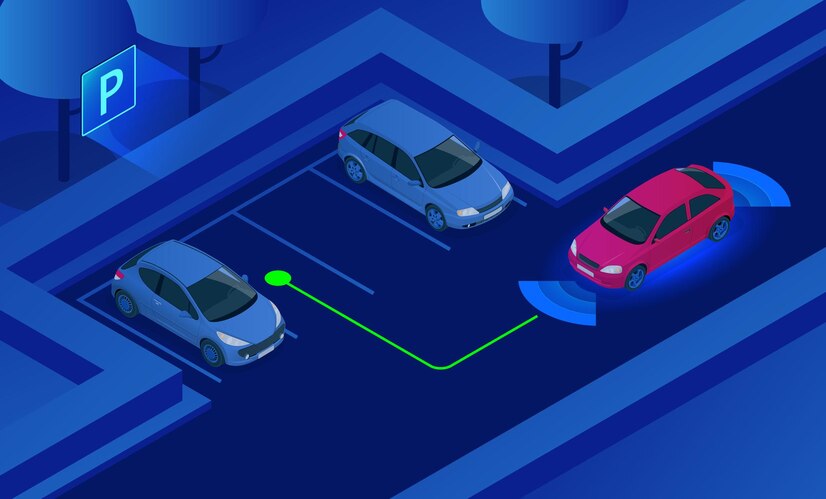
**PROJECT TITLE : SMART PARKING**

**Phase 1: Project Definition and Design Thinking**

**Project Definition:**The project involves integrating IoT sensors into public transportation vehicles to monitor ridership, track locations, and predict arrival times. The goal is to provide real-time transit information to the public through a public platform, enhancing the efficiency and quality of public transportation services. This project includes defining objectives, designing the IoT sensor system, developing the real-time transit information platform, and integrating them using IoT technology and Python.

**DESIGN THINKING: **

* **OBJECTIVES OF SMART PARKING:**

The objectives of smart parking using IoT, including real-time parking monitoring, mobile app integration, and efficient parking guidance, are :  
  
**1. Real-time parking monitoring:** One of the main objectives is to have a real-time monitoring system that continuously updates the availability of parking spaces. By using IOT sensors

**2. Mobile app integration**: The integration of smart parking systems with mobile applications allows drivers to access real-time parking information, reserve parking spots in advance, and make payments conveniently through their smartphones.

**3. Efficient parking guidance**: IoT-enabled smart parking systems can guide drivers to available parking spaces more efficiently. By leveraging sensors, cameras, or signage, the system can direct drivers to the nearest vacant parking spots, reducing the time

**4.Traffic management and congestion reduction:** Smart parking using IoT aims to contribute to better traffic management by reducing the number of vehicles circling around in search of parking.

**5. Revenue optimization:** Another objective of smart parking is to optimize revenue generation for parking operators. By implementing IoT systems, parking authorities can efficiently manage payment processes

* **Design and deployment of IoT sensors in parking spaces to detect occupancy and availability**   
    
  **1. Assess the parking area:** Begin by assessing the parking area's layout, size, and specific requirements. Identify the number of parking spaces and any unique features or challenges that need to be considered during design and deployment.  
   **2. Determine sensor placement:** Based on the assessment, determine the optimal locations to place the IoT sensors. Consider factors such as visibility, accessibility, power supply availability, and the type of parking spaces (e.g., parallel, angled, or perpendicular).  
    
  **3. Choose suitable IoT sensors:** Select IoT sensors that are reliable, accurate, and suitable for the parking space environment. Options include ultrasonic sensors, magnetic sensors, or camera-based sensors. Consider the cost, range, connectivity requirements, and compatibility with the chosen IoT platform.  
    
  **4. Establish connectivity infrastructure:** Determine the connectivity infrastructure required to transmit data from the sensors to the central system. This can be achieved through Wi-Fi, cellular networks, or dedicated IoT communication protocols like LoRaWAN or NB-IoT. Ensure sufficient coverage and network capacity.  
    
  **5. Plan power supply:** Decide on the power supply options for the IoT sensors. They can be battery-powered, solar-powered, or connected to a constant power source. Consider the maintenance requirements and the lifespan of the chosen power supply solution.  
    
  **6. Develop the central system**: Design and develop the central system that will receive, process, and analyze the data from the IoT sensors. It should provide real-time parking information and status updates. Consider integrating it with mobile apps, web portals, or smart signage for user convenience.  
    
  **7. Test and validate:** Before deployment, conduct thorough testing of the IoT sensors, connectivity, and central system. Validate the accuracy of occupancy detection and the responsiveness of the system. Make necessary adjustments or refinements if required.  
    
  **8. Deploy sensors and infrastructure:** Install the IoT sensors in the determined locations using appropriate mounting hardware. Deploy necessary infrastructure such as gateways, routers, or access points for data transmission and connectivity. Ensure proper installation, weatherproofing, and security measures.  
    
  **9. Monitor and maintain:** Establish a monitoring and maintenance plan to ensure ongoing functionality. Regularly check sensor performance, battery levels, connectivity, and data accuracy. Address any issues promptly to maintain optimal system performance.  
    
  **10. Data analysis and optimization:** Continuously analyze the collected data to identify parking patterns, occupancy trends, and utilization rates. Use this information to optimize parking operations, improve efficiency, and make informed decisions regarding pricing, capacity planning, and resource allocation.
* **A mobile app interface that displays real-time parking availability to users:**

2. **Map View:**  
- Upon entering a location, display a map view of the area with parking icons representing available spaces.  
- Show the user's current location as a marker on the map.  
- Color code the parking icons to indicate availability in real-time (e.g., green for available, orange for limited, and red for full).  
- Implement zoom and pan functionalities to allow users to explore the map.  
 **3. List View**:  
- Provide an alternative list view of nearby parking spots alongside the map view.  
- Show the distance to each parking spot from the user's current location.  
- Display the availability status of each parking spot using color-coded indicators similar to the map view.  
- Include additional details such as parking rates, hours of operation, and any special features or restrictions.  
 **4. Filter and Sorting Options:**  
- Include filter and sorting options to allow users to customize their search.  
- Provide filters for preferred parking types (e.g., covered, street, garages) and amenities (e.g., EV charging, disabled parking).  
- Offer sorting options based on distance, price, availability, or user ratings.  
 **5. Detailed Parking Spot Information:**  
- Upon selecting a specific parking spot from the map or list view, display detailed information about that spot.  
- Show real-time occupancy information (e.g., the number of available spaces out of total capacity).  
- Provide directions to the parking spot from the user's current location, including estimated travel time.  
- Include parking rates, accepted payment methods, and any additional services or requirements.  
  
**6. User Ratings and Reviews:**  
- Allow users to rate and review parking spots they have used.  
- Display average ratings and relevant reviews to help users make informed decisions.  
- Enable users to filter and sort search results based on ratings or reviews.  
  
**7. Account and Booking:**  
- Implement a user account system that allows users to save favorite parking spots and manage their bookings.  
- Integrate a secure online payment system to enable booking and pre-payment of parking spaces.  
- Provide booking confirmation and digital tickets within the app.  
  
**8. Notifications and Alerts:**  
- Send push notifications or in-app alerts to inform users about parking availability updates, promotions, or time-limited discounts.  
- Notify users when their booked parking spot is close or when there are changes to the availability status.  
  
Remember to maintain a user-friendly and intuitive interface throughout the app. Use clear and concise language, provide visual cues, and incorporate common mobile app design patterns. Regularly gather user feedback and conduct usability testing to refine and improve the interface based on user needs and preferences.

* **To collect data from sensors and update the mobile app using a Raspberry Pi,**

**1. Sensor Integration:**  
- Identify the necessary sensors for collecting the desired data (e.g., parking occupancy, temperature, humidity).  
- Connect the sensors to the appropriate GPIO (General Purpose Input/Output) pins on the Raspberry Pi.  
- Install any required libraries or drivers to enable communication with the sensors.  
  
**2. Data Collection and Processing:**  
- Write a script in a programming language like Python to read data from the connected sensors.  
- Use the appropriate libraries or APIs provided by the sensor manufacturers to access the sensor data.  
- Process the collected data as needed (e.g., calculate parking occupancy percentage).  
 **3. Real-Time Data Streaming:**  
- Establish a connection between the Raspberry Pi and the mobile app for real-time data streaming. One option is to use MQTT (Message Queuing Telemetry Transport) protocol.  
- Set up an MQTT on the Raspberry Pi or a separate server to facilitate communication.  
- Publish the collected sensor data to specific MQTT topics.  
  
**4. Mobile App Integration:**  
- Develop the mobile app using a framework like React Native or Flutter.  
- Implement MQTT client functionality in the mobile app to subscribe to relevant MQTT topics and receive sensor data updates.  
- Upon receiving new data, update the app's UI to display the real-time information (e.g., parking availability, temperature).