Below is a **complete, professional-level roadmap**, broken down into **phases**. Each phase includes **technical tasks**, **tools**, and **deliverables** — from **planning and design** to **coding, testing, documentation, and deployment**.

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

**🎯 Objectives**

* Understand problem scope
* Define users and goals
* List functional and non-functional requirements
* Identify all algorithms and data structures needed

**📌 Key Requirements (Extracted from your document)**

* Input: Source and destination (possibly with landmarks)
* Output: Route options (sorted by distance and time)
* Constraints: Traffic, landmark preference, shortest/fastest path
* Domain: UG Campus (known locations)

**📋 Functional Requirements**

* Accept input locations (text field / dropdown)
* Display multiple route options
* Show distance and estimated time
* Search by landmark
* Sort routes by criteria (distance, time)

**📋 Non-Functional Requirements**

* Fast response (optimized algorithms)
* User-friendly GUI
* Scalable graph structure
* Accurate traffic simulation (simplified or mocked)

**🔧 Tools/Technologies**

* **Java (Swing/JavaFX)** – Application GUI
* **Graph algorithms** – Dijkstra, A\*, Floyd-Warshall
* **Optimization** – Greedy, DP, Divide & Conquer
* **Mapping Tool** – Google Maps (reference only) or hardcoded distances

**📤 Deliverables**

* Requirements document
* List of campus landmarks (coordinates or names)
* Planning document with schedule and task breakdown

**✅ Phase 2: System Design**

**🧠 Conceptual Design**

* Graph-based model: Nodes = landmarks/buildings, Edges = paths (with weights: distance/time)
* Input parser (location, landmark)
* Output formatter (route visualizer, sorted lists)

**📊 Architecture Components**

* **Data Layer**: Map data, adjacency list/matrix
* **Algorithm Engine**: Routing + optimization
* **UI Layer**: Java-based GUI
* **Search Module**: Landmark-based queries

**📐 Class Diagram (suggested):**

* Node (name, coordinates)
* Edge (from, to, distance, time)
* Graph (nodes, edges, addNode, addEdge)
* RoutePlanner (Dijkstra, A\*, etc.)
* LandmarkSearch (search, filter)
* MainApp (GUI controller)

**📤 Deliverables**

* System design diagram
* Class diagram
* Brief technical spec

**✅ Phase 3: Map Data and Graph Setup**

**🔍 Tasks**

* Map UG campus with buildings, walkways, roads
* Assign distances (mock, measured, or Google Maps)
* Build static graph (Adjacency list is best for flexibility)

**📘 Resources**

* UG campus map
* Google Maps distance estimator
* JSON file to represent graph (optional)

**📤 Deliverables**

* Graph data structure initialized
* Landmarks list (e.g., Bank, Balme Library, Mensah Sarbah Hall)

**✅ Phase 4: Algorithm Development**

**🧩 Required Algorithms**

| **Algorithm** | **Purpose** |
| --- | --- |
| Dijkstra | Shortest path (distance) |
| A\* | Heuristic + distance |
| Floyd-Warshall | All-pairs shortest path |
| Critical Path Method (CPM) | Longest path in DAG (if needed) |
| Vogel Approximation & NW Corner | Load balancing (creatively applied for optimization) |
| Quick Sort / Merge Sort | Route sorting |
| Binary/Linear Search | Landmark route searching |
| Greedy / DP / Divide & Conquer | Performance tuning |

**📤 Deliverables**

* Each algorithm in a separate testable class
* JUnit tests for algorithms
* Documentation (how and where it is used)

**✅ Phase 5: User Interface Development**

**💻 Java UI Components**

* Input: Source, Destination, Landmark (Dropdowns/TextField)
* Button: “Find Route”
* Output: Table/List of routes with:
  + Route path
  + Distance
  + Time
  + Landmark(s) passed
* Optional: Simple campus map overlay

**👨‍🏫 Tools**

* Java Swing or JavaFX (FX is more modern)

**📤 Deliverables**

* GUI connected to algorithm engine
* Input/output flow working
* Sample screenshots

**✅ Phase 6: Integration & Testing**

**🧪 Unit Testing**

* Algorithms
* Landmark filtering
* Sorting logic

**🧪 Integration Testing**

* Full input-to-output route generation
* GUI + backend communication

**📤 Deliverables**

* JUnit test suite
* Test cases and results
* Bug list and fixes

**✅ Phase 7: Optimization & Enhancements**

**🚀 Advanced Tuning**

* Apply Greedy strategies for quicker approximate solutions
* Use Dynamic Programming for repeated subproblems (e.g., caching)
* Optimize search performance

**🤯 Creative Ideas**

* Add heatmap simulation (busy routes)
* Predictive suggestion of landmarks
* User profile/history (mocked)

**📤 Deliverables**

* Enhanced code
* Performance comparisons (with/without optimization)

**✅ Phase 8: Documentation**

**📘 Developer Documentation**

* System architecture
* Class/function descriptions
* Algorithm justifications

**📘 User Documentation**

* How to install
* How to use each feature
* Example use cases

**📤 Deliverables**

* PDF/Word docs
* Commented code

**✅ Phase 9: Deployment**

**🛠️ Tasks**

* Export JAR file
* Optionally create installer
* Add shortcut/run instructions

**🖥️ Suggested Tools**

* IntelliJ or Eclipse export
* JSmooth or Launch4j for Windows installer

**📤 Deliverables**

* Final executable (JAR/installer)
* Deployment instructions

**✅ Phase 10: Presentation**

**🎤 Presentation Tips**

* Explain **problem → solution**
* Show **demo** (GUI in action)
* Explain **how each algorithm is used**
* Highlight creative features
* Share **testing/performance** metrics
* Q&A prep: Be ready to explain algorithm choices

**🧰 Resource Summary**

**Tools**

* Java SDK + IDE (Eclipse, IntelliJ)
* JavaFX / Swing
* JUnit for testing
* Google Maps (for reference)
* GitHub for version control

**Deliverables Recap**

|  |  |
| --- | --- |
| **Deliverable** | **Description** |
| Requirements & Planning Docs | Initial analysis and project breakdown |
| Design Diagrams | Class + system diagrams |
| Graph and Map Data | UG campus map + graph structure |
| Algorithm Modules | Java classes for routing & optimization |
| Java GUI | User interface for inputs & results |
| Testing Suite | Unit and integration tests |
| User & Developer Documentation | Instructions and code explanations |
| Final Application (JAR or Installer) | Deployment-ready package |
| Presentation Slides | Clear overview of project and demo |