NEA Analysis

The problem:

User wants access to a self-hosted mapping platform for their organisation, from which they can import real-world map data and modify this. They want to be able to add things like notes, rough drawings, and would like the ability to calculate paths. This should be accessible on a variety of platforms - Including phones, tablets and desktop computers.

Proposed solution:

A web-based mapping app that operates on a client-server model, which can import map data in the OpenStreetMap XML format, and return this to web clients from a custom database so that it can be displayed to the user. This can then be used to find paths between points, and where this data can be modified, able to add things like annotations. This will use a authentication system.

Objectives:

See attached "Project objectives" document.

Research:

- I have conducted research on the OSM XML file format and the useful information that can be extracted from this, proposing a written alogrithm for how to do this (OSM Conversion algorithm document).
- I have been researching the Bootstrap framework and implementing it so that I can generate UI without spending lots of time worrying about implementation of UI elements. For this I have been making use of the Bootstrap reference from their website.
- I have done some research on path theory, including specifically on Dijkstra's algorithm which I have included in my "Graph theory research" document. This will be used to generate paths given connecting points on a map.

Proposed solution detail:

- See High level project structure document and diagram for an outline of my initial plan, and how different project elements will interact.
- TODO: Create database layout.

Current implementation

At the moment, I have a web client serviced by a basic web server that can display data returned from the backend on a scaleable and translateable map. The server and client use shared classes in order to represent this map data. Large parts of my OSM conversion algorithm have also been implemented, reading OSM points from a ~400MB file (Cambridgeshire) and converting them into instances of my classes, which can then be shown on the client. Displaying this OSM map data is currently buggy however and has some issues, for example some roads seem to be missing and scaling is conducted about map origin making navigating the map difficult.

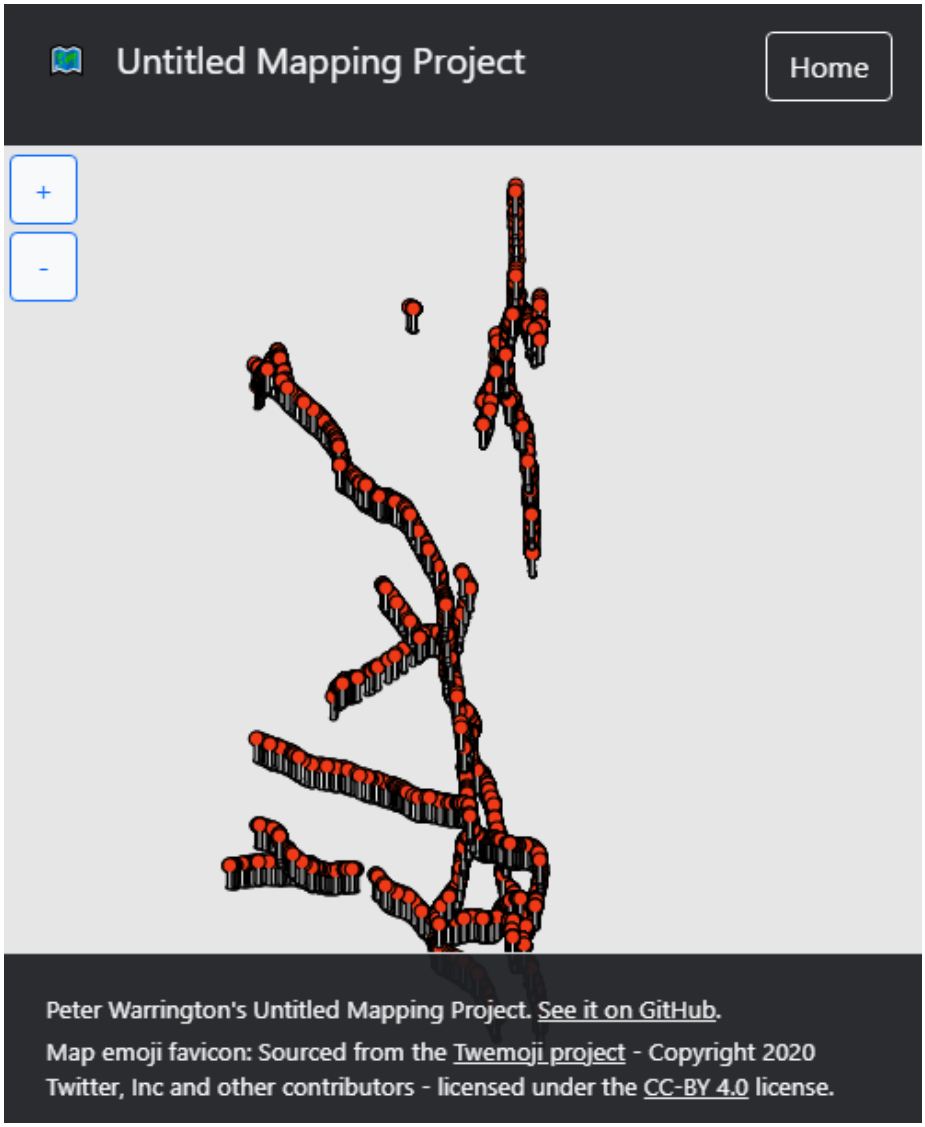

Figure 1: Screenshot showing OSM points returned to the client.

Project objectives

This document lists the objectvies I hope to be able to acheive as part of my NEA.

- Web client is able to send requests and receive appropiate data back to/from backend
- The backend is able to read from OpenStreetMap (OSM xml) data and return this to clients via this API
- The web client is then able to display and represent data returned from the server on a map that can be panned and zoomed
- A search function is available on this web client to be able to find locations using the API
- A function will be available to find a efficient route from one location on the map to another using path theory. This will take into account distance and this route must be traversable in the real world.
  - **Extension:** Take into account traffic, whether that be based on a custom specified metric or the location of traffic lights along a route
- Function serviced by database on the backend to allow a user to sign-in/register and save places/routes
- Testers reports that they are able to easily:
  - Find their home using the search function
  - Generate a route
  - Register and log-in
  - Save this location and route to their account
- Data structures and important algorithms are well-documented using comments
- API is well documented
- Effective use of object-oriented programming to be able to create effective data structures shared across the project, including the appropiate use of inheritance.
- Normalised database structure to store the data mentioned above.

Extension

- Cache the most frequently accessed places directly on the database
- Function to be able to plot new data on map that is then made available
- Ability to add notes to the map
- Ability to add other annotations to the map

OSM Conversion algorithm

Objective

To convert OSM data into the format required by the client.

Formats

Custom JS format (Desired)

- JSON
- Each road is a Path object.
- Each road is given a unique path ID
- A Path is defined in terms of its initial PathPart
- Each Point of a Path is connected in a chain of PathParts
- Each PathPart contains its point and the next PathParts in the chain

OSM (To be converted)

- XML
- Each road is defined in terms of a list of nodes
- Road metadata is given using elements in the format

  `<tag k="key" v="value"/>`
- Each road is a and has s with “type”=“route” and “route”=“road”
- Each has an “id” attribute
- Each has ways (the actual paths that make up the road) as s. Each component way is defined in the format:

  `<member type="way" ref="idOfTheWay" role=""/>`
- Each contains component nodes expressed as s in the format:

  `<nd ref="idOfTheNode"/>`
- Each has node in the format:

  `<node id="25496583" lat="51.5173639" lon="-0.140043" version="1" changeset="203496" user="80n" uid="1238" visible="true" timestamp="2007-01-28T11:40:26Z"></node>`

Objectives

- Convert OSM data from XML to JSON
- Find only those relations that make up roads
- For each of these relations, find the component ways
- For each of these ways, find the component nodes
- For each of these nodes, convert the latitude and longitude to x,y coordinates
- Convert each node to a MapPoint
- Create path for each way using connecting MapParts of MapPoints

Graph theory research

This document will look at how graph theory can be utilised with details on how this can be implemented, making notes from sources.

Notes

- Graph theory models paths in terms of pairs between nodes (also called vertices) which are connected by links (also called edges) [1]
- Pathfinding is largely based on Dijkstra's algorithm, and the use of weighted graphs [2]
- Weighting refers to the "cost" it takes to traverse a link, e.g. how long it takes, it is a number [2,3]
- For Dijkstra's algorithm: "This algorithm begins with a start node and an"open set" of candidate nodes. At each step, the node in the open set with the lowest distance from the start is examined. The node is marked "closed", and all nodes adjacent to it are added to the open set if they have not already been examined. This process repeats until a path to the destination has been found. Since the lowest distance nodes are examined first, the first time the destination is found, the path to it will be the shortest path."

References

[1] https://en.wikipedia.org/wiki/Graph_theory

[2] https://en.wikipedia.org/wiki/Pathfinding

[3] https://en.wikipedia.org/wiki/Glossary_of_graph_theory#Weighted_graphs_and_networks

High level project structure

This document outlines an initial plan for the main components of my NEA, and how these will interact.

The frontend

The frontend will be a web app built using JavaScript, HTML5 and CSS3 technologies. It will

- Display a map that is scalable and translateable
- Communicate with the backend using an API
- Support a variety of devices such as desktops, tablets and smartphones

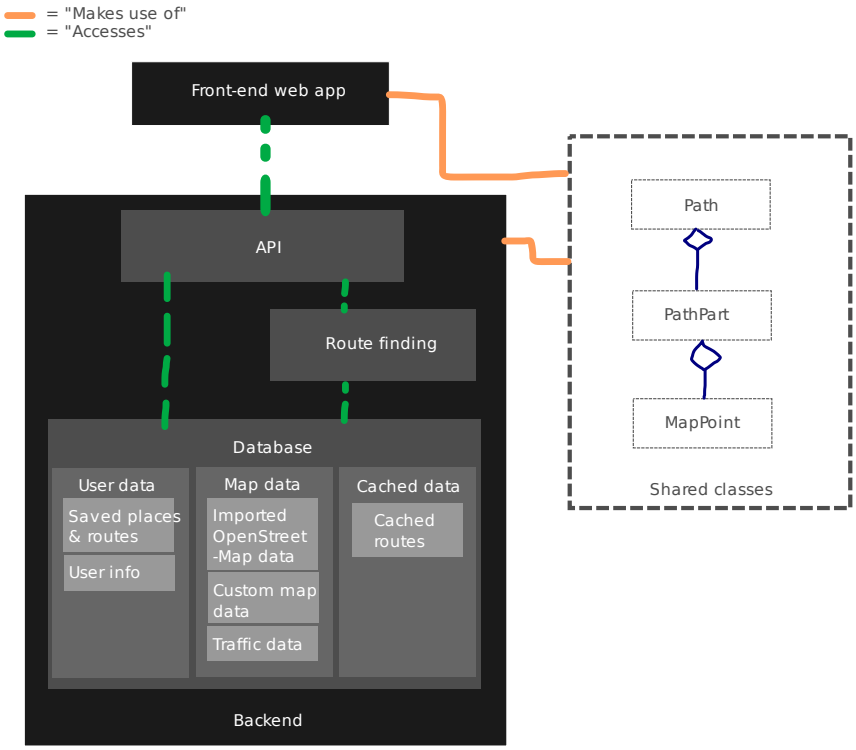

Figure 2: High level project structure diagram

It will share many of the data structures that are used on the backend representing map elements and paths. It will require extensive testing to be able to ensure it can reliably perform these functions, where serving as a web app will make user testing easier.

The backend

The backend will be built in JavaScript using the Node.JS framework. At its most basic level, it will: - Be able to read OpenStreetMap data that will be imported into a database - Serve this data via an API to the frontend - Calculate paths and serve this via an API to the frontend

It will therefore require a database and the use of path theory which I will have to research.