SF Performances/Movies

Problem Statement

Create a service that shows on a map where movies have been filmed in the San Francisco Bay Area. The user should be able to filter the view using autocompletetion search

Solution

Solution is based on full stack development.

Technical Specifications:

I have decide to use JavaScript for project development as I am proficient in JavaScript and also have previous experience working with JavaScript for full stack development. Due to time constraints, it is more appropriate to choose technology stack that I am already aware of so as to minimize learning time. It makes use of google map APIs for displaying movie locations on google map.

Front- End Development:

- HTML, CSS, JavaScript for developing front-end application. The current application is single page application with one index.html file and referencing CSS and JavaScript files located in separate directories.
- The project requirement is to create a simple service so I decide to go for simple technology stack as HTML, CSS, JavaScript. As project grows more powerful UI frameworks can be implemented such as JQuery UI etc.
- It can be structured with Model View Controller pattern by using JavaScript frameworks such as Backbone.js.
- As project scope is limited , it is possible to separate logic , responsibilities through different JavaScript(.js) files. Use of framework for small application may unnecessarily increase memory requirements and cause performance threats.
- Separation of logic is achieved by maintaining different .js files for different responsibilities. This makes sure that each module is highly cohesive and does not depend on other module. Communication between these .js files is carried out by JQuery trigger and on event which results in low coupling.

- File Structure:
    a. *Index.html* – Main html web page which loads entire application. It refers to different JavaScript modules
        i. *socket_communication.js* – establishes socket event that listens to port 3000 .Handles communication between front-end and back-end with socket events.
        ii. *ui_utilities.js*

– is responsible for handling User Interface related activities such as extracting textbox values , providing autocomplete functionality for textbox etc.

iii. *map.js*
– makes use of google map apis for initialization of google map, geocoding locations to get latitude and longitude, displaying markers for calculated latitude and longitude. Also provides clear marker function in order to remove already existing markers before plotting new markers

iv. Data exchange between these three modules is carried out by *JQuery .trigger () and .on ()* event. Providing event based communication between modules helps to reduce dependency of modules on each other thus helps in maintaining low coupling.

v. Separation of responsibilities such as UI related activities in one module, socket communication responsibilities in other module etc. makes module highly cohesive. Thus makes codebase highly maintainable and scalable.

vi. In future, it is easy to add different modules for additional responsibilities and establish communication through JQuery event.

External JavaScript Resources:

i. *Socket.io.js*
– When we install socket.io module on node it creates socket.io.js in node_modules under socket.io directory. This socket.io.js is referenced in front-end in order to create socket object for front-end. To establish communication between front-end and back-end there are multiple ways. It can be achieved through AJAX calls as well. I preferred socket communication as it provides two way communication whereas AJAX calls are more useful in scenarios where we receive data from server. Moreover, socket.io has both Server side Node.js library as well as client side library with similar API. It makes integration and coordination between back-end and front-end much simpler. As I have decided both front-end and Back-end to be JavaScript socket.io was best choice for communication. Also it reduces overhead caused by http requests as uses its own event based protocol which is added advantage from performance perspective.

ii. *autocomplete.js*
- In order to implement autocomplete, HTML provides <datalist>, but datalist allows search works best for populating

option where it starts with string that we provide. But not for options where string that we provide may appear anywhere in the option. So to improve autocomplete functionality , I have used open source JQuery autocomplete.js which enables to populate options when string that we enter is not just start but appears anywhere in an option.
Eg. If we have options as Spider Man and Bat Man then if we enter 'Man', HTML datalist does not provide functionality to populate these options as string 'Man' is not start of options.
Whereas, with autocomplete.js it is possible as it allows to populate option with string 'Man' appearing at any position.

Back End Development:

- Choice of Back-end was crucial step. Factors considered are time constraints, easy integration with front-end, maintainable technology stack, integration and support for database.
- With all these factors I decided to go for Node.js as it uses same Language as Front-End which makes it easy to integrate. It supports both relational database such as MySQL as well as Non-relational database such as Firebas.js or data in JSON format. With previous experience in Node.js, time required to develop node.js application was less rather than opting for new technology. If allowed more time and if scope of project grows I would definitely like to experiment different technology stack and their integration.
- File structure :
    i. *app.js*
    - node.js application which handles entire back-end responsibilities. It uses mysql module to integrate with MySQL database, socket.io module which helps to create server that listens to port 3000 and establish socket communication with front-end. It includes db_connection.js which is responsible for database credential settings and connection.

    ii. db_connection.js
    - it handles database connection responsibilities . Wherein we provide, database username, password, schema name. With provided credentials establishes connection with database.

Database:

- Data is available in .csv form. To process and retrieve information from database it was useful to use structured database. With MYSQL database it was efficient to fire SQL query and get desired results.

- I explored other data formats such as .JSON, it seems that JSON gives information about different counts. Eg. How many times particular movie name appears in database, how many times particular location string appears in database. It was not suitable for retrieving what was expected for current application.
- With MYSQL, it is easy to import .csv file and populate database. Also with SQL retrieving information was much simpler and time efficient. Also Node.js which serve as JavaScript server provides support for MySQL.
- Depending on application requirement, current technology stack, efficient and simple queries to retrieve information, I chose MySQL as database.

Installation and Project Setup (For Windows)

- *Database Setup :*
  i. Download mysql installer from https://dev.mysql.com/downloads/installer
  ii. Select installer for appropriate version of windows.
  iii. It is recommended to keep username and password as 'root', so that user does not need to change credentials in db_connection.js under KQED/Back-End.
  iv. If username and password for MySQL are different , detail information about setting credentials is given in 'Credential Settings' section
  v. Import SQL database from file 'kqed_data.sql'
  vi. In order to import from .csv to mysql it is required to have primary key field in table structure. To achieve, additional column 'Record Number' is added to already existing database and new database named 'Data.csv' is established.

- *Node.js setup*:
  i. Download node.js from https://nodejs.org/en/
  ii. Run the installer (.msi) file
  iii. Save the node setup to appropriate directory.
  iv. Check if node is installed or not with command :
    - node -v
  v. check if node package manager which handles file structure in node.js is installed or not with command :
    - npm -v
  vi. if everything is setup , for current project we need to additional modules to be installed on node :
    a. *mysql* – to establish communication between Node.js and MySQL database
      - command : npm install mysql
    b. *socket.io* – to establish communication between front-end and back-end
      - command : npm install socket.io

- *Credential Settings* :
  i. In ‘db_connection.js’ under KQED/Back-End, we need to provide mysql database credentials. Provide user, password as mysql username and password.
  ii. In my local setup both username and password are ‘root’. It might change from user to user depending on local installation and setup of mysql

How to run an Application

- To start sever, go to directory where ‘app.js’ exists and run command.
  - *command : node app.js*
- Node.js servers on port 3000 for current application, we can change port number if required
- To load application in browser enter url:
  - url : localhost:3000/index.html
- Once application loads, we can enter movie name in provided textbox. Autocomplete functionality provides list of suggestions (*Refer Fig.1*)

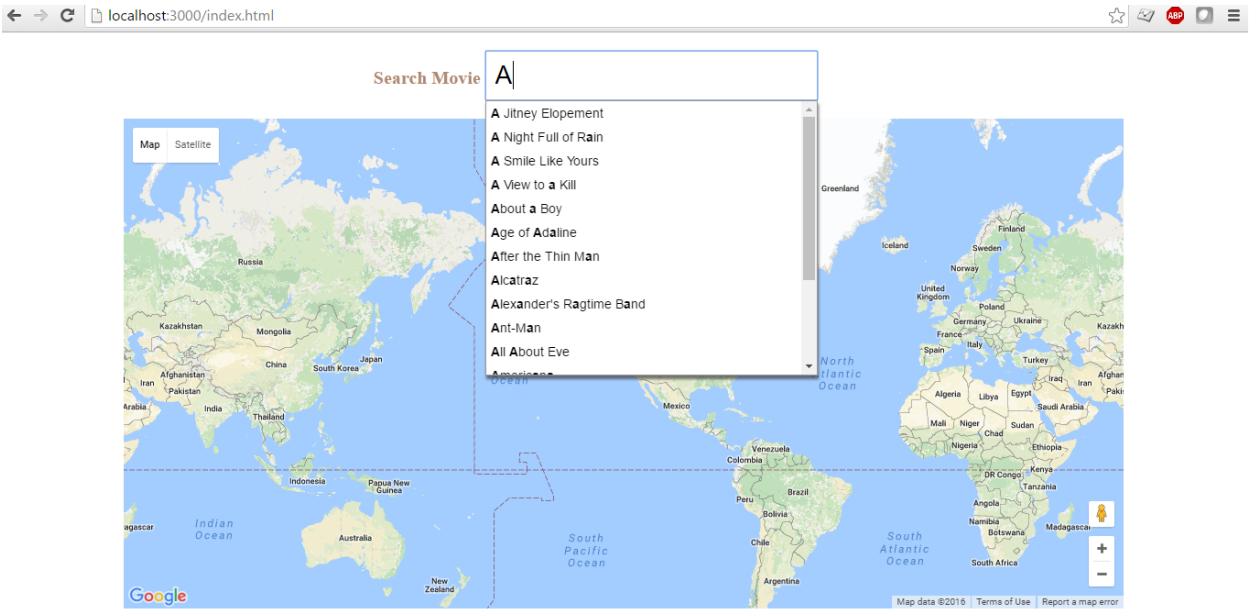

Fig.1: Illustrates autocomplete feature

- Google map shows the location markers where particular movie was shot. We can click on marker and see the detail name and description of location. (Refer Fig.2)

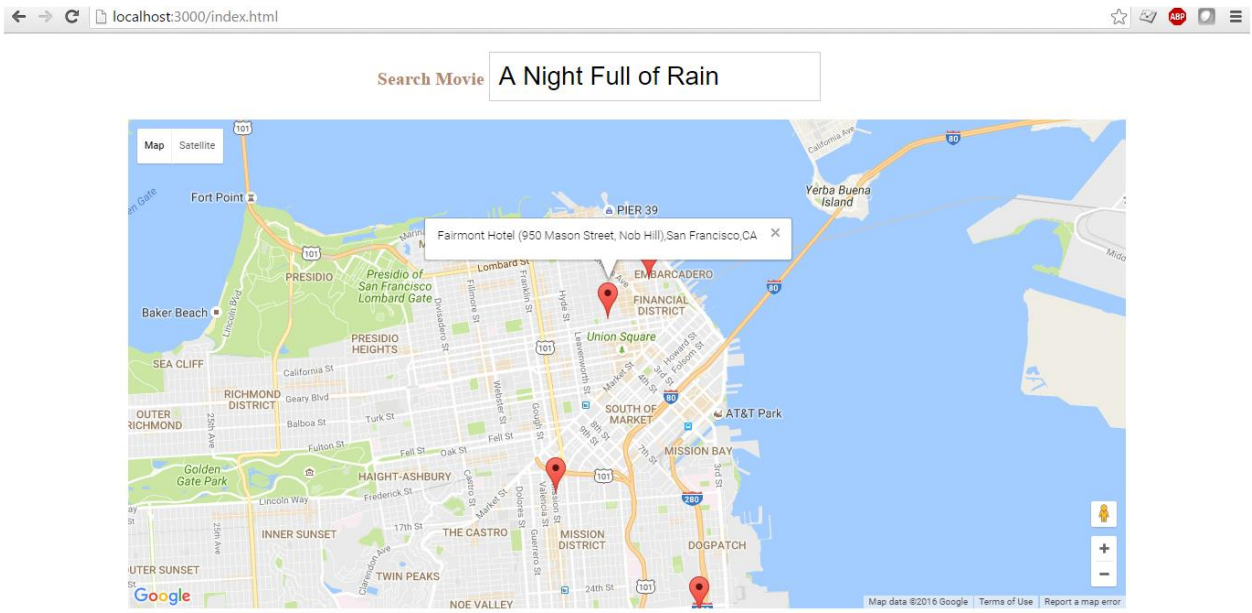

Fig. 2: Illustrates location markers feature

Future Improvements

- As scope of application grows in future , it can be structured with JavaScript frameworks such as Angular.js, Backbone.js to manage application in more precise way with model-view-control pattern
- In database there are few locations which are not in google map recognizable format. They consists of phrases like 'from street xyz …..to street abc'. Google Maps can't recognize these locations in order to geocode them and return corresponding latitude and longitude. So in future, we need to manually parse these locations. Get street names mentioned in phrase and try to find out suitable co-ordinate that lies between these street names and display on google map.
- Database contains lot of information like Fun Facts, Release Year etc. which can be used to improve application and make User Interface more informative thus increasing usability of an application.

Link to Amazon EC2: *http://ec2-54-244-109-55.us-west-2.compute.amazonaws.com:3000/index.html*

Link to GitHub project: *https://github.com/inamdarapurva/KQED*

Link to public profile: *https://www.linkedin.com/in/inamdarapurva*