Design thinking :

1. Project Objectives: Define specific objectives such as real-time parking space monitoring, mobile app integration, and efficient parking guidance.

Certainly! Design Thinking is a problem-solving approach that emphasizes understanding user needs and generating creative solutions. Here's how you can proceed with your project objectives:

Empathize: Begin by understanding the needs of your target users, both drivers and parking facility operators. Conduct surveys, interviews, and observations to gather insights into their pain points and preferences related to parking.

Define: Based on your research, clearly define the specific objectives of your project. In this case, it includes:

Real-Time Parking Space Monitoring: Develop a system that can track and display real-time information about available parking spaces.

Mobile App Integration: Create a user-friendly mobile app that integrates with the monitoring system, allowing users to check parking availability, reserve spots, and navigate to their chosen parking space.

Efficient Parking Guidance: Implement a guidance system, possibly using sensors and signage, to direct drivers efficiently to available parking spaces.

- Prototype: Develop prototypes or mockups of your parking space monitoring system and mobile app. These prototypes will help you visualize the solutions and gather feedback from potential users.
- Test: Test the prototypes with real users to identify any usability issues or improvements needed. Iterate on your designs based on user feedback.
- Implement: Once you have a refined solution, proceed with the full-scale implementation of the real-time parking space monitoring system, mobile app, and parking guidance system.

2.IoT Sensor Design: Plan the design and deployment of loT sensors in parking spaces to detect occupancy and availability

Designing and deploying IoT sensors for parking space occupancy detection involves several steps. Here's a plan to help you get started:

Define Objectives:

- Determine the specific goals of your parking space IoT system, such as reducing congestion, improving user experience, or optimizing space utilization.

Sensor Selection:

- Choose appropriate sensors for occupancy detection. Options include ultrasonic sensors, infrared sensors, or camera-based systems. Consider factors like accuracy, cost, and power consumption.

Network Connectivity:

- Decide on the connectivity technology, such as Wi-Fi, LoRaWAN, or cellular, to transmit data from sensors to a central server or cloud platform.

Power Supply;

- Ensure a reliable power source for the sensors, whether it's through batteries, solar panels, or wired connections.

Sensor Placement:

- Strategically position sensors within parking spaces for optimal coverage and accuracy. Consider factors like vehicle size and placement variability

3. Real-Time Transit Information Platform: Design a mobile app interface that displays real-time parking availability to users.

Home Screen:

1. Header: Displays the app's logo and a search bar.

2. Map:Shows the user's current location and nearby parking lots as icons.

3. Filter:Users can filter parking options by type (garage, street, lot), price range, and distance.

4. List View:A list of nearby parking options with details like name, distance, and availability status (e.g., "Open," "Full," "Limited").

5. Refresh Button: Allows users to manually update the real-time data.

Parking Lot Details Screen:

1. Header: Back button, parking lot name, and real-time availability status.

2. Map: Displays the parking lot's location with directions.

3. Availability: Shows the number of available parking spots and the total capacity.

4. Pricing: Lists hourly and daily rates.

5. Opening Hours: Displays the lot's operating hours.

6. Book Now: If available, users can reserve a parking spot in advance.

7. Reviews & Ratings: User-generated ratings and comments.

Navigation:

- Menu: Access to user profile, settings, and help/contact information.
- Profile: Allows users to set preferences, view booking history, and add payment methods.
- Settings: Customize app settings, notifications, and preferred payment methods.
- Help/Contact: Access FAQs, contact customer support, or report issues.

4. Integration Approach: Determine how Raspberry Pi will collect data from sensors and update the mobile app.

To integrate Raspberry Pi with sensors and update a mobile app, you can follow these general steps:

1. Sensor Setup:

- Connect the sensors (e.g., temperature, humidity, motion, etc.) to the Raspberry Pi. Ensure they are compatible and have the required libraries or drivers.

2. Data Collection:

- Write Python scripts on the Raspberry Pi to collect data from the connected sensors. You'll need to use libraries or APIs specific to each sensor.

3. Data Processing:

- Process and format the collected sensor data as needed. You may need to convert it to a standardized format (e.g., JSON) for easier transmission.

4. Communication Protocol:

- Choose a communication protocol to send data from Raspberry Pi to your mobile app. Common options include:
- REST API: Set up a RESTful API on the Raspberry Pi that the mobile app can make HTTP requests to.
- WebSocket: Implement a WebSocket server on the Raspberry Pi for real-time data updates.
- MQTT: Use the MQTT protocol for lightweight and efficient messaging between the Pi and the app.

5. Mobile App Integration:

    - Develop the mobile app (Android or iOS) using a programming language like Java (Android) or Swift (iOS).

6. API Endpoints (for REST):

- Create API endpoints on the Raspberry Pi to receive data from the sensors and serve it to the mobile app.
- Use secure authentication and authorization mechanisms to protect your API.

7. WebSocket Integration (if applicable):

- In the mobile app, use WebSocket libraries to establish a connection with the Raspberry Pi's WebSocket server.
- Set up event listeners to handle incoming sensor data in real-time.

8. MQTT Broker (if applicable):

- Set up an MQTT broker on the Raspberry Pi and configure topics for sensor data.
- In the mobile app, use an MQTT client library to subscribe to relevant topics and receive sensor updates.

9. Data Display:

- Design the app's interface to display sensor data, charts, or graphs as needed.
- Implement logic to update the UI when new data is received from the Raspberry Pi.

10. Error Handling and Security:

- ❖ Implement error handling in both the Raspberry Pi scripts and the mobile app to manage connectivity issues and data integrity.
- ❖ Ensure that your communication between the Raspberry Pi and the app is secure by using encryption and authentication where necessary.