UNIVERSITY MALAYSIA TERENGGANU

FACULTY OF OCEAN ENGINEERING TECHNOLOGY & INFORMATICS

CSM3114
FRAMEWORK-BASED MOBILE APP DEVELOPMENT

PROJECT REPORT

SMART PARKING SYSTEM

PREPARED BY:

ARUN MUGILAN A/L SARGUNAN S63746

PREPARED FOR:

MOHAMAD NOR BIN HASSAN HASSAN

LINK FOR GITHUB:

Bachelor of Computer Science Mobile Computing with Honors

SEMESTER I 2023/2024

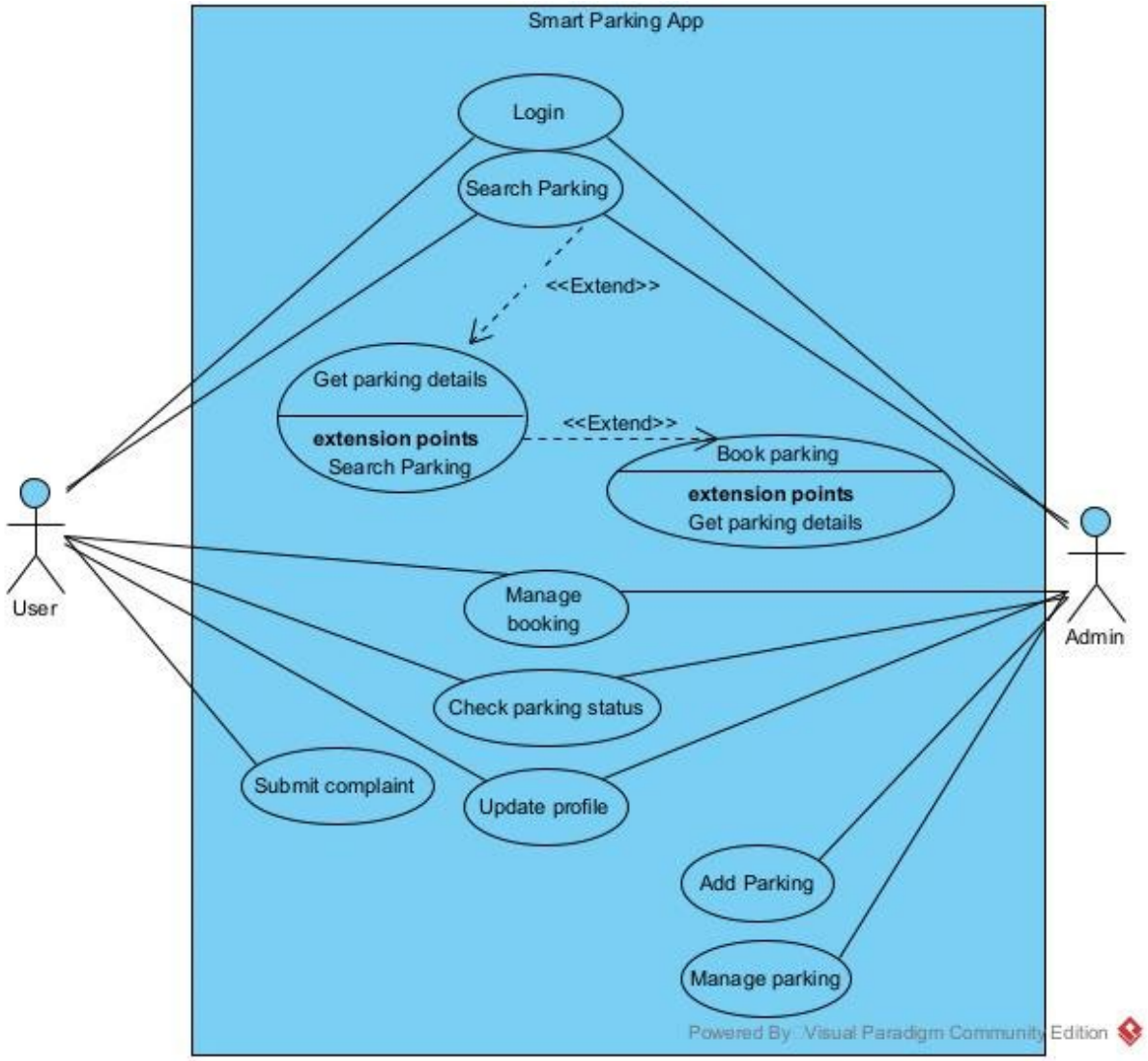
Table of Contents

Executive Summary 3
Use Case 4
Common Structures of Tree Widgets 6
Flutter Widgets & Features 9
Sample Interface With Explanation 11
Conclusion 15
References 16

Excecutive Summary

A user-focused smartphone application, the Smart Parking App aims to improve and streamline the parking experience. Users may easily search and reserve parking spaces in real time, which improves convenience and lessens anxiety associated with parking. Through the parking status checker function of the app, users can visually examine the availability of spots. Users can also provide comments and concerns, which promotes ongoing development. The software puts security first, protecting user data privacy. Real-time insights are provided by admin features for efficient parking place management. The Smart Parking App seeks to transform parking by bringing more organisation and user-friendliness to the process through its scalability for future development, dynamic user experience, and intuitive design.

Use Case



This use case describes the functionality of a Smart Parking App, which allows users to manage various parking-related tasks. The app includes the following features:

1. **Get parking details:** Users can view information about available parking spaces, such as address, slot number, and availability.
2. **Submit complaint:** Users can submit complaints or feedback about the parking experience, such as issues with the app, and parking facilities.
3. **Login:** Users can log in to their account to access their booking history, personal information, and other features.
4. **Search Parking:** Users can search for parking spaces based on various criteria, such as location, and availability.

5. **Manage booking:** Users can view, modify, or delete their existing parking bookings.
6. **Check parking status:** Users can check the status of their parking space, such as whether it is available or occupied.
7. **Update profile:** Users can update their personal information, such as name, email, and password.
8. **Book parking:** Users can book a parking space in advance, selecting the time, and location.
9. **Add Parking:** This feature may allow certain users, such as parking administrators, to add new parking spaces to the app.
10. **Admin:** This feature is for parking administrators to manage parking spaces, such as adding, editing, or removing them from the app.

Common Structures of Tree Widgets

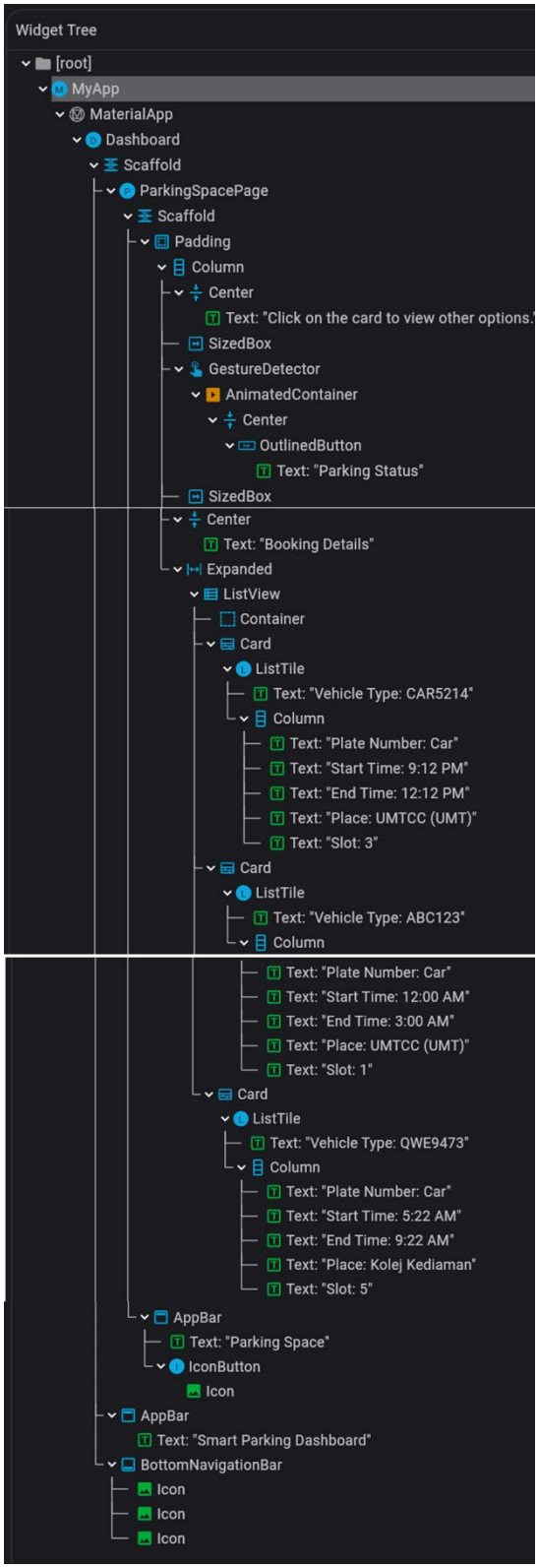


Figure 2.1 Dashboard.dart

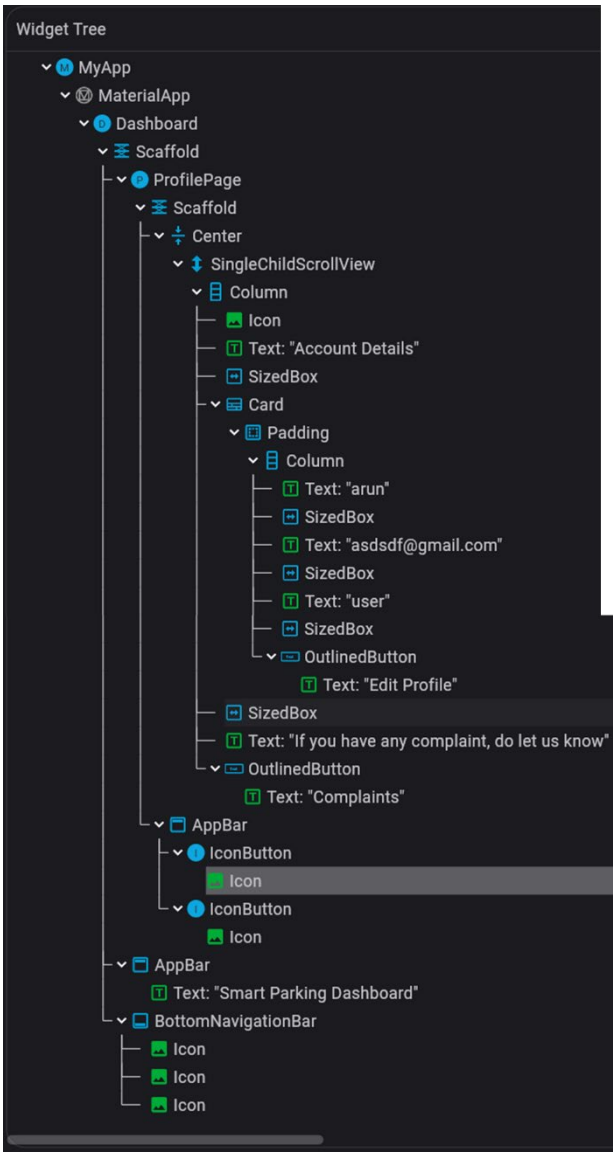


Figure2.2 Profile.dart

Figures indicate that I utilised a common structure of tree widgets (MyApp, MaterialApp, Scaffold, AppBar, BottomNavigationBar, Centre, Column, SingleScrollChildView, and many more) when designing and creating this application. I was unable to show every one of my widget trees because every screen has a unique expertise. The tree widgets listed above are the most commonly utilised when it comes to common structure. I can use the Flutter Inspector to obtain this widget tree structure. This tool is quite useful for learning about widget trees and the data they contain for all displays.

When designing and developing a Smart Parking App, a common structure for the tree widget could involve a hierarchical arrangement of widgets to represent different components and functionalities. Below is a brief explanation of a common tree widget structure for such an app:

1. **App Root (MaterialApp):**
    - The top-level widget that configures the overall structure of the app.
    - Contains MaterialApp settings, theme, and navigation.
2. **Dashboard (StatelessWidget/StatefulWidget):**
    - Represents the initial screen users see when launching the app.
    - Provides access to essential features and navigation to different sections.
3. **Navigation Bar:**
    - A bottom menu accessible from the home screen.
    - Contains links to major app sections, such as Search Parking, Manage Booking, and ViewProfile.
4. **Search Parking Screen (StatefulWidget):**
    - Allows users to search for available parking spots.
    - Includes a search bar, filters, and a list of parking spots.
5. **Booking Screen (StatefulWidget):**
    - Displays details of the selected parking spot and allows users to reserve it.

6. **User Profile Screen (StatefulWidget):**
    - Allows users to view and edit their profile information.
7. **Admin Dashboard (StatefulWidget):**
    - Accessible to administrators for adding and managing parking spots.
    - Includes options for adding, editing, and deleting parking spots, as well as reviewing user feedback.
8. **Complain Screen (StatefulWidget):**
    - Allows users to submit feedback, report issues, and provide comments.
    - Includes a form for input and a submit button.
9. **Common Widgets:**
    - Throughout the app, common widgets like AppBar, BottomNavigationBar, Card, ListViews, and Dialogs are used for consistent design and user experience.

Flutter Widgets & Features

1. **Material Components:**
    - The **MaterialApp** widget is the top-level widget for a Flutter application that follows the Material Design guidelines. It provides essential features such as navigation and theming, ensuring a consistent look and feel across Android and iOS platforms.
2. **Navigation:**
    - Flutter's **Navigator** manages a stack of **PageRoute** objects, facilitating seamless navigation between different screens or pages within the app. It enables the creation of custom page transitions, and the **BottomNavigationBar** widget is commonly used for easy navigation between primary sections of the app.
3. **User Interface Elements:**
    - The **AppBar** widget creates a customizable top app bar, often containing the app's title, navigation icons, and other actions. **Card** widgets present information in a material design card format, suitable for displaying details about parking spots or reservations. **ListView** is used for efficiently displaying scrollable lists of items.
4. **Asynchronous Operations:**
    - The **FutureBuilder** widget simplifies working with asynchronous operations, allowing widgets to react to the completion or error of a **Future**. It's often used when fetching data from a server asynchronously using the **http** package, which handles HTTP requests, and the **json** package, which encodes and decodes JSON data.
5. **Database Integration:**
    - Flutter apps often communicate with backend servers, and the **http** package is crucial for making HTTP requests. It's commonly used for CRUD (Create, Read, Update, Delete) operations with a database. The **json** package helps

with encoding and decoding JSON data exchanged between the app and the server.

6. **State Management:**
    - **StatefulWidget** and **State** work together to manage the mutable state of widgets. For more complex applications, developers may choose advanced state management solutions like Provider, Bloc, Riverpod, or others, to efficiently handle state changes.
7. **Dialogs and Alerts:**
    - The **AlertDialog** widget displays pop-up dialogs for user interaction, providing a way to communicate important information or gather user input. Additionally, the **BottomSheet** widget allows developers to show additional information or options from the bottom of the screen.
8. **Gesture Detection:**
    - The **GestureDetector** widget detects gestures such as taps, swipes, and pinches. It is used to capture user interactions, enabling developers to implement custom gestures within the app.
9. **Form Validation:**
    - The combination of the **Form** and **TextFormField** widgets is commonly used for collecting user input in forms. These widgets provide built-in validation and error handling to ensure the data entered by the user meets specified criteria.
10. **Animations:**
    - In the context of the Smart Parking App, the **AnimatedContainer** widget is likely used to create dynamic and visually appealing UI elements that change over time with smooth animations. The **AnimatedContainer** is a Flutter widget that automatically animates changes to its properties, such as height, width, color, padding, and more.

Sample Interface With Explanation

1. WelcomeScreen

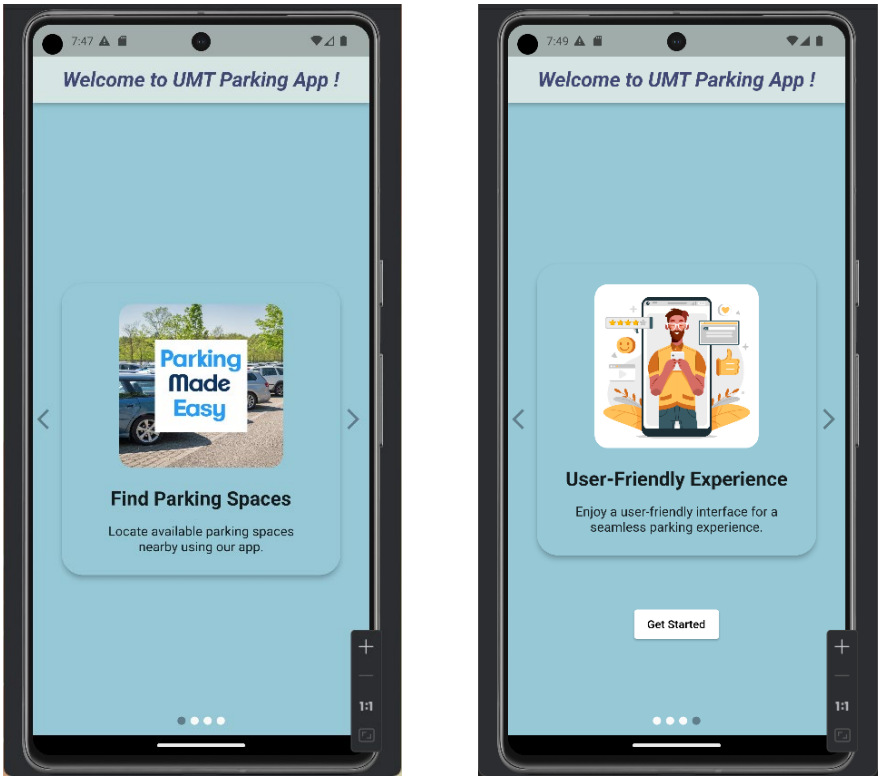


Figure 3.1 & 3.2

2. Login & Register

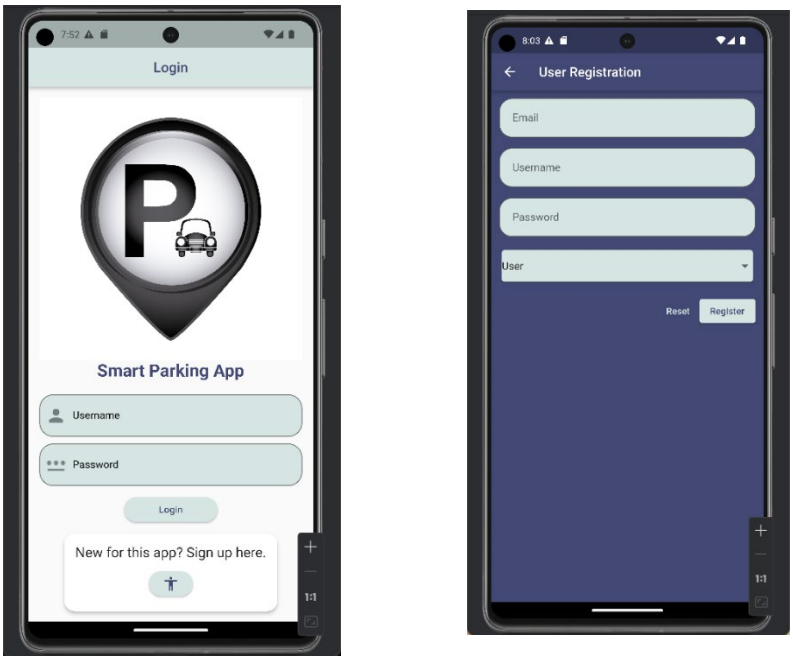


Figure 3.3 & 3.4

3. Dashboard (ParkingSpace, ManageBooking, Profile) == User

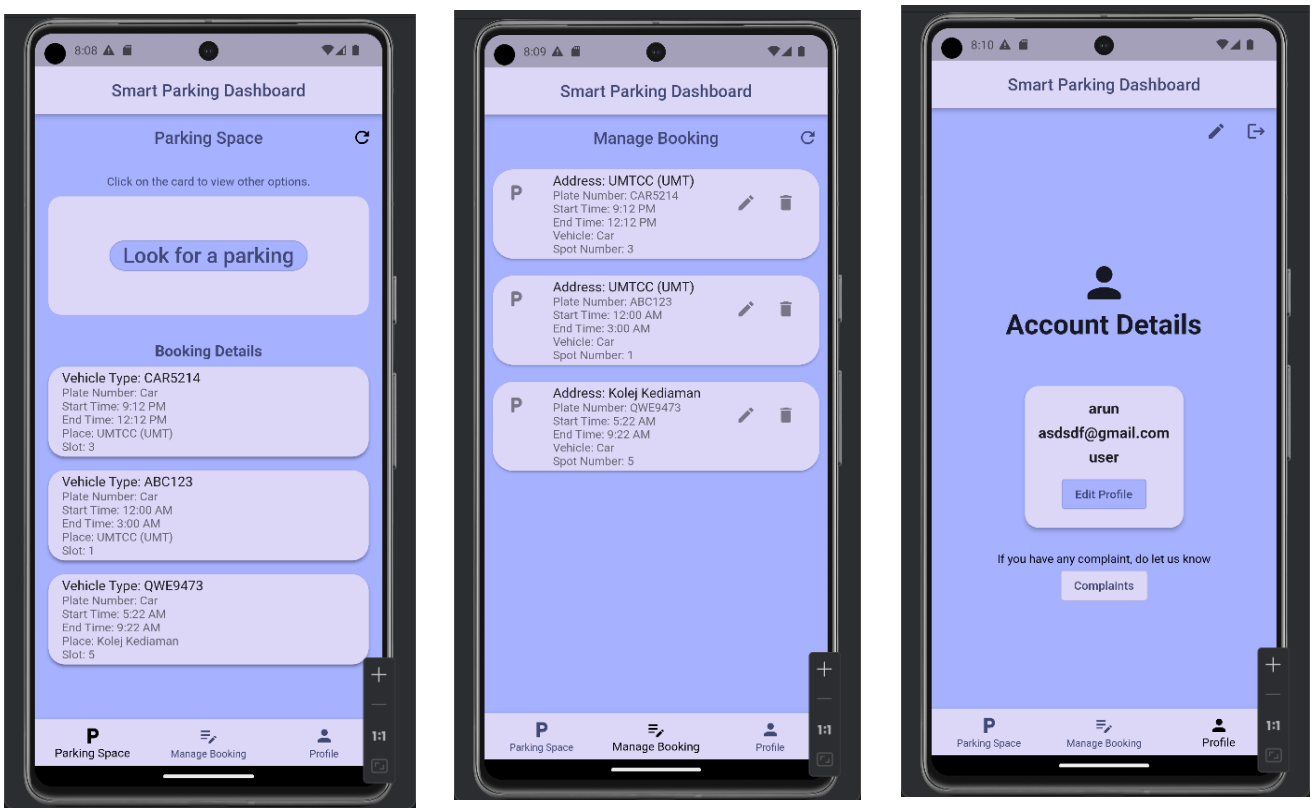


Figure 3.5 & 3.6

4. Dashboard (AddParking, ManageParking) == Only available for admin

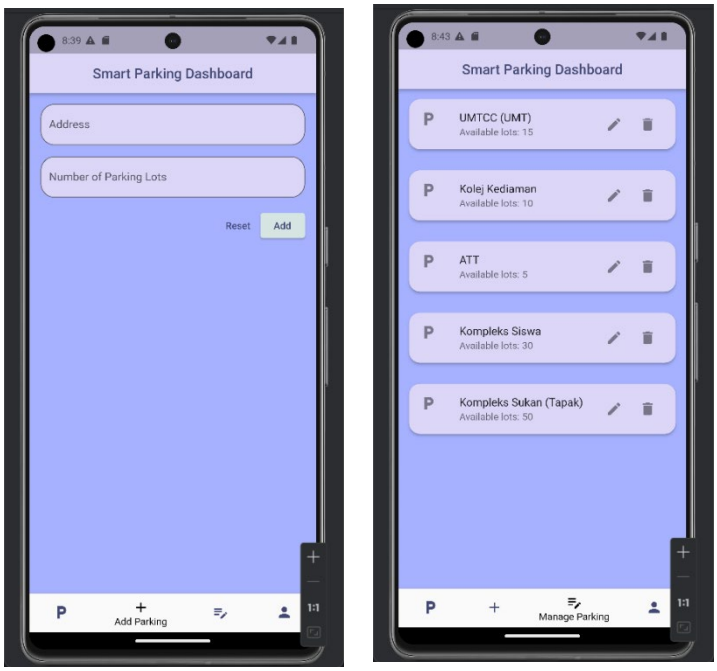


Figure 3.7 & 3.8

5. NearbyParking, ParkingInformation, BookParking

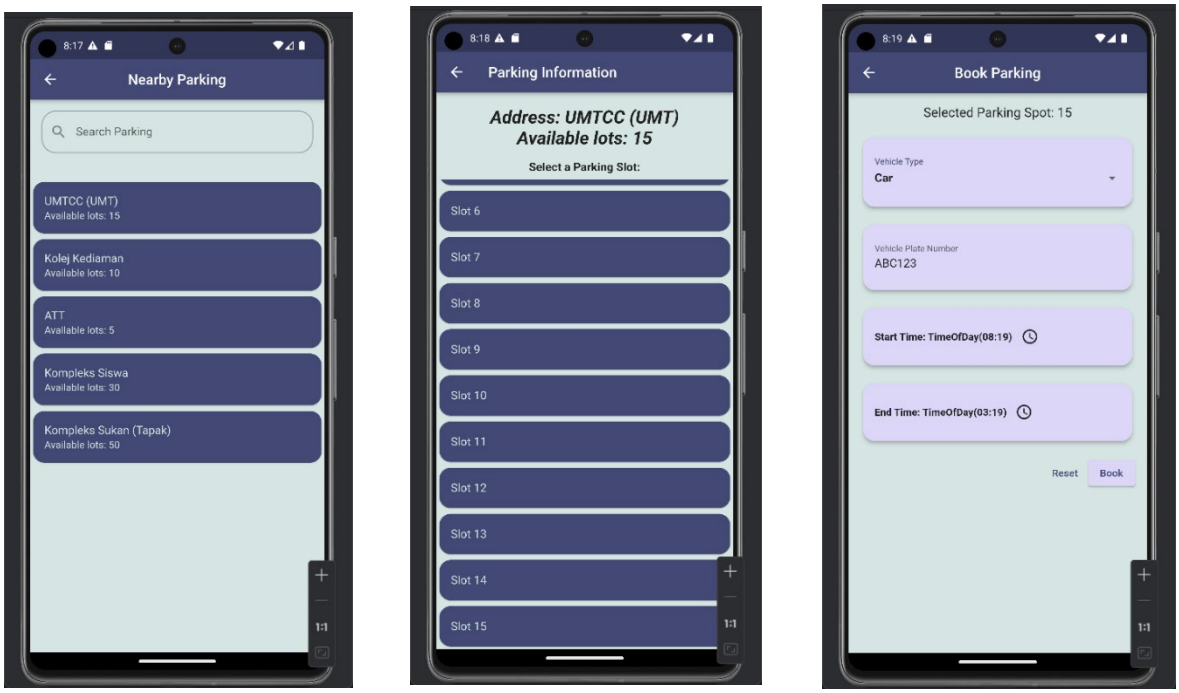


Figure 3.9 & 3.10

6. EditProfile & Complaint

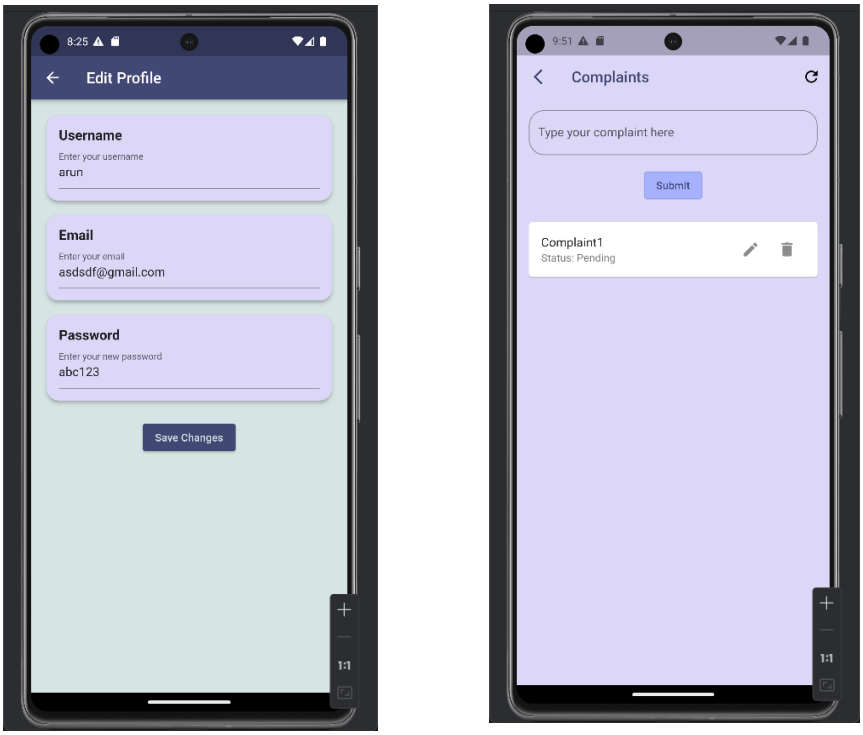


Figure 3.11 & 3.12

7. Parking Status

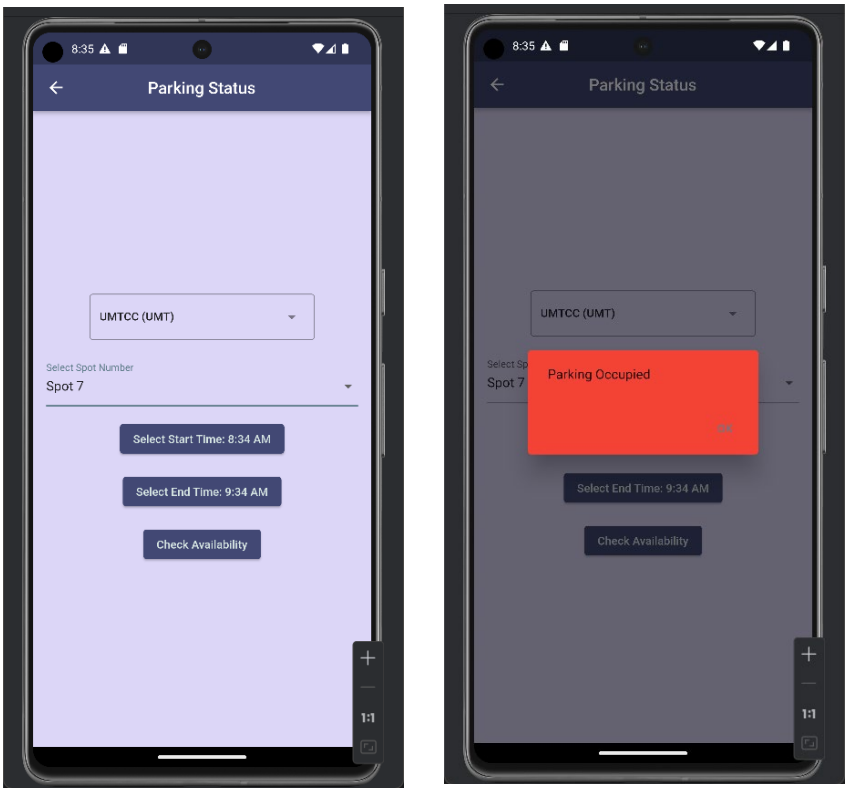


Figure 3.13 & 3.14

Conclusion

To sum up, the Smart Parking App is a complete solution made to improve and simplify users' parking experiences. With the help of contemporary technology and the flexible architecture of Flutter, the application provides a user-friendly interface that makes a variety of capabilities possible, including real-time parking space availability, booking administration, user profiles, and complaint handling. The backend database, Firebase, is used to provide user synchronization and easy data administration. The app shows a dedication to customer convenience and happiness with features including profile updating, complaint filing, and booking confirmation. An elegant and eye-catching user experience is achieved by the Smart Parking App with the use of Flutter's extensive widget collection, animations, and responsive design.

References

1. Chaythanya NairChaythanya Nair 4, Pravin RajPravin Raj 3, cegascegas 2, MizukiMizuki 2, Doan BuiDoan Bui 3, RubsRubs 82999 silver badges2121 bronze badges, Seddiq SorushSeddiq Sorush 2, DwlRathodDwlRathod 76066 silver badges1717 bronze badges, Ajit PaulAjit Paul 21733 silver badges33 bronze badges, utaridutarid 1, lavishlavish 9655 bronze badges, O Thạnh LdtO Thạnh Ldt 1, sammynisammyni 2911 bronze badge, Aryaman GodaraAryaman Godara 5111 silver badge44 bronze badges, MSaudiMSaudi 4, buckleyJohnsonbuckleyJohnson 45944 silver badges1212 bronze badges, dranzerdranzer 3122 bronze badges, Vijay Kumar KantaVijay Kumar Kanta 1, MatiasMatias 7081010 silver badges2424 bronze badges, … Pratik ButaniPratik Butani 61.2k6060 gold badges278278 silver badges441441 bronze badges. (1964, March 1). *How to implement Drop down list in flutter?*. Stack Overflow. https://stackoverflow.com/questions/49273157/how-to-implement-drop-down-list-in-flutter
2. *Futurebuilder class*. FutureBuilder class - widgets library - Dart API. (n.d.). https://api.flutter.dev/flutter/widgets/FutureBuilder-class.html
3. *Dropdownbutton class*. DropdownButton class - material library - Dart API. (n.d.). https://api.flutter.dev/flutter/material/DropdownButton-class.html
4. Wogu, J. (2022, August 19). *Firebase Firestore Crud for flutter*. Medium. https://medium.com/firebase-developers/firebase-firestore-crud-realtime-database-b476ca5f857c
5. *Realtime database: Flutterfire*. FlutterFire Blog RSS. (n.d.). https://firebase.flutter.dev/docs/database/overview/
6. Ahmed El SayedAhmed El Sayed 50911 gold badge88 silver badges1717 bronze badges, dazza5000dazza5000 7, CopsOnRoadCopsOnRoad 248k8282 gold badges667667 silver badges450450 bronze badges, griffinsgriffins 7, & darlingtonhpdarlingtonhp 122 bronze badges. (1964, September 1). *How to get data from Firebase in flutter*. Stack Overflow.

https://stackoverflow.com/questions/52130316/how-to-get-data-from-firebase-in-flutter

7. Gupta, A. (2023, May 13). *CRUD operations with the Firebase Realtime Database in flutter*. Medium. https://anmol-gupta.medium.com/crud-operations-with-the-firebase-realtime-database-in-flutter-e19617c54528
8. Understanding the date, TimeofDay, and datetime property types. (n.d.). https://docs-previous.pega.com/sites/default/files/help_v72/concepts/concepts2/conceptsdatetime.htm#:~:text=TimeofDay%20%E2%80%94%20A%20time%20of%20day,23%2C%202006%20in%20London%2C%20U.%20K.
9. Basudev NayakBasudev Nayak 99411 gold badge99 silver badges2727 bronze badges, Frank van PuffelenFrank van Puffelen 577k8080 gold badges849849 silver badges824824 bronze badges, Sayed IdreesSayed Idrees 1, & matwrmatwr 1. (1967, February 1). *How to save Flutter Timeofday to Firebase?*. Stack Overflow. https://stackoverflow.com/questions/65762491/how-to-save-flutter-timeofday-to-firebase#:~:text=There%20is%20no%20specific%20data,care%20up%20to%20millisecond%20precision)
10. *Animations tutorial*. Flutter. (n.d.). https://docs.flutter.dev/ui/animations/tutorial
11. ChatGPT - https://chat.openai.com/share/c6fef48a-b21e-4e78-b8ed-605043561174