

Semantic Visit Aware Recommendation of Hotels

DESIGN DOCUMENT

Team 34

Client: Goce Trajcevski

Advisor: Goce Trajcevski

Thomas Frohwein

Zachary Garwood

Dylan Hampton

Kevin Knack

Nathan Schenck

Britney Yu

Joseph Zuber

sdmay23-34@iastate.edu

<https://sdmay23-34.sd.ece.iastate.edu/>

Revised: 12-02-22 / Version 1.0

Executive Summary

Development Standards & Practices Used

- Agile Task Management Methodology
- Object-Oriented Programming
- Data-Structures
- IEEE Std 1012, Standard for Software Verification and Validation
- IEEE Std 1219, Standard for Software Maintenance
- IEEE/ISO/IEC 26512-2017, Requirements for acquirers and suppliers of information for users
- IEEE Std 982.1, Standard Dictionary of Measures to Produce Reliable Software
- IEEE/ISO/IEC 15288-2015, System life cycle processes

Summary of Requirements

- Design a web application that displays the determined route based on input from the user.
- The user should be able to easily input, view, and modify the current choices for points of interests and starting locations.
- The application should allow the user to specify which algorithm they want their route to be determined by.
- Design of the code should be extendable for easy addition of more areas/cities.
- Application should be able to reliably perform the task of input and creating routes.
- The user experience should be intuitive and clean.

Applicable Courses from Iowa State University Curriculum

- COMS 227: Object Oriented Programming
- COMS 228: Introduction to Data Structures
- COMS 309: Software Development Practices
- COMS 319: Construction of User Interfaces
- SE 329: Software Project Management
- SE 409: Software Requirements Engineering

New Skills/Knowledge acquired that was not taught in courses

- MapBox API
- React
- TypeScript
- Python

Table of Contents

1 Team	6
1.1 Team Members	6
1.2 Required Skill Sets for Your Project	6
1.3 Skill Sets Covered by the Team	6
1.4 Project Management Style Adopted by the Team	6
1.5 Initial Project Management Roles	6
2 Introduction	6
2.1 Problem Statement	6
2.2 Intended Users and Uses	7
2.3 Requirements & Constraints	9
2.4 Engineering Standards	9
3 Project Plan	10
3.1 Project Management/Tracking Procedures	10
3.2 Task Decomposition	10
3.3 Project Proposed Milestones, Metrics, and Evaluation Criteria	11
3.4 Project Timeline/Schedule	12
3.5 Risks And Risk Management/Mitigation	13
3.6 Personnel Effort Requirements	13
3.7 Other Resource Requirements	15
4 Design	15
4.1 Design Context	15
4.1.1 Broader Context	15
4.1.2 Prior Work/Solutions	15
4.1.3 Technical Complexity	16
4.2 Design Exploration	16
4.2.1 Design Decisions	16
4.2.2 Ideation	17
4.2.3 Decision-Making and Trade-Off	18
4.3 Proposed Design	19

4.3.1 Overview	19
4.3.2 Detailed Design and Visual(s)	20
4.3.3 Functionality	21
4.3.4 Areas of Concern and Development	22
4.4 Technology Considerations	23
4.5 Design Analysis	23
5 Testing	23
5.1 Unit Testing	23
5.2 Interface Testing	24
5.3 Integration Testing	24
5.4 System Testing	24
5.5 Regression Testing	24
5.6 Acceptance Testing	25
5.7 Results	25
6 Implementation	25
7 Professionalism	25
7.1 Areas of Responsibility	25
7.2 Project Specific Professional Responsibility Areas	27
7.3 Most Applicable Professional Responsibility Area	27
8 Closing Material	28
8.1 Discussion	28
8.2 Conclusion	28
8.3 References	28
8.4 Appendices	29
8.4.1 Team Contract	29

List of figures/tables/symbols/definitions

TABLES

Table 1: Use Cases and Affected Users	8
Table 2: Risk Probability	13
Table 3: Task and Work Hours Breakdown	13
Table 4: Broader Context	15
Table 5: Areas of Responsibility	25
Table 6: Responsibility Areas' Importance and Team Performance	27

FIGURES

Figure 1: Running Example of Diverse Path Search	7
Figure 2: Semester 1 Project Timeline	12
Figure 3: Semester 2 Front-End Project Timeline	12
Figure 4: Semester 2 Back-End Project Timeline	12
Figure 5: Semester 2 Front-End/Back-End Connection Project Timeline	13
Figure 6: Lotus Blossom	18
Figure 7: Use Case Diagram	19
Figure 8: Block Diagram	20
Figure 9. Initial User View of Front-End	21
Figure 10. Front-End View After Route Generation	22

1 Team

1.1 TEAM MEMBERS

- Thomas Frohwein
- Zachary Garwood
- Dylan Hampton
- Kevin Knack
- Nathan Schenck
- Britney Yu
- Joseph Zuber

1.2 REQUIRED SKILL SETS FOR YOUR PROJECT

- Java
- TypeScript
- Python
- React
- Spring

1.3 SKILL SETS COVERED BY THE TEAM

- Java: Thomas Frohwein, Zachary Garwood, Dylan Hampton, Kevin Knack, Britney Yu, Joseph Zuber, Nathan Schenck
- TypeScript: Nathan Schenck, Thomas Frohwein
- Python: Zachary Garwood, Dylan Hampton
- React: Nathan Schenck, Kevin Knack, Britney Yu, Zachary Garwood
- Spring: Zachary Garwood, Nathan Schenck

1.4 PROJECT MANAGEMENT STYLE ADOPTED BY THE TEAM

We have chosen a hybrid waterfall/agile development style as it is most suitable for this type of project.

1.5 INITIAL PROJECT MANAGEMENT ROLES

- Thomas Frohwein - Frontend
- Zachary Garwood - Backend
- Dylan Hampton - Frontend
- Kevin Knack - Frontend
- Nathan Schenck - Frontend
- Britney Yu - Backend
- Joseph Zuber - Backend

2 Introduction

2.1 PROBLEM STATEMENT

“Location-Based Services are often used to find proximal Points of Interest (PoI) - e.g., nearby restaurants and museums, etc. - in a plethora of applications” (Teng et al., 2021). Our client has developed an algorithm

to efficiently determine the optimal semantically diverse PoIs as well as the optimal route between them given user constraints like maximum travel distance, preferred PoI categories, etc. Furthermore, our client has compiled a dataset of New York City that contains categorized PoIs and the road network that contains them.

The main objective of this project is to create a web application that creates a visualization for the algorithm mentioned above for travelers to display the optimal PoIs and the path between them as there are no current means to visualize the algorithms. Additionally, visualizations of other algorithms that determine paths between PoIs will be created to compare the PoI diversity along the routes to demonstrate the algorithm's effectiveness. Users will be able to input constraints from which the algorithm will compute the route to semantically diverse PoIs.

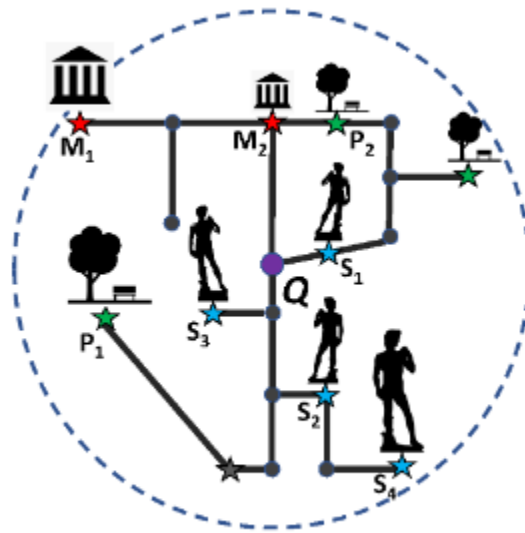


Figure 1: Running Example of Diverse Path Search

2.2 INTENDED USERS AND USES

Users:

- Tourists
 - Key characteristics:
 - They care about the locations they visit while on a trip.
 - They want to have new experiences.
 - Needs:
 - Easy to use interface.
 - Quick loading times.
 - Best route through their chosen interests.
 - A diverse route through different interests.
 - How they benefit:
 - Our application will give them the best opportunity to easily find the best route that fits their needs.

- Educator
 - Key characteristics:
 - They care about the efficiency of the application.
 - They care about the clear visibility of the algorithm's work.
 - Needs:
 - Need to be able to easily switch between routing algorithms.
 - Need an easy to use but professional interface.
 - Need the application to not slow down the algorithms for accurate comparisons between them.
 - How they benefit:
 - Our application will allow educators to directly compare multiple semantically diverse path-finding algorithms through points of interest. This will help them visually compare the algorithms for testing which is best.
- Business Professional
 - Key characteristics:
 - They care about the distance and time between points of interest.
 - Needs:
 - Easy to use interface.
 - Quick loading times.
 - How they benefit:
 - Our application will give them the best opportunity to visit points of interest within their schedule.

Use Case	Affected Roles	Details
Set starting location	All	The starting location will be from a pre-set list of hotels in NYC.
Input PoIs that interest user	All	Users can choose categories that interest them so those locations can be included on their route.
Choose which algorithms are implemented for their route	Educators	Educators can choose which algorithms are used to compare with each other.
See most optimal route on map	All	Tourists and Educators will want to see the most optimal route.
Input constraints	All	The Tourist will want to constrain distance and time to benefit their needs. The Educator will want to constrain them for testing each algorithm.

TABLE 1: USE CASES AND AFFECTED USERS

2.3 REQUIREMENTS & CONSTRAINTS

Functional Requirements:

- The application should be able to display the determined route that fulfills the input of the user.
 - In case there are multiple routes, the application should clearly indicate them.
- The user should be able to easily input, view, and modify the current choices for points of interests and starting locations.
- Stretch goal: Incorporate other attributes in ranking the selections i.e., rank hotels by price or stars; rank museums by entry fee, etc.
- The application should allow the user to specify which algorithm they want their route to be determined by.
- The application should be scalable i.e., allow multiple users to use it.

Nonfunctional Requirements:

- Design of the code should be extendable for easy addition of more areas/cities.
- Application should be able to reliably perform the task of input and creating routes.
- The user experience should be intuitive and clean.
- The code implementing every functionality should be designed with maintainability and extensibility in mind. For example,
 - UI components should be added over time
 - Data for additional cities should be easily adoptable

Constraints:

- Route should be fully displayed within seconds after input submission.
- Route visualization should be accurate.
- Total cost of implementation should not exceed \$300.
- Time: 2 semesters

2.4 ENGINEERING STANDARDS

- IEEE Std 1012, Standard for Software Verification and Validation
 - “Verification and validation (V&V) processes are used to determine whether the development products of a given activity conform to the requirements of that activity and whether the product satisfies its intended use and user needs.” (1012-2016 - IEEE Standard for System, Software, and Hardware Verification and Validation, 2016)
 - This standard will help this project to meet and or exceed the expectations and requirements of users and their needs.
- IEEE Std 1219, Standard for Software Maintenance
 - “provides the framework within which generic and specific software maintenance plans may be executed, evaluated, and tailored to the maintenance scope and magnitude of given software products.” (1219-1998 - IEEE Standard for Software Maintenance, 1995)
 - This standard will help this project to allow future developers to be able to easily maintain this software.
- IEEE/ISO/IEC 26512-2017, Requirements for acquirers and suppliers of information for users

- “This document was developed to assist users of ISO/IEC/IEEE 15288:2015 or ISO/IEC/IEEE 12207 to acquire or supply information for users as part of the system or life cycle processes. It defines the documentation process from the acquirer's standpoint and the supplier's standpoint.” (26512-2017 - ISO/IEC/IEEE International Standard - Systems and Software Engineering - Requirements for Acquirers and Suppliers of Information for Users, 2017)
- This standard will help this project to improve users quality and availability of information about the software.
- IEEE Std 982.1, Standard Dictionary of Measures to Produce Reliable Software
 - “This standard provides measures that are applicable for continual self-assessment and improvement of the software aspects of dependability.” (982.1-2005 - IEEE Standard Dictionary of Measures of The Software Aspects of Dependability, 2005)
 - This standard will help this project to create reliable and dependable software.
- IEEE/ISO/IEC 15288-2015, System life cycle processes
 - “This International Standard establishes a common framework of process descriptions for describing the life cycle of systems created by humans.” (15288-2015 - ISO/IEC/IEEE International Standard - Systems and Software Engineering -- System Life Cycle Processes, 2015)
 - This standard will aid us with definitions for the entire life cycle of the system, i.e., conception, development, production, utilization, support, and retirement.
- ACM 1.3
 - “Be honest and trustworthy.” (ACM Code of Ethics and Professional Conduct, n.d.)
 - This standard will aid this project developing software more ethically.

3 Project Plan

3.1 PROJECT MANAGEMENT/TRACKING PROCEDURES

Our group decided that we are going to use a hybrid of waterfall and agile as our project management style. For the design stage of the project, we are going to use waterfall as this stage consists mainly of the design document, and it will allow us to focus solely on each section of the design document at a time. Once we get to the implementation stage of the project, we are going to move to an agile approach that will allow us to work on the frontend and backend simultaneously, while also incorporating feedback that we get from our client/advisor during weekly meetings.

We are going to use GitLab to track version history and allow for simultaneous development. Trello will be used to assign and track tasks, and it will be used to monitor the project's overall progress. Lastly, group communication will be done through Discord.

3.2 TASK DECOMPOSITION

In accordance with the previous section for managing the project, here we list details of task decomposition where the shaded bullets represent our main tasks and the hollowed bullets represent the subtasks of their respective main task.

- Create the Design Document:
 - Identify use cases of project
 - Identify functional and non-functional requirements for project
 - Complete the project plan

- Build the Front-End UI:
 - Create a functional Figma Design
 - Set up base UI
 - Incorporate MapBox into the User Interface
 - Make MapBox features responsive to user input
- Build the Back-End:
 - Porting given algorithm
 - Storing data
- Connect front-end and back-end:
 - Develop API for communication between frontend and backend
 - Implement middleware to ensure frontend and backend work well together

3.3 PROJECT PROPOSED MILESTONES, METRICS, AND EVALUATION CRITERIA

Below are the milestones for our project broken down by semester. For estimated completion of each milestone reference the Gantt charts in section 2.4.

Fall Semester Milestones:

- Finalize tools and frameworks
- Complete preliminary system design
- Identify testing scenarios
- Refine the system design
- Refine testing scenarios and validation
- Finalize the design document

Spring Semester Milestones:

- Set up code base (Git, React project)
- Implement Visualization in MapBox
- Visualize route in MapBox
- Basic UI setup
- Use MapBox API to change map through user input
- Setup algorithms on backend
- Data is able to be stored
- Develop API for communication between frontend and backend
- Implement algorithm with frontend to change map

To measure the progress of the activities and completion of the milestones we will use the following metrics:

- General UI responds within a few seconds with correct information displayed. This will be evaluated by testing each of the UI's components.
- Displaying the locations and trajectory in Mapbox should correctly represent the one generated by the model. This will be evaluated by test cases corresponding to use case scenarios.
- Executing algorithms in the backend should complete within seconds. Correctness of the execution will be evaluated by using known scenarios (i.e., using some of the training data provided by the client).

3.4 PROJECT TIMELINE/SCHEDULE

The following Gantt charts represent the estimated project timeline. The bars represent the approximate length of the task and the diamonds represent the estimated completion date of milestones within each task.

First Semester

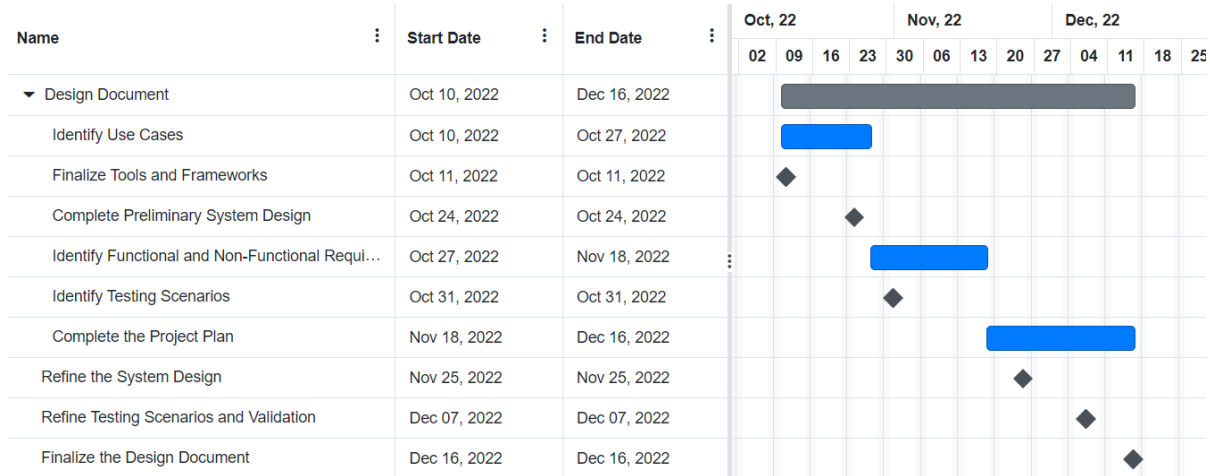


Figure 2: Semester 1 Project Timeline

Second Semester

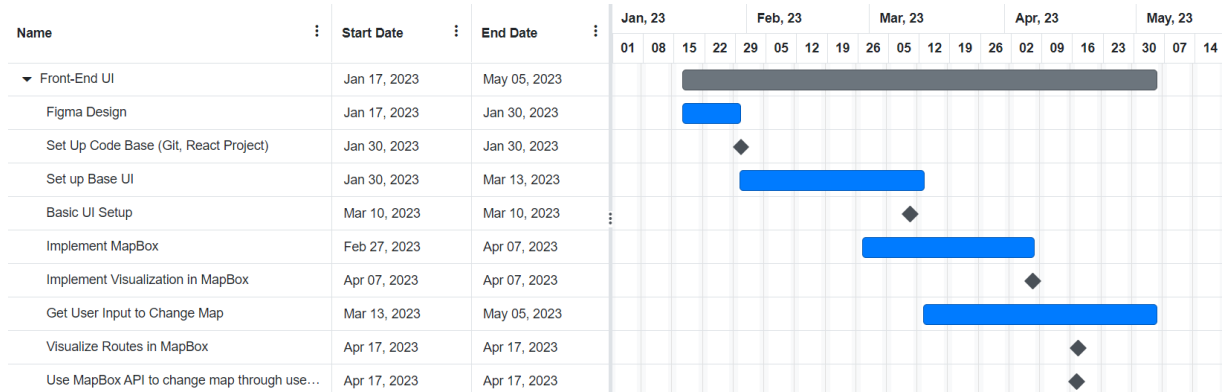


Figure 3: Semester 2 Front-End Project Timeline

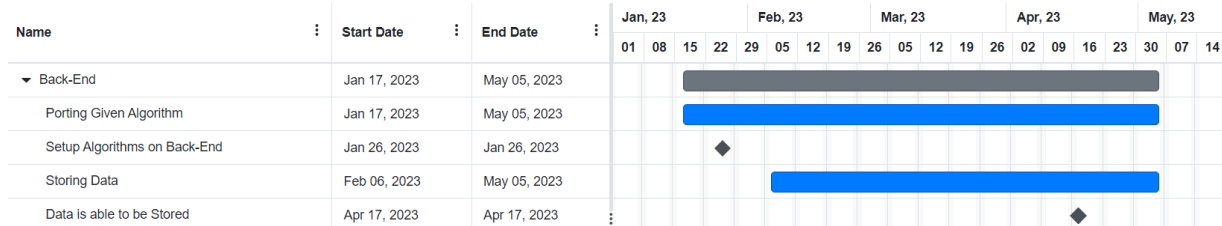


Figure 4: Semester 2 Back-End Project Timeline

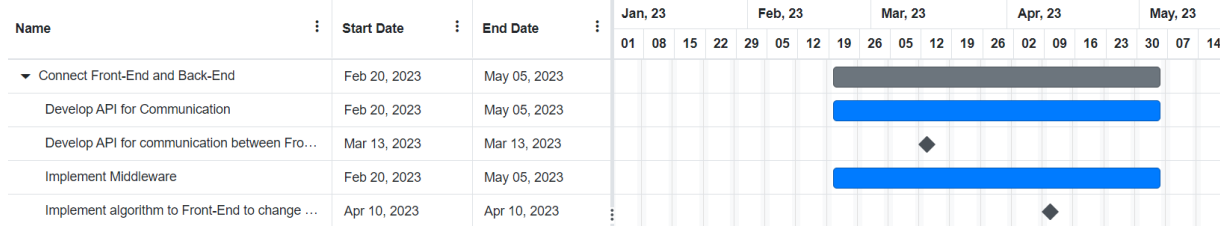


Figure 5: Semester 2 Front-End/Back-End Connection Project Timeline

3.5 RISKS AND RISK MANAGEMENT/MITIGATION

Since our implementation is an Agile approach to the project we can assess risks and risk mitigation with each sprint. Below is a table of risks that we have identified surrounding the project. Each row specifies the potential impact of the risk. Each cell on the table is the probability that the specified risk occurs during our execution of the project.

Risk	Mapbox is shutdown	Given algorithm is non-functional	Chosen UI component is unavailable	Versioning error resulting in code loss	Dataset gets corrupted
Catastrophic		1%			
Major	1%				
Moderate				10%	
Minor					5%
Insignificant			20%		

Table 2: Risk Probability

3.6 PERSONNEL EFFORT REQUIREMENTS

Below is the projected person hours divided by task.

Task	Project number of hours	Explanation
Finalize tools and frameworks	15	Determine what technologies we will use for the front and backend
Complete preliminary system design	35	Make diagrams for the system to show how the internals will communicate
Identify testing scenarios	25	Create test cases to determine our website is visualizing routes correctly

Refine the system design	20	Refine the previous diagrams and add anything in case of additions
Refine testing scenarios and validation	20	Refine the previous test cases and process to validate them
Complete the design document	70	Reviewing and revising design document and discussing with advisor
Set up code base	15	Setting up React project, backend code, and connecting code bases to GitLab
Implement UI for user input	60	Creating the UI layout for all user inputs and input processing
Implement MapBox into UI and create MapBox components	70	Figuring out how MapBox API implementation works and implementing the API into our code
Use MapBox API to change map through user input	50	Figuring out all parts of the MapBox API and how we can utilize it for our project
Setup algorithms on backend	30	Setting up how algorithms work on backend
Develop API for communication between frontend and backend	50	Developing the API between the MapBox UI and the database/backend
Implement algorithm with frontend to change map	60	This will probably be more API work, but customizing it with the algorithms given on backend
Implement test cases	50	Developing the test cases to verify our software is working as intended
Implement extra UI components	35	Extra components for more details on the routes
Stretch goal: Incorporate other attributes in ranking the selections	15	Rank the hotels by price or stars, rank museums by entry fee, etc.

Table 3: Task and Work Hours Breakdown

3.7 OTHER RESOURCE REQUIREMENTS

Below are our other resources needed to fulfill the project requirements.

- Datasets
- Algorithms
- MapBox API
- Personal Computers

4 Design

4.1 DESIGN CONTEXT

4.1.1 Broader Context

Below is a table containing examples of the ways our project impacts areas outside of the application:

Area	Impact
Public health, safety, and welfare	Increases pollution from more travelers visiting points of interest. May increase the spread of illnesses. Can be used by emergency response vehicles in case of widespread disasters.
Global, cultural, and social	Encourages more exploration. Gives travelers a more diverse set of places to visit.
Environmental	Increased energy use from running servers computing recommended routes.
Economic	Promotes tourist exploration. Can improve efficiency of rideshare routing. Beneficial to local economies.

Table 4: Broader Context

4.1.2 Prior Work/Solutions

With the rise of smartphones over the last couple decades, an increasing number of users have begun to use keyword-based search for content that is often related to their geographic location. This has sparked research focused on integrating location into keyword-based querying of geo-textual content (Chen et al., 2021). An example of such research is from a paper titled, *KORS: Keyword-aware Optimal Route Search System*, which outlines an algorithm to answer queries that find routes such that they cover a set of user-specified keywords, a specified budget constraint is satisfied, and an objective score of the routes is optimized (Cao et al.). The shortcoming of this approach is that it generates a route that brings the user to the specified keywords, but does not take into account the diversity of these keywords/PoIs.

There are many products on the market today that visualize routes between destinations and provide nearby PoIs for people to visit. Some of the more notable products in this area include Google Maps and Roadtrippers. Google Maps is known for its efficient and detailed route planning that allows users to see

things like travel time, different routes, and nearby PoIs that the user may like based on their data. Roadtrippers is well known for their travel guides where they provide PoIs that are within a given range along the route the user is taking as well as hotels for the user to stay at along the way.

Neither of the two solutions in the previous paragraph take into account PoI diversity when generating routes for the users to follow. This may lead to routes or travel guides that do not allow users to really see what the area they are visiting has to offer. Our solution fixes this problem by creating routes that maximize the PoI diversity based on constraints given by the user. In addition to this our solution has the following pros and cons as compared to the other products:

- Pros
 - Route maximizes PoI diversity based on given constraints
 - Specify certain PoI categories to visit
 - Does not store user information
- Cons
 - Limited to routes in NYC
 - Limited to starting from a known location

4.1.3 Technical Complexity

For the complexity of the project, we will be incorporating multiple data types into the project like spatial, temporal, and semantic data. We will also be utilizing multiple components/subsystems that make up the design of the project. We will have a frontend that contains the visualization of the project with a map API like MapBox. We will also have a backend that contains the API for the routing algorithms that are developed for us and incorporate that into the visualization. We will be coupling mathematical algorithmic solutions with index structure for efficient accessing of the data, and we will also be incorporating these solutions into the visualization which introduces technical complexity. These algorithms exceed current routing solutions by providing a route that is within a given range and diverse enough for the user.

4.2 DESIGN EXPLORATION

4.2.1 Design Decisions

Below are some key design decisions that will be decided on in the following sections:

Mapping API

There are a lot of possible services to use when it comes to displaying the routes in our application. The map is the center of the visualization, so it is important we select a service that meets our needs of readability, simplicity, and expandability.

Programming Language

Everyone on our team has different prior experiences so it is important that we play into everyone's strengths and select languages and frameworks that elevate the success of our team and provide the best opportunities for a good product.

Aesthetic

The choice of aesthetic, color, and font for the application is important because they affect how well our application's needs are met. The application should look good and be easy to use for our users

using it as tourists while users using it as an educational tool for the algorithms need it to be informative and presentable.

4.2.2 Ideation

In the lotus blossom activity pictured below, we have brainstormed for ideas that involve the functional software as well as the UI of our application. We divided our functional software ideas into the categories: Front end, Back end, and the Mapping API. As for the design component, we separated these ideas by the aesthetic, color scheme, and font of the application.

Front end: For the front end, we came to the consensus that the main three frameworks that we would look at were React, Angular, and Vue. Since our team knew we wanted to create a website, these three frameworks are very popular and widely used in web development. Through these choices, we evaluated our team's experience and skill sets to determine that React was the best option. In addition to the framework, we also had to decide between JavaScript and TypeScript. After discussing with the team, we decided to use TypeScript because of the several benefits of it allowing us to have static typing and can point out compilation errors at the time of development.

Back end: As for the backend, we have a plethora of options like Java, C#, Python, and COBOL. As a whole, our members have worked on various projects and classes with Java. As a result we unanimously decided to use Java as our back end language.

Mapping API: The Mapping API of choice had several options such as MapBox, Google Maps, Open Street Maps, ArcGIS, and BingMaps. Looking at google maps, we knew that holding an account could possibly create additional expenses or may not cover a whole semester for free. Other applications like ArcGIS are massive software which could require additional training and learning. One factor that played a huge role was that our client had suggested that we use MapBox. With careful consideration and looking further into the mapping api choices, we decided to go with MapBox.

UI and Design: For the design components, we thought of several ideas involving the aesthetic, color scheme, and font. With so many options, it was quite difficult for our team to decide on the exact look of our application. Through brainstorming, we realized there were several factors to consider when selecting these components. One of the issues we realized we could run into was the map colors clashing with the main colors of the app. There were other problems that we could see arising such as using a font that is too difficult to read on different backgrounds. As a team, we have decided to hold off on the decision until we familiarize ourselves with the map options and other design options that we have not explored yet. Creating the lotus blossom for this component helped our team realize other considerations of what we need to be aware of before following through with these design decisions.

			Microsoft Flight Sim	Mapbox	ArcGIS			
			Bing Maps	API Software	Google Maps			
			Open Street Map	Create your own maps	Flight Simulator			
		React		API Software		Cool Colors	Pastels	Blues
	Front End	Angular	Front End	Semantic Mapping	Color Scheme	Black and White	Color Scheme	Earthy Tones
TypeScript	JavaScript	Vue	Back End	Aesthetic	Fonts	Primary Colors	Warm Colors	Monochrome
Java	C#.NET	COBOL	Minimalistic	Bold	Modern	Comic Sans	Arial	Sans Serif
	Back End	Python		Aesthetic		Times New Roman	Fonts	Find a non-traditional font
		C					Georgia	

Figure 6: Lotus Blossom

4.2.3 Decision-Making and Trade-Off

Below are some trade-offs for each of the ideated options:

Mapping API

Certain GIS Tools have strict API limitations (such as Google or Bing Maps). Others may be very complex or hard to pick up, and may have too many or too few features for what we need (such as ArcGIS or OpenStreetMap). Because of the huge number of online tutorials available, and the recommendation of our client, we chose MapBox.

Programming Language

Our team has experience with a very wide variety of programming languages. Because of the complexity of the project, we felt we needed something higher level, but not too high such that it is not performant. We also felt we needed to choose something we were comfortable with using and we were experienced with. For this reason, we chose Java for the backend and REACT for the frontend.

Aesthetics

Decisions on aesthetics have a profound impact on the user experience. Blue light and complex fonts may be stressful or exhausting to a user. Poorly chosen colors may blend with the map. It is for this reason that we have decided on a simpler aesthetic, but our exact decisions on font and colors

are yet to be determined because we would like to get an idea of how they would look with the mapping API first.

Development Environment

Due to most of the team being most familiar with development on Windows, we will be doing all development on DOS based systems. Deciding on a single, uniform development environment is important so that all system calls will work on all systems. If we ever intended to extend our project support to UNIX based systems, we would need to rewrite all of our system calls which could be time consuming, however it would greatly increase the number of compatible systems and the portability of our software.

4.3 PROPOSED DESIGN

4.3.1 Overview

Our design will consist of three major subsystems:

- Front-end user interface that will enable the users to:
 - Specify the categories of points of interests (PoI) to be visited.
 - Specify the distance constraints for the trip.
 - Specify the starting-category.
 - Display the recommended trajectory for the visit as well as the recommended hotel (i.e., starting point).
- Back-end server
 - Store data
 - Generate route containing PoIs
- "Middleware" that will connect the front-end and back-end.

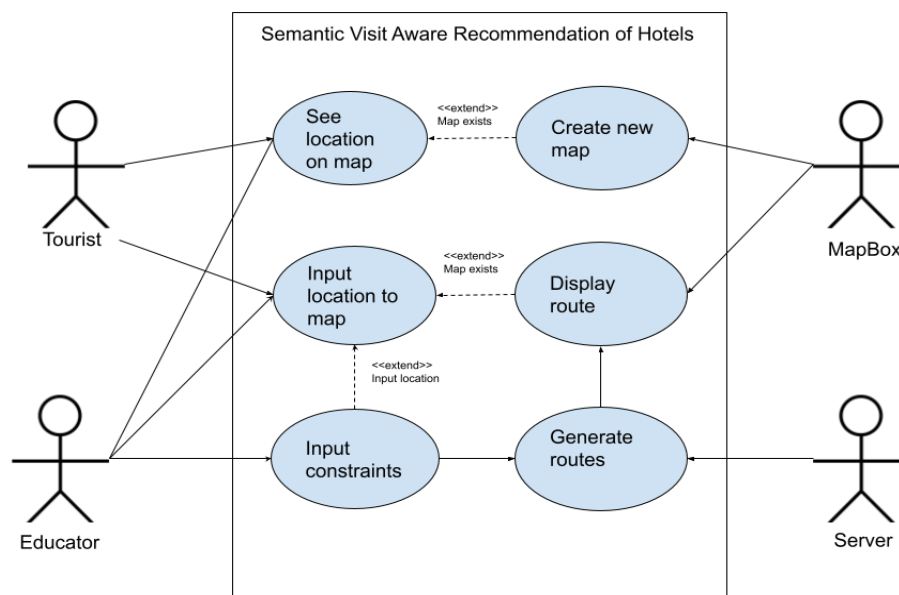


Figure 7: Use Case Diagram

4.3.2 Detailed Design and Visual(s)

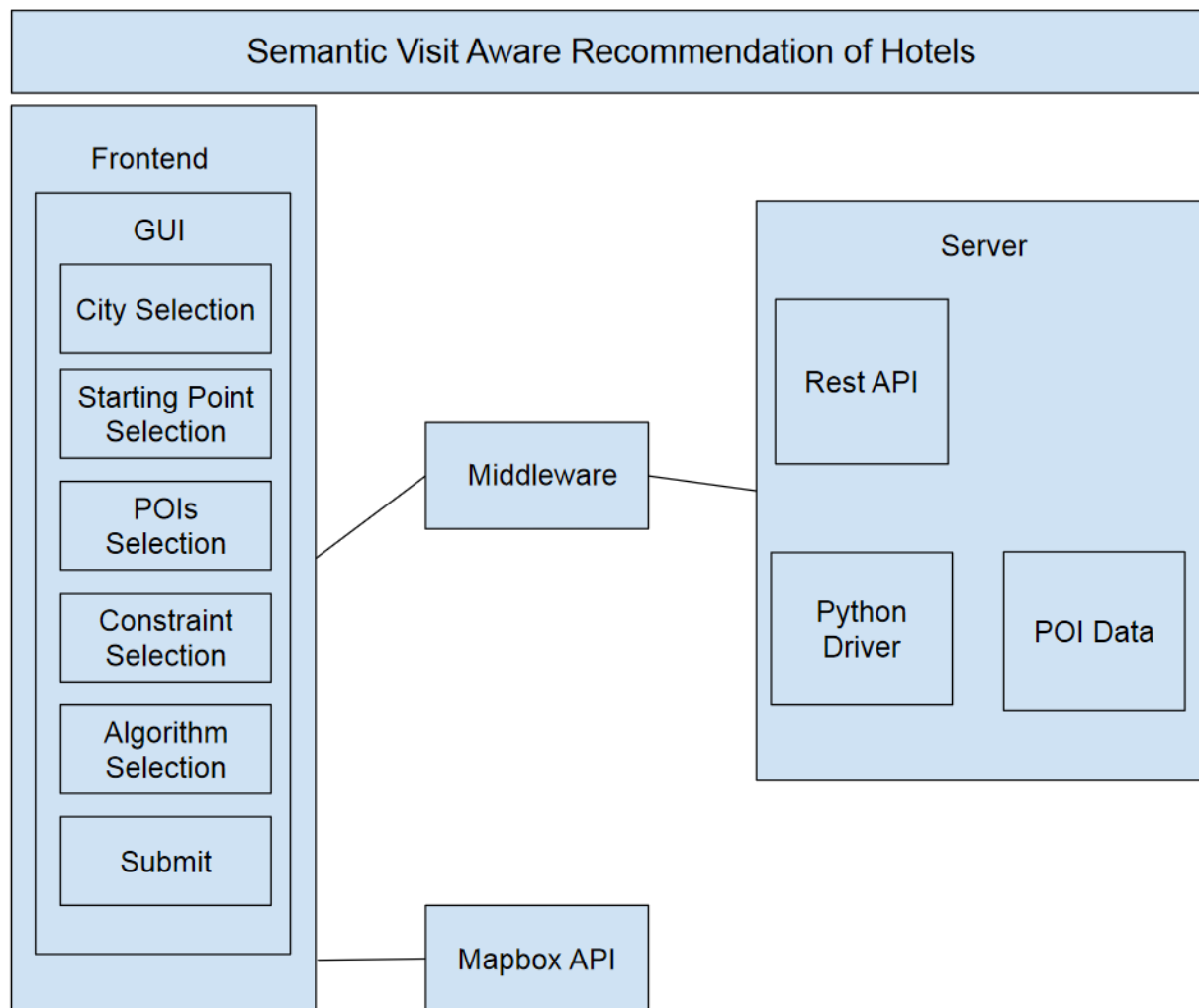


Figure 8: Block Diagram

To best describe our design, we will describe our sub systems separately. Once the design of the subsystems is complete, we can piece together the whole system.

Front-End

The user interface for our application will primarily be used in a web browser on a computer. The web application will be just a single page and made in React and TypeScript. The user experience needs to be clean and easy-to-use. This will be achieved by having a simple layout and by keeping related information together on the screen.

Back-End

The server will be used to store the given PoI data which will be used to execute the algorithms and generate the path with PoI. The backend will also communicate with the

front end to receive constraints such as starting point, preferred PoI, etc... from the user and provide the constraints to the algorithms.

Middleware

The middleware will be used to connect the frontend and backend together so we can efficiently send/receive requests made by the user in the UI and process them with the Python algorithms and location data stored in the backend.

Routing Algorithms

For our project, we received four routing algorithms. The algorithms are designed to find an efficient route from a starting point to different destinations specified by the categories given by the user. The algorithms designed to provide a best path are NSS-kDPQ and ESS-kDPQ. We were also provided a Dijkstra algorithm and Random Walk with Restart.

4.3.3 Functionality

When a user opens the web application they will be shown a map of the city that they would like to explore and plan their stay-location (e.g., a hotel) as well as a sequence of PoIs to be visited. The default/initial implementation will focus on New York City. In addition, they will be presented with a form that will allow them to specify constraints and other information for route generation. This can be seen in Figure 9. below:

Semantic Visit Aware Recommendation of Hotels

Starting Location

Destination Location

Distance Constraint

Number of PoIs

Algorithms

PoI Categories

Generate Routes

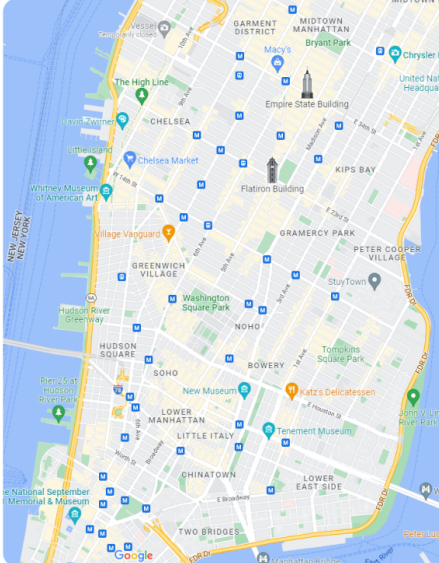


Figure 9. Initial User View of Front-End

Users will be able to enter only one value for each of the inputs in the form, except if the form element has a plus sign inside the input field. This plus sign indicates that the user can enter multiple values into the input field, and once entered, the values will appear below the input field so the user can see what has already been entered.

Once the form is filled out, the response by the system will be executed in a manner that will generate routes satisfying the given constraints. To do so, the user needs to click on the “Generate Routes” button. This will send the user-entered information to the backend, as seen in Figure 8, where the Python driver will be triggered. This, in turn will generate the following sequence of activities:

1. Access the file containing the PoI data (CF Fig. 8)
2. Invoke the route generation algorithm (selected by the user’s request)
3. Generate the routes that will be stored in a JSON file.
4. Send the JSON to the MapBox API to visualize the routes.
5. Upon completion, an event will be generated that will reflect the view on the map of the UI.

After this sequence of events, the web application will be updated to look like Figure 10.

Semantic Visit Aware Recommendation of Hotels

Starting Location
Chinatown, New York, NY

Destination Location
Midtown Manhattan, New York, NY

Distance Constraint
2 miles

Number of Poles
6

Algorithms
ESS-kDPQ

Pol Categories
Movie Theater

VSS-kDPQ RWR Restaurant Museum Park

Generate Routes

Figure 10. Front-End View After Route Generation

4.3.4 Areas of Concern and Development

Our primary concerns for delivering a product that meets our user and client needs are the following:

- We will need to learn how to use the MapBox API to incorporate visuals and also provide routes from our routing algorithm
- We will have to learn how to use the React framework to create our frontend
- We are not certain on how to develop effective integration testing to test the system’s behavior.

To address the above concerns we are going to do the following:

- We will read the MapBox documentation and test it with our solution
- We will read the React documentation and help others on the team when needed
- We will ask our client how to go about developing effective integration tests.

4.4 TECHNOLOGY CONSIDERATIONS

React - React is modern, has lots of developer support, and allows you to create components that can be used to cut down on coding time. However, it can be very slow if not well optimized. Because it is being updated so often, it can have poor documentation. There are some viable alternatives, Angular for example, but these are less popular.

MapBox API - The MapBox API is documented extensively online, making it easy to learn. The load times are quick and access to data is easy. However, the system can be slightly complex and has a steep learning curve. Alternatives for other APIs are often too simplistic and don't offer as much functionality (such as OpenStreetMaps), or have stricter API limits (such as Google Maps), or have MUCH steeper learning curves (such as ArcGIS Online). We must have a mapping API in order to display paths so there are no alternatives to having a mapping API.

[Back-end language/framework] - Java is a flexible, relatively high speed language that everyone in the group has lots of experience with. Using it to work with the CSV file and generate routes would be straightforward and efficient. The downside is that we have to have communication between these different parts of our project, which makes our design more complex. The alternative would be to use React to do these things directly, but that would be more inefficient and make our project more monolithic in design.

4.5 DESIGN ANALYSIS

Our team is planning to do the testing and implementation of the design next semester. At this time, we have not implemented or tested our design from 4.3, so we do not know if our design will work. We have thoroughly planned for what our team intends to do for each of the sections of the project for the front end, back end, and middleware. We have also created the UI design of the webpage that we plan to build to get the general idea of the look of our design.

5 Testing

The following is an overview of the testing methodology that we will be applying throughout the development of our project. We will create tests for the functional and non-functional requirements, mentioned in section 1, to verify that the functional requirements are working as intended and that the non-functional requirements are meeting the needs presented by our client. An aspect that is unique to our project is that there are no specific cost related requirements as we are developing a proof of concept system. When components of our project, as seen in Figure 8., are implemented we will run them through a series of unit, interface, integration, system, and regression tests. Once these tests have completed, we will discuss with our client the results to find areas that could be improved upon. The subsections below outline when the particular tests will be performed throughout the development cycle.

5.1 UNIT TESTING

For unit testing, we can test the individual units that make up the total project. For this, we can test the different GUI components and their individual functions. Because we are using React, we could use the Jest framework to test these individual React components to make sure that they work on their own before we integrate them together. We also want to test our backend by unit testing the individual functions on the backend to make sure that our API works correctly. We also want to make sure that the Python Driver works and that the MapBox API works with our functions.

5.2 INTERFACE TESTING

The interface testing can be tested through combining multiple units. The combination of units will ensure that the interface of the application is able to successfully implement the interface. Here are a few examples of combined units we will test for interface testing:

Algorithm & Mapbox Database: The algorithm will be tested on its ability to get the correct data from the Mapbox Database.

Algorithm & Interface: The interface will be tested on how well it is able to visualize the algorithm.

Interface & Mapbox Database: The interface and Mapbox database will work together to build the bulk of the visual for the application. Both must be able to work together to display a map on the website and get the necessary data.

Algorithm, Mapbox Database, Interface: The overall interface that the user is interacting with will comprise the all three units— Algorithm, Mapbox, and Interface. We need to test that all components are able to cohesively work together to create a working interface.

5.3 INTEGRATION TESTING

Integration testing will be done by breaking the system down into different functionalities and then testing these functionalities independently from one another. Integration tests will ensure that all of these individual functionalities are producing expected results. These functionalities combine components that are tested using interface testing. An example of one of these functionalities is selecting a city and having it immediately show up on the map. Another one which would be dependent on the first one already working is selecting a location in the city and having it show up immediately on the map. Once all of these functionalities are tested and working, the entire system will be tested using system testing.

5.4 SYSTEM TESTING

System testing will be done by the succession of multiple integration tests. i.e. System tests will follow multiple long paths and verify that at each step, what we see matches what we expect. These paths, and thus these tests, will stretch across the entire length of the system, from the Mapbox API to the user interface. This will ensure that, if a specific location is provided by the user at the user interface level, that choice will propagate correctly through all levels to the API and the API response will propagate back to the user interface correctly. To do this, corresponding unit, interface, and integration tests should be added together to chain through the system. All critical requirements (that paths are generated correctly, based on the correct algorithm, and begin at or visit the correct locations) should be verified using each of these system tests.

5.5 REGRESSION TESTING

To ensure new additions to the system do not break existing functionality, we will run existing unit, interface, and system tests, as well as, manually verifying system functionality to ensure that every existing part of the system still works as expected after implementing new features. In addition, we will also verify that all critical requirements are unchanged after the implementation of new functionality. One critical feature that will be tested and verified is the implementation of new algorithms for the route generation. After the addition of a new algorithm, we will verify that all previous functionality and important system level functionalities are still working as intended. Additionally, as a stretch goal, after mapping a

new city for our system we will verify that all previous cities still function properly and its data has not been modified.

5.6 ACCEPTANCE TESTING

In order to ensure that the functional requirements of the project are being met, we will develop a set of use cases for functions of the application. The use cases will cover all possible uses of the application. During testing, the use cases will be followed as test routes through the application to ensure that the application is performing as expected.

Non-functional requirements will primarily be tested by demonstrating the application to our client to ensure that project requirements are met. Throughout the development, with each major change or milestone, the client will give the team feedback, and we will revise and improve our product as needed. The client will continue to give feedback until satisfied with the final product.

5.7 RESULTS

As we develop our project, we will test it extensively using the tests above. Many different cases will be tested to ensure full coverage. Based on the results of those tests, we will assess our performance and make changes as necessary. This will ensure that we meet all of our goals set out in Section 3 and other sections. As we are working with a hybrid waterfall/agile development model, testing will be done in parallel with further development after the general framework of the project has been implemented.

6 Implementation

As per our plans in section 3.3, our project is moving forward according to the plan. We have a solid foundation of what our system architecture will look like and how our project will look in regards to languages, frameworks, and tools. Our team is still wrapping up our design phase of the project process and will begin implementing our design in the second semester of the class.

To start the implementation of our project, we will first set up a code base and begin working on creating visualizations with the map. At the same time, the team will also be developing the data flow from the algorithms to the visualization. These two processes will be a great kickstart to our implementation process which we will follow throughout the process using the details in section 3.3.

7 Professional Responsibility

This discussion is with respect to the paper titled “Contextualizing Professionalism in Capstone Projects Using the IDEALS Professional Responsibility Assessment”, *International Journal of Engineering Education* Vol. 28, No. 2, pp. 416–424, 2012

7.1 AREAS OF RESPONSIBILITY

Area of Responsibility	Definition	NSPE Canon	Differences between NSPE and SE Code of Ethics
Work Competence	Perform work of high quality, integrity, timelines, and professional	Perform services only in areas of their competence; Avoid deceptive acts.	The SE code says engineers should ensure their products meet the highest

	competence.		professional standards which unlike NSPE does not say engineers should only work inside their area of competence
Financial Responsibility	Deliver products and services of realizable value at reasonable costs.	Act for each employer or client as faithful agents or trustees.	The SE code says to act with the client and employers best public interest while the NSPE says to act as the client or employers trustee
Communication Honesty	Report work truthfully, without deception, and understandable to stakeholders.	Issue public statements only in an object and truthful manner; Avoid deceptive acts.	The SE code says that engineers should act with integrity while the NSPE says that engineers should only publicly speak in objective truths
Health, Safety, and Well-Being	Minimize risks to safety, health, and well-being of stakeholders.	Hold paramount the safety, health, and welfare of the public.	The SE code says engineers should act consistently with public interest while the NSPE says engineers should hold the health and safety of the public paramount
Property Ownership	Respect property, ideas, and information of clients and others.	Act for each employer or client as faithful agents or trustees.	The SE code says to act with the client and employers best public interest while the NSPE says to act as the client or employers trustee
Sustainability	Protect the environment and natural resources locally and globally.		The SE code says to act in the public's best interest while the NSPE has no input on sustainability
Social Responsibility	Produce products and services that benefit society and communities.	Conduct themselves honorable, responsibly, ethically, and lawfully so as to enhance the honor, reputation, and usefulness of the profession.	The SE code says engineers should advance the integrity and reputation of the profession while the NSPE says engineers should conduct themselves with honor, ethics, and lawfully to advance the profession

Table 5: Areas of Responsibility

7.2 PROJECT SPECIFIC PROFESSIONAL RESPONSIBILITY AREAS

Importance	Team Performance
High, It is important that quality work is done when working on our project so that it meets and or exceeds our clients expectations.	High, while our team has not yet created any prototypes, the other work done for the future work to be done has been of high quality.
Low, There is no cost associated with our project.	High, our system is planned to only include free and open source software for third party components like the mapping service.
High, It is important to communicate with our client honestly about the work done so that we can get honest feedback, as well as keeping our client informed about the progress of the project	High, our team regularly communicates with our client and is truthful in the work being done.
Low, There is little to no chance of the project impacting the safety or well-being of any of the users of the project.	N/A, our project has no impact on public safety or well-being.
Low, All client property / ideas are open source and freely available to the public.	N/A, all client code / ideas are public and open source.
Low, There is very little potential impact on the environment besides the energy required to run a web server.	High, our project has very little impact on the environment.
Medium, Our project is beneficial to a significant amount of people and can potentially impact how people plan for vacations or other traveling plans	High, our team regularly conducts themselves with honor, respect, and integrity.

Table 6: Responsibility Areas' Importance and Team Performance

7.3 MOST APPLICABLE PROFESSIONAL RESPONSIBILITY AREA

The most applicable professional responsibility to our project is communication honesty with our client. This responsibility is important to the project because there are only a few static requirements, so it is important that we communicate frequently, openly, and honestly with our client to get important feedback. Our team has demonstrated this responsibility well so far by communicating weekly with our client as well

as speaking honestly on our progress each week. This has impacted our project beneficially by allowing us to make changes as needed to better fulfill our client needs and wants for our project.

8 Closing Material

8.1 DISCUSSION

Much of our work on the project thus far has been focused on developing a design/plan for the software to implement the routing algorithms that we are given into a website. We will be using React/Typescript as the frontend with MapBox API as the mapping component in the UI. We will be using Java/Spring as the backend and also use the given Python algorithms to generate the routes that will be shown on the map. We have a list of requirements and standards that we will follow to create a quality solution and provide us with a clear understanding of our plan.

8.2 CONCLUSION

Throughout our design process so far, we have made decisions on design, architecture, and user needs. These decisions come through research, comparison, and choosing what best fits the requirements of the project. Our primary goal for this project is to create an application that can be used to display a determined route through a provided algorithm given certain constraints. To meet this goal, we must ensure that we are staying on our determined plan of action that is stated throughout this document. So far, our team has met and exceeded our goals for the end of this semester and looks forward to beginning the next phase of our project next semester.

8.3 REFERENCES

- Teng, Trajcevski, & Züfle. (2021). Searching Semantically Diverse Paths.
- 982.1-2005 - IEEE Standard Dictionary of Measures of the Software Aspects of Dependability. (2005). IEEE Xplore. <https://ieeexplore.ieee.org/document/1634994>
- 15288-2015 - ISO/IEC/IEEE International Standard - Systems and software engineering -- System life cycle processes. (2014). IEEE Xplore. <https://ieeexplore.ieee.org/document/7106435>
- 26512-2017 - ISO/IEC/IEEE International Standard - Systems and software engineering - Requirements for acquirers and suppliers of information for users. (2017). IEEE Xplore. <https://ieeexplore.ieee.org/document/8288807>
- 1219-1998 - IEEE Standard for Software Maintenance. (1995). <https://ieeexplore.ieee.org/document/720567>
- 1012-2016 - IEEE Standard for System, Software, and Hardware Verification and Validation. (2016). <https://ieeexplore.ieee.org/document/8055462>
- ACM Code of Ethics and Professional Conduct. (n.d.). Association for Computing Machinery. <https://www.acm.org/code-of-ethics>
- Chen, Z., Chen, L., Cong, G. et al. Location- and keyword-based querying of geo-textual data: a survey. The VLDB Journal 30, 603–640 (2021). <https://doi.org/10.1007/s00778-021-00661-w>

X. Cao, L. Chen, G. Cong, J. Guan, N. -T. Phan and X. Xiao, "KORS: Keyword-aware Optimal Route Search System," 2013 IEEE 29th International Conference on Data Engineering (ICDE), 2013, pp. 1340-1343, doi: 10.1109/ICDE.2013.6544939.

Code of ethics: IEEE Computer Society. Code of Ethics | IEEE Computer Society. (n.d.). Retrieved November 28, 2022, from <https://www.computer.org/education/code-of-ethics>

8.4 APPENDICES

The algorithm for the implementation of this project is provided by our client and is located in the following Github repository: <https://github.com/XTRunner/MDM2020>

Additionally, our client has provided us with datasets to use in conjunction with the algorithm for the project.

8.4.1 Team Contract

Team Members:

- 1) _Britney Yu _____ 2) _Dylan Hampton _____
 3) _Nathan Schenck _____ 4) _Zachary Garwood _____
 5) _Thomas Frohwein _____ 6) ___Kevin Knack _____
 7) __Joe Zuber _____ 8) _____

Team Procedures

1. Day, time, and location (face-to-face or virtual) for regular team meetings:
 - Sundays at 3PM in person or virtual as determined by Gocee Traveski
2. Preferred method of communication updates, reminders, issues, and scheduling (e.g., e-mail, phone, app, face-to-face):
 - Discord
3. Decision-making policy (e.g., consensus, majority vote):
 - Decisions should be civilly argued and if no agreement is reached then a vote where a majority is decided should be held.
4. Procedures for record keeping (i.e., who will keep meeting minutes, how will minutes be shared/archived):
 - We plan to rotate who takes minutes weekly. We will put them in a shared google drive, and we will have the person who takes notes also submits the weekly progress report.

Participation Expectations

1. Expected individual attendance, punctuality, and participation at all team meetings:
 - Everyone should be present in the meeting unless there are some extenuating circumstances. If someone were to miss a meeting, they should communicate with the team

about anything (progress, questions, etc) *before* the meeting instead of being stuck playing catch up afterward.

2. Expected level of responsibility for fulfilling team assignments, timelines, and deadlines:
 - Everyone's tasks should be completed on time. If their work is a possible blocker for another team member, they should work to get it done as soon as possible, well before the deadline. If for some reason someone cannot make a certain deadline they should let the team know and try to get help so the deadline can be met
3. Expected level of communication with other team members:
 - Everyone should proactively communicate questions and concerns before they become blockers or issues. On Discord, if a long conversation took place, any party not involved in the conversation should acknowledge that they have read it by a reaction or message. In addition, we plan to add some additional text channels to be made in discord for various things that may come up as the semester goes on like subgroups. We also expect members to communicate when they complete something/progress in addition to questions and concerns, so the group can get an idea of what still needs to be completed.
4. Expected level of commitment to team decisions and tasks:
 - Everyone that is a stakeholder in any decision should have a vote when those decisions are made, and stakeholders in team tasks should work hard to get them completed on time.

Leadership

1. Leadership roles for each team member (e.g., team organization, client interaction, individual component design, testing, etc.):
 - At the time of writing this, we do not have set leadership roles. The team agrees we will let these roles fall into place when the time comes, instead of forcing people into roles right now. Should no strong leadership be created by the end of the semester, Britney will help organize the roles for each team member.
2. Strategies for supporting and guiding the work of all team members:
 - We agree it's important that we outline our expectations for each team member when we meet. We also agree on the importance of communicating busy schedules and work that is stuck. If a team member has a busy schedule for a week or is stuck on their work, they should make it known so we can figure out a best plan of action whether it be deferring their work or reassigning it.
3. Strategies for recognizing the contributions of all team members:
 - We think it's important for everyone to be heard. We are thinking we could possibly implement a time at the beginning of our meetings (or class times or something) where everyone talks about what they have worked on, kind of similar to a stand-up.

Collaboration and Inclusion

1. Describe the skills, expertise, and unique perspectives each team member brings to the team.
 - Nate: I have taken pretty much all the SE requirements, just finishing out electives. Also have had a couple internships working in environments where I got full-stack experience with Java, .NET, Angular, and React. I really enjoy code design, so I'm hoping I can bring some good ideas to the table.

- Dylan: I had an internship this past summer working with C++ as well as Python. I'm pretty well versed in Java, C, and C++. I am also familiar with Python and MySQL. I have had some small experiences with JS and React but I am not confident in those. Willing to learn languages / frameworks as needed to help contribute to the team.
 - Zach: I have had a few internships where I mainly did backend work in Java and a Java-like language. I also have experience working with Python, C, C++, MySQL, and a little bit of JavaScript and Svelte. I'm open to learning any technology that is needed to help complete the project.
 - Kevin: I have experience in Java, C, C++, MySQL. I have some experience with Python, JavaScript and HTML5, but I have not used them very extensively. I will learn anything needed to help complete this project.
 - Joe: Most of my experience is with C and Java. I have some Python experience but I'm rusty. I also worked in a limited fashion with databases over the summer at my internship. I am very open to learning new languages or gaining experience in new areas (especially C++ and MySQL, which I believe are good skills to have for my future career).
 - Britney: I have taken a lot of SE courses making me experienced in C and Java. I did a few projects in React. For an internship I worked on a team doing some scripting in JavaScript. In another team in the same internship, I also was able to do some full-stack development with a focus on frontend using Vue 3, .NET.
 - Thomas: I have experience with Java, Python, JavaScript, React, and C. I also have some experience with databases and SQL. I also have some experience with AWS from my internship. I have some experience with web development also.
2. Strategies for encouraging and supporting contributions and ideas from all team members:
 - To avoid the victim-boss mentality, we will create an encouraging environment for our team so it is not only controlled by a single person. Our discussions will be focused on having everyone contribute their ideas in a round-robin style so we can be mindful of the opinions of our team members.
 3. Procedures for identifying and resolving collaboration or inclusion issues (e.g., how will a team member inform the team that the team environment is obstructing their opportunity or ability to contribute?)
 - The team will resolve collaboration or inclusion issues through team meetings, discord, and an anonymous google doc. We will ask in the beginning of each meeting if there are any problems to figure out prior to beginning the meeting. We can also post a message in the Discord any issues that arise. While members should take the initiative to express their concerns, in the case that a member wants to stay anonymous, they can put it on the [Identifying and Resolving Collaboration of Inclusion Issues](#) google doc which will be checked each week at the beginning of our meeting.

Goal-Setting, Planning, and Execution

1. Team goals for this semester:
 - We want to put ourselves in a position to have a smooth and successful second semester. To accomplish this, we will create and refine our requirements and design throughout the semester while taking user feedback into consideration. If we decide, along with our client/advisor, that our requirements and design are where we want them to be then we would like to begin creating a prototype.

