**SMART PARKING SYSTEM**

**PHASE 5: PROJECT DOCUMENTATION AND SUBMISSION**

**INTRODUCTION**

The Smart Parking IoT project aims to address the common urban problem of parking congestion by leveraging Internet of Things (IoT) technology to create a more efficient and convenient parking system. This document outlines what we have understood about the project and provides a step-by-step approach for solving the problem.

**UNDERSTANDING THE PROBLEM**

1. PARKING CONGESTION

Parking congestion is a widespread issue in urban areas, causing frustration and wasting time for drivers. Finding an available parking space can be challenging and often leads to increased traffic and pollution due to vehicles circling in search of parking.

2. SOLUTION OVERVIEW

The proposed solution involves implementing IoT sensors and a centralized system to monitor and manage parking spaces in real-time. These sensors will collect data on parking space availability and communicate this information to a mobile app or digital displays for drivers.

**APPROACH TO SOLVING THE PROBLEM**

1. DEFINE PROJECT SCOPE

The first step is to clearly define the scope of the project. This includes determining the size of the parking area to be covered, the number of parking spaces to be monitored, and the level of detail required in reporting (e.g., individual space availability or zone-based availability).

2. SENSOR DEPLOYMENT

a. Sensor Selection

Select suitable IoT sensors for detecting the presence of vehicles in parking spaces. Common sensor types include ultrasonic, infrared, or magnetic sensors. Consider factors such as cost, accuracy, and power consumption.

b. Sensor Installation

Install sensors in each parking space. Ensure proper calibration and connectivity to a central data collection system. Establish a robust communication network (e.g., Wi-Fi, LoRaWAN, or cellular) to transmit data.

3. DATA COLLECTION AND PROCESSING

a. Data Aggregation

Collect data from the deployed sensors, including real-time occupancy status, timestamp, and location of each parking space. Aggregate this data into a central database or cloud platform.

b. Data Processing

Implement data processing algorithms to filter and analyze sensor data. Determine parking space availability based on the collected information.

4. USER INTERFACE DEVELOPMENT

a. Mobile App

Develop a user-friendly mobile application for drivers. The app should display real-time parking space availability, navigation to available spaces, and notifications/alerts.

b. Digital Displays

Install digital displays at key locations within the parking area to inform drivers of space availability and guide them to open spots.

5. DATA VISUALIZATION AND ANALYTICS

Implement data visualization tools and analytics to provide insights into parking utilization patterns, peak hours, and trends. This can help optimize the parking system further.

6. USER ADOPTION AND EDUCATION

Launch a marketing and education campaign to encourage drivers to use the smart parking system. Provide user guides and support to address any questions or concerns.

7. MAINTENANCE AND MONITORING

Establish a routine maintenance schedule for sensors and the IoT infrastructure. Implement remote monitoring to detect and address issues promptly.

8. SCALABILITY

Design the system with scalability in mind, allowing for easy expansion to cover additional parking areas or spaces as needed.

9. DATA SECURITY AND PRIVACY

Implement robust security measures to protect sensor data and user information. Comply with relevant data privacy regulations.

10. CONTINUOUS IMPROVEMENT

Regularly evaluate the system's performance and gather feedback from users to identify areas for improvement and optimization.

**Hardware Setup:**

**Wiring Components:**

Connect the LEDs to the appropriate GPIO pins on the ESP32 to simulate traffic signals.

Ensure you have the necessary resistors in place to limit current flow through the LEDs.

**Writing the Code:**

//Define GPIO pins for traffic lights

const int redPin = 2;

const int yellowPin = 3;

const int greenPin = 4;

void setup() {

// Initialize GPIO pins as OUTPUT

pinMode(redPin, OUTPUT);

pinMode(yellowPin, OUTPUT);

pinMode(greenPin, OUTPUT);

}

void loop() {

// Simulate traffic management logic here

// For example, change traffic lights based on a timer or sensor input

// Turn on red light

digitalWrite(redPin, HIGH);

digitalWrite(yellowPin, LOW);

digitalWrite(greenPin, LOW);

delay(5000); // Red light for 5 seconds

// Turn on green light

digitalWrite(redPin, LOW);

digitalWrite(yellowPin, LOW);

digitalWrite(greenPin, HIGH);

delay(5000); // Green light for 5 seconds

// Simulate traffic management logic here

// You can add more complex logic as needed

}

**HARDWARE DESIGN**

Hardware equipment that we need in order to build the project are given below:

⦁ Arduino UNO

⦁ Ultrasonic Sensor

⦁ LCD Screen

⦁ Bread Board

⦁ Power Supply

⦁ Connecting Wires

**Block Diagram**

Block Diagram of our proposed system is given below :

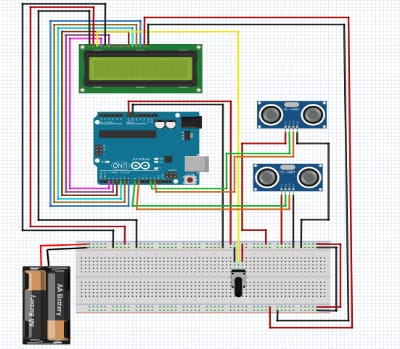


Figure-1 : Block diagram of Smart Car Parking System

**FLOW CHART**

The flow chart includes how the system works. The program flow chart is given below:

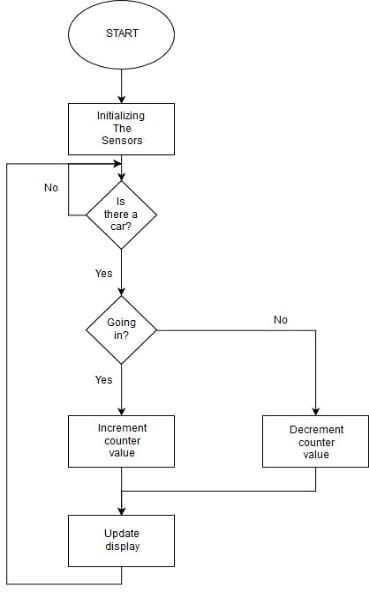


Figure-2 : Flow chart of Smart Car Parking System

**CODE**

from flask import Flask, render\_template, request, redirect, url\_for, session

from datetime import datetime

import requests, json, re, random, math, pickle

import firebase\_admin

from firebase\_admin import db

# Initialize Firebase app

firebase\_admin.initialize\_app(options={'databaseURL': 'https://vitask.firebaseio.com/'})

app = Flask(\_\_name\_\_)

# Change this to your secret key (can be anything, it's for extra protection)

app.secret\_key = 'canada$God7972#'

ref = db.reference('vitask')

slot\_a = {"x":50,"y":0,"price":15,"num":1}

slot\_b = {"x":0,"y":50,"price":20,"num":2}

slot\_c = {"x":50,"y":100,"price":25,"num":3}

slot\_d = {"x":100,"y":50, "price":30,"num":4}

# Homepage

@app.route('/')

def index():

return render\_template('index.html')

# Dashboard

@app.route('/dashboard')

def dashboard():

# Check if user is loggedin

if 'loggedin' in session:

if(session['parking\_space']=='A'):

url = "https://thingspeak.com/channels/1208300/field/1.json"

elif(session['parking\_space']=='B'):

url = "https://thingspeak.com/channels/1208301/field/1.json"

elif(session['parking\_space']=='C'):

url = "https://thingspeak.com/channels/1208302/field/1.json"

else:

url = "https://thingspeak.com/channels/1208303/field/1.json"

try:

response = requests.get(url)

data = json.loads(response.text)

record = data['feeds'][-1]['field1']

if record is None:

record = 0

except:

record = 0

slots = int(record)

lastupdated\_nonformatted = datetime.now()

# dd/mm/YY H:M:S

lastupdated = lastupdated\_nonformatted.strftime("%d/%m/%Y %H:%M:%S")

booked\_slots = []

empty\_slots = []

for i in range(0,slots):

empty\_slots.append(i+1)

for i in range(slots,10):

booked\_slots.append(i+1)

return render\_template('dashboard.html', username = session['username'], booked\_slots = booked\_slots, empty\_slots = empty\_slots, lastupdated = lastupdated, parking\_space = session['parking\_space'], distance = session['distance'], rate = session['rate'])

return redirect(url\_for('login'))

# Thingspeak write for parking spaces (1-4)

def reservedslots(parking\_space):

temp = ref.child("parkfind").child("reserve").get()

reservations = []

if temp is not None:

for key in temp:

if(temp[key]["parkingSpace"]==parking\_space):

reservations.append(temp[key])

total = str(len(reservations))

# Select the parking space

if(parking\_space=='A'):

url = "https://api.thingspeak.com/update?api\_key=7LHBQ6TZCKKWAYND&field1="+total

elif(parking\_space=='B'):

url = "https://api.thingspeak.com/update?api\_key=YBW96SSU1G299SSS&field1="+total

elif(parking\_space=='C'):

url = "https://api.thingspeak.com/update?api\_key=TOXRGRVZA1I28SIP&field1="+total

else:

url = "https://api.thingspeak.com/update?api\_key=I2L87XB3VT8P7L8V&field1="+total

try:

response = requests.get(url)

except Exception as e:

print(e)

# Distance Function

def calc\_distance(x1,y1,x2,y2):

distance = math.sqrt(((x2-x1)\*\*2)+((y2-y1)\*\*2))

return distance

# Reservation System

@app.route('/reservation')

def reservation():

# Check if user is loggedin

if 'loggedin' in session:

# Check if account exists using Firebase

reservations = []

temp = ref.child("parkfind").child("reserve").get()

if temp is not None:

for key in temp:

if(temp[key]["parkingSpace"]==session['parking\_space']):

reservations.append(temp[key])

reservedslots(session['parking\_space'])

return render\_template('reservation.html', username = session['username'], reservations = reservations)

return redirect(url\_for('login'))

# Reservation System

@app.route('/submit\_reservation', methods=['GET', 'POST'])

def submit\_reservation():

# Check if user is loggedin

if 'loggedin' in session:

if request.method == 'POST' and 'carMark' in request.form and 'carNumber' in request.form:

# Create variables for easy access

carMark = request.form['carMark']

carNumber = request.form['carNumber']

parking\_space = session['parking\_space']

username = session['username']

# Add the reservation

try:

tut\_ref = ref.child("parkfind").child("reserve")

tut\_ref.push({

'carMark': carMark,

'carNumber': carNumber,

'parkingSpace': parking\_space,

'username': username

})

except Exception as e:

print(e)

return redirect(url\_for('reservation'))

return redirect(url\_for('login'))

#Login

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

def login():

# Output message if something goes wrong...

msg = ''

# Check if "username" and "password" POST requests exist (user submitted form)

if request.method == 'POST' and 'username' in request.form and 'password' in request.form:

# Create variables for easy access

username = request.form['username']

password = request.form['password']

# Check if account exists using Firebase

account = None

temp = ref.child("parkfind").child("users").get()

for key in temp:

if(temp[key]["username"]==username):

account = temp[key]

# Fetch one record and return result

# Login successful

if(password==account["password"]):

session['loggedin'] = True

session['username'] = account["username"]

# Generate Coordinates

session['x'] = random.randint(0,100)

session['y'] = random.randint(0,100)

# Predict parking spot

filename = 'finalized\_model.sav'

distance\_a = calc\_distance(session['x'],session['y'],slot\_a['x'],slot\_a['y'])

distance\_b = calc\_distance(session['x'],session['y'],slot\_b['x'],slot\_b['y'])

distance\_c = calc\_distance(session['x'],session['y'],slot\_c['x'],slot\_c['y'])

distance\_d = calc\_distance(session['x'],session['y'],slot\_d['x'],slot\_d['y'])

data = [[distance\_a,distance\_b,distance\_c,distance\_d,15,20,25,30]]

classifier = pickle.load(open(filename, 'rb'))

parking\_space = classifier.predict(data)[0]

all\_distance = {'A': round(distance\_a), 'B': round(distance\_b), 'C': round(distance\_c), 'D': round(distance\_d)}

all\_rates = {'A': slot\_a['price'], 'B': slot\_b['price'], 'C': slot\_c['price'], 'D': slot\_d['price']}

session['parking\_space'] = parking\_space

session['distance'] = all\_distance[parking\_space]

session['rate'] = all\_rates[parking\_space]

# Redirect to dashboard

return redirect(url\_for('dashboard'))

else:

# Account doesnt exist or username/password incorrect

msg = 'Incorrect username/password!'

# Show the login form with message (if any)

return render\_template('login.html', msg=msg)

# Register

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

def register():

# Output message if something goes wrong...

msg = ''

# Check if "username", "password" and "email" POST requests exist (user submitted form)

if request.method == 'POST' and 'username' in request.form and 'password' in request.form:

# Create variables for easy access

username = request.form['username']

password = request.form['password']

# Check if account exists using Firebase

snapshot = "Default"

temp = ref.child("parkfind").child("users").get()

for key in temp:

if(temp[key]["username"]==username):

snapshot = None

if snapshot is None:

account = True

else:

account = False

# If account exists show error and validation checks

if account:

msg = 'Account already exists!'

elif not re.match(r'[A-Za-z0-9]+', username):

msg = 'Username must contain only characters and numbers!'

elif not username or not password:

msg = 'Please fill out the form!'

else:

# Account doesnt exists and the form data is valid, now insert new account into users table

try:

tut\_ref = ref.child("parkfind").child("users")

tut\_ref.push({

'username': username,

'password': password

})

msg = 'You have successfully registered!'

except Exception as e:

print(e)

print(msg)

return render\_template('login.html', msg=msg)

# Web Logout

@app.route('/logout')

def logout():

session.pop('username', None)

session.pop('loggedin', None)

session.pop('x', 0)

session.pop('y', 0)

session.pop('parking\_space', None)

session.pop('distance', 0)

session.pop('rate', 0)

return redirect(url\_for('index'))

# Run the Flask Server

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

app.run(debug=True)

Naives bayes for model training

# Naive Bayes

# Importing the libraries

import numpy as np

import matplotlib.pyplot as plt

import pandas as pd

import pickle

# Importing the dataset

dataset = pd.read\_csv('data.csv')

X = dataset.iloc[:, [0, 1, 2, 3, 4, 5, 6, 7]].values

y = dataset.iloc[:, 12].values

# Splitting the dataset into the Training set and Test set

from sklearn.model\_selection import train\_test\_split

X\_train, X\_test, y\_train, y\_test = train\_test\_split(X, y, test\_size = 0.25, random\_state = 0)

# Feature Scaling

from sklearn.preprocessing import StandardScaler

sc = StandardScaler()

X\_train = sc.fit\_transform(X\_train)

X\_test = sc.transform(X\_test)

# Fitting Naive Bayes to the Training set

from sklearn.naive\_bayes import GaussianNB

classifier = GaussianNB()

classifier.fit(X\_train, y\_train)

# Predicting the Test set results

y\_pred = classifier.predict(X\_test)

# Making the Confusion Matrix

from sklearn.metrics import confusion\_matrix

cm = confusion\_matrix(y\_test, y\_pred)

filename = 'finalized\_model.sav'

pickle.dump(classifier, open(filename, 'wb'))

print(classifier.predict([[10,50,60,70,15,20,25,30]]))

**WEB APPLICATION FOR THE SMART PARKING**

<!DOCTYPE html>

<html>

<head>

<title>that's my spot.com</title>

<meta name="viewport" content="width=device-width, initial-scale=1">

<meta http-equiv="Content-Type" content="text/html; charset=utf-8" />

<link href="css/style3.css" rel="stylesheet" type="text/css" media="all" />

<link rel="stylesheet" href="http://maxcdn.bootstrapcdn.com/bootstrap/3.3.5/css/bootstrap.min.css">

<link rel="css/bootstrap.min.css">

<link href="css/bootstrap.css" rel='stylesheet' type='text/css' />

<link rel="stylesheet" type="text/css" href="css/animate.css"/>

<script src="js/jquery.min.js"></script>

<script type="text/javascript" src="js/move-top.js"> </script>

<script src="js/bootstrap.min.js"></script>

<script type="text/javascript" src="js/wow.js"></script>

<script src="js/wow.min.js"></script>

<script type="text/javascript" src="js/easing.js"></script>

<script type="text/javascript" src="js/main.js"></script>

<script type="text/javascript">

jQuery(document).ready(function($) {

$(".scroll").click(function(event){

event.preventDefault();

$('html,body').animate({scrollTop:$(this.hash).offset().top},900);

});

});

</script><script>

new WOW().init();

</script> <style>

.carousel-inner > .item > img,

.carousel-inner > .item > a > img {

width: 100%;

margin: auto;

}

</style>

</head>

<body>

<!-- Header Section -->

<header>

<div class="container">

<div class="logo pull-left animated wow fadeInLeft">

<img src="images/logo.jpg" height="80px" width="65px" alt="" title="" >

<p style="font-style:italic;color:white;font-size:12pt;" class="animated wow fadeInLeft" data-wow-delay=".2s">That's my spot</p>

</div>

<nav class="pull-right">

<ul class="list-unstyled">

<li class="animated wow fadeInLeft" data-wow-delay="0s"><a href="Sign\_in\_customer.html">Sign In</a></li>

<li class="animated wow fadeInLeft" data-wow-delay=".2s"><a href="#gallery">Gallery</a></li>

<li class="animated wow fadeInLeft" data-wow-delay=".2s"><a href="#contact">Contact</a></li>

</ul>

</nav>

<span class="burger\_icon">menu</span> </div>

</header>

<!-- End Header Section -->

<div class="banner">

<div class="container">

<div class="banner-info">

<br>

<br>

<h3>Book your spot !!!!!!</h3>

<br>

<br>

<br>

<!-- Trigger the modal with a button -->

<div align="center" > <button type="button" class="btn btn-warning btn-lg" data-toggle="modal" data-target="#myModal">Sign Up</button>

</div>

<!-- Modal -->

<div class="modal fade" id="myModal" role="dialog">

<div class="modal-dialog">

<!-- Modal content-->

<div class="modal-content">

<div class="modal-header">

<button type="button" class="close" data-dismiss="modal">&times;</button>

h4 class="modal-title" style="color:SandyBrown;text-align:center;font-size:2em;

font-weight:600;">Sign Up</h4>

</div>

<div class="modal-body">

<div align="center">

<p>Sign Up and get started! </p>

</div>

<div align="center" >

<p> Just one step away from making the best parking facility!!!</p>

<p> <a href="Sign\_up\_customer.html">Sign Up</a></p>

<p>already signed up? <br> <a href="Sign\_in\_customer.html">Sign In</a>

</div>

</div>

</div>

</div>

<div class="modal-footer">

<button type="button" class="btn btn-default" data-dismiss="modal">Close</button>

</div>

</div> <br><br><br>

<h3>Just one step away from availing the best facility!!</h3>

</div>

</div>

</div>

<div id="specials" class="specials" >

<div class="contact-text" >

<h3>Our Specialties &nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp; <img src="images/logo.jpg" width="60px" height="70px"></h3>

<span></span>

</div>

<div class="container" >

<br>

<div id="myCarousel" class="col-md-6 carousel slide" data-ride="carousel">

<!-- Indicators -->

<ol class="carousel-indicators">

<li data-target="#myCarousel" data-slide-to="0" class="active"></li>

<li data-target="#myCarousel" data-slide-to="1"></li>

<li data-target="#myCarousel" data-slide-to="2"></li>

<li data-target="#myCarousel" data-slide-to="3"></li>

</ol>

<!-- Wrapper for slides -->

<div class="carousel-inner" role="listbox">

<div class="item active">

<img src="images/bg.jpg" width="300" height="400"> </div>

<div class="item">

<img src="images/lot1.jpg" width="300" height="400">

</div>

<div class="item">

<img src="images/lot2.jpg" width="300" height="400">

</div>

<div class="item">

<img src="images/lot3.jpg" width="300" height="400">

</div>

</div>

<!-- Left and right controls -->

<a class="left carousel-control" href="#myCarousel" role="button" data-slide="prev">

<span class="glyphicon glyphicon-chevron-left" aria-hidden="true"></span>

<span class="sr-only">Previous</span>

</a>

<a class="right carousel-control" href="#myCarousel" role="button" data-slide="next">

<span class="glyphicon glyphicon-chevron-right" aria-hidden="true"></span>

<span class="sr-only">Next</span>

</a>

</div>

<div id="myCarousel1" class="col-md-6 carousel slide" data-ride="carousel">

<!-- for customer reviews slides-->

<script src="js/responsiveslides.min.js"></script>

<script>

$(function () {

$("#slider2").responsiveSlides({

auto: true,

pager: true,

speed: 300,

namespace: "callbacks",

});

});

</script>

<div class="pricing">

<div class="container">

<div class="pricing-text">

<h3>Our Customer Reviews</h3>

<span></span>

</div>

<!-- start slider -->

<div class="pricing-grids">

<div class="slider">

<ul class="rslides" id="slider2">

<li><div class="col-md-6 pricing-plans"><p>Customer 1 review.<br> This parking system is the best available parking system in the world.

<br> These guys are really awesome!!!!!!!!</p>

<div class="pic1"><img src="images/pr2.png" alt=""/></div>

<div class="pic-info"><h5>John Doe</h5>

<a href="#">Mail id ,<br> address</a>

</div>

<div class="clearfix"></div>

</div>

<div class="col-md-6 pricing-plans">

<p>Customer 2 review.<br>that's my spot.com is really fantaaaastic.

<br>I got OMG feeling after using this. nowadays i use this app to park my kid's cycle also. hahaha</p>

<div class="pic1">

<img src="images/m2.jpg" alt=""/>

</div>

<div class="pic-info">

<h5>John Doe</h5>

<a href="#">Mail id ,<br> address</a>

</div>

<div class="clearfix"></div>

</div>

<div class="clearfix"></div>

</li>

<li>

<div class="col-md-6 pricing-plans">

<p>Customer 3 review.<br> that's my spot.com is really fantaaaastic.

<br>I got OMG feeling after using this. nowadays i use this app to park my kid's cycle also. hahaha</p>

<div class="pic1">

<img src="images/pr1.png" alt=""/></div>

<div class="pic-info">

<h5>John Doe</h5>

<a href="#">Mail id ,<br> address</a>

</div>

<div class="clearfix"></div>

</div>

<div class="col-md-6 pricing-plans">

<p>Customer 4 review.<br>This parking system is the best available parking system in the world.

<br> These guys are really awesome!!!!!!!!</p>

<div class="pic1">

<img src="images/pr2.png" alt=""/>

</div>

<div class="pic-info">

<h5>John Doe</h5>

<a href="#">Mail id ,<br> address</a>

</div>

<div class="clearfix"></div>

</div>

<div class="clearfix"></div>

</li>

</ul>

</div>

<!-- end slider -->

</div>

</div>

</div>

<!---->

<!-- gallery-->

<div class="gallery" id="gallery">

<div class="gallery-sub">

<h3> Gallery</h3>

</div>

<div class="today-grids">

<div class="col-md-4 today-grid test1">

<img src="images/bg.jpg" class="img-responsive" alt="/">

<div class="textbox">

<h4 style="color:DarkSlateGrey;

font-size:2.2em;

text-align:center;">Yeshwantpur</h4>

<p style="color:SandyBrown;

font-size:1em;

text-align:center;">The parking lot which has very neat infrastructure and ratings. …</p></div></div>

<div class="col-md-4 today-grid test1">

<img src="images/bg2.jpg" class="img-responsive" alt="/">

<div class="textbox">

<h4 style="color:DarkSlateGrey;

font-size:2.2em;

text-align:center;">Indiranagar</h4>

font-size:1em;

text-align:center;">Has lot of parking lots. Avaliable very near to each other.…</p>

</div>

</div>

<div class="col-md-4 today-grid test1">

<img src="images/lot1.jpg" class="img-responsive" alt="/">

<div class="textbox">

<h4 style="color:DarkSlateGrey;

font-size:2.2em;

text-align:center;">Malleshwaram</h4>

font-size:1em;

text-align:center;">Has lot of parking lots. Avaliable very near to each other.……</p>

</div>

</div>

<div class="col-md-4 today-grid test1">

<img src="images/lot2.jpg" class="img-responsive" alt="/">

<div class="textbox">

<h4 style="color:DarkSlateGrey;

font-size:2.2em;

text-align:center;">Yelahanka</h4>

<p style="color:SandyBrown;

font-size:1em;

text-align:center;">This parking system is the best available parking system in the world.</p></div>

</div>

<div class="col-md-4 today-grid test1">

<img src="images/lot5.jpg" class="img-responsive" alt="/">

<div class="textbox">

<h4 style="color:DarkSlateGrey;

font-size:2.2em;

text-align:center;">Yeshwantpur</h4>

<p style="color:SandyBrown;

font-size:1em;

text-align:center;">This parking system is the best available parking system in the world..</p>

</div>

</div>

<div class="col-md-4 today-grid test1">

<img src="images/lot4.jpg" class="img-responsive" alt="/">

<div class="textbox">

<h4 style="color:DarkSlateGrey;

font-size:2.2em;

text-align:center;">Malleshwaram </h4>

<p style="color:SandyBrown;

font-size:1em;

text-align:center;">This parking system is the best available parking system in the world.</p>

</div>

</div>

<div class="clearfix"></div>

</div>

</div>

<!-- gallery over-->

<div id="contact" class="contact">

<div class="container">

<div class="contact-text">

<h3>Contact us</h3>

<span></span>

</div>

<div class="contact-grids">

<div class="col-md-4 contact-grid text-center wow bounceIn" data-wow-delay="0.4s">

<div class="icon1"></div>

<p>Phone no: 9535701911</p>

</div>

<div class="col-md-4 contact-grid text-center wow bounceIn" data-wow-delay="0.4s">

<div class="icon2"></div>

<p> MSRIT</p>

<p> Mattikere, Bangalore</p>

</div>

<div class="col-md-4 contact-grid text-center wow bounceIn" data-wow-delay="0.4s">

<div class="icon3"></div>

<a href="mailto:example.com">thatsmyspot.com</a>

</div>

<div class="clearfix"></div>

</div>

</div>

</div>

<!---->

<div class="footer text-center">

<div class="container">

<a class="wow bounceIn" data-wow-delay="0.5s" href="index.html"><img src="images/logo.jpg" width="67px" height="90px" alt=""/></a>

<br>

<br>

<br>

<p class="wow bounceIn" data-wow-delay="0.4s">Copyright &copy; That's my spot All rights reserved | Design by CodeBlooded </p><br>

</div>

</div>

<script type="text/javascript">

$(document).ready(function() {

$().UItoTop({ easingType: 'easeOutQuart' });

});

</script>

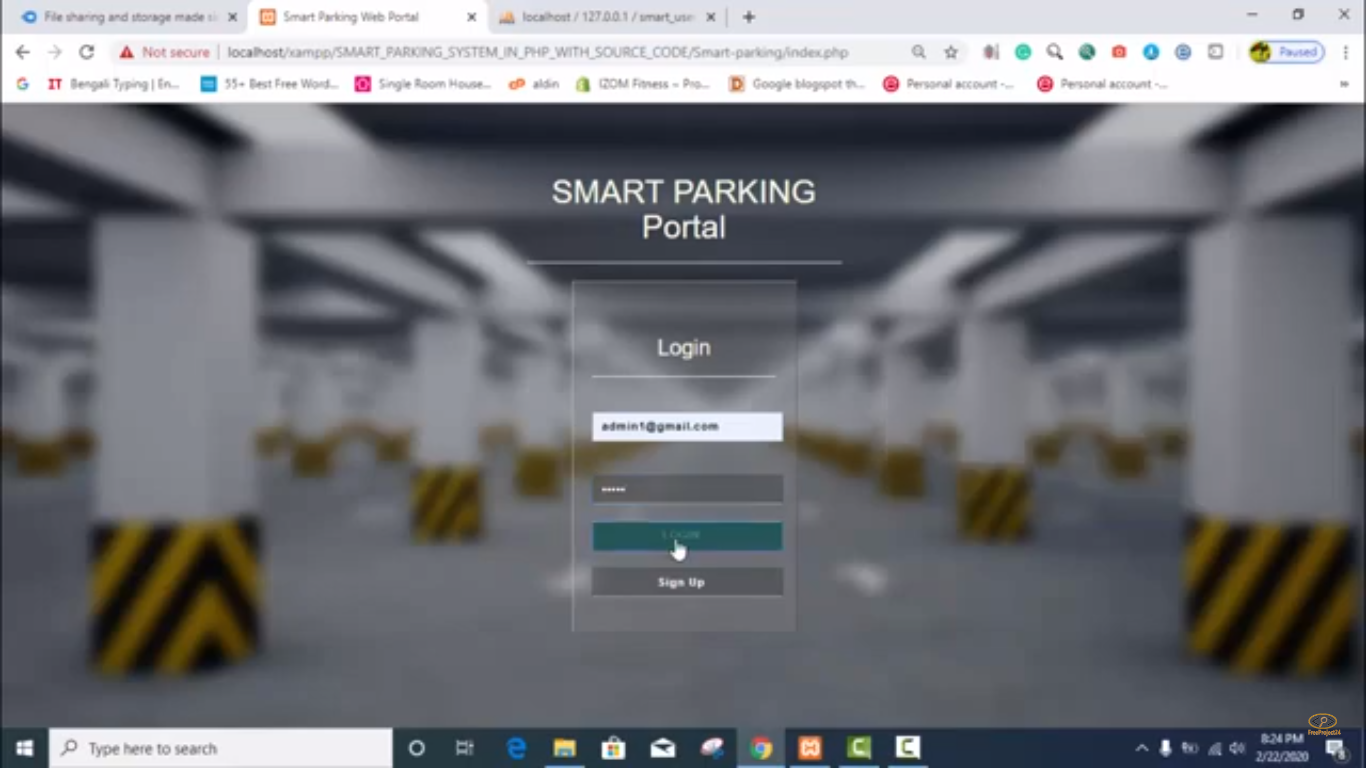
<a href="#to-top" id="toTop" style="display: block;"> <span id="toTopHover" style="opacity: 1;"> </span></a>

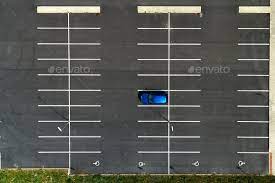
<!---->

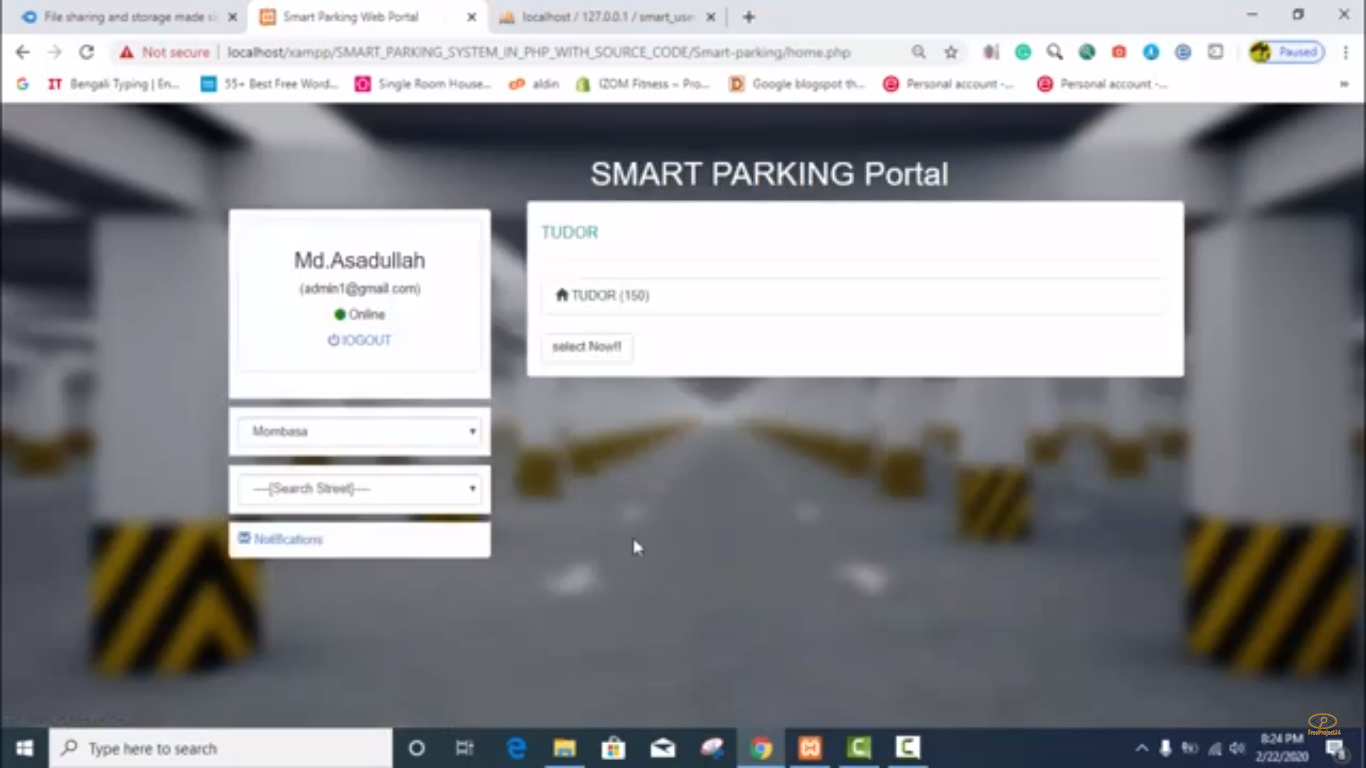
</body>

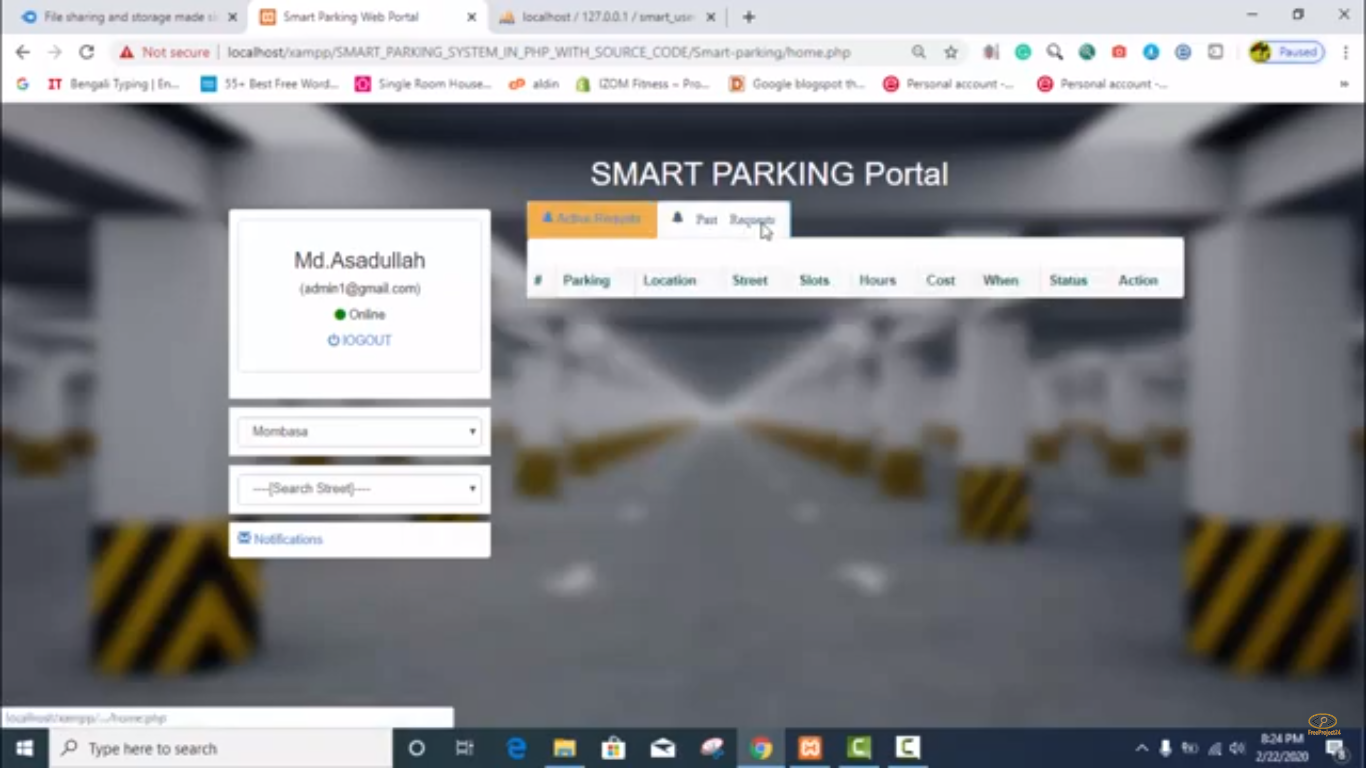
</html>

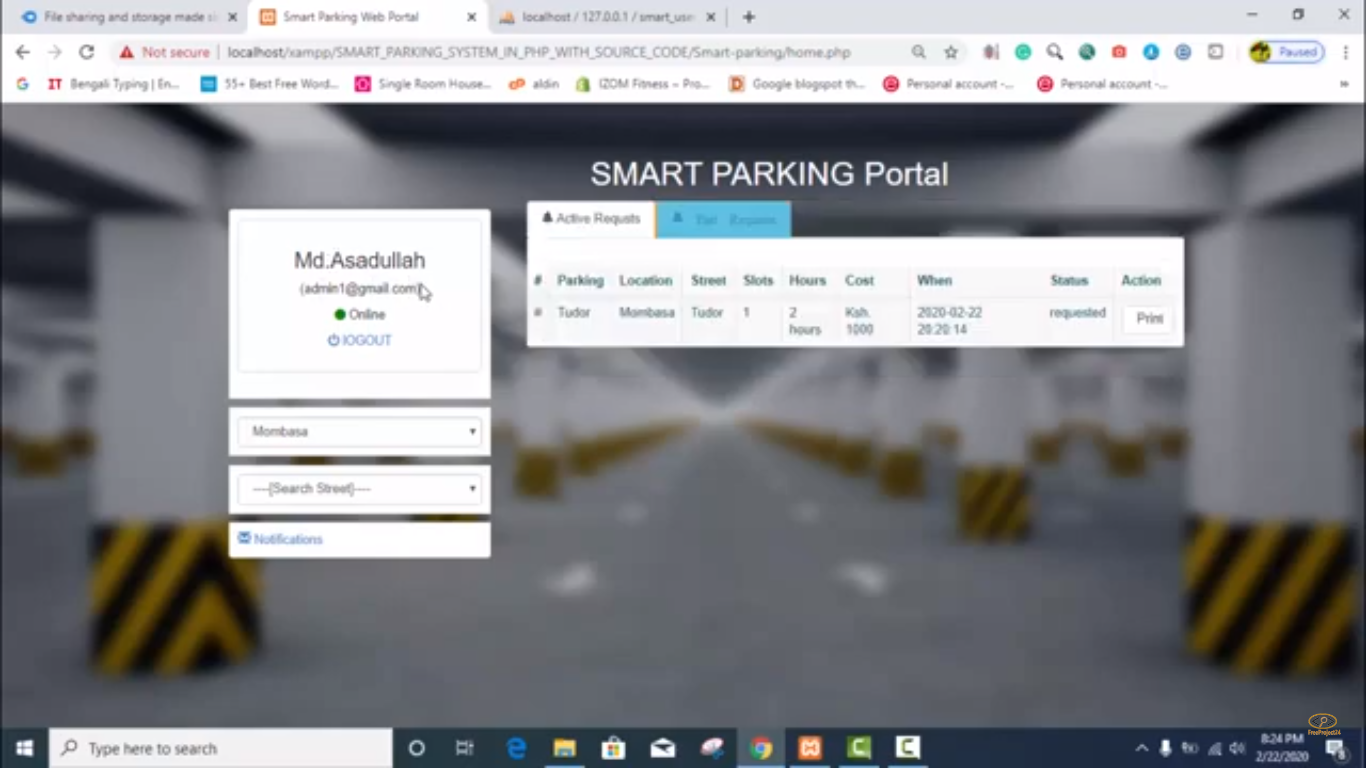
**OUTPUT**

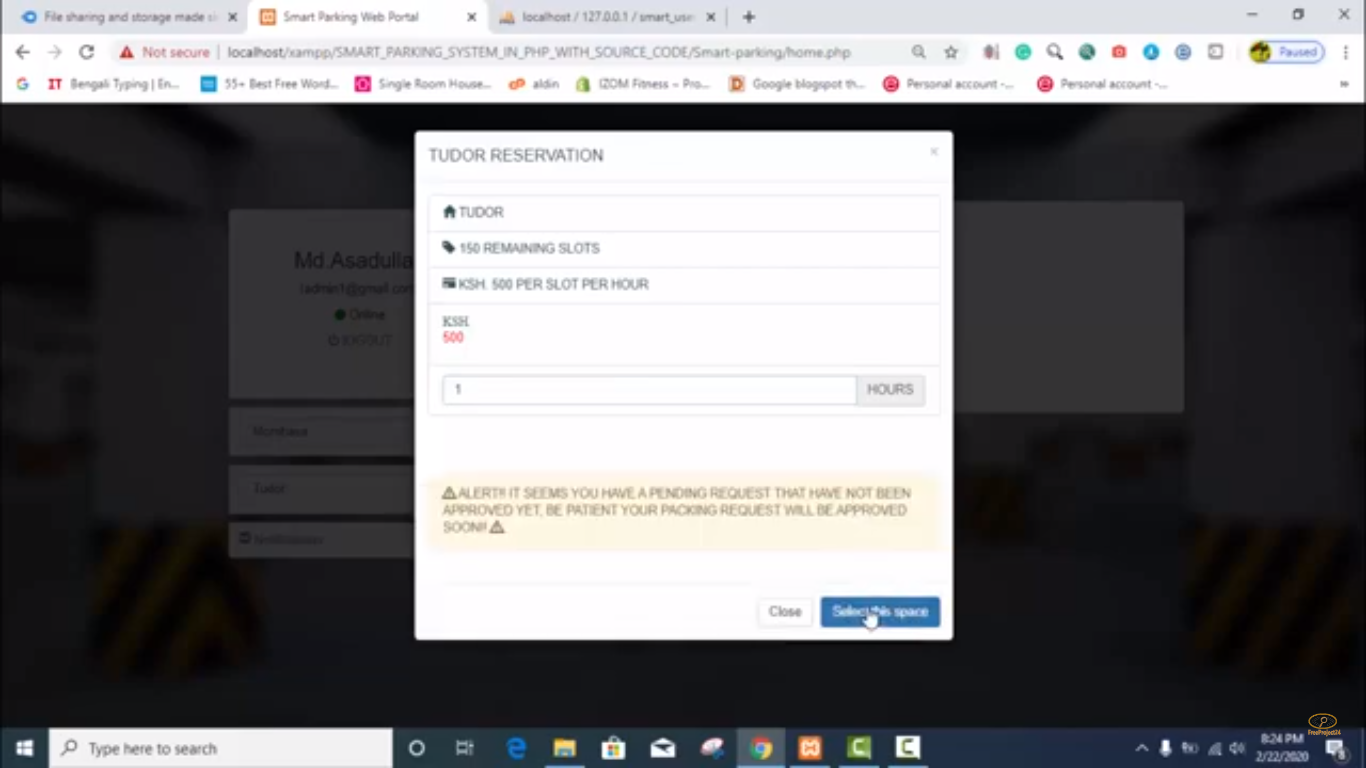














**CONCLUSION**

**The Smart Parking IoT project is a promising solution to alleviate parking congestion in urban areas. By deploying IoT sensors, developing user-friendly interfaces, and implementing data analytics, we aim to create a convenient and efficient parking experience for drivers while also contributing to reduced traffic congestion and pollution in urban environments. This approach provides a comprehensive roadmap for successfully solving the problem of parking congestion using IoT technology.**