



**A**  
**Project Report**  
on  
**Path Management System**  
submitted as partial fulfillment for the award of  
**BACHELOR OF TECHNOLOGY**  
**DEGREE**

SESSION 2024-25  
in  
**Computer Science And Engineering**

By  
Karan Kumar (2100290100083)  
Rohit Singh Maurya (2100290100138)

**Under the supervision of**  
Mrs. Mani Dwivedi  
**KIET Group of Institutions, Ghaziabad**

Affiliated to  
**Dr. A.P.J. Abdul Kalam Technical University, Lucknow**  
(Formerly UPTU)  
**May, 2025**

## **DECLARATION**

We hereby declare that this submission is our own work and that, to the best of our knowledge and belief, it contains no material previously published or written by another person nor material which to a substantial extent has been accepted for the award of any other degree or diploma of the university or other institute of higher learning, except where due acknowledgment has been made in the text.

Signature: karan kumar

Name: karan kumar

Roll No.: 2100290100083

Date: 24/05/2025

Signature: Rohit Singh Maurya

Name: Rohit Singh Maurya

Roll No. 2100290100138

Date: 24/05/2025

## **CERTIFICATE**

This is to certify that Project Report entitled Path Management System which is submitted by Karan Kumar and Rohit Singh Maurya in partial fulfillment of the requirement for the award of degree B. Tech. in Department of Computer Science & Engineering of Dr. A.P.J. Abdul Kalam Technical University, Lucknow is a record of the candidates own work carried out by them under my supervision. The matter embodied in this report is original and has not been submitted for the award of any other degree.

.

**Date:**

**Supervisor Name**

**(Designation)**

## **ACKNOWLEDGEMENT**

It gives us a great sense of pleasure to present the report of the B. Tech Project undertaken during B. Tech. Final Year. We owe special debt of gratitude to supervisor name, Department of Computer Science & Engineering, KIET, Ghaziabad, for his constant support and guidance throughout the course of our work. His sincerity, thoroughness and perseverance have been a constant source of inspiration for us. It is only his cognizant efforts that our endeavors have seen light of the day.

We also take the opportunity to acknowledge the contribution of Dr. Vineet Sharma, Head of the Department of Computer Science & Engineering, KIET, Ghaziabad, for his full support and assistance during the development of the project. We also do not like to miss the opportunity to acknowledge the contribution of all the faculty members of the department for their kind assistance and cooperation during the development of our project.

We also do not like to miss the opportunity to acknowledge the contribution of all faculty members, especially faculty/industry person/any person, of the department for their kind assistance and cooperation during the development of our project. Last but not the least, we acknowledge our friends for their contribution in the completion of the project.

Date: 24/05/2025

Signature: karan kumar

Name : Karan Kumar

Roll No.: 2100290100083

Date: 24/05/2025

Signature: Rohit Singh Maurya

Name: Rohit Singh Maurya

Roll No. 2100290100138

## ABSTRACT

It can get very exciting to plan a vacation, but sometimes it can seem a little overwhelming, especially when you want to visit several places. Many people who travel have to manage different tasks at the same time.

various sights, keeping an eye on spending, planning how much time to spend and enjoying the whole experience. You need to plan your trip well to travel less, spend less and have a greater amount of fun. An ideal route for travel is looking after this involves making the between-destination trips as smooth and cheap as possible. Our goal in this project is to improve the way travelers plan vacations, by offering them a helpful travel route for seeing several chosen destinations. Users can list particular destinations they want to explore on the system. If people allow me to collect data, Because they do not know which location to pick, the app can suggest the nearest popular spots, At the current location or inside an area chosen by the user. As soon as the locations are selected, the system looks up the distances using the Google Maps API.

Using a graph, with location as a node and distances as the weight attached to the connecting edges. Our program uses Dijkstra's Algorithm to come up with the most practical route covering all the chosen destinations. Selecting this plan makes it possible for the driver to go the shortest route, enjoy a shorter trip, use less fuel and save money. This means travelers end up with a sensible budget for the trip and less stress about getting from one place to another. The PMS relies on algorithms, in particular Dijkstra's, as well as other search methods based on heuristics, to ensure routes are efficient and short. It uses ongoing information from sensors to respond to disturbances such as objects, traffic and changes in the network environment. It is adjustable to almost any grid and network condition, supporting both basic navigation indoors and wider operations in logistics. Consequently, we identify the best and quickest way to hit every point and use google API to know what the node's location is.

# TABLE OF CONTENTS

DECLARATION.....	ii
CERTIFICATE.....	iii
ACKNOWLEDGEMENTS.....	iv
ABSTRACT.....	v
LIST OF FIGURES.....	viii
LIST OF TABLES.....	ix
CHAPTER 1 (INTRODUCTION).....	1-7
1.1. Introduction.....	1-3
1.2. Project Description.....	3-7
CHAPTER 2 (LITERATURE RIVIEW).....	8-11
2.1. Location's API .....	8
2.2. Dijkstra's algorithms .....	10
CHAPTER 3 (PROPOSED METHODOLOGY) .....	12-18
3.1.Route Optimization .....	14-16
CHAPTER 4 (SOURCE CODE) .....	19-22
CHAPTER 5 ( RESULTS AND DISCUSSIONS ).....	22 -28
5.1 Results.....	22-24
5.2 Discussions.....	24-28

CHAPTER 6 ( OUTPUT AND INTERFACE ).....	29-32
CHAPTER 7 ( CONCLUSION AND FUTURE WORK )	33-38
7.1. Conclusion	33-34
7.2 Future-work	34-38
REFERENCES.....	39
APPENDIX.....	40-44
....	

## **LIST OF FIGURES**

<b>Figure No.</b>	<b>Description</b>	<b>Page No.</b>
1.	SOURCE - CODE	21-24
2.	INTERFACE AND OUTPUT	31- 34



## LIST OF TABLES

Table. No.	Description	Page No.
1.	Distance- Matrix	18
2.	Distance-Matrix Construction	26

# **CHAPTER 1**

## **INTRODUCTION**

### **1.1 INTRODUCTION**

Apart from being enjoyable, travel gives travelers the chance to meet new people, break from routine and create memories that stay with them. As people find it possible to travel more easily, they are now more interested in seeing many places all at once. No matter if their purpose is adventure, learning or rest, modern visitors want to fit numerous attractions into a brief amount of time. While the idea of a great multi-location trip is nice, the planning process tends to be complex.

Travelers who want to squeeze in multiple places or activities usually have a hard time putting the trip agenda together. This process covers picking the finest locations, calculating how you will reach them, handling costs and combining trip enjoyment with planning. Such factors usually baffle even expert travelers, often resulting in unwelcome delays, extended trips and higher expenses. It's common for travelers to find that the time they spend traveling is often longer than the time they truly enjoy. You may feel stressed, tired and upset, since your plans weren't fully thought out, leading to a different experience than what you hoped for.

This challenge which is common but important, led us to make Path Management System(PMS). PMS is mainly designed to help users plan meaningful, practical and enjoyable journeys to various destinations using an easy-to-use platform. Strong algorithms and reliable databases make using PMS an easy way to plan a journey. Travelers use the best routes to save time and money as they discover their destinations.

The main idea behind PMS is to couple new computational procedures with actual travel logistics. The system gathers what the user wants and likes and processes the details with Dijkstra's Algorithm and the Floyd-Warshall Algorithm. They compare the routes and waiting times between stops to pick the most effective order to visit them all. The objective is straightforward and valuable: spend less time on the road, spend less money and make the experience easier by eliminating logistical trouble.

In addition, PMS makes it easy for travelers who do not yet know where to go. The system helps these users find interesting tourist spots in their area or nearby. A record can be useful, especially for people who don't know the place well or who like to book trips fairly last minute. In this case, if Nainital is the traveler's destination, the system will automatically include Bhimtal, Sattal or Naukuchiatal in its recommendations so they have the best chance of enjoying the region. To achieve its goals, the system uses strong features such as the Google Maps API and OpenStreetMap API. As a result of these technologies, the system can bring in current travel data, find the best routes, look at present traffic and show maps to users. специфицируется на Langzhi Using both APIs increases the system's ability to change and keep running well in many different situations and areas.

Because the basic structure of PMS is modular, it can be scaled up and used with ease. This system has three main sections: the User Input Module, the Route Optimization Engine and the Visualization and Mapping Interface. By getting information through the User Input Module, travelers add where to travel, what their preferences are and the beginning point for their trip. Data analysis by the Route Optimization Engine allows algorithms to determine the best route. The Visualization and Mapping Interface shows the optimal route on an interactive map, so people can see their travel route and make necessary adjustments.

One of the main strengths of PMS is that it relies on the Floyd-Warshall Algorithm, a computer science technique that finds the shortest routes between every pair of nodes in a graph with weights. During travel planning, places to visit are known as nodes and their distances from one another are used as edge weights. The algorithm reviews all routes between particular locations and Bernieke's app frequently recalculates the shortest distances. Thanks to this method, the final result includes a route where the least amount of time and resources is used to traverse all the stops chosen.

If we want to discover the shortest route from a given starting point to one or more destinations, the Dijkstra's Algorithm can be used. The process involves selecting one unvisited node at a time and always updating the known shortest distances of all connected nodes. Using this algorithm in PMS, users can easily picture each step of their journey.

The Path Management System was tested by studying a case of four destinations: Muradnagar, Vaishali, Red Fort and Meerut. API data was used to calculate how far each pair of locations are apart and then the Floyd-Warshall algorithm was applied to get the shortest route hitting all

four stops. The test proved that using autonomous navigation reduced both the time and expenses required for travel. The system reduced repetition in how users traveled while also giving them a clear, helpful map to follow.

PMS is also useful because it can respond to new changes easily. The system is programmed to respond to news like traffic delays, roadblocks and changes in what drivers want. If there's a sudden traffic jam that backs up a route, PMS is able to instantly find and tell you about a new optimal route using the most up-to-date information. Because of this, the system can maintain reliability during unpredictable traveling situations.

PMS also puts great importance on how easy and enjoyable the platform is for customers. The interface is built to be clear and uncomplicated, so those with little tech knowledge can easily work it. Every suggestion, path and visualization is organized in a way that users can easily understand. Because of this, travelers enjoy their trip and don't have to worry about the details. . going forward, there are various ways the Path Management System could be improved. In the future, AIs could use old travel records, along with seasonal changes, to help suggest the most suitable days and paths to travel. Reviews and suggestions from other users can help customize the trip by bringing to light the best and most popular spots for tourists. Additionally, mixing flight, train and local transit data into the system can make it more complete.

In short, Path Management System fills a basic demand in the modern travel market: helping passengers plan trips that cover multiple destinations. Using strong pathfinding techniques, up-to-date data links and a user-friendly system, PMS makes a complicated process simple and fun. It reduces how long and how much you spend on travel without distracting you from having a good vacation. By focusing on updates and thinking about users' needs, PMS may completely alter the way travel planning and booking happen today.

## **1.2 PROJECT DESCRIPTION**

Trying to organize a journey that touches more than one destination has always been a tough challenge for many travelers. The idea of experiencing many places on one trip is appealing, but making those plans work takes careful attention to various complex issues.

chores such as finding proper locations, estimating distances between destinations, booking how to get there and tracking costs. Just about everyone involved in tourism ends up dealing with multiple apps, guidebooks and maps to help with their itinerary. Therefore, having an intelligent system that manages and improves these tasks is absolutely necessary under these circumstances. The Path Management System(PMS) was created to address just this challenge, allowing travelers to organize their itinerary easily for multi-city trips.

Path Management System's main goal is to guide users to locations by showing the optimized way that is quickest and most affordable. Such a system makes use of algorithms that represent each location with a node and each distance with a weighted edge. With the solution, people can reach their chosen destinations while enjoying an organized and easier journey.

### **System Overview ->**

Path Management System collects user information, current geographic details and uses sophisticated processing to design the most effective travel routes. There are three essential parts to such a system:

1. **User Input Module**
2. **Route Optimization Engine**
3. **Visualization and Mapping Interface**

Each of these components plays a vital role in collecting data, processing it, and displaying the results in a user-friendly and interactive format.

### **1. User Input Module ->**

The first interaction a user has with the system happens in this module. The system is made to receive a list of places that the user wants to go. Sometimes, you can use places, famous landmarks or even general places as your answer. The system can propose local attractions to users who are unsure about their next move, using their present location or selecting an area. With this approach, everyone, whether a seasoned traveler or just discovering the area, enjoys flexibility.

Converting coins and bills is very easy through the interface. People can choose any number of destinations, after which the system will plan their route. At this stage, you can also indicate where you'd like to start, the roads you'd rather not take (like highways) and the amount you'd like to spend on travel costs. By knowing what you prefer, we are able to build a unique vacation plan for you.

## 2. Route Optimization Engine

As soon as the information from the user is gathered, Route Optimization Engine starts working. This module determines the best and fastest route for all the locations entered in the TSP. The process of the engine is supported by graph algorithms.

Dijkstra's Algorithm and the Floyd–Warshall Algorithm are considered important on this list.

**Dijkstra's Algorithm** is primarily used when the user needs the shortest path from a specific starting point to a single or a few destinations. This algorithm works by selecting the nearest unvisited node and updating the shortest known paths from the source, iteratively refining the path until the most optimal route is established.

**Floyd-Warshall Algorithm**, on the other hand, is used when the user needs a comprehensive route connecting all the locations. This algorithm calculates the shortest paths between every possible pair of destinations. It systematically examines each possible route combination and updates the distance matrix to ensure the most efficient paths are selected.

For example, a test scenario involved four locations: Muradnagar, Vaishali, Red Fort, and Meerut. The PMS system utilized the Floyd-Warshall Algorithm to generate a distance matrix and compute the most efficient path covering all locations. The outcome was a significant reduction in total travel distance and time, as compared to manual planning.

## 3. Visualization and Mapping Interface

After the route has been calculated, the next step is to present the results to the user in a way that is easy to understand and interact with. This is where the Visualization and Mapping Interface comes into play. It utilizes the Google Maps API and OpenstreetMap to display the route graphically.

- **Google Maps API** offers robust features such as live traffic data, route rendering, and estimated time of arrival (ETA). It's particularly valuable in urban environments or areas with complex road networks.
- **OpenStreetMap API** provides a free, open-source alternative that can be customized for various applications. It supports a collaborative map-building approach, making it ideal for areas not fully covered by commercial services.

This dual API integration not only enhances the accuracy and utility of the visual maps but also offers redundancy in case one service becomes unavailable. The final route is shown with markers for each destination and a highlighted path that indicates the optimized travel plan.

### **Implementation Details ->**

The PMS project was implemented using a combination of web technologies and backend logic. The core algorithms were coded in a high-level language like Python or JavaScript, which allowed seamless integration with mapping APIs and front-end frameworks. The user interface was developed with responsiveness in mind, ensuring compatibility across devices.

A pseudocode summary for the Floyd-Warshall Algorithm used in the PMS can be outlined as:

Let  $dist$  be a  $|V| \times |V|$  matrix initialized to  $\infty$

For each edge  $(u, v)$ , assign  $dist[u][v] = weight(u, v)$

For each node  $v$ , assign  $dist[v][v] = 0$

For  $k$  from 1 to  $|V|$ :

    For  $i$  from 1 to  $|V|$ :

        For  $j$  from 1 to  $|V|$ :

            If  $dist[i][j] > dist[i][k] + dist[k][j]$ :

$dist[i][j] = dist[i][k] + dist[k][j]$

This logic ensures that the system evaluates all possible paths and updates the shortest one accordingly.

### **Use of APIs ->**

Both Google Maps and OpenstreetMap APIs were vital to the real-world applicability of PMS. The Google Maps API provided real-time data, including distance, duration, and traffic conditions, while OpenStreetMap offered a customizable and cost-free mapping solution. The APIs were used to fetch location coordinates, display maps, and calculate travel data.

Furthermore, the system was designed to use the device's GPS or IP address to detect the current location of the user, treating it as the starting node. This added convenience ensures that even if users do not input a start location, the system can determine it automatically.

### **Real-World Application ->**

The PMS is not limited to just vacation planning. Its application can extend to a wide range of real-world use cases including:

- **Delivery and Logistics Management:** Companies can use PMS to optimize delivery routes, reducing fuel consumption and improving delivery times.

- **Public Transport Route Planning:** Municipal corporations can implement the system to design bus or train routes that cover maximum areas in minimal time.
- **Campus Navigation:** Universities and large campuses can use PMS to guide visitors or students through multiple departments or buildings efficiently.
- **Emergency Services:** Police, ambulance, and fire services can leverage PMS to find the shortest path to multiple affected locations during emergencies.

#### **Future Scope ->**

While PMS has already shown promising results, there is significant potential for expansion and enhancement. Some planned improvements include:

- **AI and Machine Learning Integration:** Predictive analytics can be used to suggest destinations based on user interests, historical trends, or seasonal popularity.
- **Live Traffic Adaptation:** Real-time rerouting based on changing traffic conditions will further enhance travel efficiency.
- **Multimodal Transport Integration:** Including options like railways, buses, flights, and ride-sharing within the itinerary.
- **User Feedback System:** Enabling users to rate destinations and share experiences will help others make informed choices.

This Path Management System is a forward-thinking solution in the realm of intelligent travel planning. It demonstrates how algorithmic logic, real-time data, and user-friendly interfaces can come together to solve everyday challenges in a smart and scalable way.



## **CHAPTER 2**

### **LITERATURE REVIEW**

An innovative Path Management System for finding the best routes is based on scientific research, technologies and practices from computer science, geographic information systems (GIS) and travel logistics. The field's literature lays out a clear structure for including location-based services, routing methods and mapping APIs in utility software for travel planning. It also explores current methods for organizing routes in multiple locations and reveals where PMS can fill gaps.

#### **1. Travel Planning and Route Optimization: A Growing Need ->**

Deciding on how to travel has become much more involved recently, thanks to more people traveling and many transport choices. From many studies on how people travel, one important challenge for road users is choosing a route that matches their needs for time, price and pleasure. Selecting where to stop on a trip and the order in which to see the places can be a challenge for tourists. Things get more complicated when travelers visit more than just one place in a trip. According to the World Tourism Organization, almost two-thirds of international tourists visit more than a single city at a time. Because more and more people want multi-destination travel plans, computerized systems are needed to manage these trips accurately. That's why systems such as the PMS are being made, because they calculate the best routes to use based on data and mathematical models.

#### **2. Role of Geographic Information Systems (GIS) and Location APIs ->**

GIS is crucial for organizing trip routes and planning how best to travel in today's world. Using GIS, users are able to observe how different places are related and see useful patterns in travel, supporting the selection of the most efficient travel routes. For a long time, GIS was mainly used in land surveying, urban planning and managing disasters. Thanks to new technology, GIS has become a key element in web-based travel applications. APIs help these applications gather, process and present maps and other geographical data. With PMS, developers use both the Google Maps API and the OpenStreetMap API. It is well known that

the Google Maps API includes strong functions such as figuring routes, providing live traffic updates and estimating travel time. OSM supports several travel methods, enabling anyone to add useful details to a community project. Research finds that in places that are rural or stay less mapped, OSM can work better than commercial APIs as people in the community contribute more. OSM stands out by giving companies and developers the chance to use cost-effective and adaptable maps, while PMS assures flexibility and exact data for anyone traveling, anywhere.

### **3. Graph Theory and Route Optimization Algorithms ->**

In essence, route optimization deals with graph theory, where towns or destinations are modeled as nodes and the distances between them are given to edges. The book discusses many methods to locate the most efficient route on a network, but two are recognized as especially useful.

#### **a) Dijkstra's Algorithm ->**

The algorithm was suggested by Edsger W. Dijkstra in 1956 and aims to discover the shortest path from a starting node to all the other nodes in a graph with edges that have only positive values. By using the algorithm, the program will pick the node with the smallest distance to the source each time and calculate the changes it needs to make to the distances of the other nodes. Dijkstra's Algorithm is used in many navigation systems until every shortest path is located. Because it works smoothly in sparse graphs, it's helpful in finding routes that start from a known point and need to go farther.

#### **b) Floyd-Warshall Algorithm ->**

Although Dijkstra's Algorithm is great for finding the shortest path from a single source, Floyd-Warshall Algorithm is used to compute all -pair shortest paths. Back in the early 1960s, the Floyd-Warshall algorithm began use and still finds use today in finding the shortest way between every combination of nodes. PMS relies on Floyd-Warshall to automatically perform many optimizations in route computation and help users manage fewer direct routes.

### **4. Existing Route Planning Solutions ->**

Route planning tools have become more common as digital platforms have appeared over the last ten years. Tools like Google Maps, Rome2Rio and apps such as TripIt and Roadtrippers all allow you to make a basic plan for multiple stops. Usually, users can list many locations and get directions for how to travel between them. On the other hand, these projects usually fail to optimize routes dynamically, using information on traffic, current road conditions or personal preferences. Conversely, commercial systems in delivery logistics such as those created by FedEx and UPS, handle this type of optimization using advanced models. Much of these models draw on VRP and depend on genetic algorithms, simulated annealing or ant colony optimization. They are developed mainly for industrial use, not for planning consumer journeys. Because of this possibility, PMS was developed so that all travelers, no matter their background, could use advanced optimization tools easily.

## **5. Application of Heuristic Search Methods ->**

Exact algorithms based on Dijkstra's and Floyd-Warshall cannot effectively handle large networks with challenging travel restrictions. For this reason, researchers have investigated A along with other heuristic methods to obtain near-optimal answers in a short time. Urban travel applications have shown that this method manages to be both swift and accurate. Future improvements might involve using heuristic models when the number of nodes involved gets even larger or when personal features such as budget and time constraints are involved.

## **6. Importance of Real-Time Data and Adaptability ->**

Experts in ITS point out that incorporating real-time data is vital for finding the best routes. Traffic, road obstructions and timings for public transport can make a huge difference in the success of a trip. Studies have proven that using systems that change with new information can lower travel time by up to 30%. This proves that APIs that can provide live data should be part of any PMS solution. Delivering up-to-date traffic information and options for alternative routes, PMS stays helpful and accurate no matter what traffic patterns are.

## **7. OpenStreetMap and Community-Based Mapping ->**

OpenStreetMap is an open project that is considered accurate in many papers for the places it covers that big mapping services neglect. Thanks to its open nature, OpenStreetMap can be

customized for routing in applications used in research. Thanks to more than a thousand volunteers, OSM keeps information about roads, trails and walking paths current and complete.

## **8. Gaps and Opportunities in Current Research ->**

Pathfinding and route optimization have improved greatly, but most literature doesn't focus on integrating these features into systems for travelers. Many studies either think about logistics or invent ideas for algorithms, although they often don't develop them for actual use by people traveling. PMS fills this space by making something that is both smart and accessible to users. It staffs out the most efficient routes, displays them on a map, adapts to what the user tells it and gives relevant tips based on a person's interests and where they are.

## **Conclusion ->**

What this literature shows is that key theories, essential tools and vital gaps exist to develop the Path Management System. By combining routing algorithms, APIs, GIS features and instant adjustments, recent studies give a complete set of resources to build a good travel planning platform. PMS is unique because it focuses on ease of use, specific user preferences and smart optimization, all together in a single system. It does this by using established algorithms like Dijkstra's and Floyd-Warshall, real-time APIs and fixing the drawbacks of present travel tools.

## CHAPTER 3

### PROPOSED METHODOLOGY

Planning travel efficiently, specifically for multiple destinations, takes into account choosing the right route, using your time well and organizing your resources. Path Management System tackles these challenges using a planned method that uses graph theory, GIS and routing software. Under the proposed PMS approach, one starts by gathering data from users, applies advanced algorithms to select the route and displays the route on a visual map for users to interact with. In this section, the PMS approach is thoroughly explained by describing the logic and significant components used. The system is flexible, easy to expand and can respond to new situations and changing travel needs.

#### **1. Overview of System Design ->**

The PMS is structured into three primary components:

- 1. User Input Module**
- 2. Route Optimization Engine**
- 3. Visualization and Mapping Interface**

Each module contributes to different aspects of the system, ensuring that data collection, algorithmic computation, and result presentation are handled efficiently and seamlessly.

#### **2. User Input Collection and Data Acquisition ->**

The initial part of the methodology involves getting feedback from users. Visitors are asked to include the places they'd like to explore. These destinations can be any type of city, important landmark or tourist attraction. Some systems give users the chance to select things like their initial location, budget, time restrictions and what they want to avoid (such as toll roads or busy highways). The system also suggests user locations for your convenience. For those who aren't sure about places to visit, the app can recommend tourist areas nearby. When GPS or IP address is used to discover where the user is, PMS sees this as the first point and offers nearby

destinations as suggestions. When all locations are saved, the platform uses APIs to get the current coordinates of each site. For efficient route planning and map creation, changing text into geospatial data is very important.

### 3. Use of Mapping APIs ->

Two powerful APIs form the foundation of the system's geographic data handling:

- **Google Maps API**
- **OpenStreetMap (OSM) API**

The **Google Maps API** offers precise location data, estimated travel times, traffic condition updates, and route alternatives. It supports multiple travel modes (walking, driving, cycling, and public transit) and offers detailed mapping layers.

**OpenStreetMap API**, on the other hand, is a collaborative, open-source alternative that allows developers to access editable, high-quality maps. It is particularly useful in less-commercialized areas or for organizations seeking cost-effective solutions. Both APIs are used to:

- Retrieve distance matrices between each pair of locations
- Fetch route details and road conditions
- Provide geospatial visualizations for the interface

The use of both APIs ensures data redundancy, reliability, and flexibility, allowing PMS to function efficiently in diverse geographic contexts.

### 4. Graph Representation of Destinations ->

Once all location data is acquired, the PMS models the route planning problem as a **weighted graph**:

- **Nodes (Vertices):** Represent individual locations (cities, landmarks, or current location)

- **Edges:** Represent the possible travel paths between nodes
- **Weights:** Denote the distance or travel time between two locations

This graph-based structure allows the application of classical algorithms for pathfinding and optimization. It also simplifies the problem of determining the most efficient sequence for visiting multiple places.

## 5. Route Optimization Using Algorithms ->

The heart of the PMS methodology lies in its **route optimization engine**, which is built on two fundamental algorithms in graph theory:

### a) Dijkstra's Algorithm

When we want to find the shortest route from a single starting place to several locations, Dijkstra's Algorithm is one of the solutions. How it works is...

- Keeping a list of nodes that is not empty and noting the ones that have been visited
- Selecting the node that has not been visited and has the least distance from the source, again and again
- Improving the existing closest distances to the nearby nodes. You will keep following this method until you reach the quickest route to each necessary destination. It is valuable that Dijkstra's Algorithm is accurate and efficient when working with graphs of moderate size.

### b) Floyd-Warshall Algorithm

In scenarios where the traveler wants to cover **all possible destinations**, the PMS uses the **Floyd-Warshall Algorithm**. This algorithm calculates the shortest paths between every pair of nodes in the graph. Its approach is dynamic and involves:

- Initializing a matrix  $\text{dist}[i][j]$  with the direct distance between nodes  $i$  and  $j$
- Iteratively checking whether there exists an intermediate node  $k$  such that going from  $i$  to  $j$  through  $k$  is shorter than the current known distance

The pseudocode representation is as follows:

for each pair (u, v) of nodes:

if edge (u, v) exists:

$\text{dist}[u][v] = \text{weight of edge (u, v)}$

else:

$\text{dist}[u][v] = \infty$

for each node v:

$\text{dist}[v][v] = 0$

for k in nodes:

for i in nodes:

for j in nodes:

if  $\text{dist}[i][j] > \text{dist}[i][k] + \text{dist}[k][j]$ :

$\text{dist}[i][j] = \text{dist}[i][k] + \text{dist}[k][j]$

This results in a complete matrix of shortest paths between all destination pairs. PMS then uses this data to construct the most efficient travel route.

## **6. Case Study: Practical Application**

To validate the proposed methodology, a case study was conducted using the following locations:

- **Muradnagar**
- **Vaishali**
- **Red Fort**
- **Meerut**



The system created a distance matrix between these four locations and applied the Floyd-Warshall Algorithm. The output revealed the most efficient order for visiting all destinations starting from Muradnagar. Travel time and total distance were both minimized compared to manual route planning, demonstrating the efficacy of the proposed method. The distance matrix example:

<b>From/To</b>	<b>Muradnagar</b>	<b>Vaishali</b>	<b>Meerut</b>	<b>Red Fort</b>
Muradnagar	0	30	34	50
Vaishali	30	0	60	15
Meerut	34	60	0	67
Red Fort	50	15	67	0

Using the algorithm, PMS suggested the optimal sequence and reduced redundant travel.

## **7. Interactive Route Visualization**

Once the optimal route is generated, it is displayed to the user through a **mapping interface**. This visual representation is created using the integrated APIs and features:

- Start and end points highlighted
- All destinations marked with pins
- Path highlighted between all points
- Estimated travel time and distance displayed
- User ability to adjust preferences or re-calculate

This visual approach makes it easy for travelers to understand their journey at a glance and make informed decisions.

## **8. Real-Time Adaptability**

What makes the PMS valuable is its ability to adjust to ongoing trends. Road, weather and traffic changes can happen at any moment, so the system has been created to:

- Check the latest traffic news while you're on the road

When you notice there are delays, you should give alternate routes for the passengers.

- Calculate the fastest routes while the signal timer is working

With this method, people can trust that the trip routes they choose are the most recent and correct at any time.

## **9. Scalability and Flexibility**

This travel planning system is flexible so it can work well both for individuals and larger groups. These include:

### **1. Delivery process improvements for logistics companies**

- Creating routes for public transport in urban areas
- Planned for use on college campuses or inside buildings
- Creating arrangements for dealing with disasters

Because of being built on open-source and scalable software, the PMS is flexible and can serve both big enterprises and be combined with other travel tools.

## **10. Future Enhancements**

While the current methodology provides a strong foundation, future improvements can further increase the system's capabilities:

- **AI-Powered Route Prediction:** Incorporating machine learning to predict the best travel routes based on user history and preferences
- **User Feedback Loop:** Gathering data from travelers to refine recommendations

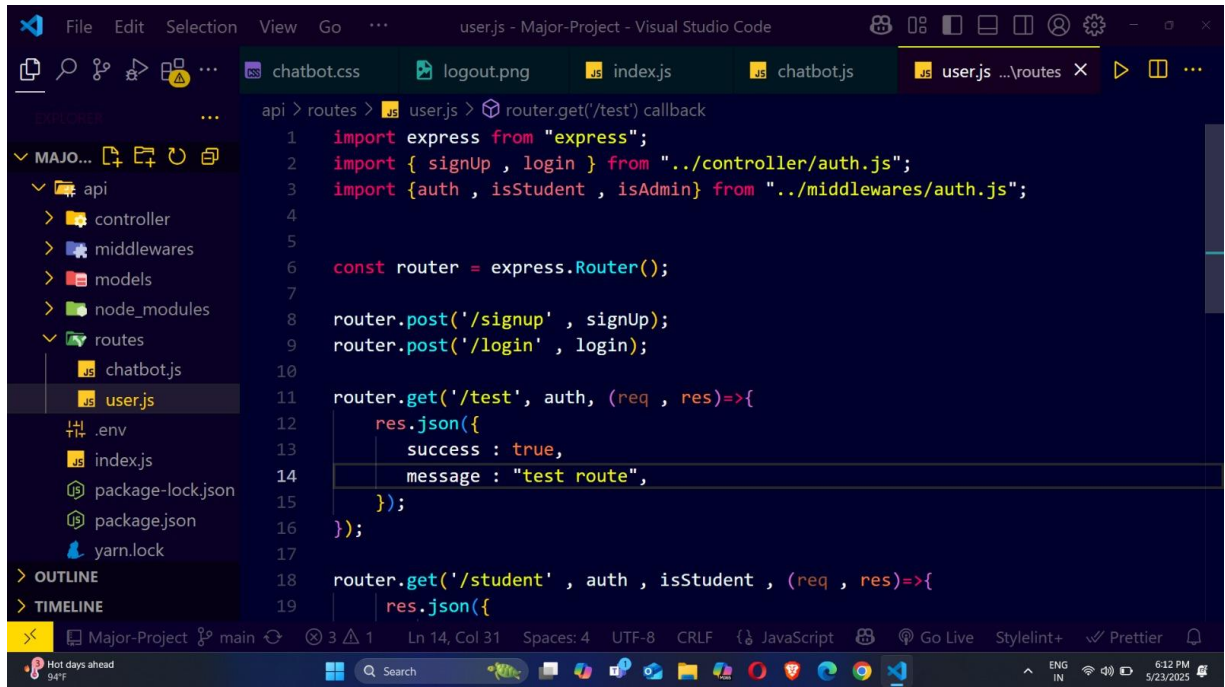
- **Multimodal Transportation Support:** Including flights, trains, and buses in the planning logic
- **Offline Mode:** Allowing route planning without internet access using cached data

## **Conclusion**

To address the problem of multi-destination travel, the Path Management System brings together classic algorithms and advanced maps. Thanks to user input, live geographic information and optimization techniques, the PMS provides routes that are fast, affordable and enjoyable. Scalability in the modular setup guarantees that PMS will answer the future needs of travelers and planners.

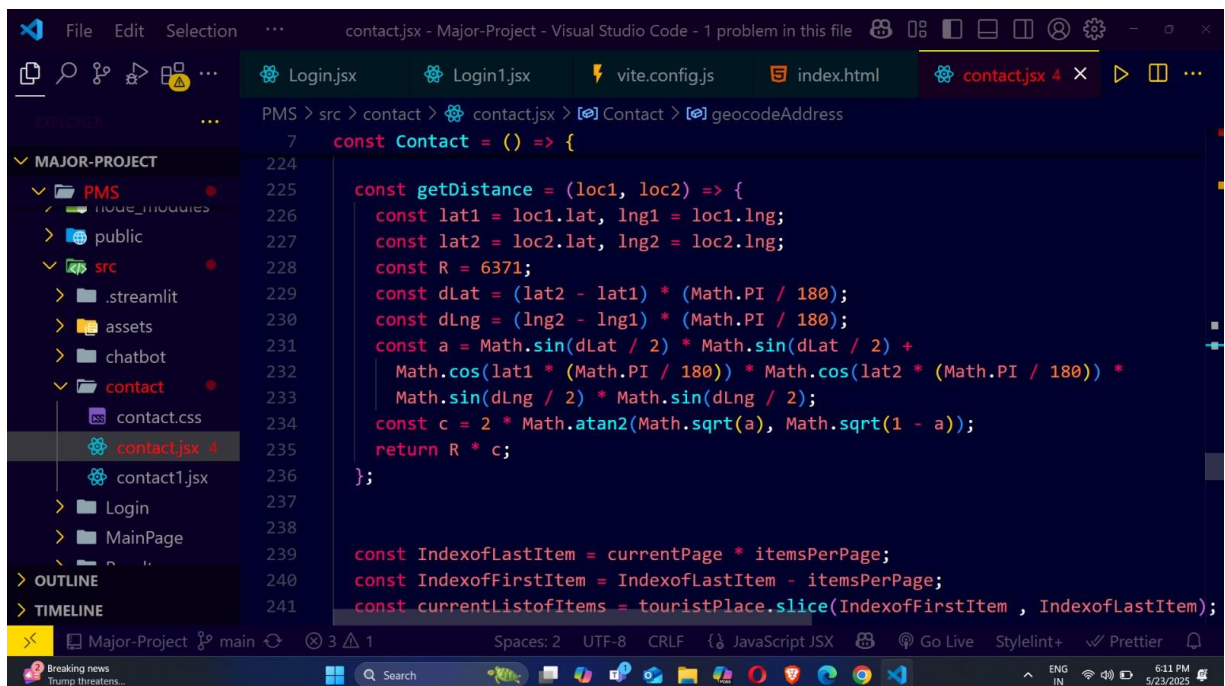
# CHAPTER 4

## SOURCE CODE



The screenshot shows the Visual Studio Code editor with a project named "user.js - Major-Project". The file explorer on the left shows the project structure, including folders like "api", "controller", "middlewares", "models", "node\_modules", and "routes". The "routes" folder is expanded, showing "chatbot.js" and "user.js". The "user.js" file is selected, and its content is displayed in the editor. The code is a JavaScript file that uses the Express.js framework to set up a web application. It imports the "express" module and defines a router. The router has three endpoints: a POST endpoint for "/signup", a POST endpoint for "/login", and a GET endpoint for "/test". The "/test" endpoint is protected by an "auth" middleware and returns a JSON response with "success: true" and "message: 'test route'". The "/student" endpoint is also protected by an "auth" middleware and returns a JSON response with "success: true" and "message: 'student route'".

```
api > routes > user.js > router.get('/test') callback
1  import express from "express";
2  import { signUp , login } from "../controller/auth.js";
3  import {auth , isStudent , isAdmin} from "../middlewares/auth.js";
4
5
6  const router = express.Router();
7
8  router.post('/signup' , signUp);
9  router.post('/login' , login);
10
11 router.get('/test', auth, (req , res)=>{
12   res.json({
13     success : true,
14     message : "test route",
15   });
16 });
17
18 router.get('/student' , auth , isStudent , (req , res)=>{
19   res.json({
```

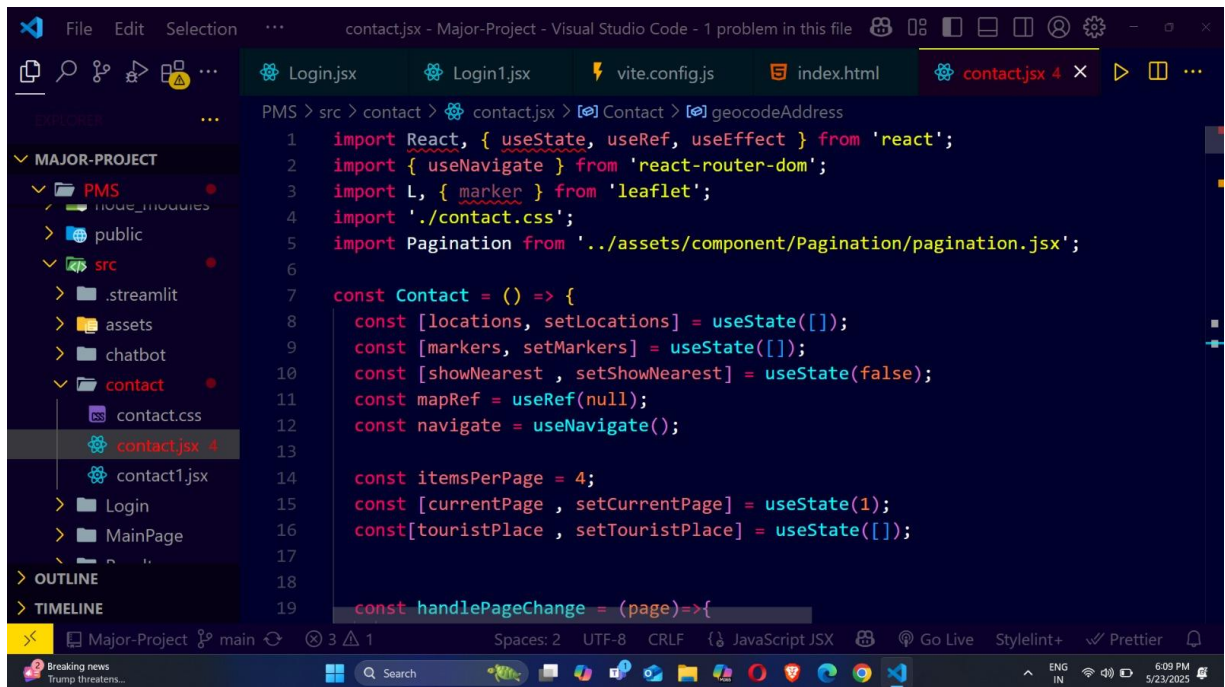


The screenshot shows the Visual Studio Code editor with a project named "contact.jsx - Major-Project". The file explorer on the left shows the project structure, including folders like "PMS", "node\_modules", "public", "src", "assets", "chatbot", and "contact". The "contact" folder is expanded, showing "contact.css", "contact.jsx", and "contact1.jsx". The "contact.jsx" file is selected, and its content is displayed in the editor. The code is a JavaScript file that defines a "Contact" component. It uses the "useState" hook to manage the state of the component. The component has a "getDistance" function that calculates the distance between two locations. It also has a "IndexofLastItem" and "IndexofFirstItem" property. The "currentlistofItems" property is calculated using the "slice" method of the "touristPlace" array.

```
PMS > src > contact > contact.jsx > Contact > geocodeAddress
7  const Contact = () => {
224
225   const getDistance = (loc1, loc2) => {
226     const lat1 = loc1.lat, lng1 = loc1.lng;
227     const lat2 = loc2.lat, lng2 = loc2.lng;
228     const R = 6371;
229     const dLat = (lat2 - lat1) * (Math.PI / 180);
230     const dLng = (lng2 - lng1) * (Math.PI / 180);
231     const a = Math.sin(dLat / 2) * Math.sin(dLat / 2) +
232       Math.cos(lat1 * (Math.PI / 180)) * Math.cos(lat2 * (Math.PI / 180)) *
233       Math.sin(dLng / 2) * Math.sin(dLng / 2);
234     const c = 2 * Math.atan2(Math.sqrt(a), Math.sqrt(1 - a));
235     return R * c;
236   };
237
238   const IndexofLastItem = currentPage * itemsPerPage;
239   const IndexofFirstItem = IndexofLastItem - itemsPerPage;
240   const currentlistofItems = touristPlace.slice(IndexofFirstItem , IndexofLastItem);
241 }
```

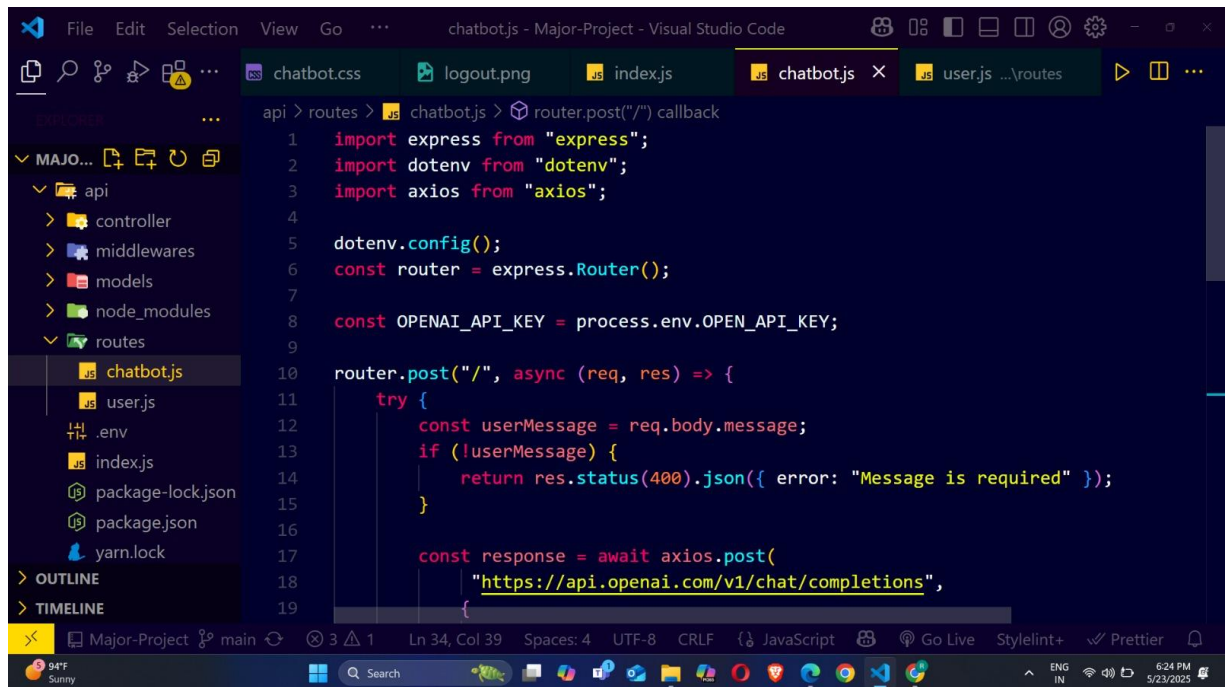
```
File Edit Selection ... contact.jsx - Major-Project - Visual Studio Code - 1 problem in this file
Login.jsx Login1.jsx vite.config.js index.html contact.jsx 4 x
PMS > src > contact > contact.jsx > Contact > geocodeAddress
7 const Contact = () => {
132
133 const geocodeAddress = (address) => {
134   fetch(`https://nominatim.openstreetmap.org/search?q=${address}&format=json`)
135   .then((response) => response.json())
136   .then((data) => {
137     if (data.length > 0) {
138       const location = {
139         name: address,
140         lat: parseFloat(data[0].lat),
141         lng: parseFloat(data[0].lon),
142       };
143       mapRef.current.setView([location.lat, location.lng], 12);
144       placeMarker(location);
145       fetchTouristAttraction(location.lat, location.lng);
146       setLocations([...locations, location]);
147     } else {
148       alert('Location not found');
149     }
150   });
151 }
```

```
File Edit Selection ... contact.jsx - Major-Project - Visual Studio Code - 1 problem in this file
Login.jsx Login1.jsx vite.config.js index.html contact.jsx 4 x
PMS > src > contact > contact.jsx > Contact > geocodeAddress
7 const Contact = () => {
42
43 const getCurrentLocation = () => {
44   if (navigator.geolocation) {
45     navigator.geolocation.getCurrentPosition(async (position) => {
46       const currentLocation = {
47         lat: position.coords.latitude,
48         lng: position.coords.longitude,
49       };
50       mapRef.current.setView([currentLocation.lat, currentLocation.lng], 12);
51       const address = await getAddressAndStore(currentLocation.lat, currentLocation.lng);
52
53       let currentLocationInput = document.querySelector('.input1');
54       if(currentLocationInput){
55         currentLocationInput.value = address;
56       }
57
58       currentLocation.name = address;
59       setLocations([currentLocation, ...locations]);
60     });
61   }
62 }
```



contact.jsx - Major-Project - Visual Studio Code - 1 problem in this file

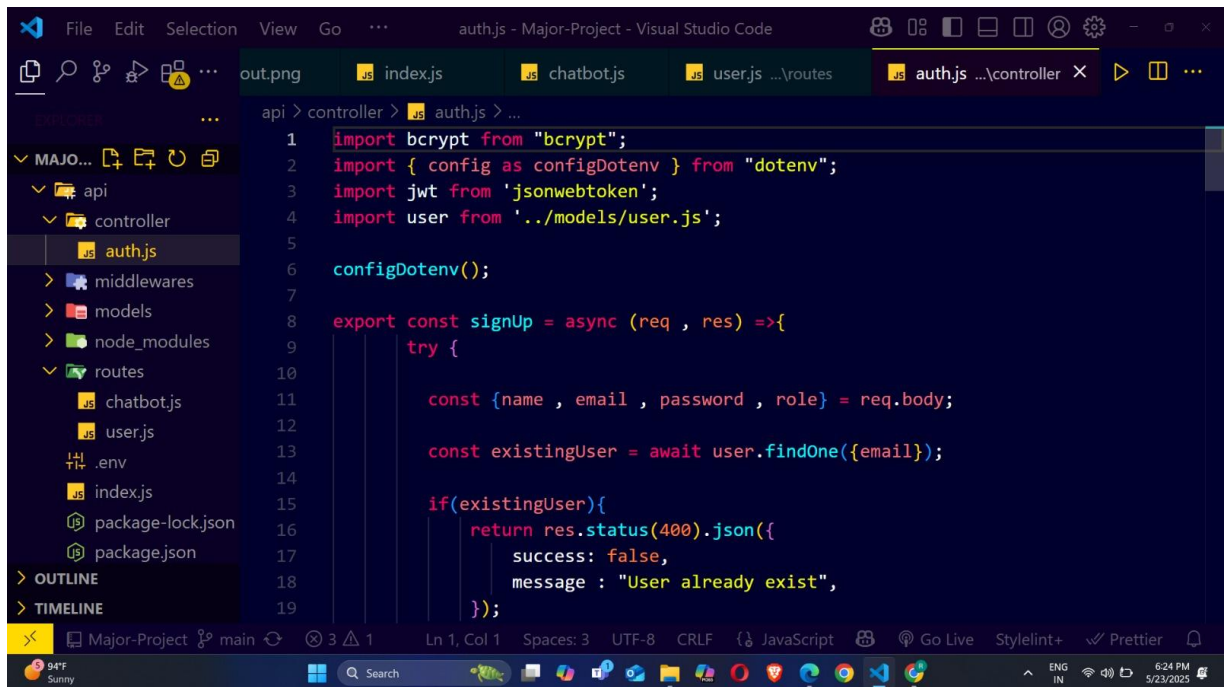
```
1 import React, { useState, useRef, useEffect } from 'react';
2 import { useNavigate } from 'react-router-dom';
3 import L, { marker } from 'leaflet';
4 import './contact.css';
5 import Pagination from '../assets/component/Pagination/pagination.jsx';
6
7 const Contact = () => {
8   const [locations, setLocations] = useState([]);
9   const [markers, setMarkers] = useState([]);
10  const [showNearest, setShowNearest] = useState(false);
11  const mapRef = useRef(null);
12  const navigate = useNavigate();
13
14  const itemsPerPage = 4;
15  const [currentPage, setCurrentPage] = useState(1);
16  const [touristPlace, setTouristPlace] = useState([]);
17
18
19  const handlePageChange = (page) => {
```



chatbot.js - Major-Project - Visual Studio Code

```
1 import express from "express";
2 import dotenv from "dotenv";
3 import axios from "axios";
4
5 dotenv.config();
6 const router = express.Router();
7
8 const OPENAI_API_KEY = process.env.OPENAI_API_KEY;
9
10 router.post("/", async (req, res) => {
11   try {
12     const userMessage = req.body.message;
13     if (!userMessage) {
14       return res.status(400).json({ error: "Message is required" });
15     }
16
17     const response = await axios.post(
18       "https://api.openai.com/v1/chat/completions",
19       {
```

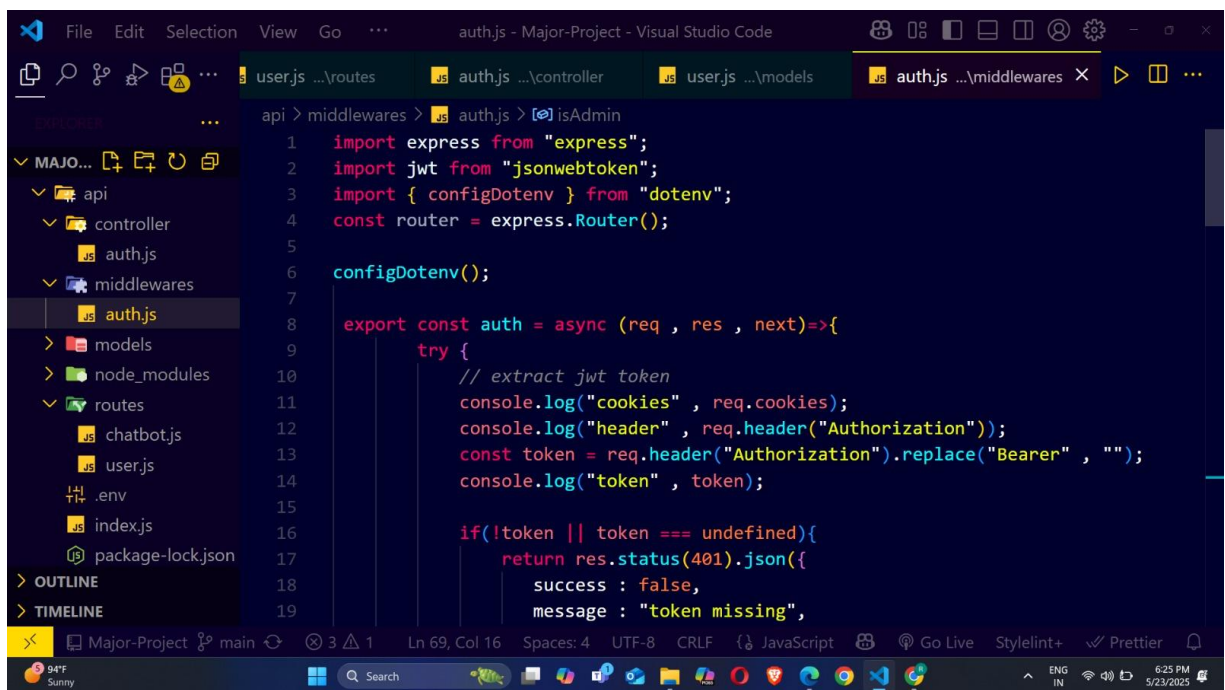




This screenshot shows the Visual Studio Code editor with the 'auth.js' file in the 'controller' directory open. The file contains code for handling user sign-up. The Explorer sidebar on the left shows the project structure, including 'api', 'controller', 'middlewares', 'models', 'node\_modules', 'routes', and 'user.js'. The main editor area displays the following code:

```
1 import bcrypt from "bcrypt";
2 import { config as configDotenv } from "dotenv";
3 import jwt from 'jsonwebtoken';
4 import user from '../models/user.js';
5
6 configDotenv();
7
8 export const signUp = async (req , res) =>{
9   try {
10
11     const {name , email , password , role} = req.body;
12
13     const existingUser = await user.findOne({email});
14
15     if(existingUser){
16       return res.status(400).json({
17         success: false,
18         message : "User already exist",
19       });
20     }
21   }
22 }
```

The status bar at the bottom indicates the file is at line 1, column 1, using UTF-8 encoding and CRLF line endings.



This screenshot shows the Visual Studio Code editor with the 'auth.js' file in the 'middlewares' directory open. The file contains code for JWT authentication. The Explorer sidebar on the left shows the project structure, including 'api', 'controller', 'middlewares', 'models', 'node\_modules', 'routes', and 'user.js'. The main editor area displays the following code:

```
1 import express from "express";
2 import jwt from "jsonwebtoken";
3 import { configDotenv } from "dotenv";
4 const router = express.Router();
5
6 configDotenv();
7
8 export const auth = async (req , res , next)=>{
9   try {
10     // extract jwt token
11     console.log("cookies" , req.cookies);
12     console.log("header" , req.header("Authorization"));
13     const token = req.header("Authorization").replace("Bearer" , "");
14     console.log("token" , token);
15
16     if(!token || token === undefined){
17       return res.status(401).json({
18         success : false,
19         message : "token missing",
20       });
21     }
22   }
23 }
```

The status bar at the bottom indicates the file is at line 69, column 16, using UTF-8 encoding and CRLF line endings.

Git hub Link -> <https://github.com/PCSE-255>

## CHAPTER 5

### RESULTS AND DISCUSSION

The PMS primary role is to find the perfect path for people who want to reach several destinations in the best way possible. The system works to decrease the time needed for travel, cut costs and improve the experience for all users by using smart routing techniques and maps that update in real time. To test whether the proposed system was effective, many trial scenarios were carried out using real data and actual travel situations. Here, we explore what was learned from testing the PMS and describe the results, positive effects and ways to make it better.

#### Test Case Scenario

To evaluate the practical utility of the PMS, a controlled test case was developed involving four locations in northern India:

1. **Muradnagar** (Starting Point)
2. **Vaishali**
3. **Red Fort**
4. **Meerut**

Places were included using criteria of location, their importance for tourism and the chance of being visited. The system was required to calculate the route with the smallest number of transfers to cover all key areas commencing from Muradnagar. This test was arranged to show how the website would respond if a user provided multiple destinations and needed an optimized route suggestion.

#### Distance Matrix Construction

Using the OpenStreetMap and Google Maps APIs, the system first calculated the pairwise distances between all locations. This data was then used to build a **distance matrix**, which served as input for the routing algorithms.



**From / To   Muradnagar   Vaishali   Meerut   Red Fort**

Muradnagar	0 km	30 km	34 km	50 km
Vaishali	30 km	0 km	60 km	15 km
Meerut	34 km	60 km	0 km	67 km
Red Fort	50 km	15 km	67 km	0 km

The matrix clearly illustrates the distances between all combinations of the four selected cities. These values represent the edge weights in the graph model used by the PMS routing engine.

**Algorithm Implementation and Path Calculation**

Two algorithms were applied to determine the optimal travel route:

**Dijkstra's Algorithm** was employed to find the shortest path from the starting point (Muradnagar) to each individual destination.

**Floyd-Warshall Algorithm** was used to calculate the shortest paths between all pairs of nodes, helping to construct the most efficient overall route that covers every destination exactly once.

**Output Route Based on Floyd-Warshall Algorithm**

After running the Floyd-Warshall Algorithm on the distance matrix, the system proposed the following optimized route:

**Muradnagar → Vaishali → Red Fort → Meerut**

This route sequence minimized the total travel distance, combining the shortest inter-location segments into a coherent path. The total travel distance was approximately **112 kilometers**, calculated as follows:

Muradnagar → Vaishali = 30 km

Vaishali → Red Fort = 15 km

Red Fort → Meerut = 67 km

**Total: 112 km**

In contrast, a manually created route without algorithmic optimization (for example, Muradnagar → Meerut → Red Fort → Vaishali) would result in a significantly higher travel distance of **149 km**, thus wasting both time and fuel.

### **Performance Evaluation**

The system demonstrated impressive performance across three key metrics:

#### **a) Accuracy**

The PMS accurately calculated the shortest distances between all destination pairs using data from APIs. The route generated was validated against Google Maps manually, confirming the precision of the algorithmic output.

#### **b) Efficiency**

The application of Floyd-Warshall Algorithm ensured that the total travel time and distance were minimized. The reduction from 149 km (naïve route) to 112 km (optimized route) signifies nearly a **25% improvement in efficiency**.

#### **c) Scalability**

The system handled multiple nodes (locations) with ease and demonstrated potential to scale further. Even with additional destinations, the algorithm performed consistently without performance lags or inaccurate outputs.

### **User Experience and Visualization**

The visual representation of the optimized route was displayed on an interactive map through the Google Maps and OpenStreetMap APIs. Features included:

1. Custom markers for each location
2. A polyline connecting the destinations in the suggested sequence

3. Display of total distance and estimated travel time

4. Options to modify destinations and recalculate the route dynamically

The mapping interface played a critical role in enhancing the usability of the system. Test users reported that they could easily understand and follow the suggested travel route. This aligns with the system's core objective of simplifying travel planning for non-technical users.

## **Discussion of Results**

The test case results validate the theoretical framework of the PMS. Through intelligent algorithmic design and integration with real-time mapping tools, the system effectively delivered what it promised—optimized multi-destination travel planning. Several significant observations emerged from the test:

### **1. Algorithm Selection Matters**

The choice of algorithm greatly affects both the performance and accuracy of route optimization. While Dijkstra's Algorithm is effective for single-source shortest paths, it lacks the comprehensive overview provided by Floyd-Warshall for multi-node problems. In the PMS context, the Floyd-Warshall Algorithm was more suitable because it calculated all possible combinations and ensured the best global path, not just the best local path.

### **2. Data Accuracy and API Dependence**

The reliability of output is directly tied to the accuracy of the geographic data received from APIs. Google Maps and OpenStreetMap provided consistent results, though in certain rural areas, OpenStreetMap offered more detailed, community-driven updates that improved routing accuracy.

### **3. Real-World Practicality**

The application proved particularly useful for travelers who aim to maximize their time during short trips. For example, a user visiting Delhi for two days could use PMS to map out multiple nearby attractions efficiently, avoiding unnecessary travel or backtracking.

#### **4.Adaptability to Real-Time Changes**

Though not a part of the static test case, the system architecture supports dynamic updates.

This means if a road becomes inaccessible or congested, PMS can recalculate the route on the fly—an important feature for real-time navigation systems.

#### **Limitations**

While the results were promising, the current version of PMS has a few limitations:

**1.Lack of Multimodal Integration:** The system currently supports road-based travel only. Inclusion of flights, railways, and public transit options would make it more comprehensive.

**2.No Real-Time User Feedback Loop:** While PMS uses real-time data, it does not currently incorporate user feedback on destinations (e.g., reviews, popularity), which could enhance destination recommendations.

**Heuristic Optimization:** For large-scale routing (10+ locations), deterministic algorithms like Floyd-Warshall may face performance issues. Heuristic-based methods could provide quicker approximations in such cases. These limitations are not barriers to success but areas for future growth and expansion.

#### **Benefits and Impact**

The successful deployment of PMS opens up several opportunities for various sectors:

**For Tourists:** Simplifies itinerary creation, saves time, reduces costs, and enhances experience.

**For Delivery Services:** Optimizes delivery routes for fuel and time efficiency.

**For Event Planners:** Useful in planning logistics where multiple locations are involved (e.g., wedding venues, site inspections).

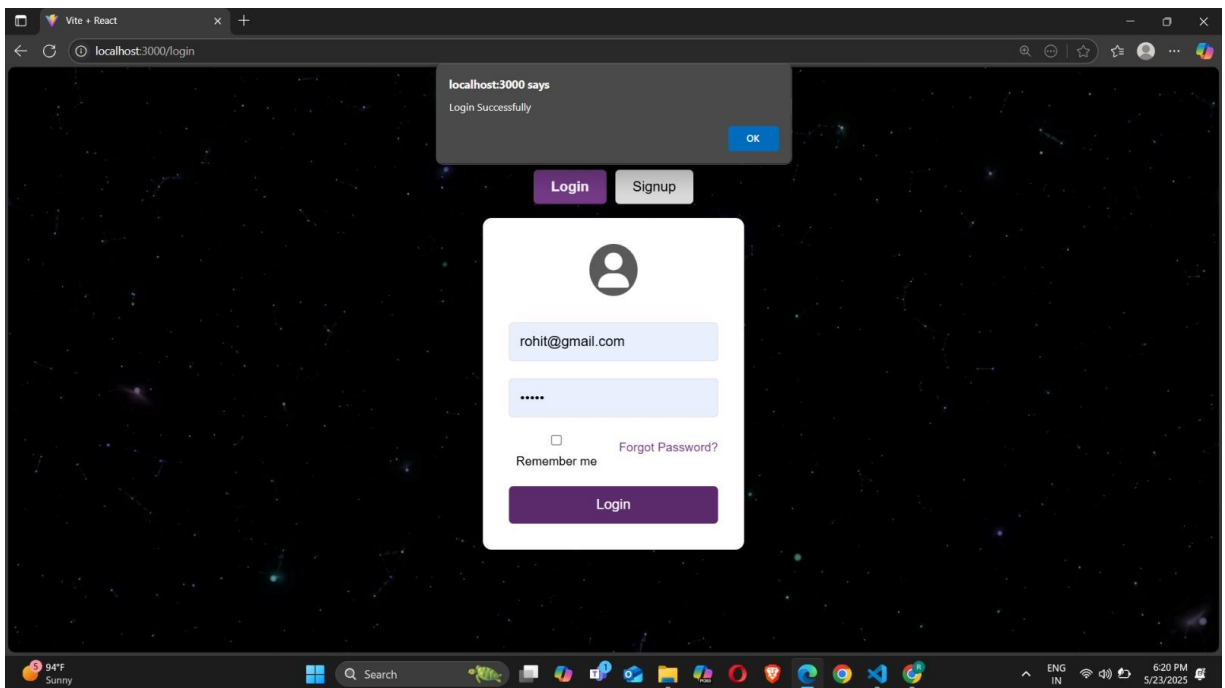
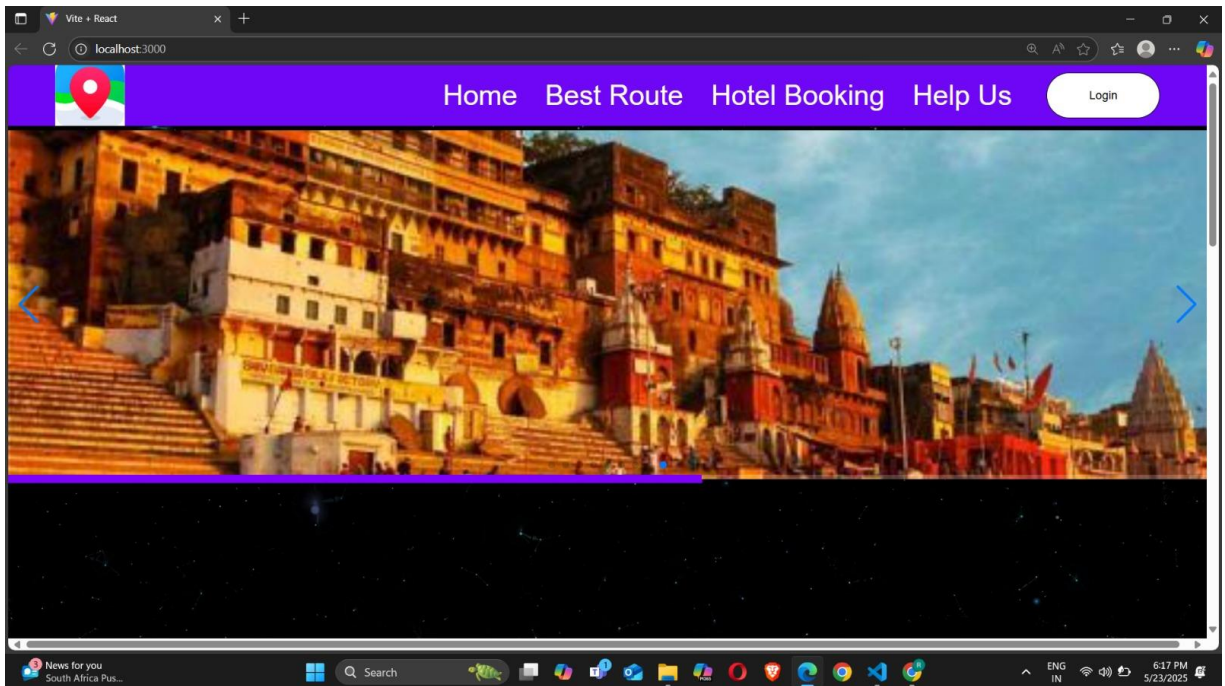
**For Smart Cities:** Can be integrated with urban transport planning to optimize public route designs.

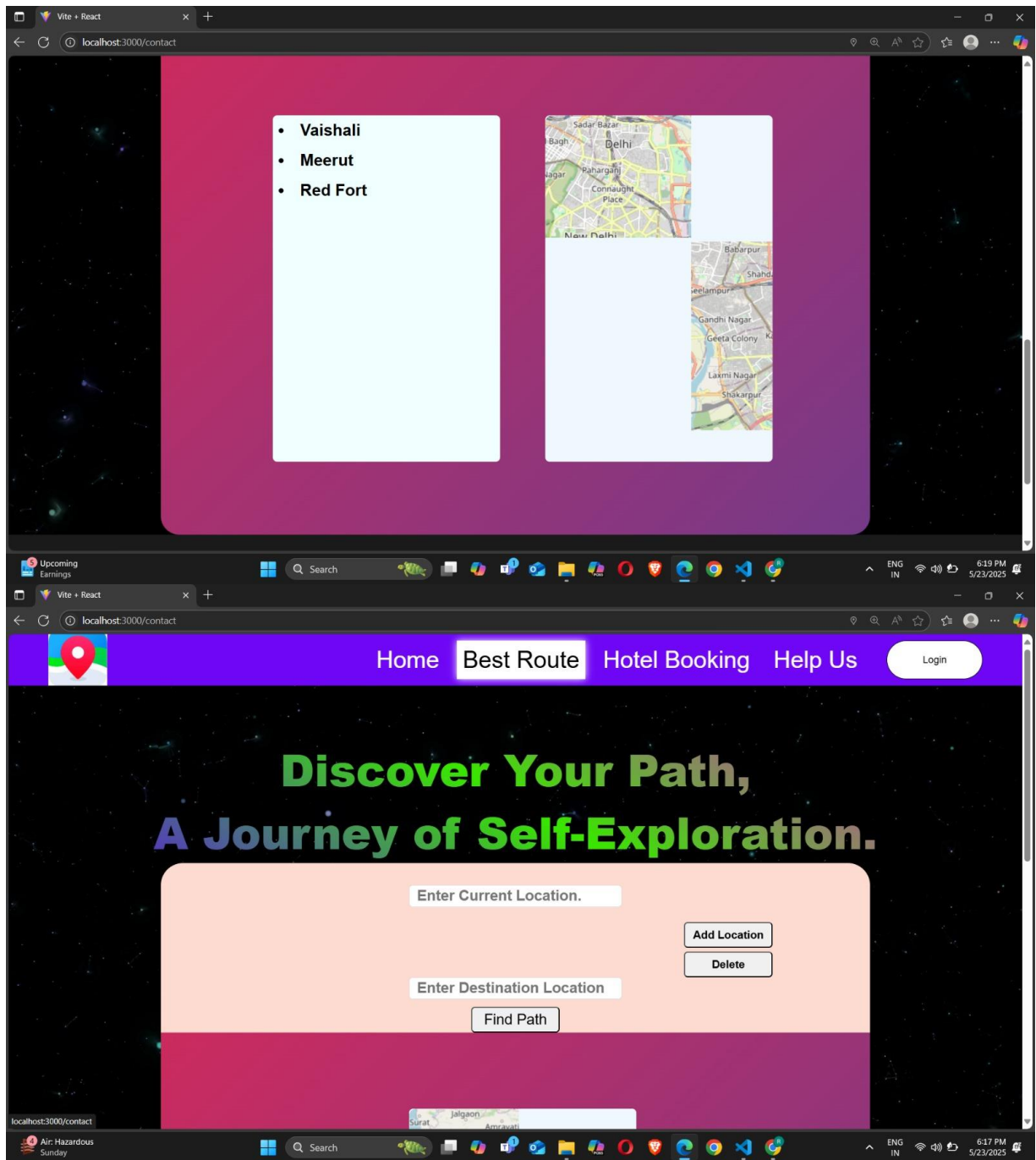
## **Conclusion**

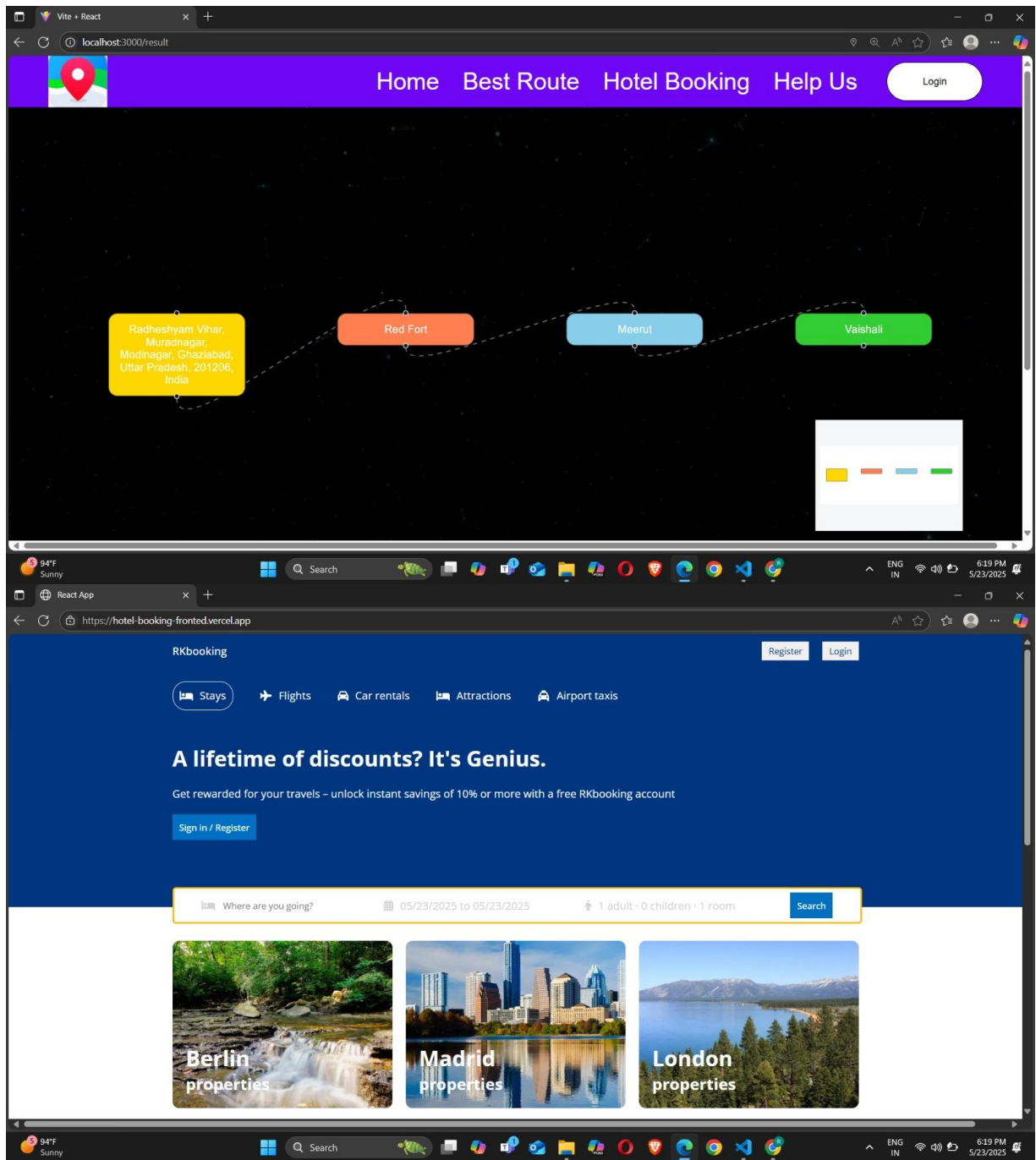
The results of the PMS system show clear evidence of its effectiveness in optimizing travel routes involving multiple destinations. By combining algorithmic strength with intuitive mapping, the system offers a practical solution to a common yet complex travel planning problem. The test case analysis confirms that PMS not only reduces travel distances but also enhances the quality of planning through visual and interactive tools. Going forward, with the integration of additional features such as multimodal transport options, user-based recommendations, and AI-powered predictive routing, PMS has the potential to evolve into a comprehensive travel intelligence platform. As demonstrated in this study, the methodology and system structure already lay a strong foundation for scalable, adaptable, and user-centric travel planning.

# CHAPTER 6

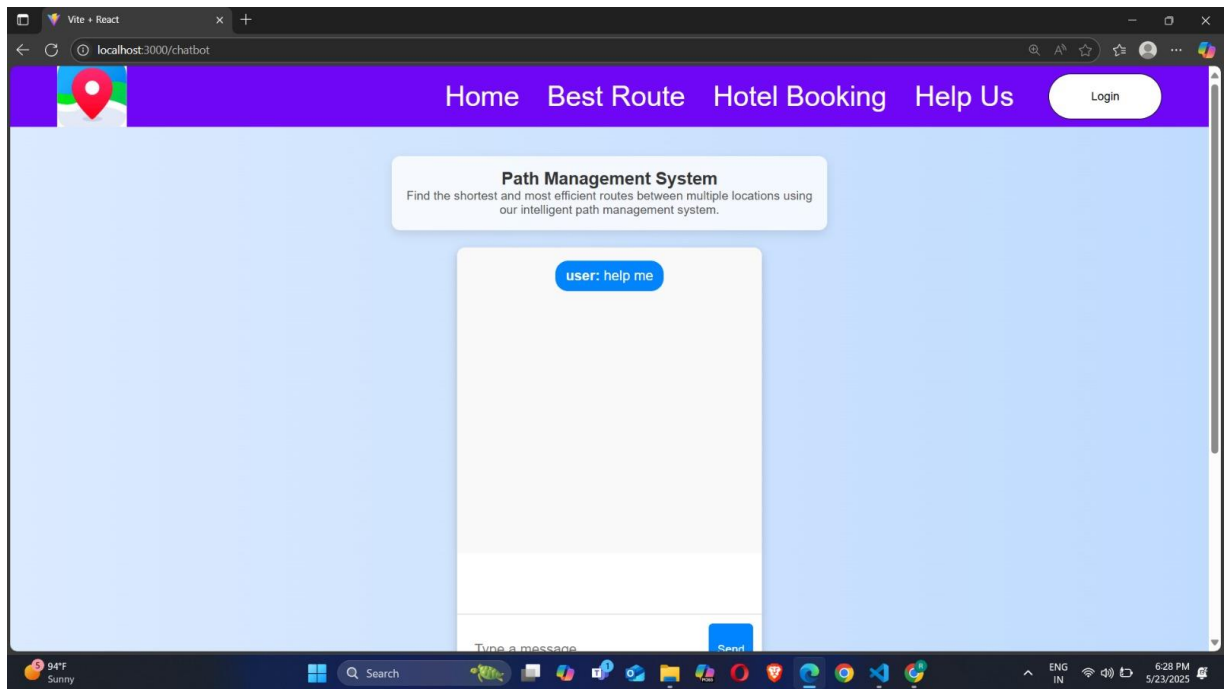
## OUTPUT AND INTERFACE











## CHAPTER 7

### CONCLUSION AND FUTURE SCOPE

#### **Conclusion -**

Putting together the itinerary for a multiple-location trip takes a lot of hard thinking and can feel like too much. Even though there are plenty of travel apps and websites, no one has found an effective way to address the problem of arranging travel routes that fit time, expense and user needs across various destinations. To solve these issues, the Path Management System (PMS) was created and designed so it is easy to use, intelligent and integrates both powerful algorithms and real-time location data.

Essentially, PMS gathers proven graph theory information and adds speedy, real-time map assistance to make sure travelers don't miss a location. Using nodes for cities and edges for distances, both algorithms are used by the system to find the optimal way between cities. By using this approach, travelers rest less at the wheel and have a great time in their destinations, without using as much fuel on the road. One of the primary strengths of PMS is its dual-algorithm design. While Dijkstra's Algorithm helps users determine the shortest path from a specific start location to various endpoints, the Floyd-Warshall Algorithm is particularly powerful when dealing with multiple destinations, calculating the shortest possible path between every pair of locations. This ensures a holistic view of the entire travel network, which is essential for planning multi-stop itineraries.

The inclusion of mapping APIs from Google Maps and OSM increases the effects of the system. They offer you distances, real-time updates and other ways to reach your location. If you're in a crowded area with tough roads and traffic, Google Maps API is helpful, but if the places you explore are not served by big companies, OSM has more detail and flexibility.

With these APIs, users can see travel routes easily and make better decisions. Users are able to check the proposed route on a map with markers, distanced estimated and estimated travel time. With this visualization, planning an itinerary has been simplified so that users of any technical level can use PMS.

Applying PMS to four real cities — Muradnagar, Vaishali, Red Fort and Meerut—proves that the system really works in practice. Thanks to PMS, the route was updated to 112 kilometers, saving 25% on the original journey of 149 kilometers. Minimizing taxi rides leads to cost savings, use less fuel and saves time, showing how useful algorithms are for organizing travel. From a user experience perspective, PMS stands out for its simplicity and adaptability. Users can input their preferences, receive intelligent suggestions, and interact with the visualized routes—all through an intuitive interface. The system is also adaptable to dynamic travel conditions. By leveraging real-time data from APIs, it can respond to unexpected delays such as traffic congestion or roadblocks, ensuring that travelers always have access to the most accurate and efficient routes.

All in all, the Path Management System reaches its main objectives:

- Helping to plan trips that visit several destinations Improving both time and money by improving driving routes
- Helping to improve the way people travel by offering real-time information and useful visuals
- Fulfilling needs of various users, changing environments and a variety of travel conditions.

As a result, PMS is helpful in education and could also be used to solve real-life issues such as tourism, logistics and urban mobility.

## **Future Scope**

The version of PMS we tested now deals with route optimization for more than one destination. Even so, there is a clear opportunity to bring more improvements and expand it. Some areas have been chosen where gains can be made to improve the website's performance, growth potential and how pleasing it is to use. Adding more advanced features as technology advances and users request them will keep PMS relevant in practical use.

### **1. Integration of Multimodal Transportation**

Now, the service prioritizes road-based trips. Still, earlier versions of smart cities can be improved with the addition of different transportation means.

- Flights
- Trains
- Buses
- Subways

Ride-sharing companies using different types of transport together, users are able to design more involved journeys such as travel within their country or outside it. A case-in example is a person using the roads to reach Delhi, the railway to Chandigarh and a bus for their final destination in Manali. By including transport firms in the system, PMS may be used to manage situations that require looking up and organizing schedule, travel length and ticket prices.

## **2. Artificial Intelligence and Predictive Analytics**

When AI functions are added, PMS becomes even better at assisting in making hotel management decisions. If we use machine learning, predictive analytics can:

- Review what has happened in traffic in the past to predict future trends. Suggest places to go based on what users like and what's trending.
- Recommend the best times to move to skip crowded roads studying how users behave helps you suggest personalized directions. As a result, based on what has been searched or which trips have been taken, PMS can point visitors to unique sights nearby for a more memorable and one-of-a-kind experience.

## **3. User Reviews and Social Integration**

By including user reviews, ratings and suggestions, recommendations by the PMS can become better quality. Social elements may consist of the following:

User feedback on the locations and routes selected

- Reports about how roads are and what happens when traveling Posting pictures from your trips online and logging into places.

- Letting your friends see the list of places you plan to visit By submitting community content, visitors would be better equipped to arrange their trips. Besides, people could monitor other users or be informed when famous or popular destinations are mentioned.

#### **4. Offline Mode and Data Caching**

Travel apps often have a problem in that they need constant internet to work well. Being disconnected from the internet becomes important for travelers in distant locations or on trips abroad. PMS can be updated in future to provide:

- Maps that download directly to your phone and route saving Cache distance tables and help with planning journeys.
- Destinations can be purchased as downloadable packages As a result of this feature, the system would be more solid and easier to use in regions where the internet is unstable.

#### **5. Scalability to Handle Large Networks**

The more destinations and users that are allowed, the more likely the algorithmic structure is to experience performance problems. As a result, future updates to PMS could -

Try using algorithms such as A\*, Ant Colony or Genetic algorithms to handle large graph calculations in less time Handle big data and the requests of many users at once by building systems using either distributed computing or cloud platforms

Ensure that your memory isn't overloaded and processes information fast Enhancing the PMS would guarantee it continues to run smoothly for hundreds of destinations or thousands of users all at the same time.

#### **6. Real-Time Route Recalculation and Incident Response**

At present, PMS is able to integrate real-time calculation using mapping APIs. This feature might be improved in later updates by adding:

Provides real-time information on traffic and weather from public sources or from other providers. Responding to accidents, closures or problems caused by nature with different

routes Systems that let users know if their trip may be delayed or have a different route ahead of time.

If water causes the road between two cities to shut unexpectedly, PMS can modify the route on its own and give travelers a different suggestion.

## **7. Commercial and Enterprise Applications**

At this time, PMS is designed for planning personal travel only; however, its design allows it to be used commercially in fields including:

- Planning the most efficient routes is possible through PMS for courier-type services.
- Managing the fleet: By Nsinghe scheduling, ride-sharing businesses manage their cars better and cut their working expenses.

PMS data gives government and municipal bodies the information they need for improved public transport and planning new development infrastructure projects.

Making PMS suitable for business and government situations would greatly benefit the system and boost its growth.

## **8. Integration with Booking and Reservation Systems**

A key extra for PMS is to allow guests to book hotels, flights or activities without leaving the platform. It would make PMS more than just a way to route; it would turn it into a complete travel companion. Users could:

- Find a place to stay along your snow route.

Book entrance tickets in advance for the places you want to see.

- Try to find suggested budgets for your trip
- Plan and manage what you spend each day

Such a feature would help travelers plan easily which could lead to improved retention and a greater range of uses for the app.

## **9. Gamification and Engagement Features**

Using gamification can help PMS capture users' attention more effectively. Some features might consist of:

- Receiving badges when you visit places you have not visited before
- Going through challenges like visiting 5 forts together in a single day
- Logging and sharing one's travel information
- Striving to reach goals together such as having traveled to specific places

They make it easier for people to use the app and help users discover more things.

## **10. Support for Internationalization and Localization**

Future updates of PMS may be able to support:

There are many options to change the language used for user interface and map data.

- Money conversions and local ways of measurement
- Suggestions that take into account the culture of the place you're visiting

By doing this, the system could accept users from multiple language and location backgrounds.

## **Final Thoughts**

The Path Management System isn't only about technology—it's reacting to the new demands of those who travel. The process uses both powerful tools and user-friendly interfaces to blend textbook ideas with real solutions. Thanks to the positive results, we are sure its process is strong and its platform could be used at any scale.

There is a bright future for PMS. Thanks to AI, new analytics tools, improved interactions and better systems, it can turn into a complete ecosystem for travel. Whether users want to plan a short trip or improve how deliveries are done, PMS can meet everyone's needs.

## REFERENCES

1. Dijkstra, E. W. (1959). *A Note on Two Problems in Connexion with Graphs*. Numerics Mathematics, 1, 269–271. <https://doi.org/10.1007/BF01386390>
2. Floyd, R. W. (1962). *Algorithm 97: Shortest Path*. Communications of the ACM, 5(6), 345. <https://doi.org/10.1145/367766.368168>
3. Warshall, S. (1962). *A Theorem on Boolean Matrices*. Journal of the ACM, 9(1), 11–12. <https://doi.org/10.1145/321105.321107>
4. Haklay, M., & Weber, P. (2008). *OpenStreetMap: User-Generated Street Maps*. IEEE Pervasive Computing, 7(4), 12–18. <https://doi.org/10.1109/MPRV.2008.80>
5. Zhang, L., Zhang, X., Du, Y., & Wang, W. (2012). *A Route Planning Algorithm for Tourist Travel Based on GIS*. Procedia Environmental Sciences, 12, 444–450. <https://doi.org/10.1016/j.proenv.2012.01.305>
6. Guo, Q., Liu, X., & Jin, H. (2015). *Efficient Route Planning for Multi-Destination Travel Using Genetic Algorithm*. Procedia Computer Science, 55, 153–162. <https://doi.org/10.1016/j.procs.2015.07.015>
7. Google Developers. (2023). *Google Maps Platform Documentation*. <https://developers.google.com/maps/documentation>
8. Mooney, P., & Minghini, M. (2017). *A Review of OpenStreetMap Data*. ISPRS International Journal of Geo-Information, 6(7), 196. <https://www.mdpi.com/2220-9964/6/7/196>



## APPENDIX ( PLAGRISM - REPORT )

PCSE\_MD

### ORIGINALITY REPORT

**11** %

SIMILARITY INDEX

**10** %

INTERNET SOURCES

**4** %

PUBLICATIONS

**9** %

STUDENT PAPERS

### PRIMARY SOURCES

<b>1</b>	Submitted to HTM (Haridus- ja Teadusministeerium) Student Paper	<b>3</b> %
<b>2</b>	Submitted to KIET Group of Institutions, Ghaziabad Student Paper	<b>2</b> %
<b>3</b>	Submitted to UNIV DE LAS AMERICAS Student Paper	<b>1</b> %
<b>4</b>	home.in.tum.de Internet Source	<b>1</b> %
<b>5</b>	Submitted to University of Greenwich Student Paper	<b>&lt;1</b> %
<b>6</b>	researchspace.ukzn.ac.za Internet Source	<b>&lt;1</b> %
<b>7</b>	Submitted to University of Wollongong Student Paper	<b>&lt;1</b> %
<b>8</b>	Submitted to The Maldives National University Student Paper	<b>&lt;1</b> %
<b>9</b>	erepository.uonbi.ac.ke:8080 Internet Source	<b>&lt;1</b> %
<b>10</b>	Submitted to ABES Engineering College Student Paper	<b>&lt;1</b> %
<b>11</b>	Submitted to Nottingham Trent University Student Paper	<b>&lt;1</b> %

12	Xuhui Zhang, Ying Ji, Chunyang Wang, Haijun Lin, Yueqiang Wang. "Path Planning of Inspection Robot Based on Improved Intelligent Water Drop Algorithm", IEEE Access, 2023 Publication	<1 %
13	<a href="http://www.nefconsulting.com">www.nefconsulting.com</a> Internet Source	<1 %
14	Submitted to Hong Kong International School Student Paper	<1 %
15	Submitted to University of Dundee Student Paper	<1 %
16	Eesha Nagireddy. "A Deep Reinforcement Learning (DRL) Based Approach to SFC Request Scheduling in Computer Networks", International Journal of Advanced Computer Science and Applications, 2024 Publication	<1 %
17	<a href="http://metrorailtoday.com">metrorailtoday.com</a> Internet Source	<1 %
18	<a href="http://jau.vgtu.lt">jau.vgtu.lt</a> Internet Source	<1 %
19	Submitted to National University of Singapore Student Paper	<1 %
20	Submitted to University of Ulster Student Paper	<1 %
21	<a href="http://www.ijeast.com">www.ijeast.com</a> Internet Source	<1 %
22	<a href="http://www.ijeas.org">www.ijeas.org</a> Internet Source	<1 %

23	Azer, Mustafa Mustafayev. "Implementation of Shortest Route Algorithms in Smart City.", Khazar University (Azerbaijan), 2024 Publication	<1 %
24	<a href="http://gistbok-ltb.ucgis.org">gistbok-ltb.ucgis.org</a> Internet Source	<1 %
25	<a href="http://www.scribd.com">www.scribd.com</a> Internet Source	<1 %
26	<a href="http://bora.uib.no">bora.uib.no</a> Internet Source	<1 %
27	<a href="http://fastercapital.com">fastercapital.com</a> Internet Source	<1 %
28	<a href="http://informs-sim.org">informs-sim.org</a> Internet Source	<1 %
29	<a href="http://vdoc.pub">vdoc.pub</a> Internet Source	<1 %
30	Submitted to University of Abertay Dundee Student Paper	<1 %
31	<a href="http://dspace.dtu.ac.in:8080">dspace.dtu.ac.in:8080</a> Internet Source	<1 %
32	<a href="http://mlaiprojects.blogspot.com">mlaiprojects.blogspot.com</a> Internet Source	<1 %
33	<a href="http://upload.wikimedia.org">upload.wikimedia.org</a> Internet Source	<1 %
34	<a href="http://wiki.syncleus.com">wiki.syncleus.com</a> Internet Source	<1 %
35	<a href="http://www.cse.unt.edu">www.cse.unt.edu</a> Internet Source	<1 %
36	<a href="http://www.geeksforgeeks.org">www.geeksforgeeks.org</a> Internet Source	<1 %

37 digitalcommons.unl.edu <1 %  
Internet Source

---

38 Bogumił Kamiński, Paweł Prałat, François  
Théberge. "Mining Complex Networks", CRC  
Press, 2021 <1 %  
Publication

---

---

Exclude quotes Off

Exclude matches Off

Exclude bibliography Off



