Deliverable 4

The Tech Sense

Smart Indoor Parking System

Group 10

Raghav Sharma	**N01537255**
Kunal Dhiman	**N01540952**
Nisargkumar Pareshbhai Joshi	**N01545986**
Rushi Manojkumar Patel	**N01539144**

Table of Contents

1. Brief Description of the Project

The "Parkit" app connects users with parking lot owners, simplifying the process of finding and booking parking spaces. Users can search for, reserve, and save parking spots, track their reservations, and customize settings such as theme and currency. Additional features like promotions, feedback collection, help support, and notifications enhance user experience and encourage repeat business.

For parking lot owners, the app offers tools to manage locations, available slots, and track their income. Owners can update location details, monitor slot availability, and view income reports, helping them optimize occupancy and attract more customers. This app benefits both users and owners by improving parking efficiency and maximizing revenue.

2. What Issue Your Product Will Solve

ParkIt addresses two major challenges in urban parking. First, it helps drivers find available parking spaces in crowded areas. By providing a real-time map of indoor parking spots, it allows users to easily locate open spaces, saving time, reducing traffic congestion, and minimizing fuel waste. This makes the parking experience more efficient and less stressful, especially in busy urban environments.

Second, ParkIt benefits parking space owners by allowing them to list their spots for rent. This optimizes the utilization of their spaces, ensuring they are not left vacant for long periods. It also creates an additional income stream for owners while providing more parking options for drivers. In this way, our app improves parking availability for users and helps owners make better use of their resources.

3. Compare Your Application with at Least Two Existing Apps

ParkMobile – Allows users to find parking spaces, pay for parking, and manage their parking sessions. [ParkMobile Link] (https://www.parkmobile.io/)
Spot Hero – Focuses on reserving parking spots in advance and provides discounts for certain parking facilities. [Spot Hero Link] (https://www.spothero.com/)

4. Highlight the Differences Between Your App and These Two Apps

1. **Indoor Parking with Sensor Data**

ParkIt uses sensor data to provide real-time updates on parking spot availability, ensuring accurate bookings and immediate availability. Additionally, sensors enhance security by notifying users if their vehicle moves from the spot.

Comparison:

ParkMobile and BestParking lack sensor-driven accuracy for indoor spaces, leading to potential delays and lack of security features.

---

2. **Real-Time Spot Availability**

ParkIt uses sensors to offer live updates, eliminating manual checks and improving efficiency. It also provides real-time security features by tracking whether a car stays in its spot.

Comparison:

BestParking relies on static data, while ParkMobile lacks sensor-based updates for indoor parking.

---

3. **Enhanced User Experience with Sensors**

ParkIt guides users to the nearest available spot using real-time navigation, offering passkey authentication for secure entry and notifications if a car is moved from a spot.

Comparison:

BestParking and ParkMobile lack sensor-based indoor navigation and security features like movement alerts.

---

5. **Benefits of Sensor Data**

Real-Time Monitoring: Continuous tracking ensures up-to-date availability and vehicle security.

Optimized Flow: Guides users to open spots in large garages, improving parking efficiency.

Analytics: Provides owners with insights to manage parking usage, optimize revenue, and track parking space activity.

ParkIt's Key Differentiators

Real-Time Availability: Accurate indoor parking data using sensors.

Indoor Navigation: Real-time guidance to available spots.

Predictive Insights: Sensor data for smarter space management and parking trends.

Security Alerts: Notifies users if their vehicle moves from its designated parking spot.

5. Pros and Cons Table for the Three Apps

Feature	**ParkIt**	**ParkMobile**	**BestParking**
Pros			
Real-Time Availability	Powered by sensor data for accurate updates	-	-
Indoor Parking Focus	Specializes in indoor parking	Limited indoor parking	Limited to outdoor parking
Payment System	Smooth, secure payment system	Easy payment system for outdoor parking	Smooth payment system
Navigation & Security	Sensor-driven data for real-time updates & spot guidance	-	-
Event-Based Parking	-	Supports event-based parking	Focuses on event-based parking
Google Sign-In	Supports Google Sign-in	Supports Google Sign-In	-
Discounts	Promotions	-	Offers discounts on reservations

App	**Cons**
ParkIt	- Limited to indoor parking
	- Still in early stages of development
	- No discounts on reservations
	- No event-based parking
ParkMobile	- Limited indoor parking options
	- No real-time availability for indoor parking
	- Lacks sensor-driven data for indoor parking

	- Average user interface (UI)
BestParking	- No real-time availability
	- No Google Sign-In
	- Limited to outdoor parking
	- Lacks sensor-driven data for indoor parking
	- Average user interface (UI)

6. Signatures Table

Name	Student ID	GitHub ID	Signature	Effort (%)
Raghav Sharma	N01537255	RaghavSharma7255	RS	100%
Kunal Dhiman	N01540952	KunalDhiman0952	KD	100%
Nisarg Kumar Paresh Bhai Joshi	N01545986	NisargJoshi5986	NJ	60%
Rushi Manojkumar Patel	N01539144	RushiPatel9144	RP	50%

7. GitHub Repo Link

https://github.com/RushiPatel9144/SmartIndoorParkingSystem

8. Login Credentials

User's Credentials

Email: aaa@bbb.com
Password: Admin101!

Owner's credentials:

Email: ccc@ddd.com
Password: Admin101!

9. Work Completed by Each Team Member in This Sprint

Raghav Sharma (RS)

Completed Tasks:

- **Integrate Payment in Checkout**: Added a secure and seamless payment gateway for finalizing bookings.
- **Create Stripe Payment Server and Client**: Set up server-side and client-side for Stripe payments, enabling secure transactions.
- **Add Google Sign-Up Using Credential Manager**: Implemented Google Sign-Up for user authentication using the credential manager.
- **Integrate Google Sign-In with Coroutine Helper**: Enhanced Google Sign-In with Coroutine Helper for smoother asynchronous operations.
- **Implement Network Monitoring and Handling**: Added functionality to monitor and manage network connectivity issues.
- **Implement No Network Activity and Utility Methods**: Developed offline handling and error displays for non-internet scenarios.
- **Develop Currency Conversion Functionality Using an API**: Integrated an API for dynamic currency conversion based on user location or preference.
- Implemented Owner functionality for Adding Parking Location, Slots and Sensors.

Kunal Dhiman (KD)

Completed Tasks:

- Updated the UI and functionality of slot spinner, time slots, and payment sheets. This includes designing the UI for the payment sheet, improving the booking bottom sheet dialog fragment, and fixing layout issues in landscape mode.
- Implemented booking functionality, including the Save Booking Method, GeneratePassKey and Expire Passkey methods, and handling cancel and clear options for bookings and booking history.
- Scheduled status updates for bookings, ensuring automated updates for booking statuses.
- Created and enhanced the Payment BottomSheetDialog class to manage payment dialogs effectively.
- Updated and improved the Help UI, ensuring a better user experience in the help section.
- Handled various booking and payment features, combining functionality for booking management and passkey handling.

Nisarg Joshi (NJ)

Completed Tasks:

- Update Feedback as per Requirements: Made necessary updates to the feedback system as per the requirements.
- Implement Functionality & UI for Favorites.
- Implement Functionality & UI for help:

Tasks Not Started:

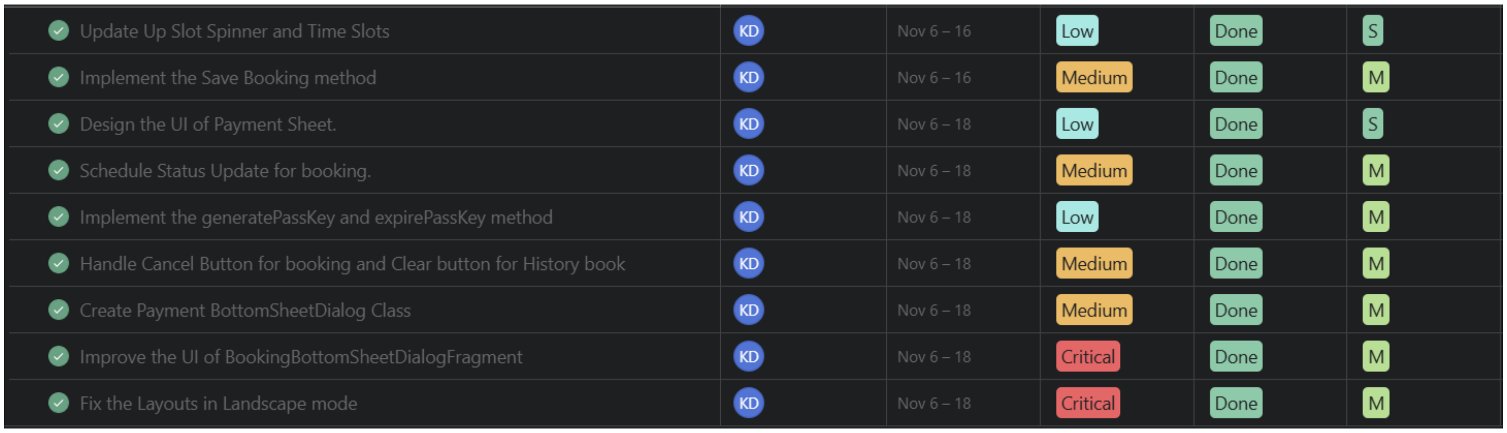
- Update Privacy Policy and Terms of Use: Yet to start.
- Implement Logic for Applied Promotion Coupon: Yet to start.
- Design and Implement Home Screen UI: Yet to start.
- Implement Quick Access Functionality in Home: Added functionality to improve navigation and quick access from the home screen. Yet to start

Rushi Patel (RP)

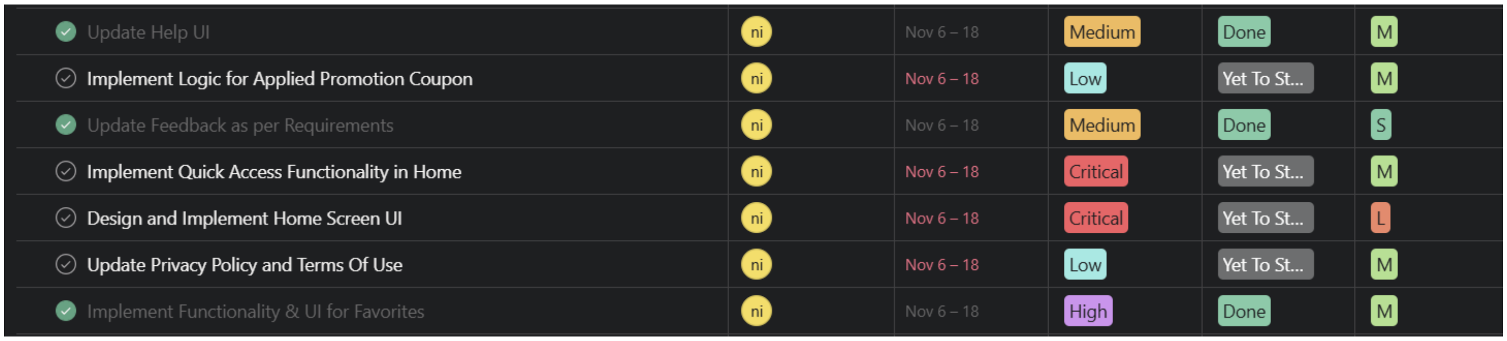
10. Sprint Goals

- ☐ **Complete user authentication and authorization.**
- ☐ **Integrate real-time availability of parking spots.**
- ☐ **Complete UI/UX design for booking and payment screens.**
- ☐ **Enhance app performance and network handling.**
- ☐ **Develop currency conversion feature.**
- ☐ **Improve core functionalities and user feedback features.**
- ☐ **Implement payment system using Stripe.**
- ☐ **Start working on the owner interface.**

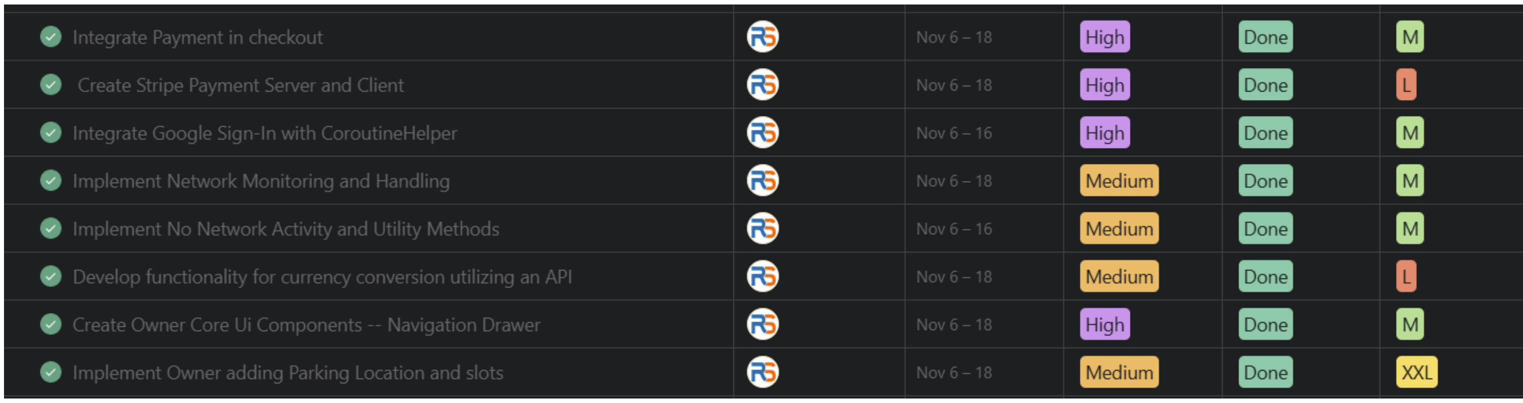
11. Sprint Dashboard



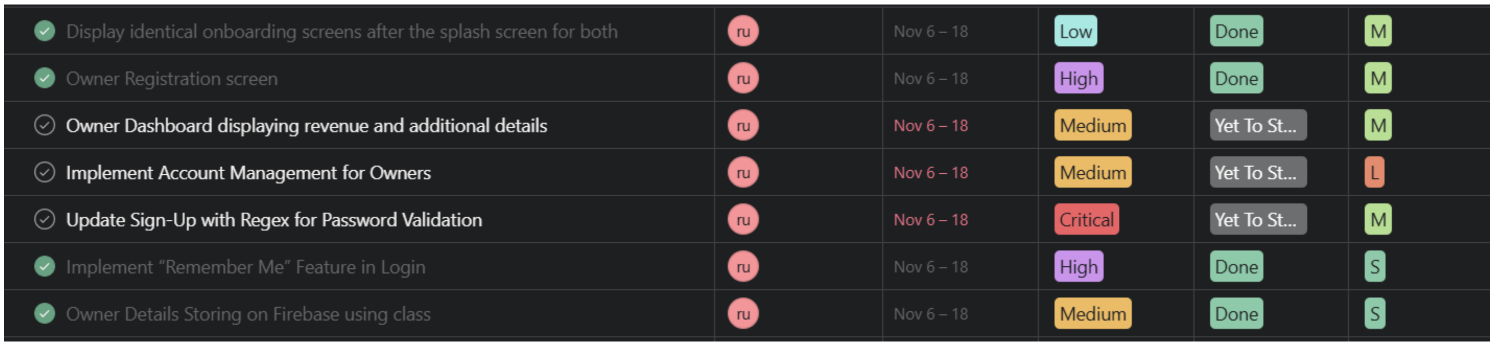
✓	Update Up Slot Spinner and Time Slots	KD	Nov 6 – 16	Low	Done	S
✓	Implement the Save Booking method	KD	Nov 6 – 16	Medium	Done	M
✓	Design the UI of Payment Sheet.	KD	Nov 6 – 18	Low	Done	S
✓	Schedule Status Update for booking.	KD	Nov 6 – 18	Medium	Done	M
✓	Implement the generatePassKey and expirePassKey method	KD	Nov 6 – 18	Low	Done	M
✓	Handle Cancel Button for booking and Clear button for History book	KD	Nov 6 – 18	Medium	Done	M
✓	Create Payment BottomSheetDialog Class	KD	Nov 6 – 18	Medium	Done	M
✓	Improve the UI of BookingBottomSheetDialogFragment	KD	Nov 6 – 18	Critical	Done	M
✓	Fix the Layouts in Landscape mode	KD	Nov 6 – 18	Critical	Done	M



✓	Update Help UI	ni	Nov 6 – 18	Medium	Done	M
○	Implement Logic for Applied Promotion Coupon	ni	Nov 6 – 18	Low	Yet To St...	M
✓	Update Feedback as per Requirements	ni	Nov 6 – 18	Medium	Done	S
○	Implement Quick Access Functionality in Home	ni	Nov 6 – 18	Critical	Yet To St...	M
○	Design and Implement Home Screen UI	ni	Nov 6 – 18	Critical	Yet To St...	L
○	Update Privacy Policy and Terms Of Use	ni	Nov 6 – 18	Low	Yet To St...	M
✓	Implement Functionality & UI for Favorites	ni	Nov 6 – 18	High	Done	M

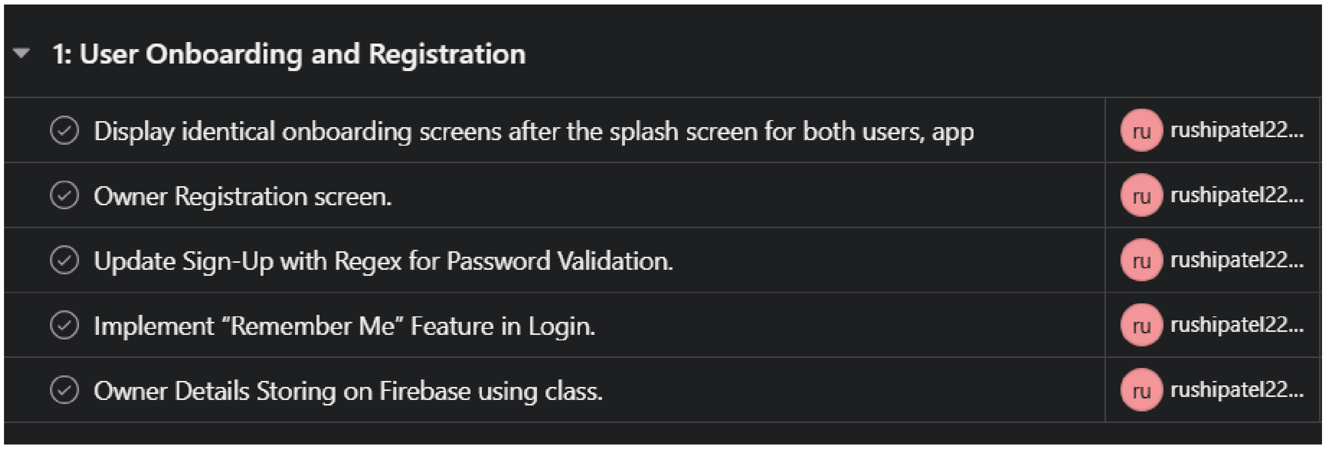


✓	Integrate Payment in checkout	RS	Nov 6 – 18	High	Done	M
✓	Create Stripe Payment Server and Client	RS	Nov 6 – 18	High	Done	L
✓	Integrate Google Sign-In with CoroutineHelper	RS	Nov 6 – 16	High	Done	M
✓	Implement Network Monitoring and Handling	RS	Nov 6 – 18	Medium	Done	M
✓	Implement No Network Activity and Utility Methods	RS	Nov 6 – 16	Medium	Done	M
✓	Develop functionality for currency conversion utilizing an API	RS	Nov 6 – 18	Medium	Done	L
✓	Create Owner Core Ui Components -- Navigation Drawer	RS	Nov 6 – 18	High	Done	M
✓	Implement Owner adding Parking Location and slots	RS	Nov 6 – 18	Medium	Done	XXL

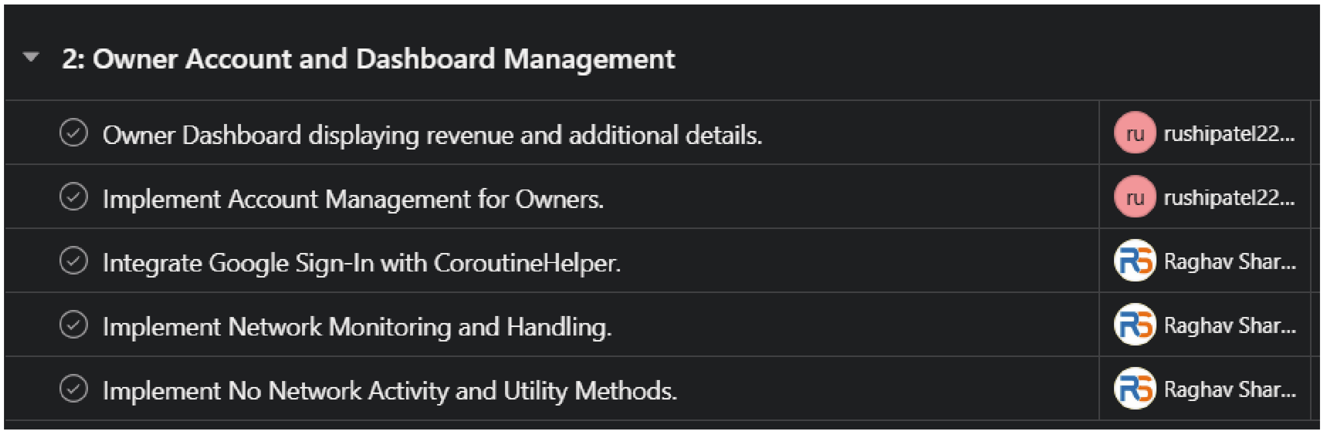


✓	Display identical onboarding screens after the splash screen for both	ru	Nov 6 – 18	Low	Done	M
✓	Owner Registration screen	ru	Nov 6 – 18	High	Done	M
○	Owner Dashboard displaying revenue and additional details	ru	Nov 6 – 18	Medium	Yet To St...	M
○	Implement Account Management for Owners	ru	Nov 6 – 18	Medium	Yet To St...	L
○	Update Sign-Up with Regex for Password Validation	ru	Nov 6 – 18	Critical	Yet To St...	M
✓	Implement "Remember Me" Feature in Login	ru	Nov 6 – 18	High	Done	S
✓	Owner Details Storing on Firebase using class	ru	Nov 6 – 18	Medium	Done	S

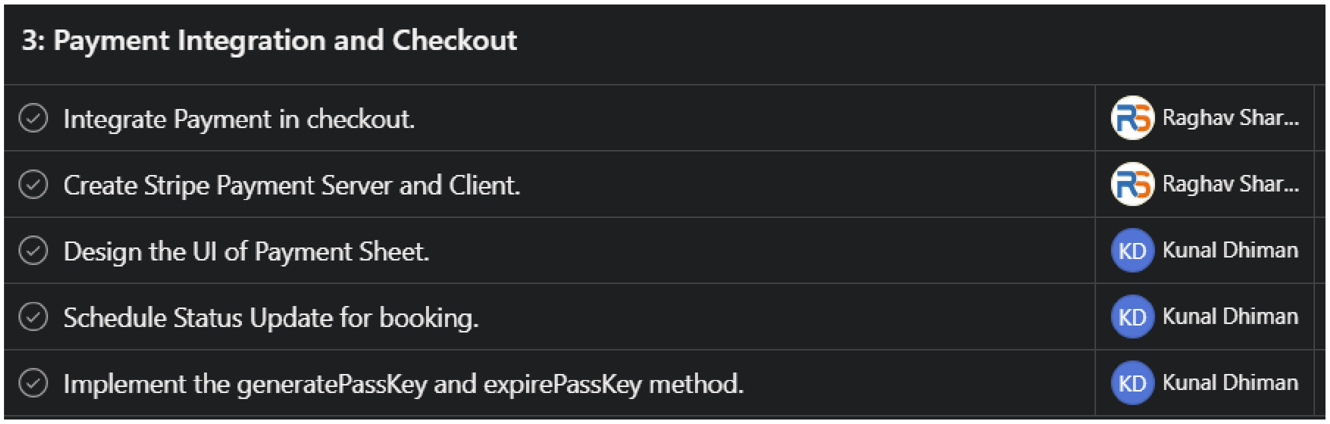
12. Sprint 4 stories



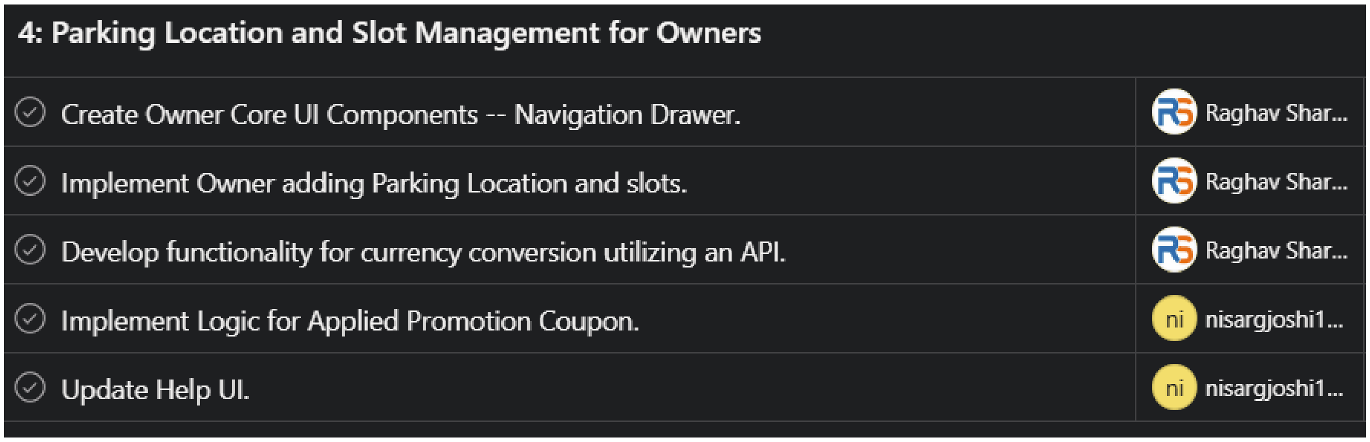
1: User Onboarding and Registration	
Display identical onboarding screens after the splash screen for both users, app	rushipatel22...
Owner Registration screen.	rushipatel22...
Update Sign-Up with Regex for Password Validation.	rushipatel22...
Implement "Remember Me" Feature in Login.	rushipatel22...
Owner Details Storing on Firebase using class.	rushipatel22...



2: Owner Account and Dashboard Management	
Owner Dashboard displaying revenue and additional details.	rushipatel22...
Implement Account Management for Owners.	rushipatel22...
Integrate Google Sign-In with CoroutineHelper.	Raghav Shar...
Implement Network Monitoring and Handling.	Raghav Shar...
Implement No Network Activity and Utility Methods.	Raghav Shar...

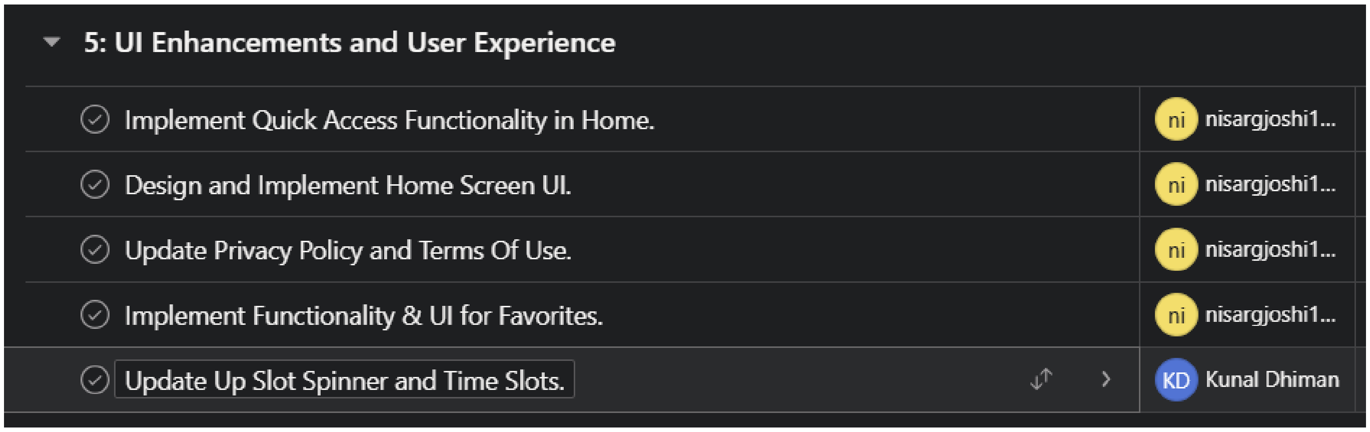


3: Payment Integration and Checkout	
Integrate Payment in checkout.	Raghav Shar...
Create Stripe Payment Server and Client.	Raghav Shar...
Design the UI of Payment Sheet.	Kunal Dhiman
Schedule Status Update for booking.	Kunal Dhiman
Implement the generatePassKey and expirePassKey method.	Kunal Dhiman



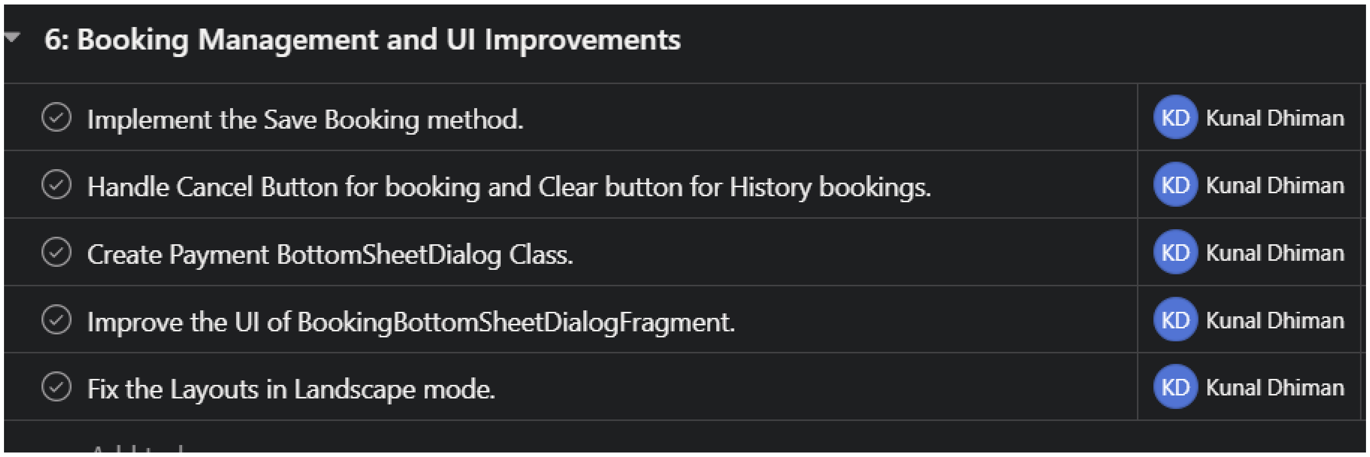
4: Parking Location and Slot Management for Owners

Create Owner Core UI Components -- Navigation Drawer.	Raghav Shar...
Implement Owner adding Parking Location and slots.	Raghav Shar...
Develop functionality for currency conversion utilizing an API.	Raghav Shar...
Implement Logic for Applied Promotion Coupon.	nisargjoshi1...
Update Help UI.	nisargjoshi1...



5: UI Enhancements and User Experience

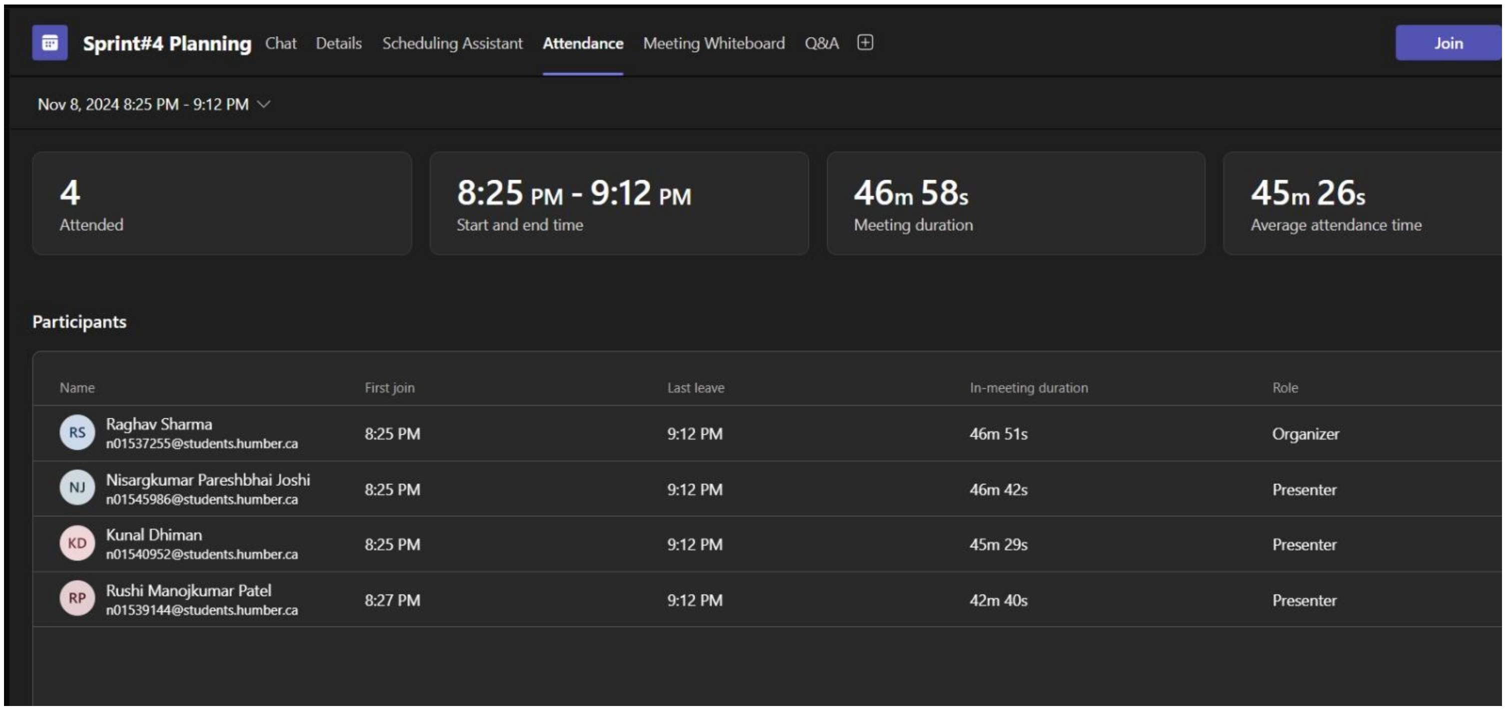
Implement Quick Access Functionality in Home.	nisargjoshi1...
Design and Implement Home Screen UI.	nisargjoshi1...
Update Privacy Policy and Terms Of Use.	nisargjoshi1...
Implement Functionality & UI for Favorites.	nisargjoshi1...
Update Up Slot Spinner and Time Slots.	Kunal Dhiman



6: Booking Management and UI Improvements

Implement the Save Booking method.	Kunal Dhiman
Handle Cancel Button for booking and Clear button for History bookings.	Kunal Dhiman
Create Payment BottomSheetDialog Class.	Kunal Dhiman
Improve the UI of BookingBottomSheetDialogFragment.	Kunal Dhiman
Fix the Layouts in Landscape mode.	Kunal Dhiman

13. Team Discussion and Topics



Nov 8, 2024 8:25 PM - 9:12 PM

4 Attended

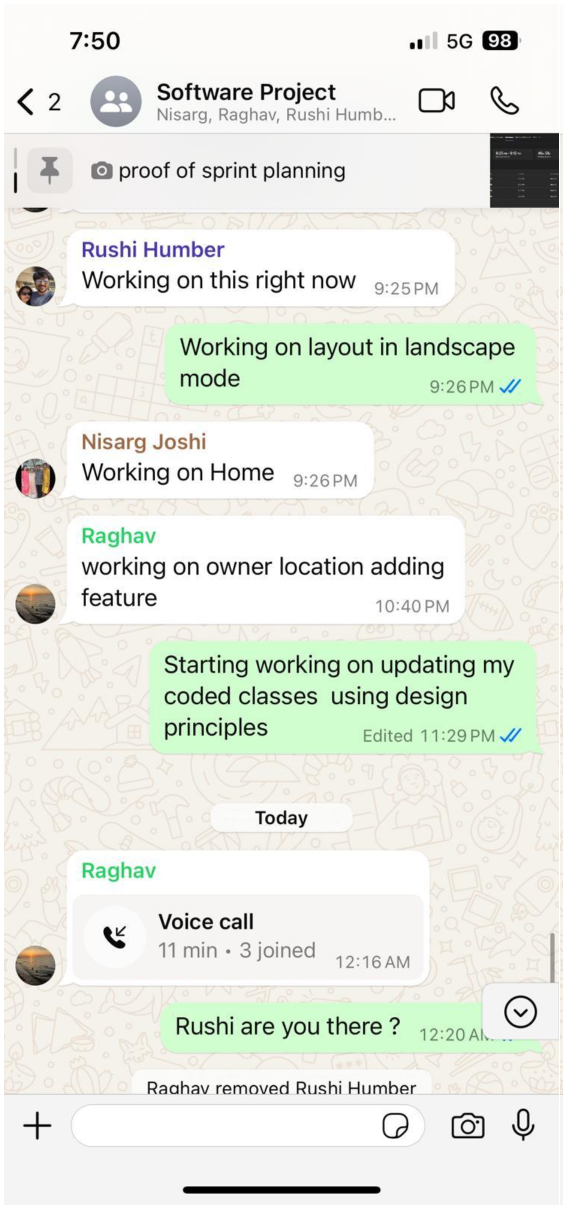
8:25 PM - 9:12 PM Start and end time

46m 58s Meeting duration

45m 26s Average attendance time

Participants

Name	First join	Last leave	In-meeting duration	Role
RS Raghav Sharma n01537255@students.humber.ca	8:25 PM	9:12 PM	46m 51s	Organizer
NJ Nisargkumar Pareshbhai Joshi n01545986@students.humber.ca	8:25 PM	9:12 PM	46m 42s	Presenter
KD Kunal Dhiman n01540952@students.humber.ca	8:25 PM	9:12 PM	45m 29s	Presenter
RP Rushi Manojkumar Patel n01539144@students.humber.ca	8:27 PM	9:12 PM	42m 40s	Presenter



Rushi Humber
Working on this right now 9:25 PM

Working on layout in landscape mode 9:26 PM

Nisarg Joshi
Working on Home 9:26 PM

Raghav
working on owner location adding feature 10:40 PM

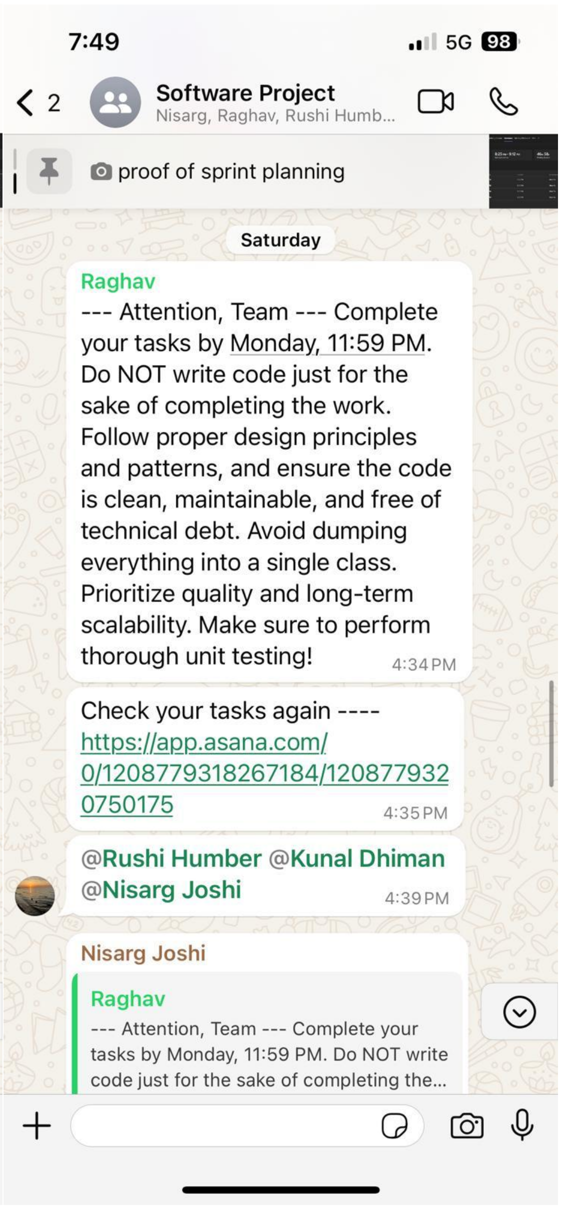
Starting working on updating my coded classes using design principles Edited 11:29 PM

Today

Raghav
Voice call
11 min • 3 joined 12:16 AM

Rushi are you there ? 12:20 AM

Raghav removed Rushi Humber



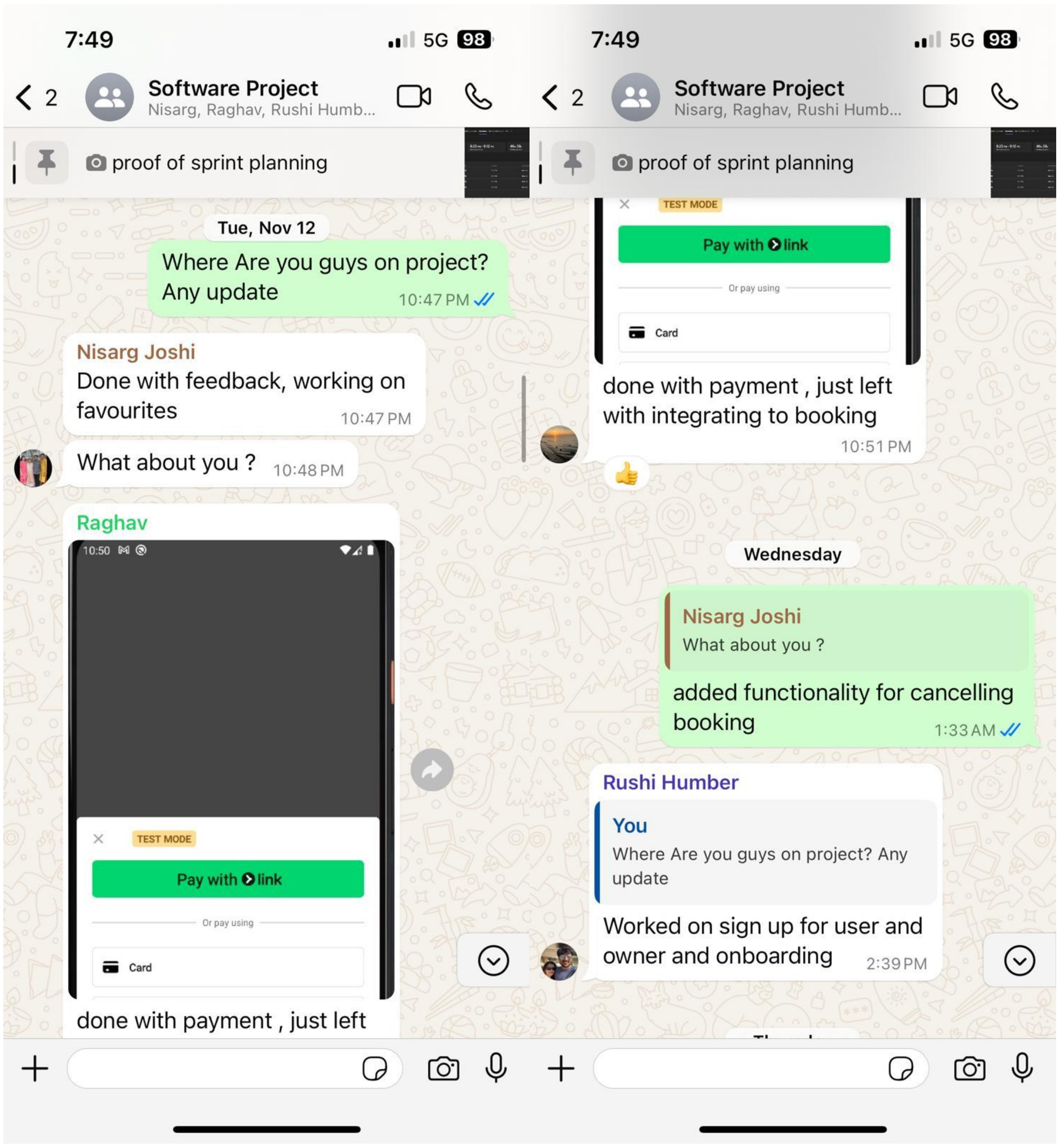
Saturday

Raghav
--- Attention, Team --- Complete your tasks by Monday, 11:59 PM. Do NOT write code just for the sake of completing the work. Follow proper design principles and patterns, and ensure the code is clean, maintainable, and free of technical debt. Avoid dumping everything into a single class. Prioritize quality and long-term scalability. Make sure to perform thorough unit testing! 4:34 PM

Check your tasks again ---- https://app.asana.com/0/1208779318267184/1208779320750175 4:35 PM

@Rushi Humber @Kunal Dhiman @Nisarg Joshi 4:39 PM

Nisarg Joshi
Raghav
--- Attention, Team --- Complete your tasks by Monday, 11:59 PM. Do NOT write code just for the sake of completing the...


7:49 5G 98

Software Project
Nisarg, Raghav, Rushi Humb...

proof of sprint planning

Tue, Nov 12

Where Are you guys on project? Any update 10:47 PM

Nisarg Joshi
Done with feedback, working on favourites 10:47 PM

What about you ? 10:48 PM

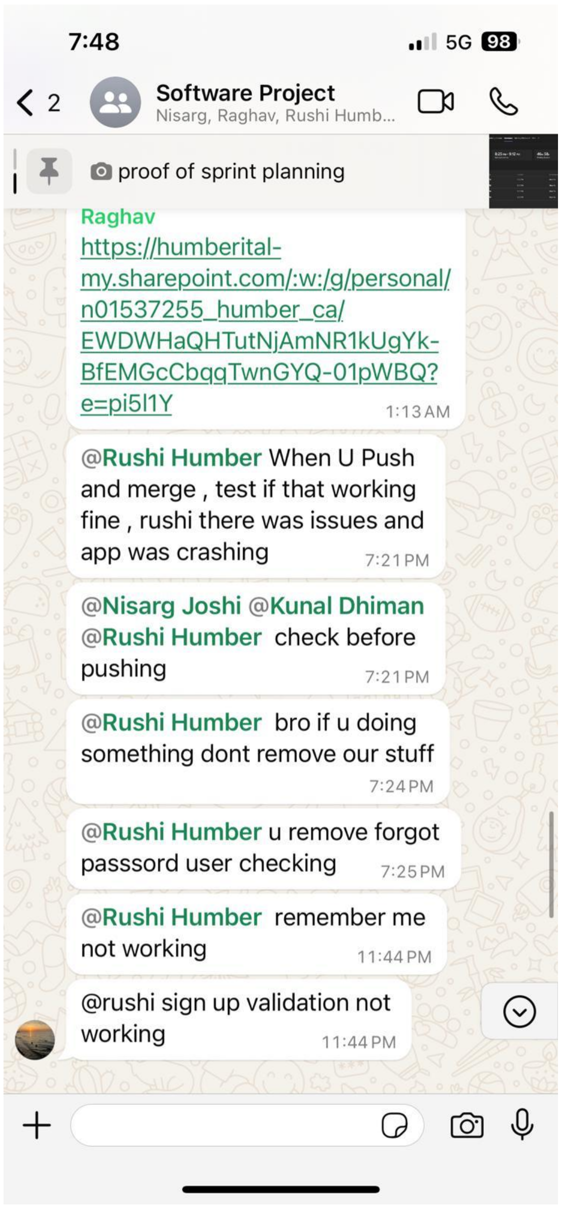
Raghav

done with payment , just left with integrating to booking 10:51 PM

Wednesday

Nisarg Joshi
What about you ?
added functionality for cancelling booking 1:33 AM

Rushi Humber
You
Where Are you guys on project? Any update
Worked on sign up for user and owner and onboarding 2:39 PM



7:48 5G 98

proof of sprint planning

Raghav
https://humberital-my.sharepoint.com/:w:/g/personal/n01537255_humber_ca/EWDWHaQHTutNjAmNR1kUgYk-BfEMGcCbqqTwnGYQ-01pWBQ?e=pi5l1Y 1:13 AM

@Rushi Humber When U Push and merge , test if that working fine , rushi there was issues and app was crashing 7:21 PM

@Nisarg Joshi @Kunal Dhiman @Rushi Humber check before pushing 7:21 PM

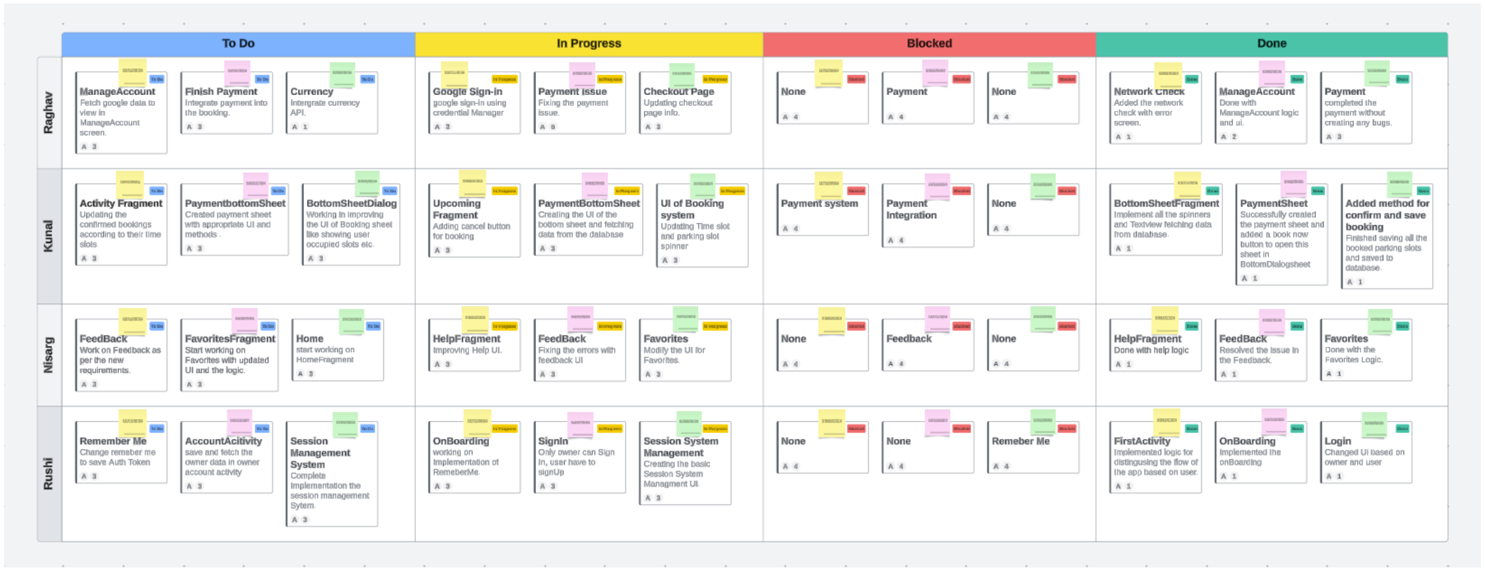
@Rushi Humber bro if u doing something dont remove our stuff 7:24 PM

@Rushi Humber u remove forgot passsord user checking 7:25 PM

@Rushi Humber remember me not working 11:44 PM

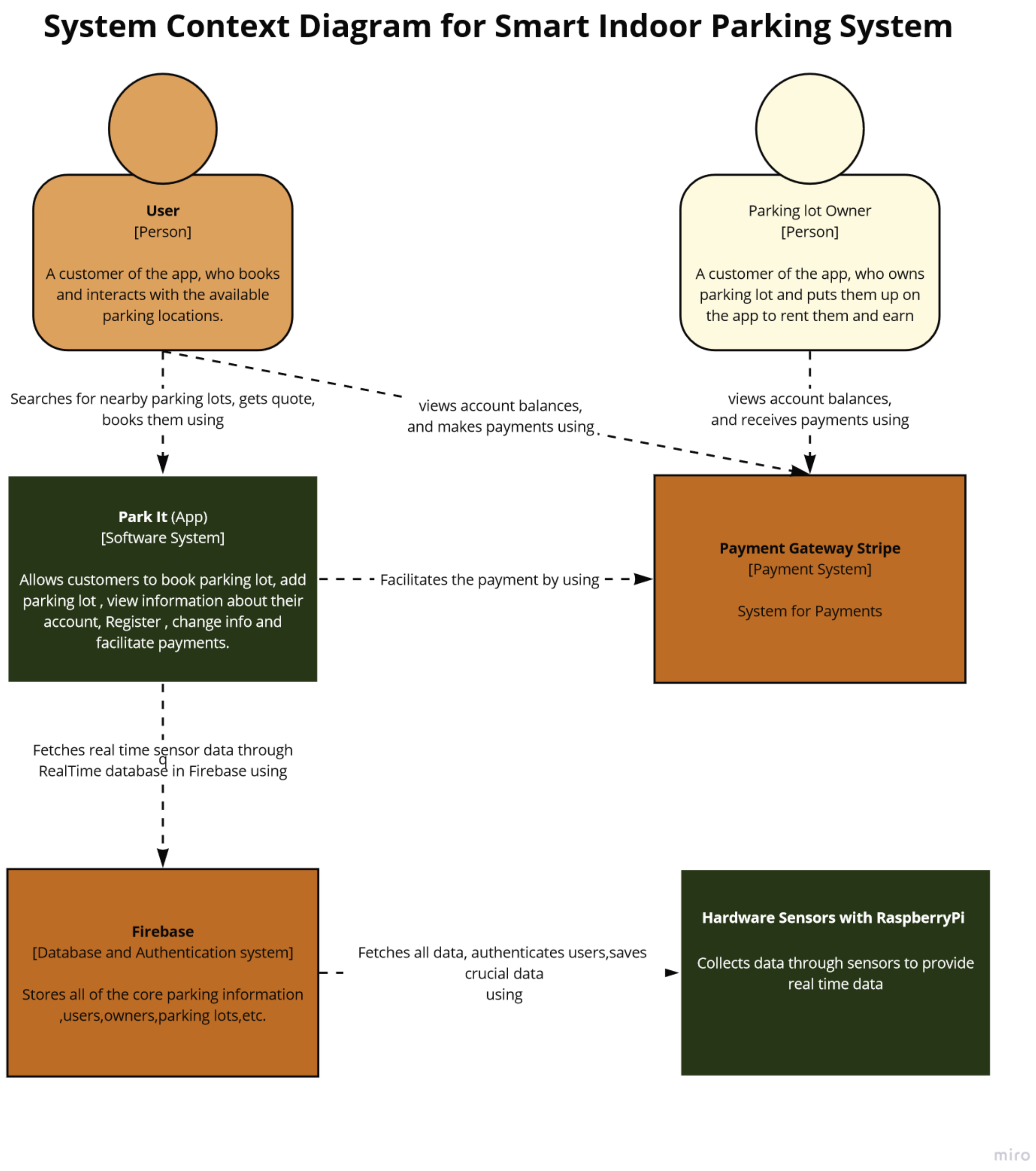
@rushi sign up validation not working 11:44 PM

14. Sprint Retrospective

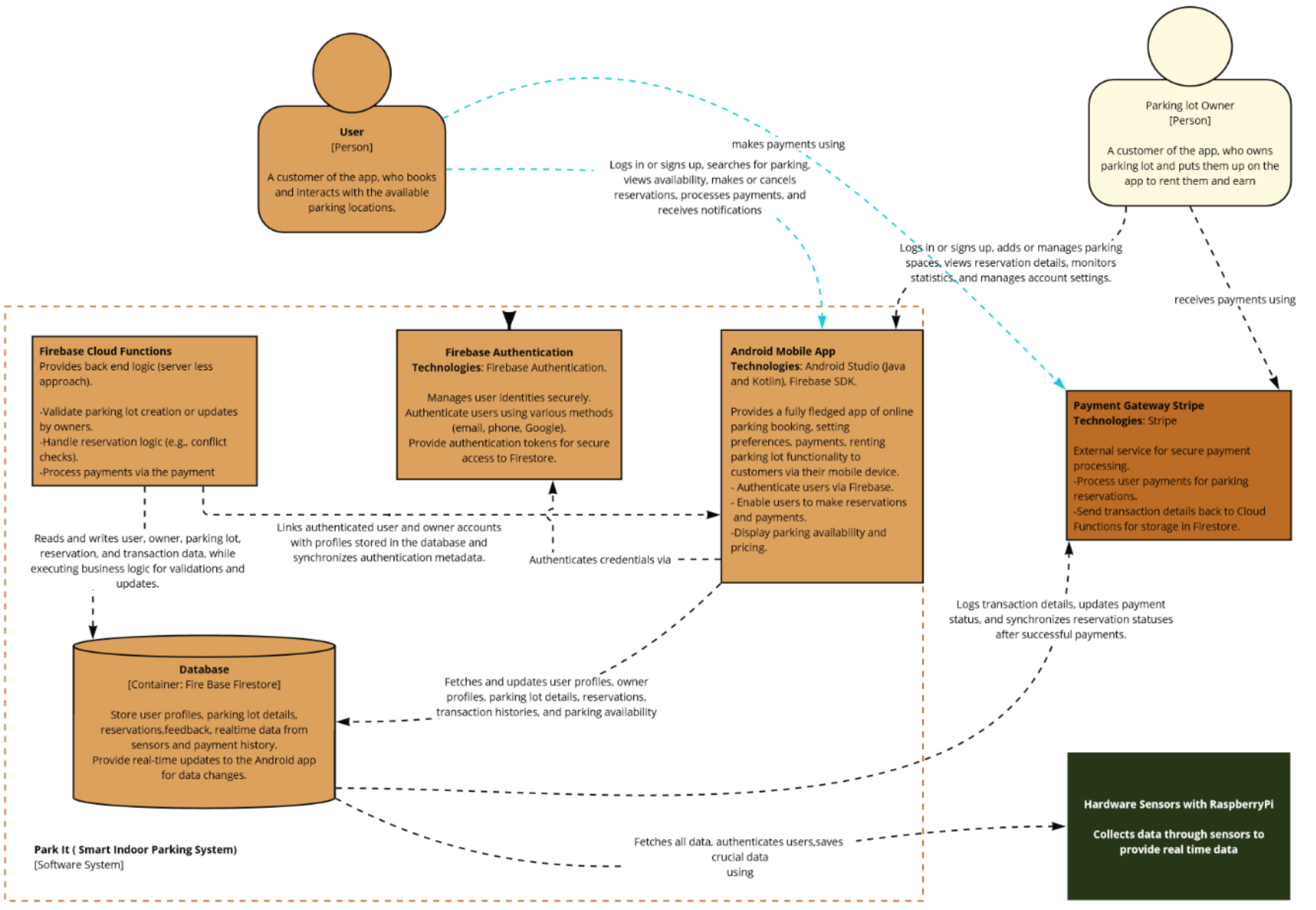


	To Do	In Progress	Blocked	Done
Raghav	**ManageAccount** Fetch google data to view in ManageAccount screen. A 3 **Finish Payment** Integrate payment into the booking. A 3 **Currency** Intergrate currency API. A 1	**Google Sign-in** google sign-in using credential Manager A 3 **Payment issue** Fixing the payment issue. A 5 **Checkout Page** Updating checkout page info. A 3	**None** A 4 **Payment** A 4 **None** A 4	**Network Check** Added the network check with error screen. A 1 **ManageAccount** Done with ManageAccount logic and ui. A 2 **Payment** completed the payment without creating any bugs. A 3
Kunal	**Activity Fragment** Updating the confirmed bookings according to their time slots A 3 **PaymentbottomSheet** Created payment sheet with appropriate UI and methods . A 3 **BottomSheetDialog** Working in improving the UI of Booking sheet like showing user occupied slots etc. A 3	**Upcoming Fragment** Adding cancel button for booking A 3 **PaymentBottomSheet** Creating the UI of the bottom sheet and fetching data from the database A 3 **UI of Booking system** Updating Time slot and parking slot spinner A 3	**Payment system** A 4 **Payment Integration** A 4 **None** A 4	**BottomSheetFragment** Implement all the spinners and Textview fetching data from database. A 1 **PaymentSheet** Successfully created the payment sheet and added a book now button to open this sheet in BottomDialogsheet A 1 **Added method for confirm and save booking** Finished saving all the booked parking slots and saved to database. A 1
Nisarg	**FeedBack** Work on Feedback as per the new requirements. A 3 **FavoritesFragment** Start working on Favorites with updated UI and the logic. A 3 **Home** start working on HomeFragment A 3	**HelpFragment** Improving Help UI. A 3 **FeedBack** Fixing the errors with feedback UI A 3 **Favorites** Modify the UI for Favorites. A 3	**None** A 4 **Feedback** A 4 **None** A 4	**HelpFragment** Done with help logic A 1 **FeedBack** Resolved the Issue in the Feedback. A 1 **Favorites** Done with the Favorites Logic. A 1
Rushi	**Remember Me** Change remeber me to save Auth Token A 3 **AccountAcitivity** save and fetch the owner data in owner account activity A 3 **Session Management System** Complete Implementation the session management Sytem. A 3	**OnBoarding** working on Implementation of RemeberMe. A 3 **SignIn** Only owner can Sign In, user have to signUp A 3 **Session System Management** Creating the basic Session System Managment UI. A 3	**None** A 4 **None** A 4 **Remeber Me** A 4	**FirstActivity** Implemented logic for distingusing the flow of the app based on user. A 1 **OnBoarding** Implemented the onBoarding A 1 **Login** Changed UI based on owner and user A 1

15. System Context Diagram (C4 Model)


System Context Diagram for Smart Indoor Parking System
User
[Person]
A customer of the app, who books and interacts with the available parking locations.
Parking lot Owner
[Person]
A customer of the app, who owns parking lot and puts them up on the app to rent them and earn
Searches for nearby parking lots, gets quote, books them using
views account balances, and makes payments using
views account balances, and receives payments using
Park It (App)
[Software System]
Allows customers to book parking lot, add parking lot , view information about their account, Register , change info and facilitate payments.
Facilitates the payment by using
Payment Gateway Stripe
[Payment System]
System for Payments
Fetches real time sensor data through RealTime database in Firebase using
Firebase
[Database and Authentication system]
Stores all of the core parking information ,users,owners,parking lots,etc.
Fetches all data, authenticates users,saves crucial data using
Hardware Sensors with RaspberryPi
Collects data through sensors to provide real time data
miro

16. Container Diagram (C4 Model)

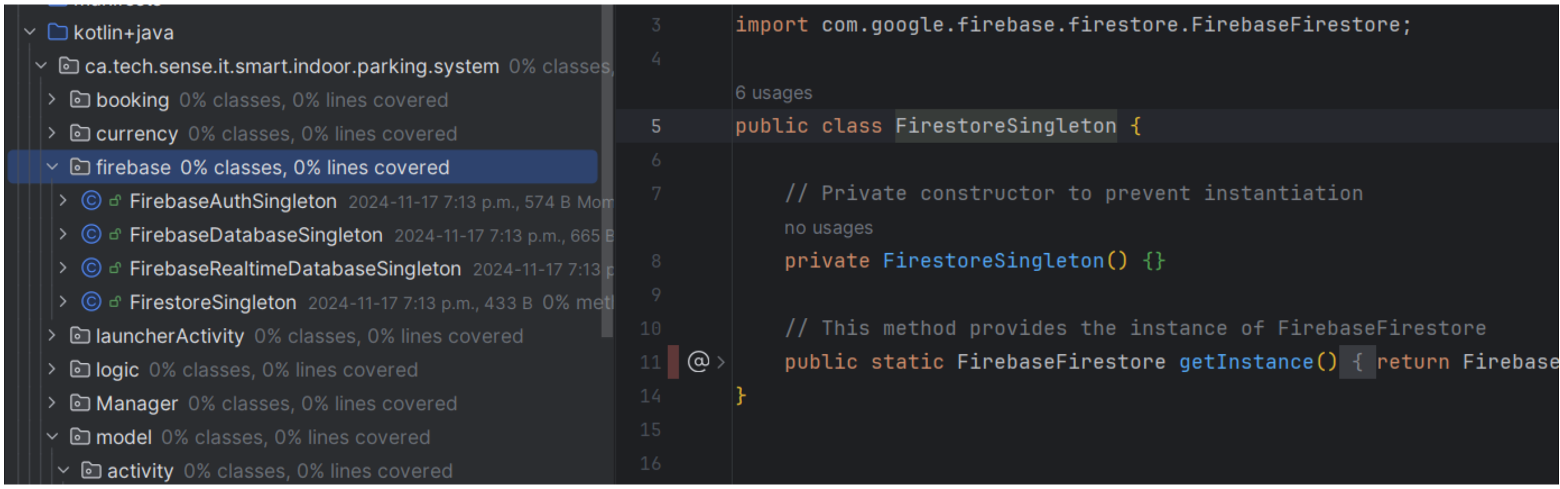

User
[Person]
A customer of the app, who books and interacts with the available parking locations.
makes payments using
Logs in or signs up, searches for parking, views availability, makes or cancels reservations, processes payments, and receives notifications
Parking lot Owner
[Person]
A customer of the app, who owns parking lot and puts them up on the app to rent them and earn
Logs in or signs up, adds or manages parking spaces, views reservation details, monitors statistics, and manages account settings.
receives payments using
Firebase Cloud Functions
Provides back end logic (server less approach).
-Validate parking lot creation or updates by owners.
-Handle reservation logic (e.g., conflict checks).
-Process payments via the payment
Firebase Authentication
Technologies: Firebase Authentication.
Manages user identities securely. Authenticate users using various methods (email, phone, Google). Provide authentication tokens for secure access to Firestore.
Android Mobile App
Technologies: Android Studio (Java and Kotlin), Firebase SDK.
Provides a fully fledged app of online parking booking, setting preferences, payments, renting parking lot functionality to customers via their mobile device.
- Authenticate users via Firebase.
- Enable users to make reservations and payments.
-Display parking availability and pricing.
Payment Gateway Stripe
Technologies: Stripe
External service for secure payment processing.
-Process user payments for parking reservations.
-Send transaction details back to Cloud Functions for storage in Firestore.
Reads and writes user, owner, parking lot, reservation, and transaction data, while executing business logic for validations and updates.
Links authenticated user and owner accounts with profiles stored in the database and synchronizes authentication metadata.
Authenticates credentials via
Logs transaction details, updates payment status, and synchronizes reservation statuses after successful payments.
Database
[Container: Fire Base Firestore]
Store user profiles, parking lot details, reservations,feedback, realtime data from sensors and payment history.
Provide real-time updates to the Android app for data changes.
Fetches and updates user profiles, owner profiles, parking lot details, reservations, transaction histories, and parking availability
Hardware Sensors with RaspberryPi
Collects data through sensors to provide real time data
Fetches all data, authenticates users,saves crucial data using
Park It (Smart Indoor Parking System)
[Software System]

Container Diagram for Smart Indoor Parking System

17. Design Patterns

- **Requirement**: Document two design patterns. Copy the code you used and add your explanation. Use the ones covered in the class. Code should take Design Principles and Design Patterns into consideration, the ones in covered in the class.
- **Answer**:

1. **Singleton Pattern**: The **Singleton Pattern** is a design pattern that restricts the instantiation of a class to a single instance. We are using it for managing the Firebase Services.

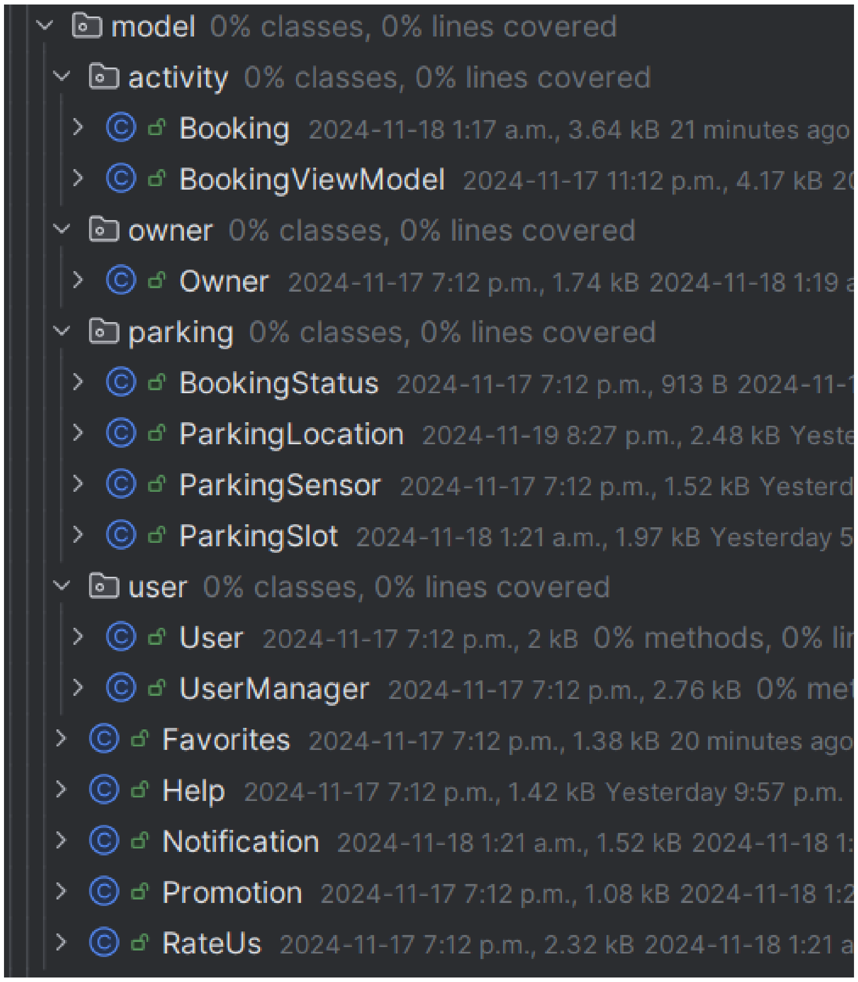


```
kotlin+java
  ca.tech.sense.it.smart.indoor.parking.system 0% classes,
    booking 0% classes, 0% lines covered
    currency 0% classes, 0% lines covered
    firebase 0% classes, 0% lines covered
      FirebaseAuthSingleton 2024-11-17 7:13 p.m., 574 B Mon
      FirebaseDatabaseSingleton 2024-11-17 7:13 p.m., 665 B
      FirebaseRealtimeDatabaseSingleton 2024-11-17 7:13 p
      FirestoreSingleton 2024-11-17 7:13 p.m., 433 B 0% met
    launcherActivity 0% classes, 0% lines covered
    logic 0% classes, 0% lines covered
    Manager 0% classes, 0% lines covered
    model 0% classes, 0% lines covered
      activity 0% classes, 0% lines covered
```

```java
import com.google.firebase.firestore.FirebaseFirestore;

6 usages
public class FirestoreSingleton {

    // Private constructor to prevent instantiation
    no usages
    private FirestoreSingleton() {}

    // This method provides the instance of FirebaseFirestore
    public static FirebaseFirestore getInstance() { return Firebase
}
```

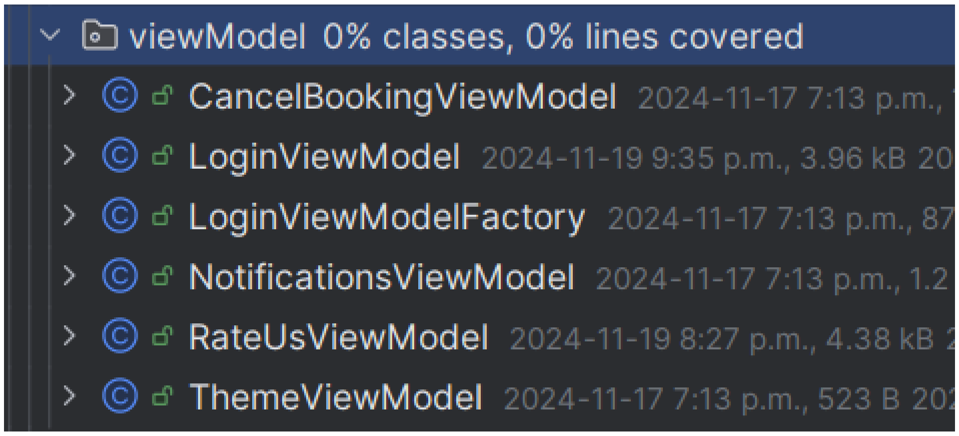
2. **MVVM (Model-View-View Model)** is a design pattern that helps to separate concerns in an application. It divides the code into three components: You u can view these classes, in different packages as per their usage.

- **Model**: Represents the data and business logic.



```
model 0% classes, 0% lines covered
  activity 0% classes, 0% lines covered
    Booking 2024-11-18 1:17 a.m., 3.64 kB 21 minutes ago
    BookingViewModel 2024-11-17 11:12 p.m., 4.17 kB 20
  owner 0% classes, 0% lines covered
    Owner 2024-11-17 7:12 p.m., 1.74 kB 2024-11-18 1:19 a
  parking 0% classes, 0% lines covered
    BookingStatus 2024-11-17 7:12 p.m., 913 B 2024-11-1
    ParkingLocation 2024-11-19 8:27 p.m., 2.48 kB Yeste
    ParkingSensor 2024-11-17 7:12 p.m., 1.52 kB Yesterd
    ParkingSlot 2024-11-18 1:21 a.m., 1.97 kB Yesterday 5
  user 0% classes, 0% lines covered
    User 2024-11-17 7:12 p.m., 2 kB 0% methods, 0% lin
    UserManager 2024-11-17 7:12 p.m., 2.76 kB 0% met
  Favorites 2024-11-17 7:12 p.m., 1.38 kB 20 minutes ago
  Help 2024-11-17 7:12 p.m., 1.42 kB Yesterday 9:57 p.m.
  Notification 2024-11-18 1:21 a.m., 1.52 kB 2024-11-18 1:
  Promotion 2024-11-17 7:12 p.m., 1.08 kB 2024-11-18 1:2
  RateUs 2024-11-17 7:12 p.m., 2.32 kB 2024-11-18 1:21 a
```

- **View Model**: Acts as a mediator between the Model and View (Fragment). It fetches data from the Model and updates the View.



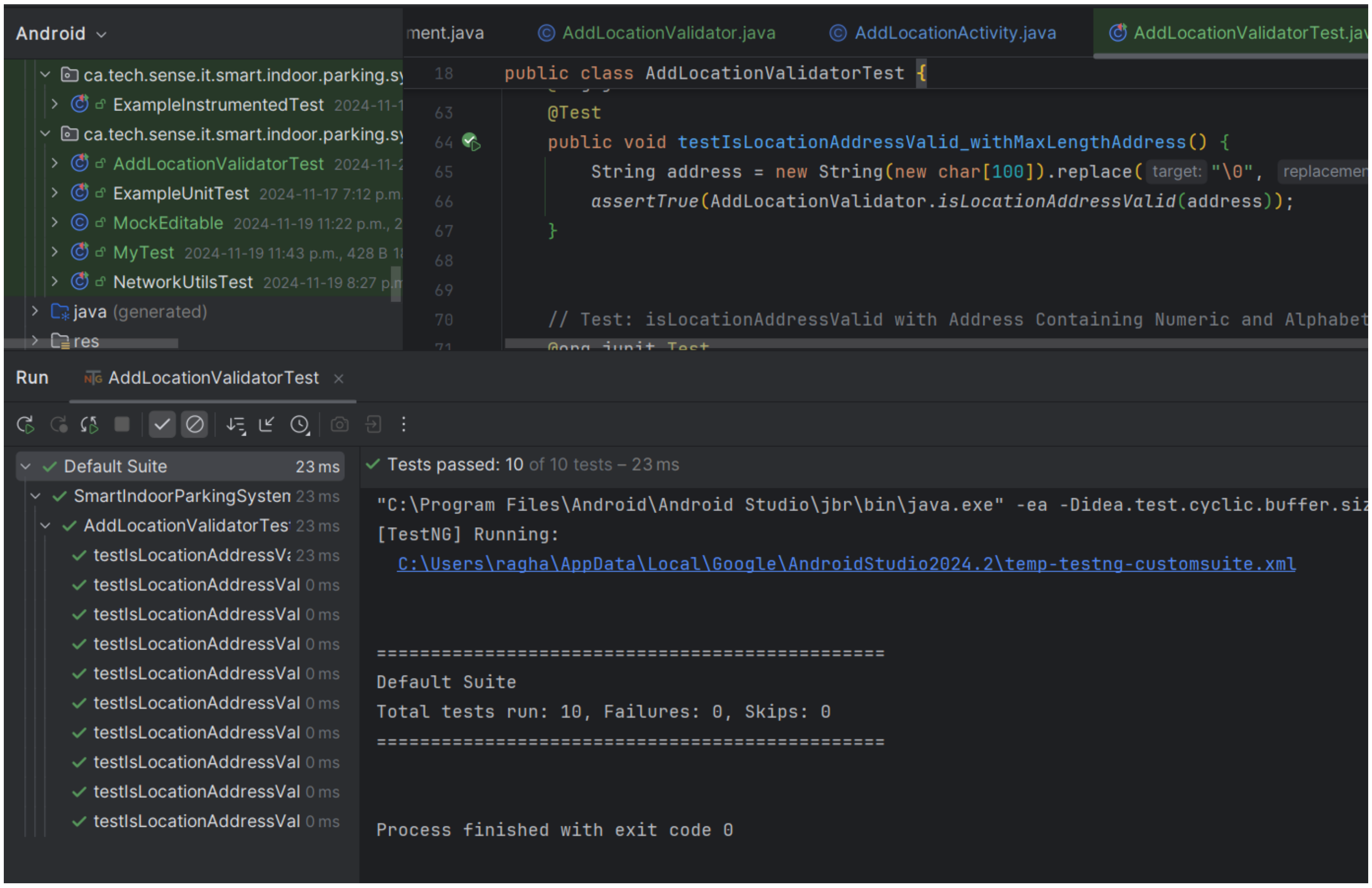
```
viewModel 0% classes, 0% lines covered
  CancelBookingViewModel 2024-11-17 7:13 p.m., 1
  LoginViewModel 2024-11-19 9:35 p.m., 3.96 kB 20
  LoginViewModelFactory 2024-11-17 7:13 p.m., 87
  NotificationsViewModel 2024-11-17 7:13 p.m., 1.2
  RateUsViewModel 2024-11-19 8:27 p.m., 4.38 kB 2
  ThemeViewModel 2024-11-17 7:13 p.m., 523 B 202
```

18. Code Progress Since Deliverable 3

- **Payment Gateway Integration:** A payment gateway was integrated, enabling users to securely process payments for parking bookings. This streamlines the booking process, providing a seamless payment experience within the app.
- **Updated Feedback Screen:** The feedback screen has been redesigned with a more user-friendly and appropriate UI. This update ensures that users can easily provide feedback on their parking experience, helping us improve the app further.
- **Signup for Parking Space Owners:** A new signup flow has been created specifically for parking space owners. This allows owners to register their parking spots on the platform, enabling them to list their available spaces for rent and manage their listings.
- **Network Monitoring and No Network Handling:** We added network monitoring functionality to the app, ensuring that users are notified when they have no network connection. This helps improve the user experience by preventing issues when the app attempts to load data without a network connection.
- **Booking Bottom Sheet:** We implemented a `BookingBottomSheetDialogFragment` to display booking details. This feature provides a seamless user experience for confirming parking bookings.

19. Unit Test Cases

- **Requirement**: Write unit tests for one of your Java classes. Write a minimum of 10 test cases.
- Answer - Wrote Unit testing for AddLocationValidator.java



```
Android
  ca.tech.sense.it.smart.indoor.parking.sy
    ExampleInstrumentedTest 2024-11-1
  ca.tech.sense.it.smart.indoor.parking.sy
    AddLocationValidatorTest 2024-11-2
    ExampleUnitTest 2024-11-17 7:12 p.m.
    MockEditable 2024-11-19 11:22 p.m., 2
    MyTest 2024-11-19 11:43 p.m., 428 B 1
    NetworkUtilsTest 2024-11-19 8:27 p.m
  java (generated)
  res
```

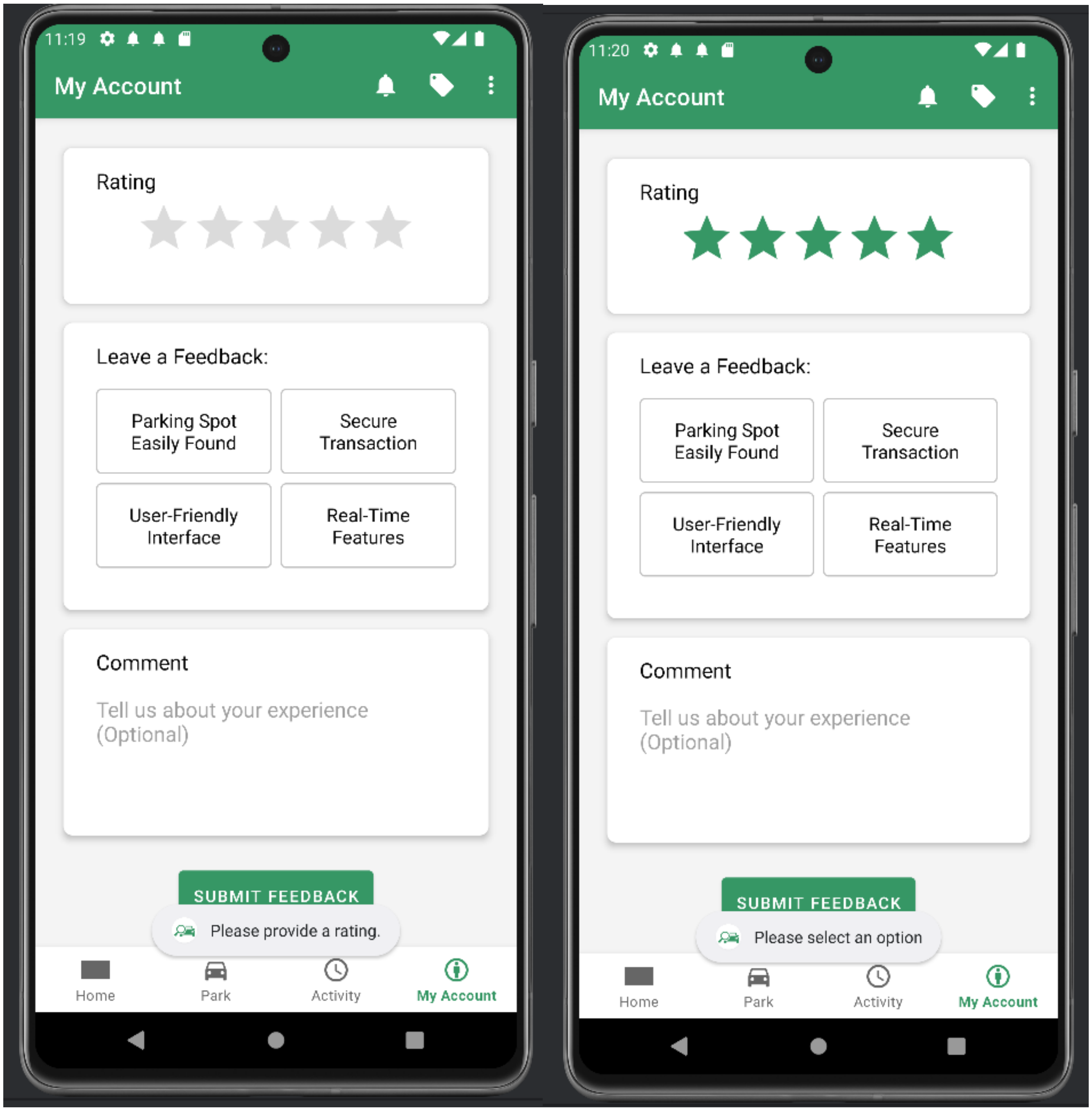
```
ment.java   AddLocationValidator.java   AddLocationActivity.java   AddLocationValidatorTest.jav

public class AddLocationValidatorTest {
    @Test
    public void testIsLocationAddressValid_withMaxLengthAddress() {
        String address = new String(new char[100]).replace( target: "\0",  replacemen
        assertTrue(AddLocationValidator.isLocationAddressValid(address));
    }


    // Test: isLocationAddressValid with Address Containing Numeric and Alphabet
    @org.junit.Test
```

```
Run   AddLocationValidatorTest

Default Suite                          23 ms
  SmartIndoorParkingSysten             23 ms
    AddLocationValidatorTes            23 ms
      testIsLocationAddressVa          23 ms
      testIsLocationAddressVal         0 ms
      testIsLocationAddressVal         0 ms
      testIsLocationAddressVal         0 ms
      testIsLocationAddressVal         0 ms
      testIsLocationAddressVal         0 ms
      testIsLocationAddressVal         0 ms
      testIsLocationAddressVal         0 ms
      testIsLocationAddressVal         0 ms
      testIsLocationAddressVal         0 ms
```

```
Tests passed: 10 of 10 tests – 23 ms

"C:\Program Files\Android\Android Studio\jbr\bin\java.exe" -ea -Didea.test.cyclic.buffer.siz
[TestNG] Running:
  C:\Users\ragha\AppData\Local\Google\AndroidStudio2024.2\temp-testng-customsuite.xml


===============================================
Default Suite
Total tests run: 10, Failures: 0, Skips: 0
===============================================


Process finished with exit code 0
```

20. Customer Feedback Screen

- **Requirement**: Implement the feedback screen with validation and progress bar. Verify you are submitting device model and stored into DB.
- **Answer**: Developed the feedback screen with fields for name, email, rating, and comment. Implemented validation and a progress bar during form submission.

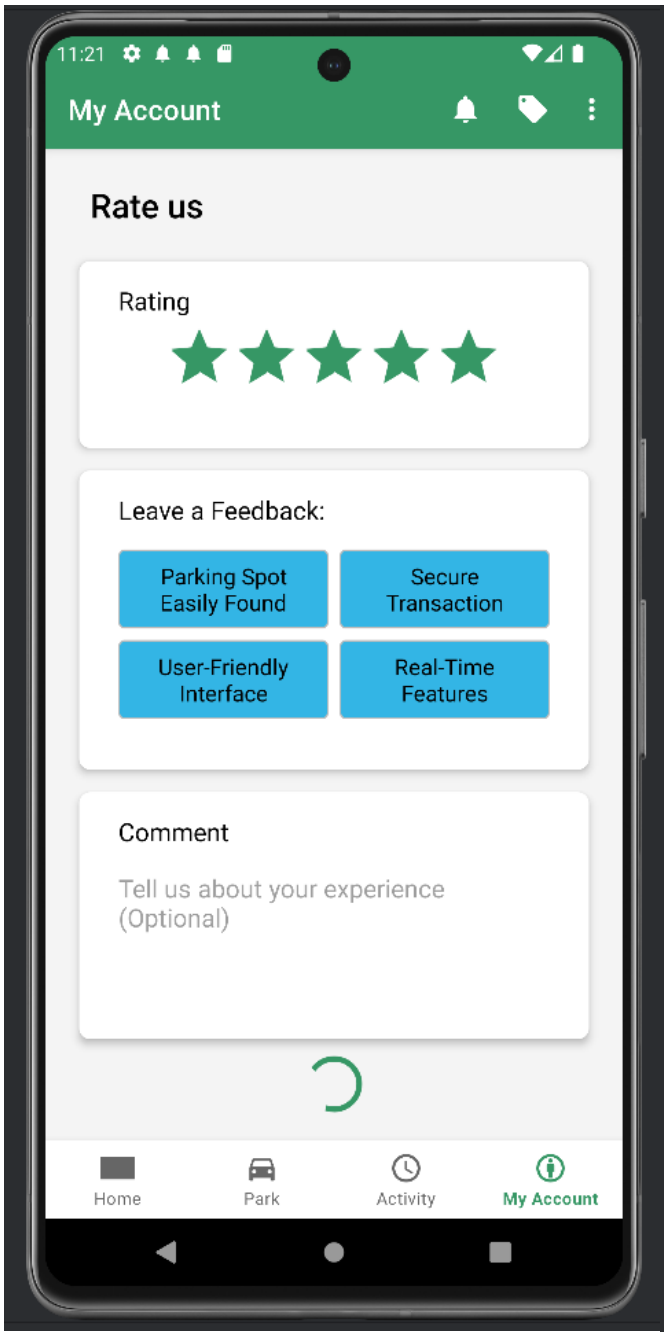


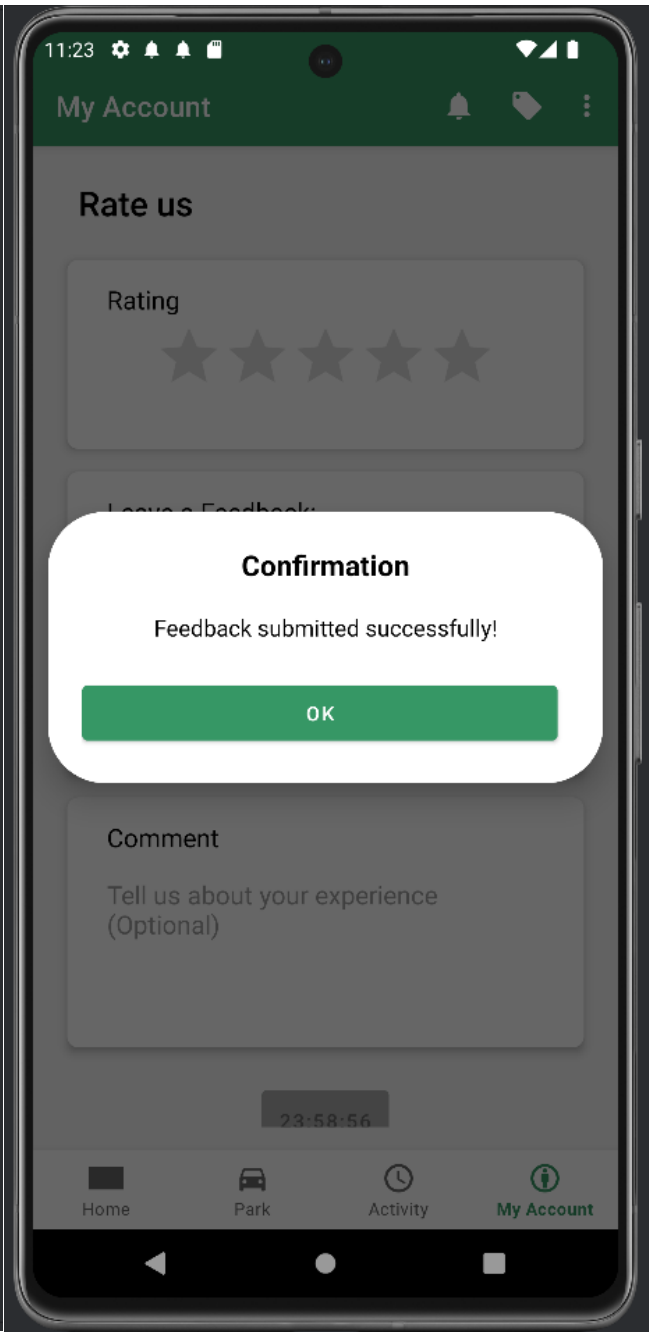
Please provide a rating.

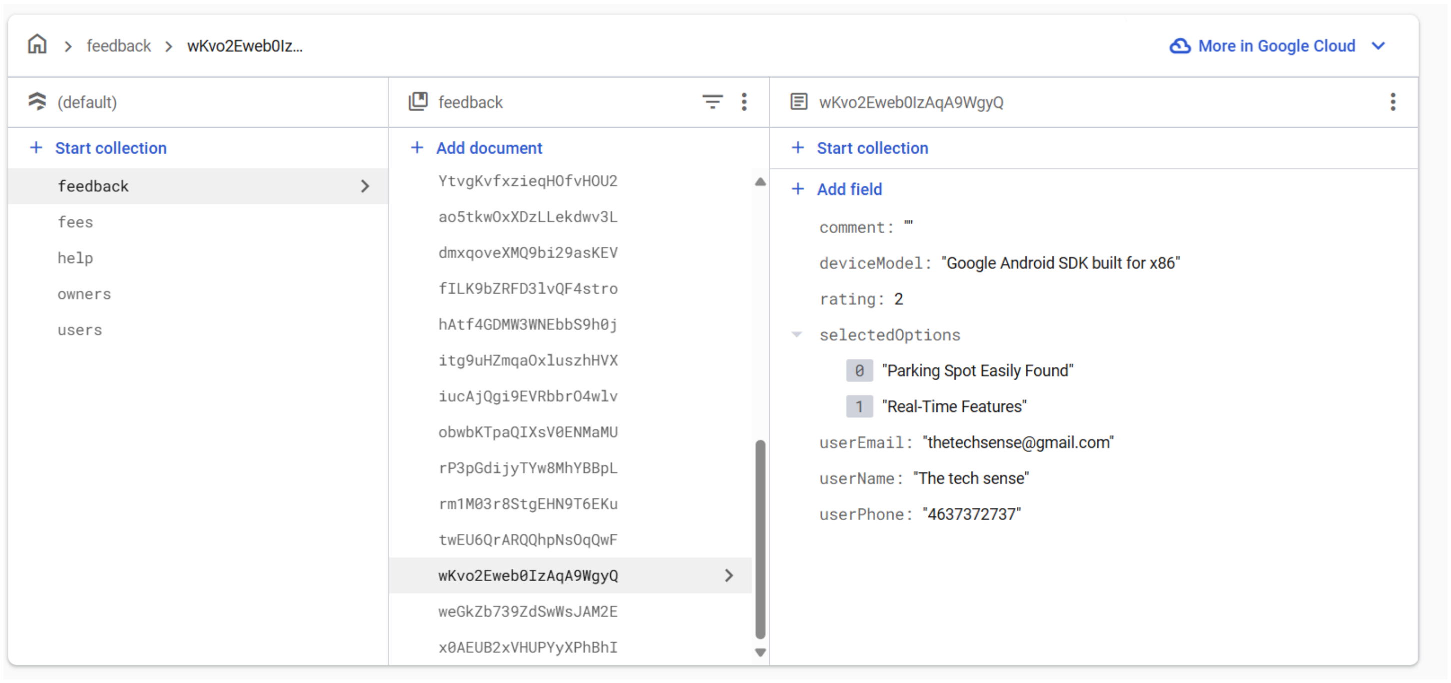
Please select an option

21. Feedback Screenshots

- **Requirement**: Add into pdf file screenshot showing the progress bar while the form is getting submitted, add into pdf file screenshot showing the Alert Dialog once the form is submitted successfully, add into pdf file screenshot showing the data stored into the DB. Must have at least 3 different entries.

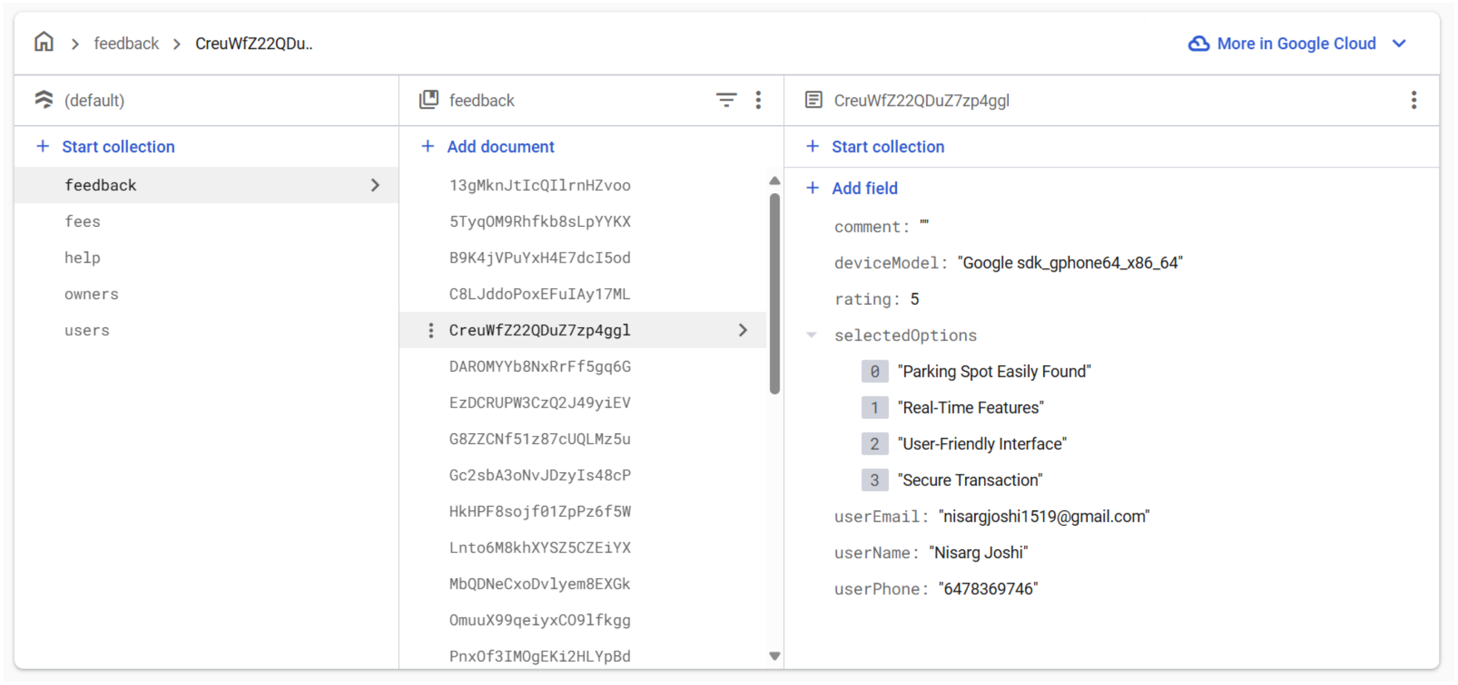






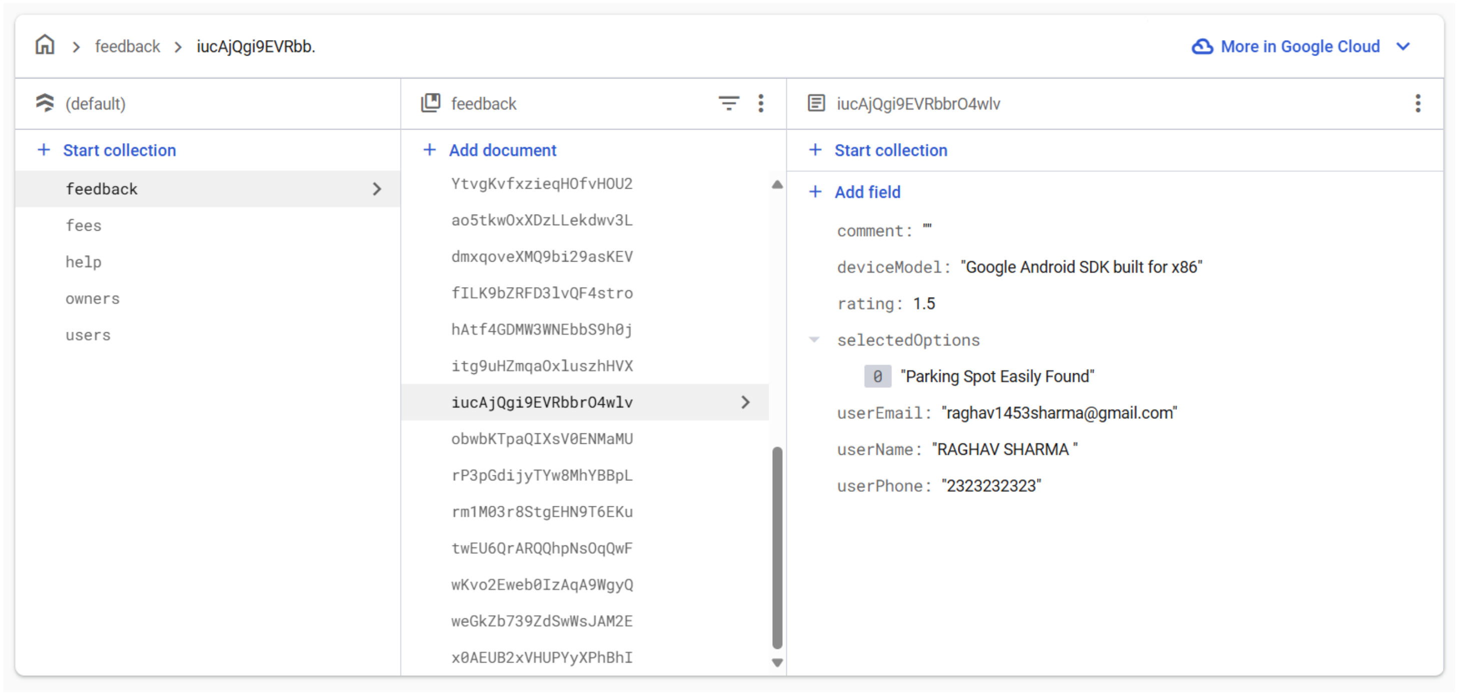
```
YtvgKvfxzieqHOfvHOU2
ao5tkw0xXDzLLekdwv3L
dmxqoveXMQ9bi29asKEV
fILK9bZRFD3lvQF4stro
hAtf4GDMW3WNEbbS9h0j
itg9uHZmqaOxluszhHVX
iucAjQgi9EVRbbrO4wlv
obwbKTpaQIXsV0ENMaMU
rP3pGdijyTYw8MhYBBpL
rm1M03r8StgEHN9T6EKu
twEU6QrARQQhpNsOqQwF
wKvo2Eweb0IzAqA9WgyQ
weGkZb739ZdSwWsJAM2E
x0AEUB2xVHUPYyXPhBhI
```

```
comment: ""
deviceModel: "Google Android SDK built for x86"
rating: 2
selectedOptions
    0 "Parking Spot Easily Found"
    1 "Real-Time Features"
userEmail: "thetechsense@gmail.com"
userName: "The tech sense"
userPhone: "4637372737"
```



```
13gMknJtIcQIlrnHZvoo
5TyqOM9Rhfkb8sLpYYKX
B9K4jVPuYxH4E7dcI5od
C8LJddoPoxEFuIAy17ML
CreuWfZ22QDuZ7zp4ggl
DAROMYYb8NxRrFf5gq6G
EzDCRUPW3CzQ2J49yiEV
G8ZZCNf51z87cUQLMz5u
Gc2sbA3oNvJDzyIs48cP
HkHPF8sojf01ZpPz6f5W
Lnto6M8khXYSZ5CZEiYX
MbQDNeCxoDvlyem8EXGk
OmuuX99qeiyxCO9lfkgg
PnxOf3IMOgEKi2HLYpBd
```

```
comment: ""
deviceModel: "Google sdk_gphone64_x86_64"
rating: 5
selectedOptions
    0 "Parking Spot Easily Found"
    1 "Real-Time Features"
    2 "User-Friendly Interface"
    3 "Secure Transaction"
userEmail: "nisargjoshi1519@gmail.com"
userName: "Nisarg Joshi"
userPhone: "6478369746"
```



feedback > iucAjQgi9EVRbb.

(default)

feedback

iucAjQgi9EVRbbrO4wlv

Collections: feedback, fees, help, owners, users

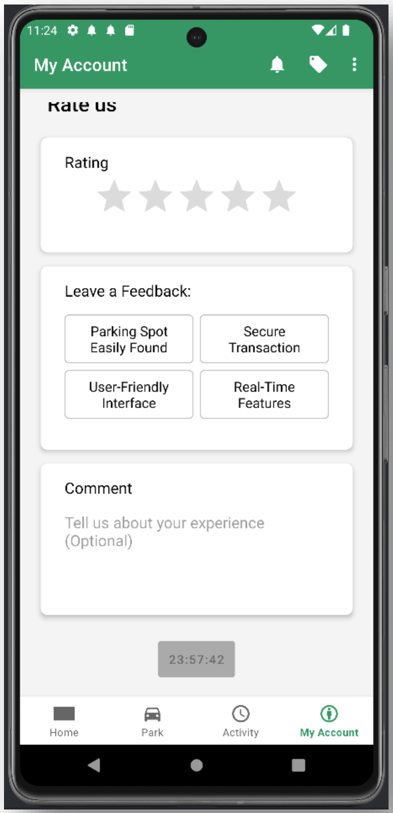
Documents in feedback:

```
YtvgKvfxzieqHOfvHOU2
ao5tkwOxXDzLLekdwv3L
dmxqoveXMQ9bi29asKEV
fILK9bZRFD31vQF4stro
hAtf4GDMW3WNEbbS9h0j
itg9uHZmqaOxluszhHVX
iucAjQgi9EVRbbrO4wlv
obwbKTpaQIXsV0ENMaMU
rP3pGdijyTYw8MhYBBpL
rm1M03r8StgEHN9T6EKu
twEU6QrARQQhpNsOqQwF
wKvo2Eweb0IzAqA9WgyQ
weGkZb739ZdSwWsJAM2E
x0AEUB2xVHUPYyXPhBhI
```

```
comment: ""
deviceModel: "Google Android SDK built for x86"
rating: 1.5
selectedOptions
    0 "Parking Spot Easily Found"
userEmail: "raghav1453sharma@gmail.com"
userName: "RAGHAV SHARMA "
userPhone: "2323232323"
```

22. Submit Button Disable and Timer Feature

- **Requirement:** Once the user submits the form successfully, gray out the submit button, and display a timer showing how many hours and minutes remaining when the user can submit another feedback. Describe in the pdf file on how you satisfied this requirement.



My Account

Rate us

Rating

Leave a Feedback:

Parking Spot Easily Found | Secure Transaction | User-Friendly Interface | Real-Time Features

Comment

Tell us about your experience (Optional)

23:57:42

Progress Bar and Delay

- showProgressBar () in RateUsFragment shows the progress bar and hides the submit button during feedback submission.
- A 5-second delay is simulated using CountDownTimer.

Confirmation AlertDialog

- After feedback submission, a confirmation dialog is shown using DialogUtil.showConfirmationDialog().

Device Model Submission

- Device model and manufacturer are extracted using Build.MODEL and Build.MANUFACTURER and submitted with the feedback.

Clearing User Input and Restricting Submissions

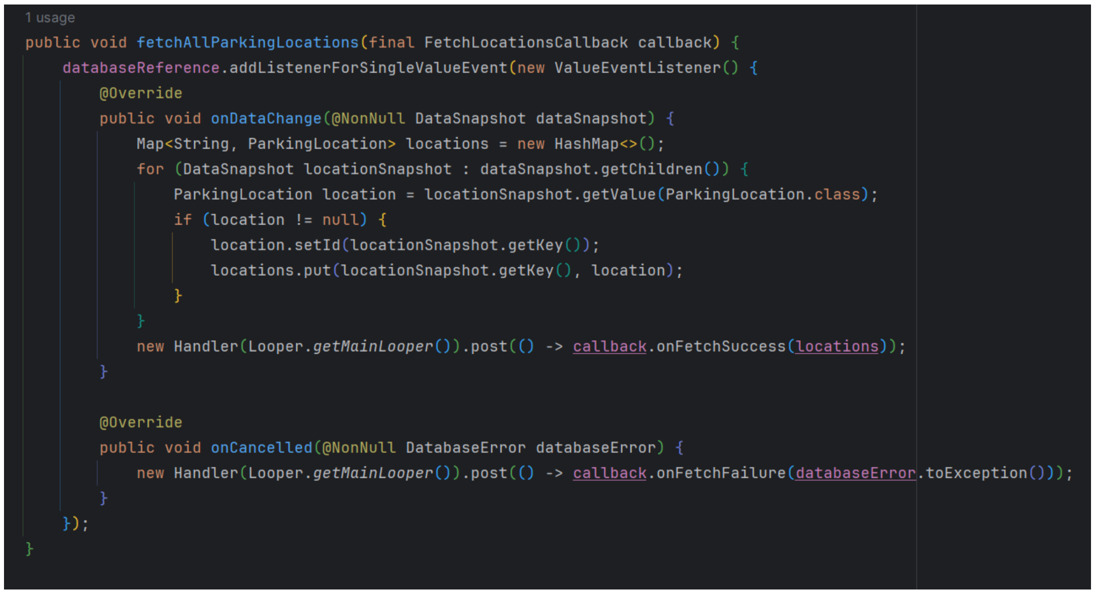
- User input is cleared after submission, and the submit button is disabled for 24 hours using CountDownTimer.

Graying Out Submit Button and Displaying Timer

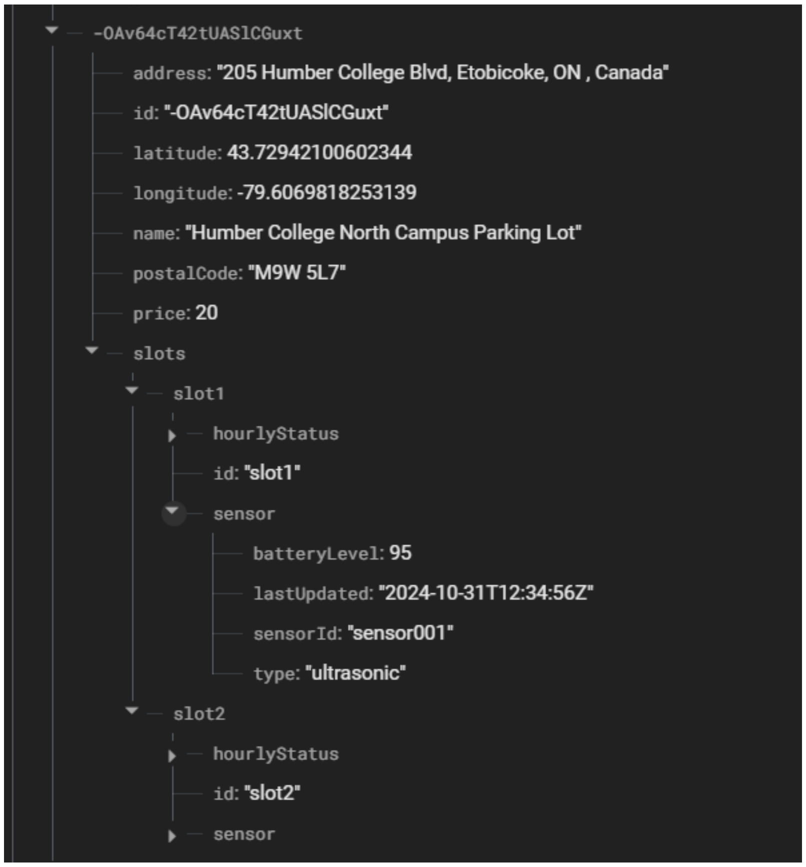
- Submit button is grayed out and a countdown timer is displayed if the user attempts to submit within 24 hours of their last feedback.

23. Sensor Data Fetched and Updated from the Database

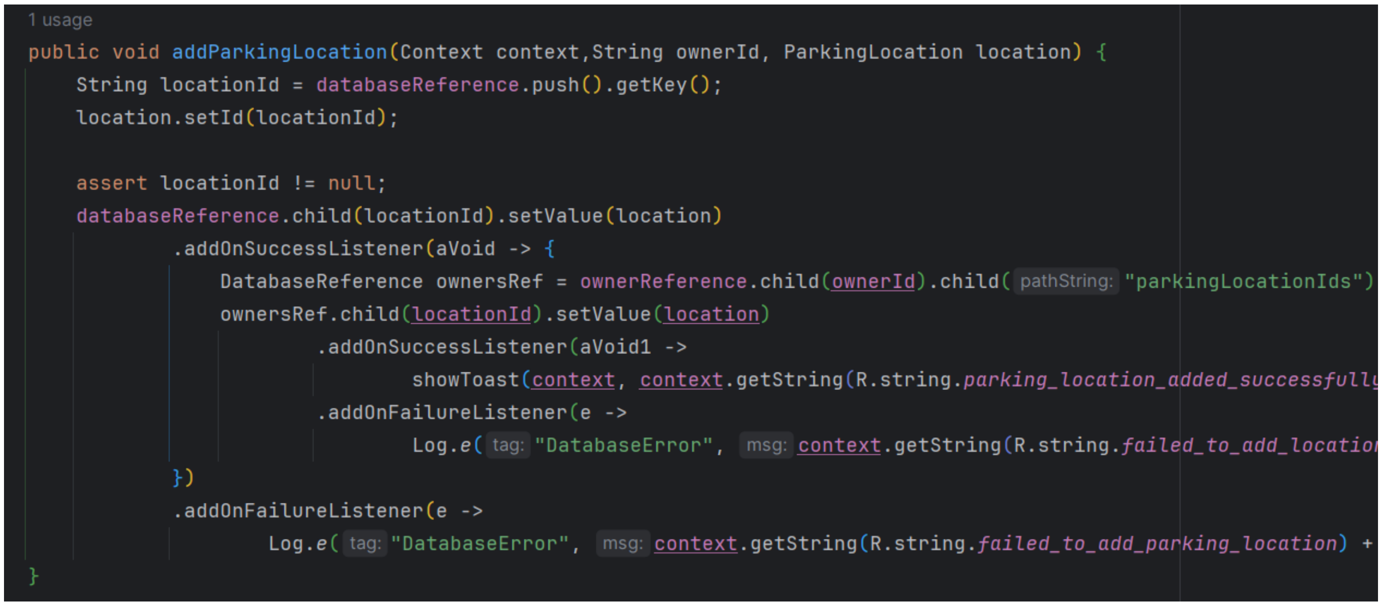
- **Requirement**: Take a screenshot showing all sensor data fetched and updated from the DB.



```
1 usage
public void fetchAllParkingLocations(final FetchLocationsCallback callback) {
    databaseReference.addListenerForSingleValueEvent(new ValueEventListener() {
        @Override
        public void onDataChange(@NonNull DataSnapshot dataSnapshot) {
            Map<String, ParkingLocation> locations = new HashMap<>();
            for (DataSnapshot locationSnapshot : dataSnapshot.getChildren()) {
                ParkingLocation location = locationSnapshot.getValue(ParkingLocation.class);
                if (location != null) {
                    location.setId(locationSnapshot.getKey());
                    locations.put(locationSnapshot.getKey(), location);
                }
            }
            new Handler(Looper.getMainLooper()).post(() -> callback.onFetchSuccess(locations));
        }

        @Override
        public void onCancelled(@NonNull DatabaseError databaseError) {
            new Handler(Looper.getMainLooper()).post(() -> callback.onFetchFailure(databaseError.toException()));
        }
    });
}
```



```
-OAv64cT42tUASlCGuxt
    address: "205 Humber College Blvd, Etobicoke, ON , Canada"
    id: "-OAv64cT42tUASlCGuxt"
    latitude: 43.72942100602344
    longitude: -79.6069818253139
    name: "Humber College North Campus Parking Lot"
    postalCode: "M9W 5L7"
    price: 20
    slots
        slot1
            hourlyStatus
            id: "slot1"
            sensor
                batteryLevel: 95
                lastUpdated: "2024-10-31T12:34:56Z"
                sensorId: "sensor001"
                type: "ultrasonic"
        slot2
            hourlyStatus
            id: "slot2"
            sensor
```



```
1 usage
public void addParkingLocation(Context context,String ownerId, ParkingLocation location) {
    String locationId = databaseReference.push().getKey();
    location.setId(locationId);

    assert locationId != null;
    databaseReference.child(locationId).setValue(location)
            .addOnSuccessListener(aVoid -> {
                DatabaseReference ownersRef = ownerReference.child(ownerId).child( pathString: "parkingLocationIds")
                ownersRef.child(locationId).setValue(location)
                        .addOnSuccessListener(aVoid1 ->
                                showToast(context, context.getString(R.string.parking_location_added_successfully
                        .addOnFailureListener(e ->
                                Log.e( tag: "DatabaseError",  msg: context.getString(R.string.failed_to_add_locatio
            })
            .addOnFailureListener(e ->
                    Log.e( tag: "DatabaseError",  msg: context.getString(R.string.failed_to_add_parking_location) +
}
```

24. Implementing Core Functionality: At Least Two Features

Feature 1: Enhanced Booking System

- **Booking Confirmation Screen:** Allows users to review and confirm their booking details. Once confirmed, the booking status is updated in Firestore, and the parking spot's availability is adjusted.
- **Booking History:** Users can view and manage past bookings, providing them with a complete overview of their reservations.

Why It Was Important: The booking system is a critical feature for the app, as it directly impacts the user experience by enabling seamless parking spot reservations. Enhancing this system allows for more efficient booking management, reducing the likelihood of booking errors, and improving user satisfaction by providing a smooth and reliable process.

How It Was Implemented:

- The **Booking Confirmation Screen** was implemented, allowing users to review their booking details before confirming. Once confirmed, the booking status is updated in Firestore, and availability for the parking spot is adjusted.
- Added **booking history functionality**, allowing users to view and manage past bookings.

Feature 2: Payment Gateway Integration

Stripe Payment Integration: Users can securely pay for parking via Stripe after confirming their booking.

Payment Success Handler: Once payment is processed successfully, the booking status is updated in Firestore.

Why It Was Important:

Integrating a payment gateway is essential to complete the booking process. By allowing users to securely pay for parking, this feature ensures that the app can be fully functional for both users and owners, creating a smooth financial transaction experience.

How It Was Implemented:

- Integrated **Stripe** payment gateway for secure credit card transactions. The user is prompted to enter payment details after confirming a booking.
- Implemented a **payment success handler** to update the booking status in Firestore once payment was successfully processed.

Feature 3: Owner Interface for Managing Parking Spots

Owner Dashboard: Allows owners to add their parking locations, spots and sensors directly in the app.

Firestore Integration: Parking spot details, including location, price, and availability, are stored in Firestore and updated dynamically whenever the owner modifies the spot's details.

Why It Was Important:

The owner interface is crucial because it allows parking space owners to list, update, and manage their parking spots directly through the app. This empowers owners to have control over their inventory, pricing, and availability, ensuring that the app remains dynamic and up to date. This functionality also creates an essential interaction between the owners and the platform, ensuring that users can find accurate and current parking spot details.

How It Was Implemented:

- Created an **Owner Dashboard** where parking space owners can add new parking spots, and view their current locations and slots.
- Integrated **Firestore** to store parking spot data, including location (latitude and longitude), price, and availability status, which is dynamically updated whenever the owner changes the spot's details.