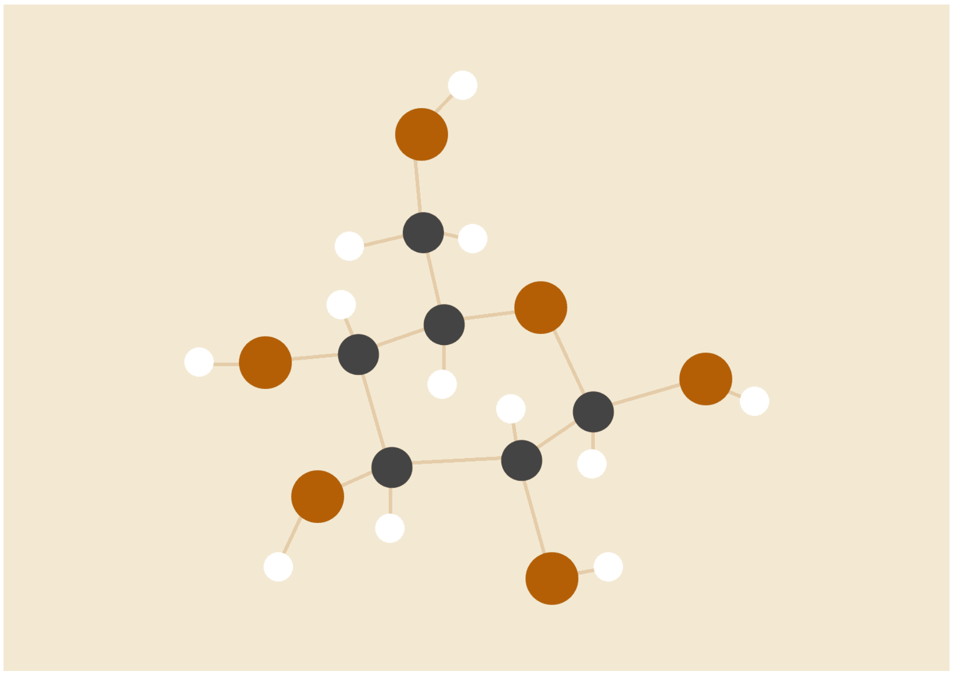
Another Covid Tracker

https://gurpinderbisla.github.io/Covid-Tracker/



Daisy Han
Gurpinder Bisla
Arminder Singh
Tom Vu

30. March 2022

CPSC 2350

Overview

Another covid tracker is a single page website that provides latest stats for active covid cases, numbers for tested, recoveries and mortality for both Canada and the world.

Upon entering the site, visitors have the option to choose between viewing data for either Canada or Worldwide. Within either components, data is displayed in number, chart, graph and map forms, to enhance the user experience and interactivity.

This website is built with React.js and its various libraries, such as Chakra,js, Charts.js, React Router. We implemented CI/CD pipeline directly with Github Action, and deployed the site using Github Pages. The testing was done with Jest and React testing library.

Reflection on SDLC

We chose Agile and Kanban framework as our SDLC. We were partially successful in following all of the Agile principles. As we started working on the project a bit later than planned. We started later because some of us were still learning React, so during this process, we did not actively work on the project.

Agile focuses on an iterative approach to push out codes as frequently as possible. We were able to really focus on the project and deliver functional features during the last couple of weeks of the project. On the other hand, we were good with another Agile principle, which is being flexible with changes, and adapting our approaches accordingly. This includes being flexible with work assignment, not strictly following the timeline set out in the WBS, and assigning work to whoever has the time and skill to work on some features. We also reschedule meetings depending on members' schedules. Moreover, we were able to change the global api because of its request restrictions only due to the agile model we choose.

Another thing we could have done better is providing each other consistent feedback, so we could improve on the way we work. Though we did not have many formal feedback-providing moments, we did utilize slack and Zoom quite frequently to update what each other is working on, and assist each other with debugging and offer suggestions when needed.

Overall, it was our first attempt using Agile to work on a group project, though we were not perfect at it, we still abided by some of the core principles. The lessons we gather from this experience would help with our future projects.

High-level view of features

Open covid Canada API

https://api.opencovid.ca/

- Stat boxes
- Bar chart for cumulative cases
- Map for active cases
- Vaccination data
- Vaccination line chart

Global covid API

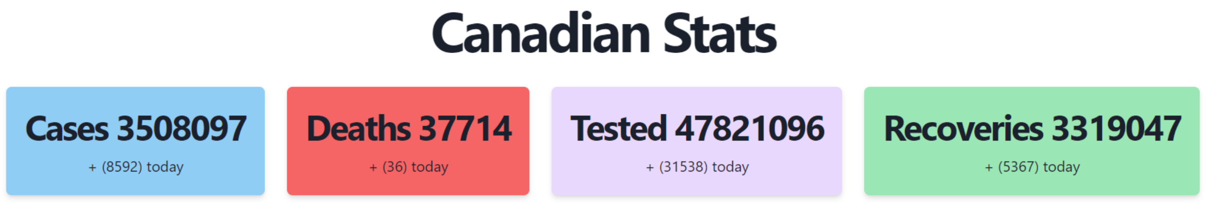
https://api.covid19api.com/summary

- Global cumulative covid stats
- Stats by country
- Line chart for specific country with data since the beginning of covid
- Worldwide map for cumulative cases
- Comparison tool for two selected countries

Detailed view on application features per API

https://api.opencovid.ca/

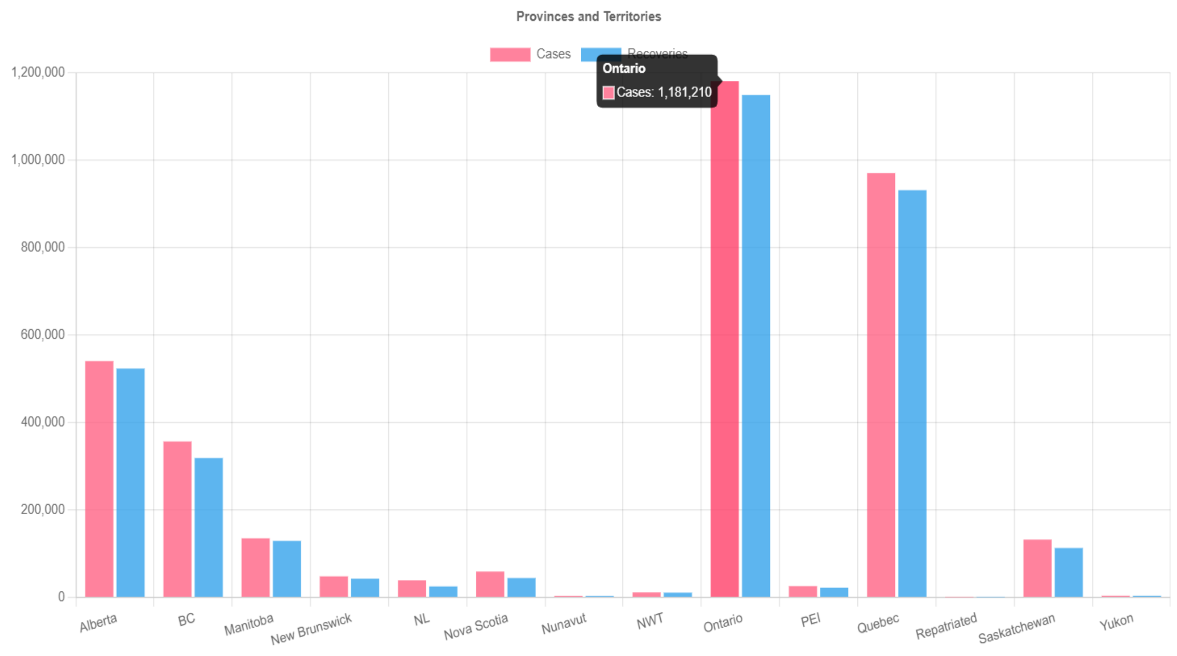
1. Data cards: Get Canadian cumulative stats for active cases, deaths, mortalities, tested and recovered and display them for Canada provincial page.



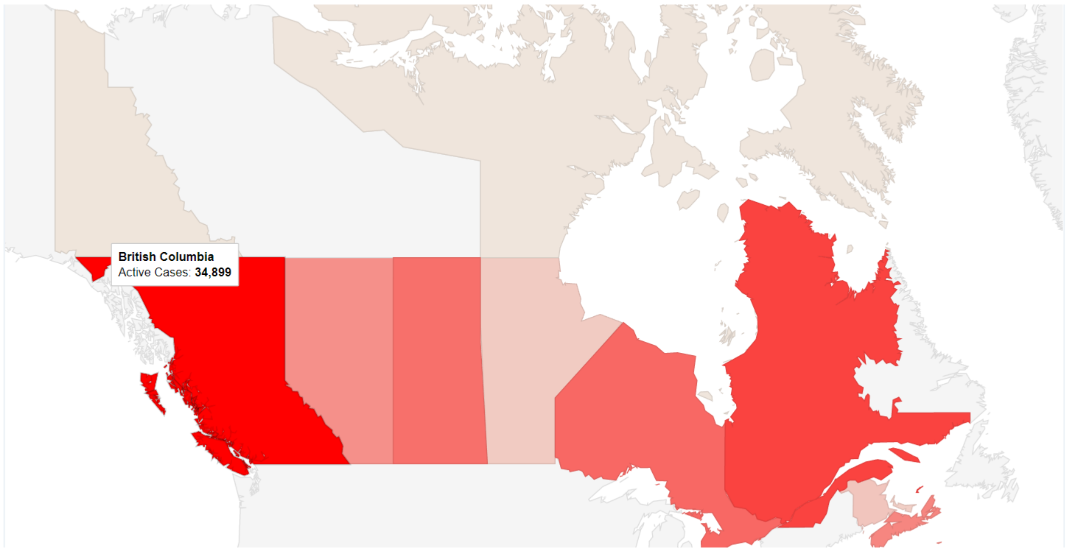
Canadian Stats

Cases 3508097	Deaths 37714	Tested 47821096	Recoveries 3319047
+ (8592) today	+ (36) today	+ (31538) today	+ (5367) today

2. Bar chart: get cumulative cases and recovered cases based on province whenever the user hovers over the bar chart.

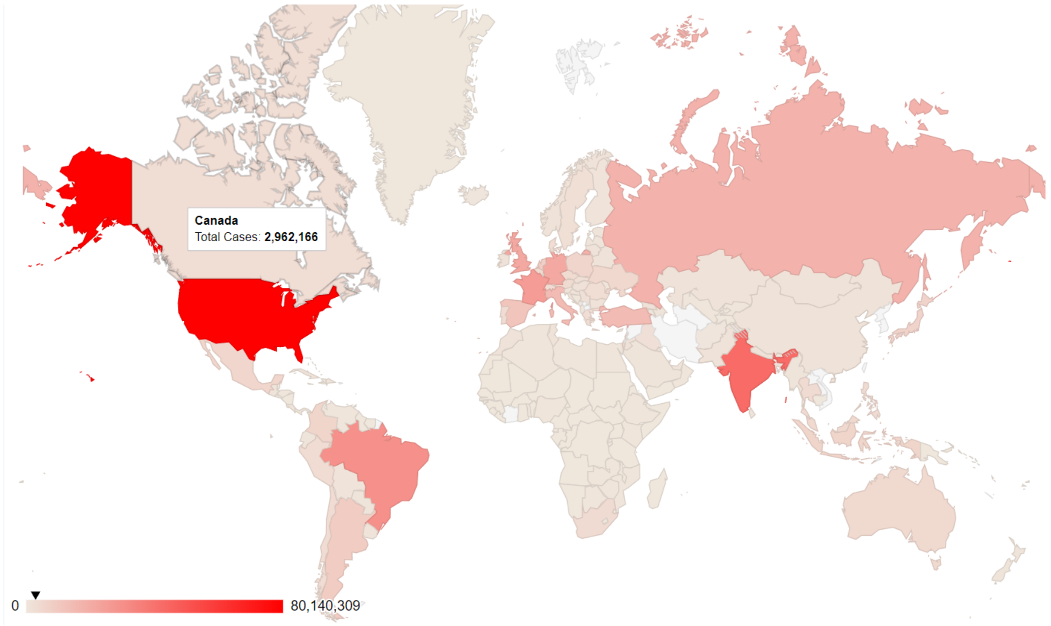


3. Map. get provincial active cases whenever the user hovers over a province in the map.

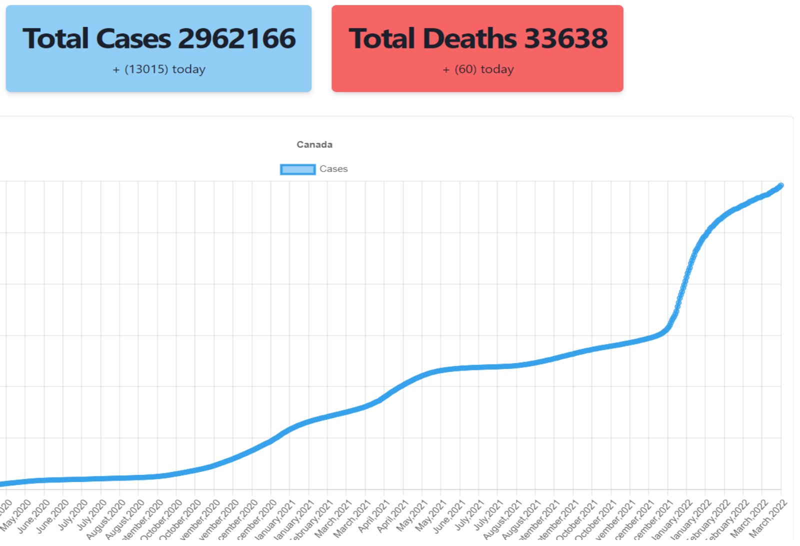


https://api.covid19api.com/summary

1. Global map: on the map, depending on which country is hovered over, the map displays the cumulative covid cases for that country.



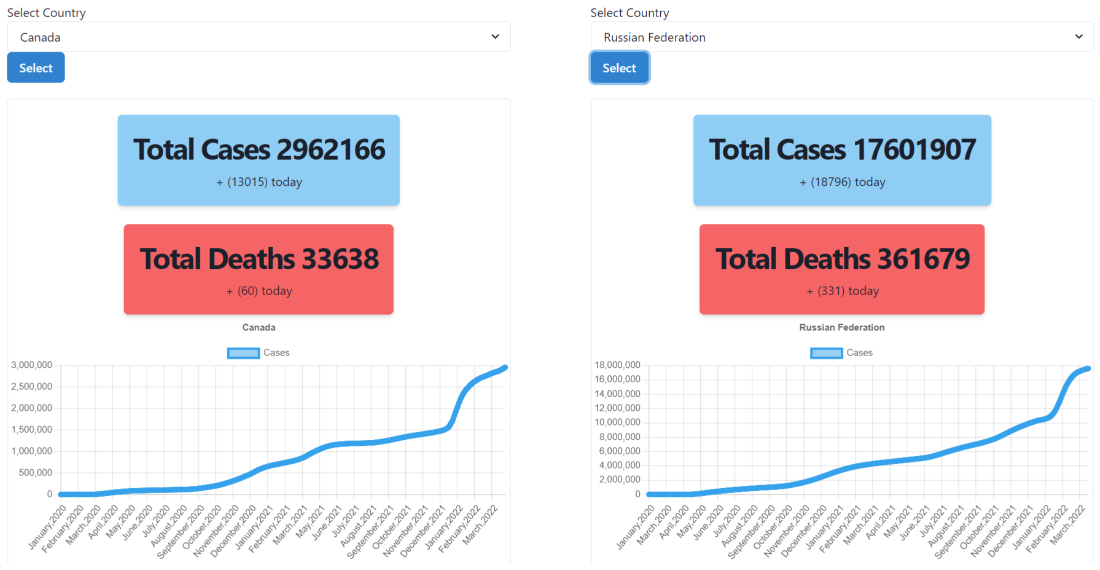
2. Country picker and line chart: once a country is chosen from the list of countries, the line graph displays the cumulative cases since the beginning of covid in line chart form.



Total Cases 2962166
+ (13015) today

Total Deaths 33638
+ (60) today

3. Country comparison tool: by selecting two countries, users can compare the cumulative covid cases since the start of covid side by side in chart form.



Select Country
Canada
Select

Select Country
Russian Federation
Select

Total Cases 2962166
+ (13015) today

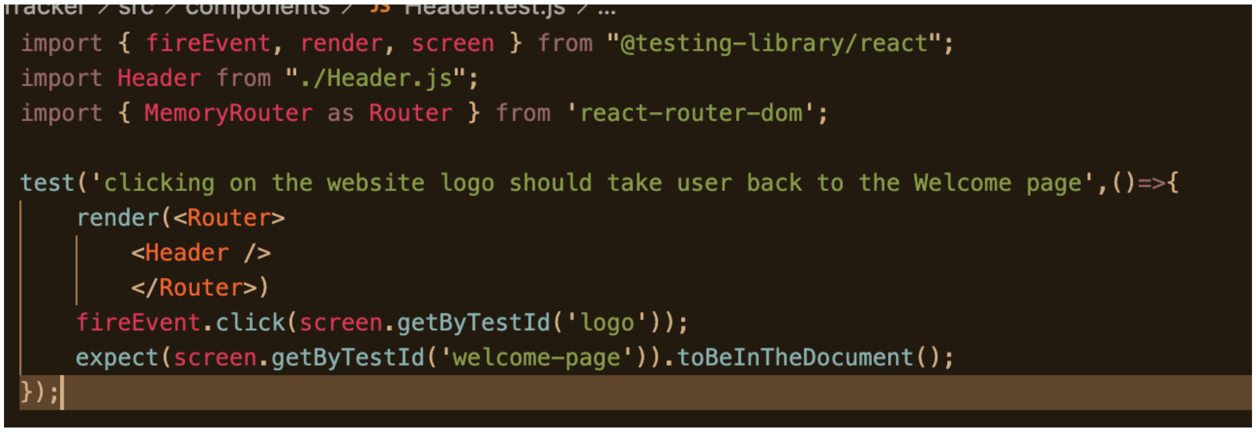
Total Deaths 33638
+ (60) today

Total Cases 17601907
+ (18796) today

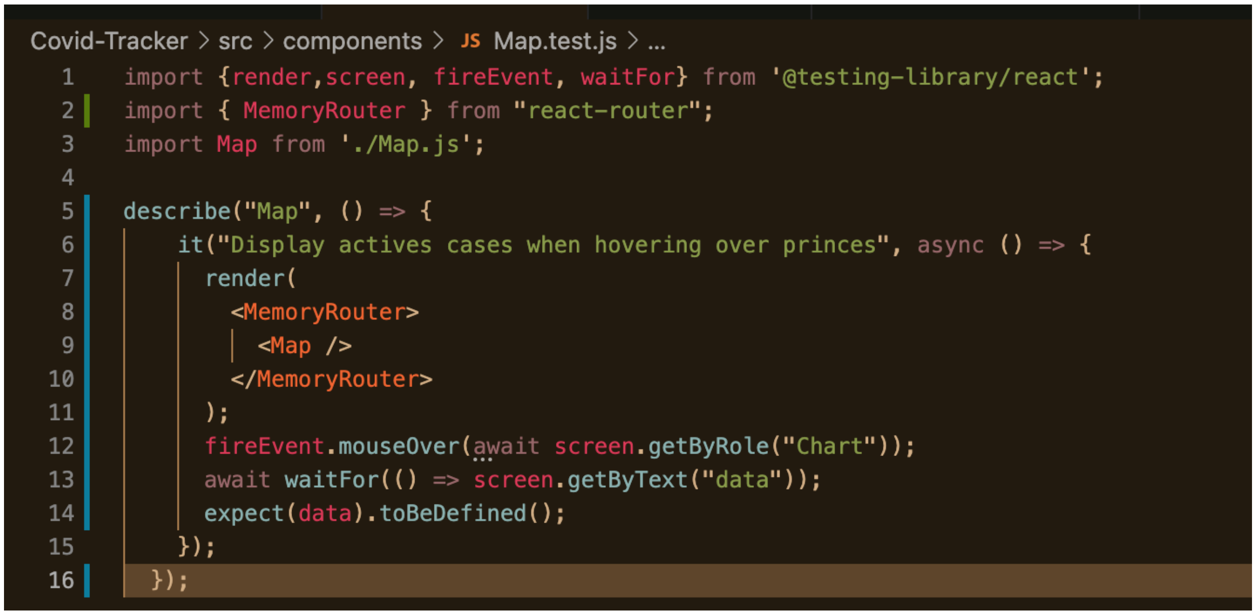
Total Deaths 361679
+ (331) today

Tests

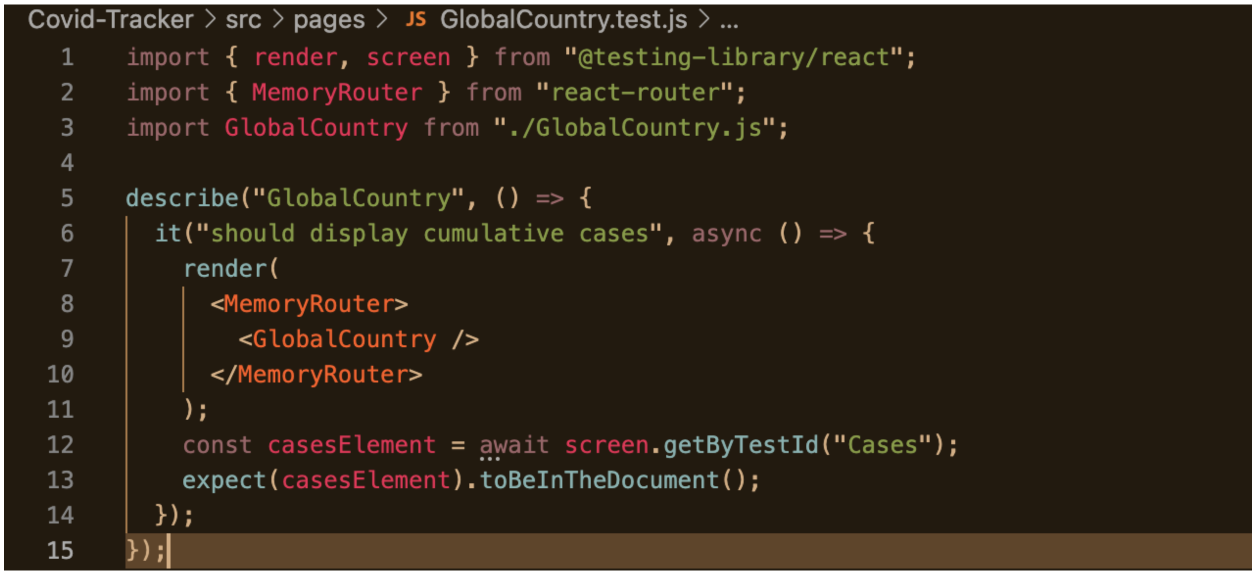
Unit Test


Tracker > src > components > JS Header.test.js > ...

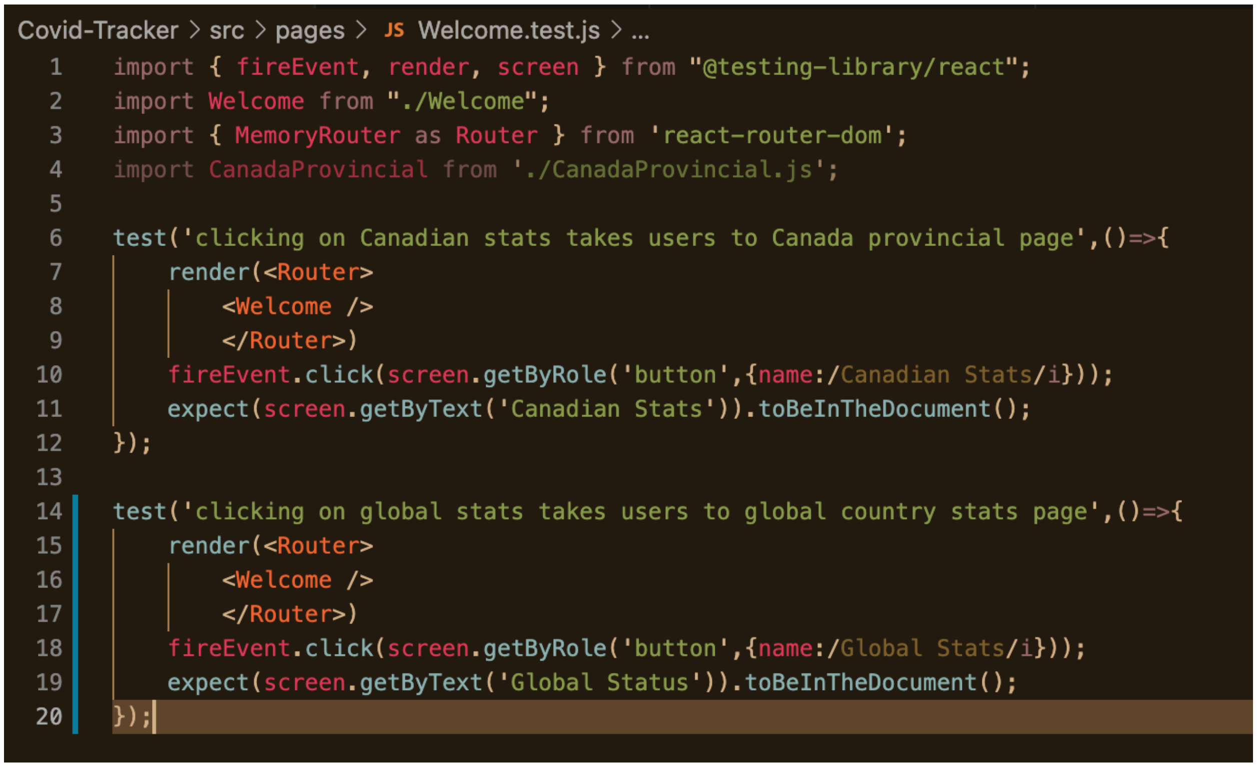
```
import { fireEvent, render, screen } from "@testing-library/react";
import Header from "./Header.js";
import { MemoryRouter as Router } from 'react-router-dom';

test('clicking on the website logo should take user back to the Welcome page',()=>{
    render(<Router>
        <Header />
        </Router>)
    fireEvent.click(screen.getByTestId('logo'));
    expect(screen.getByTestId('welcome-page')).toBeInTheDocument();
});
```


Covid-Tracker > src > components > JS Map.test.js > ...

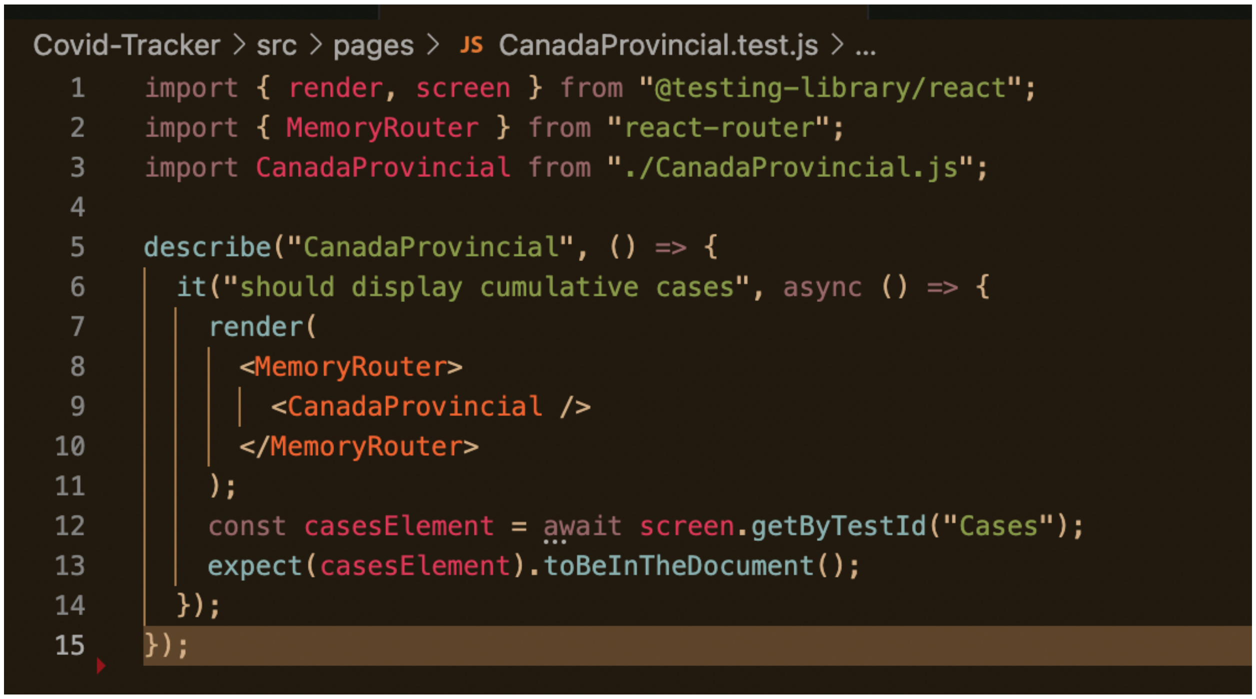
```
import {render,screen, fireEvent, waitFor} from '@testing-library/react';
import { MemoryRouter } from "react-router";
import Map from './Map.js';

describe("Map", () => {
    it("Display actives cases when hovering over princes", async () => {
      render(
        <MemoryRouter>
          <Map />
        </MemoryRouter>
      );
      fireEvent.mouseOver(await screen.getByRole("Chart"));
      await waitFor(() => screen.getByText("data"));
      expect(data).toBeDefined();
    });
  });
```


Covid-Tracker > src > pages > JS GlobalCountry.test.js > ...

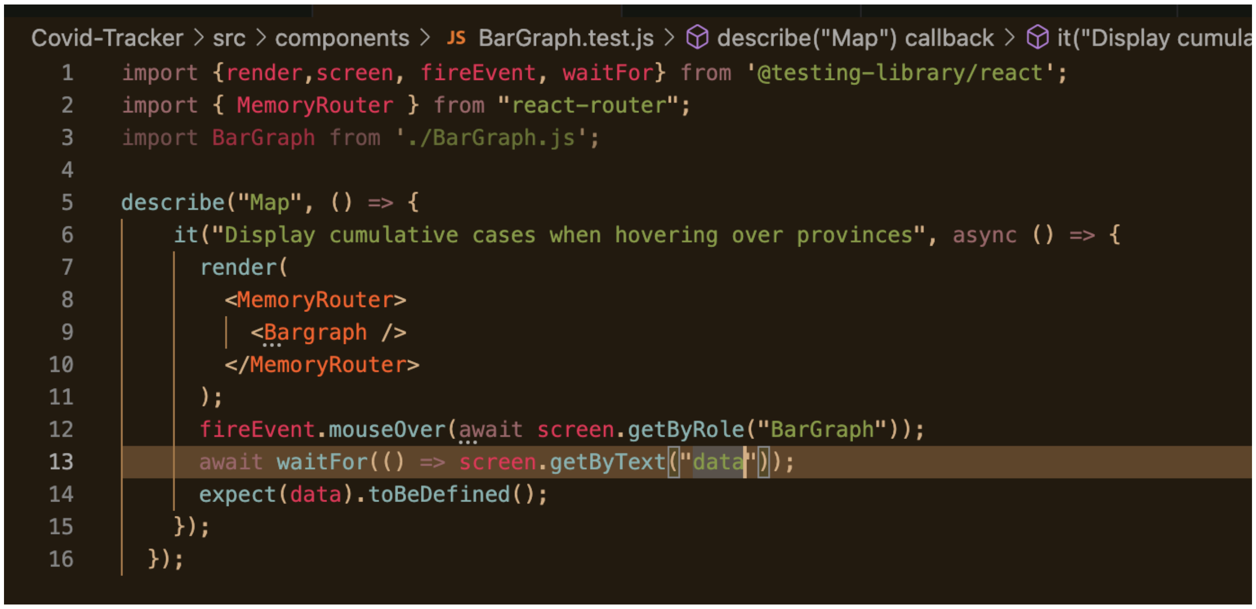
```
import { render, screen } from "@testing-library/react";
import { MemoryRouter } from "react-router";
import GlobalCountry from "./GlobalCountry.js";

describe("GlobalCountry", () => {
  it("should display cumulative cases", async () => {
    render(
      <MemoryRouter>
        <GlobalCountry />
      </MemoryRouter>
    );
    const casesElement = await screen.getByTestId("Cases");
    expect(casesElement).toBeInTheDocument();
  });
});
```


Covid-Tracker > src > pages > JS Welcome.test.js > ...

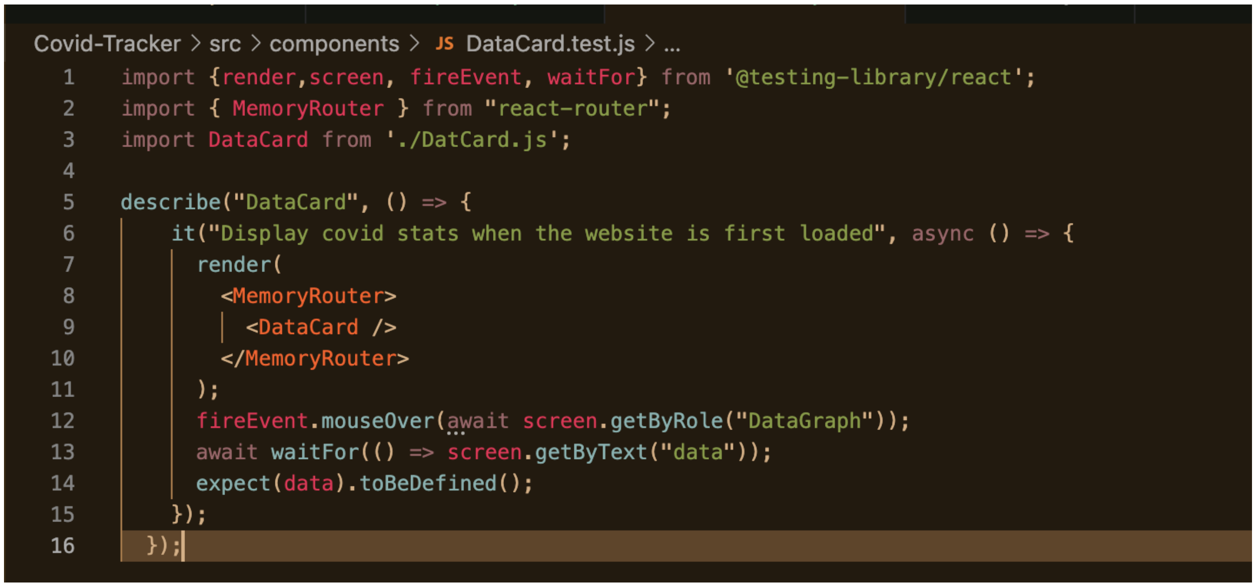
```
import { fireEvent, render, screen } from "@testing-library/react";
import Welcome from "./Welcome";
import { MemoryRouter as Router } from 'react-router-dom';
import CanadaProvincial from './CanadaProvincial.js';

test('clicking on Canadian stats takes users to Canada provincial page',()=>{
    render(<Router>
        <Welcome />
        </Router>)
    fireEvent.click(screen.getByRole('button',{name:/Canadian Stats/i}));
    expect(screen.getByText('Canadian Stats')).toBeInTheDocument();
});

test('clicking on global stats takes users to global country stats page',()=>{
    render(<Router>
        <Welcome />
        </Router>)
    fireEvent.click(screen.getByRole('button',{name:/Global Stats/i}));
    expect(screen.getByText('Global Status')).toBeInTheDocument();
});
```


Covid-Tracker > src > pages > JS CanadaProvincial.test.js > ...

```
import { render, screen } from "@testing-library/react";
import { MemoryRouter } from "react-router";
import CanadaProvincial from "./CanadaProvincial.js";

describe("CanadaProvincial", () => {
  it("should display cumulative cases", async () => {
    render(
      <MemoryRouter>
        <CanadaProvincial />
      </MemoryRouter>
    );
    const casesElement = await screen.getByTestId("Cases");
    expect(casesElement).toBeInTheDocument();
  });
});
```


Covid-Tracker > src > components > JS BarGraph.test.js > describe("Map") callback > it("Display cumula

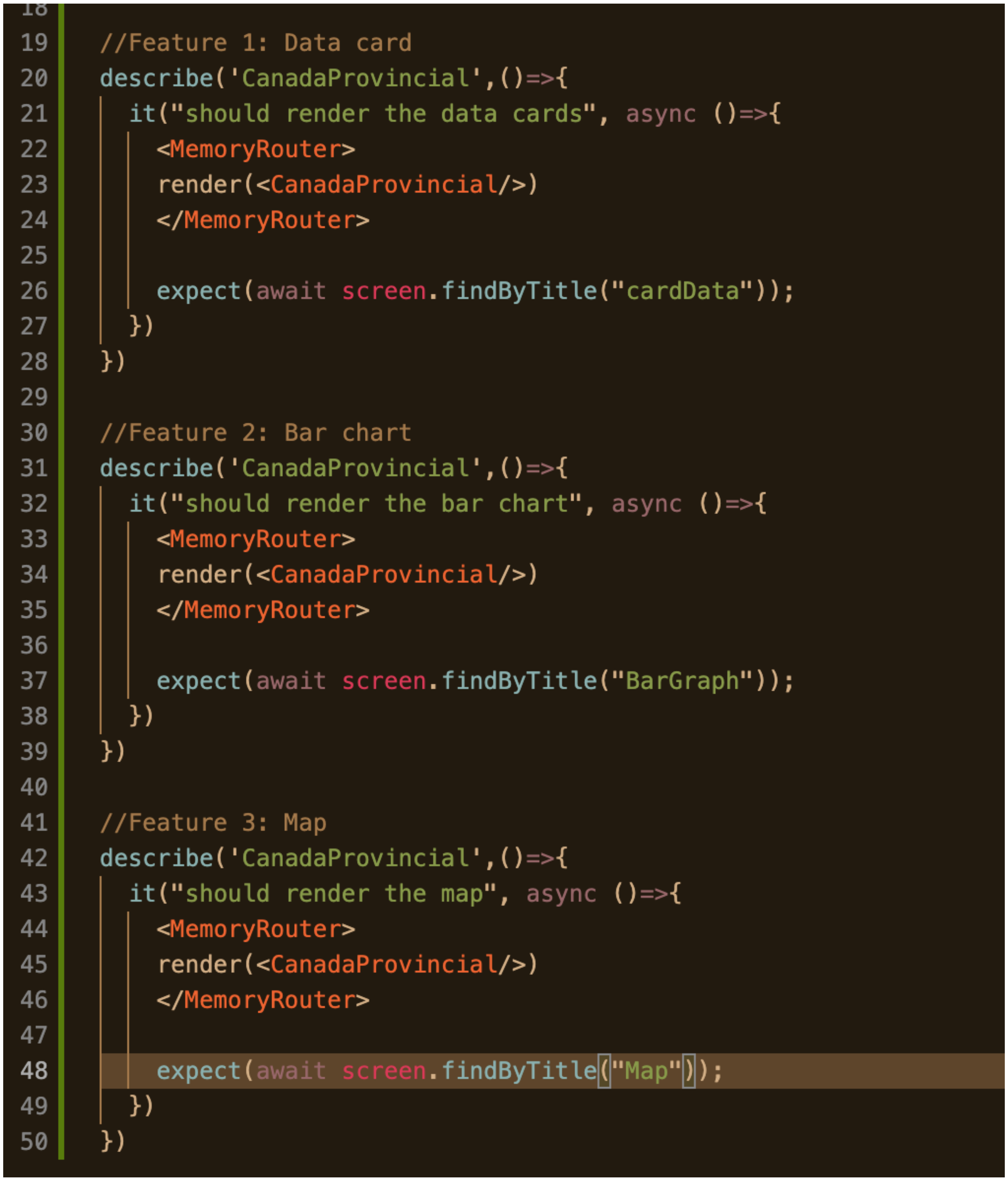
```
import {render,screen, fireEvent, waitFor} from '@testing-library/react';
import { MemoryRouter } from "react-router";
import BarGraph from './BarGraph.js';

describe("Map", () => {
    it("Display cumulative cases when hovering over provinces", async () => {
      render(
        <MemoryRouter>
          <Bargraph />
        </MemoryRouter>
      );
      fireEvent.mouseOver(await screen.getByRole("BarGraph"));
      await waitFor(() => screen.getByText("data"));
      expect(data).toBeDefined();
    });
  });
```



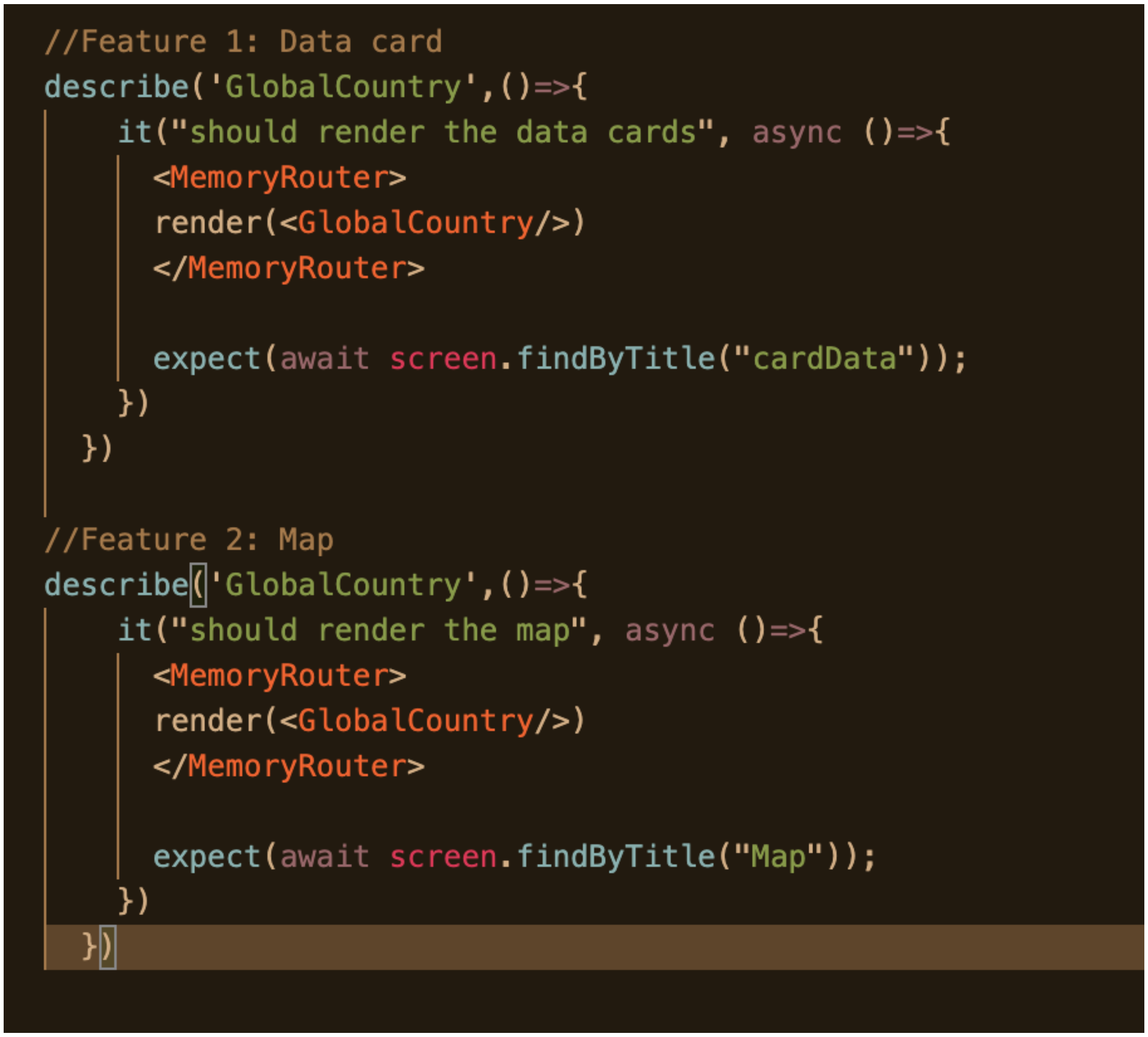
Covid-Tracker > src > components > JS DataCard.test.js > ...

```
import {render,screen, fireEvent, waitFor} from '@testing-library/react';
import { MemoryRouter } from "react-router";
import DataCard from './DatCard.js';

describe("DataCard", () => {
    it("Display covid stats when the website is first loaded", async () => {
      render(
        <MemoryRouter>
          <DataCard />
        </MemoryRouter>
      );
      fireEvent.mouseOver(await screen.getByRole("DataGraph"));
      await waitFor(() => screen.getByText("data"));
      expect(data).toBeDefined();
    });
  });
```

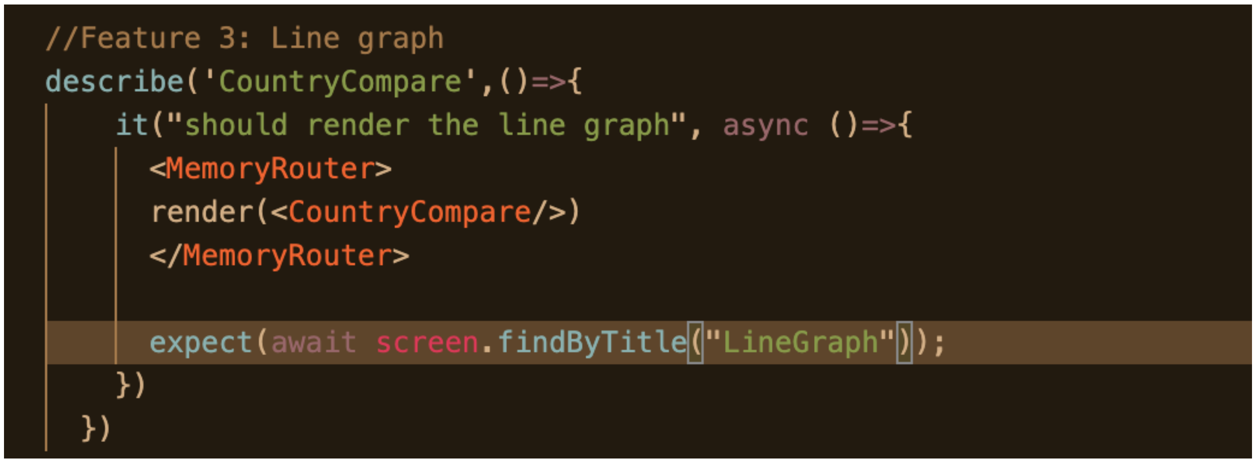
Integration test



```

//Feature 1: Data card
describe('CanadaProvincial',()=>{
  it("should render the data cards", async ()=>{
    <MemoryRouter>
    render(<CanadaProvincial/>)
    </MemoryRouter>

    expect(await screen.findByTitle("cardData"));
  })
})

//Feature 2: Bar chart
describe('CanadaProvincial',()=>{
  it("should render the bar chart", async ()=>{
    <MemoryRouter>
    render(<CanadaProvincial/>)
    </MemoryRouter>

    expect(await screen.findByTitle("BarGraph"));
  })
})

//Feature 3: Map
describe('CanadaProvincial',()=>{
  it("should render the map", async ()=>{
    <MemoryRouter>
    render(<CanadaProvincial/>)
    </MemoryRouter>

    expect(await screen.findByTitle("Map"));
  })
})
```



```
//Feature 1: Data card
describe('GlobalCountry',()=>{
    it("should render the data cards", async ()=>{
      <MemoryRouter>
      render(<GlobalCountry/>)
      </MemoryRouter>

      expect(await screen.findByTitle("cardData"));
    })
  })

//Feature 2: Map
describe('GlobalCountry',()=>{
    it("should render the map", async ()=>{
      <MemoryRouter>
      render(<GlobalCountry/>)
      </MemoryRouter>

      expect(await screen.findByTitle("Map"));
    })
  })
```



```
//Feature 3: Line graph
describe('CountryCompare',()=>{
    it("should render the line graph", async ()=>{
      <MemoryRouter>
      render(<CountryCompare/>)
      </MemoryRouter>

      expect(await screen.findByTitle("LineGraph"));
    })
  })
```

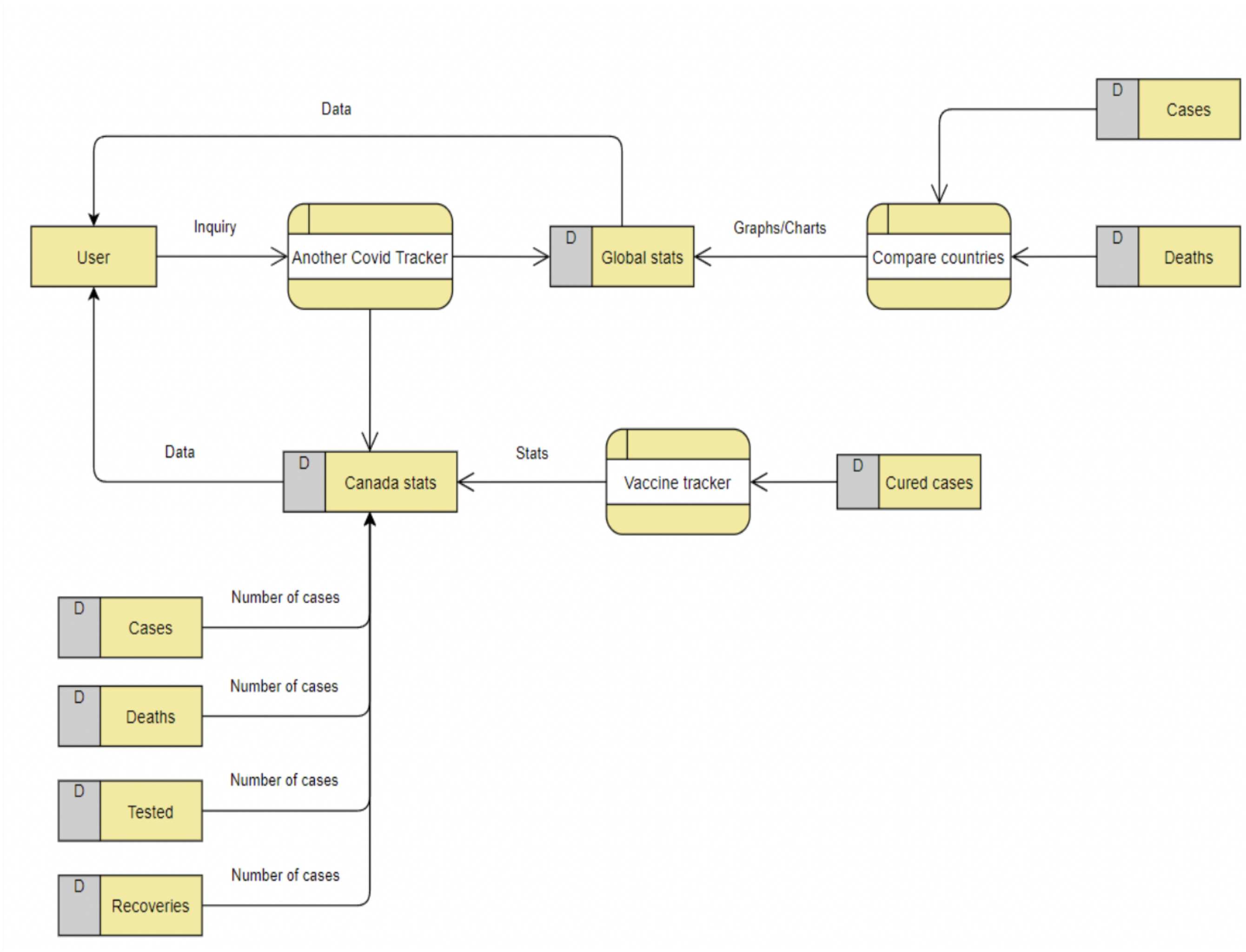
CI/CD pipeline

We chose github action as our CI/CD platform. The reason we chose this over other CI/CD tools like Jenkin or Travis CI is because github action is a built-in feature of github, this streamlines the CI/CD process as we are already using github for our version control. This allows us to simply push up our codes from our local repos to Github, and it automatically activates the CI/CD pipeline we have implemented, so we can do everything in one place, without the need to switch to a different environment, like Jenkin or so.

A closer look at our CI/CD pipeline.

In order for workflows to work, first we start creating workflow ,yml files and store them in the github folder and workflows files. We created node.js.yml for testing, building and deploying. This workflow is triggered every time there’s a push or pull request happening at the main branch. We will have the workflow run on the ubuntu host. Next comes the detailed steps for this workflow. First is checkout, which gets the code from git repo. Next we need to install the node version needed to run the code. We specified the node versions needed to increase the repeatability and predictability for the workflow to get the same results. Next it starts the building process, once that’s passed, then it will run tests to finish up. If at any step there’s any error when it runs, the response will display the error message and offer suggestions for corrections. If the previous steps passed, then our updated code will get deployed automatically to our github page. Thus completes the CI/CD pipeline.

Data flow diagram



Project Reflection

One big takeaway for all of us after working on this project is we should have started working on the project earlier than we had. This is mostly due to the fact that most members in the group don't have prior experience with using React, CI/CD pipeline, writing unit tests and such. We overlooked the fact that sometimes it could take a bit of a longer learning curve to pick up new tools. And this one of the biggest challenges for us, that we didn’t have too much knowledge and experience with the tech stack we chose. But we were all open to learning and picking up new skills, so we stuck through. In the end, despite the time crunch, we were able to pull through and delivered an app that all of us are quite proud of.

We appreciate the opportunity to work on a group coding project, for some of us, this is our first time working to build an app together. This is a refreshing experience as we get to work with teammates of different programming experience. We are able to share what we know and what we do not know, overall this process expands our programming skill. Being able to see how others write their codes when they push them to Github opens our eyes to a different way of approaching programming problems. Members with more experience shared the tools they use with members of less experience. We utilized Slack to discuss any questions and provided each other support when it comes to debugging and decision making. Overall, we concluded we all learned a lot through working on this project.

Task	Name
Implementing 6 features based on 2 APIs	Gurpinder, Arminder
Develop application interface	Gurpinder, Arminder
Unit tests and integration tests	Daisy
CI/CD	Daisy
Hosting	Gurpinder
Report	Daisy
Video	Tom
Data flow chart	Tom
Presentation	Daisy