**DCIT 204 (SEMESTER PROJECT)**

**GP-1-UG NAVIGATE**

**GROUP MEMBERS:**

Gabriel Kwame Addo Quainoo - 11298181

Boniface Amegbletor Yaw – 11301080

Yaw Peprah Dickson -

Kayor Rahina Abubakar - 11011787

Tackie Richmond - 11078425

Aboagye Christabel Afia Akwaboah - 11356456

Aaron Yeboah - 11124173

**Algorithm Overview**

The algorithm was designed to find the optimal route between two locations on the University of Ghana campus, considering factors such as distance, traffic conditions, and user preferences for landmarks. The following steps outline the process:

**1. Graph Representation**

* **Model the Campus as a Graph**: Represent the campus as a directed or undirected graph where:
  + **Nodes** represent locations (e.g., departments, halls, landmarks).
  + **Edges** represent paths between locations, with weights assigned based on distance or estimated travel time.

**2. Input Parameters**

* **User Input**: Gather user input for:
  + Starting location (Node A)
  + Destination location (Node B)
  + Optional landmarks (if the user wants routes that pass through specific points)

**3. Distance Calculation**

* **Implement Distance Algorithms**: Use algorithms such as:
  + **Dijkstra's Algorithm**: To find the shortest path from Node A to Node B based on edge weights.
  + **Floyd-Warshall Algorithm**: To calculate shortest paths between all pairs of nodes if needed for multiple queries.
  + *A Search Algorithm*\*: If heuristic estimates can be applied to improve efficiency.

**4. Pathfinding Logic**

* **Initialize Data Structures**: Create data structures to store:
  + The shortest path from the starting node to all other nodes.
  + A map to track the previous node for each node to reconstruct the path later.
* **Execute Pathfinding Algorithm**:
  + Start from Node A and explore all reachable nodes.
  + Update the shortest path and previous node information as shorter paths are found.
  + Continue until Node B is reached or all nodes have been explored.

**5. Route Options Generation**

* **Generate Multiple Routes**: If the user requests routes through specific landmarks:
  + Identify paths that pass through the specified landmarks.
  + Use a modified pathfinding approach to ensure the routes include the landmarks.

**6. Sorting and Filtering Routes**

* **Sort Routes**: Once multiple routes are generated, sort them based on:
  + Total distance
  + Estimated arrival time (considering traffic conditions)
* **Use Sorting Algorithms**: Implement sorting algorithms like Quick Sort or Merge Sort for efficient organization.

**7. Output Results**

* **Display Results**: Present the user with:
  + The best route based on their preferences.
  + Alternative routes, if available, with details on distance and estimated time.
* **User Interface**: Utilize a graphical user interface (GUI) to allow users to easily select locations and view results.

**8. Additional Features**

* **Dynamic Updates**: Incorporate real-time traffic data to adjust routes dynamically.
* **User Preferences**: Allow users to set preferences for the type of route (e.g., shortest, least traffic).

**9. Performance Optimization**

* **Enhance Algorithm Efficiency**: Apply techniques such as:
  + **Divide and Conquer**: Break down the problem into smaller subproblems.
  + **Greedy Algorithms**: Make locally optimal choices at each step with the hope of finding a global optimum.
  + **Dynamic Programming**: Store results of subproblems to avoid redundant calculations.

**10. Testing and Validation**

* **Test the Algorithm**: Validate the algorithm with various test cases to ensure accuracy and efficiency.
* **User Feedback**: Gather feedback from users to improve the application and algorithm performance.

**Flowchart**