Reaction: An Android Based Supplemental Learning Aid for Engineering Education

by

Christopher Gaddes

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
December 10, 2016

Keywords: Android, Engineering Education, Free-body Diagram

Abstract

The rapid rise of the smartphone platform is changing many facets of modern life. The platform presents exciting opportunities for supplemental education. Many mobile education applications are available e.g. Duolingo, an application that helps users learn foreign languages. Yet, the use of mobile applications in engineering education is still in its infancy. The purpose of this thesis is to explore the popularity of the mobile platform as a supplemental learning aid for engineering education. This is accomplished by the creation, distribution, and analysis of a proof-of-concept mobile learning application called *Reaction: Free-body Diagrams* which is targeted at engineering students studying mechanical free-body diagrams. The application was developed in four months, distributed for 1 month, and evaluated by survey. Over 180 downloads (installations) have been logged, and data from 50 survey respondents collected. This data suggests that the smartphone platform is a popular option among students for supplemental engineering education.

Acknowledgments

I would like to thank my committee for guiding me through this process. My adviser, Dr. Hung, has been an invaluable source of encouragement, insight, and advice. Dr. Madsen provided expertise on developing relevant statics problems for the application, and Dr. Riggs provided practical direction towards developing the data collection survey.

I am also grateful to the many people who tested my application, especially Win Sinor, who spent considerable time discovering bugs. Special thanks to James Sentell, William Hester, and Dr. Umphress, who graciously provided me with programming assistance and advice. I would also like to thank Jan Miles, who enabled me to continue my engineering education by letting me serve as her graduate assistant.

I would also like to thank my friends family for their love and support. I am especially grateful for my loving wife, who has graciously endured the many long hours I have worked on this project instead of spending time with her, and who has always been a source of creative genius. Finally, and most importantly, I would like to thank God for surrounding me with each of these people, and for providing me with this great opportunity.

Table of Contents

List of Figures

List of Tables

Chapter 1

Introduction

Can Mobile Applications Aid Engineering Education?

1.1 The Ubiquitous Mobile Platform

Processing power in personal mobile devices has increased exponentially since the 1990s [20]. As phones become more powerful, they are infiltrating more and more aspects of daily life. Smartphones are displacing functionalities of cameras, laptops, mp3 players, and many other devices. Put simply, mobile phones are everywhere. Looking around a typical college campus, one will see countless students glued to their smartphones, focused on texting, instant messaging, snapchatting, checking Instagram, and many other activities. The mobile platform is well established as a viable avenue for communication and entertainment, but what about as a platform for education? Popular educational applications (apps) exist, such as Duolingo, a mobile app to help users learn foreign languages. Duolingo has over 120 million users worldwide and teaches over 19 different languages [1]. Much of Duolingo's success can be attributed the following characteristics:

- It features a simple and intuitive user interface (UI).
- It uses gamification, which turns learning into a game where users can win or earn points to "level up".
- The app is offered to users completely free of charge.
- Duolingo now offers Duolingo for Schools, a platform that is infiltrating the formal educational system with new mobile electronic technology.

1.2 Thesis Aim

Many education applications such as Duolingo are already utilizing the mobile platform for educational purposes. However, despite being a field with an emphasis on the creation of new technologies, the engineering field seems to be behind the curve when it comes to utilizing mobile technologies for educational purposes. The aim of this thesis is to evaluate the usefulness of the smartphone platform as a supplemental learning tool for engineering education. This aim is accomplished with the following steps:

1. Develop a proof-of-concept engineering education application which runs on the Android mobile platform
2. Have users test the application and provide feedback
3. Analyze feedback and draw conclusions

1.3 Thesis Objectives

There are four main objectives in the thesis aims:

1. Review relevant literature
2. Learn mobile application development
3. Create a mobile application which allows engineering students to practice drawing free-body diagrams on their mobile device
4. Evaluate the usefulness of the app by analyzing feedback from users in context with relevant literature, and draw suitable conclusions

1.4 Thesis Structure

This thesis contains seven chapters:

- Chapter 2 - The State-of-the-Art: This chapter reviews some relevant literature from the field of engineering education.
- Chapter 3 - Technical Approach: This chapter lists existing applications related to the subject matter, defines the primary users the application is designed for, and lists the functional and non-functional requirements for the application.
- Chapter 4 - Application Design: This chapter discusses the design of the application including implementing data storage, alleviating privacy concerns, designing the user interface, and developing a process to draw free body-diagrams.
- Chapter 5 - Implementation: This chapter describes the various tools and libraries used in the application development and discusses how performance issues are overcome.
- Chapter 6 - Evaluation and Testing: This chapter discusses the specs of the final product, application testing, how the application requirements are met, and discusses user feedback.
- Chapter 7 - Conclusion: This chapter summarizes the work done in developing the application. It also discusses ideas for future work, and makes concluding remarks.

Chapter 2
The State-of-the-Art

This chapter reviews some relevant literature from the field of engineering education.

2.1 E-learning

Electronic learning or e-learning is defined as "the use of any electronic media and devices to facilitate the learning process." In recent history, e-learning has been implemented using electronic mediums such as radio, television, analog and digital audio recordings, personal computers, websites, and most recently mobile devices [20]. Although e-learning has historically been used to describe learning through each of the aforementioned mediums, the term is typically now only used to describe web-based learning. The terms e-learning and web-based e-learning will be used interchangeably throughout the remainder of this thesis. There are many benefits to e-learning including:

- **Accessibility:** The learning material is accessible from any Internet connected device with a supported web browser.
- **No restrictions of time and space:** The medium is not limited by time and space so users can study anytime and anywhere.
- **Choice:** Users typically get to choose when they work, what they work on, and for how long.
- **Flexibility:** In theory, the e-learning platform can be used to educate on any subject.

2.1.1 Challenges to e-learning

Many educational institutions use Learning Management Systems (LMSs) to implement e-learning. Popular LMSs include FirstClass, WebCT, Moodle, BlackBoard, and Canvas [20], of which the latter is currently used by Auburn University. Canvas works best on laptop/desktop environments with mouse and keyboard input and is not well suited for touch input or for small displays. Desktops are stationary by nature, and therefore limit the locations where users can learn on them. Laptops are obviously more portable than desktops, but their portability is limited by the facts that they typically must be connected to Wi-Fi, they are fairly large, they are often quite heavy, and often have a quite limited battery life.

Thus in its current form, web based e-learning is an extremely powerful platform which is limited by two major flaws:

1. Requires Internet access at all times

2. Requires a desktop environment

Although web based e-learning is extremely flexible, these two flaws can actually be quite limiting.

2.2 M-learning

M-learning, or mobile learning is a form of e-learning which is able to largely overcome the two aforementioned limitations. Although there is not a universal definition of m-learning, it will be defined in this thesis as learning by means of personal mobile electronic devices [13]. M-learning by this definition is a subset of e-learning. Also, this definition excludes devices such as laptops. In this thesis the only devices considered personal mobile devices are smartphones and tablets. The mobile learning platform is a phrase used to describe the platform which contains the devices that meet this definition.

For the purposes of this thesis, the phrase "traditional education" will be used to describe education that utilizes conventional methods to teach. In engineering education these typically include listening to in-class lectures in "brick and mortar" classrooms, attending office hours, doing homework, taking quizzes and tests on paper, reading textbooks, and often performing lab work. The intention of this thesis is not to suggest that traditional education techniques such as these should be eliminated, but to discuss the usefulness of supplementing these techniques with m-learning.

2.2.1 Characteristics of the mobile platform

There are many characteristics of the devices that comprise the mobile platform:

- **Small Screen:** Mobile devices typically feature screens that are much smaller than laptops or desktops.
- **Battery powered:** Mobile devices are powered by small batteries and must be conservative with energy usage.
- **Limited storage:** Mobile devices typically have very limited storage space.
- **Always on and nearby:** Mobile devices are often always "on" and many people have them on their person at all times.
- **Touch controls:** Mobile devices are typically controlled via touchscreen input.

2.2.2 Basic Characteristics of m-learning

According to [19], the basic characteristics of m-learning are:

- **Ubiquitous/Spontaneous:** The majority of people in the United States have mobile devices.

- **Portable:** Mobile devices used for m-learning are extremely portable. Because of this portability, many people choose to carry a mobile device with them nearly everywhere they go.

- **Blended:** Teachers can use m-learning as part of a blended educational system. Blended learning is an educational style which uses electronic sources such as those offered by e-learning and its subset m-learning to supplement"traditional education" techniques [19].

- **Private:** Mobile devices used for m-learning are personal and can be used in a private setting. Many mobile users password protect their mobile devices. Additionally, students can operate independently of students and thus learn at a pace that feels best for them.

- **Interactive** Mobile devices provide potential for a high level of interactivity and feedback. Users can work interactive problems on their mobile devices and receive feedback.

- **Collaborative:** Mobile devices can allow collaboration between students and teacher.

- **Instant information:** Since mobile devices are frequently connected to the Internet at all times via cellular data or Wi-Fi, information can be exchanges instantly;

2.2.3 Advantages of mobile learning

The mobile platform has many benefits when used for learning purposes:

- **Ubiquity:** In a 2014 US study, 86% of undergraduate students owned a smartphone and 47% owned a tablet, and these numbers are rising each year [12]. The smartphone is the most popular electronic device [18]. Since the majority of students own smartphones, it is a prime platform for supplemental education.

- **Instant feedback:** One of the main complaints students frequently give is that they don't receive enough feedback on their work before taking a test. Many professors offer graded homework for this purpose, but the difficulty of these problems can often lead students to spend excess time "spinning their wheels" on concepts that a professor could answer in seconds. Although there is value in grinding away at difficult problems, often the issues in which students get stuck are trivial in nature. Mobile learning allows instant feedback, which prevents the student from wasting excess time, and allows them to focus on learning, rather than tedium.

- **Rapid deployment:** Unlike textbooks, which feature a static list of problems, mobile learning can be changed at a moment's notice via updates to the app or remote changes to databases.

- **Lightweight and Portable** Mobile devices are small and light, especially compared to textbooks. M-learning could reduce reliance on heavy books and laptops. Even as mobile devices continue to become thinner and lighter each year, textbooks remain generally constant in size. Additionally, laptops, despite becoming lighter as technology increases, are still significantly larger and heavier than mobile devices. Carrying around heavy objects such as books and laptops in backpacks can lead to back pain and other physical health problems [21].

- **Convenience:** In order to directly learn from a professor, a student typically must either attend a lecture or visit them in their office hours. Both of these resources, although valuable, can only be accessed at fixed times. Mobile devices are extremely portable, typically always on, and many people take them everywhere. Since they are always accessible, they can be used for learning purposes at any time. Students can use mobile devices to study on a transit ride, or while waiting on a friend at a restaurant, or even while flying on a plane, all without having to worry about lugging around textbooks or laptops and without the time constraints of receiving help directly from a professor in office hours.

- **"Bite sized" learning:** Unlike lectures which are typically a fixed length of time, the mobile platform allows concepts to be broken down into bite sized chunks.

- **Touch Interface:** Touch screens are extremely efficient for certain types of activities, namely those comprised largely of swiping, and tapping, and similar gestures. Touch screens possess a unique input capability that is different than what pen and paper or a mouse can offer.

- **Personalization:** Since mobile devices are personal in nature, their content can be tailored to the individual in ways that lectures or textbooks cannot. The learning platform can adapt to the user's specific needs. For instance, if a student is having trouble with a certain concept, the mobile device could present them with problems targeted at allowing the student to practice and develop and understand of that tricky concept.

- **Cost effective:** For the 86% of undergraduate students who already have mobile devices, the mobile platform can be used as an educational platform with no additional cost. Granted, the content such as purchased apps may not be free, but the platform itself is effectively free for a large portion of the population.

2.2.4 Some Current mobile learning apps

There are currently many m-learning applications. Below are some of the most popular Android m-learning applications featured on the Google Play Store:

- Duolingo[1] / Babbel[2] - These two apps allow users to learn a foreign language on their mobile device. There are many similar applications which seek to teach foreign languages to users.

- Treehouse[3] - This app teaches users various programming languages.

[1]https://play.google.com/store/apps/details?id=com.duolingo
[2]https://play.google.com/store/apps/details?id=com.babbel.mobile.android.en
[3]https://play.google.com/store/apps/details?id=com.teamtreehouse.android

- Khan Academy[1] - This app allows users to access over 10000 educational videos from their mobile device. These videos cover an extremely diverse range of subjects.

- Quizlet Flashcards and Learning[2] - This app allows the user to memorize information using virtual flashcards.

- TED[3] - This app provides users access to more than 2000 TED Talks on their mobile device.

- Udacity[4] / Coursera[5] / EdX[6] - These apps allow users to access various educational courses from their mobile device.

2.3 Mobile Operating Systems: Two Choices

One of the project objectives is to learn mobile application development in order to create a suitable mobile application. The two most popular mobile operating systems (OSs) are Google's Android and Apple's iOS. Together, they hold the overwhelming majority of the mobile operating system market share worldwide [3]. Android is an operating system developed by Google, and is based on the Linux kernel. Android applications are typically developed in Java. However, native programming in C++ is also supported through the native development kit. This kit is typically only used to develop computationally intensive applications such as games. Android was originally designed as a mobile OS with the primary input interface being a touch screen, but has been successfully run on desktops and has support for keyboards and mice. The core functionality of Android is open source (free) and its source code is released through the Android Open Source Project (AOSP) [9]. IOS is a proprietary operating system developed by Apple, and runs exclusively on iPhones and

[1]https://play.google.com/store/apps/details?id=org.khanacademy.android
[2]https://play.google.com/store/apps/details?id=com.quizlet.quizletandroid
[3]https://play.google.com/store/apps/details?id=com.ted.android
[4]https://play.google.com/store/apps/details?id=com.udacity.android
[5]https://play.google.com/store/apps/details?id=org.coursera.android
[6]https://play.google.com/store/apps/details?id=org.edx.mobile

iPads. IOS applications are typically developed in Objective-C or Swift. Android was chosen of this work for the following reasons:

- **Access to Android Devices for Testing:** The Android platform was chosen primarily because the author owns three Android phones and one Android tablet. Testing could be done on these devices without purchasing iOS devices or relying entirely on an iPhone emulator. If the writer had owned an iPhone instead Android devices, it is likely that iOS would have been chosen instead of Android.

- **Development Cost:** Becoming an Android developer is less expensive than becoming an iOS developer. Google charges a one-time registration fee of $25 to new Android developers [10] whereas Apple charges an annual fee of $99 to iOS developers [5]. Additionally, XCode, Apple's Integrated Development Environment (IDE) for iOS development is only available on Mac OS. Conversely, Android Studio, Google's IDE for Android Development is supported on Mac OS, Windows, and Linux. Since it is likely a developer already has a computer running one of these three popular operating systems, many starting Android developers will not need to buy additional development hardware. However starting iOS developers must purchase a Mac if they currently have only a Windows or Linux machine.

- **Walled Garden:** Apple wields much tighter control over the iOS App Store than Google does over the Google Play Store. Apple uses human testers to manually test each update. This ensures applications meet Apple's quality standards, but gaining approval can take a couple days and there is always a chance the submission will be rejected completely. This approval delay can become a problem if an update needs to be pushed quickly to users in order to fix a glaring or even dangerous bug. Google also requires approval before publishing, but their process is much less stringent than Apple's and typically much faster. In development of this app, most new updates were approved by Google within half an hour. Thus, the Android platform has an advantage when it comes to pushing rapidly pushing updates to squash bugs [23].

In addition to tightly controlling the App Store, Apple also allows developers less access to OS features than Google does with Android. For instance, Google allow Android apps to "draw over other apps", which is where applications can display objects such as buttons or chat windows over other running applications. Many applications uses this powerful feature to make "floating apps", such as the chat heads used by Facebook's Messenger[1]. "Chat heads" are expandable Facebook chat windows that can "float" over the other currently running apps. This allows users to reply to chat messages without leaving their current app and opening Facebook Messenger. Apple does not give developers this ability. Android devices can also have custom home screen launchers, whereas Apple does not allow this [23]. There are many other powerful features that Android developers can access which Apple does not allow. Although the features needed for this application ended up being available on both platforms, certain features which were considered in the early planning stage would have been impossible to implement on iOS due to Apple's restrictions.

- **Android is open source:** Although more and more valuable software components on Android phones are becoming proprietary such as Gmail[2], Calculator[3], Clock[4], Phone[5], the core functionality of Android is still open source through the Android Open Source Project (AOSP) [9]. The Android platform was chosen in part because it is still mostly open source, which stands in stark contrast to Apples tightly controlled and proprietary platform.

[1]https://play.google.com/store/apps/details?id=com.facebook.orca
[2]https://play.google.com/store/apps/details?id=com.google.android.gm
[3]https://play.google.com/store/apps/details?id=com.google.android.calculator
[4]https://play.google.com/store/apps/details?id=com.google.android.deskclock
[5]https://play.google.com/store/apps/details?id=com.google.android.dialer

Based largely on the above reasons, an Android app is developed here instead of an iOS app. Developing for both Android and iOS simultaneously was briefly considered, but was decided against since it would have nearly doubled the development time.

2.3.1 Challenges to using Android as the platform

Despite its popularity, the Android platform is not without challenges. A widely publicized issue with the Android ecosystem as a whole is fragmentation. When Apple releases an update for iOS, it is immediately pushed out to all iPhones which support the new OS. Apple is able to accomplish this because they create and control both the software and the hardware of the iPhone. However, Google only produces the software (Android) and leaves the hardware creation to companies such as Samsung, Motorola, HTC, and LG, which are called Original Equipment Manufacturers (OEMs). Thus, although Google releases monthly security updates and yearly major version updates to the Android operating system via AOSP, they are unable to push the latest version of Android directly to user's devices (with the exception of Nexus and Pixel devices). Users must wait for OEMs to test the newest Android updates from Google and adapt them to fit their own specific device. This process is lengthened by the fact that OEMS often use custom user interfaces (UIs) such as Samsung's Touchwiz or HTC's Sense to modify the appearance and functionality of the basic Android source code from Google. Because of this, the overwhelming majority of Android users are not running the latest major version. Upon writing, so few phones have been updated to the latest version of Android, 7.0 Nougat, that it doesn't even appear on the Platform Versions chart that Google updates on Android's website (see Figure 3.1). This means that less than 0.1% of Android users are running the latest version of Android. This is a problem for developers because each new version of Android adds new Application Programming Interfaces (APIs) which can be used to make better apps. Apps utilizing these new features cannot run on older versions of Android without the use of support libraries (See Section 3.3.1). This often forces developers to alienate large portions of their target audience who may be running older versions of Android in order to implement powerful new features into their applications. Despite this challenge, Android is chosen as the mobile platform for the educational application of this thesis.

2.3.2 Getting started in Android App Development

Since Android applications are programmed in Java, a background in Java is often recommended for aspiring Android developers. However, this application was developed in one summer by a developer with no formal background in Java programming. While a background in Java is not absolutely necessary, it will certainly give the developer a head start. Many useful resources for learning the basics of Android Development can be found in textbooks, or found online with a cursory search. The following developer resources are used heavily in this thesis:

- **slidenerd:** Although there are many educational videos on sites like YouTube which teach Android programming skills, slidenerd, `https://www.youtube.com/user/slidenerd`, is one of the most useful, especially for beginners. Founded by developer Vivek Ramesh, slidenerd on YouTube is a fantastic free resource for beginning developers. Vivek Ramesh also produces an online course called *Design + Code an Android App from Scratch* which is available for purchase on Coursetro.com and on Udemy.com under the name *How To Launch Your App In Just 16 hrs.* Skills learned while taking this course were extremely influential in the development of this application. The course, which can be purchased for a discounted price of $15 on Coursetro, walks the student through how to completely design, code, and publish an Android application from start to finish. Courses such as this are great starting resources for beginning Android developers.

- **Stack Overflow:** A valuable resource for both experienced developers and beginners is Stack Overflow. Stack Overflow, found at `stackoverflow.com`, is a massive online community where users can ask and answer programming related questions. As development questions and problems arise, developers are encouraged to search Stack Overflow first, since others have likely faced and solved similar problems before.

- **Official Android Developer Website:** Google maintains a fairly comprehensive set of developer documentation on the official Android Developer website:
  `https://developer.android.com/training/index.html`
  This documentation is extremely useful as a reference once a rudimentary understanding of Android development has been established. This resource is listed last since there are many other resources better suited to beginners.

2.4 Android Studio

Although there are many quality IDE's that can be used for Android Development, Google recommends Android Studio, an IDE released in collaboration with JetBrains. Android Studio can be downloaded from the following link:
`https://developer.android.com/studio/index.html`
The default settings in the installation process should be adequate for most users.

2.5 Testing During Development

Since it is impractical to test an app on every supported device, emulators are often used in development. An Android emulator mimics a specific Android device and allows one to test the application on the virtual device without having to physically possess many devices. This is extremely useful for making sure that user interfaces look good on devices of various screen sizes and aspect ratios. Android Studio comes with an emulator that has improved greatly recently in terms of performance, but third-party emulators can offer additional features and improved performance. An emulator called Genymotion is used during the

development of this application and can be installed form the following link:
`https://docs.genymotion.com/Content/01_Get_Started/Installation.htm`

2.6 Release Platform

This section discusses the platform on which the application was released and how *Reaction* was marketed in order to maximize adoption.

2.6.1 Distribution

Although users can install apps from third party sources in Android by enabling it in settings[1], the primary way to install apps on Android is thought the Google Play Store. In order to maximize adoption, *Reaction* was released completely free of charge and completely ad-free on the Play Store[2]. Additionally, the entire app is open source and can be found on GitHub at: `https://github.com/ChrisGaddes/Reaction`

2.6.2 Marketing

The application was marketed in three main ways in order to maximize adoption:

- **Personal contact:** Play Store links to the application were manually given to friends and family members willing to try out the application. This method was very effective in the short term since personal friends are likely to agree to help. However, the number of personal friends having experience in drawing free-body diagrams and also owning an Android device is quite limited, so this source of users was quickly exhausted.

- **Emailed to students by professors:** Several engineering professors at Auburn University agreed to email their classes with information about the application.

- **Play Store Posting:** The Play Store listing contains high resolution screenshots of the application in action along with the following description:

[1]https://developer.android.com/distribute/tools/open-distribution.html
[2]https://play.google.com/store/apps/details?id=com.chrisgaddes.reaction

> Practice Free-body Diagrams on the go. There are currently three main problems you can solve, each with three sub-parts. This app does not currently teach you how to draw free-body diagrams (although that may be added in future updates), but it will allow you to test your knowledge and hone your skills by drawing 9 free-body diagrams of varying difficulty. This app is useful for students taking Physics, Statics, Dynamics, Systems, and many other science and engineering courses.

Engineering keywords such as Physics, Statics, Dynamics, Systems, free-body diagram, etc., are strategically included in the description so the application would be listed in a search for those terms. At the time of this writing, the application is the top result of the fourteen applications displayed if the phrase "free-body diagram" is searched for in the Google Play Store. Thus, any Android users who search for this phrase are likely to discover the application.

Additionally, users who found the application useful were encouraged to rate it on the Google Play Store. At time of writing, the app has thirteen 5/5-star ratings. Good ratings are a powerful marketing tool.

- **Reddit:** Links to the application were posted on Reddit[1,2,3,4,5,6], a massive online forum. Of the various marketing techniques, linking on Reddit seems to have been the most effective. The various Reddit posts received eighty-four "upvotes" combined and over sixty users installed the application on the day the first post was made. The application was also posted on Facebook that same day, but received little notice (only one "like") so Facebook likely did not greatly contribute to a spike in new installations.

[1] https://www.reddit.com/r/EngineeringStudents/comments/55v5wl/practice_freebody_diagrams_android_app/
[2] https://www.reddit.com/r/auburn/comments/55va0u/practice_freebody_diagrams_android_app/
[3] https://www.reddit.com/r/androidapps/comments/566flv/practice_freebody_diagrams_android_app/
[4] https://www.reddit.com/r/MET/comments/55v7xt/practice_freebody_diagrams_android_app/
[5] https://www.reddit.com/r/Android/comments/566dix/practice_freebody_diagrams_android_app/
[6] https://www.reddit.com/r/androiddev/comments/566hna/practice_freebody_diagrams_android_app/

Chapter 3
Technical Approach

This chapter lists existing applications related to the subject matter. It also defines the primary users the application is designed for, and lists the functional and non-functional requirements for the application.

3.1 Existing applications related to free-body diagrams

Before developing the application, the Google Play Store was examined to determine if there exists an Android application that meets some or all of the application requirements (see Chapter 3.3). The following Android applications describe how to draw free-body diagrams:

- Grade AP Physics: Equilibrium[2]
- Learn Physics[3]
- Mechanics - Physics[4]

Each of these applications explains the theory, but users cannot actually practice drawing free-body diagrams in the app. Additionally, none of them have been updated recently and their user interfaces do not follow Google's Material Design specifications (see Chapter 4.1.3). Based on this analysis, it was determined that no Android application exist that allow users to practice drawing free-body diagrams.

[2]https://play.google.com/store/apps/details?id=com.mikel.com.mikel.freebodydiagram
[3]https://play.google.com/store/apps/details?id=com.iphonedevro.learnphysics
[4]https://play.google.com/store/apps/details?id=learnersseries.physics.mechanics

3.2 Primary users: Students with experience in Mechanics

A critical component of creating an application is establishing the target users. The application was designed such that students with a background in drawing free-body diagrams should have success in solving the in-app problems. Free-body diagrams are typically taught in courses related to engineering Mechanics. This field is defined as "a branch of the physical sciences that is concerned with the state of rest or motion of bodies that are subjected to the action of forces."[15] Mechanics is typically featured in engineering courses such as Physics, Statics, Dynamics, and Mechanics of Materials, and many others.

3.2.1 Fundamental Mechanics concepts

This section defines several fundamental terms of used in engineering Mechanics. Readers seeking additional detail may consult a reference such as [15]

- *Mass* - Mass is "a measure of quantity of matter that is used to compare the action of one body with that of another. This property manifests itself as a gravitational attraction between two bodies and provides a measure of the resistance of matter to change in velocity" An object is called *massless* if it has no mass. Objects of extremely small mass are sometimes considered *massless* for ease of calculation.
- *Force* - A force is "a 'push' or 'pull' exerted by one object on another."
- *Moment* - A moment is a torsional (rotational) force.
- *Friction* - Friction is the term used to describe the resistance between two contacting objects which inhibits motion between them. Frictionless surfaces have no resistance to motion, and do not exist in nature. However, low friction surfaces are often approximated as frictionless in order to ease calculation.
- *Distributed Force* - A distributed force is force that is applied across a surface. It is sometimes called a distributed load.

- *Free-body diagram* - A free-body diagram is a diagram which denotes all forces and moments acting on a system.
- *Two force member* - A two-force member is a member with exactly two co-linear forces of equal magnitude and opposite direction applied to it.

3.3 Requirements

This section discusses the functional and non-functional system requirements that were used in the development of the application. These requirements were created after considering the needs of the primary users and factoring in constraints on development time.

3.3.1 Operating system

Android 4.4 “KitKat”, API level 19, was chosen as the minimum supported version of Android for the application. Version 4.4 was chosen because 81.4% of all Android devices run Android 4.4 “KitKat” or later. Android 4.3 Jelly Bean was also considered because 97% of all Android devices run Android 4.3 or later, but it was not chosen because its successor, Android 4.4, contains many useful APIs. Additionally, the developer did not have easy access to devices running Android 4.3 for testing purposes. Android 5.0 “Lollipop” was also considered as the minimum requirement because it is a major update containing many useful APIs. However, with some minor workarounds such as the use of various support libraries from Google (See Section 3.3.1), Android 4.4 was finally chosen because it is recent enough that it contains the essential functionality required to run the app while still supporting the majority of users. Table 3.1 and Figure 3.1 are provided by Google and show Android usage from September 30th - October 5th, 2016[1]. Android version containing less than 0.1% of the distribution are not included in the data. Interestingly, Android 7.x Nougat, the latest Android version, is not included because it is still represents less than 0.1% of the market even after being released more than a month ago.

[1]https://developer.android.com/about/dashboards/index.html#Screens

Version	Codename	API	Distribution
2.2	Froyo	8	0.1%
2.3.3 - 2.3.7	Gingerbread	10	1.5%
4.0.3 - 4.0.4	Ice Cream Sandwich	15	1.4%
4.1.x	Jelly Bean	16	5.6%
4.2.x		17	7.7%
4.3		18	2.3%
4.4	KitKat	19	27.7%
5.0	Lollipop	21	13.1%
5.1		22	21.9%
6.0	Marshmallow	23	18.7%

Table 3.1: Table of Android Platform Percentages

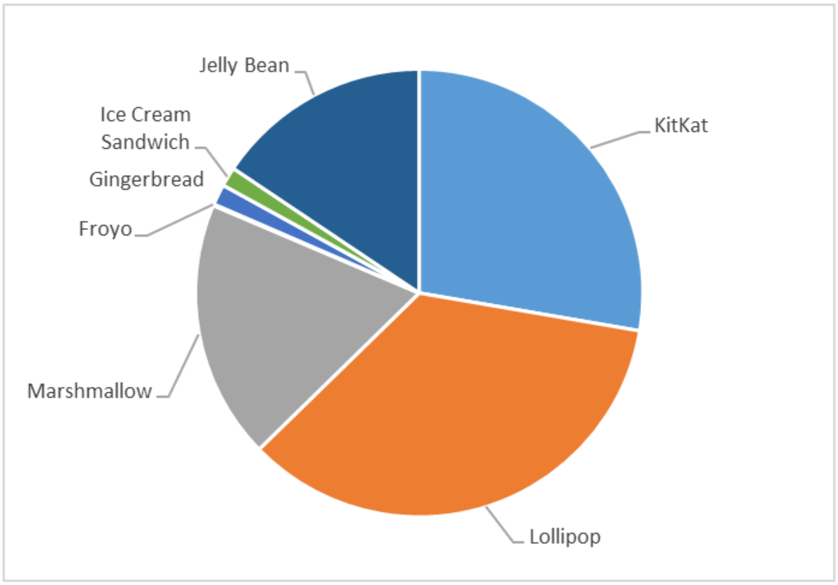


Figure 3.1: Chart of Android Platform Percentages

3.3.2 Functional requirements

Functional requirements are “statements of services the system should provide, how the system should react to particular inputs and how the system should behave in particular situations. In some cases, the functional requirements may also explicitly state what the system should not do.”[25] The functional requirements for the application are listed below:

- **Practice free-body diagrams:** Users should be able to practice drawing free-body diagrams on three discrete problems, each with three sub-parts (A, B, and C). These diagrams should collectively contain key elements of free-body diagrams including:
  - Forces
  - Moments
  - Applied forces
  - Distributed forces
  - Fixed elements
  - Gravity
  - Members with mass
  - Massless members
  - Pinned joints
  - Wheels
  - Rockers
  - Springs
- **Work offline:** The application should not depend on the Internet for any functionality (excluding initial installation and subsequent updates).
- **Use no additional permissions:** The application should not use or request any additional permissions from the user. In order to protect users and add transparency, modern versions of Android must manually request permissions from the user in order to utilize various aspects of the phone. For instance, if a developer wants their app to have access to the user’s location, they must request this from the user in a pop-up dialog, which the user can either accept or deny. The proposed application should not use or request any such additional permissions.

- **Log usage time and report feedback:** The application should log the application usage time: the total sum of the time the application is being actively used. Additionally, it should have a mechanism to anonymously report this usage time, usage feedback, and any comments to the developer for analysis.
- **Link to survey:** The application should provide a link to an anonymous survey which the user can open using a web browser of their choice.
- **Tutorial:** The application should contain a tutorial that walks the user through basic usage.

3.3.3 Non-functional requirements

Non-functional requirements are "constraints on the services or functions offered by the system[25]." These are requirements that can be used to evaluate the operation of the app, rather than specific behavior, and can be subjective in nature. The non-functional requirements include:

- **Easy to use:** The majority of reporting users should consider the application easy to use.
- **No bugs:** The application should be free of bugs and thus operate as designed.
- **Run on Android 4.4 and higher:** The application should run on devices running Android 4.4 "KitKat" or higher.
- **Good performance on low end devices:** The application should have good performance on low-end devices. Many Android devices, especially those popular in emerging countries, are extremely inexpensive. This low cost is possible because the manufacturers used low powered processors and low resolution screens. The application should have good performance on these low-end devices.

- **Responsive:** There should be no noticeable lag between a touch input and the resulting visual feedback caused by it.

- **Support common screen sizes:** The application should support common screen sizes and aspect ratios.

- **Modular code:** The application code should be designed to be modular in nature so that new features such as additional problems can be added with relative ease.

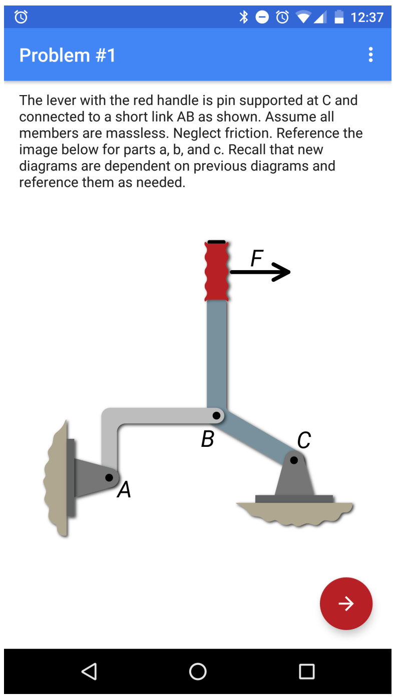
3.4 Chosen Problems

Ideally, the application would feature an enormous database of problems for users to practice with. Due to time constraints, the requirements were set at three main problems that each have three sub parts. Since this is a relatively small database of practice problems, each problem must cover a wide swath of the required free-body diagram elements listed previously. The problems were chosen carefully and the main problem statement for each can be found in Figure 3.2. The problems are ordered by difficulty with Problem 1 being the easiest and Problem 3 being the most difficult. Each problem adds important elements of free-body diagrams not contained in a preceding problem as shown below:

- Problem 1 - This problem contains only massless members. It also contains a two force member, member AB. It also contains pinned joints at A, B, and C. It also contains the anchors which are fixed to the wall and ground at points A and C respectively.

- Problem 2 - This problem adds a spring and a wheel with mass which means gravity is also used.

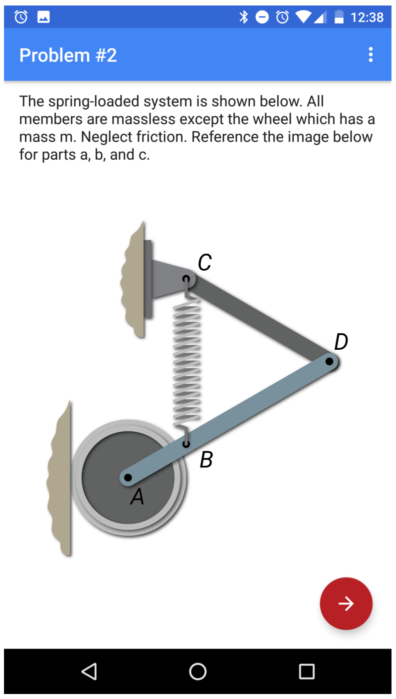
- Problem 3 - This problem adds a rocker at point C and a distributed load, w, applied to member BC.


12:37

Problem #1

The lever with the red handle is pin supported at C and connected to a short link AB as shown. Assume all members are massless. Neglect friction. Reference the image below for parts a, b, and c. Recall that new diagrams are dependent on previous diagrams and reference them as needed.

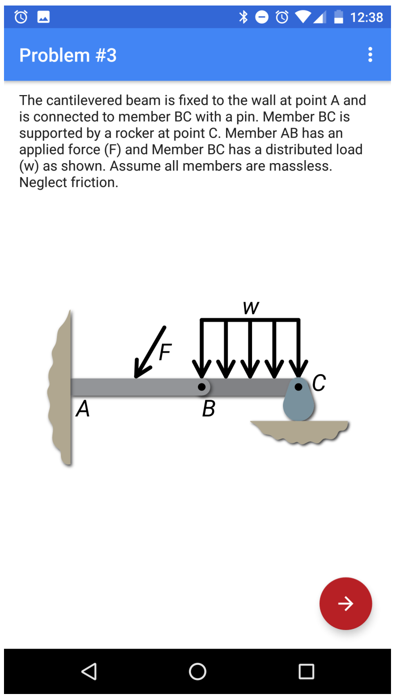
(a)


12:38

Problem #2

The spring-loaded system is shown below. All members are massless except the wheel which has a mass m. Neglect friction. Reference the image below for parts a, b, and c.

(b)


12:38

Problem #3

The cantilevered beam is fixed to the wall at point A and is connected to member BC with a pin. Member BC is supported by a rocker at point C. Member AB has an applied force (F) and Member BC has a distributed load (w) as shown. Assume all members are massless. Neglect friction.

(c)

Figure 3.2: Problem Statements

Chapter 4

Application Design

This chapter discusses the design of the application. The design was developed to meet the requirements defined in Chapter 3 and includes implementing data storage, alleviating privacy concerns, designing the user interface, and developing a process to draw free body-diagrams.

4.1 User interface (UI) design

A critical component of a mobile application is the user interface. A poorly designed user interface can cripple a functionally useful application. This section discusses various considerations that were examined in designing the application user interface. Although there are many user interface (UI) styles, three popular design styles are discussed in the following sections.

4.1.1 Skeuomorphism

Skeuomorphism refers to a user interface that is designed to mimic real world objects. For example, Apple's iBooks app used to have the appearance of a wooden bookshelf complete with realistic looking books [17]. Skeuomorphism is often used to tap into the familiarity users have with real-world objects. For instance, a skeuomorphic volume dial would use elements such as shading, textures, and drop shadows to appear three-dimensional and mimic real volume dials. When users interact with this dial they are able to use their past experience in turning real volume knobs to quickly understand how to interact with the dial in the interface. This makes skeuomorphic interfaces quite intuitive. Steve Jobs, the late CEO of Apple, was famously a huge proponent of skeuomorphism. His unwavering support explains its widespread use in Mac OSX and iOS for many years [22]. Once popular and even ubiquitous, however, skeuomorphism has dramatically decreased in popularity and use in recent years.

4.1.2 Flat design

Flat design is a broad term used to describe user interfaces that eschew the illusion of three dimensionality found in skeuomorphism in favor of flatness. This means removing elements such as gradients, drop shadows, textures such as brushed metal or wood grain, and other elements designed to make the interface appear appear three-dimensional [26]. In iOS 7, Apple replaced Steve Jobs' beloved skeuomorphism with a version of flat design. Instead of the faux textures such as wood, brushed steel, and leather, which decorated many core visual aspects of iOS 6 and before, the flat design in iOS 7 and higher is comprised of completely flat sheets. These sheets are often semitransparent, which creates the illusion of layered sheets of frosted glass. Despite its visual appeal, flat design is not as intuitive as skeuomorphism, since all elements must be flattened into a single plane and users are not able to use their past experience with real world objects to intuitively understand how to interact with the user interface elements. For example, it is sometimes hard to identify touchable elements such as buttons in flat design because they are not "raised" above their surroundings as physical buttons often are.

4.1.3 Material Design

In 2014, Google released a design language called Material Design at the Google IO conference. Material design was showcased on that latest version of Android at that time, 5.0 "Lollipop" [11], and is implemented in all later Android version to date. Material design is an attempt to take the intuitive resemblance to real-world objects that skeuomorphism offers and combine it with the minimalism of flat design. Material design is comprised of layered sheets of "paper" that appear to be "floating" at different elevations in the interface due to the strategic use of drop shadows. These shadows help differentiate buttons, which can appear raised above their surroundings. Material Design utilizes a palate of vibrant colors contrasted with generous whitespace to create beautiful user interfaces.

4.1.4 Visual Comparison

Each of these three user interface design styles can be seen in the figures below:

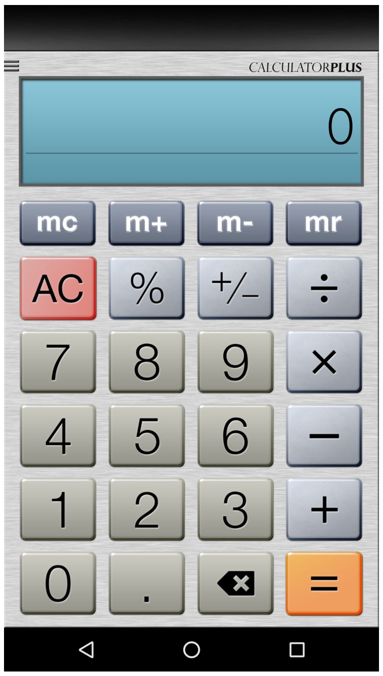


Figure 4.1: Skeuomorphism

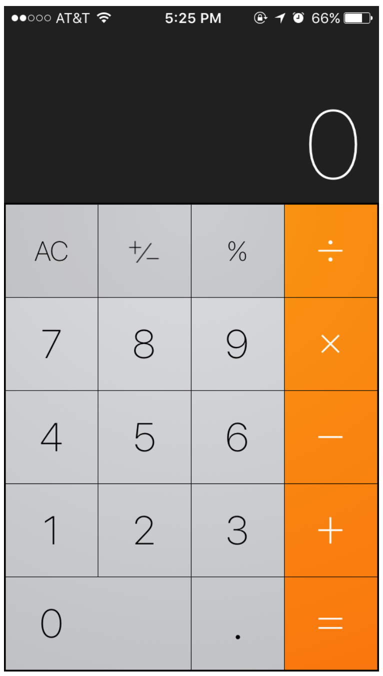


Figure 4.2: Flat Design

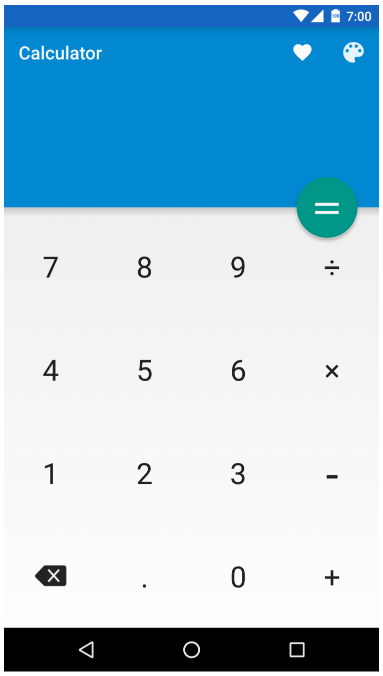


Figure 4.3: Material Design

The calculator in Figure 4.1 implements skeuomorphic design, as it is designed to resemble a physical calculator with raised buttons. The calculator in Figure 4.2 implements flat design, as it features no gradients or drop shadows. At first glance, the calculator in Figure 4.3 looks like it might implement flat design. However, there are subtle drop shadows under the "equals" button which are characteristic of Material Design.

4.1.5 Chosen design style

Material Design was chosen as the design style for this application over flat design and skeuomorphism for the following reasons:

- **Used by Google:** Google typically follows the Material Design specifications in its own applications and in the Android Operating System as a whole. Additionally, many popular Android applications implement Material Design, so using it ensures that your user interface will be consistent with other applications in the Android ecosystem[2].

- **Comprehensive Guidelines:** Google's comprehensive Material Design guidelines enable developers to quickly create beautiful and well laid out user interfaces without requiring robust background in graphic design, since Google has done much of the design work already. Additionally, Google releases many useful interface elements already designed to meet Material Design specifications. Thus, by using these elements and following Google's guidelines, even developers with little to no design experience can create beautiful and intuitive user interfaces.

- **Aesthetics:** Material Design is considered beautiful in it's judicious use of bright colors, white space, and "raised" elements via drop shadows. Additionally, there are many libraries that make implementing beautiful Material Design animations extremely simple.

- **Intuitive:** Material Design is designed to be extremely intuitive. The strategic use of drop shadows creates the illusion of three-dimensionality, which visually differentiates objects that can be interacted with such as buttons. Material design is able to combine the intuitiveness offered by skeuomorphism with the simplicity and beauty offered by flat design.

4.1.6 Icon and Name

The first thing that a user sees before using an application is the app icon as shown in Figure 4.4. This icon is displayed in the Play Store before the user installs the app and the app is launched by taping on the icon from the home screen, called the launcher icon. In developing the icon for this application, Google's Material Design guidelines were followed.

The icon portrays a block resting on a triangular shaped ramp. The black arrow is a reaction force applied to the box by the ramp, and is the origin of the app name: *Reaction: Free-body Diagrams*.

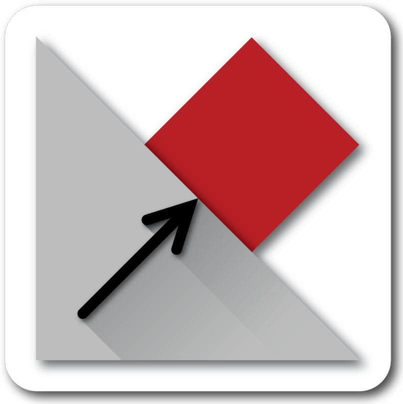


Figure 4.4: *Reaction: Free-body Diagrams* Launcher Icon

4.1.7 Meaningful Motion

An integral part of Material Design is its emphasis on realistic motion through animations. In Material Design, animations are not merely eye candy but are used to make the sheets of paper on the screen feel real and follow the laws of physics. Instead of just appearing out of nowhere, objects fade or slide into view. Meaningful motion was used in transitions between activities. Additionally, a spring-like rebound effect was added to the animation used to snap drawn arrows into place. Also, buttons either slide or fade in and out of position as appropriate.

4.2 Layout

This section discusses the layout of the user interface for the application.

4.2.1 Activities

In simple terms, an activity in Android development is a "screen" where content is held. Typically, an activity takes up all of the space on the device display with the exception of the bottom navigation bar and the top notification bar. Only one activity may be displayed at a time. The user interface of the application is composed of three Activities:

- **MainActivity:** This is the main home screen which allows the user to choose an appropriate problem to solve. A ScrollView is used to accommodate various screen sizes and to create a space for additional problems to be added. If many more problems are added in the future, then a RecyclerView is used instead of a ScrollView because it is much more efficient in memory management. Each problem is contained in a CardView as shown below in Figure 4.5. There is a CardView at the top of the screen which, encourages users to take a brief three question survey. A pulsing animation was applied to the clipboard icon in the CardView to draw attention to it in the hopes that more users would take the survey.

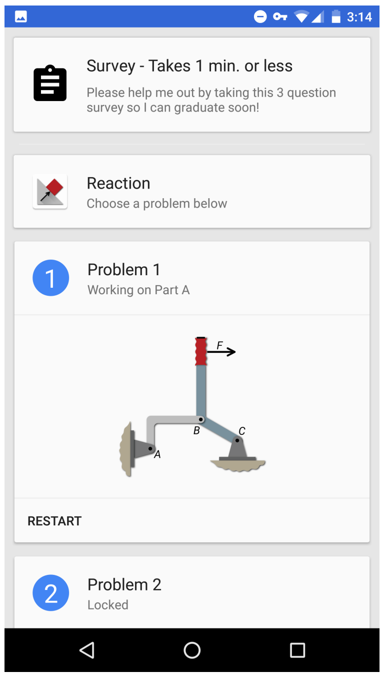
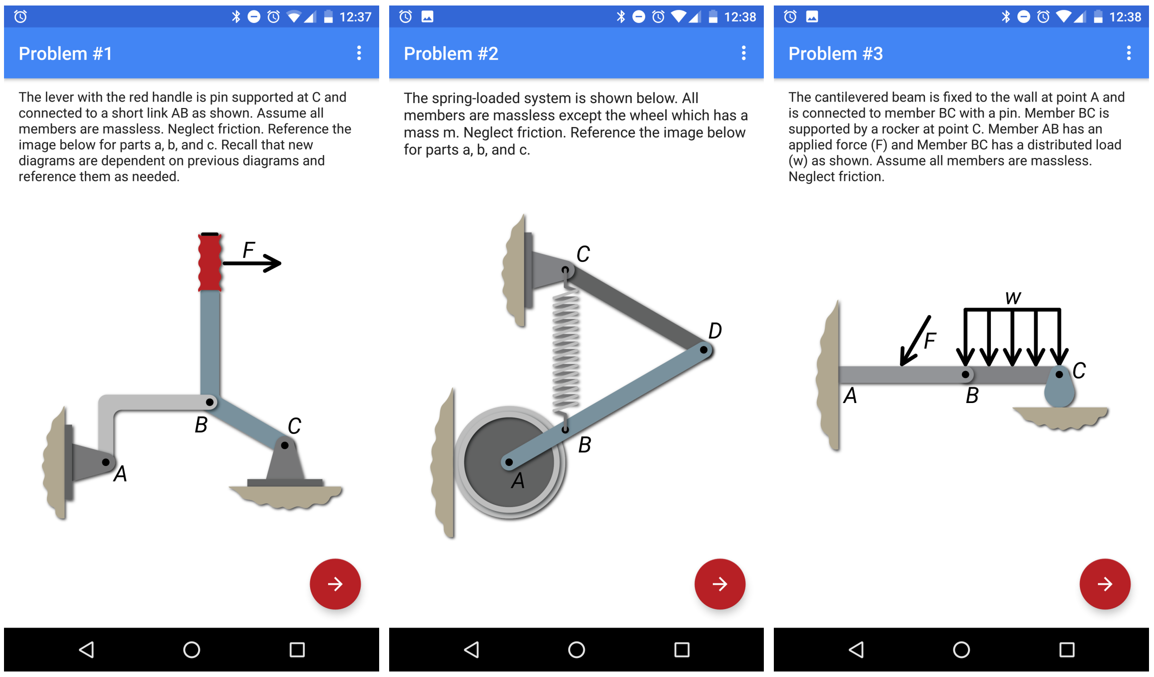

3:14
Survey - Takes 1 min. or less
Please help me out by taking this 3 question survey so I can graduate soon!
Reaction
Choose a problem below
1
Problem 1
Working on Part A
F
A
B
C
RESTART
2
Problem 2
Locked

Figure 4.5: Main Activity

- **SecondActivity:** This is where the main problem statement for each of the main problems was shown. The layout consists of a toolbar at the top of the screen, which displays the current problem number, as shown in Figure 4.6. The problem statement is shown directly below the toolbar. The main problem image is shown below the problem statement. Users click on the red button containing the forward right arrow to progress to the next activity where they will begin drawing a free-body diagram.



The lever with the red handle is pin supported at C and connected to a short link AB as shown. Assume all members are massless. Neglect friction. Reference the image below for parts a, b, and c. Recall that new diagrams are dependent on previous diagrams and reference them as needed.

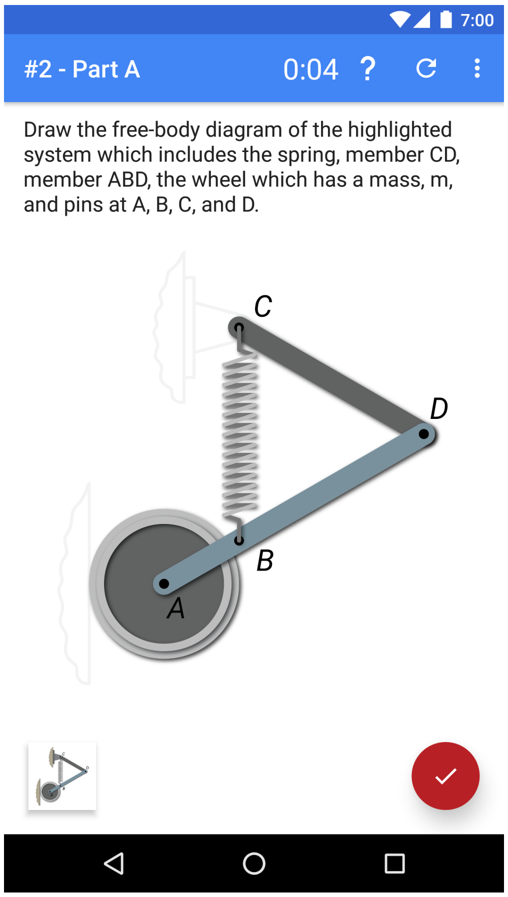
The spring-loaded system is shown below. All members are massless except the wheel which has a mass m. Neglect friction. Reference the image below for parts a, b, and c.

The cantilevered beam is fixed to the wall at point A and is connected to member BC with a pin. Member BC is supported by a rocker at point C. Member AB has an applied force (F) and Member BC has a distributed load (w) as shown. Assume all members are massless. Neglect friction.

(a) (b) (c)

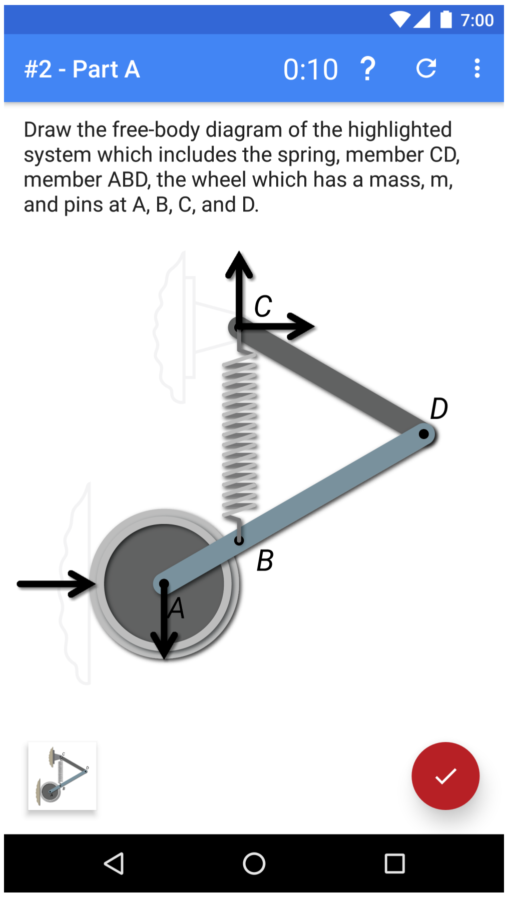
Figure 4.6: SecondActivity

- **ThirdActivity:** This is the main problem statement where users actually draw the free-body diagrams, and is shown below in Figure 4.7. The layout consists of a toolbar at the top of the screen, which displays the current problem number and part letter. The toolbar also features a stopwatch, a help button which loads a tutorial, and a reset button which removes all forces and moments which have been placed. The part statement is shown directly below the toolbar. The main part image is shown below the part statement, and users can draw forces and moments on this image. The buttons in the lower left corner allow the user to quickly view the problem statement and previously completed free-body diagrams. This is useful since the free-body diagrams are dependent on each other. The red button is pressed to check if all arrows have been paced correctly. The ThirdActivity for each problem and the solution is shown in Appendix A.



Draw the free-body diagram of the highlighted system which includes the spring, member CD, member ABD, the wheel which has a mass, m, and pins at A, B, C, and D.

(a)



Draw the free-body diagram of the highlighted system which includes the spring, member CD, member ABD, the wheel which has a mass, m, and pins at A, B, C, and D.

(b)

Figure 4.7: ThirdActivity

4.3 Drawing free-body diagrams

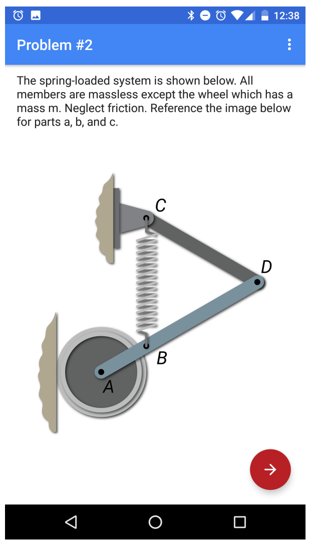
The main functional requirement of the application is for users to be able to practice drawing free-body diagrams. Thus, it is absolutely critical that drawing forces and moments be easy and intuitive.

4.3.1 Problem Images

High quality problem images are carefully constructed in Adobe Illustrator. The images are created from scratch instead of being obtained from textbooks or from online images in order to comply with copyright law and to ensure that the images conform to Google's Material Design specifications.

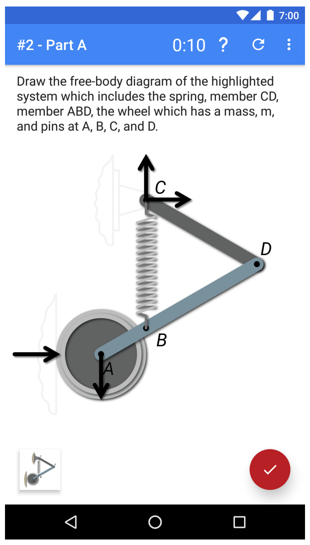
4.3.2 Isolate the Mechanical System

The first step in drawing a free-body diagram is to isolate the mechanical system. Users should isolate the appropriate mechanical system and then draw forces and moments applied at the correct locations and directions. For simplicity, it was decided that the user should be presented with problems where the member is already isolated, instead of trying to isolate them themselves. To accomplish this, users were first presented with the main problem statement and image as shown below in part (a) of Figure 4.8. Next, they were presented with a modified version of the problem image where the members outside the mechanical system are grayed as shown in parts (b), (c), and (d) of Figure 4.8. The grayed out areas are not a part of the mechanical system, but the colored areas are.


Problem #2

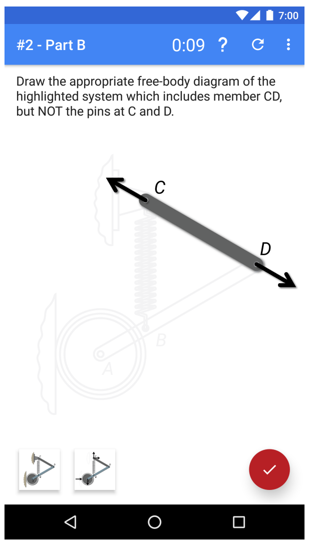
The spring-loaded system is shown below. All members are massless except the wheel which has a mass m. Neglect friction. Reference the image below for parts a, b, and c.

(a)


#2 - Part A

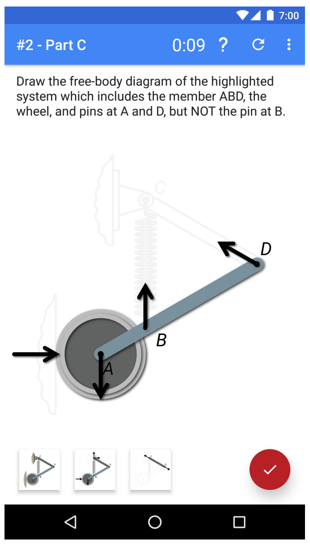
Draw the free-body diagram of the highlighted system which includes the spring, member CD, member ABD, the wheel which has a mass, m, and pins at A, B, C, and D.

(b)


#2 - Part B

Draw the appropriate free-body diagram of the highlighted system which includes member CD, but NOT the pins at C and D.

(c)


#2 - Part C

Draw the free-body diagram of the highlighted system which includes the member ABD, the wheel, and pins at A and D, but NOT the pin at B.

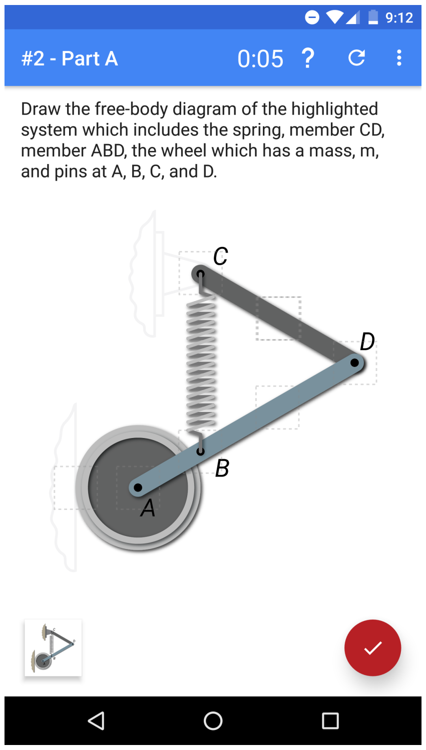
(d)

Figure 4.8: Isolated members

4.3.3 Draw Arrows

Next, the process of how to draw forces and moments was examined. When student draw free-body diagrams on paper they can physically draw forces and moments wherever they desire. For this application, it was decided that it was a bad idea to allow users to draw forces and moments at arbitrary locations on the problem canvas. Without some sort of "snapping mechanism", users would have needed to place the arrows with extreme precision which would have been tedious and frustrating, two things to be vehemently avoided in an application.

Various touch zones were overlaid on each problem. They are shown in Figure 4.9 as light gray dashed boxes. The dashed lines were changed from gray to transparent for production release. When a user touches inside one of these boxes and drags outward, an arrow is drawn from the center of the box to location the user is currently touching on the screen. When the user lifts their finger, the arrow animates inward to a preset length and snaps to 30 degree increments. This 30 degree increment was chosen because it is granular enough that users must make sure they are pointing an arrow in the correct direction, but not so much that the user is constantly trying to the get the arrow aimed in precisely the right direction.



Draw the free-body diagram of the highlighted system which includes the spring, member CD, member ABD, the wheel which has a mass, m, and pins at A, B, C, and D.

Figure 4.9: Touch zones

Free-body diagrams often contain moments, so it is considered important to implement them in the application. After consideration, the “long press” gesture was used. Moments are created by long pressing in touch zones. The arrow head of the moment can be dragged in a circle clockwise or counterclockwise to set the direction of the moment.

4.3.4 Instant feedback

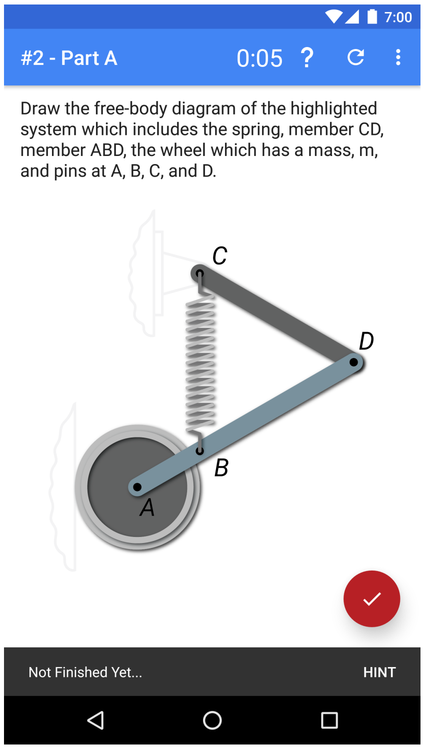
Teachers usually provide feedback to students on homework, tests, lab reports, and quizzes. While this feedback creation can occasionally take as little time as minutes or hours, it often takes teachers days to grade student work and provide feedback. Additionally, students are able to continue to later parts of a problem even after making a mistake, which can reinforce mistakes. The mobile platform is well suited to provide instant feedback to alleviate this delay and reduce the subsequent challenges.

In the application, correctly drawn arrows turn from gray to black. This instant feedback is designed to prevent users from wasting time and reinforcing mistakes. However, providing instant feedback allows users to brute force their way through the problems. One user reported having no background in free-body diagrams but still being able to finish the problems by simply drawing and removing arrows in random direction until only black arrows remain. If users use this "brute force" technique on many problems they may gain skills in drawing free-body diagrams by recognizing patterns. Being able to perform an exhaustive search to determine the correct answer is not feasible in traditional learning platforms because users do not receive immediate feedback.

4.3.5 Alerts

Android features multiple ways to alert users using visual feedback. The three most popular are: [24]

- **Toast:** A toast is a small text overlay that appears over all objects on the screen. Toasts are easy to implement and effective. However, toasts can obscure other objects quite easily, and were therefore rarely used in this application.
- **Snackbar:** The snackbar is a re-imagining of the toast. Unlike toasts, which appear on top of all elements, snackbars slide up from the bottom of the screen, and screen elements can be programmed to slide up so as to not be obscured. Additionally, snackbars can have buttons that make them extremely useful for providing users with a discrete "undo" option. In the application, snackbars are used primarily to alert the user that they have not finished a problem yet. A HINT button is placed in the snackbar shown in Figure 4.10 which users can use if they get stuck.

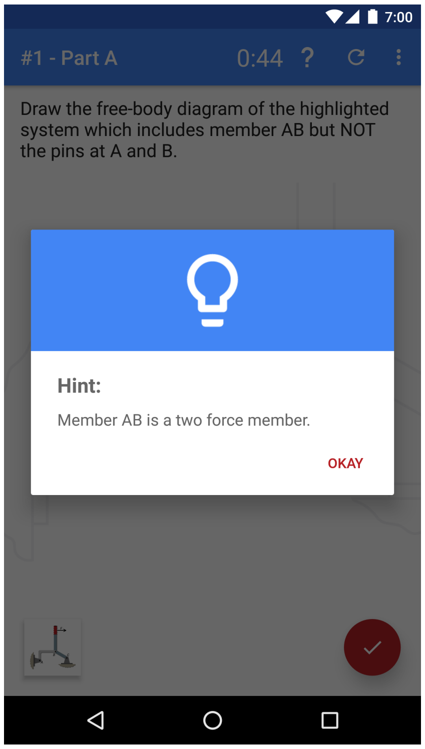


Draw the free-body diagram of the highlighted system which includes the spring, member CD, member ABD, the wheel which has a mass, m, and pins at A, B, C, and D.

Not Finished Yet...

Figure 4.10: Snackbar with button to show hint

- **Dialog:** Dialogs are invasive pop-ups that are overlaid over other views. In the application, dialogs are used to display hints as shown in Figure 4.11 and to alert the user that they have successfully completed the free-body diagram.



Draw the free-body diagram of the highlighted system which includes member AB but NOT the pins at A and B.

Hint:

Member AB is a two force member.

Figure 4.11: Dialog showing hint

Audio feedback was not used in this application since it would limit the use of the app in quiet places such as libraries, a common place for students to study and therefore a potential place the app might be used.

4.4 Language

This application was written in Java since it is the primary programming language supported by the Android platform. The user interface was largely implemented in XML, although some elements are implemented programmatically using Java because of the greater level of control it offers.

4.5 Data storage

A difficult part of the development process is choosing how to efficiently store and load data in the application. There is no “one size fits all” approach to data storage, since each option has advantages and disadvantages. Android provides five main options to persistently store data: [6]

- **Shared Preferences** - Google advises that SharedPreferences be used to store any primitive data including integers, floats, longs, strings, and booleans. Data stored in SharedPreferences persists across user sessions unless specifically cleared by the user or application. The commands to save and load data from SharedPreferences can can be quite tedious. A useful library called TinyDB[1] simplifies calls to SharedPreferences to a single line of code and is used extensively in this project [8].

- **Internal Storage** Android allows applications to store files in the device’s internal memory. These files are private and can only be accessed by the application that writes them and are removed when the application is uninstalled or the data is cleared. Internal storage is used in the application to store bitmaps.

- **External Storage** All Android devices support a shared external storage which can be used to store files. The name can be deceptive, since *external* storage consists of portions of the device’s *internal* solid-state drive (SSD). It also consists of any additional available storage from removable devices such as Secure Digital (SD) cards. The name “external” is used since these files are public and can be edited by other apps or manually by the user using a file browser. Storing files in external storage is useful if other apps will need access them, but it is also risky since other apps could delete or modify them. Additionally, external storage should not be used to store sensitive information since it can be accessed by any application which has the Storage

[1]https://github.com/kcochibili/TinyDB–Android-Shared-Preferences-Turbo

permission. This method of storage is not used in the application since implementing it would have violated the requirement that the app not use additional permissions.

- **SQLite Databases:** Android fully supports SQLite databases, which are lightweight structured databases used for a variety of applications.

- **Network Connection:** A network connection can be used to store data on an external networked database. This method is more complicated than the previous methods of storing data and many issues can arise such as collisions or loss of Internet connection.

After considering each of these options, SharedPreferences and Internal Storage were chosen as the two best ways to store data in the application. Problems are initially stored in internal storage in XML format at installation. After each problem sub part is completed, the location and angles of the arrows are saved into SharedPreferences to be used by future parts as needed. Storing larger chunks of data in SharedPreferences is not recommended by Google, but is used in this application because doing so reduced development time. Future versions of the application may better comply with Google's recommendations regarding SharedPreferences.

4.6 Alleviating privacy concerns

Applications targeting Android 6.0 Marshmallow and higher are required to request permissions from the user. Previously, app permissions were accepted by the user upon installation. The new approach allows users more granular control over what information each app has about them. For instance, in order to access the device's location, an application must first request the location permission from the user. This prevents nefarious apps from mining users' information such as location, reading text messages, etc., without the user's express knowledge and consent. In developing the application, one of the requirements was that no additional permissions would be used. This created challenges, but avoided the privacy concerns that requesting permissions such as Internet access or device location can generate. In order to further alleviate concerns about privacy, use of the app is completely anonymous. The app does not ask users to enter any information such as an email address, a birthday, a nickname, or any other kind of potentially identifying information. Users simply open the app, step through a quick walkthrough and tutorial, and can immediately get started. This alleviates privacy concerns and speeds up the onboarding process.

Chapter 5
Implementation

This chapter describes the various tools and libraries used in the application development and discusses how performance issues are overcome.

5.1 Application Development

Processes used in development include:

- Version control
- A productivity system
- Software libraries
- A debugging protocol

The following subsections describes these processes in more detail.

5.1.1 Version control

Version control is a system of keeping track of changes made to files in a programming project. At key intervals, changes to the code are committed (saved) and pushed (uploaded) to a repository which keeps track of any changes made. Changes to code can break functionality in unforeseen ways and version control allows the code to be reverted to a working version. This project was stored on a free project hosting site called GitHub which implements version control. The code for the application is open-source and can be obtained from the following repository:

`https://github.com/ChrisGaddes/Reaction.git`

5.1.2 Kanban

Although no formal coding productivity system was used such as Agile or Waterfall, a modified version of the Kanban system was used. Kanban is a system used to keep track of and prioritize tasks. A free, online board that can be used as a Kanban board is called Trello. Trello is used to keep track of tasks in development of the application[1]. The Trello board contains following columns:

- **Optional:** This column is used as a repository of interesting but non-critical ideas and features.
- **Important:** This column is used to store tasks that are important enough that they are not considered optional, but not critical in nature.
- **Critical:** This column contains tasks which must be completed in order for the app to meet the requirements.
- **Coming Up:** Tasks from the previous three columns are moved to this column if they will be worked on soon. Tasks should not be moved from important or optional until the critical column is emptied of all tasks.
- **Working On:** This column is reserved for tasks that are currently being worked on. Traditionally, the Kanban system limits the amount of tasks in the column by using work in progress (WIP) limits. Trello does not currently feature WIP limits without the use of additional plugins, so WIP limits are not strictly enforced.
- **Finished:** This column is used to store all finished tasks. Keeping track of completed tasks is useful for documentation purposes and also works as a reminder of past success which can be very encouraging.

The app was rapidly iterated and tested frequently on actual devices and/or emulators in order to verify proper functionality after changes were made. This constant testing often

[1]https://trello.com/b/V2pcQB1i/reaction-app-development

allowed bugs to be detected early and fixed before they became entrenched and more difficult to fix. In extreme circumstances when the solution to newly added bugs could not be found, the code was reverted to a working version using git (Github).

5.1.3 Description of software

This Android project consists of 3,552 lines of code in Java, 7,011 lines of XML, and over 800 lines of comments distributed throughout. This distribution is shown graphically in Figure 5.1.

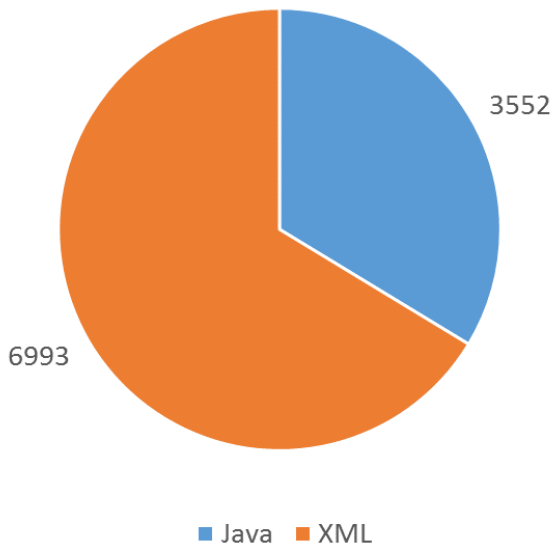


Figure 5.1: Code distribution

The following files comprise the majority of the lines of Java code and lines of XML written specifically for this application:

- MainActivity.java - 677 lines of Java code
- SecondActivity.java - 221 lines of Java code
- ThirdActivity.java - 1,011 lines of Java code

- DrawArrowsView.java - 967 lines of Java code
- activity_main.xml - 601 lines of XML
- activity_second.xml - 70 lines of XML
- activity_third.xml - 226 lines of XML
- prob*_part*.xml - 5,275 lines of XML[1]

Instead of including this large amount of Java code and XML data in this thesis, the aforementioned files and others are hosted in the application project on Github at: `https://github.com/ChrisGaddes/Reaction`

5.1.4 Tools and Libraries

Although it is certainly possible to develop quality Android apps without the use of external libraries and tools, there are many useful resources available to developers, which can greatly speed up development. By utilizing these resources, developers can avoid "reinventing the wheel". Below is a non-exhaustive list of some of the libraries that are used in developing this app:

- **Android Support libraries:**[2] The Android Support Libraries are released by Google and are designed to give developers access to features from newer version of Android on older versions of Android. For instance, Material Design was released in 2014 on Android 5.0 "Lollipop" [11]. However, many Android devices are still running older versions of Android such as Android 4.4 "KitKat" which do not support Material Design. The support libraries add backward compatibility to many new Android features and thus can allow "KitKat" users to experience Material Design applications on their devices.

[1]This XML line count includes nine files incrementing from prob1_parta to prob3_partc
[2]https://developer.android.com/topic/libraries/support-library/index.html

- **Leak Canary:**[1] Android Studio contains tools to monitor memory usage of app real-time during testing. However, Leak Canary is a tool that speeds up the process of tracing down memory leaks. A memory leak occurs when objects in memory are not properly removed (to release memory) once they are no longer needed. Memory leaks can be extremely difficult to locate, but Leak Canary can be helpful in quickly identifying them so they can be fixed.

- **Material Styled Dialogs:**[2] This is a library that simplifies the creation of dialogs.

- **Animations libraries:**[3] This animation library allows developers to quickly add animations to various views. It drastically reduces boilerplate code. For example, the code snippet below fades in "my_view" over a period of 700 ms.

```
YoYo.with(Techniques.FadeIn)
.duration(700)
.playOn(findViewById(R.id.my_view));
```

- **Material Intro View:**[4] This simple library allows developers to create simple on-boarding screens.

- **Material Intro:**[5] Material Intro is a library that allows developers to create app walkthroughs. An app walkthrough is an interactive tutorial which steps the user through how to properly use the app by highlighting various features in succession.

- **Overscroll:**[6] Overscroll is an Android library created to add an "overscrolling" effect to scroll views similar to the effect in iOS. When users scroll to the end of a view, the

[1]https://github.com/square/leakcanary
[2]https://github.com/javiersantos/MaterialStyledDialogs
[3]https://github.com/daimajia/AndroidViewAnimations
[4]https://github.com/HeinrichReimer/material-intro
[5]https://github.com/iammert/MaterialIntroView
[6]https://github.com/EverythingMe/overscroll-decor

library allows them to keep scrolling past the end, but then springs backward to the end of the view once they stop scrolling.

- **Glide:**[1] Glide is a library that simplifies asynchronous image loading.

5.1.5 Debugging

Although it would be ideal if every program is written perfectly the first time and executed flawlessly with the desired results, in reality a healthy portion of any coding project is spent debugging. Many major bugs were discovered and squashed throughout the course of the application development including:

- **Memory leaks** During testing, it was discovered that the app did not seem to be releasing memory properly. As each new problem was loaded, the previous problem would remain in memory. Eventually, this would lead to an Out Of Memory (OOM) error that would cause the application to crash. A tool called Leak Canary is used to find two small leaks involving a handler. However, after these leaks were fixed, there was still a memory leak that Leak Canary did not find. After hours of troubleshooting, the following code in the onCreate method of the custom view DrawArrowsView was found to be causing the leak:

```
    mContext = context;
    thirdActivity = (ThirdActivity) context;
```

  This causes a memory leak since the context is tied to the ThirdActivity, and thus it leaks the resources and the views that are associated with the activity. The context cannot be garbage collected since the activity still has a hold on it. These two lines were actually legacy code that had not been deleted yet. Thus, the leak was easily fixed by deleting them.

[1]https://github.com/bumptech/glide

- **Performance issues:** Initially, the app experienced slow performance while loading new problems. Upon investigation, it was discovered that the app was saving and loading bitmaps synchronously on the main UI thread. Saving and loading bitmaps is very resource intensive and should not be done on the main UI thread. The app performance speed increased once these actions were moved to a separate thread.

  Additionally, early versions of the app used the recreate() method to restart the third activity when switching between problems. This was very time consuming, and was removed since reloading the Activity was not necessary.

- **Bug in Support Library:** There exists a bug in Support Library 24.2.0 which prevents the Floating Action Button from returning to its lower elevation once the Snackbar times out. Since the bug does not exist in prior versions of the support library, early iterations of the app were released with an older support library. Google has fixed the bug in Support Library 24.2.1, so that version is used in later version of the application.

5.1.6 Supporting Multiple Screens

Since Android device screens come in many sizes, aspect ratios, and resolutions, UI elements must be scaled to properly fit each particular screen.

- **Screen Size:** The size of a screen is the physical size of the screen, measured diagonally in the desired unit of length (in., cm., etc.).

- **Screen Density:** Screen density refers to the number of pixels contained in an area of the screen; this is usually denoted in dpi (dots per inch) or ppi (pixels per inch).

- **Orientation:** The orientation of the screen from the point of view of the user. The orientation is either portrait (tall) or landscape (wide).

- **Resolution:** A screens resolution is the number of pixels in its display. This value can be calculated by multiplying the width and height of the display in pixels. For instance, a full HD display resolution is often called 1080p which means it contains 1920x1080 pixels, which equals 2,073,600 total pixels.

- **Aspect Ratio:** A screens aspect ratio is the ratio of its width to height. A 1920x1080 display has an aspect ratio of 16:9. Other common aspect ratios include 4:3, and 3:2.

- **Density-Independent Pixel (dp):** Instead of defining the size of objects in pixels, it is recommended that density-independent pixels be used instead. These virtual pixel units allow objects to remain constant in size regardless of the screen size on which they are shown [7].

- **Scaleable Pixel (sp):** Android allows users to choose the relative text size of text in all apps. Scaleable pixels are used to accomplish this [4]. Stack Overflow user Bachi[1] wrote the following methods to quickly convert back and forth between density-independent pixels and pixels:

```java
// converts dp to pixels
private int dpToPx(int dp) {
    DisplayMetrics displayMetrics = getContext().getResources().getDisplayMetrics();
    return Math.round(dp * (displayMetrics.xdpi / DisplayMetrics.DENSITY_DEFAULT));
}

// converts screen percentage to pixels
private Point percentToPx(PointF per) {
    return new Point(Math.round(per.x * viewWidth / 100), Math.round((per.y * viewHeight / 100)));

```

[1]http://stackoverflow.com/a/17410076/6388083

5.1.7 Custom Views

Android allows the creation of custom views, which is useful when the default views do not meet the specific needs of the developer. Four major custom views are used in this application:

- **DrawArrowsView** - This custom view contains the canvas on which the arrows are drawn.
- **ChronometerView**[1] - A custom stopwatch view created by Stack Overflow user CommonsWare is used to visually denote the progress of time as the user solves each problem. The custom view is displayed in the toolbar, which is the blue bar at the top of the Android application.
- **AutoResizeTextView**[2] - This custom view created by Stack Overflow user Sam Hocevar is a TextView that scales the text size to fit the dimensions of the TextView. This is used to force the text for the problem statements to fit on the screen regardless of the screen size of the device or the chosen text size in Android settings.
- **SquareImageView**[3] - This simple custom view written by GitHub user Antonio Leiva restrains the contained image within a 1:1, or square, aspect ratio.

5.2 Performance

A common complaint among mobile app users is that apps have slow performance. Great effort was made in the design of this app to ensure good performance, even on low end devices.

[1]http://stackoverflow.com/a/31120878/6388083
[2]http://stackoverflow.com/a/5535672/6388083
[3]https://git.io/vPcvn

5.2.1 Drawing

The invalidate() method, which is used to force the UI to redraw a view, is used as sparingly as possible so as to keep performance high. AsyncTask and Handlers are used for long (but still less than a couple seconds) tasks. A splash screen is shown as the app is first loading in order to communicate to the user that the app is indeed responding. This is preferable to a blank white loading screen that would have been otherwise presented.

5.2.2 Image loading

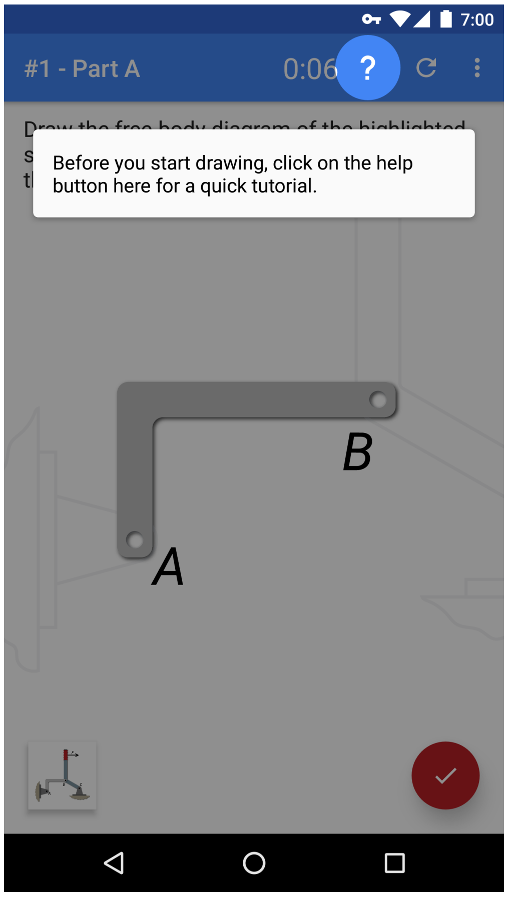
In Android, commands can be executed synchronously or asynchronously. When commands are executed synchronously, the device waits until the command is completed before executing the next command. Conversely, asynchronously executed commands can be executed simultaneously. Image loading is resource intensive and often cannot completed instantaneously. Loading an image can cause noticeable lag if the execution is done synchronously on the "main" thread, the thread responsible for updating the user interface. This is especially noticeable on low end devices or while loading large images. In order to eliminate this problem, images should be loaded asynchronously. The Glide image loading library is used in this app to load images asynchronously so as to not freeze up the main UI thread. Additionally, images are used at the appropriate resolution for the users specific device so as to not use up unnecessary memory.

5.2.3 Onboarding

In the context of mobile app development, the onboarding process is a term used to describe the mechanism through which users are taught how to use the application. Despite efforts to make the application intuitive, using it properly can be daunting to a new user. As such, a three part onboarding process was implemented as requested in the requirements.

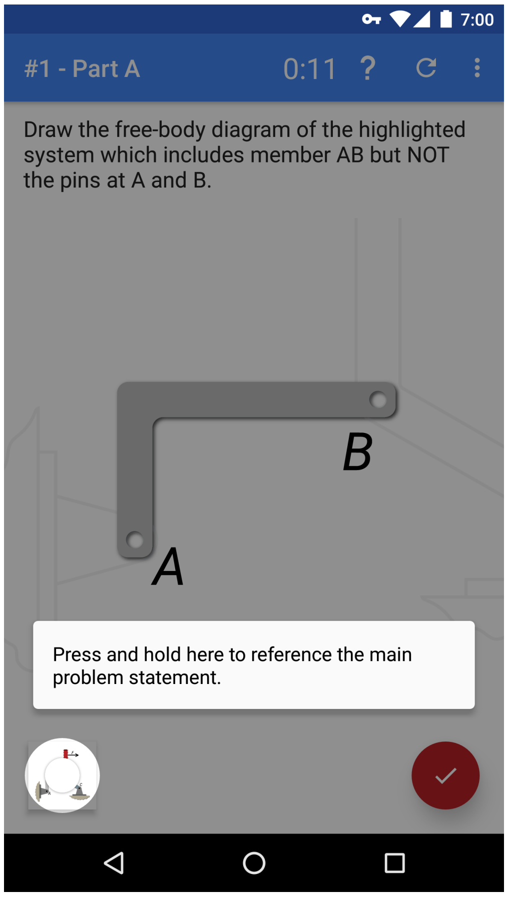
1. **Intro view:** When the app is first opened after installation, a brief intro view is shown which welcomes the user to the app. It informs the user that the app is a proof of concept and explains that there are only three problems for now.

2. **Interactive Walk-through:** Secondly, the user is presented with a white dialog box by a focused white pulsing dot as shown below in Figure 5.2 and Figure 5.3. The user is presented with several of these dialogs which explain various important aspects of the app. The dialogs are created using a library called Intro-View[1], which is inspired by an app called Fabulous.



Before you start drawing, click on the help button here for a quick tutorial.

Figure 5.2: Walkthrough: Shows where to tap help button



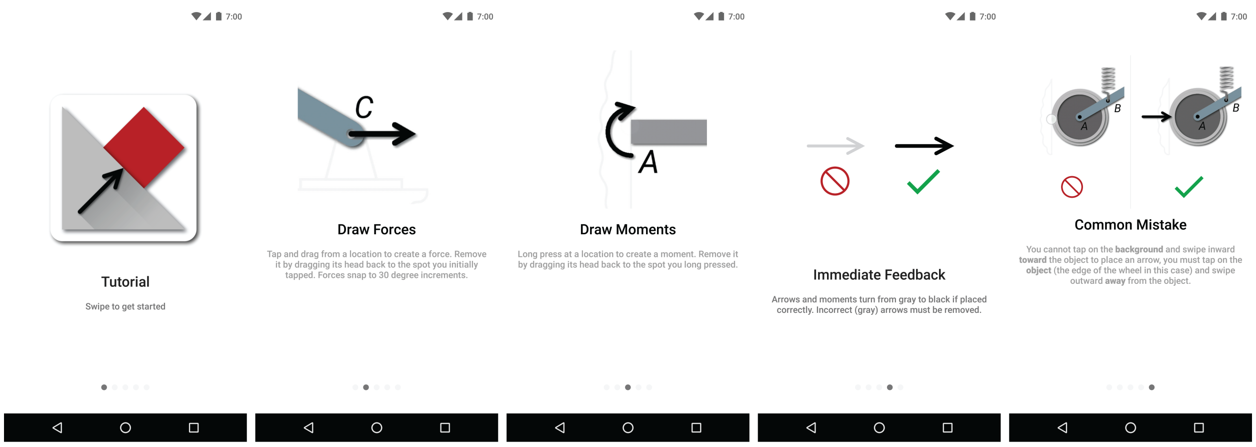
Draw the free-body diagram of the highlighted system which includes member AB but NOT the pins at A and B.

Press and hold here to reference the main problem statement.

Figure 5.3: Walkthrough: Shows where to peek button

[1]https://github.com/iammert/MaterialIntroView

3. **Tutorial:** Thirdly, a tutorial view is presented to the user, which shows animated GIFs demonstrating how to draw force arrows and place moments correctly on screens 2 and 3 respectively. It also mentions that correctly placed arrows turn from gray to black on screen 4, and alerts the user to a common mistake on screen 5 of the tutorial. Figure 5.4 shows each of the screens in the tutorial from left to right.



Tutorial

Swipe to get started

Draw Forces

Tap and drag from a location to create a force. Remove it by dragging its head back to the spot you initially tapped. Forces snap to 30 degree increments.

Draw Moments

Long press at a location to create a moment. Remove it by dragging its head back to the spot you long pressed.

Immediate Feedback

Arrows and moments turn from gray to black if placed correctly. Incorrect (gray) arrows must be removed.

Common Mistake

You cannot tap on the **background** and swipe inward **toward** the object to place an arrow, you must tap on the **object** (the edge of the wheel in this case) and swipe outward **away** from the object.

Figure 5.4: Tutorial: Shows where to tap help button

5.2.4 Problem Data

The data for each problem is stored in XML format. XML data is easy to parse, well supported in Android, and also relatively easy for humans to read. Since the touch locations are different for each part of each problem, the following steps are used to speed up the process:

1. Problems are drawn in Adobe Illustrator on a square canvas. These problems are exported as PNG files and imported into the Android Studio project.

2. Back in Illustrator, an orange circle is precisely drawn over the center of each desired touch zone location as shown below in Figure 5.5. The problem image is then removed, leaving only the orange circles. This image, shown in Figure 5.6, is then exported from Illustrator as a Scalable Vector Graphics (SVG) file.

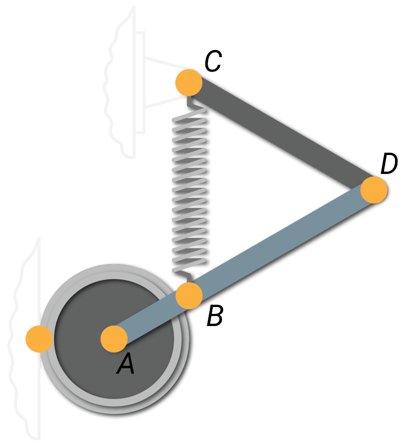


Figure 5.5: Orange Dots at Points

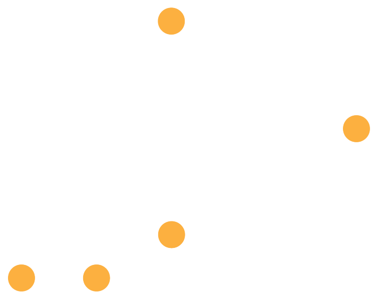


Figure 5.6: Orange Dots at Points

3. The SVG file is then imported into a Excel spreadsheet, which automatically extracts the center point locations of the orange circles from the SVG file. The spreadsheet then placed these locations in the correct XML format to be used in the program.

5.2.5 Logging usage time

Google Analytics is not used for privacy reasons, so the time that the app is actively used by each user is calculated by recording the system time in miliseconds at the onResume and onPause methods. These methods are called when an activity is started (or resumed) and stopped (or paused) respectively. The difference between these two times is added to a total time value, which was stored in SharedPreferences using TinyDB.

When users click on the button to take the survey, this total time value is automatically copied to the users clipboard in order to streamline the data logging process. Then, the user simply pastes the supplied value into the appropriate field in the survey.

5.2.6 Data collection and feedback

TO preserve user privacy, all data is collected anonymously through a Google Forms Survey.

The survey asks three questions:

1. *In your experience, how useful are mobile apps for engineering education?*
2. *How useful did you find this particular app?*
3. *Paste time value* - in this location the user is asked to paste in the total time value that they have used the app. As described earlier, this value is automatically copied to their clipboard.

Additionally, a comments box is included. All questions are optional, so users are free to not complete the survey, for any reason.

Chapter 6

Evaluation and Testing

This chapter discusses the specs of the final product, application testing, how the application requirements are met, and discusses user feedback.

6.1 Application Testing

The following devices were emulated to test the application:

- Nexus 4 running Android 4.4.4
- Nexus 6P running Android 6.0
- Nexus 9 running Android 6.0.

These devices were chosen because they cover many popular screen sizes and aspect ratios [16]. Emulators are useful for testing user interface layouts, but they often run significantly slower than physical devices, so they are not very useful in evaluating real-world performance. Physical devices should be used to evaluate applications for performance concerns. The application was tested on the following physical devices during development and testing:

- Nexus 6P running Android 7.0
- Nexus 5 running Android 6.0
- Nexus 9 running Android 7.0.

6.2 Specs of the final product

The devices above were used to develop the following specifications for the application.

- **150 MB RAM:** During testing, the random access memory (RAM) usage for the application peaked at approximately 150 MB. Any device that has 150 MB of RAM available to the application should have sufficient RAM to run the application.
- **Touchscreen:** The device must have a touch screen for input.
- **Android 4.4 or higher:** The device must be running Android 4.4 KitKat or higher.

Specifications such as minimum processor speed are not calculated, but all the Android devices which the device was tested on have had sufficient processing capability.

6.3 Requirements Testing

Each of the functional and non-functional requirements was examined in turn

- **Work offline:** This requirement was met since the application does not require Internet access for operation.
- **Request no permissions:** This requirement was met since the application does not request any permissions from the user.
- **Log usage time and report feedback:** This requirement was met since the application logs usage time and the user can both export this time and provide feedback via an external survey.
- **Link to survey:** This requirement was met since the application links to an anonymous survey which the user can open using a web browser of their choice.
- **Tutorial:** This requirement was met since the application contains a tutorial which walks the user through basic usage.

- **Easy to use:** This requirement is difficult to quantify. Some users have found the app easy to use, and other have found certain aspects of the app, namely placing moments, to be challenging. However, the majority of feedback has been positive in regards to ease of use so this requirement will be considered "mostly" met.

- **No bugs:** This requirement was not met fully since there is one minor bug which has not yet been fixed at time of writing. Users are allowed to "stack" moments on top of each other, and all the moments will turn black (as if showing they are correct), but these are not correct entries.

- **Run on Android 4.4 and higher:** This requirement was met since the application runs on devices running Android 4.4 "KitKat" or higher.

- **Good performance on low end devices:** The application was tested on low end devices such as the Moto E, Moto G and the Nexus 4 and it experienced no performance issues. Thus, this requirement was met.

- **Responsive:** This requirement was met since there is no noticeable lag between a touch input and the resulting visual feedback caused by it.

- **Support common screen sizes:** This requirement was met since the application should supports common screen sizes and aspect ratios.

- **Modular code:** This requirement was met since the application code is modular in nature and additional problems can be added with relative ease.

6.4 Metrics

Google provides developers with basic metrics via the Google Play Developer Console. From September 10th to October 10th, 2016, the application has been downloaded a total of 185 times, and uninstalled a total of 60 times which puts the number of current installations around 125. These values are shown below in Figure 6.1 and Figure 6.2. The 60+ spike in installations around Tuesday corresponds to having posted about the application on Reddit.

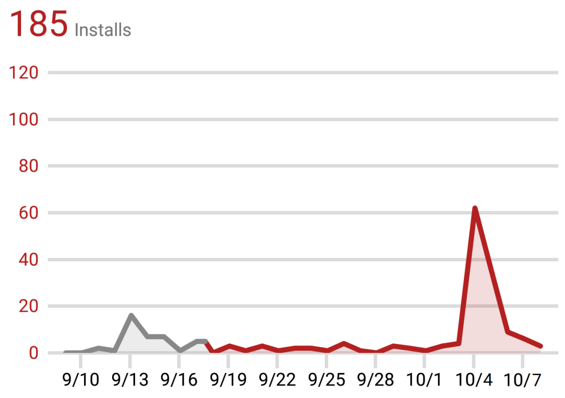


Figure 6.1: Daily Installs in 2016

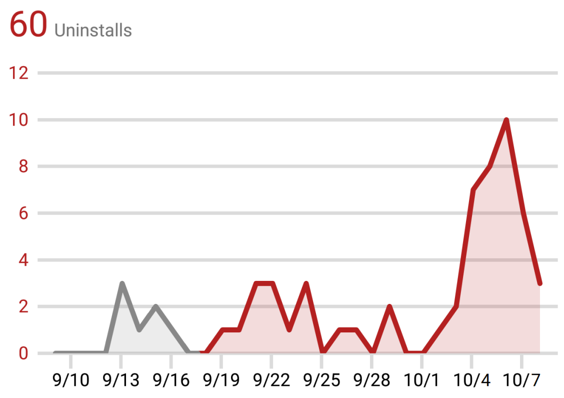


Figure 6.2: Daily Uninstalls in 2016

6.5 User Feedback

All thirteen reviews on the Google Play Store are 5-Star ratings as shown in Figure 6.3. The application is the #1 search result for the phrase “free-body diagram” on the Google Play Store.

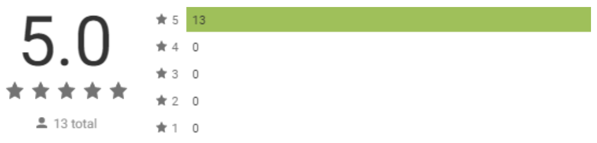


Figure 6.3: Reaction: Free-body Diagrams

In comparison, the Google Play Store Ratings for the free-body diagram related applications listed in Chapter 3.1 are shown below:

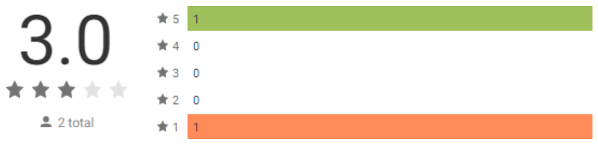


Figure 6.4: Grade AP Physics: Equilibrium

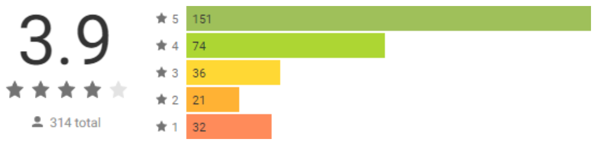


Figure 6.5: Mechanics - Physics

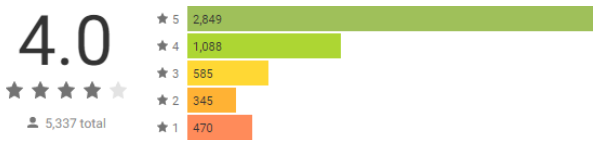


Figure 6.6: Learn Physics

6.5.1 Survey Feedback

At the time of writing, fifty people have taken the survey, which means that 27% of the 185 users who installed the application took the survey.

The survey asked just three questions. For the first two questions the users chose their answer on a scale from one to five. The first question was: *In your experience, how useful are mobile apps for engineering education?* The average answer was 4.14. The results are shown below in Figure 6.7.

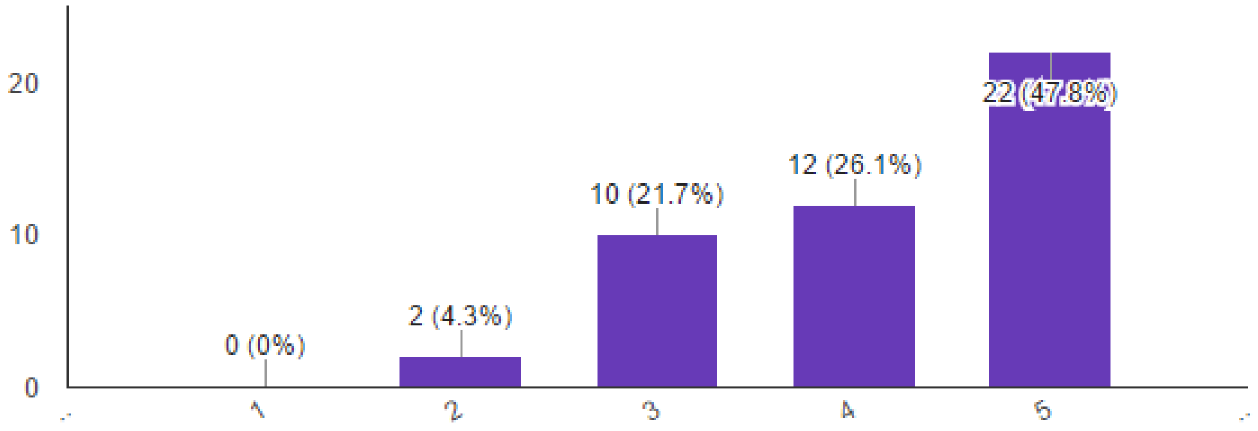


Figure 6.7: Question 1 Results

The second question was: *How useful did you find this particular app?* The average rating was 4.10. The results are shown below in Figure 6.8.

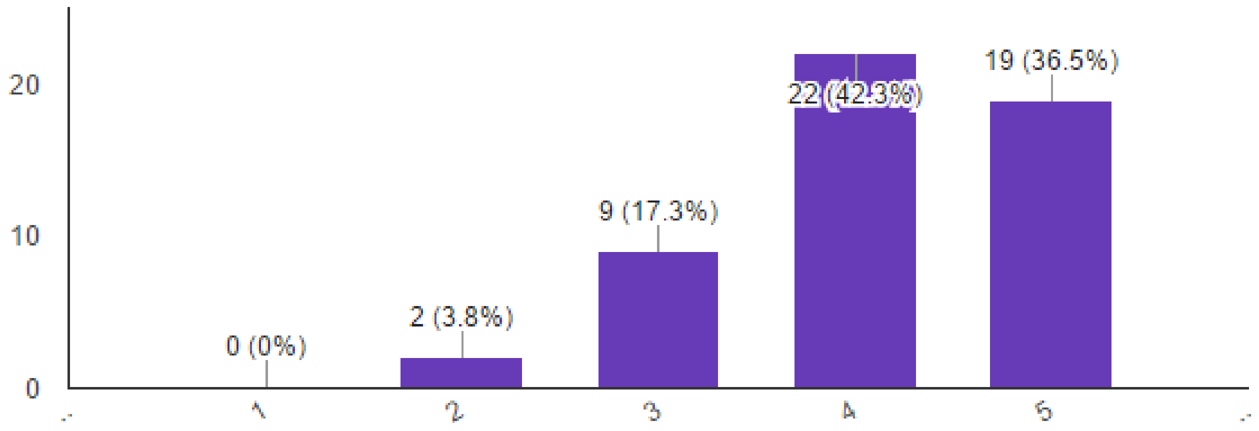


Figure 6.8: Question 2 Results

Interestingly, the average value for the second question is only slightly lower than the value of the first question. These preliminary results may suggest that users may have little experience with engineering apps, so experience with this application may effect the answer to question 1.

The third question asked users to paste in their app time usage in seconds which the app had automatically copied to their clipboard when they clicked on the survey link. The survey reported that the average user used the application for 13.42 minutes. Some users took less time than this, while other users took significantly longer. Some of the reported time was extremely low, less than a minute in some cases. It is likely that these users did not finish all three problems in such a short time.

Many of the fifty users responded with additional comments. Some of these are categorized and listed below.

Some of the comments are positive, but vague:

- “Great job”
- “Swag”
- “Make some more good [expletive]”
- “You’re awesome!”

Other comments praised the applications user interface, its ease of use, its value for practicing, and expressed desire for more engineering educational applications:

- “Very nice! Clean UI and detailed instruction.”
- “Excellent tutorial animations. The graphic design in general is awesome. Keep up the good work. ”
- “Great app, easy to comprehend and use.”
- “Looks like a great learning tool. I just wish I could remember statics better!”
- “The hints were good at helping without giving away the solution. Learning how to do free body diagrams is all about repetition and this app provided that opportunity. I wish I had this for my statics class. ”
- “Great practice, I wish there were more apps like this to interactively work on understanding engineering material”

A common request was for more problems:

- “Delightful app, I’d like to see more problems!”
- “I wish I could create my own problems”
- “Really great app, would definitely pay for it when it has more problems :-)”

Other comments contained constructive criticism or suggestions:

- “It’s not always clear which parts of the system we’re supposed to make vectors for”
- “Difficult to see gray arrows against the gray wheel”
- “Need to incorporate better hints and possibly solutions after a number of attempts (if the point is to teach).”
- “I think it is a good test of knowledge, and an okay study tool but I don’t really think it does a lot to teach. Maybe there could be an option that explains why the answer is what it is. Also, it would be good if the hint checked what you’re missing to help

you as opposed to being standardized. There were some points where I was making a simple mistake that I couldn't find and the hint told me about moments instead of the horizontal force"

- "I couldn't figure out some of the mechanics. Maybe a few bugs left? What I thought were the right answers didn't seem to work."
- "Maybe add an eraser function"

These comments are useful in analyzing the popularity of the application. The comments are predominantly positive, which aligns with the numerical feedback in the survey. These preliminary results suggest that the smartphone platform is a popular option among students for supplemental engineering education.

Chapter 7

Conclusions

This chapter summarizes the work done in developing the application. It also discusses ideas for future work, and makes concluding remarks.

7.1 Summary of work

An Android application called *Reaction: Free-Body Diagrams* is developed as a supplemental tool for engineering education. Since the developer had no prior training in Android development, various tutorials and online courses were used to develop rudimentary development skills. The finished application allowed users to practice drawing free-body diagrams for three problems, each with three sub parts. Users were asked to complete a brief three question survey. The results of this survey were analyzed to gauge the popularity of the mobile platform as a supplemental learning tool.

7.2 Future Work

Improvements that could be made to the application in the future are suggested below:

7.2.1 Add more problems

Perhaps the most obvious way to improve the usefulness of this app is to add more problems for the user to practice. Specifically, easier problems could be added at the beginning which progress to more difficult problems. If the app contained a large enough database of problems of varying difficulties, problems could be specifically tailored to the user. For example, if the app recognizes that a user is struggling to identify two force members, it would present more problems containing two force members to train them to properly identify them.

7.2.2 Community driven Framework

Although the developer could enter all the problems manually by hand, it would be useful to have a simple framework with which professors or the community as a whole could add new problems. This would allow the app to scale quickly into a comprehensive training free-body diagram practice app. Additionally, problems could each be tagged as containing key tricky components such as two-force members and the problems would be weighted on a difficulty scale. The community could then use a Reddit style upvoting and downvoting system to set the difficulty of a problem to its appropriate level.

7.2.3 Improve feedback and hints

The hints that are provided to the user for each part are predetermined and are not affected by the users current arrow placements. Intelligence could be added to tailor the hints to the users current situation. For instance, if the user is forgetting to place a gravitational force on a wheel of mass m, the hint should say something like "remember, not all members are massless" not a generic and potentially useless hint.

7.2.4 Add values to force arrows and lengths

The current version of the application considers all forces to be the same in that they are all un-labeled and of the same magnitude. It would be useful to allow the users to visually set the magnitude of the force so it can properly reflect the value of the force relative to other forces. Additionally, forces should be labeled appropriately. For instance, g refers to the gravitational force, F_n refers to a normal force, etc.

7.2.5 Improve drawing

Once of the primary challenges with drawing free-body diagrams on a tiny phone screen is that the touch points and arrows can be quite small and therefore difficult to precisely touch. Implementing a pinch-to-zoom feature would allow users to zoom in on points to make it easier to precisely place arrows.

The long press gesture to create moments is not intuitive and must be taught. Future work should include investigating a more intuitive way to place moments. Additionally, many users have reported issues with placing moments. It is often tricky to remove them, and they interfere with force arrows at times.

7.2.6 Add more teaching to the app

The app is designed to be used by people who are already familiar with drawing free-body diagrams. However, teaching could be added to the app which teaches newcomers the fundamentals of drawing free-body diagrams and then progresses them through increasingly challenging problems until they reach proficiency.

In addition to teaching the basics, the app could teach some of the tricky subtleties that must be learned to draw free-body diagrams correctly. For instance, many users have experienced difficulty in recognizing the subtle difference that including or not-including the pins of a member can have. Instead of the generic hints in the current version, the app should give feedback that is tailored to the mistakes the user is making.

7.2.7 Analytics

Google Analytics, a free analytics tool provided by Google, allows developers to glean a great deal of interesting data from users such as automatically logging the time they are in each activity of the app, among other things. This data would be more granular than the time data received via the Google Forms survey. Additionally, analytics data would be obtained from all consenting users, not simply the ones who volunteer to take the survey.

The use of Google Analytics raises privacy issues, and would require approval from the Institutional Review Board (IRB) to be used for any university research purposes. However, implementing Google Analytics or a similar data logging system might generate valuable data that could lead to improvements to the educational platform.

7.2.8 Gamification

Many popular apps use gamification. Gamification is the concept of turning an activity into a "game" to increase its enjoyability[14]. Gamification means adding typical elements from games such as scoring and competition into a platform that is not traditionally considered a game. For instance, the language learning app Duolingo will let the user "level up" once they complete a certain amount of the educational material in the app [1]. The application could be "gamified" by allowing the "level up" after each correctly completed problem. Additional points could be gained by completing the problems rapidly or with less hints.

7.2.9 Teach other skills

The mobile platform could be used to teach many engineering skills. For instance, students could practice solving circuits problems on a similar mobile application. The app could be tightly integrated with a Learning Management Program such like Canvas and licensed to professors as a supplemental learning platform for a wide variety of engineering curricula.

7.3 Concluding Remarks

The characteristics of "traditional education" such as in-class lectures in "brick and mortar" classrooms, attending office hours, doing homework, taking quizzes and tests on paper, reading textbooks, and often performing lab work, are unlikely to be replaced in the near future. As mentioned previously, the intention of this thesis is not to suggest that these traditional education techniques such as these should be eliminated, but to discuss the popularity of supplementing these techniques with mobile learning. Although small in scale, the feedback during this experiment has been overwhelmingly positive. The application which was created is not perfect, and there are certainly improvements which can be made. However, the positive feedback for this educational application suggests that the smartphone platform is a popular option among students for supplemental engineering education.

7.4 Personal Comments

When I took Statics and Dynamics at Auburn University in the fall of 2012, I spent many hours each week working homework problems in the office of my professor, Dr. Madsen. When a student would come to him with a homework related question, Dr Madsen would ask them to go to the large whiteboard in his office, where they would attempt to work through the homework problem one step at a time. If the student made a mistake, Dr. Madsen would ask them leading questions such as “why did you place a force there?” The student would often recognize their mistake in light of these questions, and correct it accordingly. Conversely, if students completed a step correctly, they would hear audible feedback from Dr. Madsen such as “exactly” or “correct”. This immediate feedback loop allowed students to learn difficult concepts without wasting hours “spinning their wheels” on concepts that a professor could answer in seconds. Although there is value in grinding away at difficult problems, often the issues on which students get stuck are trivial in nature. I personally found the immediate feedback in Dr. Madsen’s teaching approach in his office to be extremely effective, and largely contribute my A in the class to it.

As powerful as this teaching approach may be, it is scales badly. My Statics and Dynamics class contained over a hundred engineering students. If each of these students had visited his office as frequently as I, Dr. Madsen would have been absolutely overwhelmed. The personalized learning I experienced in Dr. Madsen’s office is the inspiration for *Reaction*. I set out to develop a learning platform which implements the personalized and immediate feedback offered by a professor or tutor, and combines it with the scalability offered by the ubiquitous smartphone platform. The result is admittedly flawed, and quite preliminary, but it is an interesting proof-of-concept. The data collected in this project, both empirical and anecdotal, suggest that there is a large market for well designed mobile applications which allow students to practice engineering problems, and that these applications have the potential to revolutionize the field of engineering education.

Bibliography

[1] Making Duolingo –Blog. http://making.duolingo.com/which-countries-study-which-languages-and-what-can-we-learn-from-it#fn:courses.

[2] Material Design Awards 2015. https://design.google.com/articles/material-design-awards/.

[3] Smartphone OS sales market share Kantar Worldpanel ComTech. http://www.kantarworldpanel.com/smartphone-os-market-share/.

[4] Units and measurements - Layout. https://material.google.com/layout/units-measurements.html#units-measurements-scaleable-pixels-sp.

[5] Purchase and Activation - Support - Apple Developer. https://developer.apple.com/support/purchase-activation/, 2016.

[6] Storage Options — Android Developers. https://developer.android.com/guide/topics/data/data-storage.html, 2016.

[7] Supporting Multiple Screens — Android Developers. https://developer.android.com/guide/practices/screens_support.html, 2016.

[8] TinyDB. https://github.com/kcochibili/TinyDB--Android-Shared-Preferences-Turbo, 2016.

[9] Ron Amadeo. Google's iron grip on Android: Controlling open source by any means necessary. http://arstechnica.com/gadgets/2013/10/googles-iron-grip-on-android-controlling-open-source-by-any-means-necessary/, October 2013.

[10] Android. Get Started with Publishing — Android Developers. https://developer.android.com/distribute/googleplay/start.html, 2016.

[11] Matt Brian. Google's new 'Material Design' UI coming to Android, Chrome OS and the web. https://www.engadget.com/2014/06/25/googles-new-design-language-is-called-material-design/, June 2014.

[12] Baiyun Chen, Ryan Seilhamer, Luke Bennett, and Sue Bauer. Students' Mobile Learning Practices in Higher Education: A Multi-Year Study. http://er.educause.edu/articles/2015/6/

students-mobile-learning-practices-in-higher-education-a-multiyear-study, June 2015.

[13] Helen Crompton. A historical overview of mobile learning: Toward learner-centered education. *Handbook of mobile learning*, pages 3–14, 2013.

[14] Juho Hamari, Jonna Koivisto, and Harri Sarsa. Does Gamification Work? –A Literature Review of Empirical Studies on Gamification. In *2014 47th Hawaii International Conference on System Sciences*, pages 3025–3034, January 2014.

[15] R. C. Hibbeler. *Engineering Mechanics. Statics.* Pearson, Upper Saddle River, N.J, 13th ed edition, 2013.

[16] Dave Kearney. The mobile resolutions to be aware of when designing. http://blog.fluidui.com/the-mobile-resolutions-to-be-aware-of-when-designing/, September 2016.

[17] Anna Lisnyak. Flat Design vs. Material Design: How Are They Different? http://designmodo.com/flat-vs-material/, April 2015.

[18] Monica Anderson. Technology Device Ownership: 2015, 2015-10-29T09:50:51+00:00.

[19] Fezile Ozdamli and Nadire Cavus. Basic elements and characteristics of mobile learning. *Procedia - Social and Behavioral Sciences*, 28:937–942, January 2011.

[20] S. Pal, S. Mukherjee, P. Choudhury, S. Nandi, and N. C. Debnath. M-learning in university campus scenario - Design and implementation issues. pages 1851–1856. IEEE, February 2013.

[21] K. Pant, H. Kaur, and M. Sidhu. Ergonomics Evaluation of Various Risk Factors Associated With Carrying School Bags. *International Journal of Scientific Research*, 5(2), April 2016.

[22] Steve Rose. Why Apple ditched its skeuomorphic design for iOS7. *The Guardian*, June 2013.

[23] Adam Sinicki. Developing for Android vs developing for iOS - in 5 rounds. http://www.androidauthority.com/developing-for-android-vs-ios-697304/, 2016-06-09T05:36:04-04:00.

[24] Joanna Smith. Snackbar: The appropriate interruption. https://medium.com/google-developers/snackbar-the-appropriate-interruption-ceb54d9be583#.pvwnnvwg8, January 2016.

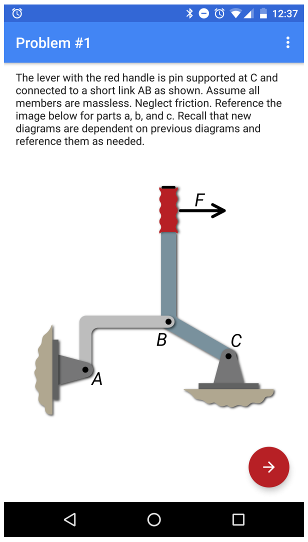
[25] Ian Sommerville. *Software Engineering*. International computer science series. Addison-Wesley, Harlow, England ; New York, 8th ed edition, 2007.

[26] Amber Leigh Turner. The History of Flat Design: Efficiency, Minimalism, Trendiness. `http://thenextweb.com/dd/2014/03/19/history-flat-design-efficiency-minimalism-made-digital-world-flat/`, 2014-03-19T15:09:01+02:00.

Appendices

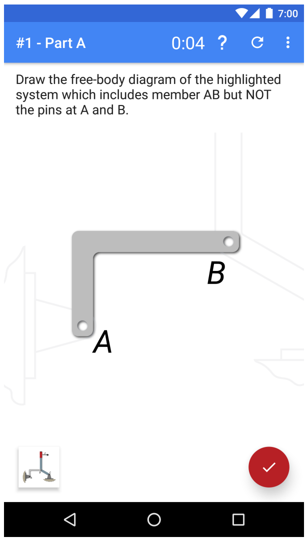
Appendix A

Problems and solutions



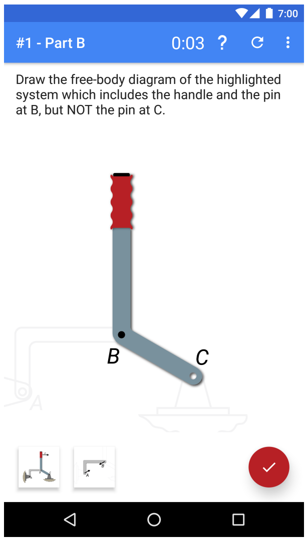
The lever with the red handle is pin supported at C and connected to a short link AB as shown. Assume all members are massless. Neglect friction. Reference the image below for parts a, b, and c. Recall that new diagrams are dependent on previous diagrams and reference them as needed.

(a)



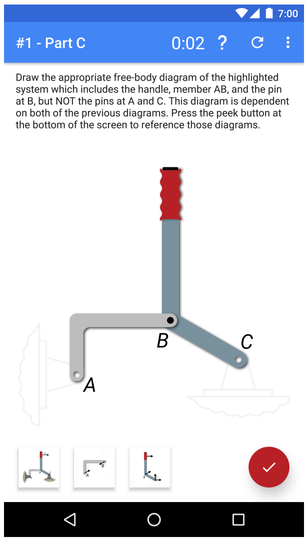
Draw the free-body diagram of the highlighted system which includes member AB but NOT the pins at A and B.

(b)



Draw the free-body diagram of the highlighted system which includes the handle and the pin at B, but NOT the pin at C.

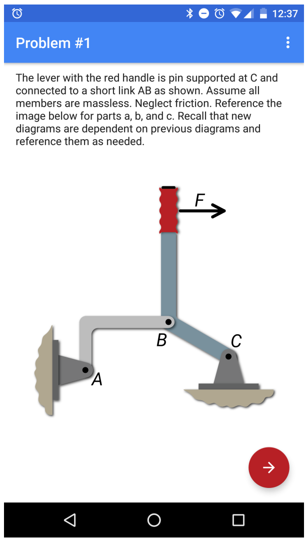
(c)



Draw the appropriate free-body diagram of the highlighted system which includes the handle, member AB, and the pin at B, but NOT the pins at A and C. This diagram is dependent on both of the previous diagrams. Press the peek button at the bottom of the screen to reference those diagrams.

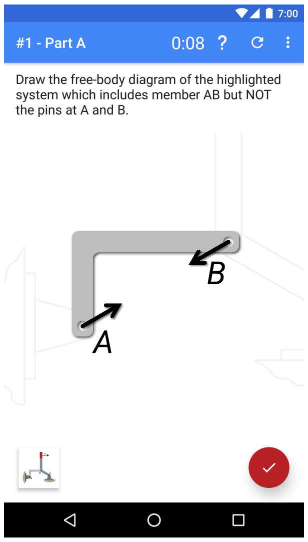
(d)

Figure A.1: Problem 1



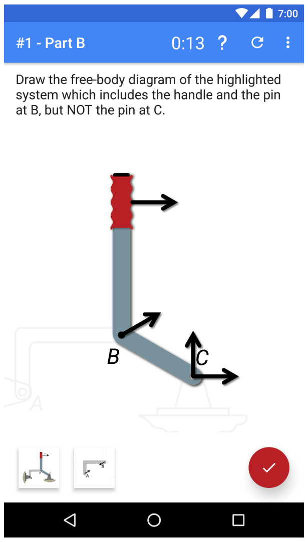
The lever with the red handle is pin supported at C and connected to a short link AB as shown. Assume all members are massless. Neglect friction. Reference the image below for parts a, b, and c. Recall that new diagrams are dependent on previous diagrams and reference them as needed.

(a)



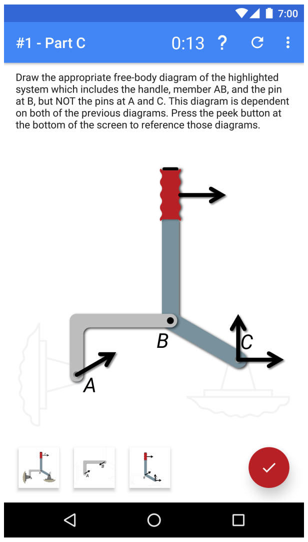
Draw the free-body diagram of the highlighted system which includes member AB but NOT the pins at A and B.

(b)



Draw the free-body diagram of the highlighted system which includes the handle and the pin at B, but NOT the pin at C.

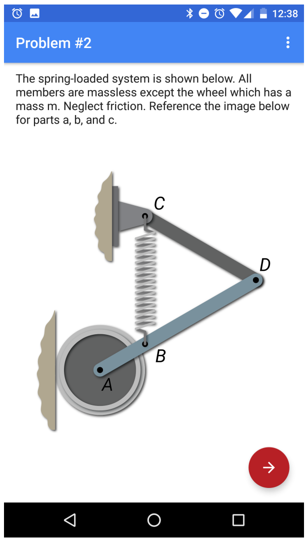
(c)



Draw the appropriate free-body diagram of the highlighted system which includes the handle, member AB, and the pin at B, but NOT the pins at A and C. This diagram is dependent on both of the previous diagrams. Press the peek button at the bottom of the screen to reference those diagrams.

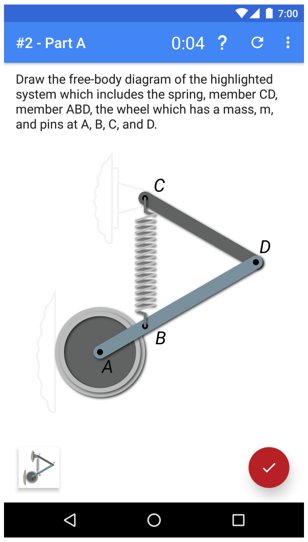
(d)

Figure A.2: Problem 1 with Arrows



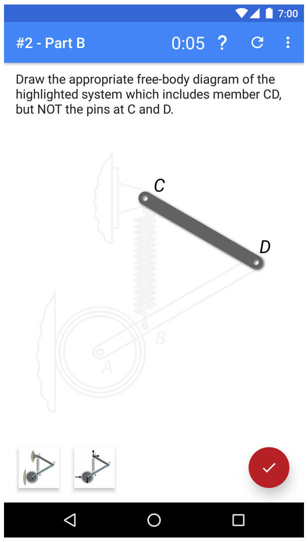
The spring-loaded system is shown below. All members are massless except the wheel which has a mass m. Neglect friction. Reference the image below for parts a, b, and c.

(a)



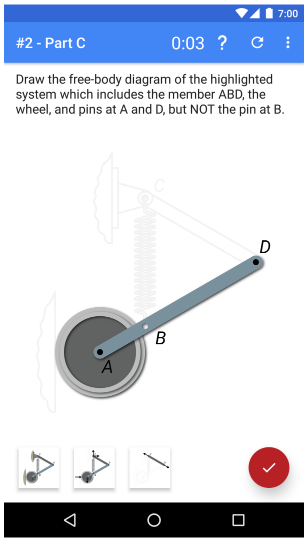
Draw the free-body diagram of the highlighted system which includes the spring, member CD, member ABD, the wheel which has a mass, m, and pins at A, B, C, and D.

(b)



Draw the appropriate free-body diagram of the highlighted system which includes member CD, but NOT the pins at C and D.

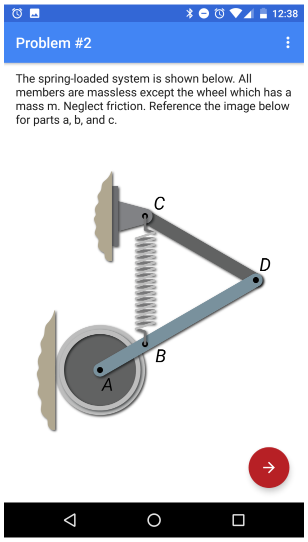
(c)



Draw the free-body diagram of the highlighted system which includes the member ABD, the wheel, and pins at A and D, but NOT the pin at B.

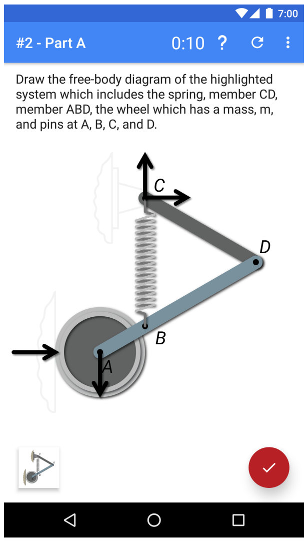
(d)

Figure A.3: Problem 2



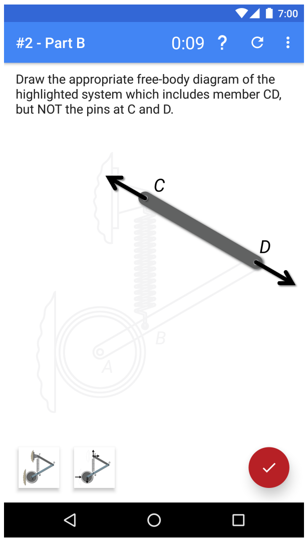
The spring-loaded system is shown below. All members are massless except the wheel which has a mass m. Neglect friction. Reference the image below for parts a, b, and c.

(a)



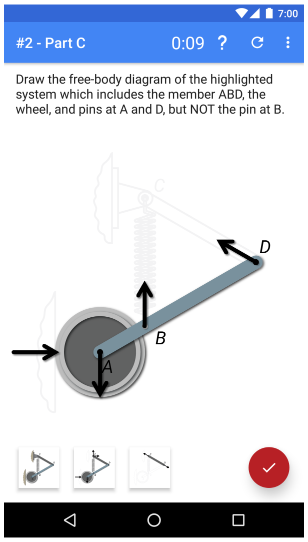
Draw the free-body diagram of the highlighted system which includes the spring, member CD, member ABD, the wheel which has a mass, m, and pins at A, B, C, and D.

(b)



Draw the appropriate free-body diagram of the highlighted system which includes member CD, but NOT the pins at C and D.

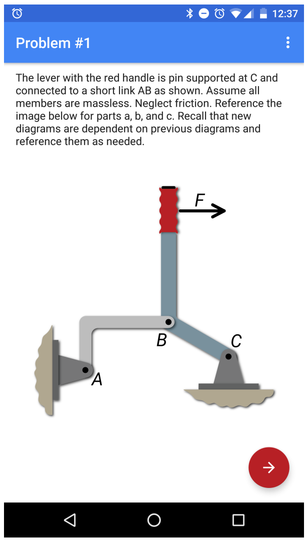
(c)



Draw the free-body diagram of the highlighted system which includes the member ABD, the wheel, and pins at A and D, but NOT the pin at B.

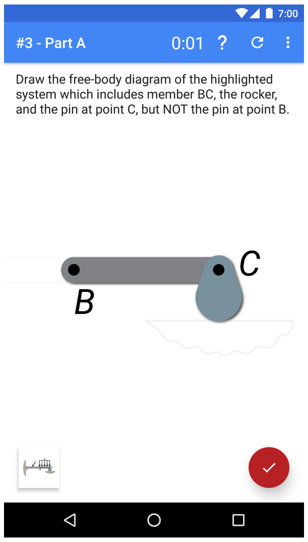
(d)

Figure A.4: Problem 2 with Arrows



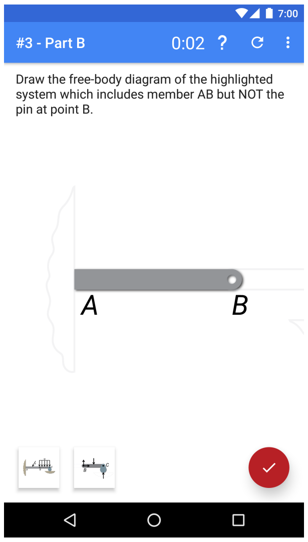
The lever with the red handle is pin supported at C and connected to a short link AB as shown. Assume all members are massless. Neglect friction. Reference the image below for parts a, b, and c. Recall that new diagrams are dependent on previous diagrams and reference them as needed.

(a)



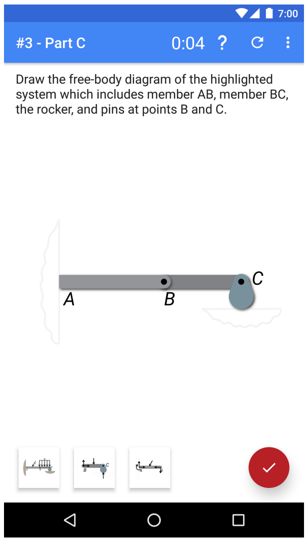
Draw the free-body diagram of the highlighted system which includes member BC, the rocker, and the pin at point C, but NOT the pin at point B.

(b)



Draw the free-body diagram of the highlighted system which includes member AB but NOT the pin at point B.

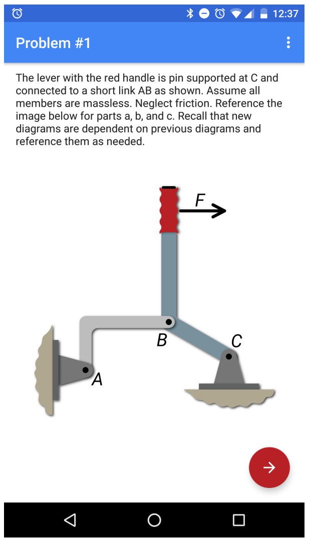
(c)



Draw the free-body diagram of the highlighted system which includes member AB, member BC, the rocker, and pins at points B and C.

(d)

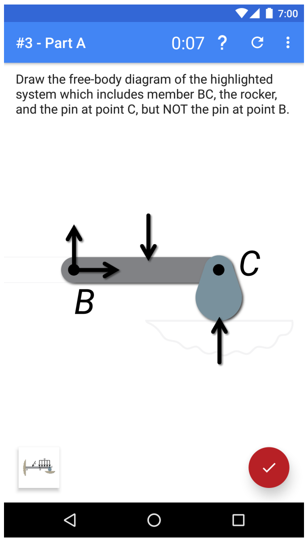
Figure A.5: Problem 3


12:37

Problem #1

The lever with the red handle is pin supported at C and connected to a short link AB as shown. Assume all members are massless. Neglect friction. Reference the image below for parts a, b, and c. Recall that new diagrams are dependent on previous diagrams and reference them as needed.

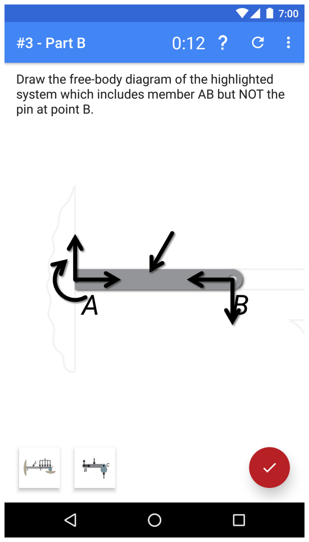
(a)


7:00

#3 - Part A 0:07

Draw the free-body diagram of the highlighted system which includes member BC, the rocker, and the pin at point C, but NOT the pin at point B.

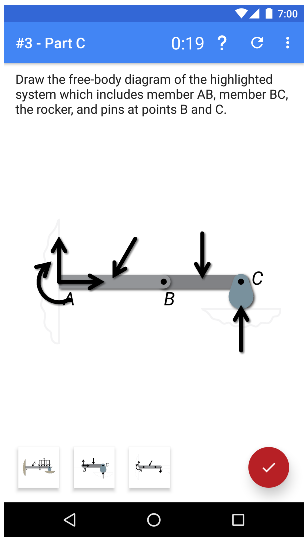
(b)


7:00

#3 - Part B 0:12

Draw the free-body diagram of the highlighted system which includes member AB but NOT the pin at point B.

(c)


7:00

#3 - Part C 0:19

Draw the free-body diagram of the highlighted system which includes member AB, member BC, the rocker, and pins at points B and C.

(d)

Figure A.6: Problem 3 with Arrows