SMART PARKING

ABSTRACT

The project involves integrating IoT sensors into public transportation vehicles to monitor ridership, track locations, and predict arrival times. The goal is to provide real-time transit information to the public through a public platform, enhancing the efficiency and quality of public transportation services. This paper investigates the likely advantages, key mechanical progressions, and useful utilizations of Shrewd Stopping in IoT, featuring its part in forming the eventual fate of metropolitan portability and working on the personal satisfaction in thickly populated regions. This project includes defining objectives, designing the IoT sensor system, developing the real-time transit information platform, and integrating them using IoT technology and Python.

Introduction:

In the present quick moving metropolitan scenes, the journey for productive and helpful stopping arrangements has turned into a squeezing concern. Clogged roads, restricted parking spots, and the steadily expanding number of vehicles out and about have led to a desperate requirement for inventive and smart leaving the executives frameworks. This need is met by the combination of two groundbreaking innovations: the Web of Things (IoT) and savvy stopping frameworks. The Web of Things, frequently condensed as IoT, is an earth-shattering organization of interconnected gadgets and items that impart and impart information to one another over the web. This innovation is upsetting the way in which we cooperate with our general surroundings, offering consistent availability and constant data sharing. Savvy stopping frameworks, then again, are intended to smooth out the stopping system by furnishing drivers with continuous data about accessible parking spots, limiting the time and dissatisfaction related with tracking down stopping. In this unique situation, the joining of IoT into stopping the board has brought forth the idea of "Brilliant Stopping in IoT." This imaginative methodology means to upgrade the stopping experience for the two drivers and parking area administrators by utilizing the force of IoT to make a more effective, easy to understand, and maintainable stopping biological system. Shrewd stopping in IoT not just works on the most common way of finding and holding parking spots yet additionally advances asset improvement, ecological manageability, and upgraded metropolitan preparation. This presentation makes way for a complete investigation of the crossing point among IoT and brilliant stopping frameworks, featuring the likely

advantages, mechanical progressions, and certifiable uses of this groundbreaking combination in metropolitan and rural conditions.

PROBLEM:

A popular shopping mall (Bharat) has parking place in first floor. Maximum 200 vehicles can park there. Is there parking space? isn't it? and how easy it can make parking? By using IoT (Internet of Things).

OBJECTIVES:

1.Real-time Parking Space Monitoring:

Develop a system that monitors parking spaces in real-time using IoT sensors to detect occupancy and vacancy accurately.

2.Mobile App Integration:

Create a user-friendly mobile application that allows drivers to access real-time parking availability information, make reservations, and navigate to available parking spots conveniently.

3.Efficient Parking Guidance:

Implement a smart parking guidance system that directs drivers to available parking spaces through clear

signage, mobile app notifications, and guidance within the parking facility.

IOT SENSOR DESIGNS:

1.Sensor Requirements:

- Determine the specific requirements for your IoT sensors, including the type of sensor (ultrasonic, infrared, magnetic) and the level of accuracy needed.

Fig: Ultrasonic sensor

2. Sensor Placement:

Identify optimal locations for sensor placement within parking spaces. Consider factors such as visibility, protection from weather, and accessibility for maintenance

3. Power Source:

- Decide on the power source for the sensors. Options include battery-powered sensors, solar-powered sensors, or wired sensors with a reliable power supply.

4. Sensor Calibration:

- Calibrate the sensors to accurately detect the presence or absence of vehicles in parking spaces. Test the sensors to ensure they provide reliable data.

5. Data Transmission:

- Establish a data transmission mechanism from the sensors to a central control system or a cloud-based platform. Ensure that data is transmitted securely and in real-time.

6. Connectivity:

Choose the communication technology for the sensors. Common options include Wi-Fi, LoRa WAN, or cellular connectivity.

- Using shopping mall cellular connectivity

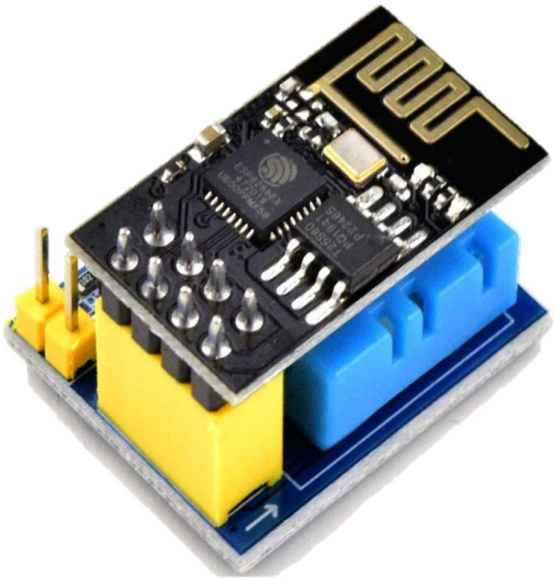


Fig: Wi-Fi-module

Real-Time Transit Information Platform:

Wireframing and Prototyping:

- Begin with wireframing and prototyping to create a basic layout and structure of the app. Use tools like tinkercat,wokewi

Map Integration:

- Integrate a map feature to display parking locations and availability. Use a mapping service like Google Maps or Map box to provide accurate and up-to-date information.

Notifications and Alerts:

Implement push notifications and alerts to inform users about available parking spots, reservation confirmations, and reminders.

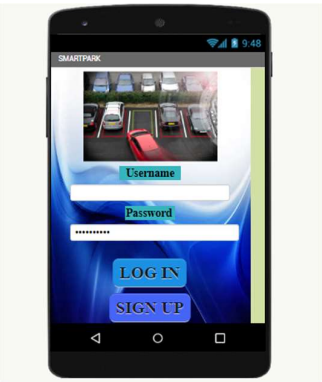


Fig: A dedicated parking app for shopping mall makes finding parking and easy for customers

Search and Filter Options:

- Include search and filter options to allow users to specify criteria such as location, price range, parking type (e.g., covered, open-air), and accessibility features (e.g., disabled parking).

CAMERA BASED PARKING:

- Camera-based smart parking in the Internet of Things (IoT) is a technology solution that leverages cameras and IoT devices to efficiently manage and monitor parking spaces.

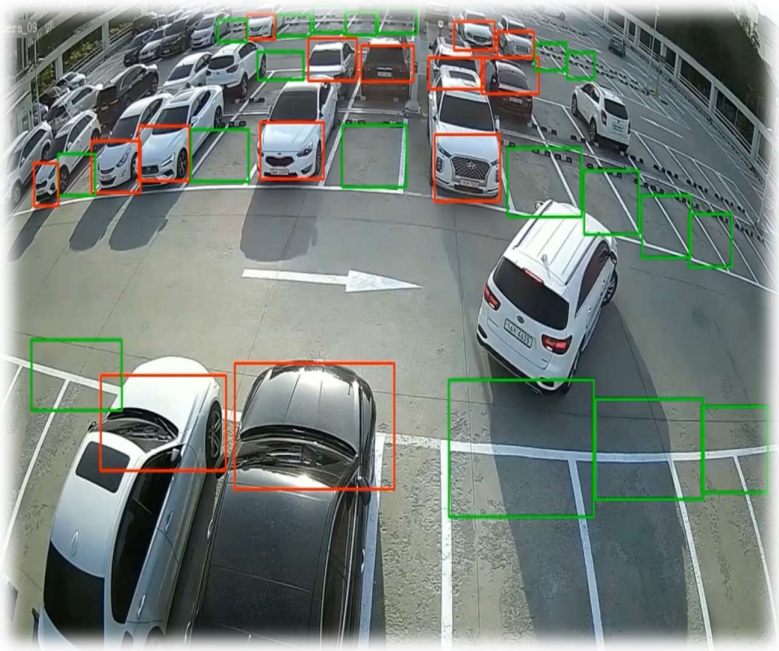


Fig: Camera based parking

Image completion:

Camera Setup:

Set up one or more cameras in the parking area to capture images or video streams.

Image Capture:

Continuously capture images or frames from the camera feed.

Preprocessing

Image Enhancement:

Apply image enhancement techniques such as brightness and contrast adjustment, histogram equalization, and noise reduction to improve image quality.

Geometric Correction:

Correct for camera distortion and perspective distortion to ensure accurate object detection.

Object Detection

Vehicle Detection:

Use object detection techniques (e.g., Haar Cascades, YOLO, SSD, Faster R-CNN) to detect vehicles in each frame of the camera feed.

Object Tracking:

Implement object tracking algorithms to track the detected vehicles across multiple frames.

Parking Space Detection

Define Regions of Interest (ROIs):

Define the areas of the camera feed where parking spaces are located.

Occupancy Classification:

Use image segmentation or classification techniques to classify each parking space within the ROIs as "occupied" or "vacant."

Visualization and User Interface

Visualization:

Overlay parking space occupancy information on the camera feed. Highlight vacant and occupied spaces.

User Interface:

Create a user interface to display parking space availability information to users.

Data Storage and Analysis

Data Storage:

Store parking space occupancy data in a database or file for historical analysis.

Data Analysis:

Analyze historical data to predict parking availability trends and make informed decisions.

Alerting (Optional)

Implement an alerting system to notify users when parking spaces become available or when the parking lot is full.

Deployment

Deploy your parking space detection system in the parking area and ensure it runs continuously.

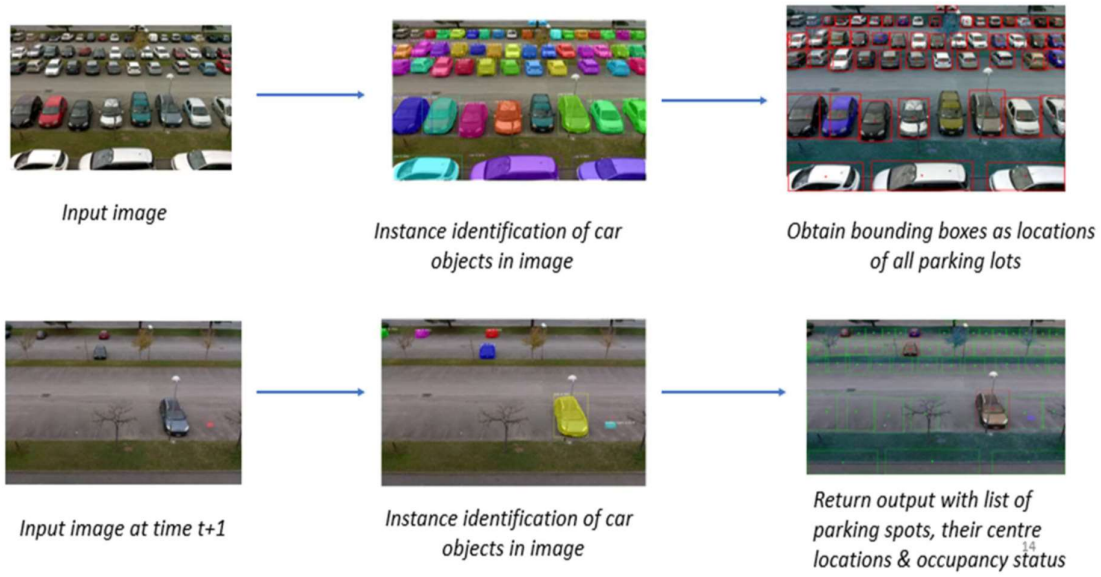
It's important to note that developing a robust parking space detection system can be a complex task that may require expertise in computer vision, machine learning, and software development. Additionally, you may need to consider factors like camera placement, lighting conditions, and environmental factors when designing your system.

PYTHON CV CODE:

```
import cv2
import numpy as np
import matplotlib.pyplot as plt
# Load the pre-trained vehicle detection classifier (Haar Cascade)
car_cascade =cv2.CascadeClassifier('/content/haarcascade_car.xml')
# Create a VideoCapture object to capture video from a file or
camera (you can specify the source)
video_source = '/content/pexels_videos_2406459 (2160p).mp4' #
Replace with your video path or use 0 for the default camera
cap = cv2.VideoCapture(video_source)
# Define parking space boundaries (x, y, width, height)
parking_spaces = [(60, 60, 50, 50),(60, 60, 50, 50),(60, 60, 50, 50),(60,
60, 50, 50),(60, 60, 50, 50)] # Example: Two parking spaces
while True:
# Read a frame from the video feed
ret, frame = cap.read()
if not ret:
break
# Convert the frame to grayscale for vehicle detection
gray = cv2.cvtColor(frame, cv2.COLOR_BGR2GRAY)
# Detect cars in the frame using the car_cascade classifier
cars = car_cascade.detectMultiScale(gray, scaleFactor=1.1,
minNeighbors=5, minSize=(50, 50))
# Initialize a list to track parking space occupancy
occupancy_status = ['Vacant'] * len(parking_spaces)
# Draw rectangles around detected cars and check parking
space occupancy
for i, (x, y, w, h) in enumerate(cars):
cv2.rectangle(frame, (x, y), (x + w, y + h), (0, 255, 0), 2)
# Check if the detected car is within any parking space
for j, (px, py, pw, ph) in enumerate(parking_spaces):
if x >= px and y >= py and x + w <= px + pw and y + h <=
py + ph:
occupancy_status[j] = 'Occupied'
```

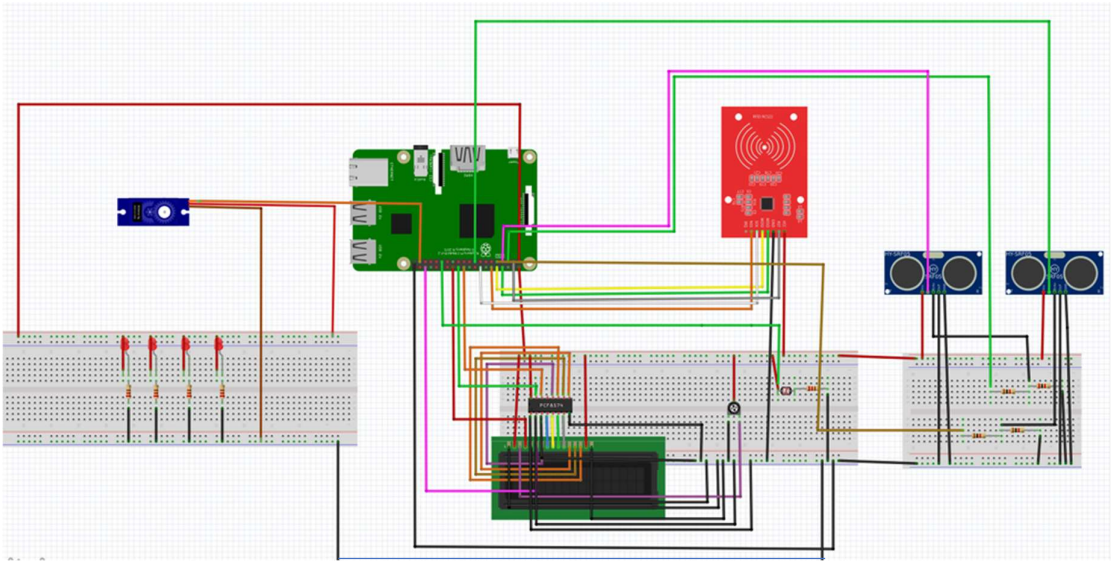
```
# Print parking space availability
for j, status in enumerate(occupancy_status):
print(f'Space {j + 1}: {status}')
# Display the frame with detected cars and parking space
occupancy using Matplotlib
frame_rgb = cv2.cvtColor(frame, cv2.COLOR_BGR2RGB)
plt.imshow(frame_rgb)
plt.axis('off')
plt.show()
# Exit the loop when the 'q' key is pressed
if cv2.waitKey(1) & 0xFF == ord('q'):
break
# Release the VideoCapture and close any OpenCV windows
cap.release()
cv2.destroyAllWindows()
```

PROTOTYPE:



```
Space 1: Vacant
Space 2: Vacant
Space 3: Vacant
Space 4: Vacant
Space 5: Vacant
```

Circuit diagram for park Monitoring system:



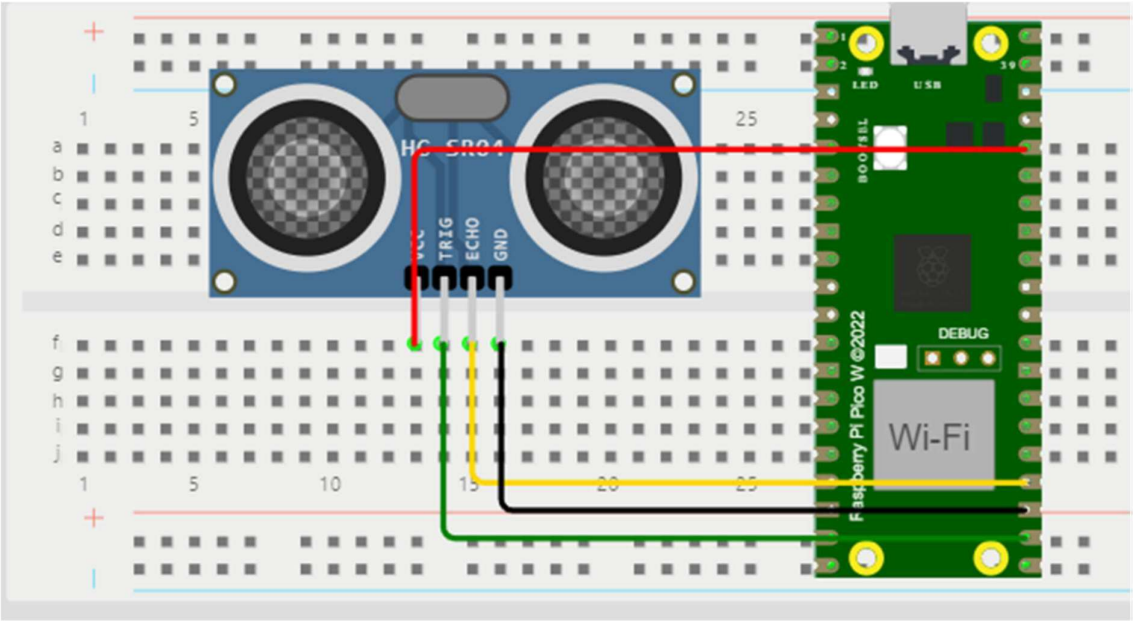
Integration Approach:

Sensor Connectivity:

- Ensure that the IoT sensors are correctly installed and connected to the Raspberry Pi. Sensors may use various communication protocols, such as GPIO, UART, I2C, or SPI, depending on the sensor type.

Raspberry Pi Configuration:

Set up the Raspberry Pi with the necessary operating system and software libraries. Ensure that the Raspberry Pi has internet connectivity via Wi-Fi or Ethernet



Python scripts for raspberry module:

```
import RPi.GPIO as GPIO
import time
import gspread
from oauth2client.service_account import ServiceAccountCredentials
# Set up Ultrasonic Sensor GPIO pins
TRIG_PIN = 17
ECHO_PIN = 18
# Initialize GPIO
```

```
GPIO.setmode(GPIO.BCM)
GPIO.setup(TRIG_PIN, GPIO.OUT)
GPIO.setup(ECHO_PIN, GPIO.IN)
# Define Google Sheets-related variables
credentials_file = 'smart-parkingiot-095b74826f5a' # Replace with your
credentials file
google_sheets_name = 'parkingdata' # Replace with your Google Sheets
document name
sheet_name = 'Sheet1' # Replace with your sheet name
# Initialize Google Sheets
scope = ['https://www.googleapis.com/auth/spreadsheets']
credentials =
ServiceAccountCredentials.from_json_keyfile_name(credentials_file, scope)
gc = gspread.authorize(credentials)
worksheet = gc.open(google_sheets_name).worksheet(sheet_name)
try:
while True:
# Measure distance using ultrasonic sensor
GPIO.output(TRIG_PIN, False)
time.sleep(2)
GPIO.output(TRIG_PIN, True)
time.sleep(0.00001)
GPIO.output(TRIG_PIN, False)
while GPIO.input(ECHO_PIN) == 0:
pulse_start = time.time()
while GPIO.input(ECHO_PIN) == 1:
pulse_end = time.time()
pulse_duration = pulse_end - pulse_start
distance = pulse_duration * 17150 # Speed of sound (343 m/s)
# Get current timestamp
timestamp = time.strftime("%Y-%m-%d %H:%M:%S")
# Determine occupancy status based on distance threshold
distance_threshold = 50
occupancy_status = "Occupied" if distance < distance_threshold else
"Vacant"
# Log data to Google Sheets
worksheet.append_row([timestamp, distance, occupancy_status])
print(f"Timestamp: {timestamp}, Distance: {distance} cm, Occupancy:
{occupancy_status}")
time.sleep(5) # Adjust the time interval as needed
```

```
except KeyboardInterrupt:
GPIO.cleanup()
```

Web java script code:

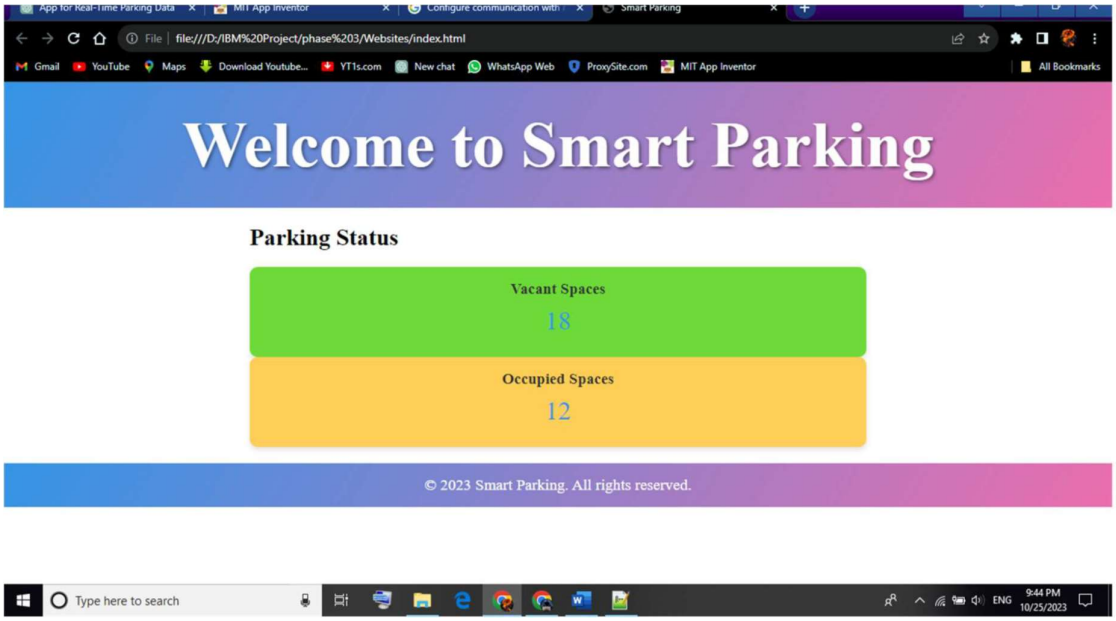
```
// JavaScript code to process CSV data and update the website
Papa.parse('parkingdata.csv', {
download: true,
header: true,
skipEmptyLines: true,
dynamicTyping: true,
complete: function(results) {
const data = results.data;
// Initialize counts
let vacantCount = 0;
let occupiedCount = 0;
// Process the data and calculate counts
data.forEach((entry) => {
const status = entry['status'];
if (status === 'vacant') {
vacantCount++;
} else if (status === 'occupied') {
occupiedCount++;
}
});
// Update the HTML with the counts
document.getElementById('vacantCount').textContent = vacantCount;
document.getElementById('occupiedCount').textContent =occupiedCount;
}
});
```

Tracking and Visualization:

Get right of entry to your Google Sheets document to reveal parking area
occupancy.
Use Google Sheets capabilities for statistics analysis and visualization.

Hardware Setup:

Securely mount the ultrasonic sensors inside the parking place.
Make sure they have got an unobstructed view of the parking spaces.



Data Transmission:

Establish a data transmission mechanism to send the processed sensor data to a central server or cloud platform. You can use protocols like HTTP, MQTT, or WebSocket for this purpose.

Data's are transmit in the shopping mall main server.

Real-Time Updates to Mobile App:

Ensure that the mobile app can periodically request and receive real-time parking availability updates from the server or cloud platform.

Mobile is developed using python.

Make app for parking .

ANDROID APP:

Fig: login & sign up

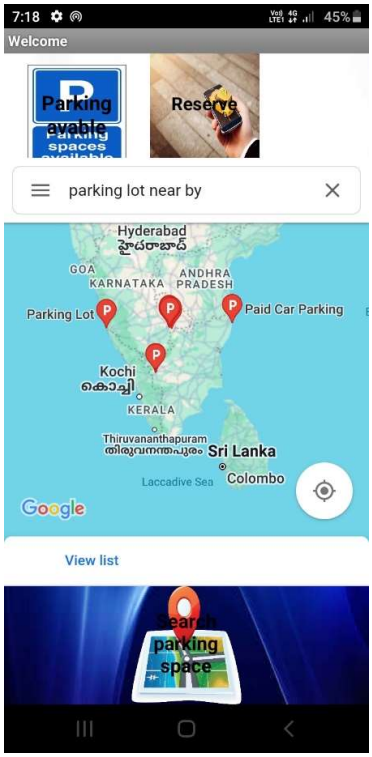


Fig: app welcome interface

Fig: parking data

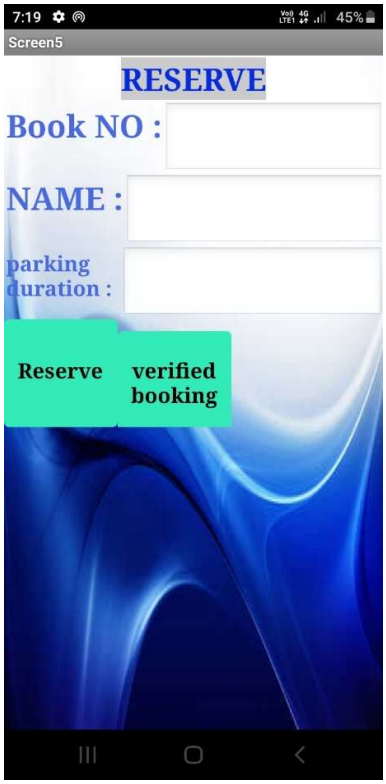


Fig: parking reserved

An ongoing stopping accessibility framework is an innovation driven arrangement that can essentially help drivers and mitigate stopping issues by giving expert data about accessible parking spots in a given region. This is the carefully guarded secret and the benefits it offers:

Decreased Pressure and Time Reserve funds:

Drivers can check the accessibility of parking spots continuously, lessening the pressure and dissatisfaction related with looking for a parking space.

This prompts time reserve funds, as drivers can make a beeline for a spot with certainty instead of surrounding around erratically.

Diminished Blockage and Outflows:

Drivers looking for stopping add to gridlock and discharges. Continuous accessibility data diminishes the time spent looking, in this manner diminishing gridlock and the related natural effect.

Cost Investment funds:

Drivers can track down stopping choices that fit their financial plans, lessening the probability of overpaying or causing fines for stopping infringement.

They can pick reasonable stopping choices and be educated about any limits or advancements.

Expanded Availability:

The framework can give data about open parking spots for individuals with incapacities, further developing availability for all individuals from the local area.

Streamlined Course Arranging:

Drivers can utilize route applications that coordinate constant stopping information to design their courses. This empowers them to pick objections in light of stopping accessibility, lessening the probability of showing up at a jam-packed area with no accessible stopping.

Worked on Metropolitan Preparation:

City organizers can utilize information from constant stopping accessibility frameworks to arrive at informed conclusions about foundation improvement and metropolitan preparation. They can distinguish regions with popularity for stopping and do whatever it takes to further develop stopping framework.

Monetary Advantages:

Neighborhood organizations can profit from expanded pedestrian activity, as potential clients are bound to visit a region on the off chance that they realize they can find stopping without any problem. This can support the neighborhood economy.

Diminished Carbon Impression:

Less vehicles revolving around for leaving implies less fuel utilization and lower fossil fuel byproducts, adding to natural supportability.

Safety:

Decreased surrounding for stopping can prompt better street security, as drivers are more averse to participate in diverted or forceful driving ways of behaving when they are focused on because of stopping hardships.

Consumer loyalty:

The accommodation of knowing where to stop and the time saved in the process can further develop generally speaking consumer loyalty, making a region more appealing to occupants and guests.

Information for Future Preparation:

Stopping accessibility information can be gathered and broke down to recognize examples and patterns. This data can assist metropolitan organizers with coming to informed conclusions about stopping the executives, extension, or rebuilding.

Reconciliation with Public Transportation:

Continuous stopping accessibility frameworks can be incorporated with public transportation choices, assisting workers with settling on informed decisions about consolidating driving with public travel.

All in all, a constant stopping accessibility framework benefits drivers by diminishing pressure, saving time, and assisting them with tracking down savvy and helpful stopping. It likewise emphatically affects gridlock, the climate, nearby organizations, and metropolitan preparation, making it an important device for tending to stopping issues in present day urban communities.

Conclusion

The development of the Internet of Things and cloud technology opens up new opportunities for smart cities. Smart parking has always been the backbone of building smart cities. IoT-based smart parking system offers real-time slots, parking procedures, information and improves users' ability to save time on proper parking. It helps to solve growing traffic congestion concerns. As for future work, users can book parking in a remote location. GPS, reservations, and license plate scanners can be included in the future.

THANK YOU