Deliverable 1

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. CENG-322 TEAM PROJECT CONTRACT

- **Please note that if cheating is discovered in a group assignment each member will be charged with a cheating offense regardless of their involvement in the offense. Each member will receive the appropriate sanction based on their individual academic honesty history.**

- **Please ensure that you understand the importance of academic honesty. Each group member is responsible for ensuring academic integrity of all submitted work, not just their own part. Placing your name on a submission indicates that you take responsibility for its content.**

Team Member Names	Signatures	Student ID	Github Id
Project Leader: Raghav Sharma	RS	N01537255	RaghavSharma7255
Kunal Dhiman	KD	N01540952	KunalDhiman0952
Rushi Manojkumar Patel	RP	N01539144	RushiPatel9144
Nisargkumar Pareshbhai Joshi	NJ	N01545986	NisargJoshi5986

- **For further information read Academic Honesty Policy on https://humber.ca/legal-and-risk-management/policies/search-by-students.html.**

- By signing this contract, we acknowledge having read the Humber Academic Honesty Policy as per the link below.

- https://academic-regulations.humber.ca/2018-2019/17.0-ACADEMIC-MISCONDUCT

1.1. Responsibilities of the Project Leader include:

1. Assigning tasks to other team members, including self, in a fair and equitable manner.
2. Ensuring work is completed with accuracy, completeness and timeliness.
3. Planning for task completion to ensure timelines are met.
4. Any other duties as deemed necessary for project completion.

1.2. What we will do if . . .

Scenario	**Accepted initials**	**We agree to do the following**
Team member does not deliver component on time due to severe illness or extreme personal problem	RS RP NJ KD	a) Team absorbs workload temporarily
Team member cannot deliver component on time due to lack of ability	RS RP NJ KD	a) Team reassigns components for one last time (then fires them). b) Team helps member for first time only. c) Team "fires" team members by not permitting his/her name on submission.
Team member does not deliver component on time due to lack of effort	RS RP NJ KD	a) Team absorbs workload only first two times and at third time we will fire. b) Team "fires" team members by not permitting his/her name on submission.

		c) Only Two warnings will be given before firing.
Team members do not attend team meetings or attend Metting without involving mentally.	RS RP NJ KD	a) Team proceeds without him/her and will assign work to the absent member. b) Team doesn't proceed and records team member's absence. c) Team proceeds for that meeting but "fires" member after 3 occurrences.
An unforeseen constraint occurs after the deliverable has been allocated and scheduled (a surprise test or assignment)	RS RP NJ KD	a) Team meets and reschedules deliverable. b) Team will cope with constraint.
Team cannot achieve consensus leaving one member feeling "railroaded", "ignored", or "frustrated" with a decision which affects all parties	RS RP NJ KD	a) Team agrees to abide by majority vote .

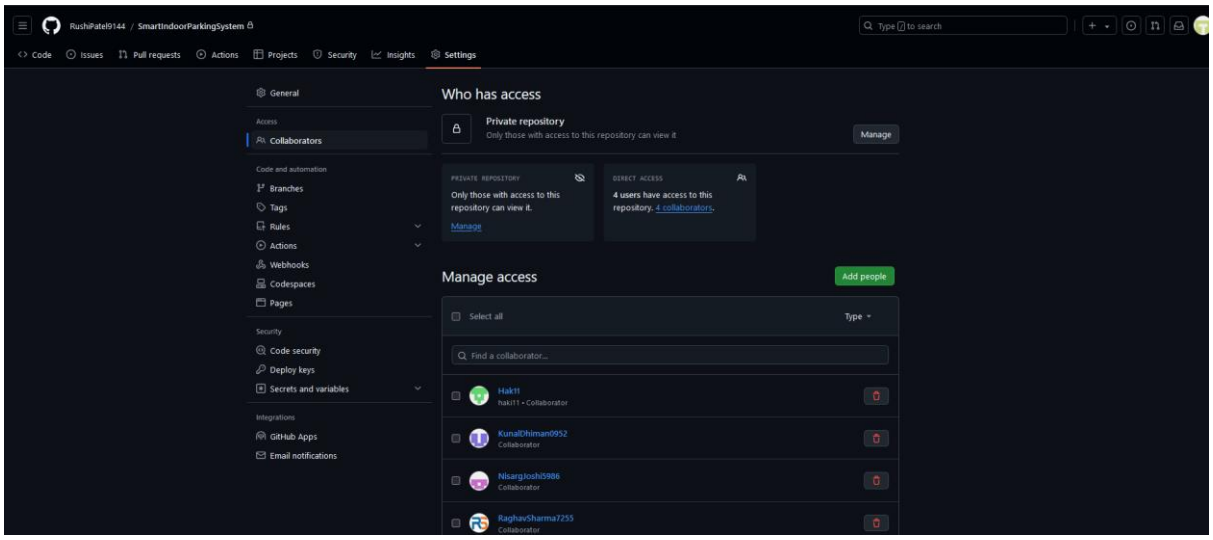
Team members do not share expectations for grade desired	RS RP NJ KD	a) Team will elect one person as "standards-bearer" who has the right to ask that work be redone. b) Team votes on each submission's quality. c) Team will ask for individual marking and will identify sections by author.
Team member behaves in an unprofessional manner by being rude or uncooperative	RS RP NJ KD	a) Team attempts to resolve the issue by airing the problem at team meetings. c) Team agrees to avoid use of all vocabulary inappropriate to the business setting. c) Team fires the team member.
Team member assumes or requests that his/her name be signed to a submission but has not participated in production of the deliverable	RS RP NJ KD	A) Team agrees that this is cheating and is unethical. B) That person's name will not be put in the submission.
There is a dominant team member who is content to make all decisions on the team's behalf leaving some team members feeling like subordinates	RS RP NJ KD	a) Team will actively solicit consensus on all decisions which affect project direction by asking for each member's decision and vote.

rather than equal members		b) Team will express subordination feelings and attempt to resolve issue. c) Majority will decide to check whether the team member is dominant.
Team has a member who refuses to participate in decision making but complains to others that s/he wasn't consulted.	RS RP NJ KD	a) Team forces decision sharing by routinely voting on all issues. b) Team routinely checks with each other about perceived roles. c) Team discusses the matter at team meeting.

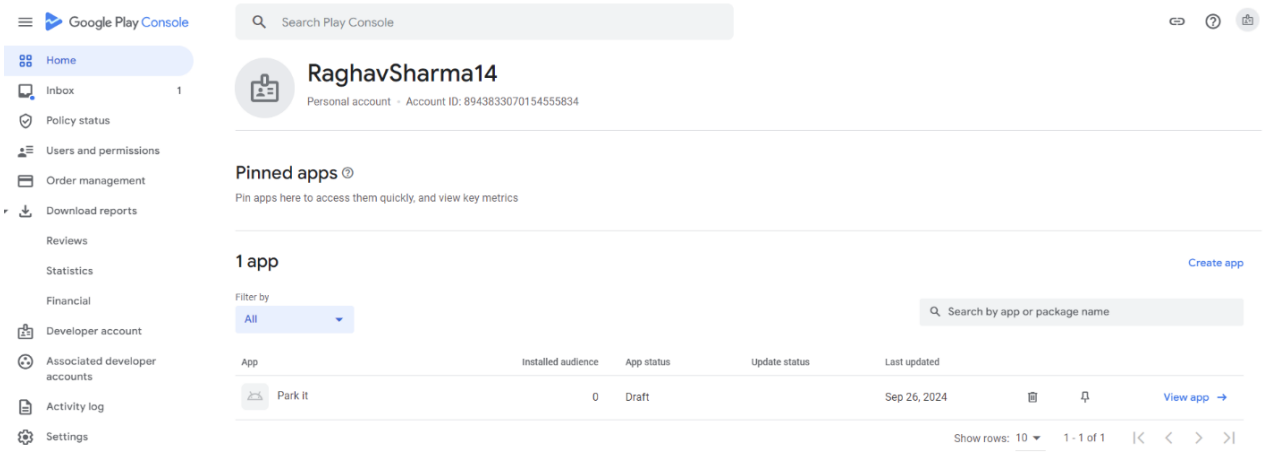
2. Screenshots

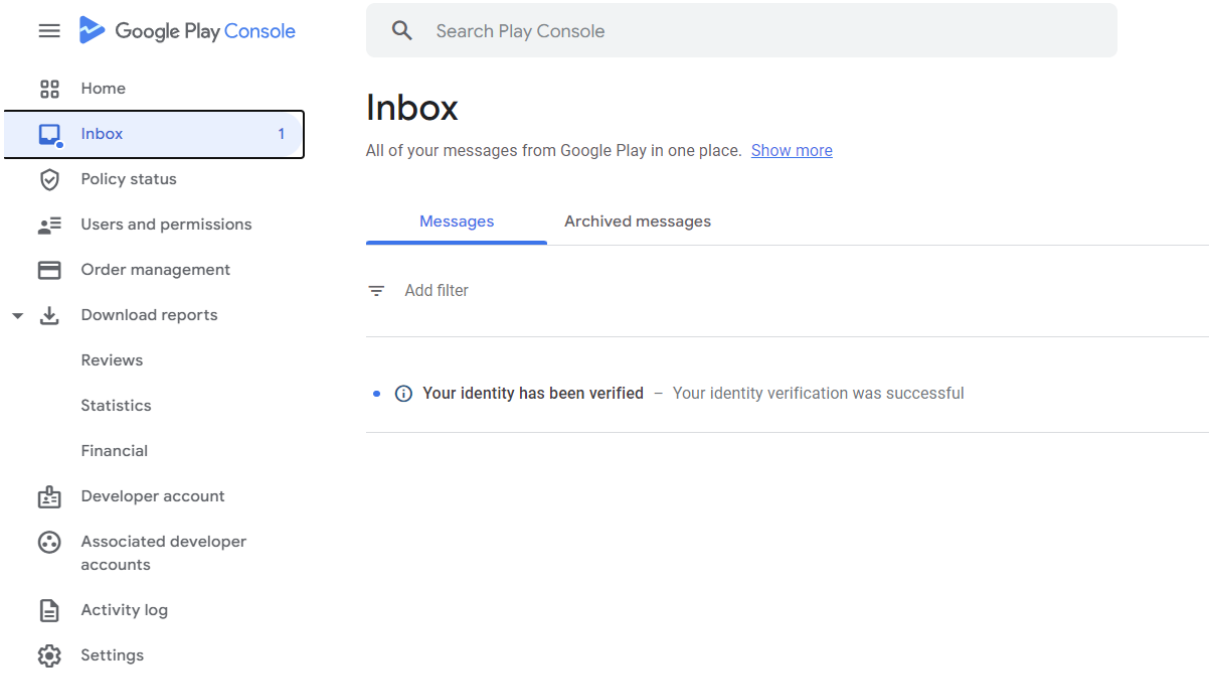
2.1 GitHub Invitation

SmartIndoorParkingSystem



2.2 Google Developer account confirmation:





3. Project Background and Description

3.1. Describe the project goals and final vision.

The goal of the **Smart Indoor Parking System** is to develop an intelligent parking system for indoor environments. The project aims to optimize parking space usage, guide users to available spots, and streamline the overall parking experience with a smart solution. The final vision is a user-friendly mobile application that integrates seamlessly with IoT hardware for real-time parking management and status updates.

3.2. Describe the software aspect and hardware.

Software Aspect: The mobile application is built using Android Studio, using API 29 as the minimum SDK and API 34 as the target. The application includes features like navigation, real-time parking status updates, and cloud-based data storage for parking spots availability. The application supports both English and French, with all UI strings stored in strings.xml.

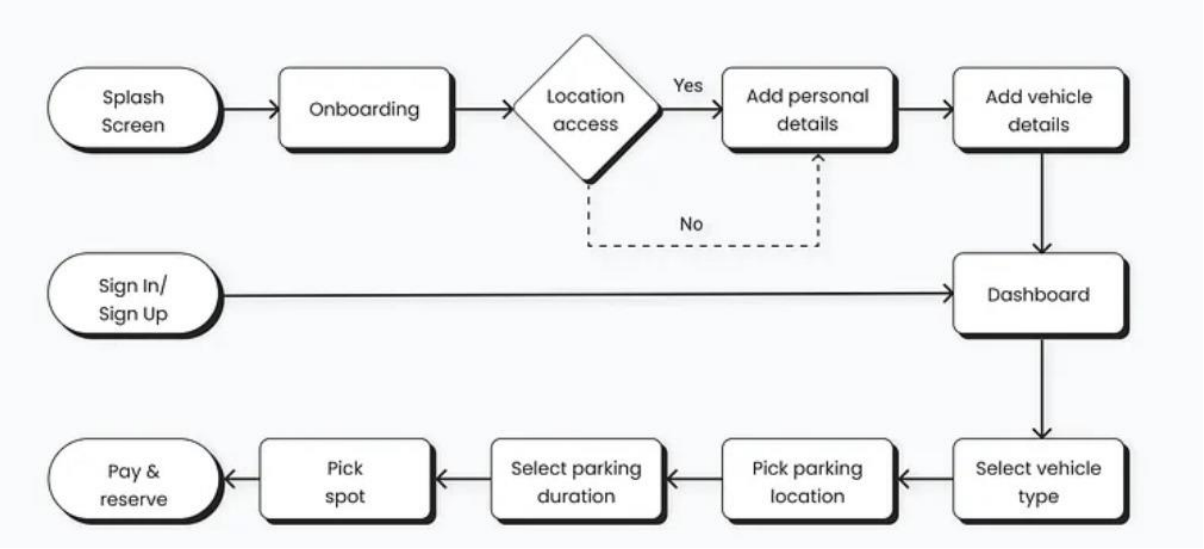
Hardware Aspect: The system will interface with indoor IoT parking sensors installed in parking spots, which will relay real-time availability data to the cloud and subsequently update the app. The hardware component of this project includes a Raspberry Pi that serves as the central controller for the indoor parking system. The Raspberry Pi is connected to four sensors that monitor the availability of parking spaces. These sensors detect the presence of vehicles in each spot and relay the information to the Raspberry Pi, which processes the data and updates it to a cloud-based database. The sensors used include:

- IR Sensors for detecting vehicle presence.
- Environmental Breakout Sensor for temperature, humidity, detection of CO_2.
- Input Keypad with screen to restrict the parking space to authorized users only and verify the user's entrance so that we can log that time as well.
- The Raspberry Pi interfaces with the cloud server to store and manage data, ensuring that users can access real-time updates via the mobile application.

This setup allows the project to monitor parking spots efficiently and relay the status to the app in real time, making the system fully interactive.

3.3. Describe the screen flows.

The application follows a structured flow with intuitive navigation elements to enhance the user experience and we will be following the best practices for ui/ux by following goggle's design guidelines. The sequence of screens is as follows:



- **First-Time User Flow: Account Creation**
  a. Sign-Up Process: When the app is first installed and launched, users will be guided through the account creation process. This includes providing personal details and location information to personalize the parking experience.

b. Post-Registration: Once the account is created, users proceed directly to the home screen.

- **Returning User Flow**
  c. Splash Screen: On subsequent launches, users are greeted with a splash screen displaying the app logo and name, which lasts for 3 seconds.
  d. Sign-In Screen: After the splash screen, users are directed to the Sign-In page. Once logged in, they are automatically taken to the Home screen.
  e. **Dashboard (Home)**: After sign-in, users land on the **Home** fragment where they can start interacting with the app.

3.3.1. Key Fragments and Their Flows

1. **Splash Screen**:
   a. Displays the app logo and name for 3 seconds before transitioning to the sign-up or sign-in process.

2. **Home Fragment with Bottom Navigation**:
   a. After login, the user lands on the Home Fragment. This screen is part of the Bottom Navigation setup, which allows easy access to the following fragments:
      i. Home: Displays an overview of parking lots, availability updates, and any relevant information.
      ii. Park: Takes users to a map where they can search for parking spots, view availability based on time and make reservations.
      iii. Activity: Allows users to review their bookings (past, current, and upcoming) and manage notifications.
      iv. My Account: Provides access to the user's profile, payment options etc.

3. **My Account Fragment with Navigation Drawer**:
   a. Within the My Account fragment, a Navigation Drawer is available for users to access additional settings and options, such as:
      i. Profile Settings: To update account details.
      ii. Payment Methods: Manage and update saved payment options.
      iii. App Support & Help: Access to FAQs, support requests, and terms of service.
      iv. Logout: Functionality for securely logging out of the app.

This flow ensures smooth navigation and accessibility for both new and returning users, making it easy to manage parking bookings, payments, and account settings within the app.

We will describe more about the navigation fragments and features further in this document, **Also, fragments are subject to change as per need or as per feedback.**

3.4. How you Incorporated the feedback provided through the interview.

Based on the feedback, we have decided to create two separate interfaces: one for users and one for parking lot owners. This way, we can avoid direct interaction with the owners.

Owners will be able to register their parking lots, set availability times, determine pricing, and track their earnings directly through the app.

3.5. Show how you plan to satisfy to read/write from the DB hosted on the cloud.

We are planning to use **Firebase Realtime Database** as our cloud-based solution to manage data read and write operations for our project. Firebase provides seamless integration with Android applications and offers real-time data synchronization across all connected clients.

- **Storing Data**: The status of parking spots, detected by the sensors connected to the Raspberry Pi, will be transmitted to Firebase, where it will be stored in real-time. This includes information about whether a parking space is occupied or vacant. Additionally, when a user enters or exits the parking area, they will input a code via the keypad sensor. This entry/exit event will also be logged in Firebase, ensuring that all user actions are recorded.
- **Reading Data: The mobile application will access the Firebase database to retrieve the latest parking availability data and user entry and exit logs.** Each time a user opens the app or navigates to the **Home** or **Maps** fragment, the app will fetch the current data from Firebase, ensuring users see the most up-to-date information. Users can view their entry and exit logs in the **My Account** section of the app, which will display timestamps for when they entered and exited the parking area.
- **Synchronization**: Firebase's real-time capabilities ensure that any changes made by the sensors or through user interactions, such as a parking spot becoming available or a user logging their entry/exit, will instantly be reflected in the app. This synchronization provides a highly responsive user experience, which is critical for the effectiveness of a smart parking system.

By leveraging Firebase's real-time database, we can efficiently manage both parking availability updates and user entry/exit tracking, delivering a comprehensive solution for our smart indoor parking system.

4. Project Scope

This project will follow an agile development approach, allowing for progress and continuous feedback. Key phases include:

1. **Requirement Gathering**:
    - Understanding user needs and defining system requirements through interviews and surveys.
2. **Design and Prototyping**:
    - Creating wireframes and flowcharts for mobile applications and system architecture.
3. **Development**:

   - Implementing the software components, ensuring seamless integration.

4. **Testing**:
   - Conducting unit, integration, and user acceptance testing with 20 real users to validate functionality and performance.
5. **Deployment**:
   - Launching the application on the Google Play Store.

4.1 Completion Criteria:

The project will be considered complete when:

- All functional requirements are met, including real-time parking availability updates, user entry/exit logging, payment system implemented, and user-friendly navigation.
- The mobile application successfully interacts with the backend services, with data accurately transmitted to and from Firebase.
- User testing feedback is positive, indicating the application is intuitive and effective in managing parking and approved by instructor.
- Basic documentation is provided on GitHub as well.

4.2 Testing and Validation:

- Conduct thorough testing of the software components to ensure reliability, robustness, and user satisfaction.
- User acceptance testing (UAT) will be performed to gather feedback and make necessary adjustments before the final release.

5. Integration of software with hardware

The integration of software and hardware components is crucial for the functionality of the Smart Indoor Parking System. The following sensors will be used:

- **Environmental Breakout Sensor (BME-680)**: This sensor will gather environmental data, such as temperature and humidity, which can help in optimizing parking conditions.
- **Qwiic PL-N823 IR Breakout (SPX-15804)**: This infrared sensor will detect vehicle presence in the parking spots, ensuring accurate availability status.

- **SparkFun Qwiic Keypad (12 Button) (COM-15290)**: This keypad will allow authorized users to enter codes, logging their entry and exit times to enhance security and track usage.
- **Zio Qwiic OLED Display (1.5 inch, 128x128) (LCD-15890)**: This display will provide real-time feedback to users regarding parking availability and system status.

The Raspberry Pi serves as the central controller for the parking system. It will collect data from the connected sensors and process this information for transmission to the mobile application.

5.1 Real-Time Data Synchronization:

- The Raspberry Pi will continuously monitor the sensors and update the Firebase database with the latest parking availability and environmental conditions using Wi-Fi connection.
- The mobile application will retrieve this data in real-time, ensuring users have access to the most current information regarding parking space availability and their entry/exit logs.

5.2 User Interactions:

- When users enter or exit the parking area, their actions will be logged through the keypad, with corresponding data sent to Firebase. The mobile app will pull this information, allowing users to view their parking history and billing details directly within the app.

6. Project Layout

We are utilizing a combination of both **navigation drawer** and **bottom navigation** in our app to provide a streamlined and intuitive user experience. The **bottom navigation** is designed for quick access to commonly used features and fragments:

- **Home**: The main dashboard or landing page.
- **Park**: Users can search for parking lots, view available options, and manage bookings (current, upcoming, and past).
- **Activity**: Shows users' current, upcoming, and past booking status.
- **My Account**: Leads users to more detailed settings and preferences via the navigation drawer.

The **navigation drawer** is designed for accessing additional features that are less frequently used, such as:

- **Account Settings**: Edit user profile and preferences.
- **App Settings**: Manage notifications, language, or appearance preferences.
- **App Review**: A section for leaving feedback on the app.
- **Notifications**: View messages and alerts.
- **Help**: Access support options and documentation.
- **Legal Documentation**: Includes privacy policy, terms of use, and legal disclaimers.
- **Logout**: A simple option to sign out of the account.

6.1 Navigation Drawer

6.1.1. How to Use:

1. Add **DrawerLayout** as the root view in your activity.
2. Use **NavigationView** for defining drawer menu items.
3. Implement **NavController** to manage fragment navigation when items are selected.
4. Add a **toggle button** in the toolbar to open/close the drawer.

6.1.2. Why to Use:

- **Ideal for secondary features**: Like settings, help, notifications, and account-related actions that don't need constant visibility.
- **Clean UI**: Keeps less-used features out of the main interface, providing a cleaner layout for primary tasks.

6.1.3 Use Cases in Existing Apps:

- **Gmail**: Labels, account settings, and additional folders are hidden in the navigation drawer, allowing quick access when needed without crowding the interface.
- **YouTube**: Access to your library, history, account, and settings via the drawer.

6.1.4 Pros Over Bottom Navigation:

- **More space**: Suitable for apps with many options and features that don’t need immediate visibility.
- **Expandable options**: You can add more items to the drawer without cluttering the screen.

6.2 Bottom Navigation

6.2.1 How to Use:

1. Add a **BottomNavigationView** to your layout.
2. Define core navigation items in XML (e.g., Home, Search, Profile).
3. Connect the **NavController** to handle fragment transactions when a tab is selected.

6.2.2 Why to Use:

- **Quick access to core features**: Ideal for apps where the user frequently switches between a few main sections.
- **Consistent visibility**: Stays visible at the bottom of the screen for easy navigation across different app sections.

6.2.3 Use Cases in Existing Apps:

- **Instagram**: Uses bottom navigation for Home, Search, Reels, Shop, and Profile, making frequently accessed sections easily available.
- **Spotify**: Navigates between Home, Search, and Your Library with bottom navigation, ensuring quick access to main functionalities.

6.2.4 Pros Over Navigation Drawer:

- **Speed and simplicity**: Allows users to quickly access the most important features with a single tap.
- **Visibility**: Bottom navigation is always visible, ensuring easy switching between the app's core features.

6.3 One vs. The Other

- **Navigation Drawer** is ideal for secondary or less frequently used features like settings, account management, and documentation.
- **Bottom Navigation** is better for primary app functions that need quick and consistent access.

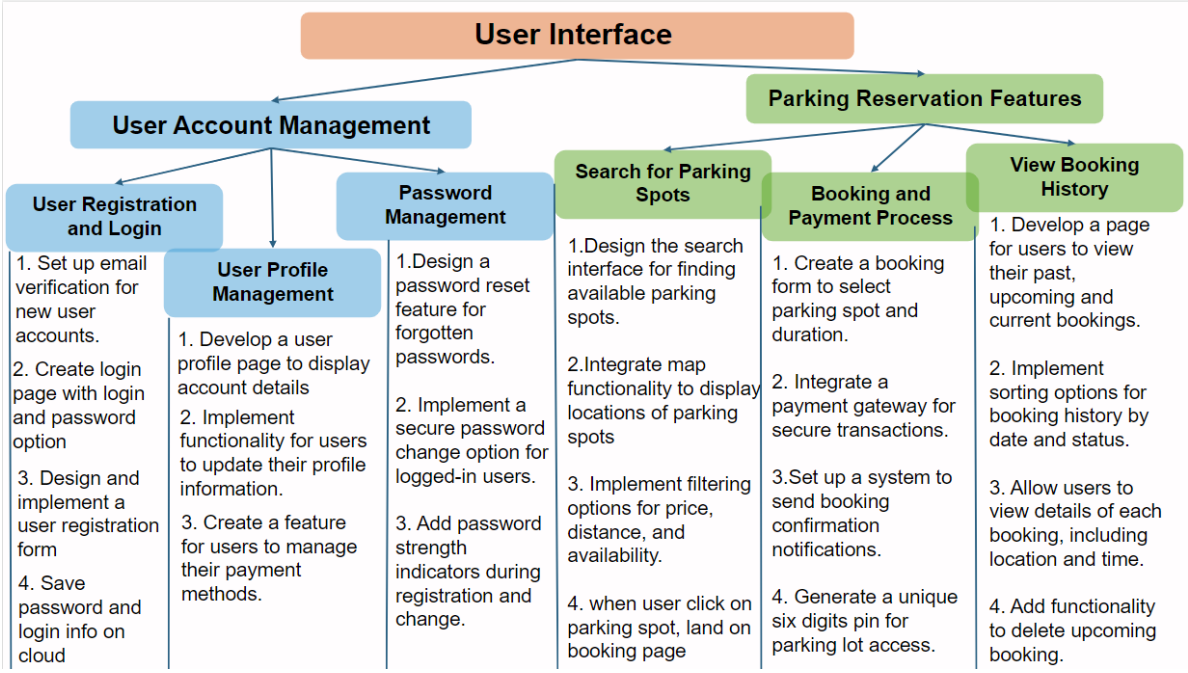
6.4 Apps That Use Both

- **YouTube**: Uses **Bottom Navigation** for core features like Home, Explore, Subscriptions, and **Navigation Drawer** for account settings, legal information, and app settings.
- **Uber**:

- **Bottom Navigation**: Used for Home, Ride, and Profile, giving users quick access to main functions like booking rides and viewing trip history.
- **Navigation Drawer**: For settings, help, legal information, and payment options.

7. Themes

7.1. Theme 1 - User Interface



7.2. Theme 2 – Owner Interface