



**UNIVERSITY MALAYSIA TERENGGANU
FACULTY OF OCEAN ENGINEERING TECHNOLOGY & INFORMATICS**

**[CSM3114]
FRAMEWORK-BASED MOBILE APPLICATION DEVELOPMENT
(GROUP 1)**

**PROJECT 2 REPORT
[SMART PARKING APP]**

**PREPARED BY:
NUR IZZATI BT ABDUL MANAF (S61964)**

**PREPARED FOR:
DR. MOHAMAD NOR HASSAN**

[BACHELOR OF COMPUTER SCIENCE (MOBILE COMPUTING) WITH HONOURS] SEMESTER II 2022/2023]

Contents

1.0 Executive Summary.....	3
2.0 USE CASE DIAGRAM	4
3.0 TREE WIDGETS	5
4.0 FLUTTER WIDGET AND FEATURES	6
5.0 USER INTERFACE	8
6.0 CONCLUSION.....	13
7.0 REFERENCE.....	14
GITHUB LINK :.....	15

1.0 Executive Summary

The Smart Parking Apps introduces a user-friendly approach to parking management, starts with a straightforward login process that ensures users can access the app's features. After registration, users can create accounts to access the apps. The data will be sent to the firebase to record users.

For admin, upon registration, admin can access features such as add or update parking details. These details include location and price per hour. The user-friendly interface simplifies the onboarding process for parking lot administrators, ensuring they can efficiently manage parking details within the app.

One of the app's standout features is its ability to provide updates on parking space availability. Users can access information about available parking spots. This function significantly reduces the time users spend searching for parking.

A built-in complaint system empowers users to report issues or express concerns related to the parking facilities. This two-way communication ensures that parking lot administrators receive timely notifications, allowing them to address and resolve concerns promptly. The feedback loop created by the complaint system contributes to continuous improvement in service quality and user satisfaction.

In conclusion, the Smart Parking App is a comprehensive solution that goes beyond mere convenience. It prioritizes user satisfaction and efficient parking management. By seamlessly integrating advanced features, the app aims to elevate the parking experience for all stakeholders involved, setting a new standard in the industry.

2.0 USE CASE DIAGRAM

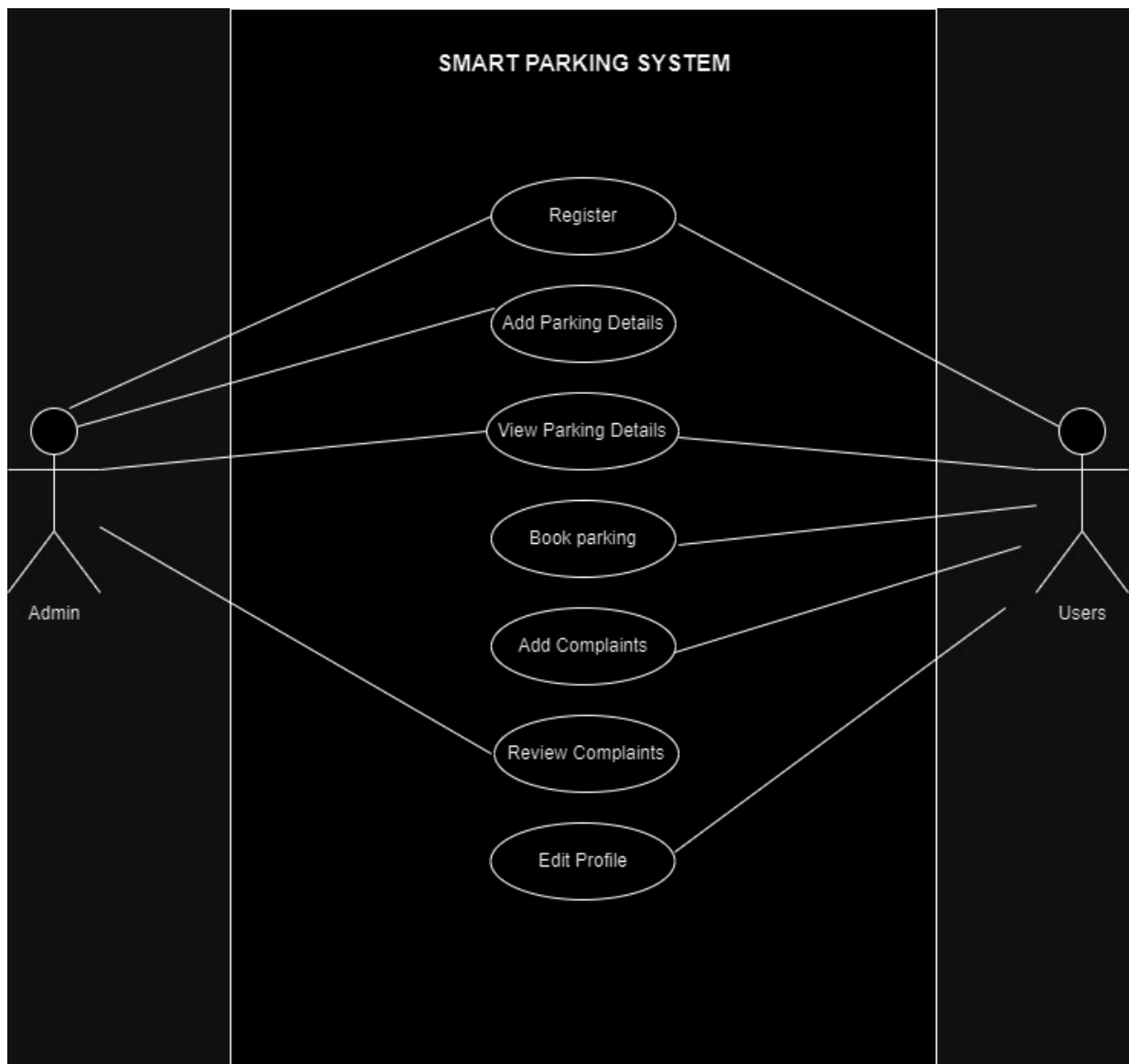


Figure 2.0 shows Smart Parking System Use case diagram

3.0 TREE WIDGETS

The widget tree consist of two users which is admin and users.

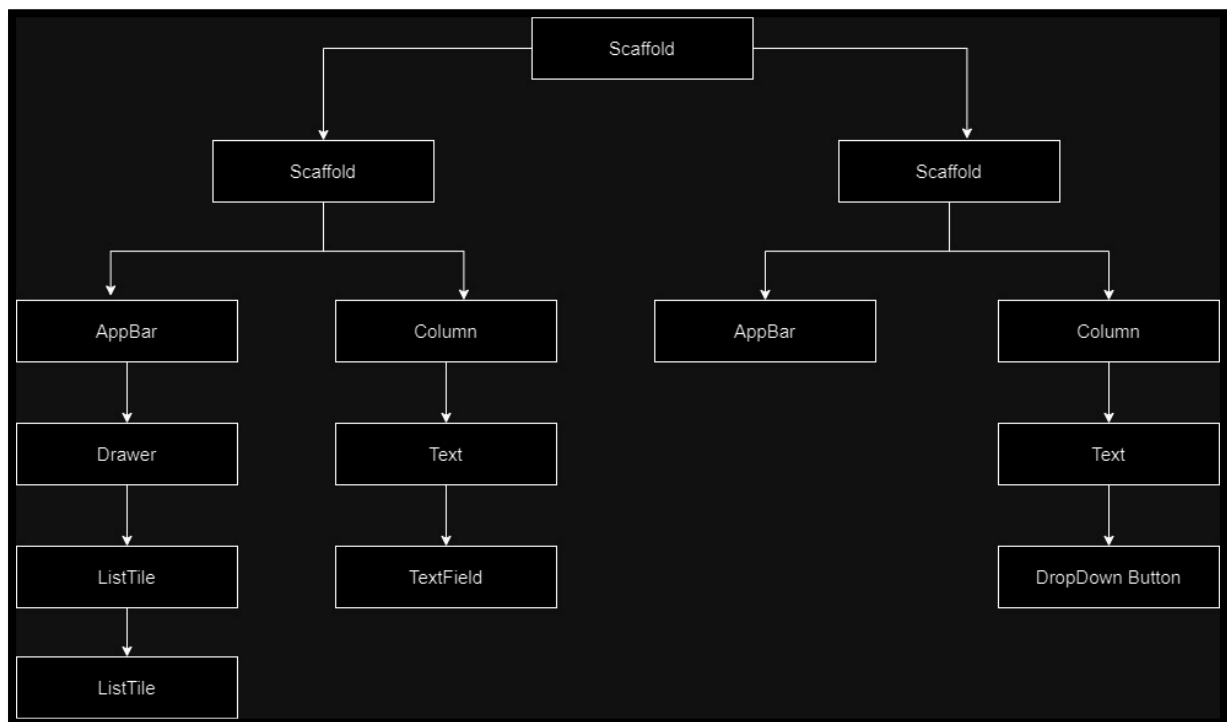


Figure 3.0 shows widget tree of smart parking app

4.0 FLUTTER WIDGET AND FEATURES

There are multiple features that I implemented such as ;

1. **http package:**

The **http** package is used to make HTTP requests. In this case, it's used to send a POST request to a Firebase Realtime Database to add parking locations.

2. **json package:**

The **dart:convert** library is used to encode data in JSON format.

3. **BuildContext:**

BuildContext is used to provide context for the navigation and for building the UI.

4. **Navigator:**

The **Navigator** is used to navigate to the **ParkingPage** when the "Go to Parking Page" button is pressed.

5. **RefreshIndicator:**

The **RefreshIndicator** widget is used to provide a pull-to-refresh functionality for updating the list of complaints.

6. **ListView.builder:**

The **ListView.builder** widget is used to efficiently display a scrolling list of complaints. It only creates widgets for the items that are currently in view.

7. **FutureBuilder:**

Although not explicitly used in the code, the **FutureBuilder** pattern is implied when handling asynchronous operations like fetching complaints. The **fetchComplaints** method returns a **Future<void>**.

8. **Error Handling:**

A basic error handling mechanism is implemented within the **fetchComplaints** method to log errors.

9. **setState:**

The **setState** method is used to trigger a rebuild of the UI whenever the state changes.

10. **DropDownButton:**

The DropDownButton widget is used to create a dropdown for selecting the parking duration in hours.

11. **GridView.builder:**

The **GridView.builder** widget is used to efficiently display a grid of available parking spots.

12. **AlertDialog:**

The **AlertDialog** widget is used to display confirmation dialogs for booking and deleting parking spaces.

13. **VoidCallback:**

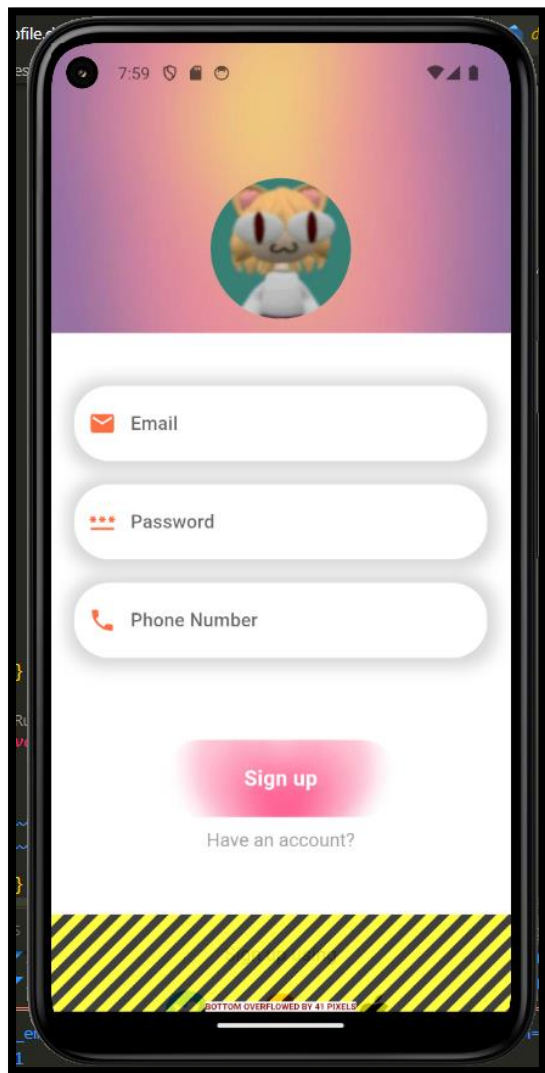
The **VoidCallback** is a callback function type used to handle tap and long-press events in the **ParkingCellWidget**.

14. **Utilization Rate Calculation:**

A method **calculateUtilizationRate** in **ParkingUtilizationService** is used to calculate the utilization rate for a given parking space.

5.0 USER INTERFACE

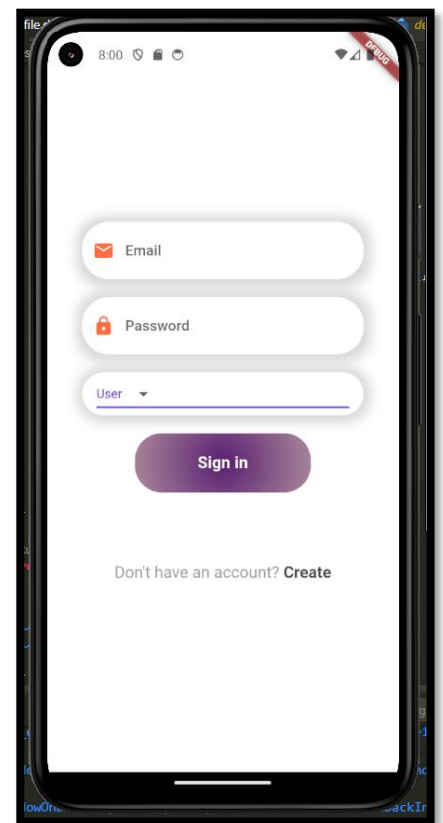
Sign Up Page



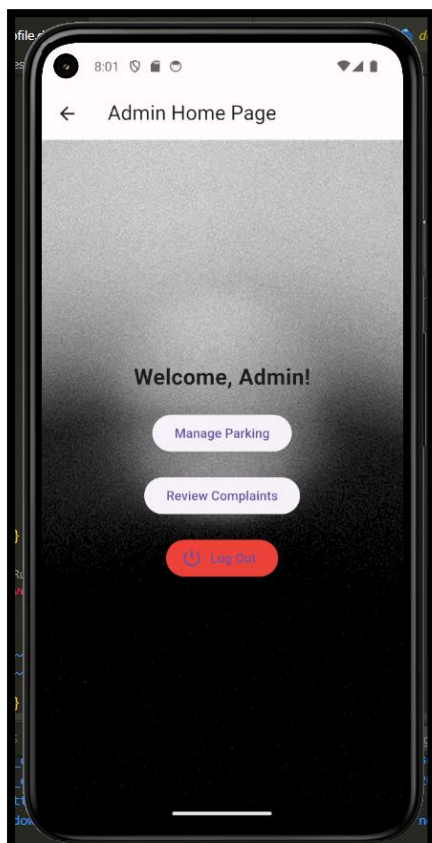
On this page, both users and admin can register their account. Their data then will be passed and stored in firebase when clicked on the button. Then they will be redirected to log in page.

Log in page

On this page, users have to fill in email and password. There will be two selection: users or admin. If choose admin, they will be navigate to Admin homepage. If choose users, will navigate to homepage

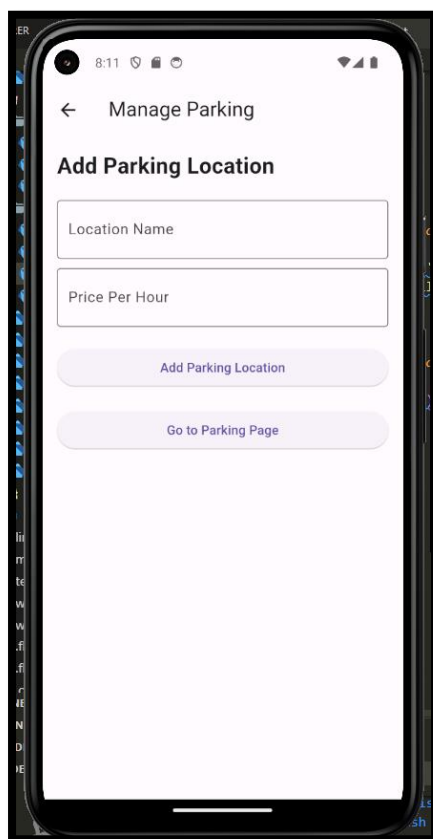


Admin Home Page



Admin can choose 'manage parking' or 'review complaints'. Admin can also log out of their account on this page.

Manage parking page



Manage Parking

Add Parking Location

Location Name

Price Per Hour

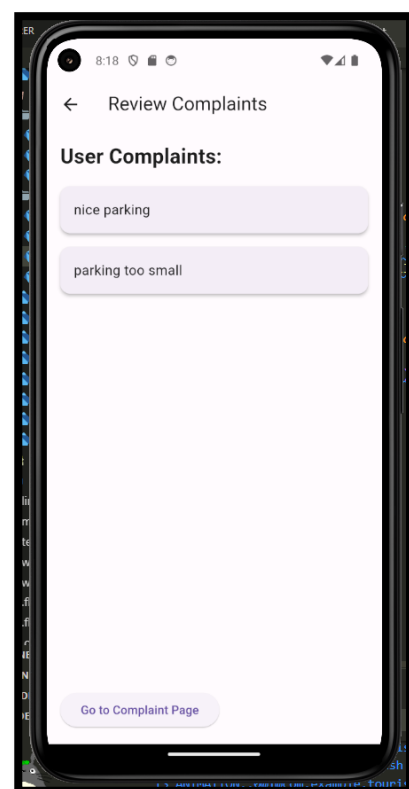
Add Parking Location

Go to Parking Page

Admin can manage parking by adding location name and price per hour for the parking location. Data from the textfield will be stored in firebase by clicking 'add parking location' button. For confirming if data added or not, admin can check it by go to parking page.

Review Complaints page

On this page, admin can review users complaint from users submitted complaint that has been retrieved from firebase.



Review Complaints

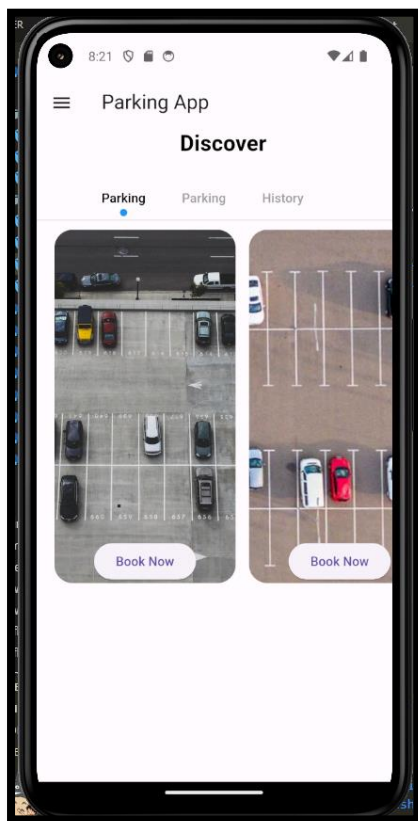
User Complaints:

nice parking

parking too small

Go to Complaint Page

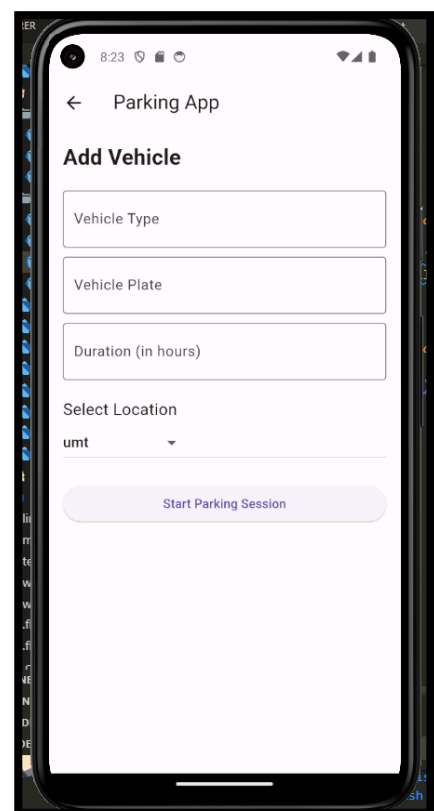
Home page



This is users homepage. There is drawer that navigate through multiple page, including log out session. Users can start book parking session by clicking on 'book now' button .

Parking location

Users can fill in car details, duration, and select location for parking in this page. The details of data then will be stored in firebase.



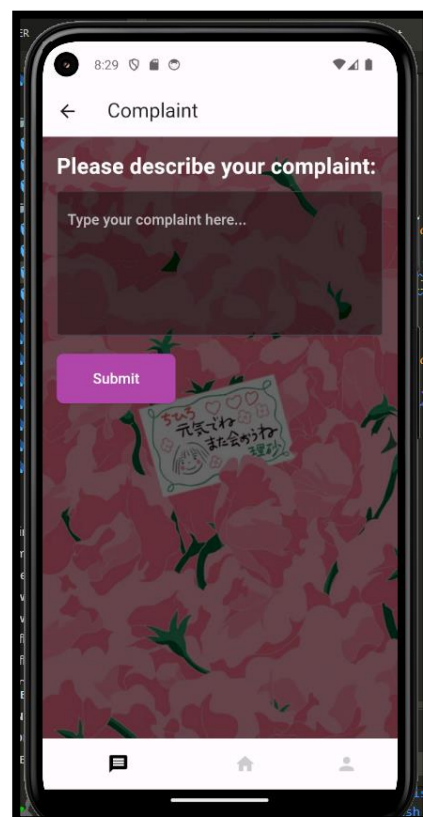
Utilization Parking



Users can select their parking slot by clicking grid displayed with ID. Confirmation dialog will pop up and data will be stored in firebase. Utilization rate of unavailable parking slot will be shown.

Complaint Page

Users can add complaint or feedback in the page. Data submitted will be stored in firebase for admin to review.



6.0 CONCLUSION

In conclusion, the Smart Parking App successfully addresses key functionalities to enhance the parking experience for users. The implemented features include user authentication with login and registration, enabling users to securely access the app and personalize their experience. The ability to register parking lots facilitates efficient management and organization of parking spaces. The app further provides a dynamic system for updating parking status, allowing users to conveniently check the availability of parking spaces.

The incorporation of a complaint system enhances user engagement and satisfaction. Users can easily express concerns or report issues related to parking facilities, fostering a collaborative approach to improving the quality of service. Additionally, the app offers a valuable feature for viewing parking utilization. Users can monitor and assess the occupancy rate of parking lots, promoting informed decision-making and potentially reducing the time spent searching for available spaces. This not only enhances user convenience but also contributes to optimizing the overall efficiency of the parking infrastructure.

Overall, the Smart Parking App successfully combines user-friendly interfaces with essential functionalities, providing a comprehensive solution for modern parking management. By addressing login security, parking lot registration, real-time updates, complaint resolution, and utilization monitoring, the app contributes to a smarter and more user-centric approach to parking solutions.

7.0 REFERENCE

1. Andrianto, I. (2021, August 22). *Flutter - using TabBar & TabBarView examples*. Woolha. <https://www.woolha.com/tutorials/flutter-using-tabbar-tabbarview-examples>
2. Modanwal, N. (2022, January 5). RefreshIndicator in Flutter - FlutterDevs. *Medium*. <https://medium.flutterdevs.com/refreshindicator-in-flutter-db6b228fc13d>
3. *EzKI SmartPark*. (n.d.). Dewan Bandaraya Kuala Lumpur. <https://www.ezsmartpark.com.my/>
4. Sharma, P. (2023, February 5). How to create Seat-booking app in flutter - Prateek Sharma - Medium. *Medium*. <https://medium.com/@sharmaprateek196/how-to-create-seat-booking-layout-in-flutter-33cff82b3edc>
5. Kuldii Project. (2021, November 6). *FLUTTER UI 04 Train Ticket - Design by Bayu Dewantoro (Tutorial Flutter Bahasa Indonesia)* [Video]. YouTube. <https://www.youtube.com/watch?v=Ktv1-vCfc4A>
6. *Create a grid list*. (n.d.). Flutter. <https://docs.flutter.dev/cookbook/lists/grid-lists>
7. Darji, P. (2021, June 14). How to create a grid list in Flutter using GridView - LogRocket Blog. LogRocket Blog. <https://blog.logrocket.com/how-to-create-a-grid-list-in-flutter-using-gridview/>
8. ChatGPT. (n.d.). <https://chat.openai.com/>

GITHUB LINK :

<https://github.com/notpunpun/S61964-Project2.git>