­­­­­­ Text

Description automatically generated

**Activity based**

**Project Report on**

**Airtificial Intelligence**

**Project Module - III**

**Submitted to Vishwakarma University, Pune**

**Under the Initiative of**

**Contemporary Curriculum, Pedagogy, and Practice (C2P2)**

**By**

**Student Name :Mudra Bhattad**

**SRN No : 202101231**

**Roll No : 23**

**Div : E**

**Third Year Engineering**

**Faculty Incharge:- Richa Agnihotri**

**Date Of Project :- 20/04/2024**

**Department of Computer Engineering**

**Faculty of Science and Technology**

**Academic Year**

**2023-2024 Term-II**

**1. Introduction:**

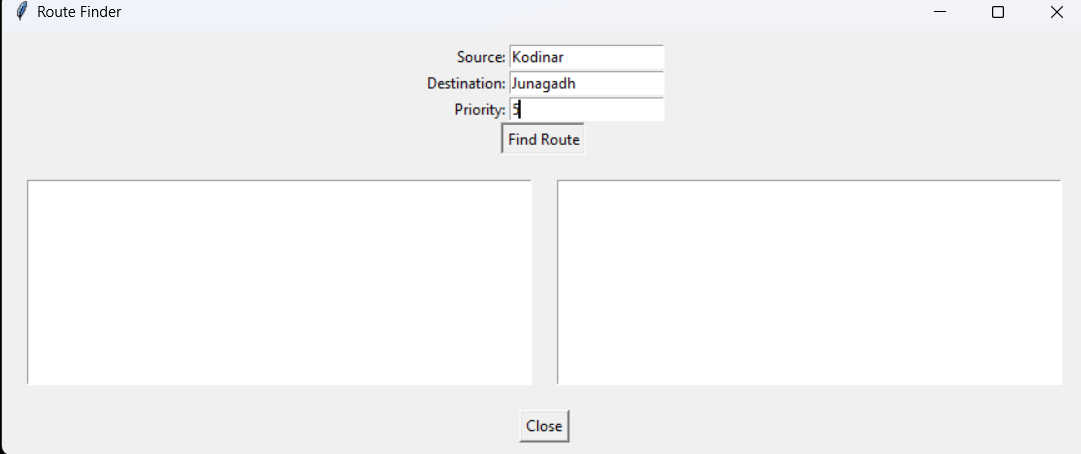
The GUI application is designed to assist users in finding routes between cities using a Best-First Search algorithm. It provides a user-friendly interface for specifying source and destination cities, along with priority criteria, and then presents possible routes and the optimal path based on user-defined priorities.

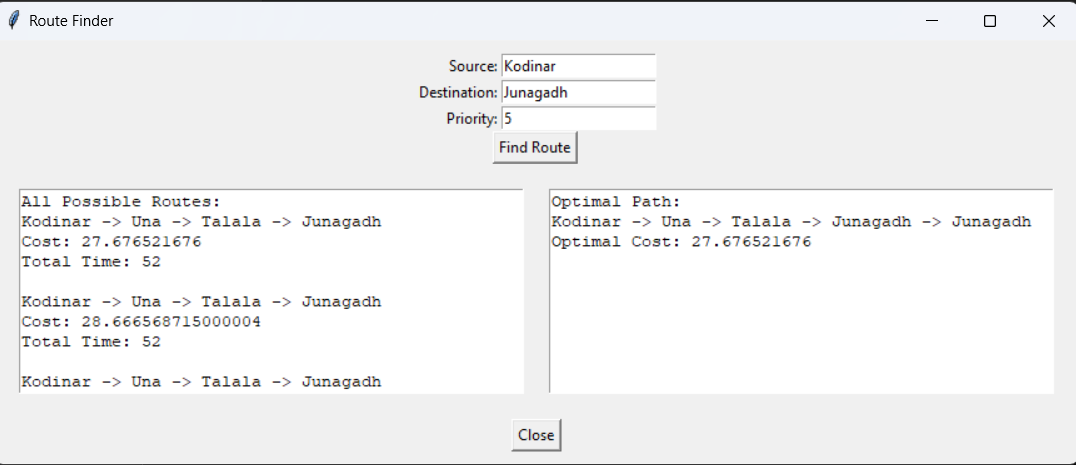
**2. End Users:**

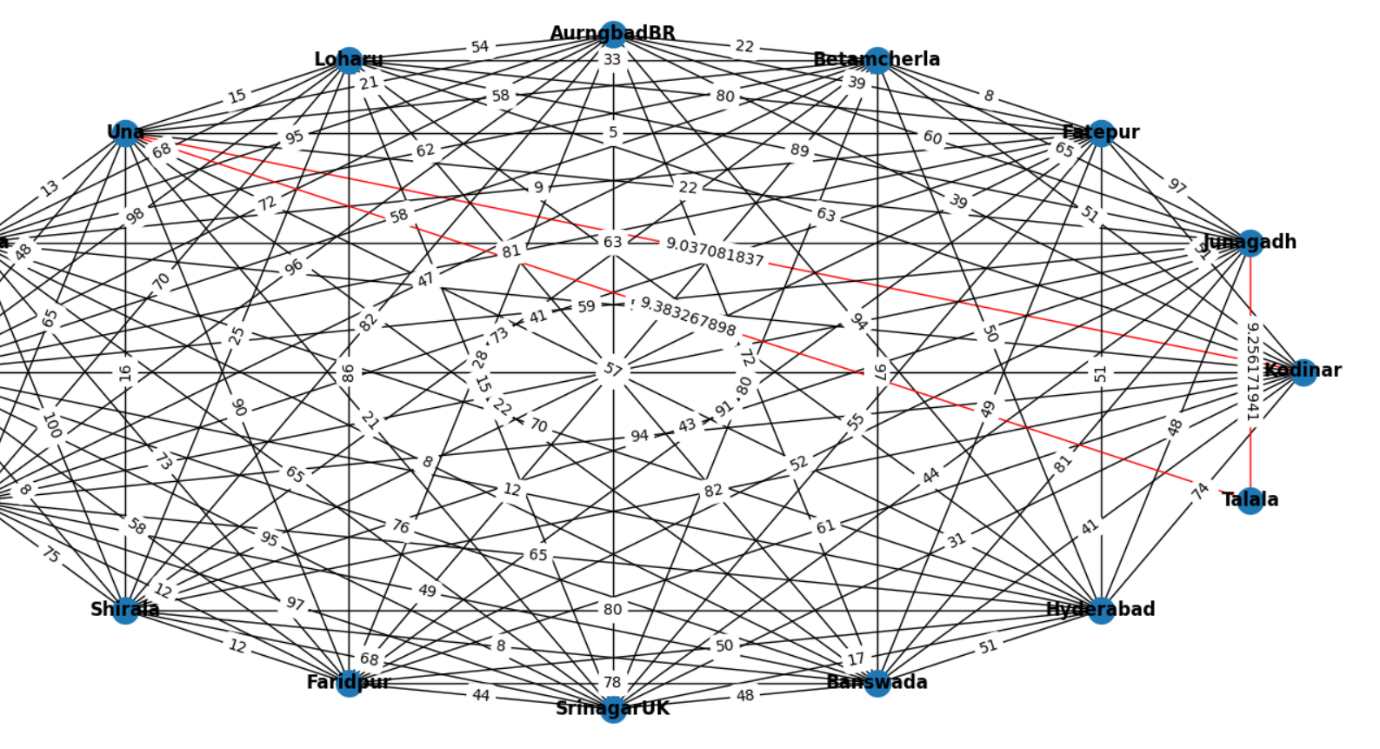
The target audience for the GUI application includes:

* Travelers seeking efficient routes between cities.
* Logistics managers planning transportation routes for goods delivery.
* Tour operators organizing itineraries for group tours.

**3. Wireframe for GUI:**







The wireframe illustrates the layout of the GUI, comprising input fields for source city, destination city, and priority, along with buttons for finding routes and closing the application. Additionally, it shows text areas for displaying possible routes and the optimal path.

**4. Feasibility of GUI:**

The GUI application is feasible for implementation, considering the following factors:

* **Technical Feasibility:** The required technologies (Python, Tkinter, Pandas, NetworkX) are widely supported and suitable for developing the application.
* **Resource Requirements:** The application's resource requirements are minimal, requiring only standard hardware and software components.
* **Time Constraints:** With efficient development practices, the GUI application can be implemented within a reasonable timeframe.
* **Challenges:** Potential challenges include optimizing the algorithm for large datasets and ensuring smooth user interaction in the graphical interface.

**5. Code of GUI:**

* **User Interface:**

import tkinter as tk

from tkinter import messagebox

import networkx as nx

import matplotlib.pyplot as plt

root = tk.Tk()

root.title("Route Finder")

def on\_find\_route():

messagebox.showinfo("Message", "Finding routes...")

def on\_close():

root.destroy()

frame\_input = tk.Frame(root)

frame\_input.pack(padx=10, pady=10)

label\_source = tk.Label(frame\_input, text="Source:")

label\_source.grid(row=0, column=0, sticky="e")

entry\_source = tk.Entry(frame\_input)

entry\_source.grid(row=0, column=1)

label\_destination = tk.Label(frame\_input, text="Destination:")

label\_destination.grid(row=1, column=0, sticky="e")

entry\_destination = tk.Entry(frame\_input)

entry\_destination.grid(row=1, column=1)

label\_priority = tk.Label(frame\_input, text="Priority:")

label\_priority.grid(row=2, column=0, sticky="e")

entry\_priority = tk.Entry(frame\_input)

entry\_priority.grid(row=2, column=1)

button\_find\_route = tk.Button(frame\_input, text="Find Route", command=on\_find\_route)

button\_find\_route.grid(row=3, columnspan=2)

button\_close = tk.Button(root, text="Close", command=on\_close)

button\_close.pack(pady=10)

root.mainloop()

* **Error Handling:**

def on\_find\_route():

# Validate user inputs

source = entry\_source.get()

destination = entry\_destination.get()

priority = entry\_priority.get()

if not source or not destination or not priority:

messagebox.showerror("Error", "Please enter source, destination, and priority.")

return

try:

priority = int(priority)

except ValueError:

messagebox.showerror("Error", "Priority must be a valid integer.")

return

find\_routes(source, destination, priority)

def find\_routes(source, destination, priority):

# Placeholder function for finding routes

messagebox.showinfo("Message", f"Finding routes from {source} to {destination} with priority {priority}...")

root = tk.Tk()

root.title("Route Finder")

root.mainloop()

**6.Analysis of output with comparison of attributes**

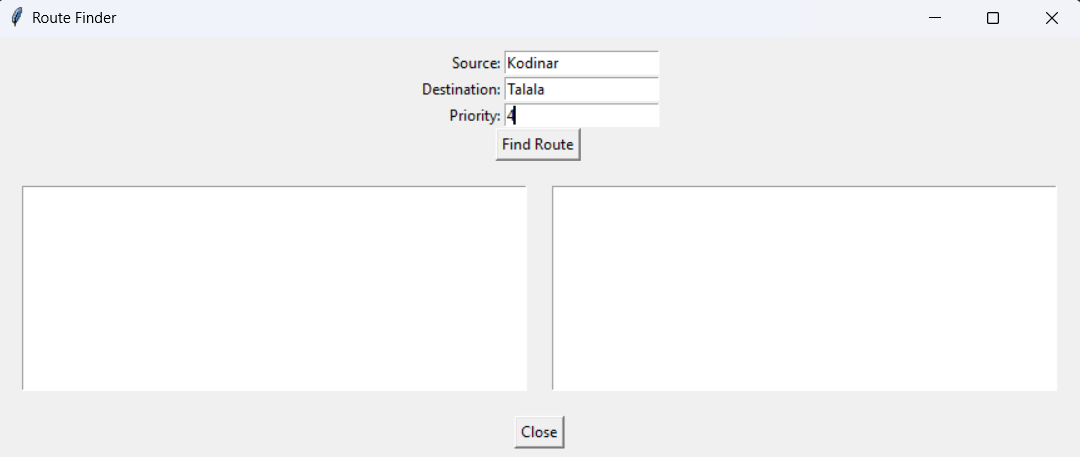
To analyze the output of the route finding process and compare the attributes of different routes, we can consider various factors such as cost, time, and priority. Here's how you can analyze the output and compare the attributes:

1. **Cost Analysis:**
   * Calculate the total cost of each route, considering factors such as distance, time, and priority score.
   * Compare the total costs of different routes to identify the most cost-effective option.
   * Visualize the cost differences between routes using graphs or tables.
2. **Time Analysis:**
   * Calculate the total time taken for each route, considering the time required to travel between cities.
   * Compare the total times of different routes to identify the fastest option.
   * Visualize the time differences between routes using graphs or tables.
3. **Priority Analysis:**
   * Consider the priority score assigned to each route and its impact on the selection process.
   * Analyze how the priority score affects the chosen routes and their attributes.
   * Compare routes with different priority scores to understand their implications on the overall route selection.
4. **Optimality Analysis:**
   * Evaluate the optimality of the selected route based on predefined criteria (e.g., lowest cost, shortest time).
   * Compare the selected route with alternative routes to determine if it meets the desired optimization goals.
5. **Visualization:**
   * Visualize the routes and their attributes using graphs or maps to provide a clear understanding of the differences.
   * Use color-coding or other visual cues to highlight important attributes such as cost, time, and priority.
6. **Comparison:**
   * Compare the attributes of different routes side by side, highlighting their strengths and weaknesses.
   * Consider trade-offs between cost, time, and priority to make informed decisions about route selection.

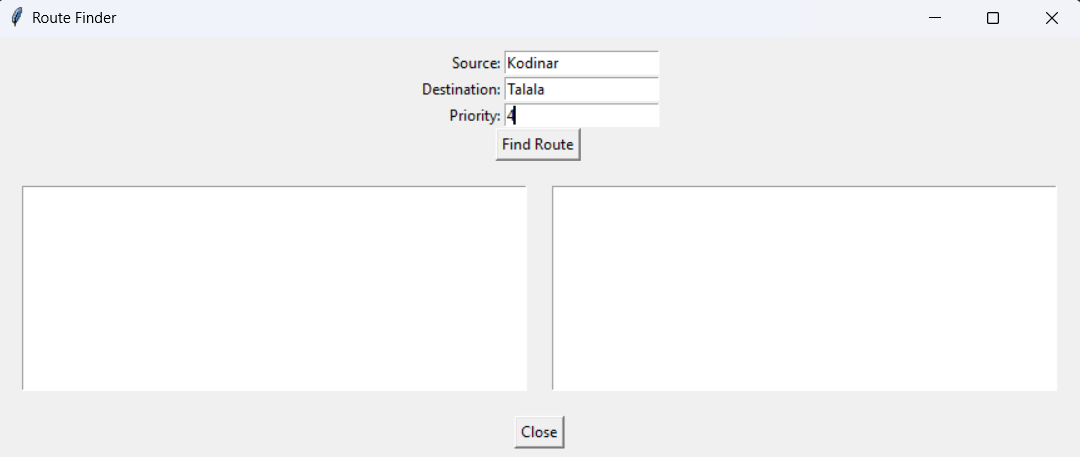
By performing a thorough analysis of the output and comparing the attributes of different routes, you can make informed decisions about route selection and optimize the route finding process to meet user needs and preferences.

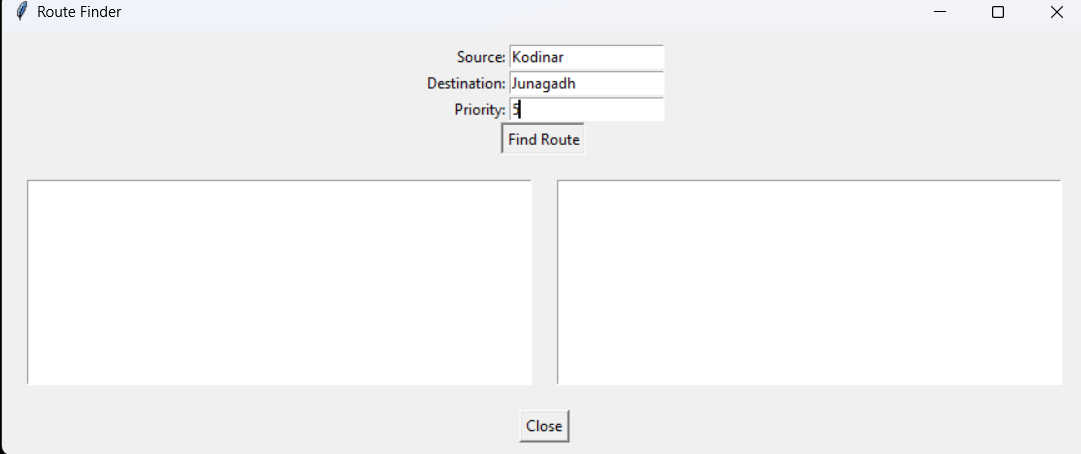
**7.Different Features**

* **Input Fields**

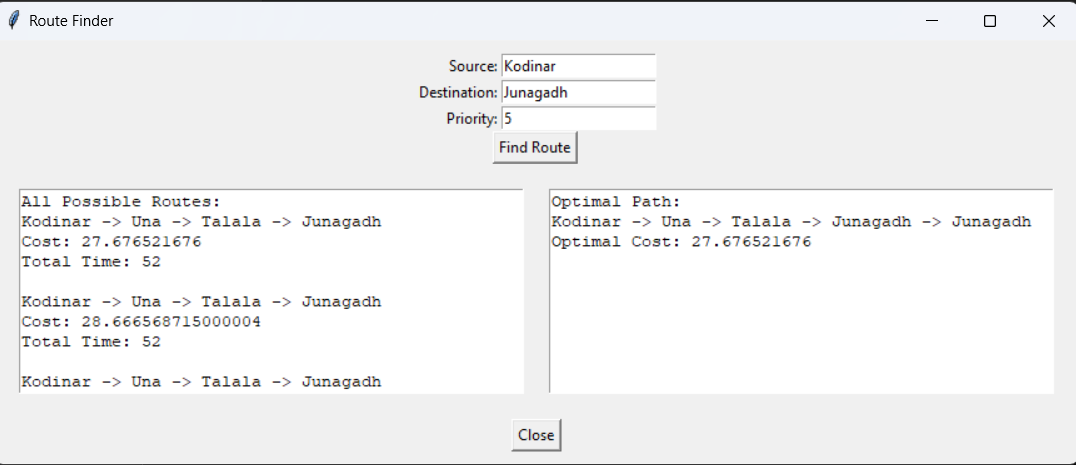
****

* **Buttons:**

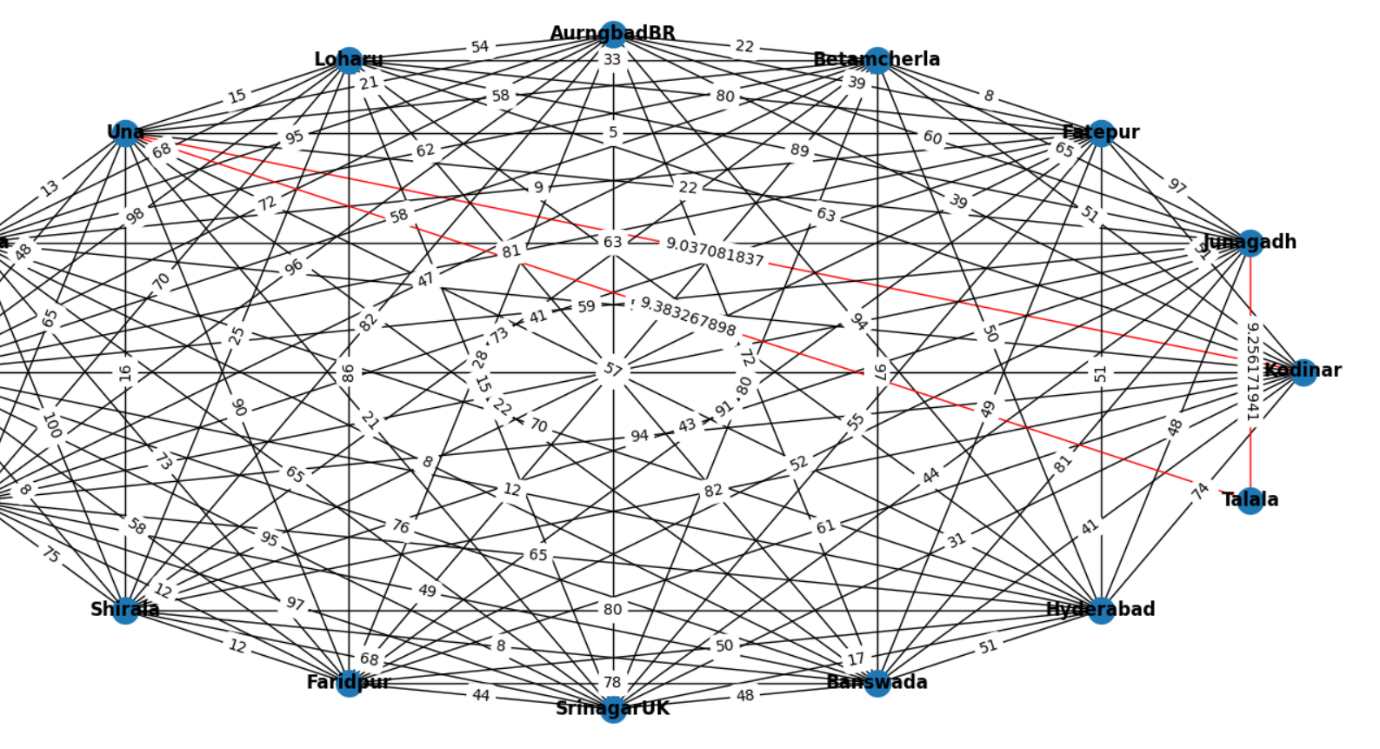
****

****

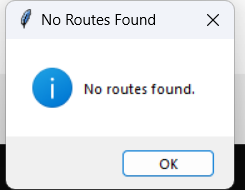
* **Text Areas:**

****

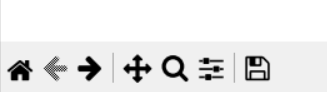
* **Graph Visualization:**

****

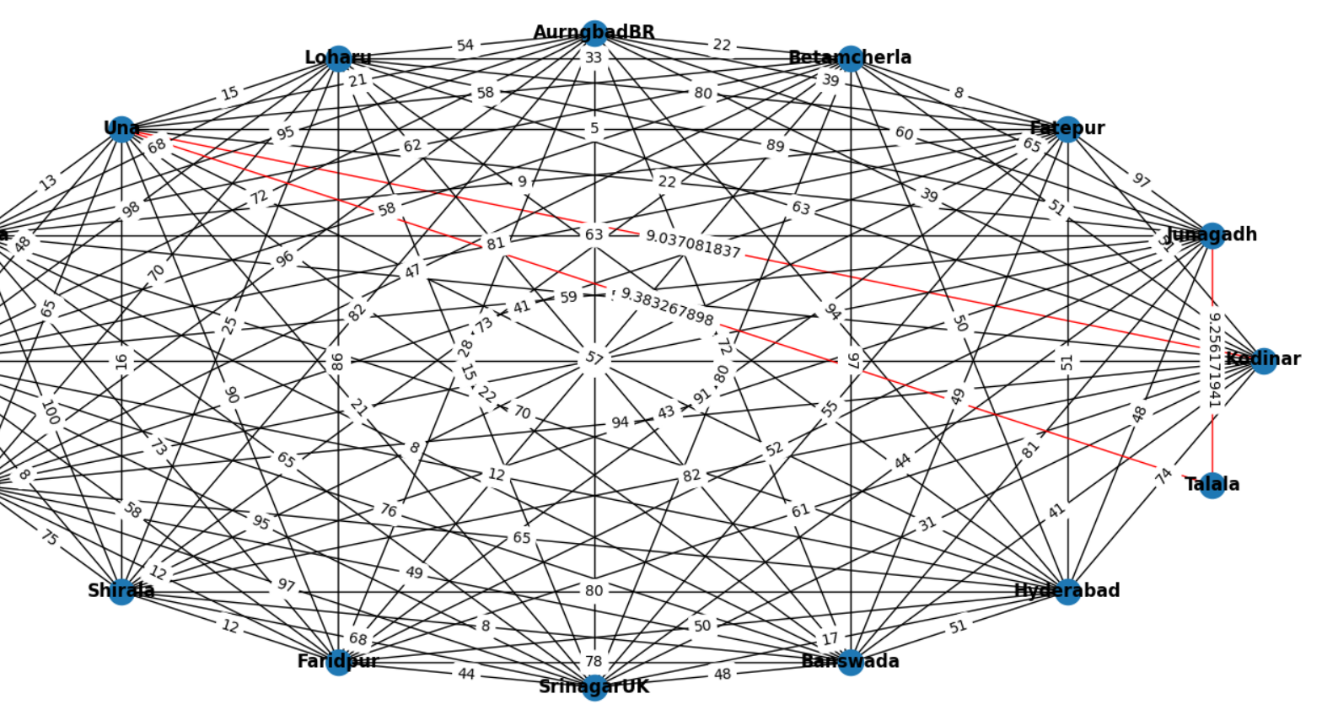
* **Error Messages:**

****

* **Zooming, Resetting, Navigation Controls,Panning :**

****

* **Highlighting:**

****

**8.Conclusion**

In summary, the GUI application for route finding offers users an intuitive interface to input source and destination cities, along with priority criteria, and obtain routes based on the Best-First Search algorithm. With robust error handling and interactive graph visualization, users can easily explore route options and compare attributes such as cost, time, and priority. The application streamlines the route finding process, catering to the needs of travelers, logistics managers, and tour operators. Future enhancements could include real-time updates and additional route planning features to further enhance user experience and decision-making capabilities.