Fail-Safe Erase

by

Zackary Dean, Spencer Bach, & Nicholas Sullivan

Submitted to
the Faculty of the School of Information Technology
in Partial Fulfillment of the Requirements for
the Degree of Bachelor of Science
in Information Technology

Zackary Dean	04/13/20
Zackary Dean	Date
Spencer Bach	04/13/20
Spencer Bach	Date
Nicholas Sullivan	04/13/20
Nicholas Sullivan	Date
Bogdan Vykhovanyuk	04/13/20
Bogdan Vykhovanyuk, Faculty Advisor	Date

University of Cincinnati
College of
Education, Criminal Justice, and Human Services

April 2020

TABLE OF CONTENTS

LIST OF ILLUSTRATIONS

ACRONYMS AND ABBREVIATIONS

GPS Global Positioning System

FCC Federal Communications Commission

SHA Secure Hash Algorithm

UI User Interface

ABSTRACT

The FCC reported that there were about 3.1 million smartphone thefts in 2013. Today nearly everyone owns at least one smart device, and many people tend to keep personal or business information on these same devices. If your device is lost the first reaction you might have is to use attempt to use 3rd party software to locate or wipe the sensitive data from your device, but what happens when the connection to those services is severed? The Fail-Safe Erase application aims to provide you with the ability to wipe your device after its disconnected from reaching network services. After setting up a personalized password and a wipe timer, the Fail-Safe Erase application will run in the background while looking for those services. Once the connection to GPS, network, and cellular service is interrupted the wipe timer begins its countdown. If the user does not access their device to interrupt the wipe timer within the time set by the user, the device will begin to wipe itself of all personal data. Fail-Safe Erase aims to ensure that if you can't gain access to your mobile device, remotely or physically, the private data on it will be securely erased, hopefully before it falls into the wrong hands.

INTRODUCTION

Problem

According to the Federal Communications Commission (FCC), the number of smartphone thefts in 2013 was about 3.1 million. In 2018 the FCC's Technological Advisory Council's Mobile Device Theft Prevention working group using a Harris Poll surveyed that number to be around 4 million smartphones. Now in 2020, the number of mobile devices a person may possess is increasing, people own multiple phones for business and personal use. The increasing number of mobile devices means an increased risk of mobile device theft. Many people now know of and are utilizing geolocation via Global Positioning System (GPS) if their device is lost, misplaced, or stolen. However, a thief that is technically educated may attempt to disable these services by using airplane mode, a Faraday bag, empty paint can, or some other means of disconnecting the mobile device from both cellular and GPS services. This would be to reduce or eliminate the risk of being tracked down using tools that utilize those services such as Google's Find My Phone. By disabling the phone's ability to connect to external services the theft disables how traditional mobile device management and remote wiping tools operate. Now with the phone unreachable and untraceable, the thief can do whatever they want with it.

Individual users and companies alike share common interests when it comes to data security for themselves or their employees. What happens if an employee or individual has sensitive information on their device and has it stolen by someone that wants to sell or misuse that information? What if the thief has the experience to disconnect the device from external connections so it can't be remotely wiped? Mobile devices are becoming more heavily used in our daily life for business and personal uses.

Meanwhile, malicious actors are becoming more creative and careful with executing their attacks. Fail-Safe Erase aims to protect its user's while decreasing the attack vector for malicious actors.

Solution

Fail-Safe Erase is an Android mobile device application that allows users to protect their data by acting as a last resort if their mobile is completely disconnected from mobile services and is unreachable via GPS. The application comes with a simple one-time set up that will allow the user to customize how they would want the application to operate depending on their mobile device usage. This includes how long the application will wait before wiping the user's data, what services to monitor, and any changes to the user's profile.

Remote wiping tools for mobile devices exist such as iCloud's Find My iPhone, Google's Find My Device, Microsoft's Intune, and DriveStrike Remote Wipe. None of these services will help a user locate their device if the GPS services are turned off and none of these services can execute a wipe of the device's hard drive if the device can't be reached using network or cellular services. This is where Fail-Safe Erase steps in, after disconnecting from external services the wipe timer begins the countdown. If the user's device doesn't reconnect to these services or if the user doesn't access the application before the timer runs out, the device is deemed lost and completely wiped of all data.

Project Goals and Methodology

Fail-Safe Erase aims to provide users the ability to have their mobile device data completely erased in case the device is misplaced in a location without internet, cellular or GPS service. Fail-Safe Erase promotes keeping the user's data secure and out of reach from potentially malicious actors when traditional remote wiping tools can't.

Fail-Safe Erase Features:

- After initial setting users create a secure user account with an encrypted passcode to make any additional setting changes to Fail-Safe Erase
- Users set a timer that will start when the wireless, cellular and/or GPS signal is lost and resets when the user logins to the application. If the mobile device loses internet and/or GPS signal once the personalized timer runs out, the mobile device is wiped

Together, these features aim to provide the user with an application that will reduce the worry of misplacing, losing, or having a mobile device and the data on that device stolen from them.

Overview

The remainder of this report details how the Fail-Safe Erase project was completed. This report includes the project concept, design objectives, methodology, budget, timeline, any problems and how these problems were addressed. Finally, the conclusion will detail the lessons we learned and the skills we improved upon during the development of the Fail-Safe Erase application.

DISCUSSION

Project Concept

Fail-Safe Erase was created during a brainstorming session between Zack Dean, Nick Sullivan, and Spencer Bach. The idea came up while discussing possible problems people experience with lost or stolen phones. Nick proposed that we create a mobile application that could defeat faraday bags or similar devices that could disable the ability to execute a remote wipe the phone.

If a mobile device is lost or stolen while it isn't connected to the internet using the mobile device's network or cellular service. The owner of that device will be unable to use remote wiping tools, software, or any similar service provided by traditional mobile device management tools. Our team envisioned a mobile application that a user could easily install and would completely wipe a user's Android device without needing a network connection or past the initial user configuration. A set and forget fail-safe in case the user's device is ever compromised or misplaced.

Design Objectives

Our team will develop a functional mobile application for Android devices that is user-friendly and visually appealing while providing data security.

Fail-Safe Erase will meet the following objectives:

- All features mentioned above in Project Goals
- Appealing and well-designed user interface
- The application will be developed using Android SDK
- Take advantage of SHA-256 to encrypt the user's passcode

During the development of Fail-Safe Erase, two specific features had to be removed due to time and development constraints. Dark mode was removed mainly due to inconsistencies between menus. It was also decided that because Fail-Safe Erase is a "set and forget" application, users won't see dark mode often since they shouldn't be opening the application frequently. Power-saving mode was originally an idea that would reduce the frequency of network checks by the Connectivity Manager. After the initial idea, it was revealed there is no way to reduce the number of checks produced by the built-in Android Connectivity Manager and Location Manager, therefore while this feature was a good idea it is not possible with the current Android processes which Fail-Safe Erase relies upon.

Methodology

While developing Fail-Safe Erase, our team relied on the design requirements and procedures crucial to meeting our project's goals. To do this, we evaluated each goal and how the design requirements would meet that goal.

- Appealing and well-designed user interface

A simple, easy-to-use, and appealing user interface was an important goal our team had in mind at the start of development. As development progressed, we decided to focus less on an appealing, good looking user interface and more on a simple, easy to use UI. Our team wanted to create a mobile application that would be quick and easy to set up, with little to no struggle or asking for help needed from users. Our team took the stance that the easier it is for a user to configure the app the more people will want to use the Fail-Safe Erase.

- The application will be developed using Android Studio

Our team wanted to take advantage of the best and most widely used Android development tools available to us. Our team choose Android Studio and SDK tools due to how popular and widely used these tools are with Android development.

- Take advantage of SHA-256 to encrypt the user's passcode

Our team wanted to take advantage of the most up-to-date and secure encryption algorithm available on Android mobile devices. SHA-256 will provide our team with the ability to ensure the user's passcode is secure and encrypted so no other users can gain access and disable the wipe timer.

User Profile

Figure 1: User Profile below this page, illustrates the user profile for Fail-Safe Erase. The user profile provides an in-depth look into the potential users that Fail-Safe Erase is developed for. This profile provides details such as previous user experience, pervious experiences with similar applications, the experience that the user will have when using this app, how frequently the user will need to use the app, and the key design requirements needed to meet the profile needs. The development team will use this user profile to ensure they meet the key target user(s) for Fail-Safe Erase.

PROJECT:

Fail-Safe Erase

POTENTIAL USERS:

- Businesses, executives, travel-heavy employees, security teams
- Tech and security enthusiasts, politicians, journalists, travelers
- Power users who have sensitive information that they don't want to be stolen

SOFTWARE, INTERFACE, AND RELATED EXPERIENCE:

This project is targeted towards anyone with personal and private data stored on their Android mobile device that they don't want to be lost or stolen by anyone with malicious intentions. It can be used in both business and personal cases for anyone that is at risk of losing their mobile device with personal or business information.

The only experience a user needs to have prior to using the Fail-Safe Erase app is how to install an app from the Play Store and how to create a basic user account. No technical experience is required, the app will walk the user through their account setup and the configuration of the wipe process.

EXPERIENCE WITH SIMILAR APPLICATIONS:

Many users may have experienced using factory reset software or software to locate their phone by GPS in the past. Other users may have their devices managed by their company's IT department who could remotely wipe their device if lost or stolen. Whatever their experience may be, this app ensures that our users have an easy way to protect their data in case of phone theft or accidental loss. Similar applications require a network connection in order to execute the remote wiping process. The Fail-Safe app addresses that issue with being installed on the device locally and actively checking for a connection to GPS and network services.

TASK EXPERIENCE:

Upon opening the application, users will be brought to a welcoming page, asking them to create a secure user account with an encrypted passcode. They will then be brought to a page that asks them to enable admin for the app and then brought to the app's main menu. The users will be able to adjust the timer depending on their preference. If a user would like to adjust their passcode, timer setting, turning the service on or off, or any other preferences this will be available in the settings menu after logging into the application.

FREQUENCY OF USE:

The setup of this product will only need to occur once on startup. Once everything is configured, the application works in the background. The user will only need to access the application if they need to enable or disable the wipe or change their settings.

KEY PROJECT DESIGN REQUIREMENTS THAT THE PROFILE SUGGESTS:

- Quick start and setup of user account and application
- Visually fluid and easy to navigate UI
- Ability to toggle technical features on/off

Figure 1 User Profile

Use Case Diagram

The diagram below, **Figure 2: Use Case Diagram**, displays the use case for the Fail-Safe Erase application. The diagram shows how the user is expected to interact with the application. It also shows how the application will utilize and communicate with external services. This diagram represents the minimum features needed for the application to meet customer expectations. Additional features may be added to improve the user's experience after meeting the base requirements.

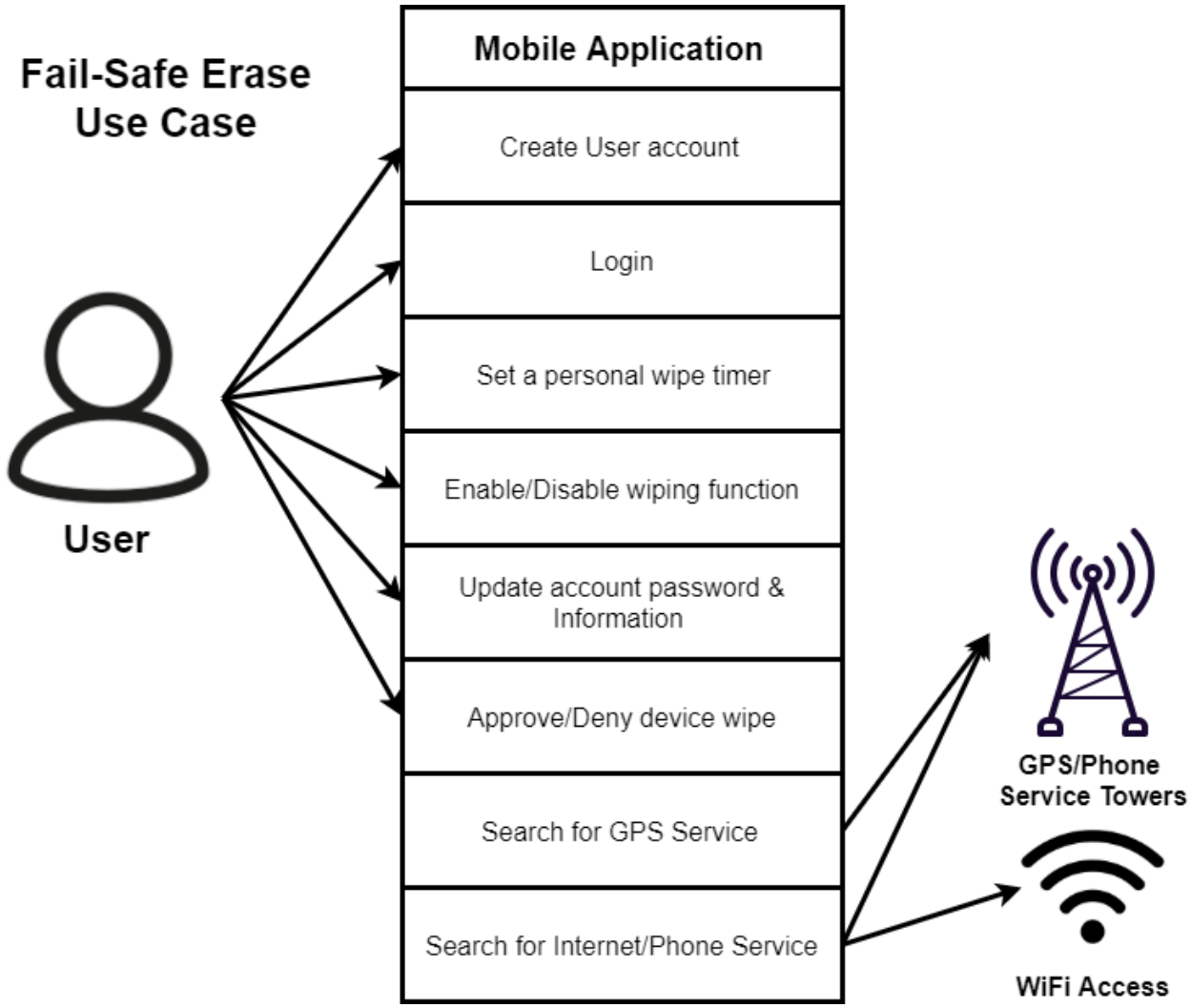

Figure 2 Use Case Diagram

Technical Elements and Architecture

Figure 3: Technical Architecture depicts the technology being utilized and how it communicates with Fail-Safe Erase. **Figures 4-10** below, show the various navigational menus available to Fail-Safe Erase users. Fail-Safe Erase is an Android application that can be installed on any rooted Android device via an APK file. It was developed with Android Studio using two programming languages: Kotlin and Java working in the backend and XML. The application works by communicating with Android's built-in "Connectivity Manager and Location Manager" which is present in every android device. Using these, Fail-Safe Erase can receive information regarding the connection states of each network connection established on that device. Using that information Fail-Safe Erase can determine whether or not external connections to GPS, cellular, and/or wireless signals are currently being made, this is shown in Figure 6. Once Fail-Safe Erase notices these services being disrupted the personalized timer begins to countdown. After the countdown begins the application will continue the countdown until connections are reestablished or the timer reaches zero and the wipe process begins. After reaching zero the application starts Android's built-in wipe process, depending on the model of the phone this process can take anywhere from 15-90 minutes to complete, this is shown in Figure 10. Once the process completes the phone will be reset to factory defaults without any user data present. The User profile information and settings are saved in a "SharedPreferences" file that is present in Android the user creates upon the first-time setup.

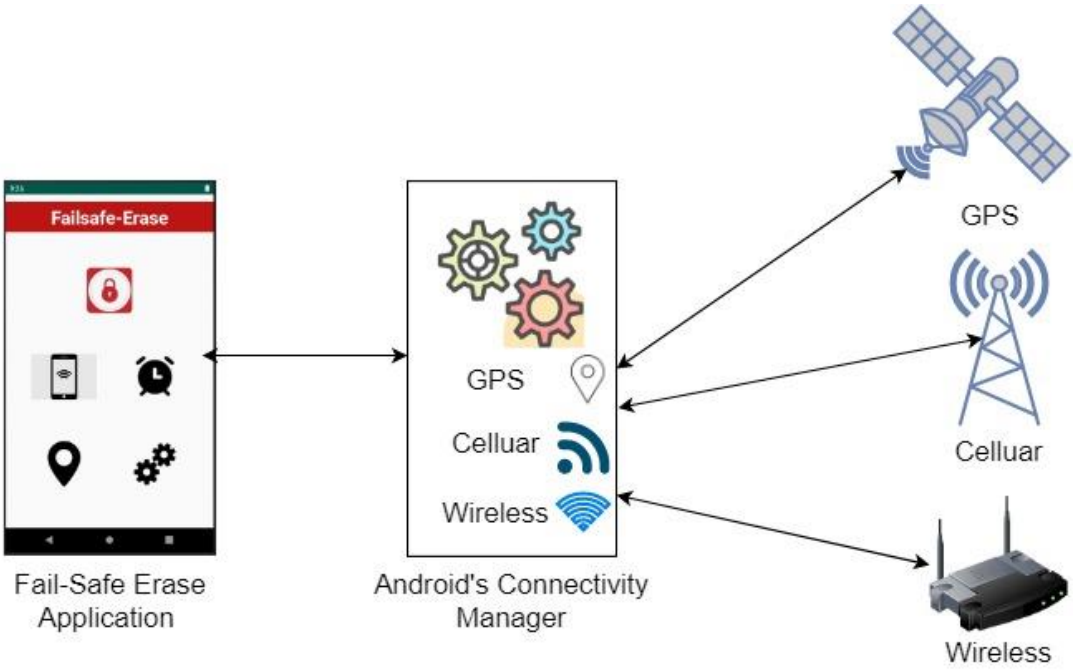

Figure 3 Technical Architecture

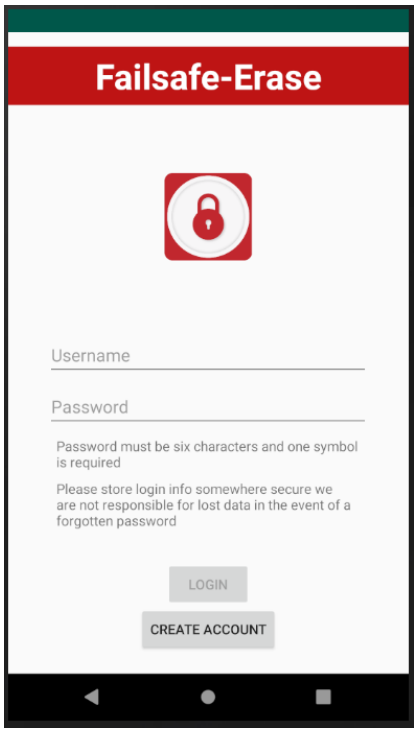

Figure 4 Main Menu

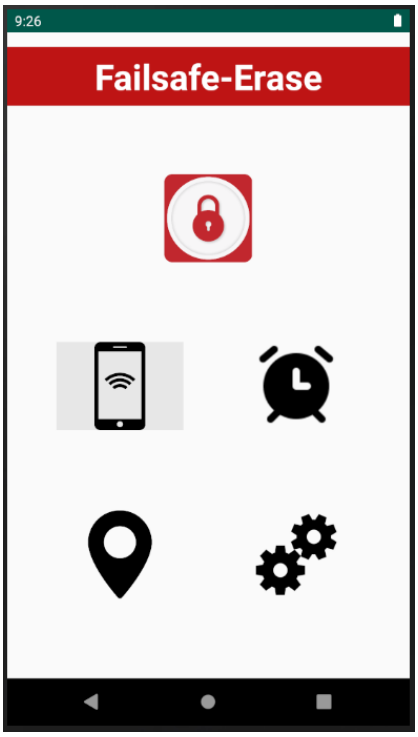

Figure 5 Login Screen

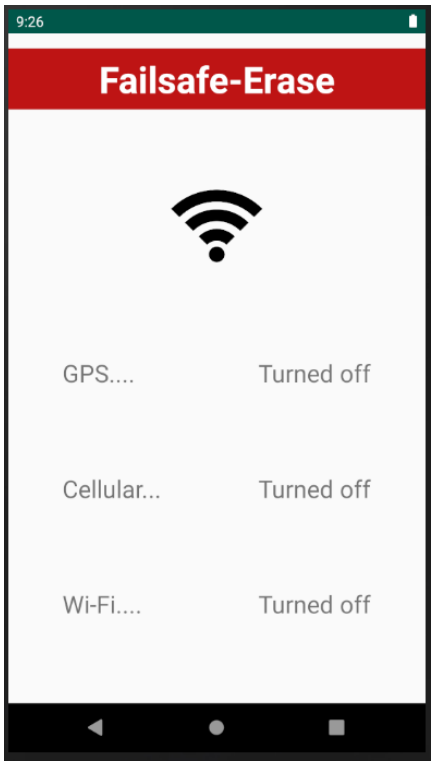

Figure 6 Connectivity Menu

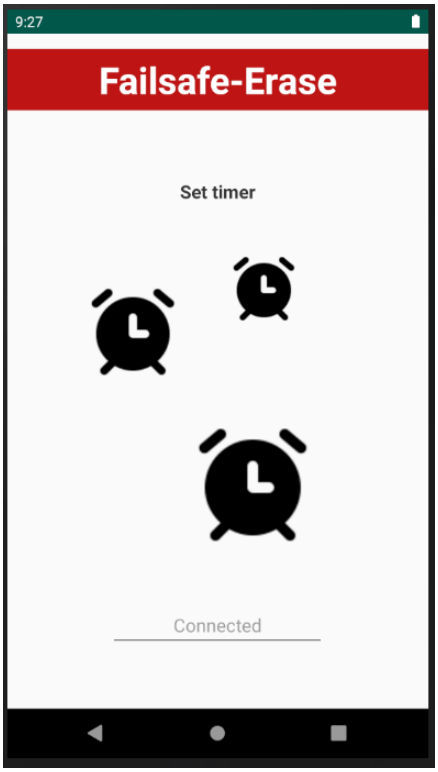

Figure 7 Timer Menu

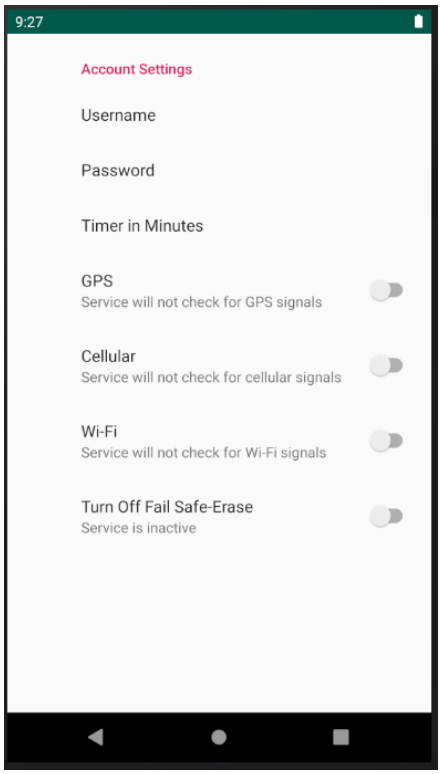

Figure 8 Account Settings

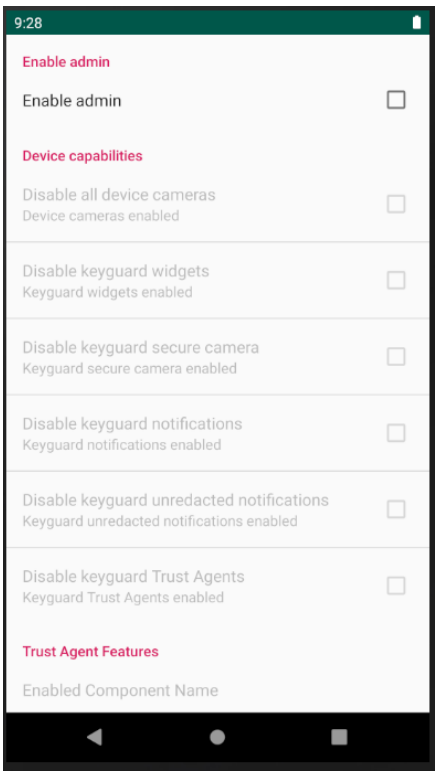

Figure 9 First Time Setup, Admin Permissions

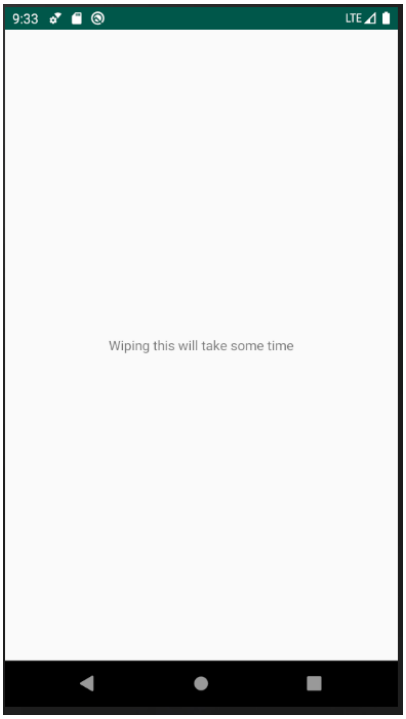

Figure 10 Wiping Process

Testing

This section contains information regarding how the Fail-Safe Erase application was tested and evaluated. The methodology that was chosen for this application was a mix between the waterfall method and the rapid application development method. Our team picked this methodology because we wanted to first meet our development goals at a rapid pace while staying within the time constraint. Our team planned testing to be conducted in phases, unit testing, integration testing, system testing, security testing, user acceptance testing, and release testing. After testing, the Fail-Safe Erase application should operate optimally, securely, and be visually pleasing to the everyday user.

Scope of Testing

The original scope of testing was to include unit testing, integration testing, system testing, security testing, user acceptance testing, and finally, release testing if needed. Unit testing would include testing of the phone wiping feature, the account creation feature, the testing of disabling specific services, and any other additional features if added. Fail-Safe Erase will make use of the Android's Connectivity Manager to monitor the GPS, wireless, and cellular services on the phone, integration showed how these services report to the Fail-Safe Erase application and ensure the data is updated and accurate. As a team, we will then conduct system testing across the whole application to ensure all basic functions are working as intended and the Connectivity Manager is fully integrated with Fail-Safe Erase. Security testing will be conducted with but not limited to Ostorlab and Immuniweb android scanning tools that will supply a report of any vulnerabilities or weaknesses in the application. User acceptance testing will be conducted by coworkers, friends, and family members of the project team. They will evaluate the functionality of each of the features, the usability or general feel of navigating through the application menus, and their feedback regarding the visual look of Fail-Safe Erase. Finally, release testing will be conducted by the project team and will involve testing the app to verify it is operating flawlessly.

Objectives of Testing

The objective of testing Fail-Safe Erase is to determine what functionalities, features, and/or graphics that may hinder the fluid operation of Fail-Safe Erase. Unit testing will ensure the main functionalities of our application including the wipe feature and the countdown timer are fully operational. Integration and system testing will ensure that the Connectivity Manager can connect and communicate with the Fail-Safe Erase application. Security testing will ensure that Fail-Safe Erase is secure and will not have any critical or high vulnerabilities that could be exploited to compromise the application. User acceptance testing will be conducted with various users to get their feedback on the overall look, feel, and operation of Fail-Safe Erase. Release testing will be conducted right before the tech expo to ensure the application works flawlessly and as intended.

Testing Procedure

Below are the tests that were performed:

1. Unit Testing

   Below, **Figure 11: Unit Testing** shows which features were testing and the result of the testing. Unit testing focused on the four main Fail-Safe Erase features that are critical to the function of the app. The Wiping Feature was tested and failed due to a permission issue, after granting appropriate permissions there were no additional errors. The account creation worked as planned there were no issues with all ten tests. Disabling services using the application worked as intended as well. The timer countdown failed during the first test and continued to count into negative numbers. The code was modified so this no longer happens, and the wipe feature is called when the timer reaches zero.

Testing Action	# Passes	# Fails	Intended action
Wiping Feature	8	2	Factory Reset mobile device
Account Creation	10	0	Creating a secure account
Disabling Services	10	0	Select which specific services to monitor
Timer Countdown	9	1	Once reaches 0:00 call the wiping function

Figure 11 Unit Testing

2. Integration Testing

Figure 12: Integration Testing shows the testing of each of the three services that Fail-Safe Erase monitors. Integration testing focused on Android's Connectivity Manager. GPS, cellular, and wireless signals were monitored for 3 hours using the Connectivity Manager. This was conducted 10 times over two weeks. During testing, we found no issues with the communication between Fail-Safe Erase, Android Connectivity Manager, and the connections that are monitored.

Testing Action	# Passes	# Fails	Intended action
GPS monitor (3 hrs.)	10	0	Constant monitor of GPS signal
Cellular monitor (3hrs.)	10	0	Constant monitor of Cellular signal
Wireless monitor (3hrs.)	10	0	Constant monitor of Wireless signal

Figure 12 Integration Testing

3. System/Release Testing

Figure 13: System Testing shows each of the five test runs and any issues encountered during each test. Due to time constraints, we merged system and release testing. This testing was to verify all intended functions of Fail-Safe Erase operate correctly and fluidly before release. This was done by Nick who conducted several dry runs of Fail-Safe Erase. These tests or dry runs included all the features included in unit testing as well as the communication tested in integration testing.

Run #	**Issues encountered**	**Remedy**
Run 1	The wipe process would fail without proper admin rights.	The app would have to be set as a Device owner via ADB commands or NFC Provisioning or if the phone is rooted. Also, if the phone is using an older version of Android, "Foreign Context Loader" can be used to wipe the phone without these privileges.
Run 2	The app would crash if the admin page was accessed multiple times in multiple ways.	Make the page accessible only once when "Create Account" was pressed
Run 3	Service would die if the application was closed or was doing too much work	The service had to be running on a separate thread on the Android phone which kept it alive indefinitely.
Run 4	The Service would die if the phone was restarted.	I had to make the application start at boot.
Run 5	No issues	N/a

Figure 13 System Testing

4. Security Testing

Figure 14: Ostorlab Scan and **Figure 15: Immuniweb Scan** displays any high-risk vulnerabilities or concerns with Fail-Safe Erase and how our team plans on addressing each risk. Security testing was conducted two separate online Android application scanning tools. Immuniweb conducted the following scans: OWASP's mobile top 10, backend APIs and web services, mobile application behavior, software composition analysis, and external communications. Ostorlabs scanned Fail-Safe Erase using static taint analysis, dynamic analysis, and smart fuzzing. Ostorlab reported one high-risk, one medium-risk, one low-risk, two potentially risky, and one important-risk vulnerability as shown in Figure 14. Immuniweb reported one medium-risk, 4 low-risk, and 6 warning or information vulnerabilities as shown in Figure 15. For best practice and due to time constraints, our team decided to focus on any high, medium, and important risks. Our team decided to review low and info risks but most of these cannot be addressed or have such a low impact it is not worth resources at the time. For any vulnerabilities that were not addressed before release, we have detailed how we plan on resolving or reducing these risks with a future update to Fail-Safe Erase.

Risk	Title	Description	Remedy
High - Ostorlab	Debug mode enabled	Application is complied with debug mode enabled	Disabled debug mode
Medium – Ostorlab	Application code not obfuscated	Application's source code is not obfuscated and can be decompiled	Obfuscate Java code with Proguard or Dexguard. Check running environment at runtime
Low - Ostorlab	Insecure Network Configuration Settings	The app does not specify a network security config	Set network security configuration setting by enabling encrypted traffic only. This can be done with a Certificate Authority
Important - Ostorlab	Exported activities, services, and broadcast receiver list	List of all exported components	Informative, no recommendations

Figure 14 Ostorlab Scan

Risk	Title	Description	Remedy
Medium - Immuniweb	Predictable Random Number Generator	Uses predictable RNG	Kotlin does not support official secure pseudorandom number generator.
Low – Immuniweb	Debug mode enabled	The mobile application has debug mode enabled	Disabled debug mode
Low - Immuniweb	Enabled Application Backup	Mobile application uses external backup functionality	Change android:allowBackup="false"
Low- Immuniweb	Missing tapjacking protection	The mobile application does not have tapjacking protection	A malicious application must be installed to exploit tapjacking. Redefine how extends are used in Fail-Safe Erase code
Low – Immuniweb	Exported Services	The mobile application contains an exported service	Remove android:exported="true"

Figure 15 Immuniweb Scan

5. User Acceptance Testing

Our team originally planned user acceptance testing but due to time constraints and the COVID-19 virus lockdown, this drastically reduced any chance for face-to-face testing. After system/release testing we determined that with the remaining time left in the semester, the time that would be required for planning, scheduling, and documenting virtual user acceptance testing via video conferencing would not be worth the small amount of feedback that we would receive. As a team, we determined it would be more beneficial to conduct user testing after the COVID-19 lockdown is lifted to receive better and more ample user feedback. Fail-Safe Erase can have any changes such as UI design or additional features added with a future software update.

Budget

Figure 16: Project Budget outlines the projected budget for Fail-Safe Erase. All tools used in the development of Fail-Safe Erase are all available freely. To calculate the simulated labor the total number of weeks of senior design was taken, 32 weeks. Each team member estimated the amount of time they worked on the project during a week with the average being around 4 hours a week. For cost per hour, the team estimated $25 per hour was reasonable compensation for the average developer. The team then took the total number of weeks (32) multiplied by the average weekly hours worked (4) multiple by 3 for each team member and then finally by the estimated hourly wage ($25). The total cost of simulated labor for the project came out to $9600. Due to the Tech Expo being online there was no need to purchase a new Android device to demonstrate Fail-Safe Erase, for users it is assumed that they have already purchased an Android device.

ITEM	UNIT HOURS	COST	TOTAL
Simulated Labor	384 hours	$25	$9600
TOTAL			$9600

Figure 16 Project Budget

Project Schedule

Figure 17: Work Breakdown Structure, Figure 18: Fall Gantt Chart and Figure 19: Spring Gantt Chart all outline the projected schedule for planning, design, development, testing, and presentation of the Fail-Safe Erase application.

Task Name	Duration (Days)	Start Date	End Date
1.0 Project Management			
1.1 Assignment 1: Team Contract	17	9/03/19	9/20/19
1.2Assigment 2: Project Abstract for Tech Expo	21	9/23/19	10/14/19
1.3 Assignment 3: Team Contract Resubmission	22	9/23/19	10/14/19
1.4 Assignment 4-5: User Profile, Use Case Diagram	7	10/14/19	10/21/19
1.5 3 Minute Elevator Pitch	14	10/07/19	10/21/19
1.6 Assignment 6: Draft Report	14	10/21/19	11/04/19
1.7 Assignment 7: Final Report	28	11/04/19	12/02/19
1.8 Final Fall Presentation	1	12/02/19	12/02/19
1.9 SS IT Expo Preparations	92	1/13/20	4/10/20
1.10 SS Assignment 1: Testing Plan/Report	14	1/27/20	2/10/20
1.11 SS Assignment 2: Abstract	7	2/10/20	2/17/20
1.12 SS Assignment 3: Draft Tech Expo Poster	8	2/24/20	3/2/20
1.13 SS Assignment 4: Final Poster Due	7	3/2/20	3/9/20
1.14 SS Final Presentation	14	3/23/20	4/6/20
1.15 SS Final Report Due	28	3/9/20	4/6/20
1.16 IT Tech Expo	-	4/14/20	4/14/20
1.17 SS Assignment 7: Final Library Copy Due	16	4/14/20	4/29/20
2.0 Initiation and Planning			
2.1 Team Building	3	8/26/19	8/28/19
2.2 Research Project	7	8/27/19	9/3/19
2.3 Identify Userbase	7	9/3/19	9/9/19
2.4 Identify Concerns	14	9/3/19	9/16/19
2.5 Define How the App will Function	7	9/10/19	9/16/19
2.6 Define Responsibilities	2	9/17/19	9/18/19
2.7 Define Requirements	21	9/10/19	9/30/19

Task Name	Duration (Days)	Start Date	End Date
3.0 Front-End Design			
3.1 Create Story Board	7	10/14/19	10/20/19
3.3 Create Mock-ups	20	10/18/19	11/07/19
3.4 Build Basic Events for Front-End	14	11/05/19	11/19/19
3.4.1 Design Login Component	1	11/10/19	11/11/19
3.4.2 Design Create Account Component	2	11/10/19	11/12/19
3.4.3 Integrate GPS Screen	10	11/10/20	11/20/19
3.4.4 Design Dashboard Screen	10	11/10/19`	11/20/19
3.4.5 Design Settings Screen	5	11/10/19	11/15/19
3.4.6 Design Failsafe-Erase Timer Screen	3	11/11/19	11/14/19
3.6 Finish Polishing Design for Back-end Development	2	12/05/19	12/07/19
4.0 Back-End Development			
4.1 Creation of Login service	10	2/05/20	2/15/20`
4.2 Creation of Services using Android Connectivity Manager	6	2/10/20	2/16/20
4.4 Creation of GPS Failsafe Erase protocol	30	2/15/20	3/15/20
5.0 Testing			
5.1 Unit Testing	15	3/05/20	3/20/20
5.2 Integration Testing	13	3/18/20	3/31/20
5.3 System/Release Testing	15	3/20/20	4/04/20
5.4 Security Testing	2	3/20/20	3/22/20
5.5 Add/removal of Additional Features	5	3/20/20	3/25/20
5.6 IT Expo Prep	20	3/23/20	4/11/20
6.0 Deployment			
6.1 IT Expo Deployment	1	4/14/20	4/14/20
	-		

Figure 17 Work Breakdown Structure

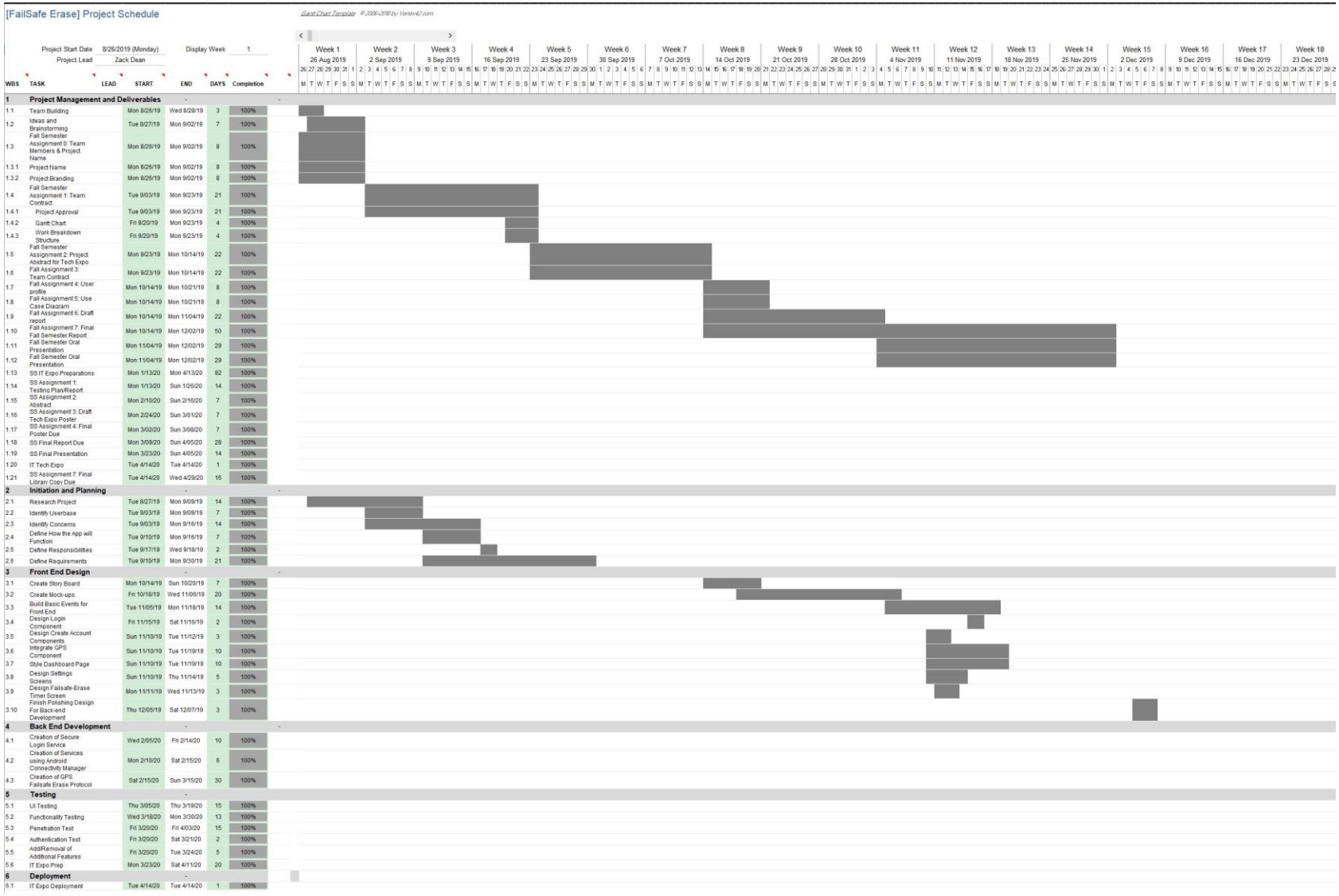

Figure 18 Fall Gantt Chart

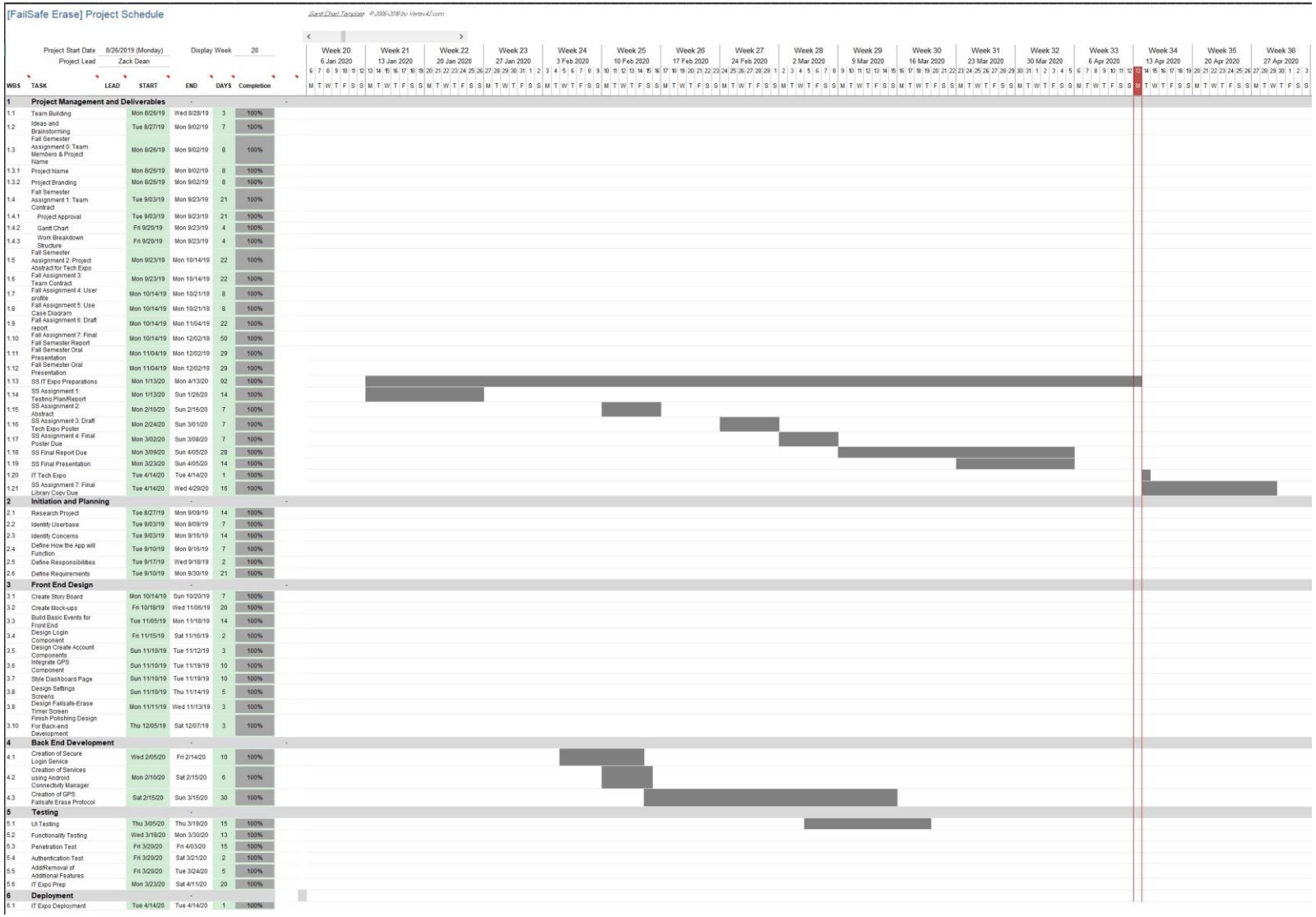

Figure 19 Spring Gantt Chart

Problems encountered and Analysis of Problems Solved

- Issue with getting a project approved

Our team had issues with getting a project that would be approved by our advisor. To solve this, we held a brainstorming session for the next two days to come up with a list of ideas to send to our advisor. After coming up with some solid ideas we typed out short descriptions for each idea and sent them to our advisor to see which he liked. After being approved our team started development on Fail-Safe Erase.

- To avoid issues with assignments we attempted to meet at least once a week

Whenever there were confusion or troubles with assignments our team would compare schedules and decide when the best day to meet would be to discuss said assignment(s).

- Application Development Issues/Progress

The development of the application has been slow to get started. To address this, we planned on meeting more often to talk in detail regarding the coding of the Fail-Safe Erase application. We have also refined our Wireframes several times to provide better references when designing the GUI and developing the backend code.

- Major Time Constraints

Our team was given two semesters (~32 weeks) to brainstorm a problem, develop a solution, test the solution, and release by tech expo. Due to all three team members having limited programing experience the development of Fail-Safe Erase took longer than we originally planned. Once completed testing was limited because of the amount of time before the tech expo and the COVID-19 virus. To address this we prioritized the development of Fail-Safe Erase and fast-tracked it to be finished by the

end of Spring Break. Once completed we began testing, to save time we combined system and release testing. Due to both time constraints and the COVID-19 lockdown user acceptance testing would be conducted at a later date.

Future recommendations and recommendations for improvement

After two semesters Fail-Safe Erase is now in an operational state that meets our project goals. However, there is much more that we can improve upon or add to the application as it matures. If our team had to start from the beginning again, we would have limited our scope to just the wipe timer. The power-saving mode and dark mode were both very interesting ideas that might be added later down the road in an update but at the time this took development time away from the main function of the app, the wipe process. The biggest struggle for our team was managing and controlling the time to took to develop Fail-Safe Erase. We went outside our scope this impacted development which impacted testing, a chain of events.

Going forward if we had additional time, we would like to continue working on power-saving mode to reduce the amount of power Fail-Safe Erase uses by continuously doing network checks. Our team would also like to add the option for dark mode back in, this would not take too much time just some more UI redesign that was missed due to the time frame. Finally, our team would like to add the option for email notifications for when a mobile device loses connections and reconnects to network connections.

The future of Fail-Safe Erase is uncertain, the code resides with lead developer Nick. Currently, there are no plans to continue development past graduation, but we may release the code as open-source for others to improve upon.

CONCLUSION

Fall Lesson Learned

In conclusion, some of our lessons learned after the fall semester would be:

- Always plan more time for tasks/assignments than needed. It's better to be done early than have to rush them at the last minute or push back other tasks
- Begin research and development very early on. While we did do some research, our team didn't feel like they did enough research into our topic at the start of the project. We also didn't begin the development of the application until the late fall semester. This could result in a demo that doesn't meet our expectations for the fall presentations.
- More detail isn't necessarily better. Our team learned that we need to focus less on the details and more on the big picture of our project. If we look at small details our team may end up behind instead of getting the main focus of the project solid and complete.

Fall Conclusion

After the fall semester, our team plans on focusing on developing the final pieces of Fail-Safe Erase that we didn't get to demo during our fall presentation. This will include smaller and backend features. During the spring we plan on replacing any placeholder art with better looking and more appealing art assets. In-between this development we will continue to focus on project assignments. To meet these objectives, we will continue to meet at least once a week if not more during the spring semester. Towards the middle to end of the semester our team will conduct penetration testing, user testing, and feedback, and finally preparing and practicing for the IT expo.

Spring Lesson Learned

After the spring semester was finished the main lesson, we learned was that time is extremely valuable. Any time that can be used early, including the winter break, should be used to further the project. It is better to start and finish early in comparison to adding additional features or impacting other parts of the project. As a team, we also learned that staying in scope is key to staying on track. After adding dark mode and power-saving mode, the time that was used to develop those features could've been used elsewhere.

Skills Developed

Some of the skills our team improved upon during the development of Fail-Safe Erase:

- Better project and people management
- Time management and balance
- Best practices for developing friendly user interfaces
- Best practices for data security, wiping, and encryption on Android devices
- Best practices for Android development and coding
- A better understanding of front-end development
- What is expected from the frontend development member on a team
- Development using Android Studio, Kotlin, and Java
- Communication both face-to-face and video conferencing

Spring Semester Completion

After the Fall semester, Fail-Safe Erase had completed menus and the ability to navigate to each of the screens as well as manually start the wipe process. Since then, a device administration prompt was added as well as the service that runs in the background and checks the connection states based on the settings the user sets. Once the device admin prompt is accepted, the actual wipe process was added which required the application to have administrator permissions. Besides these major functions, general bug fixing was done as well as the quality of life changes. Some of these include the device admin screen being called after the “Create Account” button was pressed, removing the connection state toggles from the “Connection state prompt” and kept these toggles in the settings as this was redundant. The dark mode option was removed as issues were arising with the theme not being applied to every single activity and screen. Lastly, the power-saving toggle was removed as there is currently no way to limit the number of checks that the Location Manager or Connectivity Manager makes. In addition to application development, we finalized our abstract and created a testing plan in February, draft a poster for the tech expo, and finally revised and updated our final presentation.

REFERENCES

"Report of Technological Advisory Council (TAC) Subcommittee on Mobile Device Theft Prevention (MDTP)." Federal Communications Commission, 4 Dec. 2014, https://transition.fcc.gov/bureaus/oet/tac/tacdocs/meeting12414/TAC-MDTP-Report-v1.0-FINAL-TAC-version.pdf

"Technological Advisory Council (TAC) Mobile Device Theft Prevention (MDTP) Working Group." Federal Communications Commission, 5 Dec. 2018, https://transition.fcc.gov/bureaus/oet/tac/tacdocs/reports/2018/11.30.18-MDTP-WG-Report-and-Recommendations.pdf

APPENDIX

APPENDIX A. ADDITIONAL INFORMATION

A.1. Fail-Safe Erase Presentation

Figure 20 Fail-Safe Erase Presentation

A.2. Fail-Safe Erase Poster

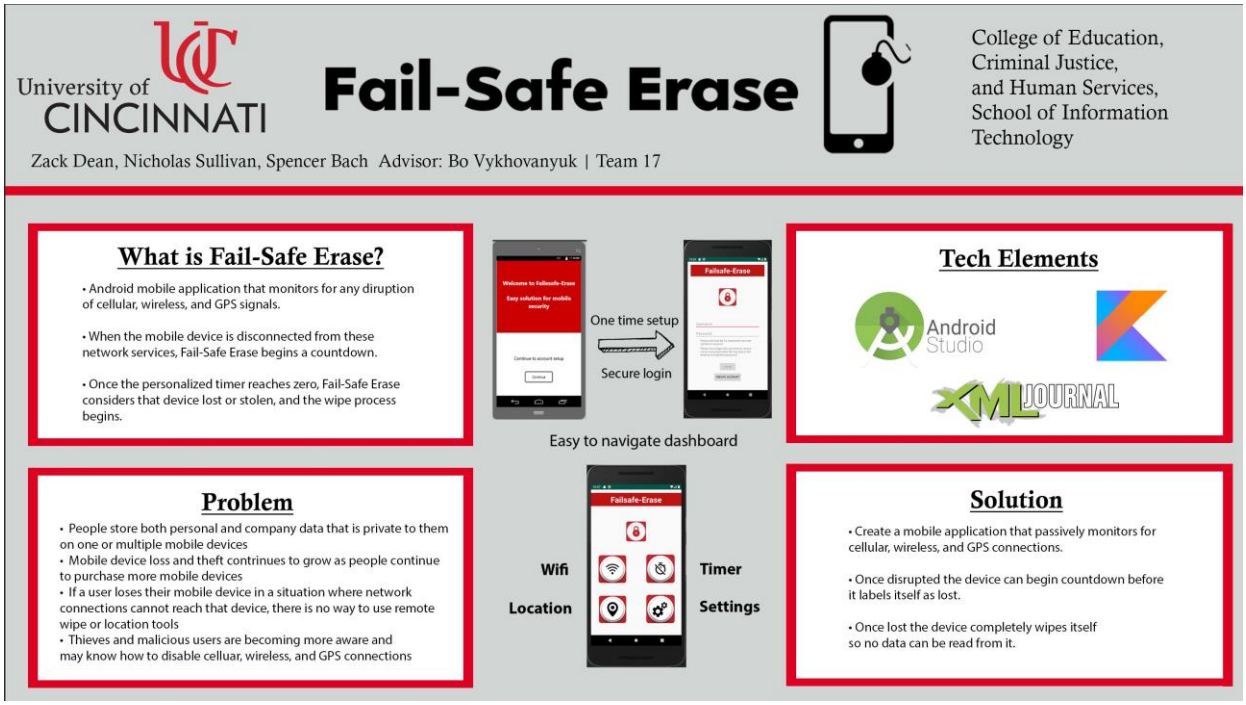

Figure 21 Poster