08 – T2 – Accessible Interactive Map

Sharon Perry, 04/28/2025

CS 4508, Spring 2025

Team Members: Justin Connick, Megan Ingram, Derrick Novak, Spencer Williams, Emily Zhu

Website: https://adamapdev.github.io/FinalReportWebsite/

Github: https://github.com/ADAMapDev/adaMapDevSite

Numbers of lines of code: 3,200+

Number of project components/tools used: 10+

Estimated total man hours: 723

Actual total man hours: 224

Table of Contents

Introduction

The Accessible Interactive Map project is a web-based navigation tool developed by a student team at Kennesaw State University. Designed with accessibility and user personalization in mind, the application enables users, especially those with physical or visual impairments, to plan optimal routes between buildings on campus. The tool integrates with APIs such as Google Maps and Open-Source Routing Machine (OSRM) to generate routes while also offering features like elevation tracking, estimated travel time, and lifestyle data such as calories burned.

This application was created to meet real-world accessibility challenges and provide inclusive technology to individuals navigating large and complex environments. The project utilizes a modern tech stack including React for the frontend, Flask for the backend, and PostgreSQL for the database. It also employs responsive design principles to ensure compatibility across desktops, tablets, and mobile devices. Future versions of the application may include features like saved routes, two-factor authentication, and real-time air quality metrics.

Requirements

Reversion History

Name	Date	Reason for Changes	Version
Spencer	04-19-2025	Changed reqs and tools used removed accessibility guidelines and standards	1.1

Introduction

1.1 Overview

The Accessible Interactive Map project is a web-based application with an interactive map where users can select start and end points to see optimized routes. The primary focus is accessibility for those with disabilities and user personalization according to lifestyle and goals. Although geared towards Kennesaw State University, the hope is to expand to other institutions.

1.2 Project Goals

The project aims to:

- Develop an accessible, web-based application

- Develop a user-friendly interface for selecting points and displaying routes
- Use existing APIs for route optimization (such as Google Maps API)
- Have cross-platform capabilities and easy configuration management
- Implement user management functionality to store preferences such as disability types and lifestyle goals
- Offer lifestyle tips such as calorie counts for different routes
- Meet accessibility standards with screen reader compatibility, high contrast modes, and keyboard navigability

1.3 Definitions and Acronyms

API = Application Programming Interface

CSS = Cascading Style Sheets

HTML = Hypertext Markup Language

KSU = Kennesaw State University

SQL = Structured Query Language

OSM = Open Street Maps

UI = User Interface

1.4 Assumptions

It is assumed the web browsers support HTML5, CSS3, and JavaScript standards. It is also assumed that they are compatible with accessibility technologies like screen readers.

It is assumed that any third-party APIs will remain free of cost and readily available for future software updates.

2.0 Design Constraints

2.1 Environment

The web-based and mobile-friendly software system functions as a navigation system to provide lifestyle guidance to users. Users will access this system through contemporary browsers available on various devices such as desktops, tablets, and smart phones. Users can access the system through any device that can support HTML, CSS3, and JavaScript. Individuals who are frequent KSU compose the main target audience for this system since

route optimization functions and accessibility features serve them best, but global expansion possibilities exist based on future data availability.

The software system will run from a dedicated server provided by Kennesaw State University for reliability and growth purposes. A relational database platform among PostgreSQL or MySQL will manage user information composed of account information along with preferred routes and accessibility preference.

User-generated routes through Google Maps API or OpenStreetMap generate route calculations the system obtains from these external mapping services to provide real-time navigation assistance. The system will connect to screen reader accessibility software so visually impaired users can utilize its features. The user interface combines responsive design to accommodate web and mobile platforms alongside high contrast features, keyboard controls, and adjustable accessibility tools to maximize usability for every user type.

2.2 User Characteristics

This application targets KSU students, especially those with walking and seeing disabilities, as its main audience. Users with disabilities can benefit from the system because it provides special routing options, disability-related features, and personalized lifestyle recommendations. The application supports users of all technical abilities who study at KSU through a user experience that remains intuitive for everyone.

Wheelchair users along with other individuals who need wheelchairs or crutches can access the system with ease because it shows accessible routes which lead through ramps and elevators alongside proper curb cuts without options for staircases and mobility limitations. User will receive instant data about disabled accessibility status and wheelchair route timing estimations through the system.

Visual impairment users can take advantage of descriptive features that make usability better through screen readers, high-contrast modes, and text-to-speech functions. Users who exclusively use assistive technologies will find priority access to keyboard operation as an alternative to traditional touch of mouse input.

The solution focuses primarily on accessibility but contains optional features which allow fitness enthusiasts to record their walking activities like step counts and calorie expenditure as they move on campus. The system interface will maintain a basic, user-friendly design for those with different technological proficiency levels.

The application will advance to serve new institutions while preparing to accept public use through ongoing developments that honor accessibility needs alongside inclusive principles.

2.3 System

A web-based application, along with a mobile-friendly navigation platform, exists to boost accessibility while directing users toward lifestyle choices. Multiple technologies will be incorporated into one system to enable smooth accessibility between desktops, laptops, tablets, and smartphones. As a main functionality, the platform serves users by helping them find optimized routes with built-in accessibility considerations during planning.

Core Components:

1. User Interface (UI):
    - Designed using HTML, CSS3, and JavaScript to support responsive web design.
    - Optimized for accessibility with high-contrast modes, keyboard navigation, and screen reader compatibility for visually impaired users.
    - Offers an intuitive and minimalistic interface to accommodate users of all technical proficiency levels.
2. Backend System:
    - Built using server-side technologies like Python (Django/Flask), Node.js, or PHP for handling business logic and data processing.
    - Utilizes PostgreSQL as the relational database for storing user accounts, preferred routes, accessibility preferences, and tracking lifestyle-related data.
3. Mapping & Navigation Services:
    - Integrates with Google Maps API or OSM for real-time route calculation and accessibility-based adjustments.
    - Provides multiple route options based on user preferences, including wheelchair-accessible paths and step-free navigation.
    - Displays real-time updates on accessibility status for elevators, ramps, and other relevant infrastructure.
4. Accessibility & Assistive Technologies:

- Supports screen reader accessibility software to assist visually impaired users with textual and navigational elements.
- Includes adjustable text sizes, customizable color contrast settings, and text-to-speech functionality.
- Offers keyboard shortcuts and voice command support for users with limited mobility.

5. Lifestyle & Fitness Tracking:
    - Users can set personal goals related to steps taken and calories burned.
    - The system dynamically updates caloric expenditure based on route selections and walking distance.
    - Provides insight into lifestyle trends and activity levels over time.
6. Hosting & Deployment:
    - Hosted on dedicated Kennesaw State University servers to ensure reliability and scalability.
    - Designed to support future expansion to other institutions and public use cases.
    - Implement routine backups, security patches, and server monitoring for optimal performance.

The system aims to unite accessibility advantages with operational convenience by delivering inclusive navigational support to students with disabilities and promoting their choice of health-promoting practices. Future development will involve the incorporation of artificial intelligence for route suggestions alongside immediate community density monitoring and expanded open accessibility capabilities starting outside the university environment.

3.0 Functional Requirements

3.1 Login

Enter username/email

Enter password

Password recovery link

Click button to login

3.2 Register / Create Account

Enter first and last name

Enter email address

Enter phone number (optional)

Enter username

Enter password (minimum 8 characters with at least 1 capital letter, 1 lowercase letter, and 1 special character or number)

Click button to register

3.3 Home Page

User types in a search bar to select start and end destinations

Display optimized route and other route options

Display lifestyle advice per route (calories burn per route)

Display time each route takes by walking or school bus / car

Click Accessibility Button

Click Lifestyle Button

3.4 Map Display

Small display incrementing calories burned and steps taken (vs set goals as a ratio)

Show precise, readable directions

Show map image

Show ETA

Click button to exit navigation

3.5 Accessibility Button

Multi-select checkboxes for selecting disability types

Accessibility options

Screen reader compatibility

High contrast modes

Keyboard navigability

3.6 Lifestyle Button

Multi-select checkboxes for selecting lifestyle advice options

Users set their own goals (steps and calories goals per day/month/year)

3.7 Account Page

View and edit personal info

Update password

Update Preferences

4.0 Non-Functional Requirements

This section outlines the necessary characteristics which design the accessible interactive map system to fulfill. These specifications cover several critical domains which include security together with capacity and usability and performance and reliability and maintainability features for meeting dual technical and user-focused targets.

4.1 Security

The system needs to implement comprehensive security procedures which keep user preferences together with account-related information secure. Only authorized user accounts holding authentication information alongside strong passwords will gain access to protected data. Data exchange will occur between client devices and Kennesaw State University's official servers.

4.2 Capacity

The system's capacity depends directly on the capacity of the server infrastructure operated by the school. The system currently operates on the existing server infrastructure, which must be designed to expand its capacity for increasing campus data amounts and user base expansion. When the user base increases, the system demonstrates its capability to maintain stable performance even during higher levels of concurrent operations. The system's dependence on third-party APIs leads to expanding

expenditure costs as user numbers grow because the system uses various third-party APIs for its features. The budget and operational costs will experience an impact from increased user numbers because API expenses increase in this situation. The system needs efficient resource management and load balancing strategies, along with additional protocols for cost optimization through API pricing tier negotiation or data caching techniques, to manage increased API call expenses. Long-term sustainability requires future capacity planners to strike a balance between technology scalability and the current expense budget for effective planning.

4.3 Usability

The interactive map prioritizes accessible usability features as its core design component. Visual components will be designed for maximum visibility through high contrast designs. We will focus on ensuring that our colors make text, buttons and the map clearly identifiable. Different components like buttons, while being clearly labeled with icons to help users identify the functions of the buttons.

5.0 External Interface Requirements

5.1 User Interface Requirements

User interface requirements include accessibility features, interaction with the map, user inputs, how the app reacts to various screen sizes (phone, desktop etc.), and error handling. For accessibility features, we want to add more support for users such as simple navigation with clearly labeled controls. Having stable interaction with the map is important and we want to add the ability to zoom in, out, and drag the map. User inputs will allow users to use a drop-down menu or input fields to filter to make it easy to find what they are looking for. If the user runs into an error or issue, there should be pop up message with the error stated as well as guidance that the user can make it work.

5.2 Hardware Interface Requirements

Starting off, the app should be available and supported on all Windows and Mac laptops. For future versions, we would like to add GPS integration so that users are able to see their location in real time to help navigate more easily as well as making this app accessible on phone and tablet devices.

5.3 Software Interface Requirements

The app should be available and supported on standard web browsers on desktop and mobile. We will be using external APIs like Google Maps, and OSM. This app will use React to support accessibility standards.

5.4 Communication Interface Requirements

The application will utilize established communication protocols to ensure secure and efficient data exchange among users, external APIs, and internal components. All data transmissions will occur over secure channels such as HTTPS, with information formatted in JSON to maintain consistency and simplify integration across various systems. Interactions with external services, including Google Maps and accessibility tools, will be facilitated through RESTful API calls that support standard HTTP methods. This design will include basic error handling measures—such as retry mechanisms and clear user notifications—to manage potential service disruptions.

In addition to these measures, the application will incorporate strategies to handle real-time data requirements. Internally, well-defined interfaces will guide communication between application modules and entities and ensure that updates to one component do not negatively impact overall performance. This integrated approach to communication is designed to deliver a secure, reliable, and scalable user experience.

Design

1.0 Introduction and Overview

1.1 Document Outline

This document provides a comprehensive design framework for the Accessible Interactive Map project, which aims to create a web-based application that enables users to select starting and ending points on an interactive map, view optimized routes, and access lifestyle-related information. The primary goal is to ensure that the application is accessible to individuals with varying disabilities and adaptable for use in different institutions, beginning with Kennesaw State University (KSU) with hopes to expand to other areas.

1.2 Document Description

1.3 Introduction

The Accessible Interactive Map project addresses the need for an inclusive navigation tool designed to accommodate diverse user needs, particularly those with disabilities. This Software Design Document (SDD) outlines the design components, technical specifications, and functional requirements necessary to implement and maintain the system. The intended audience includes software developers, quality assurance teams,

project managers, and stakeholders invested in promoting accessibility and enhancing user experience.

1.4 System Overview

The Accessible Interactive Map project is designed as a responsive, web-based platform powered by modern web technologies and frameworks such as Flask. The system will provide users with:

1) Interactive Routing: Users can select start and end points on an accessible map and view routes optimized based on user preference and physical accessibility.
2) Lifestyle Recommendations: Features such as calorie count for different routes will support fitness and weight management goals.
3) User Personalization: A user management system will allow storing preferences like types of disabilities and lifestyle goals to tailor the application experience.
4) Device Compatibility: The platform will function seamlessly across desktops, tablets, and mobile devices.

The document does not cover elements such as project scheduling, cost estimates, or testing procedures, as these are addressed in separate project management and quality assurance plans. Prototypes, user guides, and user interface storyboards will serve as supplemental reference and are documented separately.

2.0 Intended Audience and Reading Suggestions

This document is tailored for the following groups:

- Developers: To guide the implementation of the system's technical architecture.
- Testers: To understand the functional requirements and test scenarios.
- Project Managers: To ensure alignment between system design and project goals.
- Stakeholders and Clients: To validate that the design meets their requirements and expectations.

Readers are advised to begin with the system overview to understand the project's objectives and proceed to detailed design components relevant to their roles. Sections on architecture and design specifications are particularly useful for technical teams.

3.0 Product Scope

The Accessible Interactive Map project's main objective is to create an inclusive, user-friendly navigation tool. The project aligns with institutional goals to foster accessibility and support lifestyle improvement. By offering adaptable configurations, the project will pave the way for broader application practices in different settings, promoting universal design principles.

3.1 References

- Accessible Interactive Map Project Proposal (2025-01-20)
- Flask Web Framework Documentation

4.0 Design Considerations

4.1 Assumptions and Dependencies

Our project relies on the following assumptions and dependencies:

1) Related Software or Hardware:
    a) The platform assumes integration with web browsers supporting modern HTML5, CSS3, and JavaScript Standards, with React as the primary front-end framework.
    b) Compatibility with assistive technologies such as screen readers and voice control systems.
2) Operating Systems:
    a) The application is designed to be platform-agnostic, accessible via standard web browsers on both desktop and mobile devices (i.e. Microsoft Edge, Chrome, Firefox, Safari)
3) End-User Characteristics:
    a) Users will have varying degrees of technical proficiency, including individuals with physical, visual, or cognitive disabilities.
4) Possible and Probable Changes in Functionality:
    a) The system should be modular to accommodate future enhancements such as additional accessibility features or integration with third-party APIs.

4.2 General Constraints

The design of the Accessible Map Project is subject to the following constraints:

1. **Hardware or Software Environment:**
    a. Limited to the processing and memory capacities of standard consumer devices.
2. **End-User Environment:**
    a. Must perform efficiently in both high-speed and low-speed network conditions.
3. **Availability or Volatility of Resources:**
    a. Cloud-based hosting may incur downtime due to maintenance or unexpected outages.
4. **Standards Compliance:**
    a. Adheres to WCAG 2.1 guidelines for web accessibility.
5. **Interoperability Requirements:**
    a. Ensures compatibility with GIS data formats and external APIs for map services.
6. **Interface/Protocol Requirements:**
    a. Utilizes HTTPS for secure communications and JSON for data exchange.
7. **Data Repository and Distribution Requirements:**
    a. Centralized cloud-based database for storing user preferences and map data.
8. **Security Requirements:**
    a. Implements role-based access control and data encryption to protect sensitive information.
9. **Memory and Capacity Limitations:**
    a. Optimized to function within the memory and storage limits of typical consumer devices.
10. **Performance Requirements:**
    a. Routes must be calculated and displayed within two seconds of a user request.
11. **Network Communications:**
    a. Designed to handle intermittent connectivity gracefully by caching critical data.
12. **Verification and Validation Requirements:**
    a. Comprehensive unit testing and user acceptance testing are required to ensure functionality and usability.
13. **Other Requirements:**
    a. Documentation and training materials must be accessible and user-friendly.

4.4 Development Methods

The development of the Accessible Map Project follows an iterative and agile methodology to ensure continuous feedback and adaptability. Key components include:

1. **Agile Framework:**

    a. Features are prioritized in sprints, allowing incremental delivery of functional components.
2. **User-Centered Design:**
    a. Involves regular usability testing with individuals from diverse backgrounds to refine accessibility features.
3. **Component-Based Architecture:**
    a. The system is built using reusable components to facilitate maintenance and scalability.
4. **Code Standards and Best Practices:**
    a. Adherence to React development standards and other industry coding practices ensures readability and maintainability.
5. **Automated Testing:**
    a. Continuous integration and deployment pipelines include automated testing to detect and resolve issues early.
6. **Toolset:**
    a. The project uses Flask for backend development, Bootstrap for responsive design, and SQLAlchemy for database management.

By employing these methodologies, the Accessible Map Project aims to deliver a robust, accessible, and user-friendly platform that meets and exceeds stakeholder expectations.

5.0 Architectural Strategies

The architectural strategies for the project are designed to ensure scalability, maintainability, and user-centric functionality, while addressing key technological and design considerations.

5.1 Use of technologies

The project leverages React for the front-end due to its component-based structure and extensive community support. While Flask was initially considered for the backend, it may be replaced with another framework better suited for JavaScript-centric development. For data persistence, a cloud-based relational database like PostgreSQL has been selected, offering scalability and robust query capabilities. Additionally, the system integrates with GIS APIs for map rendering and route calculation, along with accessibility-focused libraries to ensure compatibility with assistive technologies.

5.2 Reuse of Software components

To accelerate development and maintain consistent language design, existing open-source components such as React UI libraries (e.g., Material-UI or Ant Design) will be utilized. On the backend, reusable modules for authentication and data validation will provide modularity and facilitate future extensibility.

5.3 Future Extension Plans

The system is designed with modularity in mind, enabling seamless integration with additional APIs, such as elevation tracking, or weather or traffic services, to enhance route recommendations. Future enhancements include the implementation of multi-language support and advanced personalization features, which can be easily incorporated due to the flexible architecture.

5.4 User Interface Paradigm

The application employs a responsive and accessible UI paradigm, ensuring usability across a wide range of devices and compliance with WCAG 2.1 standards. Input is designed to support multiple modalities, including keyboard navigation, touch gestures, and voice commands, to enhance accessibility.

5.5 Error Detection and Recovery

Real-time input validation is implemented to handle user actions, supported by robust error logging mechanisms to identify and resolve issues promptly. The system also incorporates graceful degradation strategies to ensure continued functionality in the event of non-critical failures.

5.6 Data and Persistence Management

User preferences will be saved using a combination of caching, cookies, and account creation. Temporary preferences or session-based data will be managed through caching or browser cookies for quick access. For persistent storage, user account information, preferences, and route data will be securely stored in cloud-hosted databases with nightly backups to ensure data integrity. Performance optimization will be achieved through caching mechanisms, such as Redis, to improve access times for frequently used data.

5.7 Communication and Networking

Client-server communication will rely on a suitable API framework, either RESTful APIs or GraphQL, depending on the project's evolving requirements. RESTful APIs are currently a strong candidate due to their simplicity and wide adoption, while GraphQL may be considered later for its query optimization capabilities. Regardless of the choice, secure HTTPS protocols will ensure data integrity and protection against interception.

5.8 Concurrent and Synchronization

The project employs web workers or service workers to handle concurrent processing tasks, such as route calculations, without blocking the main thread. Synchronization mechanisms are implemented to ensure consistent updates to user preferences across devices.

By integrating these strategies, the project will achieve a robust, scalable, and user-focused system architecture that aligns with the project’s goals and requirements.

6.0 System Architecture

The system is designed as a web-based application that will allow users to select starting and ending points on an interactive map with the primary goal of displaying routes based on accessibility requirements. This architecture is a client server architecture where our backend will be in Flask and our frontend will be React. We will also be using PostgreSQL as our database in which the backend will communicate with to store and retrieve user information and route data. We will also be using the libraries Google Map API and OSRM which will contribute to the processing of the routes.

These are the components of our application:

1. **Backend (Flask)**

   This is where much of the processing will be done in our application. It will utilize the libraries Google Map API and OSRM to process routes. The backend will also receive requests from the frontend and send the required data back for the frontend

to display the information. Finally, the backend will also communicate with PostgreSQL to store information regarding user information and routes.

2. **Frontend (React)**
   The frontend will be responsible for displaying information for the user to select and navigate. It will also send requests to the backend to get information to display.

3. **Database (PostgreSQL)**
   The database will store any user information and route data. This includes accessibility related data.

4. **Frameworks**
   We will be utilizing Google Maps API and OSRM in a hybrid combination to give the best data when it comes to calculating and displaying the routes.

6.1 System Architecture Diagram

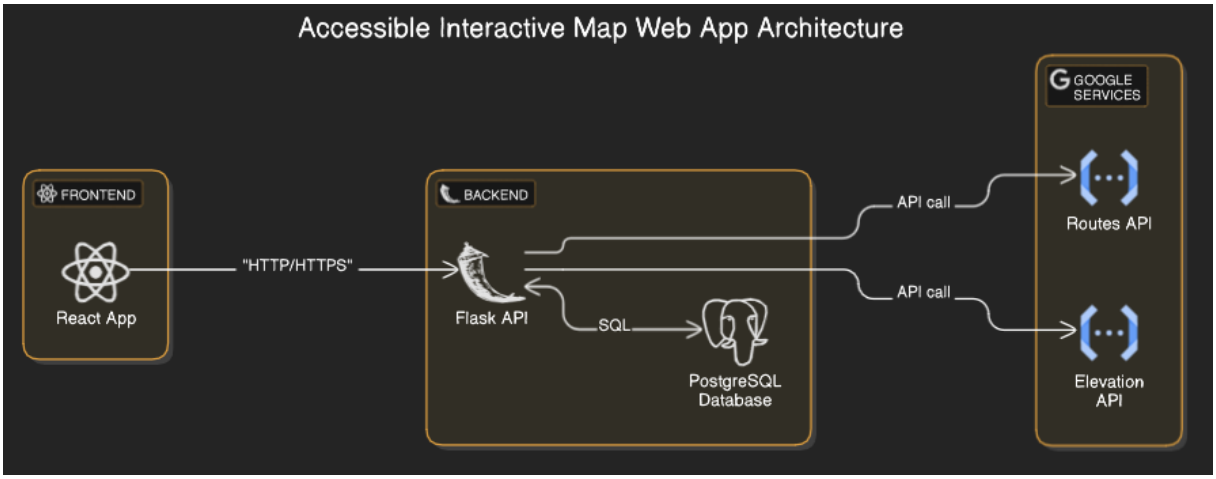

Figure 1:

7.0 Detailed System Design

Most components described in the System Architecture section will require a more detailed discussion. Other lower-level components and subcomponents may need to be described as well. Each subsection of this section will refer to or contain a detailed description of a system software component. The discussion provided should cover the following software component attributes:

7.1 Classification

The kind of component, such as a subsystem, module, class, package, function, file, etc. ....

- **Modules**
  - Frontend (React, JavaScript, HTML, CSS)
    - Components – UI elements (i.e Search bar, Tokens, Map, Buttons, Forms, Dashboard)
    - API Services – Functions handling HTTP requests to backend
  - Backend (Flask)
    - Routes – API handling route related requests from the user
    - Services – Functions processing requests and interacting with the database
  - Database (PostgreSQL)
    - Tables – Entities like Users, Routes, Feedback, Preferences, Map data
    - Relationships – Foreign Key relationships and constrains
- **Frameworks/API**
  - Google Maps API
  - OSRM

7.2 Definition

The specific purpose and semantic meaning of the component. This may need to refer to the requirements specification.

- **Frontend**
  - Display an interactive map with accessibility options along with a user-friendly interface.
  - Send requests to the backend module for route, calorie calculations, and user information
  - Utilize Google Maps API to render the map to the user
- **Backend**
  - Handle communication between the frontend and the database.
  - Will contain the logic to calculate the routes that is requested from the user
  - Process the calorie calculations of a given route
  - Communicate with the frontend to retrieve and store data related to the user and routes.
- **Database**
  - Stores user information and calculated routes.
  - Will also contain assessable information regarding the routes.

7.3 Process Flow

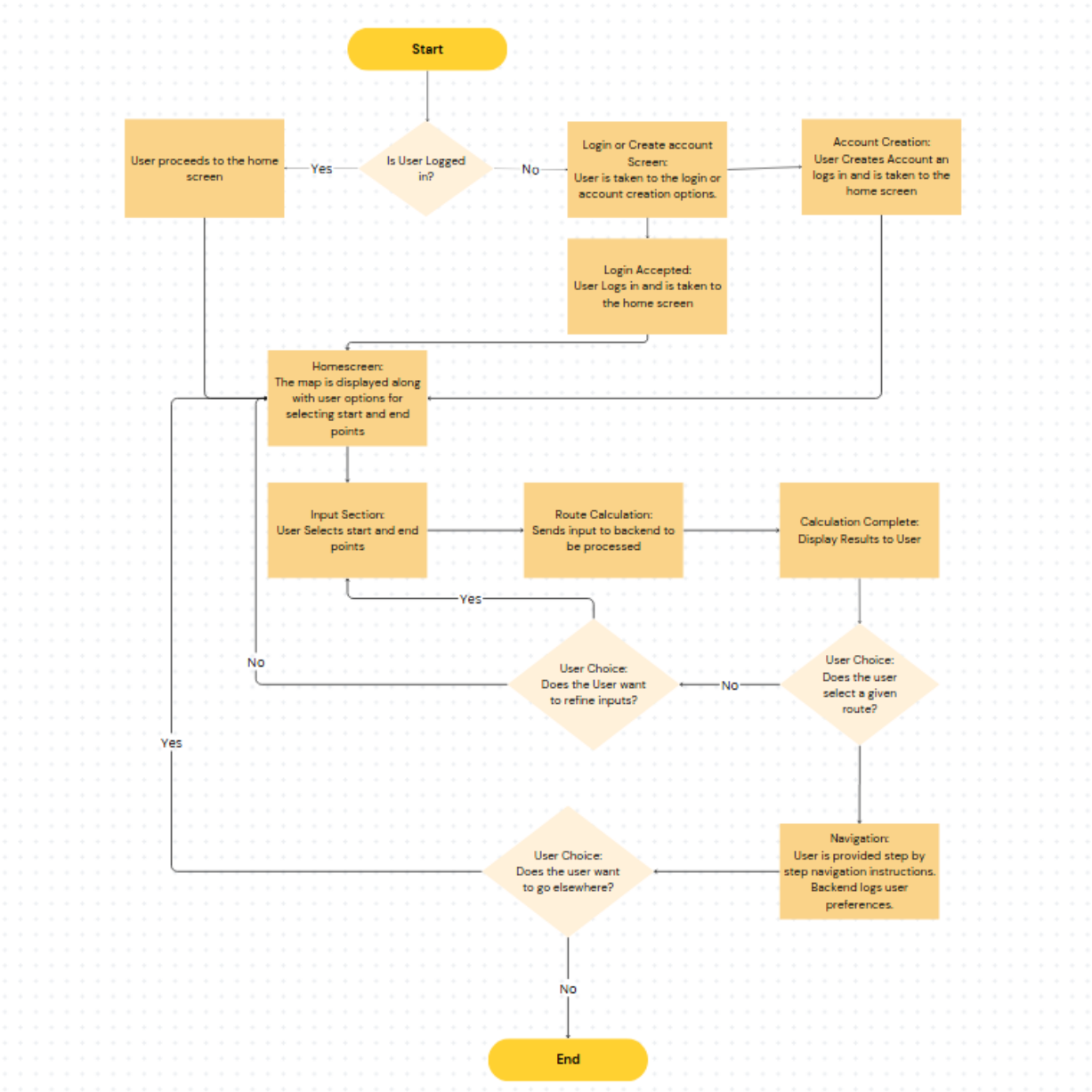

Figure 2:

7.4 Mock-Ups

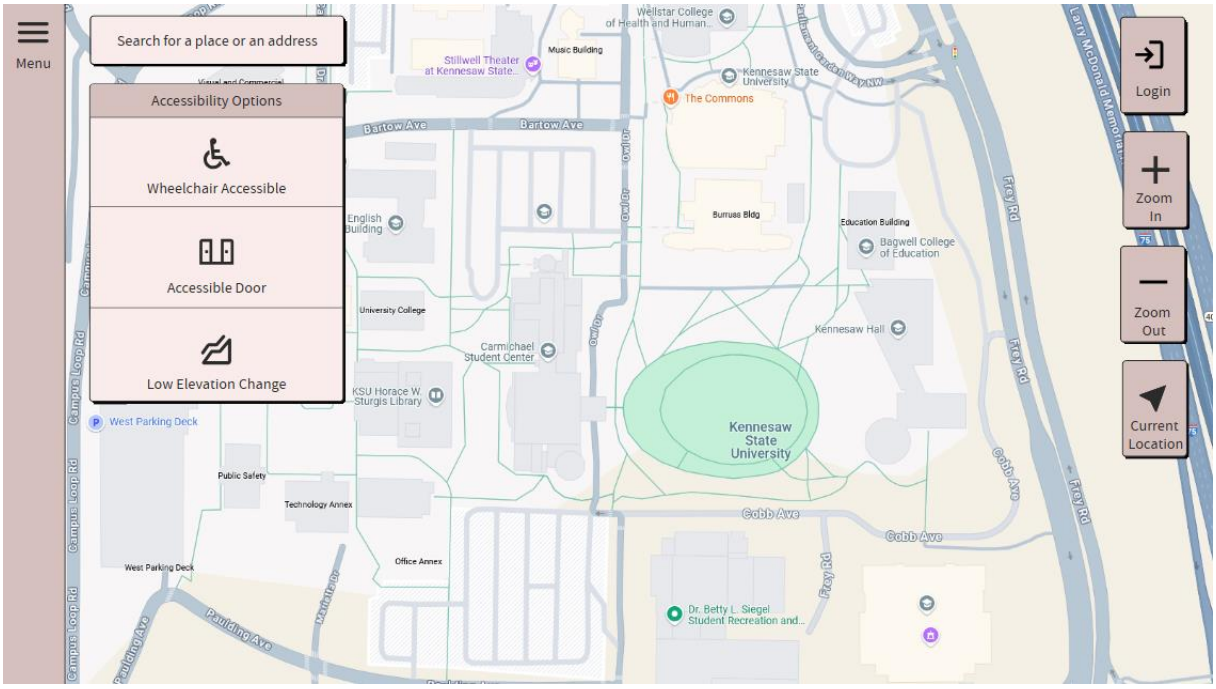

Figure 3:

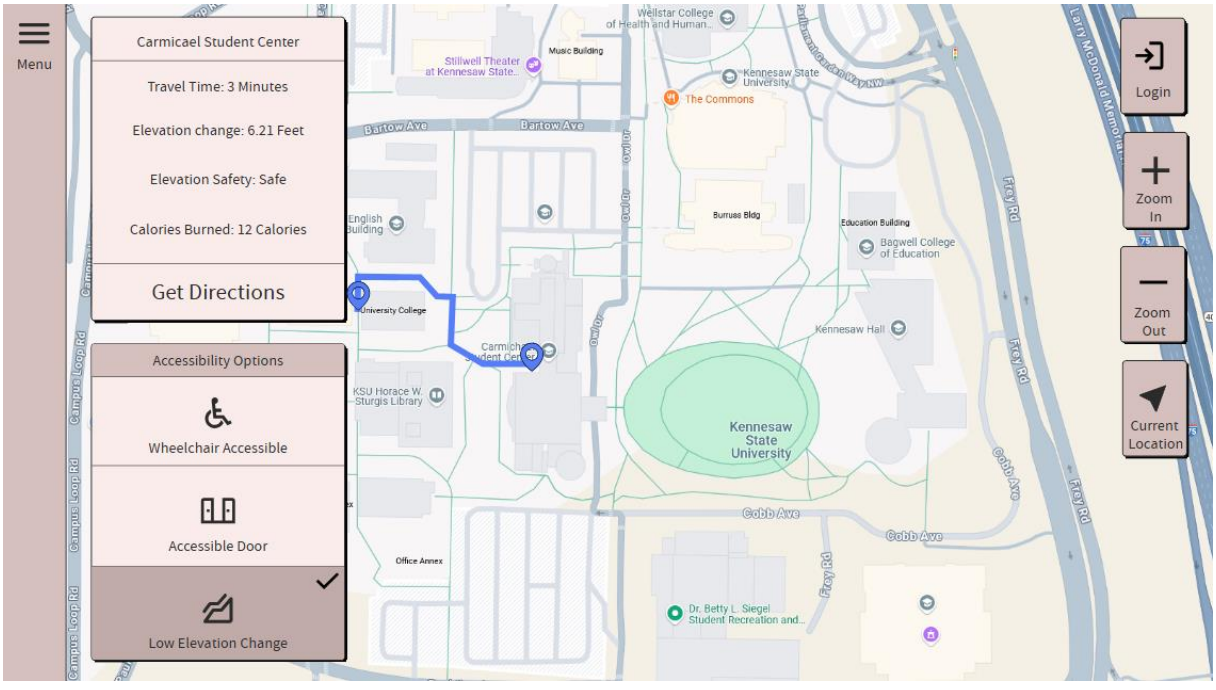

Figure 4:

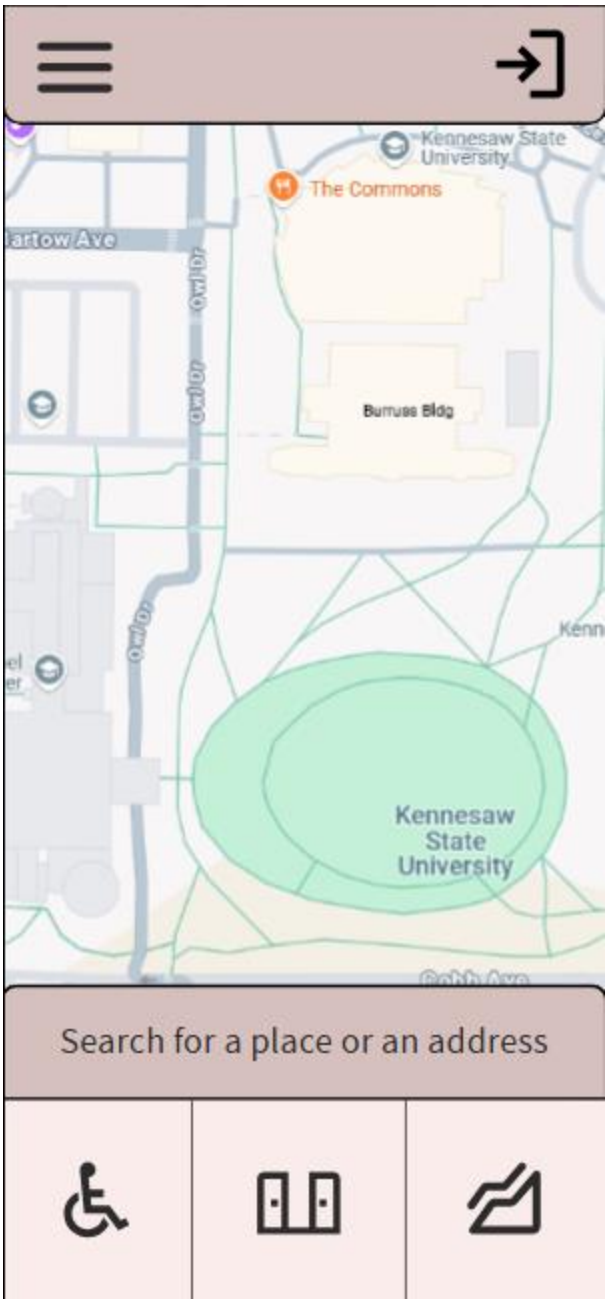

Figure 5:

7.5 Constraints

Any relevant assumptions, limitations, or constraints for this component. This should include constraints on timing, storage, or component state, and might include rules for interacting with this component (encompassing preconditions, postconditions, invariants,

other constraints on input or output values and local or global values, data formats and data access, synchronization, exceptions, etc.)

- **Performance (Frontend, Backend, Database)**
  - This is one of the more important metrics to get right in the project.
  - For the Frontend, we need the route to display in a timely manner as it will have a significant impact on the user's experience.
  - The Backend will need to calculate the routes in a timely manner after receiving a request from the Frontend.
  - The Database is extremely important to get query results in a timely manner, especially as the Database grows in size.
- **Scalability (Database)**
  - Scalability is more of a worry when it comes to providing the product to other institutions. Institutions that have bigger campuses and a larger student population would need to be taken into account during the development of the application.
  - The Database could be a bottleneck if it cannot handle a large number of requests. These requests can be due to more users and/or more routes.
- **API Limits (Backend)**
  - One limiting factor would be the API rate limit that Google Maps API has. This limit can hinder the process of getting coordinates for a destination and even the calculation of routes. This limit will need to be taken into consideration especially when it comes to an increase in users using the product.

7.6 Resources

A description of all resources that are managed, affected, or needed by this entity. Resources are entities external to the design such as memory, processors, printers, databases, or a software library. This should include a discussion of any possible race conditions and/or deadlock situations, and how they might be resolved.

- **React**
  - This will be the Frontend framework which will display a user-friendly interface. It will have components that the user can interact with such as the interactive map, search, and user account information.
  - Most of what will be displayed on the Frontend will run through React and any data retrievals will be requested from Flask (Backend).
- **Flask**
  - This is the Backend framework that handles the logic and communication between the database and the Frontend.

    - Flask will also communicate with the Frameworks/APIs to process the calorie and route calculations.
- **Google Maps API**
    - This will provide location and map rendering for the Frontend and specific locations for the Backend.
- **OSRM (Open-Source Routing Machine)**
    - Will assist in providing the shortest paths for desired routes that are already mapped out in the real world.
- **PostgreSQL**
    - Will be used as our database for the application. It will need to store information and be available to be queried to retrieve for the Backend to process the information.

7.8 Interface/Exports

The set of services (resources, data, types, constants, subroutines, and exceptions) that are provided by this component. The precise definition or declaration of each such element should be present, along with comments or annotations describing the meanings of values, parameters, etc. …. For each service element described, include (or provide a reference) in its discussion a description of its important software component attributes (Classification, Definition, Responsibilities, Constraints, Composition, Uses, Resources, Processing, and Interface).

- **Database**
    - It will provide data to the backend to retrieve for the API calls via queries. This data includes user and route information that the Backend will need to process and send to the Frontend.
    - This process is vital since it will help speed route selection for recently selected ones from the user.
- **Backend**
    - The Backend will provide an API consisting of endpoints for the Frontend to make HTTP requests.
    - The Backend will also be calling from external libraries and APIs such as Google Maps API and OSRM.

8.0 Glossary

An ordered list of defined terms and concepts used throughout the document.

Term	Meaning
API	Application Processing Interface
GIS	Geographic Information System
HTML	HyperText Markup Language
HTTPS	HyperText Transfer Protocol Secure
OSRM	Open-Source Routing Machine
SQL	Structured Query Language

9.0 Bibliography

Accessible Interactive Map Project Proposal. (2025, January 20). Kennesaw State University Internal Document.

This internal proposal outlines the project's goals, scope, and initial design considerations for the Accessible Interactive Map.

Flask Web Framework Documentation. (n.d.). Retrieved from https://flask.palletsprojects.com/

The official documentation for Flask provides detailed information on building backend web applications using Python.

React Documentation. (n.d.). Retrieved from https://reactjs.org/docs/getting-started.html

The React documentation explains the framework's component-based architecture and best practices for creating responsive user interfaces.

PostgreSQL Global Development Group. (n.d.). PostgreSQL Documentation. Retrieved from https://www.postgresql.org/docs/

This resource provides comprehensive information on using PostgreSQL for data storage and management within the project.

Google Maps Platform Documentation. (n.d.). Retrieved from https://developers.google.com/maps/documentation

Google Maps API documentation covers guidelines for integrating mapping and location-based services into web applications.

Open Source Routing Machine (OSRM) Documentation. (n.d.). Retrieved from http://project-osrm.org/docs/

OSRM documentation offers technical details for implementing route calculation features, which are integral to the project's routing functionality.

Development

1.0 Transcript Concepts

MoSCoW Requirements Model (MVP)

- Must have
  - Tracking elevation button
    - Elevation API via Google
  - Mobile & desktop friendly
  - Building-to-building route display
  - Readable directions with ETA
    - Routes API via Google
    - Geolocation API via JavaScript
  - Search bar for start and end destinations
  - User account registration & login
  - Database – PostgreSQL
  - Anonymous access (guest)
- Should have
  - Account page
    - Update password and personal information
  - Side menus with links to Kennesaw and Marietta campus maps
- Could have & Won't have – implement in future versions
  - Lifestyle goals selection
  - En-route goal achievement (ex. steps taken over set goal, calorie counter)
  - Route times for different transportation
  - Saved routes
  - Two-factor authentication
  - Air Quality Index for asthmatics
  - Accessibility features
    - Screen reader compatibility
    - High contrast modes
    - Keyboard navigability

MoSCoW Requirements Model (MVP) –

To prioritize the features of our interactive map, we used the MoSCoW requirements model, categorizing features into "Must have," "Should have," "Could have," and "Won't have" for this version. The Minimum Viable Product (MVP) focuses on essential functionalities like route planning, elevation tracking, user login, and accessibility across

devices. Additional features such as lifestyle goal tracking, saved routes, and air quality indicators are marked for future versions.

Tracking Elevation button

To improve the accessibility of our interactive map, we integrated a Tracking Elevation Button, which allows users to assess elevation changes along their route. This feature is powered by the Google Elevation API, which retrieves elevation data based on geospatial coordinates. The implementation required sending requests to Google's API whenever a route was generated, ensuring elevation feedback for users navigating the campus. It will also display the route with the lowest total elevation for the user to follow.

Mobile & Desktop Friendly

Ensuring that the interactive map functions seamlessly across both mobile and desktop devices is a key priority. Our approach involves using responsive design principles with CSS media queries and flexible layouts. Additionally, we leveraged Google's Routes API to ensure that building-to-building navigation remains intuitive, with clear directions, estimated travel times, and dynamic adjustments based on user preferences.

Building to Building Route Display

A primary function of our interactive map is to display routes between buildings with clear and concise directions. We implemented the Routes API from Google, which provides optimized navigation and estimated arrival times. To further improve usability, we integrated the Geolocation API via JavaScript, allowing users to set their starting location based on their real-time GPS position. This ensures that directions are accurate and dynamically updated for users navigating the campus.

Search Bar for Start and End Destinations

To make route planning quick and user-friendly, we incorporated a search bar for selecting start and end destinations. This feature utilizes an autocomplete function, reducing the likelihood of input errors and improving efficiency. The search functionality is built using JavaScript and interacts with our database to provide relevant location suggestions.

User Account Registration & Login

Our interactive map includes an account registration and login system, allowing users to personalize their experience. User authentication and account management are powered by a PostgreSQL database, where credentials and user preferences are securely

stored. This system enables future enhancements, such as saved routes and personalized settings.

Anonymous Access (Guest Mode)

For users who do not wish to create an account, we implemented an anonymous access (guest) mode. This feature allows anyone to use the map's core functionalities without requiring a login. While guest users cannot save routes or access personalized settings in future versions, they can still search for destinations, generate directions, and view elevation data.

Account Page

The account page provides users with an interface to update their password and personal information, which are found in the PostgreSQL database.

Lifestyle Goals Selection

A potential enhancement for future iterations is the lifestyle goals feature, which would allow users to track steps taken, calories burned, and other health metrics while navigating campus. This could be integrated with route planning to encourage fitness-oriented travel.

Saved Routes

Another feature planned for future development is saved routes, enabling users to bookmark frequently traveled paths for quick access. This would also utilize our PostgreSQL database.

Two-Factor Authentication

To enhance security, we aim to integrate two-factor authentication (2FA) in a later version, ensuring a more secure login experience for registered users.

Air Quality Index for Asthmatics

For users with respiratory conditions, we are considering an Air Quality Index (AQI) feature, which would provide real-time air quality data along selected routes. Finding an existing API would likely be the key to implementation.

Accessibility Features

Improving accessibility remains a long-term goal. Planned features include screen reader compatibility for visually impaired users, high contrast mode for better visibility,

and keyboard navigability for full usability without a mouse. These enhancements will ensure that the interactive map remains inclusive and user-friendly for all individuals.

2.0 Database Connection

1. Install PostgreSQL onto remote server
2. Run through setup wizard and set super user password
3. Edit your pg_hba.conf
    a. At the bottom of the file, under “# replication privledge.” add “host all all 0.0.0.0/0 md5”
4. Allow firewall exception for PostgreSQL port
    a. Control Panel -> System and Security -> Windows Defender Firewall -> Advanced Settings -> Inbound Rules
    b. Create a New Rule
        i. Select port, enter the PostgreSQL port, allow the connection, eave all checked, name the Rule
    c. Next repeat the previous step but for Outbound Rules
5. Use python to install pyscopg2
    a. Open terminal and enter the command listed in the next line
    b. pip install psycopg2
6. Setup your Python File
    a. Import psycopg2 and add the line below to establish a connection
    b. conn = psycopg2.connect(database="postgres", user="postgres", password="yourPostgreSQLPassword", host="your.host.ip", port=5432)
    c. You can start commands by adding this line, “cur = conn.cursor()” and execute commands with “cur.execute(“””SQL commands here”””);” and “conn.commit()”
    d. Don’t forget to close the connection with “cur.close()” and “conn.close()”

3.0 Set-Up

1. Clone the Repository
    a. Cloning the project to a version control platform makes it easier for multiple people to start working on development
    b. We have been using GitHub Organization
    c. When you have cloned the repository, open a new terminal and enter these commands on separate lines “cd adaMapDevSite” and then “npm run dev”. This will run our project with React.
2. Install Required Software
    a. Getting access to our server via Microsoft RDP
    b. Installing PostgreSQL software
    c. Installing IDE and Code editors
3. Set up the Database

    a. Edit PostgreSQL configuration file to allow remote connections
    b. Add port for PostgreSQL to firewall exceptions
    c. Creating the database and necessary tables
    d. Forming relationships between entities
    e. Install psycopg2 to connect to database
    f. May insert sample data for testing
4. Setting up Google Maps API key
    a. Navigate to the Google Cloud Developer Dashboard and create a new project
    b. Generate the API key for the necessary APIs (Directions API, Routes API)
5. Configure Database Credentials in Backend
    a. Ensuring that the backend code knows how to connect and communicate with the database server in
6. Set up Backend
    a. Setup the Flask Backend to handle the requests to the APIs
    b. Storing the Google Maps API key and ensuring proper security measures
7. Set up Frontend
    a. Ensuring the frontend files (HTML/CSS/JS) are stored in the appropriate folders
8. Testing
    a. Test features like: Viewing Buildings on the map, selecting buildings, Obtaining route information, etc. Accepting and submitting all valid feedback and adjusting the project to better handle found issues.
9. Deployment
    a. Hosting our project either on our server or a free hosting service
10. Maintenance and Troubleshooting
    a. If there are any issues that are found after deployment, a debugging session will be required until that problem is resolved and its origin is identified

Testing (plan and report)

1.0 Software Test Plan

1.1 Introduction

Our project is an accessible map for KSU students that uses React, Flask, Google Maps API, and Geolocation API. As a web application, it's available on major browsers on devices such as smartphones and desktops. Upon receiving the desired features from the project document, we started analyzing the requirements and used the MoSCoW Model to refine our scope. We also made user stories to determine our target audience which helped decide the application's primary features. After finalizing the requirements, we created a mockup in Penpot that focused on a monochromatic color palette and easily identifiable elements for the visually impaired. As we continue the development of this application, we are now working on testing the functional requirements.

1.2 Tester Selection

We will test the project by loading the project on our IDEs and passing it to peers. We were selected from class members, family members, and people we know with various disabilities. We will instruct them to look for defined test cases. As they move through the application, we will check whether the requirements passed or failed. We will also note the severity of the failure: low, medium, or high. The peers will be diverse in technical backgrounds, as some will, and some won't have programming knowledge or experience. In addition to this, we will ask peers that have special accessibility needs to take part in testing. We are testing the functional requirements, which are essentially the working features that make up the application. It will be tested on our local desktops using our IDEs since the project isn't available on browser yet due to it not being hosted on the server.

2.0 Software Test Report

2.1 Test Case Results

Requirement #	Description of Requirements	Pass/Fail	Severity of failure
1	Live tracking – The user changes their location from one building to the next. The indicator should be updated to reflect the change in location in near real-time.	Fail	Medium
2	Route creation – The user will set their location and set a destination and request a route. After their request a route should be created depending on the accessibility options.	Pass	High
3	Search bar with autofill – Users will start typing the name of several buildings. The search bar should automatically bring up names that have what the user is typing.	Pass	Low
4	Elevation tracking – Once the user creates a route, we are looking for changes in color along the route that accurately depicts the changes in elevation during the route.	Pass	High
5	Accessible doors icon/route – After the creation of the route the user will click on the accessible door option. There should be icons that	Pass	Medium/High

	represent accessible doors for the destination building.		
6	Database connection – We will test the connection between app and database with things like account information.	Pass	High
7	Building-to-building routes – Users will go to a building on campus and create a route to a different building.	Fail	Medium
8	Route leg instructions – Once a route is created under the map should show each part or “leg” of the route showing instructions.	Pass	High
9	Zoom functions – When looking at the map the user should be able to press the zoom in and out buttons to zoom in and out, respectfully, on the map or pressing “Ctrl” and scrolling in and out.	Pass	Low
10	User account registration – Users should be able to click on the top right section and create an account and have their information stored.	Fail	Low
11	User account login – Once a user creates an account, they will try logging in.	Fail	Low
12	Anonymous access – Full functionality	Pass	Medium

	remains even if the user isn't signed in.		
13	Account page – The user will be able to see their account and but able information regarding it.	Fail	Low
14	ETA approximation – Once the user creates a route, there should be an estimation of how long it will take the user to arrive at their destination.	Pass	Medium
15	User location – The app should be able to accurately obtain the users location.	Pass	High
16	Side menu with campus map links – Users will click on the side menu to ensure that it is working and the links lead to the correct campus maps.	Pass	Low
17	Map display – The map should display as the same size and in the same place on the website.	Pass	High
18	Identifiable buttons – Users should accurately be able to determine which button does what.	Fail	High
19	Mobile and desktop friendly – Our UI should look good for both desktops and mobile devices.	Fail	High

Version Control

Version control for the Accessible Interactive Map project was managed using Git and hosted on GitHub under the organization ADAMapDev. GitHub served as the central repository for collaboration among team members, allowing distributed contributions, issue tracking, and pull requests.

Each team member cloned the repository and worked within feature branches for individual tasks. Merges into the main branch were reviewed and approved to maintain code integrity. Regular commits ensured a clear history of changes, while README files, setup instructions, and documentation in the repository allowed for seamless onboarding and consistent development practices. Version tagging was used for milestone checkpoints throughout the semester.

Summary

The Accessible Interactive Map project successfully delivered a functional prototype focused on accessible, user-centered navigation for Kennesaw State University. Through the integration of technologies such as Google Maps API, OSRM, React, Flask, and PostgreSQL, the team developed a cross-platform application capable of route calculation, elevation awareness, lifestyle tracking, and real-time navigation support.

The app includes critical accessibility features such as screen reader compatibility, high contrast UI, and keyboard navigation. Testing involved both functional and usability evaluations with feedback from diverse users, including those with disabilities. While some features like login and account management encountered issues during final testing, the MVP fulfilled core requirements and laid a strong foundation for future development and deployment.

Looking ahead, the project aims to enhance accessibility further, expand to other campuses, and implement features like real-time location sharing, saved routes, and lifestyle analytics—fostering a more inclusive and intelligent campus experience.