UNIVERSITI MALAYSIA TERENGGANU
FACULTY OF OCEAN ENGINEERING TECHNOLOGY & INFORMATICS

CSM3114
FRAMEWORK-BASED MOBILE APPLICATION DEVELOPMENT

Project Report 2

NovaPlan: Project Monitoring & Tracking App

Prepared by:

Name	Matric No
Nur Siti Dahlia Binti Ab Ghani	S62584

Prepared for:
Dr. Mohamad Nor Bin Hassan

Bachelor of Computer Science (Mobile Computing) with Hons.
SEMESTER I 2023/2024

Table of Contents

1.0 Executive Summary

As the business environment became more dynamic, our creative mobile application became a vital tool for organising and enhancing organisations project management. Our innovative mobile application, NovaPlan, has evolved in response to the dynamic business climate and is now an important tool for improving and organising project management within organisations. Progress tracking, real-time activity status updates, project and activity registration, and secure user authentication are a few features that NovaPlan offers. The app also functions as a pop-up reminder system on the home screen by a Calendar Hub that is smoothly incorporated into the application. NovaPlan is prepared to completely transform project tracking and monitoring by offering a complete solution that fits the changing requirements of contemporary companies. NovaPlan aims to improve organisational efficiency and adaptability in project management through its user-friendly interface and progressive capabilities, guaranteeing smooth cooperation and well-informed decision-making in today's hectic business climate.

Driven by the increasing importance of mobile applications in modern business operations, our solutions address important challenges that companies face before they are introduced. Errors and delays are frequently caused by inefficiencies in traditional project management techniques. The absence of a real-time update system results in communication gaps, while an ineffective reminder system contributes to missed deadlines. With its dynamic, approachable, and adaptable structures, mobile apps are a transformative tool for project management. Customer interaction is enhanced with an intuitive interface, which maximises flexibility. With its ability to personalise, simplify, and increase accessibility to corporate services, the app has the potential to drastically alter the business environment. To put it briefly, it is a crucial element for companies looking to adjust to changing technological trends while maintaining the effectiveness and flexibility of their project management initiatives.

2.0 Use Case

The project monitoring and tracking mobile app use case is illustrated in the **Diagram 2.1**, offers a comprehensive set of features designed to streamline project management The relationship between these functions is depicted in this use case diagram:

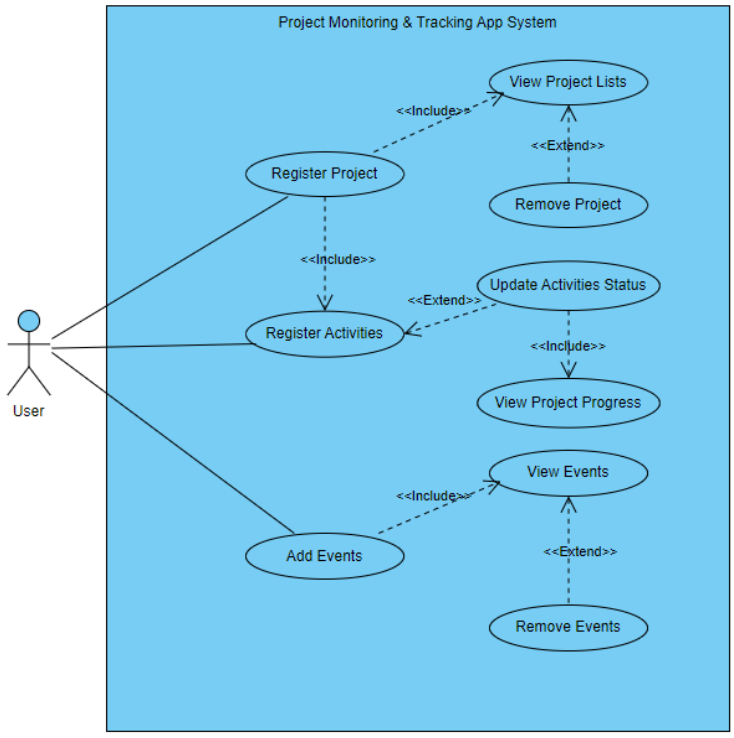


Diagram 2.1: Use Case Diagram

The use case diagram illustrates that the application system facilitates users in organizing their work, tracking project progress, and managing events associated with their activities. There are three primary use cases related to project and activity management, including the registration of projects, activities, and the addition of events. **Diagram 2.1** makes it clear that activities and projects are related, and that each project may have several related activities. Add Events, on the other hand, exist independently but can be effectively managed and removed, enhancing the overall organization and cleanliness of the system. Users are able to change the status of individual activities since activities are linked to status updates. The status of activities plays a crucial role in determining the overall progress of the project.

3.0 Common Structure of Tree Widgets Used in Designing and Developing

The widget tree is a hierarchical structure where each widget can have child widgets, forming a tree of components that define the user interface (UI). The MaterialApp widget sets up the overall structure of the app. Here are some widget trees for each main screen :

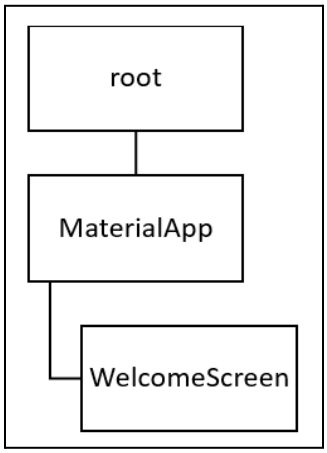


Figure 3.1: Main.dart

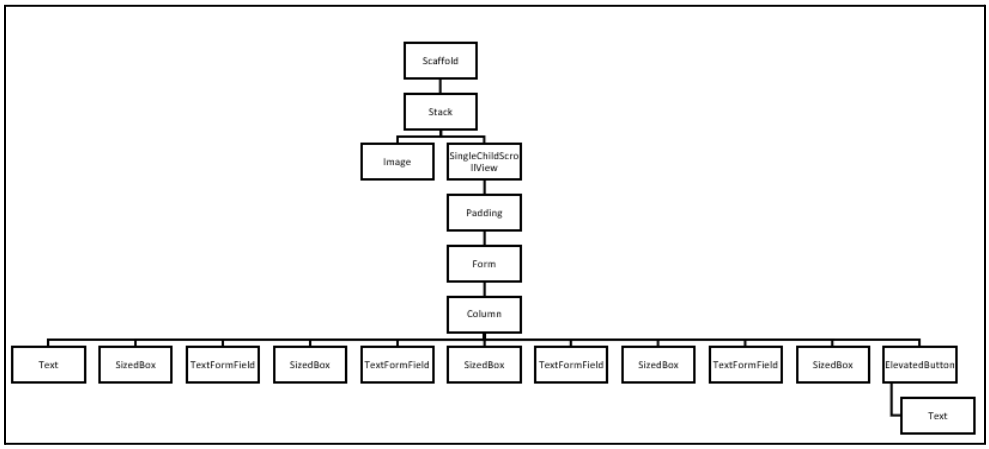


Figure 3.2: Sign Up Screen

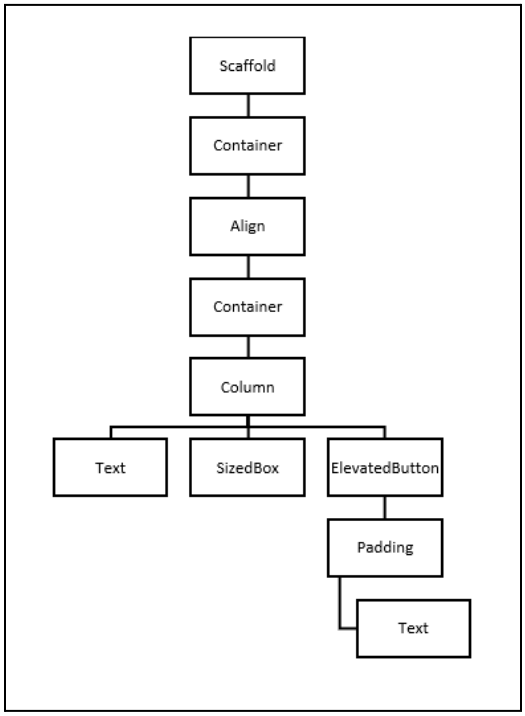


Figure 3.3: Welcome Screen

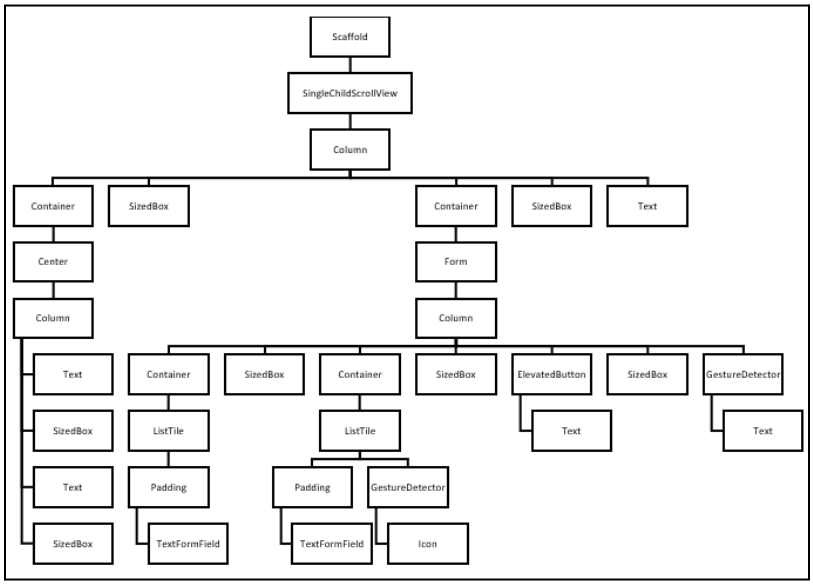


Figure 3.4: Login Screen

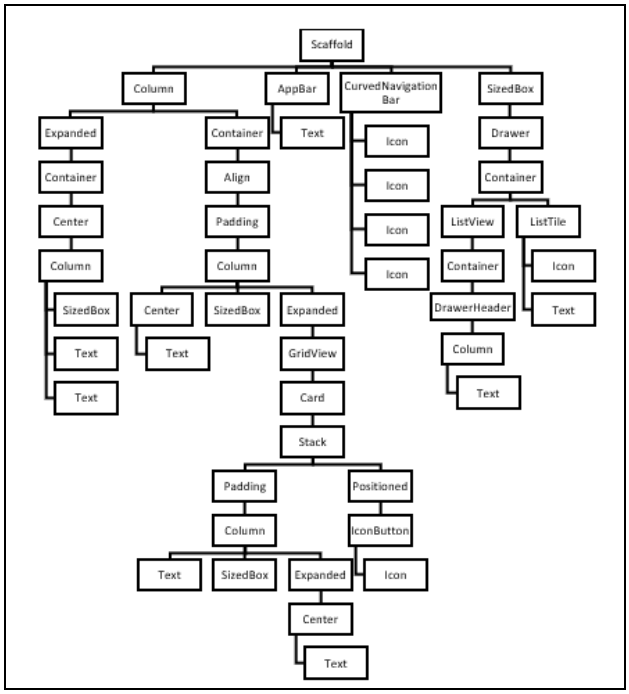


Figure 3.5: Home Screen

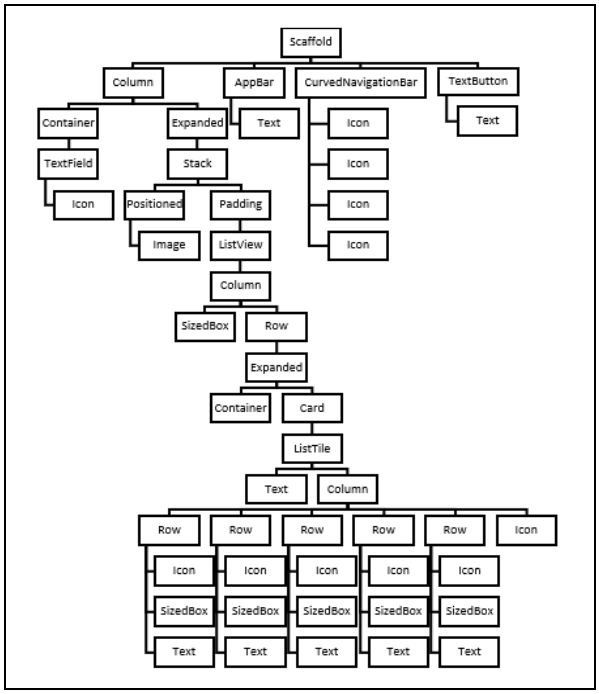


Figure 3.6: Project Screen

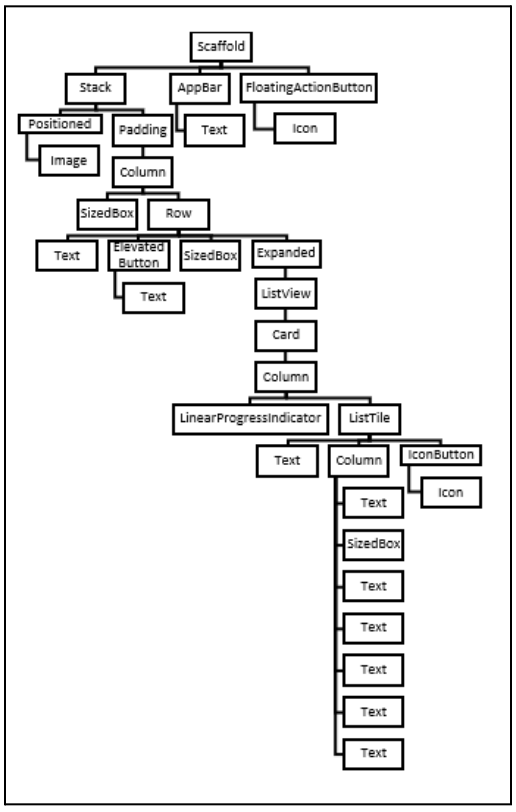


Figure 3.7: Activity Screen

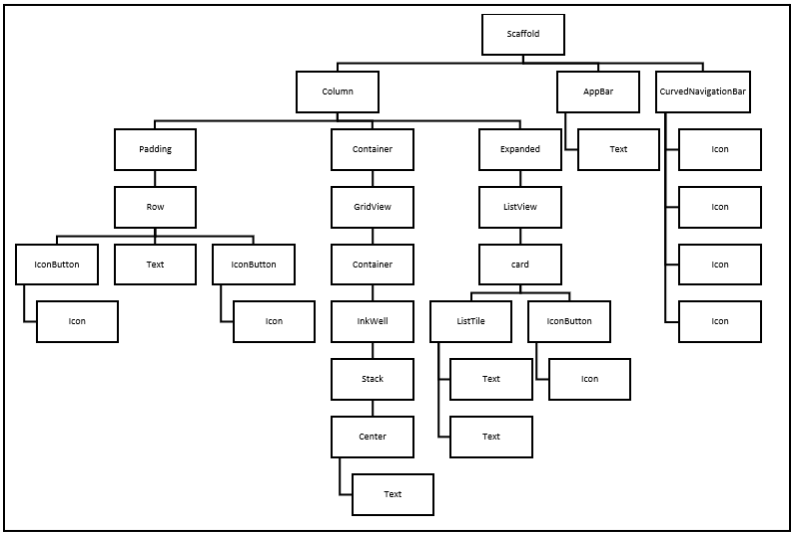


Figure 3.8: Calendar Screen

4.0 Flutter Widget and Features Adopted in Application

There are several widgets and features that have been adopted in this application. One of them is the CurvedNavigationBar widget, which serves as the base for the curved navigation bar. This widget accepts an array of items, often icons, representing individual tabs or buttons. To use this, the dependencies need to be set in pubspec.yaml which is curved_navigation_bar: ^1.0.3. Next is CustomPaint, a widget that gives a canvas on which to draw during the paint phase. This widget allows us to create a custom graphic by providing the canvas (*CustomPaint Class - Widgets Library - Dart API*, n.d.). In this application, this widget has been used to implement the StatusChart and Calendar. We are unable to use setState or markNeedsLayout during the callback because custom paint calls its painters during paint, where the layout for this frame has already done.

Another widget adopted in this application is Wrap. Wrap is a widget that shows its children in numerous horizontal or vertical runs. Each child is laid out by a Wrap, who attempts to fit them next to each other in the main axis according to the direction supplied, leaving space between them (*Wrap Class - Widgets Library - Dart API*, n.d.). This widget is particularly useful when dealing with a dynamic number of children widgets that may not fit within a single row. Next is Stack which is a widget that arranges and positions its child widgets according to the box's borders. This class is used to overlap multiple elements in a basic method, for example having some text and an image, overlaid with a gradient and a button attached to the bottom. Inside a stack, the Positioned widget is frequently used to precisely position its child. For instance, in this application it takes top, bottom, left, right, and other parameters to determine the exact placement in the Calendar screen.

AbsorbPointer is a widget that is used to prevent its subtree from receiving pointer events. This means that any interaction with the widgets within the AbsorbPointer subtree, such as taps or gestures, will be intercepted and prevented from reaching those widgets (*AbsorbPointer Class - Widgets Library - Dart API*, n.d.). In this application, the AbsorbPointer widget is used to make the child widget, _buildTextFieldWithIcon, unresponsive to user input events, specifically tap gestures in the Activity screen.

Other features adopted is AlwaysStoppedAnimation class in Flutter, it is a simple utility class that is often used to provide fixed (immutable) values for animation properties. It is especially useful in situations where animation is required, but the value of the animation remains constant over time. In this application, this class is used to specify the color of the LinearProgressIndicator within the Card widget, it ensures that color of the progress indicator is constant.

In Flutter, BoxShape is an enumeration that represents many forms for a BoxDecoration and commonly used when styling or decorating containers. In this application, BoxShape.circle is used in the BoxDecoration to create circular containers for status options. Next is the Radio widget. The Radio widget is a user interface element that simulates a radio button and lets the user choose one choice from a list of options that are mutually exclusive. Usually, each radio button in the group has a label, and only one radio button may be selected at a time. In applying this, a row of components presenting a radio button option is created using the Radio widget in the _buildStatusOption method, which uses this radio to update status for activities.

LinearProgressIndicator has been used in this application where it is used within the Card to visually represent the progress or status of an activity. This widget creates a linear progress bar and the value property is set based on the condition of activity.status == 'Complete' ? 1.0 : 1.0. This line suggests that regardless of the activity's status, the progress bar is set to 100% (1.0). In this application, Firebase also has been used for data storage and retrieval. Firebase Realtime Database is used to manage project activities and status specified on each logged in user. Firebase is a cloud-hosted database. Every connected client receives real-time data synchronization in JSON format.

5.0 Sample of Interface

The user interface (UI) of the NovaPlan Mobile App for Project Monitoring and Tracking is thoughtfully designed. Every screen in this application has been created to improve user experience and ensure a smooth and simple interaction by the user. These are the main screen in this application that have been developed:

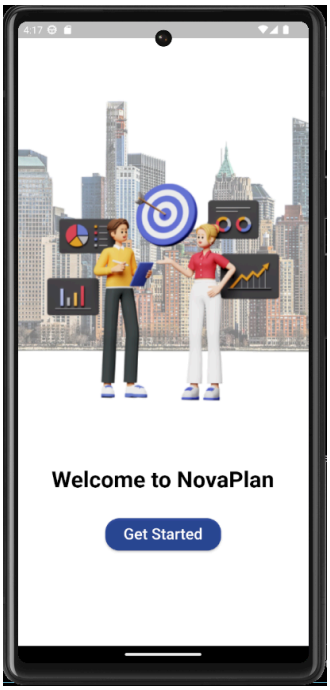


Figure 5.1: Welcome Screen

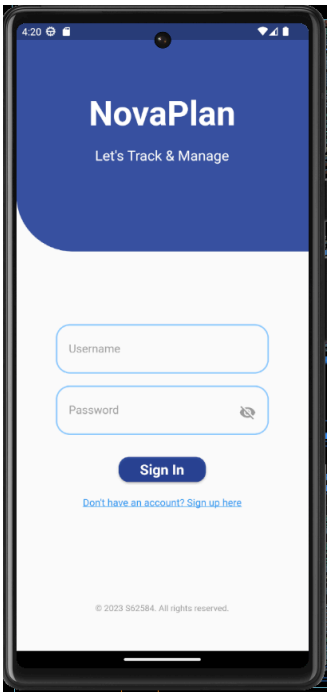


Figure 5.2: Log In Screen

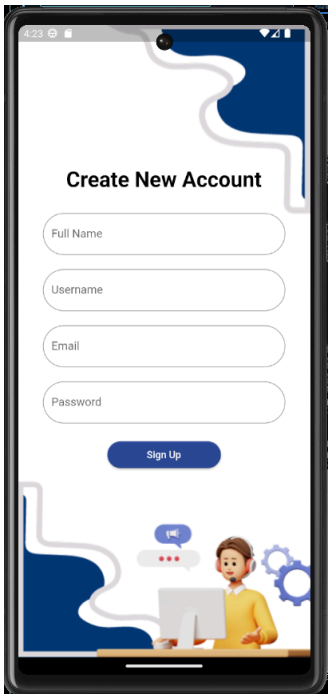
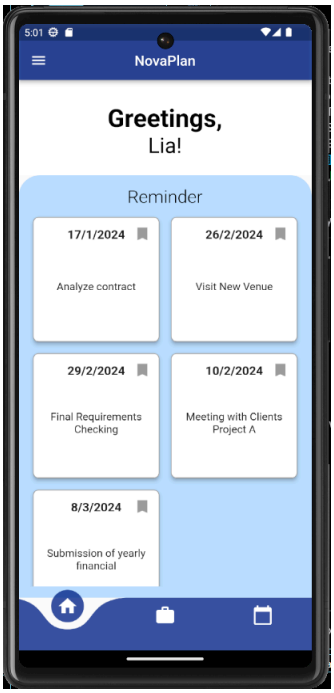
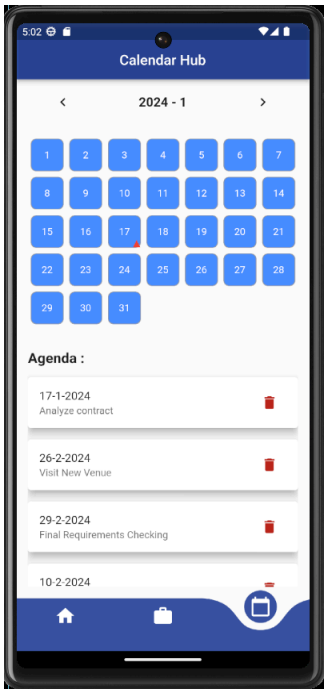


Figure 5.3: Sign Up Screen

The Welcome page begins this application and is where the journey starts. After clicking the 'Get Started' button, users start the process by logging in. Users are prompted to enter their right password and username on the Login screen. If something goes wrong, an informative error is advised by the message to correct their input before continuing. Clicking the designated link takes new users to the Signup screen. Here, in order to make an account, people must fill out a form. The data entered is automatically integrated with Firebase upon clicking the Signup button, guaranteeing safe storage. Returning to the Login screen, users can now access the application by entering the credentials established during the signup process. This robust authentication system establishes a streamlined onboarding process, allowing users to transition from the Welcome screen to actively engaging with the application's features securely and efficiently.





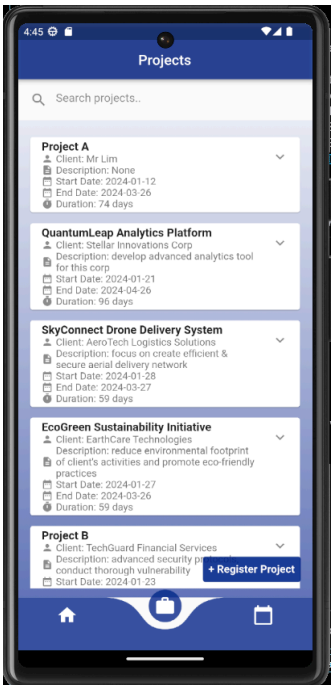


Figure 5.4: Home Screen *Figure 5.5: Calendar Screen* *Figure 5.6: Projects Screen*

The Home Screen of our application, as illustrated in **Figure 5.4** greets users by displaying their username and acts as an organizational hub. It features a card format for reminders derived from the Calendar Screen, offering a quick overview of scheduled agendas. The circular navigation bar allows seamless switching between the Project Screen and Calendar Screen. In the Calendar Screen, users can easily add, delete, and view upcoming events with the integration of Firebase Realtime Database for cross-device accessibility. Moving to the Project Screen, a bottom-right button facilitates the creation of new projects, with a search function for efficient navigation in extensive project lists. Each project on the list provides options to view activities or delete the project by pressing on the expand icon.Upon selecting "View Activities," users are seamlessly directed to the Activities Screen, allowing for detailed exploration of associated tasks. Conversely, choosing "Delete" initiates the removal of the project, synchronizing the change with Firebase Realtime Database and promptly updating the displayed project list. This comprehensive design ensures a user-friendly and efficient experience, emphasizing seamless navigation and data management.

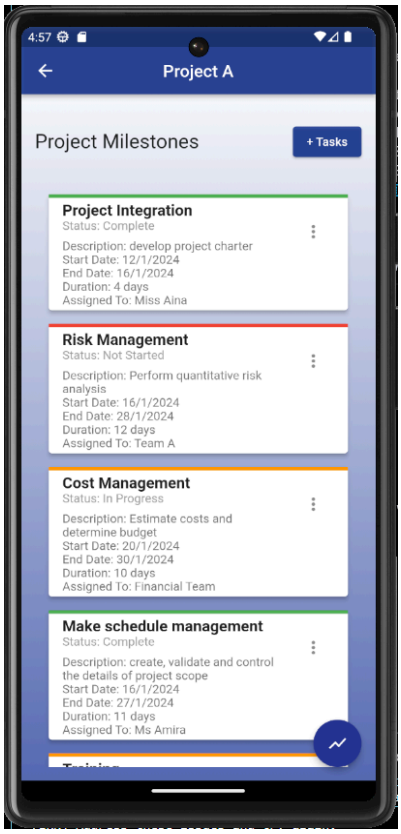


Figure 5.7: Activities Screen

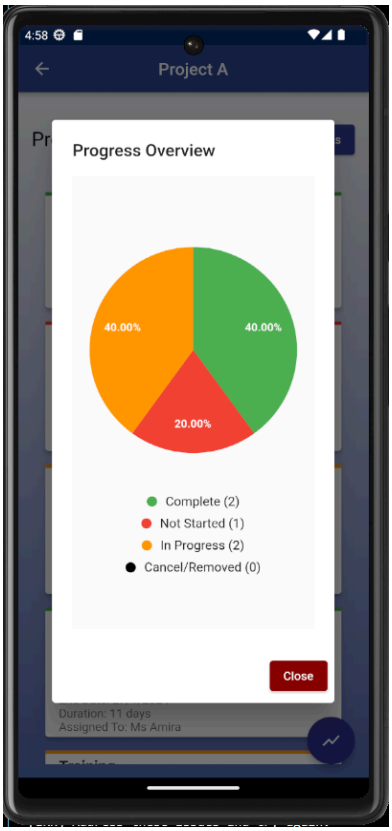


Figure 5.8: Overview Screen

Figure 5.9: Log Out Screen

In the Activities Screen, users can effortlessly manage their tasks with a color-coded status system. The addition of tasks is facilitated by an elevated button on the top right, guiding users through a detailed input process. Task details are then displayed in a list format. Task status updates are intuitive, as users click on the "more" icon to access radio buttons for quick status selection. The linear progress indicator adapts its color to reflect the chosen status, providing a visual cue for efficient task tracking.

The statistic symbol in the bottom right can be tapped by users to get insight into the overall progress of the project. Upon performing this step, a visually appealing pie chart, as in **Figure 5.8**, illustrating the task percentages for every status is shown. A detailed breakdown of the task count related to each status is available to users beneath the chart. The dual presentation ensures that users can quickly grasp the overall project status and improves user comprehension. The user's engagement with the application comes to an end when a drawer containing a logout button on the Home Screen implies a seamless return to the login page.

6.0 Conclusion

In conclusion, our revolutionary mobile application has emerged as an essential tool for current businesses, tackling critical project management issues. With its user-friendly project creation, well-organized lists, and effortless navigation, the main features in this application such as Project Screen simplifies the management of projects. Real-time data synchronization via the Firebase Realtime Database is also provided. With its dynamic updating mechanism, color-coded status system, and task adding method, the Activity Screen is an excellent task management tool all at once. An extensive summary of the project progress is given by the chart. When combined, these elements transform project management and promote effectiveness and flexibility. With features like progress monitoring, real-time updates, projects with activity registration, and a user-friendly calendar, the software significantly boosts efficiency and management. It not only overcomes the inefficiencies of traditional management methods but also improves accessibility. As such, it is a transformative tool for businesses looking to maintain effective project management practices while adapting to changing business landscapes.

7.0 Reference

AbsorbPointer class - widgets library - Dart API. (n.d.). https://api.flutter.dev/flutter/widgets/AbsorbPointer-class.html

BoxShape enum - painting library - Dart API. (n.d.). https://api.flutter.dev/flutter/painting/BoxShape.html

ChatGPT. (n.d.-b). https://chat.openai.com/c/6c1e2d37-8fb4-4607-b6de-302b810b4126

CustomPaint class - widgets library - Dart API. (n.d.). https://api.flutter.dev/flutter/widgets/CustomPaint-class.html

Firebase Realtime database. (n.d.). Firebase. https://firebase.google.com/docs/database

How do I set the animation color of a LinearProgressIndicator? (n.d.). Stack Overflow. https://stackoverflow.com/questions/41068599/how-do-i-set-the-animation-color-of-a-linearprogressindicator

LinearProgressIndicator class - material library - Dart API. (n.d.). https://api.flutter.dev/flutter/material/LinearProgressIndicator-class.html

Stack class - widgets library - Dart API. (n.d.). https://api.flutter.dev/flutter/widgets/Stack-class.html

Team, M. (n.d.). *A flutter package for easy implementation of curved navigation bar*. https://morioh.com/a/b96841d7aa56/a-flutter-package-for-easy-implementation-of-curved-navigation-bar

Wrap class - widgets library - Dart API. (n.d.). https://api.flutter.dev/flutter/widgets/Wrap-class.html