Meet app

Project Overview

- The Meet app is a serverless, progressive web app (PWA) built with React using a test-driven development technique (TDD). The app uses the Meetup API to fetch upcoming events for a selected city, and OAuth2 authentication flow. For the authorization server it uses AWS Lambda. Alerts for users are created by using an OOP approach. A service worker allows the app to work offline.
- Objective:
- To build a serverless, progressive web application (PWA) with React using a test-driven development (TDD) technique. The application uses the Google Calendar API to fetch upcoming events.

Key Features:

- Filter events by city.
- Show/hide event details.
- Specify number of events.
- Use the app when offline.
- Add an app shortcut to the home screen.
- View a chart showing the number of upcoming events by city

Technical requirements

- ● The app must be a React application.
- ● The app must be built using the TDD technique.
- ● The app must use the Google Calendar API and OAuth2 authentication flow.
- ● The app must use serverless functions (AWS lambda is preferred) for the authorization server instead of using a traditional server.
- ● The app's code must be hosted in a Git repository on GitHub.
- ● The app must work on the latest versions of Chrome, Firefox, Safari, Edge, and Opera, as well as on IE11.
- ● The app must display well on all screen sizes (including mobile and tablet) widths of 1920px and 320px.
- ● The app must pass Lighthouse's PWA checklist.
- ● The app must work offline or in slow network conditions with the help of a service worker.
- ● Users may be able to install the app on desktop and add the app to their home screen on mobile.
- ● The app must be deployed on GitHub Pages.

● The API call must use React axios and async/await.
● The app must implement an alert system using an OOP approach to show information to the user.
● The app must make use of data visualization (recharts preferred).
● The app must be covered by tests with a coverage rate >= 90%.
● The app must be monitored using an online monitoring tool.

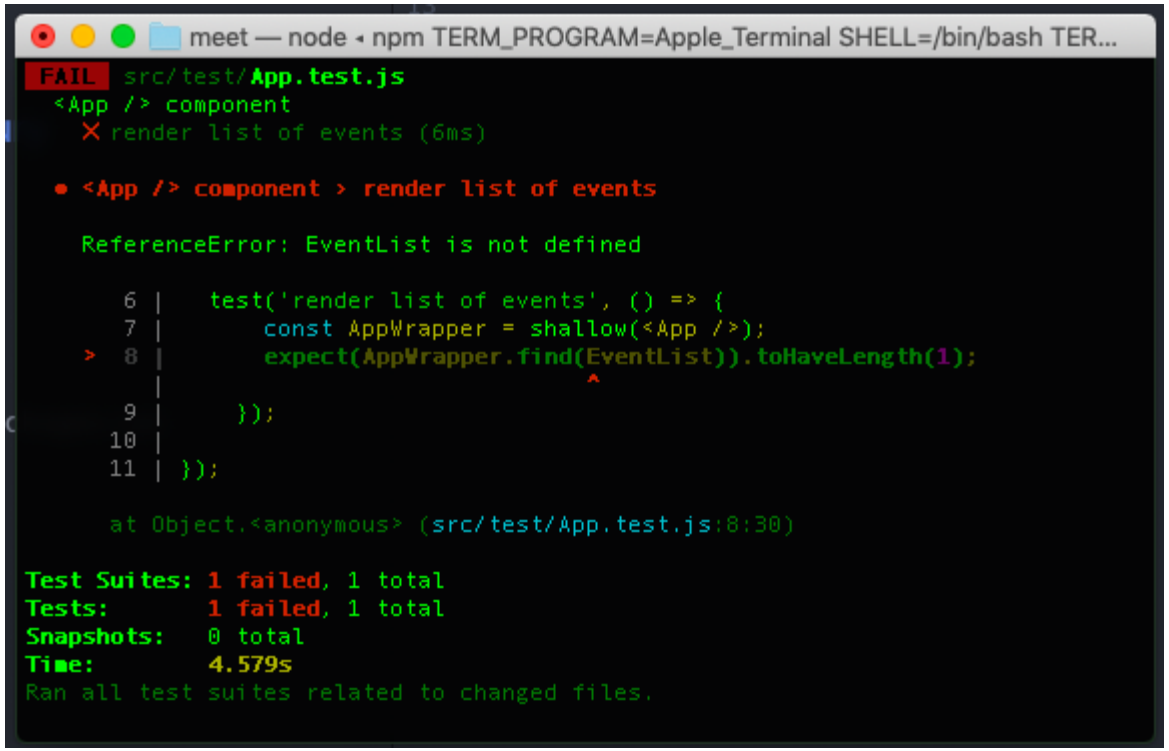
Purpose and Context/Objective

- Purpose and Context:
- The Meet app was a personal project I built as part of my web development course at CareerFoundry to demonstrate my mastery of full-stack JavaScript development.

- Objective:
- The aim of the project was to have an ambitious full-stack project I can add to my professional portfolio. The problem I wanted to solve is to build an app that allows users to be able to look up a city and than look up the and join events in a city.

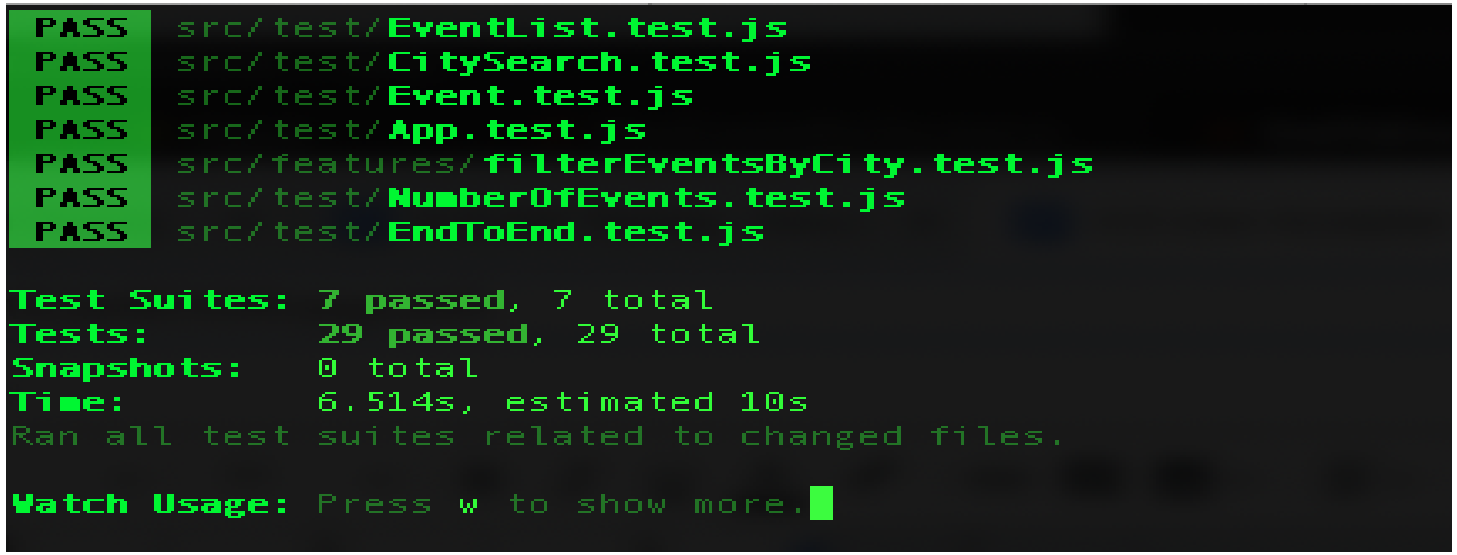
Unit Testing

- UNIT TESTING is a type of software testing where individual units or components of a software are tested. The purpose is to validate that each unit of the software code performs as expected. Unit Testing is done during the development (coding phase) of an application by the developers.



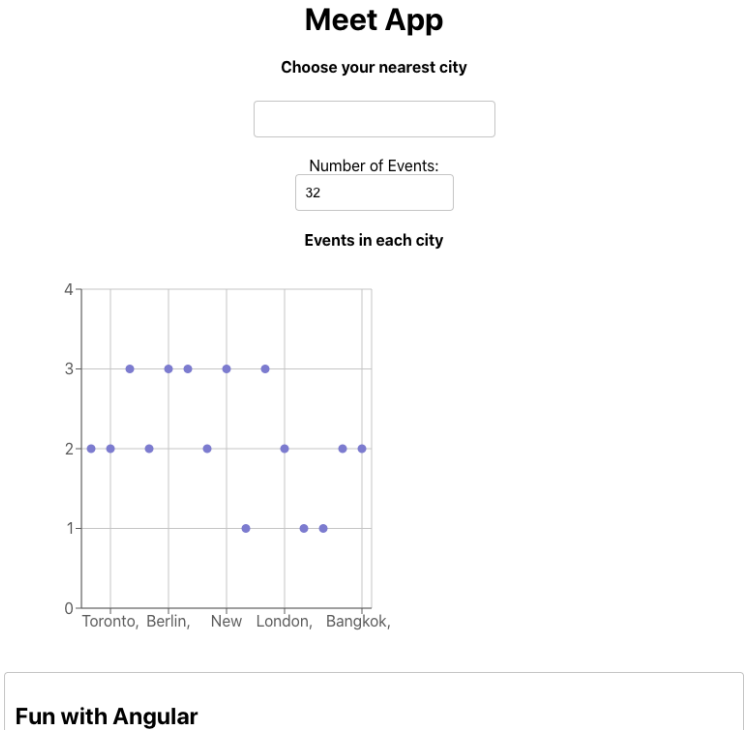
End-to-End Testing

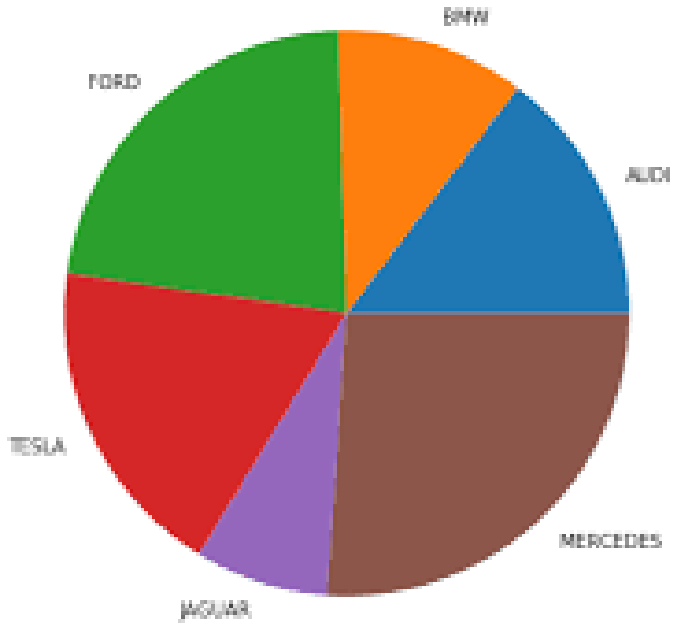
- End-to-end testing is a methodology used in the software development lifecycle to test the functionality and performance of an application under product-like circumstances and data to replicate live settings. The goal is to simulate what a real user scenario looks like from start to finish.



Data Visualization

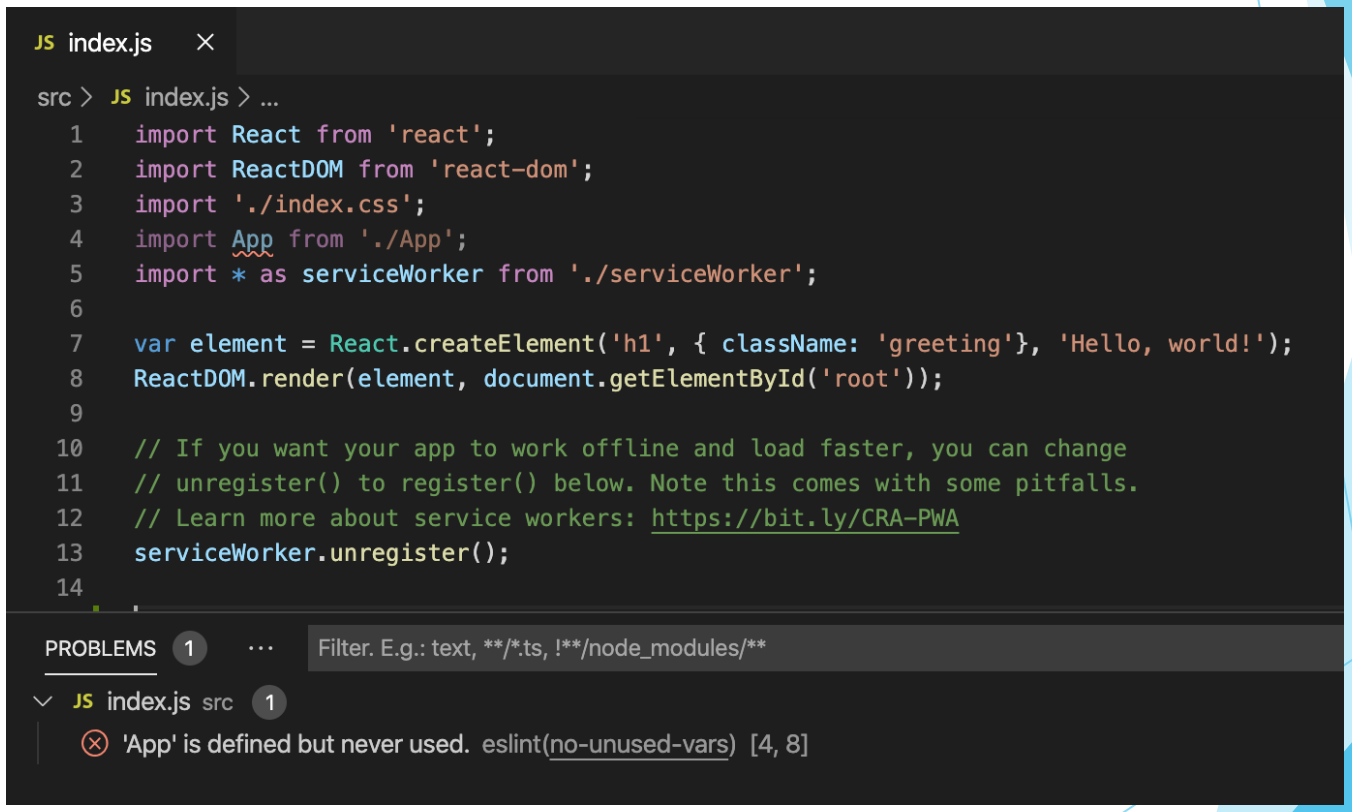
- Data Visualization is the process of communicating complex information with simple graphics and charts. Data Visualization has the power to tell data-driven stories while allowing people to see patterns and relationships found in data.





Technologies used

- Google Calendar API and OAuth2 authentication
- React
- AWS lambda
- Axios



- React:
- React JS is a JavaScript library used in web development to build interactive elements on websites. React will design simple views for each state in your application and will efficiently update and render just the right components when your data changes.

- Example:
- import React, { Component } from 'react';
- import Event from './Event';

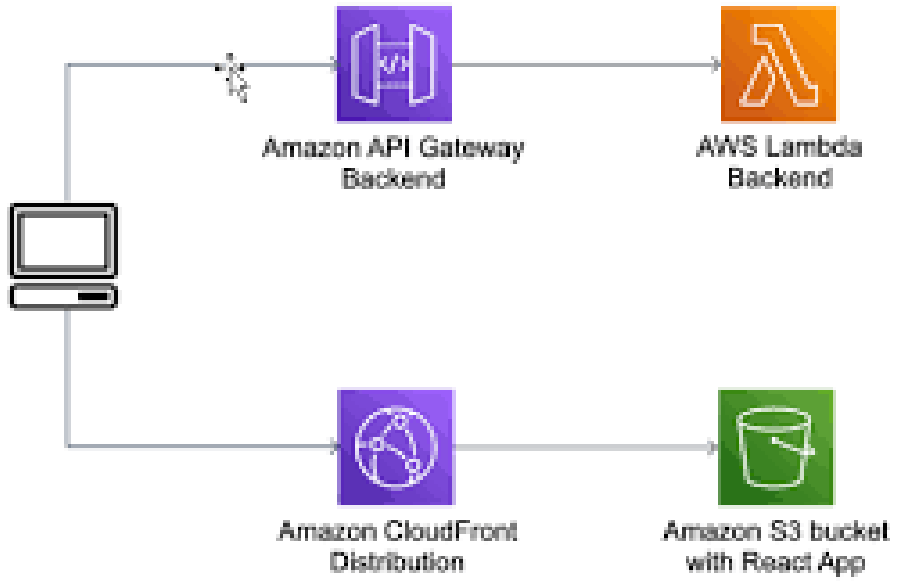
- class EventList extends Component {
- render() {
- const { events } = this.props;
- return (
- <ul className='EventList'>
- {events.map((event) => (

- Axios in React:
- It is a library which is used to make requests to an API, return data from the API, and then do things with that data in our React application. Axios is a very popular HTTP client, which allows us to make HTTP requests from the browser.

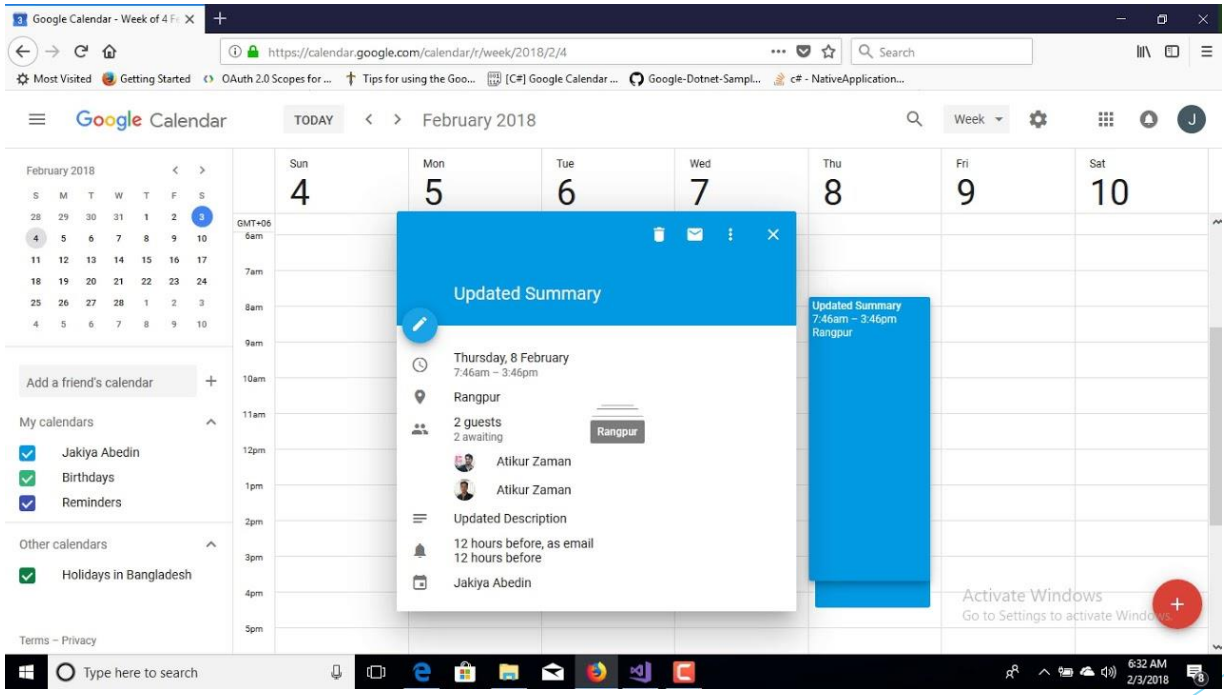
```
onSubmit(e) {
        e.preventDefault();
        const serverport = {
            name: this.state.name,
            port: this.state.port
        }
        axios.post('http://localhost:4000/serverport/add', serverport)
        .then(res => console.log(res.data));

        this.setState({
            name: '',
            port: ''
        });
    }
```

- AWS Lambda:
- AWS Lambda is a serverless compute service that runs your code in response to events and automatically manages the underlying compute resources for you. You can use AWS Lambda to extend other AWS services with custom logic or create your own back-end services that operate at AWS scale, performance, and security.

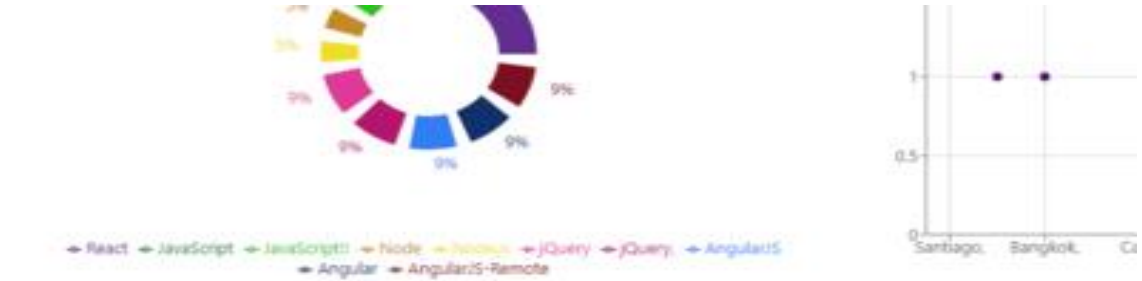


- Google Calendar API and OAuth2 authentication:
- The Google Calendar API enables developers to add full calendar data and functionality into their app using a REST interface, or through one of the client libraries Google offers for languages like Java, Python, PHP, JavaScript, and more.
- OAuth 2.0 allows users to share specific data with an application while keeping their usernames, passwords, and other information private.



My Application

- Here is my link to my GitHub repository:
- https://1998-creator.github.io/meet/



jQuery and More

Santiago, Santiago Metropolitan Region, Chile
Thu Jul 22 2021
19:00 (Eastern Daylight Time)
Do you know jQuery is used by around 70 percent of the 10 million most popular websites as of May 2019? Though many consider it dead after A
part of many websites. In our workshop, we teach basic to advanced jQuery where you will also be able to build a simple app using it. If you are fa
See details on Google Calendar

React California

California, USA
Thu Jul 22 2021
21:00 (Eastern Daylight Time)

Challenges

- Through working on this project on my windows pc, I came across various errors that were no fault of my own and took time to work around and caused frustration.
- My pc was also very slow, so it was frustrating having to wait long times when running the various tests.
- At one point my pc was running so slow that my integration test was failing, and I had to move over to another pc. This was the first time I had to set up my environment on another pc.
- This was also the first time I ran tests on my code, so it was a challenge to understand what the tests did while also coding them into the project.

Retrospective

- What went well:
- After the completion of the project, no errors were found. All test were approved after running the tests. The site is very simple for users to select a city and see the events going on in that city. Also, having to move to another pc was a good learning experience for me in having to set up my environment for that project.
- What didn’t go well:
- As stated before, my pc was very slow causing me to have to continue my project on another pc.
- What can be improved:
- More cities and events can be added to the database for a more immervise experience for the user. Also, some more CSS stylings to brighten up the app for the user experience.