METCS673 Group 6 - Software Design Document

TerrierMap

Team Members:

- Ryan Christopher
- Misael Gared
- Jasmine Hughes
- Siddhraj Parmar

Overview:

This document provides a detailed design description of TerrierMap. It will allow both developers and stakeholders to understand the architecture and design of the project. TerrierMap is intended to help guide new Terriers to their classrooms with step-by-step instructions from the classroom code provided on MyBU. TerrierMap will be web-based and accessible through a web browser.

Nomenclature:

- Building Code: a 3-letter code/abbreviation given for the location of courses for BU. It includes buildings on BU's campus and occasionally denotes whether the course is online
- Firestore Database: the cloud hosted NoSQL database that houses the building information

System Architecture:

TerrierMap will use a Client-Server-Database architecture. By utilizing this architecture structure, this will allow TerrierMap to present an intuitive frontend to a user while making calls to a database and running appropriate calculations on the server side. In the frontend, we are using Next.js as our React framework along with Tailwind CSS to create a responsive and easily interpreted user interface. The api calls to our project also exist in the Next.js project, however are at a different location and interact with both our firestore database and the leaflet routing machine in order to generate a route from the user to the correct destination. The data for our application is all stored in the firebase database, where each item in our 'classrooms' collection consists of the following layout:

Key: Classroom Code -> Value: [Name, Address, Coordinates]

High-Level Design:

Requests Mapping:

After a user enters the building code and classroom number and the search button is pressed, this will call an API route in the 'routes/API' folder with the building code and classroom number as parameters. The API route will then search the firestore database for the corresponding three-letter building code and if the entry exists it will return the building name, street address, and coordinates of the location. The longitude and latitude will then be passed into the Leaflet Routing Machine API. From here the information generated by the Leaflet Routing Machine will be rendered via a component in our Next.js project to display step-by-step directions to the building location. This route (along with the building name) will be returned to the user on the frontend.

APIs:

TerrierMap will be using two APIs: API Routes and Leaflet Routing Machine. API Routes is an API system supported by Next.js that allows api calls to be made at the same level as the root of the project. API Routes treats any file inside a "pages/api" folder as an API endpoint. This will allow an easier time to make calls to receive data from the firestore database. The firestore database is accessed through the JavaScript API that uses our authentication keys stored as environment variables to allow the project to access the data. The docs for this API can be found here: https://firebase.google.com/docs/reference/js/firestore_

Leaflet Routing Machine is an API that creates step-by-step directions for walking, driving, and cycling from a longitude and latitude of two points. The user's current location will be used as the starting point. The desired classroom will be the destination point. The docs for using the Leaflet Routing Machine can be found here: https://www.liedman.net/leaflet-routing-machine/api/

Detailed Design:

Libraries/Modules:

TerrierMap will make use of the following libraries/modules in order to properly operate: React Leaflet is an npm library that includes react components for leaflet maps. By using the leaflet maps as components, we are able to update them over time to show location

data, routes, and new information as the components are able to be re-rendered as much or as little as we want.

Next (NextAPIRequest and NextAPIResponse) is a library that is included in Next.js projects that allow us to use the API Routes mentioned previously. This library allows us to send requests to the API routes files and also allows for a response containing the data that is necessary for updating the map and routes to be sent back to the frontend and related components.

Python Unittest is the library that will be used for testing the information stored and generated in our project. We will use it to validate the information stored in the database as well as verifying that the routing machine will generate appropriate routes between two points to test various use cases.

Database Representation:
The Firebase database stores the building codes and addresses for the various classrooms. The database is broken down into the following:

- Collection: classrooms
- Documents: building codes (i.e. CAS, GSU, etc.)
- Fields: additional data for the buildings
    - address: the address of the building
    - building_name: the name of the building
    - lat_long: latitude and longitude of the address