Developing Single page application with best practices

by

Sumeet Wilkhu

A Thesis submitted to the Graduate Faculty of
Auburn University
in partial fulfillment of the
requirements for the Degree of
Master of Science

Auburn, Alabama
May 07, 2017

Keywords: Single page application, PCSE, schedule,
estimate, software process, PCSE Desk

Approved by

David A. Umphress, Chair, Professor of Computer Science & Software Engineering
Cheryl Seals, Associate Professor of Computer Science & Software Engineering
Saad Biaz, Professor of Computer Science & Software Engineering

Abstract

Browsers have been around since early 1990 and have been used for disseminating information using HTML, CSS and Javascript. Initially, Javascript was used as client side scripting language and its use was limited to data validation and manipulating HTML elements. Over the past decade, browsers have evolved to an extent that native-like applications can be developed using Javascript that resides completely on browsers. The support of browsers with the underlying operating system and with the latest version of Javascript has lead to the development of Javascript frameworks. These frameworks can be used to develop native-like applications known as Single Page Applications (SPAs).

The objective of this thesis is to introduce SPAs and how they differ from traditional web applications. The thesis examines different ways to develop an SPA and proposes a way to develop an SPA using best practices.

Acknowledgement

I take this opportunity to thank all those who helped and guided me through this research. I consider it an honor and privilege to convey my prodigious and everlasting thanks to my advisory committee chair, Dr. David A. Umphress, Computer Science & Software Engineering department, Auburn University for all the advice, guidance and support provided during this journey of working through my thesis. I also want to express my deep sense of gratitude to Dr. Cheryl Seals, Computer Science & Software Engineering department, Auburn University, and Dr. Saad Biaz, Computer Science & Software Engineering department, Auburn University, for their valuable advice and critique provided during the thesis work.

With immense pleasure and satisfaction, I express my sincere thanks to all my family members and friends for their kind help and feedback on the thesis work. I thank you all for your cooperation and companionship provided during this journey.

Table of Contents

Developing Single page application with best practices

1. Introduction

1.1 Problem description:

The browser has been a platform for viewing traditional web documents consisting of HTML, CSS, and Javascript. Over the last decade, browsers such as Safari, Chrome, and Firefox etc. have gained the ability to interact with the underlying OS and have been kept up to date with the latest version of Javascript, changing the role of Javascript from being just a scripting language of the web to a popular language for web application development. Javascript has evolved to the extent that native-like applications, which are meant to be used on a particular platform or device, are achieved using Javascript frameworks. Therefore, browser support for Javascript frameworks provides an alternative for developers to create native-like applications for different OS.

In contrast to a Javascript-based application, a native application running on different operating systems has always been challenging to develop. A native application has to be cross-platform compliant and requires a different set of technology stacks to be used for development and bundling. The lack of standards causes more time and effort for developing a native application for different OS.

With this thesis, we present a way in which a Javascript-based architecture known as Single Page Application (SPA), can be used to develop native-like applications for a browser. The thesis also examines various ways to develop an SPA and the problems/benefits associated with them. It proposes a set of practices that a developer should follow in order to develop an SPA by demonstrating how PCSE Desk, an SPA that assists in effort estimation, was written.

1.2 Overview:

Web applications have come a long way since their inception. A web application in the 1990's typically consisted of multiple static HTML pages. These web pages were sent from the server-side of the application and loaded completely from scratch at the client-side. With time, as software and hardware technology improved, loading partial views became possible, thus avoiding excessive reloading of web pages and resources such as headers, footers, navigation, images and more. Still, a lot of inconsistencies remained in terms of

A. A clear and consistent design that takes partial views into consideration.
B. An application architecture with separation of concerns.
C. A standard for development technologies that does not require a pool of expertise in multiple domains or languages.
D. A software process that supports the software development lifecycle and improves the way web applications could be developed.

With the proposed solution, the above mentioned problems are dealt with in detail and an attempt has been made to guide a developer out of these problems. PCSE Desk, an SPA, was developed to illustrate the idea behind this thesis and observations were made about the process to validate an SPA development process. The development process enlightens a developer about SPAs and lightweight processes. It also draws attention to the benefits of SPAs being used as a native application.

1.3 Objectives of Thesis:

The objectives of this thesis are as follows:

1) To demystify an SPA and how it is different from traditional web applications.

2) To understand the benefits of using AngularJS for developing an SPA over other ways to develop an SPA.

3) To illustrate an SPA development with a sample application.

1.4 Work Breakdown Structure:

Thesis chapters are broken down as follows:

1. Chapter 2 (Previous Work) provides an analysis of an SPA and various technologies used in the development and deployment of an SPA.
2. Chapter 3 (Solution) presents the complete process for developing an SPA with best practices.
3. Chapter 4 (Solution Validation) explains the application process taken to develop the features of PCSE Desk and describes the components, implementation, and deployment of the application.

4. Chapter 5 (Conclusion & Future Efforts) concludes the major lessons learned during the development process of PCSE Desk and its significance for a developer. It also details some features that could be beneficial for a developer when using PCSE Desk in future.

2.1 Traditional Web Application:

Traditionally, web applications have focussed on being 2-tier (client-server application), or 3-tier (client-server with middleware/database) in their architecture. The major actors that have played an important role have been categorized as either client-side or server-side. A web application typically involves a marriage between a client and server where the client is usually responsible for loading the static web page requested using HTTP Requests, and the server-side is responsible for handling these HTTP Requests with an appropriate response. This response most often leads to another static web page which requires all the resources to be fetched from server and loaded into the browser. A traditional web application has a clear distinction between server and client. It always assumes the client has fewest amount of responsibilities.

Figure 2.1 illustrates the architecture of a traditional web application with presentation layer, logic layer, and data layer.

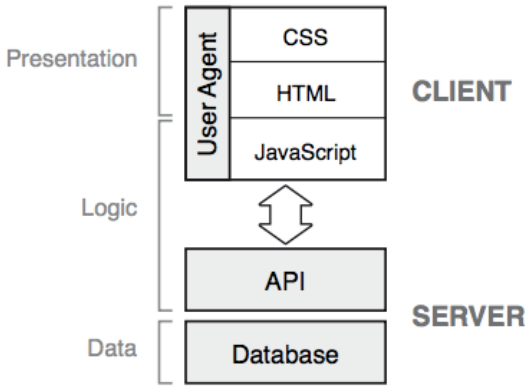

Figure 2.1: 3-tier web application architecture, adapted from [Peacock 2000].

These web applications became widely popular at the time when the devices to run them were not powerful and the memory-heavy architecture of 2-tier or 3-tier applications made a negative impact on the overall performance. Because they were the most popular means of accessing information, their application architecture was clear and definitive: the client-side consisted of HTML, CSS and Javascript, whereas, the server-side consisted of API logic and a database. At the client end, these HTML web pages were loaded on a browser via an HTTP request. HTML was used

for describing the structure and presentation of information fetched via an HTTP request, and its decoration and standardization across multiple pages was taken care of via CSS. Javascript was used for validating data, manipulating the HTML elements, and adding events to the HTML. With every HTTP request made by the events in these web pages, a new web page was loaded in the browser and all the resources were loaded from scratch once again. From a developer as well as user point of view, that led to unnecessarily loading resources like CSS, Javascript and images repeatedly. While these web requests were made, the browser had no significant role because the client could not perform any action while the web page was loading. Web applications, when compared to their native application counterparts, were much slower in performance and user interaction because of the page refresh that was involved in the process of performing actions such as submitting forms, getting images, or data, etc.

2.2 Single Page Application:

A Single Page Application (SPA) is a web application that fits in a single HTML web page, which acts as the shell for rest of the application's web pages, and whose end user interactions are implemented by using HTML, CSS and Javascript. Most of the development happens on the front-end as opposed to traditional web applications that rely heavily on server side interactions to reload new HTML web pages whenever the client changes context. An SPA keeps the business logic and the required data in the front-end and works with the local storage of the browser. All the server side interactions are limited and asynchronous in nature. These interactions are required for resource retrieval from an API end point present in the server side of the application.

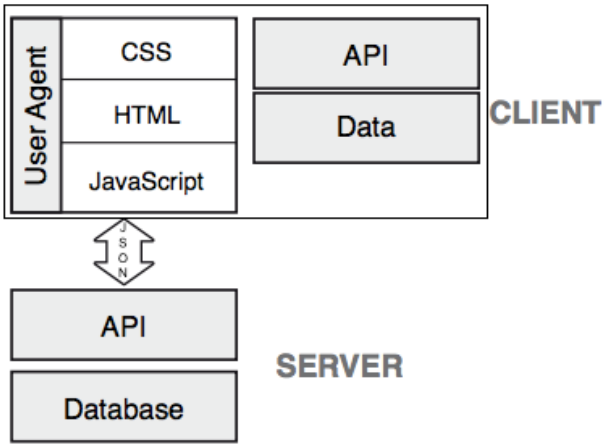

Figure 2.2: Single page application architecture, adapted from [Peacock 2000].

2.2.1 Architecture of an SPA

Generally, an SPA has a different architecture in comparison with traditional web application, as illustrated in Figure 2.2. The browser acts as a "fat" client for an SPA, meaning it gets the HTML webpage requested by the application along with the business logic and the required data to initialize the application. Once an SPA is loaded, the first set of actions that are performed by the application is loading the shell of the application, which involves fetching the HTML web page and all the relevant Javascript and CSS files in the browser.

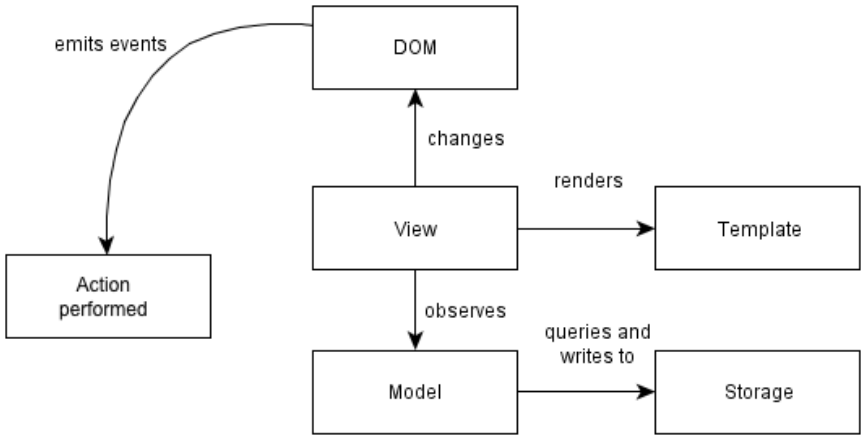

Figure 2.3: Front-end architecture for Single page application. [Takada 2012]

Based on the application framework used for developing an SPA, the application fetches the business logic and initializes the controllers which helps in fetching an HTML partial view: a reusable HTML page which can be used as a child in multiple views, the remaining business logic, and the data model to the browser. As illustrated in figure 2.3, the application framework is able to render an HTML view using the template and the data model. An HTML view is a concept in SPA architecture which allows data to be combined with the template to render a partial webpage. These partial webpages are used to append the existing structure of the application in the front-end by manipulating the existing DOM tree structure. [Takada 2012] The DOM manipulation helps in appending the partial views to the already existing webpage in the browser asynchronously, bypassing the page refresh and giving an impression of a native-like application.

Unlike traditional web applications, the view and the model have a two-way binding which means that any change made to the model is automatically reflected on the view and vice-versa. Based

on the actions performed by the user in these views, the model evolves, causing different views to be generated. The model always resides on the local storage of the browser and, when the user decides to store data, the data is synchronized with the server-side using an API request. The most important job of the server becomes to serve multiple clients, provide resources based on the API end point requested by a client, and help the client to store the data in a non-volatile storage such as database.

An SPA also gives the ability to handle and emit events from the DOM such as form submission, single click of mouse, double click of mouse, mouse hover, etc. These events exist on the DOM based on the requirement of application controllers. These events continuously redraw the DOM tree structure and help navigate to a different section of the application by manipulating the browser's current URL. This manipulation in the URL of the application is known as routing. An SPA usually consists of a configuration file which helps binding routes to its respective controllers. Hence, routing in an SPA plays a vital role in keeping the application development in harmony with the required features or components of the application. The routes in an SPA are always dynamically generated, driven by the model and compartmentalize the application into modules. This modularization advantages the application development process because it removes the cohesion among the features or components and makes unit testing as well as end-to-end testing easy.

2.2.2 AJAX for Single Page Application

AJAX, also known as Asynchronous Javascript and XML, is a technology of the web that helps developing client rich web applications. It is based on the core components of HTTP, DOM, and XMLHttpRequest.

2.2.2.1 Key concepts and components

1. HTTP

"Hypertext Transfer Protocol (HTTP) is an application-level protocol used for creating distributed, collaborative and hypertext and media information systems." [Fielding 1999] It is also used as a generic protocol for communication between user agent such as a browser and the systems that are supported by internet such as FTP, SMTP etc., providing access to resources from various applications on the Internet. [Fielding 1999]

Web applications use the HTTP protocol in order to set up a line of communication between a client and the server. It allows the client to request access to resources available on the server using different request methods such as GET, POST, PUT, DELETE, and more. The HTTP request is sent from the browser to the server as a request message consisting of request-line, headers and an optional message body. When the server receives the message, it interprets the request and responds with an appropriate HTTP response. The HTTP response consists of the status-line along with an optional message body.

From a developers perspective, the request method is an important concept to be understood as it allows the application to request resources on behalf of the client. A request method provides different means of communication with the server to access the resource required at the client-end. These methods are as follows: [Fielding 1999]

A. GET

   The GET method allows the application to request either a resource or a representation of resource using the Request-URI.

B. POST

   The POST method allows the HTTP request to enclose data in the message body and server handles the request using an application appropriate based on the Request-URI.

C. PUT

   The PUT method helps in storing the data enclosed in the message body of the Request-URI.

D. DELETE

   The DELETE method requests the server to delete a resource identified by the Request-URI.

2. DOM [Hors 2004]

The Document Object Model (DOM) is a programming interface for HTML and XML documents. It represents a map of nodes and objects connected in a logical sequence to form a document such as HTML webpage. DOM defines a relationship between different elements of the document in a parent-child relationship and provides different ways to use Javascript for accessing and manipulating the document structure and its content.

3. XMLHttpRequest

XMLHttpRequest is an API that provides the capability at the client end to create an object that helps in communication between a client and the server for accessing a resource even after the web page gets completely loaded in the browser. While requesting the resource, the application remains responsive. This allows the client to update the web page without reloading it and opens up the possibility of developing an SPA using AJAX.

2.2.2.2 Overview of AJAX for developing SPA

In order to develop an SPA, AJAX utilizes the mechanism of DOM manipulation in which events on the HTML view lead to an asynchronous action to perform HTTP requests such as GET, POST, PUT and DELETE and then, redraw the DOM tree based on the response generated from the server-side. An event-like form submission usually leads to a page refresh because it involves GET or POST operation. Usually, the operation involves storing/getting the data from server-side and responding with a different web page. With AJAX, a developer adds the ability to handle the submission event in the client-side logic and when the user performs the action to emit that event, an appropriate action is performed asynchronously using an XMLHttpRequest object. This object provides the ability to perform HTTP methods such as GET or POST on the server-side without interrupting the client side. Once the operation is performed successfully, the DOM elements are manipulated using Javascript code to append the information to the same view. This complete cycle of operations takes place asynchronously without causing the web page to refresh.

When developing an SPA using AJAX, a controller consists of the data model, business logic, and DOM manipulation logic of the application. The DOM manipulation such as adding/removing/ accessing the elements, events and style in the existing DOM, requires both the HTML view and the Javascript controller to be familiar with each other. For example, the data received from the server in the controller is supposed to be attached with the DOM tree using a specific HTML *div* tag. When the *div* tag is attached to the existing DOM, it successfully presents the data in the view. But, if another HTML view with different requirement needs the same data to be attached with its DOM tree, such as an unordered list, the controller has to either duplicate the function or logic to attach the data to another view or create a new controller altogether to handle the requirement. The changes in HTML view from a controller creates cohesion between controller and view, reducing the reusability of the

controller. Hence, AJAX can lead to tight coupling among various components of the application such as HTML and Javascript and make it hard to develop an SPA with a modular design.

2.2.2.3 Problems with AJAX

AJAX can be used to develop an SPA but it lacks in certain issues that makes it hard to develop an SPA. The major problems observed from a developer's perspective are:

1. **Increase in Complexity**

During the execution of an SPA, the DOM structure is modified many times and doing so requires the manipulation to occur asynchronously. As the nature of the transaction is asynchronous and the resources are acquired in the controller of the application, the controller itself becomes responsible for appending or manipulating the view. In order to do that, the controller accesses the HTML directives and edits or creates the DOM structure to be attached with the view. As the application grows large, the application's controller becomes complex. The complexity increases because the developer adds more features in the existing application that require the controller and view to be enhanced and modified. The enhancements and the modifications in the controller involve more DOM manipulation added on top of existing controller to handle new events generated in the view. As these changes are tied to the view, any change to the view would require the developer to track changes in the controller as well.

2. **Applications are hard to debug**

Applications developed using AJAX are not modular in design as the DOM manipulations in the controller are dependent on the events generated in a specific HTML view. The tight coupling between the controller and the view makes it hard to test and debug each of these individually.

3. **Securing resources and protecting data**

Most browsers allow the inspection of a webpage in developers mode. In this mode, AJAX-enabled application could be exposed to hackers or plagiarism, if written poorly. These services should be restricted and the access should be available to only those intended.

2.2.3 AngularJS for developing SPA

AngularJS is a Javascript based framework used for developing dynamic web applications. The framework helps developing MVC(Model-View-Controller) based SPA for client-side development. Instead of using HTML as static web pages, the framework gives the developer the ability to create reusable HTML directives and components which can be attached to the DOM without page reload. [Koppaka 2016] The major contributors to its success are features such as

A. Data binding

Data binding in a feature of AngularJS that provides communication of data between the model and the view. The communication is two-way and change to the data in model is automatically reflected in the view and vice versa. [Google 2010]

B. Dependency injection

Dependency Injection allows the various components in AngularJS environment to be created and delegated based on their requirement in the application. For example, if a service only needs to executed when a certain controller is loaded, AngularJS automatically loads the service and delegates it to the controller. AngularJS has a system which controls the management of these components and the developer can inject various components such as service, directive etc. without caring about their interdependence. [Google 2010]

C. Dynamic routing

Dynamic Routing in an AngularJS application is a module responsible for delegating requests on the basis of a URI to a controller which holds the functionality to perform an appropriate action. The URI is a combination of URL(Uniform Resource Locator) and URN(Uniform Resource Name). It is also able to capture the parameters passed in the URI to be delegated to the controller in use. The controller can use the URI for accessing resources from the server side based on the resource identifier or name and hence, routing is vital for SPA development.

D. Directives

Directives are HTML markers which have the ability to add new behaviors to the existing DOM. AngularJS has an HTML compiler which parses the HTML and AngularJS directives in form of an element or an attributes. Once compiled, the associated behavior of the directives gets attached with the DOM. [Koppaka 2016]

AngularJS has a lot of built-in directives such as ngApp, ngBind, and ngController etc. These directives have their respective functionality in an AngularJS application as they provide ways to bind the data, the view and the controller together. Developers also have the ability to create their own directives much like controllers or other modules to add new reusable components that manipulate the DOM.

2.2.3.1 Building Blocks for developing SPA [Google 2010]

AngularJS has many components that work in harmony with each other to add necessary behavior in an SPA. It uses components such as directives, templates, repeaters, modules, controllers, components, component router and more. This section discusses these components in details and understand how they work together.

A. Templates

A template is an HTML web page which consists of AngularJS directives and artifacts. A template in AngularJS is a combination of directives, expressions, filters, and controls that combine with HTML to form the view.

B. Expressions

Expressions are AngularJS's way of incorporating Javascript-like code snippets into templates by using {{expression}} syntax to produce an outcome such as data binding and function execution. These are Javascript-like because unlike Javascript expression, the AngularJS expressions do not run on the global window variable of the browser. Instead, AngularJS expressions are evaluated on a $scope variable and do not have access to functionalities such as function declaration, control statement like if-else, regular expressions, and more.

C. Repeaters

A web application usually has to display items or collections of elements that users interact with such as, list of products displayed in an e-commerce website, tweets listed on twitter, etc. Repeaters in AngularJS provide a way to iterate over a collection or array of objects including all the data types such as string, number, or custom objects. For example, AngularJS uses the ng-repeat directive to loop over a collection and display it on the view without any user-defined DOM manipulation.

D. $scope

$scope in AngularJS is the link between the controller and the HTML view. During the template linking phase the directives set up $watch expressions on the scope. The $watch expression allows the directives to be notified of property changes, which allow the directive to render the updated value to the DOM.

As AngularJS promotes separation of concerns, the $scope acts as the glue between the controllers, directives, and DOM. Each of these components has a reference to its respective scope. AngularJS also uses the idea of scope hierarchy in which multiple scopes can be created within an AngularJS application even under the same controller. This gives the ability to add one behavior to the parent attribute on the HTML view and another to its children attributes and elements. For example, if the child scope consists of a name object with “ABC" as its value and its parent scope also has the same object called name with “EFG" as its value, the child will hold “ABC”, as long as the object in the child scope exists. In case the object is not available anymore or is removed from the scope, the parent scope value would be used as a fallback, making the value of name "EFG".

E. Modules

Modules in AngularJS are a form a container in which application components such as controllers, services, directives, etc. reside. AngularJS does not have a main method to initialize the application as traditional web applications do. It provides an angular.module() function which can used to create, register, and call the modules in the application. It uses a declarative way to specify the bootstrapping of the application which gives the developer flexibility in developing the user interface and connecting the AngularJS components of the application. The major benefits of this approach are

1. The declarative process is easier to develop and understand by the developer.
2. It is easier to utilize the modules as independent piece of software.
3. It makes unit testing and end-to-end testing easier by only using relevant modules.

F. Controllers

AngularJS uses a controller as a Javascript function that has the point of entry into the web application. When the developer attaches the ng-controller directive in the HTML template and the

application starts bootstrapping, the AngularJS controller attaches the scope with the set of state and behaviors in the AngularJS application. While developing an SPA, it is considered a good practice to use the controller for developing business logic only. Controllers are not supposed to manipulate the DOM directly and, rather, use directives or services for DOM manipulation, keeping the application loosely coupled.

G. Components

AngularJS components are treated as a special form of directive. They use a simpler form of configuration suitable for component based application. Unlike directives, components use the component() method of AngularJS to attain the functionality of directive and controller. It helps in creating HTML elements which are independent and encapsulated. angular.module() is used in an application for creating a component and the resultant component has the ability to attach a controller which can be used for adding the data and the view to the $scope. With different components with their respective data and view, AngularJS is able to generate a modular application. The major benefits of using component-based architecture in AngularJS are

1. The ability to isolate the scopes of different components allowing the application to be divided into meaningful chunks.
2. Easier to upgrade to subsequent versions of AngularJS.
3. Simple to configure and easier to use than an HTML Tag.

H. Services

An AngularJS service is a shared piece of code that exists in a separate file for providing regular access to a resource or a component and is added to the controllers using dependency injection. These services are not provided to the controllers right away and are lazily instantiated, meaning, the object created by the controllers for accessing the service is injected when needed and during the bootstrapping of the application. These are also singleton objects which means only single copy of the instance is created for the lifetime of the application. For example, $http is a service used in AngularJS application for doing asynchronous communication with the server in order to retrieve any resources utilizing the server-side API end points.

I. Routing and multiple views

AngularJS uses angular-route, an built-in service provider that is used for the navigation of the application. It uses either hash-bang or HTML5 pushState for routing the application and provides

a way to navigate from one state to another. A route in an SPA consists of the path to a service along with the route parameter. Each routing parameter can be fetched using $location.url from the browser and used for retrieving the resources necessary to move to next state. For example, a location url can be used to fetch resources of a particular entity such a user id of a person in a social networking website can be used to retrieve a detailed profile view of that person.

J. Event Handlers

In order to handle events within an SPA, AngularJS prefers to use special directives already implemented for handling operation such as single click of mouse, double click of mouse and more. These directives are used to add special behavior to the DOM elements and are detected by an AngularJS application for executing the handler code. Developers can easily implement a directive to handle specific events based on the application. AngularJS uses ng-click, ng-mousedown, and more to perform event handling in the HTML view.

2.2.3.2 AngularJS application life cycle

AngularJS applications are detected by the HTML compiler by traversing the DOM and locating AngularJS based tags in the HTML templates. AngularJS promotes the idea of modularizing the application where the business logic in the controllers is always separate from the HTML templates and is bind together during AngularJS's application life cycle. The three major phases involved in the life cycle are Bootstrap, Compilation, and Runtime data binding. [Google 2010]

Bootstrapping, the first stage of application life cycle, begins when the SPA is initially loaded in the browser. It initializes the Javascript driving the application. It tries to utilize jQuery, if present, for running AngularJS framework. Otherwise, it uses an in-built jQLite, a compact version of jQuery in the AngularJS framework that keeps even the deprecated function of jQuery. jQuery and jQlite plays an important role in AngularJS because they bring in the library functions that were written with best practices in mind and are used heavily in the AngularJS framework.

Once the initial application is successfully loaded, the root of the application is detected and the application jumps into the second phase of the life cycle. In the second phase, the static DOM is accessed by the application and is connected with the scope of the controller under the root level. While doing so, the static DOM is traversed in the background to link the scope of the application

controller with the AngularJS directives present in the HTML template. This linking results in a dynamic view which is driven by the model.

Finally, in the runtime phase, the application completes loading the application and loads services when requested by the controller. The application in this phase, keeps itself in a runtime state as long as the application is either open or the application is not reloaded. While in the runtime state, the application keeps itself synchronized with the scope. Once the application is closed the scopes and views of the application are destroyed with it.

2.2.3.3 Benefits of AngularJS for developing SPA[Mikowski 2013]

The past several years have seen an improvement in the maturity of Javascript, HTML5, and CSS3. Web 2.0 gives a passage to all these browsers to be omnipresent in most of the devices in the world and be used as a platform that supports all the major applications. And, because of these advancements, an SPA enjoys most of its benefits that are mentioned below:

1. Native-Like Application

An SPA has the ability to revamp the structure of DOM tree and manipulate the model to be incorporated with the view. This act of revamping or redrawing happens without a page refresh and while the new views are generated. The current views remain responsive and user can perform action while other events take place. This gives an impression of native-like application while using them.

2. Separation of Concerns

Separation of Concerns is a software engineering principle regarding modular design. An SPA decomposes the application into its respective units and makes each unit responsible for all of its functionality. An SPA follows a MVC architecture in which the Model, the View and, the Controller are kept as separate entities and are loosely coupled with each other. The model is the representation of the data model fetched from the server side, views are the templates combined with data model to generate meaningful front-end with the help of DOM, and controllers helps bring the business logic to the front-end of the application and binds the application together. The loose coupling makes sure that there are least amount of dependence on each other and breaks the application into meaningful units which could be reused in other part of application.

3. Responsive and Reliable

An SPA controller is not just responsible for the data validation and scrubbing but also deals heavily with the business logic implemented at the client end. The business logic resides in an SPA controller, making interaction with the server-side minimalistic. This approach helps these applications be responsive to user interaction. The interaction that takes place with the server-side is also kept asynchronous in order to avoid any delay in response. The data model is stored in the browser's storage and synchronized with the server when necessary.

4. Cross-Platform

Browsers are the most effective way to interact with a web application. They do not require special instructions to be executed on different operating systems and they do not require any pre-requisites to be already installed on the system. Everything is bundled with an SPA and presented with the help of browser already present.

5. Easy deployment

The learning curve for deploying an SPA is minimal. Javascript is available in all the browsers and hence, attaching the appropriate Javascript files with the application would make the application executable on any browser. But, there are couple of techniques which helps making application deployment seamless and secure. For example, minimizing the Javascript files to combine the Javascript code from multiple sources into one huge file in order to reduce the execution time, writing modules in IIFE (Immediately invoked function execution), a practice in Javascript that protects the code by lexically scoping it inside an executable function block and more. These techniques help building a secure and unexposed application and following them is considered good practice for deployment.

6. Testing

Modularization of an SPA also comes with an added advantage that these individual pieces of application are written in such manner that they are loosely coupled and, hence, unit testing is easier in comparison with traditional web application which binds data, view, and logic together. End-to-end (E2E) testing frameworks are also mature enough to favor SPA as unit testing can only act as first line

of defense. To correctly test the integration of components and features, these frameworks play a vital role.

2.2.3.4 Disadvantages of AngularJS for developing SPA

1. Support for Javascript libraries

A rich SPA is developed using AngularJS in conjunction with other Javascript libraries to enhance the user experience. AngularJS makes it tough for a developer to directly use these libraries and inject them along with other AngularJS services and components. In order to use these external libraries, the developer has to explicitly create a custom service to make the library usable in AngularJS environment.

2. Understanding AngularJS life cycle

As a developer, it is hard to understand the concept of AngularJS life cycle and the role of $scope and $watch variable in it. The $watch variable from the life cycle can be abused, if not used properly in the controller of the application. For example, if a developer create multiple watchers in the controller to handle events, the SPA performance can degrade significantly and also lead to memory leaks.

2.3 Light-Weight process:

Light weight development methodologies embrace practices that allow programmers to build solutions more quickly and efficiently, with better responsiveness to changes in business requirements. [Khan 2011] The flexibility and agility to adapt with changes due to circumstances and business requirements makes them amenable for application development that involves factors such as small team size, low cost and low risk, and change in technology or environment etc. [Subramanian 2011]

A light weight software process encourages short and long term goals to be divided into an iteration plan. This flexibility allows short term goals to be achieved within the iteration boundary and opens a window of opportunity to discuss new requirements, if need be. These new requirements become part of new iteration plan and set new goals for rest of the project. In the beginning of development process, a lot of margin of error for planning and estimation is kept while following the light weight methodology. During this phase, the inexperienced developer tend to plan poorly. But, the idea of small iterations provides the developer with an opportunity to correct its mistakes. The

emphasis is kept on improving the accuracy with time as the developer learns more about the project and gives a chance to improve on the previously made mistakes.

Some of the benefits of using a light weight process are: [Khan 2011]

1. Rapid development due to short iterations and cost friendly.
2. Integration with other heavy or light weight processes.
3. Precise Metrics to measure performance such as Burndown, velocity etc.
4. Involves frequent demonstration to stakeholders for feedback.

2.4 Practitioner Centered Software Engineering (PCSE) for SPA development:

PCSE is a framework for binding a light-weight software engineering process. [Umphress 2015] It is the most recent embodiment of a personal self-improvement process for software engineers to control, manage and improve the act of developing applications. Using common industry practices, PCSE describes the following activities/phases that are performed within the software development process: Analysis, Architecture, Project plan, Iteration plan, Construction, Review, Refactor, Integration, Post mortem and Code complete. Each of these activities is associated with a particular artifact such as the Operational specification, Scenario-Component map, Iteration map, Conceptual design, Size matrix, Time log, Change log, Iteration map, Burn-down chart, Calendar etc. The flow of these activities is mentioned in Figure 2.4, providing an example of the lifecycle using PCSE.

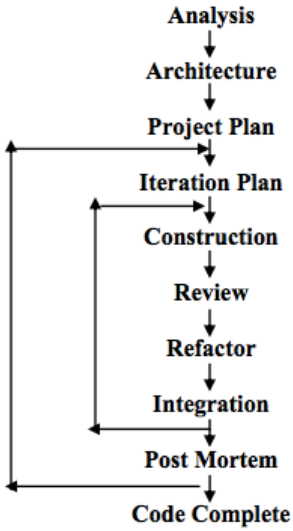

Figure 2.4 An example of PCSE Life cycle

PCSE can be practiced when developing various types of applications. It is a flexible framework that helps developer choose the activities to be performed based on the requirements of the project. It promotes use of best practices that developers need such as Test Driven Development(TDD). TDD goes hand in hand with SPA methodology. Single page architecture is modular enough to perform TDD while following PCSE life cycle. The short iterations keep the developer accountable for the deliverables and helps the developer keep track of their progress. The progress in PCSE is determined on the ratio of the components planned to the components actually built. If the progress gets stalled, PCSE gives the developer the ability to change the plan and re-estimate the time and components by either adding hours each day planned earlier or add another iteration for pending components by extending the deadline of the project. This amount of flexibility is needed while developing an SPA as it does not hinder the developers creative instincts to explore new ideas and focusses on the development of components by dividing them into tasks in an iteration map. Each task within an iteration can be tested in an SPA architecture because of the modularity, thus making the process more visible to the client.

2.5 Technologies for deployment of SPA

Continuous integration is a development practice that requires developers to integrate code into a shared repository several times in the project life cycle. Each check-in is verified by an automated build system, allowing the developer to detect problems at early stage. [Voort 2017] By integrating the application at the iteration boundary, developers can detect errors quickly and locate them more easily. Major benefits of continuous development and integration are as follows:

A. Automate the testing and building process of application.
B. Increase in the test coverage by continuously committing the code in the production repository.
C. Provide visibility to the developer(s) or across the teams. If the build is not successful, it helps determining what went wrong.

Therefore, selecting appropriate technologies for rapid and continuous development of an SPA is pivotal for its success.

2.5.1 Yeoman

Yoeman**,** a project scaffolding system which helps create projects for different ecosystems such as Java, Ruby, and Node.js, allows for rapidly getting started on new projects as well as

streamlining the maintenance of existing projects. Yeoman by itself does not make any decisions. Every decision is made by generators, which are basically plugins in the Yeoman environment that allows to create a workflow for the application development and deployment using templates provided by Yeoman. [Yeoman 2017] These generators are easy to create and match the application workflow. This workflow is a robust and opinionated client-side stack, comprising tools and frameworks that can help developers quickly build rich and beautiful SPAs. These generators provide everything needed to get started without any of the uncertainties associated with a manual setup. Some of the benefits of using Yeoman are:

A. Rapidly create a new project.
B. Create new sections of a project, like a new controller with unit tests.
C. Create modules or packages.
D. Bootstrapping new services.
E. Enforcing standards, best practices and style guides.

In order for Yeoman to get these benefits, the Yeoman ecosystem uses different Javascript-based tools for streamlining various tasks written in the Yeoman script. These tools are discussed below:

2.5.1.1 Grunt

Grunt is a JavaScript-based task runner, a tool used to automatically perform frequently used tasks such as minification, compilation, unit testing, linting, build process, etc. It uses a command-line interface to run custom tasks defined in a file (known as a Gruntfile). Grunt has a large ecosystem and already consist of reusable plugins developed by other developers with best practice in mind. These plugins help in automation of all the effort required to perform repetitive tasks during SPA development.

Gruntfile is a template based approach to define the mechanism to perform tasks that are repetitive in SPA development such as unit testing using Jasmine, generating "dist" folder for building the production level application, compiling application and deploying on the production server and automatic browser refresh in case components of SPA changes and more. The script allows to set files and folders to be tracked constantly and perform a certain set of action needed to achieve the expected outcome.

The initial configuration in Grunt allows the developer to add the project files and folders and register them with required plugins for tracking any changes. For example, Grunt file in figure 2.5 demonstrates a script which runs at the runtime of the project execution during the development and uses jshint, a plugin used in the code editor for syntax highlighting the errors in the Javascript code to register various files to be tracked. Similarly, the watch section of the code allows the runner to constantly look for changes of any form to these files. If the runner notices any change in them, it triggers the appropriate action mentioned in the script.

```
module.exports = function(grunt) {

  grunt.initConfig({
    jshint: {
      files: ['Gruntfile.js', 'src/**/*.js', 'test/**/*.js'],
      options: {
        globals: {
          jQuery: true
        }
      }
    },
    watch: {
      files: ['<%= jshint.files %>'],
      tasks: ['jshint']
    }
  });

  grunt.loadNpmTasks('grunt-contrib-jshint');
  grunt.loadNpmTasks('grunt-contrib-watch');

  grunt.registerTask('default', ['jshint']);

};
```

Figure 2.5 Sample Gruntfile for a web application

2.5.1.1 Bower

Bower is another package manager similar to NPM. It is optimized for front-end development only. The ecosystem of Bower helps in installing and linking components such as HTML, CSS, Javascript, fonts and images in an SPA. Bower does not concatenate or minify the code. Instead, it installs the right versions of the packages needed by the project and also makes sure their dependencies are in place as well. It uses a bower.json file for creating a package structure with the dependencies based on the environment such as development and production.

For example, If multiple packages used for the front-end development such as bootstrap and angular-charts depends on jQuery, Bower will install the correct version of jQuery before installing other packages. This is known as a flat dependency graph and it helps reducing the web page loading time.

3. Solution

Despite the popularity of the web, a traditional web application often suffers from responsiveness and interactivity in comparison with a native application. An SPA provides a means to overcome this barrier and helps the user to have a native-like experience. As seen earlier, there are different ways to develop an SPA and achieve the same goal in terms of the application usability. But, the software architecture and the software process to develop them can vary hugely. This chapter illustrates a way to develop an SPA using AngularJS with the best practices in software engineering.

3.1 Process for developing an SPA

Assuming that the software specification is in place, the next step is to identify the software process to be used for developing an application. In order to develop an SPA, a preference is given to a light weight process as to facilitate a rapid development of the application. While developing an SPA, a developer can focus on following the best practices and continuously integrate the existing application with properly tested and reviewed changes.

AngularJS promotes the idea of Behavioral Driven Development (BDD), which focusses on how to implement a specific behavior in the application. [Google 2010] It easies the process of converting user specification to code that must be written to satisfy them. Initially, a spec is written by the developer for testing. The spec consists of the expected behaviors from the application and BDD uses whole sentence for describing a spec, with a verb such as "Describe" or "Should" in the beginning of it. The developer then write enough code to pass the test cases written in the spec. The process of BDD inherits its traits from TDD and unit testing, and is similar to TDD with regards to the writing test cases before developing the code to pass them. BDD uses verbose sentences to describe the behavior in a spec, whereas, TDD allows short and descriptive unit test cases for testing.

BDD goes hand in hand with TDD when developing spec from user stories for development and testing. Therefore, a light weight process such as PCSE is suitable for developing an SPA. The activities in PCSE supports the development of an application that has these requirements and also promotes the idea of TDD as a practice for developing an application.

3.1.1 Initial Setup of an SPA

Dependency injection, separation of concerns, components and testing are an integral part of AngularJS. These different practices helps the application to remain modular in design. An SPA development requires these design patterns and concepts to be adopted in the development process for modularization and the directory structure of the application plays an integral role. Therefore, the directory structure needs to be understood carefully.

When it comes to the directory structure of an SPA, all the components could very well exist in a single Javascript file under one folder and the application would still initialize and run properly. From an operating system point of view, running the Javascript code present in single file is preferred over multiple files, as multiple files can slow down the process of executing the code because of the fragmentation in files. But, from developers perspective, keeping the complete Javascript code in single file would be disadvantageous in the following ways:

1. **Code maintainability**

   With a single file, there is a lack of logically compartmentalized application where every component is easily located and edited.

2. **Debugging**

   Debugging the code becomes challenging if the application is not modular in design. In order to locate the offending code, developer has to spend more time locating the appropriate section of code.

3. **Testing**

   With a tight coupling in the features present in the same file, it becomes difficult to perform unit testing without affecting other sections of the code.

4. **Scalability**

   With single file, introducing new features is difficult as there is one file to work with. Even the version control suffers, especially in a team setting where many developers are working on same piece of code.

A logical division of the code is a good practice for developing an SPA. AngularJS tends to agree with the practice of compartmentalization of code. But, it does not present any guidance as to how an application should be structured. A small-scaled SPA can easily get away with any directory structure, whereas, a large enterprise level SPA can suffer major consequences if it does not choose a

scalable and maintainable directory structure. Therefore, it is highly important to pay attention to the directory structure at the initial phase of the SPA development.

Figure 3.1 shows one way to structure an SPA and draw the benefits of AngularJS during the SPA development life cycle.

```
├── client
│   ├── app                 - All app specific components go in here
│   ├── assets              - Custom assets: fonts, images, etc…
│   ├── components          - reusable components, non-specific to to our app
│   ├── index.html
├── e2e                     - protractor end to end tests
│
└── server
    ├── api                 - apps server api
    ├── auth                - For handling authentication with different auth strategies
    ├── components          - reusable or app-wide components
    ├── config              - Where we do the bulk of our apps configuration
    │   └── local.env.js    - Keep environment variables out of source control
    │   └── environment     - Configuration specific to the node environment
    └── views               - Server rendered views
```

Figure 3.1 Overview of directory structure for an SPA

The directory structure mentioned above presents a modular way to develop an SPA following the best practices of AngularJS. It presents a structure suitable for a full stack application which involves development of both the client and the server. The client folder breaks down the main elements of client side development into its respective folder. The application structure consists of various components as follows:

1. client/app

Each component that directly relates to the feature development of an SPA is created under the app folder. The name of the component becomes the folder name for that particular feature and within each component exists the following files:

```
main
├── main.js                 - Routes
├── main.controller.js      - Controller for our main route
├── main.controller.spec.js - Test
├── main.html               - View
└── main.css                - Styles
```

Figure 3.2 Example Directory structure for a Component

F. <component>.js

This file helps setting up the route for the main Javascript application present in the outer most scope of the client directory structure. It registers the controller used by the component and the route used in the application url relative path in order to access it.

G. <component>.controller.js

This module is the main controller for the component which helps setting up the scope variable and performs the necessary data binding.

H. <component>.controller.spec.js

AngularJS uses Jasmine, a Javascript framework to perform unit testing. There are many ways to perform unit testing but Jasmine follows the best practices from agile and light weight methodology. It helps developer focus on writing testing scenarios and provides functions to help with structuring tests and also making assertions.

I. <component>.html

This file acts as the partial template which uses the controller to manipulate the DOM and attach behavior to SPA. The view is created as per the directives used in the template along with controller logic and data model associated with the $scope.

J. <component>.css

This file helps decorate the view by attaching various behaviors on the HTML elements.

2. client/components

This sub-directory is used for housing the components that are reusable by the SPA. It follows the same structure as any other component or directive. For example, an SPA usually consists of a header, a footer, and a navigation feature. These features are different components that are used repeatedly in every view and placing them as a directive main components makes sense. Therefore, these special component are placed in the reusable directory of the client.

3. client/assets

The structure of the assets sub-directory depends upon the quantity of assets used in an SPA. At production level, an asset-heavy application prefers to use content delivery network for faster retrieval of assets such as images, fonts and icons. But, for the purpose of development, the assets folder can be used to place all the application assets in folders such as fonts, images and icons.

4. client/index.html

The index HTML file is identified by the application as the shell of the application which becomes the point of entry into an SPA when loaded in the browser. It consists of the relative path to all the files necessary for an SPA and helps in bootstrapping the application with the help of AngularJS framework. The loading begins when a user makes a request to the application root level using the browser URL and an API request is made to the server. The server responds with the path to the index.html file in the client, loading the initial view of the SPA and retrieving the business logic and data model in the browser.

3.1.2 Software process for developing SPA

A light weight software process such as PCSE can be used to develop an SPA that gives the flexibility in choosing the activities to be performed during the lifecycle of the project. For further discussion on the process, we suppose that the software process includes all the activities mentioned in Figure 2.4 and discuss the role of these activities and their respective artifacts in developing an SPA.

1. Analysis

Analysis is the process of understanding the bigger picture of the project and identifying the objectives of the project. Once the project is described in broad strokes, analysis helps break complex ideas into smaller meaningful chunks. These pieces help break the application into multiple scenarios and behaviors that are expected of an SPA. The scenarios are documented on the basis of the client's clarifications and discussions about the project. The developer identifies the features and the components from these discussions and generates a number of scenarios that these features or component are expected to perform. Figure 3.3 is an example artifact on how to document a scenario.

Component: XCScore				
Scenario 1				
Objective:	To explore how a cross-country score is created.			
Spec Type:	Interface			
Tuple #	**Type**	**Actor**	**Description**	**Example**
1	Event	Test Component	Create a valid instance of XCScore	myXCScore = XCScore(rider=myRider)
2	Response	Blackbox	Return an instance of a cross-country score associated with a particular rider. It should be initialized to indicate that the rider has not completed the cross-country round.	myXCScore is an instance of *XCScore*

Figure 3.3 Example Scenario for a component

2. Architecture

In the architecture phase, the main objective is to assimilate the information gathered in the analysis and identify the components and features and their functionality. During this phase, the conceptual components from the analysis are divided into actual components using CRC (Class Responsibility Collaborator) cards. These CRC cards form the basis for planning and estimation of the project. The CRC cards are used in an SPA to identify components such as model, views, routes, controllers and HTML shell. Below is an example of CRC card for views created in an SPA.

Component Name:	studentjobs
Design Approach:	Functional
Parent Component:	None
Component Type:	View
Collaborators:	Jobs,studentjobindex.html
Operations:	studentjobs

Component Name:	jobDetail
Design Approach:	Functional
Parent Component:	None
Component Type:	View
Collaborators:	Jobs,JobHits,jobdetails.html,404.html
Operations:	jobDetail

Component Name:	jobDelete
Design Approach:	Functional
Parent Component:	None
Component Type:	View
Collaborators:	Jobs,jobsDeleted.html,404.html,accessdenied.html
Operations:	jobDelete

Figure 3.4 Example CRC cards for views

3. Project Plan

The major objective of project planning is to estimate the overall effort required for the complete project. PCSE uses historical data from other projects to produce an estimate which includes the LOC (Lines Of Code) and time spent, in actual as well as planned projections. These planned and actual projections help in generating a size matrix. A size matrix consists of different size ranges for components based on the projects developed so far. It helps in determining the raw LOC for a new component in an SPA and help identifying a planned proposal for effort, both in terms of planned LOC and planned Ep (Effort or estimate time).

Size Matrix (LOC/Method)

	Low	Mid	High
VS	1	5	6
S	6	7	8
M	8	10	11
L	11	13	15
VL	15	18	

Figure 3.5 Example of Size Matrix

For Example, assume that we have worked on projects previously and have the information related to the LOC and effort from these. PCSE helps automatically computing the size matrix using its predefined model of computation. An example of such matrix is given in Figure 3.4.

This matrix can be used for generating LOC/method based on the CRC cards generated in the architecture phase. The estimate requires the developer to identify the components being developed in an SPA along with their relative size and method count and produce raw LOC based on the newly created size matrix. For example, a controller in AngularJS contains number of functions along with the constructor. If the developer identifies the number of functions along with its relative size, then the size matrix can be used for estimation.

	Estimated							Actual	
	Existing Component				New Methods				
Components	Base Component	Deleted LOC	Modified LOC	Added LOC	Method Count	Rel Size	LOCr	Method Count	LOCa
ForecastedComponent01	Component60	3	9	21	3	M	60		
ForecastedComponent02					3	S	21		
ForecastedComponent03					2	M	20		
ForecastedComponent04					2	M	20		
ForecastedComponent05					4	L	52		
ForecastedComponent06					1	VS	5		
Totals							178		0

Figure 3.6 Example of an estimate based on size Matrix

Once the estimate occurs for the first iteration, the size matrix is not required anymore. At the beginning of second iteration, the developer gets enough confidence in the previous estimate, especially, after developing the actual components and uses the experience to either change the estimate, if the project requires or keep it intact.

4. Iteration Plan

An iteration plan is a fine-grained plan with a time-sequenced set of activities and tasks, with assigned resources, containing task dependencies, for the iteration. It also involves the redesign and implementation of a task from the specification list, and the analysis of the current version of the system. It helps identify problems or faulty assumptions at periodic intervals.

PCSE has the following major goals during this phase:

1. Select/revise scenario set

   Every iteration, the developer select a set of scenarios to be implemented and modifies or adds new scenarios identified to incorporate in the project specification.

2. Set iteration goal

   Iteration goals that have to achieved before the current iteration ends and it includes following goals.

   1) A list of major classes or packages that must be completely implemented.

   2) A list of scenarios or use cases that must be completed by the end of the iteration.

   3) A list of risks that must be addressed by the end of the iteration.

   4) A list of changes that must be incorporated in the product (bug fixes, changes in requirements) etc.

3. Schedule work

The role of this artifact is to map the components from the architecture into an iteration map. The Iteration map consists of number of iterations identified as per the schedule and based on the amount of work. This map can be modified by either by adding more time in the schedule or adding more iteration, extending the project deadline.

The developer can schedule and track the effort using the calendar, burn-down chart, and diary. The following table illustrates how each parts (methods) of a component are mapped to each iteration in the iteration map:

Component	Iteration 1 Productio	Mock	Iteration 2 roductoi	Mock	Iteration 3 Productoii	Mock	Iteration 4 Productoii	Mock	Iteration 5 Productoii	Mock	
ForecastedComponent	3		1								4
ForecastedComponent02			2	1			1				4
ForecastedComponent03			1	1			1				3
ForecastedComponent04						1	2				3
ForecastedComponent05					4						4
ForecastedComponent	1										1
Number of tasks:	4		6		5		4		0	otal tasks:	19
Ep per iteration:	114		171		142		114		0	Total Ep:	541

Figure 3.7 Example of an iteration map

Figure 3.7 is an example of iteration map where a developer can plug the components such as controller, model, view etc. identified with CRC cards and mention the number of tasks that are planned for each iteration. The PCSE model of computation for effort computed previously gets divided per iteration and gives an idea to the developer to setup the calendar and revamp the iterations, if needed. Examples of the calendar are given below.

Date	Planned Available Minutes	Planned Cumulative Minutes	Planned Burndown at Start of Day	Planned Burndown at End of Day	Planned Velocity at End of Day	Cumulative Planned Velocity	Actual Av	Cumulati	Actual Bu	Actual Bu	Earned V	Cumulati
1/1/20	60	60	541	481		0		0				0
1/2/20	60	120	481	421	4	4		0				0
1/3/20	0	120	421	421		4		0				0
1/4/20	90	210	421	331		4		0				0
1/5/20	120	330	331	211	6	10		0				0

Figure 3.8 Example of a Calendar

The calendar provides a way to plan each day in the iteration map by getting the planned available time as per the developer schedule and provides an estimated burn down for each

day. By looking at the burn down at the end of each day, the developer can set up the iteration boundary based on the planned effort per iteration identified in iteration map.

5. Construction

Construction is the activity that involves implementing the high level design using a low-level design, coding, and unit testing. As PCSE uses TDD for its development, an SPA utilizes the same principle. The test cases are written based on the scenarios and then, code is written in order to pass them. In the development of an SPA, the components are unit tested using Jasmine. Jasmine allows the developer to describe test cases directly from the scenarios. Jasmine provides functions to help with structuring user-defined tests and also making assertions. Jasmine uses the describe function to group the tests together:

```
describe('sorting the list of users', function() {
  it('sorts in descending order by default', function() {
    var users = ['jack', 'igor', 'jeff'];
    var sorted = sortUsers(users);
    expect(sorted).toEqual(['jeff', 'jack', 'igor']);
  });
});
```

Figure 3.9 An example of test case using Jasmine framework

AngularJS also provides the ngMock module which helps in mocking the test cases. This module is also used to inject and mock the services for unit testing. For example, an $http service used for asynchronous communication with the server to fetch a resource requires the server availability. This goes against the idea of unit testing due to tight coupling. With this module, the test cases can inject a mock service such as $httpBackend and add expected behavior with the service to produce right output needed by the test case and continue writing the remaining test.

6. Review

Reviewing helps in examining the test coverage for the recent iteration completed. If the test coverage is not sufficient, bugs are discovered and are fixed using TDD approach of development. Code review happens after every iteration and the bugs raised in the review are fixed in the next iteration. PCSE uses review checklist for tracking the number of defects and its description. Figure 3.9 illustrates an example of code review.

Defect Pareto Analysis (taken from all defects to date)

Type	Description	Count of Occurrences
Documentation	Flaws in comments	
Build	Problems with environment, compiler, build activity	
Product syntax	Syntax flaws in the deliverable product	31
Product logic	Logic flaws in the deliverable product	73
Product interface	Flaws in the interface of a component of the deliverable product	
Product checking	Flaws with boundary/type checking within a component of the deliverable product	
Test syntax	Syntax flaws in the test code	11
Test logic	Logic flaws in the test code	20
Test interface	Flaws in the interface of a component of the test code	0
Test checking	Flaws with boundary/type checking within a component of the test code	
Bad Smell	Refactoring changes (please note the bad smell in the defect description)	8
	Total	143

Code Review Checklist

Item	Number of times this item detected a defect during review
Does the product satisfy all the requirements stated?	2
Is the testing complete?	3
Have temporary variables been used and cleaned?	5

Figure 3.10 Code Review

In the review phase, the number of defects that occurred in the project listed on the review list is checked to see if it matches the historical defect count as seen in the above figure. If the defect count matches the list, then the developer can confirm that new changes did not introduce a new defect. The test code is checked for correctness and coverage.

7. Integration

In order to integrate the existing application with the new changes, made in recently completed iteration, a regression testing is performed. Regression testing makes sure that the application functionality remains intact even after the integration. Jasmine is used for performing the regression testing for developing an SPA. In case bugs are discovered, then they are removed using TDD in the next iteration.

8. Post Mortem

This is a brief phase that helps the developer to prepare for the next iteration. The activities that are performed in this phase are:

A. Baseline the production code in version control on the local or remote server.

B. Baseline the test code in version control on the local or remote server.

C. Analyze the activities performed in the previous iteration and the correctness of them. Understand things that went well and the major difficulties faced by the developer. If any difficulty is face, then how to adapt the process in order to overcome it.

D. Revisit the estimation and planning of the project. If things are on track, then nothing changes in these artifacts. But, revamping the estimates and plans becomes necessary, if there is a chance of derailing from the projected deadlines.

9. Code Complete

Code completion marks the end of the project development. It includes a rigorous amount of final testing and verification of production code. Once the application is verified to be correct, deploying the application is the next step. An SPA deployment involves building the application for production level and pushing it to the production server, if required. Building an SPA involves the application code validation, compilation, transpiling, minification and uglification.

4. Solution Validation

This chapter illustrates the ideas presented thus far, by demonstrating a single page application called “PCSE Desk". The SPA developed using PCSE and AngularJS, in the process of validation, represents the web version of PCSE software. “PCSE Desk” is an SPA for developers who intend to use PCSE as their software development process. The development of the application took place with the students of Computer Science and Software Engineering department at Auburn University in mind. The objective of this SPA is to help the students use their browser to access the PCSE software and eliminate the requirement of Microsoft Excel. The PCSE software uses features such as tables, lookup formulae, charts, drop downs and macros in Microsoft excel to provide its intended functionality. But, there are certain difficulties faced by the students such as:

1. Microsoft Excel cannot be installed in all operating systems such as distributions of Linux.
2. PCSE spreadsheets use features which are incompatible with softwares that are alternatives to Microsoft Excel such as OpenOffice.

4.1 Development Environment

PCSE Desk and all its features were developed using AngularJS 1.5 within the Atom development environment. Atom allows plugins to be installed that are useful for developing an SPA, and allows a continuous development and integration for the project. Yeoman was used for project scaffolding and modularization. Yeoman’s angular-fullstack generator was used for developing PCSE Desk and its components.

The server for the application was also developed to store the data locally in a non-volatile database. The server was built using Node.js and express server framework, available in the Node.js ecosystem. This API was used to provide various resources to the application. MongoDB was used as the database.

4.2 Software process for developing PCSE Desk

PCSE was used for developing PCSE Desk application by tailoring the activities that were necessary for developing an SPA. The process for developing PCSE Desk is shown in Figure 4.1:

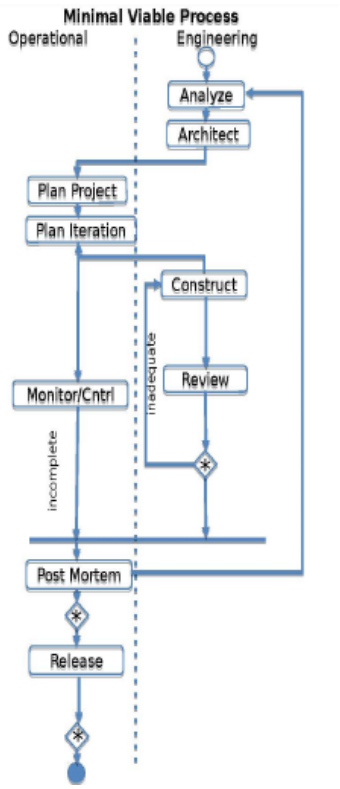

Figure 4.1 PCSE Desk - PCSE life cycle

These activities and the workflow were found suitable to develop PCSE Desk and its features over a period of time. The life cycle was kept flexible enough to provide freedom to brainstorm new ideas and continuously integrate changes in the project.

4.3 PCSE Desk features

Although PCSE involves many activities and artifacts, the scope of this application was restricted to only certain features and additional features were added to optimize and enhance the user experience while using PCSE for development.

4.3.1 Project Overview

This feature of the application is the point of entry for the developer to interact with PCSE Desk. It is responsible for viewing, creating, editing, searching, and deleting the projects. Once a project is created, it can be used for interacting with the other activities of PCSE Desk that exist within each project such as estimate, schedule, and setup of the project. PCSE requires a certain set of activities to be performed in order to develop an application. This feature is the gateway to all the other activities that help in performing the PCSE life cycle within each project scope.

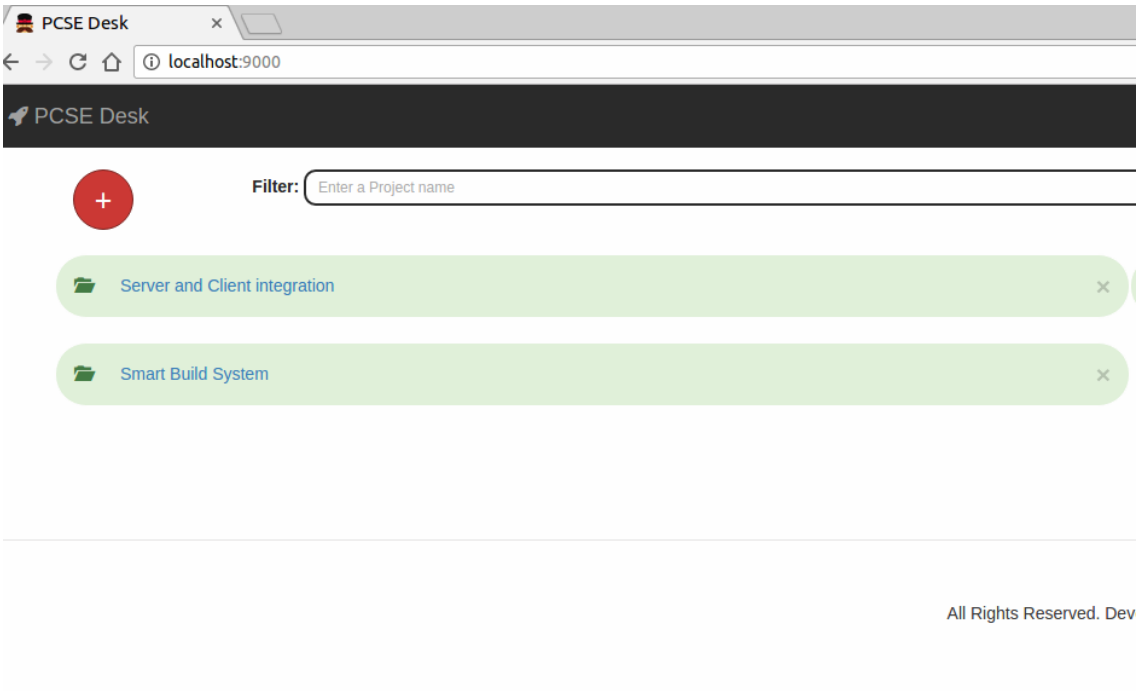

Figure 4.2 PCSE Desk - Project Overview

In order to add a new project, the developer can click on the add button on the view, given on the top left corner, and submit the appropriate details in the popup modal. After validating and verifying the information, a new project gets created and all the other PCSE Desk features are exposed to the project. In order to perform any action in a project, the developer clicks on a particular project name and the project opens, making sure all the necessary existing details of the project gets loaded in the application.

Once these projects are created, the developer also has the ability to view a list of all existing projects and delete them, if necessary. In order to retrieve a particular project from a long list, the developer can search the project using the filter option available on the view.

4.3.2 Project Setup

Project setup helps in setting up the configuration of the project currently in use by the developer. This is the initial step before proceeding with other features within each project. This feature makes sure that the project gets all the required artifacts before preceding to the next step. If the developer does not provide these details, the application would not allow the proper access to other features available to the project. The details necessary in this step are:

1. Start Date of the project
2. JSON configuration file that consists of:
    1. MGI (Minimal Guiding Indicators)
    2. MVP (Minimal Viable Process)
    3. MEP (Minimal Effective Practices)

In order to use this feature, the developer initially provides the starting date of the project. An iteration for the project gets created automatically. The next step is to upload the JSON configuration file which is available from the developer and provided by the client of the project. The client is responsible for setting up this file and providing it to the developer to upload. This file is a large JSON file created with a certain guidelines in mind which are as follows:

1. The JSON file needs to include MGI, MVP and MEP in a JSON structure.
2. The MVP object consists of all the activities from the PCSE life cycle and it needs to have the coordinates for the flow chart created from these activities.
3. MGI needs to specify the details such as the cost, the schedule and the performance of the project and MEP needs to detail the activities to be performed in the process of development.

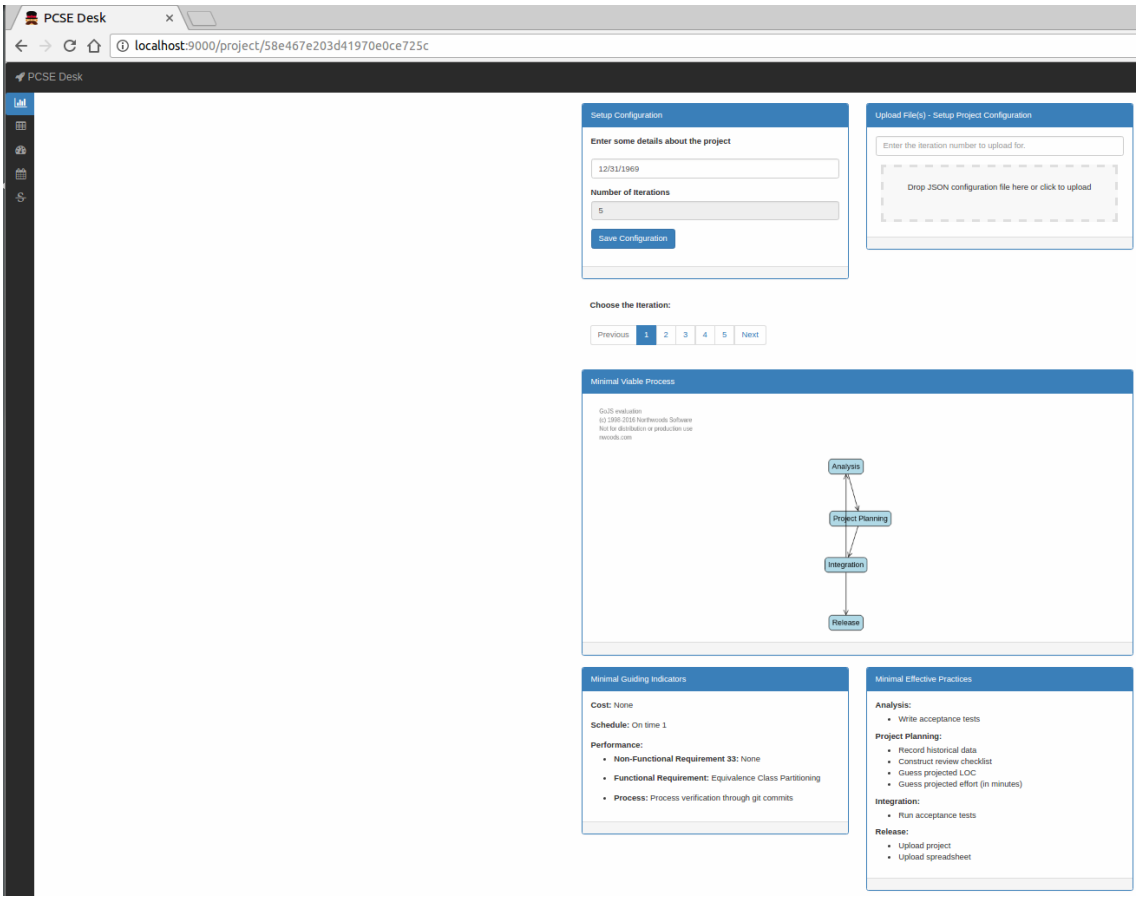

Figure 4.3 PCSE Desk - Project Setup

To submit the JSON configuration file, the ability to upload this file is provided in the application. As soon as the file is uploaded, the project gets prepared to be used. This file helps setting up the expectations for the planned iterations of the project by providing a way to view the MGI, MEP and MVP of the project. The MGI helps depicting the cost, schedule and performance of the project. The MVP is used by developer to understand the flow of the software process in order to develop the project. And, The MEP guides the developer to understand the minimum effective practices that have to be performed while performing the MVP.

4.3.3 Planning

This feature is a combination of estimation and scheduling which are integral for planning a project using PCSE. Assuming the developer has the access to the CRC cards from the architecture of the project, the developer can plan for coming up with an estimate for lines of code and effort for every component to be built in the project and distribute the effort among the iterations planned for the project. The planning consist of two sub features:

1. Estimation
2. Scheduling

4.3.3.1 Estimation

Estimation helps the developer determine the planned LOC and effort for the project as per the components planned for development. In order to generate the estimate, the application collects the historical data from previously completed projects and generates the size matrix. The size matrix is then used for creating new components.

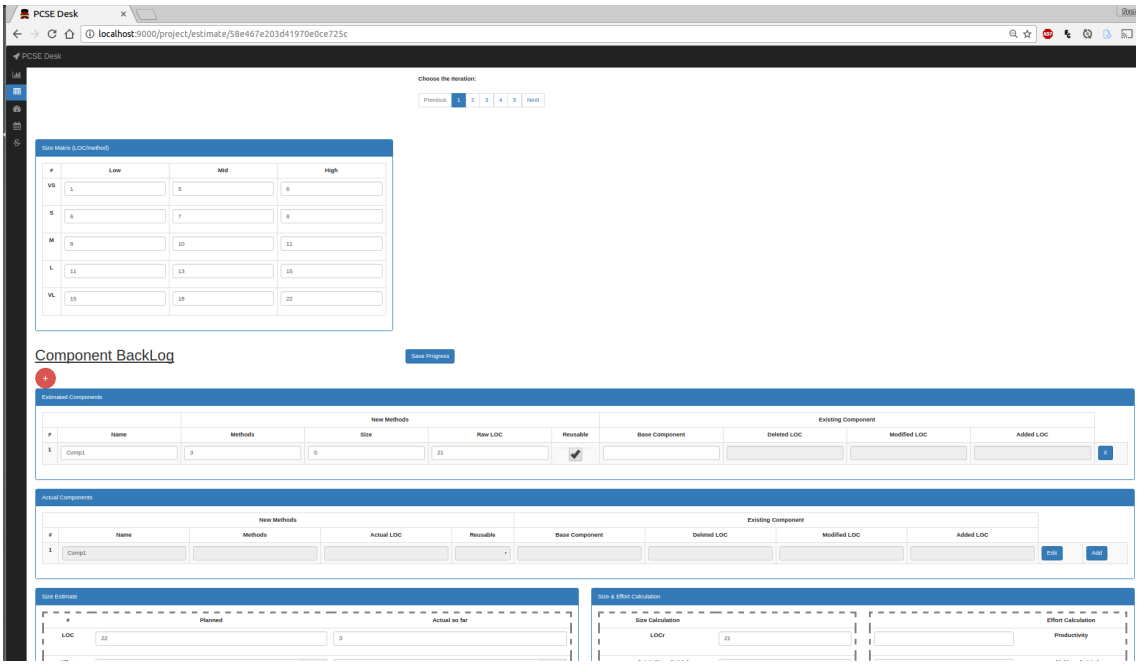

Figure 4.4 PCSE Desk - Project Estimate

In order to create new components, the developer is provided with an *add* component button on top left corner. By clicking it, the developer provides the necessary details such as component name, method count and base component or relative size to create the component. The size matrix compares the relative size of either the size provided by the user or relative size of base component and computes the raw LOC for the new component. Based on the total components and their respective raw LOC, the estimation feature is able to compute the planned LOC and effort for the project. If the developer discovers new components, this feature can be to revisited to estimate after

the iteration ends and change the estimate as per new requirements. If the estimate does not change after the iteration ends, the same estimate is used for next iteration.

4.3.3.1 Scheduling

Scheduling provides the developer with a way to divide the effort and tasks for every component identified in the estimation among the planned iterations. The project set up to create one iteration for the complete project. The developer is provided the ability to use the scheduling feature to add new iterations as per the estimate. Scheduling gives the developer the ability to add/remove an iteration in the iteration map. The iteration map provides a way to add and divide tasks in the iterations added for the project and computes the total number of tasks per component, total number of tasks per iteration and total effort required per iteration.

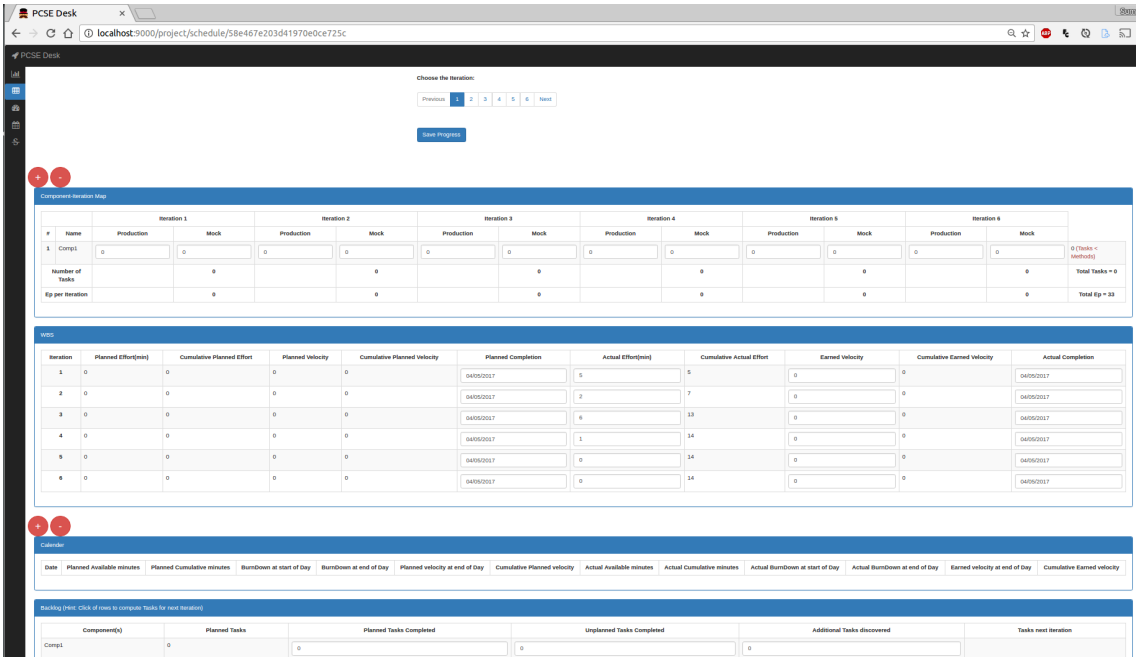

Figure 4.5 PCSE Desk - Project Schedule

The other artifacts used for scheduling are the WBS, Calendar, and Backlog. Once the iteration map is in place, the developer can view the cumulative effort and planned effort per iteration. Using this information, the developer can use the calendar to provide the planned available minutes and planned velocity at the end of each day.

By looking at the data provided in the calendar, the application computes the iteration boundary for each iteration and estimates the burndown at the end of each day. This allows the developer to understand the overall progress of the project.

The WBS artifact provides a summarization of each iteration with the help of the data available in calendar and other features such as the timelog and the estimate. The summary includes the planned and actual values of the completion date per iteration, effort per iteration, and velocity per iteration.

Once the project construction finishes, the feature also allows the developer to provide the actual values for both estimation and scheduling. This helps the developer understand the progress of the project and helps improve the planning for the next iteration.

The Backlog provides a way for the developer to understand the new burndown for the next iteration. By looking at the planned tasks, completed tasks, and discovered tasks, the burndown for the next iteration can be computed. This information can be used for changing the estimate for the next iteration.

4.3.4 Dashboard

The purpose of this feature is to provide a way to visually analyze the progress of the project using a burndown chart. It provides the overall effort and the tasks left versus the remaining time, helping the developer figure out the outstanding work to be finished before the expected deadline for either each day, iteration and complete project.

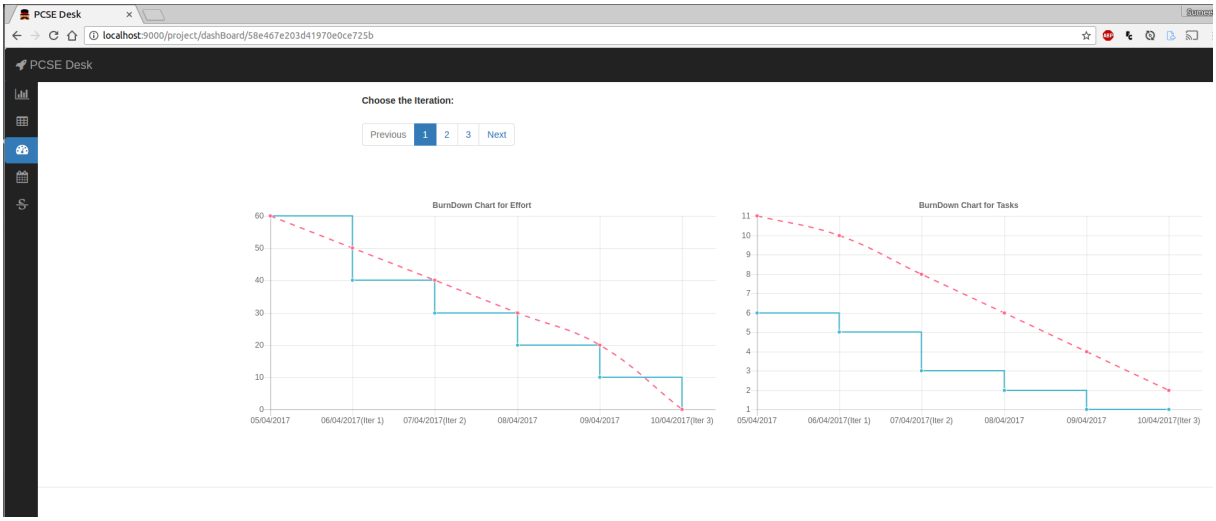

Figure 4.6 PCSE Desk - Project Dashboard

4.3.5 Timelog

The timelog is an important artifact for the developer to record the time spent on each activity in the project life cycle. This feature uses the software process uploaded during the project set up for generating the timelogs. Within each iteration, the developer can access the timelogs and manipulate them. In order to create the timelog, the developer uses the flowchart provided automatically for each iteration. The activities in the flowchart can be selected, as per the activity in progress and a time counter begins computing the time spent on the current activity. Once the activity is finished, the developer can stop the timer and a new timelog gets generated automatically. The developer has the ability to view, edit, save, and delete the timelogs. The timelog records the date, start time, stop time, activity performed, total time taken, interrupt and, comments.

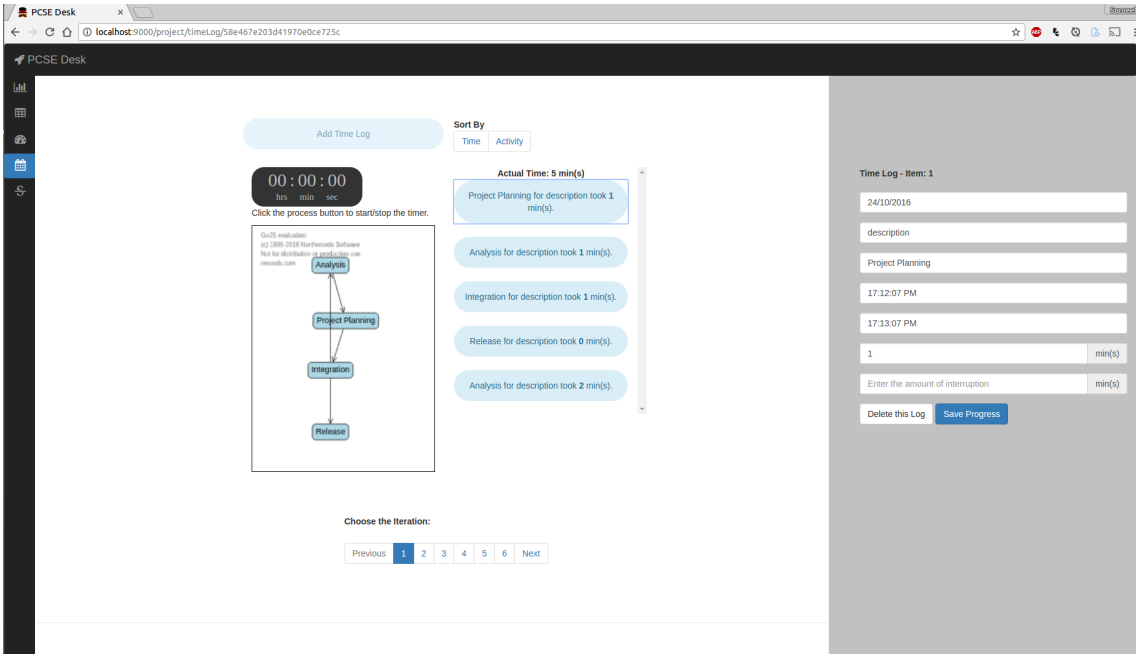

Figure 4.7 PCSE Desk - Project Timelog

4.3.6 Changelog

The changelog is an artifact which helps the developer track defects detected in the project. The defect is recorded whenever the developer encounters an anomaly in the program. The developer provides the necessary details to the changelog feature. The details include the type of defect, injected in which activity, removed in which activity, time to fix the defect, injected in which iteration, removed in which iteration, fix reference for providing a reference to an already existing defect, and a brief description about the defect.

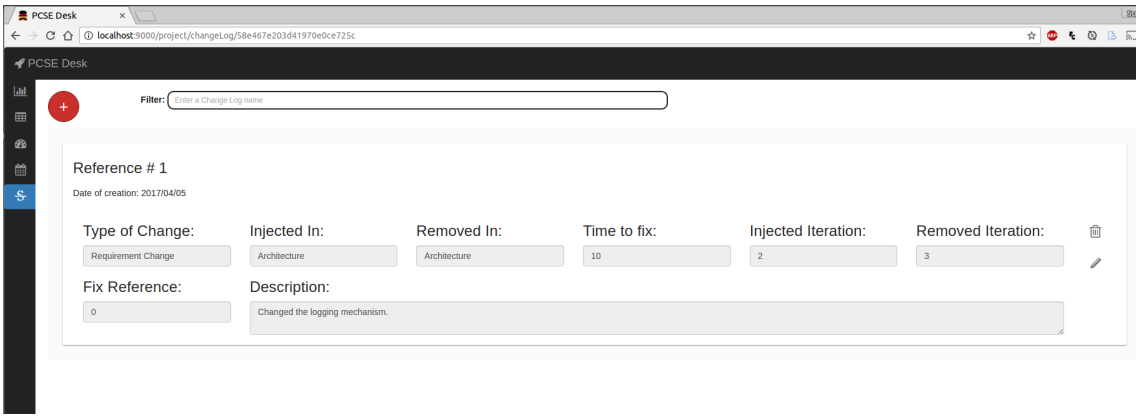

Figure 4.7 PCSE Desk - Project Changelog

4.4 Development of PCSE Desk

PCSE Desk was developed using the process illustrated in figure 4.1. In order to understand the development process completely, we need to understand the activities performed and artifacts produced while using the PCSE for developing PCSE Desk. The activities performed in the development process are as follows:

1. Analysis

While doing analysis for PCSE Desk, we focused on identifying the features of the application. We understood the requirements of the application and broke down the features into their respective components. These components were used for understanding the application features at a high-level and develop scenarios using user stories that would take place using these components. A user story provides a means to write the expectations from the component in plain English text.

Story Set	1		
Objective	To explore the functionality of Add Project component.		
Story	**As a …**	**I want to …**	**So I can …**
Story -1	user (anonymous/logged in)	have the ability to add project name and description	append a new project in the existing list of projects in system.
Story -2	user (anonymous/logged in)	have the ability to edit the project	Either change the project name/description or delete the project itself.
Story -3	user (anonymous/logged in)	view all the existing projects	choose either of them and do the needful tasks within that project scope.
Story -4	software component in PCSEDesk	take the details from user about the project	store them in the system and enable proper services required for a project.

Figure 4.8 PCSE Desk - User Story for Project overview

Figure 4.8 illustrates some of the user stories identified during the initial phase of the first iteration of the project. These user stories explain some of the scenarios of the project overview feature that have to be satisfied by the application for its completion.

2. Architecture

After a successful analysis, we recognized the components of the application, based on the user stories gathered for the application. The user stories identified helped in determining the

components using CRC cards. The user stories were mapped onto their respective components to determine the components that would form the complete feature.

Component Name	Router – Add Project
Design Approach:	
Parent Component:	
Attributes (optional):	
Component Type:	
Collaborators:	Controller – Add Project, View – Add Project
Operations:	Responsible for routing to the proper path of main project page along with respecti

Figure 4.9 CRC card of router for Project overview

Component Name	Controller – Add Project
Design Approach:	
Parent Component:	
Attributes (optional):	
Component Type:	
Collaborators:	Model – Add Project, View – Add Project
Operations:	Responsible for controlling the input data, applying validation and storing for further

Figure 4.10 CRC card of controller for project overview

Component Name	View – Add Project
Design Approach:	
Parent Component:	
Attributes (optional):	
Component Type:	
Collaborators:	Controller – Add Project, Model – Add Project
Operations:	Responsible for generating a view which gives all the abilities covered in user storie

Figure 4.11 CRC card of view for project overview

Figure 4.9 to 4.11 illustrates some of the components identified in the first iteration for the project overview feature. The major components required for building the features in AngularJS corresponded with the components identified in the CRC cards. The one-to-one correspondence between AngularJS and CRC cards helped making the process of architecture seamless.

We also kept the software process flexible enough to accommodate new features. If a new feature was discovered during an iteration, we added it to the existing architecture and identified new user stories for it.

3. Project Plan

The project planning involved estimation which played an important role in the development of the application. The objective of planning the project was to provide the overall effort required for the complete project. All the application features were identified during the project planning.

In order to generate an estimate, a size matrix was required. But, due to the lack of historical data, we made an educated guess for the size matrix and used it to develop an estimate for the first iteration. At the end of the first iteration, we got confidence in the planned effort and planned lines of code for the complete project.

	Estimated				Actual		
New Components	ethod Cou	Rel Size	LOCr	Reuseable?	ethod Cou	LOCa	Reusable?
Router – Add Project	1	S	3	Yes	1	6	Yes
Controller – Add Proje	4	M	68	Yes	5	50	Yes
Model – Add Project	1	VL	552	Yes	1	1	Yes
View – Add Project	1	L	97	Yes	1	64	Yes
app.js	2	M	34	Yes	2	23	Yes
style.css	1	VL	552	Yes	1	200	Yes
index.html	1	L	97	Yes	1	43	Yes
Router – Dashboard	1	S	3	Yes			
Controller – Dashboar	4	L	388	Yes			
View – Dashboard	1	L	97	Yes			
Router – TimeLog	1	S	3	Yes	1	3	Yes
Controller – TimeLog	4	M	68	Yes	8	88	Yes
View – TimeLog	1	M	17	Yes	1	101	Yes
Router – ChangeLog	1	S	3	Yes			
Controller – ChangeL	4	M	68	Yes			
View – ChangeLog	1	M	17	Yes			
Router – Burndown	1	S	3	Yes			
Controller – Burndown	10	VL	5520	Yes			
View – Burndown	1	VL	552	Yes			
			0				
Totals			8142	8142		579	579

Figure 4.12 PCSE Desk - Project estimation at iteration 2

Figure 4.12 illustrates the estimate of the complete project at beginning of the second iteration. We modified the existing estimate and corrected it based on the experience gained during the first iteration.

4. Iteration plan

An iteration plan is a fine-grained plan with a time-sequenced set of activities and tasks, with assigned resources, containing task dependencies, for the iteration. We used iteration map for

distributing the tasks based on the available time for each calendar day. The iterations were kept to one week to give enough time for constructing the complete feature. We divided the tasks based on the feature and its corresponding components in each iteration. The objective of dividing the tasks in such way was to make sure that one feature would be completed in single iteration. If the feature needed more time for completion, we pushed it to the next iteration by changing the iteration map and accommodating it with other proposed tasks already present for the next iteration.

		Iteration01	Iteration02	Iteration03	Iteration04	Iteration05	Iteration06
New Components	Router – Add Project	1					
	Controller – Add Project	5					
	Model – Add Project	1					
	View – Add Project	1					
	app.js	2					
	style.css	1					
	index.html	1					
	Router – Dashboard			1			
	Controller – Dashboard			4			
	View – Dashboard			1			
	Router – TimeLog		1				
	Controller – TimeLog		4				
	View – TimeLog		1				
	Router – ChangeLog				1		
	Controller – ChangeLog				2	1	1
	View – ChangeLog				1		
	Router – Burndown				1		
	Controller – Burndown				5	3	2
	View – Burndown				1		
Number of tasks:		12	6	6	11	4	3
Effort:		514	257	257	471	171	129

Figure 4.13 PCSE Desk - Project iteration map at the beginning of iteration 2

In order to adjust the change in iteration plan, we either added more time in the calendar for the next iteration or added another iteration based on the backlog generated at the end of previous iteration. Figure 4.13 illustrates the iteration map for PCSE desk at the beginning of iteration 2 based on the available time per day and provides a projection for the completion of project at the end of iteration 6.

5. Construction & Review

At the initial stage, the project was set up with the directory structure similar to the illustration in figure 3.1 and each feature of the SPA was constructed in the client folder. Figure 4.14 illustrates the complete client directory structure at the end of the the project.

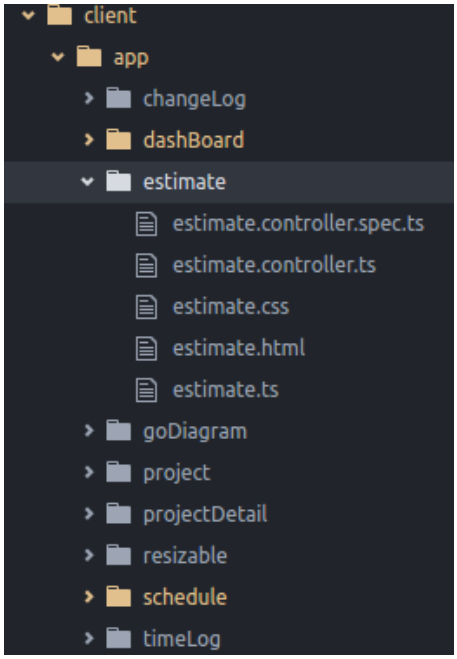

Figure 4.14 PCSE Desk - Directory structure

In order to begin development of PCSE Desk, we started with development of the project overview feature. It involved developing the index.html shell document to bootstrap the SPA. A shell of AngularJS consists of a special directive called “ng-app”. The ng-app directive is usually attached with the body tag of HTML to recognize an AngularJS application. Figure 4.14 illustrates a snippet of shell document.

```
<body ng-app="pcseappApp">
  <!--[if lt IE 7]>
  <p class="browserupgrade">You are using an <strong>o
  <![endif]-->

  <!-- Add your site or application content here -->
  <header></header>
  <div ui-view=""></div>
  <footer></footer>
```

Figure 4.15 PCSE Desk - Snippet of index.html

The “project" directory in figure 4.14 is the complete project overview feature. It consists of a router, a controller, a css file, a view and a testing file.

We used TDD for the construction of PCSE Desk and wrote all the necessary test cases based on the user stories discovered in the analysis for the components involved in development of the project overview feature. We used Jasmine, a Javascript framework for unit testing. When all the possible test cases were covered in the test file, we start constructing the components in order to complete the feature. The construction of an SPA component involves programming AngularJS based template, controller and route.

```
(function(){

  describe('A suite for PCSE', function() {
    beforeEach(module('ngRoute'));
    beforeEach(module('pcseDeskApp'));

    it('contains spec with an expectation', function() {
        expect(true).toBe(true);
        });

    it('has version to be 0.0.1', inject(function(version) {
        expect(version).toBe('0.0.1');
        }));

    describe('A suite for Add Project component', function() {
      var scope, projectCtrl, $httpBackend, fakeProjects;

      beforeEach(inject(function(_$httpBackend_, $rootScope, $controller) {
        $httpBackend = _$httpBackend_;
        fakeProjects = [{name: 'Project Testing', desc: 'gdfgdfg'}];
        $httpBackend.expectGET('Project').respond(fakeProjects);
        scope = $rootScope.$new();
        projectCtrl = $controller('projectController', {$scope: scope});
      }));

      describe('All specs for project Controller', function() {

        it('Should get 1 fake project from data model', function() {
            expect(scope.Projects).toEqual([]);
            $httpBackend.flush();
            expect(scope.Projects).toEqual(fakeProjects);
            expect(scope.Projects.length).toEqual(1);
            });
```

Figure 4.16 PCSE Desk - project overview testing file

For example, in order to construct the project overview feature, the following components were built after writing the test cases in “project.controller.spec.js” as illustrated in figure 4.16:

1. project.html

We constructed the template as if we were constructing a static web page. This helped in mocking the user experience of the feature. Once the mock up was ready, we identified the template sections that required data model or event handling and used various AngularJS components or concepts to construct the application.

```
<div class="modal-body">
<div class="form-group">
  <label>Project name</label>
  <input class="form-control" ng-model="project.name" name="name" id="nam
  <div ng-show="projectForm.name.$invalid && projectForm.name.$touched">
    <small style="color: Red; display: block;">Enter a Valid Project name
  </div>
</div>
<div class="form-group">
  <label>Description</label>
  <textarea class="form-control" ng-model="project.desc" id="desc" placeh
</div>
div>
iv class="modal-footer">
<button class="btn btn-default" type="button" ng-click="$ctrl.reset()" da
<input class="btn btn-primary btn-create" ng-disabled="projectForm.$inval
v>
```

Figure 4.17 PCSE Desk - project overview template

For example, in figure 4.17, the modal was mocked initially without the awareness of AngularJS directives and once the mock design was ready, the AngularJS directives such as ng-modal, ng-disabled, etc. were added to bind the $scope which communicates with the data model present in the controller.

2. project.controller.js

The controller in an SPA is responsible for handling the business logic and is continuously developed along with the template to fulfill the feature requirement. While programming the controller and the template, we continuously tested the functionality to pass the test cases written for the feature. Once all the test cases were passed, we got enough confidence to mark the feature for completion. For example, figure 4.18 presents a snippet of controller for project overview. It consist of the various functions which are responsible for injecting the components, services and user-defined functions.

```
class ProjectComponent {
  constructor($http, $scope, socket, $state) {
    this.$http = $http;
    this.socket = socket;
    this.$scope = $scope;
    this.$state = $state;
    this.$scope.Projects = [];
    this.defaultValues = {
          name : '',
          desc : ''
      };
      $scope.$on('$destroy', function() {
        socket.unsyncUpdates('projects');
      });
  }

  $onInit() {
    // Add a new project
```

Figure 4.18 PCSE Desk - project overview controller

3. project.js

The routing file in a component is responsible for injecting the component based on the URL requested by the user of PCSE Desk. It allows the application to change from one state to another using browser's URL and helps binding the template and the controller associated with the location specified in the URL.

```
'use strict';

angular.module('pcseappApp')
  .config(function ($stateProvider) {
    $stateProvider
      .state('project', {
        url: '/',
        template: '<project></project>'
      });
  });
```

Figure 4.19 PCSE Desk - project overview route

After constructing these files to completion, we reviewed the test code for its coverage and made sure that the feature was tested thoroughly. At this point, the feature was constructed completely.

When the user requests the application at the root level, the application automatically redirects the user to the project overview feature. The routing also makes sure that the transition from one state to another happens without page refresh.

All the other features were constructed in the same way and we made sure that we adhere to the best practices of SPA development mentioned in chapter 3. With each feature with its own route, we made sure that the application developed follows the single page architecture and gives an impression of a native-like application.

4. Post mortem

At the end of each iteration, we understood the difficulties faced during the current iteration and prepared for the next iteration. If there was some backlog from previous iteration, then we changed the next iteration plan and managed the backlog along with existing tasks to meet the project deadlines. If during the construction, new features or components were discovered, then we discussed about these features during the weekly meeting and added them to the plan for upcoming iterations.

4.5 Deployment of PCSE Desk

In order to deploy the application, the application server had to be executed in the operating system. The application's "dist" folder was prepared for distribution using the "*grunt build*" command in the Atom development environment. Building was a one-time process for producing the final application using the grunt task runner and once built, an SPA was available for distribution in the “dist" folder. It was downloaded from a common repository for utilization.

After installing these prerequisites and cloning the repository, PCSE Desk can be used with any browser present on the operating system.

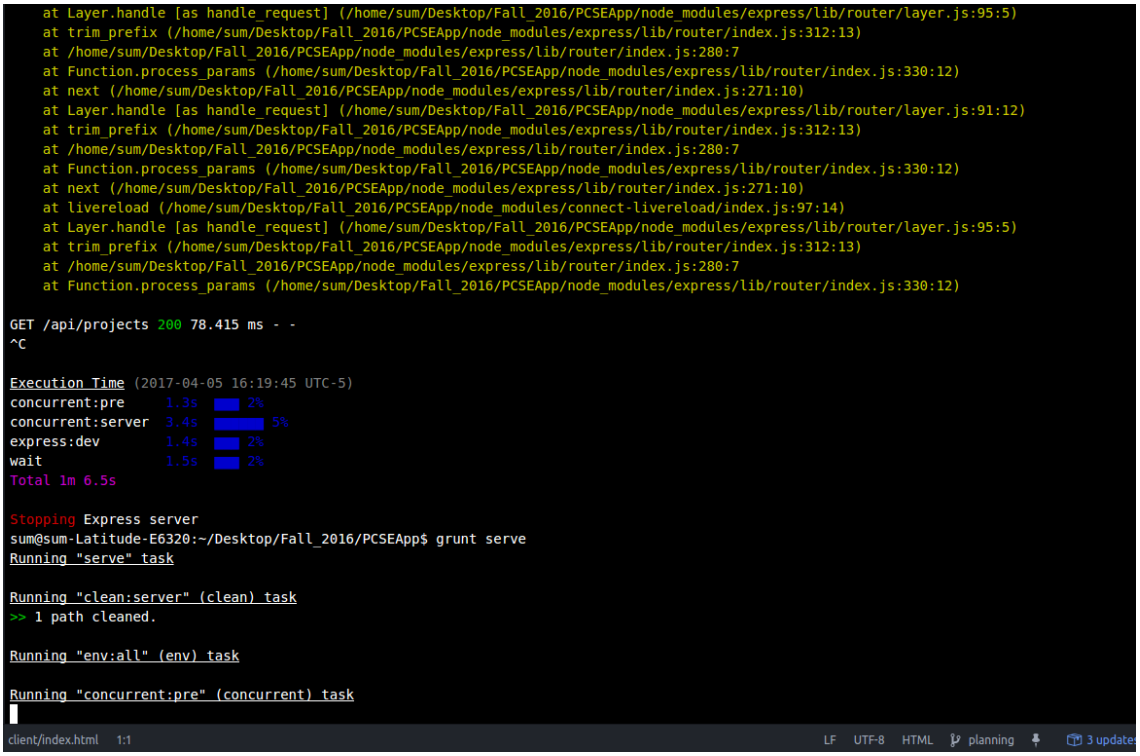

Figure 4.8 Deployment of an SPA using grunt task runner

5. Conclusion and Future work

5.1 Conclusion

This study explored the best practices for developing SPAs. While examining different ways to develop SPAs, AngularJS was found to be a suitable way in conjunction with a light weight process. AngularJS encourages practices such as modularization, separation of concerns, dependency injection, and BDD which helped in making SPAs easier to develop and deploy for utilization. These practices also took advantage from the agility of the light weight process allowing the developer to continuously integrate and deploy SPAs. Using a browser and its capability of using Javascript allowed AngularJS to create an SPA which resides completely on the browser and only uses server for synchronizing the data.

In the present work an SPA, PCSE Desk was developed using AngularJS and PCSE. PCSE Desk works on multiple clients using a browser and provides an experience of a native-like application. As browser never reloads the page to navigate between different features of the SPA, it requires relatively less amount of network and computing bandwidth as compared to traditional web applications such as 3-tier and 2-tier web applications. It easily adapts to different form factors and its performance is found to be consistent across different platforms. It is also observed that the artifacts created by the end user remains synchronized while simultaneously using PCSE Desk from multiple clients. In summary, it is concluded that the single page architecture can be used by the developer to create superior applications in comparison to the architecture of traditional web applications.

5.2 Future Work

Although the application, PCSE Desk, developed to demonstrate the best practices of AngularJS is completed successfully, there are certain enhancements that can provide a developer with a better experience. The following are some of these enhancements:

1. The scheduling feature in planning does not show the iteration boundary for the first iteration, while using calendar. The page has to refreshed in order to view the changes. Implementing the calendar with $watch variable of AngularJS can improve the behavior of the calendar.
2. The visualizations created in the dashboard feature can be enhanced by providing the developers with the ability to manipulate the efforts and tasks in their respective visualizations and view the effect of these changes on the project deadline. This can help

developers get a better idea about the distribution of the tasks and effort, and they can make wise decision about the project completion.

3. While using PCSE Desk, the developer can provide the path to the source code. This information can be used for fetching the actual LOC written by the user for each component. The enhancement of including source code directory of a project can be used for automatically parsing the source code in order to fetch LOC and methods. This can be a great enhancement and improve the overall experience of using PCSE Desk.
4. Also, the application can be enhanced to encompass every activity involved in PCSE. The new features such as Analysis and Architecture can be developed in future and integrated with the existing application to improve the user experience with PCSE Desk.

References

[Fielding 1999] Fielding, R., Gettys J. and Mogul J. , Hypertext Transfer Protocol -- HTTP/1.1, June 1999

[Peacock 2000] Peacock, R. Distributed architecture technologies. IT Professional 2, 3, 2000.

[Hors 2004] Hors, Arnaud Le, Hégaret, Philippe Le, Document Object Model (DOM) Level 3 Core Specification, version 1.0, W3C, April 2004

[Google 2010] AngularJS official documentation by Google, 2010

[Khan 2011] Khan, AI, Qurashi, RJ and Khan, UA A comprehensive study of commonly practiced heavy and light weight software methodologies, arXiv, 2011

[Subramanian 2011] Subramanian, P. A PCSE (Practice Centered Software Engineering) tool for Eclipse environment, Auburn University, 2011

[Takada 2012] Takada, Mikito Single page apps in depth, 2012

[Mikowski 2013] Mikowski, S. and Powell, J.C. Single page web applications, B and W, 2013.

[Umphress 2015] Software Process course notes, Fall 2015, By Dr. David A. Umphress, Computer Science and Software Engineering Department, Auburn University.

[Koppaka 2016] Koppaka, Venkata and Addie, Scott, Single-page Application Frameworks in Enterprise Software Development

[Voort 2017] Voort, Job van der, Benefits of Continuous Integration, DZone.com, 2017

[Yeoman 2017] Yeoman Official documentation for learning by Yeoman, 2017

Appendix 1 (Source code)

index.html

```
<!doctype html>
<!--[if lt IE 7]>      <html class="no-js lt-ie9 lt-ie8 lt-ie7"> <![endif]-->
<!--[if IE 7]>         <html class="no-js lt-ie9 lt-ie8"> <![endif]-->
<!--[if IE 8]>         <html class="no-js lt-ie9"> <![endif]-->
<!--[if gt IE 8]><!--> <html class="no-js"> <!--<![endif]-->
<head>
  <meta charset="utf-8">
  <meta http-equiv="x-ua-compatible" content="ie=edge">
  <base href="/">
  <title> PCSE Desk</title>
  <meta name="description" content="">
  <meta name="viewport" content="width=device-width">
  <link rel="stylesheet" type="text/css" href="bower_components/bootstrap/dist/css/bootstrap.min.css">
  <link rel="stylesheet" type="text/css" href="http://code.jquery.com/ui/1.9.2/themes/base/jquery-ui.css">
  <link rel="stylesheet" type="text/css" href="bower_components/font-awesome/css/font-awesome.css">
  <!-- Place favicon.ico and apple-touch-icon.png in the root directory -->
  <!-- build:css(client) app/vendor.css -->
  <!-- bower:css -->
  <link rel="stylesheet" href="bower_components/angular-material/angular-material.css" />
  <!-- endbower -->
  <!-- endbuild -->
  <!-- build:css({.tmp,client}) app/app.css -->
  <link rel="stylesheet" href="app/app.css">
  <!-- injector:css -->
  <link rel="stylesheet" href="app/app.css">
  <link rel="stylesheet" href="app/changeLog/changeLog.css">
  <link rel="stylesheet" href="app/dashBoard/dashBoard.css">
  <link rel="stylesheet" href="app/estimate/estimate.css">
  <link rel="stylesheet" href="app/project/project.css">
  <link rel="stylesheet" href="app/projectDetail/projectDetail.css">
  <link rel="stylesheet" href="app/schedule/schedule.css">
  <link rel="stylesheet" href="app/timeLog/timeLog.css">
  <link rel="stylesheet" href="components/footer/footer.css">
  <link rel="stylesheet" href="components/modal/modal.css">
  <!-- endinjector -->
  <!-- endbuild -->
</head>
<body ng-app="pcseappApp">
  <!--[if lt IE 7]>
  <p class="browserupgrade">You are using an <strong>outdated</strong> browser. Please <a
href="http://browsehappy.com/">upgrade your browser</a> to improve your experience.</p>
  <![endif]-->

  <!-- Add your site or application content here -->
```

```
<header></header>
<div ui-view=""></div>
<footer></footer>

<!-- Google Analytics: change UA-XXXXX-X to be your site's ID -->
<script>
(function(i,s,o,g,r,a,m){i['GoogleAnalyticsObject']=r;i[r]=i[r]||function(){
  (i[r].q=i[r].q||[]).push(arguments)},i[r].l=1*new Date();a=s.createElement(o),
  m=s.getElementsByTagName(o)[0];a.async=1;a.src=g;m.parentNode.insertBefore(a,m)
})(window,document,'script','https://www.google-analytics.com/analytics.js','ga');

ga('create', 'UA-XXXXX-X');
ga('send', 'pageview');
</script>
<!--[if lt IE 9]>
<script src="bower_components/es5-shim/es5-shim.js"></script>
<script src="bower_components/json3/lib/json3.min.js"></script>
<![endif]-->
<!-- build:js({client,node_modules}) app/vendor.js -->
<!-- bower:js -->
<script src="bower_components/jquery/dist/jquery.js"></script>
<script src="bower_components/angular/angular.js"></script>
<script src="bower_components/bootstrap/dist/js/bootstrap.js"></script>
<script src="bower_components/angular-resource/angular-resource.js"></script>
<script src="bower_components/angular-cookies/angular-cookies.js"></script>
<script src="bower_components/angular-sanitize/angular-sanitize.js"></script>
<script src="bower_components/angular-bootstrap/ui-bootstrap-tpls.js"></script>
<script src="bower_components/lodash/dist/lodash.compat.js"></script>
<script src="bower_components/angular-socket-io/socket.js"></script>
<script src="bower_components/angular-ui-router/release/angular-ui-router.js"></script>
<script src="bower_components/ng-file-upload/ng-file-upload.js"></script>
<script src="bower_components/angular-filter/dist/angular-filter.js"></script>
<script src="bower_components/jquery-ui/jquery-ui.js"></script>
<script src="bower_components/angular-animate/angular-animate.js"></script>
<script src="bower_components/chart.js/dist/Chart.js"></script>
<script src="bower_components/angular-chart.js/dist/angular-chart.js"></script>
<script src="bower_components/angular-ui-sortable/sortable.js"></script>
<script src="bower_components/angular-aria/angular-aria.js"></script>
<script src="bower_components/angular-messages/angular-messages.js"></script>
<script src="bower_components/angular-material/angular-material.js"></script>
<!-- endbower -->
<script src="socket.io-client/socket.io.js"></script>
<!-- endbuild -->
<script src="app/app.js"></script>
<script src="app/goDiagram/go.js"></script>
<!-- build:js({.tmp,client}) app/app.js -->
<!-- injector:js -->
<script src="components/util/util.module.js"></script>
<script src="app/schedule/schedule.controller.js"></script>
```

```
  <script src="app/dashBoard/dashBoard.controller.js"></script>
  <script src="app/dashBoard/dashBoard.js"></script>
  <script src="app/estimate/estimate.controller.js"></script>
  <script src="app/estimate/estimate.js"></script>
  <script src="app/goDiagram/goDiagram.directive.js"></script>
  <script src="app/project/project.controller.js"></script>
  <script src="app/project/project.js"></script>
  <script src="app/projectDetail/projectDetail.controller.js"></script>
  <script src="app/projectDetail/projectDetail.js"></script>
  <script src="app/resizable/resizable.directive.js"></script>
  <script src="app/changeLog/changeLog.js"></script>
  <script src="app/schedule/schedule.js"></script>
  <script src="app/timeLog/timeLog.controller.js"></script>
  <script src="app/timeLog/timeLog.js"></script>
  <script src="components/footer/footer.directive.js"></script>
  <script src="components/header/header.directive.js"></script>
  <script src="components/modal/modal.service.js"></script>
  <script src="components/navbar/navbar.controller.js"></script>
  <script src="components/navbar/navbar.directive.js"></script>
  <script src="components/socket/socket.service.js"></script>
  <script src="app/changeLog/changeLog.controller.js"></script>
  <script src="components/util/util.service.js"></script>
  <script src="app/app.constant.js"></script>
  <!-- endinjector -->
  <!-- endbuild -->
  </body>
  </html>
```

app.js

```
'use strict';

angular.module('pcseappApp', [
  'pcseappApp.constants',
  'ngCookies',
  'ngResource',
  'ngAnimate',
  'ngSanitize',
  'btford.socket-io',
  'ui.router',
  'ui.bootstrap',
  'ngFileUpload',
  'angular.filter',
  'chart.js',
  'ui.sortable',
  'ngMaterial'
])
  .config(function($urlRouterProvider, $locationProvider) {
```

```
    $urlRouterProvider
      .otherwise('/');

    $locationProvider.html5Mode(true);
  });
```

app.css

```
/**
 * Bootstrap Fonts
 */

@font-face {
  font-family: 'Glyphicons Halflings';
  src: url('../bower_components/bootstrap/fonts/glyphicons-halflings-regular.eot');
  src: url('../bower_components/bootstrap/fonts/glyphicons-halflings-regular.eot?#iefix')
format('embedded-opentype'),
  url('../bower_components/bootstrap/fonts/glyphicons-halflings-regular.woff') format('woff'),
  url('../bower_components/bootstrap/fonts/glyphicons-halflings-regular.ttf') format('truetype'),
  url('../bower_components/bootstrap/fonts/glyphicons-halflings-regular.svg#glyphicons_halflingsregular')
format('svg');
}

/**
 *Font Awesome Fonts
 */

@font-face {
  font-family: 'FontAwesome';
  src: url('../bower_components/font-awesome/fonts/fontawesome-webfont.eot?v=4.1.0');
  src: url('../bower_components/font-awesome/fonts/fontawesome-webfont.eot?#iefix&v=4.1.0')
format('embedded-opentype'),
  url('../bower_components/font-awesome/fonts/fontawesome-webfont.woff?v=4.1.0') format('woff'),
  url('../bower_components/font-awesome/fonts/fontawesome-webfont.ttf?v=4.1.0') format('truetype'),
  url('../bower_components/font-awesome/fonts/fontawesome-webfont.svg?v=4.1.0#fontawesomeregular')
format('svg');
  font-weight: normal;
  font-style: normal;
}

/**
 * App-wide Styles
 */

.browserupgrade {
  margin: 0.2em 0;
  background: #ccc;
  color: #000;
```

```
  padding: 0.2em 0;
}

/* Toggle Styles */

 #wrapper {
    padding-left: 0;
    -webkit-transition: all 0.5s ease;
    -moz-transition: all 0.5s ease;
    -o-transition: all 0.5s ease;
    transition: all 0.5s ease;
    height: 100%;
}

div#timeLog{
        cursor: pointer;
        text-align: center;
        border-left:1px solid white;
        border-right:1px solid white;
        border-top:1px solid white;
        border-bottom:1px solid white;
        border-radius: 50px;
        opacity: 0.65;
}

div#tasks{
        cursor: pointer;
        text-align: center;
        border-left:1px solid white;
        border-right:1px solid white;
        border-top:1px solid white;
        border-bottom:1px solid white;
        border-radius: 50px;
}

div#tasks:hover{
        border: 1px solid;
}

#wrapper.toggled {
    padding-left: 250px;
    height: 100%;
    overflow: visible;
}

#sidebar-wrapper {
    z-index: 1000;
    position: fixed;
    left: 250px;
```

```
    width: 0;
    height: 100%;
    margin-left: -250px;
    margin-top: 50px;
    overflow: hidden;
    background: #222;
    -webkit-transition: all 0.5s ease;
    -moz-transition: all 0.5s ease;
    -o-transition: all 0.5s ease;
    transition: all 0.5s ease;
}
#wrapper.toggled #sidebar-wrapper {
    width: 250px;
}

#page-content-wrapper {
    position: absolute;
    padding: 15px;
    width: 100%;
    overflow-x: visible;
    height: 100%;
}

#addButton {
        border-radius: 50px;
        width: 50px;
        height: 50px;
        font-size: 25px;
}

#addProject{
        position: relative;
        padding-bottom: 10px;

}
#filter input:focus {
        outline:none;
        border: 3px Solid #222;
        box-shadow: 0 0 10px #222;
}
#filter input {
        width:50%;
        border-radius: 10px;
        color: #222;
        border: 2px Solid #222;
}
#folder {
        padding-right: 20px;
}
```

```
textarea {
	resize:vertical ;
}

#noProject {
	color: grey;
	display: block;
	text-align: center;
	z-index: -1;
}

#addProject button:focus {
	outline:0;
}
.navbar-header a:focus {
	outline:0;
}
.navbar {
   position: fixed;
   min-height: 50px;
   width: 100%;
   margin-bottom: 20px;
   border: 1px solid transparent;
   z-index: 1000;
}

#wrapper.toggled #page-content-wrapper {
   position: relative;
   margin-right: 0px;
   height: 100%;
}
.fixed-brand{
   width: 100%;
}
/* Sidebar Styles */

.sidebar-nav {
   position: absolute;
   top: 0;
   width: 250px;
   margin: 0;
   padding: 0;
   list-style: none;
   margin-top: 2px;
}

.sidebar-nav li {
   text-indent: 15px;
   line-height: 40px;
```

```
}

.sidebar-nav li a {
    display: block;
    text-decoration: none;
    color: #999999;
    cursor: pointer;
}

.sidebar-nav li a:hover {
    text-decoration: none;
    color: #fff;
    background: rgba(255,255,255,0.2);
    border-left: red 2px solid;
}

.sidebar-nav li a:active,
.sidebar-nav li a:focus {
    text-decoration: none;
    outline:0;
}

.sidebar-nav > .sidebar-brand {
    height: 65px;
    font-size: 18px;
    line-height: 60px;
}

.sidebar-nav > .sidebar-brand a {
    color: #999999;
}

.sidebar-nav > .sidebar-brand a:hover {
    color: #fff;
    background: none;
}
.no-margin{
    margin:0;
}

@media(min-width:480px) {
    #wrapper {
        padding-left: 250px;
        height: 100%;
    }
    .fixed-brand{
        width: 100%;
    }
    #wrapper.toggled {
```

```
    padding-left: 0;
    height: 100%;
}

#sidebar-wrapper {
    width: 250px;
}

#wrapper.toggled #sidebar-wrapper {
    width: 250px;
}
#wrapper.toggled-2 #sidebar-wrapper {
    width: 50px;
}
#wrapper.toggled-2 #sidebar-wrapper:hover {
    width: 250px;
}


#page-content-wrapper {
    padding: 20px;
    position: relative;
    height: 100%;
    -webkit-transition: all 0.5s ease;
    -moz-transition: all 0.5s ease;
    -o-transition: all 0.5s ease;
    transition: all 0.5s ease;
}

#wrapper.toggled #page-content-wrapper {
    position: relative;
    height: 100%;
    margin-right: 0;
    padding-left: 250px;
}
#wrapper.toggled-2 #page-content-wrapper {
    position: relative;
    height: 100%;
    margin-right: 0;
    margin-left: -200px;
    -webkit-transition: all 0.5s ease;
    -moz-transition: all 0.5s ease;
    -o-transition: all 0.5s ease;
    transition: all 0.5s ease;
    width: auto;

}
.container {
  padding-left: 200px;
```

```
   }
}
input.ng-invalid.ng-touched{
        border: 2px solid Red;
}

#filter-label{
        color: #222;
}

#projectRow{
        border-left:1px solid white;
        border-right:1px solid white;
        border-top:1px solid white;
        border-bottom:1px solid white;
        border-radius: 50px;
}
#outer {
        position: fixed;
        /*margin: 0;*/
        /*float: right;*/
  /*right: 0 !important;*/
  right: 0px;
  width: 0px;
  max-width: 500px;
  height: 100%;
  overflow: hidden;
}
#resizable {
        z-index: 900;
        position: absolute;
        /*margin-right: -60px;*/
  /*right: 0px;*/
  /*width: 0px;*/
  float: right;
  height: 100%;
  overflow: hidden;
  padding-top:100px;
  background-color: rgb(193, 193, 193);
}

.container-fluid {
        padding-top: 50px;
  padding-left: 50px;
}
.container {
        padding-top: 50px;
}
#charts-wrapper {
```

```
  margin-left: 300px;
}

#Timer {
        background: #333;
        color: #bbb;
        font-family: Menlo;
        text-align:center;
        font-size:30px;
        height: 100%;
        width: 180px;
        border-radius: 20px;
}
#scrollable {
        height: 400px;
        overflow-y: scroll;
}
#Iteration {
        margin-top: -25px;
}
.dropzone {
  height: 200px;
  border-width: 2px;
  margin-bottom: 20px;
}

.dropzone {
  color: #ccc;
  border-style: dashed;
  border-color: #ccc;
  line-height: 200px;
  text-align: center
}
.dropzone {
  color: #222;
  border-color: #222;
}
```

project.js

```
'use strict';

angular.module('pcseappApp')
  .config(function ($stateProvider) {
    $stateProvider
      .state('project', {
        url: '/',
        template: '<project></project>'
```

```
    });
  });
```

project.controller.js

```
'use strict';
(function(){

class ProjectComponent {
  constructor($http, $scope, socket, $state) {
    this.$http = $http;
    this.socket = socket;
    this.$scope = $scope;
    this.$state = $state;
    this.$scope.Projects = [];
    this.defaultValues = {
                                          name : '',
                                          desc : ''
                        };
      $scope.$on('$destroy', function() {
        socket.unsyncUpdates('projects');
      });
  }

  $onInit() {
    this.$http.get('/api/projects').then(response => {
      this.$scope.Projects = response.data;
      this.socket.syncUpdates('projects', this.$scope.Projects);
    });
  }
  // Add a new project
  addProject(){

    var addProjectItem = {
      name: this.$scope.project.name,
      desc: this.$scope.project.desc
    };
    var req = angular.toJson(addProjectItem);
    this.$http.post('/api/projects', req).success(function(){
      console.log('Post Request sucessful');
    });
    this.reset();
    $('#myModal').modal('toggle');
  }
  reset(){
    this.$scope.projectForm.$setPristine();
    this.$scope.projectForm.$setUntouched();
    this.$scope.project = angular.copy(this.defaultValues);
  }
```

```
  deleteProject(project){
    this.$http.delete('/api/projects/'+ project._id).success(function(){
      console.log('Delete Request sucessful');
    });
  }

  projectAvailable(){
          return this.$scope.Projects.length > 0 ? true:false;
  }

  setSelectProject(project) {
    this.$state.go('projectDetail', {
      id: project._id
    });
  }
}

angular.module('pcseappApp')
  .component('project', {
    templateUrl: 'app/project/project.html',
    controller: ProjectComponent
  });

})();
```

project.html

```
<!-- Modal -->
<div class="modal fade" id="myModal" role="dialog">
          <div class="modal-dialog">
                    <form name="projectForm" ng-submit="$ctrl.addProject()" novalidate>
                    <!-- Modal content-->
                    <div class="modal-content">
                    <div class="modal-header">
                              <button type="button" class="close" data-dismiss="modal">&times;</button>
                              <h4 class="modal-title"> Add New Project</h4>
                    </div>

                    <div class="modal-body">
                                        <div class="form-group">
                                        <label>Project name</label>
                                        <input class="form-control" ng-model="project.name"
name="name" id="name" type="text" ng-required="true" placeholder="Enter a project Name"/>
                                        <div ng-show="projectForm.name.$invalid &&
projectForm.name.$touched">
```

```
                                        <small style="color: Red; display: block;">Enter a Valid
Project name</small>
                                    </div>
                                    </div>
                                    <div class="form-group">
                                    <label>Description</label>
                                    <textarea class="form-control" ng-model="project.desc"
id="desc" placeholder="Enter a project description"></textarea>
                                    </div>
                                </div>
                                <div class="modal-footer">
                                <button class="btn btn-default" type="button" ng-click="$ctrl.reset()"
data-dismiss="modal">Cancel</button>
                                <input class="btn btn-primary btn-create" ng-disabled="projectForm.
$invalid || adding" type="submit" value="Create"/>
                                </div>

                    </div>
            </form>
            </div>
 </div>

<!-- Page Content -->
<div id="page-content-wrapper">
            <div class="container-fluid">
                    <div id="addProject">
                            <div class="row">
                                    <div class="col-sm-1">
                                            <button id="addButton" type="button" class="btn btn-danger"
data-toggle="modal" data-target="#myModal"  data-toggle="tooltip" data-placement="top" title="Add a
New Project">+</button>
                                    </div>
                                    <div class="col-sm-11">
                                            <div id="filter" class="form-group">
                                                    <label id="filter-label" class="form-
group">Filter:</label>
                                            <input id="filter-input" class="form-group input-sm" ng-
model="search" type="text" placeholder="Enter a Project name">
                                            </div>
                                    </div>
                            </div>
                    </div>
                    <div class="row">
                    <div id="projectRow" ng-repeat="project in Projects | filter: search" ng-
show="$ctrl.projectAvailable()" class="col-sm-6 alert alert-success" role="alert">
                            <span id="folder" class="glyphicon glyphicon-folder-open" aria-
hidden="true"></span>
                            <a href="" ng-click="$ctrl.setSelectProject(project)">{{project.name}}</a>
                            <button type="button" class="close" ng-
```

```
click="$ctrl.deleteProject(project)">&times;</button>
                    </div>
                    </div>
                    <div ng-show="!$ctrl.projectAvailable()">
                    <h1 id="noProject">No Projects Available.</h1>
          </div>
          </div>
</div>
```

project.controller.spec.js

```
/* global describe, it, expect, inject, beforeEach */

(function(){

          describe('A suite for PCSE', function() {
                    beforeEach(module('ngRoute'));
                    beforeEach(module('pcseDeskApp'));

                    it('contains spec with an expectation', function() {
                      expect(true).toBe(true);
                      });

                    it('has version to be 0.0.1', inject(function(version) {
                      expect(version).toBe('0.0.1');
                      }));

                    describe('A suite for Add Project component', function() {
                              var scope, projectCtrl, $httpBackend, fakeProjects;

                              beforeEach(inject(function(_$httpBackend_, $rootScope, $controller) {
                                        $httpBackend = _$httpBackend_;
                                        fakeProjects = [{name: 'Project Testing', desc: 'gdfgdfg'}];
                                        $httpBackend.expectGET('Project').respond(fakeProjects);
                                        scope = $rootScope.$new();
                                        projectCtrl = $controller('projectController', {$scope: scope});
                              }));

                              describe('All specs for project Controller', function() {

                                        it('Should get 1 fake project from data model', function() {
                                          expect(scope.Projects).toEqual([]);
                                          $httpBackend.flush();
                                          expect(scope.Projects).toEqual(fakeProjects);
                                          expect(scope.Projects.length).toEqual(1);
                                          });
                                        it('Should add 1 fake project to the data model', function() {
                                                  scope.project = fakeProjects;
                                                  scope.projectForm = {
```

```
                                        name : 'ds',
                                        desc : 'sd'
                                };
                                scope.projectForm.$setPristine = function(){
                                        return true;
                                };
                                scope.projectForm.$setUntouched = function(){
                                        return true;
                                };
                                $httpBackend.flush();
                                scope.addProject();
                                expect(scope.Projects.length).toEqual(2);
                        });
                        it('Should delete 1 fake project from the data model', function() {
                                $httpBackend.flush();
                                scope.deleteProject();
                                expect(scope.Projects.length).toEqual(0);
                        });

                        it('Should not have any projects if data model is empty', function() {
                                scope.projectAvailable();
                                expect(scope.Projects.length).toEqual(0);
                        });

                });
                describe('All specs from Project view', function() {
                        it('Should reset the form to have default values(empty name and
description)', function() {
                                var defaultValues = {
                                        name : '',
                                        desc : ''
                                };
                                scope.projectForm = {
                                        name : 'ds',
                                        desc : 'sd'
                                };
                                scope.projectForm.$setPristine = function(){
                                        return true;
                                };
                                scope.projectForm.$setUntouched = function(){
                                        return true;
                                };
                                scope.project = scope.projectForm;
                                scope.reset();
                           expect(scope.project).toEqual(defaultValues);
                           });
                });
```

```
});

describe('A suite for Time Log component', function() {
	var scope, scope1, timLogCtrl, projectCtrl, $httpBackend, fakeProject, $filter;

	beforeEach(inject(function(_$httpBackend_, $rootScope, Scopes, $controller,
$filter) {
		$httpBackend = _$httpBackend_;
		scope = $rootScope.$new();
		projectCtrl = $controller('projectController', {$scope: scope});
		fakeProject = [{name: 'Project Testing', desc: 'gdfgdfg', timeLog:
[{"startTime":"2016\/04\/20
23:00","dateFormat":"04\/04\/2016","comments":"fghf","activity":"Analysis","interrupt":"55","stopTime"
:"2016\/04\/30 23:00"}]}];
		scope.setSelectProject(fakeProject);
		$httpBackend.expectPOST('TimeLog').respond(fakeProject);
		scope1 = $rootScope.$new();
		timLogCtrl = $controller('timeController', {$scope: scope1});
		scope1.SelectProject = fakeProject;
	}));

	describe('All specs for time Controller', function() {

		it('Should add 1 Time Log for a fake project from data model', function()
{
			expect(scope1.SelectProject.timeLog).toBeUndefined();
			scope1.addTimeLogs();
		expect(scope1.SelectProject.timeLog.length).toEqual(1);
		});
		it('Should open 1 fake project to the data model', function() {
			scope1.addTimeLogs();
			scope1.openLog();
			expect(scope1.SelectProject.timeLog.length).toEqual(1);
			var today = new Date();
			var dd = today.getDate();
		var mm = today.getMonth()+1; //January is 0!

		var yyyy = today.getFullYear();
		if(dd<10){
		    dd='0'+dd
		}
		if(mm<10){
		    mm='0'+mm
		}
		var today = dd+'/'+mm+'/'+yyyy;

	expect(scope1.SelectProject.timeLog[0].dateFormat).toEqual(today);
		});
		it('Should delete 1 fake time log from the data model', function() {
```

```
                                        scope1.addTimeLogs();
                                        scope1.deleteLog();
                                        expect(scope1.SelectProject.timeLog.length).toEqual(0);
                                    });

                            });
                    });

                    });
}());
```

projectDetail.js

```
'use strict';

angular.module('pcseappApp')
  .config(function ($stateProvider) {
    $stateProvider
      .state('projectDetail', {
        url: '/project/:id',
        template: '<project-detail></project-detail>'
      });
  });
```

projectDetail.controller.js

```
'use strict';
(function(){

  class ProjectDetailComponent {
    constructor($http, $scope, socket, $stateParams, Upload) {
      this.$scope = $scope;
      this.$http = $http;
      this.socket = socket;
      this.$stateParams = $stateParams;
      this.Upload = Upload;
      this.$scope.Project = [];
      $scope.iterations = [];
      $scope.totalItems = $scope.iterations.length*10;
      $scope.currentPage = 1;
      $scope.maxSize = 8;
      $scope.itemsPerPage = 1;

      $scope.createModel = function(iteration) {
        //console.log(iteration.projectConfig);
        if(iteration.projectConfig.length > 0) {
          var config = JSON.parse(iteration.projectConfig);
          //console.log(config);
```

```
  $scope.model = new go.GraphLinksModel(
    config[0].nodeDataArray,
    config[0].linkDataArray);
    $scope.model.selectedNodeData = null;
    $scope.mgi = config[1];
    $scope.mep = config[2];
    $scope.isUploadedAfter = true;
  }
  else{
    $scope.isUploadedAfter = false;
  }
}

$scope.pageChanged = function() {
  //console.log($scope.currentPage);
  $scope.createModel($scope.Project.iterations[$scope.currentPage-1]);
}
this.$scope.setupConfig = {};
this.$scope.log = '';
this.$scope.uploadFile;
this.setupActive = 'active';
this.$scope.isConfigured = false;
this.$scope.isUploadedAfter = false;
this.$scope.stringTitle = 'Setup Project Configuration';
this.$scope.model = new go.GraphLinksModel();

this.$scope.upload = function(files, iter) {
  if (iter && files && files.length) {
    for (var i = 0; i < files.length; i++) {
      var file = files[i];
      if(file) {
        Upload.base64DataUrl(file).then(response => {
          console.log(response);
          var str = response.toString().slice(29);
          while(str.charAt(str.length-1) === '='){
            str = str.slice(0, -1);
          }
          //console.log(str);
          $scope.uploadFile = JSON.parse(window.atob(str));
          //console.log($scope.uploadFile);
          //console.log($scope.Project.iterations[iter-1]);
          $scope.Project.iterations[iter-1].isUploaded = true;
          $scope.Project.iterations[iter-1].projectConfig = JSON.stringify($scope.uploadFile);
          var data = {
            iterations: $scope.Project.iterations
          };
          $http.put('/api/projects/'+$stateParams.id, data).then(response => {
            console.log(response);
          });
```

```
            });
            $scope.stringTitle = 'Upload File(s) - Setup Project Configuration';
            $scope.isUploadedAfter = true;
           }
          }
         }
        };
        $scope.$watch('files', function () {
          console.log($scope.setupConfig.noOfIteration);
          if($scope.iterToUpload && $scope.iterToUpload > 0 && $scope.iterToUpload <=
$scope.setupConfig.noOfIteration ){
            $scope.upload([$scope.files], $scope.iterToUpload);
          }
        });
      }

      $onInit() {
        this.getData();
      }
      getData() {
        this.$http.get('/api/projects/'+this.$stateParams.id).then(response => {
          this.$scope.Project = response.data;
          console.log(this.$scope.Project);
          if(this.$scope.Project.Setup){
            //this.$scope.setupConfig.lengthOfIteration = this.$scope.Project.Setup.lengthOfIteration;
            console.log(new Date(this.$scope.Project.Setup.startProject));
            this.$scope.setupConfig.startProject = new Date(this.$scope.Project.Setup.startProject);
            //this.$scope.setupConfig.endProject = new Date(this.$scope.Project.Setup.endProject);
            this.$scope.setupConfig.noOfIteration = this.$scope.Project.Setup.noOfIteration;
            console.log(this.$scope.Project.iterations);
            this.$scope.iterations = this.$scope.Project.iterations;
            this.$scope.totalItems = this.$scope.iterations.length*10;
            this.$scope.createModel(this.$scope.iterations[0]);
            this.$scope.stringTitle = 'Upload File(s) - Setup Project Configuration';

            this.$scope.isConfigured = true;
          }
        });
      }
      saveConfig(numberOfIter) {
        var iter = [];
        console.log(this.$scope.setupConfig.startProject);
        // console.log(Date.parse("2016-01-10"));
        // var timeDiff = this.$scope.setupConfig.endProject - this.$scope.setupConfig.startProject;
        // var numberOfDays = Math.floor(timeDiff/(60*60*24*1000));
        // var numberOfIter = numberOfDays / this.$scope.setupConfig.lengthOfIteration;
        for(var i=0; i<numberOfIter; i++) {
          iter.push({
            projectConfig: '',
```

```
          isUploaded: false,
          timeLog: []
        });
      }
      var data = {
        Setup: {
          startProject: this.$scope.setupConfig.startProject,
          noOfIteration: numberOfIter
        },
        iterations: iter,
        changeLog: []
      };
      this.$http.put('/api/projects/'+this.$stateParams.id, data).then(response => {
        console.log(response);
      });
      this.getData();
      this.$scope.isConfigured = true;
    }



  }

  angular.module('pcseappApp')
  .component('projectDetail', {
    templateUrl: 'app/projectDetail/projectDetail.html',
    controller: ProjectDetailComponent
  });

})();
```

projectDetail.css

```
.drop-box {
  background: #F8F8F8;
  border: 5px dashed #DDD;
  width: auto;
  height: 100px;
  text-align: center;
  padding-top: 25px;
  margin: 10px;
}
.dragover {
  border: 5px dashed blue;
}
```

projectDetail.html

```
<navbar></navbar>
```

```
<div id="page-content-wrapper">
	<div class="container">
		<div class="row">
			<div class="col-md-6">
				<div class="panel panel-primary">
					<div class="panel-heading">
						Setup Configuration
					</div>
					<div class="panel-body">
						<form name="SetupConfig" ng-
submit="$ctrl.saveConfig(1)">
							<div class="form-group">
								<label>Enter some details about the
project</label>
							</div>
							<div class="form-group">
								<input class="form-control" ng-
model="setupConfig.startProject" type="date" ng-required="true" placeholder="Enter a start date for this
Project"/>
							</div>
							<!-- <div class="form-group">
								<input class="form-control" ng-
model="setupConfig.endProject" type="date" ng-required="true" placeholder="Enter an end date for this
project"/>
							</div> -->
							<div class="form-group">
								<label>Number of Iterations</label>
								<input class="form-control" ng-
model="setupConfig.noOfIteration" type="text" ng-readonly="true" placeholder="Enter the number of
iterations"/>
							</div>
							<!-- <div class="form-group">
								<div class="input-group">
									<input class="form-control" ng-
model="setupConfig.lengthOfIteration" type="text" ng-required="true" placeholder="Enter the number
of days in each iteration"/>
									<span class="input-group-
addon">days(s)</span>
								</div>
							</div> -->

							<div class="form-group">
								<input class="btn btn-primary btn-
create" ng-disabled="SetupConfig.$invalid" type="submit" value="Save Configuration"/>
							</div>
						</form>
					</div>
					<div class="panel-footer">
					</div>
```

```
                    </div>
                </div>
                <div ng-show="isConfigured" class="col-md-6">
                    <div class="panel panel-primary">
                        <div class="panel-heading">
                            {{stringTitle}}
                        </div>
                        <div class="panel-body">
                            <div class="form-group">
                                <input class="form-control" type="text" ng-
model="iterToUpload" ng-required="true" placeholder="Enter the iteration number to upload for."/>
                            </div>

                            <div ngf-drop ngf-select ng-model="files" class="drop-
box" ngf-change="readFile($files)"
                            ngf-drag-over-class="'dragover'" ngf-multiple="false"
ngf-allow-dir="false"
                            accept="application/json"
                            ngf-pattern="'application/json'">Drop JSON
configuration file here or click to upload</div>

                            <div ngf-no-file-drop>File Drag/Drop is not supported
for this browser</div>

                        </div>
                        <div class="panel-footer">
                        </div>
                    </div>
                </div>
            </div>
            <div ng-show="isConfigured" class="panel-body">
                <label>Choose the Iteration:</label>
                <div><ul uib-pagination total-items="totalItems" ng-model="currentPage" max-
size="maxSize" ng-change="pageChanged()"></ul></div>
            </div>
            <div ng-repeat="iteration in iterations.slice(((currentPage-1)*itemsPerPage),
((currentPage)*itemsPerPage))">
                <div class="row">
                    <div ng-show="isUploadedAfter" class="col-md-12">
                        <div class="panel panel-primary">
                            <div class="panel-heading">
                                Minimal Viable Process
                            </div>
                            <div class="panel-body">
                                <go-diagram go-model="model"
style="width:100%; height:400px"></go-diagram>
                            </div>
                            <div class="panel-footer">
                            </div>
```

```
                                    </div>
                                </div>
                            </div>
                            <!-- /.row -->
                            <div ng-show="isUploadedAfter" class="row">
                                <div class="col-md-6">
                                    <div class="panel panel-primary">
                                        <div class="panel-heading">
                                            Minimal Guiding Indicators
                                        </div>
                                        <div class="panel-body">
                                            <p><label>Cost:</label> {{mgi.cost}}</p>
                                            <p><label>Schedule:</label>
{{mgi.schedule}}</p>
                                            <label>Performance:</label>
                                            <ul ng-repeat="(key, value) in
mgi.performance">
                                                <li><label>{{key}}:</label>
{{value}}</li>
                                            </ul>
                                        </div>
                                        <div class="panel-footer">
                                        </div>
                                    </div>
                                </div>
                                <div ng-show="isUploadedAfter" class="col-md-6">
                                    <div class="panel panel-primary">
                                        <div class="panel-heading">
                                            Minimal Effective Practices
                                        </div>
                                        <div class="panel-body">
                                            <div ng-repeat="(key, value) in mep">
                                                <label>{{key}}:</label>
                                                <ul>
                                                    <li ng-repeat="items in
value">{{items}}</li>
                                                </ul>
                                            </div>
                                        </div>
                                        <div class="panel-footer">
                                        </div>
                                    </div>
                                </div>
                            </div>
                        </div>
                    </div>
                </div>
```

timeLog.js

```
'use strict';

angular.module('pcseappApp')
  .config(function ($stateProvider) {
    $stateProvider
      .state('timeLog', {
        url: '/project/timeLog/:id',
        template: '<time-log></time-log>'
      });
  });
```

timeLog.controller.js

```
'use strict';
(function(){

  class TimeLogComponent {
    constructor($http, $scope, $state, $stateParams, $filter) {
      this.timeActive = 'active';
      this.$http = $http;
      this.$state = $state;
      this.$scope = $scope;
      this.$stateParams = $stateParams;
      this.$filter = $filter;
      this.$scope.model = new go.GraphLinksModel();
      this.$scope.visible = false;
      this.logView = true;
      this.$scope.indexedItems = [];

      $scope.iterations = [];
      $scope.totalItems = $scope.iterations.length*10;
      $scope.currentPage = 1;
      $scope.maxSize = 8;
      $scope.itemsPerPage = 1;
      $scope.pageChanged = function() {
        //console.log($scope.currentPage);
        $scope.initGoJs($scope.Project.iterations[$scope.currentPage-1]);
        document.getElementById('outer').style.width = '0px';
      }

      $scope.initGoJs = function(iter) {
        if(iter.projectConfig && iter.projectConfig.length >= 0){
          var config = JSON.parse(iter.projectConfig);
          $scope.model = new go.GraphLinksModel(
            config[0].nodeDataArray,
            config[0].linkDataArray);
            $scope.model.selectedNodeData = null;
            $scope.visible = true;
```

```
        }
        else{
          $scope.visible = false;
        }
      }

    }
    $onInit() {
      this.$http.get('/api/projects/'+this.$stateParams.id).then(response => {
        this.$scope.Project = response.data;
        this.$scope.iterations = this.$scope.Project.iterations;
        this.$scope.totalItems = this.$scope.iterations.length*10;
        if(this.$scope.iterations.length > 0 ){
            this.$scope.initGoJs(this.$scope.iterations[0]);
        }
      });
    }

    $onDestroy() {
      console.log('Destroying TimeLog Controller!!');
    }


    addTimeLogs(time, selActivity){
      if(this.$scope.Project.iterations[this.$scope.currentPage-1].hasOwnProperty('timeLog')){
        console.log('Yeah!!');
      }
      else{
        this.$scope.Project.iterations[this.$scope.currentPage-1].timeLog = [];
        console.log(this.$scope.Project);
      }
      this.$scope.date = this.$filter('date')(new Date(), "MM/dd/yyyy");
      var dateTime = new Date();
      var starting = this.$filter('date')(dateTime, "HH:mm:ss a");
      var stop = this.$filter('date')(new Date(dateTime.getTime() + 60000*Math.ceil(time)), "HH:mm:ss
a");
      var emptyLog = {
        dateFormat : this.$scope.date,
        timeTaken: Math.ceil(time),
        startTime: starting,
        stopTime: stop,
        activity: selActivity,
        interrupt:",
        comments: "
      };
      this.$scope.Project.iterations[this.$scope.currentPage-1].timeLog.push(emptyLog);
      console.log(this.$scope.Project.iterations[this.$scope.currentPage-1].timeLog);

    }
```

```
    switchView(val){
      this.logView = val;
      console.log(this.logView);
    }

    projectLength(){
      return this.$scope.Project ? this.$scope.Project.iterations.length > 0 : false;
    }

    totalTimeTaken(iter){
      this.$scope.total = 0;
      for(var i=0; i<iter.timeLog.length; i++){
        this.$scope.total += (iter.timeLog[i].timeTaken - iter.timeLog[i].interrupt);
      }
      console.log("the total: "+this.$scope.total);
      return true;
    }

    timePerActivity(iter, activity){
      var total = 0;
      for(var i=0; i < iter.timeLog.length; i++){
        if(activity == iter.timeLog[i].activity){
          total += (iter.timeLog[i].timeTaken - iter.timeLog[i].interrupt);
        }
      }
      return total;
    }
    openLog(logs){
      console.log('The item clicked is: ');
      console.log(logs);
      this.$scope.selectedLog = logs;
      console.log(this.$scope.selectedLog);
      this.$scope.indexVal = this.$scope.Project.iterations[this.$scope.currentPage-
1].timeLog.indexOf(logs)+1;
      $("#outer").css('width', '500px');
    }
    deleteLog(){
      this.$scope.Project.iterations[this.$scope.currentPage-1].timeLog.splice(this.$scope.indexVal-1, 1);
      $("#outer").css('width', '0px');
    }

    checkIfNaN(){
      if(isNaN(this.$scope.selectedLog.interrupt) ||
      this.$scope.selectedLog.timeTaken < this.$scope.selectedLog.interrupt){
        this.$scope.selectedLog.interrupt = '';
      }
    }
    saveTimeLog() {
```

```
      console.log(this.$scope.Project);
      var data = {
        iterations: this.$scope.Project.iterations
      };
      this.$http.put('/api/projects/'+this.$stateParams.id, data).then(response => {
        console.log(response);
      });
    }
  }

  angular.module('pcseappApp')
  .component('timeLog', {
    templateUrl: 'app/timeLog/timeLog.html',
    controller: TimeLogComponent
  });

})();
```

timeLog.html

```
<navbar></navbar>

<div class="container-fluid">
  <div class="row">
    <div class="col-lg-10 col-md-10">
      <div id="page-content-wrapper">
        <div ng-repeat="iteration in iterations.slice(((currentPage-1)*itemsPerPage),
((currentPage)*itemsPerPage))">
          <div class="container" id="leftTimeLog">
            <div class="row">
              <div class="col-sm-4 alert alert-info" id="timeLog" ng-click="initGoJs(iteration)">
                <span>Add Time Log</span>
              </div>
              <div class="col-sm-4">
                <div class="row">
                  <div class="col-sm-12">
                  </div>
                </div>
                <div class="row">
                  <div class="col-sm-12">
                    <label>Sort By</label>
                    <div id="Iteration">
                      <nav>
                        <ul class="pagination">
                          <li><a href="" ng-click="$ctrl.switchView(true)">Time</a></li>
                          <li><a href="" ng-click="$ctrl.switchView(false)">Activity</a></li>
                        </ul>
                      </nav>
                    </div>
```

```
            </div>
          </div>
        </div>
      </div>
      <div class="row">
        <div class="col-md-4">
          <div id="Timer">
            <div>
              <span id="hour" style="padding-left: 5px; padding-right: 5px;">00</span>:<span id="min"
style="padding-left: 5px; padding-right: 5px;">00</span>:<span id="demo" style="padding-left: 5px;
padding-right: 5px;">00</span>
            </div>
            <div style="margin-top:-20px;">
              <span style=" font-size: 15px; padding-left: 10px; padding-right: 10px;">hrs</span><span
style="font-size: 15px; padding-left: 10px; padding-right: 10px;">min</span><span style="font-size:
15px; padding-left: 10px; padding-right: 10px;">sec</span>
            </div>
          </div>
          <p id="demo1">Click the process button to start/stop the timer.</p>
          <go-diagram ng-show="visible" go-model="model" style="border:1px solid black;
width:100%; height:400px"></go-diagram>
        </div>
        <div ng-show="$ctrl.logView && $ctrl.projectLength() && $ctrl.totalTimeTaken(iteration)"
id="scrollable" class="col-md-4">
          <div align="center"><b>Actual Time: {{total}} min(s)</b></div>
          <div ng-repeat="timeLogs in iteration.timeLog | orderBy: 'dateFormat' | orderBy: 'startTime'
track by $index">
            <div class="col-lg-12 alert alert-info" ng-click="$ctrl.openLog(timeLogs)" id="tasks">
              <span>{{timeLogs.activity}} <span ng-show="timeLogs.comments.length > 0">for
{{timeLogs.comments}}</span> took <b>{{timeLogs.timeTaken - timeLogs.interrupt}}</b>
min(s).</span>
            </div>
          </div>
        </div>
        <div ng-show="!$ctrl.logView && $ctrl.projectLength()" id="scrollable" class="col-md-4">
          <!-- <div ng-repeat="timeLogs in SelectProject.timeLog | orderBy: 'activity' track by $index">
          <div class="col-sm-4 alert alert-info" ng-click="openLog($index)" id="tasks">
          <span>{{timeLogs.activity}} - {{timeLogs.comments}}</span>
        </div>
      </div> -->
      <div ng-repeat="(key, value) in iteration.timeLog | groupBy: 'activity'">
        <div align="center"><b>{{value[0].activity}} : {{$ctrl.timePerActivity(iteration,
value[0].activity)}} min(s)</b></div>
        <div id="tasks" class="col-lg-12 alert alert-info" ng-click="$ctrl.openLog(Logs)" ng-
repeat="Logs in value">
          Duration: {{Logs.startTime}} - {{Logs.stopTime}} & time taken: <b>{{Logs.timeTaken -
Logs.interrupt}}</b> min(s).
        </div>
      </div>
```

```
        </div>
       </div>
     </div>
   </div>
  </div>
</div>
<div class="col-lg-2 col-md-2">
  <div id="outer">
    <div resizable id="resizable">
      <div class="container-fluid">
       <form name="selectedLog" ng-submit="$ctrl.saveTimeLog()">
         <div class="form-group">
           <label>Time Log - Item: {{ indexVal }}</label>
         </div>
         <div class="form-group">
           <input class="form-control" ng-model="selectedLog.dateFormat" type="text" ng-required="true"
placeholder="Enter a date for this log"/>
         </div>
         <div class="form-group">
           <input class="form-control" value="{{selectedLog.comments}}" ng-
model="selectedLog.comments" type="text" placeholder="Enter a comment about this log"/>
         </div>
         <div class="form-group">
           <input class="form-control" value="{{selectedLog.activity}}" ng-model="selectedLog.activity"
type="text" ng-required="true" placeholder="Enter a process activity for this log"/>
         </div>
         <div class="form-group">
           <input class="form-control" value="{{selectedLog.startTime | date: 'HH:mm:ss'}}" ng-
model="selectedLog.startTime" type="text" ng-required="true" placeholder="Enter a start time for this
log"/>
         </div>
         <div class="form-group">
           <input class="form-control" value="{{selectedLog.stopTime | date: 'HH:mm:ss'}}" ng-
model="selectedLog.stopTime" type="text" ng-required="true" placeholder="Enter a stop time for this
log"/>
         </div>
         <div class="form-group">
           <div class="input-group">
             <input class="form-control" value="{{selectedLog.timeTaken}}" ng-
model="selectedLog.timeTaken" type="text" ng-required="true" placeholder="Enter the time taken for
this log"/>
             <span class="input-group-addon">min(s)</span>
           </div>
         </div>
         <div class="form-group">
           <div class="input-group">
             <input class="form-control" value="{{selectedLog.interrupt}}" ng-change="$ctrl.checkIfNaN()"
ng-model="selectedLog.interrupt" type="text" placeholder="Enter the amount of interruption"/>
             <span class="input-group-addon">min(s)</span>
```

```
          </div>
        </div>
        <div class="form-group">
          <button class="btn btn-default" type="button" ng-click="$ctrl.deleteLog()">Delete this
Log</button>
          <input class="btn btn-primary btn-create" ng-disabled="timeLogForm.$invalid" type="submit"
value="Save Progress"/>
        </div>
      </form>
    </div>
  </div>
 </div>
</div>
</div>
<div class="container" id="leftTimeLog" ng-show="visible">
 <label>Choose the Iteration:</label>
 <div><ul uib-pagination total-items="totalItems" ng-model="currentPage" max-size="maxSize" ng-
change="pageChanged()"></ul></div>
</div>
</div>
```

dashBoard.js

```
'use strict';

angular.module('pcseappApp')
 .config(function ($stateProvider) {
   $stateProvider
     .state('dashBoard', {
       url: '/project/dashBoard/:id',
       template: '<dash-board></dash-board>'
     });
 });
```

dashBoard.controller.js

```
'use strict';
(function(){

 class DashBoardComponent {
   constructor($http, $scope, $state, $stateParams, $filter) {
     this.$http = $http;
     this.dashActive = 'active';
     this.$state = $state;
     this.$scope = $scope;
     this.$stateParams = $stateParams;
     this.$filter = $filter;
     this.iterationEnd = 0;
     this.$scope.message = 'Dashboard';
```

```
    $scope.visible = true;
    $scope.iterations = [];
    $scope.totalItems = $scope.iterations.length*10;
    $scope.currentPage = 1;
    $scope.maxSize = 8;
    $scope.itemsPerPage = 1;
    $scope.pageChanged = function() {
      var iterationEnd = $scope.currentPage-1;
      var len = $scope.iterations[$scope.currentPage-1].schedule.calendar.length;
      var start = $scope.iterations[$scope.currentPage-
1].schedule.calendar[0].ActualBurnDownStartOfDay;
      var difference = start/len;
      var len1 = 0;
      var cumulativeVelocity = 0;
      var cumulativeEarnedVelocity = 0;
      $scope.labels = [];
      $scope.data = [
       [],
       []
      ];
      $scope.labels1 = [];
      $scope.data1 = [
       [],
       []
      ];
      $scope.iterations[$scope.currentPage-1].schedule.calendar.forEach(function(day, index) {
       if(new Date(day.date).getTime() == new Date($scope.iterations[$scope.currentPage-
1].schedule.wbs[iterationEnd].plannedCompletionDate).getTime()) {
         $scope.labels.push($filter('date')(day.date, "dd/MM/yyyy")+'(Iter '+(iterationEnd+1)+')');
         $scope.labels1.push($filter('date')(day.date, "dd/MM/yyyy")+'(Iter '+(iterationEnd+1)+')');
         iterationEnd++;
       }
       else {
         $scope.labels.push($filter('date')(day.date, "dd/MM/yyyy"));
         $scope.labels1.push($filter('date')(day.date, "dd/MM/yyyy"));
       }
       if(index == len-1) {
         $scope.data[0].push(0);
       }
       else{
         $scope.data[0].push(start);
       }
       if(index == 0) {
         $scope.data[1].push(day.ActualBurnDownStartOfDay);
       }
       else {
         $scope.data[1].push(day.ActualBurnDownEndOfDay);
       }
       start -= difference;
```

```
        len1 += 1;
        difference = start/(len-len1);
        cumulativeVelocity += day.PlannedVelocityEndOfDay;
        cumulativeEarnedVelocity += day.EarnedVelocityEndOfDay;

      });

      $scope.iterations[$scope.currentPage-1].schedule.calendar.forEach(function(day, index) {
        $scope.data1[0].push(cumulativeVelocity);
        $scope.data1[1].push(cumulativeEarnedVelocity);
        cumulativeVelocity -= day.PlannedVelocityEndOfDay;
        cumulativeEarnedVelocity -= day.EarnedVelocityEndOfDay;
      });

      $scope.series = ['Ideal BurnDown', 'Actual BurnDown'];
      $scope.series1 = ['Planned Tasks', 'Actual Tasks'];
      $scope.colors = ['#ff6384', '#45b7cd'];
      $scope.colors1 = ['#ff6384', '#45b7cd'];

      $scope.onClick = function (points, evt) {
        console.log(points, evt);
      };
      $scope.onClick1 = function (points, evt) {
        console.log(points, evt);
      };
      $scope.datasetOverride = [{ yAxisID: 'tasksDone', fill: false, borderDash: [8] }, { XAxisID:
'BurnDown', fill: false, steppedLine: true}, { yAxisID: 'tasksDone', fill: false, borderDash: [8] },
{ XAxisID: 'BurnDown', fill: false, steppedLine: true}];
      $scope.datasetOverride1 = [{ yAxisID: 'tasksDone', fill: false, borderDash: [8] }, { XAxisID:
'BurnDown', fill: false, steppedLine: true}, { yAxisID: 'tasksDone', fill: false, borderDash: [8] },
{ XAxisID: 'BurnDown', fill: false, steppedLine: true}];

      $scope.options = {
        title: {
          display: true,
          text: 'BurnDown Chart for Effort'
        },
        scales: {
          yAxes: [
            {
              id: 'tasksDone',
              type: 'linear',
              display: true,
              position: 'left'
            }
          ]
        }
      };
      $scope.options1 = {
```

```
      title: {
        display: true,
        text: 'BurnDown Chart for Tasks'
      },
      scales: {
        yAxes: [
          {
            id: 'tasksDone',
            type: 'linear',
            display: true,
            position: 'left'
          }
        ]
      }
    };
  }
}

$onInit() {
  this.getData();
}

getData() {
  this.$http.get('/api/projects/'+this.$stateParams.id).then(response => {
    this.$scope.Project = response.data;
    this.$scope.iterations = this.$scope.Project.iterations;
    this.$scope.totalItems = this.$scope.iterations.length*10;
    console.log(this.$scope.iterations[0].schedule.calendar);
    var len = this.$scope.iterations[0].schedule.calendar.length;
    var start = this.$scope.iterations[0].schedule.calendar[0].ActualBurnDownStartOfDay;
    var difference = start/len;
    var len1 = 0;
    var cumulativeVelocity = 0;
    var cumulativeEarnedVelocity = 0;
    this.$scope.labels = [];
    this.$scope.data = [
      [],
      []
    ];
    this.$scope.labels1 = [];
    this.$scope.data1 = [
      [],
      []
    ];
    this.$scope.iterations[0].schedule.calendar.forEach(function(day, index) {

      if(new Date(day.date).getTime() == new Date(this.
$scope.iterations[0].schedule.wbs[this.iterationEnd].plannedCompletionDate).getTime()) {
        this.$scope.labels.push(this.$filter('date')(day.date, "dd/MM/yyyy")+'(Iter '+
```

```
(this.iterationEnd+1)+')');
          this.$scope.labels1.push(this.$filter('date')(day.date, "dd/MM/yyyy")+'(Iter '+
(this.iterationEnd+1)+')');
          this.iterationEnd++;
        }
        else {
          this.$scope.labels.push(this.$filter('date')(day.date, "dd/MM/yyyy"));
          this.$scope.labels1.push(this.$filter('date')(day.date, "dd/MM/yyyy"));
        }


        if(index == len-1) {
          this.$scope.data[0].push(0);
        }
        else{
          this.$scope.data[0].push(start);
        }
        if(index == 0) {
          this.$scope.data[1].push(day.ActualBurnDownStartOfDay);
        }
        else {
          this.$scope.data[1].push(day.ActualBurnDownEndOfDay);
        }
        start -= difference;
        len1 += 1;
        difference = start/(len-len1);
        cumulativeVelocity += day.PlannedVelocityEndOfDay;
        cumulativeEarnedVelocity += day.EarnedVelocityEndOfDay;

      }, this);
      this.$scope.iterations[0].schedule.calendar.forEach(function(day, index) {
        this.$scope.data1[0].push(cumulativeVelocity);
        this.$scope.data1[1].push(cumulativeEarnedVelocity);
        cumulativeVelocity -= day.PlannedVelocityEndOfDay;
        cumulativeEarnedVelocity -= day.EarnedVelocityEndOfDay;
      }, this);

      this.$scope.series = ['Ideal BurnDown', 'Actual BurnDown'];
      this.$scope.series1 = ['Planned Tasks', 'Actual Tasks'];
      this.$scope.colors = ['#ff6384', '#45b7cd'];
      this.$scope.colors1 = ['#ff6384', '#45b7cd'];

      this.$scope.onClick = function (points, evt) {
        console.log(points, evt);
      };
      this.$scope.onClick1 = function (points, evt) {
        console.log(points, evt);
      };
      this.$scope.datasetOverride = [{ yAxisID: 'tasksDone', fill: false, borderDash: [8] }, { XAxisID:
```

```
'BurnDown', fill: false, steppedLine: true}, { yAxisID: 'tasksDone', fill: false, borderDash: [8] },
{ XAxisID: 'BurnDown', fill: false, steppedLine: true}];
      this.$scope.datasetOverride1 = [{ yAxisID: 'tasksDone', fill: false, borderDash: [8] }, { XAxisID:
'BurnDown', fill: false, steppedLine: true}, { yAxisID: 'tasksDone', fill: false, borderDash: [8] },
{ XAxisID: 'BurnDown', fill: false, steppedLine: true}];

      this.$scope.options = {
        title: {
          display: true,
          text: 'BurnDown Chart for Effort'
        },
        scales: {
          yAxes: [
            {
              id: 'tasksDone',
              type: 'linear',
              display: true,
              position: 'left'
            }
          ]
        }
      };
      this.$scope.options1 = {
        title: {
          display: true,
          text: 'BurnDown Chart for Tasks'
        },
        scales: {
          yAxes: [
            {
              id: 'tasksDone',
              type: 'linear',
              display: true,
              position: 'left'
            }
          ]
        }
      };
    });

  }

}

angular.module('pcseappApp')
.component('dashBoard', {
  templateUrl: 'app/dashBoard/dashBoard.html',
  controller: DashBoardComponent
});
```

})();

dashBoard.html

```
<navbar></navbar>
<div id="page-content-wrapper">
        <div class="container" ng-show="visible">
   <label>Choose the Iteration:</label>
   <div><ul uib-pagination total-items="totalItems" ng-model="currentPage" max-size="maxSize" ng-
change="pageChanged()"></ul></div>
  </div>
        <div ng-repeat="iteration in iterations.slice(((currentPage-1)*itemsPerPage),
((currentPage)*itemsPerPage))">
                <div class="container-fluid">
                        <div id="charts-wrapper">
                                <div class="row">
                                        <div class="col-md-6">
                                                <canvas id="line" class="chart chart-line" chart-
data="data"
                                                chart-labels="labels" chart-series="series" chart-
options="options" chart-colors="colors"
                                                chart-dataset-override="datasetOverride" chart-
click="onClick">
                                                </canvas>
                                        </div>
                                        <div class="col-md-6">
                                                <canvas id="line" class="chart chart-line" chart-
data="data1"
                                                chart-labels="labels1" chart-series="series1" chart-
options="options1" chart-colors="colors1"
                                                chart-dataset-override="datasetOverride1" chart-
click="onClick1">
                                                </canvas>
                                        </div>
                                </div>
                        </div>
                </div>
        </div>
</div>
```

changeLog.js

```
'use strict';

angular.module('pcseappApp')
  .config(function ($stateProvider) {
    $stateProvider
      .state('changeLog', {
```

```
      url: '/project/changeLog/:id',
      template: '<change-log></change-log>'
    });
  });
```

changeLog.controller.js

```
'use strict';
(function(){

class ChangeLogComponent {
  constructor($http, $scope, $state, $stateParams) {
    this.$http = $http;
    this.$state = $state;
    this.$scope = $scope;
    this.$stateParams = $stateParams;
    this.changeActive = 'active';
    this.$scope.message = 'Change Log';
    this.$scope.editSwitch = false;
    this.$scope.visible = false;
    this.$scope.removedIterOptions = [];
    this.defaultValues = {
      date: '',
      type: '',
      inject: '',
      remove: '',
      fixTime: '',
      fixReference: '0',
      desc: '',
      injectedIter: '',
      removedIter: ''
    };
  }
  $onInit() {
    this.$http.get('/api/projects/'+this.$stateParams.id).then(response => {
      this.$scope.Project = response.data;
      this.$scope.changeLog = this.$scope.Project.changeLog;
      if(this.$scope.changeLog.length > 0){
        this.$scope.currentLogNum = this.$scope.changeLog[this.$scope.changeLog.length-1].changeNum
+ 1;
      }
      else{
        this.$scope.currentLogNum = 1;
      }
      this.$scope.injectedIterOptions = new Array(this.$scope.Project.iterations.length);
      for(var i=0; i<this.$scope.injectedIterOptions.length; i++){
        this.$scope.injectedIterOptions[i] = i+1;
      }
      if(this.$scope.Project.hasOwnProperty('Setup')){
```

```
      this.$scope.visible = true;
    }
    this.$scope.change = angular.copy(this.defaultValues);
  });
}
// Add a new change log
addChangeLog(){
  var addLog = {
    changeNum: this.$scope.currentLogNum,
    startDate: this.$scope.change.date,
    typeOfChange: this.$scope.change.type,
    injectIn: this.$scope.change.inject,
    removedIn: this.$scope.change.remove,
    fixTime: this.$scope.change.fixTime,
    fixReference: this.$scope.change.fixReference,
    desc: this.$scope.change.desc,
    injectedIter: this.$scope.change.injectedIter,
    removedIter: this.$scope.change.removedIter
  };
  this.$scope.Project.changeLog.push(addLog);
  var data = {
    changeLog: this.$scope.Project.changeLog
  };
  this.$http.put('/api/projects/'+this.$stateParams.id, data).then(response => {
    console.log(response);
  });
  this.reset();
  $('#myModal').modal('toggle');
  this.$scope.currentLogNum++;
}
reset(){
  this.$scope.changeForm.$setPristine();
  this.$scope.changeForm.$setUntouched();
  this.$scope.change = angular.copy(this.defaultValues);
  this.$scope.removedIterOptions = [];
}

deleteChangeLog(log) {
  var indexVal = this.$scope.Project.changeLog.indexOf(log);
  console.log(indexVal);
  this.$scope.Project.changeLog.splice(indexVal, 1);
  var data = {
    changeLog: this.$scope.Project.changeLog
  };
  this.$http.put('/api/projects/'+this.$stateParams.id, data).then(response => {
    console.log(response);
  });
}
```

```
  editChangeLog(log) {
    this.$scope.editSwitch = !this.$scope.editSwitch;
    if(this.$scope.editSwitch){
      console.log(log);
      this.$scope.indexVal = this.$scope.Project.changeLog.indexOf(log);
      console.log(this.$scope.indexVal);
      var setValues = {
        changeNum: log.changeNum,
        startDate: log.startDate,
        typeOfChange: log.typeOfChange,
        injectIn: log.injectIn,
        removedIn: log.removedIn,
        fixTime: log.fixTime,
        fixReference: log.fixReference,
        desc: log.desc,
        injectedIter: log.injectedIter,
        removedIter: log.removedIter
      };
      this.$scope.changeVal = angular.copy(setValues);
    }
    else{
      console.log('save');
      this.$scope.Project.changeLog.splice(this.$scope.indexVal, 1, this.$scope.changeVal);
      var data = {
        changeLog: this.$scope.Project.changeLog
      };
      this.$http.put('/api/projects/'+this.$stateParams.id, data).then(response => {
        console.log(response);
      });
    }
  }
  setRemovedOptions() {
    this.$scope.removedIterOptions = [];
    for(var i=this.$scope.change.injectedIter; i <= this.$scope.injectedIterOptions.length; i++){
        this.$scope.removedIterOptions.push(i);
    }
  }

}

angular.module('pcseappApp')
  .component('changeLog', {
    templateUrl: 'app/changeLog/changeLog.html',
    controller: ChangeLogComponent
  });

})();
```

changeLog.css

```
.list {
	list-style: none outside none;
	margin: 10px 0 30px;
}

.apps-container {
  margin: 10px 10px 0 0;
  padding: 5px;
  min-width:200px;
  min-height:50px;
}

.app {
	padding: 5px 10px;
	margin: 5px 0;
	border: 2px solid #444;
	background-color: grey;
  height: 50px;
	font-size: 1.1em;
	font-weight: bold;
	text-align: center;
	cursor: move;
}
#arrow-center {
  text-align: center;
  margin-top: 100px;
}
```

changeLog.html

```
<navbar></navbar>
<div id="page-content-wrapper" >
	<!-- Modal -->
	<div ng-show="visible">
	<div class="modal fade" id="myModal" role="dialog">
		<div class="modal-dialog">
			<form name="changeForm" ng-submit="$ctrl.addChangeLog()" novalidate>
				<!-- Modal content-->
				<div class="modal-content">
					<div class="modal-header">
						<button type="button" class="close" data-
dismiss="modal">&times;</button>
						<h4 class="modal-title"> Add New Change Log</h4>
					</div>

					<div class="modal-body">
						<div class="row">
							<div class="col-md-6">
```

```
                                        <label>Date</label>
                                        <input class="form-control" ng-
model="change.date" type="date" ng-required="true" placeholder="Enter a date"/>
                                    </div>
                                    <div class="col-md-6">
                                        <label>Type of Change</label>
                                        <select class="form-control" ng-
model="change.type" ng-required="true">
                                            <option value="Requirement
Change">Requirement Change</option>
                                            <option value="Requirement
Clarification">Requirement Clarification</option>
                                            <option value="Product
Syntax">Product Syntax</option>
                                            <option value="Product
Logic">Product Logic</option>
                                            <option value="Product
Interface">Product Interface</option>
                                            <option value="Product
Checking">Product Checking</option>
                                            <option value="Test
Syntax">Test Syntax</option>
                                            <option value="Test
Logic">Test Logic</option>
                                            <option value="Test
Interface">Test Interface</option>
                                            <option value="Test
Checking">Test Checking</option>
                                            <option value="Bad
Smell">Bad Smell</option>
                                        </select>
                                    </div>
                                </div>

                                <div class="row">
                                    <div class="col-md-6">
                                        <label>Inject in</label>
                                        <select class="form-control" ng-
model="change.inject" ng-required="true">
                                            <option
value="Analysis">Analysis</option>
                                            <option
value="Architecture">Architecture</option>
                                            <option
value="Planning">Planning</option>
                                            <option value="Integration
Planning">Integration Planning</option>
                                            <option
value="Construction">Construction</option>
```

```
                                        <option
value="Refactoring">Refactoring</option>
                                        <option
value="Review">Review</option>
                                        <option value="Integration
Testing">Integration Testing</option>
                                        <option
value="Repatterning">Repatterning</option>
                                        <option
value="Postmortem">Postmortem</option>
                                        <option
value="Sandbox">Sandbox</option>
                                    </select>
                                </div>
                                <div class="col-md-6">
                                    <label>Removed in</label>
                                    <select class="form-control" ng-
model="change.remove" ng-required="true">
                                        <option
value="Analysis">Analysis</option>
                                        <option
value="Architecture">Architecture</option>
                                        <option
value="Planning">Planning</option>
                                        <option value="Integration
Planning">Integration Planning</option>
                                        <option
value="Construction">Construction</option>
                                        <option
value="Refactoring">Refactoring</option>
                                        <option
value="Review">Review</option>
                                        <option value="Integration
Testing">Integration Testing</option>
                                        <option
value="Repatterning">Repatterning</option>
                                        <option
value="Postmortem">Postmortem</option>
                                        <option
value="Sandbox">Sandbox</option>
                                    </select>
                                </div>
                            </div>

                            <div class="row">
                                <div class="col-md-6">
                                    <label>Inject iteration</label>
                                    <select class="form-control" ng-
model="change.injectedIter" ng-required="true" ng-change="$ctrl.setRemovedOptions()">
```

```
                                        <option ng-repeat="options in injectedIterOptions" value="{{options}}">{{options}}</option>
                                    </select>
                                </div>
                                <div class="col-md-6">
                                    <label>Removed iteration</label>
                                    <select class="form-control" ng-model="change.removedIter" ng-required="true">
                                        <option ng-repeat="options in removedIterOptions" value="{{options}}">{{options}}</option>
                                    </select>
                                </div>
                            </div>

                            <div class="row">
                                <div class="col-md-6">
                                    <label>Fix time</label>
                                    <input class="form-control" ng-model="change.fixTime" type="text" ng-required="true" placeholder="Enter a fix time"/>
                                </div>
                                <div class="col-md-6">
                                    <label>Fix Reference</label>
                                    <select class="form-control" ng-model="change.fixReference" ng-required="true">
                                        <option ng-repeat="options in changeLog" value="{{options.changeNum}}">{{options.changeNum}} - {{options.typeOfChange}} - {{options.desc}}</option>
                                    </select>
                                </div>
                            </div>
                            <div class="form-group">
                                <label>Description</label>
                                <textarea class="form-control" ng-model="change.desc" ng-required="true" placeholder="Enter a change description"></textarea>
                            </div>
                        </div>
                        <div class="modal-footer">
                            <button class="btn btn-default" type="button" ng-click="$ctrl.reset()" data-dismiss="modal">Cancel</button>
                            <input class="btn btn-primary btn-create" ng-disabled="changeForm.$invalid" type="submit" value="Create"/>
                        </div>

                    </div>
                </form>
            </div>
        </div>
        <!-- End Modal -->
```

```
<div class="container-fluid">
    <div id="addProject">
        <div class="row">
            <div class="col-sm-1">
                <button id="addButton" type="button" class="btn btn-danger"
data-toggle="modal" data-target="#myModal"  data-toggle="tooltip" data-placement="top" title="Add a
New ChangeLog">+</button>
            </div>
            <div class="col-sm-11">
                <div id="filter" class="form-group">
                    <label id="filter-label" class="form-
group">Filter:</label>
                    <input id="filter-input" class="form-group input-sm"
ng-model="search" type="text" placeholder="Enter a Change Log name">
                </div>
            </div>
        </div>
            <md-content class="md-padding" layout-lg="column" layout="row">
                <md-card ng-show="editSwitch">
                    <md-card-title>
                        <md-card-title-text>
                            <span class="md-headline">Edit
Change Log - {{changeVal.changeNum}}</span>
                        </md-card-title-text>
                    </md-card-title>
                    <md-card-content layout="row" layout-align="space-
between">
                        <md-list-item class="md-5-line" ng-
click="null">
                            <div class="row">
                                <div class="col-sm-2">
                                    <h3>Type of
Change:</h3> <input class="form-control" type="text" ng-model=changeVal.typeOfChange></input>
                                </div>
                                <div class="col-sm-2">
                                    <h3>Injected In:</h3>
<input class="form-control" type="text" ng-model=changeVal.injectIn></input>
                                </div>
                                <div class="col-sm-2">
                                    <h3>Removed In:</h3>
<input class="form-control" type="text" ng-model=changeVal.removedIn></input>
                                </div>
                                <div class="col-sm-2">
                                    <h3>Time to fix:</h3>
<input class="form-control" type="text" ng-model=changeVal.fixTime></input>
                                </div>
                                <div class="col-sm-2">
                                    <h3>Injected
Iteration:</h3> <input class="form-control" type="text" ng-model=changeVal.injectedIter></input>
```

```
                                                                    </div>
                                                                    <div class="col-sm-2">
                                                                        <h3>Removed
Iteration:</h3> <input class="form-control" type="text" ng-model=changeVal.removedIter></input>
                                                                    </div>
                                                                    <div class="col-sm-2">
                                                                        <h3>Fix
Reference:</h3> <input class="form-control" type="text" ng-model=changeVal.fixReference></input>
                                                                    </div>
                                                                    <div class="col-sm-10">
                                                                        <h3>Description:</h3>
<textarea class="form-control" ng-model=changeVal.desc>{{changeVal.desc}}</textarea>
                                                                    </div>
                                                                </div>
                                                            </md-list-item>
                                                        </md-card-content>
                                                    </md-card>
                                                    <div ng-repeat="changeLogs in changeLog | filter: search">
                                                        <div flex-lg flex-gt-lg="50" layout="column">
                                                            <md-card>
                                                                <md-card-title>
                                                                    <md-card-title-text>
                                                                        <span class="md-
headline">Reference # {{changeLogs.changeNum}}</span>
                                                                        <span class="md-
subhead">Date of creation: {{changeLogs.startDate | date:'yyyy/MM/dd'}}</span>
                                                                    </md-card-title-text>
                                                                </md-card-title>
                                                                <md-card-content layout="row" layout-
align="space-between">
                                                                    <md-list-item class="md-5-line"
ng-click="null">
                                                                        <div class="row">
                                                                            <div
class="col-sm-2">

                                                                                <h3>Type of Change:</h3> <input class="form-control" ng-readonly="true" type="text"
value={{changeLogs.typeOfChange}}></input>
                                                                            </div>
                                                                            <div
class="col-sm-2">

                                                                                <h3>Injected In:</h3> <input class="form-control" ng-readonly="true" type="text"
value={{changeLogs.injectIn}}></input>
                                                                            </div>
                                                                            <div
class="col-sm-2">

                                                                                <h3>Removed In:</h3> <input class="form-control" ng-readonly="true" type="text"
```

```
value={{changeLogs.removedIn}}></input>
                                                                    </div>
                                                                    <div
class="col-sm-2">

        <h3>Time to fix:</h3> <input class="form-control" type="text" ng-readonly="true"
value={{changeLogs.fixTime}}></input>
                                                                    </div>
                                                                    <div
class="col-sm-2">

        <h3>Injected Iteration:</h3> <input class="form-control" type="text" ng-readonly="true"
value={{changeLogs.injectedIter}}></input>
                                                                    </div>
                                                                    <div
class="col-sm-2">

        <h3>Removed Iteration:</h3> <input class="form-control" type="text" ng-readonly="true"
value={{changeLogs.removedIter}}></input>
                                                                    </div>
                                                                    <div
class="col-sm-2">

        <h3>Fix Reference:</h3> <input class="form-control" type="text" ng-readonly="true"
value={{changeLogs.fixReference}}></input>
                                                                    </div>
                                                                    <div
class="col-sm-10">

        <h3>Description:</h3> <textarea class="form-control" ng-
readonly="true">{{changeLogs.desc}}</textarea>
                                                                    </div>
                                                                </div>
                                                            </md-list-item>

                                                            <md-card-actions
layout="column">
                                                                <md-button class="md-
icon-button" aria-label="Remove" ng-click="$ctrl.deleteChangeLog(changeLogs)">
                                                                    <md-icon md-
svg-icon="../assets/images/Trash.svg"></md-icon>
                                                                </md-button>
                                                                <md-button class="md-
icon-button" aria-label="Edit" ng-click="$ctrl.editChangeLog(changeLogs)">
                                                                    <md-icon md-
svg-icon="../assets/images/Edit_icon.svg"></md-icon>
                                                                </md-button>
                                                            </md-card-actions>
                                                        </md-card-content>
```

```
                                                    </md-card>
                                                </div>
                                            </div>
                                        </md-content>

                    </div>
                    </div>
            </div>
```

estimate.js

```
'use strict';

angular.module('pcseappApp')
  .config(function ($stateProvider) {
    $stateProvider
      .state('estimate', {
        url: '/project/estimate/:id',
        template: '<estimate></estimate>'
      });
  });
```

estimate.controller.js

```
'use strict';
(function(){

  class EstimateComponent {
    constructor($http, $scope, $state, $stateParams) {
      this.$http = $http;
      this.$state = $state;
      this.$scope = $scope;
      this.$stateParams = $stateParams;
      this.estimateActive = 'active';
      $scope.visible = true;
      this.$scope.sizeHide = false;
      this.$scope.historicalComponent = {};
      this.$scope.edit = true;

      $scope.iterations = [];
      $scope.totalItems = $scope.iterations.length*10;
      $scope.currentPage = 1;
      $scope.trackPages = [0, 1];
      $scope.maxSize = 8;
      $scope.itemsPerPage = 1;
      $scope.pageChanged = function() {
        $scope.trackPages = [$scope.trackPages[1], $scope.currentPage];
        console.log($scope.trackPages);
        if($scope.currentPage-1 > 0 && $scope.trackPages[0] < $scope.trackPages[1]) {
```

```
        if(this.Project.iterations[$scope.currentPage-1].estimate.hasOwnProperty('pLOC') &&
this.Project.iterations[$scope.currentPage-1].estimate.pLOC > 0) {
          console.log('No Need to copy');
        }
        else {
          console.log('copy');
          this.Project.iterations[$scope.currentPage-1].estimate = {
            pLOC: this.Project.iterations[$scope.currentPage-2].estimate.pLOC,
            pEffort: this.Project.iterations[$scope.currentPage-2].estimate.pEffort,
            aLOC: this.Project.iterations[$scope.currentPage-2].estimate.aLOC,
            aEffort: this.Project.iterations[$scope.currentPage-2].estimate.aEffort
          }
          this.Project.iterations[$scope.currentPage-1].estimate.newComponents = function() {
            var theCopy = [];
            var arr1 = $scope.Project.iterations[$scope.currentPage-2].estimate.newComponents;
            console.log(arr1);
            for (var i = 0, len = arr1.length; i < len; i++) {
              theCopy[i] = Object.assign({}, arr1[i]);
            }
            console.log(theCopy);
            return theCopy;
          }();
        }
      }
    };
    this.$scope.sortableOptions = {
      placeholder: "app",
      connectWith: ".apps-container"
    };
    this.defaultNewValues = {
      name: '',
      methods: '',
      size: '',
      rLOC: '',
      reusable: false
    };
    this.$scope.sumLOCRaw = 0;
    // Try to use mid 0-4
    this.$scope.sizeMatrix = {
      low: {
        VS:1,
        S: 6,
        M: 8,
        L: 11,
        VL: 15
      },
      mid: {
        VS: 5,
        S: 7,
```

```
      M: 10,
      L: 13,
      VL: 18
     },
     high: {
      VS: 6,
      S: 8,
      M: 11,
      L: 15,
      VL: 22
     }
    };
   }

   $onInit() {
    this.$http.get('/api/projects/'+this.$stateParams.id).then(response => {
     this.$scope.Project = response.data;
     this.$scope.iterations = this.$scope.Project.iterations;
     this.$scope.totalItems = this.$scope.iterations.length*10;
    });
    this.$http.get('/api/projects/').then(response => {
     this.$scope.allProjects = response.data;
     var ___id = this.$stateParams.id;
     var naturalLog = 0;
     var count = 0;
     var averageLog = 0;
     var sumOfSquares = 0;
     var sumLOCaLOCr = 0;
     var sumRawLOC = 0
     var std = 0;
     this.$scope.allProjects.forEach(function(project) {
      if(project._id !== ___id){
       if(project.iterations.length > 0) {
        var iteration = project.iterations[project.iterations.length-1];
        if(iteration.hasOwnProperty('estimate')){
         if(iteration.estimate.hasOwnProperty('newComponents')) {
          iteration.estimate.newComponents.forEach(function(component) {
           if(component.hasOwnProperty('actual_LOC') && component.actual_LOC > 0 &&
component.hasOwnProperty('actual_methods') && component.actual_methods > 0){
            var currentVal = component.actual_LOC/component.actual_methods;
            var currentLog = Math.log(currentVal);
            naturalLog += currentLog;
            count++;

            sumRawLOC += component.rLOC;
           }
          });
         }
        }
```

```
    }
  }
});
averageLog = naturalLog/count;
console.log(averageLog);
this.$scope.allProjects.forEach(function(project) {
  if(project._id !== ___id){
    if(project.iterations.length > 0) {
      var iteration = project.iterations[project.iterations.length-1];
      if(iteration.hasOwnProperty('estimate')){
        if(iteration.estimate.hasOwnProperty('newComponents')) {
          iteration.estimate.newComponents.forEach(function(component) {
            if(component.actual_LOC > 0){
              var currentVal = component.actual_LOC/component.actual_methods;
              var currentLog = Math.log(currentVal);
              sumOfSquares += Math.pow(currentLog - averageLog, 2);
            }
          });
        }
      }
    }
  }
});
std = Math.sqrt(sumOfSquares/count);
console.log(std);
if(isNaN(averageLog) || isNaN(std)){
  console.log('Aye');
}
else{
  this.$scope.sizeMatrix = {
    low: {
      VS:1,
      S: Math.round(Math.exp(averageLog - (3*std/2))),
      M: Math.round(Math.exp(averageLog - (std/2))),
      L: Math.round(Math.exp(averageLog + (std/2))),
      VL: Math.round(Math.exp(averageLog + (3*std/2)))
    },
    mid: {
      VS: Math.round(Math.exp(averageLog - (2*std))),
      S: Math.round(Math.exp(averageLog - std)),
      M: Math.round(Math.exp(averageLog)),
      L: Math.round(Math.exp(averageLog + std)),
      VL: Math.round(Math.exp(averageLog + (2*std)))
    },
    high: {
      VS: Math.round(Math.exp(averageLog - (3*std/2))),
      S: Math.round(Math.exp(averageLog - (std/2))),
      M: Math.round(Math.exp(averageLog + (std/2))),
      L: Math.round(Math.exp(averageLog + (3*std/2))),
```

```
          VL: 22
        }
      };
    }
    var sumALOC = 0;
    var sumEa = 0;
    var sumEaLOCa = 0;
    var sumAEffort = 0;
    var productivity = 0;
    var histComponents = [];
    this.$scope.allProjects.forEach(function(project) {
      if(project._id !== ___id){
        if(project.iterations.length > 0) {
          var iteration = project.iterations[project.iterations.length-1];
          if(iteration.hasOwnProperty('estimate')){
            if(iteration.estimate.aLOC > 0 && iteration.estimate.aEffort > 0){
              sumALOC += iteration.estimate.aLOC;
              sumEa += iteration.estimate.aEffort;
              sumAEffort += iteration.estimate.aEffort;
              iteration.estimate.newComponents.forEach(function(component) {
                histComponents.push({component: component.name, size: component.size, project:
project.name});
              });
            }
          }
        }
      }
    });
    productivity = sumALOC/sumAEffort;
    sumLOCaLOCr = sumALOC/sumRawLOC;
    sumEaLOCa = sumEa/sumALOC;
    console.log(productivity);
    console.log(sumLOCaLOCr);
    console.log(sumEaLOCa);

    this.$scope.sizeCalc = {
      productivity: productivity,
      sumLOCaLOCr: sumLOCaLOCr,
      sumEaLOCa: sumEaLOCa
    }
    //All components from other projects
    this.$scope.historicalComponents = histComponents;
    var sumRaw = 0;
    this.$scope.Project.iterations[this.$scope.currentPage-
1].estimate.newComponents.forEach(function(comp) {
      sumRaw+=comp.rLOC;
    });
    this.$scope.sumLOCRaw = sumRaw;
    if(this.$scope.Project.iterations[this.$scope.currentPage-1].estimate.hasOwnProperty('aLOC') &&
```

```
(this.$scope.Project.iterations[this.$scope.currentPage-1].estimate.aLOC == 0 || this.
$scope.Project.iterations[this.$scope.currentPage-1].estimate.aLOC == null)) {
        var sumActual_LOC = 0;
        this.$scope.Project.iterations[this.$scope.currentPage-
1].estimate.newComponents.forEach(function(comp) {
          sumActual_LOC+=(comp.actual_LOC+(comp.hasOwnProperty('actual_mLOC')?
comp.actual_mLOC:0)+(comp.hasOwnProperty('actual_aLOC')?comp.actual_aLOC:0));
        });
        this.$scope.Project.iterations[this.$scope.currentPage-1].estimate.aLOC = sumActual_LOC;
      }
    });
  }
  getTotalTime(iter) {
    var timeTaken = 0;
    var bool = true;
    this.$scope.Project.iterations.forEach(function(iteration) {
      if(bool) {
        iteration.timeLog.forEach(function(log) {
          timeTaken += parseInt(log.timeTaken);
          if(log.interrupt) {
            timeTaken -= parseInt(log.interrupt);
          }
        });
        if(iteration._id === iter._id){
          bool = false;
          iteration.estimate.aEffort = timeTaken;
        }
      }
    });
  }

  addNewComponent() {
    if(this.$scope.Project.iterations[this.$scope.currentPage-1].hasOwnProperty('estimate')){
      console.log('Yeah!!');
    }
    else{
      this.$scope.Project.iterations[this.$scope.currentPage-1].estimate = {
        newComponents: []
      };
      console.log(this.$scope.Project);
    }
    this.$scope.Project.iterations[this.$scope.currentPage-1].estimate.newComponents.push(this.
$scope.newComponent);
    this.computeRawLOC(this.$scope.newComponent);
    console.log(this.$scope.Project.iterations[this.$scope.currentPage-1].estimate);
    var len = this.$scope.Project.iterations[this.$scope.currentPage-1].estimate.newComponents.length;
    console.log(this.$scope.Project.iterations[this.$scope.currentPage-1].estimate.newComponents[len-
1].rLOC);
```

```
    //Change EP && LOCp
    if(isNaN(this.$scope.Project.iterations[this.$scope.currentPage-1].estimate.newComponents[len-
1].rLOC) && isNaN(this.$scope.Project.iterations[this.$scope.currentPage-
1].estimate.newComponents[len-1].mLOC) && isNaN(this.$scope.Project.iterations[this.
$scope.currentPage-1].estimate.newComponents[len-1].aLOC)) {
      console.log('Nope');
    }
    else if(isNaN(this.$scope.Project.iterations[this.$scope.currentPage-1].estimate.newComponents[len-
1].mLOC) && isNaN(this.$scope.Project.iterations[this.$scope.currentPage-
1].estimate.newComponents[len-1].aLOC)){
      this.$scope.sumLOCRaw += this.$scope.Project.iterations[this.$scope.currentPage-
1].estimate.newComponents[len-1].rLOC;
    }
    else if(isNaN(this.$scope.Project.iterations[this.$scope.currentPage-1].estimate.newComponents[len-
1].mLOC)) {
      this.$scope.sumLOCRaw += this.$scope.Project.iterations[this.$scope.currentPage-
1].estimate.newComponents[len-1].rLOC + this.$scope.Project.iterations[this.$scope.currentPage-
1].estimate.newComponents[len-1].aLOC;
    }
    else if(isNaN(this.$scope.Project.iterations[this.$scope.currentPage-1].estimate.newComponents[len-
1].aLOC)){
      this.$scope.sumLOCRaw += this.$scope.Project.iterations[this.$scope.currentPage-
1].estimate.newComponents[len-1].rLOC + this.$scope.Project.iterations[this.$scope.currentPage-
1].estimate.newComponents[len-1].mLOC;
    }
    else {
      this.$scope.sumLOCRaw += this.$scope.Project.iterations[this.$scope.currentPage-
1].estimate.newComponents[len-1].rLOC + this.$scope.Project.iterations[this.$scope.currentPage-
1].estimate.newComponents[len-1].mLOC+ this.$scope.Project.iterations[this.$scope.currentPage-
1].estimate.newComponents[len-1].aLOC;
    }
    this.$scope.Project.iterations[this.$scope.currentPage-1].estimate.pLOC = Math.ceil(this.
$scope.sumLOCRaw*this.$scope.sizeCalc.sumLOCaLOCr);
    this.$scope.Project.iterations[this.$scope.currentPage-1].estimate.pEffort = Math.ceil(this.
$scope.Project.iterations[this.$scope.currentPage-1].estimate.pLOC * this.$scope.sizeCalc.sumEaLOCa);
    console.log(this.$scope.sumLOCRaw);
    this.resetNew();
    $('#myModal2').modal('toggle');
    this.$scope.sizeHide = false;
  }

  resetNew(){
    this.$scope.newForm.$setPristine();
    this.$scope.newForm.$setUntouched();
    this.$scope.newComponent = angular.copy(this.defaultNewValues);
    this.$scope.sizeHide = false;
  }
  saveProgress() {
    console.log(this.$scope.Project.iterations);
```

```
    var data = {
      iterations: this.$scope.Project.iterations
    };
    this.$http.put('/api/projects/'+this.$stateParams.id, data).then(response => {
      console.log(response);
    });
  }
  deleteComponent(index) {
    this.$scope.sumLOCRaw -= (this.$scope.Project.iterations[this.$scope.currentPage-
1].estimate.newComponents[index].rLOC+this.$scope.Project.iterations[this.$scope.currentPage-
1].estimate.newComponents[index].aLOC+this.$scope.Project.iterations[this.$scope.currentPage-
1].estimate.newComponents[index].mLOC);
    //Change EP && LOCp
    this.$scope.Project.iterations[this.$scope.currentPage-1].estimate.pLOC = Math.ceil(this.
$scope.sumLOCRaw * this.$scope.sizeCalc.sumLOCaLOCr);
    this.$scope.Project.iterations[this.$scope.currentPage-1].estimate.pEffort = Math.ceil(this.
$scope.Project.iterations[this.$scope.currentPage-1].estimate.pLOC * this.$scope.sizeCalc.sumEaLOCa);

    this.$scope.Project.iterations[this.$scope.currentPage-1].estimate.newComponents.splice(index, 1);
    console.log(this.$scope.Project.iterations[this.$scope.currentPage-1].estimate);
  }

  computeRawLOC(component) {
    component.rLOC = this.$scope.sizeMatrix.mid[component.size] * component.methods;
  }
  hideRelSize(Comp) {
    var base = JSON.parse(Comp);
    this.$scope.sizeHide = true;
    this.$scope.newComponent.baseComponent = base.component;
    this.$scope.newComponent.size = base.size;
  }
  submitActual(index) {
    console.log('here');
      if(isNaN(this.$scope.Project.iterations[this.$scope.currentPage-
1].estimate.newComponents[index].actual_mLOC) && isNaN(this.$scope.Project.iterations[this.
$scope.currentPage-1].estimate.newComponents[index].actual_aLOC)){
        if(isNaN(this.$scope.Project.iterations[this.$scope.currentPage-1].estimate.aLOC)){
          this.$scope.Project.iterations[this.$scope.currentPage-1].estimate.aLOC = this.
$scope.Project.iterations[this.$scope.currentPage-1].estimate.newComponents[index].actual_LOC;
        }
        else {
          this.$scope.Project.iterations[this.$scope.currentPage-1].estimate.aLOC += this.
$scope.Project.iterations[this.$scope.currentPage-1].estimate.newComponents[index].actual_LOC;
        }
      }
      else if(isNaN(this.$scope.Project.iterations[this.$scope.currentPage-
1].estimate.newComponents[index].actual_mLOC)) {
        this.$scope.Project.iterations[this.$scope.currentPage-1].estimate.aLOC += this.
$scope.Project.iterations[this.$scope.currentPage-1].estimate.newComponents[index].actual_LOC + this.
```

```
$scope.Project.iterations[this.$scope.currentPage-1].estimate.newComponents[index].actual_aLOC;
      }
      else if(isNaN(this.$scope.Project.iterations[this.$scope.currentPage-
1].estimate.newComponents[index].actual_aLOC)){
        this.$scope.Project.iterations[this.$scope.currentPage-1].estimate.aLOC += this.
$scope.Project.iterations[this.$scope.currentPage-1].estimate.newComponents[index].actual_LOC + this.
$scope.Project.iterations[this.$scope.currentPage-1].estimate.newComponents[index].actual_mLOC;
      }
      else {
        this.$scope.Project.iterations[this.$scope.currentPage-1].estimate.aLOC += this.
$scope.Project.iterations[this.$scope.currentPage-1].estimate.newComponents[index].actual_LOC + this.
$scope.Project.iterations[this.$scope.currentPage-1].estimate.newComponents[index].actual_mLOC+
this.$scope.Project.iterations[this.$scope.currentPage-1].estimate.newComponents[index].actual_aLOC;
      }
      console.log(this.$scope.Project.iterations[this.$scope.currentPage-1].estimate.aLOC+' '+this.
$scope.currentPage);
   }
   changeEdit(index){
     this.$scope.edit = !this.$scope.edit;
     if(isNaN(this.$scope.Project.iterations[this.$scope.currentPage-
1].estimate.newComponents[index].actual_LOC) && isNaN(this.$scope.Project.iterations[this.
$scope.currentPage-1].estimate.newComponents[index].actual_mLOC) && isNaN(this.
$scope.Project.iterations[this.$scope.currentPage-1].estimate.newComponents[index].actual_aLOC)) {
       console.log('Nope');
     }
     else if(isNaN(this.$scope.Project.iterations[this.$scope.currentPage-
1].estimate.newComponents[index].actual_mLOC) && isNaN(this.$scope.Project.iterations[this.
$scope.currentPage-1].estimate.newComponents[index].actual_aLOC)){
       this.$scope.Project.iterations[this.$scope.currentPage-1].estimate.aLOC -= this.
$scope.Project.iterations[this.$scope.currentPage-1].estimate.newComponents[index].actual_LOC;
     }
     else if(isNaN(this.$scope.Project.iterations[this.$scope.currentPage-
1].estimate.newComponents[index].actual_mLOC)) {
       this.$scope.Project.iterations[this.$scope.currentPage-1].estimate.aLOC -= this.
$scope.Project.iterations[this.$scope.currentPage-1].estimate.newComponents[index].actual_LOC + this.
$scope.Project.iterations[this.$scope.currentPage-1].estimate.newComponents[index].actual_aLOC;
     }
     else if(isNaN(this.$scope.Project.iterations[this.$scope.currentPage-
1].estimate.newComponents[index].actual_aLOC)){
       this.$scope.Project.iterations[this.$scope.currentPage-1].estimate.aLOC -= this.
$scope.Project.iterations[this.$scope.currentPage-1].estimate.newComponents[index].actual_LOC + this.
$scope.Project.iterations[this.$scope.currentPage-1].estimate.newComponents[index].actual_mLOC;
     }
     else {
       this.$scope.Project.iterations[this.$scope.currentPage-1].estimate.aLOC -= this.
$scope.Project.iterations[this.$scope.currentPage-1].estimate.newComponents[index].actual_LOC + this.
$scope.Project.iterations[this.$scope.currentPage-1].estimate.newComponents[index].actual_mLOC+
this.$scope.Project.iterations[this.$scope.currentPage-1].estimate.newComponents[index].actual_aLOC;
     }
```

```
    }

    checkIteration() {
      console.log(this.$scope.currentPage);
      return this.$scope.currentPage == 1 ? true:false;
    }

  }

  angular.module('pcseappApp')
  .component('estimate', {
    templateUrl: 'app/estimate/estimate.html',
    controller: EstimateComponent
  });

})();
```

estimate.css

```
th {
  text-align: center;
}
table {
  border-style: dashed;
}
.app {
  height: 100px;
}
```

estimate.html

```
<navbar></navbar>
<div id="page-content-wrapper">
  <div class="container" ng-show="visible">
    <label>Choose the Iteration:</label>
    <div><ul uib-pagination total-items="totalItems" ng-model="currentPage" max-size="maxSize" ng-
change="pageChanged()"></ul></div>
  </div>
  <div ng-repeat="iteration in iterations.slice(((currentPage-1)*itemsPerPage),
((currentPage)*itemsPerPage))">
    <div class="container-fluid">

      <div class="row" ng-show="$ctrl.checkIteration()">
        <div class="col-md-4 col-lg-4">
          <div class="panel panel-primary">
            <div class="panel-heading">
              Size Matrix (LOC/method)
            </div>
            <div class="panel-body">
```

```
<table class="table table-bordered table-sm">
  <thead>
    <tr>
      <th>#</th>
      <th>Low</th>
      <th>Mid</th>
      <th>High</th>
    </tr>
  </thead>
  <tbody>
    <tr>
      <th scope="row">VS</th>
      <td>
        <div class="form-group">
          <input class="form-control" ng-model="sizeMatrix.low.VS" type="text"/>
        </div>
      </td>
      <td>
        <div class="form-group">
          <input class="form-control" ng-model="sizeMatrix.mid.VS" type="text"/>
        </div>
      </td>
      <td>
        <div class="form-group">
          <input class="form-control" ng-model="sizeMatrix.high.VS" type="text"/>
        </div>
      </td>
    </tr>
    <tr>
      <th scope="row">S</th>
      <td>
        <div class="form-group">
          <input class="form-control" ng-model="sizeMatrix.low.S" type="text"/>
        </div>
      </td>
      <td>
        <div class="form-group">
          <input class="form-control" ng-model="sizeMatrix.mid.S" type="text"/>
        </div>
      </td>
      <td>
        <div class="form-group">
          <input class="form-control" ng-model="sizeMatrix.high.S" type="text"/>
        </div>
      </td>
    </tr>
    <tr>
      <th scope="row">M</th>
      <td>
```

```
    <div class="form-group">
      <input class="form-control" ng-model="sizeMatrix.low.M" type="text"/>
    </div>
  </td>
  <td>
    <div class="form-group">
      <input class="form-control" ng-model="sizeMatrix.mid.M" type="text"/>
    </div>
  </td>
  <td>
    <div class="form-group">
      <input class="form-control" ng-model="sizeMatrix.high.M" type="text"/>
    </div>
  </td>
</tr>
<tr>
  <th scope="row">L</th>
  <td>
    <div class="form-group">
      <input class="form-control" ng-model="sizeMatrix.low.L" type="text"/>
    </div>
  </td>
  <td>
    <div class="form-group">
      <input class="form-control" ng-model="sizeMatrix.mid.L" type="text"/>
    </div>
  </td>
  <td>
    <div class="form-group">
      <input class="form-control" ng-model="sizeMatrix.high.L" type="text"/>
    </div>
  </td>
</tr>
<tr>
  <th scope="row">VL</th>
  <td>
    <div class="form-group">
      <input class="form-control" ng-model="sizeMatrix.low.VL" type="text"/>
    </div>
  </td>
  <td>
    <div class="form-group">
      <input class="form-control" ng-model="sizeMatrix.mid.VL" type="text"/>
    </div>
  </td>
  <td>
    <div class="form-group">
      <input class="form-control" ng-model="sizeMatrix.high.VL" type="text"/>
    </div>
```

```
                </td>
               </tr>
              </tbody>
             </table>
            </div>
           </div>
          </div>
        </div>

        <div class="row">
          <div class="col-md-4 col-lg-4">
            <h1>
              <u>Component BackLog</u>
            </h1>
          </div>
          <div class="col-md-4 col-lg-4">
            <h1>
              <button class="btn btn-primary btn-create" type="button" ng-click="$ctrl.saveProgress()" data-
dismiss="modal">Save Progress</button>
            </h1>
          </div>
        </div>
        <div class="row">
          <div class="col-md-12 col-lg-12">
            <button id="addButton" type="button" class="btn btn-danger" data-toggle="modal" data-
target="#myModal2"  data-toggle="tooltip" data-placement="top" title="Add a New
Component">+</button>
            <div class="panel panel-primary">
              <div class="panel-heading">
                Estimated Components
              </div>
              <div class="panel-body">
                <table class="table table-bordered table-striped table-sm">
                  <thead>
                    <tr>
                      <th colspan="2"></th>
                      <th colspan="4">New Methods</th>
                      <th colspan="4">Existing Component</th>
                    </tr>
                  </thead>
                  <thead>
                    <tr>
                      <th>#</th>
                      <th>Name</th>
                      <th>Methods</th>
                      <th>Size</th>
                      <th>Raw LOC</th>
                      <th>Reusable</th>
                      <th>Base Component</th>
```

```
                <th>Deleted LOC</th>
                <th>Modified LOC</th>
                <th>Added LOC</th>
              </tr>
            </thead>
            <tbody ng-repeat="component in iteration.estimate.newComponents">
              <tr>
                <th scope="row">{{$index+1}}</th>
                <td><input class="form-control" ng-model="component.name" type="text" /></td>
                <td><input class="form-control" ng-model="component.methods" type="number" /></td>
                <td><input class="form-control" ng-model="component.size" type="text" /></td>
                <td><input class="form-control"
value="{{component.rLOC+component.mLOC+component.aLOC}}" type="number" /></td>
                <td><input class="form-control" ng-model="component.reusable"
value="{{(component.reusable) || "false"}}" type="checkbox" /></td>
                <td><input class="form-control" ng-model="component.baseComponent" type="text"
/></td>
                <td><input class="form-control" ng-model="component.dLOC" type="number" ng-readonly
= "true"/></td>
                <td><input class="form-control" ng-model="component.mLOC" type="number" ng-
readonly = "true"/></td>
                <td><input class="form-control" ng-model="component.aLOC" type="number" ng-readonly
= "true"/></td>
                <td><button class="btn btn-primary btn-create" type="button" ng-
click="$ctrl.deleteComponent($index)">X</button></td>
              </tr>
            </tbody>
          </table>
        </div>
      </div>
    </div>
    <div class="row">
      <div class="col-md-12 col-lg-12">
        <div class="panel panel-primary">
          <div class="panel-heading">
            Actual Components
          </div>
          <div class="panel-body">
            <table class="table table-bordered table-striped table-sm">
              <thead>
                <tr>
                  <th colspan="5">New Methods</th>
                  <th colspan="4">Existing Component</th>
                </tr>
              </thead>
              <thead>
                <tr>
                  <th>#</th>
```

```
                <th>Name</th>
                <th>Methods</th>
                <th>Actual LOC</th>
                <th>Reusable</th>
                <th>Base Component</th>
                <th>Deleted LOC</th>
                <th>Modified LOC</th>
                <th>Added LOC</th>
              </tr>
            </thead>
            <tbody ng-repeat="component in iteration.estimate.newComponents">
              <tr>
                <th scope="row">{{$index+1}}</th>
                <td><input class="form-control" ng-model="component.name" type="text"  ng-
readonly="edit"/></td>
                <td><input class="form-control" ng-model="component.actual_methods" type="number"
ng-readonly="edit"/></td>
                <td><input class="form-control" ng-model="component.actual_LOC" type="number" ng-
readonly="edit" /></td>
                <td>
                  <select class="form-control" ng-model="component.actual_reusable" ng-required="true"
ng-readonly="edit">
                    <option value="true" ng-selected="component.actual_reusable == true">True</option>
                    <option value="false" ng-selected="component.actual_reusable == false">False</option>
                  </select>
                </td>
                <td><input class="form-control" ng-model="component.actual_baseComponent"
type="text"  ng-readonly="edit"/></td>
                <td><input class="form-control" ng-model="component.actual_dLOC" type="number"  ng-
readonly="edit"/></td>
                <td><input class="form-control" ng-model="component.actual_mLOC" type="number"  ng-
readonly="edit"/></td>
                <td><input class="form-control" ng-model="component.actual_aLOC" type="number"  ng-
readonly="edit"/></td>
                <td><button class="btn btn-primary btn-create" type="button" ng-
click="$ctrl.changeEdit($index)">Edit</button></td>
                <td><button class="btn btn-primary btn-create" type="button" ng-
click="$ctrl.submitActual($index)">Add</button></td>
              </tr>
            </tbody>
          </table>
        </div>
      </div>
    </div>
  </div>

  <div class="row">
    <div class="col-md-6 col-lg-6">
      <div class="panel panel-primary">
```

```
        <div class="panel-heading">
          Size Estimate
        </div>
        <div class="panel-body">
          <table class="table">
            <thead>
              <tr>
                <th>#</th>
                <th>Planned</th>
                <th>Actual so far</th>
              </tr>
            </thead>
            <tbody>
              <tr>
                <th scope="row">LOC</th>
                <td>
                  <div class="form-group">
                    <input class="form-control" ng-model="iteration.estimate.pLOC" type="number" ng-
required="true" placeholder="Enter the Planned LOC"/>
                  </div>
                </td>
                <td>
                  <div class="form-group">
                    <input class="form-control" ng-model="iteration.estimate.aLOC" type="number" ng-
required="true" placeholder="Enter the Actual LOC"/>
                  </div>
                </td>
              </tr>
              <tr>
                <th scope="row">Effort</th>
                <td>
                  <div class="form-group">
                    <div class="input-group">
                      <input class="form-control" ng-model="iteration.estimate.pEffort" type="number" ng-
required="true" placeholder="Enter the Planned Effort (in min.)"/>
                      <span class="input-group-addon">min(s)</span>
                    </div>
                  </div>
                </td>
                <td>
                  <div class="form-group">
                    <div class="input-group">
                      <!-- value={{$ctrl.getTotalTime(iteration)}} ng-readonly="true"
-->{{$ctrl.getTotalTime(iteration)}}
                      <input class="form-control" ng-model="iteration.estimate.aEffort" type="text" ng-
required="true" placeholder="Enter the Actual Effort"/>
                      <span class="input-group-addon">min(s)</span>
                    </div>
                  </div>
```

```
            </div>
           </td>
          </tr>
         </tbody>
        </table>
       </div>
      </div>
    </div>


    <div class="col-md-6 col-lg-6" ng-show="$ctrl.checkIteration()">
      <div class="panel panel-primary">
       <div class="panel-heading">
        Size &amp Effort Calculation
       </div>
       <div class="panel-body">
        <div class="row">
         <div class="col-sm-6 col-md-6 col-lg-6">
          <table class="table">
           <thead>
            <tr>
             <th>Size Calculation</th>
             <th></th>
            </tr>
           </thead>
           <tbody>
            <tr>
             <th scope="row">LOCr</th>
             <td>
              <div class="form-group">
               <input class="form-control" type="text" ng-model="sumLOCRaw"/>
              </div>
             </td>
            </tr>
            <tr>
             <th scope="row">sum(LOCa)/sum(LOCr)</th>
             <td>
              <div class="form-group">
               <input class="form-control" type="text" ng-model="sizeCalc.sumLOCaLOCr"/>
              </div>
             </td>
            </tr>
            <tr>
             <th scope="row">LOCp</th>
             <td>
              <div class="form-group">
               <input class="form-control" type="text" ng-model="iteration.estimate.pLOC"/>
              </div>
             </td>
```

```
      </tr>
      <tr>
        <th scope="row">Confidence</th>
        <td>
          <div class="form-group">
            <input class="form-control" type="text"/>
          </div>
        </td>
      </tr>
    </tbody>
  </table>
</div>
<div class="col-sm-6 col-md-6 col-lg-6">
  <table class="table">
    <thead>
      <tr>
        <th></th>
        <th>Effort Calculation</th>
      </tr>
    </thead>
    <tbody>
      <tr>
        <td>
          <div class="form-group">
            <input class="form-control" type="text" ng-model="sizeCalc.productivity"/>
          </div>
        </td>
        <th scope="row">Productivity</th>
      </tr>
      <tr>
        <td>
          <div class="form-group">
            <input class="form-control" type="text" ng-model="sizeCalc.sumEaLOCa"/>
          </div>
        </td>
        <th scope="row">sum(Ea)/sum(LOCa)</th>
      </tr>
      <tr>
        <td>
          <div class="form-group">
            <input class="form-control" type="text" ng-model="iteration.estimate.pEffort"/>
          </div>
        </td>
        <th scope="row">Ep</th>
      </tr>
      <tr>
        <td>
          <div class="form-group">
            <input class="form-control" type="text"/>
```

```
                </div>
              </td>
              <th scope="row">LPI</th>
            </tr>
            <tr>
              <td>
                <div class="form-group">
                  <input class="form-control" type="text"/>
                </div>
              </td>
              <th scope="row">UPI</th>
            </tr>
            <tr>
              <td>
                <div class="form-group">
                  <input class="form-control" type="text"/>
                </div>
              </td>
              <th scope="row">Confidence</th>
            </tr>
          </tbody>
        </table>
      </div>
    </div>
  </div>
</div>

  </div>

 </div>
</div>
</div>
<!-- Modal -->

<div class="modal fade" id="myModal2" role="dialog">
  <div class="modal-dialog">
    <form name="newForm" ng-submit="$ctrl.addNewComponent()" novalidate>
      <!-- Modal content-->
      <div class="modal-content">
        <div class="modal-header">
          <button type="button" class="close" data-dismiss="modal">&amp;times;</button>
          <h4 class="modal-title">Add New Component</h4>
        </div>

        <div class="modal-body">
          <div class="row">
            <div class="col-md-12">
              <div class="form-group">
```

```
            <label>Component name</label>
            <input class="form-control" ng-model="newComponent.name" type="text" ng-required="true"
placeholder="Enter a Component Name"/>
            <div ng-show="newForm.name.$invalid && newForm.name.$touched">
              <small style="color: Red; display: block;">Enter a Valid Component name</small>
            </div>
          </div>
        </div>
      </div>
      <div class="row">
        <div class="col-md-6">
          <div class="form-group">
            <label>Method Count</label>
            <input class="form-control" ng-model="newComponent.methods" type="number" ng-
required="true"/>
          </div>
        </div>
        <div class="col-md-6">
          <div class="form-group">
            <label>Base Component</label>
            <select class="form-control" ng-change="$ctrl.hideRelSize(historicalComponent)" ng-
model="historicalComponent" >
              <option ng-repeat="histComp in historicalComponents"
value="{{histComp}}">{{histComp.component+ ' from '+ histComp.project}}</option>
            </select>
          </div>
        </div>
      </div>
      <div class="row" ng-show="sizeHide">
        <div class="col-md-4">
          <div class="form-group">
            <label>Deleted LOC</label>
            <input class="form-control" ng-model="newComponent.dLOC" type="number"/>
          </div>
        </div>
        <div class="col-md-4">
          <div class="form-group">
            <label>Modified LOC</label>
            <input class="form-control" ng-model="newComponent.mLOC" type="number"/>
          </div>
        </div>
        <div class="col-md-4">
          <div class="form-group">
            <label>Added LOC</label>
            <input class="form-control" ng-model="newComponent.aLOC" type="number"/>
          </div>
        </div>
      </div>
      <div class="row">
```

```
          <div class="col-md-6" ng-hide="sizeHide">
            <div class="form-group">
              <label>Relative Size</label>
              <select class="form-control" ng-model="newComponent.size" ng-required="true">
                <option value="VS">VS</option>
                <option value="S">S</option>
                <option value="M">M</option>
                <option value="L">L</option>
                <option value="VL">VL</option>
              </select>
            </div>
          </div>
          <div class="col-md-6">
            <div class="checkbox">
              <label><input type="checkbox" ng-model="newComponent.reusable">Reusable</label>
            </div>
          </div>
        </div>
      </div>
      <div class="modal-footer">
        <button class="btn btn-default" type="button" ng-click="$ctrl.resetNew()" data-
dismiss="modal">Cancel</button>
        <input class="btn btn-primary btn-create" ng-disabled="newForm.$invalid" type="submit"
value="Create"/>
      </div>

    </div>
  </form>
 </div>
</div>
```

schedule.js

```
'use strict';

angular.module('pcseappApp')
 .config(function ($stateProvider) {
   $stateProvider
     .state('schedule', {
       url: '/project/schedule/:id',
       template: '<schedule></schedule>'
     });
 });
```

schedule.controller.js

```
'use strict';
(function(){
```

```
class ScheduleComponent {
  constructor($http, $scope, $state, $stateParams) {
    this.$http = $http;
    this.$state = $state;
    this.$scope = $scope;
    this.$stateParams = $stateParams;
    this.scheduleActive = 'active';
    this.$scope.totalEp = 0;
    this.$scope.totalTasks = 0;
    this.$scope.remainingTasks = 0;
    this.$scope.actualMinPerTask = 0;
    this.$scope.iterationEndsValue = 0;

    $scope.visible = true;
    $scope.iterations = [];
    $scope.totalItems = $scope.iterations.length*10;
    $scope.currentPage = 1;
    $scope.maxSize = 8;
    $scope.itemsPerPage = 1;
    $scope.trackPages = [0, 1];
    $scope.pageChanged = function() {
      //console.log('page Changed');
      $scope.iterationEndsValue = $scope.currentPage-1;
      $scope.trackPages = [$scope.trackPages[1], $scope.currentPage];
      //console.log($scope.trackPages);
      if($scope.currentPage-1 > 0 && $scope.trackPages[0] < $scope.trackPages[1]) {
        //console.log(this.Project.iterations[$scope.currentPage-1].schedule);
        if($scope.iterations[$scope.currentPage-1].hasOwnProperty('schedule') &&
$scope.iterations[$scope.currentPage-1].schedule.hasOwnProperty('components') &&
$scope.iterations[$scope.currentPage-1].schedule.components.length > 0){
          //console.log('Already copied');
          $scope.iterations[$scope.currentPage-1].schedule.wbs.forEach(function(date) {
            date.plannedCompletionDate = new Date(date.plannedCompletionDate);
            date.ActualCompletionDate = new Date(date.ActualCompletionDate);
          });
        }
        else {
          for(var i=0; i<$scope.iterations[$scope.currentPage-1].estimate.newComponents.length; i++) {
            $scope.iterations[$scope.currentPage-1].schedule.components.push({
              name: $scope.iterations[$scope.currentPage-1].estimate.newComponents[i].name,
              backLog: {
                PlannedTaskCompleted: 0,
                UnplannedTaskCompleted: 0,
                AdditionaltaskDiscovered: 0
              },
              componentIterationMap: []
            });
          }
          //console.log($scope.iterations[$scope.currentPage-1].schedule.components);
```

```
      var len = $scope.iterations.length;
      $scope.iterations[$scope.currentPage-1].schedule.components.forEach(function(component) {
       var arr1 = [];
       for(var j=0; j<len; j++) {
        var obj = {
         production: 0,
         mock: 0
        };
        arr1.push(Object.assign({}, obj));
       }
       component.componentIterationMap = Object.assign([], arr1);
      });
      //console.log($scope.iterations[$scope.currentPage-1].schedule.components);
     }
     $scope.totalEp = $scope.iterations[$scope.currentPage-1].estimate.pEffort;
    }
    else {
     //console.log(this.Project.iterations[$scope.currentPage-1].schedule);
     $scope.totalEp = $scope.iterations[$scope.currentPage-1].estimate.pEffort;
    }

    for(var i=0; i<$scope.trackPages[0]-1; i++) {
     if($scope.currentPage != 1) {
      $scope.iterations[$scope.currentPage-1].schedule.components.forEach(function(component) {
       component.componentIterationMap[i].showIt = false;
      });
     }
    }
    for(var i=0; i<$scope.trackPages[1]-1; i++) {
     if($scope.currentPage != 1) {
      $scope.iterations[$scope.currentPage-1].schedule.components.forEach(function(component) {
       component.componentIterationMap[i].showIt = true;
      });
     }
    }
   }
}
$onDestroy() {
  this.saveProgress();
}
$onInit() {
  this.getData();
}
getData() {
  this.$http.get('/api/projects/'+this.$stateParams.id).then(response => {
    this.$scope.Project = response.data;
    this.$scope.iterations = this.$scope.Project.iterations;
    //console.log(this.$scope.iterations);
    this.$scope.totalItems = this.$scope.iterations.length*10;
```

```
    if(this.$scope.iterations[this.$scope.currentPage-1].hasOwnProperty('estimate') && this.
$scope.iterations[this.$scope.currentPage-1].estimate.hasOwnProperty('newComponents') && this.
$scope.iterations[this.$scope.currentPage-1].estimate.newComponents.length > 0) {
      if(this.$scope.iterations[this.$scope.currentPage-1].hasOwnProperty('schedule') && this.
$scope.iterations[this.$scope.currentPage-1].schedule.hasOwnProperty('components') && this.
$scope.iterations[this.$scope.currentPage-1].schedule.components.length > 0){
        //console.log('Already copied');
        this.$scope.iterations[this.$scope.currentPage-1].schedule.wbs.forEach(function(date) {
          date.plannedCompletionDate = new Date(date.plannedCompletionDate);
          date.ActualCompletionDate = new Date(date.ActualCompletionDate);
        });
      }
      else {
        for(var i=0; i<this.$scope.iterations[this.$scope.currentPage-1].estimate.newComponents.length; i+
+) {
          this.$scope.iterations[this.$scope.currentPage-1].schedule.components.push({
            name: this.$scope.iterations[this.$scope.currentPage-1].estimate.newComponents[i].name,
            backLog: {
              PlannedTaskCompleted: 0,
              UnplannedTaskCompleted: 0,
              AdditionaltaskDiscovered: 0
            },
            componentIterationMap: []
          });
        }
        //console.log(this.$scope.iterations[this.$scope.currentPage-1].schedule.components);
        var len = this.$scope.iterations.length;
        this.$scope.iterations[this.$scope.currentPage-
1].schedule.components.forEach(function(component) {
          var arr1 = [];
          for(var j=0; j<len; j++) {
            var obj = {
              production: 0,
              mock: 0
            };
            arr1.push(Object.assign({}, obj));
          }
          component.componentIterationMap = Object.assign([], arr1);
        });
        //console.log(this.$scope.iterations[this.$scope.currentPage-1].schedule.components);
      }
      this.$scope.totalEp = this.$scope.iterations[this.$scope.currentPage-1].estimate.pEffort;
    }
    this.AssessIterationMap();
  });
 }
 AssessIterationMap() {
  var totalSum = 0;
  this.$scope.iterations[this.$scope.currentPage-1].schedule.components.forEach(function(component) {
```

```
      var sum = 0;
      component.componentIterationMap.forEach(function(itermap) {
        sum += (itermap.production+itermap.mock);
      });
      var atleastMethods = 0;
      //console.log(this.$scope.iterations[this.$scope.currentPage-1].estimate.newComponents);
      this.$scope.iterations[this.$scope.currentPage-1].estimate.newComponents.forEach(function(comp) {
        if(comp.name == component.name) {
          atleastMethods = comp.methods;
        }
      });
      //console.log(atleastMethods);
      if(sum >= atleastMethods) {
        component.computeComponentSum = sum;
        component.error = false;
      }
      else {
        component.computeComponentSum = sum;
        component.error = true;
      }
      totalSum += component.computeComponentSum;
    }, this);

    this.$scope.totalTasks = totalSum;
  }
  computeIterationSumTasks(iteration) {
    var sum = 0;
    //console.log('heeeee');
    //console.log(this.$scope.iterations);
    this.$scope.iterations[this.$scope.currentPage-1].schedule.components.forEach(function(component) {
      //console.log(component);
      sum += component.componentIterationMap[iteration].production  +
component.componentIterationMap[iteration].mock;
    });
    this.$scope.tasksValue = sum;
    return this.$scope.tasksValue;
  }
  computeIterationSumEp(iteration) {
    var sum = 0;
    this.$scope.iterations[this.$scope.currentPage-1].schedule.components.forEach(function(component) {
      sum += component.componentIterationMap[iteration].production  +
component.componentIterationMap[iteration].mock;
    });

    this.$scope.EpValue = this.$scope.totalTasks == 0 ? 0 : Math.ceil((this.$scope.totalEp/this.
$scope.totalTasks)*sum);
    this.computeIterationSumTasks(iteration);
    this.createWBS(iteration);
    return this.$scope.EpValue;
```

```
  }
  computeNextIteration(component) {
    //console.log(component);

    if(component.computeComponentSum >= component.backLog.PlannedTaskCompleted) {
      component.backLog.nextIter = component.computeComponentSum -
component.backLog.PlannedTaskCompleted + component.backLog.AdditionaltaskDiscovered;
      //console.log(component.backLog.nextIter);
    }
    else {
      component.backLog.nextIter = 'Error';
    }

  }
  addIteration() {
    ////console.log(this.$scope.Project);
    this.$scope.Project.Setup.noOfIteration++;
    this.$scope.Project.iterations.push({
      projectConfig: '',
      isUploaded: false,
      timeLog: [],
      schedule: {}
    });
    // Copy previous components
    ////console.log('copy');
    this.$scope.Project.iterations[this.$scope.Project.Setup.noOfIteration-1].estimate = {
      pLOC: this.$scope.Project.iterations[this.$scope.Project.Setup.noOfIteration-2].estimate.pLOC,
      pEffort: this.$scope.Project.iterations[this.$scope.Project.Setup.noOfIteration-2].estimate.pEffort,
      aLOC: this.$scope.Project.iterations[this.$scope.Project.Setup.noOfIteration-2].estimate.aLOC,
      aEffort: this.$scope.Project.iterations[this.$scope.Project.Setup.noOfIteration-2].estimate.aEffort
    }
    this.$scope.Project.iterations[this.$scope.Project.Setup.noOfIteration-1].estimate.newComponents =
function(project) {
      var theCopy = [];
      var arr1 = project.iterations[project.Setup.noOfIteration-2].estimate.newComponents;
      //console.log(arr1);
      for (var i = 0, len = arr1.length; i < len; i++) {
        theCopy[i] = Object.assign({}, arr1[i]);
      }
      //console.log(theCopy);
      return theCopy;
    }(this.$scope.Project);
    // setup last iteration, just added
    //console.log(this.$scope.Project.iterations[this.$scope.Project.Setup.noOfIteration-1].schedule);
    this.$scope.Project.iterations[this.$scope.Project.Setup.noOfIteration-1].schedule.components = [];
    this.$scope.iterations = this.$scope.Project.iterations;
    this.$scope.Project.iterations[this.$scope.currentPage-1].schedule.components = [];
    for(var i=0; i<this.$scope.Project.iterations[this.$scope.currentPage-
1].estimate.newComponents.length; i++) {
```

```
      this.$scope.Project.iterations[this.$scope.currentPage-1].schedule.components.push({
        name: this.$scope.iterations[this.$scope.currentPage-1].estimate.newComponents[i].name,
        backLog: {
          PlannedTaskCompleted: 0,
          UnplannedTaskCompleted: 0,
          AdditionaltaskDiscovered: 0
        },
        componentIterationMap: []
      });
    }

    //console.log(this.$scope.iterations[this.$scope.currentPage-1].schedule.components);
    var len = this.$scope.iterations.length;
    this.$scope.Project.iterations[this.$scope.currentPage-
1].schedule.components.forEach(function(component) {
      var arr1 = [];
      for(var j=0; j<len; j++) {
        var obj = {
          production: 0,
          mock: 0
        };
        arr1.push(Object.assign({}, obj));
      }
      component.componentIterationMap = Object.assign([], arr1);
    });
    var data = {
      Setup: this.$scope.Project.Setup,
      iterations: this.$scope.Project.iterations
    };
    this.$http.put('/api/projects/'+this.$stateParams.id, data).then(response => {
      //console.log(response);
    });
    this.$scope.totalItems = this.$scope.iterations.length*10;
    //this.getData();

    //Adding code for hiding
    for(var i=0; i<this.$scope.trackPages[0]-1; i++) {
      if(this.$scope.currentPage != 1) {
        this.$scope.iterations[this.$scope.currentPage-1].schedule.components.forEach(function(component)
{
          component.componentIterationMap[i].showIt = false;
        });
      }
    }
    for(var i=0; i<this.$scope.trackPages[1]-1; i++) {
      if(this.$scope.currentPage != 1) {
        this.$scope.iterations[this.$scope.currentPage-1].schedule.components.forEach(function(component)
{
          component.componentIterationMap[i].showIt = true;
```

```
      });
    }
   }
  }

  removeIteration() {
   if(this.$scope.Project.iterations.length > this.$scope.currentPage) {
    this.$scope.Project.Setup.noOfIteration--;
    this.$scope.Project.iterations.splice(this.$scope.Project.iterations.length-1, 1);
    this.$scope.Project.iterations[this.$scope.currentPage-
1].schedule.components.forEach(function(component) {
      component.componentIterationMap.splice(component.componentIterationMap.length-1, 1);
    });
    ////console.log(this.$scope.Project.iterations[this.$scope.currentPage-1].schedule.wbs.length +' : '+
this.$scope.Project.iterations[this.$scope.currentPage-
1].schedule.components[0].componentIterationMap.length);
    if(this.$scope.Project.iterations[this.$scope.currentPage-1].schedule.wbs.length > this.
$scope.Project.iterations[this.$scope.currentPage-
1].schedule.components[0].componentIterationMap.length) {
      ////console.log('hhhhhhhhhhhhhhhhhhhh');
      while(this.$scope.Project.iterations[this.$scope.currentPage-1].schedule.wbs.length != this.
$scope.Project.iterations[this.$scope.currentPage-
1].schedule.components[0].componentIterationMap.length) {
       this.$scope.Project.iterations[this.$scope.currentPage-1].schedule.wbs.splice(this.
$scope.Project.iterations[this.$scope.currentPage-1].schedule.wbs.length-1, 1);
      }
      this.$scope.iterations[this.$scope.currentPage-1].schedule.wbs = this.$scope.Project.iterations[this.
$scope.currentPage-1].schedule.wbs;
    }
    var data = {
     Setup: this.$scope.Project.Setup,
     iterations: this.$scope.Project.iterations
    };
    this.$http.put('/api/projects/'+this.$stateParams.id, data).then(response => {
     //console.log(response);
    });
    this.$scope.totalItems = this.$scope.iterations.length*10;
   }
  }

  saveProgress() {
   console.log(this.$scope.Project.iterations);
   var data = {
    iterations: this.$scope.Project.iterations
   };
   this.$http.put('/api/projects/'+this.$stateParams.id, data).then(response => {
    //console.log(response);
   });
  }
```

```
  createWBS(iter) {
    this.$scope.iterations[this.$scope.currentPage-
1].schedule.components[0].componentIterationMap.forEach(function(iter){
      if( this.$scope.iterations[this.$scope.currentPage-1].schedule.wbs.length < this.$scope.iterations[this.
$scope.currentPage-1].schedule.components[0].componentIterationMap.length) {
        this.$scope.iterations[this.$scope.currentPage-1].schedule.wbs.push({
          plannedEffort: 0,
          cumulativeEffort: 0,
          plannedVelocity: 0,
          cumulativePlannedVelocity: 0,
          plannedCompletionDate: new Date(),
          ActualEffort: 0,
          cumulativeActualEffort: 0,
          EarnedVelocity: 0,
          cumulativeEarnedVelocity: 0,
          ActualCompletionDate: new Date()
        });
      }
    }, this);
    //console.log(iter);
    this.$scope.iterations[this.$scope.currentPage-1].schedule.wbs[iter].plannedEffort = this.
$scope.EpValue;
    this.$scope.iterations[this.$scope.currentPage-1].schedule.wbs[iter].plannedVelocity = this.
$scope.tasksValue;
    if(iter == 0) {
      this.$scope.iterations[this.$scope.currentPage-1].schedule.wbs[iter].cumulativeEffort = this.
$scope.iterations[this.$scope.currentPage-1].schedule.wbs[iter].plannedEffort;
      this.$scope.iterations[this.$scope.currentPage-1].schedule.wbs[iter].cumulativePlannedVelocity =
this.$scope.iterations[this.$scope.currentPage-1].schedule.wbs[iter].plannedVelocity;
      this.$scope.iterations[this.$scope.currentPage-1].schedule.wbs[iter].cumulativeActualEffort = this.
$scope.iterations[this.$scope.currentPage-1].schedule.wbs[iter].ActualEffort;
      this.$scope.iterations[this.$scope.currentPage-1].schedule.wbs[iter].cumulativeEarnedVelocity = this.
$scope.iterations[this.$scope.currentPage-1].schedule.wbs[iter].EarnedVelocity;
    }
    else {
      this.$scope.iterations[this.$scope.currentPage-1].schedule.wbs[iter].cumulativeEffort = this.
$scope.iterations[this.$scope.currentPage-1].schedule.wbs[iter].plannedEffort + this.
$scope.iterations[this.$scope.currentPage-1].schedule.wbs[iter-1].cumulativeEffort;
      this.$scope.iterations[this.$scope.currentPage-1].schedule.wbs[iter].cumulativePlannedVelocity =
this.$scope.iterations[this.$scope.currentPage-1].schedule.wbs[iter].plannedVelocity + this.
$scope.iterations[this.$scope.currentPage-1].schedule.wbs[iter-1].cumulativePlannedVelocity;
      this.$scope.iterations[this.$scope.currentPage-1].schedule.wbs[iter].cumulativeActualEffort = this.
$scope.iterations[this.$scope.currentPage-1].schedule.wbs[iter].ActualEffort + this.$scope.iterations[this.
$scope.currentPage-1].schedule.wbs[iter-1].cumulativeActualEffort;
      this.$scope.iterations[this.$scope.currentPage-1].schedule.wbs[iter].cumulativeEarnedVelocity = this.
$scope.iterations[this.$scope.currentPage-1].schedule.wbs[iter].EarnedVelocity + this.
$scope.iterations[this.$scope.currentPage-1].schedule.wbs[iter-1].cumulativeEarnedVelocity;
    }
```

```
   var compareDate = new Date(this.$scope.iterations[this.$scope.currentPage-
1].schedule.wbs[iter].plannedCompletionDate);
   var bool = true;
   this.$scope.iterations[this.$scope.currentPage-1].schedule.calendar.forEach(function(day) {
     if(bool & day.hasOwnProperty('IterationEndHere') && day.IterationEndHere == true ) {
       var compDate = new Date(day.date);
       if(iter == 0 && compDate.getDate() == compareDate.getDate() && compDate.getDay() ==
compareDate.getDay() && compDate.getFullYear() == compareDate.getFullYear() ) {
         this.$scope.iterations[this.$scope.currentPage-1].schedule.wbs[iter].EarnedVelocity =
day.EarnedCumVelocityEndOfDay;
         bool = false;
       }
       if(iter > 0 && compDate.getDate() == compareDate.getDate() && compDate.getDay() ==
compareDate.getDay() && compDate.getFullYear() == compareDate.getFullYear() ) {
         this.$scope.iterations[this.$scope.currentPage-1].schedule.wbs[iter].EarnedVelocity =
day.EarnedCumVelocityEndOfDay - this.$scope.iterations[this.$scope.currentPage-1].schedule.wbs[iter-
1].EarnedVelocity;
         bool = false;
       }
     }
   }, this);
  }
  getBackLogData() {
   var sum = 0;
   var sum1 = 0;
   this.$scope.actualMinPerTask = this.$scope.iterations[this.$scope.currentPage-
1].schedule.components.forEach(function(comp) {
     sum += comp.backLog.PlannedTaskCompleted+comp.backLog.UnplannedTaskCompleted;
     sum1 += comp.backLog.nextIter;
   });
   this.$scope.actualMinPerTask = this.$scope.iterations[this.$scope.currentPage-1].schedule.wbs[this.
$scope.currentPage-1].cumulativeActualEffort / sum;
   this.$scope.remainingTasks = sum1;

  }

  addCalendar() {
   if(this.$scope.iterationEndsValue < this.$scope.iterations[this.$scope.currentPage-
1].schedule.wbs.length) {

     if(this.$scope.iterations[this.$scope.currentPage-1].schedule.calendar.length == 0) {
       //console.log(this.$scope.Project.Setup.startProject);
       this.$scope.iterations[this.$scope.currentPage-1].schedule.calendar = [];

       //copy previous
       if(this.$scope.currentPage-1 > 0) {
         for(var i=0; i<this.$scope.currentPage-1; i++) {
           this.$scope.iterations[this.$scope.currentPage-1].schedule.wbs[i].plannedCompletionDate = this.
$scope.iterations[this.$scope.currentPage-2].schedule.wbs[i].plannedCompletionDate
```

```
        }
        var first = new Date(this.$scope.iterations[this.$scope.currentPage-2].schedule.wbs[this.
$scope.currentPage-2].plannedCompletionDate).getTime()+24*60*60*1000;
        this.$scope.iterations[this.$scope.currentPage-1].schedule.calendar.push({
          date: new Date(first).getTime(),
          plannedAvailableMinutes : 0,
          PlannedVelocityEndOfDay : 0,
          ActualAvailableMinutes : 0,
          ActualBurnDownStartOfDay : 0,
          ActualBurnDownEndOfDay : 0,
          EarnedVelocityEndOfDay : 0
        });
       }
       else {
        var first = new Date(this.$scope.Project.Setup.startProject);
        this.$scope.iterations[this.$scope.currentPage-1].schedule.calendar.push({
          date: first.getTime(),
          plannedAvailableMinutes : 0,
          PlannedVelocityEndOfDay : 0,
          ActualAvailableMinutes : 0,
          ActualBurnDownStartOfDay : 0,
          ActualBurnDownEndOfDay : 0,
          EarnedVelocityEndOfDay : 0
        });
       }

     }
     else {
       var len = this.$scope.iterations[this.$scope.currentPage-1].schedule.calendar.length;
       var nextDate = new Date(this.$scope.iterations[this.$scope.currentPage-1].schedule.calendar[len -
1].date).getTime()+24*60*60*1000;
       this.$scope.iterations[this.$scope.currentPage-1].schedule.calendar.push({
         date: nextDate,
         plannedAvailableMinutes : 0,
         PlannedVelocityEndOfDay : 0,
         ActualAvailableMinutes : 0,
         ActualBurnDownStartOfDay : 0,
         ActualBurnDownEndOfDay : 0,
         EarnedVelocityEndOfDay : 0
       });
     }
   }
  }
  removeCalendar() {
     if(this.$scope.iterations[this.$scope.currentPage-1].schedule.calendar.length > 0) {
       if(this.$scope.iterations[this.$scope.currentPage-1].schedule.calendar[this.$scope.iterations[this.
$scope.currentPage-1].schedule.calendar.length-1].IterationEndHere == true) {
         this.$scope.iterationEndsValue--;
       }
```

```
      this.$scope.iterations[this.$scope.currentPage-1].schedule.calendar.splice(this.$scope.iterations[this.
$scope.currentPage-1].schedule.calendar.length-1, 1);
    }
  }

  plannedCumulativeMinutes(index) {
    if(index == 0) {
      this.$scope.iterations[this.$scope.currentPage-1].schedule.calendar[index].plannedCumMinutes =
this.$scope.iterations[this.$scope.currentPage-1].schedule.calendar[index].plannedAvailableMinutes;
      this.$scope.iterations[this.$scope.currentPage-1].schedule.calendar[index].BurnDownStartOfDay =
this.$scope.totalEp;
      // this.$scope.iterations[this.$scope.currentPage-
1].schedule.calendar[index].ActualBurnDownStartOfDay = this.$scope.totalEp;
      this.$scope.iterations[this.$scope.currentPage-1].schedule.calendar[index].BurnDownEndOfDay =
this.$scope.iterations[this.$scope.currentPage-1].schedule.calendar[index].BurnDownStartOfDay - this.
$scope.iterations[this.$scope.currentPage-1].schedule.calendar[index].plannedAvailableMinutes;
      this.$scope.iterations[this.$scope.currentPage-
1].schedule.calendar[index].ActualBurnDownEndOfDay = this.$scope.iterations[this.$scope.currentPage-
1].schedule.calendar[index].ActualBurnDownStartOfDay - this.$scope.iterations[this.$scope.currentPage-
1].schedule.calendar[index].ActualAvailableMinutes;
      if(this.$scope.iterations[this.$scope.currentPage-
1].schedule.calendar[index].ActualBurnDownEndOfDay < 0) {
        this.$scope.iterations[this.$scope.currentPage-
1].schedule.calendar[index].ActualBurnDownEndOfDay = 0;
      }
      this.$scope.iterations[this.$scope.currentPage-
1].schedule.calendar[index].PlannedCumVelocityEndOfDay = this.$scope.iterations[this.
$scope.currentPage-1].schedule.calendar[index].PlannedVelocityEndOfDay;
      this.$scope.iterations[this.$scope.currentPage-
1].schedule.calendar[index].ActualAvailableCumMinutes = this.$scope.iterations[this.
$scope.currentPage-1].schedule.calendar[index].ActualAvailableMinutes;
      this.$scope.iterations[this.$scope.currentPage-
1].schedule.calendar[index].EarnedCumVelocityEndOfDay = this.$scope.iterations[this.
$scope.currentPage-1].schedule.calendar[index].EarnedVelocityEndOfDay;
    }
    else {
      this.$scope.iterations[this.$scope.currentPage-1].schedule.calendar[index].plannedCumMinutes =
this.$scope.iterations[this.$scope.currentPage-1].schedule.calendar[index].plannedAvailableMinutes +
this.$scope.iterations[this.$scope.currentPage-1].schedule.calendar[index-1].plannedCumMinutes;
      this.$scope.iterations[this.$scope.currentPage-1].schedule.calendar[index].BurnDownStartOfDay =
this.$scope.iterations[this.$scope.currentPage-1].schedule.calendar[index-1].BurnDownEndOfDay;
      this.$scope.iterations[this.$scope.currentPage-
1].schedule.calendar[index].ActualBurnDownStartOfDay = this.$scope.iterations[this.
$scope.currentPage-1].schedule.calendar[index-1].ActualBurnDownEndOfDay;
      this.$scope.iterations[this.$scope.currentPage-1].schedule.calendar[index].BurnDownEndOfDay =
this.$scope.iterations[this.$scope.currentPage-1].schedule.calendar[index].BurnDownStartOfDay - this.
$scope.iterations[this.$scope.currentPage-1].schedule.calendar[index].plannedAvailableMinutes;
      this.$scope.iterations[this.$scope.currentPage-
1].schedule.calendar[index].ActualBurnDownEndOfDay = this.$scope.iterations[this.$scope.currentPage-
```

```
1].schedule.calendar[index].ActualBurnDownStartOfDay - this.$scope.iterations[this.$scope.currentPage-
1].schedule.calendar[index].ActualAvailableMinutes;
    if(this.$scope.iterations[this.$scope.currentPage-
1].schedule.calendar[index].ActualBurnDownEndOfDay < 0) {
      this.$scope.iterations[this.$scope.currentPage-
1].schedule.calendar[index].ActualBurnDownEndOfDay = 0;
    }
    this.$scope.iterations[this.$scope.currentPage-
1].schedule.calendar[index].PlannedCumVelocityEndOfDay = this.$scope.iterations[this.
$scope.currentPage-1].schedule.calendar[index].PlannedVelocityEndOfDay + this.$scope.iterations[this.
$scope.currentPage-1].schedule.calendar[index-1].PlannedCumVelocityEndOfDay;
    this.$scope.iterations[this.$scope.currentPage-
1].schedule.calendar[index].ActualAvailableCumMinutes = this.$scope.iterations[this.
$scope.currentPage-1].schedule.calendar[index].ActualAvailableMinutes + this.$scope.iterations[this.
$scope.currentPage-1].schedule.calendar[index-1].ActualAvailableCumMinutes;
    this.$scope.iterations[this.$scope.currentPage-
1].schedule.calendar[index].EarnedCumVelocityEndOfDay = this.$scope.iterations[this.
$scope.currentPage-1].schedule.calendar[index].EarnedVelocityEndOfDay + this.$scope.iterations[this.
$scope.currentPage-1].schedule.calendar[index-1].EarnedCumVelocityEndOfDay;
  }
  var compareDate = new Date(this.$scope.iterations[this.$scope.currentPage-
1].schedule.calendar[index].date);
  if(index == this.$scope.iterations[this.$scope.currentPage-1].schedule.calendar.length-1 && this.
$scope.iterationEndsValue > this.$scope.iterations[this.$scope.currentPage-1].schedule.wbs.length-1 &&
this.$scope.iterations[this.$scope.currentPage-1].schedule.wbs[this.$scope.iterationEndsValue-
1].cumulativeEffort > this.$scope.iterations[this.$scope.currentPage-
1].schedule.calendar[index].plannedCumMinutes) {
    this.$scope.iterations[this.$scope.currentPage-1].schedule.calendar[index].IterationEnd = '';
    this.$scope.iterations[this.$scope.currentPage-1].schedule.calendar[index].IterationEndHere = false;
    this.$scope.iterationEndsValue--;
  }
  var wb = this.$scope.iterations[this.$scope.currentPage-1].schedule.wbs[this.
$scope.iterationEndsValue];
  //wb.plannedCompletionDate = 0;

  if(wb && wb.hasOwnProperty('cumulativeEffort')) {
    var effortSoFar = wb.cumulativeEffort;
    if( effortSoFar <= this.$scope.iterations[this.$scope.currentPage-
1].schedule.calendar[index].plannedCumMinutes ) {
      wb.plannedCompletionDate = new Date(this.$scope.iterations[this.$scope.currentPage-
1].schedule.calendar[index].date);
      this.$scope.iterations[this.$scope.currentPage-1].schedule.calendar[index].IterationEnd = 'text-
danger bg-danger';
      this.$scope.iterations[this.$scope.currentPage-1].schedule.calendar[index].IterationEndHere = true;
      this.$scope.iterationEndsValue++;
    }
    else if(effortSoFar > this.$scope.iterations[this.$scope.currentPage-
1].schedule.calendar[index].plannedCumMinutes && this.$scope.iterations[this.$scope.currentPage-
1].schedule.calendar[index].IterationEndHere == true) {
```

```
      this.$scope.iterations[this.$scope.currentPage-1].schedule.calendar[index].IterationEnd = '';
      this.$scope.iterations[this.$scope.currentPage-1].schedule.calendar[index].IterationEndHere = false;
      this.$scope.iterationEndsValue--;
     }
     else if(effortSoFar > this.$scope.iterations[this.$scope.currentPage-
1].schedule.calendar[index].plannedCumMinutes){
      this.$scope.iterations[this.$scope.currentPage-1].schedule.calendar[index].IterationEnd = '';
      this.$scope.iterations[this.$scope.currentPage-1].schedule.calendar[index].IterationEndHere = false;
     }
    }

   ('Itera'+this.$scope.iterationEndsValue);
   return this.$scope.iterations[this.$scope.currentPage-1].schedule.calendar[index].plannedCumMinutes;
  }

  getTotalTime(index) {
   var timeTaken = 0;
   var bool = true;
   this.$scope.Project.iterations.forEach(function(iteration, i) {
    iteration.timeLog.forEach(function(log) {
     if(i == index) {
      timeTaken += parseInt(log.timeTaken);
      if(log.interrupt) {
       timeTaken -= parseInt(log.interrupt);
      }
      this.$scope.iterations[this.$scope.currentPage-1].schedule.wbs[index].ActualEffort = timeTaken;
      bool = false;
     }
    }, this);
   }, this);
   return true;
  }

  getTimePerDay(iter, index) {
   // ////console.log(index);
   var timeTaken = 0;
   this.$scope.Project.iterations.forEach(function(iteration, i) {
    iteration.timeLog.forEach(function(log) {
     var logDate = new Date(Date.parse(log.dateFormat));
     var calendarDate = new Date(this.$scope.iterations[this.$scope.currentPage-
1].schedule.calendar[index].date);
     if(iter._id == iteration._id && logDate.getDate() == calendarDate.getDate() && logDate.getDay()
== calendarDate.getDay() && logDate.getFullYear() == calendarDate.getFullYear()) {
      timeTaken += parseInt(log.timeTaken);
      if(log.interrupt) {
       timeTaken -= parseInt(log.interrupt);
      }
      this.$scope.iterations[this.$scope.currentPage-1].schedule.calendar[index].ActualAvailableMinutes
= timeTaken;
```

```
      }
    }, this);
  }, this);
  return true;
 }

}

angular.module('pcseappApp')
 .component('schedule', {
  templateUrl: 'app/schedule/schedule.html',
  controller: ScheduleComponent
 });

})();
```

schedule.html

```
<navbar></navbar>
<div id="page-content-wrapper">
 <div class="container" ng-show="visible">
  <label>Choose the Iteration:</label>
  <div><ul uib-pagination total-items="totalItems" ng-model="currentPage" max-size="maxSize" ng-
change="pageChanged(); $ctrl.AssessIterationMap()"></ul></div>
  <div class="row">
   <div class="col-md-4 col-lg-4">
    <h1>
     <button class="btn btn-primary btn-create" type="button" ng-click="$ctrl.saveProgress()" data-
dismiss="modal">Save Progress</button>
    </h1>
   </div>
  </div>
 </div>

 <div ng-repeat="iteration in iterations.slice(((currentPage-1)*itemsPerPage),
((currentPage)*itemsPerPage))">
  <div class="container-fluid">
   <div class="row">
    <button id="addButton" type="button" class="btn btn-danger" ng-click="$ctrl.addIteration()"
title="Add a New Iteration">+</button>
    <button id="addButton" type="button" class="btn btn-danger" ng-click="$ctrl.removeIteration()"
title="Remove last Iteration">-</button>
    <div class="col-md-12 col-lg-12">
     <div class="panel panel-primary">
      <div class="panel-heading">
       Component-Iteration Map
      </div>
      <div class="panel-body">
       <table class="table table-bordered table-striped table-sm">
```

```
            <thead>
              <tr>
                <th colspan="2"></th>
                <th ng-hide="iteration.showIt" ng-repeat="iteration in
iteration.schedule.components[0].componentIterationMap" colspan="2">Iteration {{$index+1}}</th>
              </tr>
            </thead>
            <thead>
              <tr>
                <th>#</th>
                <th>Name</th>
                <th ng-hide="iteration.showIt" ng-repeat-start ="iteration in
iteration.schedule.components[0].componentIterationMap">Production</th>
                <th ng-hide="iteration.showIt" ng-repeat-end>Mock</th>
              </tr>
            </thead>
            <tbody ng-repeat="component in iteration.schedule.components">
              <tr>
                <th scope="row">{{$index+1}}</th>
                <td>{{component.name}}</td>

                <td ng-repeat-start="iterationMap in component.componentIterationMap" ng-
hide="iterationMap.showIt"><input ng-change="$ctrl.AssessIterationMap()" class="form-control"
type="number" ng-model="iterationMap.production"/></td>
                <td ng-repeat-end ng-hide="iterationMap.showIt"><input class="form-control"
type="number" ng-model="iterationMap.mock" ng-change="$ctrl.AssessIterationMap()"/></td>
                <td>{{component.computeComponentSum}}<span class="text-danger" ng-
show='component.error'> (Tasks &lt Methods)</span></td>
              </tr>
            </tbody>
            <tfoot>
              <tr>
                <th colspan="2">Number of Tasks</th>
                <th ng-repeat-start ="iteration in iteration.schedule.components[0].componentIterationMap"
ng-hide="iteration.showIt"></th>
                <th ng-repeat-end ng-
hide="iteration.showIt">{{$ctrl.computeIterationSumTasks($index)}}</th>
                <th>Total Tasks = {{totalTasks}}</th>
              </tr>
              <tr>
                <th colspan="2">Ep per Iteration</th>
                <th ng-repeat-start ="iteration in iteration.schedule.components[0].componentIterationMap"
ng-hide="iteration.showIt"></th>
                <th ng-repeat-end ng-
hide="iteration.showIt">{{$ctrl.computeIterationSumEp($index)}}</th>
                <th>Total Ep = {{totalEp}}</th>
              </tr>
            </tfoot>
          </table>
```

```
        </div>
       </div>
      </div>
     </div>
     <div class="row">
      <div class="col-md-12 col-lg-12">
       <div class="panel panel-primary">
        <div class="panel-heading">
         WBS
        </div>
        <div class="panel-body">
         <table class="table table-bordered table-striped table-sm">
          <thead>
           <tr>
            <th>Iteration</th>
            <th>Planned Effort(min)</th>
            <th>Cumulative Planned Effort</th>
            <th>Planned Velocity</th>
            <th>Cumulative Planned Velocity</th>
            <th>Planned Completion</th>
            <th>Actual Effort(min)</th>
            <th>Cumulative Actual Effort</th>
            <th>Earned Velocity</th>
            <th>Cumulative Earned Velocity</th>
            <th>Actual Completion</th>
           </tr>
          </thead>
          <tbody>
           <tr ng-repeat="wbs in iteration.schedule.wbs">
            <th scope="row">{{$index+1}}</th>
            <td>{{wbs.plannedEffort}}</td>
            <td>{{wbs.cumulativeEffort}}</td>
            <td>{{wbs.plannedVelocity}}</td>
            <td>{{wbs.cumulativePlannedVelocity}}</td>
            <td><input class="form-control" type="date" ng-
model="wbs.plannedCompletionDate"/></td>
            <td><input class="form-control" type="number" ng-model="wbs.ActualEffort" ng-
if="$ctrl.getTotalTime($index)" /></td>
            <td>{{wbs.cumulativeActualEffort}}</td>
            <td><input class="form-control" type="number" ng-model="wbs.EarnedVelocity"/></td>
            <td>{{wbs.cumulativeEarnedVelocity}}</td>
            <td><input class="form-control" type="date" ng-
model="wbs.ActualCompletionDate"/></td>
           </tr>
          </tbody>
         </table>
        </div>
       </div>
      </div>
```

```
    </div>
    <div class="row">
      <button id="addButton" type="button" class="btn btn-danger" ng-click="$ctrl.addCalendar()"
title="Add a New Day">+</button>
      <button id="addButton" type="button" class="btn btn-danger" ng-click="$ctrl.removeCalendar()"
title="Remove last Day">-</button>
      <div class="col-md-12 col-lg-12">
        <div class="panel panel-primary">
          <div class="panel-heading">
            Calender
          </div>
          <div class="panel-body">
            <table class="table table-bordered table-sm">
              <thead>
                <tr>
                  <th>Date</th>
                  <th>Planned Available minutes</th>
                  <th>Planned Cumulative minutes</th>
                  <th>BurnDown at start of Day</th>
                  <th>BurnDown at end of Day</th>
                  <th>Planned velocity at end of Day</th>
                  <th>Cumulative Planned velocity</th>
                  <th>Actual Available minutes</th>
                  <th>Actual Cumulative minutes</th>
                  <th>Actual BurnDown at start of Day</th>
                  <th>Actual BurnDown at end of Day</th>
                  <th>Earned velocity at end of Day</th>
                  <th>Cumulative Earned velocity</th>
                </tr>
              </thead>
              <tbody ng-repeat="dayItem in iteration.schedule.calendar">
                <tr ng-class="dayItem.IterationEnd">
                  <th scope="row">{{dayItem.date | date:'MM/dd/yyyy' }}</th>
                  <td><input class="form-control" type="number" ng-
model="dayItem.plannedAvailableMinutes" ng-
change="$ctrl.plannedCumulativeMinutes($index)"/></td>
                  <td>{{$ctrl.plannedCumulativeMinutes($index)}}</td>
                  <td>{{dayItem.BurnDownStartOfDay}}</td>
                  <td>{{dayItem.BurnDownEndOfDay}}</td>
                  <td><input class="form-control" type="number" ng-
model="dayItem.PlannedVelocityEndOfDay" /></td>
                  <td>{{dayItem.PlannedCumVelocityEndOfDay}}</td>
                  <td><input class="form-control" type="number" ng-
model="dayItem.ActualAvailableMinutes" ng-if="$ctrl.getTimePerDay(iteration, $index)" /></td>
                  <td>{{dayItem.ActualAvailableCumMinutes}}</td>
                  <td><input class="form-control" type="number" ng-
model="dayItem.ActualBurnDownStartOfDay" /></td>
                  <td><input class="form-control" type="number" ng-
model="dayItem.ActualBurnDownEndOfDay" /></td>
```

```
                <td><input class="form-control" type="number" ng-
model="dayItem.EarnedVelocityEndOfDay" /></td>
                <td>{{dayItem.EarnedCumVelocityEndOfDay}}</td>
                <td ng-show="dayItem.IterationEndHere">Iteration End Here</td>
              </tr>
            </tbody>
          </table>
        </div>
      </div>
    </div>
    <div class="row">
      <div class="col-md-12 col-lg-12">
        <div class="panel panel-primary">
          <div class="panel-heading">
            Backlog (Hint: Click of rows to compute Tasks for next Iteration)
          </div>
          <div class="panel-body" ng-click="$ctrl.getBackLogData()">
            <table class="table table-bordered table-striped table-sm">
              <thead>
                <tr>
                  <th>Component(s)</th>
                  <th>Planned Tasks</th>
                  <th>Planned Tasks Completed</th>
                  <th>Unplanned Tasks Completed</th>
                  <th>Additional Tasks discovered</th>
                  <th>Tasks next iteration</th>
                </tr>
              </thead>
              <tbody ng-repeat="component in iteration.schedule.components" ng-
click="$ctrl.computeNextIteration(component)">
                <tr>
                  <td>{{component.name}}</td>
                  <td>{{component.computeComponentSum}}</td>
                  <td><input class="form-control" type="number" ng-
change="$ctrl.computeNextIteration(component)" ng-
model="component.backLog.PlannedTaskCompleted"/></td>
                  <td><input class="form-control" type="number" ng-
change="$ctrl.computeNextIteration(component)" ng-
model="component.backLog.UnplannedTaskCompleted"/></td>
                  <td><input class="form-control" type="number" ng-
change="$ctrl.computeNextIteration(component)" ng-
model="component.backLog.AdditionaltaskDiscovered"/></td>
                  <td>{{component.backLog.nextIter}}</td>
                </tr>
              </tbody>
            </table>
            <div class="row">
              <div class="col-md-4 col-lg-4">
```

```
            Actual minutes per tasks: {{actualMinPerTask}}
           </div>
           <div class="col-md-4 col-lg-4">
            Remaining tasks: {{remainingTasks}}
           </div>
           <div class="col-md-4 col-lg-4">
           New burndown: {{actualMinPerTask * remainingTasks}}
           </div>
          </div>
         </div>
        </div>
       </div>
      </div>
     </div>
    </div>
</div>
```

Iteration 01

User Stories

Story Set	1		
Objective:	To explore the functionality of Add Project component.		
Story	**As a …**	**I want to …**	**So I can …**
Story -1	user (anonymous/logged in)	have the ability to add project name and description	append a new project in the existing list of projects in system.
Story -2	user (anonymous/logged in)	have the ability to edit the project	Either change the project name/description or delete the project itself.
Story -3	user (anonymous/logged in)	view all the existing projects	choose either of them and do the needful tasks within that project scope.
Story -4	software component in PCSEDesk	take the details from user about the project	store them in the system and enable proper services required for a project.
Story -5	software component in PCSEDesk	read the data from the system related to the existing projects	Restore the system back to how it was before system was closed and display all the existing projects even after the system is restarted.
Story -6	software component in PCSEDesk	validate the information	store them and use them for business logic and making changes to their respective views.
Story -7	user (anonymous/logged in)	filter the projects	filter the resulting project for further usage.
Story -8	user (anonymous/logged in)	click the project name	open and work on the specific project being clicked.
Story -9	software component in PCSEDesk	inject separate views based on the availability of the projects	display a meaningful message to the user about unavailability of projects.

Story Set	2		
Objective:	To explore the functionality of Dashboard component.		
Story	**As a …**	**I want to …**	**So I can …**
Story -1	user (anonymous/logged in)	have the ability to plan the project details like project duration, no. of iterations, no. of working days in a week	setup the project for different iterations and create their respective views on the system.
Story -2	user (anonymous/logged in)	have the ability to add the details within each iteration the expected/actual artifacts like Lines of code, no. of methods/components and Effort	keep a track of the project and its progress so far.
Story -3	software component in PCSEDesk	take the details from user about the iterations	store them and use them for business logic and making changes to their respective views.
Story -4	software component in PCSEDesk	compare the expected data from user and actual data from time log	produce meaningful views on Burn down component.
Story -5	user (anonymous/logged in)	have the ability to edit the iterations which are still to come	keep the project deadline within the threshold.
Story -6	software component in PCSEDesk	lock the iteration which corresponds to the current date	only have the ability to modify the iterations from next cycle onwards.
Story -7	software component in PCSEDesk	validate the information	store them and use them for business logic and making changes to their respective views.

Story Set	3		
Objective:	To explore the functionality of Time Log component.		
Story	**As a …**	**I want to …**	**So I can …**
Story -1	user (anonymous/logged in)	have the ability to add iteration details related to time like date, start time, end time, interrupt, activity, special comments	track the time taken for each activity.
Story -2	user (anonymous/logged in)	have the ability to edit these details	override the system generated information.
Story -3	software component in PCSEDesk	generate date and delta for the user activities	calculate the total time taken for each/individual activity within an iteration
Story -4	software component in PCSEDesk	store the information mentioned above in the system	Use it along with business logic and restore the information for their respective views.
Story -5	user (anonymous/logged in)	view all the time logs for each iteration completed so far/in running state	keep a track of time logs from past and view the painful areas of concern.
Story -6	software component in PCSEDesk	validate the information	store them and use them for business logic and making changes to their respective views.

Story Set	4		
Objective:	To explore the functionality of Change Log component.		
Story	**As a …**	**I want to …**	**So I can …**
Story -1	user (anonymous/logged in)	have the ability to add iteration details related to changes to the system like type, inject activity, remove activity, fix time,fix Sequence, Description of change	track the changes done so far to the system and bugs found so far.
Story -2	user (anonymous/logged in)	have the ability to edit these details	override the system generated information.
Story -3	software component in PCSEDesk	generate sequence number, date for the user activities	calculate the total no. of bugs so far and filter the area of concern in PCSE development
Story -4	software component in PCSEDesk	store the information mentioned above in the system	Use it along with business logic and restore the information for their respective views.
Story -5	user (anonymous/logged in)	view all the change logs for each iteration completed so far/in running state	keep a track of change logs from past and view the painful areas of concern where most of the bugs occur.
Story -6	software component in PCSEDesk	validate the information	store them and use them for business logic and making changes to their respective views.

Story Set	5		
Objective:	To explore the functionality of Burn Down component.		
Story	**As a …**	**I want to …**	**So I can …**
Story -1	user (anonymous/logged in)	view and manipulate the data visualizations	override the existing iteration projections and scale the iteration to match the needs of project.
Story -2	user (anonymous/logged in)	filter the results based on different criteria	visualize different data and track the progress.
Story -3	user (anonymous/logged in)	View 2 different visualization namely Iteration Burn down chart and Release burn down chart	track the progress
Story -4	software component in PCSEDesk	pull the information from the system regarding each iteration for a project	create the data visualizations.
Story -5	software component in PCSEDesk	track the canvas for any changes done by user to the visualization	produce the changes on the views and their respective visualizations.

Architecture

Use this worksheet to describe components required to implement the specifications. List new components and base comp include extraneous components.

Component Name:	Router – Add Project
Design Approach:	
Parent Component:	
Component Type:	
Collaborators:	Controller – Add Project, View – Add Project
Operations:	Responsible for routing to the proper path of main project page along with respective view.

Component Name:	Controller – Add Project
Design Approach:	
Parent Component:	
Component Type:	
Collaborators:	Model – Add Project, View – Add Project
Operations:	Responsible for controlling the input data, applying validation and storing for further usage.

Component Name:	Model – Add Project
Design Approach:	
Parent Component:	
Component Type:	
Collaborators:	
Operations:	Responsible for storing the projects and their respective details.

Component Name:	View – Add Project
Design Approach:	
Parent Component:	
Component Type:	
Collaborators:	Controller – Add Project, Model – Add Project
Operations:	Responsible for generating a view which gives all the abilities covered in user stories.

Component Name:	app.js
Design Approach:	
Parent Component:	
Component Type:	
Collaborators:	All Controllers and Routers
Operations:	Responsible for compiling, executing and running the application.

Component Name:	style.css
Design Approach:	
Parent Component:	
Component Type:	
Collaborators:	
Operations:	Responsible for styling the views appropriately.

Component Name:	index.html
Design Approach:	
Parent Component:	
Component Type:	
Collaborators:	
Operations:	Responsible for generating a dynamic view port for application.

Component Name:	Router – Dashboard
Design Approach:	
Parent Component:	
Component Type:	
Collaborators:	
Operations:	Responsible for routing to the proper path of main project dashboard along with respective view.

Component Name:	Controller – Dashboard
Design Approach:	
Parent Component:	
Component Type:	
Collaborators:	
Operations:	Responsible for controlling the input data, applying validation and storing for further usage.

Component Name:	View – Dashboard
Design Approach:	
Parent Component:	
Component Type:	
Collaborators:	Controller – Dashboard, Model – Add Project
Operations:	Responsible for generating a view which gives all the abilities covered in user stories.

Component Name:	Router – Time Log
Design Approach:	
Parent Component:	
Attributes (optional):	
Component Type:	
Collaborators:	
Operations:	Responsible for routing to the proper path of main project Time Log along with respective view.

Component Name:	Controller – Time Log
Design Approach:	
Parent Component:	
Attributes (optional):	
Component Type:	
Collaborators:	
Operations:	Responsible for controlling the input data, applying validation and storing for further usage.

Component Name:	Router – Time Log
Design Approach:	
Parent Component:	
Attributes (optional):	
Component Type:	
Collaborators:	
Operations:	Responsible for routing to the proper path of main project Time Log along with respective view.

Component Name:	View – Time Log
Design Approach:	
Parent Component:	
Attributes (optional):	
Component Type:	
Collaborators:	Controller – Time Log, Model – Add Project
Operations:	Responsible for generating a view which gives all the abilities covered in user stories.

Component Name:	Router – Change Log
Design Approach:	
Parent Component:	
Attributes (optional):	
Component Type:	
Collaborators:	
Operations:	Responsible for routing to the proper path of main project Change Log along with respective view.

Component Name:	Controller – Change Log
Design Approach:	
Parent Component:	
Attributes (optional):	
Component Type:	
Collaborators:	
Operations:	Responsible for controlling the input data, applying validation and storing for further usage.

Component Name:	View – Change Log
Design Approach:	
Parent Component:	
Attributes (optional):	
Component Type:	
Collaborators:	Controller – Change Log, Model – Add Project
Operations:	Responsible for generating a view which gives all the abilities covered in user stories.

Component Name:	Router – Burn Down
Design Approach:	
Parent Component:	
Attributes (optional):	
Component Type:	
Collaborators:	
Operations:	Responsible for routing to the proper path of main project Burn Down along with respective view.

Component Name:	Controller – Burn Down
Design Approach:	
Parent Component:	
Attributes (optional):	
Component Type:	
Collaborators:	
Operations:	Responsible for controlling the input data, applying validation and storing for further usage.

Component Name:	View – Burn Down
Design Approach:	
Parent Component:	
Attributes (optional):	
Component Type:	
Collaborators:	Controller – Burn Down, Model – Add Project
Operations:	Responsible for generating a view which gives all the abilities covered in user stories.

	Estimated				Actual		
New Components	Method Count	Rel Size	LOCr	Reuseable?	Method Count	LOCa	Reusable?
Router – Add Project	1	S	1	Yes	1	6	Yes
Controller – Add Project	4	M	4	Yes	5	50	Yes
Model – Add Project	1	VL	1	Yes	1	1	Yes
View – Add Project	1	L	1	Yes	1	64	Yes
app.js	2	M	2	Yes	2	23	Yes
style.css	1	VL	1	Yes	1	200	Yes
index.html	1	L	1	Yes	1	43	Yes
Router – Dashboard	1	S	1	Yes			
Controller – Dashboard	4	L	4	Yes			
View – Dashboard	1	L	1	Yes			
Router – TimeLog	1	S	1	Yes			
Controller – TimeLog	4	M	4	Yes			
View – TimeLog	1	M	1	Yes			
Router – ChangeLog	1	S	1	Yes			
Controller – ChangeLog	4	M	4	Yes			
View – ChangeLog	1	M	1	Yes			
Router – Burndown	1	S	1	Yes			
Controller – Burndown	10	VL	10	Yes			
View – Burndown	1	VL	1	Yes			
			0				
Totals			41	41		387	387

Component-Iteration Map

	Component	Iteration 1	Iteration 2	Iteration 3	Iteration 4	Iteration 5	Iteration 6	Iteration 7	Iteration 8	Iteration 9	Iteration 10
Base Components											
New Components	Router –	1									
	Controller –	4									
	Model – Add	1									
	View – Add	1									
	app.js	2									
	style.css	1									
	index.html	1									
	Router –			1							
	Controller –			4							
	View –			1							
	Router –		1								
	Controller –		4								
	View –		1								
	Router –				1						
	Controller –				2	1	1				
	View –				1						
	Router –				1						
	Controller –				5	3	2				
	View –				1						
:Number of tasks		11	6	6	11	4	3	0	0	0	0
Effort:		0	0	0	0	0	0	0	0	0	0

WBS

Iteration Number	Planned Effort (minutes)	Cumulative Planned Effort	Planned Velocity	Cumulative Planned Velocity	Planned Completion Date	Actual Effort (minutes)	Cum Actual Effort	Earned Velocity	Cumulative Earned Velocity	Actual Completion Date
1	3600	3600	220	220	3/23/2016	3/13/1909	3360	8/7/1900	220	Due Date
2	3600	7200		220	4/6/2016		3360		440	
3	3600	10800		220	4/20/2016		3360		440	
4	3600	14400		220	5/4/2016		3360		440	
5	3600	18000		220	8/15/2016		3360		440	
6	3600	21600		220	8/31/2016		3360		440	
7		21600		220			3360		440	
8		21600		220			3360		440	
9		21600		220			3360		440	
10		21600		220			3360		440	

Calendar

Day #	Date	Available Minutes	Cumulative Minutes	Planned Velocity	Cumulative Planned Velocity	Actual Minutes	Cumulative Actual Minutes	Earned Velocity	Cumulative Earned Velocity
1	Due Date - 13		0		0		0		0
2	Due Date - 12		0		0		0		0
3	Due Date - 11		0		0		0		0
4	Due Date - 10		0		0		0		0
5	Due Date - 9		0		0		0		0
6	Due Date - 8		0		0		0		0
7	Due Date - 7		0		0		0		0
8	Due Date - 6		0		0		0		0
9	Due Date - 5		0		0		0		0
10	Due Date - 4		0		0		0		0
11	Due Date - 3		0		0		0		0
12	Due Date - 2		0		0		0		0
13	Due Date - 1		0		0		0		0
14	Due Date - 0		0		0		0		0
15	Due Date - 1		0		0		0		0
16	Due Date		0		0		0		0
17	Due Date + 1		0		0		0		0
18	Due Date + 2		0		0		0		0

Change Log

Use this worksheet to record changes to made to artifacts after the project is underway. Record repairs to defects, as well as changes that result from requirements clarifications/changes.

Change Designation	Description
Requirements Change	Changes to requirements
Requirements Clarification	Clarifications to requirements
Product syntax	Syntax flaws in the deliverable product
Product logic	Logic flaws in the deliverable product
Product interface	Flaws in the interface of a component of the deliverable product
Product checking	Flaws with boundary/type checking within a component of the deliverable product
Test syntax	Syntax flaws in the test code
Test logic	Logic flaws in the test code
Test interface	Flaws in the interface of a component of the test code
Test checking	Flaws with boundary/type checking within a component of the test code
Bad Smell	Refactoring changes (please note the bad smell in the defect description)

Number	Date	Type	Where Injected	Inject Iteration	Where Removed	Remove Iteration	Fix Time	Fix Reference	Description
1									
2									
3									

Time Recording Log

Date	Start Time	Stop Time	Interrupt	Delta	Activity	Iteration	Comments
3/2/2016	10:00 AM	12:00 PM	0	120	Analysis	1	Made a mock up for design and analysis
3/2/2016	5:00 PM	6:00 PM	10	50	Analysis	1	Analyzed and wrote the User stories
3/3/2016	12:00 PM	6:00 PM	30	330	Architecture	1	Constructed CRC cards for the project components
3/4/2016	12:00 PM	6:00 PM	20	340	Project planning	1	Estimation for the project using Component-based Estimation
3/7/2016	12:00 PM	2:00 PM	0	120	Interation plannir	1	Worked on Component-iteration map
3/7/2016	2:00 PM	6:00 PM	0	240	Construction	1	Worked on Add Project feature.
3/8/2016	3:00 PM	11:00 PM	0	480	Construction	1	Worked on Add Project feature.
3/9/2016	11:00 AM	2:00 PM	0	180	Construction	1	Worked on Add Project feature.
3/10/2016	11:00 AM	8:00 PM	30	510	Construction	1	Worked on Add Project feature.
3/11/2016	11:00 AM	5:00 PM	20	340	Construction	1	Worked on Add Project feature.
3/21/2016	2:00 PM	6:00 PM	0	240	Review	1	Verification using test cases.
3/22/2016	3:00 PM	10:00 PM	10	410	Postmortem	1	Calculated LOC and effort.

Acceptance Test Report

Use this worksheet to record tests that, if passed, tell you that your software is complete. List only tests that are directly linked to requirements (in other words, do not list tests for methods or components that you create over and above what is required to fulfill the specified interface definition -- these tests will appear in your individual unit test files). Write the expected results at planning time. Modify if necessary later. Don't write actual results in advance.

Scenario	Test	Expected Results	Actual Results
Story-3	Should get 1 fake project from data model	Get 1 project.	Passed
Story-1	Should add 1 fake project to the data model	Add 1 project.	Passed
Story-2	Should delete 1 fake project from the data model	Delete 1 project.	Passed
Story-3	Should not have any projects if data model is empty	There should be zero projects.	Passed
Story-1	Should reset the form to have default values(empty name and description)	The form should reset itself to default values.	Passed

Iteration 02

New Components	Estimated Method Count	Rel Size	LOCr	Reuseable?	Actual Method Count	LOCa	Reusable?
Router – Add Project	1	S	3	Yes	1	6	Yes
Controller – Add Project	4	M	68	Yes	5	50	Yes
Model – Add Project	1	VL	552	Yes	1	1	Yes
View – Add Project	1	L	97	Yes	1	64	Yes
app.js	2	M	34	Yes	2	23	Yes
style.css	1	VL	552	Yes	1	200	Yes
index.html	1	L	97	Yes	1	43	Yes
Router – Dashboard	1	S	3	Yes			
Controller – Dashboard	4	L	388	Yes			
View – Dashboard	1	L	97	Yes			
Router – TimeLog	1	S	3	Yes	1	3	Yes
Controller – TimeLog	4	M	68	Yes	8	88	Yes
View – TimeLog	1	M	17	Yes	1	101	Yes
Router – ChangeLog	1	S	3	Yes			
Controller – ChangeLog	4	M	68	Yes			
View – ChangeLog	1	M	17	Yes			
Router – Burndown	1	S	3	Yes			
Controller – Burndown	10	VL	5520	Yes			
View – Burndown	1	VL	552	Yes			
			0				
Totals			8142	8142		579	579

Component-Iteration Map

	Component	Iteration 1	Iteration 2	Iteration 3	Iteration 4	Iteration 5	Iteration 6	Iteration 7	Iteration 8	Iteration 9	Iteration 10
Base Components											
New Components	Router –	1									
	Controller –	5									
	Model – Add	1									
	View – Add	1									
	app.js	2									
	style.css	1									
	index.html	1									
	Router –			1							
	Controller –			4							
	View –			1							
	Router –		1								
	Controller –		4								
	View –		1								
	Router –				1						
	Controller –				2	1	1				
	View –				1						
	Router – Controller –				15	3	2				
	View –				1						
:Number of tasks		12	6	6	11	4	3	0	0	0	0
Effort:		514	257	257	471	171	129	0	0	0	0

WBS

Iteration Number	Planned Effort (minutes)	Cumulative Planned Effort	Planned Velocity	Cumulative Planned Velocity	Planned Completion Date	Actual Effort (minutes)	Cum Actual Effort	Earned Velocity	Cumulative Earned Velocity	Actual Completion Date
1	3600	3600	220	220		3/13/1909	3360	8/7/1900	220	
2	1800	5400	43	263			3360		440	
3	1800	7200	43	306			3360		440	
4	3600	10800	80	386			3360		440	
5	1500	12300	40	426			3360		440	
6	1200	13500	30	456			3360		440	
7		13500		456			3360		440	
8		13500		456			3360		440	
9		13500		456			3360		440	
10		13500		456			3360		440	

Calendar

Day #	Date	Available Minutes	Cumulative Minutes	Planned Velocity	Cumulative Planned Velocity	Actual Minutes	Cumulative Actual Minutes	Earned Velocity	Cumulative Earned Velocity
1	Due Date - 13		0		0		0		0
2	Due Date - 12		0		0		0		0
3	Due Date - 11		0		0		0		0
4	Due Date - 10		0		0		0		0
5	Due Date - 9		0		0		0		0
6	Due Date - 8		0		0		0		0
7	Due Date - 7		0		0		0		0
8	Due Date - 6		0		0		0		0
9	Due Date - 5		0		0		0		0
10	Due Date - 4		0		0		0		0
11	Due Date - 3		0		0		0		0
12	Due Date - 2		0		0		0		0
13	Due Date - 1		0		0		0		0

Change Log

Use this worksheet to record changes to made to artifacts after the project is underway. Record repairs to defects, as well as changes that result from requirements clarifications/changes.

Change Designation	Description
Requirements Change	Changes to requirements
Requirements Clarification	Clarifications to requirements
Product syntax	Syntax flaws in the deliverable product
Product logic	Logic flaws in the deliverable product
Product interface	Flaws in the interface of a component of the deliverable product
Product checking	Flaws with boundary/type checking within a component of the deliverable product
Test syntax	Syntax flaws in the test code
Test logic	Logic flaws in the test code
Test interface	Flaws in the interface of a component of the test code
Test checking	Flaws with boundary/type checking within a component of the test code
Bad Smell	Refactoring changes (please note the bad smell in the defect description)

Number	Date	Type	Where Injected	Inject Iteration	Where Removed	Remove Iteration	Fix Time	Fix Reference	Description
1	3/29/2016	Product logic	Construction	2	Construction	2	30		Resizable function not working as jquery load not asynchronously.
2									

Time Recording Log

Date	Start Time	Stop Time	Interrupt	Delta	Activity	Iteration	Comments
3/23/2016	10:00 AM	3:00 PM	10	290	Analysis		
3/24/2016	10:00 AM	1:00 PM	2	178	Architecture		
3/25/2016	10:00 AM	2:00 PM	5	235	Project planning		
3/28/2016	10:00 AM	1:00 PM	0	180	Interation planning		
3/29/2016	12:00 PM	3:00 PM	0	180	Construction		
3/30/2016	1:00 PM	2:00 PM	2	58	Construction		
3/31/2016	2:00 PM	4:00 PM	5	115	Construction		
4/1/2016	3:00 PM	7:00 PM	0	240	Review		
4/4/2016	4:00 PM	5:00 PM	10	50	Construction		
4/5/2016	5:00 PM	8:00 PM	0	180	Postmortem		

Acceptance Test Report

Use this worksheet to record tests that, if passed, tell you that your software is complete. List only tests that are directly linked to requirements (in other words, do not list tests for methods or components that you create over and above what is required to fulfill the specified interface definition -- these tests will appear in your individual unit test files). Write the expected results at planning time. Modify if necessary later. Don't write actual results in advance.

Scenario	Test	Expected Results	Actual Results
Story-1	Should add 1 Time Log for a fake project from data model	Add 1 project	Passed
Story-2	Should add 1 Time Log for a fake project from data model	Edit 1 project	Passed
Story-3	Should add 1 Time Log for a fake project from data model	Add 1 project	Passed
Story-4	Should add 1 Time Log for a fake project from data model	Store 1 project	Passed
Story-5	Should Open 1 Time Log for a fake project from data model	Opens the time log	Passed
Story-6	Should delete 1 fake time log from the data model	Delete the time log	Passed

Iteration 03

New Components	Estimated Method Count	Rel Size	LOCr	Reuseable?	Actual Method Count	LOCa	Reusable?
Router – Add Project	1	S	4	Yes	1	6	Yes
Controller – Add Project	4	M	68	Yes	5	50	Yes
Model – Add Project	1	VL	444	Yes	1	1	Yes
View – Add Project	1	L	86	Yes	1	64	Yes
app.js	2	M	34	Yes	2	23	Yes
style.css	1	VL	444	Yes	1	200	Yes
index.html	1	L	86	Yes	1	43	Yes
Router – Dashboard	1	S	4	Yes			
Controller – Dashboard	10	L	860	Yes			
View – Dashboard	1	L	86	Yes			
Router – TimeLog	1	S	4	Yes	1	3	Yes
Controller – TimeLog	10	M	170	Yes	10	120	Yes
View – TimeLog	1	L	86	Yes	1	140	Yes
Router – ChangeLog	1	S	4	Yes			
Controller – ChangeLog	4	M	68	Yes			
View – ChangeLog	1	M	17	Yes			
Router – ProjectSetup	1	S	4	Yes	1	6	Yes
Controller – ProjectSetup	6	VL	2664	Yes	4	79	Yes
View – ProjectSetup	1	VL	444	Yes	1	140	Yes
setupFlowChart.js	2	L	172	Yes	2	90	Yes
gojs.js	2	L	172	Yes	2	150	Yes
Totals			5921	5921		1115	1115

Component-Iteration Map

	Component	Iteration 1	Iteration 2	Iteration 3	Iteration 4	Iteration 5	Iteration 6	Iteration 7	Iteration 8	Iteration 9	Iteration 10
Base Components											
New Components	Router – Add Project	1									
	Controller – Add Project	5									
	Model – Add Project	1									
	View – Add Project	1									
	app.js	2									
	style.css	1									
	index.html	1									
	Router – Dashboard				1						
	Controller – Dashboard				5	3	2				
	View – Dashboard				1						
	Router – TimeLog		1								
	Controller – TimeLog		4	6							
	View – TimeLog		1								
	Router – ChangeLog					1					
	Controller – ChangeLog					1	3				
	View – ChangeLog					1					
	Router – ProjectSetup			1							
	Controller – ProjectSetup			4	2						
	View – ProjectSetup			1							
	setupFlowChart.js			2							
	gojs.js			2							
	Number of tasks:	12	6	16	9	6	5	0	0	0	0
	Effort:	400	200	533	300	200	167	0	0	0	0

WBS

Iteration Number	Planned Effort (minutes)	Cumulative Planned Effort	Planned Velocity	Cumulative Planned Velocity	Planned Completion Date	Actual Effort (minutes)	Cum Actual Effort	Earned Velocity	Cumulative Earned Velocity	Actual Completion Date
1	3600	3600	220	220	3/23/2016	3360	3360	220	220	
2	1800	5400	43	263		1706	5066	43	440	4/6/2016
3	1800	7200	33	296	4/19/2016	1696	6762	33	483	4/19/2016
4	3600	10800	80	376			6762		516	
5	1500	12300	40	416			6762		516	
6	1200	13500	30	446			6762		516	
7		13500		446			6762		516	
8		13500		446			6762		516	
9		13500		446			6762		516	
10		13500		446			6762		516	

Calendar

Day #	Date	Available Minutes	Cumulative Minutes	Planned Velocity	Cumulative Planned Velocity	Actual Minutes	Cumulative Actual Minutes	Earned Velocity	Cumulative Earned Velocity
1	4/6/2016	200	200		0		0		0
2	4/7/2016	200	400		0		0		0
3	4/8/2016	200	600		0		0		0
4	4/9/2016	0	600		0		0		0
5	4/10/2016	0	600		0		0		0
6	4/11/2016	200	800		0		0		0
7	4/12/2016	200	1000		0		0		0
8	4/13/2016	200	1200		0		0		0
9	4/14/2016	200	1400		0		0		0
10	4/15/2016	200	1600		0		0		0
11	4/16/2016	0	1600		0		0		0
12	4/17/2016	0	1600		0		0		0
13	4/18/2016	100	1700		0		0		0
14	4/19/2016	100	1800		0		0		0

Acceptance Test Report

Use this worksheet to record tests that, if passed, tell you that your software is complete. List only tests that are directly linked to requirements (in other words, do not list tests for methods or components that you create over and above what is required to fulfill the specified interface definition -- these tests will appear in your individual unit test files). Write the expected results at planning time. Modify if necessary later. Don't write actual results in advance.

Scenario	Test	Expected Results	Actual Results
Story-1	Should add 1 Time Log for a fake project from data model	Add 1 project	Passed
Story-2	Should add 1 Time Log for a fake project from data model	Edit 1 project	Passed
Story-3	Should add 1 Time Log for a fake project from data model	Add 1 project	Passed
Story-4	Should add 1 Time Log for a fake project from data model	Store 1 project	Passed
Story-5	Should Open 1 Time Log for a fake project from data model	Opens the time log	Passed
Story-6	Should delete 1 fake time log from the data model	Delete the time log	Passed

Time Recording Log

Date	Start Time	Stop Time	Interrupt	Delta	Activity	Iteration
4/6/2016	10:00 AM	3:00 PM	15	285	Analysis	3
4/7/2016	10:00 AM	1:00 PM	2	178	Architecture	3
4/8/2016	10:00 AM	2:00 PM	0	240	Project planning	3
4/11/2016	10:00 AM	1:00 PM	0	180	Interation planni	3
4/12/2016	12:00 PM	3:00 PM	10	170	Construction	3
4/13/2016	1:00 PM	2:00 PM	2	58	Construction	3
4/14/2016	2:00 PM	4:00 PM	5	115	Construction	3
4/15/2016	3:00 PM	7:00 PM	0	240	Review	3
4/18/2016	4:00 PM	5:00 PM	10	50	Construction	3
4/19/2016	5:00 PM	8:00 PM	0	180	Postmortem	3

Change Designation	Description
Requirements Change	Changes to requirements
Requirements Clarification	Clarifications to requirements
Product syntax	Syntax flaws in the deliverable product
Product logic	Logic flaws in the deliverable product
Product interface	Flaws in the interface of a component of the deliverable product
Product checking	Flaws with boundary/type checking within a component of the deliverable product
Test syntax	Syntax flaws in the test code
Test logic	Logic flaws in the test code
Test interface	Flaws in the interface of a component of the test code
Test checking	Flaws with boundary/type checking within a component of the test code
Bad Smell	Refactoring changes (please note the bad smell in the defect description)

Number	Date	Type	Where Injected	Inject Iteration	Where Removed	Remove Iteration	Fix Time	Fix Reference	Description
1	3/29/2016	Product logic	Construction	2	Construction	2	30		Resizable function not working as jquery load not asynchronously.
2	4/6/2016	Product logic	Construction	2	Construction	3	60		Changes to time log for different views
3	4/6/2016	Product logic	Analysis	2	Analysis	3	20		Removed burndown feature. Instead added project setup feature.
4									

Iteration 04

New Components	Estimated Method Cou	Rel Size	LOCr	Reuseable?	Actual Method Cou	LOCa	Reusable?
State – Add Project	1	S	5	Yes	1	10	Yes
Controller – Add Project	4	M	88	Yes	5	69	Yes
Model – Project	1	M	22	Yes	1	29	Yes
View – Add project	1	L	102	Yes	1	62	Yes
app.js	1	M	22	Yes	2	22	Yes
style.css	1	VL	475	Yes	1	368	Yes
index.html	1	L	102	Yes	1	115	Yes
State – Dashboard	1	S	5	Yes			
Controller – DashBoard	10	L	1020	Yes			
View – Dashboard	1	L	102	Yes			
State – TimeLog	1	S	5	Yes	1	10	Yes
Controller – Timelog	10	M	220	Yes	10	148	Yes
View – TimeLog	1	L	102	Yes	1	121	Yes
State – ChangeLog	1	S	5	Yes	1	10	Yes
Controller – ChangeLog	6	M	132	Yes	6	115	Yes
View – Changelog	1	L	102	Yes	1	197	Yes
Controller – ProjectSetup	6	M	132	Yes	4	140	Yes
View – ProjectSetup	1	L	102	Yes	1	121	Yes
setupFlowChart.js	2	L	204	Yes	2	90	Yes
go.js	2	L	204	Yes	2	150	Yes
State – Estimate	1	S	5	Yes			
Controller – Estimate	10	M	220	Yes			
View – Estimate	1	L	102	Yes			
State – Schedule	1	S	5	Yes			
Controller – Schedule	10	M	220	Yes			
View – Schedule	1	L	102	Yes			
Totals			3805	3805		1777	1777

Component-Iteration Map

	Component	Iteration 1	Iteration 2	Iteration 3	Iteration 4	Iteration 5	Iteration 6	Iteration 7	Iteration 8	Iteration 9	Iteration 10
Base Components											
New Components	State – Add Project	1									
	Project	5									
	Model – Project	1									
	View – Add project	1									
	app.js	2									
	style.css	1									
	index.html	1									
	State – Dashboard				1						
	DashBoard						5	5			
	View – Dashboard						1				
	State – TimeLog		1								
	Controller – Timelog		4	6							
	View – TimeLog		1								
	State – ChangeLog				1						
	ChangeLog				6						
	View – Changelog				1						
	ProjectSetup			1							
	ProjectSetup			6							
	View – ProjectSetup			1							
	setupFlowChart.js			2							
	go.js			2							
	State – Estimate				1						
	Controller – Estimate					10					
	View – Estimate					1					
	State – Schedule				1						
	Controller – Schedule						5	5			
	View – Schedule							1			
	Number of tasks:	12	6	18	11	11	11	11	0	0	0
	Effort:	5820	2910	8730	5335	5335	5335	5335	0	0	0

WBS

Iteration Number	Planned Effort (minutes)	Cumulative Planned Effort	Planned Velocity	Cumulative Planned Velocity	Planned Completion Date	Actual Effort (minutes)	Cum Actual Effort	Earned Velocity	Cumulative Earned Velocity	Actual Completion Date	
1	3600	3600	220	220	3/23/2016	3360	3360	220	220	3/23/2016	
2	1800	5400	43	263	4/5/2016	1706	5066	43	440	4/5/2016	
3	1800	7200	33	296	4/19/2016	1696	6762	33	483	4/19/2016	
4	800	8000	11	307	11/15/2016	821	7583	11	516	11/15/2016	**Current Iteration**
5	800	8800	11	318	11/30/2016		7583		527		
6	1800	10600	11	329			7583		527		
7	1800	12400	11	340			7583		527		

Calendar

Day #	Date	Available Minutes	Cumulative Minutes	Planned Velocity	Cumulative Planned Velocity	Actual Minutes	Cumulative Actual Minutes	Earned Velocity	Cumulative Earned Velocity	
1	3/9/2016		0		0		0		0	begin 1st Iteration
2	3/10/2016		0		0		0		0	
3	3/11/2016		0		0		0		0	
4	3/12/2016		0		0		0		0	
5	3/13/2016		0		0		0		0	
6	3/14/2016		0		0		0		0	
7	3/15/2016		0		0		0		0	
8	3/16/2016		0		0		0		0	
9	3/17/2016		0		0		0		0	
10	3/18/2016		0		0		0		0	
11	3/19/2016		0		0		0		0	
12	3/20/2016		0		0		0		0	
13	3/21/2016		0		0		0		0	
14	3/22/2016		0		0		0		0	end 1st Iteration
15	3/23/2016		0		0		0		0	begin 2nd Iteration
16	3/24/2016		0		0		0		0	
17	3/25/2016		0		0		0		0	
18	3/26/2016		0		0		0		0	
19	3/27/2016		0		0		0		0	
20	3/28/2016		0		0		0		0	
21	3/29/2016		0		0		0		0	
22	3/30/2016		0		0		0		0	
23	3/31/2016		0		0		0		0	
24	4/1/2016		0		0		0		0	
25	4/2/2016		0		0		0		0	
26	4/3/2016		0		0		0		0	
27	4/4/2016		0		0		0		0	
28	4/5/2016		0		0		0		0	end 2nd Iteration
29	4/6/2016	200	200	3	3	200	200	3	3	begin 3rd Iteration
30	4/7/2016	200	400	3	6	200	400	3	6	
31	4/8/2016	200	600	3	9	200	600	3	9	
32	4/9/2016	0	600	0	9	0	600	0	9	
33	4/10/2016	0	600	0	9	0	600	0	9	
34	4/11/2016	200	800	3	12	200	800	3	12	
35	4/12/2016	200	1000	3	15	200	1000	3	15	
36	4/13/2016	200	1200	3	18	200	1200	3	18	
37	4/14/2016	200	1400	3	21	200	1400	3	21	
38	4/15/2016	200	1600	3	24	200	1600	3	24	
39	4/16/2016	0	1600	0	24	0	1600	0	24	
40	4/17/2016	0	1600	0	24	0	1600	0	24	
41	4/18/2016	100	1700	5	29	100	1700	5	29	
42	4/19/2016	100	1800	4	33	100	1800	4	33	end 3rd Iteration
43	11/2/2016	100	1900	2	35	60	1860	0	33	begin 4th iteration
44	11/3/2016	100	2000	2	37	60	1920	0	33	
45	11/4/2016	100	2100	2	39	120	2040	0	33	
46	11/5/2016	0	2100	0	39	0	2040	0	33	
47	11/6/2016	0	2100	0	39	0	2040	0	33	
48	11/7/2016	0	2100	0	39	0	2040	0	33	
49	11/8/2016	0	2100	0	39	0	2040	0	33	
50	11/9/2016	100	2200	1	40	120	2160	2	35	
51	11/10/2016	100	2300	1	41	110	2270	2	37	
52	11/11/2016	100	2400	1	42	118	2388	2	39	

Time Recording Log

Date	Start Time	Stop Time	Interrupt	Delta	Activity	Iteration
11/2/2016	10:00 AM	11:00 AM	0	60	Analysis	4
11/3/2016	10:00 AM	11:00 AM	0	60	Architecture	4
11/4/2016	10:00 AM	11:00 AM	0	60	Project planning	4
11/4/2016	11:00 AM	12:00 PM	0	60	Interation plann	4
11/9/2016	11:00 AM	1:00 PM	0	120	Construction	4
11/10/2016	11:00 AM	1:00 PM	10	110	Construction	4
11/11/2016	11:00 AM	1:00 PM	2	118	Construction	4
11/14/2016	11:00 AM	1:00 PM	2	118	Review	4
11/15/2016	11:00 AM	1:00 PM	5	115	Postmortem	4

Change Log

Use this worksheet to record changes to made to artifacts after the project is underway. Record repairs to defects, as well as changes that result from requirements clarifications/changes.

Change Designation	Description
Requirements Change	Changes to requirements
Requirements	Clarifications to requirements
Product syntax	Syntax flaws in the deliverable product
Product logic	Logic flaws in the deliverable product
Product interface	Flaws in the interface of a component of the deliverable product
Product checking	Flaws with boundary/type checking within a component of the deliverable product
Test syntax	Syntax flaws in the test code
Test logic	Logic flaws in the test code
Test interface	Flaws in the interface of a component of the test code
Test checking	Flaws with boundary/type checking within a component of the test code
Bad Smell	Refactoring changes (please note the bad smell in the defect description)

Number	Date	Type	Where Injected	nject Iteratio	Where Remove	emove Iteratio	Fix Time	Fix Reference	Description
1	11/9/2016	Requirements Change	Analysis	4	Construction	4	10		Changed the Navigation to add new features of Planning i.e. Estimation and scheduling.
2									

Iteration 05

	Estimated				Actual		
New Components	Method Cou	Rel Size	LOCr	Reuseable?	Method Cou	LOCa	Reusable?
State – Add Project	1	S	6	Yes	1	10	Yes
Controller – Add Project	4	M	96	Yes	5	69	Yes
Model – Project	1	M	24	Yes	1	29	Yes
View – Add project	1	L	106	Yes	1	62	Yes
app.js	1	M	24	Yes	2	22	Yes
style.css	1	VL	479	Yes	1	368	Yes
index.html	1	L	106	Yes	1	115	Yes
State – Dashboard	1	S	6	Yes			
Controller – DashBoard	10	L	1060	Yes			
View – Dashboard	1	L	106	Yes			
State – TimeLog	1	S	6	Yes	1	10	Yes
Controller – Timelog	10	M	240	Yes	10	148	Yes
View – TimeLog	1	L	106	Yes	1	121	Yes
State – ChangeLog	1	S	6	Yes	1	10	Yes
Controller – ChangeLog	6	M	144	Yes	6	115	Yes
View – Changelog	1	L	106	Yes	1	197	Yes
Controller – ProjectSetup	6	M	144	Yes	4	140	Yes
View – ProjectSetup	1	L	106	Yes	1	121	Yes
setupFlowChart.js	2	L	212	Yes	2	90	Yes
go.js	2	L	212	Yes	2	150	Yes
State – Estimate	1	S	6	Yes	1	10	Yes
Controller – Estimate	10	M	240	Yes	6	151	Yes
View – Estimate	1	VL	479	Yes	1	655	Yes
State – Schedule	1	S	6	Yes			
Controller – Schedule	10	M	240	Yes			
View – Schedule	1	L	106	Yes			
Totals			4372	4372		2593	2593

Component-Iteration Map

	Component	Iteration 1	Iteration 2	Iteration 3	Iteration 4	Iteration 5	Iteration 6	Iteration 7	Iteration 8	Iteration 9	Iteration 10
Base Components											
New Components	State – Add Project	1									
	Project	5									
	Model – Project	1									
	View – Add project	1									
	app.js	2									
	style.css	1									
	index.html	1									
	State – Dashboard				1						
	DashBoard						5	5			
	View – Dashboard						1				
	State – TimeLog		1								
	Controller – Timelog		4	6							
	View – TimeLog		1								
	State – ChangeLog				1						
	ChangeLog				6						
	View – Changelog				1						
	ProjectSetup			1							
	ProjectSetup			6							
	View – ProjectSetup			1							
	setupFlowChart.js			2							
	go.js			2							
	State – Estimate				1						
	Controller – Estimate					10					
	View – Estimate					1					
	State – Schedule				1						
	Controller – Schedule						5	5			
	View – Schedule							1			
	Number of tasks:	12	6	18	11	11	11	11	0	0	0
	Effort:	3591	1796	5387	3292	3292	3292	3292	0	0	0

WBS

Iteration Number	Planned Effort (minutes)	Cumulative Planned Effort	Planned Velocity	Cumulative Planned Velocity	Planned Completion Date	Actual Effort (minutes)	Cum Actual Effort	Earned Velocity	Cumulative Earned Velocity	Actual Completion Date	
1	3600	3600	220	220	3/23/2016	3360	3360	220	220	3/23/2016	
2	1800	5400	43	263	4/5/2016	1706	5066	43	440	4/5/2016	
3	1800	7200	33	296	4/19/2016	1696	6762	33	483	4/19/2016	
4	800	8000	11	307	11/15/2016	821	7583	11	516	11/15/2016	
5	800	8800	11	318	11/29/2016	760	8343	8	527	11/29/2016	Current Iteration
6	1800	10600	11	329			8343		535		
7	1800	12400	11	340			8343		535		

Calendar

Day #	Date	Available Minutes	Cumulative Minutes	Planned Velocity	Cumulative Planned Velocity	Actual Minutes	Cumulative Actual Minutes	Earned Velocity	Cumulative Earned Velocity	
1	3/9/2016		0		0		0		0	begin 1st Iteration
2	3/10/2016		0		0		0		0	
3	3/11/2016		0		0		0		0	
4	3/12/2016		0		0		0		0	
5	3/13/2016		0		0		0		0	
6	3/14/2016		0		0		0		0	
7	3/15/2016		0		0		0		0	
8	3/16/2016		0		0		0		0	
9	3/17/2016		0		0		0		0	
10	3/18/2016		0		0		0		0	
11	3/19/2016		0		0		0		0	
12	3/20/2016		0		0		0		0	
13	3/21/2016		0		0		0		0	
14	3/22/2016		0		0		0		0	end 1st Iteration
15	3/23/2016		0		0		0		0	begin 2nd Iteration
16	3/24/2016		0		0		0		0	
17	3/25/2016		0		0		0		0	
18	3/26/2016		0		0		0		0	
19	3/27/2016		0		0		0		0	
20	3/28/2016		0		0		0		0	
21	3/29/2016		0		0		0		0	
22	3/30/2016		0		0		0		0	
23	3/31/2016		0		0		0		0	
24	4/1/2016		0		0		0		0	
25	4/2/2016		0		0		0		0	
26	4/3/2016		0		0		0		0	
27	4/4/2016		0		0		0		0	
28	4/5/2016		0		0		0		0	end 2nd Iteration
29	4/6/2016	200	200	3	3	200	200	3	3	begin 3rd Iteration
30	4/7/2016	200	400	3	6	200	400	3	6	
31	4/8/2016	200	600	3	9	200	600	3	9	
32	4/9/2016	0	600	0	9	0	600	0	9	
33	4/10/2016	0	600	0	9	0	600	0	9	
34	4/11/2016	200	800	3	12	200	800	3	12	
35	4/12/2016	200	1000	3	15	200	1000	3	15	
36	4/13/2016	200	1200	3	18	200	1200	3	18	
37	4/14/2016	200	1400	3	21	200	1400	3	21	
38	4/15/2016	200	1600	3	24	200	1600	3	24	
39	4/16/2016	0	1600	0	24	0	1600	0	24	
40	4/17/2016	0	1600	0	24	0	1600	0	24	
41	4/18/2016	100	1700	5	29	100	1700	5	29	
42	4/19/2016	100	1800	4	33	100	1800	4	33	end 3rd Iteration
43	11/2/2016	100	1900	2	35	60	1860	0	33	begin 4th iteration
44	11/3/2016	100	2000	2	37	60	1920	0	33	
45	11/4/2016	100	2100	2	39	120	2040	0	33	
46	11/5/2016	0	2100	0	39	0	2040	0	33	
47	11/6/2016	0	2100	0	39	0	2040	0	33	
48	11/7/2016	0	2100	0	39	0	2040	0	33	
49	11/8/2016	0	2100	0	39	0	2040	0	33	
50	11/9/2016	100	2200	1	40	120	2160	2	35	
51	11/10/2016	100	2300	1	41	110	2270	2	37	
52	11/11/2016	100	2400	1	42	118	2388	2	39	
53	11/12/2016	0	2400	0	42	0	2388	0	39	
54	11/13/2016	0	2400	0	42	0	2388	0	39	
55	11/14/2016	100	2500	1	43	118	2506	4	43	
56	11/15/2016	100	2600	1	44	115	2621	1	44	End 4th Iteration
57	11/16/2016	100	2700	2	46	100	2721	0	44	begin 5th iteration
58	11/17/2016	100	2800	2	48	100	2821	0	44	
59	11/18/2016	0	2800	0	48	0	2821	0	44	
60	11/19/2016	100	2900	2	50	100	2921	1	45	
61	11/20/2016	0	2900	0	50	0	2921	0	45	
62	11/21/2016	0	2900	0	50	100	3021	2	47	
63	11/22/2016	0	2900	0	50	120	3141	2	49	
64	11/23/2016	100	3000	1	51	0	3141	0	49	
65	11/24/2016	100	3100	1	52	0	3141	0	49	
66	11/25/2016	100	3200	1	53	0	3141	0	49	
67	11/26/2016	0	3200	0	53	0	3141	0	49	
68	11/27/2016	0	3200	0	53	0	3141	0	49	
69	11/28/2016	100	3300	1	54	240	3381	3	52	
70	11/29/2016	100	3400	1	55	0	3381	0	52	
71	11/30/2016		3400		55		3381	0	52	end 5th iteration

Change Designation	Description
Requirements Change	Changes to requirements
Requirements Clarification	Clarifications to requirements
Product syntax	Syntax flaws in the deliverable product
Product logic	Logic flaws in the deliverable product
Product interface	Flaws in the interface of a component of the deliverable product
Product checking	Flaws with boundary/type checking within a component of the deliverable product
Test syntax	Syntax flaws in the test code
Test logic	Logic flaws in the test code
Test interface	Flaws in the interface of a component of the test code
Test checking	Flaws with boundary/type checking within a component of the test code
Bad Smell	Refactoring changes (please note the bad smell in the defect description)

Number	Date	Type	Where Injected	Inject Iteration	Where Removed	Remove Iteration	Fix Time	Fix Reference	Description
1	11/9/2016	Requirements Change	Analysis	4	Construction	5	10		Changed the changeLog feature as per discussion with singl iteration in mind.
2									

Time Recording Log

Date	Start Time	Stop Time	Interrupt	Delta	Activity	Iteration
11/15/2016	10:00 AM	11:40 AM	0	100	Analysis	5
11/16/2016	10:00 AM	11:40 AM	0	100	Architecture	5
11/18/2016	10:00 AM	11:40 AM	0	100	Project planning	5
11/21/2016	11:00 AM	12:00 PM	0	60	Interation planni	5
11/21/2016	12:00 PM	12:40 PM	0	40	Construction	5
11/22/2016	11:00 AM	1:00 PM	0	120	Construction	5
11/28/2016	11:00 AM	1:00 PM	0	120	Construction	5
11/28/2016	1:00 PM	2:00 PM	0	60	Review	5
11/28/2016	2:00 PM	3:00 PM	0	60	Postmortem	5

Iteration 06

User Stories

Story Set	1		
Objective:	To explore the functionality of Planning - Scheduling component.		
Story	As a …	I want to …	So I can …
Story -1	user (anonymous/logged in)	have the ability to add planned components in the project iteration.	Get information to compute size estimate
Story -2	user (anonymous/logged in)	have the ability to add actual components in the project iteration.	Get information to compute size estimate
Story -3	user (anonymous/logged in)	have the ability to edit or delete components in the project iteration.	update the plan
Story -4	user (anonymous/logged in)	change the iteration	view or update the estimate for each iteration
Story -5	software component in PCSEDesk	validate and verify the components	store them in the database
Story -6	software component in PCSEDesk	load the existing data for all the components	restore the last saved state in the UI.

Story Set	2		
Objective:	To explore the functionality of Planning – estimation component.		
Story	As a …	I want to …	So I can …
Story -1	user (anonymous/logged in)	have the ability to add planned iteration in the project.	store artifacts for an iteration related to scheduling of project
Story -2	user (anonymous/logged in)	have the ability to divide the tasks among the iteration.	compute the iteration plan
Story -3	user (anonymous/logged in)	have the ability to edit or delete task in the project iteration map.	update the plan
Story -4	user (anonymous/logged in)	navigate the iterations	view the burndown for each iteration
Story -5	software component in PCSEDesk	use the available information from the project	generate the burndown chart

Story Set	3		
Objective:	To explore the functionality of Dashboard component.		
Story	**As a …**	**I want to …**	**So I can …**
Story -1	user (anonymous/logged in)	view the burndown chart	track the project progress
Story -2	software component in PCSEDesk	use the information about the project	generate the burndown chart
Story -3	software component in PCSEDesk	use the chart service	use it for generating the chart
Story -4	software component in PCSEDesk	store the information mentioned above in the system	Use it along with business logic and restore the information for their respective views.
Story -5	user (anonymous/logged in)	view all the time logs for each iteration completed so far/in running state	keep a track of time logs from past and view the painful areas of concern.
Story -6	software component in PCSEDesk	validate the information	store them and use them for business logic and making changes to their respective views.

Architecture

Use this worksheet to describe components required to implement the specifications. List new components and base comp include extraneous components.

Component Name:	estimate.js
Design Approach:	Functional
Parent Component:	
Component Type:	Logic
Collaborators:	estimate.controller.js, estimate.html, estimate.css
Operations:	1 method

Component Name:	estimate.controller.js
Design Approach:	Functional
Parent Component:	
Component Type:	Logic
Collaborators:	estimate.js, estimate.html
Operations:	10 methods

Component Name:	estimate.css
Design Approach:	Functional
Parent Component:	
Component Type:	Logic
Collaborators:	
Operations:	1 Methods

Component Name:	estimate.html
Design Approach:	Functional
Parent Component:	
Component Type:	I/O
Collaborators:	estimate.controller.js, estimate.css
Operations:	1 method

Component Name:	schedule.js
Design Approach:	Functional
Parent Component:	
Component Type:	Logic
Collaborators:	schedule.controller.js, schedule.html, schedule.css
Operations:	1 method

Component Name:	schedule.controller.js
Design Approach:	Functional
Parent Component:	
Component Type:	Logic
Collaborators:	schedule.js, schedule.html
Operations:	15 methods

Component Name:	schedule.css
Design Approach:	Functional
Parent Component:	
Component Type:	Logic
Collaborators:	
Operations:	1 Methods

Component Name:	schedule.html
Design Approach:	Functional
Parent Component:	
Component Type:	I/O
Collaborators:	schedule.controller.js, schedule.css
Operations:	1 Methods

	Estimated				Actual		
New Components	ethod Cou	Rel Size	LOCr	Reuseable?	Method Cou	LOCa	Reusable?
State – Add Project	1	S	6	Yes	1	10	Yes
Controller – Add Project	4	M	96	Yes	5	69	Yes
Model – Project	1	M	24	Yes	1	29	Yes
View – Add project	1	L	106	Yes	1	62	Yes
app.js	1	M	24	Yes	2	22	Yes
style.css	1	VL	479	Yes	1	368	Yes
index.html	1	L	106	Yes	1	115	Yes
State – Dashboard	1	S	6	Yes			
Controller – DashBoard	10	L	1060	Yes			
View – Dashboard	1	L	106	Yes			
State – TimeLog	1	S	6	Yes	1	10	Yes
Controller – Timelog	10	M	240	Yes	10	148	Yes
View – TimeLog	1	L	106	Yes	1	121	Yes
State – ChangeLog	1	S	6	Yes	1	10	Yes
Controller – ChangeLog	6	M	144	Yes	6	115	Yes
View – Changelog	1	L	106	Yes	1	197	Yes
Controller – ProjectSetup	6	M	144	Yes	4	140	Yes
View – ProjectSetup	1	L	106	Yes	1	121	Yes
setupFlowChart.js	2	L	212	Yes	2	90	Yes
go.js	2	L	212	Yes	2	150	Yes
State – Estimate	1	S	6	Yes	1	10	Yes
Controller – Estimate	10	M	240	Yes	10	372	Yes
View – Estimate	1	VL	479	Yes	1	505	Yes
State – Schedule	1	S	6	Yes			
Controller – Schedule	10	M	240	Yes			
View – Schedule	1	L	106	Yes			
Totals			4372	4372		2664	2664

WBS

Iteration Number	Planned Effort (minutes)	Cumulative Planned Effort	Planned Velocity	Cumulative Planned Velocity	Planned Completion Date	Actual Effort (minutes)	Cum Actual Effort	Earned Velocity	Cumulative Earned Velocity	Actual Completion Date	
1	3600	3600	12	12	3/23/2016	3360	3360	220	220	3/23/2016	
2	1800	5400	6	18	4/5/2016	1706	5066	43	440	4/5/2016	
3	1800	7200	18	36	4/19/2016	1696	6762	33	483	4/19/2016	
4	800	8000	11	47	11/15/2016	821	7583	11	516	11/15/2016	
5	800	8800	11	58	11/29/2016	760	8343	8	527	11/29/2016	
6	2000	10800	11	69	2/2/2017	1900	10243	15	535	2/2/2017	Curre
7	2000	12800	17	86	2/9/2017		10243		550	2/9/2017	
8	2000	14800	11	97	2/16/2017		10243		550	2/16/2017	

Calendar

Day #	Date	Available Minutes	Cumulative Minutes	Planned Velocity	Cumulative Planned Velocity	Actual Minutes	Cumulative Actual Minutes	Earned Velocity	Cumulative Earned Velocity	
1	1/26/2017		0		0		0		0	
2	1/27/2017		0		0		0		0	
3	1/28/2017		0		0		0		0	
4	1/29/2017		0		0		0		0	
5	1/30/2017		0		0		0		0	
6	1/26/2017	400	400	2	2	200	200	2	2	begin 6th Iteration
7	1/27/2017	400	800	2	4	100	300	2	4	
8	1/28/2017	0	800	2	6	0	300	1	5	
9	1/29/2017	0	800	2	8	0	300	2	7	
10	1/30/2017	400	1200	2	10	520	820	1	8	
11	1/31/2017	400	1600	2	12	420	1240	2	10	
12	2/1/2017	400	2000	1	13	420	1660	3	13	
13	2/2/2017	400	2400	0	13	240	1900	2	15	End 6th Iteration
14	2/3/2017	400	2800	4	17		1900		15	begin 7nd Iteration
15	2/4/2017	0	2800	2	19		1900		15	
16	2/5/2017	0	2800	2	21		1900		15	
17	2/6/2017	400	3200	4	25		1900		15	
18	2/7/2017	400	3600	1	26		1900		15	
19	2/8/2017	400	4000	2	28		1900		15	
20	2/9/2017	400	4400	2	30		1900		15	end 7nd Iteration
21	2/10/2017	400	4800	1	31		1900		15	begin 8nd Iteration
22	2/11/2017	0	4800	1	32		1900		15	
23	2/12/2017	0	4800	1	33		1900		15	
24	2/13/2017	400	5200	1	34		1900		15	
25	2/14/2017	400	5600	1	35		1900		15	
26	2/15/2017	400	6000	1	36		1900		15	
27	2/16/2017	400	6400	1	37		1900		15	end 8nd Iteration
Number of tasks:		12	6	18	11	11	11	17	11	0	0
Effort:		2962	1481	4443	2715	2715	2715	4196	2715	0	0

Time Recording Log

Date	Start Time	Stop Time	Interrupt	Delta	Activity	Iteration
1/26/2017	10:00 AM	11:40 AM	0	100	Analysis	6
1/26/2017	12:00 AM	1:00 AM	0	60	Architecture	6
1/26/2017	2:00 AM	2:40 AM	0	40	Project planning	6
1/27/2017	10:00 AM	11:40 AM	0	100	Interation plann	6
1/30/2017	10:00 AM	6:40 PM	0	520	Construction	6
1/31/2017	10:00 AM	5:00 PM	0	420	Construction	6
2/1/2017	10:00 AM	5:00 PM	0	420	Review	6
2/2/2017	10:00 AM	2:00 PM	0	240	Postmortem	6

Change Log

Use this worksheet to record changes to made to artifacts after the project is underway. Record repairs to defects, as well as changes that result from requirements clarifications/changes.

Change Designation	Description
Requirements Change	Changes to requirements
Requirements	Clarifications to requirements
Product syntax	Syntax flaws in the deliverable product
Product logic	Logic flaws in the deliverable product
Product interface	Flaws in the interface of a component of the deliverable product
Product checking	Flaws with boundary/type checking within a component of the deliverable product
Test syntax	Syntax flaws in the test code
Test logic	Logic flaws in the test code
Test interface	Flaws in the interface of a component of the test code
Test checking	Flaws with boundary/type checking within a component of the test code
Bad Smell	Refactoring changes (please note the bad smell in the defect description)

Number	Date	Type	Where Injected	nject Iteratio	Where Remove	emove Iteratio	Fix Time	Fix Reference	Description
1	1/26/2017	Requirements Change	Analysis	4	Construction	6	50		Added estimation and scheduling under plan for UI

Iteration 07

New Components	Estimated Method Cou	Rel Size	LOCr	Reuseable?	Actual Method Cou	LOCa	Reusable?
State – Add Project	1	S	6	Yes	1	10	Yes
Controller – Add Project	4	M	96	Yes	5	69	Yes
Model – Project	1	M	24	Yes	1	29	Yes
View – Add project	1	L	106	Yes	1	62	Yes
app.js	1	M	24	Yes	2	22	Yes
style.css	1	VL	479	Yes	1	368	Yes
index.html	1	L	106	Yes	1	115	Yes
State – Dashboard	1	S	6	Yes			
Controller – DashBoard	10	L	1060	Yes			
View – Dashboard	1	L	106	Yes			
State – TimeLog	1	S	6	Yes	1	10	Yes
Controller – Timelog	10	M	240	Yes	10	148	Yes
View – TimeLog	1	L	106	Yes	1	121	Yes
State – ChangeLog	1	S	6	Yes	1	10	Yes
Controller – ChangeLog	6	M	144	Yes	6	115	Yes
View – Changelog	1	L	106	Yes	1	197	Yes
Controller – ProjectSetup	6	M	144	Yes	4	140	Yes
View – ProjectSetup	1	L	106	Yes	1	121	Yes
setupFlowChart.js	2	L	212	Yes	2	90	Yes
go.js	2	L	212	Yes	2	150	Yes
State – Estimate	1	S	6	Yes	1	10	Yes
Controller – Estimate	10	M	240	Yes	10	372	Yes
View – Estimate	1	VL	479	Yes	1	505	Yes
State – Schedule	1	S	6	Yes	1	10	Yes
Controller – Schedule	10	M	240	Yes	15	512	Yes
View – Schedule	1	L	106	Yes	1	200	Yes
Totals			4372	4372		3386	3386

Component-Iteration Map

	Component	Iteration 1	Iteration 2	Iteration 3	Iteration 4	Iteration 5	Iteration 6	Iteration 7	Iteration 8	Iteration 9	Iteration 10
Base Components											
New Components	State – Add Project	1									
	Project	5									
	Model – Project	1									
	View – Add project	1									
	app.js	2									
	style.css	1									
	index.html	1									
	State – Dashboard				1						
	DashBoard								10		
	View – Dashboard								1		
	State – TimeLog		1								
	Controller – Timelog		4	6							
	View – TimeLog		1								
	State – ChangeLog				1						
	ChangeLog				6						
	View – Changelog				1						
	ProjectSetup			1							
	ProjectSetup			6							
	View – ProjectSetup			1							
	setupFlowChart.js			2							
	go.js			2							
	State – Estimate				1						
	Controller – Estimate					10	10				
	View – Estimate					1	1				
	State – Schedule				1						
	Controller – Schedule							16			
	View – Schedule							1			
	Number of tasks:	12	6	18	11	11	11	17	11	0	0
	Effort:	2962	1481	4443	2715	2715	2715	4196	2715	0	0

WBS

Iteration Number	Planned Effort (minutes)	Cumulative Planned Effort	Planned Velocity	Cumulative Planned Velocity	Planned Completion Date	Actual Effort (minutes)	Cum Actual Effort	Earned Velocity	Cumulative Earned Velocity	Actual Completion Date	
1	3600	3600	12	12	3/23/2016	3360	3360	220	220	3/23/2016	
2	1800	5400	6	18	4/5/2016	1706	5066	43	440	4/5/2016	
3	1800	7200	18	36	4/19/2016	1696	6762	33	483	4/19/2016	
4	800	8000	11	47	11/15/2016	821	7583	11	516	11/15/2016	
5	800	8800	11	58	11/29/2016	760	8343	8	527	11/29/2016	
6	2000	10800	11	69	2/2/2017	1900	10243	15	535	2/2/2017	Cur
7	2000	12800	17	86	2/9/2017	2100	12343	15	550	2/9/2017	
8	2000	14800	11	97	2/16/2017		12343		565	2/16/2017	

Calendar

Day #	Date	Available Minutes	Cumulative Minutes	Planned Velocity	Cumulative Planned Velocity	Actual Minutes	Cumulative Actual Minutes	Earned Velocity	Cumulative Earned Velocity	
1	1/26/2017		0		0		0		0	
2	1/27/2017		0		0		0		0	
3	1/28/2017		0		0		0		0	
4	1/29/2017		0		0		0		0	
5	1/30/2017		0		0		0		0	
6	1/26/2017	400	400	2	2	200	200	2	2	begin 6th Iteratior
7	1/27/2017	400	800	2	4	100	300	2	4	
8	1/28/2017	0	800	2	6	0	300	1	5	
9	1/29/2017	0	800	2	8	0	300	2	7	
10	1/30/2017	400	1200	2	10	520	820	1	8	
11	1/31/2017	400	1600	2	12	420	1240	2	10	
12	2/1/2017	400	2000	1	13	420	1660	3	13	
13	2/2/2017	400	2400	0	13	240	1900	2	15	End 6th Iteration
14	2/3/2017	400	2800	4	17	200	2100	2	17	begin 7nd Iteratio
15	2/4/2017	0	2800	2	19	0	2100	2	19	
16	2/5/2017	0	2800	2	21	0	2100	1	20	
17	2/6/2017	400	3200	4	25	460	2560	2	22	
18	2/7/2017	400	3600	1	26	520	3080	1	23	
19	2/8/2017	400	4000	2	28	420	3500	2	25	
20	2/9/2017	400	4400	2	30	660	4160	5	30	end 7nd Iteration
21	2/10/2017	400	4800	1	31		4160		30	begin 8rd Iteratior
22	2/11/2017	0	4800	1	32		4160		30	
23	2/12/2017	0	4800	1	33		4160		30	
24	2/13/2017	400	5200	1	34		4160		30	
25	2/14/2017	400	5600	1	35		4160		30	
26	2/15/2017	400	6000	1	36		4160		30	
27	2/16/2017	400	6400	1	37		4160		30	end 8rd Iteration

Time Recording Log

Date	Start Time	Stop Time	Interrupt	Delta	Activity	Iteration
2/3/0107	10:00 AM	11:40 AM	0	100	Analysis	7
2/3/0107	12:00 AM	1:00 AM	0	60	Architecture	7
2/3/0107	2:00 AM	2:40 AM	0	40	Project planning	7
2/6/0107	10:00 AM	5:40 PM	0	460	Interation plann	7
2/7/0107	10:00 AM	6:40 PM	0	520	Construction	7
2/8/0107	10:00 AM	5:00 PM	0	420	Construction	7
2/9/0107	10:00 AM	5:00 PM	0	420	Review	7
2/9/0107	10:00 AM	2:00 PM	0	240	Postmortem	7

Change Log

Use this worksheet to record changes to made to artifacts after the project is underway. Record repairs to defects, as well as changes that result from requirements clarifications/changes.

Change Designation	Description
Requirements Change	Changes to requirements
Requirements	Clarifications to requirements
Product syntax	Syntax flaws in the deliverable product
Product logic	Logic flaws in the deliverable product
Product interface	Flaws in the interface of a component of the deliverable product
Product checking	Flaws with boundary/type checking within a component of the deliverable product
Test syntax	Syntax flaws in the test code
Test logic	Logic flaws in the test code
Test interface	Flaws in the interface of a component of the test code
Test checking	Flaws with boundary/type checking within a component of the test code
Bad Smell	Refactoring changes (please note the bad smell in the defect description)

Number	Date	Type	Where Injected	nject Iteratio	Where Remove	emove Iteratio	Fix Time	Fix Reference	Description
1	1/26/2017	Requirements Change	Analysis	4	Construction	6	140		Add iteration from schedule ad change project setup

Iteration 08

New Components	Estimated Method Cou	Rel Size	LOCr	Reuseable?	Actual Method Cou	LOCa	Reusable?
State – Add Project	1	S	6	Yes	1	10	Yes
Controller – Add Project	4	M	96	Yes	5	69	Yes
Model – Project	1	M	24	Yes	1	29	Yes
View – Add project	1	L	106	Yes	1	62	Yes
app.js	1	M	24	Yes	2	22	Yes
style.css	1	VL	479	Yes	1	368	Yes
index.html	1	L	106	Yes	1	115	Yes
State – Dashboard	1	S	6	Yes	1	10	Yes
Controller – DashBoard	10	L	1060	Yes	3	240	Yes
View – Dashboard	1	L	106	Yes	1	27	Yes
State – TimeLog	1	S	6	Yes	1	10	Yes
Controller – Timelog	10	M	240	Yes	10	148	Yes
View – TimeLog	1	L	106	Yes	1	121	Yes
State – ChangeLog	1	S	6	Yes	1	10	Yes
Controller – ChangeLog	6	M	144	Yes	6	115	Yes
View – Changelog	1	L	106	Yes	1	197	Yes
Controller – ProjectSetup	6	M	144	Yes	4	140	Yes
View – ProjectSetup	1	L	106	Yes	1	121	Yes
setupFlowChart.js	2	L	212	Yes	2	90	Yes
go.js	2	L	212	Yes	2	150	Yes
State – Estimate	1	S	6	Yes	1	10	Yes
Controller – Estimate	10	M	240	Yes	10	372	Yes
View – Estimate	1	VL	479	Yes	1	505	Yes
State – Schedule	1	S	6	Yes	1	10	Yes
Controller – Schedule	10	M	240	Yes	15	512	Yes
View – Schedule	1	L	106	Yes	1	200	Yes
Totals			4372	4372		3663	3663

Component-Iteration Map

	Component	Iteration 1	Iteration 2	Iteration 3	Iteration 4	Iteration 5	Iteration 6	Iteration 7	Iteration 8	Iteration 9	Iteration 10
Base Components											
New Components	State – Add Project	1									
	Project	5									
	Model – Project	1									
	View – Add project	1									
	app.js	2									
	style.css	1									
	index.html	1									
	State – Dashboard				1						
	DashBoard								10		
	View – Dashboard								1		
	State – TimeLog		1								
	Controller – Timelog		4	6							
	View – TimeLog		1								
	State – ChangeLog				1						
	ChangeLog				6						
	View – Changelog				1						
	ProjectSetup			1							
	ProjectSetup			6							
	View – ProjectSetup			1							
	setupFlowChart.js			2							
	go.js			2							
	State – Estimate				1						
	Controller – Estimate					10	10				
	View – Estimate					1	1				
	State – Schedule				1						
	Controller – Schedule							16			
	View – Schedule							1			
	Number of tasks:	12	6	18	11	11	11	17	11	0	0
	Effort:	2962	1481	4443	2715	2715	2715	4196	2715	0	0

WBS

Iteration Number	Planned Effort (minutes)	Cumulative Planned Effort	Planned Velocity	Cumulative Planned Velocity	Planned Completion Date	Actual Effort (minutes)	Cum Actual Effort	Earned Velocity	Cumulative Earned Velocity	Actual Completion Date	
1	3600	3600	12	12	3/23/2016	3360	3360	220	220	3/23/2016	
2	1800	5400	6	18	4/5/2016	1706	5066	43	440	4/5/2016	
3	1800	7200	18	36	4/19/2016	1696	6762	33	483	4/19/2016	
4	800	8000	11	47	11/15/2016	821	7583	11	516	11/15/2016	
5	800	8800	11	58	11/29/2016	760	8343	8	527	11/29/2016	
6	2000	10800	11	69	2/2/2017	1900	10243	15	535	2/2/2017	Curre
7	2000	12800	17	86	2/9/2017	2100	12343	15	550	2/9/2017	
8	2000	14800	11	97	2/16/2017	2120	14463	9	565	2/16/2017	

Calendar

Day #	Date	Available Minutes	Cumulative Minutes	Planned Velocity	Cumulative Planned Velocity	Actual Minutes	Cumulative Actual Minutes	Earned Velocity	Cumulative Earned Velocity	
1	1/26/2017		0		0		0		0	
2	1/27/2017		0		0		0		0	
3	1/28/2017		0		0		0		0	
4	1/29/2017		0		0		0		0	
5	1/30/2017		0		0		0		0	
6	1/26/2017	400	400	2	2	200	200	2	2	begin 6th Iteration
7	1/27/2017	400	800	2	4	100	300	2	4	
8	1/28/2017	0	800	2	6	0	300	1	5	
9	1/29/2017	0	800	2	8	0	300	2	7	
10	1/30/2017	400	1200	2	10	520	820	1	8	
11	1/31/2017	400	1600	2	12	420	1240	2	10	
12	2/1/2017	400	2000	1	13	420	1660	3	13	
13	2/2/2017	400	2400	0	13	240	1900	2	15	End 6th Iteration
14	2/3/2017	400	2800	4	17	200	2100	2	17	begin 7nd Iteration
15	2/4/2017	0	2800	2	19	0	2100	2	19	
16	2/5/2017	0	2800	2	21	0	2100	1	20	
17	2/6/2017	400	3200	4	25	460	2560	2	22	
18	2/7/2017	400	3600	1	26	520	3080	1	23	
19	2/8/2017	400	4000	2	28	420	3500	2	25	
20	2/9/2017	400	4400	2	30	660	4160	5	30	end 7nd Iteration
21	2/10/2017	400	4800	1	31	200	4360	1	31	begin 8rd Iteration
22	2/11/2017	0	4800	1	32	0	4360	0	31	
23	2/12/2017	0	4800	1	33	0	4360	0	31	
24	2/13/2017	400	5200	1	34	420	4780	3	34	
25	2/14/2017	400	5600	1	35	420	5200	4	38	
26	2/15/2017	400	6000	1	36	420	5620	1	39	
27	2/16/2017	400	6400	1	37	660	6280	0	39	end 8nd Iteration

Change Log

Use this worksheet to record changes to made to artifacts after the project is underway. Record repairs to defects, as well as changes that result from requirements clarifications/changes.

Change Designation	Description
Requirements Change	Changes to requirements
Requirements	Clarifications to requirements
Product syntax	Syntax flaws in the deliverable product
Product logic	Logic flaws in the deliverable product
Product interface	Flaws in the interface of a component of the deliverable product
Product checking	Flaws with boundary/type checking within a component of the deliverable product
Test syntax	Syntax flaws in the test code
Test logic	Logic flaws in the test code
Test interface	Flaws in the interface of a component of the test code
Test checking	Flaws with boundary/type checking within a component of the test code
Bad Smell	Refactoring changes (please note the bad smell in the defect description)

Number	Date	Type	Where Injected	nject Iteratio	Where Remove	emove Iteratio	Fix Time	Fix Reference	Description
1									
2									
3									

Time Recording Log

Date	Start Time	Stop Time	Interrupt	Delta	Activity	Iteration
2/10/0107	10:00 AM	11:40 AM	0	100	Analysis	8
2/10/0107	12:00 AM	1:00 AM	0	60	Architecture	8
2/10/0107	2:00 AM	2:40 AM	0	40	Project planning	8
2/13/0107	10:00 AM	5:00 PM	0	420	Construction	8
2/14/0107	10:00 AM	5:00 PM	0	420	Construction	8
2/15/0107	10:00 AM	5:00 PM	0	420	Construction	8
2/16/0107	10:00 AM	5:00 PM	0	420	Review	8
2/16/0107	10:00 AM	2:00 PM	0	240	Postmortem	8