Technical Manual

Project 7: Visualising our Transport Networks
Client: Chris Vallyon
Date: 11 October 2019

Overview

The Technical Manual is code documentation for out project. It describes the code architecture and semantics of the program. It includes some code snippets to describe code behaviour.

The application was developed using a three layer architecture.

- Application layer - Front-end built with `React.js`
- Middleware - Business Logic built with `Java Spring`
- Data layer - Data managed with an `SQL Database`

SQL Database - Data Layer (Backend)

The SQL database is a core part of the application as it holds all necessary data; travel times, demographic information etc. to support the application. The main queries will be processed based on travel time.

- The data component uses MySQL data management tool.
- This requires the installation of MySQL server in the hosting server.
- The user name and password of the MySQL server should be root and password respectively.
- Once the MySQL server is set up, a database named react_spring should be created.
- The backend schema should follow the same table name and column names as mentioned in the below sample schema.

  Sample database schema

- user:

user_id	email_id	password
7	project7	$2a$10$vuDcy6bbNMpAVw3h1dLnVe2mXQVSuHp7IDC6z0PaysWWlUqz3zl06

- role:

roleid	role
2	ADMIN

- user_role:

user_id	roleid

user_id	roleid
7	2

- bus_from

lat	lng	duration	distance	city	date	time
-41.2792895	-41.2792895	3494	25057	Wellington	2018-11-28 00:00:00	0944

- demograph

area	population
Highbury	3120

Server/Middleware Component

The middleware contains classes responsible for loading data from the SQL Database in the backend and passing it the front-end Application using Spring Rest APIs. The server establishes the connection to the database, builds query based on the user inputs from the frontend, query the database and return the result to the user terminal.

- The server component uses the spring framework.
- This requires the installation of maven to run the spring framework.
- A Java version 1.8 is required to be installed in the hosting server.

React.js - Frontend Component

The frontend contains modular classes and components that enables the user to interact with the application. This includes logging in, interacting with the Map using filters and conditions, displaying a graph and a menubar for additional information.

- The frontend uses the react componenet.
- This requires the installation of node package manager (npm).

Getting set up

This section provides detailed, step-by-step system operating instructions.

As the systems uses a three-layer architecture, it introduces dependencies between each of those layers. In this case, the frontend depends on the middleware architecure which inturn depends on the backend database. So the application reqiures the processes in the below mentioned order.

- Database - reuires to start the MySQL server (ex: /etc/init.d/mysql start)
- Middleware - requires to start the executable jar using Maven (ex: mvn clean install && java -jar target/visualising-transport-networks-0.0.1-SNAPSHOT.jar)
- Frontend - requires to start the react-app using node manager (ex: npm start)

React.js

With exception to data storage, the entire system is built as one single application using the current version of the React framework. Future development of this application will need take versioning into consideration as React.js is a frequently updated framework and some components/syntax may or may not be compatible with future versions of React.

This application currently uses React version: **16.9.0**

React is a very flexible library for buliding JavaScript web applications. The whole UI is divided into different components.

For documentation for React, see: https://reactjs.org/docs/getting-started.html

React Component Lifecycle

Lifecycle methods are a set of functions called during different phases of the component lifecycle. The main React lifecycle method used in this project.

`ComponentDidMount()`

This method is called right after UI is first presented on the screen. All the Spring API calls and loading the data from the SQL Database is done in this method.

`ComponentWillMount()`

This is call before the UI is rendered so that data is fully loaded and will not cause the application to display and function incorrectly.

`ComponentDidUpdate()`

This method is called when after the component updated. This is useful to derive the state from props if the props do not match the previous state.

`ComponentWillUnMount()`

This is the last method called before the component unmounts and any subscriptions clean up.

```
componentDidMount() {
  var geojson;
  // Black and white tile layer
  let mapboxLayer = L.tileLayer(orgURL, {
    zoom: 10,
    maxZoom: 18,
    id: 'mapbox.high-contrast',
    accessToken: MAPBOX_KEY
  });

  map = L.map(mapid, {
    layers: [mapboxLayer]
  }).setView([-41.2858, 174.78682], 14);
```

Mapbox and Leaflet

Mapbox is an opensource mapping platform used in this application to visualise a map appropriate to the clients wants and needs.

Leaflet is an opensource javascript library and we used their components for the legends, icons and interactions with the map.

Together Mapbox and Leaflet are used in this application to display the dynamic visualisation specific to the needs of the project scope.

To use Mapbox, a future project developer will need to make their own account and may need generate a API key, if their key is not public like the one currently used in this project.

More specific documentation can be found on the following websites:

https://www.mapbox.com/

https://leafletjs.com/

Material-UI

This application follows the by Material UI design guidelines created by Google. These guidelines provides helpful attributes that allow a consistent application layout. This also enables effective user experience. This is made possible by using the Material UI React components package. This holds a bunch of reusable components that can be utilized to follow the Material UI design guidelines.

For more information, see: https://material-ui.com/.

Spring Framework

Spring framework is used to support frontend with all the data required. All the data retrieval and processing is done in this middleware component. The data is exposed via REST APIs which are called by the frontend React application.

The application currently uses the spring version 1.5.10. The application is run by using Maven. Below are some of the componenets of spring used to develop this application.

Spring-Security

Spring Security framework of Spring enables security restrictions to this Web-based application. Spring security uses AutheticationManager to handle authentication and authorization of the incoming Web request. A token generated on the frontend, is passed to the authentication API of spring, which inturn authenticates the request. It returns with a success/failure response based on authorisation.

Spring-JDBC

Spring JdbcTemplate is a powerful mechanism to connect to the database and execute SQL queries. It internally uses JDBC api and reduces the challenges in using normal Java JDBC connections like,

- The need to write a lot of code before and after executing the query, such as creating connection, statement, closing resultset, connection etc.
- The need to perform exception handling code on the database logic.

- The need to handle transaction.
- Repetition of all these codes from one to another database logic is a time consuming task.

A model created in the Spring MVC framework, would automatically create all the required dependencies (tables, columns, foreign keys etc.) in the backend MySQL database. It reduces the repetition work of maintaining the same schema both in backend and middleware.

Testing

Jacoco-plugin

Code coverage is a software metric used to measure how many lines of our code are executed during automated tests. Jacoco is used as a plugin of Spring to generated automated test reports. Running the test using JUnit will automatically set in motion the JaCoCo agent, thus, it will create a coverage report in binary format in the target directory – target/jacoco.exec. A report of the Unit test cases is generated automatically in target/site/jacoco/index.html.

Jest and Enzyme

The tests in this project are written with React libraries Jest and Enzyme.

Jest is a node-based test runner that allows fast parallel running of test is a node environment.

Enzyme is a testing utility package that makes testing React components easier. Additional to Jest which comes with React, this tool provides more streamlined methods to carry out tests.

To install Enzyme:

run: `npm install --save-dev enzyme enzyme-adapter-react-16 react-test-renderer`

To run existing test, run `npm test` at root level.

Possible Errors

Error code 404 - API not found

- The project requires the backend Spring application to be up and running to support the frontend application.
- Failure to start the backend would result in this error.
- The solution is to start the server/spring application before starting the client/react componenet.

JDBC Communications link failure

- Failure to start the backend database results in this error.
- The user is required to start the backend MySQL server before starting the server/spring componenet of the application.

Unknown database react_spring

- The application uses a backend database named react_spring.
- Failure to create this database on the backend will result in this error.

- The user is expected to run the SQL scripts provided as part of the handover process or is expected to create the databsae react_spring manually in the backend MySQL data managemet tool.

Invalid user name or password

- Entering bad credentials on the login screen pops up this error message.
- The user is required to enter the correct credentials.