Software Requirements Specification

on

Pathfinder AI

Team Details:

Group No. 41

Mentored By – Dr. Panduranga Raviteja

S.No.	SAP Id	Name	Course Spl.
01	500107149	Keertivallabh Sharma	FSAI- B1
02	500107065	Jyotiraditya Singh	FSAI- B1
03	500101806	Shrey Sharma	FSAI- B1
04	500105685	Rishav Singh	FSAI- B1

Signatures-

Mentor: **Panel Members:**

School Of Computer Science
University of Petroleum and Energy Studies,
Dehradun

Table of Contents

1. Introduction

1.1 Purpose

*The purpose of this Software Requirements Specification (SRS) document is to provide a comprehensive description of the **Pathfinder AI**. This project aims to assist in determining the optimal routes for the construction of roads and highways, particularly in urban areas. By utilizing a modified version of path determining algorithms, the system will allow users to upload city maps, select cities for connection, and mark critical points of interest such as hospitals, government buildings, airports, and stations. The primary goal is to develop a tool that helps optimize infrastructure development in India, keeping in mind constraints like traffic, distance, and accessibility.*

1.2 Document Conventions

The following standards and typographical conventions have been applied throughout this SRS document:

- **Fonts and Styles**:
    - **Headings** are formatted in bold to differentiate sections and subsections.
    - **Normal text** uses a standard font size of 12 for body content.
- **Numbering**:
    - All sections and subsections follow a hierarchical numbering format (e.g., 1, 1.1, 1.2) for easy reference.
    - Requirements are uniquely identified with sequential tags (e.g., REQ-1, REQ-2).
- **Prioritization**:
    - Each requirement is assigned its own priority (High, Medium, Low) to indicate its importance.
    - If not explicitly mentioned, the priority for detailed requirements will inherit the priority of their parent feature.
- **Highlighting**:
    - **Bold** text is used for emphasis on key terms and concepts.
    - Requirements, assumptions, and constraints are often listed in bullet points to enhance readability.

1.3 Intended Audience and Reading Suggestions

*The **Pathfinder AI** SRS document is intended for the following audience:*

- ***Developers**: To understand the functional and technical requirements of the system for proper implementation.*
- ***Urban Planners & Civil Engineers**: To assess the relevance of the system in solving real-world infrastructure problems and provide insights on user needs.*
- ***Testers**: To derive test cases based on the requirements and ensure the system behaves as intended.*
- ***Academic Reviewers and Evaluators**: To evaluate the technical depth and real-world applicability of the project for academic grading or feedback purposes.*

Reading Suggestions:

- *For developers, this section provides detailed system requirements, including functional and non-functional requirements that will guide the system's implementation.*
- *Stakeholders and project managers should focus on the system's use cases and external interface requirements to understand how the system will interact with users and external systems.*
- *Urban planners, civil engineers, and testers will benefit from reviewing the performance and system attributes to ensure the system meets expected standards and performs under various constraints.*

By following these reading suggestions, each group can focus on the sections most relevant to their roles and responsibilities within the project

1.4 Project Scope

*The **Pathfinder AI** is designed to assist in optimizing road and highway construction by identifying the most efficient routes between selected cities, while considering critical infrastructure points like hospitals, government offices, airports, and stations. The system will primarily be used by urban planners, civil engineers, and government authorities responsible for city development projects.*

The project will utilize a modified version path determining algorithms to calculate the shortest and most cost-effective routes. The system will take into account various constraints, such as:

- ***Distance and Connectivity**: Ensuring the shortest possible route while maintaining connectivity between essential locations.*
- ***Traffic Density**: Considering areas with higher traffic and providing alternate routes to avoid congestion.*
- ***Urban Development**: Factoring in the development status of different regions, prioritizing routes that enhance connectivity in developing areas.*

Key features of the system include:

- ***Map Upload**: Users can upload a city-wide map to the system.*
- ***City and Landmark Selection**: Users can select cities and mark important places like hospitals, airports, government offices, etc.*
- ***Route Calculation**: The system will compute optimal routes using a pathfinding algorithm.*
- ***Visualization**: A graphical interface will display the calculated routes on the map for better visualization.*

The scope of this project includes the development of the core route optimization algorithm, a user-friendly interface, and visualization capabilities. Future expansions may integrate live traffic data, environmental impact assessments, and cost estimations for construction projects.

This project is aimed at aiding the infrastructure development efforts, particularly in developing nations like India, where optimized route planning can have a significant impact on economic growth and urban development.

1.5 References

The following sources and documents were referenced during the development of this SRS:

1. **IEEE Standard 830-1998** – Recommended Practice for Software Requirements Specifications.
   URL: https://ieeexplore.ieee.org/document/720574

2. **Geographic Information Systems (GIS) Standards and Practices** – Guidelines for map data formats and integration.
   URL: https://www.gisstandards.org

3. **Dijkstra's Algorithm and Pathfinding Literature** – Papers and textbooks on graph theory, shortest path algorithms, and their applications.
   Example Reference: "Introduction to Algorithms" by Thomas H. Cormen, Charles E. Leiserson, Ronald L. Rivest, and Clifford Stein.

4. **User Interface Design Guidelines** – Principles for designing user-friendly interfaces.
   URL: https://www.usability.gov

5. **Software Engineering Practices** – General guidelines for software development, coding standards, and system architecture.
   URL: https://www.sei.cmu.edu/research-capabilities/software-engineering

6. **APIs and Data Integration for Urban Planning Tools** – Documentation for third-party APIs and real-time data services.
   URL: https://developer.here.com/documentation

These references provide foundational support and best practices for the specifications and design considerations in this document.

2. Overall Description

2.1 Product Perspective

The Pathfinder AI System is a standalone tool for urban planners, civil engineers, and government authorities to optimize road and highway construction projects using advanced pathfinding algorithms. Key elements include:

- **System Interfaces**:
    - *Map Upload*: Supports formats like PNG, JPEG, and GIS data.
    - *User Input*: Users mark cities and landmarks for route calculations.
    - *Output Visualization*: Displays optimized routes on the uploaded map.
- **User Interface**:
    - Intuitive *GUI* designed for technical and non-technical users, allowing map interaction, data input, and route visualization.
- **Hardware Interfaces**:
    - Runs on desktop systems capable of handling large maps and computations.
- **Software Interfaces**:
    - *OS Compatibility*: Supports Windows, macOS, and Linux.
    - Optional *GIS Integration* for advanced map features in future versions.
- **Product Functions**:
    - Map upload, city/landmark marking, route calculation via a modified Dijkstra's algorithm, and visual route display.
- **Constraints**:
    - Must handle large datasets efficiently.
    - Compatible with common map file formats.
    - Route calculation should be quick, even for large regions.
- **Assumptions & Dependencies**:
    - Assumes access to accurate maps.
    - Relies on user input accuracy.

- Future enhancements may include real-time data integration via third-party APIs.

This system focuses on usability, efficiency, and scalability to assist with infrastructure route planning.

2.2 Product Features

The **Pathfinder AI** offers several key features aimed at simplifying and optimizing the process of planning road and highway construction. Below is a detailed description of the main product features:

1. City-Wide Map Upload

- **Description**: Users can upload detailed maps of cities or regions in various formats (e.g., PNG, JPEG, or GIS data). The system will parse the uploaded map to allow for route planning and visualization.
- **Purpose**: Provides a geographical basis for users to define routes and mark critical points of interest.

2. City and Landmark Selection

- **Description**: Users can select specific cities or points on the map to define the start and end locations of the route. They can also mark key landmarks such as hospitals, government offices, airports, and stations.
- **Purpose**: Enables the selection of important locations that must be considered in the route planning process.

3. Optimal Route Calculation

- **Description**: The system uses a modified version of Dijkstra's algorithm to calculate the most efficient route between selected cities. The algorithm considers distance, traffic density, and connectivity to compute the shortest and most feasible path.
- **Purpose**: Automates the complex task of determining the best route for road construction, accounting for various constraints.

4. Multiple Route Options

- **Description**: The system will not only generate one optimal route but may also provide multiple alternative routes. These alternatives could prioritize different factors such as shortest distance, least traffic, or better connectivity to landmarks.
- **Purpose**: Provides flexibility to users by offering several viable route options, allowing them to choose based on specific project requirements.

5. Route Visualization

- **Description**: Once the route has been calculated, the system will display it visually on the uploaded map, with clear distinctions for different types of roads and highways. Marked landmarks will also be highlighted.
- **Purpose**: Provides users with a clear and interactive visualization of the proposed routes, making it easier to analyze and validate route options.

6. Editable Markers and Paths

- **Description**: Users can adjust their selections by moving city markers or landmark points on the map. The system will recalculate the route dynamically based on these changes.
- **Purpose**: Allows flexibility in route planning by enabling users to modify their selections without restarting the process.

7. Cost and Time Estimation (Future Expansion)

- **Description**: A feature that provides an estimate of the construction cost and time based on the selected route, the length of the roads, and predefined metrics.
- **Purpose**: Assists stakeholders in making informed decisions by giving them an estimate of the project's feasibility in terms of cost and time.

8. Exportable Reports

- **Description**: The system will generate reports summarizing the calculated routes, distance covered, important landmarks along the way, and any other relevant information. These reports can be exported in formats such as PDF or CSV.
- **Purpose**: Allows users to share the results with other stakeholders or use them for presentations and project planning.

9. Scalable for Larger Maps

- **Description**: The system is designed to handle maps of varying sizes, from small cities to large metropolitan areas. It will be optimized to ensure performance remains efficient even with complex and large datasets.
- **Purpose**: Ensures the system can be used for both small and large-scale infrastructure projects.

These features are designed to provide comprehensive support for route planning while ensuring ease of use and flexibility for users involved in infrastructure development and urban planning. Future features, like cost estimation and real-time traffic integration, could further enhance the system’s capabilities.

2.3 User Classes and Characteristics

The **Pathfinder AI** is designed to cater to a range of users with different roles and responsibilities within infrastructure planning and development. Below are the primary user classes along with their respective characteristics and interaction with the system:

1. Urban Planners

- **Description**: Professionals responsible for planning and designing city layouts, infrastructure development, and road networks.
- **Characteristics**:
    - Familiar with city planning principles and regulations.
    - Moderate technical expertise, typically capable of using Geographic Information Systems (GIS) and related tools.
    - Focus on optimizing city layout and traffic management.
- **System Interaction**:
    - Upload city maps, mark important urban landmarks, and select cities for connectivity.
    - Use the route calculation feature to design efficient road networks within urban areas.
    - View multiple route options and finalize optimal routes for city planning purposes.

2. Civil Engineers

- **Description**: Engineers involved in the construction and development of physical infrastructure such as roads, highways, and bridges.
- **Characteristics**:
    - High technical expertise in infrastructure and civil engineering.
    - Familiar with construction costs, time estimates, and logistical challenges.
    - Focus on practical execution, cost-efficiency, and feasibility.
- **System Interaction**:
    - Utilize the system to assess the feasibility of proposed routes.
    - Export detailed reports that include route distances and potential construction costs (if available).
    - Adjust route options based on technical constraints and construction requirements.

3. Government Authorities

- **Description**: Local, regional, or national government officials responsible for approving and funding public infrastructure projects.
- **Characteristics**:
    - Limited technical expertise but high-level decision-making roles.
    - Focus on policy compliance, budget considerations, and regional development.
- **System Interaction**:
    - Review route visualization and cost estimation to assess the viability of proposed infrastructure projects.
    - Export reports for discussion with various departments and stakeholders.
    - Monitor overall progress and ensure the project aligns with national or regional development goals.

4. Stakeholders and Sponsors

- **Description**: Investors, business leaders, and other stakeholders with a financial interest in infrastructure development projects.
- **Characteristics**:
    - Limited technical background, primarily focused on the financial and logistical aspects of projects.
    - Focus on project timelines, cost efficiency, and return on investment (ROI).
- **System Interaction**:
    - Review high-level reports on proposed routes and estimated costs.
    - Use visualizations to gain a clear understanding of the project's impact.
    - Evaluate the project based on time and cost estimation outputs (if available).

5. Testers

- **Description**: Quality assurance engineers responsible for testing the system to ensure it meets functional and performance requirements.
- **Characteristics**:
    - High technical expertise in software testing methodologies and tools.
    - Familiarity with both functional and non-functional system requirements.
    - Focus on identifying bugs, performance issues, and usability concerns.

- **System Interaction**:
    - Test the system for route accuracy, performance under load (handling large maps), and correct visualization of routes.
    - Ensure that user interactions (e.g., map uploads, city selection) work seamlessly and are user-friendly.

6. Academic Researchers and Students

- **Description**: Researchers and students working in fields related to urban development, GIS, computer science, or civil engineering.
- **Characteristics**:
    - Varying levels of technical expertise depending on experience.
    - Focus on research-oriented use cases, such as developing new algorithms or exploring new ways to enhance route optimization.
- **System Interaction**:
    - Use the system for research purposes, such as testing different pathfinding algorithms or optimizing route calculations.
    - Explore possible integrations with other tools (e.g., GIS software, real-time traffic data) to enhance the system's capabilities.

This diverse user base highlights the need for a system that is intuitive yet powerful, offering varying levels of complexity based on the user's technical expertise and the nature of their role in infrastructure development.

2.4 Operating Environment

The **Pathfinder AI** is designed to operate in a variety of environments to accommodate users from different sectors. Below is an outline of the required operating environments:

1. Hardware Requirements

- **Processor**: Minimum 2 GHz dual-core processor (recommended: 3.0 GHz quad-core or higher).
- **Memory (RAM)**: Minimum 4 GB RAM (recommended: 8 GB or higher, especially for handling large city maps).
- **Storage**: 500 MB of free storage for application installation, with additional space required for map storage and data files.

- **Graphics**: A dedicated GPU is not required, but hardware acceleration is recommended for faster processing of large maps.
- **Display**: Minimum screen resolution of 1280x720 (recommended: 1920x1080 for better map visualization).

2. Software Requirements

- **Operating Systems**:
  - Windows 10 or higher
  - macOS 10.13 or higher
  - Linux distributions (e.g., Ubuntu 18.04 or higher, Fedora, etc.)
- **Programming Languages**:
  - The system will be developed using languages like JavaScript (for the frontend), Python or Node.js (for backend services), and Java (for core algorithms).
- **Database**:
  - A relational database such as PostgreSQL or MySQL may be used to store data related to maps, routes, landmarks, and user configurations.
- **Map Support**:
  - The system will support map files in formats like PNG, JPEG, and possibly GIS data formats like GeoJSON or KML for future integration with GIS tools.

3. Development and Testing Environment

- **Development Tools**:
  - The development environment will include Visual Studio Code or similar IDEs, version control using Git, and testing frameworks such as Jest (for JavaScript) and JUnit (for Java).
- **Testing Platforms**:
  - The system will be tested on different platforms to ensure compatibility across Windows, macOS, and Linux environments. It will also undergo load testing to ensure it performs efficiently with large datasets.

4. Networking and Internet Requirements

- **Offline Mode**:
  - The system can function in offline mode for basic route planning and map handling.
- **Online Mode**:

- For future versions or expansions, such as real-time traffic data integration, the system will require an active internet connection to fetch live data via APIs from third-party services.

5. Security Considerations

- **User Authentication**: If the system involves saving user-specific configurations or map data to a cloud server, secure authentication mechanisms will be required.
- **Data Encryption**: Any stored or transmitted data related to maps and routes will need to be encrypted to protect sensitive geographical information.

This operating environment is designed to ensure smooth performance and compatibility across a wide range of hardware and software setups while allowing scalability for future feature enhancements.

2.5 Design and Implementation Constraints

The **Pathfinder AI** must adhere to several constraints that will affect its design and implementation. These constraints are dictated by technical, regulatory, and operational factors, as well as the requirements of the development and target environments.

1. Hardware Limitations

- **Processing Power**: The system will be designed to run on moderate hardware (2 GHz dual-core processor, 4 GB RAM minimum). Heavy computation tasks, such as pathfinding for large maps, must be optimized to prevent performance bottlenecks on lower-end machines.
- **Memory Requirements**: Efficient memory management is required, especially when working with large map files or city datasets. This could limit the complexity of algorithms and the size of data handled at once.
- **Storage Constraints**: Users with limited storage should be able to work with compressed map files or offload map storage to external databases or cloud storage.

2. Regulatory and Compliance Constraints

- **Data Privacy**: The system must comply with data privacy regulations, such as GDPR (General Data Protection Regulation), when handling geographic data or user-specific information (e.g., saved configurations, uploaded maps).
- **Geographic Restrictions**: Some map data or user information might be subject to geographic or national security restrictions. This will require the system to comply with local laws governing the usage and storage of geographic information.

3. Technological Constraints

- **Algorithm Efficiency**: The modified Dijkstra's algorithm or any pathfinding technique used must be efficient enough to handle large datasets without causing significant delays. This could limit the choice of algorithms to those with known performance characteristics.
- **Technology Stack**: The system is limited by the choice of technologies and tools:
    - **Frontend**: JavaScript frameworks such as React or Vue.js for interactive map visualization.
    - **Backend**: Node.js or Python for server-side route calculation.
    - **Database**: Relational databases (e.g., PostgreSQL or MySQL) will be used for data storage, limiting the complexity of relationships that can be handled.
    - **Map Handling**: Integration with GIS data and real-time traffic data may introduce constraints on the types of APIs or services that can be integrated, as well as licensing restrictions on map providers.

4. Security Constraints

- **User Authentication**: If user-specific configurations are stored or accessed, secure user authentication must be implemented, including session management and encryption protocols. This could introduce limitations on user interactions when offline.
- **Data Encryption**: Geographic data, user information, and saved configurations must be encrypted both at rest and in transit, which could introduce performance overheads, particularly for larger files.

5. Interface Constraints

- **APIs and External Tools**: If the system integrates with third-party services, such as GIS tools or real-time traffic APIs, the design will be constrained by the availability, licensing, and technical limitations of these services.
- **Platform Compatibility**: The system must operate seamlessly across Windows, macOS, and Linux, requiring cross-platform compatibility. This may limit the use of platform-specific tools and libraries.

6. Design and Programming Standards

- **Modular Design**: The system will follow a modular architecture to ensure ease of maintenance and scalability, which may limit the ability to implement certain tightly coupled features.
- **Coding Standards**: The development team must adhere to specific coding standards and best practices (e.g., PEP 8 for Python or ESLint for JavaScript), which may influence design decisions.

- **Maintainability**: As the system might be maintained by different developers over time, clean code practices, proper documentation, and the use of well-known frameworks are essential. This could limit the use of experimental or less commonly used technologies.

7. Performance Constraints

- **Real-Time Data Processing**: If real-time traffic data or other live inputs are integrated, the system will need to efficiently process this information without significantly impacting performance, placing constraints on how real-time updates are handled.
- **Response Time**: The system should respond quickly, even when processing large maps or complex route calculations. This limits the complexity of algorithms and the potential use of machine learning models, which might introduce latency.

8. Scalability and Future Expansion

- **Scalability Constraints**: The system should be designed with scalability in mind, which may limit the use of certain monolithic architectures in favor of microservices. Additionally, the backend must support scalability for handling larger datasets and concurrent users.

These constraints provide a framework within which the development team must work, ensuring that the system meets technical, operational, and regulatory requirements while remaining efficient, secure, and user-friendly.

2.6 User Documentation

The **Pathfinder AI** will include a comprehensive set of user documentation to ensure that users can effectively interact with the software. The following components will be delivered as part of the user documentation:

1. User Manual

- A detailed **User Manual** will be provided that outlines the following:
    - **Installation Guide**: Instructions for installing the system on supported platforms (Windows, macOS, and Linux), including prerequisites like necessary software and hardware configurations.
    - **System Overview**: A description of the software's purpose, key features, and a basic overview of its interface.
    - **How-To Guides**: Step-by-step guides to help users accomplish tasks such as:
        - Uploading and importing maps.
        - Defining routes between cities.

- Marking important landmarks (e.g., hospitals, airports, offices).
- Running the route optimization algorithm.
- Saving, exporting, and sharing routes.

2. Online Help System

- The system will include an **Online Help** section, accessible through the application's interface. It will provide:
  - Context-sensitive help for various sections of the system, such as the map import, route selection, and optimization process.
  - Tooltips and in-app hints that provide quick explanations of buttons and features.

3. Tutorials

- **Interactive Tutorials** will be provided to guide new users through the system. These tutorials will focus on:
  - Basic system navigation.
  - Uploading a city map and understanding the UI.
  - Running the pathfinding algorithm for different city configurations.
  - Customizing routes and visualizing results.

4. FAQs and Troubleshooting

- A **Frequently Asked Questions (FAQ)** section will address common issues and queries, such as:
  - Map format issues.
  - Performance optimization for larger maps.
  - Interpretation of route optimization results.
- **Troubleshooting Guide**: A section dedicated to troubleshooting common problems, with solutions for issues like system crashes, map incompatibility, or errors in route calculations.

5. Video Tutorials

- A set of **Video Tutorials** will be available online (YouTube or other platforms) that demonstrate key features and workflows of the system. Topics covered may include:
  - Importing maps and setting city boundaries.
  - Running and interpreting the optimized routes.

- Managing user settings and map configurations.

6. API Documentation (If Applicable)

- If an API is exposed for external services or third-party integrations, **API Documentation** will be provided, including:
  - A description of the available endpoints.
  - Usage instructions for interacting with the API.
  - Code samples and common use cases for developers.

7. Delivery Formats

- The user documentation will be delivered in the following formats:
  - **PDF** for the **User Manual** and **Installation Guide**.
  - **HTML** for the **Online Help** system, accessible via a web browser.
  - **Interactive In-App Help** embedded directly into the system's interface.
  - **Video Tutorials** available on online platforms.
  - **API Documentation** in markdown or HTML format for easy reference.

By offering a range of documentation formats and interactive help, the system ensures that users of all skill levels can effectively use and maintain the **Pathfinder AI**.

2.7 Assumptions and Dependencies

The development and implementation of the **Pathfinder AI** rely on several assumptions and dependencies that could impact the project. These assumptions and dependencies pertain to the technology, data sources, and operating environments involved in the project. If any of these assumptions prove to be incorrect or change during the course of development, they could potentially affect the project's timeline, functionality, or scope.

Assumptions

1. **Map Data Availability**
   - It is assumed that users will provide valid and accurate map data in a compatible format (e.g., GeoJSON, KML) for the system to process. The system depends on this data for route optimization.
   - It is also assumed that users will be able to access publicly available map data or purchase geographic data if necessary.

2. **Access to Third-Party APIs**
    - The system assumes continuous access to third-party APIs (e.g., real-time traffic data, geographic information services) if such features are incorporated.
    - It is assumed that the APIs will remain free or available at a reasonable cost during the development and operational phases of the project.
3. **User Hardware and Software Environment**
    - It is assumed that end users will have access to a modern, web-capable browser (e.g., Chrome, Firefox, Safari) and devices that meet the minimum hardware requirements (4 GB RAM, 2 GHz processor) to run the system without performance issues.
    - It is assumed that the users will operate the system on supported platforms (Windows, macOS, or Linux) without the need for additional customization.
4. **Team Skillset**
    - It is assumed that the development team possesses the required skill set to work with the selected technologies (e.g., Node.js, JavaScript frameworks, GIS tools) and can deliver the project on time.
    - It is also assumed that the team will have access to the necessary development tools and software libraries needed to build and maintain the system.
5. **Data Accuracy**
    - The system assumes that all data provided by users, such as city locations, important landmarks, and road networks, are accurate and up-to-date. Any inaccuracies in user-supplied data may result in incorrect route optimizations.

Dependencies

1. **Third-Party GIS Tools and APIs**
    - The project is dependent on **third-party geographic information systems (GIS)**, APIs, and data sources for importing, displaying, and analyzing map data. If these services experience outages, changes in pricing, or policy changes, the project may be affected.
    - The dependency on **Google Maps, OpenStreetMap, or similar mapping services** may also impose limitations on the volume of requests or specific features that can be accessed.
2. **Backend Framework (Node.js)**
    - The project is dependent on the stability and continued support of the **Node.js** or **Python** backend framework for handling server-side operations and route calculations.

- Any critical updates, deprecations, or security vulnerabilities in the selected backend technology could delay development or require significant rework.

3. **Database System**
    - The system relies on the performance and scalability of relational databases (e.g., **PostgreSQL** or **MySQL**) to store map data, user configurations, and route information.
    - Any outages, capacity issues, or migration problems with the database system could impact the project's ability to handle large datasets or scale to support more users.
4. **Real-Time Traffic Data (If Implemented)**
    - If the system incorporates **real-time traffic data**, it will depend on external sources for up-to-date information on traffic conditions. The availability and accuracy of this data may vary depending on the provider and could affect the route calculations.
5. **Licensing of Mapping Services**
    - The system depends on licensing agreements or pricing plans for third-party map services. Changes in pricing models or usage limitations could impact the cost of running the application or restrict certain features for users.
6. **Internet Connectivity**
    - The system assumes that users will have stable internet access to utilize real-time data, cloud-based services, and API requests. Limited or intermittent connectivity may affect the system's performance, especially in retrieving map data or updating routes.

By identifying these assumptions and dependencies, the project team can proactively monitor potential risks and ensure that any changes to these factors are promptly addressed to minimize disruptions to the project's development and delivery.

3. System Features

3.1 Feature: Map Upload and City Selection

3.1.1 Description and Priority

Description:
This feature allows users to upload a map in supported formats (e.g., GeoJSON, KML) and select the cities they want to connect through a road network. It is the foundational step for route planning and is essential for the operation of the system.

Priority: High
Benefit: 9
Penalty: 7
Cost: 4
Risk: 6

3.1.2 Stimulus/Response Sequences

- **Stimulus**: The user clicks the "Upload Map" button and selects a valid map file.
- **Response**: The system validates the file and displays the map with city markers.
- **Stimulus**: The user selects multiple cities to be connected via road.
- **Response**: The system highlights the selected cities on the map and confirms the selection.

3.1.3 Functional Requirements

- **REQ-1**: The system shall allow users to upload maps in the supported formats (GeoJSON, KML).
- **REQ-2**: The system shall validate the uploaded file for correct format and structure.
- **REQ-3**: The system shall display an error message if the file is invalid or incompatible.
- **REQ-4**: The system shall visually render the uploaded map in the interface.
- **REQ-5**: The system shall allow users to interactively select cities on the map for route planning.
- **REQ-6**: The system shall highlight the selected cities once chosen.
- **REQ-7**: The system shall display a confirmation dialog once the cities have been selected for routing.

3.2 Feature: Route Optimization

3.2.1 Description and Priority

Description:
This feature calculates the most optimal route for road construction between the selected cities, considering various factors like distance, key landmarks, and terrain. It provides the core functionality of the system.

Priority: High
Benefit: 9
Penalty: 8
Cost: 5
Risk: 6

3.2.2 Stimulus/Response Sequences

- **Stimulus**: The user clicks the "Optimize Route" button after selecting cities.
- **Response**: The system computes the most efficient route between the cities and displays it on the map, along with the calculated distance and estimated cost of construction.
- **Stimulus**: The user changes the selection criteria by adding or removing a landmark (e.g., hospitals, airports).
- **Response**: The system recalculates the route based on the new parameters and updates the display.

3.2.3 Functional Requirements

- **REQ-1**: The system shall compute the shortest route between the selected cities using Dijkstra's or a modified algorithm.
- **REQ-2**: The system shall allow users to add or remove important landmarks such as hospitals, government buildings, or airports to be included in the route calculation.
- **REQ-3**: The system shall visually display the optimized route on the map.
- **REQ-4**: The system shall provide an estimated distance and cost for the selected route.
- **REQ-5**: The system shall recalculate the route dynamically when the user adjusts the selection criteria (e.g., landmarks or city choice).
- **REQ-6**: The system shall provide feedback to users in the event of calculation errors or infeasible routes.

3.3 Feature: Traffic and Environmental Factors (Future Enhancement)

3.3.1 Description and Priority

Description:
This feature allows the system to integrate real-time traffic and environmental data (e.g.,

weather conditions) for route optimization. This will help ensure that the recommended route takes into consideration live factors that could affect the construction process.

Priority: Medium
Benefit: 7
Penalty: 5
Cost: 7
Risk: 8

3.3.2 Stimulus/Response Sequences

- **Stimulus**: The user enables the "Real-Time Data" option before route calculation.
- **Response**: The system fetches current traffic and environmental data and factors this into the route optimization.
- **Stimulus**: The user disables the real-time data option.
- **Response**: The system reverts to using static data for route calculation.

3.3.3 Functional Requirements

- **REQ-1**: The system shall integrate with third-party services to retrieve real-time traffic and weather data.
- **REQ-2**: The system shall allow users to toggle the use of real-time data in route calculations.
- **REQ-3**: The system shall display traffic and environmental warnings or considerations if they impact the recommended route.
- **REQ-4**: The system shall recalculate routes dynamically when real-time data is toggled on or off.

The functional requirements outlined for each feature ensure that the system fulfills its purpose of efficiently planning road networks while considering key factors such as user input, geographic information, and future enhancements with real-time data integration.

4. External Interface Requirements

4.1 User Interfaces

The **Pathfinder AI** will provide a user-friendly interface for interacting with the map, selecting cities, optimizing routes, and visualizing data. Below are the key elements and characteristics of the user interface:

1. ***General Layout***

- **Map View**: The central part of the screen will be dedicated to displaying the uploaded map, with city markers and routes. Users can zoom in/out and drag the map for easier navigation.
- **Sidebar**: A collapsible sidebar on the left side will contain input fields, buttons for uploading maps, selecting cities, and optimizing routes. It will also have options for additional settings like selecting landmarks and toggling real-time data.
- **Header**: The header will display the application title and navigation buttons such as "Help," "Settings," and "User Profile."
- **Footer**: The footer will include links to terms of service, privacy policy, and contact information.

2. ***Standard Buttons and Functions***

- **Upload Map**: Button for uploading maps in supported formats.
- **Select Cities**: Button that allows the user to select multiple cities on the map for route planning.
- **Optimize Route**: Button that triggers the route optimization process.
- **Reset**: Resets the selection to start the process over.
- **Help**: A button that provides quick access to documentation and guidance for using the software.
- **Submit Feedback**: Allows users to provide feedback or report issues.

3. ***Keyboard Shortcuts***

- **Ctrl + U**: Opens the file dialog to upload a new map.
- **Ctrl + S**: Saves the current map view and selected cities.
- **Ctrl + R**: Resets the current selection.
- **Ctrl + Z**: Undo the last action (e.g., deselecting a city).

4. Screen Layout Constraints

- The map view must always be visible and responsive to different screen sizes.
- Buttons and user inputs must follow a clean and consistent design, ensuring accessibility for all users, including those with disabilities (e.g., screen readers, large buttons for touch interfaces).

5. Error Message Display Standards

- All errors (e.g., file upload issues, invalid city selection) will appear in red text within a popup window that overlays the main screen.
- Error popups will include clear instructions on how to resolve the issue, along with a "Close" or "Retry" button.
- Notifications for successful actions (e.g., "Map uploaded successfully," "Route optimized") will appear as green popups in the lower right corner of the screen.

6. User Interface Components

1. **Map View**: Core visual interface to display the uploaded map and selected cities, supporting drag-and-zoom features.
2. **City Selection Tool**: Allows users to click and select cities directly from the map.
3. **Route Visualization**: Displays the optimized route between selected cities, highlighting important places like hospitals or government buildings.
4. **Real-time Data Integration Toggle**: Switch to enable or disable the use of real-time traffic or weather data during route optimization.
5. **Interactive Messages**: For errors, notifications, and confirmations, ensuring users are always aware of the system's status.

Details of specific design guidelines will be included in a separate user interface specification document.

4.2 Communications Interfaces

Although this application is designed to function offline, several local communication interfaces may still be relevant:

1. **Local File System Access**:
    - The application will need to read from and write to the local file system for tasks such as storing map data, user configurations, or logs. Supported file formats may include JSON, CSV, or proprietary formats specific to the application.

2. **Internal Messaging**:
    - If the system is designed with modular components, internal messaging techniques may be employed to enable communication between these subsystems. This could include methods for sharing data or coordinating tasks across different parts of the application.
3. **Application Programming Interface (API)**:
    - Although the application primarily operates offline, it may expose a local API for internal components to communicate with each other or for future integration with online services. This API could follow RESTful principles and allow for structured data exchange and interaction within the application, such as querying map data or submitting user inputs.

These interfaces ensure that the application can effectively manage local data and interact with any connected devices, while still maintaining its offline functionality.

5. Other Nonfunctional Requirements

5.1 Performance Requirements

The **Pathfinder AI** must meet several performance criteria to ensure fast, accurate, and user-friendly interactions. The following performance requirements detail how the system is expected to behave under normal and peak usage conditions:

1. Response Time

- **User Interaction**: The system should provide responses to user actions (e.g., selecting cities, generating routes) within **2 seconds** under normal conditions. For more complex operations like map uploads and route optimization, the system should respond within **5 to 10 seconds**.
    - **Rationale**: Quick responses are crucial to maintain user engagement and satisfaction, particularly when dealing with map navigation and route visualizations.
- **Error Handling**: If an error occurs (e.g., invalid input or server issues), the system should notify the user within **3 seconds** with appropriate error messages.

2. Data Processing Speed

- **Route Calculation**: The optimal route planning algorithm should process and compute the best route between selected cities within **5 seconds** for small datasets (up to 5 cities) and within **10 seconds** for larger datasets (up to 50 cities).
    - **Rationale**: Users will expect timely results when calculating routes for important infrastructure projects. Delays in the optimization process could negatively affect the overall user experience.

3. Concurrent Users

- **Scalability**: The system must support at least **100 concurrent users** under peak load without degradation in performance.
    - **Rationale**: Given the potential for multiple users (e.g., government officials, planners) accessing the system simultaneously, it is crucial that the application handles concurrent operations without reducing the speed of map rendering or data retrieval.
- **Simultaneous Requests**: The system must handle at least **50 simultaneous route calculation requests** without exceeding the defined response time limits.

4. Data Throughput

- **File Upload and Download**: The system must support the upload and download of large map files (up to **100MB**) with minimal lag, taking no more than **30 seconds** on a standard broadband connection (10 Mbps).
    - **Rationale**: Since city maps and other geospatial data files can be large, the system must handle these operations efficiently to prevent user frustration.

5. System Load

- **Peak Load Handling**: The system should be able to handle a peak load of **500 API requests per minute** across all features (map loading, route calculation, data retrieval) without significant performance degradation.
    - **Rationale**: Ensuring peak performance under heavy load is critical for public-facing systems, especially when government officials or infrastructure planners might be using the system during critical times.

6. Database Performance

- **Query Speed**: Database queries (e.g., retrieving city data, route histories) should execute within **1 second** for standard searches and no longer than **5 seconds** for complex queries involving multiple criteria.
    - **Rationale**: A fast database response is crucial for providing users with a seamless experience when accessing historical route data or performing multi-city searches.

7. Real-Time Synchronization

- **Map Updates**: The system should display real-time map updates and changes (e.g., when users modify routes or add new cities) within **1 second** of the change being made.
    - **Rationale**: For a planning tool, real-time synchronization is essential to keep all users on the same page and ensure that data is accurately reflected across sessions.

8. System Uptime

- **Availability**: The system must have an uptime of **99.9%**, allowing for minimal downtime and ensuring it is accessible at all times, especially for urgent infrastructure planning.
    - **Rationale**: High availability is critical for public or governmental systems that might be used for critical operations, and prolonged downtime could lead to significant project delays.

By adhering to these performance requirements, the system will provide a reliable, scalable, and responsive experience for users involved in route planning, ensuring smooth operation even under demanding conditions.

5.2 Safety Requirements

The **Pathfinder AI** must include measures to ensure the safety and security of data and operations. This section outlines the safeguards and requirements to prevent harm, damage, or data loss during system usage, as well as compliance with external safety standards.

1. Data Integrity and Protection

- **Data Loss Prevention**: The system must implement regular data backup protocols, ensuring that all user data, route calculations, and uploaded maps are automatically backed up every **12 hours**. In case of system failure, these backups will be used for recovery.
    - **Rationale**: Given that the system will be handling sensitive city infrastructure data, loss of this information could cause significant setbacks for ongoing projects.
- **Safeguard**: Automated rollback mechanisms must be in place in case of system crashes or disruptions to avoid corruption or loss of data.

2. Regulatory Compliance

- **Data Privacy**: The system must comply with privacy regulations, including **GDPR** (General Data Protection Regulation) and **Indian IT Act 2000** (for user data protection and privacy). Sensitive data like user credentials, project information, and map data must be encrypted during transmission (using **SSL/TLS**) and storage.

- **Rationale**: Since the system will handle potentially sensitive data related to national infrastructure, regulatory compliance is essential to ensure user privacy and data security.

3. System Access and Authorization

- **Role-Based Access Control**: The system must implement strict role-based access controls (RBAC) to prevent unauthorized users from accessing sensitive data or performing critical operations (e.g., route modification or deletion). Only users with appropriate roles should be able to edit or delete maps and route information.
    - **Rationale**: Misuse of system features could lead to incorrect route planning or loss of important data, which could severely affect decision-making for infrastructure projects.
- **Prevent Unauthorized Changes**: System administrators must be able to review and approve route modifications before they are applied to the final project.

4. Physical System Security

- **Server Security**: The system servers must be hosted in a secure environment with **24/7 monitoring**, fire protection, and protection against physical tampering.
    - **Rationale**: Physical access to servers must be restricted to prevent any hardware-based attacks or data breaches.

5. System Failures and Recovery

- **Failure Detection**: The system should have mechanisms to detect critical failures (e.g., system crashes, server overloads) and automatically notify administrators, with a maximum detection time of **30 seconds**. Immediate action must be taken to minimize downtime.
    - **Rationale**: Continuous system uptime is critical for infrastructure planning and analysis. Failure detection mechanisms will allow administrators to react swiftly and avoid data loss.
- **Recovery**: In the event of a failure, the system should be able to restore the last saved state within **10 minutes** from the backup to minimize operational delays.

6. Security Auditing

- **Audit Logs**: The system must maintain detailed audit logs of user activities, such as logins, data uploads, route creation, and deletion activities. Logs must be tamper-proof and maintained for a minimum of **1 year** to assist in identifying and investigating any safety incidents or unauthorized actions.
    - **Rationale**: In case of any harmful actions, having a complete audit trail is necessary to trace and prevent future incidents.

7. Malware and Threat Protection

- **Virus Scanning**: Any files uploaded (e.g., maps, documents) must be scanned for viruses and malware to prevent introducing harmful code into the system. The system should reject any files found to contain malicious content.
    - **Rationale**: Since users will upload files, malware could potentially compromise the system, risking data loss or system integrity.

8. Incident Reporting

- **Safety Certification**: The system must comply with safety certifications, such as **ISO 27001** for information security, to ensure adherence to best practices for security, risk management, and compliance with safety standards.
    - **Rationale**: Compliance with international standards for information security and safety practices will mitigate potential risks and ensure system reliability.

By meeting these safety requirements, the system will ensure the protection of data, prevent misuse, and maintain compliance with safety and regulatory standards, reducing the potential for harmful incidents and ensuring long-term operational security.

5.3 Security Requirements

The **Pathfinder AI** must implement robust security measures to protect user data, safeguard the system, and prevent unauthorized access or malicious activities. This section outlines the security requirements necessary to ensure that the system operates safely in compliance with privacy and security regulations.

1. User Authentication

- **Multi-Factor Authentication (MFA)**: All users accessing the system must go through a multi-factor authentication (MFA) process to ensure that only authorized individuals can log in. This could involve a combination of password-based authentication and one-time codes sent via email or SMS.
    - **Rationale**: MFA significantly reduces the risk of unauthorized access, even if passwords are compromised.
- **Password Policies**: Passwords must follow stringent complexity rules (minimum 8 characters, mix of upper/lowercase, numbers, and symbols) and must be changed every **60 days**.
    - **Rationale**: Strong password policies ensure that user accounts are not easily compromised.

2. Data Encryption

- **Encryption in Transit**: All data exchanged between the client (users) and the server must be encrypted using **SSL/TLS** protocols to prevent eavesdropping or man-in-the-middle attacks during communication.
    - **Rationale**: Data being transmitted across networks must be protected to avoid interception of sensitive information such as user credentials or route details.
- **Encryption at Rest**: Sensitive data, such as user credentials, uploaded maps, and saved route plans, must be encrypted using **AES-256** encryption at rest in the database.
    - **Rationale**: Encrypting stored data ensures that even if the database is compromised, the data remains inaccessible without proper decryption.

3. Access Control

- **Role-Based Access Control (RBAC)**: The system must support role-based access control, where users are assigned roles such as admin, user, or guest. Each role will have specific permissions, limiting access to data and functionalities based on their role.
    - **Rationale**: Implementing RBAC ensures that users can only access features and data relevant to their permissions, reducing the likelihood of data misuse.
- **Data Segmentation**: Each user or group (e.g., governmental agencies, urban planners) will only have access to the data they upload or are authorized to view. Users should not be able to access or modify data belonging to other groups.
    - **Rationale**: Prevents unauthorized access or tampering with other users' data, ensuring data privacy and integrity.

4. Audit Trails and Monitoring

- **Audit Logs**: The system must maintain detailed logs of user activities such as login attempts, changes to routes, and data uploads. These logs must be protected against tampering and regularly reviewed for suspicious activities.
    - **Rationale**: Audit trails enable administrators to track and investigate any unauthorized actions, helping in the identification of potential security threats.
- **Intrusion Detection and Prevention**: The system must integrate with intrusion detection/prevention systems (IDS/IPS) to detect and respond to malicious activities such as brute-force attacks or suspicious network traffic.
    - **Rationale**: Monitoring for malicious activities will help identify and neutralize potential threats before they escalate.

5. Data Privacy

- **Privacy Policies**: The system must comply with global privacy regulations, including **GDPR** and **India's Data Protection Bill**, to ensure user data is handled in accordance with legal requirements for privacy and confidentiality.
    - **Rationale**: Adhering to privacy regulations protects users' personal and sensitive information, preventing legal liabilities for mishandling data.
- **User Data Control**: Users should be able to view, modify, or delete their personal data, and be informed about how their data will be used. Consent must be obtained before storing any personal information.
    - **Rationale**: Providing users with control over their own data aligns with privacy principles and increases user trust in the system.

6. Security Certifications

- **Compliance with ISO 27001**: The system must adhere to the **ISO 27001** standards for information security management, ensuring that the product meets industry standards for managing sensitive company information.
    - **Rationale**: ISO 27001 certification demonstrates that the system follows best practices in security, mitigating the risk of data breaches and other security incidents.
- **Security Testing**: Regular penetration testing and security audits must be conducted to identify and resolve potential vulnerabilities in the system.
    - **Rationale**: Periodic testing ensures that the system remains resilient against evolving security threats.

7. Third-Party Dependencies

- **Secure Third-Party Components**: Any third-party libraries, APIs, or services used within the system (e.g., mapping APIs or machine learning tools) must be vetted for security and kept up to date with the latest security patches.
    - **Rationale**: Third-party components may introduce vulnerabilities if not regularly updated or properly secured.

8. Data Breach Response Plan

- **Incident Response Protocol**: In the event of a data breach or security incident, a documented incident response plan must be in place. This includes notifying affected users within **72 hours**, investigating the breach, and implementing corrective measures.
    - **Rationale**: Quick and efficient responses to data breaches can limit damage and maintain user trust.

By implementing these security requirements, the **Pathfinder AI** ensures the safety, integrity, and privacy of user data while maintaining compliance with industry security standards and legal regulations.

5.4 Software Quality Attributes

The **Pathfinder AI** must meet several key quality attributes that are important to both customers and developers. These attributes will ensure the system performs reliably, is easy to maintain, and provides a high level of usability and adaptability. The following quality attributes are defined to guide the development process and guarantee the desired outcomes:

1. Adaptability

- The system must be easily adaptable to changes in user requirements or environmental conditions, such as adding new route planning features, supporting additional types of maps, or integrating with other systems.
- **Metric**: The system should support at least 80% of feature changes or new requirements without significant architectural modifications.

2. Availability

- The system should be highly available, with a target uptime of **99.9%**, ensuring that users can access the platform at any time, except for scheduled maintenance windows.
- **Metric**: System downtime should not exceed 9 hours per year, and the system should provide notifications to users in case of planned maintenance.

3. Correctness

- The system must correctly compute optimal routes based on the data input by users and provide accurate distance, time, and resource estimates.
- **Metric**: Route calculation accuracy must be at least 99%, with a goal of returning correct results in at least 99 out of 100 test cases.

4. Flexibility

- The system must allow flexible input methods for data entry, such as map uploads, manual selection of points, and input from external systems like GIS.
- **Metric**: The system should integrate at least two methods of data input with ease, and changes to input types should be accomplished in under two development cycles.

5. Interoperability

- The system must integrate and communicate seamlessly with external applications, including map services (e.g., Google Maps, OpenStreetMap), government databases, or urban planning software.
- **Metric**: The system must be able to import and export data using at least two industry-standard formats (e.g., JSON, XML, GeoJSON).

6. Maintainability

- The system must be easy to maintain, allowing developers to modify, extend, or fix parts of the software without requiring complete system overhauls. Clear and modular code should be used to simplify troubleshooting.
- **Metric**: The mean time to resolve (MTTR) any system bugs or issues should not exceed **2 hours**, with high-priority issues resolved within 24 hours.

7. Portability

- The system must be portable across different platforms, supporting multiple operating systems (e.g., Windows, macOS, Linux) and deployment environments (cloud and on-premises).
- **Metric**: The system must be deployable in at least two different cloud environments (e.g., AWS, Azure) and run without modification on at least three different operating systems.

8. Reliability

- The system must consistently perform as expected under normal and peak loads, without crashes or data corruption.
- **Metric**: The system should be able to handle up to **1,000 concurrent users** with minimal performance degradation, maintaining at least 95% of its response time targets.

9. Reusability

- System components should be designed with reusability in mind, allowing future projects to leverage core features such as pathfinding algorithms or map integration modules.
- **Metric**: At least 30% of the codebase should be reusable in other applications or projects with minimal modifications.

10. Robustness

- The system must be robust and handle a wide range of input errors or unexpected user behavior, providing appropriate error messages or corrective measures.
- **Metric**: The system should handle invalid inputs (e.g., incorrectly formatted maps or coordinates) in at least 95% of cases, returning meaningful error messages without crashing.

11. Testability

- The system must be designed to be easily testable at all levels (unit, integration, system) to ensure continuous integration and delivery (CI/CD) processes can be applied efficiently.
- **Metric**: At least 80% of the system codebase must be covered by automated tests, with critical features having a minimum of 95% test coverage.

12. Usability

- The system must provide an intuitive and user-friendly interface, minimizing the learning curve for new users and providing clear guidance at each stage of the process.
- **Metric**: Usability tests should show that at least 90% of users can complete basic tasks (e.g., uploading a map, selecting cities, calculating routes) without needing external assistance or more than 5 minutes of training.

6. Other Requirements

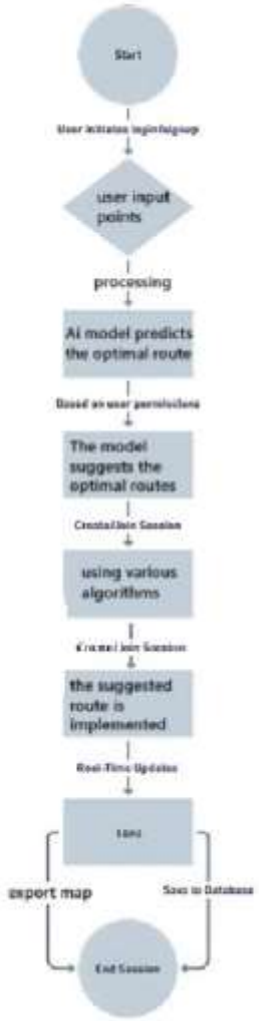
Appendix A: Glossary

- **API (Application Programming Interface)**: A set of tools and protocols that allows different software applications to communicate with each other.
- **DFD (Data Flow Diagram)**: A diagram that depicts how data flows within a system, illustrating the relationships between processes, data stores, and data sources/destinations.
- **ERD (Entity-Relationship Diagram)**: A diagram used to visually represent the relationships between entities (objects) in a database.
- **GIS (Geographic Information System)**: A system designed to capture, store, manipulate, analyze, manage, and present spatial or geographic data.
- **SRS (Software Requirements Specification)**: A document that describes the functional and non-functional requirements for a software product.
- **TBD (To Be Determined)**: A placeholder for information that is not yet available.
- **UI (User Interface)**: The space where interactions between humans and machines occur, allowing users to control the system and receive feedback.

Appendix B: Analysis Models

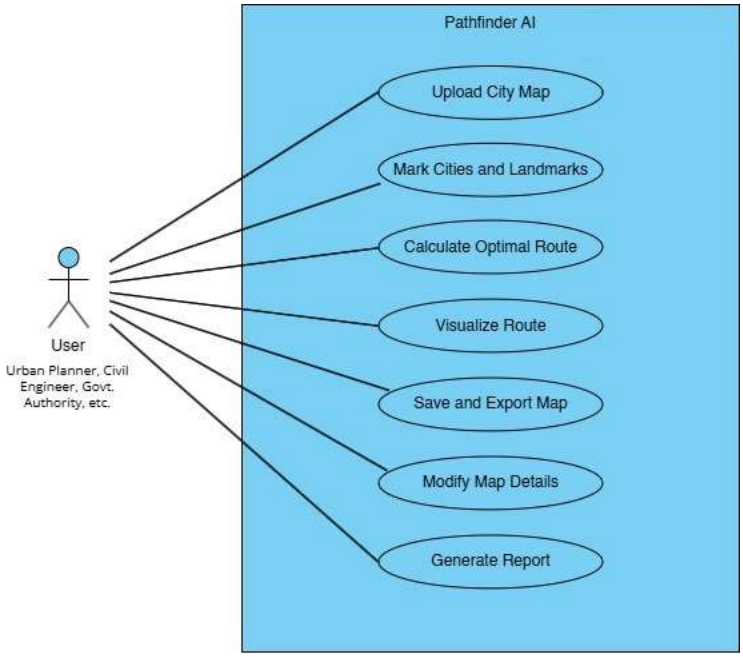
User Flow Diagram:

A DFD illustrating how data moves through the **Pathfinder AI**, from user inputs such as map uploads and city selection to the system's pathfinding algorithm and outputs of the optimal route.



 Use Case Diagram:

The **Use Case Diagram** for the Pathfinder AI System illustrates the interactions between the system and its users, including urban planners, civil engineers, and government authorities. It defines how users will interact with the application through key functions such as uploading city maps, marking cities and landmarks, and calculating optimal routes. The diagram highlights the system's core features, including map upload, user input for route calculations, and the visualization of the optimized routes. It provides a clear overview of the user's role in the system and the system's functionality in delivering efficient route planning for infrastructure projects.



Appendix C: Issues List

- **TBD-001**: The specific communication protocol for third-party map service integration has not yet been determined.
- **TBD-002**: Pending decision on the final data format for user-uploaded maps (e.g., GeoJSON, KML, or another format).
- **TBD-003**: Confirmation of performance requirements under peak load conditions (e.g., 1,000 concurrent users).
- **TBD-004**: Finalization of security requirements concerning data encryption and user authentication mechanisms.
- **Pending Decision-005**: Clarification required on the exact scope of supported map services (Google Maps, OpenStreetMap, etc.).
- **Conflict-006**: Potential conflict between user-friendliness and advanced customization features; resolution needed to determine the balance between ease of use and functionality.
- **Information Needed-007**: Details on the operating systems where the system will be deployed for compatibility testing.
- **Pending Decision-008**: Decision needed on whether to support offline map functionality, affecting portability requirements.