Smart parking

NAME:G.PAVAN KUMAR

REG:711121106026

- **Smart Home Automation for Enhanced Comfort and Energy Efficiency**

Smart home automation systems leverage technology to provide homeowners with enhanced comfort, convenience, and energy efficiency. These systems enable the seamless control and management of various devices and appliances within the home, making everyday life more convenient and sustainable.

Objectives of the project

1. Smart Home Hub: A central control hub or controller that connects and manages all smart devices in the home. It acts as the brain of the system.
2. Sensors: Various sensors (e.g., motion, light, temperature, humidity, door/window sensors) placed throughout the house to collect data and trigger actions.
3. Smart Devices: A wide range of smart devices that can be controlled and automated, including smart thermostats, lights, locks, cameras, speakers, and appliances.
4. Mobile App: An intuitive mobile app that allows homeowners to remotely control and monitor their smart home from their smartphones or tablets.

Voice Assistants: Integration with voice-controlled devices like Amazon Alexa or Google Assistant for hands-free control.

Use Case Scenario: Let's explore a use case that illustrates the benefits of smart home automation in action.

Scenario: Enhanced Comfort and Energy Efficiency

User: Sarah, a homeowner

1. Morning Routine:

- Sarah's day starts at 7:00 AM. The smart home system gradually raises the bedroom blinds to let in natural light as the sun rises.

2. Energy Efficiency:

- The smart thermostat adjusts the temperature based on Sarah's schedule, optimizing heating and cooling to reduce energy consumption when she's not home.

3. Home Security:

- While Sarah is at work, motion sensors and cameras detect any unusual activity. If an unexpected movement is detected, the system sends an alert to her phone and activates outdoor security lights.

4. Remote Access: Sarah realizes she forgot to lock her front door. She uses her mobile app to remotely lock the smart lock and receive a confirmation.

5. Entertainment:

- In the evening, Sarah can use voice commands to control her smart TV, lights, and sound system for an immersive home theater experience.

6. Energy Savings:

- The system monitors energy consumption, and Sarah receives recommendations for further optimizing her energy usage, such as using energy-efficient LED bulbs or setting the thermostat to an eco-friendly mode.

7. Bedtime Routine:

- As Sarah goes to bed, she can simply say, "Good night," and the system will dim the lights, lock the doors, and adjust the thermostat to her preferred sleeping temperature.

Benefits:

1. Enhanced Comfort: Smart home automation adapts the environment to the user's preferences, creating a more comfortable living space.
2. Convenience: Automation simplifies everyday tasks, such as controlling lights, locks, and appliances, saving time and effort.
3. Energy Efficiency: Smart thermostats and energy monitoring systems help reduce energy consumption, leading to cost savings and environmental benefits.
4. Security: Automated security features enhance home safety by providing real-time alerts and remote monitoring.

Remote Control: The ability to control and monitor the home remotely provides peace of mind and convenience, especially while away from home.

5. Personalization: Smart systems can learn user preferences and adapt to individual routines.
6. Integration: Devices and systems can be seamlessly integrated, creating a cohesive and user-friendly experience.

Code implementation

What is Computer Vision? Computer vision is the interdisciplinary field that allows computers to acquire, process, and analyze visual data from the real world. It involves tasks such as image and video recognition, object detection, facial recognition, image segmentation, and more. Computer vision is used in various applications, including autonomous vehicles, medical imaging, surveillance, augmented reality, and robotics.

Key Concepts in Computer Vision:

1. **Image Processing:** This involves the manipulation and enhancement of images. Operations like filtering, edge detection, and noise reduction are essential for preparing images for analysis.
2. **Feature Extraction:** Identifying distinctive features within images, such as corners, edges, or blobs, which are essential for object recognition.
3. **Object Detection:** Locating and classifying objects within an image or video stream. This is widely used in applications like face detection and autonomous vehicles.
4. **Image Segmentation:** Dividing an image into meaningful segments, such as regions or objects, to facilitate further analysis.
5. **Deep Learning:** Deep neural networks, particularly Convolutional Neural Networks (CNNs), have revolutionized computer vision. They can automatically learn and extract features from images, enabling state-of-the-art results in various tasks.

Python Libraries for Computer Vision:

1. **OpenCV (Open Source Computer Vision Library):** OpenCV is a powerful and widely-used open-source computer vision library that provides a broad range of functions for image and video processing, object detection, and feature extraction.
2. **Pillow (PIL Fork):** Pillow is a Python Imaging Library that allows for basic image processing tasks such as opening, manipulating, and saving images.
3. **scikit-image:** Part of the scikit-learn ecosystem, scikit-image offers a collection of algorithms for image processing.
4. **TensorFlow and PyTorch:** These deep learning frameworks provide tools and pre-trained models for deep learning-based computer vision tasks.
5. **Dlib:** A library that includes tools for facial recognition, object detection, and shape prediction.

Getting Started: To begin your journey in computer vision with Python, you can install the necessary libraries and start with simple tasks like image manipulation and visualization. As you gain confidence, you can explore more complex tasks like object detection and image classification.

Python, with its easy-to-understand syntax and vast community support, is an excellent language for diving into the exciting world of computer vision. Whether you're a beginner or an experienced programmer, Python offers the flexibility to explore and develop computer vision applications that can solve real-world problems and create innovative solutions.

Arduino code for smart parking

```
class ParkingSpace:
  def _init_(self, number):
    self.number = number
    self.occupied = False

class SmartParkingSystem:
  def _init_(self):
    self.spaces = [ParkingSpace(i) for i in range(1, 3)]

  def check_availability(self):
    for space in self.spaces:
      status = "Occupied" if space.occupied else "Available"
      print(f"Parking Space {space.number}: {status}")

  def park_car(self, space_number):
    space = self.get_space_by_number(space_number)
    if space:
      if not space.occupied:
        space.occupied = True
        print(f"Car parked in Space {space.number}")
      else:
        print(f"Parking Space {space.number} is already occupied")
    else:
      print(f"Invalid parking space number: {space_number}")

  def remove_car(self, space_number):
    space = self.get_space_by_number(space_number)
```

```
        if space:
            if space.occupied:
                space.occupied = False
                print(f"Car removed from Space {space.number}")
            else:
                print(f"Parking Space {space.number} is already vacant")
        else:
            print(f"Invalid parking space number: {space_number}")

    def get_space_by_number(self, space_number):
        for space in self.spaces:
            if space.number == space_number:
                return space
        return None

# Usage example
parking_system = SmartParkingSystem()

# Simulate parking and removing cars
parking_system.check_availability()
parking_system.park_car(1)
parking_system.park_car(2)
parking_system.remove_car(1)
parking_system.check_availability()
```

HTML CODE :

```
<!DOCTYPE html>
<html lang="en">
<head>
    <meta charset="UTF-8">
    <meta name="viewport" content="width=device-width, initial-scale=1.0">
    <title>Smart Parking</title>
    <link rel="stylesheet" href="styles.css">
</head>
<body>
    <h1>Smart Parking System</h1>
    <div id="parkingSpaces"></div>
    <button id="checkAvailability">Check Availability</button>
    <label for="parkCarSpace">Park Car in Space:</label>
    <input type="number" id="parkCarSpace" min="1" max="10">
    <button id="parkCar">Park Car</button>
    <label for="removeCarSpace">Remove Car from Space:</label>
    <input type="number" id="removeCarSpace" min="1" max="10">
    <button id="removeCar">Remove Car</button>

    <script src="script.js"></script>
</body>
</html>
```

JAVA SCRIPT CODE

```
document.addEventListener('DOMContentLoaded', function() {
    const parkingSpacesDiv = document.getElementById('parkingSpaces');
    const checkAvailabilityButton =
document.getElementById('checkAvailability');
    const parkCarButton = document.getElementById('parkCar');
    const removeCarButton = document.getElementById('removeCar');
    const parkCarSpaceInput = document.getElementById('parkCarSpace');
    const removeCarSpaceInput =
document.getElementById('removeCarSpace');

    const parkingSpaces = [];
    for (let i = 1; i <= 10; i++) {
        parkingSpaces.push({ number: i, occupied: false });
    }

    function updateParkingSpaces() {
        parkingSpacesDiv.innerHTML = '';
        parkingSpaces.forEach(space => {
            const status = space.occupied ? 'Occupied' : 'Available';
            const spaceDiv = document.createElement('div');
            spaceDiv.textContent = `Space ${space.number}: ${status}`;
            parkingSpacesDiv.appendChild(spaceDiv);
        });
    }

    checkAvailabilityButton.addEventListener('click', function() {
        updateParkingSpaces();
    });
```

```
  parkCarButton.addEventListener('click', function() {
    const spaceNumber = parseInt(parkCarSpaceInput.value);
    const space = parkingSpaces.find(space => space.number ===
spaceNumber);
    if (space && !space.occupied) {
      space.occupied = true;
      updateParkingSpaces();
    } else {
      alert('Invalid parking space or space is already occupied.');
    }
  });

  removeCarButton.addEventListener('click', function() {
    const spaceNumber = parseInt(removeCarSpaceInput.value);
    const space = parkingSpaces.find(space => space.number ===
spaceNumber);
    if (space && space.occupied) {
      space.occupied = false;
      updateParkingSpaces();
    } else {
      alert('Invalid parking space or space is already vacant.');
    }
  });
});
```