Report On

ROUTE OPTIMIZATION

Submitted in partial fulfilment of the requirements of the Skill Based Language (SBL) Course project in
Semester IV of Second Year Computer Engineering

by
Veer Jain (Roll No. 62)
Akash Nadar (Roll No. 63)
Satyam Yadav (Roll No. 70)

Supervisor
Prof. Sneha Mhatre

Vidyavardhini's College of Engineering & Technology

Department of Computer Engineering

(2023-24)

2

Department of Computer Engineering
Vidyavardhini's College of Engineering & Technology

Department of Computer Engineering
CERTIFICATE

This is to certify that the course project entitled "ROUTE OPTIMIZATION" is a bonafide work of "Veer Jain (Roll No. 62), Akash Nadar (Roll No. 63), Satyam Yadav
(Roll No. 70)" submitted to the University of Mumbai in partial fulfillment of the requirement for the Course project in semester IV of Second Year Computer Engineering.

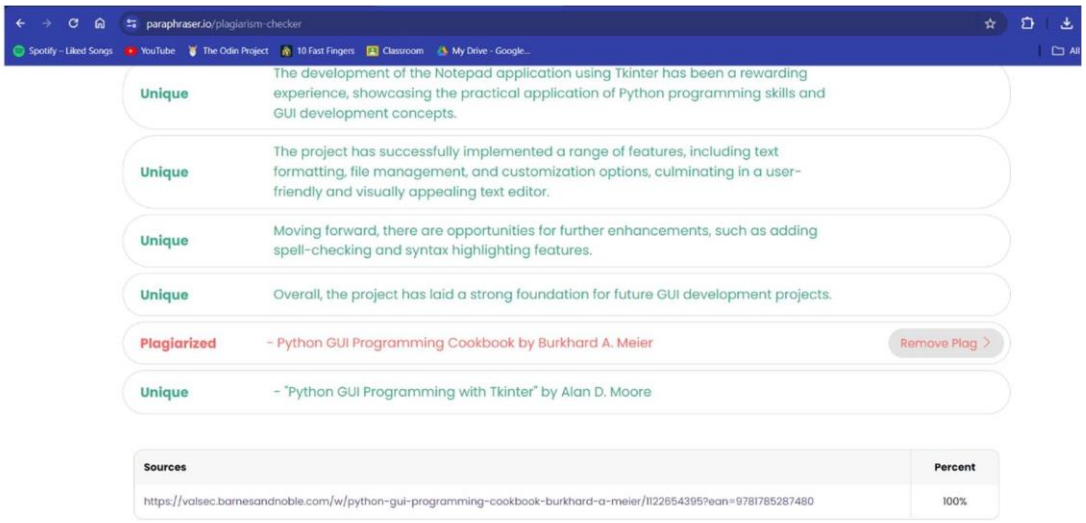
Supervisor

Prof. Sneha Mhatre

Dr Megha Trivedi
Head of Department

Dr. H.V. Vankudre
Principal

3 PLAGIARISM REPORT:



4 Abstract:

Route optimization plays a crucial role in various domains such as transportation, logistics, and urban planning. This project aims to develop a route planner application using streamlit, pandas, osmnx, networkx, and matplotlib in Python. The application allows users to find the shortest route between two locations within a specified area, leveraging road network data from OpenStreetMap.

The workflow involves several key steps:

1. Data Loading: The application loads depot and bus stop data from CSV files to create a comprehensive dataset of all stops.
2. Graph Initialization: The road network graph for the specified area (in this case, Mumbai, India) is initialized using osmnx, with options to customize network type and other parameters.
3. Route Calculation: The application calculates the shortest route between the selected origin and destination using networkx's shortest path algorithm, considering the road network's edges' lengths.
4. Visualization: The resulting route is visualized on the road network graph using matplotlib, highlighting the chosen path in red.

The user interacts with the application through a simple user interface created with streamlit, selecting the origin and destination from dropdown menus and triggering the route calculation with a button click. The application then displays the total distance of the route and the visual representation of the path on the map.

Overall, this project demonstrates the practical application of route optimization techniques and provides a user-friendly tool for finding the shortest route between two points within a specified area.

5) INDEX

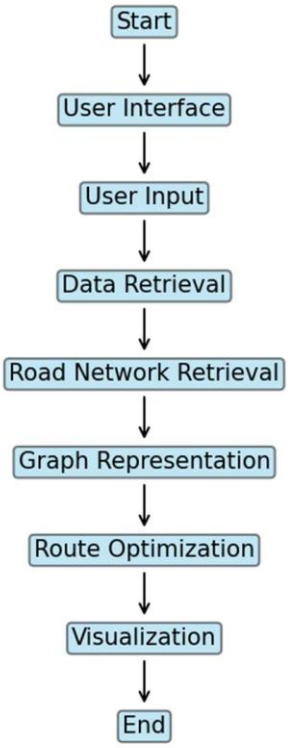
Contents

Problem Statement:

The aim is to develop a robust route optimization web application tailored for Mumbai, India, to facilitate efficient navigation between locations. The application should utilize OpenStreetMap data and advanced algorithms to optimize routes, thereby minimizing travel time, distance, and fuel consumption for users.

7 Block Diagram, description and Working:

- Block Diagram:



Description:

- User-Friendly Interface : The application boasts a user-friendly interface that allows users to easily input their origin and destination locations within Mumbai. Dropdown menus or search bars facilitate the selection of locations, ensuring a seamless user experience.
- Data Retrieval and Processing: Upon selecting the origin and destination locations, the application retrieves relevant data from various sources. This includes depot and bus stop data, which is crucial for route planning. The data is processed and formatted using Pandas to ensure compatibility with further analysis.
- Road Network Representation: Utilizing OSMnx, the application accesses OpenStreetMap data to retrieve detailed information about Mumbai's road network. This data is transformed into a graph representation using NetworkX, where nodes represent points of interest (e.g., intersections, depots) and edges represent road segments connecting these nodes.
- Route Optimization: NetworkX provides powerful algorithms for route optimization, including Dijkstra's algorithm. This algorithm is employed to find the shortest path between the selected origin and destination nodes on the road network graph. Factors such as distance, travel time, and user preferences are taken into account during the optimization process.
- Visualization: Matplotlib is used to visualize the optimized route on an interactive map. Users can see the selected path highlighted on the map, along with relevant details such as distance and notable landmarks. This visualization aids users in understanding the recommended route and making informed decisions about their travel plans.
- Feedback and Customization: The application may provide feedback to users, such as alternative routes or potential traffic delays. Additionally, users can customize their route preferences, prioritizing factors such as shortest distance or quickest travel time. This customization enhances the user experience and allows users to tailor their routes according to their specific needs.
- Optimization and Efficiency: To ensure optimal performance and efficiency, the application implements various optimization techniques. Caching mechanisms store previously calculated routes to avoid redundant computations, while parallel processing and optimization algorithms reduce processing time for large datasets. These optimizations enhance the responsiveness and usability of the application.

- Working:

- User Input and Interface Interaction :
  - Users access the route optimization application through a web browser.
  - They are greeted with a user-friendly interface where they can input their origin and destination locations within Mumbai, India.
  - The interface, designed using Streamlit, provides dropdown menus or search bars for users to select their locations.
- Location Selection and Data Retrieval:
  - Once users select their origin and destination locations, the application retrieves relevant data.
  - This includes fetching depot and bus stop data, which is necessary for route planning.
  - The application utilizes Pandas to load and preprocess this data, ensuring it is in a suitable format for further analysis.
- Road Network Acquisition :
  - With the selected locations in hand, the application proceeds to fetch road network data for Mumbai, India.
  - OSMnx comes into play here, allowing the application to access OpenStreetMap data and extract information about the city's street network.
  - The retrieved road network data forms the foundation for route optimization.
- Graph Representation and Optimization:
  - The road network data is then transformed into a graph representation using NetworkX.
  - Nodes in the graph correspond to points of interest such as intersections, depots, and bus stops, while edges represent road segments connecting these nodes.
  - NetworkX provides algorithms for route optimization, such as Dijkstra's algorithm, which is used to find the shortest path between the selected origin and destination nodes.
  - This optimization process takes into account factors like distance, travel time, and any specified user preferences.
- Route Calculation and Visualization:

- Once the shortest path is determined, the application visualizes the optimized route on an interactive map.
- Matplotlib is used to plot the road network graph and overlay the calculated route.
- Users can see the selected path highlighted on the map, along with relevant details such as distance and any notable landmarks or waypoints along the route.
- This visualization aids users in understanding the recommended route and helps them make informed decisions about their travel plans.

- Feedback and Additional Features:
  - The application may provide feedback to users, such as alternative routes or potential traffic delays.
  - Additional features, such as the ability to adjust route preferences (e.g., prioritize shortest distance or quickest travel time) or save favorite routes for future reference, enhance the user experience. o Users can interact with the interface to explore different routes and customize their travel plans according to their preferences and requirements.
- Optimization and Efficiency:
  - To ensure optimal performance and efficiency, the application may implement various optimization techniques.
  - o Caching mechanisms may be used to store previously calculated routes and avoid redundant computations.
  - o Parallel processing and optimization algorithms can help reduce processing time for large datasets, improving the overall responsiveness of the application.

Module Description:

1. Data Acquisition and Preprocessing : Retrieve and preprocess

geographic data including road networks, depots, and bus stops from OpenStreetMap and other relevant sources. This step involves data cleaning, integration, and formatting.

2. Graph Representation: Construct a graph representation of the road network using OSMnx and NetworkX libraries. Nodes represent intersections or points of interest, and edges represent road segments. Ensure the graph is weighted with attributes such as distance, travel time, and traffic conditions.
3. Route Optimization Algorithms: Implement advanced optimization algorithms such as Dijkstra's algorithm, A* algorithm, or genetic algorithms to find the most efficient routes between origin and destination pairs. Consider factors like traffic congestion, road closures, and timedependent variations.
4. User Interface and Interaction: Develop an intuitive and interactive user interface using Streamlit to allow users to input their starting point, destination, and any additional preferences or constraints. Provide options for users to prioritize factors such as shortest distance, quickest travel time, or minimal fuel consumption.
5. Visualization and Feedback: Visualize optimized routes on an interactive map using Matplotlib or other mapping libraries. Display route details including distance, estimated travel time, and any relevant landmarks or waypoints. Provide feedback to users regarding alternate routes, traffic updates, and potential delays.
6. Performance Optimization: Optimize the application's performance to handle large-scale route optimization tasks efficiently. Implement caching mechanisms, parallel processing, and other optimization techniques to reduce computation time and resource utilization.

Brief description of software and hardware used and its programming:

1. Streamlit : Streamlit is a popular Python library used for building interactive web applications for data science and machine learning tasks. It simplifies the process of creating web interfaces by allowing developers to write Python scripts that are automatically converted into web apps. In this project, Streamlit is used to create the user interface where users can input their origin and destination points and visualize the optimized routes.

2. Pandas: Pandas is a powerful data manipulation and analysis library for Python. It provides data structures and functions to efficiently manipulate and analyze structured data, such as CSV files or database tables. In this project, Pandas is used to read and preprocess depot and bus stop data from CSV files, as well as to manipulate the data within the application. 3. OSMnx: OSMnx is a Python library for working with OpenStreetMap (OSM) data. It allows users to retrieve, construct, analyze, and visualize street networks from OSM data anywhere in the world. In this project, OSMnx is used to retrieve road network data for Mumbai, which is essential for route optimization.

4. NetworkX: NetworkX is a Python library for the creation, manipulation, and study of complex networks. It provides tools for analyzing the structure and properties of networks, as well as algorithms for network optimization and traversal. In this project, NetworkX is used to represent the road network graph and to perform route optimization algorithms such as Dijkstra's algorithm.
5. Matplotlib: Matplotlib is a comprehensive library for creating static, animated, and interactive visualizations in Python. It provides a wide range of plotting functions and customization options for creating highquality plots and figures. In this project, Matplotlib is used to visualize the road network graph and optimized routes on interactive maps.

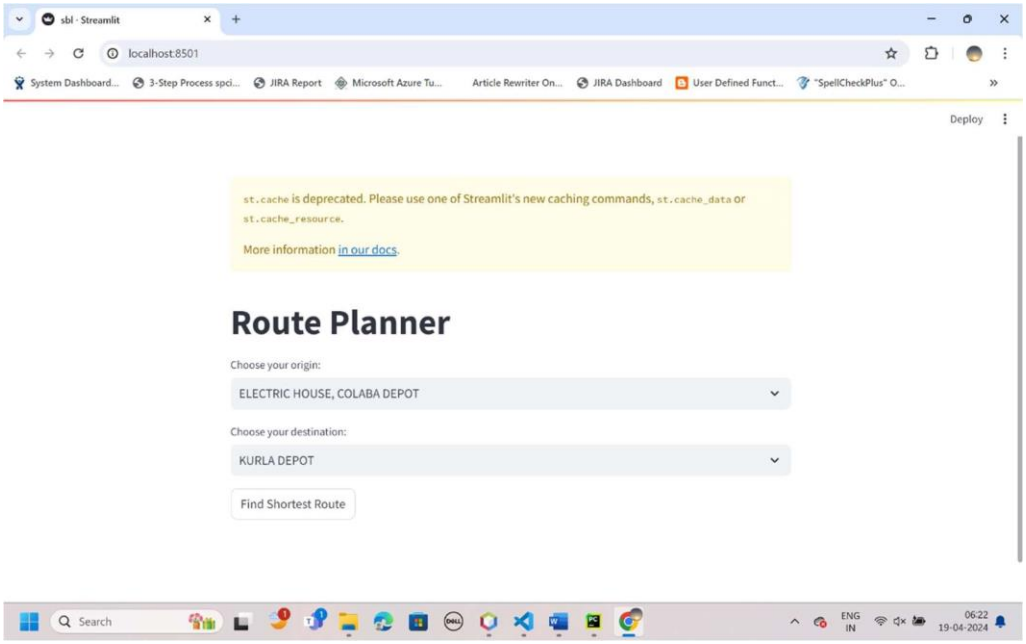
Code :

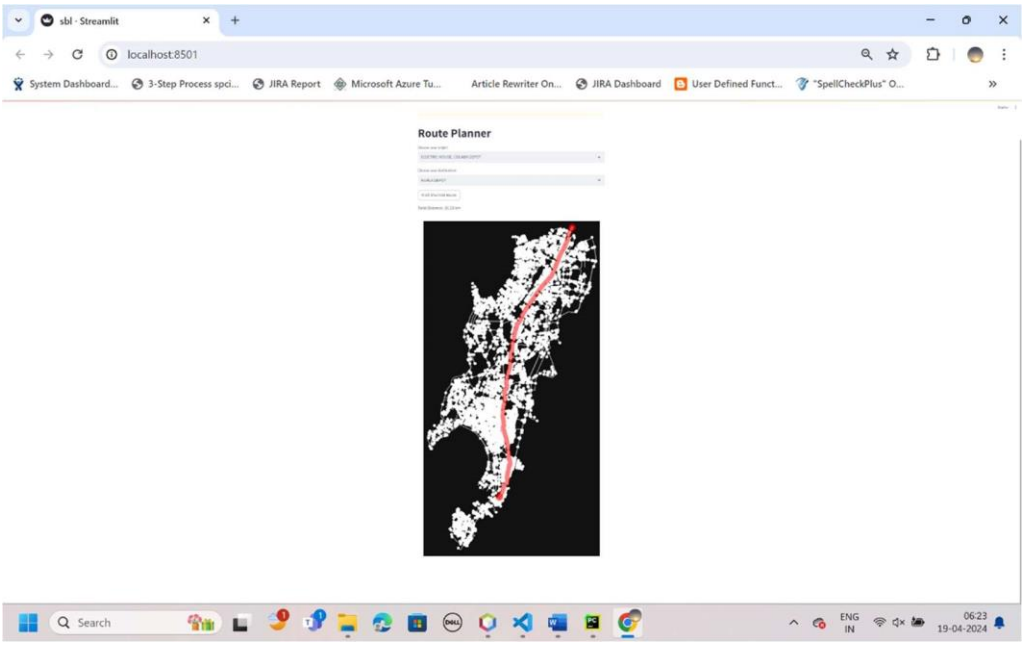
```
import streamlit as st import
pandas as pd import osmnx as
ox import networkx as nx
import matplotlib.pyplot as plt
@st.cache_data def
load_data():
    depots_data = pd.read_csv("depots_data.csv")     bus_stops_data =
pd.read_csv("bus_stops_data.csv")     all_stops = pd.concat([depots_data,
bus_stops_data], ignore_index=True)     return all_stops all_stops =
load_data()
@st.cache(allow_output_mutation=True, show_spinner=False) def
init_graph():     return ox.graph_from_place('Mumbai, India',
network_type='drive') G = init_graph() def get_route(origin,
destination):
    origin_node = ox.distance.nearest_nodes(G, origin[1], origin[0])
destination_node = ox.distance.nearest_nodes(G, destination[1],
destination[0])     route = nx.shortest_path(G, origin_node, destination_node,
weight='length')     total_distance = sum(
ox.utils_graph.get_route_edge_attributes(G, route, 'length')) / 1000  # convert
meters to kilometers
    fig, ax = ox.plot_graph(G, show=False, close=False)
ox.plot_graph_route(G, route, route_color='red', route_linewidth=6, ax=ax,
node_size=0)     plt.close(fig)  # Properly close the plot to avoid memory
issues     return fig, total_distance st.title("Route Planner") origin_name =
st.selectbox("Choose your origin:", all_stops['name'].unique())
destination_name = st.selectbox("Choose your destination:",
all_stops['name'].unique())

if st.button("Find Shortest Route"):
    origin_coords = all_stops[all_stops['name'] == origin_name][['latitude',
'longitude']].iloc[0]     destination_coords =
all_stops[all_stops['name'] ==
destination_name][['latitude', 'longitude']].iloc[0]
```

13

```
    fig, total_distance = get_route((origin_coords['latitude'],
origin_coords['longitude']),
                                    (destination_coords['latitude'],
destination_coords['longitude']))

    st.write(f"Total Distance: {total_distance:.2f} km")
    st.pyplot(fig)
```

Results:


sbl · Streamlit
localhost:8501
Deploy
st.cache is deprecated. Please use one of Streamlit's new caching commands, st.cache_data or st.cache_resource.
More information in our docs.
Route Planner
Choose your origin:
ELECTRIC HOUSE, COLABA DEPOT
Choose your destination:
KURLA DEPOT
Find Shortest Route


sbl · Streamlit
localhost:8501
Route Planner

Conclusion:

By leveraging cutting-edge technologies and algorithms, the developed route optimization application aims to streamline travel logistics and enhance transportation efficiency within Mumbai. It empowers users to make informed decisions, save time, and reduce environmental impact by choosing optimized routes.

References:

1. **Streamlit**:
   - Documentation: Streamlit Documentation
   - GitHub Repository: Streamlit on GitHub
2. **Pandas**:
   - Documentation: Pandas Documentation
   - GitHub Repository: Pandas on GitHub
3. **OSMnx**:
   - Documentation: OSMnx Documentation
   - GitHub Repository: OSMnx on GitHub
4. **NetworkX**:
   - Documentation: NetworkX Documentation
   - GitHub Repository: NetworkX on GitHub
5. **Matplotlib**:
   - Documentation: Matplotlib Documentation
   - GitHub Repository: Matplotlib on GitHub

These references provide comprehensive documentation and resources for understanding and utilizing the libraries effectively.