CS5412 Project Proposal

Marauder's Map on London Tube

[Same idea but a little less ambitious.] Or, ok, why not just actually build a cloud-hosted marauder's map to track something, like "Poker Games near me now" or "Zombie Sightings"?

Our idea comes from the idea of the "Marauder's Map in Harry Potter" and we aim to build an integrated web map visualization for optimized subway operation.

Team Member:
bx83 Bin Xu
hw727 Hongyi Wu

Project Goals and Ideas

The project focuses on several functions which could improve the passenger ride experience. The goals could be found in the following figures, including:

1. Real time train positioning tracking;
2. Comfortable information carriage detection

Background: The London Underground has its origins in the Metropolitan Railway, the world's first underground passenger railway. However, with the development of Intelligent Technology, the underground system can be monitored with more advanced IoT systems with sensors. For example, there is no visualization of a bunch of tubes including their speeds, current position and other information. We design a real time system to track, send, track, and display the information of the underground by their carriages.

For the purpose of better resource allocation for the tubes in London, we propose an application for London Tube which collects the data including speed, location and passenger flow to improve the passenger experience on public transportation.

Data source exploration:
The input data includes two parts including the access records of passengers in transits and trains and the locations of the train recording by the GPS IoT devices. All of the data are simulated as we cannot get the accurate real data.

Reference:

1. Passenger flow and station information: https://tfl.gov.uk/

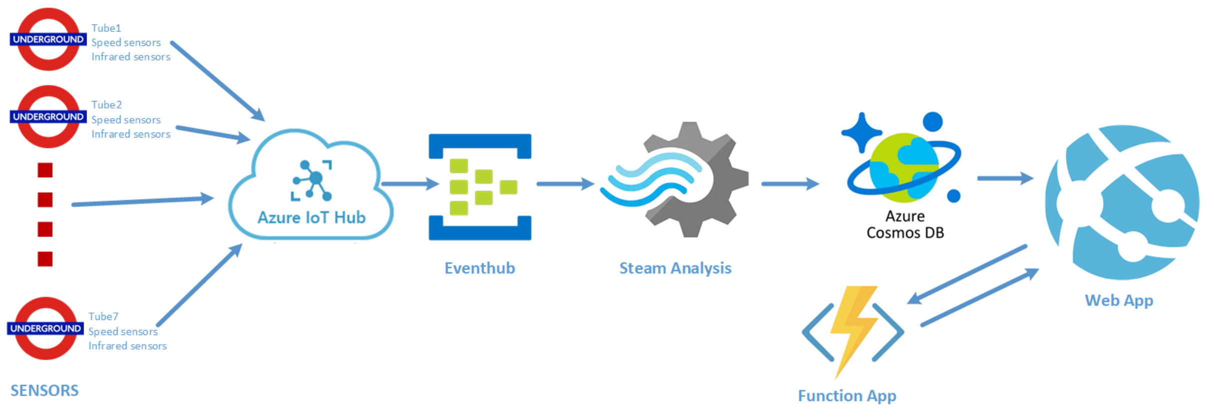
The simulation of the data is based on the basic model of uniform accelerated rectilinear motion. Therefore, the tube would stop approaching the station and come to increase in speed until getting to the maximum value.

We simulate the data only in speed as it is hard to get the accurate position by GPS in underground because of the poor signal. We then calculate the position by the algorithm in the backend. The speed and time data could be used for several times as we simulate a circular route, or a round trip when reaching the end.

Design Architecture & Implementation

Design of could platform on Azure

- Azure IoT Hub
- Cosmos DB
- Azure serverless Functions



The data was uploaded from the local IoT device to the IoT hub per second. One piece of the stream data only contains:

- timestamp: generated to store the unique collect time of the data
- line_id: The current line this tube follows
- metro_id: The unique id of the tube
- stop_id: The latest station this tube has arrived
- speed: The current speed of the tube

Then It goes to the event hub by the Endpoint.
The event hub would send the cosmos DB via Stream Analysis toolkit.
The toolkit sends the stream data to Azure Cosmos DB one by one.
Then a serverless function reads the data from the Cosmos DB and sends it to the web app via HTTP.

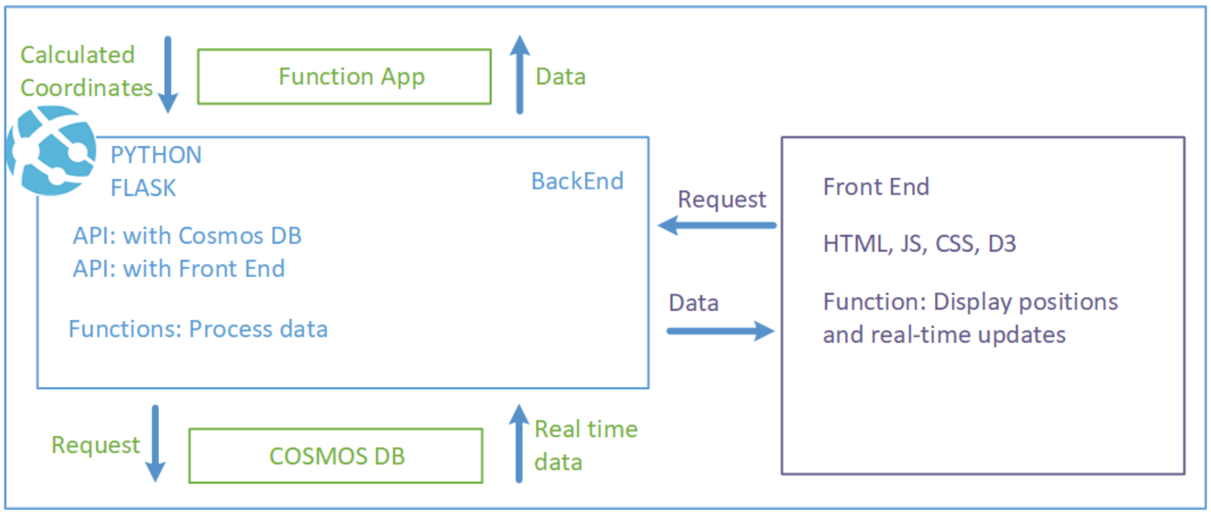
Design of Web App:

A web app is deployed to present the front and backend of the system. The azure web app has the function to deploy a python-based flask server. Flask is a popular, extensible web microframework which is suitable for a cloud service. By deploying flask, the system can easily route to other apis for post or get data.
In this case, the python flask have four functions:

1. provide an API with Cosmos DB to get the data: train ID, line ID, speed, station ID and others.
2. provide an API with Front End JS to provide data: train ID, coordinates on the map, speed, and others
3. provide an API with the Azure Function App to provide data
4. the most basic function: render websites or data through self-defined api

The front end uses simple web building tools, there are very few functions for interactive in this project. The main component is a metro line map of London. The map is simplified into seven lines and most of the stations are deleted. To make the demo simpler, one train operates on one line. The trains use a light green dot and change on the dark-color line map. Every second, the map will be refreshed with new coordinates and data. The front end is mainly built with d3.js, which is a tool to create visualization projects. All the elements are packaged and applied zoom in zoom out function for clearer vision.

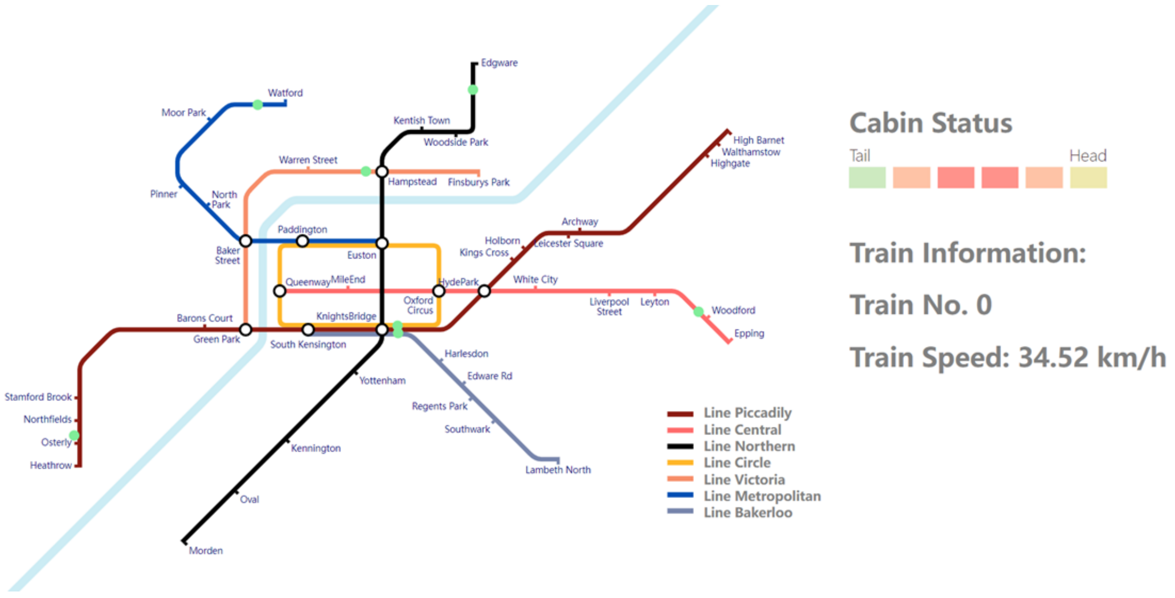
Azure web app deployment uses VS code to complete. The VS code has a special tool for app service. By simply uploading the directory, the web app could established the environment and activate the website.
Azure function app is used for calculation the coordinates. Flask send data to HTTP API and function app transfer the sensor data into visualized coordinates.



Evaluation and results:

The project primarily met the goals set for the visualization of the London tube. The performance is acceptable, for the stream data is applied every second and there are 7 metros operating on 7 lines of the tube. IoT hub is applied for live data streaming to the cosmos db. The critical function is to convert sensor data to front end coordinates. The computation process is done on Azure Function App HTTP API.

However, the estimation of the position need the history data. In this case, we use a light-weighted framework flask to allocate and distribute data. (Only distribute data and no computation). Although the front end can also perform data allocation, however we are trying to avoid any situations where the front /back end gets heavy.



Cabin Status

Train Information:

Train No. 0

Train Speed: 34.52 km/h

Conclusions & Contributors

Bin Xu: development of front end (d3 visualization), development of flask (in backend), deployment of Azure web App, deployment of Azure Function App

Hongyi Wu: Simulate tube speed and data, design and implement the algorithm to calculate the position of each tube on the web, deployment of cosmos DB, IoT hub and serverless function.