**ParkPoint – Smart Parking Finder**

**ParkPoint** is a comprehensive smart parking management system consisting of a mobile application for users and a web dashboard for parking spot owners. The system addresses urban parking challenges by providing real-time information about parking spot availability, pricing, and location while offering a seamless booking and management experience.

**Table of Contents**

* [Features](#features)
* [Tech Stack](#tech-stack)
* [Installation](#installation)
  + [Mobile App Setup](#mobile-app-setup)
  + [Owner Website Setup](#owner-website-setup)
* [Usage](#usage)
  + [Mobile App Usage](#mobile-app-usage)
  + [Owner Website Usage](#owner-website-usage)
* [Firebase Configuration](#firebase-configuration)
* [Google Pay Integration](#google-pay-integration)

**Features**

**User Mobile App**

* **User Registration and Login**: Users sign up or log in using phone number authentication with Firebase.
* **Map View**: Users can search for nearby parking spots displayed on a map.
* **Parking Spot Booking**: Users can view parking spot details, such as time, price, and distance, and book spots.
* **QR Code-Based Parking**: QR code scanning for entry and exit, updating parking spot availability in real-time.
* **Search Suggestions**: Location-based suggestions as users type in the search bar.
* **Payment Integration**: Integration with Google Pay for seamless payment of parking fees.
* **Booking Management**: Users can extend their parking time and manage existing bookings.

**Owner Web Dashboard**

* **Spot Registration**: Owners can register multiple parking spots with details such as location, pricing, and capacity.
* **Dashboard**: Owners can manage all their registered spots and view real-time booking data.
* **Booking Management**: Automated management of parking spots, with booking status updates based on QR code scans.
* **Session Handling**: Owners can securely log in and manage sessions with Firebase Authentication.

**Tech Stack**

* **Mobile App**: Java, Android SDK, Firebase (Authentication, Realtime Database)
* **Owner Website**: Node.js, Express, EJS, Firebase (Authentication, Realtime Database)
* **Payment Gateway**: Google Pay API integration
* **Maps**: Google Maps API for displaying parking spots

**Installation**

**Mobile App Setup**

1. Clone the repository:

git clone https://github.com/VatFranklin/ParkPoint.git

cd parkpoint-mobile-app

1. Open the project in Android Studio.
2. Add Firebase configuration:
   * Download the google-services.json file from Firebase Console and place it in the app/ directory.
3. Build and run the project on your device or emulator.

**Owner Website Setup**

1. Clone the repository:

git clone https://github.com/VatFranklin/ParkPoint.git

cd parkpoint-owner-website

1. Install dependencies:

npm install

1. Add Firebase configuration:
   * Set up your Firebase project.
   * Add your Firebase configuration details to .variables.env.
2. Start the server:

npm start

1. Access the web dashboard at http://localhost:3000.

**Usage**

**Mobile App Usage**

1. **User Login**: Users log in using their phone number (OTP-based authentication).
2. **Search Parking**: Use the search bar to find nearby parking spots based on location.
3. **Book a Spot**: View spot details and book a parking space.
4. **Payment**: Complete the booking using Google Pay.
5. **QR Code Entry/Exit**: Scan QR codes for parking entry and exit.

**Owner Website Usage**

1. **Owner Login**: Log in to the dashboard using Firebase Authentication.
2. **Register Parking Spots**: Add new spots with location, pricing, and capacity.
3. **Manage Bookings**: View real-time bookings and manage spot availability.

**Firebase Configuration**

1. Create a Firebase project in the Firebase Console.
2. Enable Authentication and Realtime Database.
3. Download google-services.json (for mobile) and configure Firebase Admin SDK (for web).
4. Set up your Firebase Realtime Database structure for storing parking spot and booking data.

**Google Pay Integration**

For integrating Google Pay in the mobile app:

1. Include the following in your build.gradle:

gradle

implementation 'com.google.android.gms:play-services-wallet:18.1.3'

1. Follow the Google Pay API documentation to integrate payment functionality in your app.

|  |  |  |
| --- | --- | --- |
| In your activity, set up the payment request and handle the transaction. |  |  |