PROJECT REPORT ON

"GPS Navigation Tracker"

Submitted By:

Raja Kumar, UID- 24MCA20229

Under the Guidance of :

Mrs. Deepali Saini

April, 2025

University Institute of Computing

Chandigarh University,

Mohali, Punjab

CERTIFICATE

This is to certify that Raja Kumar (UID: - 24MCA20229) have successfully completed the project title “Admin Panel with CRUD” at University Institute of Computing under my supervision and guidance in the fulfilment of requirements of fourth semester, Master of Computer Application. Of Chandigarh University, Mohali, Punjab.

Dr. Krishan Tuli
Head of the Department
University Institute of Computing

Mrs. Deepali Saini
Project Guide Supervisor
University Institute of Computing

ACKNOWLEDGEMENT

We deem it a pleasure to acknowledge our sense of gratitude to our project guide Mrs Sweta Mam under whom we have carried out the project work. Her incisive and objective guidance and timely advice encouraged us with constant flow of energy to continue the work. We wish to reciprocate in full measure the kindness shown by Dr. Krishan Tuli (H.O.D, University Institute of Computing) who inspired us with his valuable suggestions in successfully completing the project work. We shall remain grateful to Dr. Manisha Malhotra, Additional Director, University Institute of Technology, for providing us a strong academic atmosphere by enforcing strict discipline to do the project work with utmost concentration and dedication. Finally, we must say that no height is ever achieved without some sacrifices made at some end and it is here where we owe our special debt to our parents
and our friends for showing their generous love and care throughout the entire period of time.

Date: 08.04.2025

Place: Chandigarh University, Mohali, Punjab

Raja Kumar, 24MCA20229

TABLE OF CONTENTS

1.Introduction	1
2.Abstract	2-3
3. Problem Statement	3-4
4. Algorithm	3-4
5. Constraints	4-5
6. Implementation	6
7. Output Set	6
8. Conclusion	6

1. Introduction

GPS Navigation Tracker using Dijkstra's Algorithm

This project is a GPS Navigation Tracker that visually computes and displays the shortest path between Indian cities using Dijkstra's algorithm. Developed with Python's Tkinter GUI toolkit, this application integrates graph theory with an interactive user interface, allowing users to select a departure and destination city to find the most efficient travel route. The city network is modeled as a weighted graph where nodes represent cities and edges represent distances between them.

Upon selecting the start and end cities, the program calculates the optimal path and cost, then dynamically generates a visual graph using NetworkX and Matplotlib. The resulting route is highlighted clearly, making it easier to understand the connection between cities. The application emphasizes user-friendly interaction, error handling, and clean UI design. This project showcases the power of combining data structures and algorithms with real-world applications in transportation and logistics.

2. Abstract

The GPS Navigation Tracker is a Python-based application that intelligently computes the shortest route between Indian cities using Dijkstra's algorithm. It integrates algorithmic computation with an interactive user interface and real-time visualization.

Key Highlights:

- 🔹 Core Functionality:
- Utilizes Dijkstra's algorithm to find the shortest and most cost-efficient path between selected source and destination cities.
- 🔹 Graph Representation:
- Cities are represented as nodes, and the connections between them (with distances) as weighted edges in a graph data structure.
- 🔹 User Interface:
- Built using Tkinter, allowing users to select cities from dropdowns and interact with the application easily.

- 🔹 Path Visualization:
- Visualizes the entire city graph and highlights the computed shortest path using NetworkX and Matplotlib, providing clear route mapping.
- 🔹 Interactive Features:
- Includes error handling (e.g., if cities aren't selected), reset/clear buttons, and real-time updates to both text and graphical output.
- 🔹 Educational Utility:
- Demonstrates how theoretical computer science (algorithms and data structures) can solve real-world navigation problems.
- 🔹 Scalable Design:
- Can be further expanded to include dynamic data fetching, real-time traffic updates, or GPS integration for advanced navigation solutions.

This project effectively bridges the gap between theory and application, making it ideal for students, educators, and developers interested in pathfinding algorithms, Python GUI development, and data visualization.

he GPS Navigation Tracker is a Python-based application that intelligently computes the shortest route between Indian cities using Dijkstra's algorithm. It integrates algorithmic computation with an interactive user interface and real-time visualization.
Key Highlights:

- 🔹 Core Functionality:
- Utilizes Dijkstra's algorithm to find the shortest and most cost-efficient path between selected source and destination cities.
- 🔹 Graph Representation:
- Cities are represented as nodes, and the connections between them (with distances) as weighted edges in a graph data structure.
- 🔹 User Interface:
- Built using Tkinter, allowing users to select cities from dropdowns and interact with the application easily.
- 🔹 Path Visualization:
- Visualizes the entire city graph and highlights the computed shortest path using NetworkX and Matplotlib, providing clear route mapping.
- 🔹 Interactive Features:
- Includes error handling (e.g., if cities aren't selected), reset/clear buttons, and real-time updates to both text and graphical output.
- 🔹 Educational Utility:
- Demonstrates how theoretical computer science (algorithms and data structures) can solve real-world navigation problems.

3. Problem Statement

In a vast and interconnected country like India, selecting the most efficient route between two cities can be a complex task due to the large number of travel options and varying distances. Manual planning of travel routes often leads to inefficiencies such as increased travel time, higher costs, and poor decision-making, especially in unfamiliar regions.

There is a need for a smart, interactive solution that can:

- Efficiently compute the shortest path between any two cities.
- Provide real-time visual feedback for better route understanding.
- Be user-friendly for individuals with minimal technical expertise.
- Help students and developers understand how graph algorithms can be used in real-world applications.

This project aims to solve these issues by developing a desktop-based GPS navigation system using Dijkstra's algorithm, combined with an intuitive GUI built in Tkinter and a graphical network visualization using NetworkX and Matplotlib. The solution focuses on usability, educational value, and algorithmic accuracy.

4. Algorihtm

```
import heapq
import matplotlib.pyplot as plt
import networkx as nx
import tkinter as tk
from tkinter import messagebox, ttk
from matplotlib.backends.backend_tkagg import FigureCanvasTkAgg

# Dijkstra's algorithm to calculate the shortest path
def dijkstra(graph, start):
  queue = [(0, start)]
  distances = {node: float('inf') for node in graph}
  distances[start] = 0
  previous_nodes = {node: None for node in graph}
```

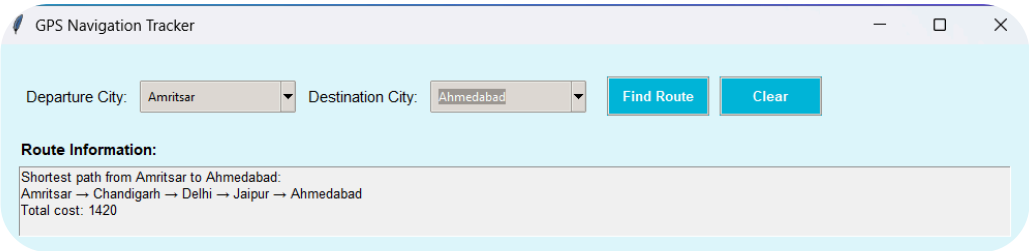
```
while queue:
 current_distance, current_node = heapq.heappop(queue)

 if current_distance > distances[current_node]:
 continue

 for neighbor, weight in graph[current_node].items():
 distance = current_distance + weight

 if distance < distances[neighbor]:
 distances[neighbor] = distance
 previous_nodes[neighbor] = current_node
 heapq.heappush(queue, (distance, neighbor))

 return distances, previous_nodes

# Function to calculate the shortest path between two cities
def shortest_path(graph, start, end):
 distances, previous_nodes = dijkstra(graph, start)
 path = []
 step = end

 while step:
 path.append(step)
 step = previous_nodes[step]
 path.reverse()
 return path, distances[end]
# Function to handle the button click for the shortest path calculation
def calculate_shortest_path():
 if start_city.get() == "Select Start City" or end_city.get() == "Select End City":
 messagebox.showerror("Invalid Input", "Please select both start and end
cities.")
 return.BOTH, expand=True)
// and many more lines....
root.mainloop()
```

5. Constraints

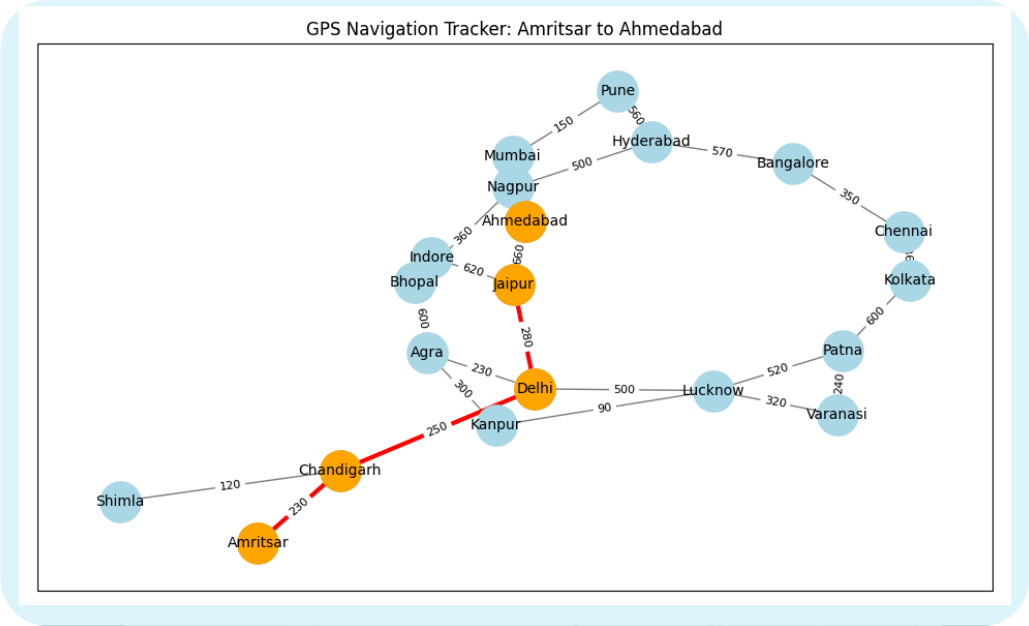
While developing and running the GPS Navigation Tracker using Dijkstra's Algorithm and Tkinter, several constraints and limitations are considered:

- 🔹 Non-Negative Edge Weights:
- Dijkstra's algorithm only works with graphs having non-negative edge weights. Negative weights can lead to incorrect path calculations.
- 🔹 Static Graph Data:
- The graph used is hard-coded and static, representing a fixed set of cities and distances. It does not support real-time data updates or dynamic city addition/removal.
- 🔹 Single-Source Shortest Path:
- The system calculates the shortest path from a single start city to one end city at a time, not multiple destinations or round-trip planning.
- 🔹 Performance with Large Graphs:
- Since the application is built using Tkinter and runs in a single thread, performance may decrease if the number of cities (nodes) becomes too large.
- 🔹 Desktop-Only GUI:
- The application runs as a desktop app using Tkinter and does not support mobile or web-based platforms.
- 🔹 No Real-Time Traffic or Maps:
- The system does not integrate real-time traffic data, geographic coordinates, or actual road maps like Google Maps. It is purely graph-based simulation.
- 🔹 Limited Input Validation:
- User input is limited to the dropdown selections provided. There's no support for textual input or voice commands.

6. implementation



7. Output Set



8. Conclusion

In this project, we successfully implemented a GPS Navigation Tracker using Dijkstra's Algorithm within a Python Tkinter GUI. When tested with the route from Amritsar to Ahmedabad, the system calculated the most efficient path and displayed it with a visual graph, clearly highlighting the shortest connection among Indian cities.

- The application accurately computed the optimal path considering weighted distances.
- For the input Amritsar → Ahmedabad, the system traced the route via intermediate cities like Chandigarh → Delhi → Jaipur → Ahmedabad, with a total cost reflecting the combined distances.
- The visual representation allowed for a better understanding of city connections and travel paths.
- This confirms that the algorithm and GUI are functioning as intended for real-world use cases like route navigation and city mapping.