Eagle Maps – Technical Documentation

1. **PROJECT OVERVIEW**

* Project Name: Eagle Maps
* Description: A c++ command-line navigation tool built to enable students to efficiently navigate the University of Southern Mississippi campus. The program allows users to find the shortest path, view nearby locations, plan out their schedule using muti-building routes and update the campus map.
* Contributors: Pearl Esezobor, Cody Boudreaux, Brian Perry, Riley Wilson
* Tech Stack: C++, HTML, Custom Graph Implementation.

1. **SYSTEM ARCHITECTURE**

Eagle maps is divided into 4 core modules:

* Main.cpp: Entry point that creates and runs an instance of EagleMaps
* Eaglemaps.h: Declares the EagleMaps class which encapsulates all application logic and serves as the controller of the entire navigation system. It includes functions like:  
  - InitializeMap (\_) - which builds the graph with nodes and connections.
* findShortestPath (), planScheduleRoute () – which are interactive features
* modifyMap () – allows dynamic updates to the graph (add/remove nodes)
* Eaglemaps.cpp: Implements all the logic defined in eaglemaps.h including building setup with alias, shortest path logic using Dijkstra’s algorithm, menu navigation and distance-based searches.
* Graph.h: custom graph structure implementation using a Node class logic. It includes:
* Node class which represents campus buildings
* Graph class that manages connections, path finding using Dijkstra’s algorithm, aliases and route planning. This is also the part of the program where all CRUD operations are implemented.

1. **FEATURES**

* **Display All Buildings**  
  Lists all current buildings in the system.
* **Find Shortest Path**  
  Uses Dijkstra’s algorithm to find the shortest walking route between two buildings.
* **Plan a Schedule Route**  
  Generates a route through multiple buildings in a specified order (ideal for class schedules).
* **Find Nearby Buildings**  
  Lists all buildings within a given radius (miles) of a selected location.
* **Map Modification**  
  Dynamically add or remove buildings and their connections.
* **Campus Map Display**  
  Lists all connections in the graph with distances and times.

1. **DATA STRUCTURES**

* Node Structure (in graph.h): Each building is represented as a node

A screen shot of a computer code

AI-generated content may be incorrect.

* Graph Structure: which stored all buildings and their connections and supports bi-directional edge creation, alias mapping and graph display.
* EagleMaps Class: Contains main logic flow and manages user interaction.

1. **INSTALLATION AND SETUP**

Requirements:

* C++ compiler supporting C++ 17
* A terminal or an IDE preferably Visual Studio Code.

1. **USAGE GUIDE.**

Upon launching the program, the user will be met with an interactive menu:

A black screen with white text

AI-generated content may be incorrect.

Input is numeric and this tool is necessary to validate entries and provide user friendly prompts.

1. **TESTING AND DEBUGGING**

* Manuel testing was done for each feature through the CLI, and every function/ method was tested as implemented.
* Input validation for edge cases
* Console outputs to confirm path results and the integrity of the graph

1. **CHALLENGES AND SOLUTIONS**

* **Ambiguous building names**  
  Solved using alias mapping (Alias Map) so "Thad", "Student Union", and "Thad Cochran Center" all resolve correctly.
* **Graph updates in real time:** Created functions like addNewBuilding () and remove Building() with user-friendly prompts.
* **User experience via terminal**  
  Enhanced with clear formatting, route arrows (→), and clean menus.

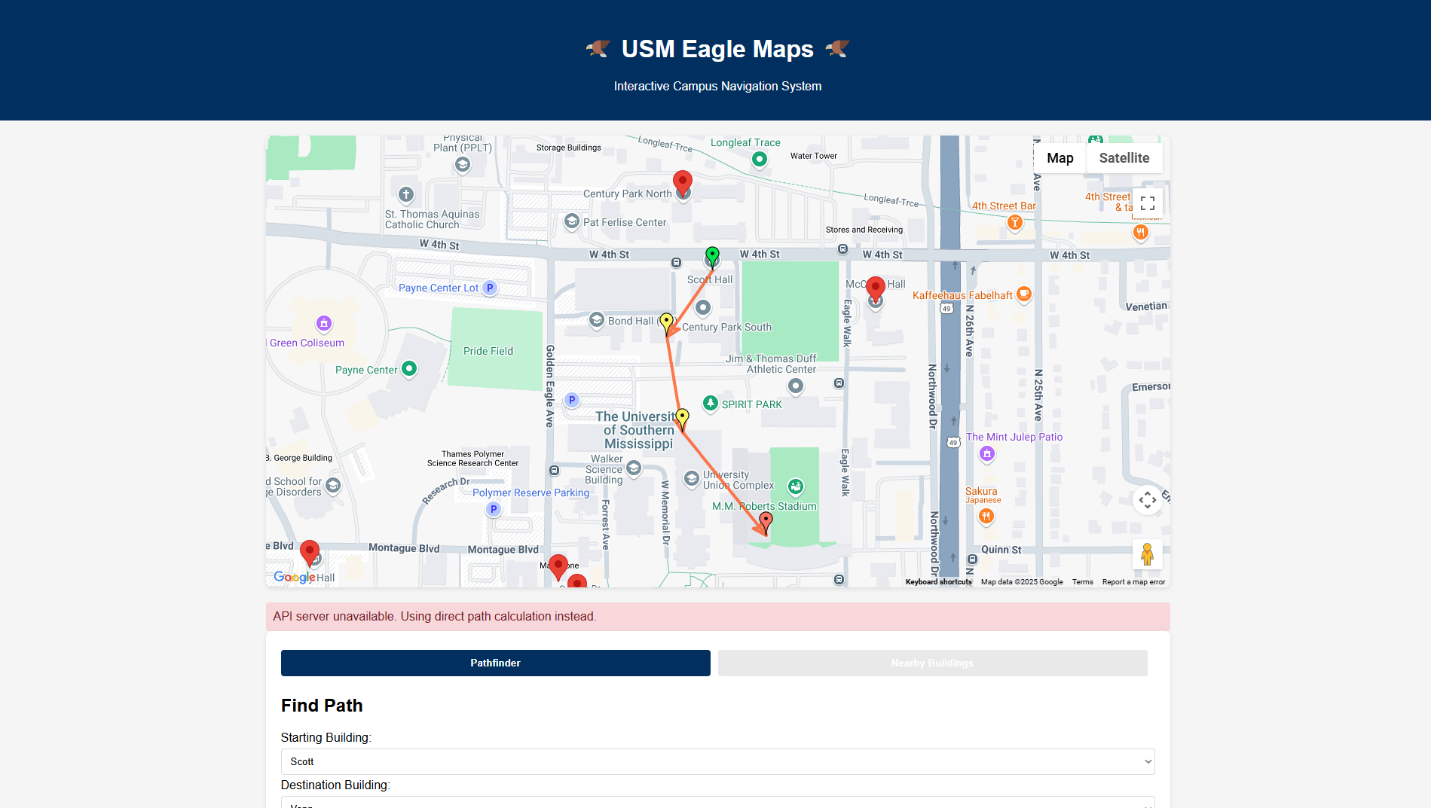
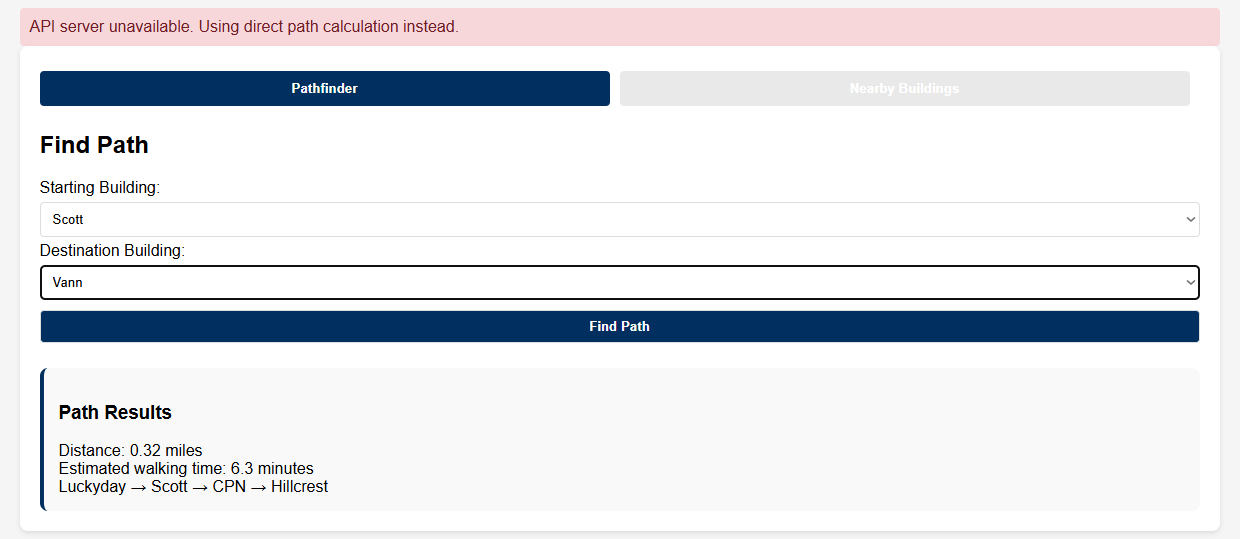
1. **Overview of USM Eagle Maps Interface**

**The images show the USM Eagle Maps interface, an interactive campus navigation system for the University of Southern Mississippi. This application provides navigation assistance for students to find their way around campus.**

**Path Finding Functionality**

**These images show the path finding feature in action:**

* **Users can select a starting building (set to "Scott" in the examples)**
* **Users can select a destination building (set to "Venn" in the examples)**
* **After clicking "Find Path," the system displays path results showing:** 
  + **Distance: 0.32 miles**
  + **Estimated walking time: 6.3 minutes**
  + **Route description: Luckyday → Scott → CPN → Hillcrest**

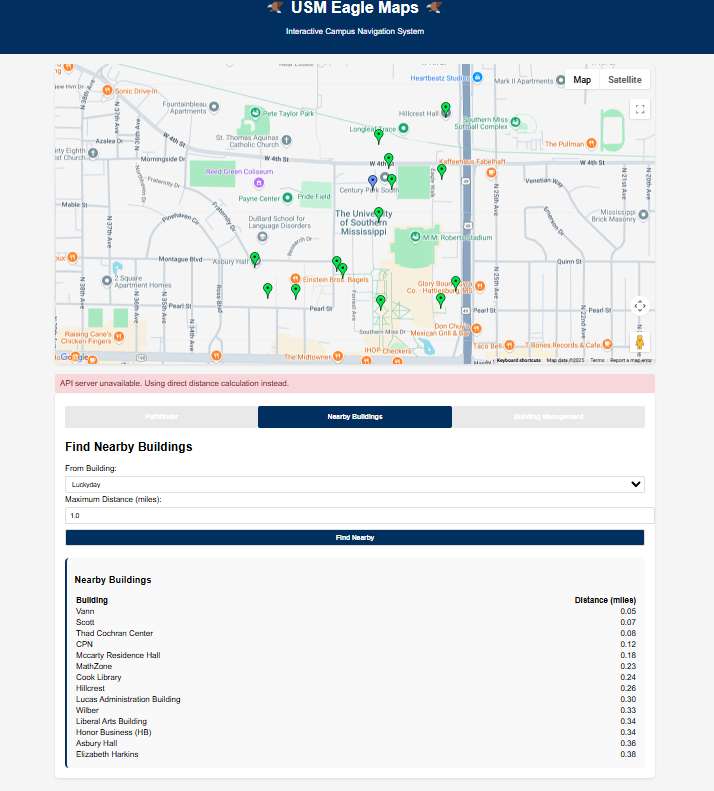
 

**Nearby Buildings Feature Interface**

The image shows the USM Eagle Maps interface with the "Nearby Buildings" tab selected instead of the "Pathfinder" tab from the previous images. This feature allows users to find buildings within a specified distance of a selected starting point.

Key elements visible in this interface:

* The map displays the USM campus with multiple building locations now marked with green pins (rather than red pins seen in the pathfinder view)
* The same API error message appears: "API server unavailable. Using direct distance calculation instead"
* The interface includes:
  + A dropdown to select the "From Building" (currently set to "Luckyday")
  + A field to enter "Maximum Distance (miles)" (set to 1.0)
  + A "Find Nearby" button to execute the search



**Building Management Interface**

The final image shows the building management interface where administrators can:

* Add new buildings by providing:
  + Building name
  + Latitude and longitude coordinates
* Remove existing buildings through a selection dropdown
* A success message is shown: "Building 'Johnson Science Tower (JST)' removed successfully!"

This management interface allows the campus map to be dynamically updated, adding or removing buildings as the campus evolves - demonstrating the application's flexibility mentioned in the technical documentation where "Map Modification" was listed as a key feature.

