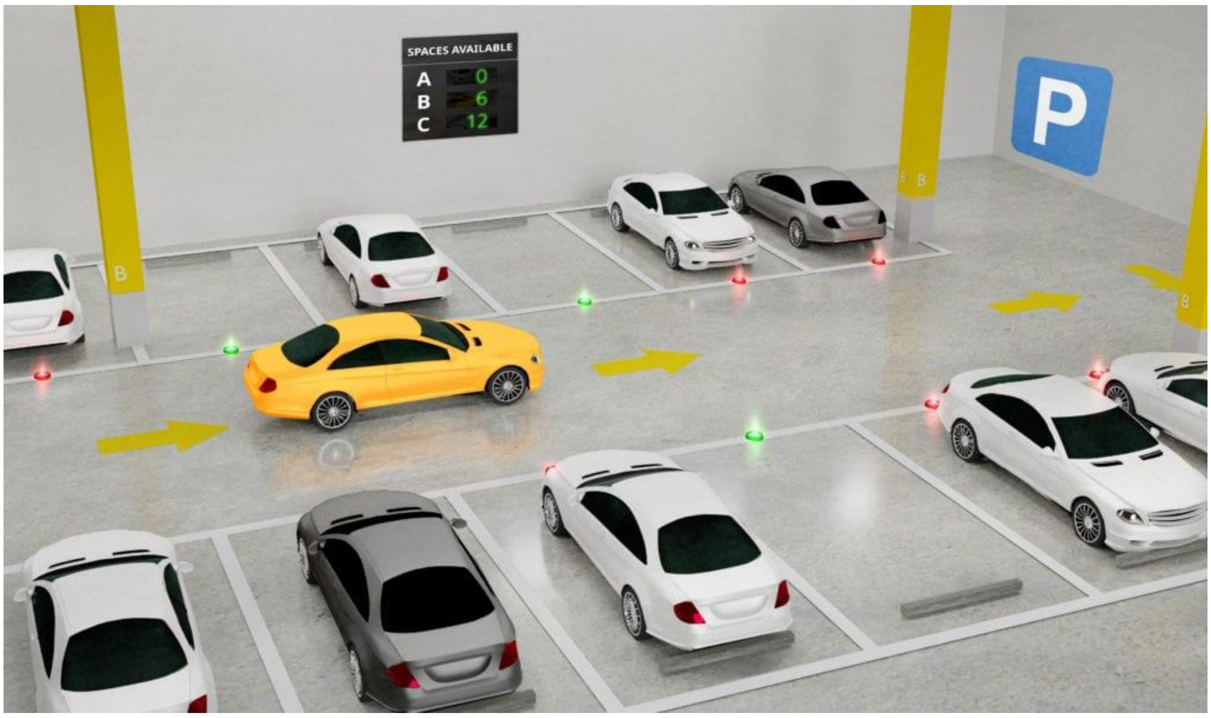
Reg no: 510421104028

Name: DIVYA.A

Phase-4 Document submission

Project: *SMART PARKING*



INTRODUCTION

- In this phase of the project, the focus shifts towards the development of a mobile application for the IoT Smart Parking system.
- Leveraging the versatility and power of Python, we aim to create an intuitive and user-friendly interface that will seamlessly interact with the IoT infrastructure.
- The successful development of the mobile app will mark a substantial milestone in the realization of the IoT Smart Parking system's full potential.

OBJECTIVE

- Develop a mobile app for the IoT Smart Parking system using Python. plan the design and deployment of IoT sensors in parking spaces to detect occupancy and availability.

- In phase 3 we discussed about the raspberry pi , sensors with example coding and cloud integration in IoT

Phase 4: Development Part 2

1) In this part you will continue building your project.

2) Continue building the project by developing the mobile app using Python.

3) Use a mobile app development framework (e.g., Flutter) to create an app that displays realtime parking availability.

4) Design app functions to receive and display parking availability data received from the Raspberry Pi.

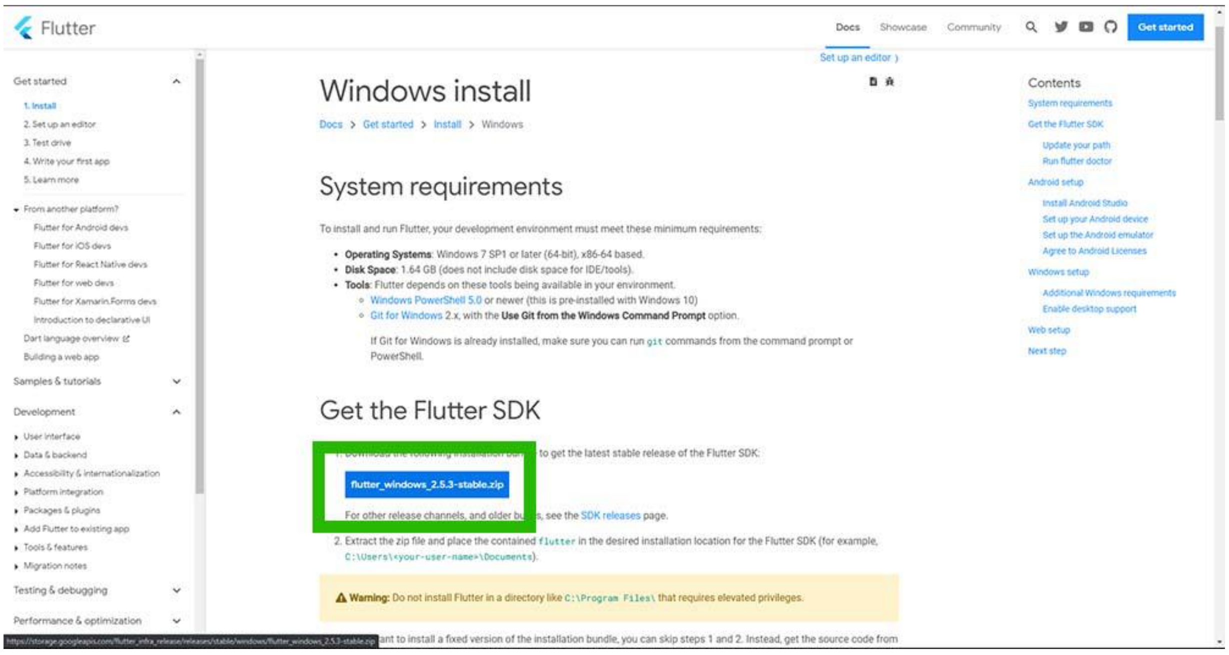
STEPS:

1. Set Up Development Environment

- Install necessary tools and libraries for mobile app development in Python.(e.g., Flutter)

- Install Flutter SDK on your development machine. Follow the official Flutter documentation for instructions on installation: Flutter Installation Guide

- Set up an IDE (Integrated Development Environment) for Flutter, such as Visual Studio Code or Android Studio.



Windows install

Docs > Get started > Install > Windows

System requirements

To install and run Flutter, your development environment must meet these minimum requirements:

- **Operating Systems**: Windows 7 SP1 or later (64-bit), x86-64 based.
- **Disk Space**: 1.64 GB (does not include disk space for IDE/tools).
- **Tools**: Flutter depends on these tools being available in your environment.
  - Windows PowerShell 5.0 or newer (this is pre-installed with Windows 10)
  - Git for Windows 2.x, with the **Use Git from the Windows Command Prompt** option.

  If Git for Windows is already installed, make sure you can run `git` commands from the command prompt or PowerShell.

Get the Flutter SDK

1. [illegible] to get the latest stable release of the Flutter SDK:

   For other release channels, and older bu[illegible]s, see the SDK releases page.
2. Extract the zip file and place the contained `flutter` in the desired installation location for the Flutter SDK (for example, `C:\Users\<your-user-name>\Documents`).

Warning: Do not install Flutter in a directory like `C:\Program Files\` that requires elevated privileges.

ant to install a fixed version of the installation bundle, you can skip steps 1 and 2. Instead, get the source code from

2. Define App Features

Here are some potential features you might consider including in your parking availability app:

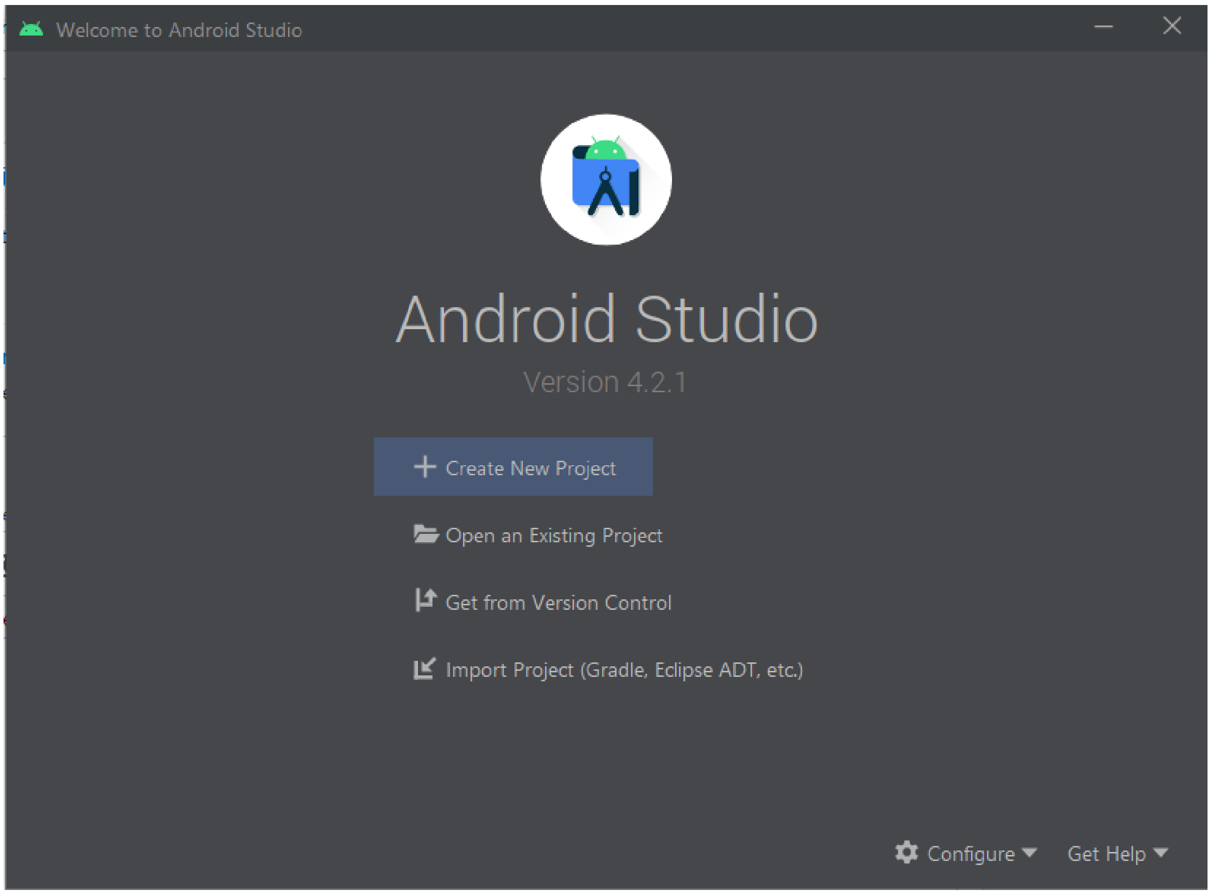
1. Real-time Parking Availability
2. Search and Navigation
3. User Authentication
4. Reservation and Booking
5. Notifications
6. Parking Details
7. Parking Space Navigation
8. Multi-language Support
9. User Reviews and Ratings
10. Emergency Assistance

Remember to prioritize features based on the core functionality you want to provide to your users.

Start with a minimum viable product (MVP) and then consider adding more advanced features as your app gains traction and user feedback.

3. Create a New Flutter Project

Once Flutter is installed, follow these steps to create a new Flutter project:



1. Open a terminal or command prompt.
2. Use the following command to create a new Flutter project:

   ```
   flutter create your_project_name
   ```

Replace your_project_name with the desired name of your project.

For example, if you want to create a project named parking_app, you would use:

```
flutter create parking_app
```

1) Once the project is created, you'll see a message indicating that the project has been successfully created.

   It will also provide you with some instructions on how to get started with your new project.

2) Change into the project directory using the following command:

```
cd your_project_name
```

Again, replace your_project_name with the actual name of your project.

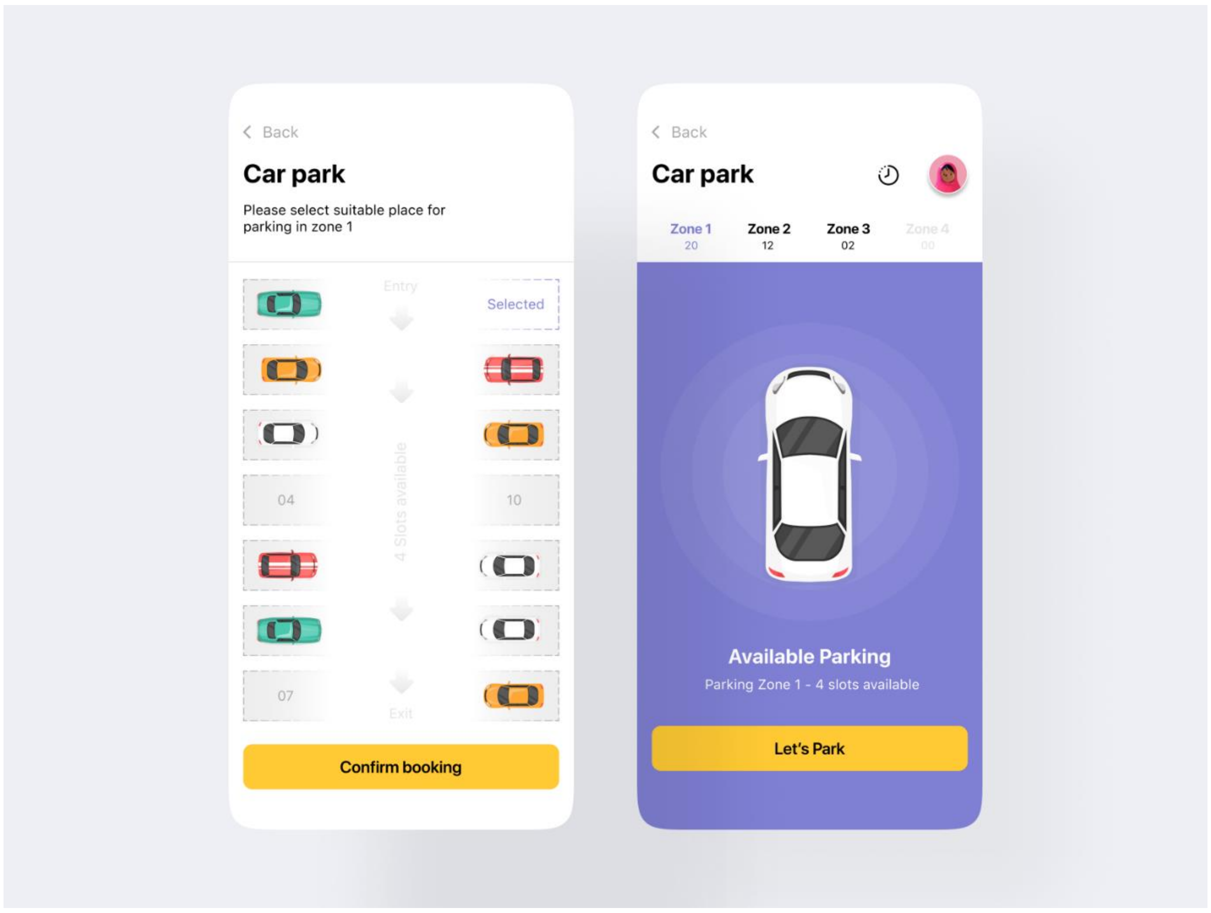
4. Designing the App UI

- Open the project in your chosen IDE.

- Define the UI layout of the app.

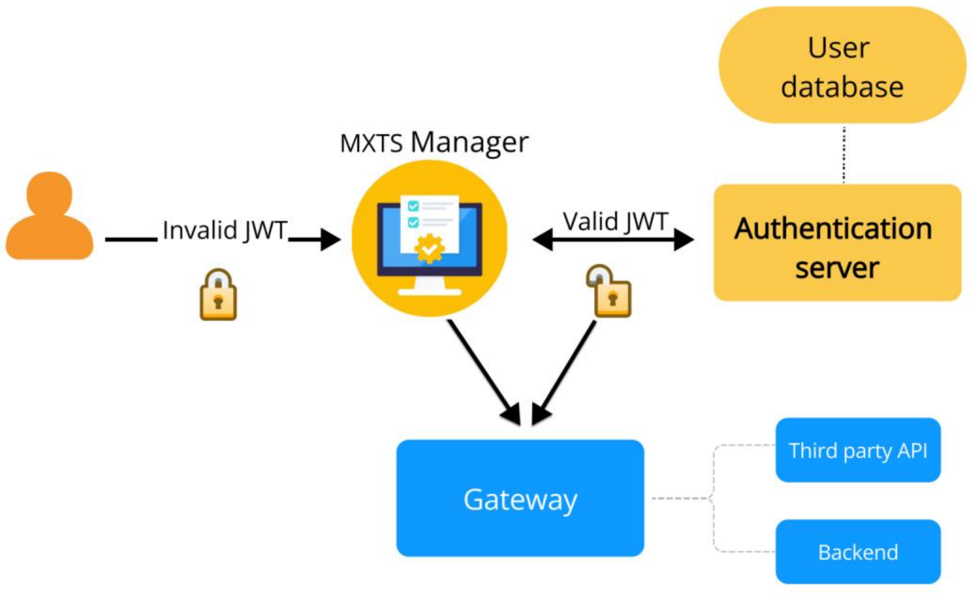
- This may include elements like buttons, labels, and a space for displaying parking availability.

- Use Flutter widgets to create a visually appealing and user-friendly interface.



5. Implement User Authentication

- Develop a user authentication system to secure the app.



Backend: Use a server-side language (e.g., Python, Node.js, Java) and a framework (e.g., Django, Express.js, Spring Boot) for handling server logic.

Database: Choose a database system (e.g., PostgreSQL, MySQL, MongoDB) to store user data.

Authentication server : An authentication server is responsible for managing user identities, authenticating users, and granting access based on their credentials.

5. Integrate IoT Data

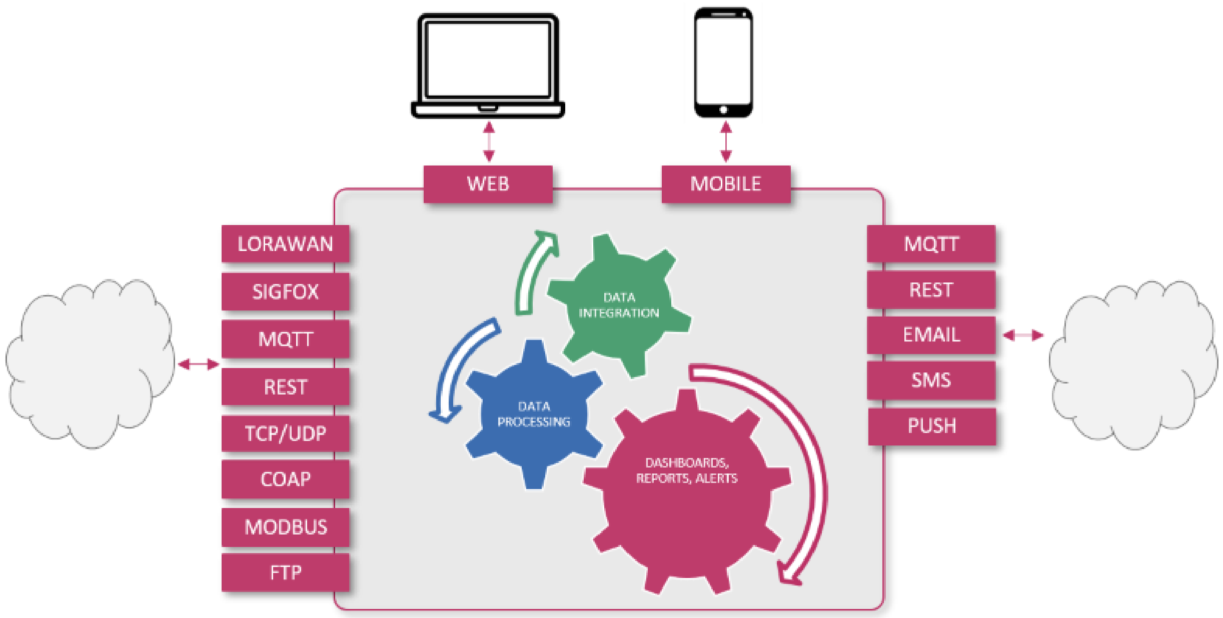
- Connect the mobile app to the IoT Smart Parking system to retrieve real-time data on parking availability.

a. Set Up IoT Devices and Sensors: Install occupancy sensors, cameras, or any other relevant IoT devices in the parking area to monitor parking space availability.

b. Choose a Communication Protocol: Select a communication protocol for your IoT devices to communicate with the gateway MQTT, HTTP, or WebSocket are common choices for real-time applications.

c. Set Up a Mobile App: Develop a mobile app that will interact with the IoT Smart Parking system. This app should be able to request real-time data on parking availability.

d. Data Processing and Transformation: Process the raw data received from the IoT devices. This may involve filtering, aggregating, and transforming the data to make it suitable for your application.



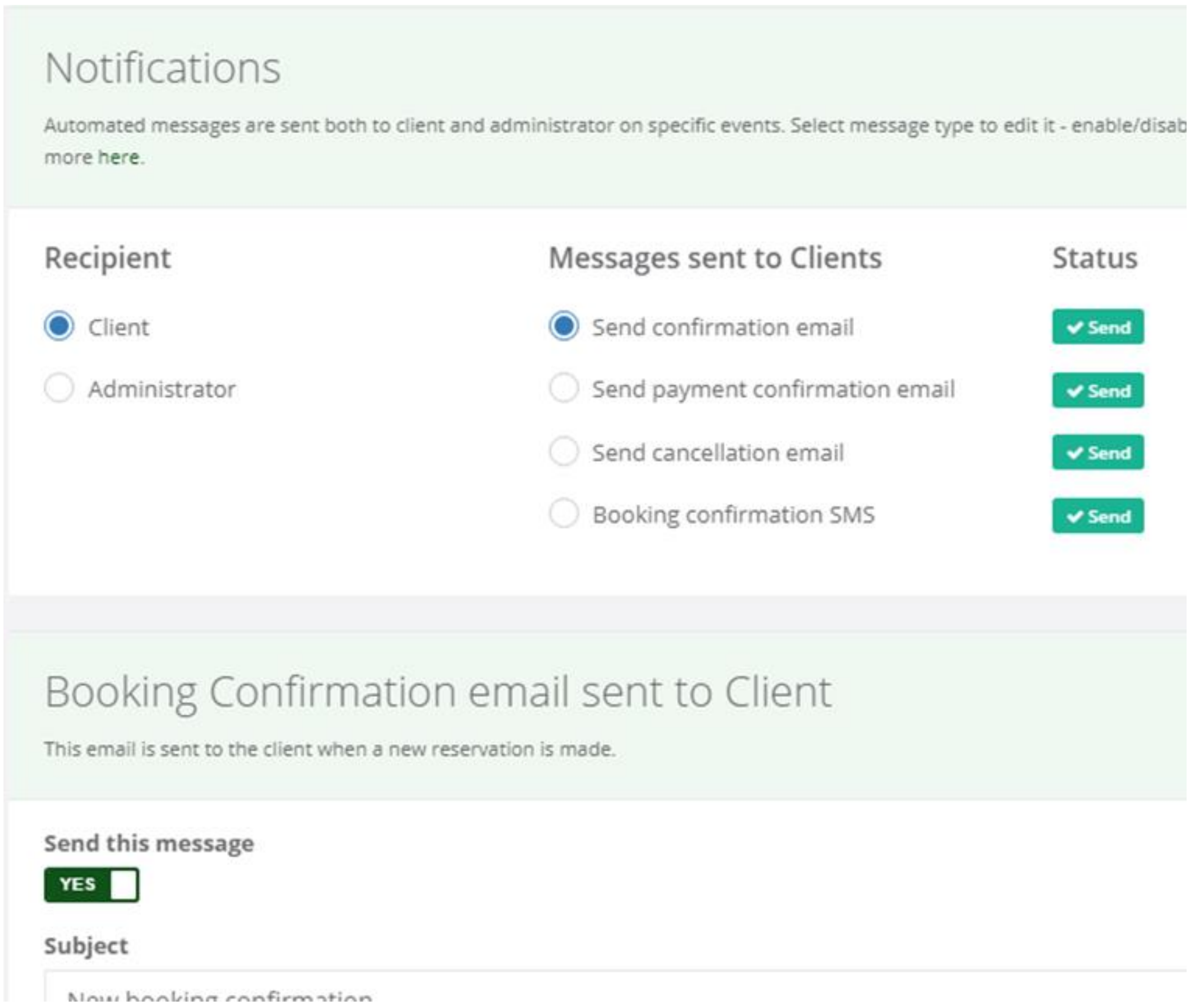
6. Add Navigation Features

- Integrate navigation capabilities to guide users to their selected parking spot.

User Panel
Register
Track vehicle location
Search
Arrival and Departure time
Booking
Payments
Parking History
Ratings & Reviews

7. Implement Notifications/Alerts

- Set up notifications or alerts to inform users of important events (e.g., successful reservation, parking time limit reached).



Notifications

Automated messages are sent both to client and administrator on specific events. Select message type to edit it - enable/disab
more here.

Recipient	Messages sent to Clients	Status
● Client	● Send confirmation email	✓ Send
○ Administrator	○ Send payment confirmation email	✓ Send
	○ Send cancellation email	✓ Send
	○ Booking confirmation SMS	✓ Send

Booking Confirmation email sent to Client

This email is sent to the client when a new reservation is made.

Send this message

YES

Subject

New booking confirmation

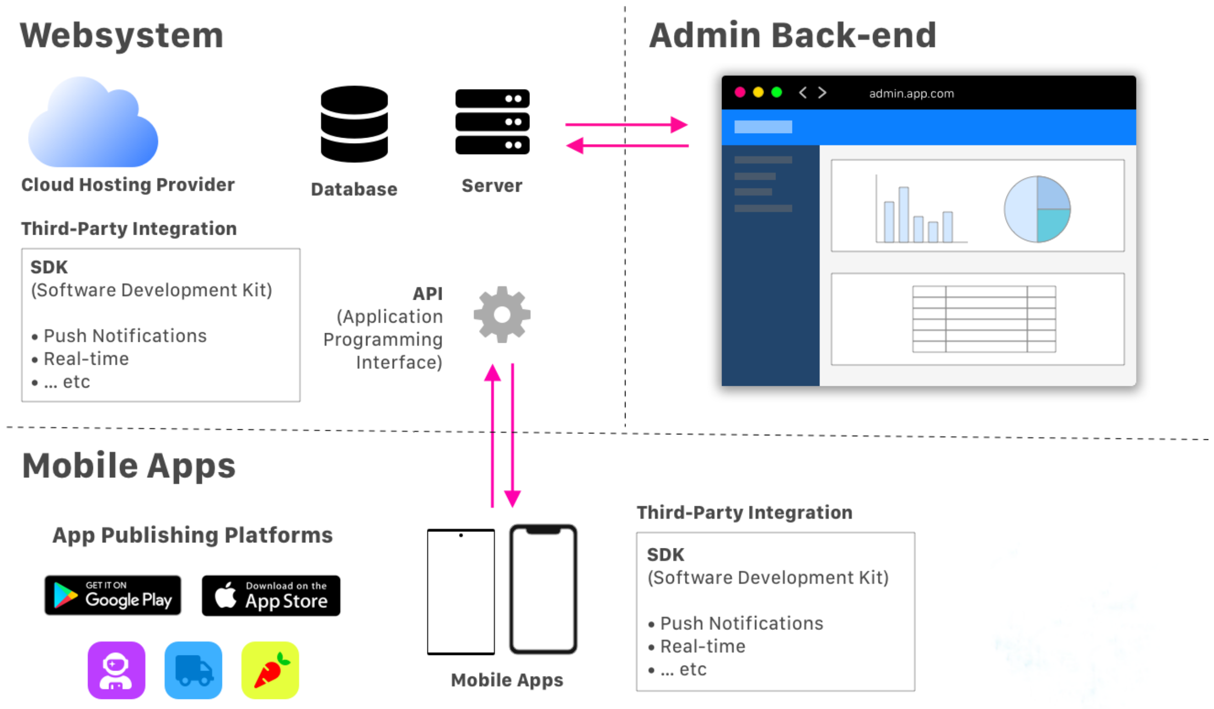
9. Testing and Debugging

- Thoroughly test the app to ensure all features work as expected.
- Address any bugs or issues that arise during testing.

10.Overview of mobile app

It includes

- Web system
- Admin Back-end
- App publishing platforms
- Third-party integration
- SDK
- API



11. Documentation

- Create documentation that includes:
  - Overview of the mobile app
  - Technical details (libraries used, architecture)
  - User guide

12. Deployment

- Publish the app to an appropriate platform (e.g., Google Play Store, Apple App Store) or distribute it to intended users.

CONCLUSION

- Phase 4 saw the successful development of the mobile app using Python.
- This critical component enhances user interaction, providing real-time updates on parking availability, navigation, and notifications.
- The fusion of technology and user-centric design ensures a seamless parking experience. With Phase 4 completed, we are poised for integration and testing in Phase 5.