Software Design

Team Shining Sky

Rosze Voronin, Skyler Hanson, Ashleea Holloway, Logan O'Donnell

February 14th, 2022 - Version 2

Sponsored by: Dr. Morgan Vigil-Hayes

Mentored by: Felicity Escarzaga

1. Introduction

Rural Native Americans struggle daily with mental health, with youth being susceptible to rising rates of depression, anxiety, and behavioral issues. This is due to Native American communities often not having funding for mental health resources, and a lack of network connectivity preventing members from accessing online help. Native American communities therefore may seek help from academic research labs and other sources of low-cost software development to attempt to address these problems.

The client, Dr. Morgan Vigil-Hayes, is the director of CANIS Lab (Community Aware Networks and Information Systems) and the founder of the ARORA mobile application. Dr. Vigil-Hayes' research, funded by issued research grants, focuses largely on asynchronous networking and network infrastructure that does not rely on a constant internet connection. This research is primarily used to address the lack of network connection within rural Native American communities. CANIS lab's goal is to take the products developed for the tribal communities such as the Navajo Nation and the Hopi tribe and genericize them to be offered to other Native American communities.

One such product is the ARORA mobile application, currently being used by Hopi's younger generations. The ARORA app represents a resource for Hopi youth to track their mood and stress levels over time and engage in activities that may improve mental well-being. The goal of the app is to aid in preventing the decline of mental health of Native American youth, however it is still missing some features. This includes the lack of connection between users and external mental health resources, including mentoring. The Hopi Tribe is currently creating a Community Mentorship program that will connect community mentors with Hopi youth mentees, but the program still needs a software tool to aid in communication and organization that the existing ARORA app does not provide.

1.1 Problem and Solution

The mentorship app within this document will be a separate sister application to the ARORA app that addresses the missing features needed for Hopi's Community Mentorship program. The Hopi Community Mentorship program does not have a technological or physical resource that connects mentors with younger Hopi youth (now mentees within the program) to

address mentees' declining mental health. CANIS Lab's ARORA app was only made for mentees to use to track their own mental health, so the mood reports and stress levels reported within the app are not currently viewable to potential mentors within the mentorship program. Therefore, the ARORA app is currently missing ways for mentors to:

- View mood report data of mentees from the ARORA app.
- Track mentees showing signs of struggling with mental health.
- Communicate with mentees that show signs of struggling with mental health.
- Answer questions mentees may have related to mental health.

The solution to these missing features is the mentorship app outlined in this document. The mentorship app will provide a clean UI that organizes mentee data and allows mentors to directly communicate with mentees that may be struggling with their mental health. The mentorship app will include features such as:

- A login system to ensure that only authorized mentors and supervisors of the mentorship program can access the application.
- Mood report and stress level mentee data from the ARORA app organized in list format filterable by the mentor.
- A question list system that displays anonymous questions sent in from mentees or other Hopi youth for the mentor to respond to.
- A chat system for mentors to directly communicate with mentees.
- A calendar system for mentors to create and set appointments with mentees for further communication.

2. Implementation Overview

In order to provide the missing features, developers are producing the Community Mentor Portal mobile app. The app targets the functionalities that the Community Mentorship program needs to be more effective, providing a way for mentors to see the data that their mentees provide about their mental health through mood reports, initiate communication via chats and setting of appointments, and answer anonymous questions. Some major elements of the in-progress implementation are as follows:

React Native, a mobile app framework, enables exporting to both iOS and Android. It also can produce a web app, which the client has considered developing further after completion of the mobile version. Due to the framework having a large community, various feature packages are available to use, which aids implementation and allows developers to include as many features as possible from the wishlists of the client and the Hopi mentorship program. React Native also provides the majority of UI elements that contribute to the implementation of dynamic lists, page navigation, and overall usage design.

The “react-native-gifted-chat” package by Stream is currently being explored as an option for the chat page within the app. Chat implementation will enable the mentor user to send and receive messages from their mentees, as well as view their message history. In addition, supervisor users will be able to view chat logs in order to keep track of the mentors under them, helping to ensure that youth in the program are being handled safely and receiving adequate advice.

A simple calendar view will be included so that mentors can set appointments with their mentees and keep track of when they are and what they will be about. This will ensure that the mentors can remain organized and not have to dig through chats or emails to remember their appointments. Including it in the Community Mentor Portal app also contributes to the overall design philosophy of a software tool that allows mentors to have most of what they need in one place.

The SHA1 Encryption Algorithm is a simple scheme developed by the NSA that will be used for password hashing in the mentor portal app. As the mentorship program deals with mental health and includes personally identifying information in some places, ensuring that only

approved mentors have access to mentee data, and only for their own mentees, is important in ensuring that the program remains a safe option for mentees.

The app will use SQLite for a local database to store mentee mood reports, account log-ins, and anonymous questions sent in for mentors to answer. Storing relevant information locally allows for less required network transactions, which is important in a rural area with less consistent connectivity.

The Recovery Email API is used to allow the app to send recovery emails when a user has forgotten their password.

3. Architectural Overview

In order to visualize the main components of the mentor mobile application, Figure 3.1 and Figure 3.2 demonstrate a color-coded diagram of the app's navigation representing the two user types and their respective view of the app. The two user types are a mentor who views mentee information and a supervisor who views mentor information. Each color-coded level represents the depth a user must go to reach that screen. Each color-coded module represents a screen that can be navigated to using the navigation bar, buttons, or other UI elements. The navigation bar module is the heart of the navigation system and can be accessed on multiple screens from multiple levels, so it is not included.

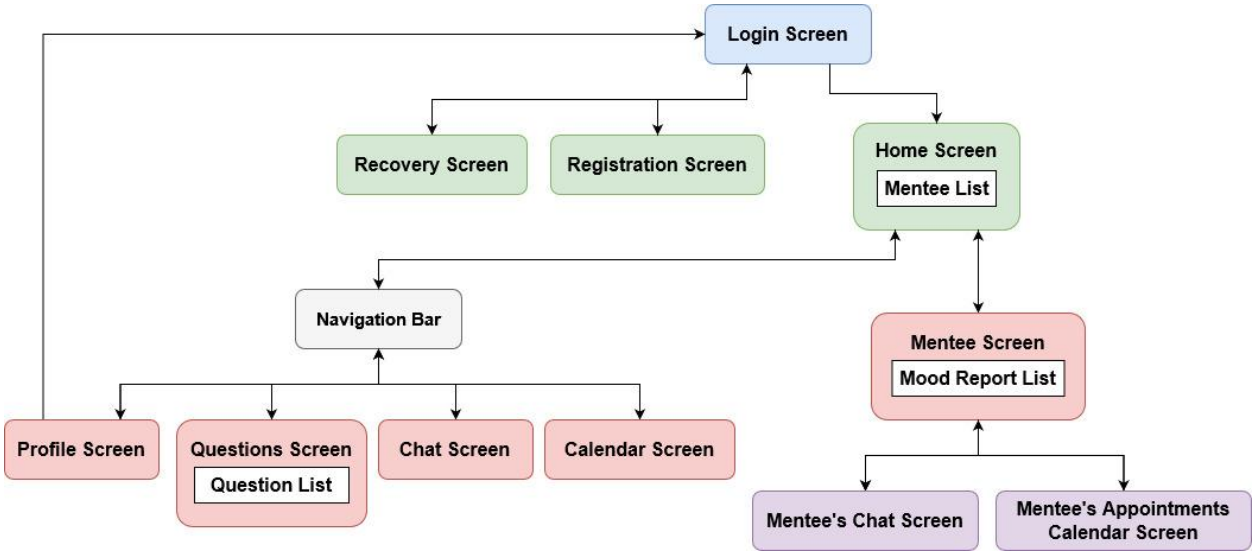

Figure 3.1 Architecture Diagram for Mentor View

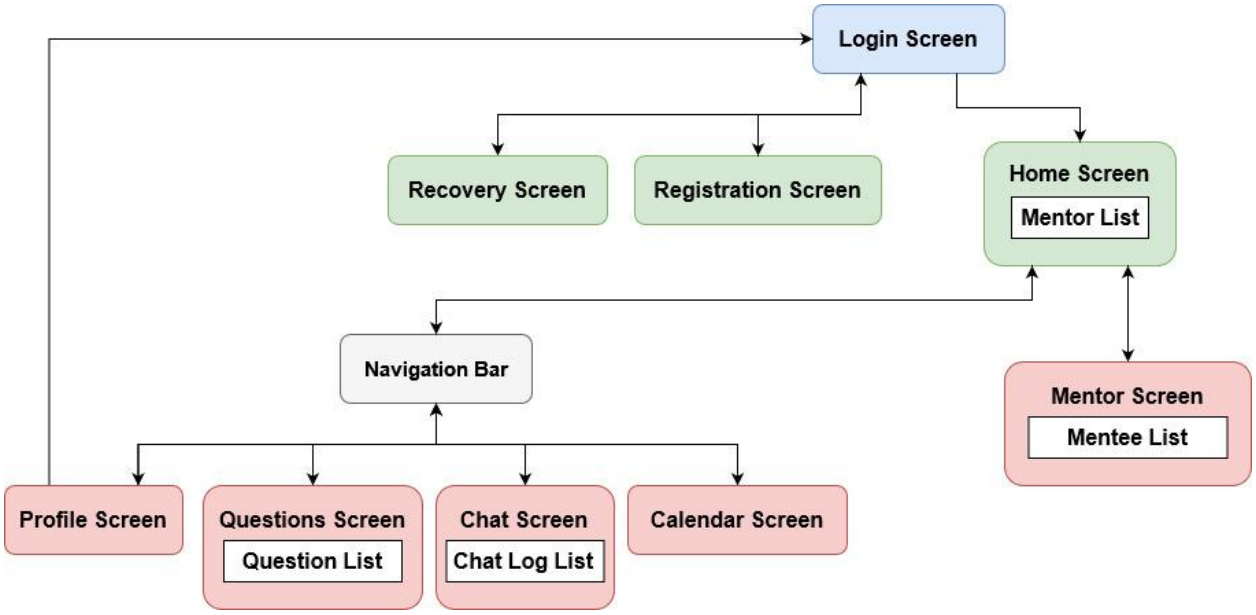

Figure 3.2 Architecture Diagram for Supervisor View

Each module has its key responsibilities and functions, with some navigation triggering the generation of information onto the next page. The overall architecture of the mobile application follows a three-layer architecture pattern, in which implementation is divided into the presentation layer, the business layer, and the data layer. The presentation layer will represent the visual UI that the app's user will see and use. The business layer conducts the beginning of back-end implementation, which would include saving chat logs between mentor and mentee. Finally, the data layer consists of protecting the database that holds mentee information, account information, and all data stored for the app's use. This would be the interaction between the implementation and the database, such as when the mentee screen must pull mentee information from the database for the mentor to view. The architectural three-layer pattern is demonstrated by the screens and their inner modules and can be described as follows:

Level 1 (Blue)

- Login Screen where users must log in with a username and password to access the rest of the app.

Level 2 (Green)

- Recovery Screen where users enter an email to recover their credentials.
- Registration Screen that enables the user to create an account.

- Home Screen is accessed from the login screen if correct credentials are entered and displays a list of the mentor's mentees that are pressable and can navigate to a mentee screen.

Level 3 (Red)

- Profile Screen that displays the user's account information. They may change or view their credentials. Logging out here will send the user to the login screen.
- Questions Screen that displays a forum-style list of anonymous questions sent in for mentors to answer. The mentor can view, answer, or flag questions for supervisor review.
- Chat Screen that allows the mentor to view a list of initiated chats between themselves and mentees and send/receive messages. For supervisors, this screen allows viewing of chat logs.
- Calendar Screen that displays a simple calendar allowing mentors to manage events such as appointments with mentees.
- Mentee Screen that displays the name, mood reports, and mental health status of the mentee clicked on. For supervisors, this screen allows viewing of which mentees are assigned to a particular mentor.

Level 4 (Purple)

- Mentee's Chat Screen that is accessed from a mentee screen, displaying that mentee's chat messages with the mentor.
- Mentee's Appointments Calendar Screen that is accessed from a mentee screen, displaying that mentee's appointments with the mentor.

4. Module and Interface Descriptions

Each module, represented as a screen, can be technically described by defining the functions and processes within. The modules and their inner subfunctions are as follows:

Login

This module allows the user to log into the app if they are an authorized user. It includes text boxes to store and send username and password, and buttons to navigate to registration, password recovery, and for submitting login information. SHA1 Encryption will be used to obscure sensitive log in data before it is sent to and checked against the database, in order to reduce the risk of a data leak. Encrypted login information will be stored in a local SQLite database (using functionality from version 6.0.1 of the react-native-sqlite-storage package in order to load the database and run queries during runtime) in order to enable increased functionality when network connectivity is unavailable. When the client continues development on the project, the plan is for login information to be also stored on their central server so that local databases have something to check their information against.

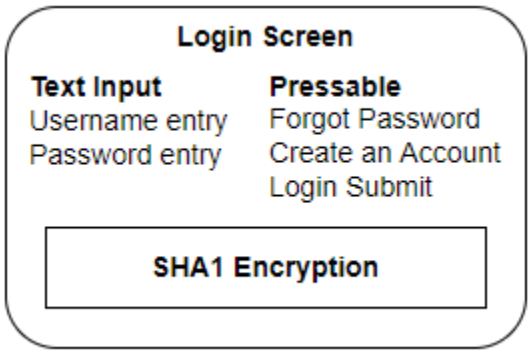

Figure 4.1 Login Screen Module Diagram

Registration

This module allows the user to register for an account, provided they are an authorized user with an access code. Access codes are generated in the supervisor view and create a blank row in the database's user table to accommodate the new user information that the registrant inputs. The module's UI includes text boxes to store access code, first name, last

name, and email. Once a user registers successfully, they are placed back at login so that they can log into the app.

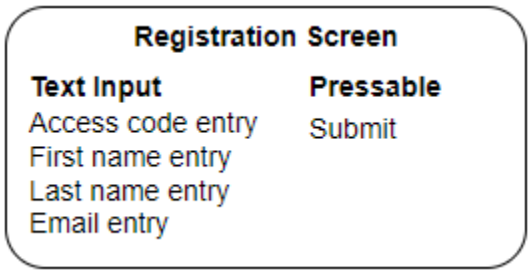

Figure 4.2 Registration Screen Module Diagram

Recovery

This module allows the users to recover their account by requesting a password reset email. It includes text boxes to store email addresses to be used to reset passwords. It also includes a button to submit a text form which will begin the password reset process. Emailing API will be used to send the recovery emails. A check is run against the database in order to identify the account the email is associated with.

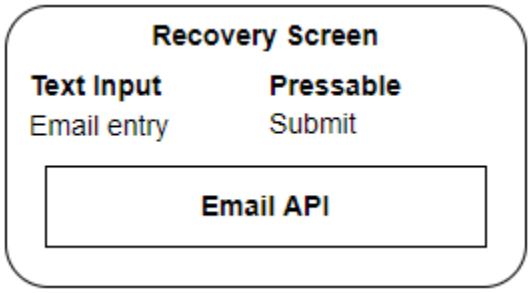

Figure 4.3 Recovery Screen Module Diagram

Home Screen, Mentee List

This module allows the mentor to view a list of their mentees. The mentee names are pressable to generate a screen of their information via a call to the database. It includes a Mentee List component that uses the map function. Mentee Names and their icon (using an image component to display their current mental health risk: red, yellow, or green), represent the information pulled from the database. This data will be stored within the local database, and

added to when network connectivity is present once the client seats the application on their central server. The navigation bar using React Native's Bottom Tab Navigation is located at the bottom of the screen.

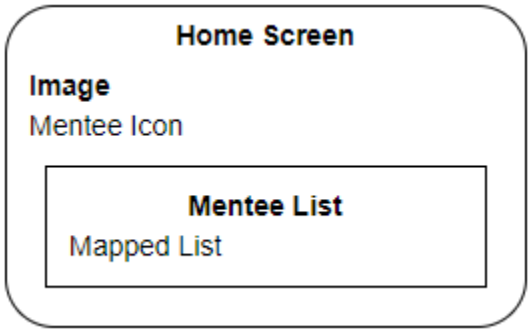

Figure 4.4 Home Screen Module Diagram

Home Screen, Mentor List (Supervisor View)

This module allows a supervisor to view the list of active mentors. The mentors' names are pressable to generate a screen of their information including assigned mentees. It includes a Mentor List component that uses the map function to display the mentor names. Mentor and mentee information is populated via a call to the database. The navigation bar using React Native's Bottom Tab Navigation is located at the bottom of the screen.

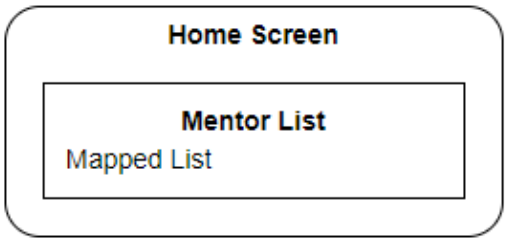

Figure 4.5 Supervisor Home Screen Module Diagram

Mentee Page, Mood Report List, and Mood Report

This module allows the mentor to view a mentee's information. Pressables are used to flag a mentee, open the chat between the mentor and mentee, and open appointments between the mentor and mentee. There is an icon using an Image component that displays the mentee's current mental health (red, yellow, or green). All information is populated using SQL queries on

the local database, with the client's future plan being to update the locally stored information via their central server. There is a Mood Report List component that uses the map function to display a list of the mentee's mood reports. The Mood Report List displays a list of the Mood Report component, which comprises the date posted, the mood level, the stress level, and an automatically generated risk level.

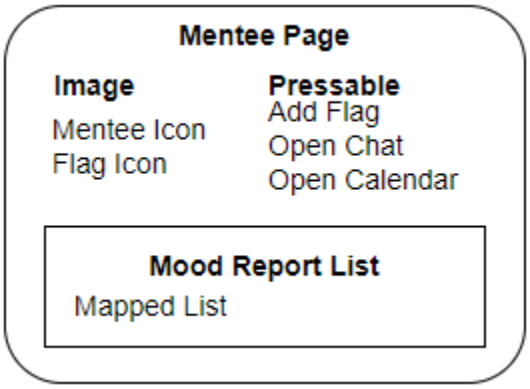

Figure 4.6 Mentee Page Module Diagram

Mentor Page, Mentee List (Supervisor View)

This module allows the supervisor to view a mentor's mentees. It uses the Mentor List component that uses the map function to display a list of mentee names. As with the previous section, information is pulled from the local database with future plans to include information updates from CANIS Lab's central ARORA server.

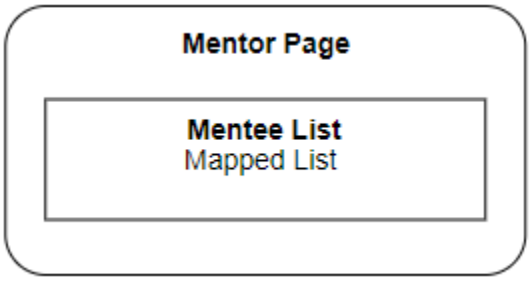

Figure 4.7 Mentor Page Module Diagram

Questions Screen, Question

This module allows the mentor or supervisor to view a forum-style list of questions. The Question List component uses the map function to display each question. The Question component consists of the question text, a flag pressable, a text box for the mentor to write a response in, and a pressable to submit the response. The navigation bar using React Native's Bottom Tab Navigation is located at the bottom of the screen. Questions will be stored in the database, with additional topic-tagging so that questions can be piped to mentors who may be more familiar with a certain issue and thus able to better answer them. In the absence of that, questions will be delivered using a Round Robin methodology for even distribution.

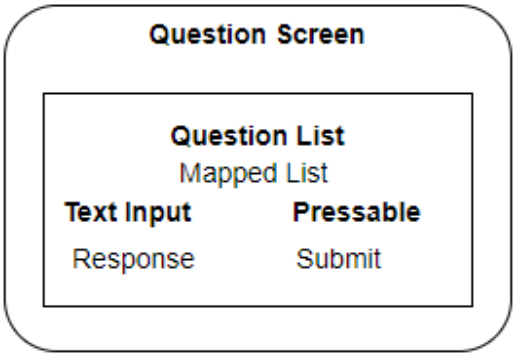

Figure 4.8 Question Screen Module Diagram

Chat Screen, Chat

This module uses a chat package to represent a traditional chat window for chats between the mentor and the mentee. The Chat component representing each message consists of the message text, time sent, and the delivery success result. The navigation bar using React Native's Bottom Tab Navigation is located at the bottom of the screen. Chat logs will be stored as an encrypted file, allowing for reloading later or opening by the supervisor in order to monitor mentor behavior surrounding mentees.

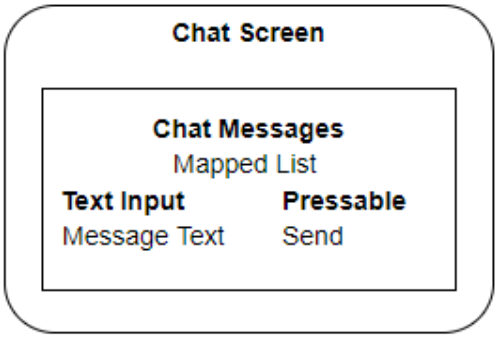

Figure 4.9 Chat Screen Module Diagram

Chat Screen (Supervisor View)

This module displays a list of mentors and their respective chat logs with mentees. The Mentor List component and Chat Log List component use the map function to display items. Each mentor name is pressable to open the Chat Log List component, and each chat log is pressable to open a log text file. The navigation bar using React Native's Bottom Tab Navigation is located at the bottom of the screen.

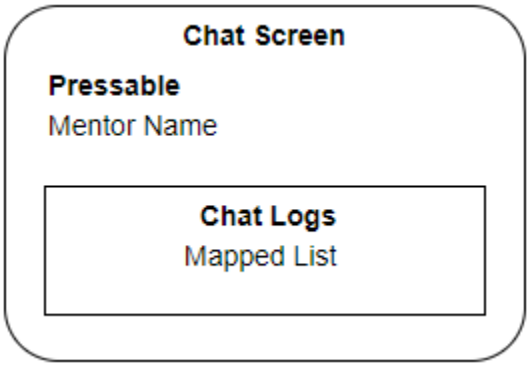

Figure 4.10 Supervisor Chat Screen Module Diagram

Calendar Screen/ Calendar Event

This module represents a calendar view that allows users to view, set, and delete appointments. Pressables are used for the set, delete, and save buttons. The navigation bar using React Native's Bottom Tab Navigation is located at the bottom of the screen. Appointment information will be stored in the local database, and per the client's request there are currently no plans to offer any complexities like calendar exportation or other features.

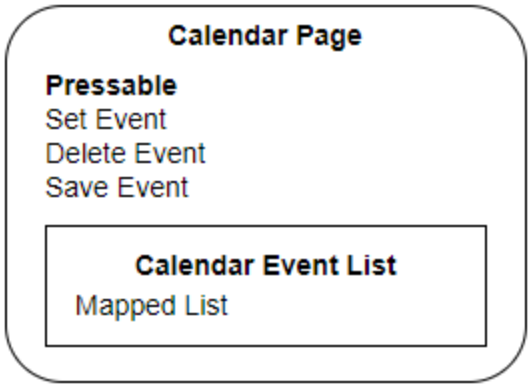

Figure 4.11 Calendar Screen Module Diagram

5. Implementation Plan

The implementation plan for the previously discussed modules has been laid out on a weekly basis, with implementation being focused on until mid-March (UGrads), and final documents, tweaks, and user testing occupying the time beyond that. So far through January, design for the modules was worked on with the client's input, and some of the core screens and their inner modules within the app, including Login screen, Home screen, Question Screen, and Mentee Screen began to be developed. At the beginning of February, most of the mentor screens will be finished while the Chat module, Calendar module, Recovery module, Registration module, and Supervisor View are being worked on. Below (Figure 5.1) is a visual representation of the order in which modules were started or will be started, and the expected time implementation will end.

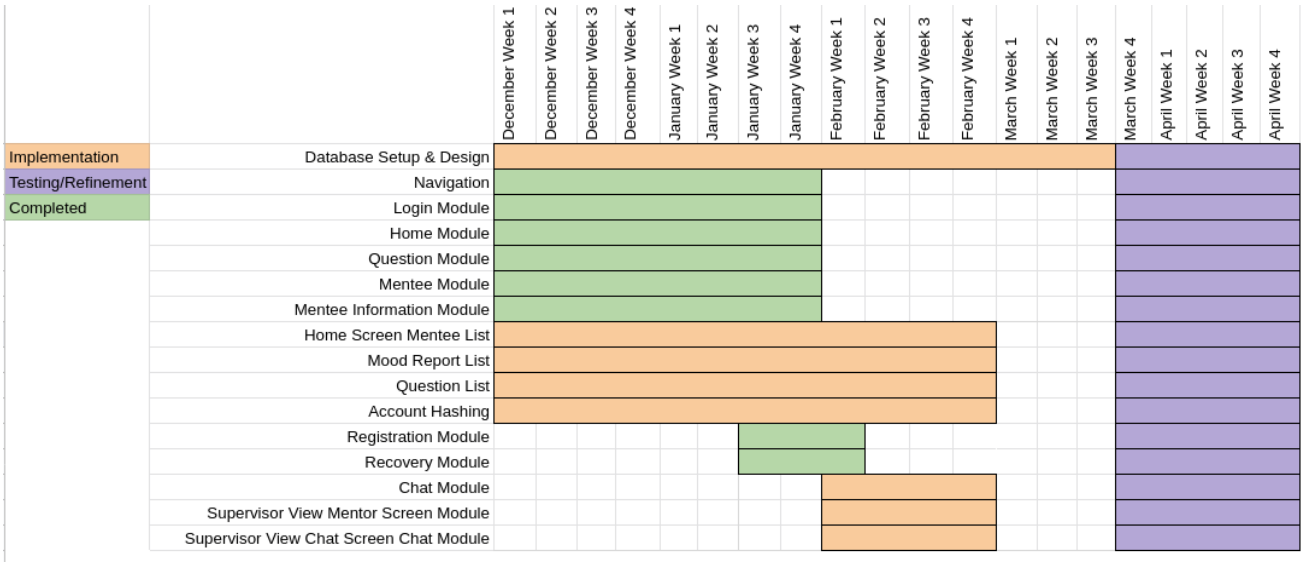

Figure 5.1 Implementation Plan

So far, implementation has focused on the modules currently showing as completed in Figure 5.1. At the moment, the team is working to complete the local database module (for which additional items will be added as needed for new features), allowing for completed modules to have their SQL calls added in. In addition, a separate view for supervisor accounts is being implemented. After that, implementation will turn to the Chat and Question List modules. Further down the line, the last few features will be implemented and early testing will begin, including tests on a few mobile devices provided by the client. Towards the end of the

semester, implementation will focus on final tweaks, adding in assets that better match the ARORA aesthetic (client-provided), and documentation for the handover to CANIS Lab.

6. Conclusion

The ARORA mentor app is going to close the final technological gap between a beneficial community mentor program and the youth of the Hopi Tribe. The client, Dr. Vigil-Hayes of CANIS lab, is aiming to achieve this through the relationship between a new mentor-focused app and CANIS lab's existing youth-focused app, which the mentees will use. Currently, Hopi youth struggle with higher rates of suicide, depression, anxiety, and substance abuse due to the lack of mental health resources within their community. The Hopi community mentor program, made up of older Hopi individuals trained in providing mental health resources and guidance, needs a utility to further reach and connect to the Hopi youth in need. The Community Mentor Portal app project, as an expansion of the ARORA project, will represent such a utility, with mentors able to view youth mentees' mood reports, answer questions they may have, and allow themselves to easily contact youth or be contacted.

Team Shining Sky is currently working to implement the ARORA mentor app. Part of our process has been to lay out the implementation plan for each feature, assess our overall architecture, and then divide everything into modules to more fully plan each portion of the mentor app. With the app's feature set and implementation plan firmly defined, development can continue to progress, and the team is confident that the extra organization will help ensure the project remains on track for a timely and complete delivery.