Team ReBoot.

Final “as-built” Report

Version 1.2

Date: May 7th, 2020

Team Members:
Bianca Altman, Jennie Dallas,
Victoria Leafgren, Isai Martinez,
Cindy Valentin

Sponsor: Dr. Emery Eaves

Mentor: Dr. Eck Doerry

This is a modified version of the final report. Please update this document with new information about the project as it is discovered/needed. Prefer to use this final report over the one found in the iMATter web repository so we can have a single source of truth. The pdf in the Documentation directory may not be up to date but is there for convenience.

Table of Contents

1.0 Introduction……………………………………………………1

2.0 Process Overview……………………………………………….3

3.0 Requirements……………………………………………………4

4.0 Architecture and Implementation……………………………11

5.0 Testing……………………………………………………………24

6.0 Project Timeline…………………………………………………29

7.0 Future Work……………………………………………………32

8.0 Conclusion………………………………………………………32

Appendix A…………………………………………………………34

1.0 Introduction

Opioid addiction among pregnant women is a growing issue in the United States that has costly consequences for the mother, child, and for the country. Between 1999 and 2014, the number of women with opioid use disorder at the time of delivery quadrupled, leaving the current rate at 6.5 per 1,000 deliveries[1]. Use of opioids during pregnancy is a particularly serious problem, as it can result in newborns being born with a withdrawal syndrome known as neonatal abstinence syndrome (NAS). Babies born with NAS may experience poor feeding, breathing problems, body shakes, excessive crying, and more[2]. Longer term, NAS can cause cognitive or behavioral impairments[3]. From 2004 to 2014, there was a five-fold increase in the number of babies born with NAS in the United States, which has consequently resulted in a steady increase of the amount of money spent on treating these babies. In 2014, an estimated $563 million in hospital costs was spent on treating infants with NAS, compared to the $90.9 million spent in 2004[4].

To combat the rising rates of opioid addiction, pregnant women with opioid use disorder are enrolled in a treatment program early in their pregnancy. This not only helps decrease the rates of opioid addiction, but also helps minimize the number of infants born with NAS. However, while plenty of women enter treatment programs, the rates of relapse are relatively high: 56% of women who are six months postpartum experience relapse[5]. Explaining the cause of high relapse rates calls upon the expertise of our client.

Our client, Dr. Emery Eaves, is a medical anthropologist at Northern Arizona University whose research focuses on the opioid epidemic among pregnant women in Arizona. Through recovery programs and support groups in Phoenix, Dr. Eaves finds pregnant women with opioid use disorder to interview; using this research to come to a better understanding of these women's situations and to identify their biggest recovery obstacles. Dr. Eaves has found that the three major obstacles of traditional treatment programs that contribute to high relapse rates are:

- **Attending appointments and obligations** - Pregnant women recovering from opioid use disorder usually receive a daily dose of Methadone as part of treatment, which can only be dispensed to them one dose at a time at a Methadone clinic. These daily appointments combined with multiple counseling appointments a week as well as

1 National Institute on Drug Abuse. (2019, January 22). Opioid Overdose Crisis.

2 The Recovery Village. (2019, October 14). Long-Term Effects of Neonatal Abstinence Syndrome.

3 Seattle Children's. (2018). *Neonatal Nursing Education Brief: The Long-Term Outcomes of Infants with Neonatal Abstinence Syndrome*.

4 National Institute on Drug Abuse. (2019, January 22). Dramatic Increases in Maternal Opioid Use and Neonatal Abstinence Syndrome.

5 *Goodman, Daisy. Should Maternity Care and Treatment for Opioid Use Disorder Be Integrated? [PowerPoint slides].*

prenatal care appointments can add up to be nearly impossible to accommodate in a typical woman’s schedule.

- **Lack of social support** - If pregnant women in recovery barely have enough time to attend the required appointments and other obligations they have, finding time to make new friends and socialize can become out of the question. Due to the stigma surrounding pregnant women with opioid addictions, it’s especially difficult to find people who are supportive of their recovery process. This can leave these women feeling isolated and unsupported in their recovery.
- **Lack of information access** - Similar to a lack of social support, not having any leisure time makes it more difficult for a pregnant woman in recovery to seek out key health and pregnancy information specific to their recovery. This leads to a lack of knowledge of certain recovery strategies that these women should adhere to, or medical advice that could help make the recovery process easier.

These challenges are amplified if the woman in recovery lives in a rural area far away from clinics or has trouble accessing reliable transportation and makes the recovery process even more difficult.

To aid pregnant women in recovery and help them overcome these challenges, we have developed a mobile application called iMATter that will provide a secure, virtual environment that supports aspects of their recovery process. Key features of iMATter include:

- **Chat Room** - A place that connects pregnant women with opioid use disorder with each other to offer each other advice, speak on shared experiences, and eliminate the lack of social support.
- **Information Desk** - A forum where women may ask pregnancy or recovery related questions and receive answers from credible and knowledgeable sources in a timely manner. This will help women easily obtain answers to questions they have throughout pregnancy and recovery.
- **Learning Center** - A place that provides learning materials that helps users expand their knowledge of pregnancy and recovery information. This will help improve ease of access to information that pregnant women in recovery might seek out.

Other features include an in-app calendar, surveys, pregnancy updates, a map of recovery and pregnancy resources, and more. This mobile application will be coupled with a web application utilized by administrators, allowing them to create, edit, delete, and otherwise view content being displayed in the mobile application. Administrators will also be able to add or delete users and configure other settings for both applications. Providers will also access this web application, as their role is to answer questions being posted in the information desk forum by pregnant women in recovery.

With iMATter, pregnant women recovering from opioid use disorder will have a source of social support and easily accessible informational materials tailored to their pregnancy and recovery process. This leaves them with more free time, alleviates the feeling of isolation that they often face, and provides them with a tool to help guide them towards a successful recovery.

Now that we have given an introduction to the iMATter application, let us discuss the processes we used in developing this product.

2.0 Process Overview

To develop iMATter, we had two components to create: the mobile application and the web application. When the development process started, we decided the best route for us was to separate the development into our main modules based on our requirements. With this in mind, we also decided that each person would develop their module in both the mobile and web applications and do so in the same time period. The reason we decided this would be best is because the web application is used for creating and managing data for the mobile application; therefore, if the modules were being developed on both platforms concurrently, integration testing could take place during development.

As a team we meet weekly with our mentor and client, receiving feedback and ideas on how to improve the iMATter applications. As well, each week we met as a team to discuss the completed tasks and tasks to come for each member in a scrum style. We began with all meetings in person and any other communications through a group GroupMe chat room. However, due to certain circumstances in the Spring 2020 semester, meetings were moved to online using Zoom. We updated our repositories during our meetings after the group accepted everyone’s branches as good to merge. We started by getting the functionalities of the modules working as expected, considering user interface design later in the Spring 2020 semester. Once we had developed all of our modules to be testable, we distributed the application to our client to ensure we were developing everything as expected.

Originally, our team distributed roles for user interface design, back-end and front-end development. Due to changing our development process to be module based, some roles changed. The roles were:

- **Bianca Altman**: team lead, technical writing editor, client communications, mobile and web application development
- **Jennie Dallas**: GitHub release manager, technical writing editor, video editor, mobile and web development

- **Victoria Leafgren**: mobile and web application development
- **Isai Martinez**: mobile and web application development
- **Cindy Valentin**: mobile and web application development

The tools we used are as follows:

- **GitHub**: GitHub was chosen for version control, allowing us to all have access to a shared repository and easily manage updates and merges.
- **GroupMe**: GroupMe was used for group communication purposes.
- **Zoom**: We used Zoom for our online meetings.
- **Google Drive**: For documents, presentations, and sharing documentation we used Google Drive and provided tools (i.e. Google Docs).

Next, we will discuss how we gathered our requirements through our meetings with our client and outline those requirements.

3.0 Requirements

Over the course of the Fall 2019 semester, we met with Dr. Emery Eaves about twice every month, as well as had meetings with both Dr. Eaves and Dr. Doerry that occurred weekly. During these meetings, we acquired higher level requirements, with which we created use cases to better understand what functionalities would be expected from these requirements. With the use cases, we were able to develop around 57 functional requirements and 25 performance requirements. During our Spring 2020 semester, as functionalities developed and new needs were discovered or some requirements deemed unnecessary, some of these requirements were updated verbally.

3.1 Functional Requirements (FR)

Functional requirements are specific functionalities the system must provide to support the domain-level requirements. The following is a list of some of our high-priority functional requirements with a few of the more important specifications about them (*Note*: this is a summarized list and does not contain every detail. Please visit our website and check out our Requirements Document for more information on our requirements).

FR1 Secure Accounts and Information

One of the domain level requirements was that this app should be able to support accounts. In order for users, administrators, and medical professionals to be able to use the applications, they will need to be able to create accounts for themselves. These accounts should be password protected to protect personal information and restrict accessibility to authorized users.

FR1.1 User Accounts

We define a user as an expecting mother in recovery that received a unique code from an administrator.

- Users must enter a unique code given to them to be brought to the sign-up page.
- User accounts require the following to create their profile: pseudonym for username, password, avatar picture, due date, recovery email address, answer to a security question with the option of including the following in their profile: location, short biography.
- After signing up, users will be automatically placed into a cohort that is based on their due month.

FR1.2 Administrative Accounts

We define an administrator as someone is allowed access to user information and has authority to add and remove data to the application.

- Administrators will require the following to create a profile: username, password

FR1.3 Medical Professional Accounts

We define medical professionals as clinic workers or opioid recovery resource center volunteers/workers that have been given access to the information desk from an administrator.

- Accounts will be initialized by administrator with the following required information: first and last name, date of birth, email.
- Medical professionals will require the following to create a profile: username, password, avatar picture, and have the option to write a short biography about themselves and their work and assistance with recovery that will be displayed on their profile.

FR.2.0 Levels of Privilege

Each account type should only be allowed access to certain functionalities of the applications depending on account type. After login, the program will immediately decide based on credentials what authority and access a person logging in has. The accessibilities for each account types are defined in this section.

FR.2.1 Users

- Users will exclusively use the mobile application

FR.2.2 Administrators

- Administrators will exclusively use the web application to: moderate forum and chat to ensure no negative dynamics are developing, add learning modules, surveys, and videos, maintain user and medical professional accounts, send out notifications.

FR.2.3 Medical Professionals

- Medical professionals will only be able to access what is called the information desk

FR3.0 Threaded Forum

There will be a threaded forum that serves as a place for users to ask medical professionals questions. This will serve as a place where users may go to ask questions, they may either be embarrassed to ask their cohort members or want the opinion or answer from a medical professional. This forum is also called the information desk.

FR5.1 User Capabilities

- Users will be able to post medical-related questions that will be answered by a medical professional.

- To avoid users not wanting to feel embarrassed to ask a question, the user asking the question will have the option to keep their identity completely anonymous with no username displayed.

- All users will have the ability to respond to thread.

FR5.2 Medical Professional Capabilities

- Medical professionals can answer any questions and continue discussions in the comments on the main thread.

FR4.0 Calendar

A calendar will be provided to users to serve as a place where they can maintain and keep track of their recovery.

- Events on the calendar will trigger push notifications for reminding the user of the event before the event occurs.

FR5.0 Cohort Chat Room

To provide users with a space where they may support one another, give and receive advice, and share experiences, users will have a chat room where they may talk to one another in real time.

FR5.1 User Capabilities

- Users can send messages in the room, which will then be visible to other users in the same cohort for a certain period of time decided by an administrator.

FR5.2 Administrative Capabilities

- Administrators will be able to view all user discussion threads from the web application.
- Administrators can delete any threads or comments.
- Administrators can choose a set time from a list for how long chats should be viewable.

FR6.0 Learning Center

As this application is a tool to help aid in recovery, there will be a center where users may go to learn about recovery and their pregnancy. Data in this center will be decided and added in by administrators.

- Certain material in the learning center will change based on the user's stage in pregnancy.
- Users will be able to do the following: watch provided videos on the application that were provided by an administrator, take interactive learning modules here, answer multiple choice question/answer prompts, earn points based on their participation, interact with a provided link.

FR7.0 Widgets

We define widgets as components that will be displayed in the application that users may interact with.

FR7.1 Mood Survey

- Users will have a mood survey on the home page at all times where they may select their current mood from a selection of faces.
- Any mood selected will be sent to the user's chat room to publicly show their current mood.

FR7.2 Baby Development Fact

- Users will be able to see how their baby is developing.

FR7.3 Resources and Helplines

- Users will have a list of phone numbers and links for resources they might need for their opioid addiction recovery accessible to them.

FR8.0 Rewards Tracker

In order to keep users interacting with the application, a rewards system will be put in place to allow users to earn points that may be used for rewards.

- Users will maintain points that they can gain through participation in the app.
- Users can see how many points they have

FR9.0 Notifications

As another way to keep users interacting with the application and keeping them on track with their recovery, notifications will be sent to users for different types of events.

FR9.1 Notification Events

Users will be sent push notifications for the following events and reminders: calendar events, new messages in cohort chat room, new questions at information desk, mood updates

FR9.2 Notification Configuration

- Users will be able to configure how far in advance they wish to be notified of calendar events
- Users will be able to configure if they wish to receive all notifications or just specific ones.

DR8.FR10. Research Analytics Tool

As this application will serve as a functioning prototype for a grant, Dr. Eaves will need to be able to prove the feasibility of this application as a tool for recovery for feature implementations and releases. To be able to do this, we will need to be able to see how users are interacting with the application.

- Information the administrator will be able to view will all be logged into a database table. The administrators will want to view this data from the users to prove the app's feasibility.
- Information and statistics should be presented in a graph to give administrators a visualization of the data.

Now that we have outlined our functional requirements, we will go into the performance requirements we developed for them.

3.2 Non-functional Requirements (NFR)

Our performance requirements outline what is to be expected from our functional requirements in terms of metrics. These were determined by our best judgment, also considering that users may have never used an application like iMATter before (*Note*: this is a summarized list and does not contain every detail. Please visit our website and check out our Requirements Document for more information on our requirements).

NFR1 Account Creation and Login

NFR1.1 Users

The signup pages and login page will all be clearly labeled, with specifications as to what is required for signup/login, and what is optional for signup. Considering this, the work through should be minimal. Taking into consideration that not everyone will understand these components immediately, the time to learn about and create accounts are the following:

- For 90% of first-time users, user account creation should take in total ≤ 120 seconds.
- User account login should take in total ≤ 30 seconds the first time, ≤ 10 for experienced users.

NFR1.2 Administrators

- For 90% of first-time administrators, new administrator accounts signup should take in total ≤ 60 seconds.
- Administrator account login should take in total ≤ 30 seconds the first time, ≤ 10 after the first time.

NFR1.3 Medical Professionals

- For 90% of first-time, new medical professional accounts signup should take in total ≤ 60 seconds.
- Medical professional account login should take in total ≤ 30 seconds the first time, ≤ 10 after the first time.

NFR2.0 Information Desk

Information on how the information desk works and how users can interact by asking questions and commenting will be provided to users, so we expect the following times for users to understand the forum:

- Posting a question for the first time should take 80% of new users ≤ 2 minutes to understand, and ≤ 30 seconds after the first time or for experienced users.
- Notifications should send in ≤ 30 seconds to medical professionals.
- New questions posted by users should show up in the information desk within ≤ 5 seconds.

NFR3.0 Calendar

- Understanding of how to interact with the calendar should take ≤ 2 minutes for 90% of users, ≤ 30 seconds for experienced users.

- Adding events to the calendar should take ≤ 10 seconds.

NFR4.0 Cohort Chat Room

- As this component is more complex than others, we expect full understanding of this to take ≤ 5 minutes. For users who have experience with chat rooms, ≤ 1 minute.
- New messages sent by other users should show up in discussion threads within ≤ 5 seconds.

NFR5.0 Interactive Component Updates

- Due to the fact that this app is to be personalized on the homepage, computation speed to display may take up to 15 seconds.
- General navigation will be easy to understand, having suggested interaction options on the homepage that will navigate to the appropriate page. Otherwise, common icons with occasional labeling will be used. We expect 90% of users to be able to fully understand all the app features in ≤ 1 hour.

NFR6.0 Reward System Updates

- Points should update in the database in ≤ 5 seconds after the user accomplishes a task that would add to points or points are reset.
- Points should update immediately on the user page after a task is completed or points are reset.

NFR7.0 Notification Sending Time

- The majority of notifications sent should take ≤ 30 seconds to send to the user after being triggered.

NFR8.0 Analytics Loading

- Analytic Graphs and Charts should take ≤ 45 seconds to fully load.

Now that we have outlined some of our primary functionalities, we will discuss the architecture and implementation we developed to fulfill these requirements.

4.0 Architecture and Implementation

The architectural layout of iMATter is driven by the need for dynamically adding and editing mobile application content, as administrators will be actively viewing and changing content from the web application.

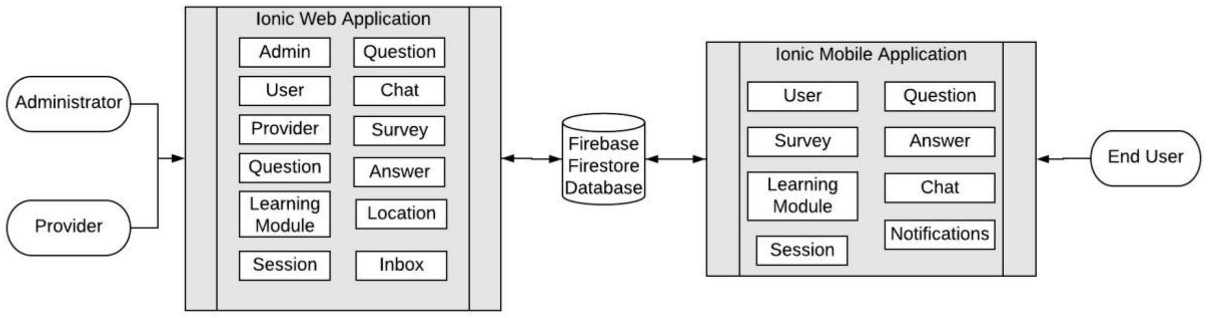


Figure 4.0 - iMATter System Diagram

Our general solution can be seen in Figure 4.0. Our solution contains two major components, a mobile application for end-users and a web application for administrators and providers, which will share information through a shared Firebase database. This application uses a Firestore database, allowing for synchronization of the connected applications through Google's real time listeners.

The mobile application provides pregnant women in recovery a judgement free space to obtain information about recovery and where they can receive social support from other women in the same position, all while remaining completely anonymous. This application also provides customized resources based on pregnancy to the users, a way for users to talk to one another, and provides learning opportunities that users may not easily access otherwise. The web application for administrators provides a place for them to maintain and manage users and content, such as being able to create and distribute learning modules and surveys to the users of the mobile application. Providers are able to communicate with mobile users through the information desk forum. We will now go into more detail about these key features for each user.

Key Responsibilities and Features

The responsibilities and features differ based on the type of user. All three users will have different experiences using their application.

Administrators and providers will only be using the web application, but for different functionalities. Administrators, at a high level, will be using the web application to be able to do the following:

- Manage mobile application user, providers, and other administrators accounts, including creating, editing, and deleting the accounts.
- Manage forum questions/answers and chat logs posted on the application, meaning having the ability to delete forum questions/answers or chats sent.
- Create and add learning modules, surveys, and pregnancy updates to be administered to the mobile application.
- View click-path analytics of the user application.

Providers are only using this application for answering information desk forum questions. At a high level, providers will be using the web application to:

- Answer questions posted in the information desk forum.
- Create a profile that contains a username, provider type, and short bio for users of the mobile application to view.
- View negative emotion updates from users.

The mobile application provides social support, informational resources, and other tools to help pregnant women in opioid recovery, which at a high level include:

- Interact with a chat room based on the cohort they are placed in, meaning they can send chats and view chats from other users that are also in their cohort.
- Interact with a calendar that uses local storage to store user input dates and receive notifications to remind them of the event.
- Ask questions in the information desk, as well submit answers on their own questions or other questions.
- Answer surveys.
- Take learning modules, which can include watching a video and answering questions.
- View information about their current pregnancy status.
- Earn points from completing surveys and learning modules, which can be redeemed for gift cards.

With these key features in mind, next we will discuss how these components communicate with one another.

Communication Mechanisms and Data Flow

As seen in Figure 4.0, using the Firebase Firestore database, the web and mobile applications will be able to share data in real time. These two components will be sharing data for multiple different application functionalities. For example, the general flow of mobile app user creation will begin with the administrator creating a blank user, which is done by pressing a button that will provide a randomly generated code. This code is then stored in the database and the administrator may then give the code to the user. When the user goes to sign up in the mobile application, they will first be prompted into providing the required code. If the code is correctly entered and has not been entered before, the user is brought to the signup page and asked to enter their required signup data. Upon submitting this, this data will be added to the database attached to code that was entered and will then show on the web application to the administrator where mobile application users are listed. This is the general concept for most app to app communication.

Now that we have established a high-level detail of the workings of iMATter, we will discuss our application's components in more detail.

4.1 Module and Interface Descriptions

As previously stated, iMATter consists of two applications, the mobile and web. We have broken this down into three main modules: the web application for administrators, the web application for providers, and the mobile application for users. In this section, we will discuss the classes and interfaces that make up each module, as well as why they are necessary components and how they will be used. To begin, we will discuss the administrator web application module.

4.1.1 Administrator Web Application Module

The administrator web application module provides administrators of iMATter with a user interface to view and manage content being displayed in the mobile application. This includes managing mobile application user accounts, provider accounts, creating surveys and learning modules, viewing mobile application chat and forum activity, and more. Figure 4.1 below is the UML of our administrative web application.

To better explain the functionalities of the administrator web application module, the details of each component are listed below:

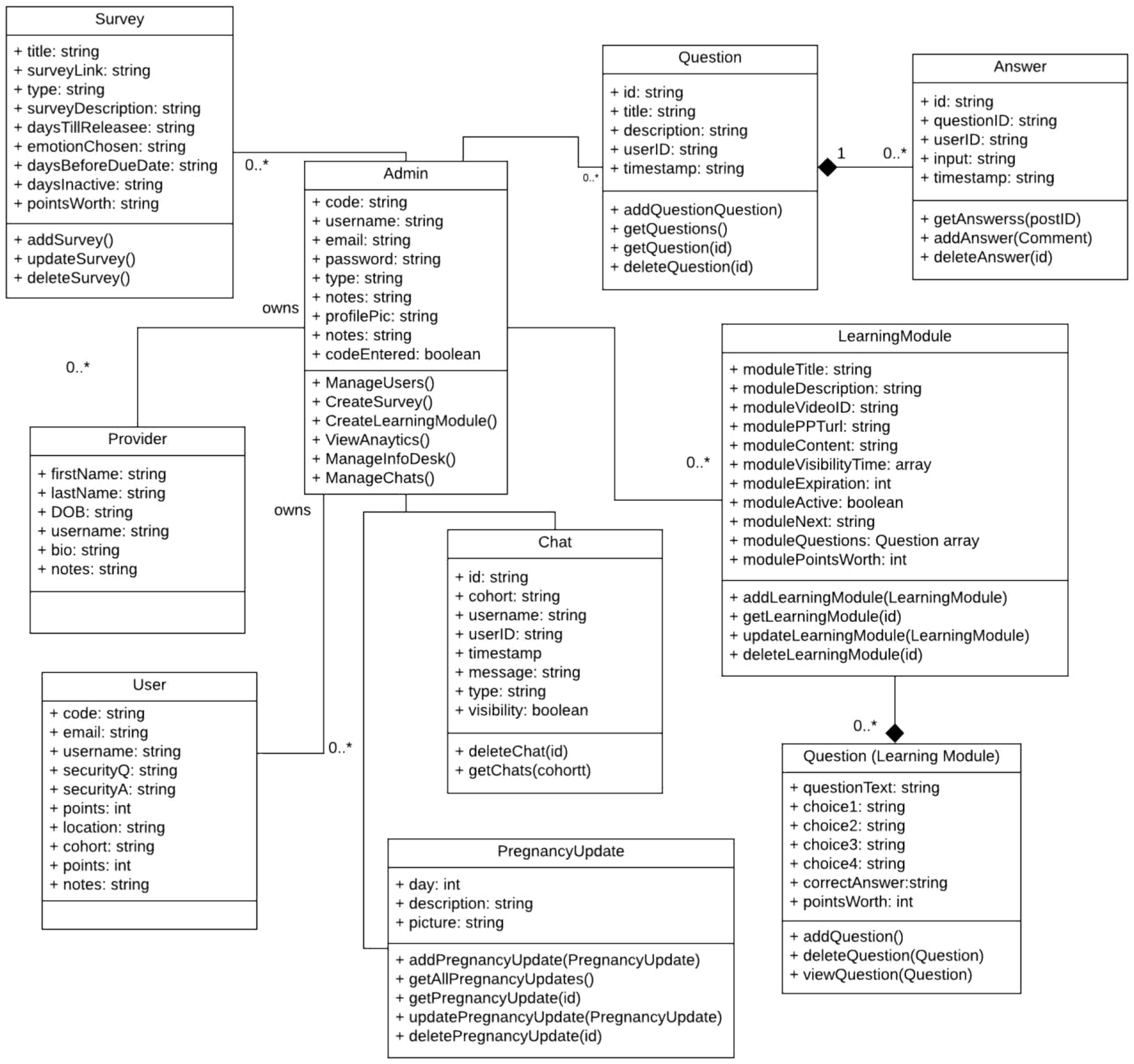


Figure 4.1 - Administrator Web Application Module UML

Admin:

We define an administrator as someone that has access to viewing all user created data and managing data. The admin type will be able to interact with multiple components. Admins will be able to create, edit, and delete all user types. Admins can create, update, and delete survey objects which will be viewable by not only themselves but also mobile application users. Learning module and pregnancy update objects are also created, updated, and deleted by admins, which are also viewable by both admins and users. Admins can manage user created data by being able to delete certain objects, including question, answer, and chat objects that are created by any user type. Admins will also be able to view analytic data which is created by users on the mobile application.

User:

We define a user as a pregnant woman in recovery for opioid use disorder that has been given access to this mobile application. Since users do not use the web application at all, only admins will be interacting with the user component. Administrators will be able to view a list of users and certain user fields, such as their sign-up code and username. Admins can delete users, as well as edit certain fields of each user, such as their cohort and total number of reward points.

Provider:

We define providers as volunteers and medical professionals who are given access to the web application in order to assist the users by answering questions in the information desk. The administrators will be able to view a list of providers and some of their fields, as well as have the ability to update a provider’s name and email or delete providers. The administrators will be able to see a provider’s code, email, name, username, provider type, and any notes that were added to that provider’s account during the creation process.

Survey:

Surveys are Qualtrics surveys that users will take to help administrators get feedback on their experiences. Administrators can create new surveys by providing links to existing Qualtrics surveys as well as other information such as the time period they should be displayed to users, or which event should trigger the survey. Currently, the four types of surveys are: after user joining, before due date, inactive user, and mood surveys. Each survey has an associated point value to count towards the user’s reward points. Admins also have the ability to delete surveys they have created if needed, as well as edit existing surveys.

LearningModule:

A learning module is a page containing informational media such as a video or presentation, followed by questions for users to answer regarding the information presented. In the web application, the learning module is responsible for taking in administrator input from its fields and saving the input in the database, including the title, description, and total points worth (calculated from adding the individual question’s points). The learning module component in the web application gives administrators the ability to create, edit, and delete learning modules that appear for the mobile application user to view. Therefore, the admin component will be interacting with the fields and functionality of this component. The questions component will also be utilized in the learning module, as one of the learning module’s fields is storing an array of questions. These questions can be added and deleted by the administrator within each learning module.

Question (Learning Module):

Learning module questions are objects that are created by the administrator and added to learning modules. They contain questions on learning module materials and answer selections that will be displayed to mobile application users, as well as the point value of each question. The purpose of this component is to provide a quiz-like feature in the mobile application. Questions are exclusively used in learning modules and cannot be created anywhere else. Therefore, the question component only interacts with the admin and learning module components.

Question (Information Desk):

A question in the information desk module is an object that is created when a user enters a question in the information desk forum. Its main contents are a title and description. A question object also saves user information of the user that created the question object, which is then used for showing who created the question by displaying a picture, username, and timestamp. It is viewable by admins, providers, and other users. An admin also has the ability to delete a question, which will also delete all answers attached to that question.

Answer:

An answer is what is created when an admin enters a reply/answer on a question in the information desk forum. When an answer object is created, it saves the admin's data, including username, userID, timestamp of when entered, profile picture, and the answer entered. With this data, an answer is then displayed with a picture identifying the admin as an administrator with the admin username as well. Answers are created and added on individual questions. All answers by any user type are viewable by admins. Any answer can also be deleted by an admin.

Chat:

A chat is an object created by a mobile user but is viewable by an admin for management purposes. A chat saves a userID, cohort, username, profile picture, timestamp, and message as fields. Admins can view chats based on what cohort they want to see the log for. As well, they will see who sent the chat and at what time. Admins have the ability to monitor chats by cohort and delete chats if they feel necessary.

Analytics:

The analytics module gathers information about user sessions (i.e., user's time spent in certain pages of the mobile application and their click paths), calculates different statistics about the sessions, and then displays it graphically using Chart.js for the administrators to see. The analytics module takes in a date range to show the number of page visits for certain pages within the date range, or to show how long users spent in certain pages.

Now that we have described the administrator module and how administrators will be able to interact with the web application, we will now describe the provider side of the web application.

4.1.2 Provider Web Application Module

The provider web application module is a user interface that allows providers to log into iMATter to view and respond to information desk forum questions. Providers are also able to edit their profile information, which is displayed publicly for other iMATter users to see. Figure 4.2 below illustrates the components of the provider web application module.

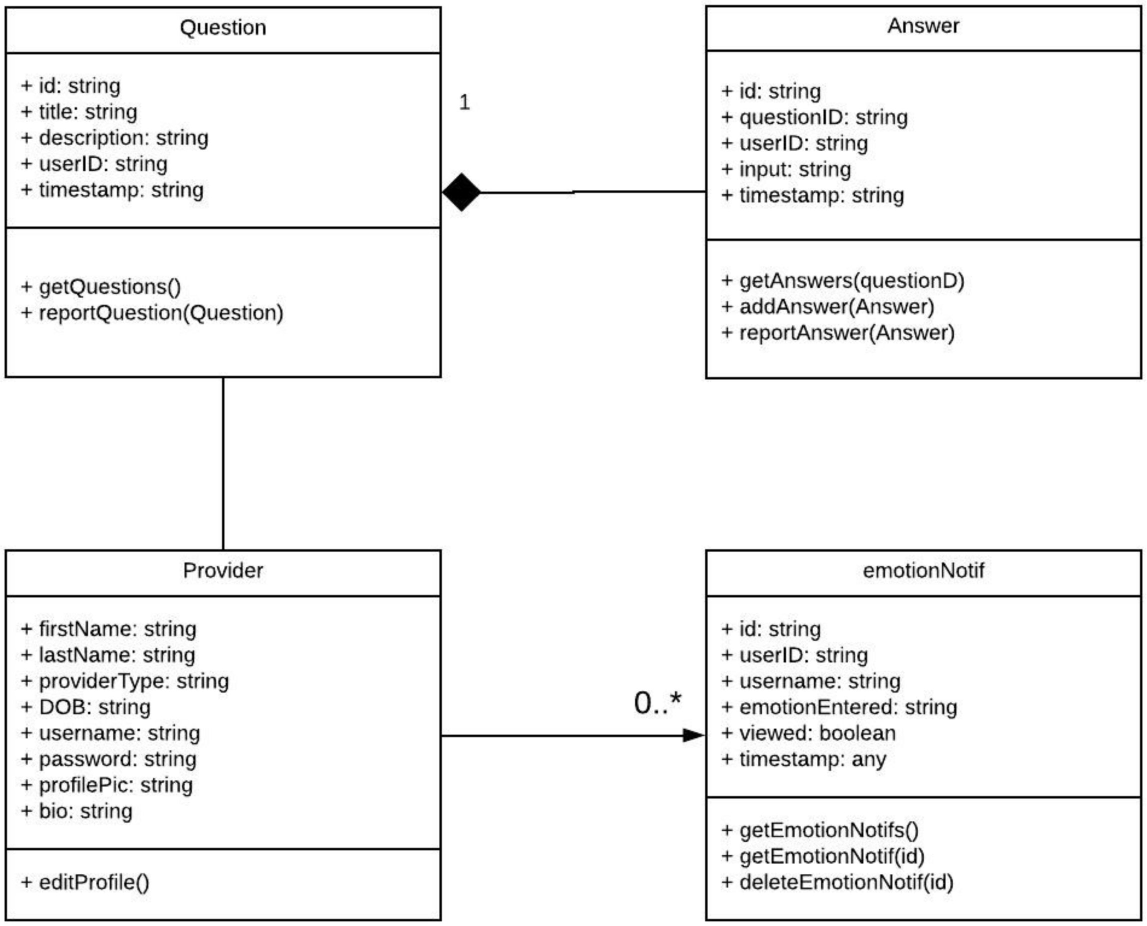


Figure 4.2 - Provider Web Application Module UML

Provider:

The provider is tasked with answering users' questions in the information desk forum. Provider's update their account when they sign up with the fields email, password, and bio.

Providers have already saved fields when they sign up, including firstName, lastName, DOB, and provider type. They have the ability to edit their own profile and can change the fields bio, email, and password. The provider interacts with users through the information desk by answering questions.

Question (Information Desk):

A question in the information desk module is an object that is created when a user enters a question in the information desk forum. Its main contents are a title and description. A question object also saves user information of the user that created the question object, which is then used for showing who created the question by displaying a picture, username, and timestamp. It is viewable by admins, providers, and other users. A provider also has the ability to report a question, which will send a message to administrators.

Answer:

An answer is what is created when an admin enters a reply/answer on a question in the information desk forum. When an answer object is created, it saves the admin's data, including username, userID, timestamp of when entered, profile picture, provider type and the answer entered. With this data, an answer is then displayed with a picture identifying the provider as the provider's saved provider type with the admin username as well. Answers are created and added on individual questions. All answers by any user type are viewable by providers. Any answer can also be reported by a provider.

EmotionNotif:

A provider receives an EmotionNotif object in their inbox when a mobile user updates their emotion to sad, overwhelmed, or angry. This allows providers to monitor for any frequent negative emotions from a user. Providers can view fields including the user's username and the emotion they updated as.

4.1.3 Mobile Application

The mobile application is what users (pregnant women in opioid recovery) will be using. This application will allow them to interact with other users and have a place to learn and keep track of events related to their recovery. Figure 4.3 below shows a UML diagram for the mobile application implementation of iMATter.

A more detailed explanation of the functionalities of the mobile application implementation is as follows:

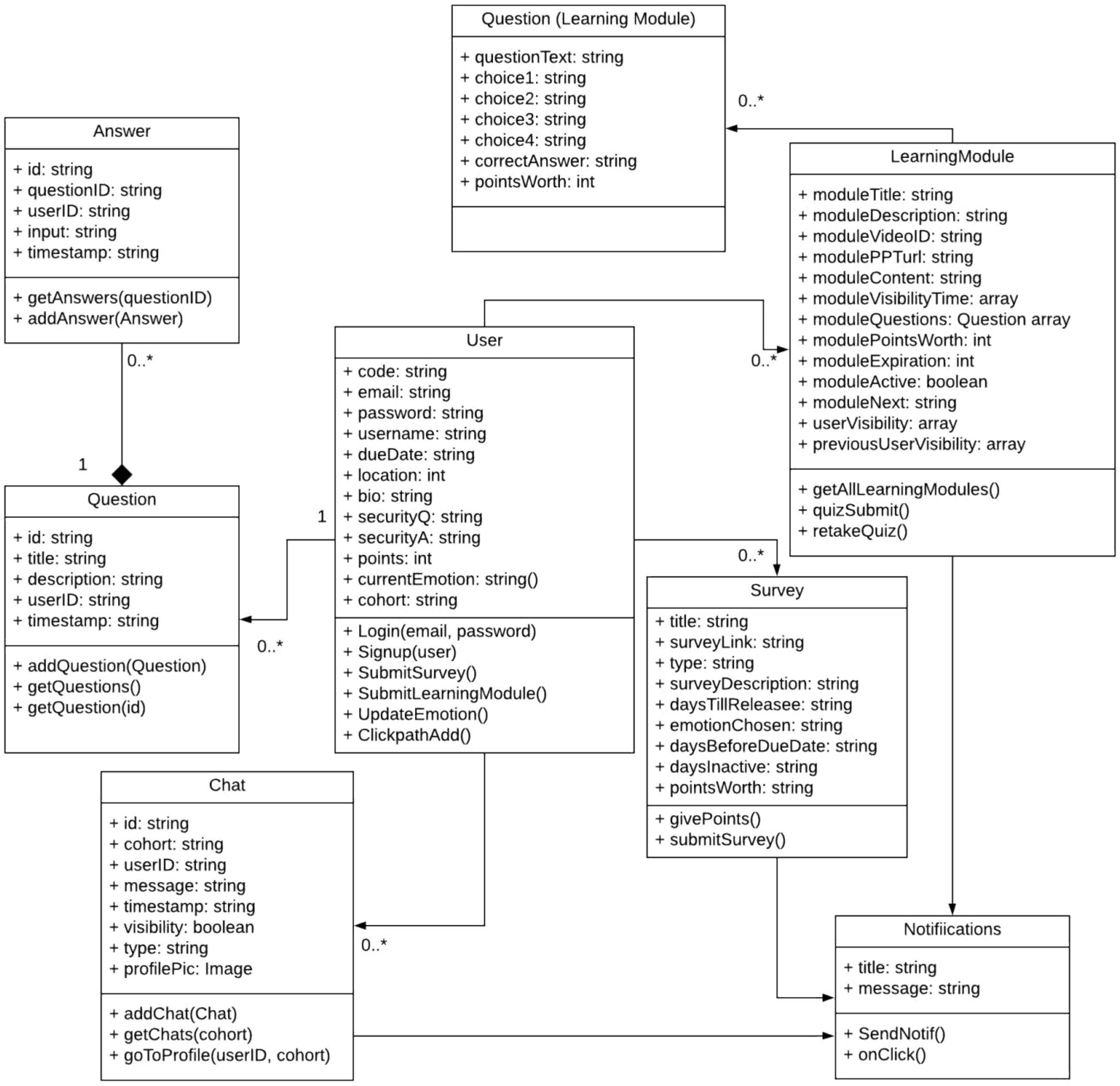


Figure 4.3 - User Mobile Application Module UML

User:

We define a user as a pregnant woman in opioid recovery that has been given access to this mobile application. They are the users that will be interacting with the classes shown in Figure 4.3. A user will be able to sign up for the mobile application after receiving a code from an administrator. Once they are signed up, they will have the ability to login to the application. Once logged in, users can interact with multiple different components. Users can update specific fields saved, including bio, email, password, and location. Users can update their emotion by clicking a button, which depending on the emotion will trigger different protocols. Users can interact with the information desk forum, which allows them to create question objects or answer objects by answering questions. Another functionality users have is creating chat objects in the chatroom, which will be sent to and visible to only other users that are also in their cohort. Users can interact with the learning module component by watching videos provided and answering questions. Users can interact with survey objects by clicking a button that will bring them to a provided URL. Users will be able to earn reward points by completing learning modules and surveys, which can then be redeemed for gift cards provided by the administrator. They will also be able to view pregnancy and recovery resources near them on an interactive map, as well as add events to a calendar and receive notifications for those events. Lastly, users will be able to configure certain settings for their application, such as turning on and off push notifications.

Survey:

A survey consists of a Qualtrics link, criteria for the user to view the object, and a time period that the survey will be displayed to users. If the user fits the criteria to view the survey, such as a specific emotion, the number of days they have been a user, the number of days they have not logged in, and the days they have left before delivery, the user will be able to view and interact with the survey. Users can take the survey and earn points for completing it. The survey object will display a button which when the user presses, will open the Qualtrics link inside an in-app browser. The purpose of this component is to provide mobile application users with a way to access surveys that administrators wish for them to regularly complete.

LearningModule:

A learning module is a page containing informational media such as videos or PowerPoints followed by questions for users to answer regarding the information presented. The learning module component in the mobile application is responsible for retrieving learning module fields from the database and displaying them for the user to see. It also checks the user's week of pregnancy against the module visibility set by the administrator to ensure that users are only viewing learning modules they are meant to. The learning module is also responsible for handling checking user's quiz question answers for correctness and displaying their quiz scores.

Notifications:

Notifications are a vital tool for engaging mobile application users with the application by reminding them of appointments or alerting them of new questions or learning materials in the application. Users will be able to receive notifications when new learning modules are added, surveys are added, a chat has been sent to the chat room, a new question has been submitted to the information desk, or calendar events are occurring. Notification objects are created using Firebase Cloud Messaging, where a title and message is set.

Question (Information Desk):

A question in the information desk module is an object that is created when a user enters a question in the information desk forum. Its main contents are a title and description. A question object also saves user information of the user that created the question object, which is then used for showing who created the question by displaying a picture, username, and timestamp. Users can also choose to set the username to anonymous.

Answer:

An answer is created when an admin enters an answer on a question in the information desk forum. When an answer object is created, it saves the user's data. With this data, an answer is then displayed with a picture identifying the user with a username as well, unless the answer is flagged as anonymous. This picture can be clicked by any user type to be brought to a viewable profile. Answers are created and added on individual questions and are viewable by admins, providers, and other users.

Chat:

A chat is an object created by a user. A chat saves a userID and cohort as fields, so users will only see chats from their cohort. The ID is used to display user information with the chat, such as a profile picture and username. Users may send chats, which saves the user's username, userID, profile picture, the message entered, and the time submitted.

Overall, we were able to implement our project off of our original architecture designs. The main differences came from adding extra fields and interfaces/objects that were necessary to implement more functionalities.

4.2 Firebase Cloud Functions

Cloud Functions in Firebase is a serverless framework. These functions are called by Firebase trigger events or HTTPS requests, allowing for backend code to run upon triggering. These functions are stored in Google's cloud, and Google handles the running of the function. We have eleven cloud functions in total. To better understand cloud functions and how to access them in the iMATter mobile repository and Firebase console, please see Appendix A. We will now go over the functionalities of these functions.

emotionSurveyNotification

This cloud function's purpose is to send a push notification to users to alert them when a new mood survey is available to them. It is invoked whenever a user's document changes in the database, as that is where the user’s mood is stored. It checks to see if a user’s mood has changed and if a mood survey fitting that mood exists. It then checks to ensure the user’s survey notifications are on before sending them a push notification.

newSurveyNotification

The purpose of this cloud function is to send a push notification to users to alert them when any new survey other than a mood survey is available to them. It is invoked via an HTTP call by the Cloud Scheduler on a regular basis. This function goes through the list of surveys and iterates through users to see which users fit the criteria for the survey. The criteria is determined based on each survey’s type. If a user fits the criteria, they will be added to the userVisibility array to indicate that this survey is visible to them, and if their survey notifications are on they will be sent a push notification. Only after users are added to the userVisibility array will the new survey be visible to them. When users are sent a notification with this function, the notification will also appear in the user’s recent notifications on the home page of the mobile application.

newLearningModuleNotification

The purpose of this cloud function is to send a push notification to users to alert them when a new learning module is available to them. It is invoked via an HTTP call by the Cloud Scheduler on a regular basis. This function goes through the list of active learning modules and iterates through users to see which users fit the criteria for a learning module. The criteria is determined by the number of days pregnant the user is. If the user is within the interval that the administrator has set for a given module, they will be added to the userVisibility array and be sent a push notification if their learning module notifications are on. Only after users are added to the userVisibility array will the new learning module be visible to them. When users are sent a notification with this function, the notification will also appear in the user’s recent notifications on the home page of the mobile application. This cloud function also keeps track of and updates the previousUserVisibility array. This array is used in the learning center in the mobile application to determine when a learning module is appearing as a recurrence.

sendProviderRecoveryEmail

This cloud function is invoked when there is a new document added to the Firestore database in the ‘recovery_email’ collection. It sends an email to the provider’s email that they have entered containing a randomly generated code for the provider to enter to allow them to reset their password.

sendRecoveryEmail

This cloud function is invoked when there is a new document added to the Firestore database in the ‘recovery_email’ collection. It sends an email to the mobile user’s email that they have entered containing a randomly generated code for the user to enter to allow them to reset their password.

sendChatNotification

This cloud function’s purpose is to send a user a push notification when a new chat message has been added to their cohort chat room. It is invoked when a new chat has been added to the ‘chats’ collection in the database. It takes the fields saved of the new message and checks if this is an automated message or user sent message first. Then, it checks all the users with the same cohort as the sent chat message and iterates through those users, checking if they have their chat room notifications set to true. If they do, the function uses Firebase’s cloud messaging service and sends it to the devices of the users.

sendEmailNotification

This cloud function sends an email to all provider’s saved in the database when a new question has been posted. It is invoked upon the writing of a new question object. It iterates through all providers and sends to the saved email address for each.

sendGCRequestEmail

The purpose of this cloud function is to allow a set administrator to receive an email when a mobile user has redeemed points for a gift card. The function is invoked when a new document has been created in the ‘userPointsRedeem’ collection. It takes the fields of the document and sends an email to the saved admin set email with the contents of the email being the user that redeemed points, which gift card they have requested, and the user’s email.

sendInfoDeskNotification

This cloud function’s purpose is to send a user a push notification when a new question has been added to the information desk. It is invoked when a new question has been added to the ‘questions’ collection in the database. It iterates through all mobile users, checking if they have their information desk notifications set to true. If they do, the function uses Firebase’s cloud messaging service and sends it to the devices of the users.

updateDays

This cloud function is invoked once a day via an HTTP by the Cloud Scheduler and is responsible for updating the totalDaysPregnant, daysPregnant, daysSinceLogin, and weeksPregnant fields of each user collection in the database. As these fields are being displayed to users of the mobile application, this function is crucial for keeping these values up to date.

deleteOldChatMessages

The purpose of this notification is to help maintain the data being stored in the ‘chats’ collection. As many chats are added to this collection, auto and user, it is a way to ensure there is no unnecessary amounts of data being stored. This function is invoked when an administrator clicks a button labeled ‘Delete Not Visible Chats’ using an HTTPS request. When this function is invoked, it collects all chats that have the visibility set to false and batch deletes them.

5.0 Testing

For this project we have implemented integration and usability tests and attempted to implement unit testing. Unit tests would have tested individual sections of our code to check that it behaved as intended and no unforeseen issues occurred. Our integration tests examined how well separate components of our project worked together to produce the desired functionality. In our case, this applies to how well web application data was relayed to the mobile application data through the database, and vice versa. Usability testing assessed how user-friendly our application is by asking people unfamiliar with our project to navigate our mobile and web applications and complete a variety of tasks. This uncovered any issues with our user interface and other software components. Usability testing was very important for our project because it ultimately simulated how our end users would be interacting with our product and gave us helpful feedback for improving our product. Given how important this was, our testing plan focused more heavily on usability testing.

5.1 Unit Testing

Unit testing breaks software down into units to be tested individually. The purpose of unit testing is to ensure that each unit of code is performing the way it is expected. We had planned to write our unit tests using Ionic’s built in page (.spec.ts file) that is automatically generated when a new page is generated. These files utilize a framework called Jasmine, which would’ve provided functions for us to properly implement our unit tests in our spec.ts file. During our attempts at unit testing, we ran into a series of issues we were unable to solve. Due to this, we were also unable to code any unit tests. For this reason, team members went through and personally unit tested modules by using them in different ways to find bugs or unexpected behavior. While not ideal, it did allow for us to ensure everything was being handled as expected and the appropriate messages were showing.

As our project consists of a web and mobile application, we tested these applications separately, including the components that are virtually the same in both applications, to verify that both are working and producing results exactly as they should. To begin, we will discuss the unit testing for our web application. As our unit testing section contains a significant number of unit tests, we will only be highlighting an example of how many of our tests worked. For this example, we will be writing about how we used unit testing to make sure creating a pregnancy update worked properly. Many of our other creation unit tests that used forms were similar. If you would like to see the more in-depth version, our software testing plan document is on our website.

Unit Test	Description	Boundary Values	Example Input	Expected Response
Create Pregnancy Update	In order to create a pregnancy update, the admin must input a day, description, and image.	-A valid day: an integer within the pregnancy days range (0-240) -A valid description: length>=1 -a valid image: Any file in an image format(JPG, PNG)	-Day *Valid:* 239 *Invalid:* two -Description: 'The baby is currently this large' -image: *Valid:* Day249.png *Invalid:* day249.rtf	-Pregnancy update cannot be created if a field is missing, or an invalid input was entered. -If all fields valid, pregnancy update is created and added to database. Image is uploaded to firebase cloud storage.

The mobile application will only be used by pregnant women that are in recovery for opioid use disorder. This is where they will view and access content being added by administrators as well as chat with other pregnant women in recovery, ask questions in the information desk forum, and more. In this next example, we will discuss a unit test for a user posting and posting an answer in the forum. This is similar to the other unit testing we did as it shows how a form must have valid input. To check out more unit testing for the mobile application, refer to our software testing plan.

Unit Test	Description	Boundary Values	Example Input	Expected Response
Submit a Question	A user can start a discussion by creating a question.	-A valid title: string of length >=1 -A valid question: string of length >=1	-Title: 'Breastfeeding on Methadone' -Question: 'Is it okay to	-User cannot submit a question if either field is empty. -If neither are empty, user can

			breastfeed on methadone?’	submit. Question will be displayed in list of question titles can be clicked on to see the question. Question is added to database.
Submit an Answer	A user can enter an anonymous answer that will be displayed under a question.	-A valid answer: string of length >= 1	-Answer: ‘Hi, I also am having this issue’	-User cannot submit answer if nothing is in the form. -If answer is not empty, user can submit. -Answer will be displayed. -Answer is added to database.

5.2 Integration Testing

We performed integration tests for all of our modules that had data being created or managed in the web application for the mobile application. This included testing our surveys, learning center, pregnancy update, and resource location modules. As the functionalities behind these modules are very similar, the tests were all generally in the same format. To provide an example of how we tested these modules, below is our integration test for the learning modules.

Integration Test	Description	Expected Response
Administrator Adds New Learning Module	An administrator creates a new learning module and marks it as active	- At least once a day a scheduled Firebase cloud function will run. Once this job runs, a push notification will be sent to users that this new learning module is available (if they fit the visibility criteria) - The learning module is made visible to users who fit the visibility criteria

Administrator Deletes a Learning Module	An administrator deletes a learning module	- The learning module is immediately removed from visibility of all users who were able to view and access it, and is deleted from the database
Administrator Changes an Active Learning Module to Inactive	An administrator changes active learning module that currently visible to some users to inactive	- The learning module is immediately removed from visibility of all users who were able to view and access it

We also created integration tests to test the communications between our mobile applications, as this is an integral part of our information desk and chat room. To provide an example of how we performed integration testing between mobile applications, here is the integration test for our cohort chat room. Mobile app users will be talking to other users in the same cohort in the chatroom. These chats are to be sending in real-time, so the users can see the chats seconds after the chat is sent. These chats are all uploaded to the Firebase database.

Integration Test	**Description**	**Expected Response**
User Sends Message in Cohort Chat Room	A user goes into the chatroom and sends a message.	- A push notification is sent to all users in the cohort. - The message is viewable by users only in the same cohort. - Admin can view the message that was sent.

5.3 User Testing

User testing allowed us to get an idea of how real users would interact with our mobile application, and what kind of obstacles they might run into that we didn't account for. User testing was done by recruiting volunteers that knew nothing about how our app worked beforehand and asking them to navigate through our app performing specific tasks without our guidance following a provided user manual. Three of our volunteers were recruited by our client, Dr. Eaves.

We performed our user testing sessions using Zoom. The sessions were recorded to allow us to be able to go back and watch the sessions again if needed; however, all members were present during both sessions taking notes. Before the scheduled sessions, participants were sent a user manual that they would be following for the sessions, allowing them time to look over and understand the actor they would be portraying. The sessions began with the team lead explaining the process, then all team members would mute their audio and stop video. We chose this way of performing the testing so we would not interfere with the testing process and could better understand what this application experience was for first time users.

After the first session, we took into consideration the feedback from the participants and improved our application accordingly. We then ran our second user testing session after our updates were made. The following subsections review our results from the user testing sessions of our mobile application.

5.3.1 User Testing First Session

From the first session, we were able to find a few bugs in our user interface, such our navigation bar disappearing occasionally, making certain pages of our application inaccessible. A piece of feedback we obtained from this session was for certain pages, such as learning modules and surveys, there were so many navigations to new pages that it was taking going back with the back arrow a few too many times to reach the navigation bar again. We also found that some fields of the calendar events would disappear when attempting to edit them, which was not ideal as even information would have to be re-entered. Other minor but important user interface changes were also suggested, such as adding toast pop-ups for confirmation for certain events. These user interface suggestions and bugs were updated before the next session.

5.3.2 User Testing Second Session

During the second session, we found from user feedback that being given confirmation of a new account being created successfully would be helpful to users. Many user interface changes to calendar events were also suggested to make it easier and less tedious to create and manage an event.

From these two sessions we were able to gather important feedback for making our mobile application more intuitive to new users. Taking this user input into account allowed us to see flaws and bugs we had not before, so we could improve our app overall. These user testing sessions were extremely helpful to us, as user experience is a major factor in having iMATter become a successful tool to aid in opioid recovery. These two sessions were very helpful, and we learned a lot from them.

Now that we have discussed our testing process and results for iMATter, we will go over the timeline of our project and some of the major milestones.

6.0 Project Timeline

Our project timeline was segmented into five sections: Outline User Interface, Web Application Development, Mobile Application Development, Database Integration, and Testing and Refinement. As a novel end-user focused product aimed at largely non-technical users, careful and early design of user interfaces and user flows was critical. Therefore, we prioritized close collaboration with our client on these early on, while simultaneously developing basic internal elements and increasing our understanding of the tasks. We developed our timeline by considering the requirements of our mobile application, web application, storing user information, our flow, and improvements that could've been done. Each of these sections were assigned smaller tasks and separated among all our team members. Figure 6.0 on the following page is a visual representation of our timeline.

Outline User Interface

During this phase we aimed to outline our user interface, which served as a guideline to account for user flow. This task was fundamental for providing a user experience that satisfied the use cases and requirements we had developed. This task included having a detailed plan on user workflow and mapping out where each feature of our application should be placed:

- General structure of web and mobile applications
- User interface mock-up for web and mobile application
- User workflow planned out from page to page

Web and Mobile Application Development

We concurrently developed the main features of our web and mobile applications during this part of our timeline. The user interface was cleaned up to be more suitable as a final product, and the web application was given administrative capabilities and the ability to communicate with the mobile application via our database. During this time, we also implemented the major functionalities of our mobile application, such as the chatroom, learning center, surveys, and information desk. This implementation served as a solid prototype, and refinements were continuously made as the semester progressed.

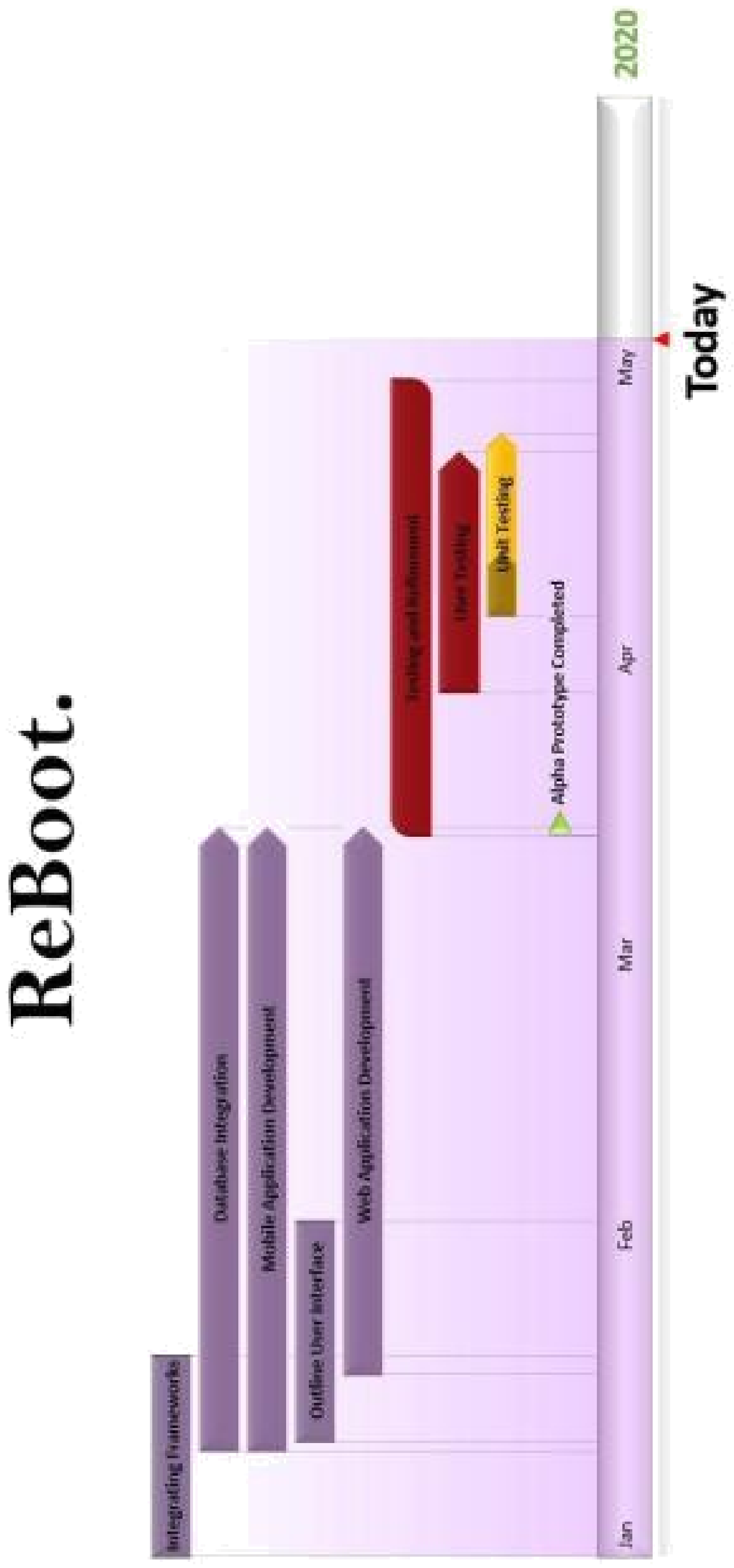


Figure 6.0 - ReBoot’s Project Development Timeline

Through all of our developments, each team member had specific areas of the mobile application they worked on, as well as its corresponding counterpart in the web application and the communication between these two parts in the database. In addition to the web application counterparts for mobile application features, some team members worked on other components of the web application.

The team was divided as follows:

	Mobile Application	**Web Application**
Bianca Altman	Account management, chat room, information desk forum, profiles, iOS, push notifications	Account management, pregnancy updates, information desk, chat room log, inbox
Jennie Dallas	Learning center, push notifications	Learning module management, Cloud Scheduler configuration
Victoria Leafgren	Calendar with local notifications, password recovery, emotion update bar	Password recovery
Isai Martinez	Survey center, recent notifications	Survey management
Cindy Valentin	Resource locations, analytics tracking	Analytics, resource location management

Database Integration

In parallel to creating all accounts, models and features for the mobile and web applications, all data was being stored in the database for future usage. Database integration was done concurrently when working on the mobile and web applications. Some usage of the database includes:

- Storing information
- Accessing the database

Milestone: Alpha Prototype Completion

As a team, completing our alpha prototype was a huge milestone for us. This part of the process allowed us to take our alpha prototype and start making changes to make it our final product. During this phase, all mobile and web applications had their main functionalities. The next phase was to test and refine.

Testing and Refinement

With the expectation of needing to make adjustments to our alpha prototype, we planned more than a month for testing and refinements. These refinements included any concerns or comments that our team and clients had. Once we made refinements, we then tested them to make sure that our new implementations cooperated with our applications.

Unit, Integration, and Unit Testing

During this part of our process, as described in section 5.0, we attempted to implement several unit tests and user testing sessions. Once completing a single test, we then made the needed changes. For our user sessions, not only did we make changes based on user feedback, but we also took into consideration how the user interacted with the application versus how we expected the user to interact with our application, which led us to make more changes.

7.0 Future Work

Team ReBoot worked to create an application for pregnant women in opioid recovery that provides a virtual environment for them to have social interactions and learning opportunities. iMATter is a great tool to assist the recovery process, and it has the potential to improve as development continues. As the application develops, the user interface should become more intuitive. While Team ReBoot was able to achieve a fairly intuitive user interface that was tested during user testing, some modules could use updating and refinements as the application grows and adds more functionality. As well, due to time restrictions, we were unable to create a private messaging platform for providers and users to talk to one another. Having a private chatting system could give users that are frequently experiencing negative emotions an outlet for direct, individual support from a professional.

8.0 Conclusion

Rising rates of pregnant women with opioid use disorder is an issue with potentially serious consequences, as it can result in newborns suffering from short- and long-term complications, including cognitive and behavioral impairments. The number of babies born with a withdrawal syndrome as a result of their mother using opioids before or during pregnancy has increased drastically over the years in the United States. To help combat this, pregnant women with opioid use disorder who want to start on the path to recovery are immersed in comprehensive treatment programs. However, while many women begin treatment, the intensity of the requirements of traditional treatment programs has caused relapse rates to be relatively high.

Through her research as a medical anthropologist, our client Dr. Emery Eaves has found that the major obstacles that contribute to high relapse rates of pregnant women with opioid use disorder are the number of weekly appointments and obligations, a lack of information access, and a lack of social support. To help address these obstacles and improve the recovery process for these women, we have developed an application called iMATter which includes the following main features:

- **Chat Room** - a place which will be able to help women in recovery interact with other women that are experiencing similar situations to offer advice and share experiences. This feature will help them receive support and advice.

- **Information Desk** - a place where a safe and reliable source of information will be found. This is where users may ask questions and have them answered from credible and knowledgeable sources. This will help women easily obtain answers to questions they have throughout pregnancy and recovery.

- **Learning Center** - a place that provides learning materials that helps users expand their knowledge of pregnancy and recovery information. This will help improve ease of access to information that pregnant women in recovery might seek out.

iMATter is aimed at revolutionizing treatment of opioid use disorder by leveraging modern web and mobile technologies to make effective treatment accessible to thousands of women, many of which currently have no practical options. Specifically, pregnant women in opioid recovery who live in rural areas or who don't have access to reliable transportation have an especially difficult time completing the requirements of treatment programs. Not only will our product help further the research of our client Dr. Emery Eaves, but it will help pregnant women in recovery receive social support, provide them ways to learn about recovery and pregnancy, and give them resources to aid in their recovery. We aim to help these women feel heard and safe, as well as provide a way for them to talk about exactly what they are going through without being judged.

Our application has the ability to be used as a model for a mobile application for addiction recovery. Using the same context from iMATter with some modifications, our clients can create other applications that could potentially impact the lives of those who are looking for a helping hand in recovering from an addiction. iMATter is a product that could expand the market of an application that aids in recovery.

Overall, this capstone class and creating iMATter was a learning experience for Team ReBoot. We learned the meaning of time management, pushed our coding capabilities to an extent we did not know we had, worked as a team, created two applications, and understood the process that it takes to create a functional software application. We are extremely proud of the product we created and will take our learned experiences moving forward.

Appendix A: Development Environment and Toolchain

This section will go over how our team prepared our environments and configured our machines for development. For this project most members used Windows and one member used macOSX. However, Ionic should run on any preferred operating system.

Development for cross-platform was necessary for iMATter, so it is important to note that continuation of development for iOS will require OSX to build and deploy.

The following steps will go over how to set up for developing with Ionic.

1. Install Git

To begin, it is necessary to install Git. Go to the website https://git-scm.com/downloads and select your operating system, then follow the directions to get Git properly installed on your machine. If you are using a Windows machine, the following commands will be run in Git Bash.

2. Install Node.js

In order to install Cordova, Node.js is required. To install Node.js, go to the website http://nodejs.org/ and install the version for your machine.

3. Install Apache Cordova

To install Cordova on your machine, in your terminal run the command

```
npm install -g cordova
```

*Note you may need to add **sudo** before npm if you do not have a Windows computer.

4. Install Ionic

To install the Ionic framework, run the command

```
npm install -g ionic
```

*Note you may need to add **sudo** before npm if you do not have a Windows computer.

Once Ionic is installed, the iMATter repository will need to be cloned or downloaded to your machine.

5. Download iMATter Repositories

The iMATter repositories can be found at: https://github.com/NAU-ReBoot

There are two repositories, one for the web application and one for the mobile application. There are two approaches to downloading the repositories.

Approach 1 - GitHub Website:

1. Go to the GitHub page for the desired repository.
2. Click the ‘Clone or download’ button (make sure you are in the master branch).
3. Click ‘Download ZIP’
4. Unzip the ZIP file

Approach 2 - Git Bash Terminal:

1. Go to the GitHub page for the desired repository.
2. Click the ‘Clone or download’ button (make sure you are in the master branch).
3. Copy the HTTPS link
4. Open the Git Bash terminal and cd to the directory you wish to save the iMATter application repository in.
5. Run the following command, substituting in the URL from step 3:

```
git clone <REPOSITORY URL>
```

6. Install Project Dependencies

Now that a local copy of the repository is saved on the machine, in order to build and run the projects the dependencies will need to be installed first. First, in the terminal cd to the directory where the iMATter project is saved. Then you will need to make sure you cd into the root folder.

The root folders are as follows:

Mobile Application

/iMATterMobile

Web Application

/iMATterWeb

Once you are in the desired root folder directory, run the command

```
npm install
```

This command will install all necessary dependencies to build and run the projects.

7. Edit and Run the Projects

To continue development on the projects, open the root directory of the project (directory cd into from step 6) in your prefered IDE. With this, development can continue. To run the projects, while in the same root directory folder, run the command

```
ionic serve
```

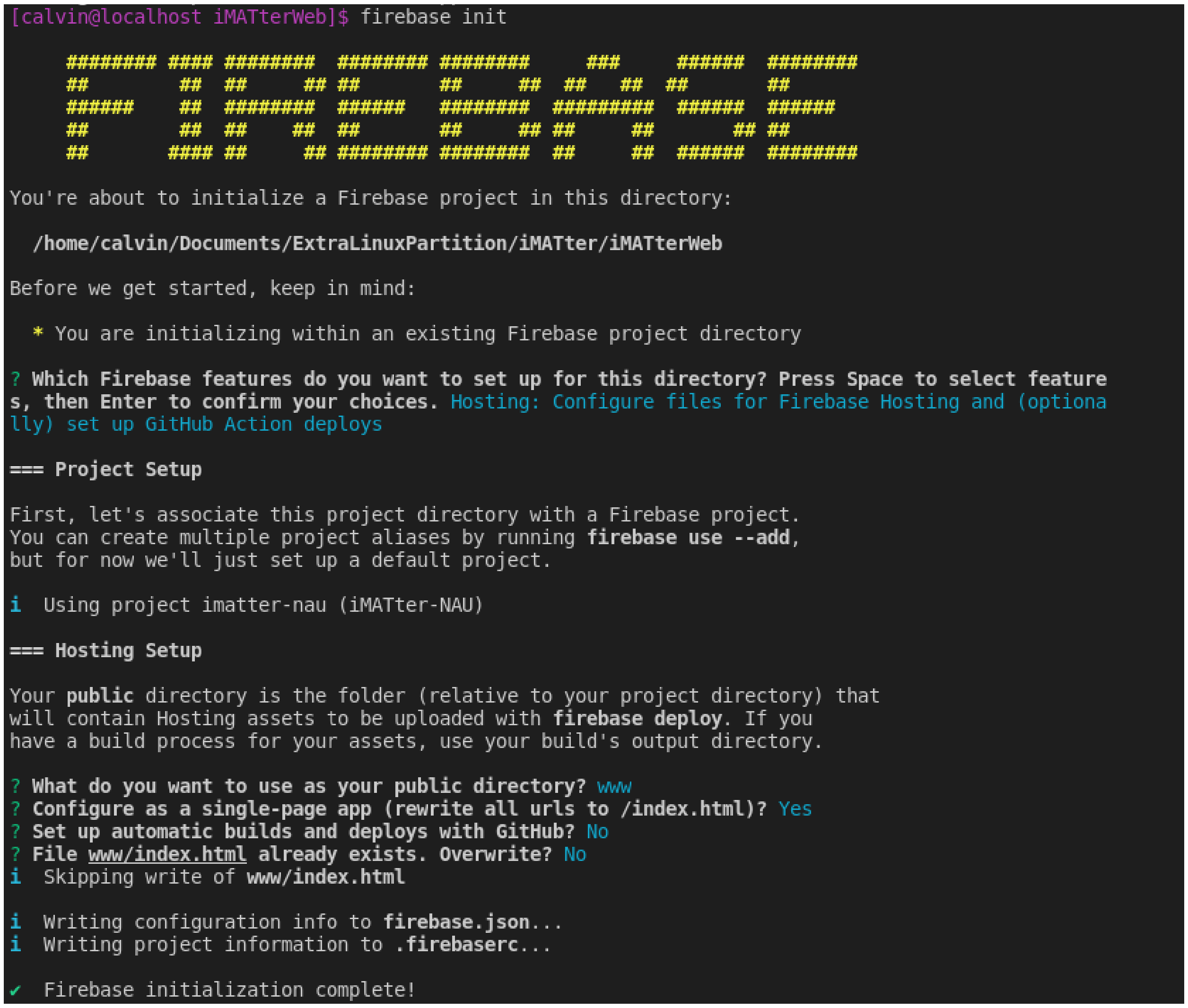
This command will run the project locally. It is a good option for development and testing as it will live update.

8. Deployment and Distribution

The iMATter website is currently live and being hosted through Firebase. There is also an APK file for the mobile application for installing and running on an Android. Steps to redeploy and create new APK files for new versions are also in the user manual, which is saved in each GitHub repository. We will go over these steps again here.

To rehost the website, follow these steps:

1. In a terminal, cd into the directory where you have saved the application folder.
2. In the root folder, run the following command:
   ```
   ionic build --prod --release
   ```
   This will generate the necessary folder (www) for Firebase to host the website.
3. If you have not already installed the Firebase client tools, enter this command: **`npm install -g firebase -tools`**
4. Now you will need to login to your firebase account
   ```
   firebase login
   ```
   This will prompt you to enter your credentials
5. Next you will enter the command: **`firebase init`**
6. This will bring up a selection for you to choose from, use the keyboard arrows to go down to the 'Hosting' option, hit space to select this, then enter.
7. You will now be asked multiple questions to properly initialize the website for deployment. These are the inputs:



```
[calvin@localhost iMATterWeb]$ firebase init

     ######## #### ########  ######## ########     ###     ######  ########
     ##        ##  ##     ## ##       ##     ##  ##   ##  ##       ##
     ######    ##  ########  ######   ########  #########  ######  ######
     ##        ##  ##    ##  ##       ##     ## ##     ##       ## ##
     ##       #### ##     ## ######## ########  ##     ##  ######  ########

You're about to initialize a Firebase project in this directory:

  /home/calvin/Documents/ExtraLinuxPartition/iMATter/iMATterWeb

Before we get started, keep in mind:

  * You are initializing within an existing Firebase project directory

? Which Firebase features do you want to set up for this directory? Press Space to select feature
s, then Enter to confirm your choices. Hosting: Configure files for Firebase Hosting and (optiona
lly) set up GitHub Action deploys

=== Project Setup

First, let's associate this project directory with a Firebase project.
You can create multiple project aliases by running firebase use --add,
but for now we'll just set up a default project.

i  Using project imatter-nau (iMATter-NAU)

=== Hosting Setup

Your public directory is the folder (relative to your project directory) that
will contain Hosting assets to be uploaded with firebase deploy. If you
have a build process for your assets, use your build's output directory.

? What do you want to use as your public directory? www
? Configure as a single-page app (rewrite all urls to /index.html)? Yes
? Set up automatic builds and deploys with GitHub? No
? File www/index.html already exists. Overwrite? No
i  Skipping write of www/index.html

i  Writing configuration info to firebase.json...
i  Writing project information to .firebaserc...

✔  Firebase initialization complete!
```

- As you can see, I selected the **Hosting: Configure files for Firebase Hosting and (optionally) set up GitHub Action deploys** option although it will look different when you select it.
- And then near the bottom in the blue text, you can see the options I selected for the public directory, SPA, auto builds, and index.html. Choosing to not configure the site as single page application (SPA) will make it so reloading anywhere on the site causes a 404 error.

8. Last, you will enter the command:

```
firebase deploy --only hosting
```

To create a new .APK file for distribution, follow the steps:

1. In a terminal, cd into the directory where you have saved the application folder.
2. In the root folder, run the following command:

   ```
   ionic cordova build --release android
   ```

   This will generate the unsigned .APK file.
3. After the file has finished generating, run the following command:

   ```
   jarsigner -verbose -sigalg SHA1withRSA -digestalg SHA1 -
   keystore my-release-key.keystore
   ```

```
platforms/android/app/build/outputs/apk/release/app-
release-unsigned.apk alias_name
```

This will sign the .APK file.

4. For this .APK to be able to be properly downloaded, you will have to run one last command:

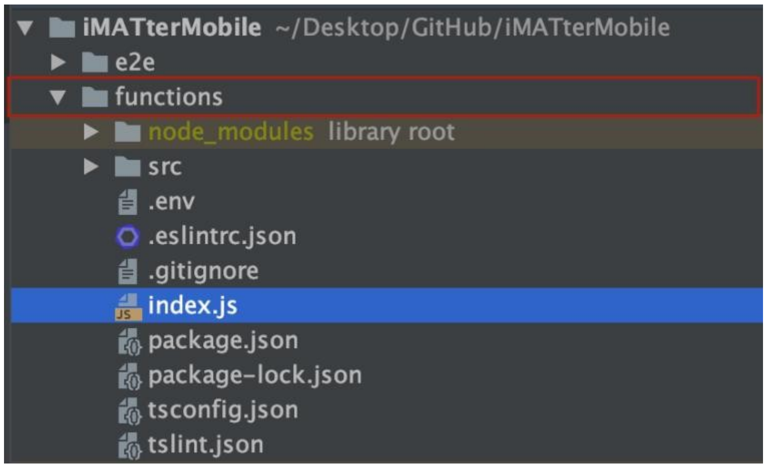
```
platforms/android/app/build/outputs/apk/zipalign -v 4 platforms/android/app/build/outputs/apk/release/app-release-unsigned.apk iMATter_V#.#.apk
```

Where the #.# is replaced with your current version number of the application.

After following these steps, the new version will be listed in the files of the root folder as '**iMATter_V#.#.apk**'. This is the file you will distribute.

9. Firebase Cloud Functions

Firebase cloud functions can be found under the functions folder of the iMATterMobile folder. In this folder, there is an index.js file, here is where you will find where the functions are written. This can be seen in Figure 9.0.



```
▼ iMATterMobile ~/Desktop/GitHub/iMATterMobile
  ▶ e2e
  ▼ functions
    ▶ node_modules library root
    ▶ src
      .env
      .eslintrc.json
      .gitignore
      index.js
      package.json
      package-lock.json
      tsconfig.json
      tslint.json
```

Figure 9.0 - iMATterMobile Cloud Function index.js file

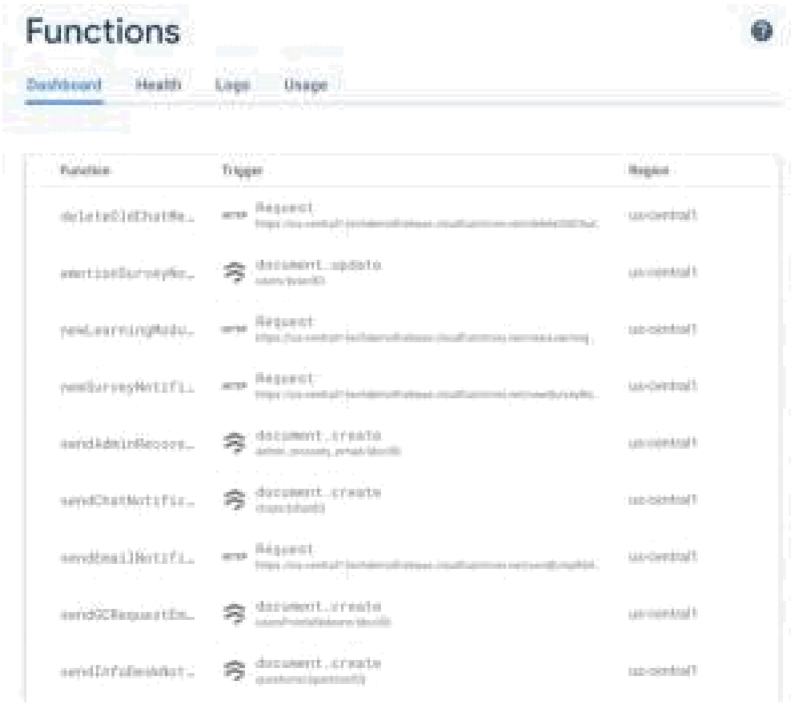
Here is where the cloud functions may be edited. Once ready to deploy the function to the cloud, run the following command:

```
firebase deploy --only name_of_function_deploying
```

Example: firebase deploy --only functions:sendChatNotifications

This function will only deploy the function specified. To redeploy all functions, run:

```
firebase deploy
```



Function	Trigger	Region
deleteOldChatMe…	Request [illegible]	us-central1
advertiseSurveyNo…	document.update [illegible]	us-central1
newLearningModu…	Request [illegible]	us-central1
newSurveyNotifi…	Request [illegible]	us-central1
sendAdminReoccu…	document.create [illegible]	us-central1
sendChatNotific…	document.create [illegible]	us-central1
sendEmailNotifi…	Request [illegible]	us-central1
sendGCRequestEm…	document.create [illegible]	us-central1
sendInfoDeskNot…	document.create [illegible]	us-central1

Figure 9.0 - iMATterMobile Cloud Function index.js file

To view deployed cloud functions, go to the Firebase console as seen above in Figure 9.0. In the 'Logs' tabs, it can be viewed when functions run and if they throw an error or not.

10. Google Cloud Platform

iMATter utilizes Google APIs for Google Maps as well as accessing Cloud Scheduler capabilities for sending and configuring when notifications are sent for learning modules and surveys. The API keys and OAuth credentials for these APIs can be found and managed here: https://console.cloud.google.com/apis/credentials?folder=&organizationId=&project=imatter-nau

The Google Cloud Platform console was also used to create the scheduled jobs for a few of our cloud functions. Those can be found and managed here: https://console.cloud.google.com/cloudscheduler?folder=&organizationId=&project=imatter-nau

11. Creating a New Splash Screen With New App Versions

1. First open the config.xml file in the projects root folder
2. Now change the version number in line 2 of this file.
    1. `<widget android-versionCode="12" id="com.nau.iMATter" version="5.4.1" xmlns="http://www.w3.org/ns/widgets" xmlns:cdv="http://cordova.apache.org/ns/1.0">`
3. Now the about page will grab the version number from this file for display.
4. Now, delete splash.png from /resources/ folder.
5. Now run the command
    1. **convert -pointsize 72 -fill \'#494949\' OriginalResourcePictures/splashOrg.png -gravity south -annotate +0+625 ' Version 5.4.1' resources/splash.png**

2. -pointsize is fontsize

3. OriginalResourcePictures/splashOrg.png is the non-edited splash screen that we want to edit with a version number

4. -gravity south centers the text in the middle of the page, I think

5. annotate +0+625 specifies a coordinate ie +x+y. So we place the version number at x=0 y=500 on the image.

6. ‘5.4.1’ is the version number

7. resources/splash.png is the output image.

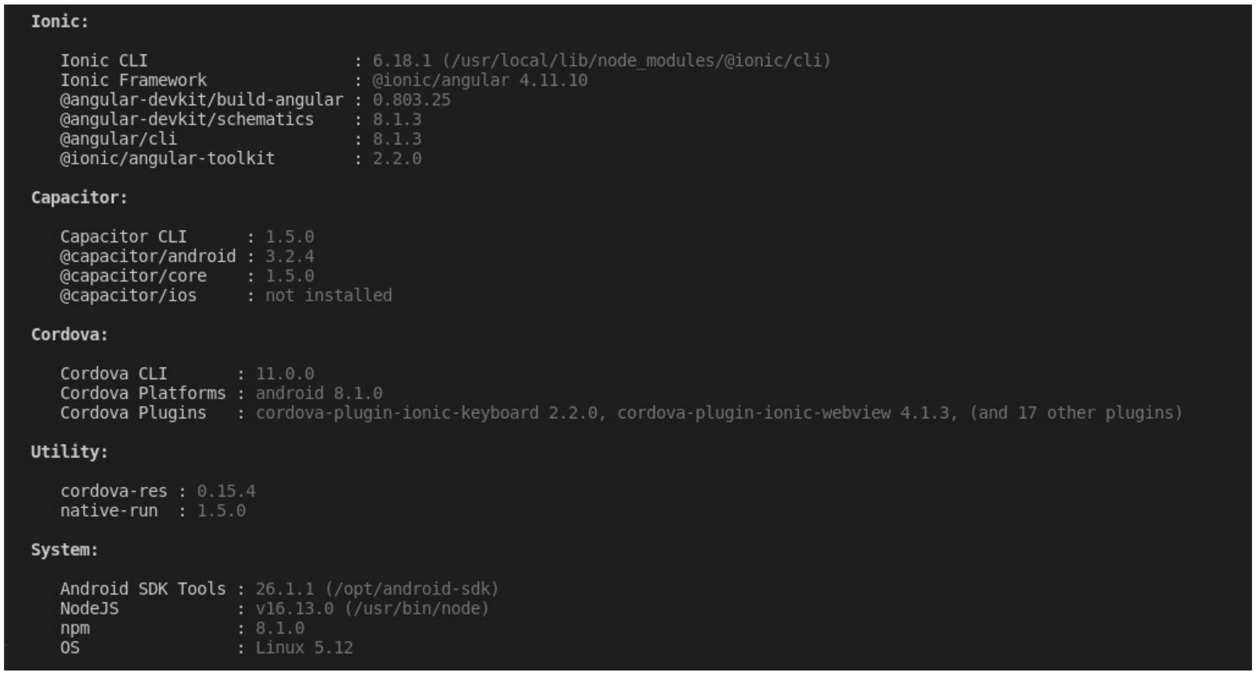
8. Now run **`ionic cordova resources`** `to refresh the resoruces used to build the app. You WILL NOT see the new splash screen if you don’t run this command.`

9. `You might get some errors when running the` **`convert`** `command, if you do make sure to check if the picture was created because in most cases the picture still gets made correctly if you get erros.`

10. Enjoy the new splash screen.

12. Ionic Info Output on Working Build

1. Be extremely cautious when changing the android platform version, the gradle version or any other build tool version. It can be extremely frustrating to debug and you may need to re-install your tools such as cordova using NPM including re-downloading the repo. Make sure you’re on a separate branch when trying to upgrade anything and be willing to sink hours into troubleshooting possibly. Also, do not upgrade any of these tools without purpose as it will only make your life harder.
2. I changed the version of some of these build tools and changing them back did not even fix the issue.



```
Ionic:

   Ionic CLI                     : 6.18.1 (/usr/local/lib/node_modules/@ionic/cli)
   Ionic Framework               : @ionic/angular 4.11.10
   @angular-devkit/build-angular : 0.803.25
   @angular-devkit/schematics    : 8.1.3
   @angular/cli                  : 8.1.3
   @ionic/angular-toolkit        : 2.2.0

Capacitor:

   Capacitor CLI      : 1.5.0
   @capacitor/android : 3.2.4
   @capacitor/core    : 1.5.0
   @capacitor/ios     : not installed

Cordova:

   Cordova CLI       : 11.0.0
   Cordova Platforms : android 8.1.0
   Cordova Plugins   : cordova-plugin-ionic-keyboard 2.2.0, cordova-plugin-ionic-webview 4.1.3, (and 17 other plugins)

Utility:

   cordova-res : 0.15.4
   native-run  : 1.5.0

System:

   Android SDK Tools : 26.1.1 (/opt/android-sdk)
   NodeJS            : v16.13.0 (/usr/bin/node)
   npm               : 8.1.0
   OS                : Linux 5.12
```