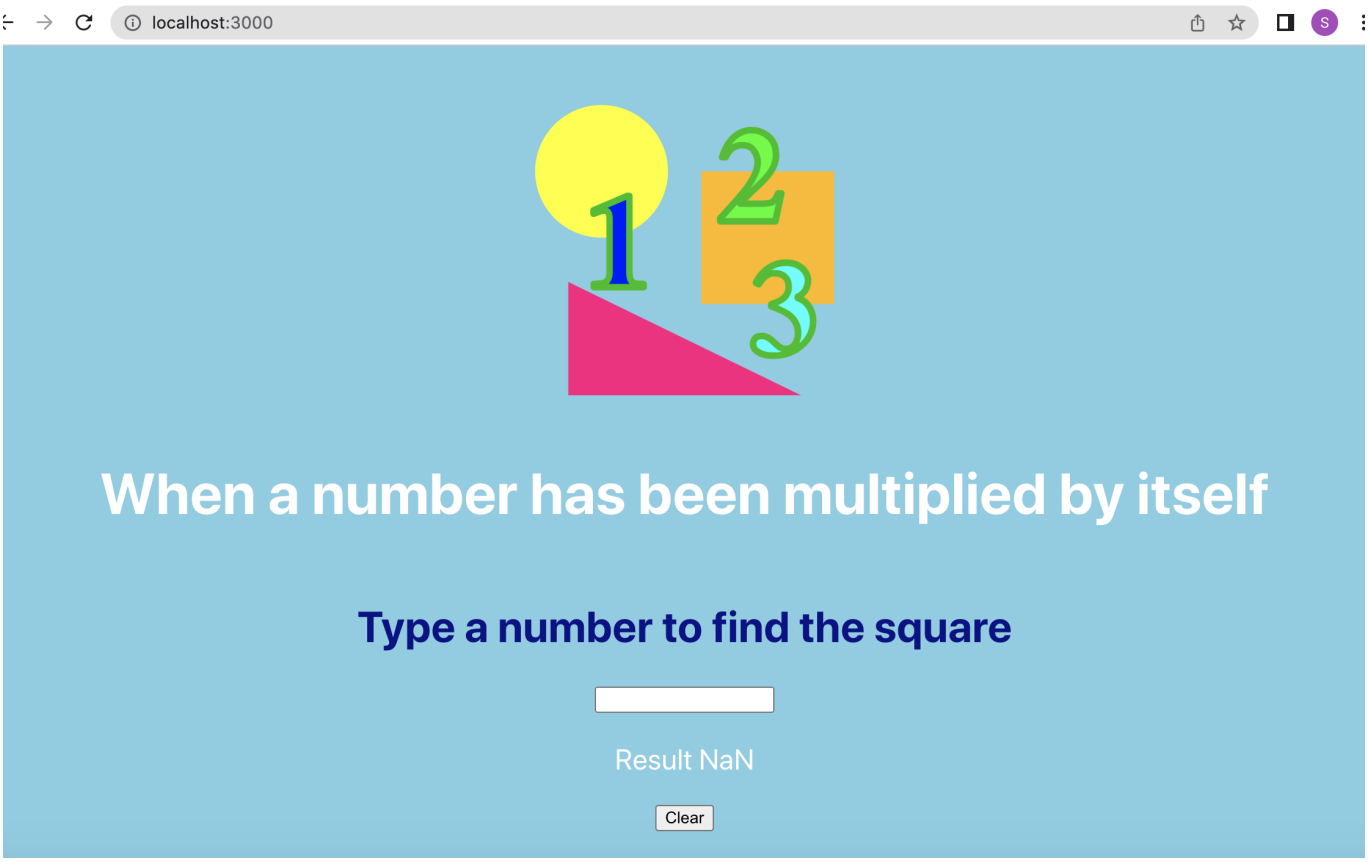
Summative Assignment One, System Testing

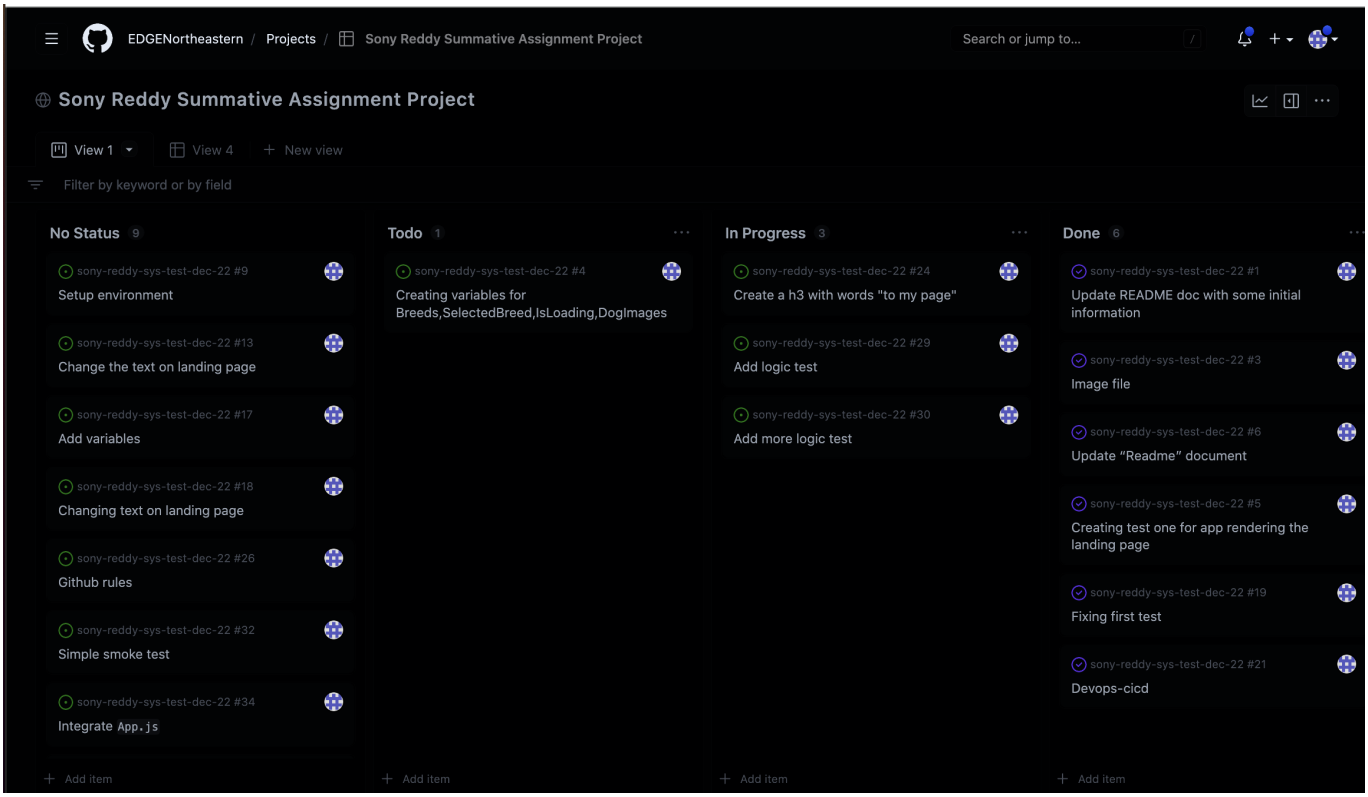
For Northeastern University



Project on Squared

- When a number is multiplied by itself, the result is a square number. For instance, 8 is a square number since it is equal to four lots of four, or four x four.
- This project is about knowing square of the number you enter.As soon as you type a number it will show the result by multiplying itself.There is also a clear button to clear the text entered.

📂 Project Management



- Git project tracker was employed. Our work is better organised and prioritised thanks to project boards on GitHub AE. For the development of particular features, we can make project boards, thorough roadmaps, or even release checklists. We have the freedom to design unique workflows with project boards that meet our demands.

Environments

- Prod
- UAT

Production -The final destination for all completed and approved work is production servers. When code is put on a production server, it's been given the all-clear to go live. Never work on a production server without using some sort of version control, as there is a great chance that something will go wrong while the product is being used. When a product or production server goes offline, it may sometimes cost a firm a lot of money, which is absolutely not something

The ultimate goal is always to launch a product into production with the fewest number of flaws feasible to guarantee a completely functional interface and satisfactory customer experience.

UAT - The developers or programmers will deploy their finished product to a test server after they have concluded their work. The team will set up and configure a test server for internal use with the relevant configurations. This makes it possible for the group to access the work for validation. A Tester is typically used by the internal team to finish the testing step. To verify that the product is operating as it should, the tester will run a variety of use cases. The developer or programmer will be given tasks to complete if the tester finds bugs or other problems.

Here production enviroment is the intial enviroment where the project started and UAT is the enviroment the application developed for this project.

Git Rules

- Create a new Git repository for every new project
- Create a **new ticket** for every **new feature** to be tested
- Add acceptance criteria to feature ticket
- Create a new branch for every new feature
- One feature - one ticket - one branch - one Pull Request - many commits
- A branch name should be of small alpha characters and dashes (-)
- Use Pull requests to merge code to uat and then uat to main in a release

Prerequisities

Three key tools are used in this project: Create React App, Jest, and React Testing Library. A single-page React application is bootstrapped using Create React App. In order to organise tests around user interactions, Jest is utilised as the test runner while the React Testing Library offers test tools.

Make sure npm is installed because this project needs it to run the files.

1. Run the following command in a terminal to verify that node and npm are installed.

```
node -v
```

```
npm -v
```

2. Run the following command in your terminal to clone the project from Github:

```
git clone https://github.com/EDGENortheastern/sony-reddy-sys-test-dec-22.git
```

Navigate to the folder you just cloned to find the code.

3. Install the project dependencies.

```
npm install
```

The npm install command will install all the project dependencies defined in the package.json file.

4. After installing the dependencies, you can either view the deployed version of the app or you can run the app locally with the following command:

```
npm start
```

Runs the app in the development mode.
Open [http://localhost:3000] to view it in your browser.

5. You can view the deployed version of the app after the dependencies have been installed, or you can use the following command to execute the app locally:

```
npm test
```

```
a
```

Launches the test runner in the interactive watch mode.
See the section about running tests for more information.

7.view full tests and code coverage:

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

8.React configs in-depth and detach from react-scripts:

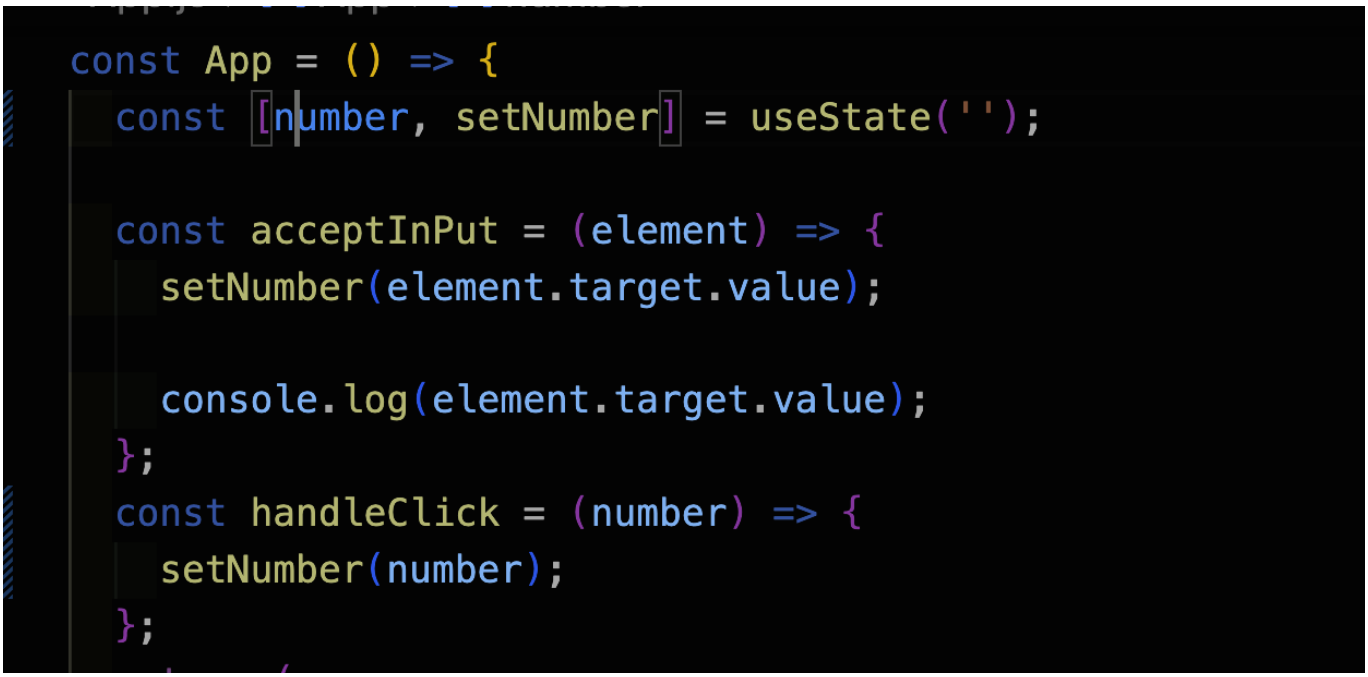
```
npm run eject
```

**Note: This is a one-way procedure, please note. You are unable to return after you eject! You can always eject if the build tool and configuration choices don't meet your needs. With this command, you can decouple your project from the single build dependency. Instead, all the configuration files and transitive dependencies (such as webpack, Babel, ESLint, etc.) will be copied directly into your project, giving you complete control over them. Every command, with the exception of eject, will continue to function, but they will now point to the copied scripts so you can edit them. You're on your own at this point. You're never required to use eject. You shouldn't feel pressured to adopt the curated feature set because it is appropriate for small and middle-sized deployments.

Coding Pratice

Naming conventions

We frequently use JSX (JavaScript Extension) files while working with React. Therefore, camel case should be used for every component that we develop for React. This translates to names without spaces and every other word's first letter being capitalised. Additionally, the team ensured that the code was properly indented throughout.



```
const App = () => {
  const [number, setNumber] = useState('');

  const acceptInPut = (element) => {
    setNumber(element.target.value);

    console.log(element.target.value);
  };
  const handleClick = (number) => {
    setNumber(number);
  };
```

Comments

Only add comments where they are required in the code. In addition to adhering to React best practises, this accomplishes two goals at once:

It will keep the code clean and uncluttered. In the event that you decide to change the code in the future, you'll prevent any potential conflicts between the comment and the code.

Code should execute as expected and be testable

We should write code that works as expected and is quick and simple to test.test suffixed test files that are exact copies of the source files. Finding the test files will thereafter be simple.

Regression Testing

Regression testing was utilised by the team to keep the code base consistent. When performing regression testing, a UI component is rendered, a snapshot is taken, and the result is compared to a reference snapshot file kept with the test. If the two snapshots differ, the test will fail because either the change is unexpected or the reference snapshot has to be updated to reflect the new UI component.

React Testing Library

Testing Logic

For our project, Jest is the best testing framework because it supports projects written in Babel, Typescript, Node, React, Angular, Vue, and other languages. Jest is a javascript testing framework that was created with simplicity in mind. The group used Jest to collaborate on creating testing blocks that quickly achieved 90–100% code coverage. For this project, utilising Jest is advantageous because it enables us to make our project more agile because we are confident that the code we have won't break quickly.

```
describe("testing function convertToNum",()=>{
    test("that function returns a number",()=>{
        expect(convertToNum("2")).toBe(2);
        expect(convertToNum("3")).toBe(3);
        expect(convertToNum("35")).toBe(35);
    })
    test("that function works for negitive numbers",()=>{
        expect(convertToNum("-127")).toBe(-127);
        expect(convertToNum("-3")).toBe(-3);
        expect(convertToNum("-90")).toBe(-90);
    })
    test("that function works for decimals",()=>{
        expect(convertToNum("12.5")).toBe(12);
        expect(convertToNum("67.5")).toBe(67);
        expect(convertToNum("54.8")).toBe(54);
    })
    test("operation typeof works",()=>{
        expect(typeof convertToNum("50")).toBe('number');
        expect(typeof ("true")).toBe('string');
```

Performance and Accessibility Testing

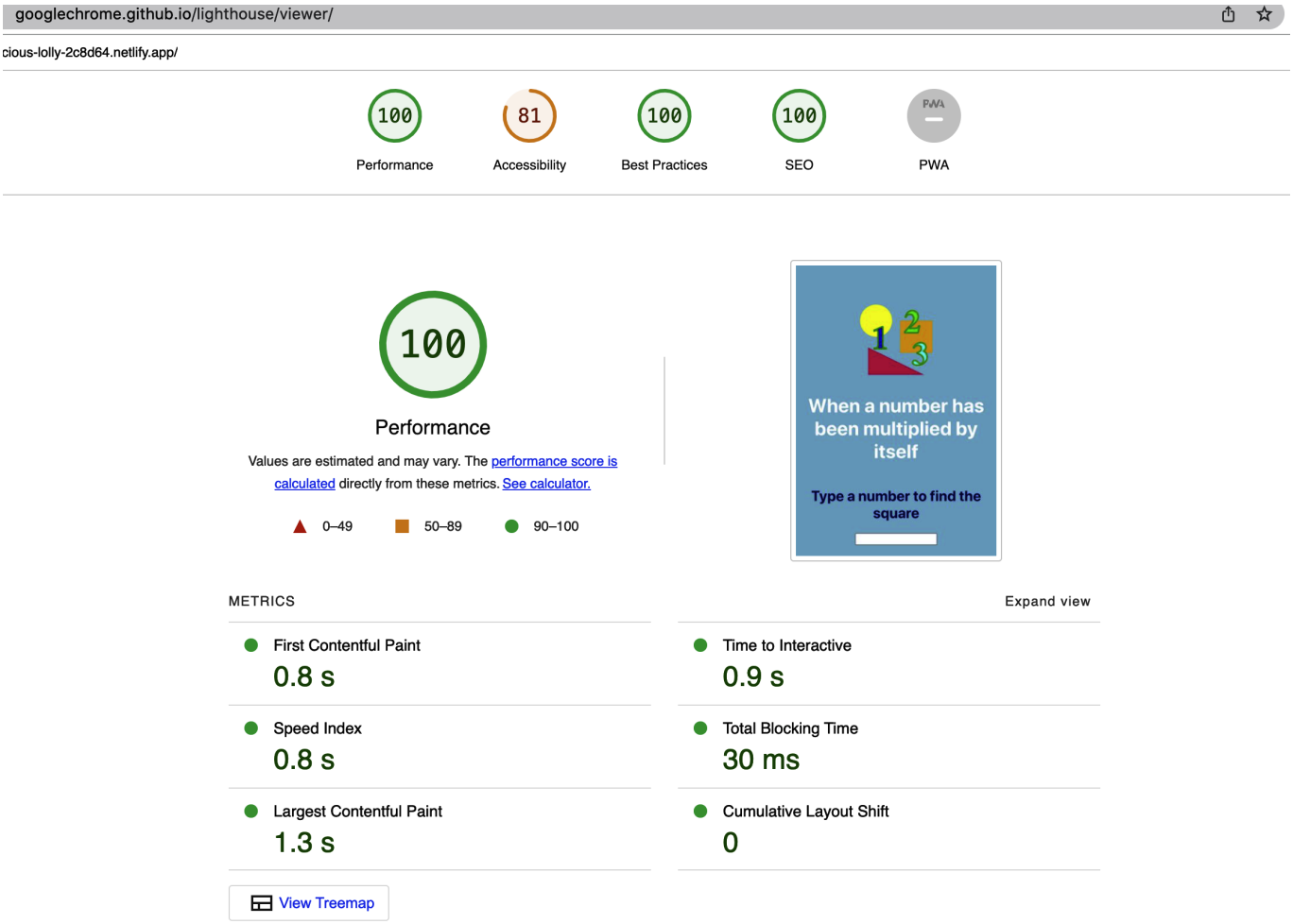
In order to evaluate the effectiveness of our website, we used Google Lighthouse. An open source tool called Google Lighthouse can be used to audit and enhance the functionality and quality of your website. Due to its adaptability and capacity for measuring a variety of metrics, including online performance, accessibility, best practises, SEO, and progressive web apps, it is a preferred option.

Performance rating - An extremely optimised website will offer a streamlined and seamless experience, enhancing the user's overall experience while using our website, according to the site performance score of 100. Additionally, it reduces the number of users who abandon a website after trying to access it because it is taking too long to load. Poor performance hurts the user experience as a whole and makes it less appealing to stay.

Accessibility rating - The platform is easier to use for all users, regardless of age, thanks to the website's accessibility grade of 81, which also improves our SEO performance.

Best practices rating - The best practises grade of 100 indicates that the website's layout is uncomplicated and easy to use.

Search engine optimisation (SEO) - The website has a search engine optimization (SEO) score of 100, which means that its layout attracts higher-quality internet traffic that will arrive on its own.



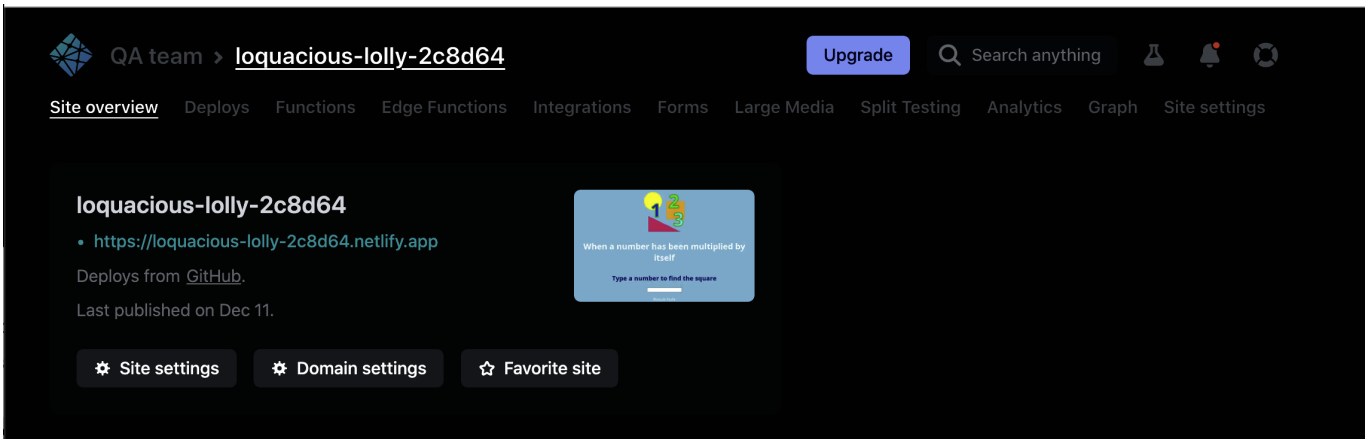
Manual testing:

Google Docs were used to complete and record the manual testing. Since manual testing is a continuous process that necessitates human verification at regular intervals throughout the software development lifecycle, it is impossible to completely avoid it. Teams must therefore strike the ideal mix between manual and automated testing.

Manual testing will always be necessary, despite the agile approach to software development always advocating a shift toward test automation. In reality, manual testing is a useful method for evaluating user-relevant aesthetic features, such as how smoothly web items render, how simple it is to navigate through an application, etc.

Find test cases here

Continuous Integration / Continuous Deployment



The live app has been deployed to Netlify, one of the best sites to automate React projects and to create CD pipelines for React projects. Every time some code is merged into either main or uat branches, the Netlify sites update. When changes are merged to uat the uat environment updates here: https://uat--loquacious-lolly-2c8d64.netlify.app/ Every time code is merged into main, the production site updates: https://loquacious-lolly-2c8d64.netlify.app/

Moreover, Netlify runs some checks with every Pull Request. As a result, every PR is checked by Netlify for things such as broken links and also by CI (GitHub Actions) that runs all tests suites. Netlify lets developers preview the checks and produces meaningful error messages if something goes wrong.

When code is merged into main, an event known as a release happens. So far, the app had one release. The Pull Request for the release was called release 1.0 and it can be seen here:

https://github.com/EDGENortheastern/sony-reddy-sys-test-dec-22/pull/22

The next release is yet due to happen. The release happens after all manual tests are completed and all features that were due to be released in a sprint are built.

GitHub Actions

```yaml

name: Node.js CI

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

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
```

GitHub Actions are used to automate the test cycle each time code is pushed or a request is submitted. Now that our merged pull requests have been put to production, the team may create processes to build each pull request before it is added to the repository. Additionally, by automatically adding the required labels, the team employs GitHub Actions to improve team efficiency.

For additional examples of GitHub Actions click here

The boilerplate explained

The application was created with a boilerplate - Create React App.

The boilerplate comes with React, React Testing Library and many other useful packages pre- installed.