**✅ Phase 1: Requirements Gathering & Planning**

**🔍 1.1. Define the Problem Clearly**

**Problem Statement:**

Students and staff on the University of Ghana campus often need to find the best path from one campus location to another. Routes may vary based on distance, traffic, and preferred landmarks. There is a need for a system that intelligently determines the best route options, considering both shortest paths and user preferences.

**🎯 1.2. Identify Project Goals**

**✔️ Primary Goals**

* Build a **Java application** that finds the **optimal route** between two UG campus locations.
* Provide **multiple route options** based on:
  + **Shortest distance**
  + **Fastest estimated arrival time**
  + **Landmark preferences**
* Enable **searching** for routes through specific landmarks.

**✔️ Secondary Goals**

* Use **various algorithms** to optimize performance.
* Offer a **user-friendly** and interactive GUI.
* Simulate or integrate **traffic-like behavior**.

**👥 1.3. Define Stakeholders and Users**

| **Role** | **Description** |
| --- | --- |
| End Users | UG students, staff, visitors |
| Developers | You (student group) |
| Instructors | Evaluators; want to see algorithmic use |

**📝 1.4. Gather Functional Requirements**

| **Feature** | **Description** |
| --- | --- |
| **Route Finder** | User inputs start & end locations → app shows best routes. |
| **Sort Routes** | Routes sorted by distance or arrival time. |
| **Search by Landmark** | User inputs a landmark → app shows routes via that landmark. |
| **Multiple Algorithms** | Use Dijkstra, A\*, Floyd-Warshall, etc. |
| **Landmark-based Input** | Input “Bank” → app suggests relevant routes. |
| **View Details** | Route displays: path, total distance, time estimate. |

**📐 1.5. Gather Non-Functional Requirements**

| **Requirement** | **Description** |
| --- | --- |
| **Performance** | Fast computation of results even for multiple routes. |
| **Usability** | Easy-to-use UI with minimal learning curve. |
| **Portability** | Application should run on Windows/macOS/Linux (Java). |
| **Maintainability** | Code should be modular and well-documented. |

**🧠 1.6. Identify Constraints & Assumptions**

**🔒 Constraints**

* Application must be written in **Java**.
* Only **UG campus locations** should be supported.
* **Offline access**: Assume the application does **not use live traffic APIs** like Google Maps.

**💡 Assumptions**

* Distances and times are either:
  + Pre-defined/hardcoded
  + Pulled from tools (like Google Maps) **once** during setup
* "Traffic" may be simulated via random weighting.

**🧮 1.7. List Algorithms to Be Used (as required)**

| **Type** | **Algorithms** |
| --- | --- |
| **Graph Search** | Dijkstra, A\*, Floyd-Warshall |
| **Sorting** | Merge Sort, Quick Sort |
| **Optimization** | Greedy, Dynamic Programming |
| **Heuristic Routing** | A\* |
| **Transport Modeling** | Vogel, Northwest Corner (creative use) |
| **Critical Path** | For max-time paths (optional) |
| **Search** | Linear Search, Binary Search for landmarks |

**📦 1.8. Resources Needed**

| **Resource** | **Purpose** |
| --- | --- |
| Java SDK | Development |
| JavaFX or Swing | GUI development |
| GitHub | Version control |
| Campus Map | Location data |
| Distance Table | Preprocessed or manually created |
| Testing Library (JUnit) | Test automation |
| Diagram Tool (draw.io, Lucidchart) | Design diagrams |
| Word/PDF editor | Documentation |

**🗓️ 1.9. Project Planning and Task Breakdown**

**🧾 Deliverables of Phase 1:**

* ✅ Requirements Specification Document (we can start drafting this next)
* ✅ Functional & Non-functional Requirements list
* ✅ Stakeholders & Use Cases
* ✅ Feature list (must have, nice to have)
* ✅ Tool & Technology list
* ✅ Schedule (initial timeline or Gantt chart)

**🧰 1.10. Suggested Roles in Your Group**

| **Role** | **Responsibility** |
| --- | --- |
| Project Manager | Overall coordination, tracking tasks |
| UI Developer | GUI in JavaFX/Swing |
| Algorithm Developer | All routing, sorting, searching logic |
| Data Modeler | Build the graph/map structure |
| Tester & QA | Unit testing, integration testing |
| Documenter | Write all deliverables |

**✅ Summary**

By the end of Phase 1, you should have:

* A **clear understanding** of what the app will do
* A complete **list of features**
* An overview of **tools and algorithms**
* A shared **vision among group members**
* The foundation for the **system design phase**