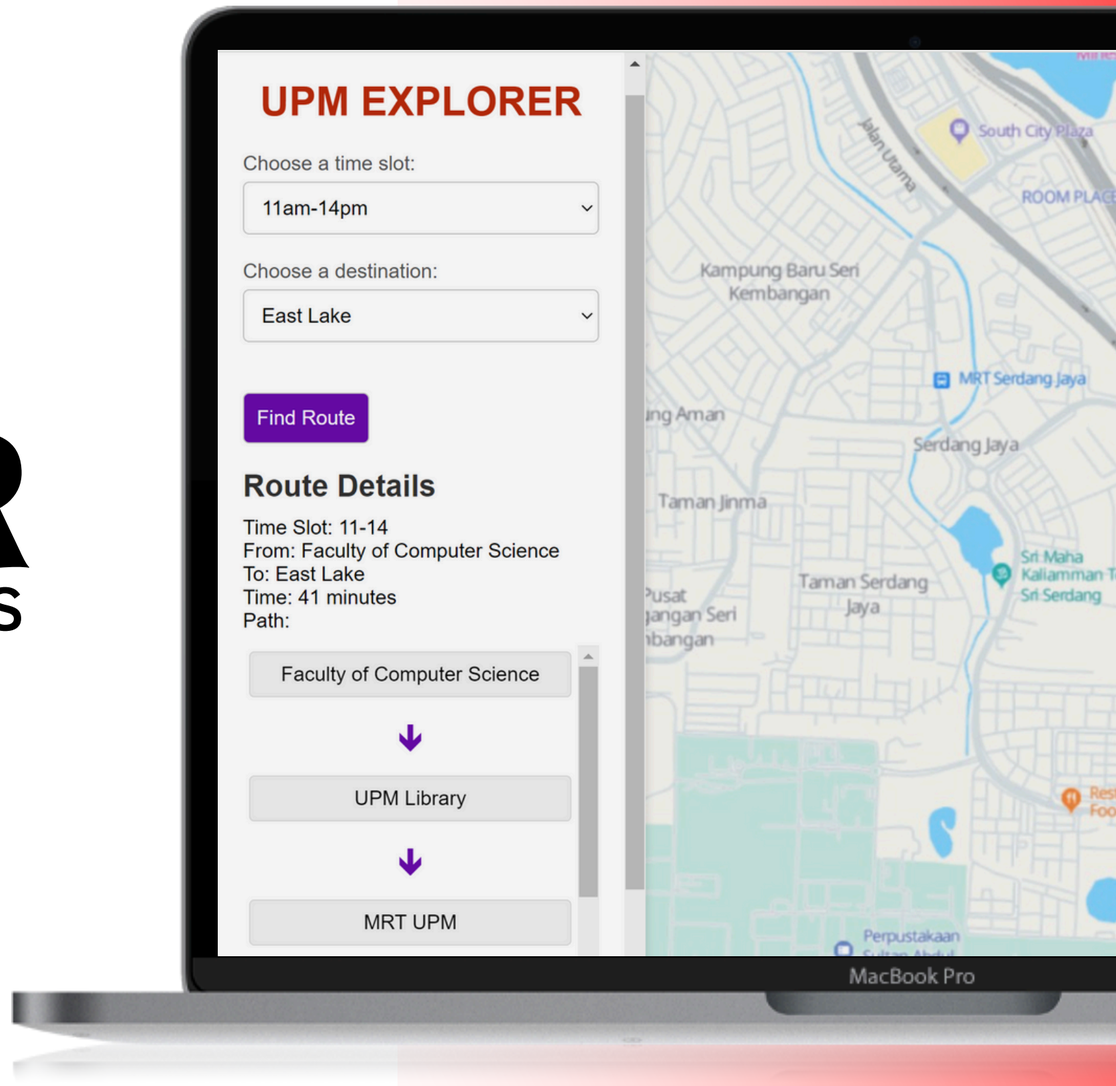
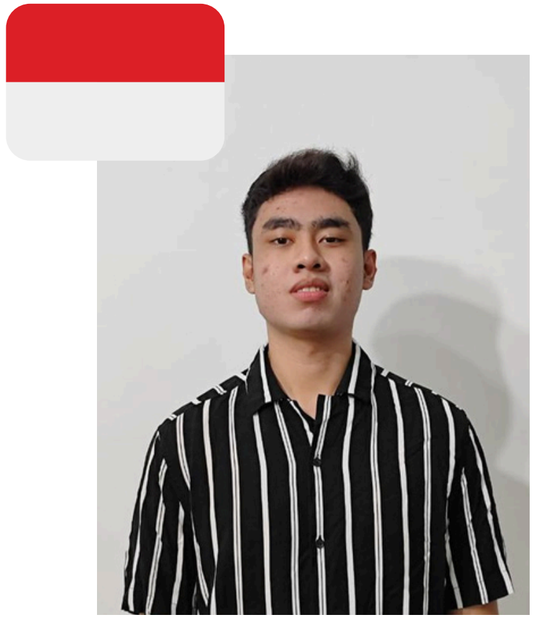
GROUP PROJECT

UPM EXPLORER
MAPS

CSC4202
DESIGN AND ANALYSIS OF ALGORITHMS



GROUP MEMBERS



MUHAMMAD HANIF MURTAZA
213793

YANG ZIXUN
213844

CAI ZESHUO
213733

BACKGROUND (PROBLEM STATEMENT)

MOST INTERNATIONAL STUDENTS AT UPM LIVE IN FIVE MAIN APARTMENTS (ASTETICA, EAST LAKE RESIDENCE, IOI RESORT, UNIV 360, AND TROPICAL VILLA).

ALL THESE STUDENTS COMPLAIN THAT WHEN THEY GO BACK HOME FROM OUR FACULTY THEY OFTEN FACE HEAVY TRAFFIC, ESPECIALLY DURING 3 PEAK HOURS (9-11 AM, 11 AM-2 PM, AND 2-6 PM)

IN THIS PROJECT WE AIM TO HELP THESE STUDENTS FIND THE FASTEST WAY HOME BY USING ALGORITHMS BASED ON OUR TRAFFIC DATA.

MOTIVATION

- Reducing travel time for students allows them to spend more time on productive activities.
- Providing optimal routes can reduce stress and improve the overall quality of life for students.
- It can help in better traffic management and reduce congestion during peak hours.

OBJECTIVES

- **TO MODEL THE UPM CAMPUS AND ITS SURROUNDING AREAS AS A GRAPH:** NODES REPRESENT KEY LOCATIONS WHILE EDGES REPRESENT THE PATHS BETWEEN THESE LOCATIONS WITH ASSOCIATED TRAVEL TIMES.

- **TO DESIGN AND IMPLEMENT AN ALGORITHM:** THE ALGORITHM WILL COMPUTE THE SHORTEST PATHS

- **TO CONSIDER TRAFFIC DATA:** THE ALGORITHM WILL PROVIDE DIFFERENT ROUTE RECOMMENDATIONS BASED ON TRAFFIC CONDITIONS DURING THREE TIME SLOTS: 9-11 AM, 11 AM-2 PM, AND 2-6 PM.

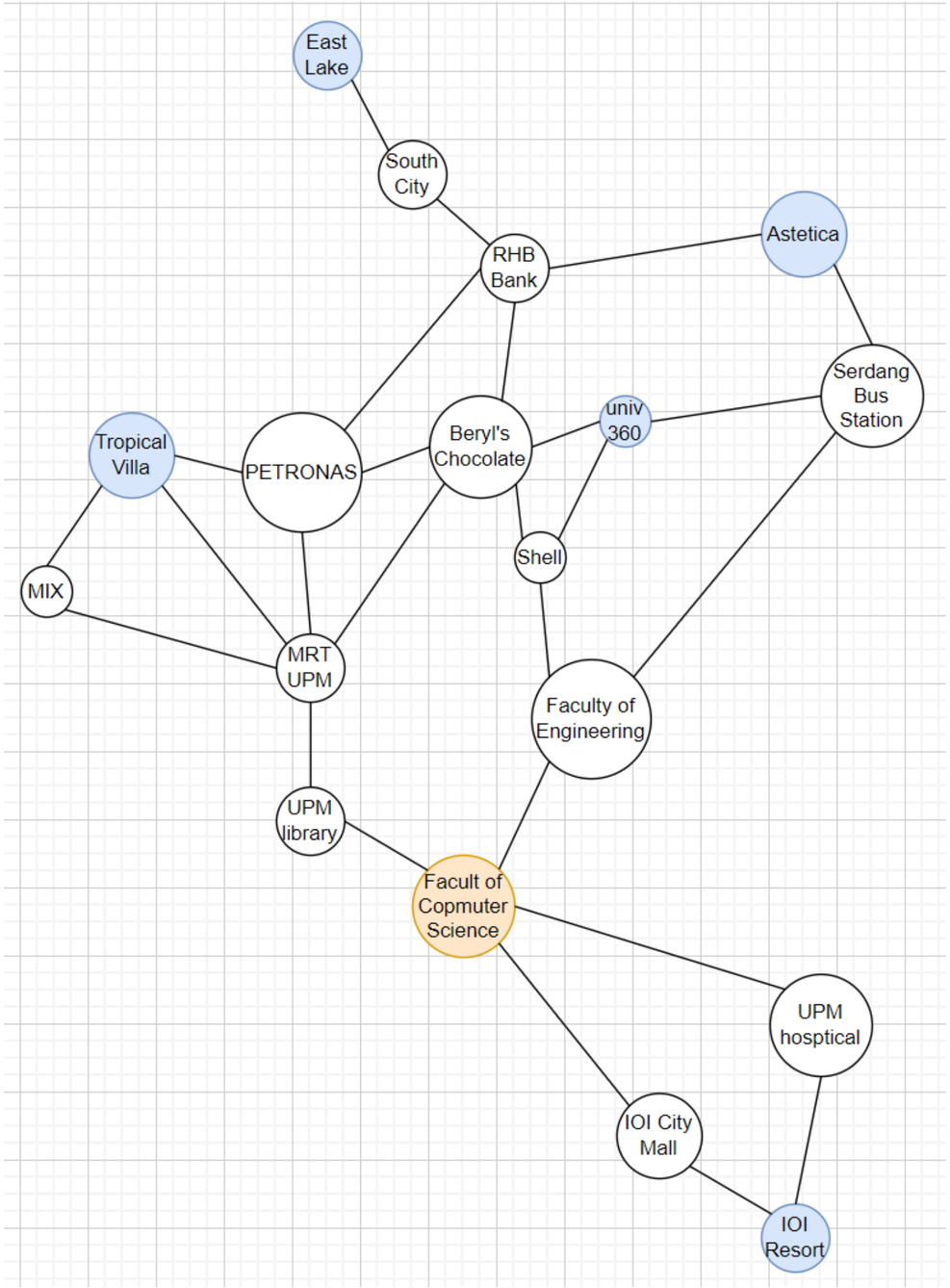
- **TO ENSURE THE ALGORITHM'S EFFICIENCY: BY ANALYZING THE TIME COMPLEXITY OF THE ALGORITHM IN THE BEST, WORST, AND AVERAGE CASE SCENARIOS, WE ENSURE IT IS SUITABLE FOR REAL-TIME APPLICATION.**

GRAPH

YELLOW = START (FSKTM)

BLUE = FINISH (APARTMENT)

WHITE = LANDMARK/CHECKPOINT



```python
import heapq

2 个用法
def dijkstra(graph, start):
    distances = {vertex: float('infinity') for vertex in graph}
    distances[start] = 0
    previous_nodes = {vertex: None for vertex in graph}
    priority_queue = [(0, start)]

    while priority_queue:
        current_distance, current_vertex = heapq.heappop(priority_queue)

        for neighbor, weight in graph[current_vertex].items():
            distance = current_distance + weight

            if distance < distances[neighbor]:
                distances[neighbor] = distance
                previous_nodes[neighbor] = current_vertex
                heapq.heappush(priority_queue, __item: (distance, neighbor))

    return distances, previous_nodes

2 个用法
def reconstruct_path(start, end, previous_nodes):
    path = []
    current_vertex = end

    while current_vertex is not None:
        path.append(current_vertex)
        current_vertex = previous_nodes[current_vertex]


path.reverse()

return path if path[0] ==start else[]
```

ALGORITHM CHOICE

DIJKSTRA'S ALGORITHM WAS CHOSEN FOR THIS PROJECT DUE TO ITS EFFICIENCY IN FINDING THE SHORTEST PATH IN A GRAPH WITH NON-NEGATIVE EDGE WEIGHTS. IT IS WELL-SUITED FOR OUR SCENARIO WHERE WE NEED TO FIND THE OPTIMAL ROUTES FROM THE FACULTY OF COMPUTER SCIENCE TO VARIOUS APARTMENTS AROUND THE UPM CAMPUS. THE ALGORITHM USES A PRIORITY QUEUE TO EXPLORE THE SHORTEST PATHS, ENSURING THAT THE COMPUTATIONAL EFFORT IS MINIMIZED WHILE GUARANTEEING THE SHORTEST PATH IS FOUND.

MADE IN PYTHON

DEMONSTRATION

ALGORITHM TIME COMPLEXITY ANALYSIS

BEST CASE

Scenario: The start and end nodes are connected with minimal nodes and edges.
Explanation: In the best-case scenario, the algorithm quickly finds the shortest path with the least number of nodes and edges.
Time Complexity: O((V + E) log V), where V is the number of vertices and E is the number of edges.

WORST CASE

Scenario: The graph is dense, and the algorithm has to explore all nodes and edges to find the shortest path.
Explanation: In the worst-case scenario, every node is connected to many other nodes, requiring the algorithm to process all possible edges to find the shortest path. This typically happens in a densely connected graph where each node has multiple connections.
Time Complexity:

- O(V^2) for an adjacency matrix representation.
- O((V + E) log V) for an adjacency list representation using a priority queue.

AVERAGE CASE

Scenario: The graph has a moderate number of nodes and edges with balanced connections.
Explanation: In the average case, the algorithm processes nodes and edges in a balanced manner. The graph is neither too sparse nor too dense, representing a typical scenario for route finding.
Time Complexity: O((V + E) log V) for an adjacency list with a priority queue.

BEST CASE: FACULTY OF COMPUTER SCIENCE TO UNIV 360
WORST CASE: FACULTY OF COMPUTER SCIENCE TO IOI RESORT
AVERAGE CASE: FACULTY OF COMPUTER SCIENCE TO TROPICAL VILLA

CONCLUSION

THE UPM EXPLORER PROJECT SUCCESSFULLY IMPLEMENTED AN ALGORITHM TO FIND THE OPTIMAL ROUTE FOR UPM STUDENTS CONSIDERING VARYING TRAFFIC CONDITIONS.

THE RESULTS DEMONSTRATE THE EFFECTIVENESS OF THE ALGORITHM IN PROVIDING RELIABLE ROUTE RECOMMENDATIONS. THIS TOOL CAN SIGNIFICANTLY ENHANCE THE COMMUTING EXPERIENCE FOR STUDENTS BY MINIMIZING TRAVEL TIME AND IMPROVING NAVIGATION.

FOR FUTURE WORK COULD INCLUDE INTEGRATING REAL-TIME TRAFFIC DATA USING APIS TO PROVIDE DYNAMIC ROUTE UPDATES.

ADDITIONALLY, IMPLEMENTING A MOBILE APPLICATION VERSION OF THE TOOL COULD FURTHER ENHANCE ACCESSIBILITY AND USER EXPERIENCE.

Thank you!