INNOVATION

Problem:

The problem domain for a smart car parking system encompasses various challenges and considerations, including:

Space Optimization:

- Efficiently utilizing parking space to accommodate as many vehicles as possible while maintaining safety and accessibility.

User Experience:

- Ensuring a seamless and user-friendly experience for drivers when entering, finding, and exiting parking facilities.

Traffic Management:

- Minimizing traffic congestion around parking areas caused by vehicles searching for available spots.

Security and Safety:

- Implementing security measures like surveillance, lighting, and emergency response systems to ensure the safety of both vehicles and users.

Payment and Billing:

- Developing a reliable and convenient payment system for parking fees, which can include options like mobile payments, contactless payments, or subscription models.

Data Management:

- Handling and analyzing data related to parking usage, payment transactions, and user preferences to make informed decisions.

Accessibility:

- Designing parking facilities to accommodate people with disabilities and providing accessible features such as reserved spots and signage.

Infrastructure:

- Installing the necessary hardware and software components, including sensors, cameras, and communication networks, to enable smart parking functionality.

Environmental Impact:

- Implementing eco-friendly practices to reduce the environmental footprint of parking facilities, such as incorporating electric vehicle charging stations.

Regulations and Compliance:

- Adhering to local laws and regulations related to parking, data privacy, and accessibility.

Maintenance and Reliability:

- Ensuring the ongoing functionality and reliability of the system, including regular maintenance and troubleshooting.

Integration:

- Integrating with other transportation systems and urban planning initiatives to create a holistic approach to mobility and parking management.

Addressing these challenges and considerations is crucial for the successful implementation and operation of a smart car parking system, benefiting both users and the overall urban environment.

Need for smart parking system:

A smart car parking system offers several benefits:

Efficiency:

- It optimizes parking space usage, reducing congestion and the time it takes to find a parking spot.

Convenience:

- Users can easily locate available parking spots using mobile apps or signs, saving time and reducing stress.

Cost Savings:

- It can lower operational costs for parking facility owners by reducing the need for manual labor and maximizing space usage.

Environmental Impact:

- Reduced idling and searching for parking spots can lead to lower carbon emissions.

Data Insights:

- Smart systems can collect data on parking patterns, helping city planners and businesses make informed decisions.

Improved Safety:

- It can enhance security through features like surveillance cameras and emergency response systems.

Accessibility:

- Smart systems can offer features like reserved spots for disabled individuals, improving accessibility.

Overall, a smart car parking system can enhance the overall parking experience while benefiting both users and facility operators

Design Thinking

Components Needed:

IR Sensor:

- This sensor is used to detect the presence of a car. It typically has three pins - Vcc, GND, and OUT.

7805 Voltage Regulator:

- This component regulates the voltage to a stable 5V output.

LEDs (3):

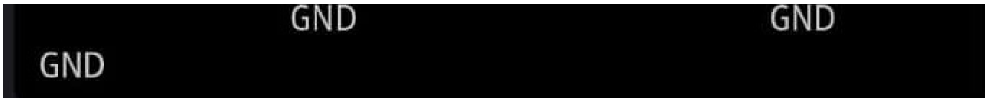
- These will be used to indicate the parking status (occupied or vacant).

Current-Limiting Resistors for LEDs:

- To prevent the LEDs from burning out, you'll need a resistor in series with each LED.

Circuit Diagram

```
5V
          |
         7805 (Voltage Regulator)
          |
          +---------> Vcc of IR Sensor
          |
         GND
          |
          |
        [220Ω Resistor]      [220Ω
Resistor]      [220Ω Resistor]
          |                   |
|
          |                   |
|
          |                   |
|
          +-------> Collector of LED 1
Collector of LED 2   Collector of LED 3
          |                   |
|
          |                   |
|
          |                   |
|
        LED 1               LED 2
LED 3
          |                   |
|
```



Circuit Connections:

IR Sensor:

- Connect Vcc of the IR sensor to the output pin of the 7805 voltage regulator.
- Connect GND of the IR sensor to the ground rail of the circuit.
- Connect the OUT pin of the IR sensor to a GPIO pin on your microcontroller (e.g., Raspberry Pi).

7805 Voltage Regulator:

- Connect the input pin of the 7805 to your power source (e.g., 9V battery or a suitable power supply).
- Connect the ground pin of the 7805 to the ground rail of the circuit.
- Connect the output pin (5V) of the 7805 to the Vcc pins of the IR sensor and LEDs.

LEDs:

- Connect the anode (longer leg) of each LED through a current-limiting resistor to the 5V output of the 7805.
- Connect the cathode (shorter leg) of each LED to a separate GPIO pin on your microcontroller (these will be used to control the LEDs).

CODING:

```
import RPi.GPIO as GPIO

import time

# Set up GPIO

GPIO.setmode(GPIO.BCM)

# Define pin numbers

IR_SENSOR_PIN = 17  # Example GPIO pin, adjust as needed

LED1_PIN = 18

LED2_PIN = 19

LED3_PIN = 20

# Initialize GPIO pins

GPIO.setup(IR_SENSOR_PIN, GPIO.IN)

GPIO.setup(LED1_PIN, GPIO.OUT)
```

```
GPIO.setup(LED2_PIN, GPIO.OUT)

GPIO.setup(LED3_PIN, GPIO.OUT)

def check_parking_status():

    if GPIO.input(IR_SENSOR_PIN) == GPIO.LOW:

        return True  # Car detected

    else:

        return False  # No car detected

def indicate_parking_status(status):

    if status:

        GPIO.output(LED1_PIN, GPIO.HIGH)  # LED1 ON

        GPIO.output(LED2_PIN, GPIO.LOW)   # LED2 OFF

        GPIO.output(LED3_PIN, GPIO.LOW)   # LED3 OFF

    else:

        GPIO.output(LED1_PIN, GPIO.LOW)   # LED1 OFF

        GPIO.output(LED2_PIN, GPIO.HIGH)  # LED2 ON

        GPIO.output(LED3_PIN, GPIO.LOW)   # LED3 OFF

try:

      while True:

       status = check_parking_status()

        indicate_parking_status(status)

        time.sleep(1)

except KeyboardInterrupt:

GPIO.cleanup()
```