

Coursework for Systems Testing

 Node.js CI passing

 codecov 47%

About

BMI Calculator

Weight (kg)

Height (metres)

Submit

Reload

Your BMI is:

The Body Mass Index (BMI) calculator is an application that allows the user to measure their body fat based on their height and weight. This gives the user the ability to work out if they are of a healthy weight based on

how tall the user is and it applies to both adult men and women. This BMI calculator will give you an estimated number rounded to two decimal places. Below are some of the key features I aim to display.

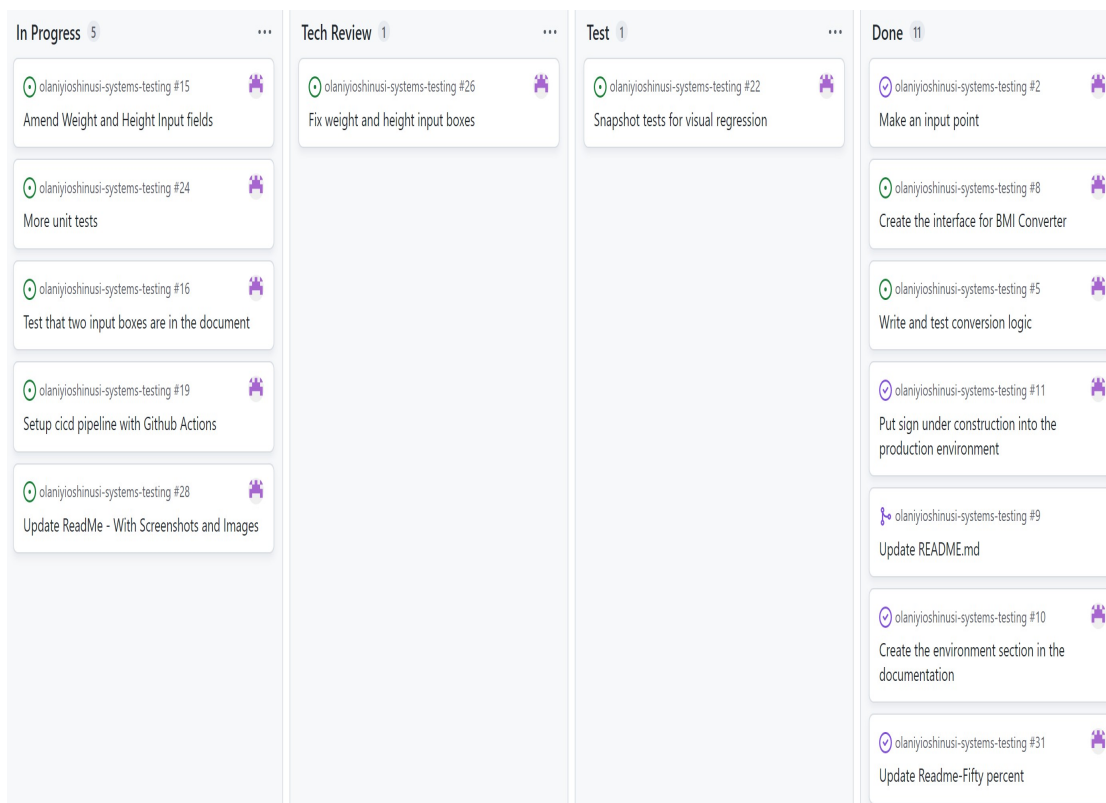
- Inputs that the BMI Calculator will accept.
- Outputs that the BMI Calculator will display.
- Accuracy testing that the value displayed is indeed correct.
- A clickable button to give the user a result.

Environments

Project has two environments, the Production environment and the Development environment. I tested features in the development environment.

- [Live App](#)
- [Dev Environment](#)

Project Management



I used GitHub's Project Tracker for project management to manage issues and used the kanban methodology to maintain a workflow where I could continuously improve, visualise my workflow on a regular basis and sort tasks in order of priority.

! Prerequisites

This project requires npm to execute the files, so ensure that it is installed.

1. Ensure node and npm are installed by running the following commands in your terminal:

```
node -v
```

```
npm -v
```

If they are not installed, follow the steps on [npm Docs](#).

2. To run end-to-end tests, please install 'serve' globally in your machine:

```
npm install -g serve
```

If you are getting some errors after using that command and you are on Linux/MacOS, try running it as a superuser (sudo)

3. Afterwards, clone this repo:

```
git clone https://github.com/EDGENortheastern/olaniyioshinusi-systems-testing.git
```

Now, you should have everything that you need to proceed! Navigate into the folder you just cloned to find our code 😊



Getting Started

0. Before trying out any of the next commands:

```
npm install
```

If you get some vulnerability warnings after executing this command, don't worry 😊, it's the React team that has to do some catch-up...

1. In the project directory, you can run:

```
npm start
```

Runs the app in the development mode. 🖱️

Open <http://localhost:3000> to view it in your browser.


The page will reload when you make changes.

You may also see any lint errors in the console.

2. To run logic and UI tests for my application, run the following command:

```
npm test
```

```
a
```

This launches the test runner in the interactive watch mode. 

Clicking on the 'a' key runs all tests (excluding End-to-End tests). See the section about [running tests](#) for more information.

3. To view full tests and code coverage for my code:

```
npm test -- --coverage --watchAll=false
```

4. To run our E2E tests:

Note that the following command might be different depending on your Operating System:


```
npm run build
```

Builds the app for production to the `build` folder. 

It correctly bundles React in production mode and optimizes the build for the best performance.

See the section about [deployment](#) for more information.

```
PORT=4571 serve -s build & npm run test:e2e
```

'PORT=4571' is used to set the port where the production server will be listening for incoming connections. I decided to use a very specific number so that it doesn't interfere with any of your servers  (especially if you are testing multiple apps for some scholar reason?)

5. To customise your React configs in-depth and detach from react-scripts:

```
$ npm run eject
```

Note: this is a one-way operation. Once you `eject`, you can't go back!

If you aren't satisfied with the build tool and configuration choices, you can `eject` at any time. This command will remove the single build dependency from your project.

Instead, it will copy all the configuration files and the transitive dependencies (webpack, Babel, ESLint, etc) right into your project so you have full control over them. All of the commands except `eject` will still work, but they will point to the copied scripts so you can tweak them. At this point you're on your own.

You don't have to ever use `eject`. The curated feature set is suitable for small and middle deployments, and you shouldn't feel obligated to use this feature. However we understand that this tool wouldn't be useful if you couldn't customize it when you are ready for it.

Coding Practice

Throughout this project I ensured that the code is easily readable, whilst still maintaining the correct naming conventions; by using the camelCase practice throughout the entire code, and while still making sure that that all lines are correctly indented, ensuring that any other programmer could still read and understand the code. Furthermore, the code also has comments on every line of code to ensure that anyone can understand what is happening at every step of the code. I also made sure the code had the correct indentations throughout the entire code.

camelCase

```
// state
const [weight, setWeight] = useState("")
const [height, setHeight] = useState("")
const [bmi, setBmi] = useState('')
const [message, setMessage] = useState('')
```

Comments

```
let calcBmi = (event) => {  
  //prevent submitting, e=event  
  event.preventDefault()  
  
  if (weight === 0 || height === 0) {  
    alert('Please enter a valid weight and height')  
  } else {  
    let bmi = (weight / (height * height))  
    setBmi(bmi.toFixed(1))  
  
    // Logic for message  
  
    if (bmi < 25) {  
      setMessage('You are underweight')  
    } else if (bmi >= 25 && bmi < 30) {  
      setMessage('You are a healthy weight')  
    } else {  
      setMessage('You are overweight')  
    }  
  }  
}
```

📷 Regression Testing

I used regression testing to maintain code base consistency. Regression testing creates a snapshot of the DOM (HTML) for each of the components, if any of those components change it will throw an error if not the test will pass.

📖 React Testing library



Testing Library

The react testing library is used for testing the UI in units. This is because react testing library is a set of helper functions that allow us to test React components without relying on their implementation details.

Testing Logic

For testing the logic of the website I used react testing library's Jest. Jest (standalone) is utilised to test the logic of the application, this is because Jest is a javascript testing framework designed with simplicity in mind and Jest supports projects coded using Babel, Typescript, Node, **React**, Angular, Vue and more; Making Jest an ideal testing framework for my project. I produced testing blocks that lead to huge code coverage using Jest within a short period of time. Unit testing using Jest makes my project more agile as it allows me to be assured that the code I have will not break easily as I have already found and prevented bugs earlier in the development cycle.

End-to-End Testing

Puppeteer is a Node library which provides a high-level API to control Chrome or Chromium over the DevTools Protocol. I used Puppeteer to automate End-to-End testing. I have also integrated React E2E Testing with Puppeteer into a GitHub action, which allowed me to test my code with Puppeteer whenever I push or make a pull request.

Performance and Accessibility Testing

In order to improve the quality of the webpage, I used the Google Lighthouse tool to generate a report containing a summary of the quality of our application. Lighthouse is a free and open-source tool that can be accessed through the development console for Chromium-based browsers. There are 4 metrics which I have focussed on which are:

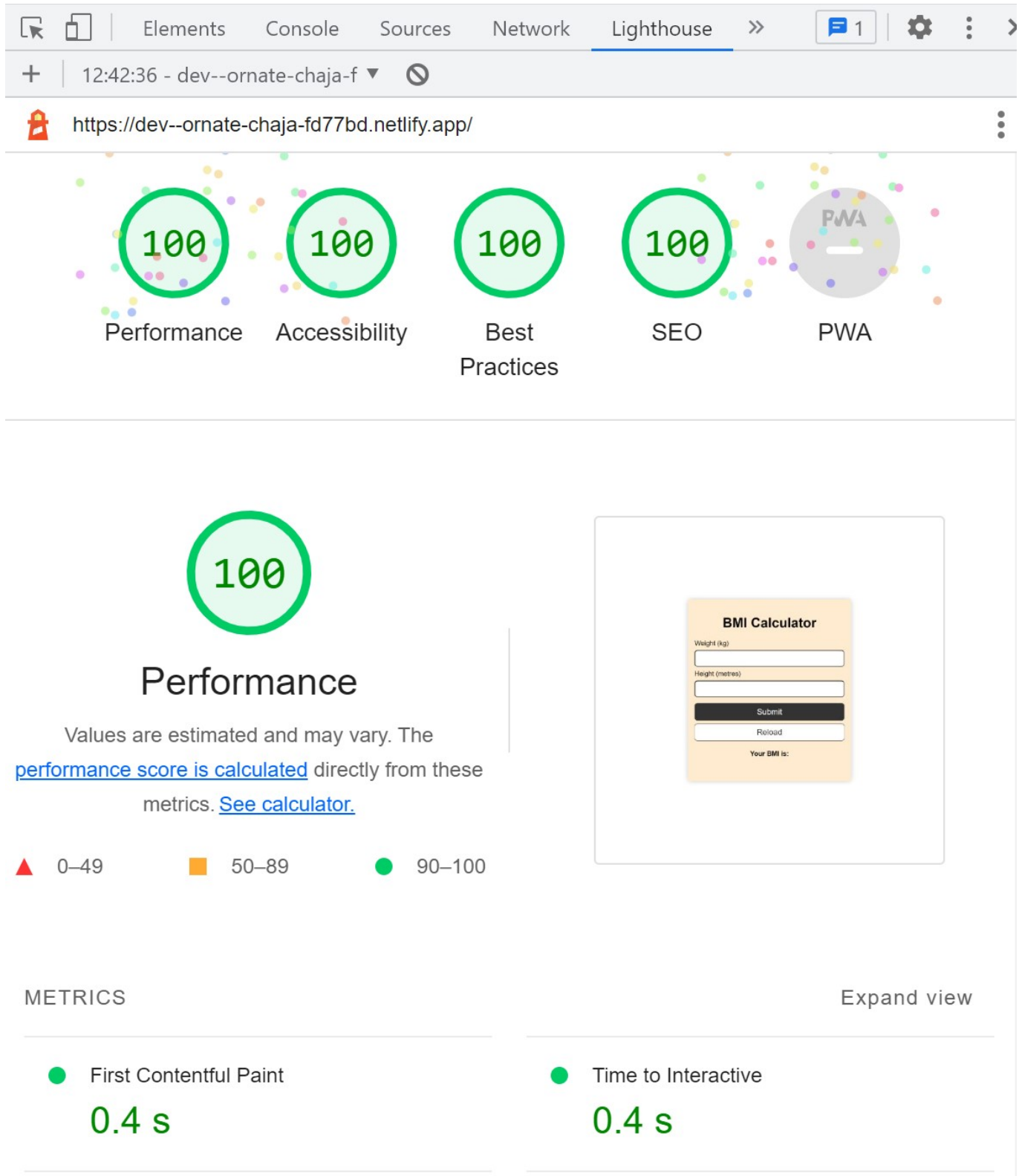
- Performance which is an aggregation of how the page progressed in aspects such as loading speed, time taken for loading basic frames and displaying meaningful content.
- Accessibility which is an aggregation of how accessible the website is through audio captions, button names and 'aria-' attributes.
- Best Practices which is an aggregation of many practices that have been deemed 'best' such as use of HTTPS and avoiding the use of deprecated code.
- SEO which is an aggregation of scores in features such as meta description, presence of titles, legible font sizes.

The websites performance rating of 100 suggests it is highly optimised and as a result, the website will offer a streamlined and smoother experience giving the user a better experience as a whole when using our website, it also decreases the bounce rate of users who attempt to access the website incase it takes too long to load. Poor performance hurts the overall user experience and offers less incentive to stay.

The accessibility rating of 100 improves our search engine optimisation results and makes it easier for each user to use the website regardless of age or disability. The website's design also includes contrasting colours. Users would be able to interact with the website/application properly without any problems while accessing it.

The best practices rating of 100 suggests that the websites design is simple and easy to navigate.

The search engine optimisation (SEO) rating of 100 suggests the websites design attracts higher quality internet traffic and traffic will be directed to the website naturally without having to advertise it.



Manual Testing

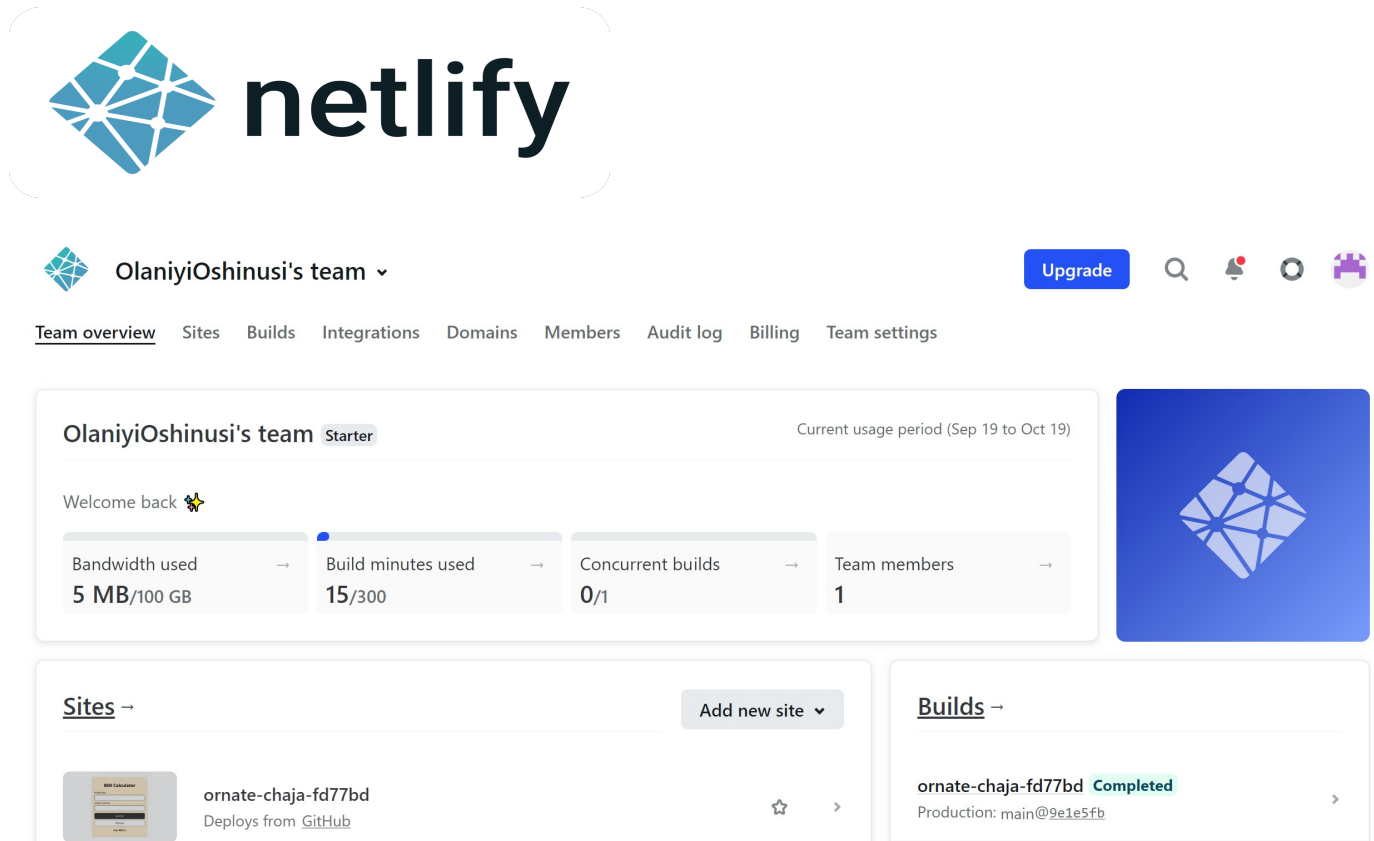
Manual testing was completed on the application. The purpose of Manual Testing is to identify any bugs, issues, or defects in the website. Manual software testing helps to find critical bugs in the website design such as features not working as intended or not working at all. This allows to debug the website and improve features which did not have necessary code implemented to fix specific bugs which may occur when the user is using my BMI calculator to calculate body mass index. Although Manual website Testing requires more effort, I realised it was imperative to check automation feasibility; Additionally, Manual Testing concepts do not require knowledge of any testing tool.

[BMI Manual Test Script.xlsx](#)

Resolution of an Issue

This is an example of an issue which was discovered while manually testing the design.

Continuous Integration / Continuous Deployment



Netlify is used to continuously deploy my code or any changes, every time I push to the main branch, the website goes live with a new version. When a pull request is made in my workflow, netlify also checks if there are any warnings, and makes a preview for that specific pull request. This is beneficial because Netlify is incredibly quick and easy to use for getting started with my website, saving time and increasing productivity as a result. Netlify will also save more time as you can lock onto a main branch for main deployments. It can also deploy other branches and give you a URL to test those deploys, improving the amount of time it takes for you test your code before merging it to your main; Also giving you an easier option to manually test other branches.

GitHub Actions

GitHub Actions is used to automate the testing pipeline every time code is pushed or a request is made. This allowed me to create workflows that build and test every pull request to the repository, or deploy my merged pull requests to production. GitHub Actions also helped improve my workflow by automatically adding the appropriate labels.

```
> workflows > ! nodejs.yml
# This workflow will do a clean installation of node dependencies, cache/restore them, build the source code and
# For more information see: https://help.github.com/actions/language-and-framework-guides/using-nodejs-with-github-actions

name: Node.js CI

on:
  push:
    branches: [ "main", "dev" ]
  pull_request:
    branches: [ "main", "dev" ]

jobs:
  build:
    runs-on: ubuntu-latest

    strategy:
      matrix:
        node-version: [14.x, 16.x, 18.x]
        # See supported Node.js release schedule at https://nodejs.org/en/about/releases/

    steps:
      - uses: actions/checkout@v3
      - name: Use Node.js ${{ matrix.node-version }}
        uses: actions/setup-node@v3
        with:
          node-version: ${{ matrix.node-version }}
          cache: 'npm'
      - run: npm ci
      - run: npm run build --if-present
      - run: npm test
```

For additional examples of GitHub Actions click [here](#)

Learn More

You can learn more in the [Create React App documentation](#).

To learn React, check out the [React documentation](#).

Code Splitting

This section has moved here: <https://facebook.github.io/create-react-app/docs/code-splitting>

Analyzing the Bundle Size

This section has moved here: <https://facebook.github.io/create-react-app/docs/analyzing-the-bundle-size>

Making a Progressive Web App

This section has moved here: <https://facebook.github.io/create-react-app/docs/making-a-progressive-web-app>

Advanced Configuration

This section has moved here: <https://facebook.github.io/create-react-app/docs/advanced-configuration>

Deployment

This section has moved here: <https://facebook.github.io/create-react-app/docs/deployment>

`npm run build` fails to minify

This section has moved here: <https://facebook.github.io/create-react-app/docs/troubleshooting#npm-run-build-fails-to-minify>