

**MVJ College of Engineering, Bengaluru (An Autonomous Institute)**

Affiliated to VTU, Belagavi, Approved by AICTE, New Delhi, Recognized by UGC with 12(f) & 12 (B), Accredited by NBA & NAAC

JUNE 2023

ITINERARY PLANNER

A Mini Project Report

submitted in partial fulfillment of the requirement for the award of the degree

Bachelor of Engineering In

**INFORMATION SCIENCE AND ENGINEERING**

Submitted by

**NAME: ABHIRAM DINESH 1MJ20IS003**

**NAME: H.C PRAJWAL 1MJ20IS030**

**NAME: JATHIN D 1MJ20IS034**

**NAME: HITHESH KP 1MJ20IS037**

Under the guidance of

**Prof K.L Sharmila Thagoor,**

**ISE DEPT, MVJCE**



**MVJ COLLEGE OF ENGINEERING, BENGALURU-560067**

**(Autonomous Institution Affiliated to VTU, Belagavi)**

DEPARTMENT OF INFORMATION SCIENCE AND ENGINEERING

**CERTIFICATE**

Certified that the minor project work titled ***‘*ITINERY PLANNER*’*** is carried out by **ABHIRAM DINESH(1MJ20IS003), H C PRAJWAL(1MJ20IS030), JATHIN D(1MJ20IS034) and HITHESH K P(1MJ20IS037)** who are confide students of MVJ College of Engineering, Bengaluru, in partial fulfillment for the award of Degree of **Bachelor of Engineering in Information science and Engineering** of the Visvesvaraya Technological University, Belagavi during the year 2022- 2023. It is certified that all corrections/suggestions indicated for the Internal Assessment have been incorporated in the mini project report deposited in the departmental library. The mini project report has been approved as it satisfies the academic requirements in respect of mini project work prescribed by the institution for the said Degree.

|  |  |  |
| --- | --- | --- |
| **Signature of Guide** | **Signature of Head of the Department** | **Signature of Principal** |
| **Prof K.L Sharmila Thagoor** | **Mr. R KUMAR** | **Dr.** |

**External Viva**

**Name of Examiners Signature with Date**

**1**

**2**



**MVJ COLLEGE OF ENGINEERING, BENGALURU-560067**

**(Autonomous Institution Affiliated to VTU, Belagavi) DEPARTMENT OF INFORMATION SCIENCE AND ENGINEERING**

### DECLARATION

We **Abhiram Dinesh, HC Prajwal, Jathin D, Hithesh KP** students of sixth semester BE., Information science and Engineering, MVJ College of Engineering, Bengaluru, hereby declare that the mini project titled ***‘*ITINERY PLANNER*’*** has been carried out by us and submitted in partial fulfillment for the award of Degree of **Bachelor of Engineering** in **Information science and Engineering** during the year 2022-2023.

Further we declare that the content of the dissertation has not been submitted previously by anybody for the award of any Degree or Diploma to any other University.

We also declare that any Intellectual Property Rights generated out of this project carried out at MVJCE will be the property of MVJ College of Engineering, Bengaluru and we will be one of the authors of the same.

Place: Bengaluru Date:

**Name**

**ABHIRAM DINESH (1MJ20IS003)**

**HC PRAJWAL (1MJ20IS030)**

**JATHIN D (1MJ20IS034)**

**HITHESH KP (1MJ20IS037)**

**Signature**

# ACKNOWLEDGMENT

We are indebted to our guide**, Prof Sharmila Thagoor,** Professor**, Dept. of Information Science and Engineering, MVJ College of Engineering** for his wholehearted support, suggestions and invaluable advice throughout our project work and also for the help in the preparation of this report.

Our sincere thanks to **Dr. R KUMAR**, Associate Professor and Head, Department of Information Science and Engineering, MVJCE for his support and encouragement.

We express sincere gratitude to our beloved Principal, **Dr.** for all his support towards this project work.

Lastly, we take this opportunity to thank our **family** members and **friends** who provided all the backup support throughout the project work.

III

**ABSTRACT**

* The aim of this project is to develop a travel guide website that provides users with an itinerary based on the radius they enter in kilometers.
* The website utilizes location data and Google Places API to retrieve location data and store it in an Elasticsearch database.
* Generate a one-day itinerary based on the user's specified radius.
* We can filter and sort the locations based on various criteria such as popularity, ratings, distance, and relevance to recommend the best places to visit within the specified radius.
* This website aims to simplify travel planning by providing users with personalized itineraries and cost-effective transportation options.

# ACRONYMS

|  |  |
| --- | --- |
| **Acronym** | **Abbreviation** |
| USN | University Seat Number |
| SEM | Semester |

**TABLE OF CONTENTS**

###### **Page No.**

**Certificate Declaration Acknowledgment Abstract Acronyms**

**List of Figures**

I II III IV V VI

**CHAPTER 1**

1. Introduction 9
   1. [Existing System](#_TOC_250004)
   2. [Proposed System](#_TOC_250003)

CHAPTER 2

1. Theory and Concepts 14
   1. [System Review](#_TOC_250002)
   2. [Technology used](#_TOC_250001)
   3. Previous Existing system

CHAPTER 3

1. System Requirements 18
   1. Software Review
   2. Hardware Review
   3. [Requirement Analysis](#_TOC_250000)

CHAPTER 4

1. Methodology 21

CHAPTER 5

1. Applications 30

## **List of Figures**

###### **Figure no. Figure Title Page No.**

**1.2 Proposed Model** **13**

**2.2. Features of Proposed Model** **16**

CHAPTER 1

# INTRODUCTION

## **Introduction**

This project aims to create a travel guide website that assists users in planning their trips by generating personalized itineraries based on a specified radius. By leveraging location data and the Google Places API, the website will retrieve and store relevant information in an Elasticsearch database. Users can then input their desired radius, and the website will curate a one-day itinerary with the best places to visit within that range. The locations can be filtered and sorted based on popularity, ratings, distance, and relevance to ensure the recommendations meet the user's preferences.

### EXISTING SYSTEM

* When it comes to travel planning, many tourists rely on guided tours provided by established companies such as TripAdvisor and MakeMyTrip. While these tours can be a great way to explore a new destination, they often come with a fixed itinerary, leaving little room for customization. Additionally, these tours can be expensive, making them less accessible to budget-conscious travelers.
* A picture containing font, symbol, logo, graphics

  Description automatically generated

#### **Drawbacks of Existing Systems**

* Lack of Local Knowledge: Trip planners often rely on general information and popular tourist attractions, which may not take into account the local insights or hidden gems that can make a trip more memorable. They might overlook lesser-known destinations, local events, or cultural nuances that could enhance the travel experience.
* Limited Flexibility: Many trip planners offer pre-determined itineraries or fixed options for accommodations, activities, and transportation. This can limit the ability to customize the trip according to individual preferences and unique travel need.

### PROPOSED SYSTEM

* Our website aims to solve these issues by providing a new approach to travel planning.
* The process is simple: users enter their starting location and the radius they are willing to travel within. Our website then generates a list of recommended destinations based on proximity, popularity, and other factors. Users can then select the destinations that interest them and receive a detailed itinerary for their trip, restaurants, and landmarks to visit.
* Our website also aims to offer users a cost-effective transportation solution. We understand that getting around in a new city can be confusing and expensive, which is why we plan to include a rate comparison feature for public various transportation options, such as Ola, Uber, and NammaAuto.

#### **Advantages of proposed system**

* **Simplicity**: The code is written using Flask, a lightweight and easy-to-use web framework in Python. It follows a simple structure and uses decorators to define routes and handle requests, making it easy to understand and modify.
* **Modularity**: The code separates different functionalities into different functions and routes. This improves code readability and maintainability, as each function is responsible for a specific task. It also allows for easy expansion of the application by adding new routes and functions.
* **HTM**
* **L Templating**: The code uses the **render\_template** function from Flask to render HTML templates. This enables the separation of logic and presentation, making it easier to manage and update the frontend code. It also allows for the reuse of common HTML components across different routes.
* **User Input Handling**: The code properly handles user input by retrieving form data from the request object. It performs necessary validations and conversions on the input data before using it in API requests. This helps prevent common security vulnerabilities such as SQL injection or cross-site scripting.
* **API Integration**: The code integrates with the Google Places API to retrieve nearby places based on user input. It constructs API requests dynamically using the provided parameters and handles API responses to extract relevant data. This allows the application to provide personalized recommendations to the user.
* **Session Management**: The code uses Flask's session object to store and retrieve user-specific data, such as the top-rated places. This enables the application to remember user preferences across different requests and provide a personalized experience.
* **Error Handling**: The code includes error handling mechanisms to handle cases where the API request fails or no places are found. It provides appropriate error messages or renders a specific template to inform the user about the issue.
* **Code Reusability**: The code promotes code reusability by encapsulating repetitive tasks within functions. For example, the get\_place\_details function can be reused to retrieve detailed information about a specific place. This modular approach improves code organization and reduces redundancy.
* **Deployment**: The code includes a conditional block at the end that allows the application to be run as a standalone server. This makes it easy to deploy the Flask application on a server or cloud platform, making it accessible to users.
* Overall, the code provides a solid foundation for building a web application that interacts with the Google Places API to retrieve and display nearby places based on user input. It follows best practices for web development in Python, ensuring a clean and maintainable codebase
* **Figure**: Proposed model and Future Enhancement

A picture containing diagram, text, sketch, technical drawing

Description automatically generated

# CHAPTER 2

# THEORY AND CONCEPTS

**THEORY AND CONCEPTS**

### TECHNOLOGY USED

* **Front End: For** the front-end, we use HTML, CSS, and JavaScript to create a user-friendly interface.
* **API:** To gather location information and provide accurate recommendations, we Plan to utilize the Google Places API.
* **Database:** For the back-end, we use the Elasticsearch database. Which enables for auto suggestion and fuzzy search.
* **Back End:** When it comes to back-end programming languages, we plan to use Spring Boot. These are popular and widely-used frameworks for developing scalable and secure web applications. we can ensure that our website is both reliable and efficient for users.
* **To Test API:** To check our end point we will use postman.
* **Version Control:** Planning to use git to make and review changes during our project

## **Software requirements**

* 1. **SOFTWARE REQUIREMENTS.**
* **Python**: The code is written in Python, so you need to have Python installed on your system. The code is compatible with Python 3.1.0 versions.
* **Libraries:** The code requires the following libraries to be installed:
  + Flask
  + Render\_template
  + Request
  + Session

### HARDWARE REQUIREMENTS.

* The hardware requirements include a computer with sufficient processing power. A modern multi-core processor, at least 4 GB of RAM, and a GPU (optional but beneficial for performance), Internet Support are recommended.

### 

### REQUIREMENT ANALYSIS

* **Python:** Python is the basis of the program that we wrote. It utilizes many of the python libraries.
* **Libraries**:
  + Flask
  + Render\_template
  + Request
  + session
* **OS**: Program is tested on Windows 11 using Laptop to run our code.
* **Internet:** Internet required to use our web

# CHAPTER 4 METHODOLOGY

**Methodology:**

1. **Problem Statement**: Aims to Develop a web application for self-guided travel planning, allowing users to explore and plan their own travel experiences within a specified radius.

2**. Research**: Understanding user requirements and expectations for a self-guided travel application. Researched APIs for retrieving nearby places of interest.

3. **Environment Setup**: Set up the development environment, including installing necessary dependencies and frameworks like Flask., python

4. **Application Structure**: Design the Flask application structure, including routes, templates(html), and static files (image, CSS)

5. **User Interface**: Create a user-friendly interface to capture user input, specifically the desired travel radius.

6. **API Integration**: We Integrated with PlacesAPI to retrieve information about nearby places of interest.

7**. Data Retrieval**: Constructed API requests to fetch nearby places based on the user's specified radius.

8. **Data Processing**: Extracted relevant information from the API response, such as names, ratings, and photos of the retrieved places.

9. **Presentation**: Designed the user interface to display the retrieved places, including maps, images, and descriptions.

10. **Itinerary Planning**: Enabled users to select and create their own itinerary by choosing places of interest.

11. **Error Handling**: Implemented error handling mechanisms for API requests and responses, providing informative error messages to users. Such as displaying No Places found

12. **Testing**: Conducted comprehensive testing of the application to ensure functionality and user experience.

13**. Deployment**: Deployed Application Using Render so Listen to All devices

### SAMPLE CODE:

#### **Main.py**

from flask import Flask, render\_template, request, session

import requests, sys

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

rating= []

latitude=0

longitude=0

@app.get('/')

def home():

return render\_template('index.html')

@app.route('/process\_form', methods=['POST'])

def process\_form():

global rating

global longitude

global latitude

radius = request.form.get('radius')

radiuskm = int(radius) \* 1000

print(radiuskm)

place\_types = request.form.getlist('placeType')

latitude=request.form.get('latitude')

longitude=request.form.get('longitude')

print(latitude)

print(longitude)

# Do something with the retrieved values, such as storing them in a database

api\_key = 'AIzaSyAfWP\_iKshvMZmoUhz4VKXC4y3xoDTmE-4'

if 'all' in place\_types:

url = f'https://maps.googleapis.com/maps/api/place/nearbysearch/json?location={latitude},{longitude}&radius={radiuskm}&key={api\_key}'

else:

#place\_type\_str = ','.join(place\_types)

url = f'https://maps.googleapis.com/maps/api/place/nearbysearch/json?location={latitude},{longitude}&radius={radiuskm}&keyword={place\_types}&key={api\_key}'

#url = f'https://maps.googleapis.com/maps/api/place/nearbysearch/json?location={latitude},{longitude}&radius={radius}&keyword={place\_type}&key={api\_key}'

response = requests.get(url)

data = response.json()

places = data['results']

if not places:

return render\_template('none.html')

# Initialize the top-rated places list

max\_rating = 0

top\_rated\_places = session.get('top\_rated\_places', [])

for place in places:

place['photo\_url'] = place['photos'][0]['photo\_reference'] if 'photos' in place else None

place['rating'] = place.get('rating', None)

if place['rating'] is not None and place['rating'] > max\_rating:

top\_rated\_places = [place] # If a higher-rated place is found, replace the top-rated places list

max\_rating = place['rating']

elif place['rating'] is not None and place['rating'] == max\_rating:

top\_rated\_places.append(place) # If a place has the same rating as the current maximum, add it to top-rated places

rating+=top\_rated\_places

for place in rating:

encoded\_name = place['name'].encode('utf-8', errors='replace')

print(encoded\_name.decode('utf-8', errors='replace'))

return render\_template('places.html', places=places)

@app.route('/plan')

def plan():

return render\_template('itinerary.html', top\_rated\_places=rating,user\_latitude=latitude,user\_longitude=longitude)

@app.route('/place\_info/<place\_id>')

def place\_info(place\_id):

def get\_place\_details(place\_id):

api\_key = 'AIzaSyAfWP\_iKshvMZmoUhz4VKXC4y3xoDTmE-4' # Replace with your own API key

url = f'https://maps.googleapis.com/maps/api/place/details/json'

params = {

'place\_id': place\_id,

'key': api\_key

}

response = requests.get(url, params=params)

data = response.json()

place = data['result']

return place

place = get\_place\_details(place\_id)

return render\_template('place\_info.html', place=place)

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

app.run(host='0.0.0.0',port)

**INDEX.HTML**

<!DOCTYPE html>

<html>

<head>

<link rel="stylesheet" type="text/css" href="{{ url\_for('static', filename='main.css') }}">

<title>ITINERARY PLANNER</title>

<script>

function getLocation() {

if (navigator.geolocation) {

navigator.geolocation.getCurrentPosition(showPosition);

} else {

console.log("Geolocation is not supported by this browser.");

}

}

function showPosition(position) {

document.getElementById('latitude').value = position.coords.latitude;

document.getElementById('longitude').value = position.coords.longitude;

}

</script>

</head>

<body>

<h1>Place Filter</h1>

<div class="bi">

<form class="container" action="/process\_form" method="post" >

<label for="radius">Enter Radius:</label>

<input type="number" id="radius" name="radius" required>

<label for="placeType">Filter by Place Type:</label>

<select id="placeType" name="placeType">

<option value="all" selected>All</option>

<option value="temple">Temple</option>

<option value="hotel">Hotel</option>

<option value="park">Park</option>

<option value="Busstop">BUS STOP</option>

<option value="metrostation">metro</option>

<option value="school">school/colleges</option>

<option value="policestation">policestation</option>

<option value="hospital">hospital</option>

<option value="movietheatre">Movie Theatre</option>

</select>

<input type="hidden" id="latitude" name="latitude">

<input type="hidden" id="longitude" name="longitude">

<button type="submit">Submit</button>

</form>

</div>

<script>

getLocation();

</script>

</body>

</html>

**ITNIERARY.HTML**

<!DOCTYPE html>

<html>

<head>

<title>Itinerary</title>

<link rel="stylesheet" type="text/css" href="{{url\_for('static', filename='itinerary.css') }}">

</head>

<body>

<h1>Itinerary</h1>

<div class="roadmap">

{% for place in top\_rated\_places %}

<div class="roadmap-item">

{% if place.photo\_url %}

<img src="https://maps.googleapis.com/maps/api/place/photo?maxwidth=400&maxheight=400&photoreference={{ place.photo\_url }}&key=AIzaSyAfWP\_iKshvMZmoUhz4VKXC4y3xoDTmE-4" alt="Place Image">

{% endif %}

<div>

<h3>{{ place.name }}</h3>

{% if place.rating %}

<p>Rating: {{ place.rating }}</p>

{% endif %}

</div>

</div>

{% endfor %}

</div>

<button onclick="showDirections()">Show Directions</button>

<div id="map"></div>

<script>

function initMap() {

const directionsService = new google.maps.DirectionsService();

const directionsRenderer = new google.maps.DirectionsRenderer();

const map = new google.maps.Map(document.getElementById("map"), {

center: { lat: {{ user\_latitude }}, lng: {{ user\_longitude }} },

zoom: 12

});

directionsRenderer.setMap(map);

directionsRenderer.setPanel(document.getElementById("directionsPanel"));

const waypoints = [];

{% for place in top\_rated\_places %}

let waypoint{{ loop.index }} = {

location: new google.maps.LatLng({{ place.geometry.location.lat }}, {{ place.geometry.location.lng }}),

stopover: true

};

waypoints.push(waypoint{{ loop.index }});

{% endfor %}

const request = {

origin: new google.maps.LatLng({{ user\_latitude }}, {{ user\_longitude }}),

destination: new google.maps.LatLng({{ user\_latitude }}, {{ user\_longitude }}),

waypoints: waypoints,

optimizeWaypoints: true,

travelMode: google.maps.TravelMode.DRIVING

};

directionsService.route(request, function(result, status) {

if (status == "OK") {

directionsRenderer.setDirections(result);

}

});

}

function showDirections() {

if (navigator.geolocation) {

navigator.geolocation.getCurrentPosition(function(position) {

const userLat = position.coords.latitude;

const userLng = position.coords.longitude;

const destLat = {{ top\_rated\_places[0].geometry.location.lat }};

const destLng = {{ top\_rated\_places[0].geometry.location.lng }};

const url = `https://www.google.com/maps/dir/${userLat},${userLng}/${destLat},${destLng}`;

window.open(url);

});

} else {

alert("Geolocation is not supported by your browser.");

}

}

</script>

<script src="https://maps.googleapis.com/maps/api/js?key=AIzaSyAfWP\_iKshvMZmoUhz4VKXC4y3xoDTmE-4&callback=initMap" async defer></script>

<div id="directionsPanel"></div>

</body>

</html>

**None.html**

<!DOCTYPE html>

<html>

<head>

<title>No Places Available</title>

<link rel="stylesheet" type="text/css" href="{{url\_for('static', filename='none.css') }}">

</head>

<body>

<div class="image">

<div>

<h1>No Places Available!..</h1>

<p>There are no places available near the entered radius.</p>

</div>

</div>

</body>

</html>

**PLACEINFO.HTML**

<!DOCTYPE html>

<html>

<head>

<title>Place Information</title>

<style>

body {

font-family: Arial, sans-serif;

margin: 0;

padding: 20px;

}

h1 {

font-size: 28px;

margin-top: 0;

}

p {

margin: 0;

}

a {

color: #0000EE;

text-decoration: none;

}

#map {

height: 400px;

width: 100%;

margin-top: 20px;

}

button {

margin-top: 20px;

padding: 10px 20px;

font-size: 16px;

background-color: #4CAF50;

color: #FFFFFF;

border: none;

cursor: pointer;

}

button:hover {

background-color: #45a049;

}

ul {

padding-left: 20px;

}

</style>

</head>

<body>

<h1>{{ place.name }}</h1>

{% if place.rating %}

<p>Rating: {{ place.rating }}</p>

{% endif %}

{% if place.formatted\_address %}

<p>Address: {{ place.formatted\_address }}</p>

{% endif %}

{% if place.website %}

<p>Website: <a href="{{ place.website }}">{{ place.website }}</a></p>

{% endif %}

{% if place.geometry and place.geometry.location %}

<div id="map"></div>

<script>

function initMap() {

const map = new google.maps.Map(document.getElementById("map"), {

center: { lat: {{ place.geometry.location.lat }}, lng: {{ place.geometry.location.lng }} },

zoom: 14

});

new google.maps.Marker({

position: { lat: {{ place.geometry.location.lat }}, lng: {{ place.geometry.location.lng }} },

map,

title: "{{ place.name }}"

});

}

</script>

<script src="https://maps.googleapis.com/maps/api/js?key=AIzaSyAfWP\_iKshvMZmoUhz4VKXC4y3xoDTmE-4&callback=initMap" async defer></script>

{% endif %}

{% if place.types and 'tourist\_attraction' in place.types %}

<h2>Things to Do:</h2>

<ul>

{% for activity in place.activities %}

<li>{{ activity }}</li>

{% endfor %}

</ul>

{% endif %}

<!-- Add the "Show Directions" button -->

{% if place.geometry and place.geometry.location %}

<button onclick="showDirections()">Show Directions</button>

<script>

function showDirections() {

if (navigator.geolocation) {

navigator.geolocation.getCurrentPosition(function(position) {

const userLat = position.coords.latitude;

const userLng = position.coords.longitude;

const destLat = {{ place.geometry.location.lat }};

const destLng = {{ place.geometry.location.lng }};

const url = `https://www.google.com/maps/dir/${userLat},${userLng}/${destLat},${destLng}`;

window.open(url);

});

} else {

alert("Geolocation is not supported by your browser.");

}

}

</script>

{% endif %}

</body>

</html>

**PLACES.HTML**

<!DOCTYPE html>

<html>

<head>

<title>Places</title>

<link rel="stylesheet" type="text/css" href="{{url\_for('static', filename='places.css') }}">

<script>

function getLocation() {

if (navigator.geolocation) {

navigator.geolocation.getCurrentPosition(showPosition);

} else {

console.log("Geolocation is not supported by this browser.");

}

}

function showPosition(position) {

document.getElementById('latitude').value = position.coords.latitude;

document.getElementById('longitude').value = position.coords.longitude;

}

**CSS**

body {

font-family: Arial, sans-serif;

margin: 0;

padding: 0;

}

h1 {

text-align: center;

margin-top: 20px;

}

.roadmap {

display: flex;

flex-wrap: wrap;

justify-content: center;

}

.roadmap-item {

width: 300px;

margin: 20px;

padding: 20px;

border: 1px solid #ccc;

border-radius: 5px;

box-shadow: 0 2px 5px rgba(0, 0, 0, 0.1);

background-color: #fff;

text-align: center;

}

.roadmap-item img {

max-width: 100%;

height: auto;

margin-bottom: 10px;

}

.roadmap-item h3 {

margin-top: 0;

}

.roadmap-item p {

margin: 0;

}

#map {

height: 400px;

margin-top: 20px;

}

#directionsPanel {

margin-top: 20px;

padding: 10px;

border: 1px solid #ccc;

border-radius: 5px;

background-color: #f9f9f9;

}

button {

display: block;

margin: 20px auto;

padding: 10px 20px;

font-size: 16px;

border: none;

border-radius: 5px;

background-color: #007bff;

color: #fff;

cursor: pointer;

}

h1 {

text-align: center;

margin: 20px;

}

/\* Styles for the form \*/

form {

max-width: 400px;

margin: auto;

margin-top: 20px;

padding: 20px;

display: flex;

flex-direction: column;

align-items: auto;

justify-content: center;

height: 100vh;

}

.bi {

background: url("background-image.png");

background-position: top;

background-repeat: no-repeat;

background-size: auto;

margin-top:top;

margin-left: auto;

margin-right: auto;

margin-bottom: auto;

}

/\* Styles for the form labels \*/

label {

display: block;

margin-bottom: 10px;

font-weight: bold;

}

/\* Styles for the form input fields \*/

input[type="number"] {

width: 100%;

padding: 8px;

}

/\* Styles for the form dropdown menu \*/

select {

width: 100%;

padding: 8px;

}

/\* Styles for the form submit button \*/

button[type="submit"] {

width: 100%;

padding: 10px;

margin-top: 10px;

background-color: #4CAF50;

color: #fff;

border: none;

border-radius: 4px;

cursor: pointer;

}

button[type="submit"]:hover {

background-color: #45a049;

}

/\* Additional styles for the page \*/

body {

font-family: Arial, sans-serif;

margin: 0;

padding: 20px;

}

.image {

background-image: url("not\_found.jpg");

background-size: 100% 100%;

background-repeat: no-repeat;

height: 100vh;

display: flex;

justify-content: center;

align-items: center;

}

h1{

margin-left: 600px;

color:white;

font-size: 100px;

}

p{

margin-left: 600px;

color:white;

font-size: 40px;

}

/\* Global Styles \*/

body {

font-family: Arial, sans-serif;

margin: 0;

padding: 20px;

}

h1 {

text-align: center;

}

form {

margin-bottom: 20px;

}

label {

display: block;

margin-bottom: 10px;

font-weight: bold;

}

input[type="number"],

select {

width: 100%;

padding: 8px;

border: 1px solid #ccc;

border-radius: 4px;

box-sizing: border-box;

margin-bottom: 10px;

}

button[type="submit"],

button[type="button"] {

padding: 10px 20px;

background-color: #4CAF50;

color: #fff;

border: none;

border-radius: 4px;

cursor: pointer;

}

button[type="submit"]:hover,

button[type="button"]:hover {

background-color: #45a049;

}

/\* Places List \*/

ul {

list-style-type: none;

padding: 0;

}

**OUTPUT**: web Application Successfully Run on Port 8000 in Local Server

A screenshot of a computer

Description automatically generated with medium confidence

A screenshot of a computer

Description automatically generated with low confidence

A screenshot of a computer

Description automatically generated with medium confidence

A screenshot of a map

Description automatically generated with medium confidence

A picture containing text, map, atlas, diagram

Description automatically generated

# CHAPTER 5

# APPLICATIONS

###### **APPLICATIONS**

The travel guide website and itinerary planner application you described can be used in various scenarios and by different types of users. Here are some examples:

* **Individual Travelers:** Solo travelers who want to explore new destinations within a specified radius can use the application to plan their day and discover popular attractions, restaurants, and landmarks.
* **Tourists:** Tourists visiting a new city or location can utilize the website to get recommendations for nearby places to visit and create a customized itinerary based on their preferences and time constraints.
* **Travel Agencies:** Travel agencies can integrate the application into their websites or platforms to provide itinerary planning services to their customers. The application can assist travel agents in creating personalized itineraries for their clients based on the specified radius and other preferences.
* **City Guides:** City guides and tourism organizations can incorporate the application into their websites to offer visitors a comprehensive travel planning tool. The application can provide recommendations for attractions, restaurants, and transportation options, enhancing the overall tourist experience.
* **Travel Blogs and Websites:** Travel bloggers and websites can leverage the application to provide their readers with itinerary suggestions and recommendations for specific locations or travel themes.
* **Educational Institutions:** Educational institutions conducting field trips or study tours can utilize the application to plan and organize activities within a certain radius of their destination.

Overall, the travel guide website and itinerary planner application can be used by individuals, tourists, travel agencies, city guides, transportation service providers, travel bloggers, educational institutions, and more to enhance travel planning, discover new destinations, and optimize the travel experience within a specified radius.

**References**

* **HTML**: Html is markup language also known to give structure to website has been implemented using Reference: <https://developer.mozilla.org/en-US/docs/Learn/HTML>
* **CSS:** Cascading Style Sheet is used to beautify website using Reference: <https://developer.mozilla.org/en-US/docs/Learn/CSS>
* **JAVASCRIPT:** JavaScript is a scripting language use to implement logic and responsiveness in a website it has been implemented using Reference: <https://developer.mozilla.org/en-US/docs/Learn/JAVASCRIPT>
* **Flask:** Flask is backend Framework used in website as server-side Language it has been implemented using Reference: <https://flask.palletsprojects.com/en/2.3.x/>
* **API: Application** Program Interference is used to fetch the detail from other server database in website in our website Places API has been implemented using Reference: <https://developers.google.com/maps/documentation/places/web-service/overview>