# Optimal Delivery Route System Using TSP Algorithms

Group-6

Sai Satya Jagannadh Doddipatla / Akshay Patil / Sumanth Mittapally / Neema Tabarani / Adam Harb / Anuj Patel

## Roles & Responsibilities

- Akshay Patil : Project Coordinator
- Anuj Patel & Sumanth Mittapally : UI Design
- Sai Satya Jagannadh Doddipatla : Backend Developer
- Neema Tabarani & Adam Harb : Presentation & Testing

## Introduction

- Optimizing delivery routes is crucial for businesses to minimize costs, enhance efficiency, and improve customer satisfaction.
- In this project, we leverage the Traveling Salesperson Problem (TSP) algorithm to develop an optimal route delivery system. By integrating some tools, we aim to create a streamlined solution for finding the shortest distance between two locations, thereby facilitating efficient route planning in logistics and transportation.

## Algorithms

- **Nearest Neighbor Algorithm:** Offers a simple and greedy approach to find an approximate solution by starting from a random location and visiting the nearest unvisited location until all locations are covered.

- **Brute Force Algorithm:** Provides an exhaustive approach by generating and evaluating all possible permutations of locations to find the optimal solution, guaranteeing the shortest path but with exponential time complexity.

- **Random Sampling Algorithm**: Implements a balance between computational complexity and solution quality by generating a fixed number of random permutations of locations and selecting the best permutation as the solution.

- **Genetic Algorithm:** Inspired by natural selection and evolution, it iteratively improves a population of candidate solutions through selection, crossover, and mutation operations, effectively exploring the search space for near-optimal solutions.

- **Simulated Annealing Algorithm:** A probabilistic algorithm that explores the search space by occasionally accepting wrong solutions to escape local optima, capable of finding good solutions to complex optimization problems with appropriate cooling schedules.

## Tool development

• **Our methodology focused on four main areas:** Location input handling, algorithm implementation, result visualization, and user interface.

• **Technologies Used:**

- Python: Core programming language
- Streamlit Library: Frontend development and deployment
- Folium: Visualizing geospatial data on interactive maps
- Geopy: Calculating geodesic distances
- NumPy, Pandas, SciPy: Numerical operations and data manipulation
- Matplotlib: Creating visualizations (bar charts, line graphs)
- Random, Math, Itertools: Implementing algorithms

# Tool development

• **Data Processing and Algorithm Integration**

- Location Input Handling: Input validation for latitude/longitude values and duplicate coordinates.

- Algorithm Implementation: Implementing various TSP algorithms (Nearest Neighbor, Brute Force, Random Sampling, Genetic Algorithm, Simulated Annealing) with efficient data structures and optimizations.

- Integration and Testing: Defining the code to accept user input for locations and algorithm selection, and determining relevant test cases for each algorithm and edge cases.

• **User Interface Design and Interaction**

- Streamlit Application Development: Building the interactive web application using Streamlit, including location input forms, algorithm selection checkboxes, and result visualization components.

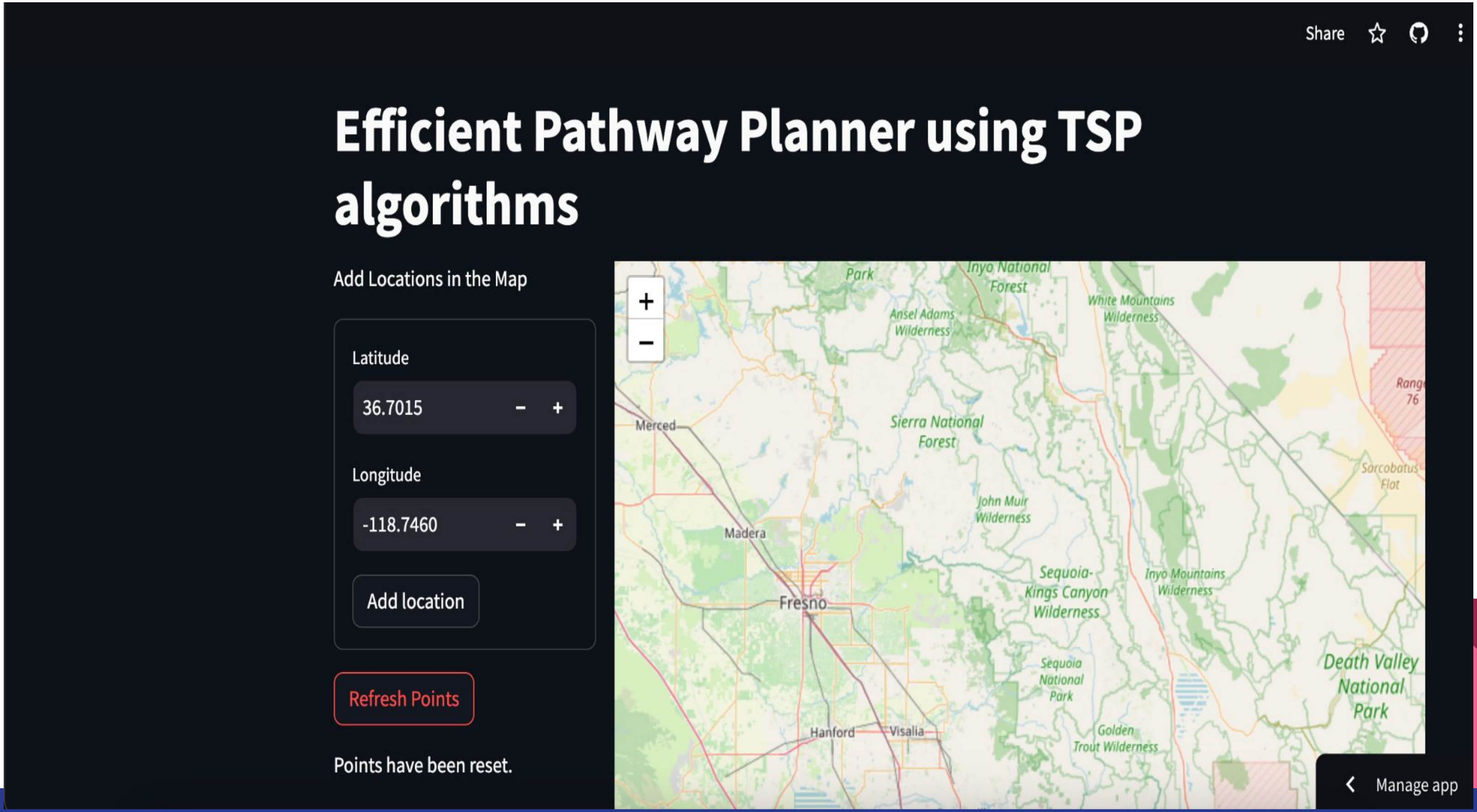
- Interactive Results Display: Ensuring clear and intuitive display of results, including the optimal route on the map, comparative analysis table, distance matrix, execution time bar chart, and distance comparison line graph.

- User Experience Enhancements: Implementing features like marker placement on the map, path visualization for algorithms, and the ability to refresh or reset added locations.

• **Deployment**

- Cloud Deployment: Deploying the Streamlit application to Streamlit Cloud for easy access and sharing.

## User Interface



Share

# Efficient Pathway Planner using TSP algorithms

Add Locations in the Map

Latitude

36.7015

Longitude

-118.7460

Add location

Refresh Points

Points have been reset.

## Demo

https://drive.google.com/file/d/1iXNHBj_R4E_xlLO_5ovqgC9Hhitkxi1F/view?usp=drive_link

## Test-cases

| Test Case | Objective | Expected Result | Actual Result | Status |
|---|---|---|---|---|
| Add valid latitude/longitude values | Test input validation | Coordinates should be accepted and marker added to map | Coordinates should be accepted and marker added to map | Pass |
| Add unique location coordinates | Test input validation | New unique coordinates should be accepted and marker added to map | New unique coordinates should be accepted and marker added to map | Pass |
| Select combination of algorithms | Test algorithm selection | Results for selected combination of algorithms should be displayed | Results for selected combination of algorithms should be displayed | Pass |
| Run all algorithms with small number of locations (2-5) | Test all algorithms | Correct shortest path should be found for each algorithm | Correct shortest path should be found for each algorithm | Pass |
| Run all algorithms with larger number of locations (10-20) | Test all algorithms | Correct shortest path should be found for each algorithm | Correct shortest path should be found for each algorithm | Pass |

| **Test Case** | **Objective** | **Expected Result** | **Actual Result** | **Status** |
|---|---|---|---|---|
| Marker placement on the map | Test visualization | Markers should be placed correctly on the map | Markers should be placed correctly on the map | Pass |
| Path visualization for algorithms | Test visualization | Path should be visualized correctly on the map | Path should be visualized correctly on the map | Pass |
| Bar chart visualization | Test visualization | Bar chart should be displayed correctly | Bar chart should be displayed correctly | Pass |
| Line graph visualization | Test visualization | Line graph should be displayed correctly | Line graph should be displayed correctly | Pass |

## Challenges & Solutions

**Challenges:**

1. **Brute Force Algorithm Efficiency**

2. **Route Breakdown Visualization**

**3.Integrating the Frontend and the Backend**

**4.Launching the Streamlit Application Locally**

**Solutions:**

1. **Optimization of Brute Force Algorithm:** The brute force algorithm becomes highly inefficient as the number of locations increases due to the factorial growth in the number of permutations. Implement alternative algorithms to handle larger datasets more efficiently.

2. **Route Breakdown Visualization:** Enhance the application to display a detailed breakdown of the optimized route for each algorithm. This breakdown should include distances between consecutive points, allowing users to understand the solution better.

3.Used the Steamlit library to create frontend as it was more intuitive and flexible to integrate.

4.We also used Streamlit Cloud to deploy the app online so that all team members can access the application without running into issues.

## Scan the Barcode for the quiz

Thank You!