

CIS 4951: Capstone 2

Dr. Masoud Sadjadi, Florida International University

Stress and Anxiety Management Application

Team members:

Sidney Bobadilla, Santiago Boscan, Adolfo Cabrera, Nicholas Juman, David Vasquez

Product Owner:

Liane Sangollo

December 15, 2025

1. Executive Summary

According to the academic article “Application for Population Mental Health” by J. Adler and D. Van Brunt, more than 40 million adults in the US experience a clinical anxiety disorder. This study highlights the rise in reported cases of stress and anxiety among American adults. At the same time, the study also shows that 80% of Americans own a smartphone, which creates a powerful opportunity. With mobile devices being such an integral part of Americans’ daily life, a new opportunity arises for these technologies to provide an accessible platform for mental health support. As such, the goal of this project is to address the need of an interactive, maintainable, and cross-platform mobile application to help users manage stress and anxiety.

For this project, our team worked on the HowRu.Life project owned by Liane Sangollo. This project originally started as an Android application developed by a previous student team, which required changes to the codebase to improve maintainability, usability, and scalability. Our team addressed this need by refactoring the Android application using Jetpack Compose, enabling reusable and responsive user interface architecture. In parallel, the project expanded the platform’s reach by developing a functional iOS prototype that mirrors the core functionalities of the Android application. The iOS version was built in Flutter and a local database was implemented using SQFlite. Across both platforms, the main goal was to deliver interactive, visually and audibly engaging activities that support users in understanding how to manage stress and anxiety effectively.

To ensure that the project met the quality criteria set by the product owner, our team performed unit tests for the Android application and functional testing using emulators for both versions of the application. The application was reviewed and presented to the product owner for feedback and approval. In summary, the final version of the project provides a maintainable, scalable, and cross-platform application designed to support users in their mental health management journey through guided self-reflection questions, calming breathing exercises, and mood tracker tools.

2. Problem & Requirements

The original HowRu.life Android application, developed by a previous student team, lacked a maintainable interface architecture, which made it difficult to easily extend support for new features and cross-platform compatibility. As a result, the application’s scalability was limited. Additionally, there was not a functional iOS version, which limited the application’s reach and accessibility for users seeking stress and anxiety management tools in iOS devices. As such, the project was driven by two needs:

- 1) Modernize the Android application by implementing Jetpack Compose and restructuring the ViewModel to improve maintainability and development efficiency.
- 2) Develop a functional iOS prototype that delivers the core functionalities of the Android application.

To address the identified problems, the project needed to satisfy the following functional and technical requirements:

2.1. Functional Requirements

- Provide interactive activities that support users in their mental health journey.
- Deliver self-reflection questions that promote emotional awareness
- Include guided breathing exercises with customizable calming sounds.
- Implement mood tracker to help users analyze their emotional patterns.
- Offer visually and audibly engaging activities.
- Apply the color palette selected by the product owner.
- Ensure consistent functionalities between the Android and iOS version of the application.

2.2. Technical Requirements

- Implement a reusable user interface system using Jetpack Compose on Android.
- Develop the iOS prototype in Flutter, with local data storage to simulate user's data storage using SQFlite.
- Test the application using an emulator for both versions and unit testing for the Android version.

2.3. Deliverable Requirements

- A refactored and functioning Android application that incorporates Jetpack Compose architecture and the new color palette selected by the product owner.
- A working iOS prototype implementing the functionalities of the Android version.

3. System Overview & Architecture

The Android implementation modernizes the original build of the project by introducing Jetpack Compose and refactoring the ViewModel layer for improved reusability and maintainability. The iOS prototype, developed from scratch using Flutter, simulates the structure

and functionalities of the Android version while employing Flutter's widget based user interface and SQLite for the local database. In both platforms, the system is structured to be scalable and consistent in user experience.

3.1. Android Architecture Overview

The Android version adopts a hybrid implementation of Model View ViewModel (MVVM) architecture with Jetpack Compose. This hybrid approach allows the application to remain compatible with existing XML based layouts while transitioning to the complete implementation of Jetpack Compose. The elements used in this architecture are the following ones:

- **XML Layouts:** used for screens' designs and user interface components.
- **Jetpack Compose:** used to update the ViewModels and create reusable user interface components.
- **ViewModels:** used to manage states and data.
- **Firebase:** Cloud based backend used for authentication and user data.

A sample screen that was modified for the new hybrid implementation is the HomeActivity screen, which serves as the entry point for users after successfully logging in the system. In this screen the following components can be observed:

Component	Technology	Purpose
DrawerLayout	XML	Used for the navigation menu.
Toolbar	Material	Host the navigation toggle.
NavigationView	XML	Shows the menu items.
ComposeView	Jetpack Compose	Renders the new user interface components.
ViewModel	AndroidX	Handles user state and data.

One reusable component implemented in the HomeActivity screen through ComposeView is the MotivationalCard(username) component which enables the following:

- Smooth user interface update.
- Declarative user interface structure.
- Separation between the ComposeView and ViewModel.

Navigation in the Android application consists of the following elements:

- **Navigation Drawer:** includes the setting, dashboard, membership, about, exercise and logout screen.
- **Navigation Bar:** allows users to navigate to other screens from one place. This element consists of a reusable button at the top left corner of the screen.
- **Activity Based Navigation:** uses helper methods to switch to other activities screens.

The data layer in the Android application is handled with Firebase and consists of the following elements:

- **Authentication:** stores user credentials and handles the functionality of the login, logout, and session control. The storage of data is done through a cloud system and real time syncing is performed when needed.

3.2. iOS Architecture Overview

The iOS application uses Flutter's widget based architecture paired with a ViewModel approach. The elements that were used are the following ones:

- **Reusable Components:** user interface components, such as, title cards, buttons, calendar, and navigation bar.
- **Screen Specific Logic and Layout:** screens were designed with specific needs in mind, prioritizing isolation of components as needed.
- **ViewModels:** used for state management and data manipulation.

- **SQLite:** local database was designed using SQLite.
- **Navigation:** implemented via named routes.

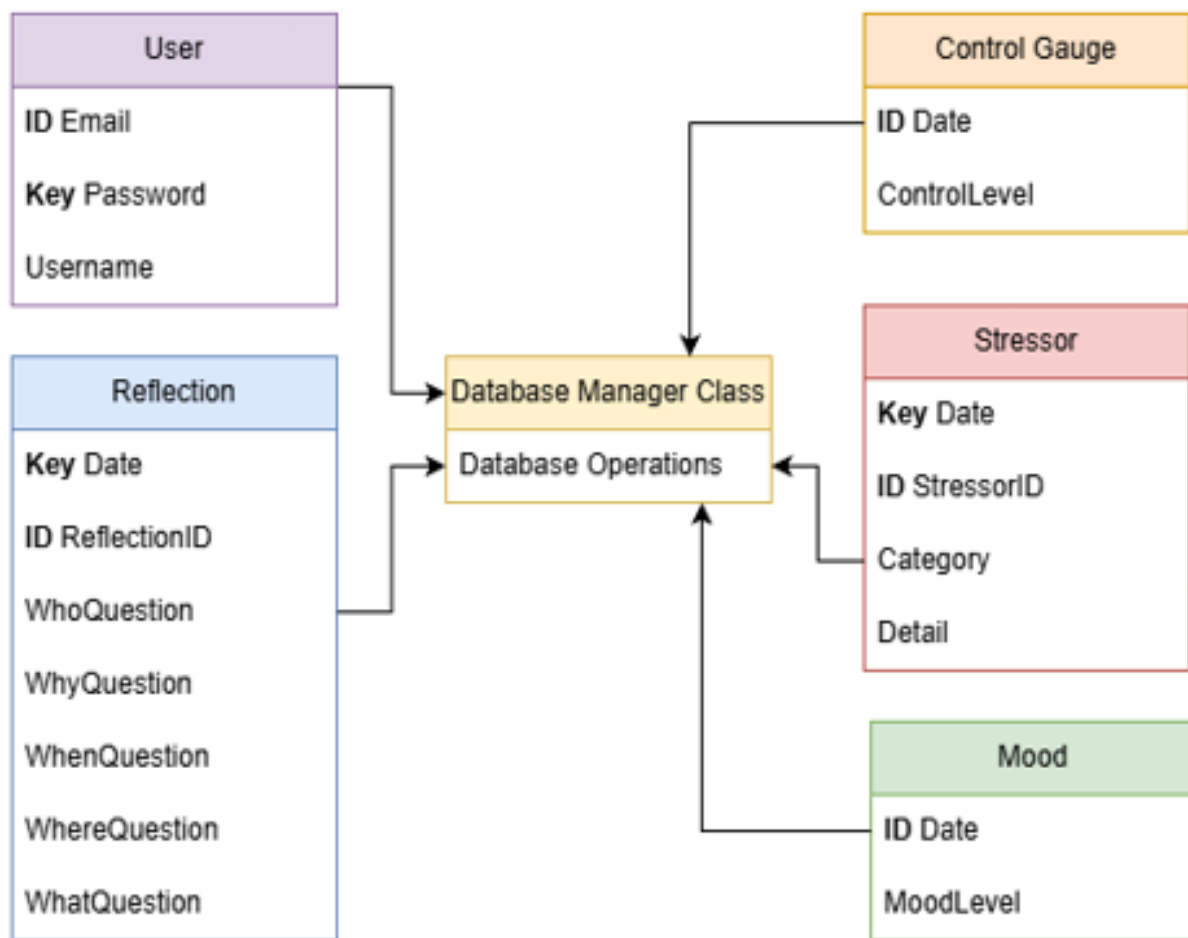
A sample of core screens developed in the iOS application are the following ones:

- **HomeScreen:** displays logo, welcome message and main menu to select exercises. Uses a reusable MainScaffold element for a navigation bar. The main menu contains reusable ActionButton components to switch between screens.
- **AboutScreen:** shows description of HowRu.life using a reusable AboutTextBox component to display text. This screen includes a fully scrollable layout for responsiveness.
- **Exercise Screen:** template screen that uses named routes for navigation and basic validation for proper screen switching. All exercises are constructed from this screen.
- **Calendar Screen:** reusable screen that works alongside exercises to implement data storage for users to log activity results on a daily basis.

The Flutter application uses an embedded SQLite database managed through a component called DatabaseHelper which contains the following elements:

Table	Purpose
Reflections	Stores self reflection questionnaire entries
Moods	Stores daily moods for tracking and chart generation
Control Gauge	Stores stress control level value
Stressor	Stores categories and details of stressors
User	Stores user credentials and preferred name

The following figure provides a graphical overview of the database structured used in the iOS prototype:



3.3. iOS Application File Structure

The file structure for the iOS application follows a modular organization that reinforces reusability and clarity to support future development for the prototype. Each folder in the file structure contains exclusively components that correspond to their respective category. All files within these folders were developed following the hybrid approach.

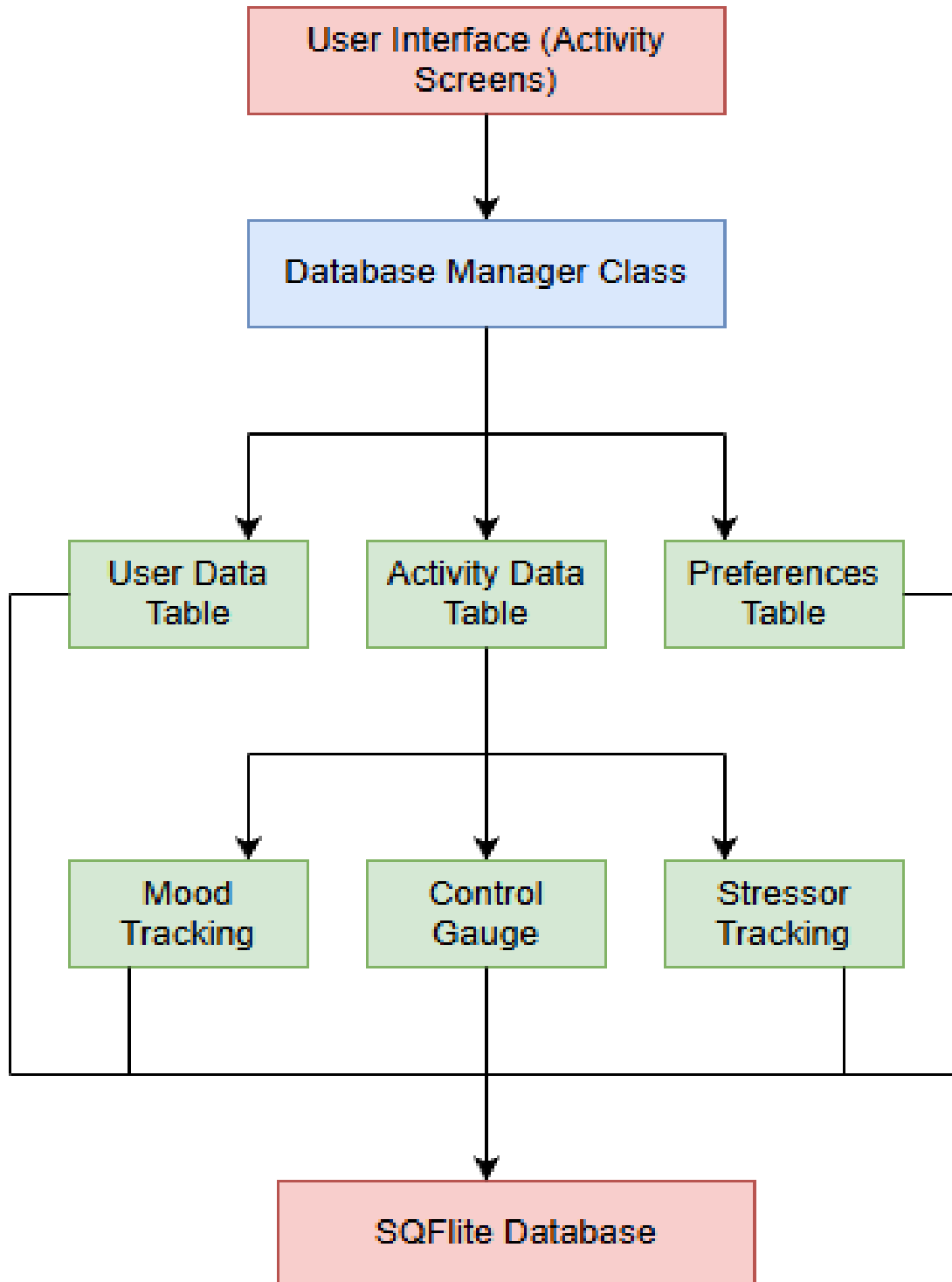
```

lib/
| — components/    # Reusable Components
| — database/      # Database Component and Operations
| — screens/       # Application Screens (Home Screen and Exercise Screens)
| — viewmodel/     # Logic for the Application Screens
| — main.dart      # Entry Point for the Application
  
```

3.4. High Level System Flow

- 1) The user opens the app and lands on the Home Screen.
- 2) The ViewModel loads user profile and activity data.
- 3) The user selects one of the following options:
 - a) Breathing Exercise
 - b) Self Reflection Questions
 - c) Mood Tracker
 - d) Dashboard
- 4) The user provides input and it is stored in the database.
- 5) The data is used to generate graphs and summary cards.
- 6) The user logs out and ends the session.

The following figure provides a graphical overview of the high level system flow used in both versions of the application:



4. Discipline-Specific Depth

The development of HowRu.life mobile application required the integration of multiple discipline specific competencies across Computer Science, Software Engineering, Mobile Development, UI/UX Design, and Data Management. Each component of the project required a technical approach that aligned with the expectations from the product owner. For each field our team performed industry best practices for efficiency in the development process.

4.1. Mobile Development

Updating the Android application required understanding of modern Android architectural patterns and the Jetpack Compose Framework. The transition from only using traditional XML user interface to a hybrid approach with Jetpack Compose enabled a declarative programming model, improved user interface state management, and reusable components. Building the iOS prototype from scratch required applying cross-platform engineering principles using Flutter. The team designed the core functionalities to closely align with the Android experience.

4.2. UI/UX Design Consideration

A strong focus was placed on delivering activities that are visually calming and intuitive for users managing stress and anxiety. The color palette was selected by the product owner to reinforce emotional comfort while sticking to the brand identity. The color palette is the following one:



4.3. Data Management

The project involved implementing a database system capable of tracking user interactions and emotional patterns over time. As such, a database was designed to support storage of moods, awareness questions, and summary cards. Alongside exercise data, the database also supports user credentials and preferred information.

4.4. Quality Assurance & Testing

Emulators were used to perform testing for both the Android and iOS application to verify responsiveness and functional correctness. Iterative reviews with the product owner were performed throughout the semester to ensure the implemented features aligned with the project's requirements and vision.

5. Implementation Notes

The Android application was refactored using Jetpack Compose, transitioning from only using XML layout to a hybrid approach. ViewModels were rewritten to accommodate for the new framework and reusable components were developed to implement text cards, buttons, and navigation bars.

The iOS application was developed from the ground up using Flutter, allowing for rapid user interface prototyping and consistent structure across screens. The following features from the Android application were developed: awareness questions, guided breathing exercises, and mood tracking. The prototype uses a local database to simulate user authentication systems and user data storage for activities. User interface elements were built using Flutter widgets, ensuring responsive layouts compatible across multiple devices with different screen sizes.

5.1. User Stories

Over the course of six sprints, our team worked through a structured backlog of user stories that outline essential tasks needed to advance the project's goal. These tasks included:

Project Setup:

- Configured the development environment for Android.
- Set up the iOS/Flutter development environment.
- Documented the complete environment setup process for both Android and iOS platforms.

Architecture & Documentation:

- Created the project architecture documentation detailing hybrid approach on Android.

- Created the database documentation describing table schemas, entity relationship, and SFQLite operations.
- Maintained a structured and continuous change log documentation to write down changes across all sprints.
- Updated the README file to reflect project structure, installation, and development workflow.

User Interface Component Development:

- Designed and documented reusable user interface components for Android.
- Designed and documented reusable user interface components in Flutter.

Jetpack Compose Migration:

- Refactored Android ViewModels to integrate Jetpack Compose framework.

iOS Flutter Application Development:

- Built Home Screen user interface with reusable components and responsive layout.
- Implemented the Login and Signup Screens, including input validation and navigation logic.
- Implemented core navigation flow using named routed.
- Developed activity screens for reflection questions, mood tracker, and breathing routine.

Database Implementation

- Designed and implemented SQLite databases tables for users, mood, reflection, stressor, and control gauge.
- Added a component for database operations and local storage.

6. Testing & Evaluation

6.1. Android Testing

- **Unit Testing:** ViewModels and screen logic were validated using unit tests developed by the previous student team to ensure functional states, proper data handling, and consistent behavior across the application.
- **Functional testing:** The application was tested on multiple emulator configurations to verify screen responsiveness, navigation flow, and valid input correctness.

6.2. iOS Prototype Testing

- **Functional testing:** The application was tested on multiple emulator configurations to verify consistent layout, functional navigation, and proper handling of data.

6.3. Stakeholder Evaluation

- **Product Owner Review:** Both versions of the application were demonstrated to the product owner for approval. The feedback focused on visual consistency, usability, color palette implementation, and alignment with the vision of HowRu.life.
- **Iterative Improvements:** Adjustments were made based on testing findings and product owner input to refine user experience across both platforms.

7. Security, Privacy, Accessibility & Compliance

Both versions of the application do not collect, store, or transmit user's personal information. All data stored locally in the iOS application remains on the user's device. To meet the privacy transparency requirements of the Google Play Store, clear disclosures were added to the application Frequently Asked Question Screen stating that no user data is collected and no data is shared with third parties. Privacy expectations and app behavior were reviewed and confirmed with the product owner to ensure full alignment with Google Play Store guidelines and platform expectations. To address accessibility the team used a legible font for text and audio cues were added to exercises to support users who benefit from auditory interactions.

8. Deployment & Operations

8.1. Android Deployment

- **Google Play Store Release:** The Android version has been successfully deployed to the Google Play Store with the changes made by our team. Prior to the deployment the application underwent reviews using Android Studio, the application was sent for review to meet the verification for Google Play policies, and screenshots were added in Play Console to display the current state of the application. Ongoing operations involve monitoring feedback and maintaining the codebase for newer Android versions.

8.2. iOS Prototype Deployment

- **Development Build:** The iOS is currently only a functional prototype developed using Flutter. It is not deployed to the App Store and must be run through development tools. The prototype can be executed through Android Studio with Flutter support or any Flutter compatible IDE.
- **Execution Process:** The following steps showcase how to run the iOS prototype in Android Studio:
 - 1) Open the Flutter project in Android Studio.
 - 2) Select a device to act as an emulator via the device manager.
 - 3) Run the project using Flutter's build and debug tools.

9. Limitations & Future Work

The main limitation for the iOS version is that the application is currently a functional prototype and has not been optimized to comply with the privacy policies of the App Store. Additionally, the prototype lacks unit testing for error handling. And because data is stored locally, data does not sync across devices and user authentication remains minimal.

The iOS prototype can be expanded and prepared for the App Store deployment by implementing additional user interface refinements so that the application has the same layout as the Android application. Ensure that the iOS application complies with the App Store privacy policies. A production ready database is implemented to handle user authentication and data manipulation. And a device with MacOS is used to create a deployable build of the application for the App Store.

10. References

Android Developers, “Get started with Compose — Android Developers”

J. Adler and D. Van Brunt, “It is time to realize the promise of the digital mental health transformation: Application for population mental health,” J. Med. Internet Res., vol. 27, no. 6.

Appendix

Appendix A Installation Guide

A.1. Android Application

The Android version of the HowRu.life application is publicly available through the Google Play Store. The following steps showcase the process of how to install the application:

A.1.1. Installations Steps

1. Open the [Google Play Store](#) on an Android Device.
2. In the search bar, type “HowRu.life” and locate the application published by HOWRU.LIFE
3. Select application from the search results.
4. Click the install button to download.

HOWRU.LIFE

HOWRU.LIFE

1+
Downloads

Everyone

Install

Share

Add to wishlist

You don't have any devices



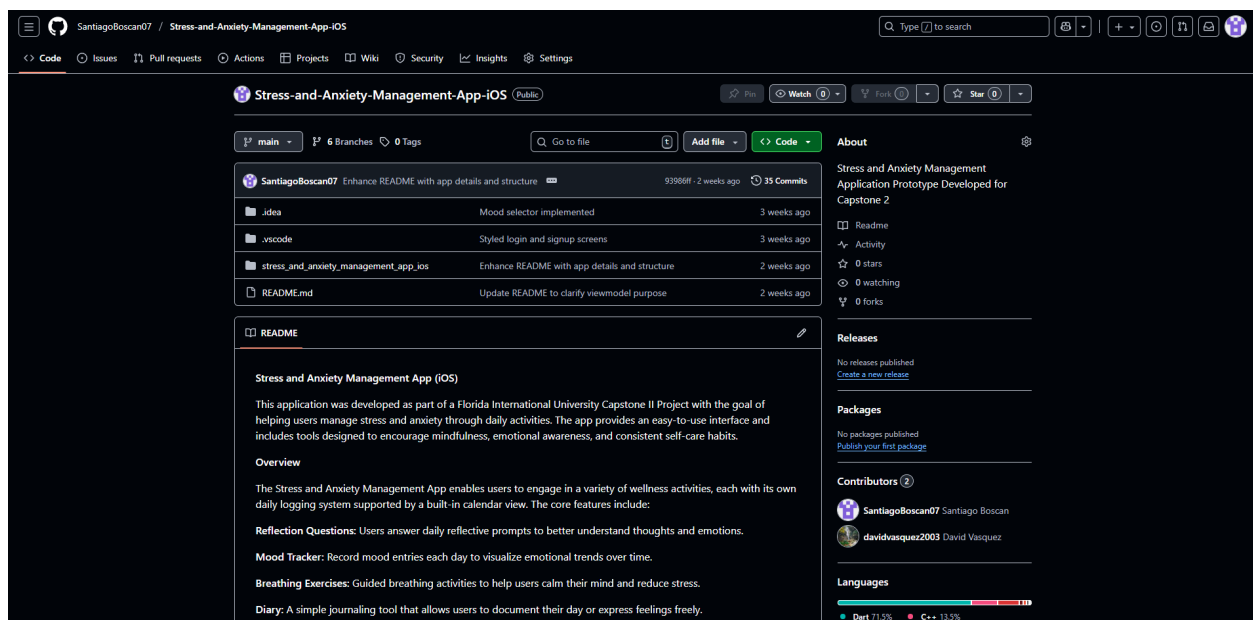
5. Once the download is completed, locate the application in the device and tap the icon.
6. Follow the sign up instructions to begin using the application.

A.2. iOS Prototype

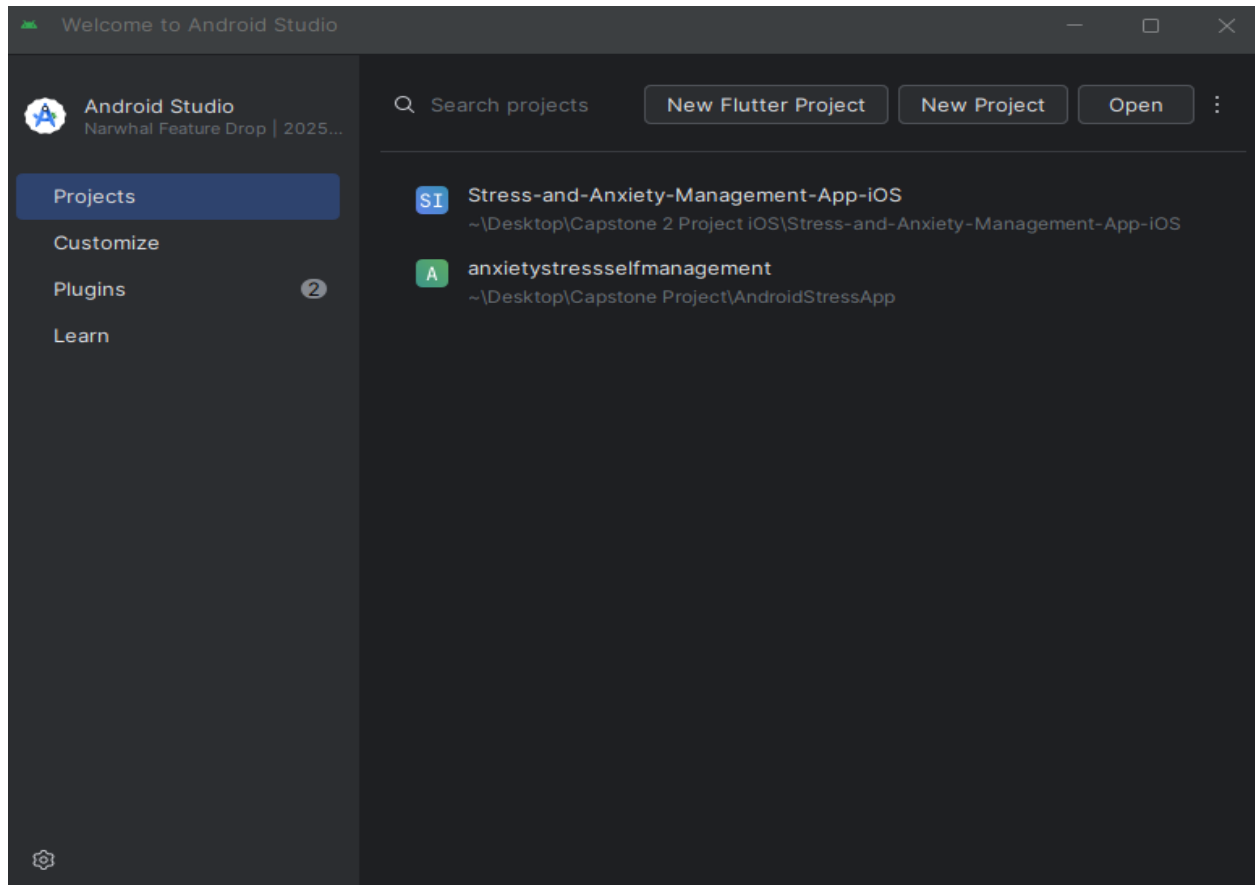
Since the iOS version is a development prototype, it is not available on the Apple App Store. Instead it must be downloaded from GitHub and executed in a Flutter compatible IDE, such as Android Studio. The following steps showcase the process of installing the application in Android Studio:

A.2.1. Installation Steps

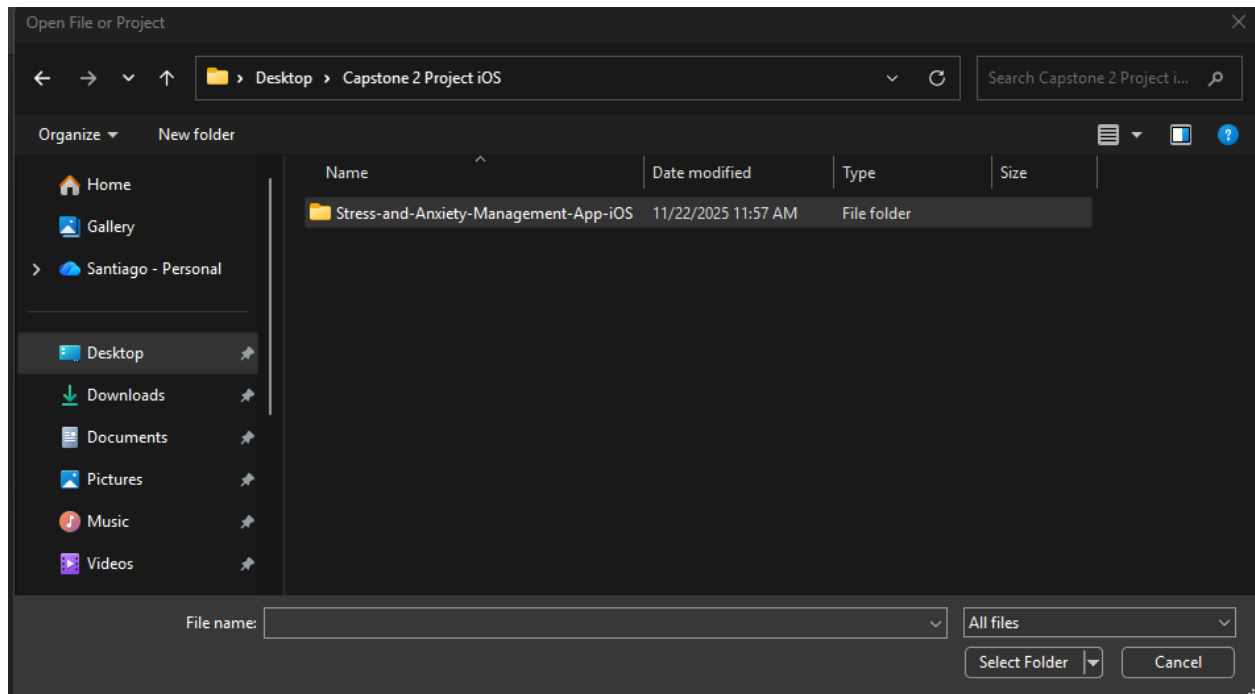
1. Navigate to the [project repository](#) on GitHub.
2. Click the code button, and then click the download zip button.



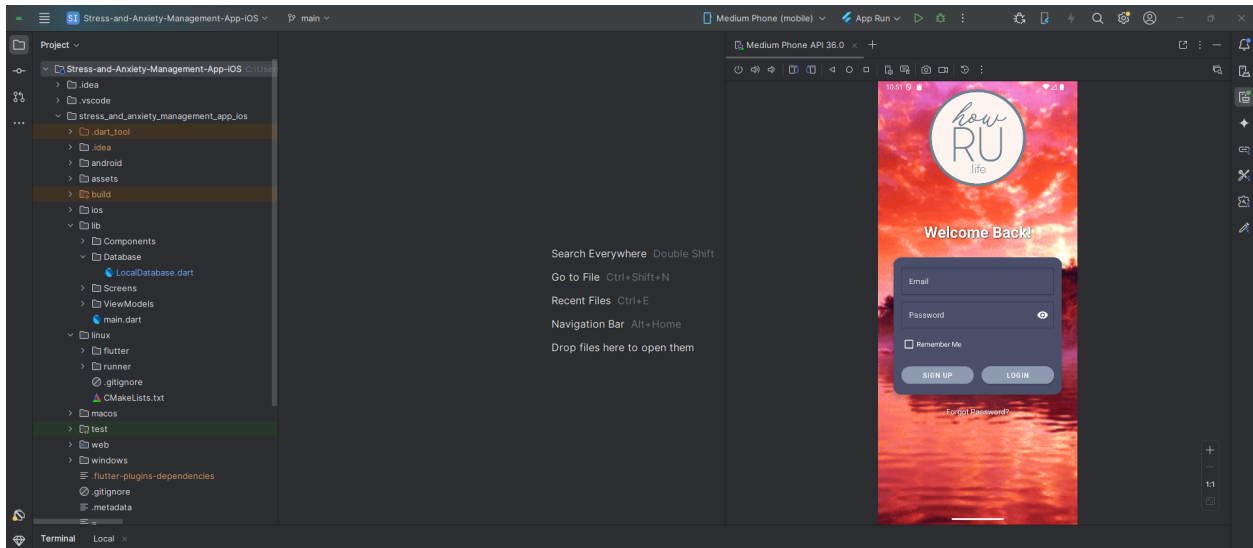
3. Once the file has been downloaded, locate where the file was saved and extract the zip.
4. Launch Android Studio and press open.



5. Locate the folder where the zip was extracted and press the select folder button.

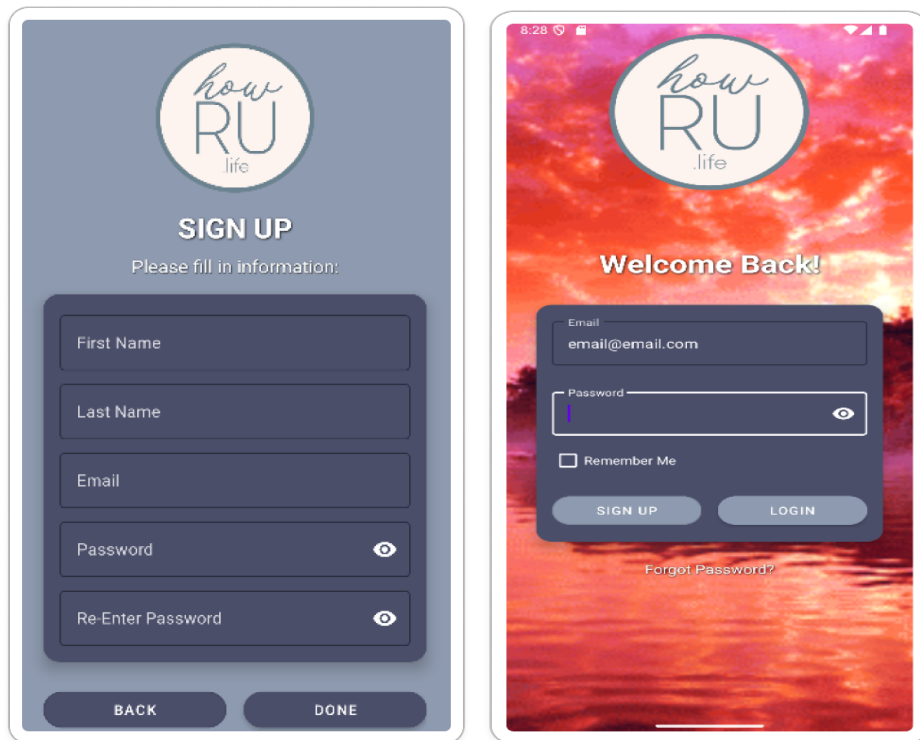


6. To run the application select a mobile device to emulate in the device manager and press the run button to start the program.



Appendix B User Manual

B.1. Login and Signup Screen

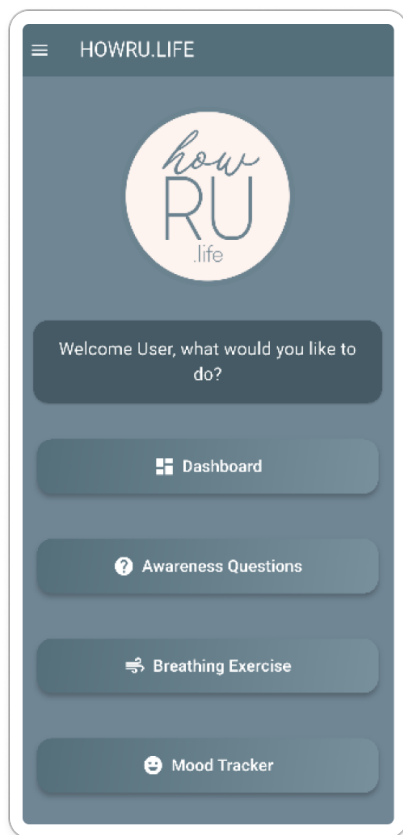


The user must create an account in the sign up screen by providing their full name, email, and password. Once valid information has been provided, the user will be able to log in in the application from the log in screen by providing the email and password that they set up during registration. In case that the user forgets their password, they can press the forget password button to begin the password reset process.

B.1.1. Password Reset Process

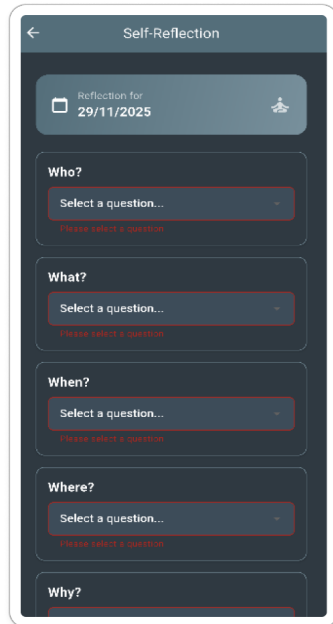
If users provide a valid email registered in the system, the user will receive an email message to create a new password for their account.

B.2. Dashboard Screen



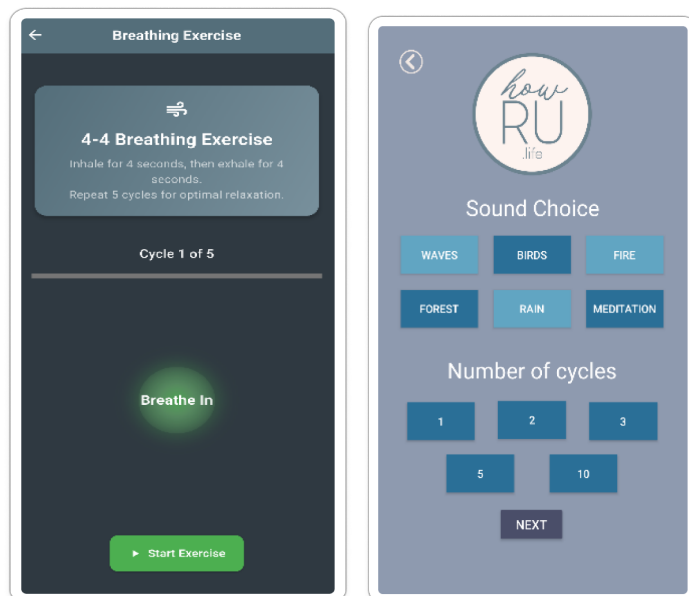
After successfully logging in the system, the user will be presented by the dashboard screen which offers access to the main features of the application: self reflection questionnaires, breathing exercises, mood tracker, and dashboard summary.

B.3. Self Reflection Screen



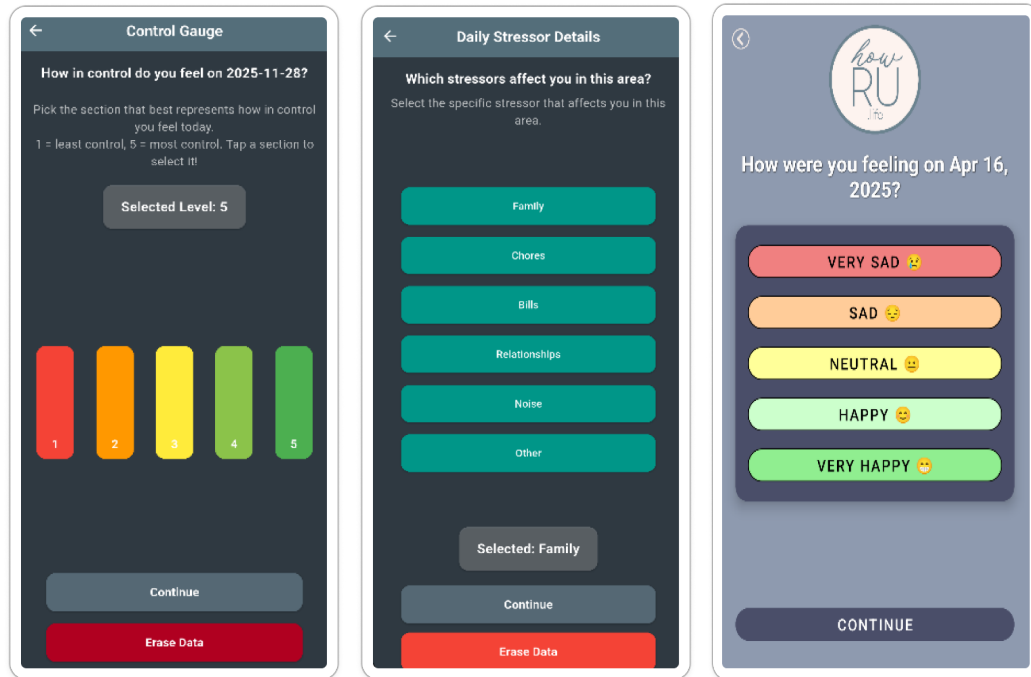
To access the self reflection screen the user must tap the self reflection button on the dashboard screen. Before the self reflection starts, the user will be given the option to select the current date or a different date to record the responses from the activity. After the date is selected, multiple prompts will be displayed to the user. Once all questions have been read and responded, the user can save the responses by pressing the save button at the bottom of the page.

B.4. Breathing Exercise Screen



To access the breathing exercise screen the user must tap the breathing button on the dashboard screen. Before the breathing exercise begins, the user will be given the option to select a sound and the number of cycles that will be performed. After the sound and cycles are selected, the screen will transition to another one where sounds and imagery will support the user as they perform the exercise.

B.5. Mood Tracker



To access the mood tracker, the user must select the mood tracker button in the dashboard screen. Before the mood tracker activity starts, the user will be given the option to select the current date or a different date to record the responses from the activity. Once the date is selected, multiple prompts will be displayed to the user to respond. First the user must select from one of five options when asked about their feelings. Second the user must select a level from one to five to reflect on how in control they feel. Lastly the user will be asked what their stressor of the day is.

Appendix C Admin/Operations Guide

C.1. Codebase Management

The Android version is maintained in a GitHub repository and the application is deployed to the Google Play Store. The iOS version is maintained in a separate GitHub repository and runs through Android Studio or another Flutter compatible IDE. The admin responsibilities includes:

- Keep both repositories updated with the latest stable version of Flutter and Android dependencies.
- Conduct periodic code cleanups to remove unused assets, deprecated libraries, and warnings.

C.2. Troubleshooting

For the iOS application run the “flutter clean” command in the terminal to resolve most environment related issues with the application. Update Flutter’s dependencies using the “flutter doctor” command in the terminal.

Appendix D API Reference

The application does not use any external web APIs, functionalities were implemented using internal classes and functions that serve as the project’s core programming interface.

D.1. Android Internal API

The android implementation uses Jetpack Compose and MVVM architecture.

D.2. iOS Prototype Internal API

The iOS implementation uses Flutter and SQLite to manage local SQLite database creation and queries.

Appendix E Poster, Slides, and Video Link

E.1. Poster

The poster can be accessed from the following link: [Stress and Anxiety Management Application Poster Fall 2025.](#)

E.2. Presentation Slides

The presentation slides can be accessed from the following link: [Stress and Anxiety Management Application Presentation Slides Fall 2025.](#)

E.3. Introduction Video and Demo Video

E.3.1. Introduction Video

The introduction video can be accessed from the following link: [Stress and Anxiety Management Application Introduction Video Fall 2025.](#)

E.3.2. Demo Video

The demo video can be accessed from the following link: [Stress and Anxiety Management Application Demo Video Fall 2025.](#)

Appendix F Scrum Evidence Index

F.1. Daily Scrum Meetings

Daily scrum meetings can be accessed from the following link: [Daily Scrum Meetings Document Fall 2025](#).

F.2. Sprint Backlog Grooming Meetings

Sprint backlog grooming meetings can be accessed from the following link: [Sprint Backlog Grooming Meetings Document Fall 2025](#).

F.3. Sprint Retrospective Meetings

Sprint retrospective meetings can be accessed from the following link: [Sprint Retrospective Meetings Document Fall 2025](#).

F.4. Sprint Review Meetings

Sprint review meetings can be accessed from the following link: [Sprint Review Meetings Document Fall 2025](#).

F.5. Sprint Planning Meetings

Sprint planning meetings can be accessed from the following link: [Sprint Planning Meetings Document Fall 2025](#).