**IOT Based Smart Parking System**

A Project report submitted in partial fulfillment of the requirements for the degree of B.E in

Computer Science Engineering

By

**M.JAYASHREE (513221104009)**

Under the supervision of the Professor & HOD department

|  |
| --- |
| parking space to detect occupancy. These sensors should be able to communicate data wirelessly.  2. \*Data Communication:\* Set up a reliable communication network to transmit data from the sensors to a central server. This could be Wi-Fi, LoRa, cellular, or a combination of technologies.  3. \*Central Server:\* Develop a central server or cloud platform to collect and process data from the sensors. This server should be capable of handling a large volume of real-time data.  4. \*User Interface:\* Create a user-friendly interface for both parking lot operators and users. This interface should show real-time parking space availability and may include mobile apps and websites.  5. \*Data Processing and Analysis:\* Use data analytics to process the information received from sensors. This can help in identifying trends, predicting peak usage times, and optimizing parking space allocation.  6. \*Mobile Apps:\* Develop mobile apps for users to check parking space availability, reserve spots, and receive navigation directions to available spaces.  7. \*Payment Integration:\* If required, integrate payment processing into the app to allow users to pay for parking through the system.  8. \*Security:\* Implement robust security measures to protect user data and the system from cyber threats.  9. \*Remote Monitoring and Maintenance:\* Implement remote monitoring and maintenance capabilities to ensure the system operates smoothly and address any issues promptly.  10. \*Scalability:\* Ensure the system can scale to accommodate additional sensors or parking lots as needed.  11. \*Sustainability:\* Consider the environmental impact and energy efficiency of the system, as well as the use of renewable energy sources where applicable. |

|  |  |
| --- | --- |
| 12. \*Sensor Deployment:\* Install IoT sensors (e.g., ultrasonic, magnetic, or infrared sensors) in each 12. \*Regulatory Compliance:\* Be aware of and comply with local regulations and privacy laws that may apply to your system. | |
| PROGRAM  # Import necessary libraries  from flask import Flask, request, jsonify  app = Flask(\_\_name)  # Create a dictionary to simulate parking space occupancy status  parking\_spaces = {  'A1': False,  'A2': False,  'B1': False,  'B2': False,  } | |

# Route to get parking space status

@app.route('/parking', methods=['GET'])

def get\_parking\_status():

return jsonify(parking\_spaces)

# Route to update parking space status

@app.route('/parking/update', methods=['POST'])

def update\_parking\_status():

data = request.get\_json()

space\_id = data.get('space\_id')

status = data.get('status')

if space\_id in parking\_spaces:

parking\_spaces[space\_id] = status

return "Parking status updated for space " + space\_id

else:

return "Invalid space ID"

if \_name\_ == '\_main\_':

app.run(debug=True)

```

In this example, we've created a simple Flask server that allows you to:

1. Get the status of parking spaces.

2. Update the status of parking spaces using a POST request.

You would need to deploy this server on a cloud platform or a local server.

For the IoT part, you would need to integrate sensor data to send POST requests to update parking space statuses. This code doesn't cover the hardware and sensor integration, which may involve various IoT platforms and technologies.

As for the web interface, you can create HTML templates and use JavaScript to display the parking space status in a user-friendly manner. This part involves front-end development, which is beyond the scope of this code example.

Output

1. When you access the route `'/parking'` with a GET request, the server will return a JSON response showing the simulated parking space occupancy status. For example:

{

"A1": false,

"A2": false,

"B1": false,

"B2": false

}

This indicates that all parking spaces are currently unoccupied (False).

2. When you access the route `'/parking/update'` with a POST request and provide valid data, it will update the parking space status accordingly. For example, if you send a POST request with JSON data:

json

{

"space\_id": "A1",

"status": true

}

The server will respond with:

Parking status updated for space A1

This means that parking space A1 is now marked as occupied (True).