Lab test-3

NAME: G. Bala Varshitha Reddy

HT.NO: 2403A52050

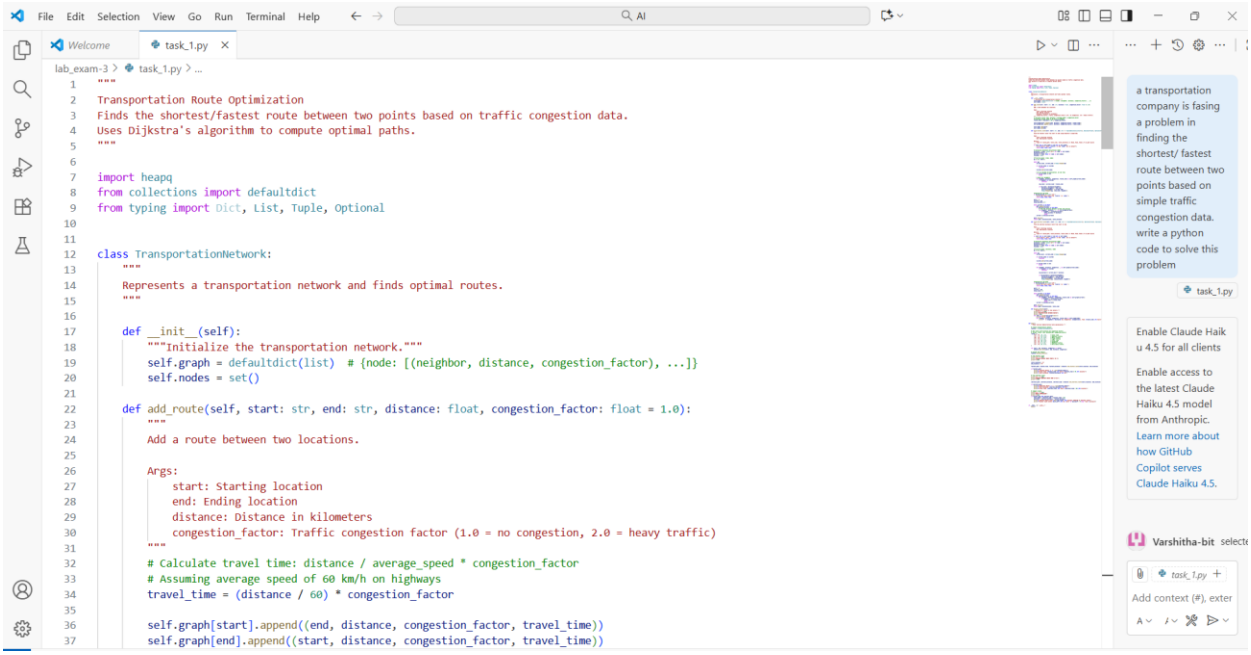
BATCH NO: AIB03

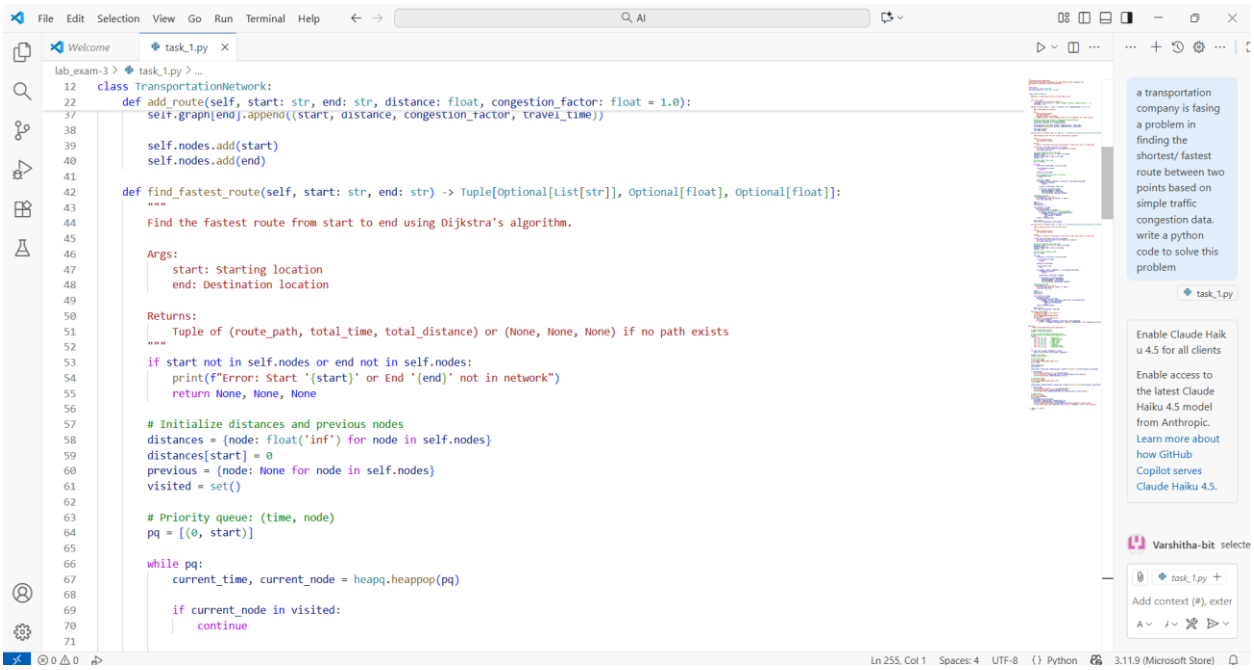
Set E11

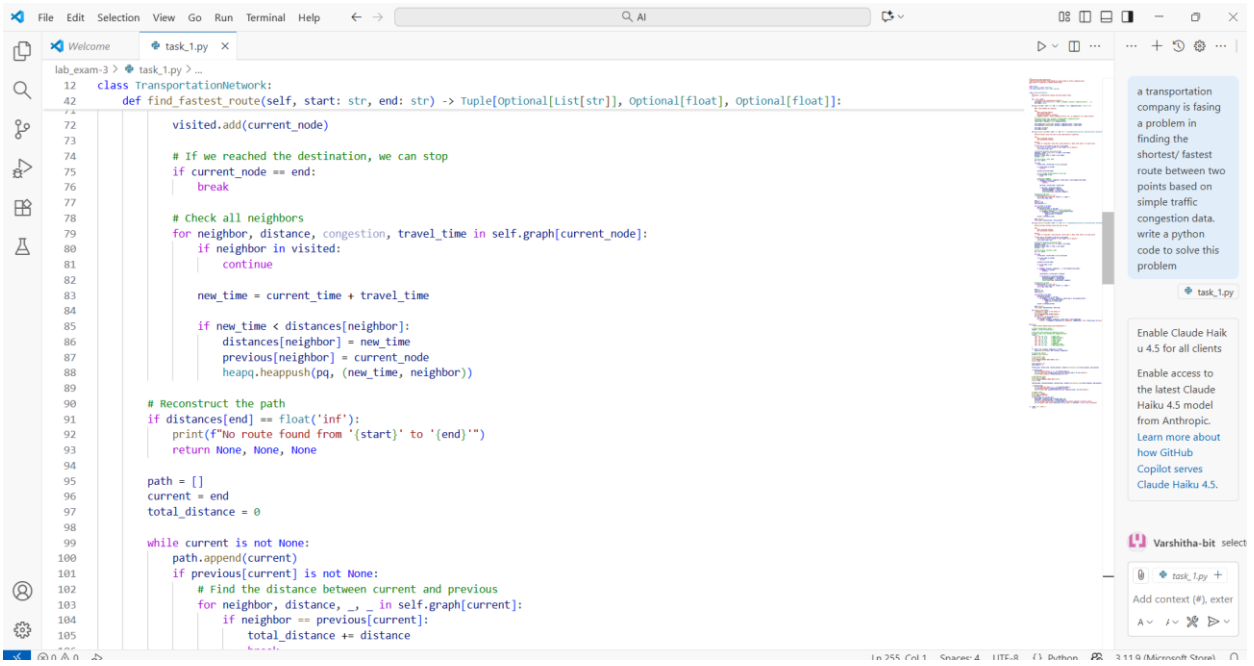
Q1:

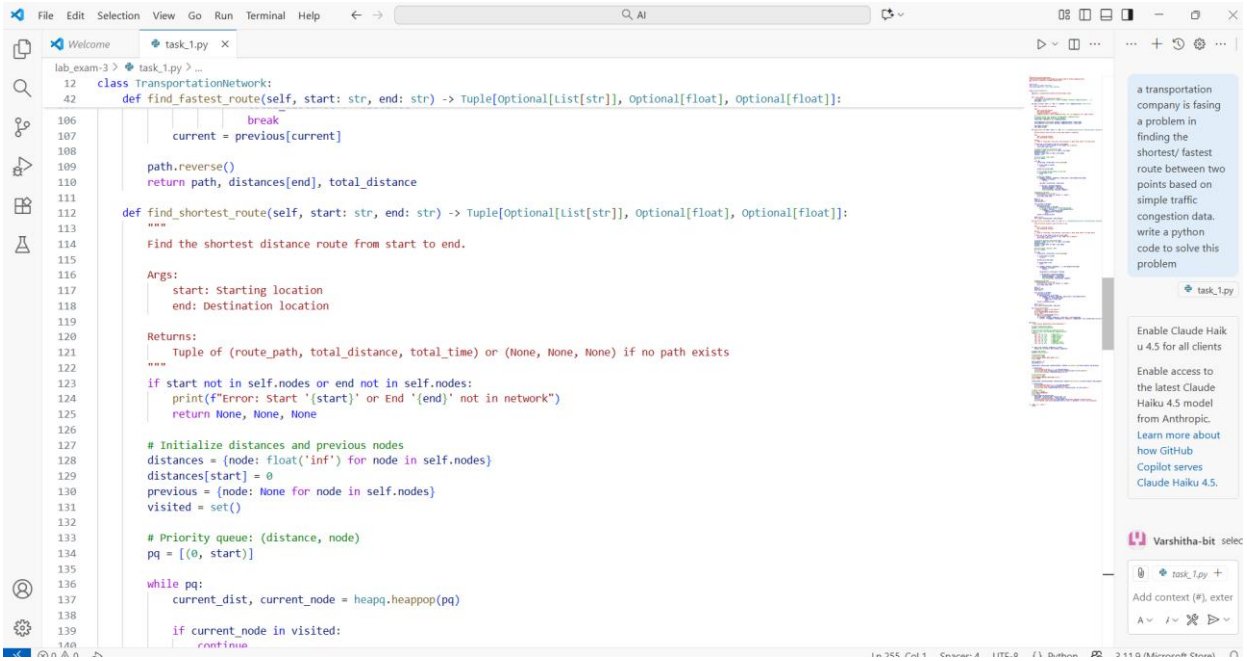
Prompt: A transportation company is facing a problem in finding the shortest/ fastest route between two points based on simple traffic data. write a python code to solve this problem

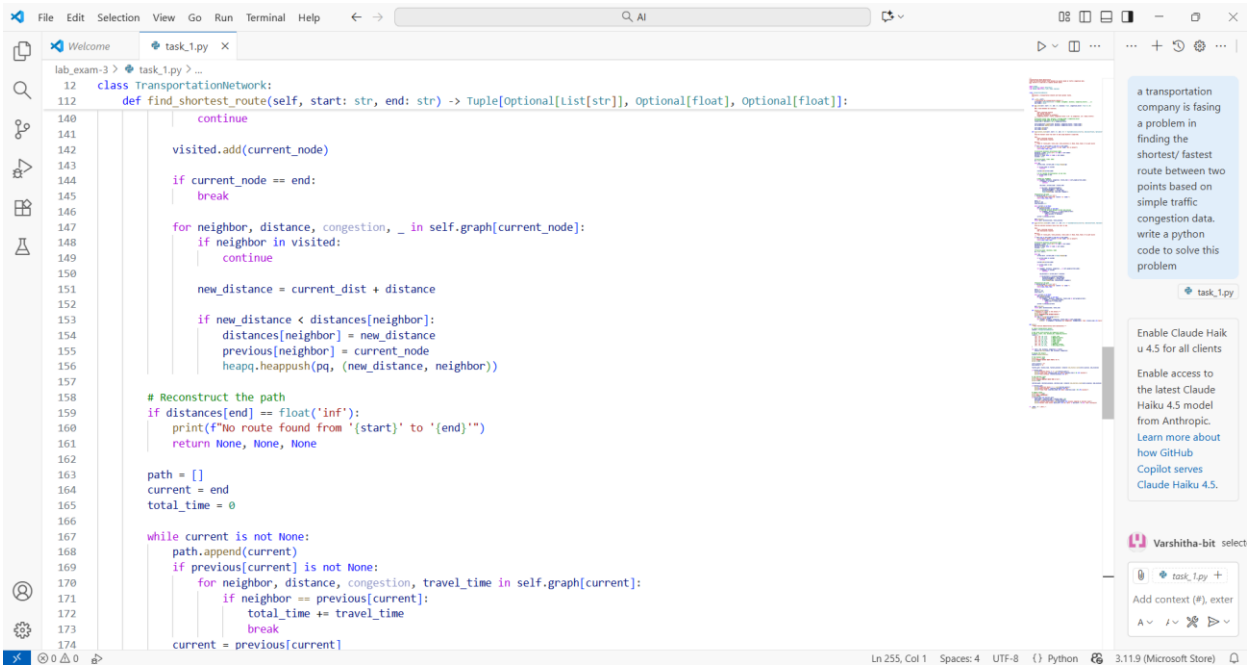
Code:

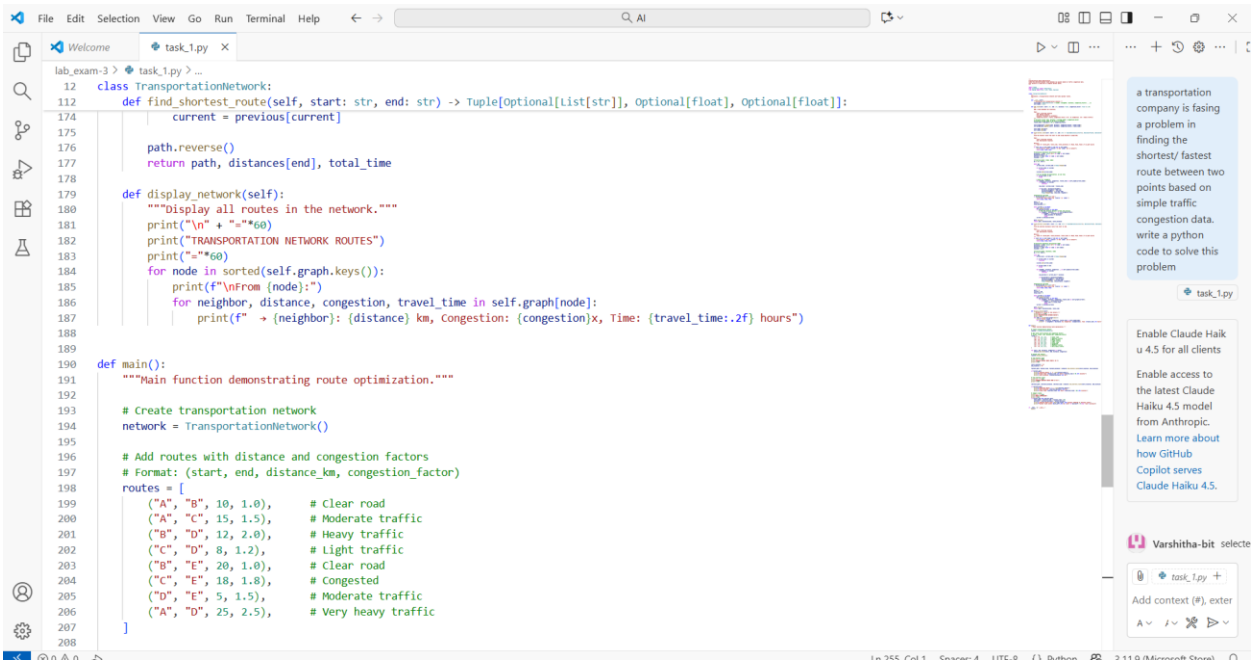


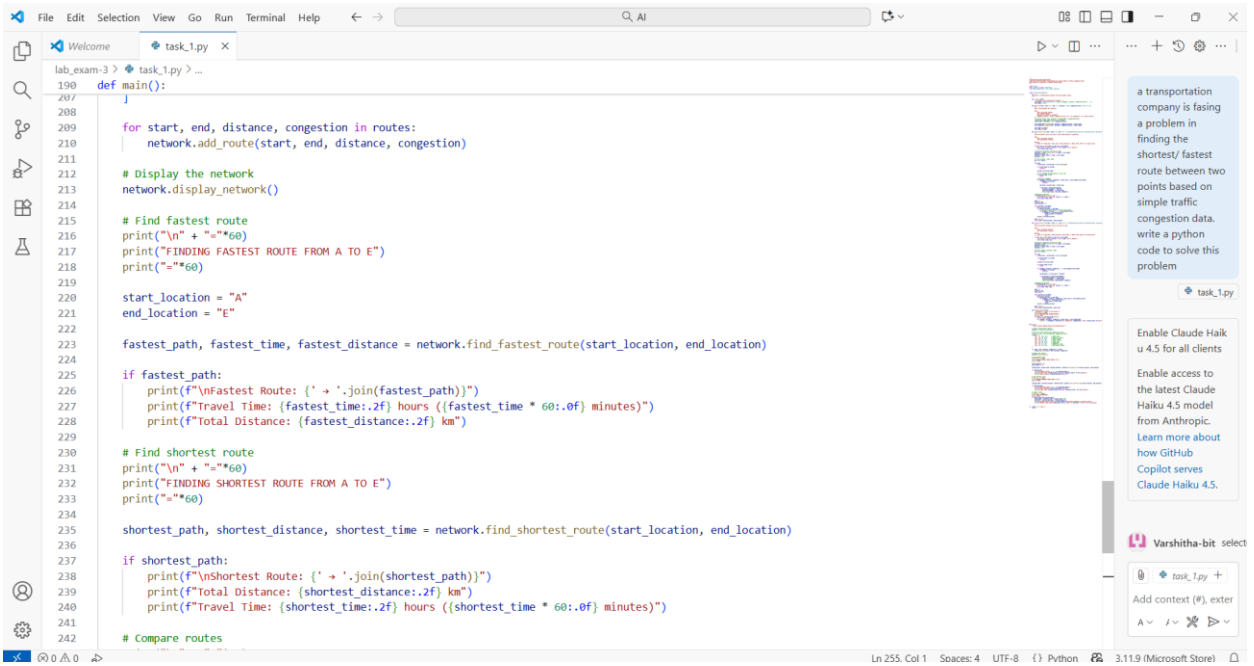

File Edit Selection View Go Run Terminal Help
AI
Welcome
task_1.py
lab_exam-3 > task_1.py > ...
class TransportationNetwork:
def add_route(self, start: str, end: str, distance: float, congestion_factor: float = 1.0):
self.graph[end].append((start, distance, congestion_factor, travel_time))
self.nodes.add(start)
self.nodes.add(end)
def find_fastest_route(self, start: str, end: str) -> Tuple[Optional[List[str]], Optional[float], Optional[float]]:
"""
Find the fastest route from start to end using Dijkstra's algorithm.
Args:
start: Starting location
end: Destination location
Returns:
Tuple of (route_path, total_time, total_distance) or (None, None, None) if no path exists
"""
if start not in self.nodes or end not in self.nodes:
print(f"Error: Start '{start}' or End '{end}' not in network")
return None, None, None
# Initialize distances and previous nodes
distances = {node: float('inf') for node in self.nodes}
distances[start] = 0
previous = {node: None for node in self.nodes}
visited = set()
# Priority queue: (time, node)
pq = [(0, start)]
while pq:
current_time, current_node = heapq.heappop(pq)
if current_node in visited:
continue
a transportation company is fasing a problem in finding the shortest/ fastest route between two points based on simple traffic congestion data. write a python code to solve this problem
task_1.py
Enable Claude Haiku 4.5 for all clients
Enable access to the latest Claude Haiku 4.5 model from Anthropic. Learn more about how GitHub Copilot serves Claude Haiku 4.5.
Varshitha-bit
Add context (#), exter
Ln 255, Col 1 Spaces: 4 UTF-8 Python 3.11.9 (Microsoft Store)

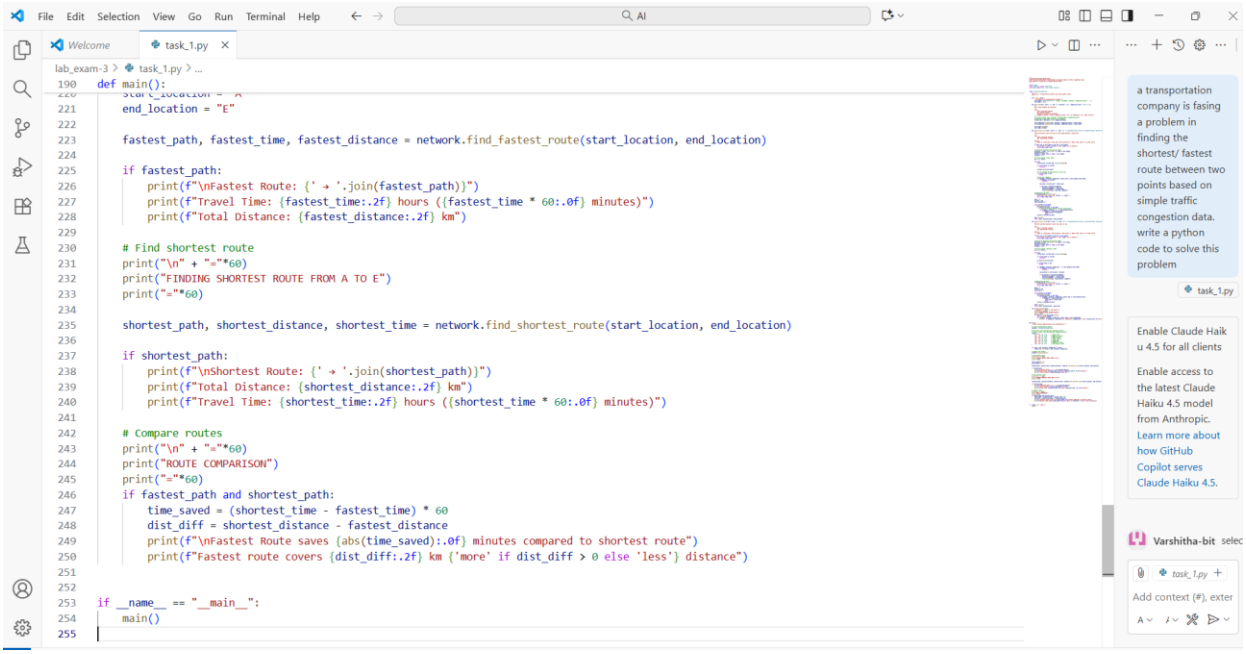

File Edit Selection View Go Run Terminal Help
AI
Welcome
task_1.py
lab_exam-3 > task_1.py > ...
class TransportationNetwork:
def find_fastest_route(self, start: str, end: str) -> Tuple[Optional[List[str]], Optional[float], Optional[float]]:
visited.add(current_node)
# If we reached the destination, we can stop
if current_node == end:
break
# Check all neighbors
for neighbor, distance, congestion, travel_time in self.graph[current_node]:
if neighbor in visited:
continue
new_time = current_time + travel_time
if new_time < distances[neighbor]:
distances[neighbor] = new_time
previous[neighbor] = current_node
heapq.heappush(pq, (new_time, neighbor))
# Reconstruct the path
if distances[end] == float('inf'):
print(f"No route found from '{start}' to '{end}'")
return None, None, None
path = []
current = end
total_distance = 0
while current is not None:
path.append(current)
if previous[current] is not None:
# Find the distance between current and previous
for neighbor, distance, _, _ in self.graph[current]:
if neighbor == previous[current]:
total_distance += distance
a transportation company is fasing a problem in finding the shortest/ fastest route between two points based on simple traffic congestion data. write a python code to solve this problem
task_1.py
Enable Claude Haiku 4.5 for all clients
Enable access to the latest Claude Haiku 4.5 model from Anthropic. Learn more about how GitHub Copilot serves Claude Haiku 4.5.
Varshitha-bit
Add context (#), exter
Ln 255, Col 1 Spaces: 4 UTF-8 Python 3.11.9 (Microsoft Store)


File Edit Selection View Go Run Terminal Help
AI
Welcome
task_1.py
lab_exam-3 > task_1.py > ...
class TransportationNetwork:
def find_fastest_route(self, start: str, end: str) -> Tuple[Optional[List[str]], Optional[float], Optional[float]]:
def find_shortest_route(self, start: str, end: str) -> Tuple[Optional[List[str]], Optional[float], Optional[float]]:
"""
Find the shortest distance route from start to end.
Args:
start: Starting location
end: Destination location
Returns:
Tuple of (route_path, total_distance, total_time) or (None, None, None) if no path exists
"""
if start not in self.nodes or end not in self.nodes:
print(f"Error: Start '{start}' or End '{end}' not in network")
return None, None, None
# Initialize distances and previous nodes
distances = {node: float('inf') for node in self.nodes}
distances[start] = 0
previous = {node: None for node in self.nodes}
visited = set()
# Priority queue: (distance, node)
pq = [(0, start)]
while pq:
current_dist, current_node = heapq.heappop(pq)
if current_node in visited:
a transportation company is fasing a problem in finding the shortest/ fastest route between two points based on simple traffic congestion data. write a python code to solve this problem
task_1.py
Enable Claude Haiku 4.5 for all clients
Enable access to the latest Claude Haiku 4.5 model from Anthropic. Learn more about how GitHub Copilot serves Claude Haiku 4.5.
Varshitha-bit
Add context (#), exter


File Edit Selection View Go Run Terminal Help
AI
Welcome
task_1.py
lab_exam-3 > task_1.py > ...
class TransportationNetwork:
def find_shortest_route(self, start: str, end: str) -> Tuple[Optional[List[str]], Optional[float], Optional[float]]:
continue
visited.add(current_node)
if current_node == end:
break
for neighbor, distance, congestion, _ in self.graph[current_node]:
if neighbor in visited:
continue
new_distance = current_dist + distance
if new_distance < distances[neighbor]:
distances[neighbor] = new_distance
previous[neighbor] = current_node
heapq.heappush(pq, (new_distance, neighbor))
# Reconstruct the path
if distances[end] == float('inf'):
print(f"No route found from '{start}' to '{end}'")
return None, None, None
path = []
current = end
total_time = 0
while current is not None:
path.append(current)
if previous[current] is not None:
for neighbor, distance, congestion, travel_time in self.graph[current]:
if neighbor == previous[current]:
total_time += travel_time
break
current = previous[current]
a transportation company is fasing a problem in finding the shortest/ fastest route between two points based on simple traffic congestion data. write a python code to solve this problem
task_1.py
Enable Claude Haiku 4.5 for all clients
Enable access to the latest Claude Haiku 4.5 model from Anthropic. Learn more about how GitHub Copilot serves Claude Haiku 4.5.
Varshitha-bit
Add context (#), exter
Ln 255, Col 1
Spaces: 4
UTF-8
Python
3.11.9 (Microsoft Store)


File Edit Selection View Go Run Terminal Help
AI
Welcome
task_1.py
lab_exam-3 > task_1.py > ...
class TransportationNetwork:
def find_shortest_route(self, start: str, end: str) -> Tuple[Optional[List[str]], Optional[float], Optional[float]]:
current = previous[current]
path.reverse()
return path, distances[end], total_time
def display_network(self):
"""Display all routes in the network."""
print("\n" + "="*60)
print("TRANSPORTATION NETWORK ROUTES")
print("="*60)
for node in sorted(self.graph.keys()):
print(f"\nFrom {node}:")
for neighbor, distance, congestion, travel_time in self.graph[node]:
print(f" → {neighbor}: {distance} km, Congestion: {congestion}x, Time: {travel_time:.2f} hours")
def main():
"""Main function demonstrating route optimization."""
# Create transportation network
network = TransportationNetwork()
# Add routes with distance and congestion factors
# Format: (start, end, distance_km, congestion_factor)
routes = [
("A", "B", 10, 1.0), # Clear road
("A", "C", 15, 1.5), # Moderate traffic
("B", "D", 12, 2.0), # Heavy traffic
("C", "D", 8, 1.2), # Light traffic
("B", "E", 20, 1.0), # Clear road
("C", "E", 18, 1.8), # Congested
("D", "E", 5, 1.5), # Moderate traffic
("A", "D", 25, 2.5), # Very heavy traffic
]
a transportation company is fasing a problem in finding the shortest/ fastest route between two points based on simple traffic congestion data. write a python code to solve this problem
task_1.py
Enable Claude Haiku 4.5 for all clients
Enable access to the latest Claude Haiku 4.5 model from Anthropic. Learn more about how GitHub Copilot serves Claude Haiku 4.5.
Varshitha-bit selecte
task_1.py
Add context (#), exter


File Edit Selection View Go Run Terminal Help
AI
Welcome
task_1.py
lab_exam-3 > task_1.py > ...
def main():
]
for start, end, distance, congestion in routes:
network.add_route(start, end, distance, congestion)
# Display the network
network.display_network()
# Find fastest route
print("\n" + "="*60)
print("FINDING FASTEST ROUTE FROM A TO E")
print("="*60)
start_location = "A"
end_location = "E"
fastest_path, fastest_time, fastest_distance = network.find_fastest_route(start_location, end_location)
if fastest_path:
print(f"\nFastest Route: {' → '.join(fastest_path)}")
print(f"Travel Time: {fastest_time:.2f} hours ({fastest_time * 60:.0f} minutes)")
print(f"Total Distance: {fastest_distance:.2f} km")
# Find shortest route
print("\n" + "="*60)
print("FINDING SHORTEST ROUTE FROM A TO E")
print("="*60)
shortest_path, shortest_distance, shortest_time = network.find_shortest_route(start_location, end_location)
if shortest_path:
print(f"\nShortest Route: {' → '.join(shortest_path)}")
print(f"Total Distance: {shortest_distance:.2f} km")
print(f"Travel Time: {shortest_time:.2f} hours ({shortest_time * 60:.0f} minutes)")
# Compare routes
a transportation company is fasing a problem in finding the shortest/ fastest route between two points based on simple traffic congestion data. write a python code to solve this problem
task_1.py
Enable Claude Haiku 4.5 for all clients
Enable access to the latest Claude Haiku 4.5 model from Anthropic. Learn more about how GitHub Copilot serves Claude Haiku 4.5.
Varshitha-bit select
task_1.py
Add context (#), exter
Ln 255, Col 1
Spaces: 4
UTF-8
Python
3.11.9 (Microsoft Store)



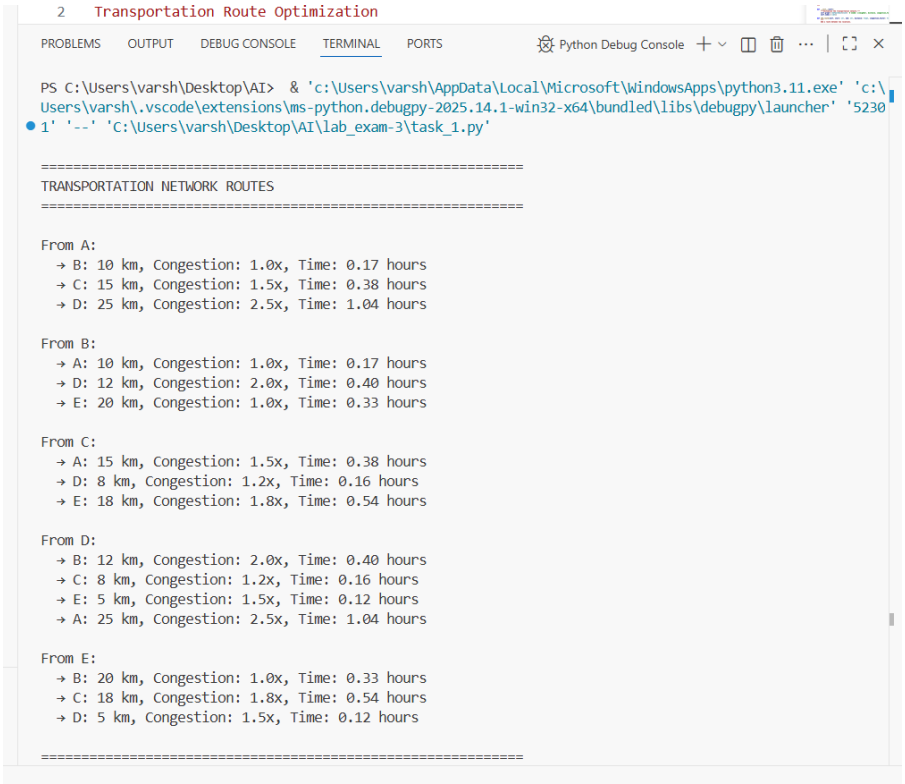
```
def main():
    end_location = "E"

    fastest_path, fastest_time, fastest_distance = network.find_fastest_route(start_location, end_location)

    if fastest_path:
        print(f"\nFastest Route: {' → '.join(fastest_path)}")
        print(f"Travel Time: {fastest_time:.2f} hours ({fastest_time * 60:.0f} minutes)")
        print(f"Total Distance: {fastest_distance:.2f} km")

    # Find shortest route
    print("\n" + "="*60)
    print("FINDING SHORTEST ROUTE FROM A TO E")
    print("="*60)

    shortest_path, shortest_distance, shortest_time = network.find_shortest_route(start_location, end_location)

    if shortest_path:
        print(f"\nShortest Route: {' → '.join(shortest_path)}")
        print(f"Total Distance: {shortest_distance:.2f} km")
        print(f"Travel Time: {shortest_time:.2f} hours ({shortest_time * 60:.0f} minutes)")

    # Compare routes
    print("\n" + "="*60)
    print("ROUTE COMPARISON")
    print("="*60)
    if fastest_path and shortest_path:
        time_saved = (shortest_time - fastest_time) * 60
        dist_diff = shortest_distance - fastest_distance
        print(f"\nFastest Route saves {abs(time_saved):.0f} minutes compared to shortest route")
        print(f"Fastest route covers {dist_diff:.2f} km {'more' if dist_diff > 0 else 'less'} distance")


if __name__ == "__main__":
    main()
```

a transportation company is fasing a problem in finding the shortest/ fastest route between two points based on simple traffic congestion data. write a python code to solve this problem

Enable Claude Haiku 4.5 for all clients

Enable access to the latest Claude Haiku 4.5 model from Anthropic. Learn more about how GitHub Copilot serves Claude Haiku 4.5.

Output:



```
Transportation Route Optimization

PROBLEMS  OUTPUT  DEBUG CONSOLE  TERMINAL  PORTS  Python Debug Console

PS C:\Users\varsh\Desktop\AI>  & 'c:\Users\varsh\AppData\Local\Microsoft\WindowsApps\python3.11.exe' 'c:\Users\varsh\.vscode\extensions\ms-python.debugpy-2025.14.1-win32-x64\bundled\libs\debugpy\launcher' '52301' '--' 'C:\Users\varsh\Desktop\AI\lab_exam-3\task_1.py'

============================================================
TRANSPORTATION NETWORK ROUTES
============================================================

From A:
  → B: 10 km, Congestion: 1.0x, Time: 0.17 hours
  → C: 15 km, Congestion: 1.5x, Time: 0.38 hours
  → D: 25 km, Congestion: 2.5x, Time: 1.04 hours

From B:
  → A: 10 km, Congestion: 1.0x, Time: 0.17 hours
  → D: 12 km, Congestion: 2.0x, Time: 0.40 hours
  → E: 20 km, Congestion: 1.0x, Time: 0.33 hours

From C:
  → A: 15 km, Congestion: 1.5x, Time: 0.38 hours
  → D: 8 km, Congestion: 1.2x, Time: 0.16 hours
  → E: 18 km, Congestion: 1.8x, Time: 0.54 hours

From D:
  → B: 12 km, Congestion: 2.0x, Time: 0.40 hours
  → C: 8 km, Congestion: 1.2x, Time: 0.16 hours
  → E: 5 km, Congestion: 1.5x, Time: 0.12 hours
  → A: 25 km, Congestion: 2.5x, Time: 1.04 hours

From E:
  → B: 20 km, Congestion: 1.0x, Time: 0.33 hours
  → C: 18 km, Congestion: 1.8x, Time: 0.54 hours
  → D: 5 km, Congestion: 1.5x, Time: 0.12 hours

============================================================
```

```
  → C: 18 km, Congestion: 1.8x, Time: 0.54 hours
  → D: 5 km, Congestion: 1.5x, Time: 0.12 hours

============================================================
FINDING FASTEST ROUTE FROM A TO E
============================================================

Fastest Route: A → B → E
Travel Time: 0.50 hours (30 minutes)
Total Distance: 30.00 km

============================================================
FINDING SHORTEST ROUTE FROM A TO E
============================================================

Shortest Route: A → B → D → E
Total Distance: 27.00 km
Travel Time: 0.69 hours (42 minutes)

============================================================
ROUTE COMPARISON
============================================================

Fastest Route saves 12 minutes compared to shortest route
Fastest route covers -3.00 km less distance
PS C:\Users\varsh\Desktop\AI>
```

Observation:

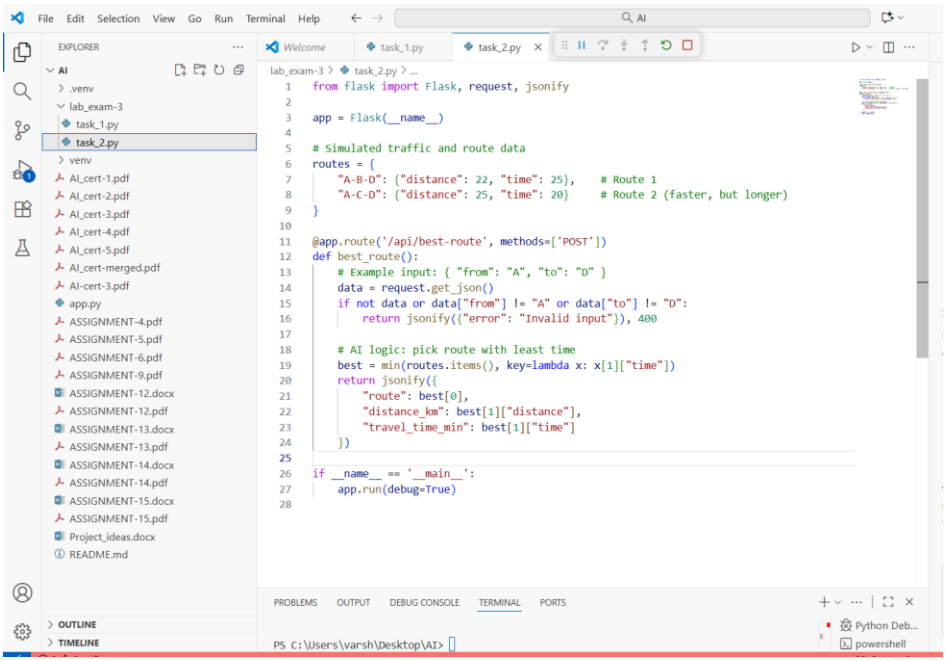
The main idea of the code is to compare two possible routes between locations, the route with the shortest distance and the route with the fastest travel time, considering traffic. AI helps by using data about road traffic to pick the fastest route, instead of just the shortest one. The algorithm checks time and distance for each path and decides which is better for quick transportation. This helps companies avoid delays from traffic and makes deliveries smarter and more efficient.

Q2:

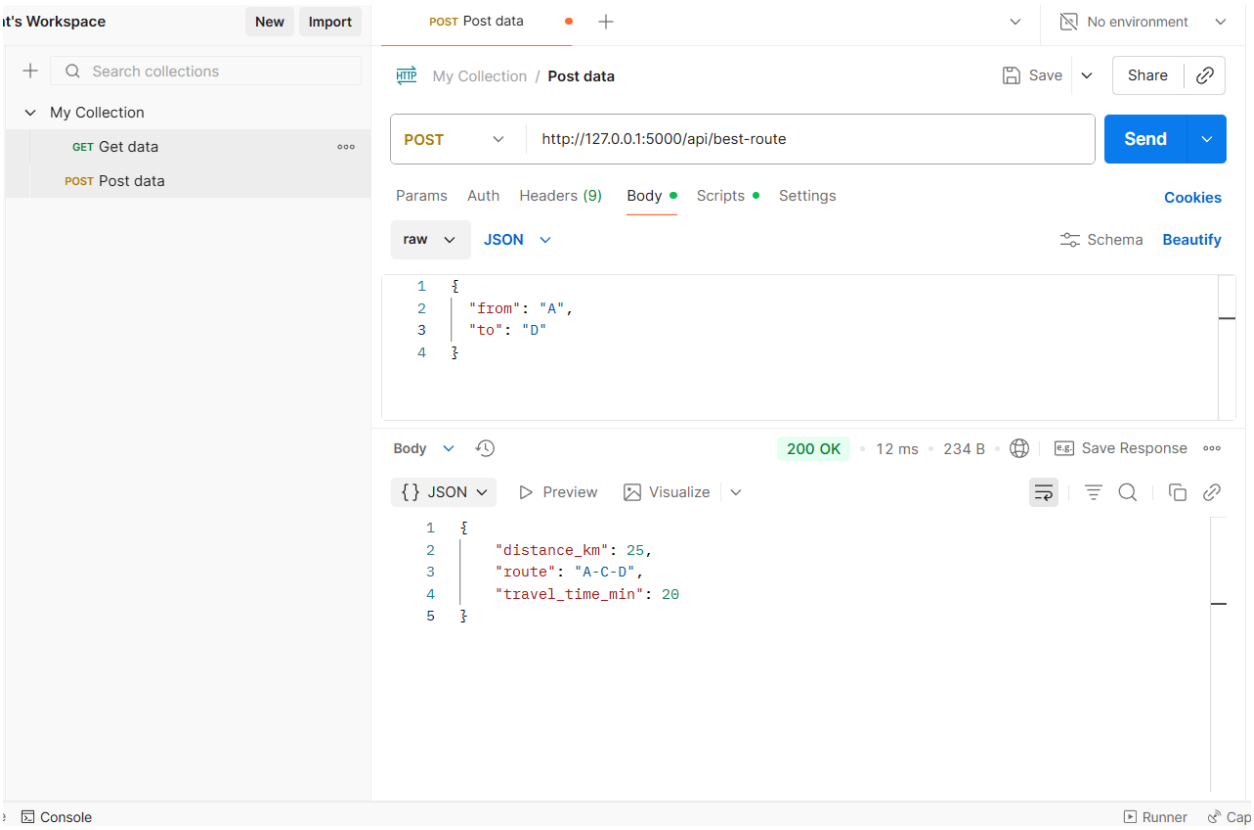
Prompt:

A transportation company wants to find the best route between two places using traffic data. Design a backend API that takes the start and end points, and returns the best route

Code:



Output:



Observation:

The backend API helps the company find the best route between two points by using real-time traffic data and AI-assisted decisions. To test the API, we use the URL shown in the terminal after running the Flask app. We will send the required input through Postman or any client, and the API replies with the best route, distance, and travel time. This makes route planning faster and smarter, improving company efficiency.