<ECO CHARGE>
A
Mini Project Report
Submitted in partial fulfilment of the Requirements for the award of the Degree of
BACHELOR OF ENGINEERING
IN
INFORMATION TECHNOLOGY
By
<Mohammed Nomaan><1602-22-737-094>
<Mohammed Muneeb Ur Rahaman><1602-22-737-093>
<Kogooru Bhargavi><1602-22-737-071>
Department of Information Technology Vasavi College of Engineering (Autonomous)
ACCREDITED BY NAAC WITH 'A++' GRADE
(Affiliated to Osmania University and Approved by AICTE) Ibrahimbagh, Hyderabad-31
2023
Vasavi College of Engineering (Autonomous)
ACCREDITED BY NAAC WITH 'A++' GRADE
(Affiliated to Osmania University and Approved by AICTE) Hyderabad-500 031
Department of Information Technology
DECLARATION BY THE CANDIDATE
We, Mohammed Nomaan, Mohammed Muneeb Ur Rahman, and Kogooru Bhargavi,
bearing hall ticket numbers, 1602-22-737-094, 1602-22-737-093 and 1602-22-737-071, hereby declare that the project report enti
This is a record of bonafide work carried out by us and the results embodied in this project report have not been submitted to any
Mohammed Nomaan
1602-22-737-094
< Mohammed Muneeb Ur Rahaman >
1602-22-737-093
Kogooru Bhargavi
1602-22-737-071
(Faculty In-Charge)
(Head,Dept of IT)
ACKNOWLEDGMENT:
We extend our sincere thanks to Dr. S. V. Ramana, Principal, Vasavi College of Engineering for his encouragement.
We express our sincere gratitude to Dr. K. Ram Mohan Rao, Professor & Head, Department of Information Technology, Vasavi C
We also want to thank and convey our gratitude towards our mini project coordinators MS S.Aruna and
MS Soume Sanyal for guiding us in understanding the process of project development & giving us timely suggestions at every pha
We would also like to sincerely thank the project reviewers for their valuable inputs and suggestions.
Abstract :
The provided HTML template represents a page displaying information about the charging cost for an electric vehicle. It includes c
Here's a simplified abstraction:
Styling:
Defines styling rules for paragraphs, spans, and a directions button.
Specifies the appearance of the container with a title.
Content:
Displays charging-related information using paragraphs and spans.
Calculates and displays the total charging cost.
Interaction:
Includes a Google Maps Directions button with a link generated based on the charging station's name and location.

This template aims to provide a visually appealing and informative representation of charging-related details, coupled with an inter

Table Of Contents

Introduction:

Introduction to Eco Charge: A Flask-Powered EV Charging Station Locator

Eco Charge is a robust web application designed to simplify the user experience of finding and interacting with electric vehicle (EV

Key Features and Database Integration:

The backbone of Eco Charge lies in its database models—ChargingStation and ChargerType—which capture vital information abo

Interactive Maps and User-Friendly Interface:

Eco Charge goes beyond a conventional EV charging locator by incorporating an interactive map powered by Leaflet.js. Users can

Technology

Flask:

Flask is a web framework for Python that simplifies the process of building web applications. It is lightweight, modular, and provide

Flask-SQLAlchemy:

Flask-SQLAlchemy is an extension for Flask that simplifies database integration using SQLAlchemy, a popular SQL toolkit and Ob

SQLite:

SQLite is a lightweight, serverless relational database management system (RDBMS). In this code, SQLite serves as the backend

Leaflet.js:

Leaflet.js is a JavaScript library for interactive maps. The code incorporates Leaflet.js to create an interactive map displaying the lo

HTML and CSS:

HTML (Hypertext Markup Language) is used for structuring the content of web pages, while CSS (Cascading Style Sheets) is emp

JavaScript:

JavaScript is a scripting language that enables dynamic interactions within web pages. The code utilizes JavaScript to implement

Google Maps API:

Although not explicitly mentioned in the provided code, the mention of Google Maps in the 'charging_cost.html' template suggests

These technologies collectively contribute to the development of a feature-rich and user-friendly EV charging station locator applic

Implementation(Code):

```
# app.py
from flask import Flask, render_template, request, redirect, url_for
from flask_sqlalchemy import SQLAlchemy
app = Flask(_name_)
app.config['SQLALCHEMY_DATABASE_URI'] = 'sqlite:///charging_stations.db'
with app.app_context():
db = SQLAlchemy(app)
class ChargingStation(db.Model):
id = db.Column(db.Integer, primary_key=True)
name = db.Column(db.String(255), nullable=False)
location = db.Column(db.String(255), nullable=False)
charger_types = db.relationship('ChargerType', backref='charging_station', lazy=True)
class ChargerType(db.Model):
id = db.Column(db.Integer, primary_key=True)
name = db.Column(db.String(50), nullable=False)
cost_per_kwh = db.Column(db.Float, nullable=False)
station_id = db.Column(db.Integer, db.ForeignKey('charging_station.id'), nullable=False)
def create_default_data():
# Check if data already exists
if ChargingStation.query.filter_by(name='Tata Power Charging Station', location='Banjara Hills').first() is None:
with app.app_context():
# Create instances and add to the database
```

```
station1 = ChargingStation(name='Tata Power Charging Station', location='Banjara Hills')
charger_type1 = ChargerType(name='CCS', cost_per_kwh=18, charging_station=station1)
charger_type2 = ChargerType(name='AC', cost_per_kwh=6, charging_station=station1)
station2 = ChargingStation(name='Tata Charging Station', location='Somajiguda')
charger_type3 = ChargerType(name='CCS', cost_per_kwh=20, charging_station=station2)
charger_type4 = ChargerType(name='AC', cost_per_kwh=7, charging_station=station2)
# Add instances to the session
db.session.add(station1)
db.session.add(charger_type1)
db.session.add(charger_type2)
db.session.add(station2)
db.session.add(charger_type3)
db.session.add(charger_type4)
# Commit the changes
db.session.commit()
@app.route('/')
def index():
charging_stations = ChargingStation.query.all()
return render_template('index_leaf.html', charging_stations=charging_stations)
@app.route('/find_charging_stations', methods=['POST'])
def find_charging_stations():
location = request.form.get('location')
if location:
# Query charging stations based on the entered location
charging_stations = ChargingStation.query.filter(ChargingStation.location.ilike(f'%{location}%')).all()
return render_template('index_leaf.html', charging_stations=charging_stations, entered_location=location)
else:
# If no location entered, redirect back to the index page
return redirect(url_for('index'))
@app.route('/select_charger/<int:station_id>', methods=['GET', 'POST'])
def select_charger(station_id):
station = ChargingStation.query.get_or_404(station_id)
if request.method == 'POST':
charger_type_id = request.form['charger_type']
units = request.form['units']
charger_type = ChargerType.query.get_or_404(charger_type_id)
# Calculate charging cost based on selected charger type, units, and cost_per_kwh
charging_cost = float(units) * charger_type.cost_per_kwh
return render_template('charging_cost.html', station=station, charger_type=charger_type, units=units, charging_cost=charging_co
return render_template('select_charger.html', station=station)
if _name_ == '_main_':
with app.app_context():
db.create_all()
create_default_data()
# Run the Flask application in debug mode
app.run(debug=True)
<!-- index_leaf.html -->
<!DOCTYPE html>
```

```
<html lang="en">
<head>
<meta charset="UTF-8">
<meta name="viewport" content="width=device-width, initial-scale=1.0">
<title>Welcome to Eco Charge</title>
<!-- Add your other head elements here -->
<!-- Add a link to your custom CSS file if needed -->
<link rel="stylesheet" href="styles.css">
<style>
body {
font-family: 'Arial', sans-serif;
margin: 0;
padding: 0;
background-color: #f4f4f4;
}
header {
background-color: #333;
color: #fff;
padding: 20px;
text-align: center;
}
#logo {
height: 120px;
width: 120px;
position: absolute; /* Position the logo absolutely */
top: 10px; /* Distance from the top */
left: 300px; /* Distance from the left */
}
h1 {
margin: 0;
}
form {
margin-top: 20px;
}
label {
color: #fff;
}
input {
padding: 10px;
margin-right: 10px;
}
button {
padding: 10px 20px;
background-color: #4caf50;
color: #fff;
border: none;
cursor: pointer;
}
```

```
button:hover {
background-color: #45a049;
}
#map {
height: 400px;
width: 100%;
}
</style>
</head>
<body>
<header>
<!-- Add a heading for the page -->
<h1>Welcome to Eco Charge</h1>
<!-- Add your logo here -->
<img src="{{ url_for('static', filename='image.jpeg') }}"  id="logo">
<!-- Search bar for finding charging stations -->
<form id="searchForm">
<label for="search">Find Charging Stations:</label>
<input id="search" type="text" name="location" placeholder="Enter location">
<button type="button" onclick="findChargingStations()">Find Charging Stations</button>
</form>
</header>
<!-- Embed Leaflet.js map as before -->
<div id="map"></div>
<!-- Include Leaflet.js from CDN -->
<link rel="stylesheet" href="https://unpkg.com/leaflet/dist/leaflet.css" />
<script src="https://unpkg.com/leaflet/dist/leaflet.js"></script>
<script>
var map;
var chargingStations = [
{ id: 1, name: 'Tata Power Charging Station', location: 'Banjara Hills', lat: 17.4065, lon: 78.4772 },
{ id: 2, name: 'Tata Charging Station', location: 'Somajiguda', lat: 17.4256, lon: 78.4346 },
// Add more charging stations as needed
];
document.addEventListener('DOMContentLoaded', function () {
// Initialize the map with default center coordinates
map = L.map('map').setView([17.4065, 78.4772], 10);
// Use OpenStreetMap as the base layer
L.tileLayer('https://{s}.tile.openstreetmap.org/{z}/{x}/{y}.png', {
attribution: '&copy; OpenStreetMap contributors'
}).addTo(map);
// Add markers for charging stations
for (var i = 0; i < chargingStations.length; i++) {
var station = chargingStations[i];
var marker = L.marker([station.lat, station.lon]).addTo(map)
.bindPopup("<b>" + station.name + "</b><br>Click to select charger");
// Attach click event to the marker
marker.on('popupopen', function (e) {
```

```
// Retrieve information about the clicked charging station
var clickedStation = chargingStations.find(s => s.lat === e.popup._latlng.lat && s.lon === e.popup._latlng.lng);
if (clickedStation) {
// Redirect to the page for selecting a charger (modify the URL as needed)
window.location.href = "/select_charger/" + clickedStation.id;
}
});
}
});
function findChargingStations() {
var locationInput = document.getElementById('search');
var location = locationInput.value;
// Geocode the entered location using a geocoding service (adjust the service URL as needed)
fetch('https://nominatim.openstreetmap.org/search?format=json&q=' + location)
.then(response => response.json())
.then(data => {
if (data.length > 0) {
var foundLocation = data[0];
var foundLat = parseFloat(foundLocation.lat);
var foundLon = parseFloat(foundLocation.lon);
// Update the map view to the found location
map.setView([foundLat, foundLon], 12);
} else {
alert('Location not found. Please enter a valid location.');
}
})
.catch(error => {
console.error('Error fetching geolocation data:', error);
});
}
</script>
</body>
</html>
<!-- select_charger.html -->
<!DOCTYPE html>
<html lang="en">
<head>
<meta charset="UTF-8">
<meta name="viewport" content="width=device-width, initial-scale=1.0">
<title>Select Charger Type and Units</title>
<style>
body {
font-family: 'Arial', sans-serif;
background-color: #f4f4f4;
margin: 0;
padding: 0;
}
h2 {
```

```
color: #333;
}
form {
max-width: 400px;
margin: 20px auto;
padding: 20px;
background-color: #fff;
border-radius: 8px;
box-shadow: 0 0 10px rgba(0, 0, 0, 0.1);
}
label {
display: block;
margin-bottom: 8px;
color: #555;
}
select, input {
width: 100%;
padding: 10px;
margin-bottom: 16px;
box-sizing: border-box;
}
button {
background-color: #4caf50;
color: #fff;
padding: 10px 20px;
border: none;
border-radius: 4px;
cursor: pointer;
font-size: 16px;
}
button:hover {
background-color: #45a049;
}
</style>
<!-- ... (other head elements) ... -->
</head>
<body>
<h2>Select Charger Type and Units</h2>
<form action="{{ url_for('select_charger', station_id=station.id) }}" method="post">
<label for="charger_type">Charger Type:</label>
<select id="charger_type" name="charger_type" required>
{% for charger_type in station.charger_types %}
<option value="{{ charger_type.id }}">{{ charger_type.name }}</option>
{% endfor %}
</select>
<br>
<label for="units">Units:</label>
<input type="number" id="units" name="units" required>
```

```
<br>
<button type="submit">Calculate Charging Cost</button>
</form>
</body>
</html>
<!-- charging_cost.html -->
<!DOCTYPE html>
<html lang="en">
<head>
<meta charset="UTF-8">
<meta name="viewport" content="width=device-width, initial-scale=1.0">
<title>Charging Cost</title>
<style>
body {
font-family: 'Arial', sans-serif;
background-color: #f7f7f7;
margin: 0;
padding: 0;
display: flex;
align-items: center;
justify-content: center;
height: 100vh;
}
.container {
text-align: center;
padding: 20px;
background-color: #fff;
border-radius: 8px;
box-shadow: 0 0 20px rgba(0, 0, 0, 0.1);
}
h2 {
color: #333;
}
p {
margin: 10px 0;
color: #555;
}
p span {
font-weight: bold;
color: #333;
}
p:last-child {
font-size: 1.4em;
color: #4caf50;
}
.directions-button {
margin-top: 20px;
background-color: #4285f4;
```

```
color: #fff;
padding: 10px 20px;
border: none;
border-radius: 4px;
cursor: pointer;
text-decoration: none;
display: inline-block;
}
.directions-button:hover {
background-color: #357ae8;
}
</style>
<!-- ... (other head elements) ... -->
</head>
<body>
<div class="container">
<h2>Charging Cost</h2>
<p><span>Charging Station:</span> {{ station.name }}</p>
<p><span>Location:</span> {{ station.location }}</p>
<p><span>Charger Type:</span> {{ charger_type.name }}</p>
<p><span>Units:</span> {{ units }}</p>
<p><span>Cost per kWh:</span> &#x20B9; {{ charger_type.cost_per_kwh }}</p>
{% set total_cost = units|float * charger_type.cost_per_kwh %}
<p><span>Total Charging Cost:</span> &#x20B9; {{ total_cost }}</p>
<!-- Google Maps Directions Button with search query -->
<a class="directions-button" href="https://www.google.com/maps/search/?api=1&query={{ station.name }} {{ station.location }}" tar
</div>
</body>
</html>
```

Output:

User Interface :

Page showing charging cost :

Conclusion and Future Scope:

The Flask application, with its robust database structure and user-friendly interface, effectively manages and presents information

Future Scope:

Potential enhancements include user authentication for personalized experiences, improved error handling, and input validation fo

Overall Impact:

The application's potential contribution to the electric vehicle infrastructure can be further realized by exploring partnerships, real-t

References :