PathSync:
Intelligent Supply Chain Route Optimization using Geospatial Analysis, ML, Graph Neural Network

Name: Vidhu Prabha

ID: 2022DA4087

A Dissertation Submitted in Partial Fulfillment of the Requirements for the Degree of Master of Technology.

Birla Institute of Technology and Science

Data Science and Engineering

[September 2024]

The Reasons to support it is a Data Science project, there are several key aspects that highlight its nature:

1. Data Collection and Storage

- **Snowflake Integration:** The project connects to a Snowflake database to fetch facility and customer address details, demonstrating the ability to interact with large datasets and databases.

- **Google Maps API Usage:** It uses the Google Maps API to retrieve geographical data, such as distances and travel times, which is crucial for route optimization.

2. Data Processing and Preparation

- **Data Retrieval Functions:** The functions get_facility_details and get_addresses_for_city retrieve and prepare data for further analysis. They involve querying a database and processing the results into usable formats (like lists of dictionaries).

- **Handling Geospatial Data:** The project manages latitude and longitude data, a critical aspect of spatial data science.

3. Machine Learning and Modeling

- **Graph Neural Network (GNN):** The implementation of a GNN for optimizing routes is a significant component. This shows the application of advanced machine learning techniques to solve a complex problem like the Traveling Salesman Problem (TSP).

- **Optimization Algorithms:** The use of Euclidean distance heuristics and K-means clustering indicates an understanding of optimization and clustering algorithms in machine learning.

4. Route Optimization

- **TSP Optimization Logic:** The tsp_optimized_route_gnn function encapsulates the logic for optimizing routes using a neural network. This is a critical aspect of data science, as it applies statistical methods to improve decision-making processes.

- **Distance and Duration Calculations:** Functions like get_distance_and_time demonstrate how the model calculates real-world metrics, essential for evaluating performance.

5. Data Visualization

- **Plotting with Matplotlib and Plotly:** The project includes visualization capabilities using libraries like Matplotlib and Plotly, which are essential for presenting results and insights. This helps in interpreting data and communicating findings effectively.

6. Analysis and Insights

- **Clustering Customer Addresses:** The project includes a function to cluster customer addresses using K-means clustering, highlighting the analysis of spatial relationships among data points.

- **Comparative Metrics:** It computes and compares optimized versus original distances and durations, providing insights into the effectiveness of the route optimization process.

7. Reproducibility and Control

- **Seed Initialization:** The project sets random seeds for reproducibility in experiments, a best practice in data science to ensure consistent results.

Conclusion

Overall, this project integrates various data science methodologies, including data collection, processing, machine learning modeling, and visualization, to address a specific problem in route optimization. The use of GNNs and clustering algorithms adds complexity and showcases a sophisticated understanding of data science principles.

Objective

The primary objective of **PathSync** is to develop a Graph Neural Network (GNN) model to optimize supply chain routing by effectively predicting the most efficient routes for deliveries. This model aims to enhance delivery speed and accuracy while minimizing costs and environmental impact through data-driven decision-making. Specifically, the project aims to achieve the following goals:

- **Develop a GNN-based Model**: Create a robust Graph Neural Network model to optimize routing in supply chain management, leveraging facility and customer location data.
- **Integrate Real-time Data**: Incorporate real-time traffic and weather data to enhance the accuracy and reliability of route predictions.
- **Develop interactive visual reports** using Python libraries such as Folium and Matplotlib to visualize route optimization results, including distance, time, and cost metrics.
- **Provide data-driven forecasts** of route efficiency over various time periods (daily, weekly, monthly, yearly) and estimate their potential impact on operations.
- **Deliver Actionable Insights**: Provide actionable insights and recommendations for supply chain managers to improve operational efficiency and reduce costs through optimized routing solutions.
- **Assess Scalability**: Test the model's scalability by applying it to various supply chain scenarios with differing complexities and sizes.
- **Enhance Prediction Accuracy**: Aim to improve the accuracy of delivery time predictions by utilizing the inherent relationships between facilities and customer locations in the GNN framework.
- **Optimize Resource Utilization**: Investigate methods to optimize resource allocation (e.g., vehicles and personnel) based on the optimized routes generated by the GNN model.
- **Develop a User-friendly Interface**: Create an intuitive interface for stakeholders to visualize route optimizations and assess their implications on supply chain operations.
- **Facilitate Decision-Making**: Support decision-making processes by providing comprehensive reports and visualizations that highlight the benefits of adopting GNN- based route optimization.

Project Steps and Modules

Structured list of steps involved in this project on supply chain route optimization, including the modules and functionalities, this structured approach ensures a comprehensive handling of each aspect of the project, from data acquisition to analysis and reporting.:

Project Initialization

- Set up the virtual environment.
- Install necessary libraries (e.g., Pandas, NumPy, Matplotlib, Geopy, Scikit-learn, TensorFlow).

Data Acquisition

- Load UPS facility and customer data into Snowflake database
- Retrieve customer address data from the Snowflake database.

Geocoding

- Use the Google Maps API to convert customer addresses into latitude and longitude coordinates.
- Store the geocoded coordinates in a DataFrame.

Data Preprocessing

- Cleab UPS facilty data before loading it into Snowflake databse.
- Clean and preprocess the customer address data.
- Validate the presence of necessary columns (e.g., Address, City, State, Latitude, Longitude).

Clustering Customer Addresses

- Implement KMeans clustering to group customer addresses based on proximity (e.g., clustering into groups of 5).
- Calculate centroids for each cluster.

Route Optimization Logic

For each cluster of customer addresses:

- Retrieve nearest UPS facility coordinates based on the cluster centroid.
- Implement a Traveling Salesman Problem (TSP) solution using a Graph Neural Network (GNN) to determine the optimal route.
- Calculate distances and durations for the optimized route.

Distance and Duration Estimation

- Estimate original and optimized distances and durations for each cluster.
- Store these estimations in a structured format.

Cost and Savings Calculation

- Calculate fuel costs and savings based on distances, fuel efficiency, and fuel prices.
- Implement functions to estimate these values for various time periods (daily, weekly, monthly, yearly).

Data Storage

- Store the results (original distances, optimized distances, durations, fuel costs, and savings) into a DataFrame for further analysis.

Visualization

- Create visualization for optimized route map.
- Create visualizations to compare original vs. optimized distances, durations, and fuel costs using Matplotlib.
- Generate summary statistics plots, including comparisons of distances, durations, fuel costs, and savings.

Results Analysis

- Analyze the results based on visualizations and summary statistics.
- Generate insights on optimization effectiveness.

Documentation and Reporting

- Document the code with comments and docstrings.
- Prepare a report summarizing findings, methodologies, and visualizations.

Deployment (Optional)

- Consider deploying the Flask application for user interactions.
- Ensure the application integrates with the Snowflake database and provides route optimization services.

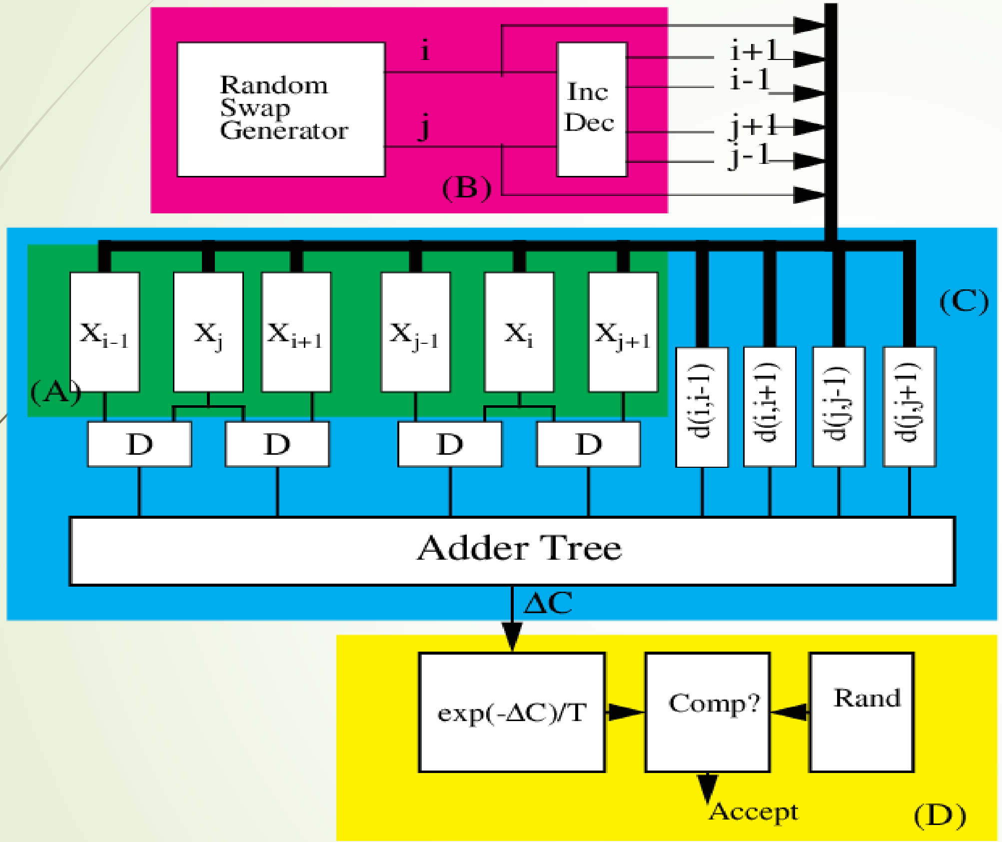
Modules Overview

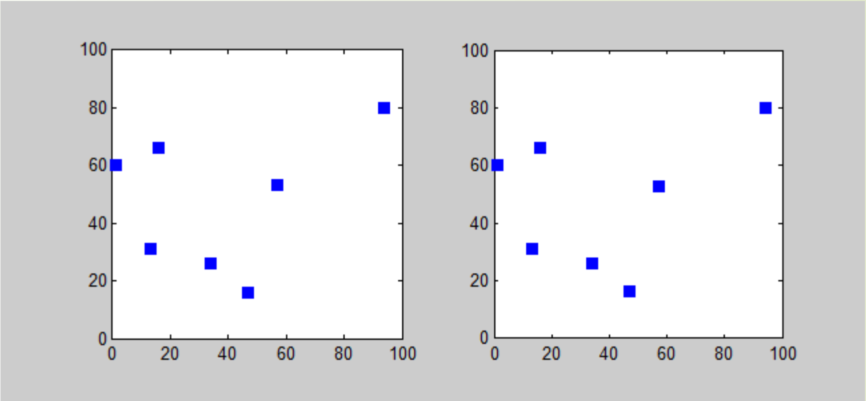
- **Data Loading Module**: Handles loading customer and UPS facility data from Snowflake.
- **Geocoding Module**: Manages the conversion of addresses to coordinates using the Google Maps API.
- **Clustering Module**: Implements KMeans clustering for customer addresses.
- **Route Optimization Module**: Includes logic for TSP solution using GNN and distance calculation.
- **Estimation Module**: Performs calculations for distances, durations, fuel costs, and savings.
- **Visualization Module**: Handles the generation of plots and visual comparisons of results.

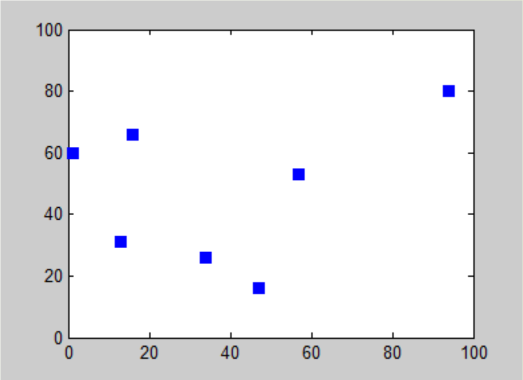
Architecture

- Layered Architecture Overview
- Presentation Layer (Frontend)
- Application Layer (Flask)
- Service Layer (Route Optimization and Data Processing)
- Integration Layer (APIs like Google Maps and OpenWeather)
- Data Layer (Snowflake Database)

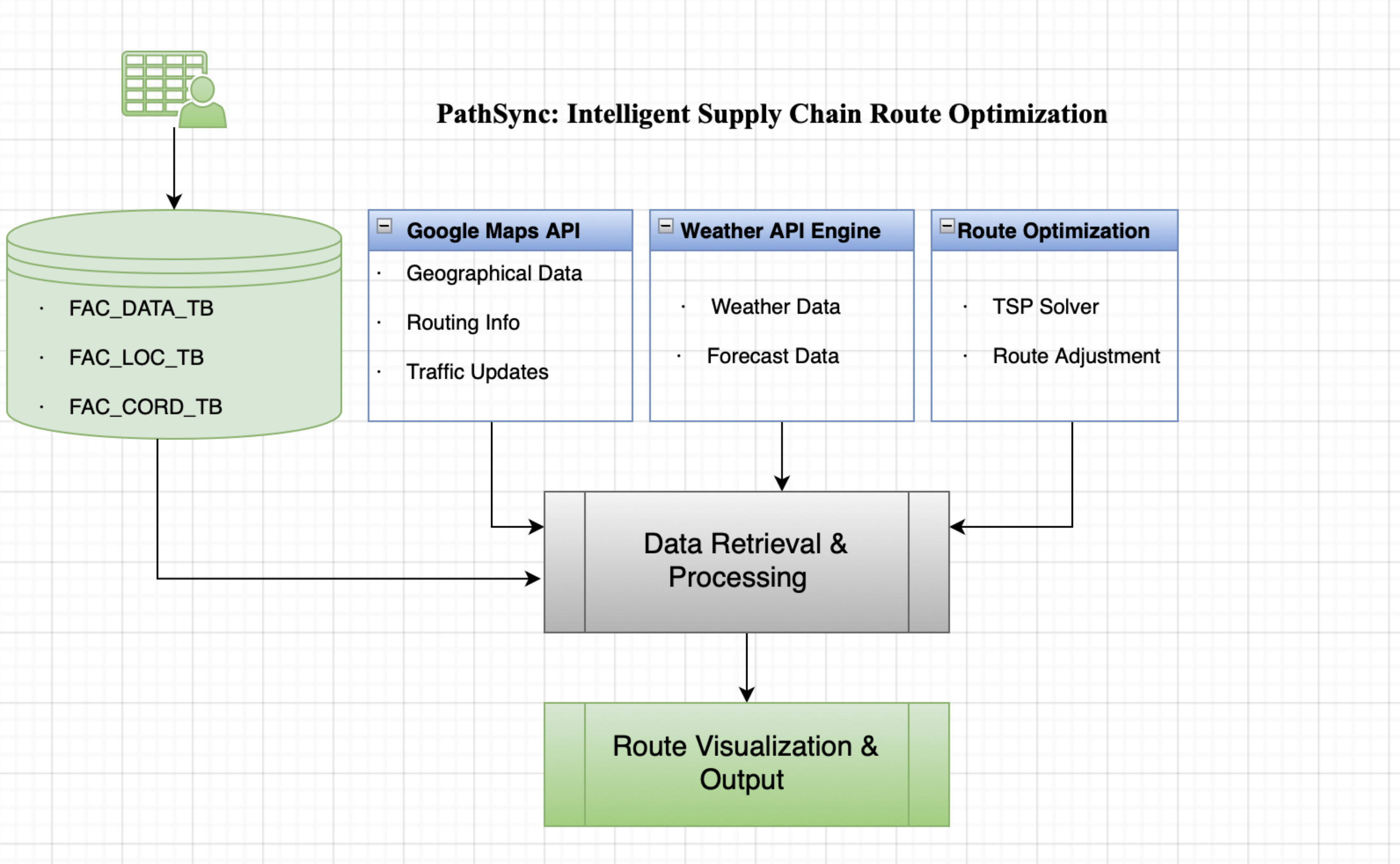
Architecture of solving TSP (Travelling Salesman Problem)



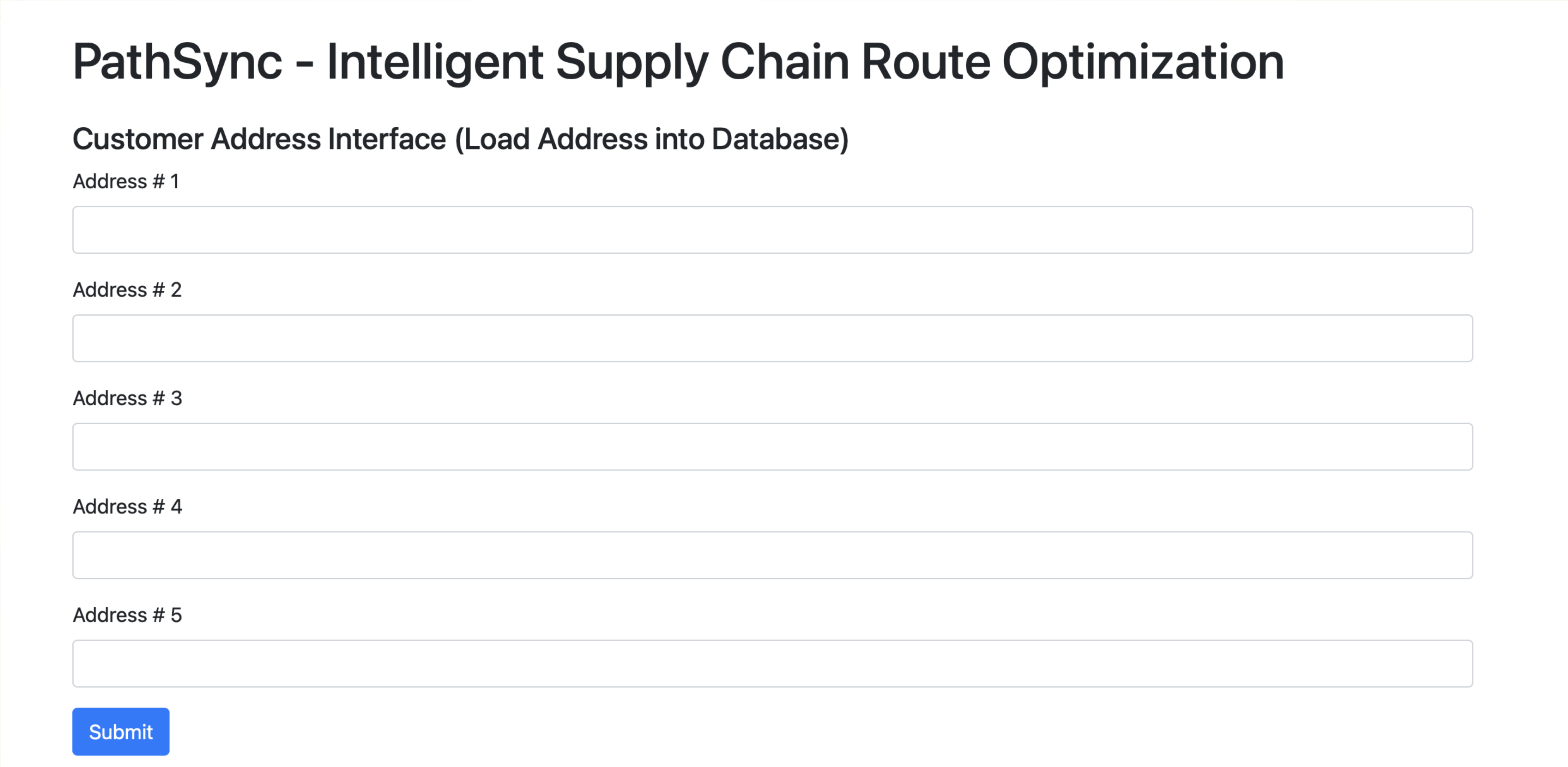




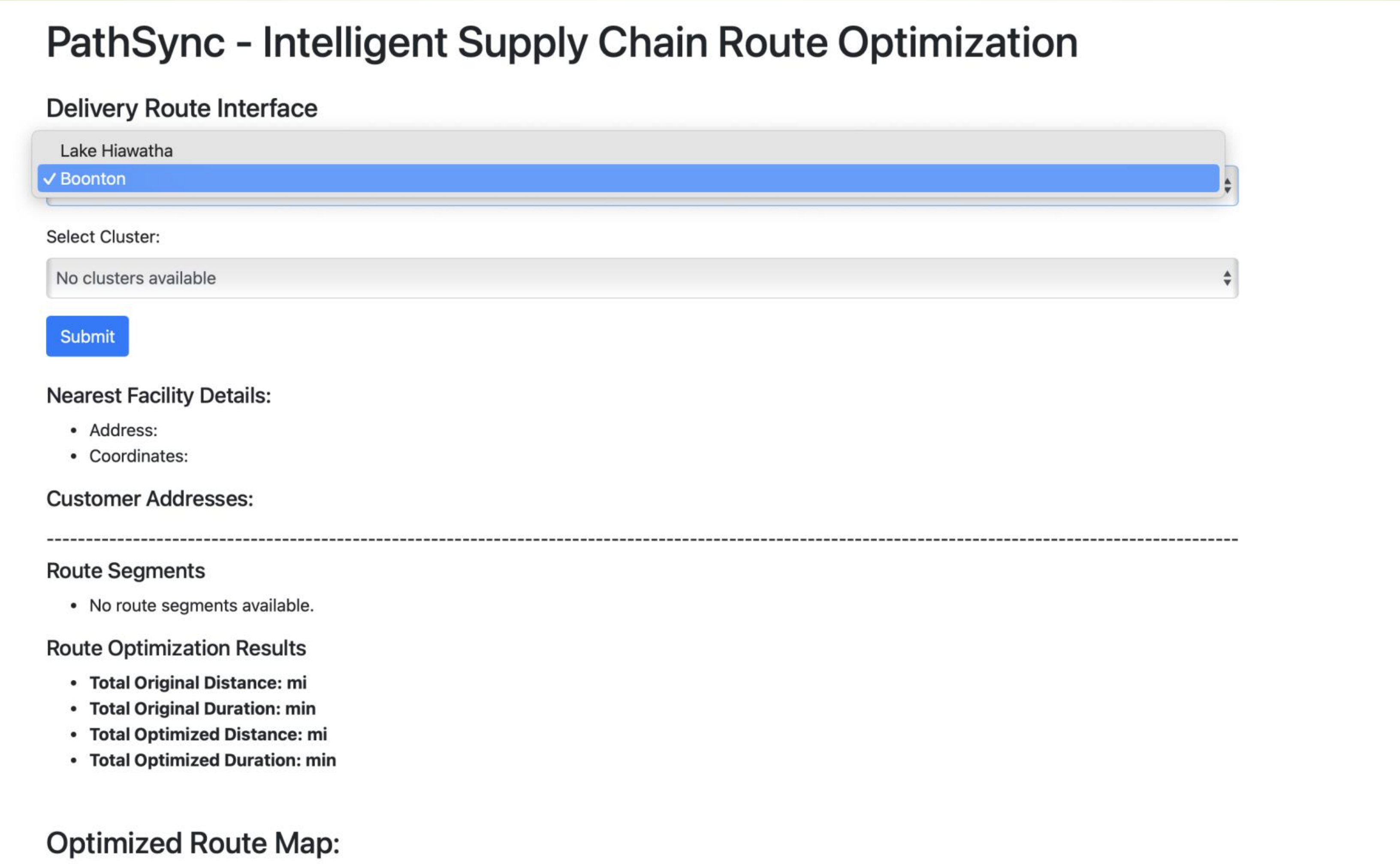
FLOW-DIAGRAM



Presentation Layer (Customer Interface)



Presentation Layer (User Interface)



Presentation Layer (User Interface)

PathSync - Intelligent Supply Chain Route Optimization

Delivery Route Interface

Select City:

No city available

Select Cluster:

No clusters available

Submit

Nearest Facility Details:

- Address: The UPS Store 144 N BEVERWYCK RD, Lake Hiawatha, NJ
- Coordinates: 40.885266 -74.381078

Lake Hiawatha Customer Addresses:

- Address 1: 456 Oak Avenue, Parsippany, NJ 07054, City: Parsippany-Troy Hills, State: NJ, Lat: 40.8648051, Lon: -74.3766393
- Address 2: 789 Maple Drive, Parsippany, NJ 07054, City: Parsippany-Troy Hills, State: NJ, Lat: 40.8595077, Lon: -74.3511793
- Address 3: 202 Elm Street, Parsippany, NJ 07054, City: Parsippany-Troy Hills, State: NJ, Lat: 40.8601751, Lon: -74.3520073
- Address 4: 345 Lincoln Ave, Parsippany, NJ 07054, City: Parsippany-Troy Hills, State: NJ, Lat: 40.8796256, Lon: -74.3855317
- Address 5: 115 Lake Shore Dr, Lake Hiawatha, NJ, City: Parsippany-Troy Hills, State: NJ, Lat: 40.8782538, Lon: -74.3773036
- Address 6: 26.5 Cherokee Ave, Lake Hiawatha, NJ, City: Parsippany-Troy Hills, State: NJ, Lat: 40.8779471, Lon: -74.3744137
- Address 7: 116 N Beverwyck Rd, Lake Hiawatha, NJ, City: Parsippany-Troy Hills, State: NJ, Lat: 40.8834309, Lon: -74.3818934
- Address 8: 101 Hiawatha Blvd, Lake Hiawatha, NJ 07034, City: Parsippany-Troy Hills, State: NJ, Lat: 40.8819937, Lon: -74.3778634

Presentation Layer (User Interface)

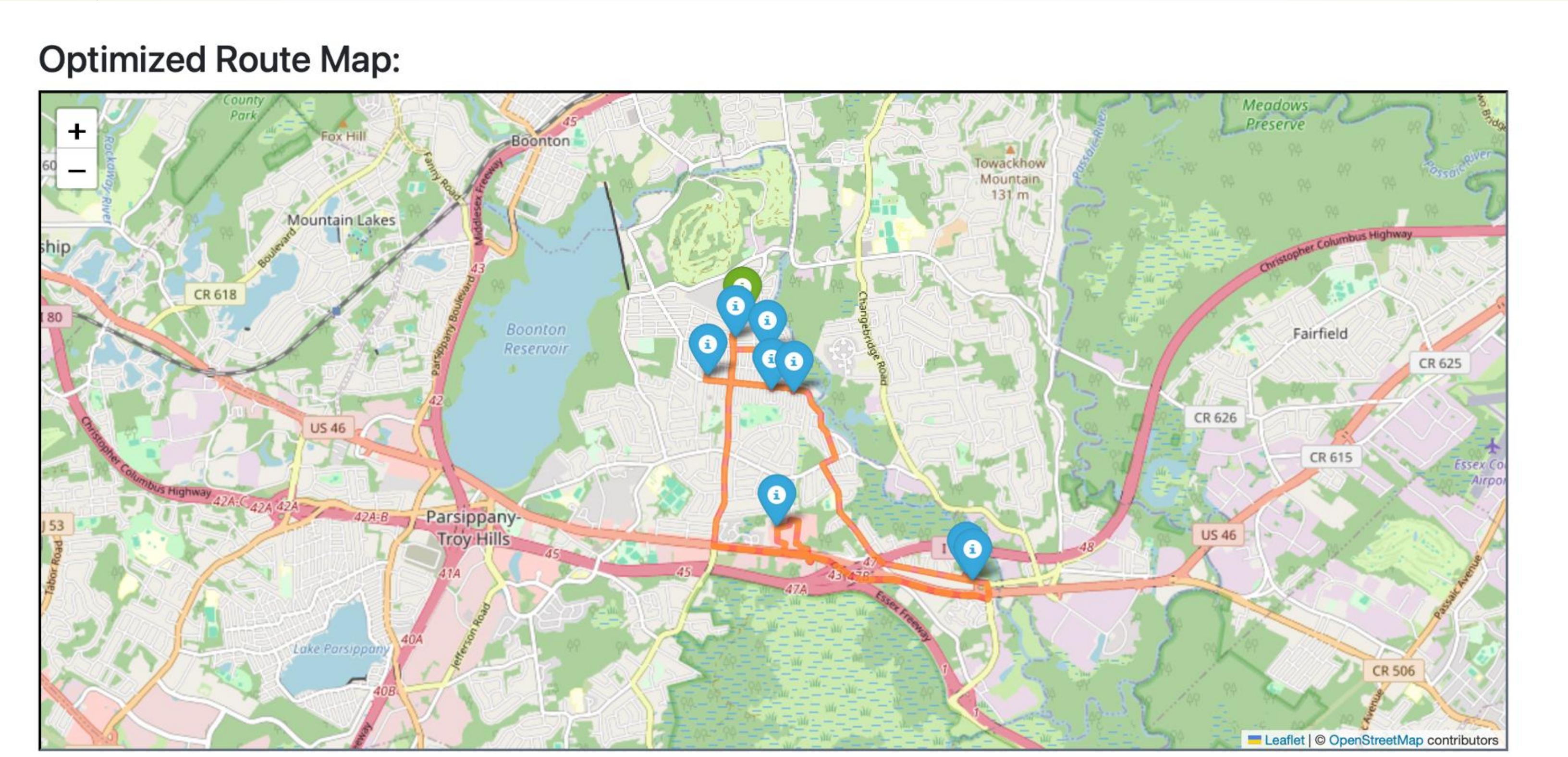
Lake Hiawatha Route Segments

- Destination 1: Address: 456 Oak Avenue, Parsippany, NJ 07054, City: Parsippany-Troy Hills, State: NJ, Coordinates: 40.8648051, -74.3766393
  Optimized Distance: 1.97 miles, Optimized Duration: 6.98 minutes
- Destination 2: Address: 789 Maple Drive, Parsippany, NJ 07054, City: Parsippany-Troy Hills, State: NJ, Coordinates: 40.8595077, -74.3511793
  Optimized Distance: 0.1 miles, Optimized Duration: 0.83 minutes
- Destination 3: Address: 202 Elm Street, Parsippany, NJ 07054, City: Parsippany-Troy Hills, State: NJ, Coordinates: 40.8601751, -74.3520073
  Optimized Distance: 2.74 miles, Optimized Duration: 7.33 minutes
- Destination 4: Address: 345 Lincoln Ave, Parsippany, NJ 07054, City: Parsippany-Troy Hills, State: NJ, Coordinates: 40.8796256, -74.3855317
  Optimized Distance: 0.5 miles, Optimized Duration: 1.95 minutes
- Destination 5: Address: 115 Lake Shore Dr, Lake Hiawatha, NJ, City: Parsippany-Troy Hills, State: NJ, Coordinates: 40.8782538, -74.3773036
  Optimized Distance: 0.17 miles, Optimized Duration: 0.72 minutes
- Destination 6: Address: 26.5 Cherokee Ave, Lake Hiawatha, NJ, City: Parsippany-Troy Hills, State: NJ, Coordinates: 40.8779471, -74.3744137
  Optimized Distance: 0.77 miles, Optimized Duration: 2.5 minutes
- Destination 7: Address: 116 N Beverwyck Rd, Lake Hiawatha, NJ, City: Parsippany-Troy Hills, State: NJ, Coordinates: 40.8834309, -74.3818934
  Optimized Distance: 0.32 miles, Optimized Duration: 1.5 minutes
- Destination 8: Address: 101 Hiawatha Blvd, Lake Hiawatha, NJ 07034, City: Parsippany-Troy Hills, State: NJ, Coordinates: 40.8819937, -74.3778634
  Optimized Distance: 0.46 miles, Optimized Duration: 1.58 minutes

Route Optimization Results

- **Total Original Distance: 8.61 mi**
- **Total Original Duration: 26.82 min**
- **Total Optimized Distance: 7.03 mi**
- **Total Optimized Duration: 23.4 min**

Presentation Layer (**User Interface**)



Presentation Layer (User Interface)

PathSync - Intelligent Supply Chain Route Optimization

Delivery Route Interface

Select City:

No city available

Select Cluster:

No clusters available

Submit

Nearest Facility Details:

- Address: The UPS Store 121 HAWKINS PL, Boonton, NJ
- Coordinates: 40.906949 -74.424813

Boonton Customer Addresses:

- Address 1: 112 Sycamore St, Boonton, NJ 07005, City: Boonton, State: NJ, Lat: 40.9078612, Lon: -74.4054601
- Address 2: 567 Cedar Ln, Boonton, NJ 07005, City: Boonton, State: NJ, Lat: 40.9074497, Lon: -74.4090519
- Address 3: 234 Pine St, Boonton, NJ 07005, City: Boonton, State: NJ, Lat: 40.9090073, Lon: -74.4148275
- Address 4: 890 Birch Way, Boonton, NJ 07005, City: Boonton, State: NJ, Lat: 40.9076395, Lon: -74.4136805
- Address 5: 678 Elm St, Boonton, NJ 07005, City: Boonton, State: NJ, Lat: 40.904019, Lon: -74.4087405
- Address 6: 131 Main St, Boonton, NJ 07005, City: Boonton, State: NJ, Lat: 40.9093314, Lon: -74.4182865
- Address 7: 123 Main Street, Boonton, NJ 07005, City: Boonton, State: NJ, Lat: 40.9089067, Lon: -74.4182729
- Address 8: 789 Washington St, Parsippany, NJ 07054, City: Boonton, State: NJ, Lat: 40.8954376, Lon: -74.4103717
- Address 9: 123 Parsippany Blvd, Parsippany, NJ 07054, City: Parsippany-Troy Hills, State: NJ, Lat: 40.8925653, Lon: -74.4159037

Presentation Layer (User Interface)

Boonton Route Segments

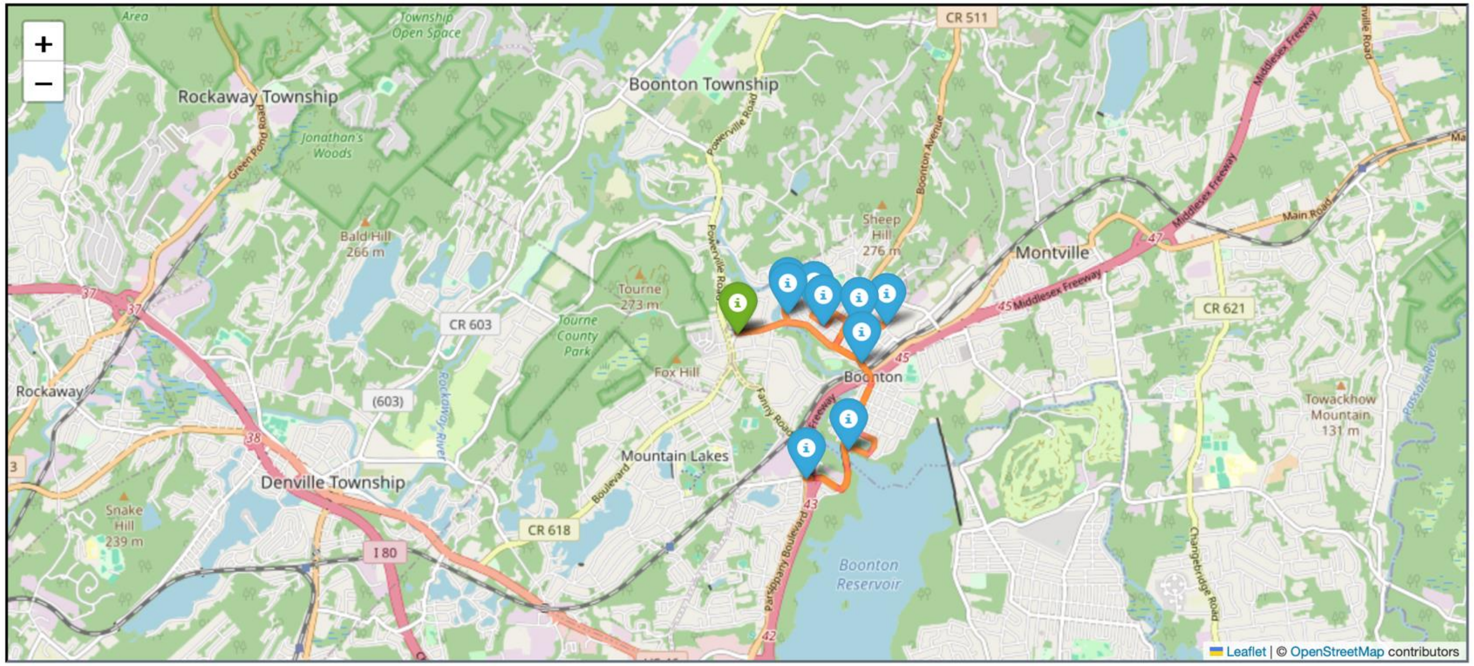
- Destination 1: Address: 112 Sycamore St, Boonton, NJ 07005, City: Boonton, State: NJ, Coordinates: 40.9078612, -74.4054601 Optimized Distance: 0.26 miles, Optimized Duration: 1.4 minutes
- Destination 2: Address: 567 Cedar Ln, Boonton, NJ 07005, City: Boonton, State: NJ, Coordinates: 40.9074497, -74.4090519 Optimized Distance: 0.39 miles, Optimized Duration: 1.72 minutes
- Destination 3: Address: 234 Pine St, Boonton, NJ 07005, City: Boonton, State: NJ, Coordinates: 40.9090073, -74.4148275 Optimized Distance: 0.12 miles, Optimized Duration: 0.88 minutes
- Destination 4: Address: 890 Birch Way, Boonton, NJ 07005, City: Boonton, State: NJ, Coordinates: 40.9076395, -74.4136805 Optimized Distance: 0.64 miles, Optimized Duration: 2.75 minutes
- Destination 5: Address: 678 Elm St, Boonton, NJ 07005, City: Boonton, State: NJ, Coordinates: 40.904019, -74.4087405 Optimized Distance: 0.68 miles, Optimized Duration: 2.13 minutes
- Destination 6: Address: 131 Main St, Boonton, NJ 07005, City: Boonton, State: NJ, Coordinates: 40.9093314, -74.4182865 Optimized Distance: 0.03 miles, Optimized Duration: 0.07 minutes
- Destination 7: Address: 123 Main Street, Boonton, NJ 07005, City: Boonton, State: NJ, Coordinates: 40.9089067, -74.4182729 Optimized Distance: 0.43 miles, Optimized Duration: 1.07 minutes
- Destination 8: Address: 789 Washington St, Parsippany, NJ 07054, City: Boonton, State: NJ, Coordinates: 40.906949, -74.424813 Optimized Distance: 1.6 miles, Optimized Duration: 4.8 minutes
- Destination 9: Address: 123 Parsippany Blvd, Parsippany, NJ 07054, City: Parsippany-Troy Hills, State: NJ, Coordinates: 40.8954376, -74.4103717 Optimized Distance: 1.21 miles, Optimized Duration: 3.68 minutes

Presentation Layer (User Interface)

Route Optimization Results

- **Total Original Distance: 5.68 mi**
- **Total Original Duration: 19.1 min**
- **Total Optimized Distance: 5.35 mi**
- **Total Optimized Duration: 18.5 min**

Optimized Route Map:



Presentation Layer (User Interface)

[8]:

```python
print(tabulate(summary_stats_rounded.transpose(), headers='keys', tablefmt='pretty'))
```

	count	mean	std	min	25%	50%	75%	max
daily_original_distance	2.0	28.58	8.29	22.72	25.65	28.58	31.51	34.44
daily_optimized_distance	2.0	24.76	4.75	21.4	23.08	24.76	26.44	28.12
daily_original_duration	2.0	91.84	21.84	76.4	84.12	91.84	99.56	107.28
daily_optimized_duration	2.0	83.8	13.86	74.0	78.9	83.8	88.7	93.6

[9]:

```python
print(tabulate(daily_stats, headers='keys', tablefmt='pretty'))
```

	count	mean	std	min	25%	50%	75%	max
daily_original_distance	2.0	28.58	8.29	22.72	25.65	28.58	31.51	34.44
daily_optimized_distance	2.0	24.76	4.75	21.4	23.08	24.76	26.44	28.12
daily_original_fuel_cost	2.0	4.0	1.16	3.18	3.59	4.0	4.41	4.82
daily_optimized_fuel_cost	2.0	3.47	0.67	3.0	3.23	3.47	3.7	3.94
daily_fuel_savings	2.0	0.53	0.49	0.18	0.36	0.53	0.71	0.88

[10]:

```python
print(tabulate(weekly_stats, headers='keys', tablefmt='pretty'))
```

	count	mean	std	min	25%	50%	75%	max
weekly_original_distance	2.0	200.06	58.01	159.04	179.55	200.06	220.57	241.08
weekly_optimized_distance	2.0	173.32	33.26	149.8	161.56	173.32	185.08	196.84
weekly_original_fuel_cost	2.0	28.01	8.12	22.27	25.14	28.01	30.88	33.75
weekly_optimized_fuel_cost	2.0	24.26	4.66	20.97	22.62	24.26	25.91	27.56
weekly_fuel_savings	2.0	3.74	3.46	1.29	2.52	3.74	4.97	6.19

[11]:

```python
print(tabulate(monthly_stats, headers='keys', tablefmt='pretty'))
```

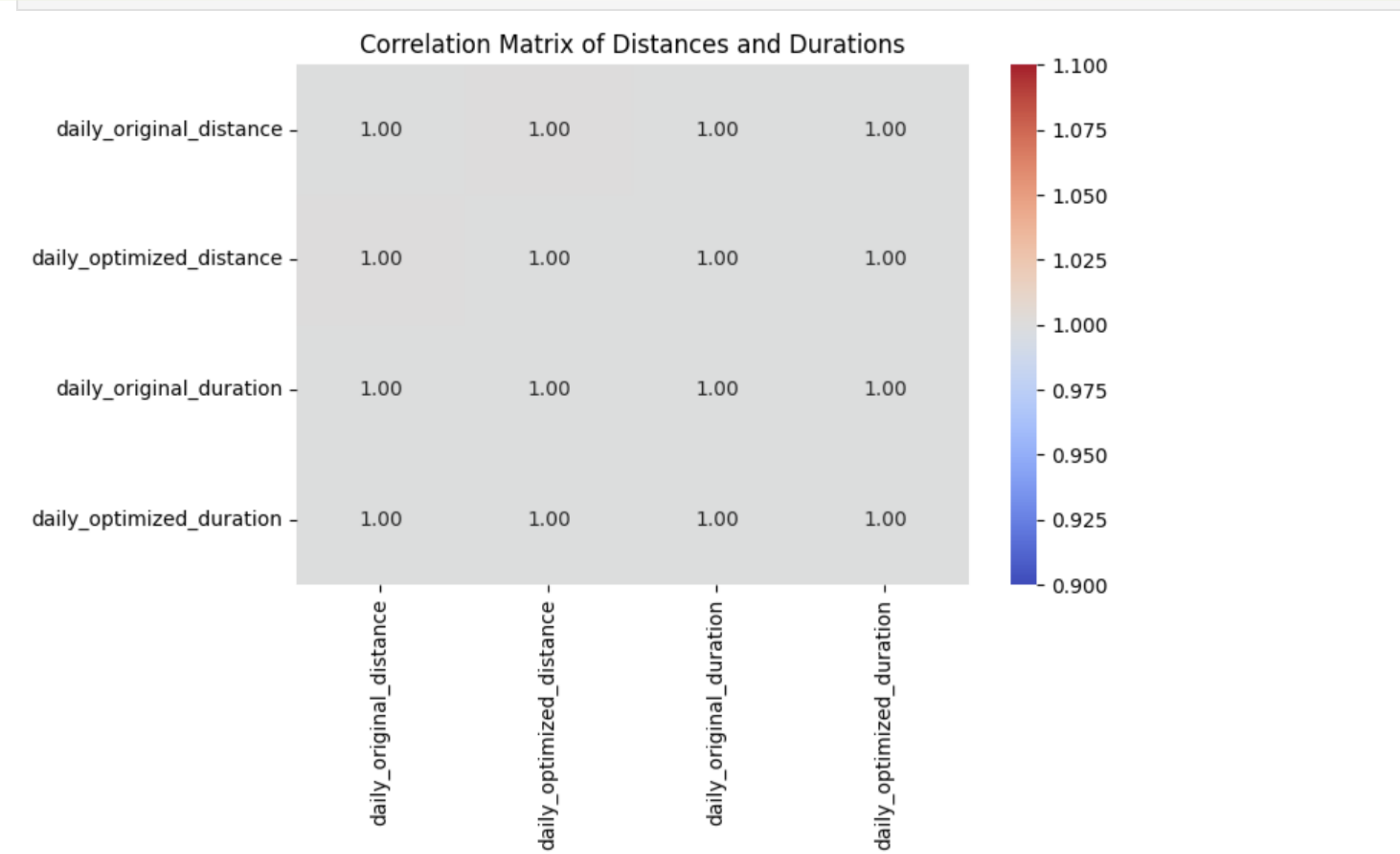
	count	mean	std	min	25%	50%	75%	max
monthly_original_distance	2.0	857.4	248.62	681.6	769.5	857.4	945.3	1033.2
monthly_optimized_distance	2.0	742.8	142.55	642.0	692.4	742.8	793.2	843.6
monthly_original_fuel_cost	2.0	120.04	34.81	95.42	107.73	120.04	132.34	144.65
monthly_optimized_fuel_cost	2.0	103.99	19.96	89.88	96.94	103.99	111.05	118.1
monthly_fuel_savings	2.0	16.04	14.85	5.54	10.79	16.04	21.29	26.54

[12]:

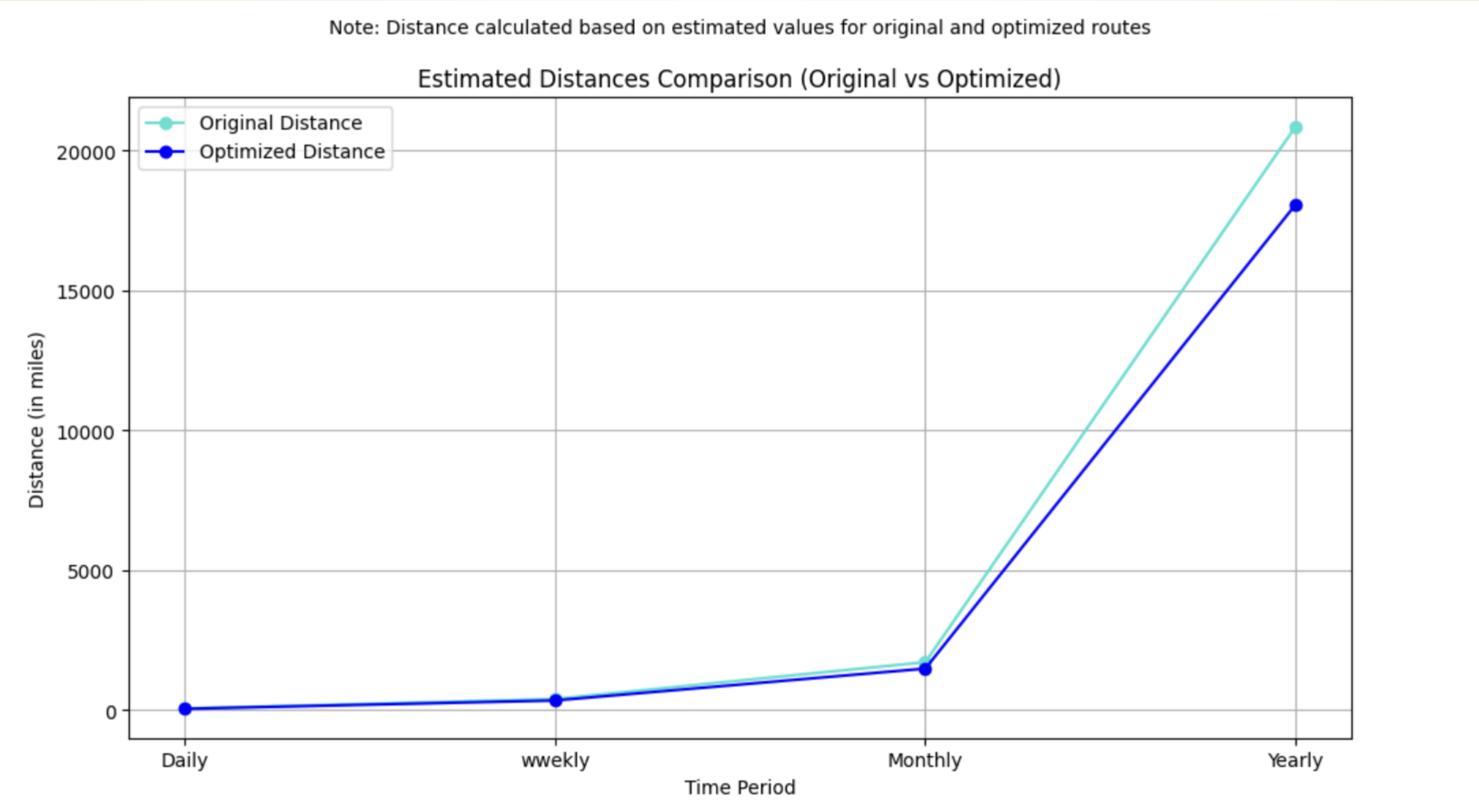
```python
print(tabulate(yearly_stats, headers='keys', tablefmt='pretty'))
```

	count	mean	std	min	25%	50%	75%	max
yearly_original_distance	2.0	10431.7	3024.86	8292.8	9362.25	10431.7	11501.15	12570.6
yearly_optimized_distance	2.0	9037.4	1734.39	7811.0	8424.2	9037.4	9650.6	10263.8
yearly_original_fuel_cost	2.0	1460.44	423.48	1160.99	1310.71	1460.44	1610.16	1759.88
yearly_optimized_fuel_cost	2.0	1265.24	242.81	1093.54	1179.39	1265.24	1351.08	1436.93
yearly_fuel_savings	2.0	195.2	180.67	67.45	131.33	195.2	259.08	322.95

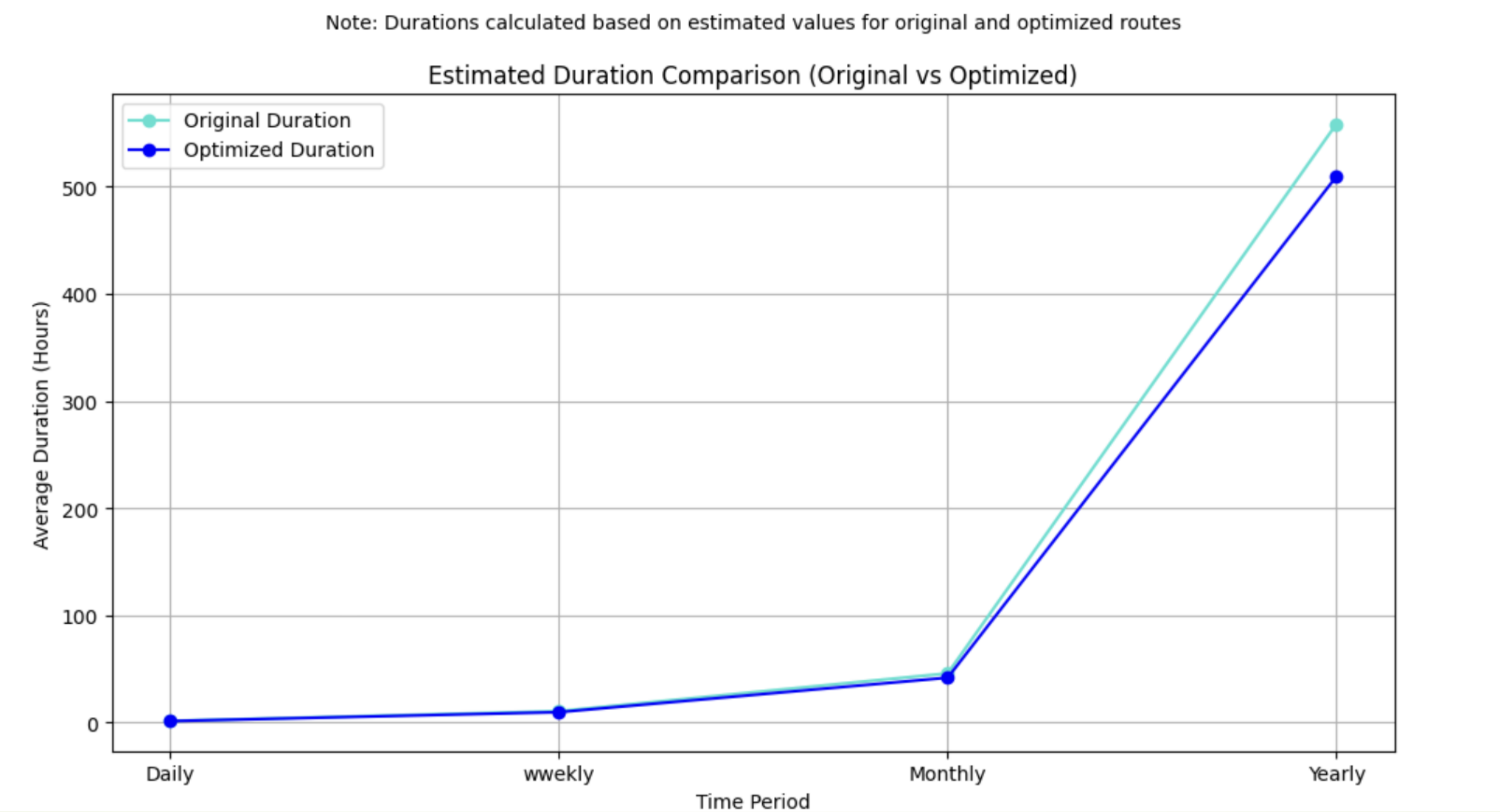
Presentation Layer (User Interface)



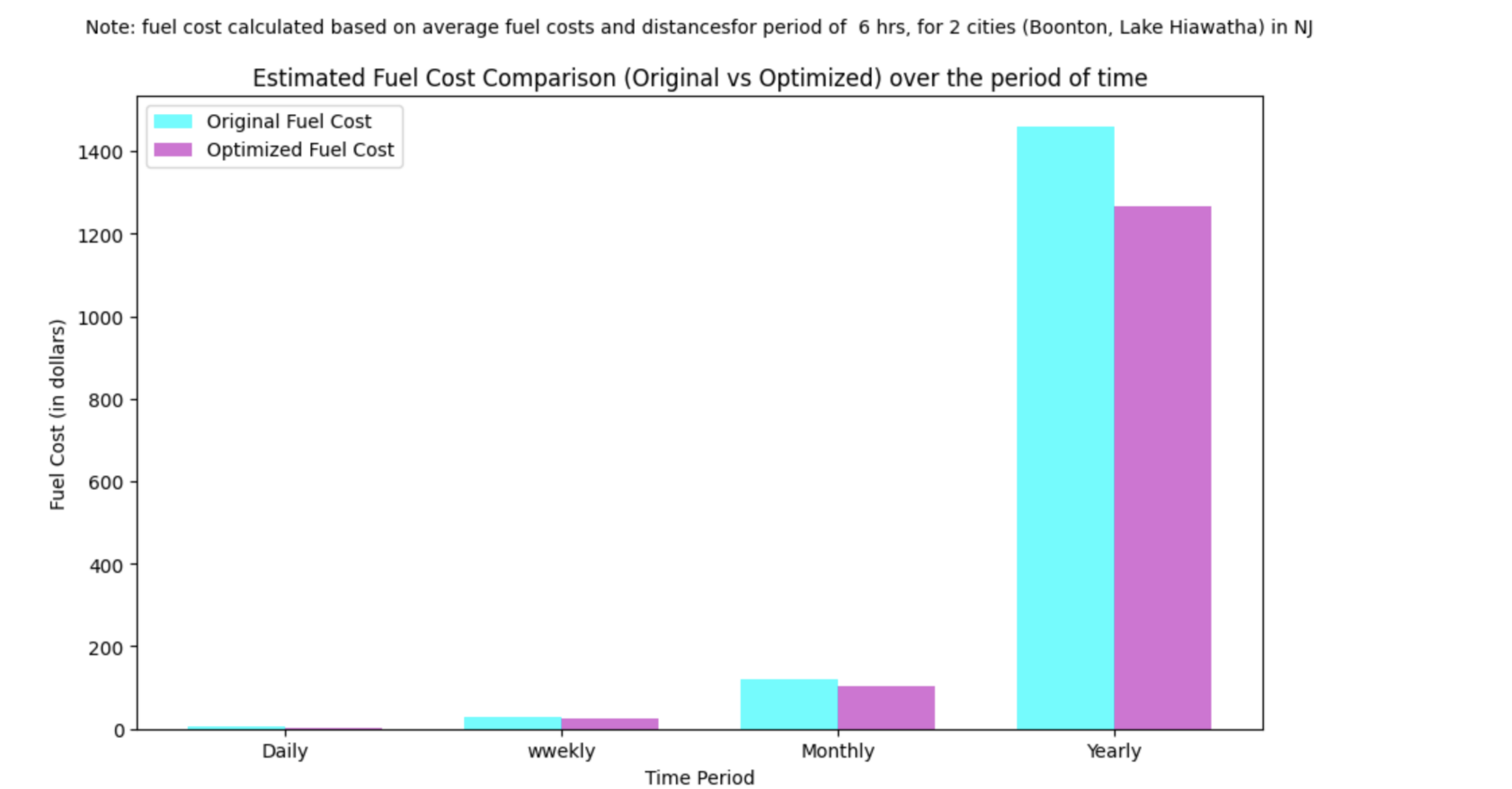
Presentation Layer (User Interface)



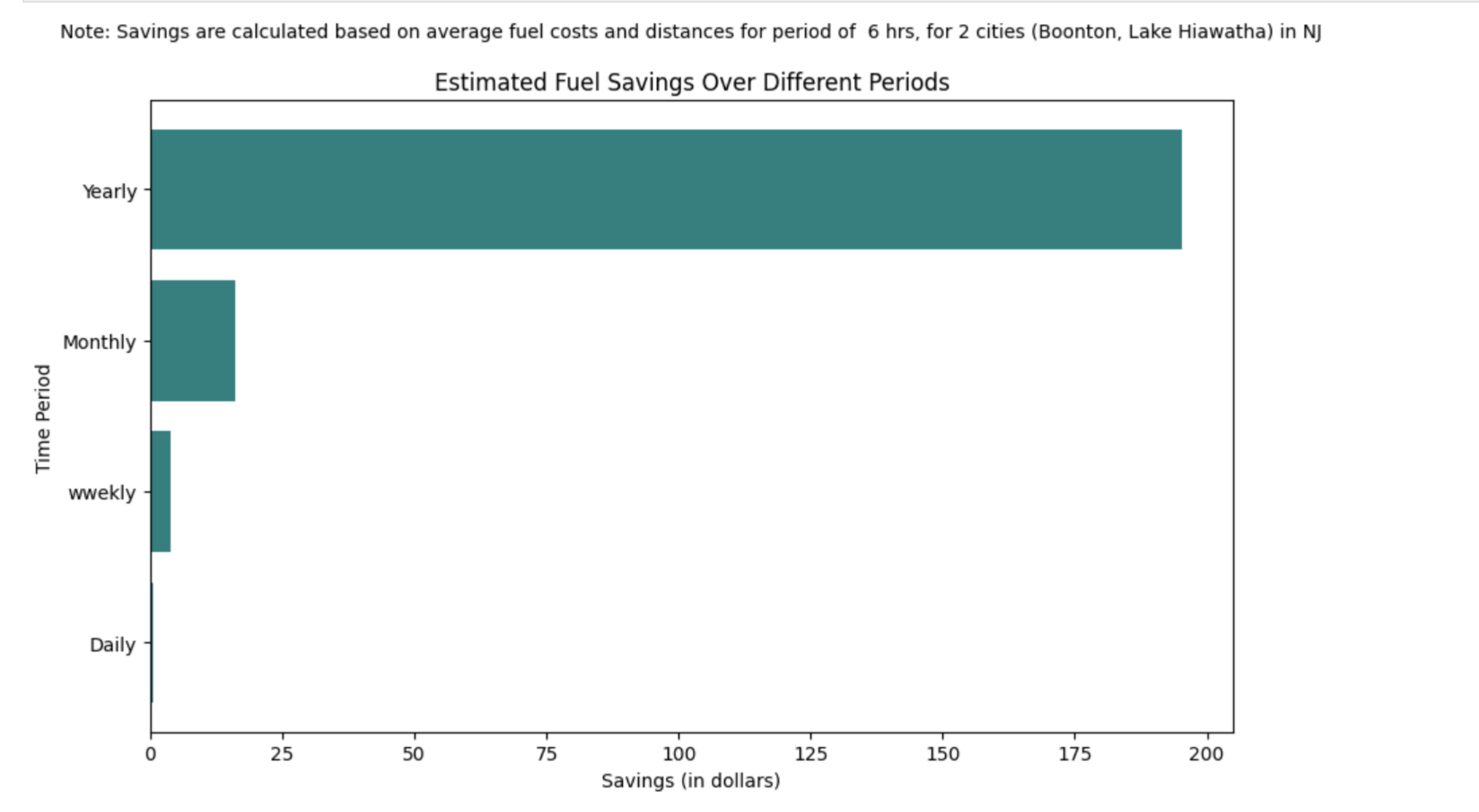
Presentation Layer (User Interface)



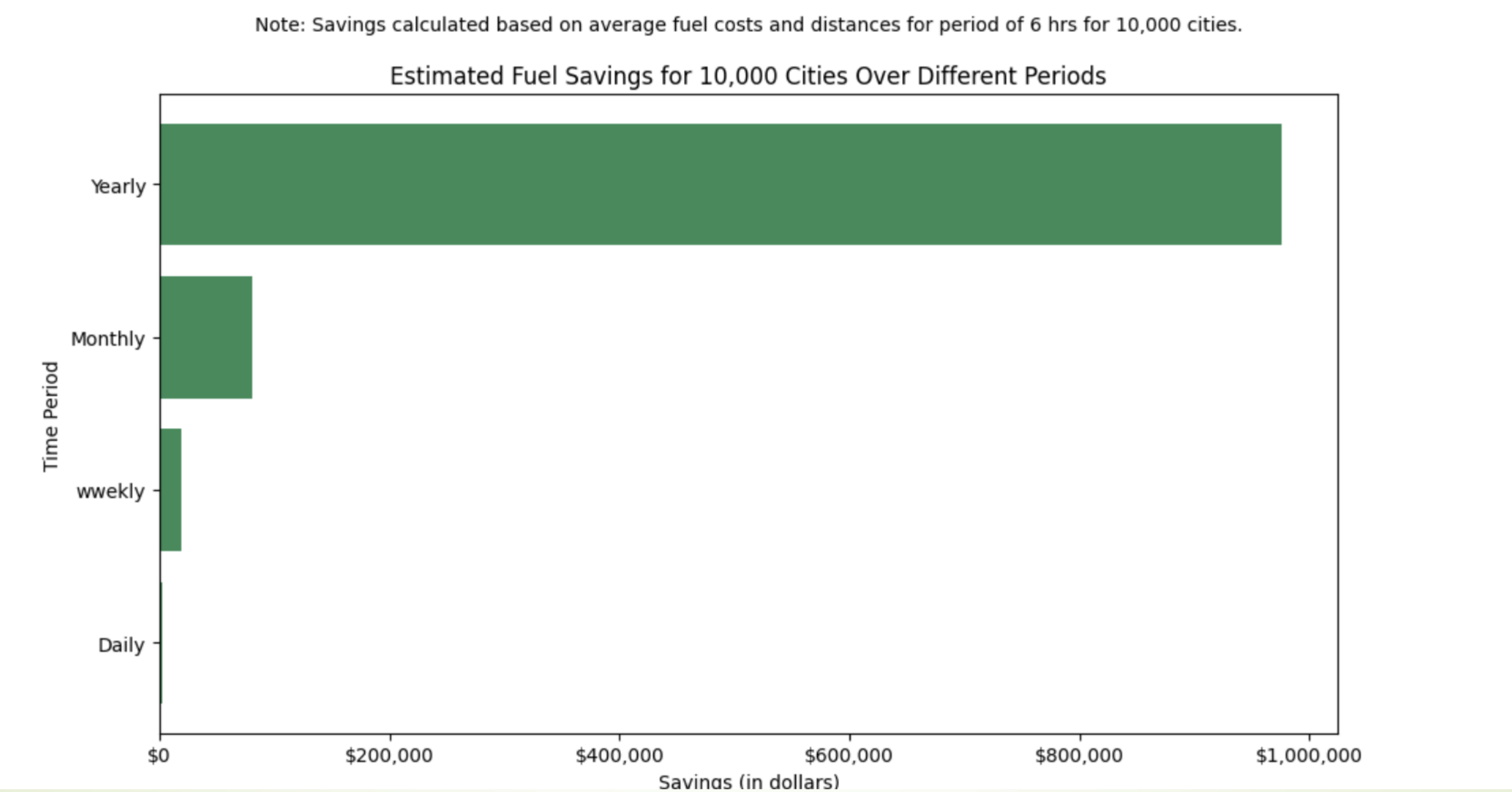
Presentation Layer (User Interface)



Presentation Layer (User Interface)



Presentation Layer (User Interface)



Application Layer (Flask API)

- **Flask Web Server**:

  Flask Web Application:

  - Displays optimized routes, distances, durations, and savings to users.
  - Handles user interactions and requests.
  - The **Flask** framework powers backend API and processes user requests.
  - When a user submits addresses, the frontend sends an HTTP request to Flask, which handles the routing logic and interacts with the service and integration layers and store the customer data into snowflake tables.
  - When a user requests optimized routes, the frontend sends an HTTP request to Flask, which handles the routing logic and interacts with the service and integration layers, to display optimized path, distance duration and web-map.

- **API Endpoints**:

  - POST /loading customer data: This endpoint takes the user's input (addresses) and triggers the storage of customer data into tables..
  - POST /optimize-route: This endpoint takes the user's input (city options from drop-down menu) and triggers the route optimization logic.
  - GET /map: This could return a map visualization after the optimization process is complete.
  - **JSON responses**: Flask sends back the results (routes, distances, etc.) as JSON for rendering on the frontend.
  - Used for geocoding customer addresses to latitude and longitude.
  - Provides route details and distance information.

Service Layer (Route Optimization and Data Processing)

- **Route Optimization Algorithm**

  Route Optimization Service:

  - Handles the logic for solving the Traveling Salesman Problem (TSP) using Graph Neural Networks (GNN).
  - Calculates optimized routes for each cluster of customer addresses.

  Estimation Service:

  - Computes distances, durations, fuel costs, and savings for original and optimized routes.

- **Distance Calculation**:

  - Geodesic or **great-circle distance** formulas are used to compute the shortest path between two latitude/longitude points.
  - This is useful when finding the nearest UPS facility or calculating route distances.

  Method:

  - calculate_distance(): Implements Haversine or geodesic distance calculations to measure physical distance between customer addresses and UPS facilities.

- **Traffic and Weather Integration**:

  - **Google Maps API**: To factor in real-time traffic, this layer communicates with the Google Maps API to get directions and travel times between points, considering current traffic conditions.
  - **OpenWeather API**: Retrieves real-time weather data and adjusts the routes dynamically if any locations face severe weather conditions.

  Methods:

  - get_shortest_route_with_traffic(): Retrieves traffic-aware routes from the Google Maps API.
  - adjust_route_for_weather(): Modifies routes based on weather data to avoid delays or hazardous conditions.

Integration Layer (APIs like Google Maps and OpenWeather)

- **Google Maps API**:

  The code interacts with the **Google Maps Directions API** to:

  - Get geocoded locations for customer addresses (latitude and longitude).
  - Calculate the **optimal route** between points, considering traffic.
  - Retrieve real-time travel duration and distances.

  Method:

  - get_shortest_route_with_traffic(): Calls the Google Maps API to fetch the shortest route based on traffic conditions.

- **OpenWeather API**:

  - Integrates real-time weather data to adjust the route if necessary.
  - Weather can impact routes, and the system will dynamically reroute around hazardous weather (like storms, heavy rain).

  Method:

  - get_weather_data(): Fetches weather conditions based on customer address coordinates.
  - adjust_route_for_weather(): Adjusts routes based on weather alerts.

- **Data Retrieval Services**:

  - Fetches customer and UPS facility data from the Snowflake database.
  - Responsible for querying and returning data in structured formats.

  Clustering Service:

  - Implements KMeans clustering to group customer addresses.
  - Determines centroids for each cluster for route optimization.

❖ **Clustering Service**:

- Implements KMeans clustering to group customer addresses.
- Determines centroids for each cluster for route optimization.

❖ **Visualization Layer**

Visualization Libraries (Matplotlib, etc.):

- Generates plots and charts for visual comparisons of original vs. optimized routes.
- Displays summary statistics and results of the analysis.

❖ **High-Level Data Flow**

- **User Input**: Users input addresses through the web interface.
- **Data Retrieval**: The Flask application retrieves customer and facility data from Snowflake.
- **Geocoding**: The application sends requests to the Google Maps API for geocoding customer addresses.
- **Clustering**: The clustered addresses are processed to find optimal routes.
- **Route Optimization**: The TSP solution is calculated using the GNN model.
- **Estimation**: Distances, durations, and costs are estimated for both original and optimized routes.
- **Visualization**: Results are visualized and displayed on the web interface for user review.
- **Storage**: Results are stored in the Snowflake database for future analysis and reporting.

Data Layer (Snowflake Database)

FACILITY_DB / FACILITY_SC / CUST_DATA_TB

FACILITY_DB / FACILITY_SC / CUST_DATA_TB

Load Data

Table ACCOUNTADMIN 37 minutes ago 16 6.5KB

Table Details Columns Data Preview Copy History

COMPUTE_WH 16 Rows • Updated 1 minute ago

	CUST_ADDRESS	CUST_CITY	CUST_STATE	CUST_ZIP
1	116 N Beverwyck Rd, Lake Hiawatha, NJ	Parsippany-Troy Hills	NJ	07034
2	123 Main Street, Boonton, NJ 07005	Boonton	NJ	07005
3	456 Oak Avenue, Parsippany, NJ 07054	Parsippany-Troy Hills	NJ	07054
4	789 Maple Drive, Parsippany, NJ 07054	Parsippany-Troy Hills	NJ	07054
5	101 Pine Road, Roseland, NJ 07068	Roseland	NJ	07068
6	202 Elm Street, Parsippany, NJ 07054	Parsippany-Troy Hills	NJ	07054
7	23 Maple Avenue, Claymont, DE 19703	Claymont	DE	19703
8	456 Oak Street, Claymont, DE 19703	Claymont	DE	19703
9	875E Maple Drive, Paramus, NJ	Paramus	NJ	07652
10	202 Birch Lane, Claymont, DE 19703	Claymont	DE	19703
11	1101 Elm Street, Claymont, DE 19703	Claymont	DE	19703
12	26.5 Cherokee Ave,Lake Hiawatha, NJ	Parsippany-Troy Hills	NJ	07034
13	115 Lake Shore Dr,Lake Hiawatha, NJ	Parsippany-Troy Hills	NJ	07034
14	345B Main Street, Dover, NJ	Dover	NJ	07801
15	52C Oakwood Avenue, Montclair, NJ	Montclair	NJ	07043
16	654D Elm Street, Morristown, NJ	Morristown	NJ	07960

Data Layer (Snowflake Database)

FACILITY_DB / FACILITY_SC / FAC_DATA_TB

FACILITY_DB / FACILITY_SC / CUST_DATA_TB

... Load Data

Table ACCOUNTADMIN 41 minutes ago 16 6.5KB

Table Details Columns Data Preview Copy History

COMPUTE_WH 16 Rows • Updated just now

	CUST_ADDRESS	CUST_CITY	CUST_STATE	CUST_ZIP
1	116 N Beverwyck Rd, Lake Hiawatha, NJ	Parsippany-Troy Hills	NJ	07034
2	123 Main Street, Boonton, NJ 07005	Boonton	NJ	07005
3	456 Oak Avenue, Parsippany, NJ 07054	Parsippany-Troy Hills	NJ	07054
4	789 Maple Drive, Parsippany, NJ 07054	Parsippany-Troy Hills	NJ	07054
5	101 Pine Road, Roseland, NJ 07068	Roseland	NJ	07068
6	202 Elm Street, Parsippany, NJ 07054	Parsippany-Troy Hills	NJ	07054
7	23 Maple Avenue, Claymont, DE 19703	Claymont	DE	19703
8	456 Oak Street, Claymont, DE 19703	Claymont	DE	19703
9	875E Maple Drive, Paramus, NJ	Paramus	NJ	07652
10	202 Birch Lane, Claymont, DE 19703	Claymont	DE	19703
11	1101 Elm Street, Claymont, DE 19703	Claymont	DE	19703
12	26.5 Cherokee Ave,Lake Hiawatha, NJ	Parsippany-Troy Hills	NJ	07034
13	115 Lake Shore Dr,Lake Hiawatha, NJ	Parsippany-Troy Hills	NJ	07034
14	345B Main Street, Dover, NJ	Dover	NJ	07801
15	52C Oakwood Avenue, Montclair, NJ	Montclair	NJ	07043
16	654D Elm Street, Morristown, NJ	Morristown	NJ	07960

Data Layer (Snowflake Database)

FACILITY_DB / FACILITY_SC / FAC_LOC_TB

FACILITY_DB / FACILITY_SC / FAC_LOC_TB

... Load Data

Table ACCOUNTADMIN 22 hours ago 6.4K 226.0KB

Table Details Columns Data Preview Copy History

COMPUTE_WH 100 of 6.4K Rows • Updated just now

	NAME	ADDRESS	CITY	STATE	ZIP
1	UPS Alliance Location	150 SHALLOTTE CROSSING PKWY	Shallotte	NC	28470
2	The UPS Store	120 SHALLOTTE CROSSING	Shallotte	NC	28470
3	UPS Customer Center	1620 N 23RD ST	Wilmington	NC	28405
4	UPS Alliance Location	11845 RETAIL DR	Wake Forest	NC	27587
5	UPS Alliance Location	2950 MILLBROOK RD	Raleigh	NC	27604
6	The UPS Store	361 S COLLEGE RD	Wilmington	NC	28403
7	The UPS Store	3600 S COLLEGE RD E	Wilmington	NC	28412
8	The UPS Store	1121 MILITARY CUTOFF RD	Wilmington	NC	28405
9	The UPS Store	310 N FRONT ST	Wilmington	NC	28401
10	The UPS Store	773 MAIN ST	N Myrtle Bch	SC	29582
11	The UPS Store	9650 STRICKLAND RD	Raleigh	NC	27615
12	The UPS Store	9654 N KINGS HWY	Myrtle Beach	SC	29572
13	The UPS Store	13200 STRICKLAND RD	Raleigh	NC	27613
14	The UPS Store	8311 BRIER CREEK PKWY	Raleigh	NC	27617
15	The UPS Store	7413 SIX FORKS RD	Raleigh	NC	27615
16	The UPS Store	1444 JEFFREYS RD	Rocky Mount	NC	27804
17	The UPS Store	2664 TIMBER DR	Garner	NC	27529
18	The UPS Store	3126 W CARY ST	Richmond	VA	23221

Data Layer (Snowflake Database)

FACILITY_DB / FACILITY_SC / FAC_CORD_TB

FACILITY_DB / FACILITY_SC / FAC_CORD_TB

Table ACCOUNTADMIN 22 hours ago 6.4K 85.0KB

Table Details Columns **Data Preview** Copy History

COMPUTE_WH 100 of 6.4K Rows • Updated just now

	LATITUDE	LONGITUDE
1	33.978476	-78.374769
2	33.976327	-78.383009
3	34.260614	-77.919585
4	35.969576	-78.541809
5	35.849672	-78.581892
6	34.237177	-77.872078
7	34.180734	-77.891004
8	34.23606	-77.828461
9	34.239049	-77.949056
10	33.82714	-78.67878
11	35.901658	-78.655326
12	33.78404	-78.78437
13	35.906967	-78.72595
14	35.908406	-78.785256
15	35.880731	-78.646012
16	35.974472	-77.811352
17	35.715414	-78.639863
18	37.553478	-77.481341

Model Components

- **Geocoding Model**

  API-Driven Optimization Model

  - The overall model integrates several APIs to gather real-time data that enhances the optimization algorithm:
  - Google Maps API: For geocoding addresses, calculating directions, and adjusting for traffic.
  - OpenWeather API: For retrieving weather data and adjusting routes accordingly.
  - These APIs feed dynamic, real-world data into the TSP and route optimization model, ensuring that the solution is not only theoretically optimized but also practical in real-world conditions.

- **Clustering Model**

  KMeans Clustering:

  - Used to group customer addresses into clusters based on their proximity.
  - Helps to reduce the complexity of route optimization by minimizing the number of stops per route.
  - Determines centroids for each cluster, which serve as target points for optimization.

- **Route Optimization Model**

  Graph Neural Network (GNN):

  - Used to solve the Traveling Salesman Problem (TSP) for optimizing routes among multiple stops.
  - Leverages graph structures to represent facilities and customer addresses, capturing relationships and distances effectively.
  - Trains on historical data to predict efficient paths considering dynamic factors like traffic and weather.

- **Estimation Models**

  Statistical Estimation:

  - Calculates distances and durations for original and optimized routes.
  - Estimates fuel costs and potential savings based on vehicle efficiency and travel distances.
  - Provides monthly and yearly statistics to understand performance over time.

- **Great-Circle Distance Calculation Model**

- For each step in the route, the model calculates the distance between two geographic points (latitude and longitude) using the **Haversine formula** or **Geodesic distance**:
  - calculate_distance(): This function calculates the **shortest distance over the earth's surface** between two points.
  - It's based on **geodesic** calculations, often used for geographical distance measurement over long distances.
  - This component ensures that the model calculates **accurate physical distances** between points like customer addresses and UPS facilities, which are then fed into the route optimization algorithm.

- **Traffic and Weather Models** (Optional)

  Traffic Prediction Models:

  - If integrated, these models use historical traffic data to predict travel times under different conditions.

  Weather Impact Models:

  - Analyze the impact of weather conditions on travel times and route efficiency.
  - Can inform adjustments to routes based on expected weather events.

- **UPS Facility Nearest Location Model**
  - Another critical part of the model is finding the nearest UPS facility for a given set of customer addresses.
  - get_filtered_facilities(): This function queries the Snowflake database to return the UPS facilities that match the selected city or region.
  - The model then calculates the distance from each customer address to each facility, choosing the nearest one.
  - This ensures that the delivery originates from the most efficient starting point, minimizing unnecessary travel.

- **Weather-Aware Route Adjustment Model**
  - The model integrates real-time weather data from the OpenWeather API to adjust routes based on weather conditions (e.g., avoiding areas with heavy rain, storms, etc.).
  - The weather-aware component helps the model avoid delays or unsafe conditions, improving overall route reliability.

Summary of Model Interactions

- **Geocoding Model**: Transforms addresses into coordinates for subsequent processing.
- **Clustering Model**: Groups customer locations to simplify route optimization.
- **Route Optimization Model (GNN)**: Finds the most efficient paths considering multiple factors.
- **Estimation Models**: Provide analytics and insights on route performance and cost-effectiveness.

Algorithms

❖ **Geocoding Algorithm**

Google Maps Geocoding API:

- Converts customer addresses into geographic coordinates (latitude and longitude).
- Utilizes RESTful API calls to retrieve location data based on input addresses.
- Clustering Algorithm

❖ **KMeans Clustering:**

- Partitions customer addresses into kk clusters based on proximity.
- Steps involved:
- Select the number of clusters kk.
- Randomly initialize kk centroids.
- Assign each data point (customer address) to the nearest centroid.
- Recalculate centroids as the mean of the assigned points.
- Repeat steps 3-4 until convergence (centroids no longer change).

❖ **Route Optimization Algorithm**

Graph Neural Network (GNN):

- Solves the Traveling Salesman Problem (TSP) by modeling the customer addresses and facilities as nodes in a graph.
- Steps involved:
- Construct a graph with nodes representing locations and edges representing distances.
- Use message-passing techniques to aggregate information from neighboring nodes.
- Optimize the sequence of visits to minimize total travel distance or time.
- Train the model on historical route data to learn patterns.

❖ **Distance Calculation Algorithm**

Haversine Formula:

- Calculates the great-circle distance between two points on the Earth's surface given their latitude and longitude.
- Used for both estimating distances between facilities and customer addresses.

❖ **Estimation Algorithms**

Statistical Analysis Algorithms:

- Used to calculate monthly and yearly statistics for optimized route durations and distances.
- Includes methods for calculating averages, medians, and standard deviations to provide insights into performance.

- **Traffic and Weather Impact Algorithms (Optional)**

  Predictive Modeling Algorithms:

  - If integrated, these algorithms analyze historical traffic and weather data to forecast future conditions.
  - Techniques like time series analysis or regression may be used to predict how these factors will impact travel times.

Summary of Algorithm Interactions

- **Geocoding Algorithm**: Transforms addresses into coordinates for further processing.
- **Clustering Algorithm**: Reduces complexity by grouping customer addresses.
- **Route Optimization Algorithm (GNN)**: Finds optimal routes using graph structures and learned patterns.
- **Distance Calculation Algorithm**: Provides accurate distance estimates for routing decisions.
- **Estimation Algorithms**: Analyze and summarize performance metrics over time.

These algorithms work together to create an efficient and effective solution for optimizing supply chain routes, enhancing delivery efficiency, and reducing costs.