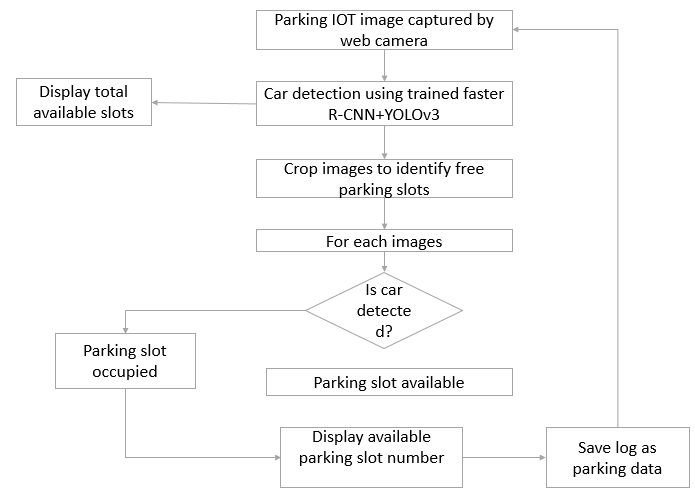
**SMART PARKING**

**Project Objectives:**

This objective involves the implementation of a system that continuously tracks and updates the availability of parking spaces in a given area. It utilizes sensors or technology to provide up to the minute information on vacant and occupied parking spots, helping drivers find parking quickly.  It allows users to access real-time parking information, reserve parking spaces, make payments, and receive navigation guidance via their smartphones. This integration enhances user convenience and accessibility.  Efficient parking guidance aims to optimize the utilization of parking spaces by providing drivers with clear directions to available spots. This can involve the use of digital signage, mobile apps, or guidance systems that direct drivers to the nearest available parking spaces, reducing congestion and improving overall parking efficiency. These objectives collectively contribute to enhancing the parking experience by providing real-time information, convenient mobile access, and efficient parking solutions.

**IOT Sensor Design:**

****

**Real-Time Transit Information Platform:**

Creating a mobile app interface for real-time parking availability requires careful consideration of user experience and functionality.  Real-time updates on available parking spots nearby, with color-coded markers indicating availability, Users can switch to a list view to see parking options in a scrollable list format. Integration with GPS navigation apps for turn-by-turn directions. A separate admin interface for parking facility managers to monitor occupancy, revenue, and user reviews.

**Integration Approach:**

There are the several steps to follow in the raspberry pi in this smart parking

1. **Select and Connect Sensors:**

* Choose suitable sensors for detecting parking space occupancy, such as ultrasonic sensors or infrared sensors.
* Connect these sensors to the Raspberry Pi using GPIO pins or suitable sensor modules.

1. **Program Raspberry Pi:**

* Write Python or another suitable programming language code to interface with the sensors. This code will read data from the sensors and determine if parking spaces are occupied.
* Create a script to process sensor data, interpret it, and store it if needed.

1. **Data Processing and Storage**

* Depending on your requirements, you may store the sensor data in a database on the Raspberry Pi or transmit it directly to a cloud-based database for scalability.
* Use suitable libraries or APIs to manage the data.

1. **Mobile App Integration**

* Develop a mobile app for users to access parking information. You can use platforms like Android Studio (for Android) or Xcode (for iOS) to create the app.
* Implement a feature in the app to request and receive parking availability information.

1. **API or Communication Protocol**

* Create an API or communication protocol (e.g., RESTful API, MQTT) that allows the mobile app to communicate with the Raspberry Pi.

1. **Real-Time Updates**

* Set up a mechanism for the Raspberry Pi to push updates to the mobile app whenever there is a change in parking space availability. This can be done through push notifications or periodic polling of the API.

1. **User Interface**

* Design a user-friendly interface in the mobile app that displays parking availability information in real-time.

1. **Testing and Deployment**

* Thoroughly test the entire system to ensure sensor data accuracy and app functionality.
* Deploy the Raspberry Pi and sensors in the parking lot.

1. **Security Measures**

* Implement security measures to protect data transmission between the Raspberry Pi and the mobile app, especially if sensitive data is involved.

1. **Monitoring and Maintenance:**

* Set up monitoring to detect and address any issues with the Raspberry Pi or sensors in real-time.
* Regularly maintain and update the system as needed.