Backend – Achievement 4 – Testing in the Development Process

1) Description of the project

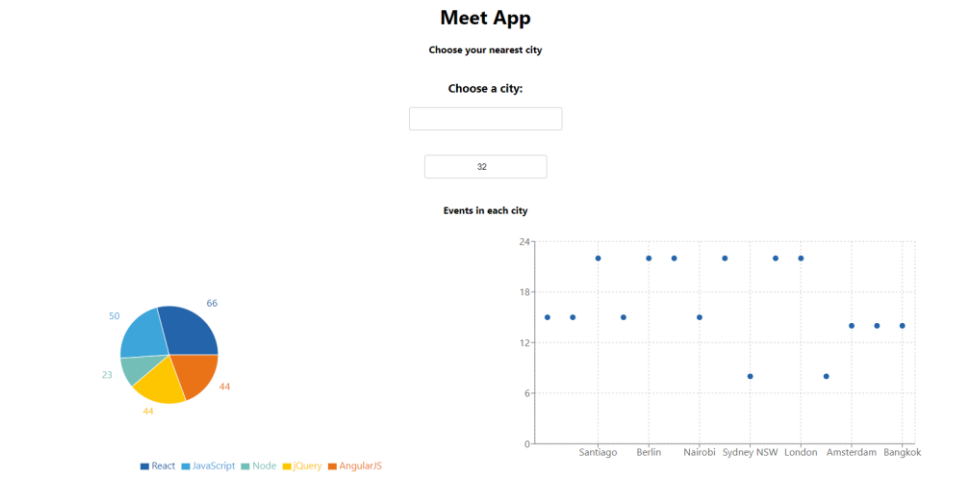
- What was my role for this project and what tasks did I face?
  - built serverless, progressive web app (MEET) that fetches data from the Google Calendar API
    - Serverless: No backend maintenance, easy to scale, always available, no cost for idle time
    - PWAs: Instant loading, offline support, push notifications, “add to home screen” prompt, responsive design, and cross-platform compatibility
- 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 is a React application
  - The app is built using the TDD technique
  - The app uses the Google Calendar API and OAuth2 authentication flow
  - The app uses serverless functions (AWS lambda is preferred) for the authorisation server instead of using a traditional server
  - The app’s code is hosted in a Git repository on GitHub
  - The app works on the latest versions of Chrome, Firefox, Safari, Edge, and Opera, as well as on IE11
  - The app displays well on all screen sizes (including mobile and tablet) widths of 1920px and 320px
  - The app passes Lighthouse’s PWA checklist
  - The app works offline or in slow network conditions with the help of a service worker
  - The users are able to install the app on desktop and add the app to their home screen on mobile
  - The API call used React axios and async/await
  - The app implements an alert system using an OOP approach to show information to the user
  - The app makes use of data visualization
  - The app is covered by tests with a coverage rate >= 90%
  - The app is monitored using an online monitoring tool

- Lessons I learned / decisions I made during this project
  - Wrote user stories based on the app’s key features
  - Translated user stories for each feature into multiple test scenarios
  - Used create-react-app to create a React application and push it to GitHub
  - Obtained a consumer secret from the Google Calendar API
  - Wrote a simple serverless function to refresh the access token
  - Called this function from a static page
  - Used test scenarios, wrote frontend unit tests using mock data for the app’s key features
  - Developed another feature for the app

- Wrote integration tests to test the interaction between the app's React components
- Wrote integration tests to test the data received from the mock API
- Wrote user acceptance tests for two key features, covering all defined test scenarios
- Set up app monitoring for the application to monitor its performance
- Used an OOP approach to create alerts for the application
- Learned advantages and disadvantages of regular web apps and native apps
- Used a service worker to ensure your app works offline
- Used a "manifest.json" file to make my app installable
- Showed a notification to the user to inform them the app is working offline (when the user is offline)
- Added charts, such as a ScatterChart, to my app's UI to visualize data using the recharts library
- Made visualizations responsive

2) A screenshot to represent the project



3) A link to the project's GitHub repository

- https://github.com/Luisa-Inc/meet
- User Stories and Scenarios

4) A link to the live, hosted version of my app

- https://luisa-inc.github.io/meet/

5) A list of the technologies used for each project

- React
- TDD
- Google Calendar API
- OAuth2 flow
- AWS Lambda
- Autorisation server
- Enzyme
- Jest
- Passing Lighthouse's PWA checklist
- React Axios and async/await
- alert system using an OOP approach

6) Any other relevant materials I created for the project

Serverless function

- Created a serverless deployment package with AWS Lambda
- Using the Serverless Toolkit and the Google Calendar API
- own authorization server for issuing OAuth2 access tokens
- Created an OAuth Consumer
- Setup an AWS Lambda Function
- Created an Authentication Server
- Created a Serverless Service
- Configured AWS Credentials
- Deployed the Authentication Process
- Wrote functions:
    - GET AUTH URL: https://59i7ltvzyg.execute-api.eu-central-1.amazonaws.com/dev/api/get-auth-url
    - return: USER AUTHORIZATION CODE: {"authUrl":"https://accounts.google.com/o/oauth2/v2/auth?access_type=offline&scope=https%3A%2F%2Fwww.googleapis.com%2Fauth%2Fcalendar.readonly&response_type=code&client_id=250900529821-rit2jpo282rokk6p0vdrbuo6rcnq0f9o.apps.googleusercontent.com&redirect_uri=https%3A%2F%2FLuisa-Inc.github.io%2Fmeet%2F"}

    - GET TOKEN: https://59i7ltvzyg.execute-api.eu-central-1.amazonaws.com/dev/api/token/{code}

    - return: ACCESS TOKEN: {"access_token":"ya29.a0AVvZVso71dic2NfpOw726rnwnLPYTLoQ5_yf8Pd4kHjjKAM-IlNERg7VONhC7OYRLPMPasmIySwszNTbRCPpL6u2oti2R2BKUFzGInfEafTLylY12EC3T9-K1N8-SKrgpevm5E7TglZsQK7_TrSwta4dTqZpOQaCgYKAcsSARISFQGbdwaI69Zl6BUbmGm3fPlseFKsGA0165","scope":"https://www.googleapis.com/auth/calendar.readonly","token_type":"Bearer","expiry_date":1675335063675}

    - GET CALENDAR EVENTS: https://59i7ltvzyg.execute-api.eu-central-1.amazonaws.com/dev/api/get-events/{access_token}

- Tested a Serverless Function Using a Static Site
    - Set up a Local Node.js HTTP Server
    - Created HTML file

- can be found here: https://coach-courses-us.s3.amazonaws.com/exercises/1114/40489/99a9b7da98553b8dfae1cded99e29b2f/test-auth-server.html