SSN COLLEGE OF ENGINEERING

DEPARTMENT OF INFORMATION TECHNOLOGY

UIT2402---ADVANCED DATA STRUCTURES AND ALGORITHM

PROJECT TITLE : DEPARTMENT LIBRARY MANAGEMENT SYSTEM

PROJECT REPORT

SUBMITTED BY

HARIKRISHNA V 3122235002304

ABSTRACT:

OptimRoute is a smart delivery optimization system designed for efficient last-mile logistics management. Tailored for companies like Big Basket, it enables drivers to plan and follow optimized routes, allows administrators to manage the delivery fleet and monitor performance, and empowers customers to track deliveries in real time. The system utilizes a Flask-based backend, MongoDB for data storage, and Google OR-Tools to solve the Traveling Salesman Problem (TSP) using Haversine-calculated distances between delivery points. By optimizing routes, OptimRoute minimizes fuel consumption, reduces delivery times, and lowers operational costs. Even a 1% gain in efficiency across thousands of deliveries can lead to massive cost savings and improved sustainability.

INTRODUCTION / MOTIVATION:

As the demand for online shopping increases, last-mile delivery becomes a bottleneck in logistics. Big Basket, with its large delivery fleet, requires a system that not only plans optimal delivery paths but also provides real-time driver tracking and centralized fleet control. OptimRoute addresses these needs through a unified system. Drivers sign up and input delivery locations, which are geocoded and optimized using TSP algorithms. Admins can manage driver information and track live locations, while customers can monitor deliveries by entering a driver's ID. This comprehensive system integrates real-time route optimization, role-based access, and map visualizations, helping companies improve logistics efficiency and reduce environmental impact.

PROBLEM STATEMENT:

Develop a delivery route optimization system for Big Basket's fleet of over 5,000 trucks, where each truck starts at a depot, visits n delivery points ($20 \leq n \leq 50$), and returns to the depot. Compute a cost matrix representing the distance-based costs between every pair of delivery points. Use this matrix to solve the Traveling Salesman Problem (TSP) for each truck, ensuring the route has the lowest overall cost. The system should include a backend for route computation, user management, and data storage, with interfaces for administrators, drivers, and customers to manage and track deliveries

Real-Time Application

OptimRoute provides a complete solution for urban delivery optimization. Drivers get guided paths, admins get operational insights, and customers receive live tracking. This enhances delivery efficiency, saves time and fuel, and improves the overall customer experience.

Requirement Analysis

Functional Requirements

- User signup/login for drivers, admins, and customers.
- Geocoding of delivery places to latitude/longitude.
- TSP-based route optimization using OR-Tools.
- Admin functionalities: add, edit, delete, view drivers.
- Driver route input and tracking.
- Customer route view via driver ID.
- Map-based route visualization.

Non-Functional Requirements

- Scalable to handle thousands of deliveries.
- Secure password storage with bcrypt.
- Fast response time for route optimization.
- Modular, responsive UI for all user roles.

Challenges

- Real-time distance calculations for 20–50 locations.
- Efficient TSP solving under time constraints.
- Secure handling of user data across roles.
- Concurrent usage by multiple users.
- Visualizing optimized paths interactively.

ALGORITHM DESIGN & DEVELOPMENT SOLUTION:

OptimRoute design addresses the need for efficient route optimization, secure data management, and seamless user interaction. The solution comprises multiple components, each leveraging appropriate algorithms and data structures.

1. Route Optimization: Google OR-Tools TSP Solver

- Nodes: delivery locations.
- Edges: weighted with Haversine distances.
- OR-Tools solves TSP using PATH_CHEAPEST_ARC.

2. Distance Matrix: Haversine Formula

- Calculates geospatial distance between each point.
- Efficient for real-time calculations.

3. User Authentication: MongoDB + bcrypt

- Users stored in separate collections.
- Secure password hashing using bcrypt.

4. Route Storage: MongoDB

- Stores optimized route data by driver.
- Enables retrieval and historical analysis.

Justification for Problem-Solving Method

The TSP is a well-known NP-hard problem, but Big Basket's delivery scenario (20–50 stops) is manageable with modern solvers like OR-Tools, which balance speed and optimality. The Haversine formula ensures accurate cost matrices, critical for realistic route planning. MongoDB's flexibility supports the system's data needs, while Flask provides a lightweight, scalable backend. The modular design separates concerns (authentication, route optimization, visualization), ensuring maintainability and extensibility.

PROCESS MANAGEMENT:

Agile methodology:

The project follows an Agile approach with iterative sprints to deliver functional components incrementally.

Sprint 1: Authentication & Project Setup (Week 1)

- Set up Flask, MongoDB, and frontend framework (e.g., React).
- Implement signup/login for admins, drivers, and customers with JWT.
- Create backend APIs for user data storage.

Sprint 2: Cost Matrix & TSP Solver (Week 2)

- Implement Haversine-based cost matrix computation.
- Integrate Google OR-Tools for TSP optimization.
- Store optimized routes in MongoDB.

Sprint 3: Route Visualization (Week 3)

- Develop map interface using Leaflet.js to display routes.
- Enable drivers to view optimized paths.
- Allow customers to track driver routes.

Sprint 4: Admin Dashboard & History (Week 4)

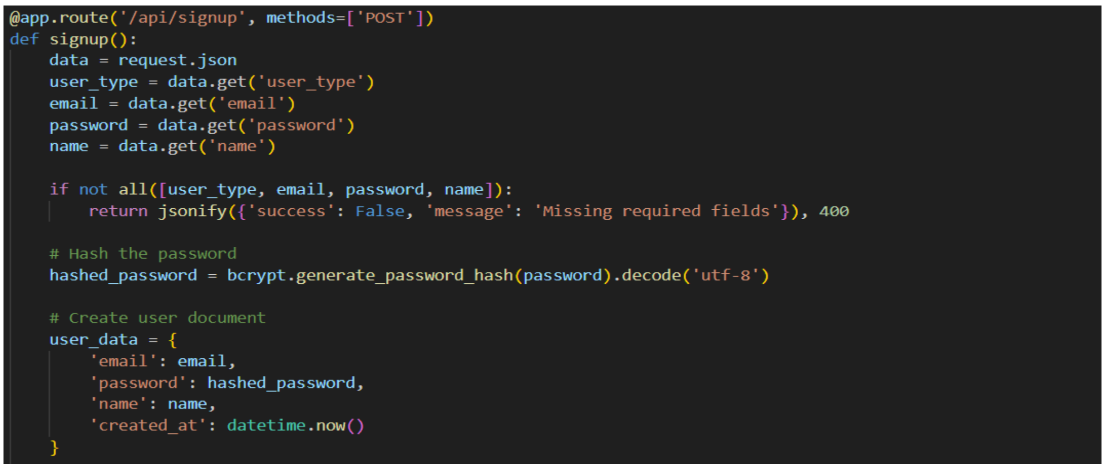
- Build admin interface to manage drivers and view routes.
- Implement route history retrieval and display.
- Add analytics for route efficiency.

Sprint 5: UI Refinement & Testing (Week 5)

- Refactor frontend into reusable components.
- Enhance UX with responsive design and styling.

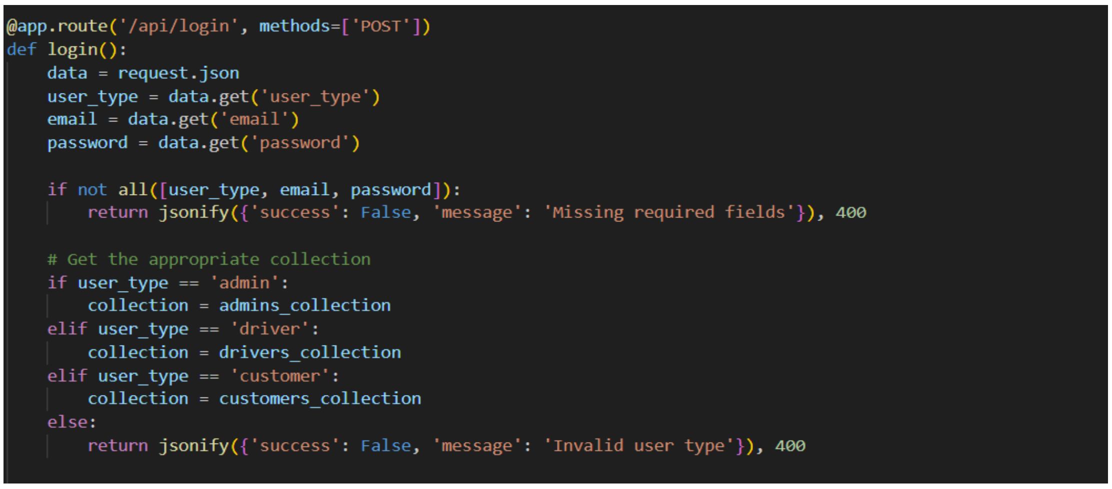
CODING & IMPLEMENTATION / DEPLOYMENT & TEST CASES:

i) User Registration Endpoint:



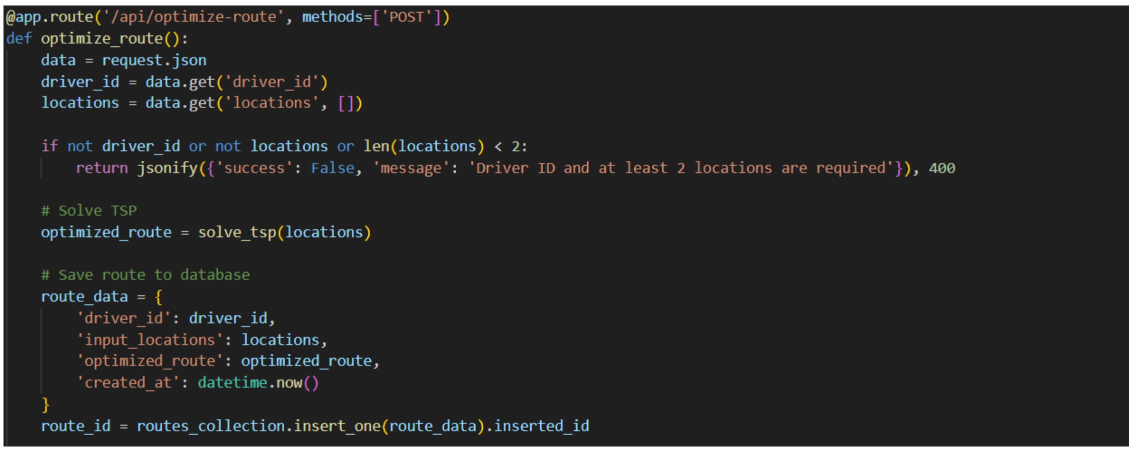
```python
@app.route('/api/signup', methods=['POST'])
def signup():
    data = request.json
    user_type = data.get('user_type')
    email = data.get('email')
    password = data.get('password')
    name = data.get('name')

    if not all([user_type, email, password, name]):
        return jsonify({'success': False, 'message': 'Missing required fields'}), 400

    # Hash the password
    hashed_password = bcrypt.generate_password_hash(password).decode('utf-8')

    # Create user document
    user_data = {
        'email': email,
        'password': hashed_password,
        'name': name,
        'created_at': datetime.now()
    }
```

ii) User Login Endpoint:



```python
@app.route('/api/login', methods=['POST'])
def login():
    data = request.json
    user_type = data.get('user_type')
    email = data.get('email')
    password = data.get('password')

    if not all([user_type, email, password]):
        return jsonify({'success': False, 'message': 'Missing required fields'}), 400

    # Get the appropriate collection
    if user_type == 'admin':
        collection = admins_collection
    elif user_type == 'driver':
        collection = drivers_collection
    elif user_type == 'customer':
        collection = customers_collection
    else:
        return jsonify({'success': False, 'message': 'Invalid user type'}), 400
```

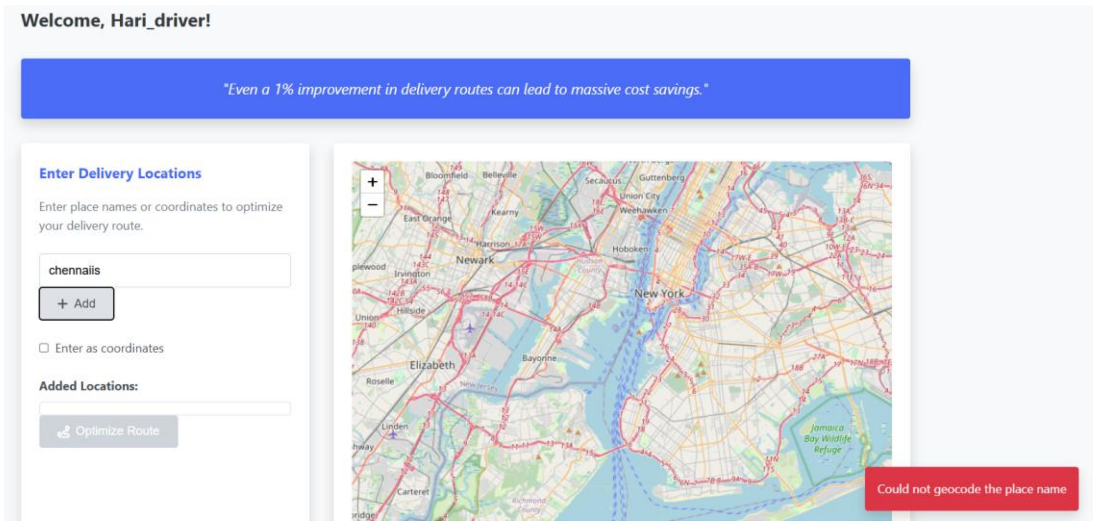
v) Solving the Traveling Salesman Problem (TSP):



```python
@app.route('/api/optimize-route', methods=['POST'])
def optimize_route():
    data = request.json
    driver_id = data.get('driver_id')
    locations = data.get('locations', [])

    if not driver_id or not locations or len(locations) < 2:
        return jsonify({'success': False, 'message': 'Driver ID and at least 2 locations are required'}), 400

    # Solve TSP
    optimized_route = solve_tsp(locations)

    # Save route to database
    route_data = {
        'driver_id': driver_id,
        'input_locations': locations,
        'optimized_route': optimized_route,
        'created_at': datetime.now()
    }
    route_id = routes_collection.insert_one(route_data).inserted_id
```

TEST CASES:

i) Error Message for Insufficient Destinations:


Welcome, Hari_driver!

"Even a 1% improvement in delivery routes can lead to massive cost savings."

Enter Delivery Locations

Enter place names or coordinates to optimize your delivery route.

chennaiis

+ Add

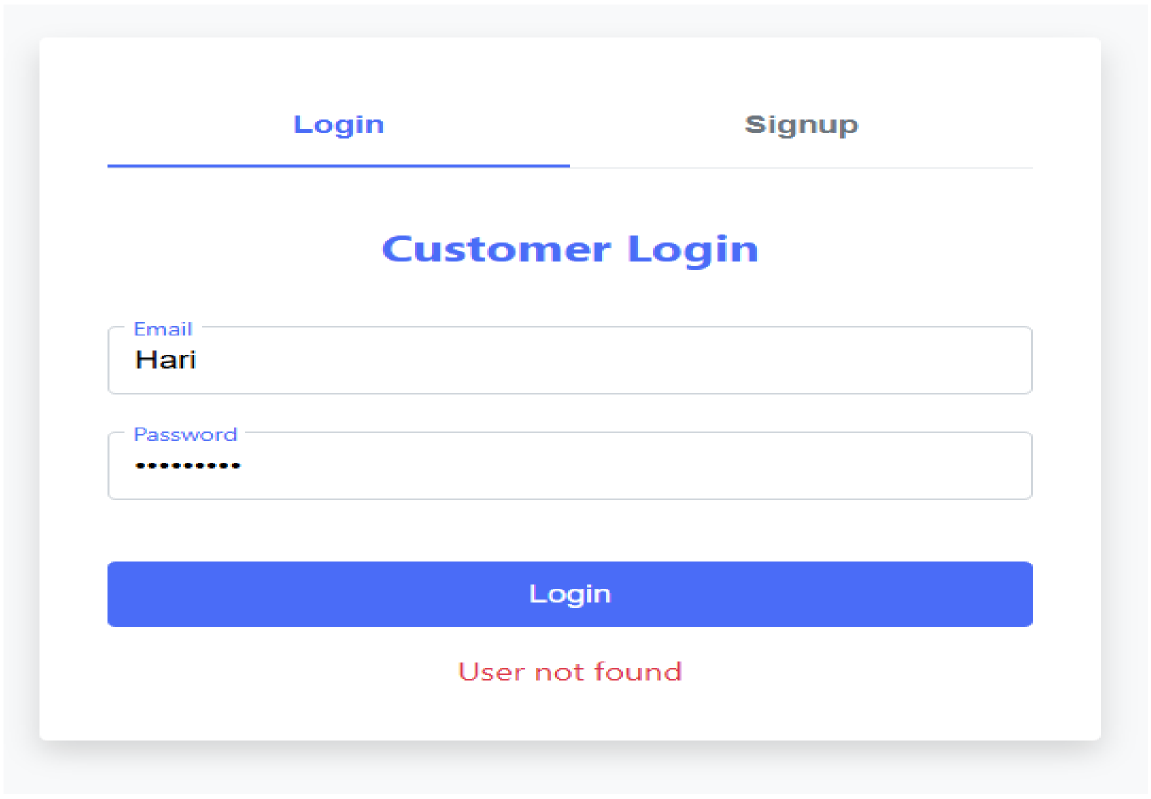
☐ Enter as coordinates

Added Locations:

Optimize Route

Could not geocode the place name

ii) Invalid Login Credentials:


Login Signup

Customer Login

Email: Hari

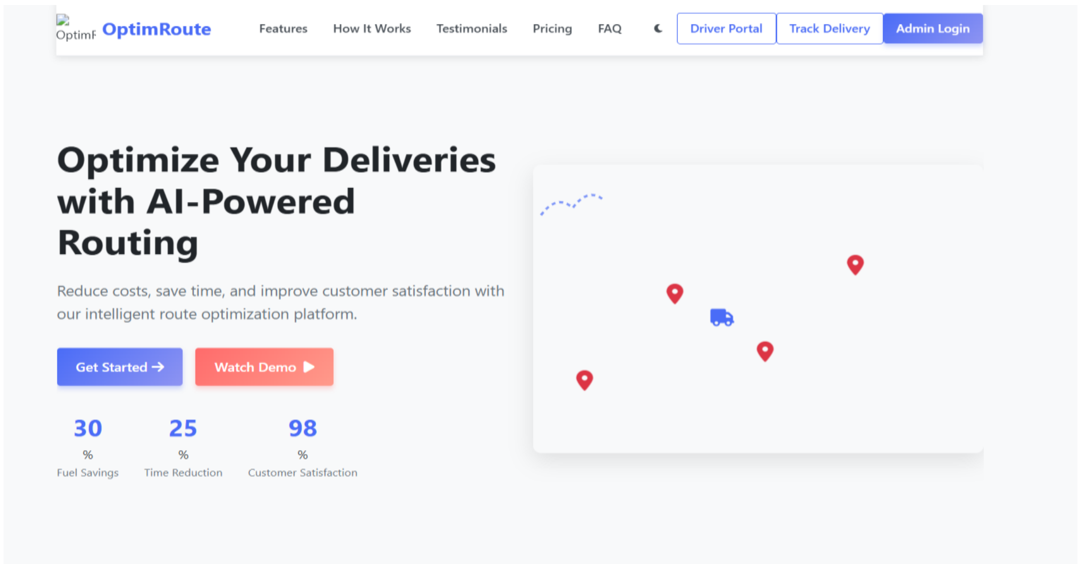
Password: •••••••••

Login

User not found

RESULTS (SCREENSHOTS):

i) Home page:

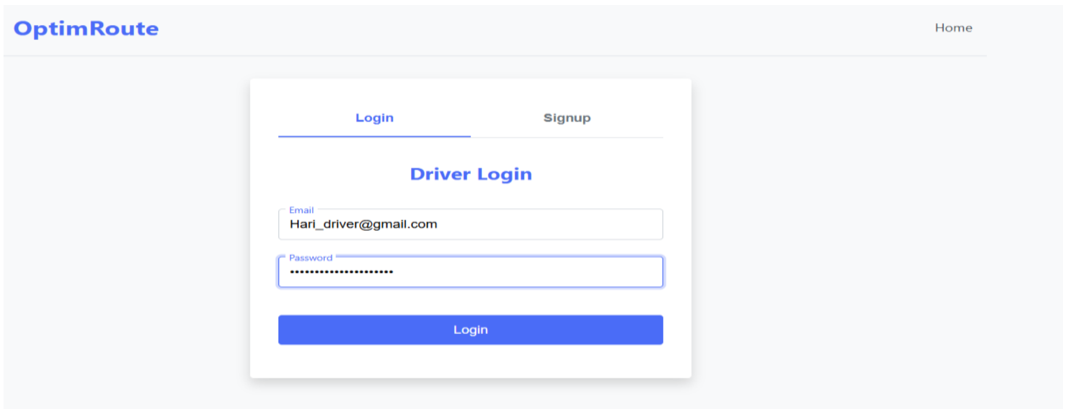


Optimize Your Deliveries with AI-Powered Routing

Reduce costs, save time, and improve customer satisfaction with our intelligent route optimization platform.

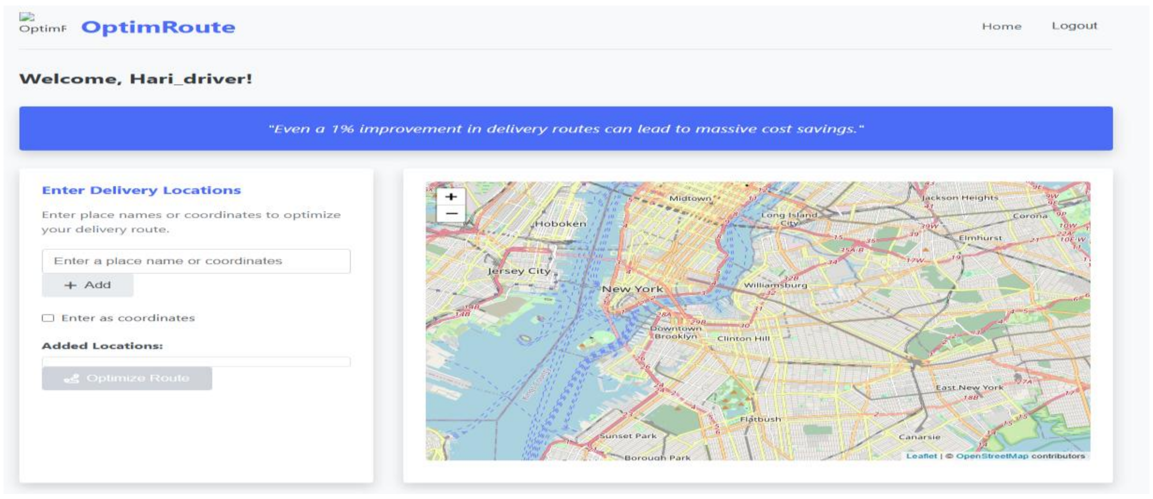
30 % Fuel Savings
25 % Time Reduction
98 % Customer Satisfaction

ii) Login page:



Email: Hari_driver@gmail.com
Password: ••••••••••••••••••••••

iii) Route Optimization Interface:

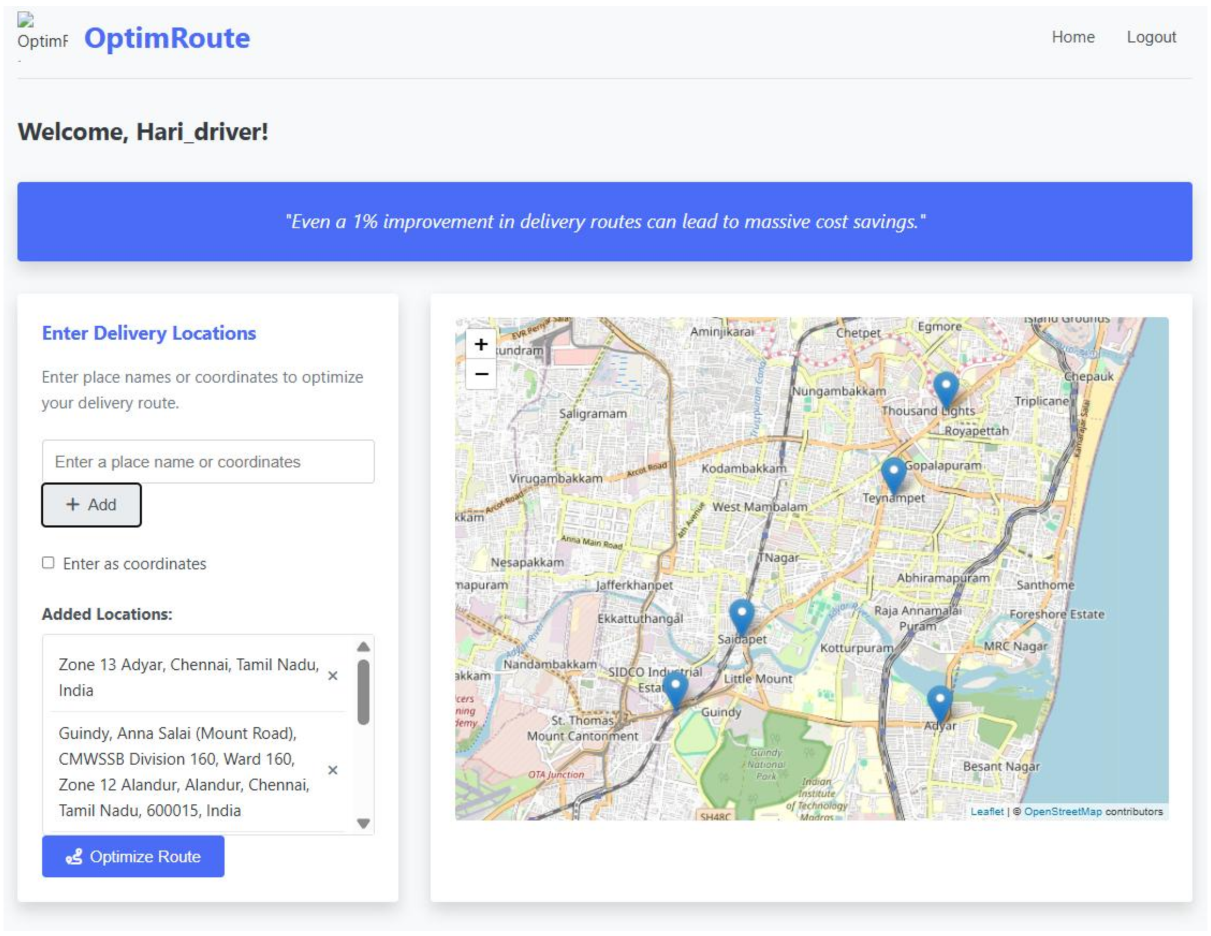


Welcome, Hari_driver!

"Even a 1% improvement in delivery routes can lead to massive cost savings."

Enter place names or coordinates to optimize your delivery route.

iv) Route Optimization with Multiple Destinations:



Welcome, Hari_driver!

"Even a 1% improvement in delivery routes can lead to massive cost savings."

Enter Delivery Locations

Enter place names or coordinates to optimize your delivery route.

Enter a place name or coordinates

+ Add

☐ Enter as coordinates

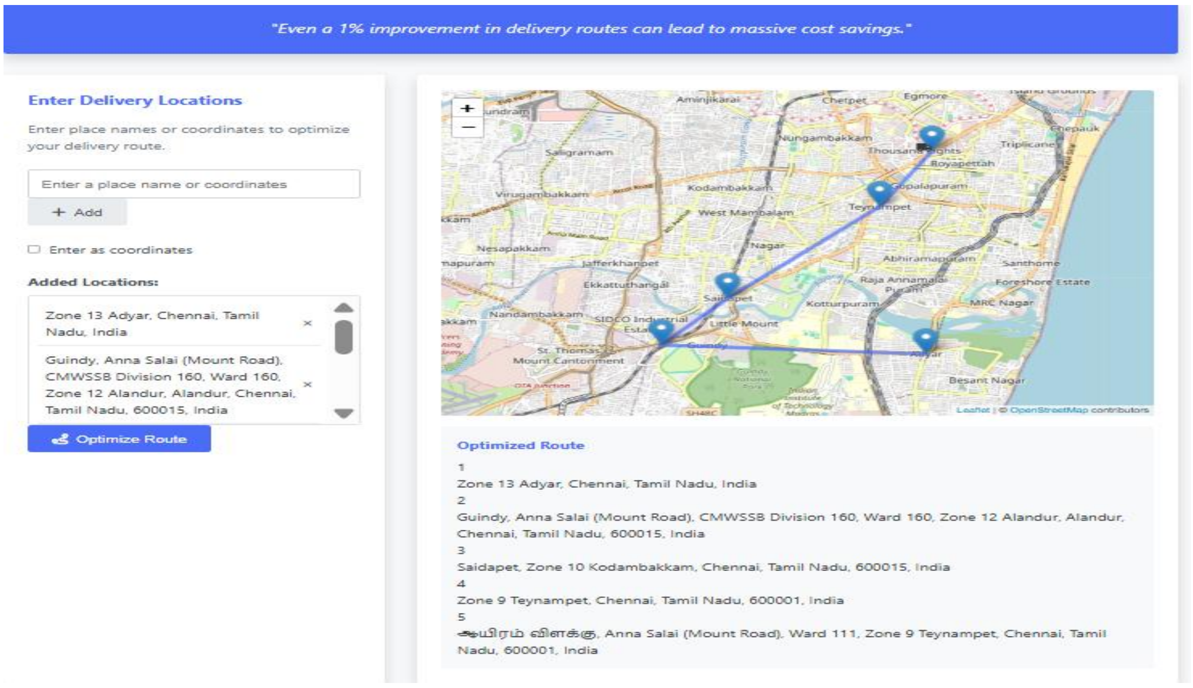
Added Locations:

Zone 13 Adyar, Chennai, Tamil Nadu, India ×

Guindy, Anna Salai (Mount Road), CMWSSB Division 160, Ward 160, Zone 12 Alandur, Alandur, Chennai, Tamil Nadu, 600015, India ×

Optimize Route

v) Optimized path:



"Even a 1% improvement in delivery routes can lead to massive cost savings."

Enter Delivery Locations

Enter place names or coordinates to optimize your delivery route.

Enter a place name or coordinates

+ Add

☐ Enter as coordinates

Added Locations:

Zone 13 Adyar, Chennai, Tamil Nadu, India ×

Guindy, Anna Salai (Mount Road), CMWSSB Division 160, Ward 160, Zone 12 Alandur, Alandur, Chennai, Tamil Nadu, 600015, India ×

Optimize Route

Optimized Route

1
Zone 13 Adyar, Chennai, Tamil Nadu, India
2
Guindy, Anna Salai (Mount Road), CMWSSB Division 160, Ward 160, Zone 12 Alandur, Alandur, Chennai, Tamil Nadu, 600015, India
3
Saidapet, Zone 10 Kodambakkam, Chennai, Tamil Nadu, 600015, India
4
Zone 9 Teynampet, Chennai, Tamil Nadu, 600001, India
5
ஆயிரம் விளக்கு, Anna Salai (Mount Road), Ward 111, Zone 9 Teynampet, Chennai, Tamil Nadu, 600001, India

GAP ANALYSIS:

Current State

1. **Functionalities Implemented:**
   - User authentication for admins, drivers, customers.
   - Cost matrix computation using Haversine formula.
   - TSP solver for route optimization.
   - Route storage and retrieval in MongoDB.
   - Basic interfaces for role-based access.
2. **Challenges:**
   - Limited scalability testing for 5,000+ trucks.
   - Basic visualization without real-time updates.
   - No integration with external factors (e.g., traffic, weather).
   - Manual input of delivery coordinates.

Desired State

1. **Expected Functionalities:**
   - Real-time route updates based on traffic or delays.
   - Automated coordinate input via address geocoding.
   - Scalable cloud deployment for fleet-wide use.
2. **Performance Benchmarks:**
   - TSP solution computed in <1 second for 50 stops.
   - System uptime of 99.9% for daily operations.
   - Support for 10,000 concurrent users.
3. **User Experience Goals:**
   - Intuitive interfaces with minimal learning curve.
   - Real-time feedback on route status.
   - Personalized dashboards for each role.

Identified Gaps

1. **Scalability:**
   - Current: Local deployment with limited load testing.
   - Desired: Cloud-based system handling thousands of trucks.
2. **Real-Time Integration:**
   - Current: Static distance-based costs.
   - Desired: Dynamic costs incorporating traffic and weather.
3. **Visualization:**
   - Current: Basic map rendering.
   - Desired: Interactive, real-time maps with analytics.
4. **Data Input:**
   - Current: Manual coordinate entry.
   - Desired: Automated geocoding from addresses.

Action Plan to Bridge Gaps

1. **Enhance Scalability:**
   - Deploy on AWS with load balancing.
   - Optimize MongoDB queries for high throughput.
2. **Integrate Real-Time Data:**
   - Use APIs (e.g., Google Maps) for traffic updates.
   - Incorporate weather data for cost adjustments.
3. **Improve Visualization:**
   - Add Leaflet plugins for interactive maps.
   - Implement analytics dashboards with D3.js.
4. **Automate Inputs:**
   - Integrate geocoding APIs for address-to-coordinate conversion.
   - Validate inputs to prevent errors.

CONCLUSION & FUTURE WORK:

OptimRoute effectively showcases how modern optimization algorithms and geospatial tools can address real-world logistics challenges. By integrating route optimization through the Traveling Salesman Problem (TSP), real-time location tracking, and user-specific interfaces for drivers, administrators, and customers, the system streamlines the entire delivery process. It reduces operational costs, improves delivery efficiency, and provides greater visibility into fleet activities. The backend built with Flask and MongoDB offers a scalable and secure foundation, while the use of Google OR-Tools ensures accurate and efficient route planning.Looking ahead, several future enhancements are envisioned to elevate OptimRoute's capabilities. A dedicated mobile application for both drivers and customers would improve accessibility and provide real-time updates on the go. Integrating live traffic data through Google Maps APIs could make the system more dynamic, adjusting routes based on real-world conditions. Predictive analytics features could be introduced to estimate delivery times and proactively manage delays. Furthermore, advanced dashboards for administrators—complete with delivery heatmaps and performance analytics—would offer deeper insights into operations and help in making data-driven decisions.

REFERENCES:

- https://developers.google.com/optimization/routing/tsp – Google OR-Tools for TSP.
- https://flask.palletsprojects.com/ – Flask documentation.
- https://www.mongodb.com/docs/ – MongoDB documentation.
- https://en.wikipedia.org/wiki/Travelling_salesman_problem – Overview of TSP.
- https://arxiv.org/abs/1803.08475 – "Solving the Traveling Salesman Problem with Reinforcement Learning" by Khalil et al.