

Deliverable 2

**The Tech Sense**

Smart Indoor Parking System

Group 10

Raghav Sharma  
Kunal Dhiman  
Nisargkumar Pareshbhai Joshi  
Rushi Manojkumar Patel

[N01537255](#)  
[N01540952](#)  
[N01545986](#)  
[N01539144](#)

1. Student's Names and IDs .....	3
2. Members Info and Participation .....	3
3. Project Scope and Goals .....	4
3.1 Completion Criteria .....	4
4. GitHub Repo Link and Strategy.....	5
4.1 GitHub Strategy.....	5
5. Screenshot of Hardware Professor Invitation .....	6
6. Monday Stories and Tasks .....	6
6.1. Sprint Dashboard .....	6
6.2 Sprint-1 .....	8
7. Screenshot of Menu Bar .....	8
8. Explanation of How DoD Criteria Were Met .....	8
9. Coding Work Progress Since Deliverable 1 .....	10
New Features Added.....	10
Bugs Fixed .....	10
11. Cloud Database.....	10
11.1. Core Functionalities .....	10
11.2. Data Flow in the Application .....	11
12. Daily Stand-ups Record.....	12
12.1 Day 1: Oct 1, 2024 .....	12
12.2. Day 2: Oct 6, 2024.....	13
12.3 Day 3: Oct 8, 2024 .....	13
13. Business Model Canvas .....	14
13.1. Customer Segments .....	14
13.2. Value Propositions .....	14
13.3. Channels .....	15
13.5. Revenue Streams.....	15
13.6. Key Activities .....	15
13.7. Key Resources .....	15
13. 8. Key Partnerships .....	15
13.9. Cost Structure.....	15
14. Design Principles .....	16

## 1. Student's Names and IDs

Name	Student Id	GitHub Id	Signature	Effort
<b>Raghav Sharma</b>	<b>N01537255</b>	<a href="#">RaghavSharma7255</a>	RS	100
<b>Kunal Dhiman</b>	<b>N01540952</b>	<a href="#">KunalDhiman0952</a>	KD	100
<b>Nisargkumar Pareshbhai Joshi</b>	<b>N01545986</b>	<a href="#">NisargJosh9856</a>	NJ	100
<b>Rushi Manojkumar Patel</b>	<b>N01539144</b>	<a href="#">RushiPatel9144</a>	RP	100

---

## 2. Members Info and Participation

- **Raghav Sharma:** Database Setup, adding menu item and their fragments, creating sign up page and checking code for meeting the design principles – 100%.
  - **Nisarg Joshi:** Worked in adding functionality to the app by adding images and runtime permissions. – 100%.
  - **Rushi Patel:** Created the login page, managed flow of screens after splash. Added all the necessary code for the required task. – 100%.
  - **Kunal Dhiman:** Worked in adding functionality of the fragments and modified the UI of the app for both portrait and landscape. – 100%.
-

## 3. Project Scope and Goals

### Scope:

The scope of the **Smart Indoor Parking System** focuses on developing a robust and user-friendly parking management application. The software components include:

1. **Real-time Parking Spot Availability:** Implementing an IoT-driven system to provide users with real-time updates on parking spot availability within an indoor parking facility.
2. **Mobile Application:** A mobile app that allows users to view available spots, reserve them, log entry/exit times, and complete payments seamlessly.
3. **Backend Services:** Integration with a backend service (Firebase) to manage user accounts, parking data, and transaction histories.
4. **Payment System:** Implementing a secure and efficient payment system that allows users to pay for parking directly through the mobile app.

### Goals:

The primary goals of the **Smart Indoor Parking System** are to:

- Provide users with real-time parking availability updates to reduce parking search times.
- Ensure seamless interaction between the mobile app and backend services, using Firebase for data storage and synchronization.
- Implement a fully functioning payment system within the app, allowing for smooth and secure financial transactions.
- Validate functionality and user experience through unit, integration, and user acceptance testing with at least 20 real users.
- Launch the mobile application on the Google Play Store by the end of the project.

---

### 3.1 Completion Criteria

The project will be considered complete when:

- All functional requirements are fully implemented, including:
  - Real-time parking availability updates.
  - Logging user entry and exit times.
  - A fully functioning payment system.
  - User-friendly navigation and intuitive app interface.
- The mobile app successfully interacts with backend services, ensuring accurate data flow between the app and Firebase.

- Feedback from user testing (with 20 real users) indicates that the app is intuitive and meets expectations for parking management.
  - The application is approved by the instructor.
  - Basic documentation, including usage instructions and API details, is provided on GitHub.
- 

## 4. GitHub Repo Link and Strategy

### GitHub Repository Link

Here is the GitHub repository link for the **Smart Indoor Parking System** project:

[SmartIndoorParkingSystem - GitHub Repo Link!](#)

### 4.1 GitHub Strategy

We currently not following everything listed below, but we are trying our best to implement this strategy.

#### *Main Branches:*

- **master/main:** The production-ready branch containing stable, fully tested code.
- **develop:** The main development branch where all feature branches are merged after review and testing.

#### *Name Branches:*

- Each team member works on a personal branch named after them, created from **develop**.

#### *Pull Requests (PRs):*

- Upon completing a feature, a PR is created to merge the feature branch into **develop**.
- Code is reviewed by another team member before merging, typically by the group leader or any reviewer.

#### *Commit Guidelines:*

- Commits follow a clear naming convention, ensuring clarity and traceability.

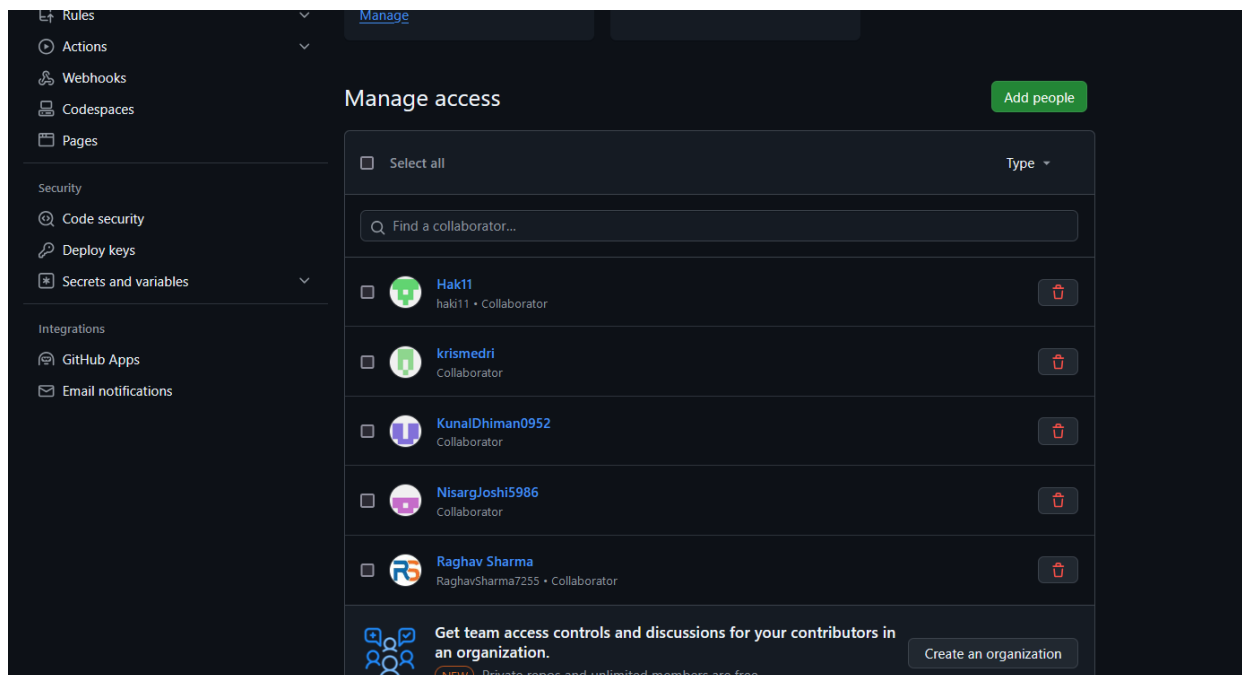
#### *Issue Tracking:*

- GitHub Issues are used for tracking tasks, bugs, and enhancements, with each issue linked to the relevant PR for transparency.

### Team Collaboration:

- Regular reviews and testing are conducted to maintain stability in **develop**, with team members assisting each other as needed.

## 5. Screenshot of Hardware Professor Invitation



## 6. Monday Stories and Tasks

Click on this link to join our Monday board using your Humber Email-

[https://humber605941.monday.com/users/sign\\_up?invitationId=44422400140197700000&inviter\\_id=66695764](https://humber605941.monday.com/users/sign_up?invitationId=44422400140197700000&inviter_id=66695764)

The stories and tasks are available in the Main table. For the first sprint, we completed tasks from each story based on priority and the requirements for Deliverable 2. Some tasks remain unassigned and will be allocated during the Sprint 2 planning meeting, where sprint tasks will be selected from backlog/stories and assigned. We'll also be adding more stories and tasks in the future.

### 6.1. Sprint Dashboard

## The Tech Sense – Smart Indoor Parking System – Group 10 – Deliverable 2

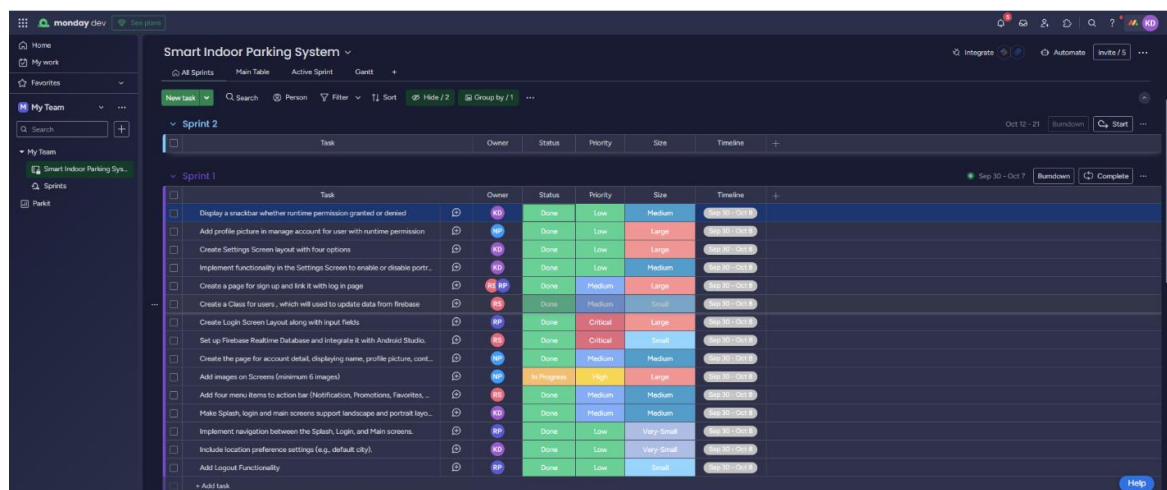
Smart Indoor Parking System							Integrate	Automate	Invite / 5	...
Main Table	New task	Search	Person	Filter	Sort	Hide	Group by	...		
> User Authentication		9 Tasks		Status	Priority	Size				
> User Profile and Account Details		9 Tasks		Status	Priority	Size				
> App Navigation and Screens		7 Tasks		Status	Priority	Size				
> Parking Spot Finder		4 Tasks		Status	Priority	Size				
> Booking a Parking Spot		4 Tasks		Status	Priority	Size				
> App Settings		5 Tasks		Status	Priority	Size				
> Payment System		5 Tasks		Status	Priority	Size				

User Authentication							
	Task		Owner	Status	Priority	Size	Sprint
<input type="checkbox"/>	Create a page for sign up and link it with log in page	+	RS RP	Done	Medium	Large	Sprint 1
<input type="checkbox"/>	Create a Class for users , which will used to update data from firebase	+	RS	Done	Medium	Small	Sprint 1
<input type="checkbox"/>	Create Login Screen Layout along with input fields	+	RP	Done	Critical	Large	Sprint 1
<input type="checkbox"/>	Set up Firebase Realtime Database and integrate it with Android Studio.	+	RS	Done	Critical	Small	Sprint 1
<input type="checkbox"/>	Implement input validation for user details (name, email, password).	+			Missing		
<input type="checkbox"/>	Add error messages and success confirmation.	+			Missing		
<input type="checkbox"/>	Implement password encryption for secure login.	+			Missing		
<input type="checkbox"/>	Integrate third-party authentication (Google sign-in)	+			Missing		
<input type="checkbox"/>	Handle invalid login attempts with error messages.	+			Missing		
<input type="checkbox"/>	+ Add task						

You can view the other stories task from the below link, if u were unable to join Monday board.

<https://view.monday.com/7522718638-d148e04f069dd77ecde0c35c46f85eed?r=use1>

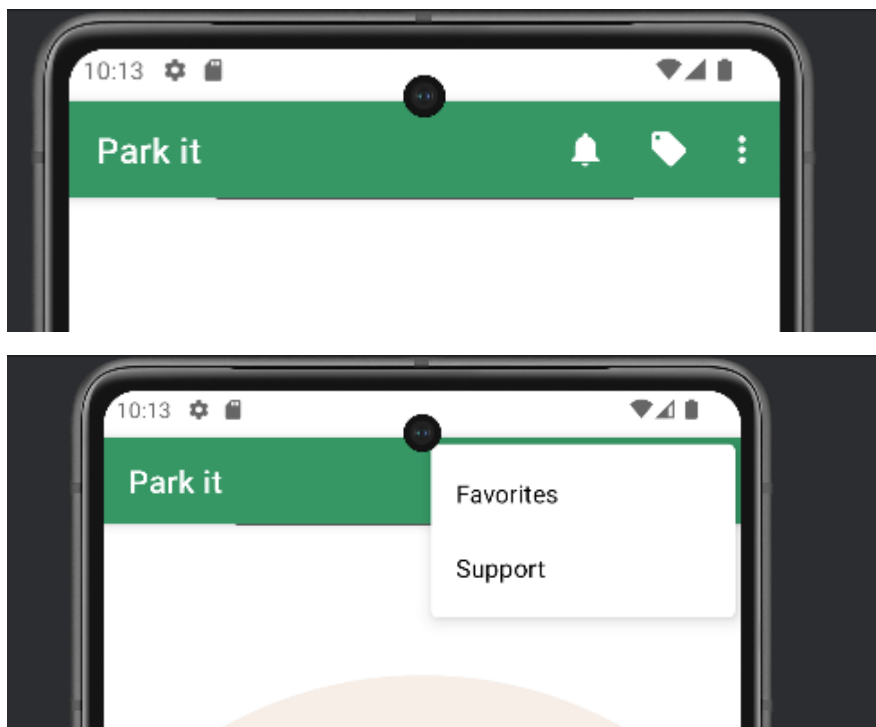
## 6.2 Sprint-1



The screenshot shows a Monday.com project board for the 'Smart Indoor Parking System'. The board is organized into two sprints: Sprint 1 and Sprint 2. Each task is represented by a card with a title, owner, status, priority, size, and timeline. The tasks are color-coded by status: green for 'Done', yellow for 'In Progress', and red for 'Not Started'. The timeline for each task is shown as a bar chart.

Sprint	Task	Owner	Status	Priority	Size	Timeline
Sprint 1	Display a trackbar whether runtime permission granted or denied	JP	Done	Low	Medium	Oct 10 - Oct 10
	Add profile picture in manage account for user with runtime permission	JP	Done	Low	Large	Oct 10 - Oct 10
	Create Settings Screen layout with four options	JP	Done	Low	Large	Oct 10 - Oct 10
	Implement functionality in the Settings Screen to enable or disable port...	JP	Done	Low	Medium	Oct 10 - Oct 10
	Create a page for sign up and link it with log in page	JP	Done	Medium	Large	Oct 10 - Oct 10
	Create a Class for users, which will used to update data from firebase	JP	Done	Medium	Small	Oct 10 - Oct 10
	Create Login Screen Layout along with input fields	JP	Done	Critical	Large	Oct 10 - Oct 10
	Set up Firebase Realtime Database and integrate it with Android Studio.	JP	Done	Critical	Small	Oct 10 - Oct 10
	Create the page for account detail, displaying name, profile picture, coat...	JP	Done	Medium	Medium	Oct 10 - Oct 10
	Add images on Screens (minimum 6 images)	JP	In Progress	High	Large	Oct 10 - Oct 10
Sprint 2	Add four menu items to action bar (Notification, Promotions, Favorites, ...)	JP	Done	Medium	Medium	Oct 10 - Oct 10
	Make Splash, login and main screens support landscape and portrait layo...	JP	Done	Medium	Medium	Oct 10 - Oct 10
	Implement navigation between the Splash, Login, and Main screens.	JP	Done	Low	Very Small	Oct 10 - Oct 10
	Include location preference settings (e.g., default city).	JP	Done	Low	Very Small	Oct 10 - Oct 10
	Add Logout Functionality	JP	Done	Low	Small	Oct 10 - Oct 10
	+ Add task					

## 7. Screenshot of Menu Bar



## 8. Explanation of How DoD Criteria Were Met

To mark a task as "Done," we made sure to follow the **Definition of Done (DoD)** criteria closely:



- **Acceptance Criteria Checked:**  
We made sure that each feature worked the way it was supposed to, according to the requirements or user stories.
  - **Coding Tasks Finished:**  
All the coding that needed to be done for each task was completed before we moved on.
  - **Exploratory Testing Done:**  
Our team members tested the app on their own to find any hidden issues that might not have been covered by regular tests. Once that was done, we all agreed that it was good to go.
  - **Regression Tests Passed:**  
We ran tests to make sure new changes didn't break any existing features. These tests were reviewed and passed before marking tasks as done.
  - **Unit Tests Written and Passed:**  
We wrote small tests for individual pieces of code to make sure they worked correctly. If something broke, we fixed it and tested again.
  - **Code Reviews Completed:**  
After writing code, a teammate reviewed it to check for mistakes, and then it was merged into the main project once everyone was happy with it.
  - **No Critical Bugs Left:**  
Any major bugs were either fixed or agreed upon by the product owner (in this case, our professor), so there were no deal-breaking issues left.
  - **Story Accepted by Product Owner:**  
Once we finished everything for a task, our professor (or whoever was in charge) reviewed it and accepted the work.
  - **Regression Tests Passed Again:**  
We ran final checks to make sure the new features didn't mess up the old ones. If something broke, we fixed it.
  - **Smoke Tests Run (if needed):**  
We ran quick tests to make sure the app still worked after adding new features.
  - **Checked for Memory Leaks:**  
We used tools to make sure the app wasn't using up more memory than it should, which could slow things down.
  - **Automated Unit Tests Checked:**  
We set up automated tests that checked our code every time we made changes, so we knew if something broke right away.
-

## 9. Coding Work Progress Since Deliverable 1

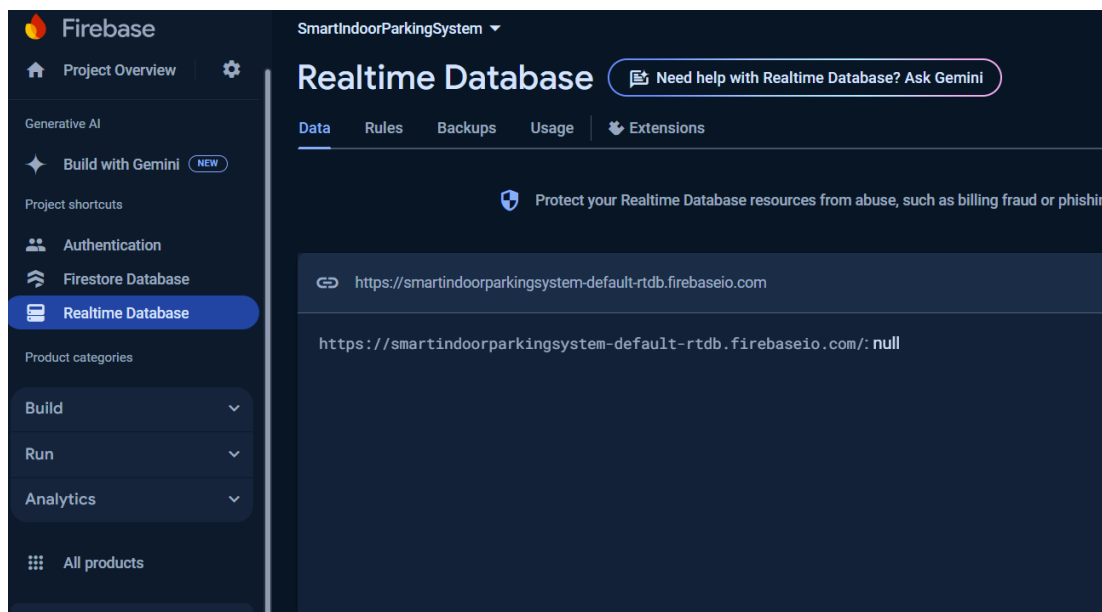
### New Features Added:

Runtime permission for gallery access, setup the firebase database, implemented login and signup functionality, also sign in without userID. Added images to background. Organized the code in more clear way and added functionality to menu bar and settings fragment.

### Bugs Fixed:

Resolved various inconsistencies due to difference in themes, fonts, dependencies, etc. Removed depreciated code from the application and replaced it with up-to-date code. We moved back to JDK 17 and Gradle version 8.7 due to compatibility issues with Firebase implementation. Aligned all the layout files in a proper manner and gave uniform IDs to objects along with proper and clear naming conventions.

## 11. Cloud Database



### 11.1. Core Functionalities

#### A. Real-Time Data Management

- **Parking Spot Availability:**
  - Firestore will manage the availability status of parking spots, allowing users to see which spots are free in real-time. Changes made by any user (e.g., a reservation or cancellation) are reflected immediately for all users.

## B. User Profiles and Reservations

- **User Information Storage:**
  - Users will have unique profiles stored in Firestore, containing personal information and reservation history.
- **Reservation System:**
  - When a user reserves a parking spot, the reservation details (user ID, spot ID, start time, end time) will be stored. This allows the system to keep track of currently occupied spots and user reservations.

## C. Payment Processing

- **Transaction Logging:**
  - Each payment transaction will be logged in Firestore with details such as amount, user ID, spot ID, and payment status (pending, completed).

## D. Entry and Exit Logging

- **User Activity Tracking:**
  - The application will log each user's entry and exit times to calculate the parking duration. This will help with billing and analyzing usage patterns.

## E. Real-Time Updates and Notifications

- **Live Updates:**
  - Using Firestore's snapshot listeners, the app will provide real-time updates on parking availability and reservation statuses, enhancing user engagement.

## F. Security and User Authentication

- **Access Control:**
  - Security rules will be implemented in Firestore to ensure that users can only access and modify their own data, thereby maintaining privacy and security.

## G. User Feedback and Improvements

- **Feedback Collection:**
  - User feedback collected after usage can be stored for analysis and to improve the system. This feedback can be linked to specific reservations or user profiles.

# 11.2. Data Flow in the Application

## A. User Registration and Login

- Upon registration, user details are added to the users collection.
- On login, the user profile is fetched from Firestore to display personalized information.

## B. Making a Reservation

- When a user reserves a parking spot:
  1. The app checks the availability of the spot by querying the parkingSpots collection.
  2. If available, the reservation details are updated in the users collection and the parkingSpots collection.
  3. The system logs the reservation in the logs collection.

## C. Processing Payments

- When a payment is made:
  1. The payment details are added to the payments collection.
  2. The payment status is updated accordingly.

## D. Logging Entry and Exit

- Upon entry, the app creates a new document in the logs collection.
- On exit, the app updates the existing log entry with the exit time and calculates the duration.

## E. Gathering Feedback

- After parking, users can submit feedback, which is stored in the feedback collection for analysis and improvements.

# 12. Daily Stand-ups Record

## 12.1 Day 1: Oct 1, 2024

Member	What have we done?	What are we going to do?	Issues faced
KD	Started working on creating the settings screen layout with four options.	Complete the settings screen layout and add functionality for enabling/disabling portrait mode.	No major issues faced.
RS	Set up Firebase Realtime Database and created a basic user class to interact with the database.	Integrate Firebase with Android Studio and link the database to the sign-up page.	No major issues faced.

RP	Worked on the login screen layout .	Continue with Firebase integration and finalize the login screen features.	Struggled with understanding the Firebase SDK.
NP	Started working on the six background images with 3 different resolutions.	Create the manage account layout	Major issues with the sizing of different resolutions.

## 12.2. Day 2: Oct 6, 2024

Member	What have we done?	What are we going to do?	Issues faced
KD	Completed the settings screen layout, added functionality to lock screen menu item,	Start working on UI and start doing the documentation of the deliverable 2.	No major issues encountered.
RS	Linked the sign-up page with login functionality and continued Firebase database setup, added 4 menu items to the menu bar.	Begin testing Firebase data updates .	No major issues encountered.
RP	Completed login screen layout and done the signup.	Focus on implementing logout functionality and testing user data updates with Firebase.	Firebase integration issues, but resolved with debugging.
NP	Created the layout of the Account Fragment, and added the runtime permission for the imagebutton.	Add the remaining background images and make the bussiness canvas model.	Minor UI challenges in the Fragment.

## 12.3 Day 3: Oct 8, 2024

Member	What have we done?	What are we going to do?	Issues faced
KD	Created the documentation and make sure that every condition is satisfied.	Prepare for the presentation.	No significant issues.
RS	Made few changes in the menu bar and did the documentation and made the Gantt chart.	Made the sprint dashboard and prepare for the presentation.	No significant issues.
RP	Completed Firebase user data integration.	Ensure proper functionality of the entire login/logout flow and fix any remaining bugs and prepare for the presentation.	Encountered small bugs in the login flow.
NP	Finalized adding images and created the business canvas model.	Conduct final UI/UX improvements and ensure consistency in all screen layouts and prepare for the presentation.	Faced issues in adding shared preferences.

## 13. Business Model Canvas

### 13.1. Customer Segments

- Vehicle owners in urban areas, businesses with parking facilities, and city planners.
- They think about the hassle of finding parking, the safety of their vehicles, and the convenience of automated systems.
- They see crowded parking lots, limited parking spaces, and inefficient parking management.
- They feel frustrated with traditional parking methods and are anxious about vehicle safety.
- They look for efficient, secure, and convenient parking solutions.

### 13.2. Value Propositions

- The system offers a hassle-free, time-saving, and secure parking experience using smart technology.
- Customers use the system to save time, reduce stress, and ensure the safety of their vehicles.

### 13.3. Channels

- Through digital marketing, partnerships with local businesses, and a user-friendly mobile app.
- These channels effectively reach tech-savvy individuals and businesses looking for efficient parking solutions.
- Yes, as it targets the right audience and provides a seamless user experience.

### 13.4. Customer Relationships

- Through a mobile app that guides them from finding and reserving parking spots to payment processing and customer support.

### 13.5. Revenue Streams

- Through fees charged per parking session, subscription models for frequent users, and partnerships with businesses.

### 13.6. Key Activities

- Maintaining the software platform, managing partnerships with parking lot owners, and updating the system's database with real-time availability.

### 13.7. Key Resources

- Advanced software algorithms, agreements with property owners, and a customer service team.

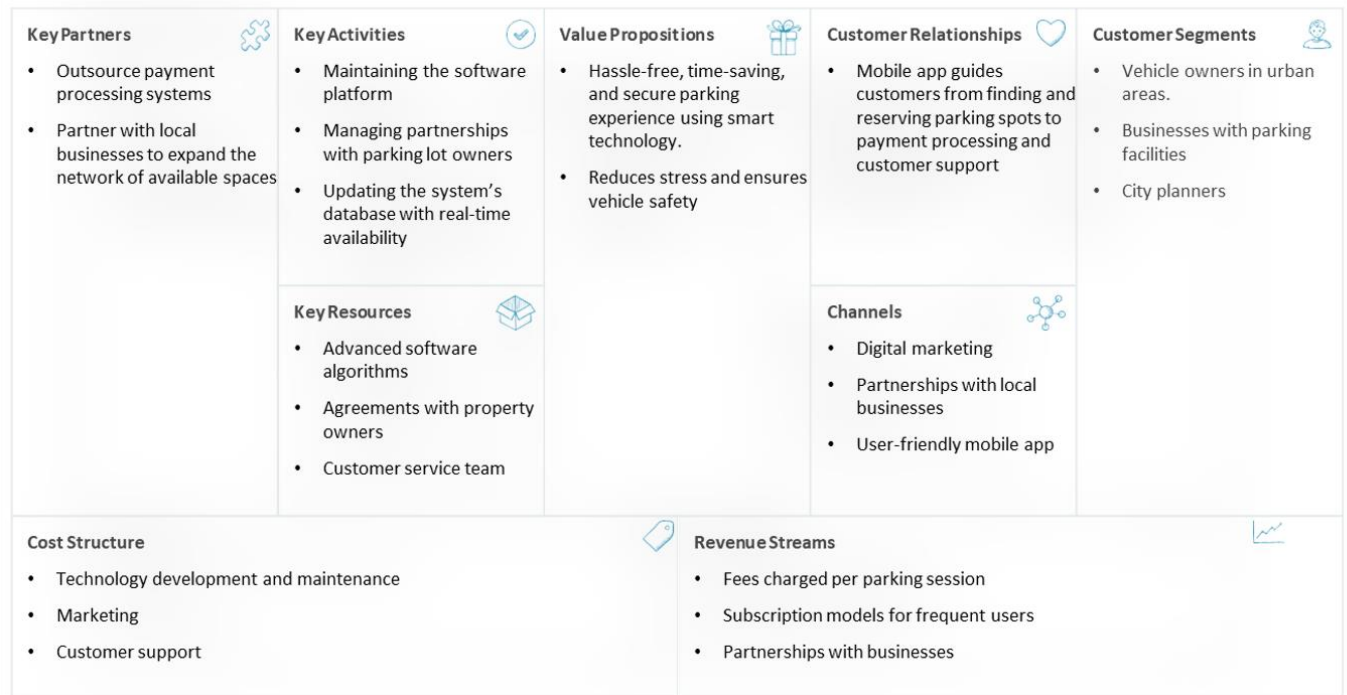
### 13.8. Key Partnerships

- Outsource payment processing systems and partner with local businesses to expand the network of available spaces.

### 13.9. Cost Structure

- Technology development and maintenance, marketing, and customer support.
- These costs are directly linked to the number of users and transactions processed through the system.

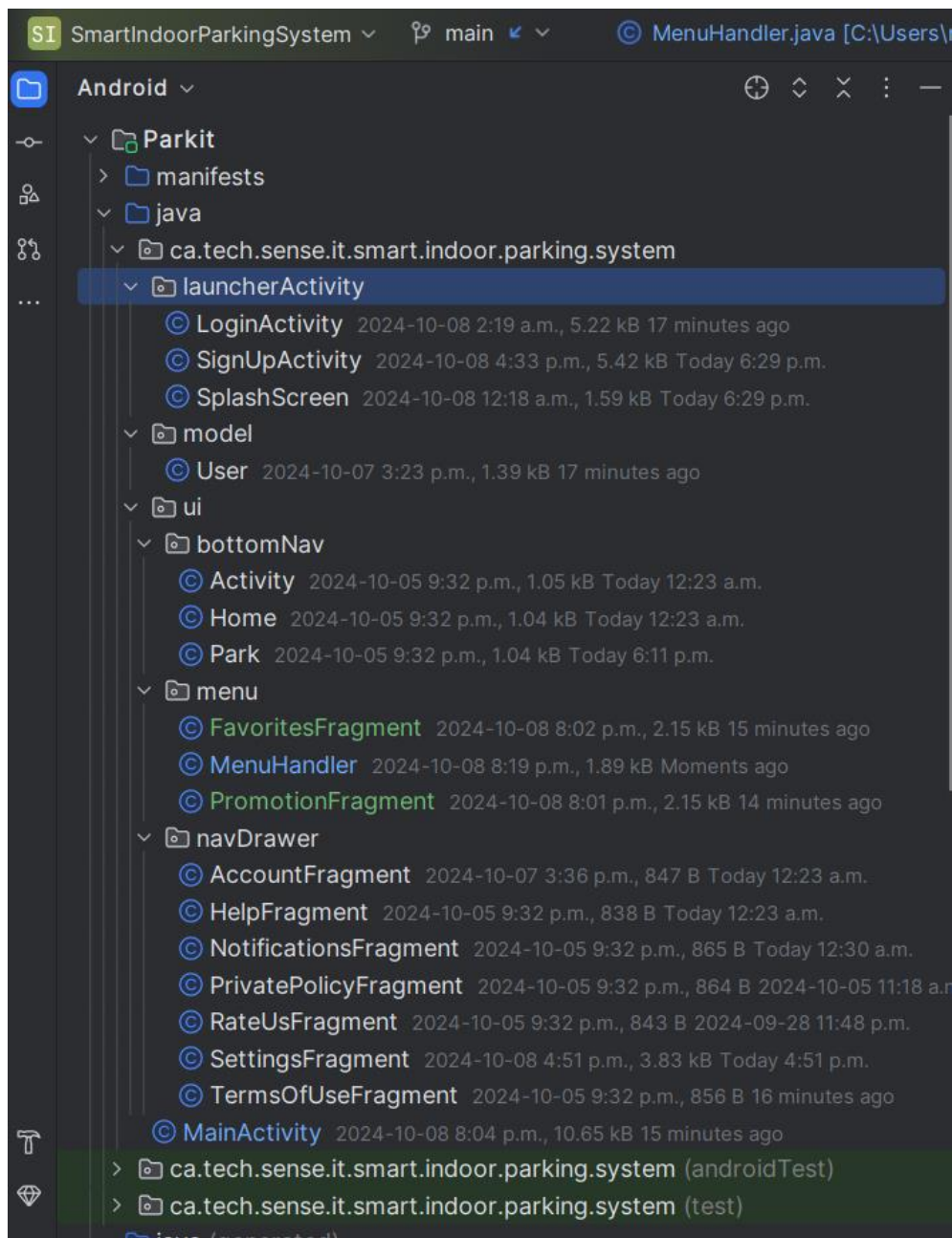
## Business Model Canvas



## 14. Design Principles

We used the modularity design principle in our smart parking app by organizing code into packages based on different functionalities. This separation of concerns allows each package to handle specific tasks (e.g., UI components, data models, and menu handling), making the codebase easier to maintain and enhancing scalability. This approach not only improves collaboration among team members but also simplifies debugging and testing.





**Here We used DRY Design Principle. Created methods for not writing code again and again in Java class.**

```
private void openGallery() { 2 usages
    Intent intent = new Intent(Intent.ACTION_PICK);
    intent.setType(getString(R.string.image));
    pickImageLauncher.launch(intent);
}

private void saveProfilePictureUri(Uri uri) { 1 usage
    SharedPreferences sharedPreferences = getActivity().getSharedPreferences(PREFS_NAME, Context.MODE_PRIVATE);
    SharedPreferences.Editor editor = sharedPreferences.edit();
    editor.putString(KEY_PROFILE_PICTURE_URI, uri.toString());
    editor.apply();
}

private void loadProfilePictureUri() { 1 usage
    SharedPreferences sharedPreferences = getActivity().getSharedPreferences(PREFS_NAME, Context.MODE_PRIVATE);
    String uriString = sharedPreferences.getString(KEY_PROFILE_PICTURE_URI, defValue: null);
    if (uriString != null) {
        Uri uri = Uri.parse(uriString);
        try (InputStream inputStream = getActivity().getContentResolver().openInputStream(uri)) {
            profilePictureButton.setImageURI(uri);
        } catch (Exception e) {
            profilePictureButton.setImageResource(R.mipmap.ic_launcher);
        }
    }
    else {
        profilePictureButton.setImageResource(R.mipmap.ic_launcher);
    }
}
```