

**KGiSL INSTITUTE OF TECHNOLOGY**

(Approved By AICTE, New Delhi, Affiliate to Anna University

Recognized by UGC, Accredited by NBA(IT)

265, KGISL Campus, Thudiyalur Road, Saravanampatti, Coimbatore-641035**.)**

**DEPARTMENT OF**

**ARTIFICIAL INTELLIGENCE AND DATA SCIENCE**

**NAAN MUDHALVAN - INTERNET OF THINGS**

**SMART PARKING**

**NAME: MADHUMITHA P**

**REG NO:** 711721243055

**NM ID:** au711721243055

**TEAM MENTOR:** Mr**.** Mohankumar M

**TEAM EVALUATOR:** Ms. Akilandeeshwari M

**SMART PARKING**

**Objectives:**

The Smart Parking project aims to develop a real-time parking availability system that can benefit both drivers and parking lot operators. The system will use IoT sensors to detect the occupancy status of each parking space and transmit this data to a central cloud-based server. A mobile app will then be developed to allow drivers to view the availability of parking spaces in real time and navigate to the nearest available space.

**IoT Sensor Setup**

The IoT sensor setup consists of the following components:

* Ultrasonic sensors: These sensors are used to detect the presence or absence of a vehicle in a parking space.
* Microcontrollers: These devices are used to process the data from the ultrasonic sensors and transmit it to the cloud-based server.
* Communication modules: These modules allow the microcontrollers to communicate with the cloud-based server.

The ultrasonic sensors are mounted on the ceiling or walls of the parking lot, with one sensor for each parking space. The microcontrollers are also mounted in the parking lot, and are connected to the ultrasonic sensors and communication modules.

**Mobile App Development:**

The mobile app is developed using a cross-platform development framework, such as Flutter or React Native. This allows the app to be deployed on both iOS and Android devices.

The app has the following features:

* Real-time parking availability map: The map shows the availability of parking spaces in real time, using different colors to indicate whether a space is available or occupied.
* Navigation: The app can provide turn-by-turn navigation to the nearest available parking space.
* Parking reservations: Drivers can use the app to reserve a parking space in advance.

**Raspberry Pi Integration:**

The Raspberry Pi is a low-cost, single-board computer that can be used to integrate the IoT sensors and the mobile app.

The Raspberry Pi is installed in the parking lot and is connected to the microcontrollers and the mobile app. The Raspberry Pi acts as a gateway between the IoT sensors and the cloud-based server. It also serves the mobile app with the real-time parking availability data.

**CODE:**

**import RPi.GPIO as GPIO**

**import time**

**import requests**

**ENTRY\_TRIG = 23**

**ENTRY\_ECHO = 24**

**EXIT\_TRIG = 25**

**EXIT\_ECHO = 8**

**GPIO.setmode(GPIO.BCM)**

**GPIO.setup(ENTRY\_TRIG, GPIO.OUT)**

**GPIO.setup(ENTRY\_ECHO, GPIO.IN)**

**GPIO.setup(EXIT\_TRIG, GPIO.OUT)**

**GPIO.setup(EXIT\_ECHO, GPIO.IN)**

**SERVER\_URL = 'http://your-server-url/api/update-parking'**

**def measure\_distance(trig\_pin, echo\_pin):**

**try:**

**while True:**

**entry\_distance = measure\_distance(ENTRY\_TRIG, ENTRY\_ECHO)**

**exit\_distance = measure\_distance(EXIT\_TRIG, EXIT\_ECHO)**

**data = {'entry\_distance': entry\_distance, 'exit\_distance': exit\_distance}**

**response = requests.post(SERVER\_URL, json=data)**

**time.sleep(5)**

**except KeyboardInterrupt:**

**GPIO.cleanup()**

**Code Implementation:**

The project is implemented using Python. The following are some of the key Python libraries that are used:

* **Flask:** A web framework that is used to develop the cloud-based server.
* **SocketIO:** A library that is used to implement real-time communication between the cloud-based server and the Raspberry Pi.
* **PyMySQL:** A library that is used to connect to and interact with a MySQL database.

**CODE:**

import React, { useState, useEffect } from 'react';

import { View, Text } from 'react-native';

function App() {

const [parkingData, setParkingData] = useState({ entry: 0, exit: 0 });

useEffect(() => {

// Fetch parking spot availability data from the server

fetch('http://your-server-url/api/get-parking-availability')

.then((response) => response.json())

.then((data) => setParkingData(data))

.catch((error) => console.error('Error fetching data:', error));

}, []);

return (

<View>

<Text>Entry Distance: {parkingData.entry} cm</Text>

<Text>Exit Distance: {parkingData.exit} cm</Text>

</View>

);

}

export default App;

**Benefits of the Real-Time Parking Availability System**

The real-time parking availability system can benefit both drivers and parking lot operators in the following ways:

**Drivers:**

* Reduced time spent searching for parking
* Improved fuel efficiency
* Reduced stress
* Increased convenience

**Parking lot operators:**

* Increased revenue
* Reduced traffic congestion in the parking lot
* Improved customer satisfaction
* Better understanding of parking usage patterns

**Submission**

The GitHub repository link containing the project's code and files is as follows:

[GitHub repository link]

**Instructions on How to Replicate the Project**

To replicate the project, follow these steps:

1. Clone the GitHub repository.
2. Install the required Python libraries.
3. Configure the IoT sensors and the Raspberry Pi.
4. Deploy the cloud-based server.
5. Develop and deploy the mobile app.

**How to Deploy IoT Sensors**

To deploy the IoT sensors, follow these steps:

1. Mount the ultrasonic sensors on the ceiling or walls of the parking lot, with one sensor for each parking space.
2. Connect the ultrasonic sensors to the microcontrollers.
3. Connect the microcontrollers to the communication modules.
4. Connect the communication modules to the Raspberry Pi.

**How to Develop and Deploy the Transit Information Platform**

To develop and deploy the transit information platform, follow these steps:

1. Install Python and the required Python libraries.
2. Develop the cloud-based server using Flask.
3. Deploy the cloud-based server to a cloud hosting provider, such as Amazon Web Services or Google Cloud Platform.

**How to Integrate the IoT Sensors and the Transit Information Platform Using Python**

To integrate the IoT sensors and the transit information platform using Python, follow these steps:

1. Install the PyMySQL library.
2. Connect to the MySQL database that is used by the transit information platform.
3. Write a Python script to read the data from the IoT sensors and insert it into the MySQL database.
4. Deploy the Python script on the Raspberry Pi.

**Conclusion:**

This document has described the Smart Parking project in detail, including its objectives, IoT sensor setup, mobile app development, Raspberry Pi integration.