**README**

San Francisco Muni application shows the real-time positions of San Francisco’s buses and trains. The application shows the position of the vehicles in the last 15 minutes. The position is updated every 15 seconds with the information from NextBus real-time data feed. The Application has been hosted on http://ec2-18-220-227-179.us-east-2.compute.amazonaws.com:8080

**Getting Started**

The application has been built using Angular.js and d3.js. Since the application has js file that accesses some local data files, a flask server aids in running the code locally. The unit tests can be executed using karma.

**San Francisco Muni**

**Prerequisites**

For the application to run successfully, the environment requires

1. Python version 2.7.6
2. NPM and Node.js

**Installing**

**To install flask:**

1. pip install flask

**To install karma:**

1. npm install karma

**Real-time Data**

NextBus provides the necessary XML/JSON feed. To understand more about the configuration request, the following documentation can be used:

<http://www.nextbus.com/xmlFeedDocs/NextBusXMLFeed.pdf>

**Deployment**

**To start the server:**

Navigate to the folder where the project has been downloaded and navigate to \sf-challenge. Open the command prompt from this location and use command *python server.py* to start the server.

**To run the application:**

On local machine: run <http://localhost:8082> on browser

On EC2 instance: run <http://ec2-18-220-227-179.us-east-2.compute.amazonaws.com:8080> on browser

**To run unit tests:**

From the command prompt, use command *karma start* to run the tests.

**Features**

1. Multiple routes can be selected and only vehicles for those selected routes are drawn on the map.
2. Vehicle position refresh every 15 seconds.
3. Zoom functionality enables to focus on specific area by scrolling up and down.

**Working of the application**

1. The base map of San Francisco is built using neighborhood.json and streets.json
2. From NextBus server, the list of routes are obtained and populated in a table.
3. For each of the routes, the path and vehicles are drawn on top of the map.
4. By selecting one or more routes from the table, the path and vehicles are displayed.
5. The position of the vehicles for the selected routes are updated every 15 seconds.
6. On selection of the route, if no vehicles are available, a message is displayed to the user saying “No vehicles found on this route”.
7. Deselecting a route will remove its path and vehicle positions on the map.