Bachelor’s thesis

Information and Communications Technology

2022

Huong Nguyen

SINGLE-PAGE APPLICATION AND FRONT-END TESTING METHODS

– Built with React and React Router, Tested with Jest and Cypress

TURKU AMK
TURKU UNIVERSITY OF APPLIED SCIENCES

BACHELOR’S THESIS | ABSTRACT

TURKU UNIVERSITY OF APPLIED SCIENCES

Information and Communications Technology

2022 | 69 pages

Huong Nguyen

Nowadays, web technology brings numerous advantages to its users. Everyone searches for information stay up to date on current news and keep in touch with others through various websites and applications.

Web technology is divided into 2 segments: Front-end development and Back-end development. In order to have a specific view about Front-end development, one of the best Front-end technologies - React was chosen as the focus of this thesis.

The main purpose of this thesis was to study React and its relevant technologies in order to create a single-page application, as well as to carry out front-end testing using Jest and Cypress. A Portfolio was built with React to examine how React actually works in reality, and its front-end was tested with selected testing methods to ensure the application works as expected.

The result of the study was a functional Portfolio single-page application that describes the information of the thesis owner about qualifications, projects/works, and a contact form. The Portfolio runs the client-side with features that change routes without loading the entire page, animation on the background, and a contact form that can send a visitor’s message directly to the owner’s email. Furthermore, after being tested with Jest and Cypress, the application is ensured to run perfectly.

Keywords:

Front-end development, Web development, React, Front-end testing.

Contents

Tables

Figures

LIST OF ABBREVIATIONS

CSS	Cascading Style Sheets
CV	Curriculum Vitae
DOM	Document Object Model
GUI	Graphical User Interface
HR	Human resources
HTML	Hypertext Markup Language
JS	JavaScript
JSX	JavaScript XML
MVC	Model View Controller
SASS	Syntactically Awesome Style Sheets
SCSS	Sassy CSS
UI	User Interface
VSC	Visual Studio Code
XML	Extensible Markup Language

1 INTRODUCTION

In this day and age, daily tasks and activities can be easily done via the Internet by interacting on websites or applications, for example, people can search for information, read online newspapers, shop online, or watch movies through any applications or websites. Therefore, humans invest in the Web development industry to become dynamic, grow continually, and innovate all the time.

Web development is divided into two main sections. The first one is Front-end development, also known as client-side development. It is a practical combination of HTML, CSS, and JavaScript. Users can see and interact with front-end programming through User Interface (UI) (Lindley, 2018). The second section is Back-end development, which is known as server-side development. It involves many behind-the-scenes activities and features that humans do not see and have interaction with when they access the website.

One of the most popular Front-end libraries nowadays is React. It was built with a JavaScript library, conserved by Facebook and a group of developers and organizations (reactjs, Reactjs, 2021 b). React can be utilized as a foundation in the process of developing a single-page application or mobile application.

In Front-end development, UI testing is also a critical part to ensure the UI works as expected. The target of Front-end testing is to investigate functionalities and verify that the presentation layer of a website or application is error-free. There are several types of UI testing, including unit testing, integration testing, performance, and accessibility testing.

This thesis aims to study React and its relevant technologies in order to build a single-page application. Furthermore, it indicates the advantages and disadvantages of React, how React has become popular and is different from other Front-end frameworks. In addition, the thesis also investigates a variety of Front-end testing methods to warrant the application runs smoothly. In the thesis, a Portfolio was built as a single-page application with React to examine how React works in reality. In this application, a Front-end testing part was established with Jest and Cypress to test the application functionalities and components.

The thesis comprises six chapters. Chapter 1 briefly introduces Front-end development, React, Front-end testing, and the objectives of this thesis. Chapter 2 discusses the general knowledge of the theoretical background about React and relevant technologies to build a single-page application. Chapter 3 discusses how React grows in the market by analyzing its positive features and adverse characteristics, and compares React to a similar framework, which is Angular. Chapter 4 discusses concepts and requirements of Front-end testing and introduces some types of testing tools, which are cross-browser testing, js testing, and functional testing. Chapter 5 discusses the value of a Portfolio in job hunting and demonstrates the flow of developing a Portfolio with React. Chapter 6 presents the conclusion of the thesis.

2 THEORETICAL BACKGROUND TO BUILDING AND TESTING A SINGLE-PAGE APPLICATION

What is a Single-page application (SPA)? In general terms, SPA is a web application that avoids refreshing the page every time users interact with a link (Nassi, 2021), for example Trello or Facebook. It uses HTML5 and AJAX to build responsive apps. In order to handle the heavy lifting on the client-side of a SPA, some JavaScript frameworks including React, Angular, Vue were developed and launched, and they are managing their function really well. The main advantages of a SPA are that it is fast and responsive since it only updates the required content, not the entire page, therefore, the application's loading speed is rapid (Z, 2018).

Among the above-mentioned JavaScript frameworks for SPA, the writer decided chose to React for developing her SPA (Portfolio) due to its fast performance and scalability, furthermore, it is also a well-known library in market trend.

This chapter is about the general knowledge of React and technologies involved in building a SPA, such as React Router and SASS. Two testing frameworks that are used to test the application in this thesis, Jest, and Cypress, are also introduced in this chapter.

2.1 The purpose of the thesis

After graduation, every student faces many problems in job hunting, they have to prepare themselves a good Curriculum Vitae (CV) demonstrate good professional and personal skills, and how to perform as well as they can in the jon interview. However, before they can reach to the interview round, they need to impress Human Resources (HR) through their CV and Portfolio. For IT students, especially students who majored in Web development, Portfolio is very important, it presents the students' work and which level of coding skills they have. Therefore, most of students invest their time and try their best to compile a good Portfolio with an attractive UI design.

The writer of this thesis decided to do the Portfolio as her thesis works since she realized the importance of the Portfolio for her career path later. She will do the Portfolio as a SPA by React because she would like to demonstrate her programming skills in React through the Portfolio. Furthermore, she will also make it different from the original web application, which is defining different APIs for routes by using React Router.

The Portfolio is designed in different UI layouts according to different purposes (presented by APIs), such as about (introduction of applicant), projects (applicant’s work), contact (how to contact the applicant). If using the traditional web programming method, it is pretty hard to organize these layouts in accordance with different APIs in a beautiful overall look, therefore she decided to build it as a SPA by using React and React Router.

2.2 React

React is a free and open-source JavaScript library that has attracted most web developers’ attention over the world from the date it was deployed by Facebook. It is firstly announced on Facebook’s newsfeed in 2011, appeared on Instagram in 2012 (Pandit, 2021), and was officially released on May 29, 2013, (Wiki, 2021).

For the original web application, the Document Object Model (DOM) is used to represent data (structure and content of a web document). With DOM, the web document is organized as nodes and objects, programmers can interact with the structure, content, and style of the page via programming languages (Docs, 2022). An example of DOM is illustrated in Figure 1.

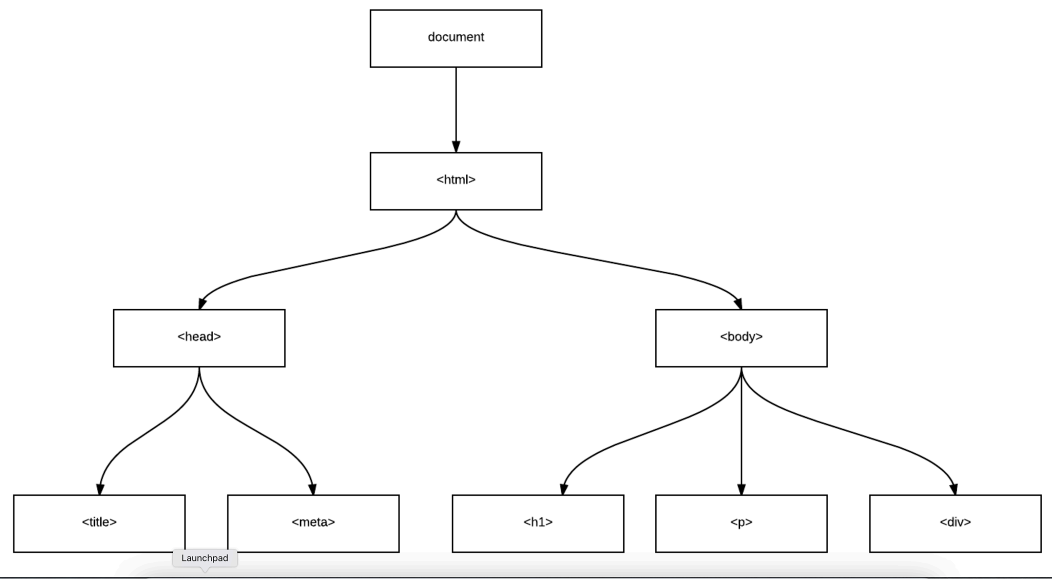


Figure 1. Document Object Model – DOM Example (Suthwal, 2020).

The main reason for making React well-known is that it uses Virtual DOM for better, faster performance, compared to DOM of traditional web development. In React, a Virtual DOM represents a DOM object and performs as the lightweight copy (Codecademy, 2021).

To clarify the point of saying Virtual DOM is high-speed and effective, let's have a deep look into how the Virtual Dom actually works. A Virtual DOM – formed as a tree – is generated when the UI has new elements added, every element represent a node on this tree. A new Virtual DOM is produced afterward if the state of those elements has any changes. Subsequently, there is a comparison or diffing process between this Virtual DOM tree and the previous. When this process is completed, the real DOM is updated by the best possible method calculated by the Virtual DOM. This manages to have minimal operations on the actual DOM (Hamedani, 2018).

Figure 2 is a demonstration of how the actual DOM is updated after the diffing process from Virtual DOM.

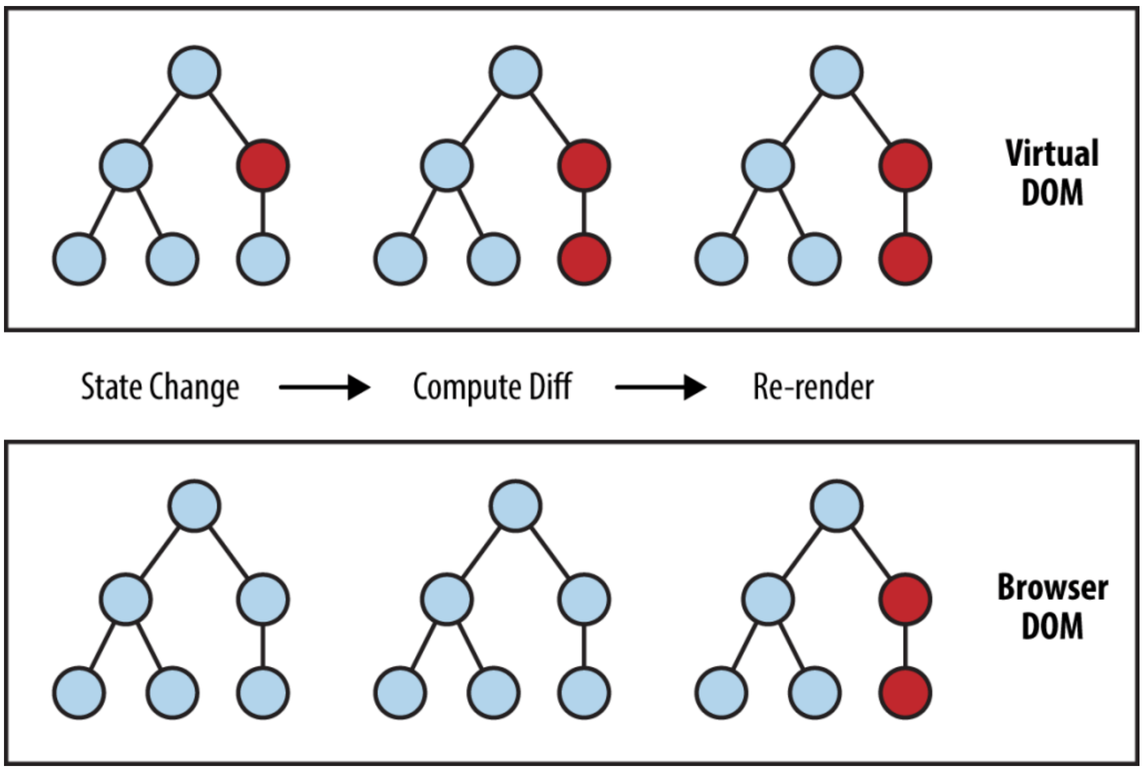
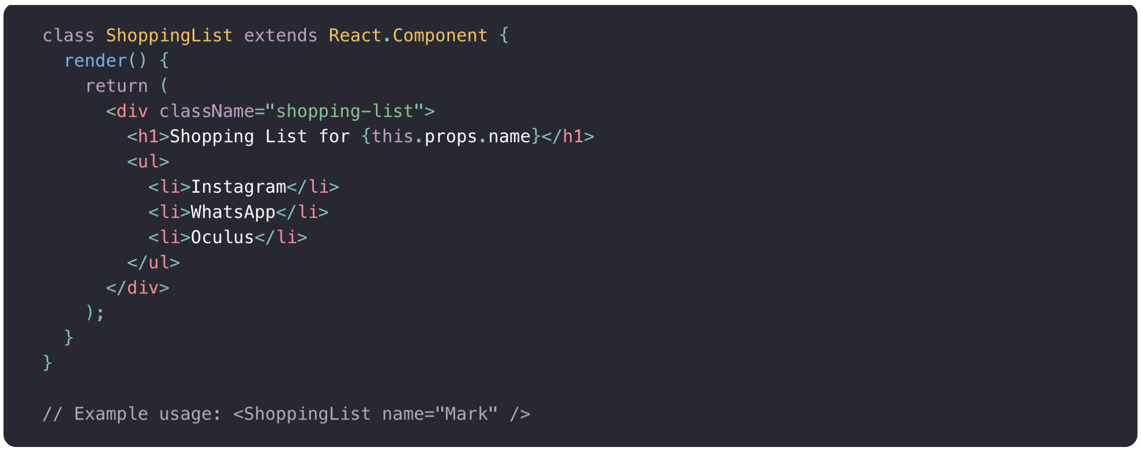


Figure 2. Virtual Dom tree vs Real DOM tree and the diffing process (Hamedani, 2018).

Based on the above analysis, using React is more beneficial to modify the content and style of the Portfolio in the most efficient and fastest way.

Besides the Virtual DOM, another highlight feature of React is that it lets developers build complex UI from independent pieces of code called components. In this way, the programmer can manage the code in an organized structure.

Below is a React component sample (Figure 3).



```
class ShoppingList extends React.Component {
  render() {
    return (
      <div className="shopping-list">
        <h1>Shopping List for {this.props.name}</h1>
        <ul>
          <li>Instagram</li>
          <li>WhatsApp</li>
          <li>Oculus</li>
        </ul>
      </div>
    );
  }
}

// Example usage: <ShoppingList name="Mark" />
```

Figure 3. React component (reactjs, Reactjs, 2021 b).

There are two main types of React components (Lin, 2019):

- Functional Component (Figure 4): it takes in props, returns JSX. State or lifecycle methods are not included in this type of component. Developers have to implement React Hooks to utilize state and some relevant functions.

```
// Functional Component Example

import React from 'react';

const HelloWorld = () => {
   return (
      <div>
         <p>Hello World!</p>
      </div>
   )
}

export default HelloWorld;
```

Figure 4. Functional Component example (Lin, 2019).

- Class Component (Figure 5): prime functions of React as state, lifecycle methods or props can be managed in this component. Programmer can manipulate state or props in some lifecycle methods which are componentDidMount(), or componentDidUpdate(), and componentWillUnmount(), etc.

```
// Class Component Example

import React from 'react';

class HelloWorld extends React.Component {
   render() {
      return (
         <div>
            <p>Hello World!</p>
         </div>
      )
   }
}

export default HelloWorld;
```

Figure 5. Class Component example (Lin, 2019).

Taking the advantage of the component feature of React, the author can divide the Portfolio into separate components by functions of each page, such as component Navigation Bar, Homepage, Projects, and Contact.

2.3 React-Router

When mentioning routing in a SPA, it is not impossible to notice React Router. It is a standard library to manipulate multiple routes in a React application. It allows navigation between views of numerous components, switch between routes in browser URL, and manage UI synchronous with URL (Shruti, 2021). Nowadays, React Router is used to render multiple views on most social media websites including Facebook, Instagram. (javatpoint, React Router, 2021 c).

There are three main routing packages to be considered to install in a React application (javatpoint, React Router, 2021 c):

- react-router: implement the core routing functionality for React Router.

- react-router-native: is utilized for a React Native application – mobile version.
- react-router-dom: is applied for web applications.

However, developers do not need to install *react-router* directly in the application, since when they install *react-router-native or react-router-dom*, the package *react-router* is a dependency already included in those packages.

In React Router, there is 4 main components most used: <Router>, <Route>, <Link>, <Switch>. In fact, React Router has two main <Router> components including <HashRouter> and <BrowserRouter>.

Browser Router is used to handle dynamic URLs and send information about the routes to the server. Furthermore, it has a parent/ child relationship with other components since it stores other components entirely. When the application is deployed on the server, the link should be like "https://www.hosting.com/". (reactrouter, BrowserRouter, 2021 a)

HashRouter uses the hash portion of the URL and handles the static request. A good advantage of this component is the information of routes is not sent to the server. When the application is hosted on the Github page, the URL will be as "https://www.hosting.com/#". (reactrouter, HashRouter, 2021 b)

Based on the characteristic of each Router, Brower Router is chosen to manage routes in the Portfolio of this thesis, since the author wants to make the URL look nice and well-ordered without the hash.

Beside <Router>, <Route> is also a essential component in React Router. It is a children component of <Router> and responsible for rendering UI when the current URL is matching with its path (reactrouter, Route, 2021 c). Therefore, routes in the application can be set up as the Figure 6 below.

```
import React from "react";
import ReactDOM from "react-dom";
import { BrowserRouter as Router, Route } from "react-router-dom";

ReactDOM.render(
  <Router>
    <div>
      <Route exact path="/">
        <Home />
      </Route>
      <Route path="/news">
        <NewsFeed />
      </Route>
    </div>
  </Router>,
  node
);
```

Figure 6. BrowserRouter and Route component (reactrouter, BrowserRouter, 2021 a).

In Figure 6, when the current URL is matching with path “/”, the Home component is rendered and if the path “/news” is matched to the URL, it renders <NewsFeed>.

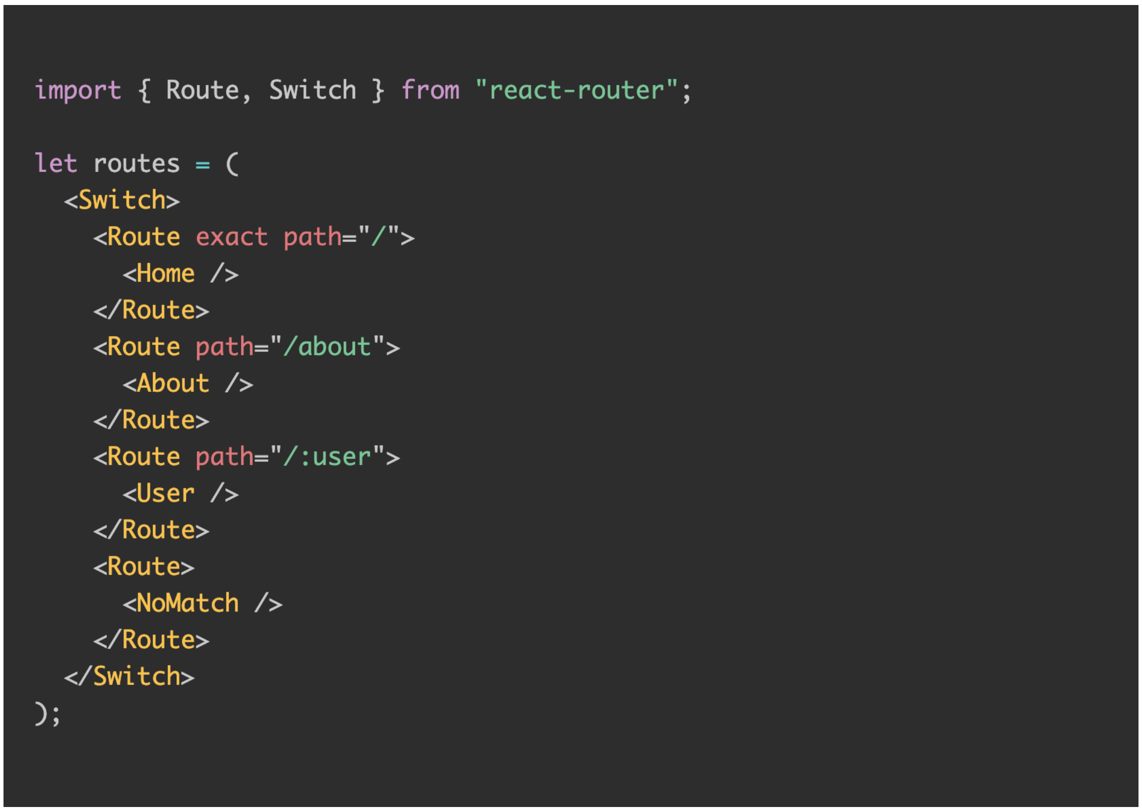
In order to navigate distinct routes and generate navigation between routes, <Link> is used, it works like a <a> tag in HTML. See the Figure 7 below.

```
<Link to="/about">About</Link>
```

Figure 7. Link component.

In Figure 7, it creates a link to route “/about”. When the user clicks on “About”, the URL is navigated to “/about”.

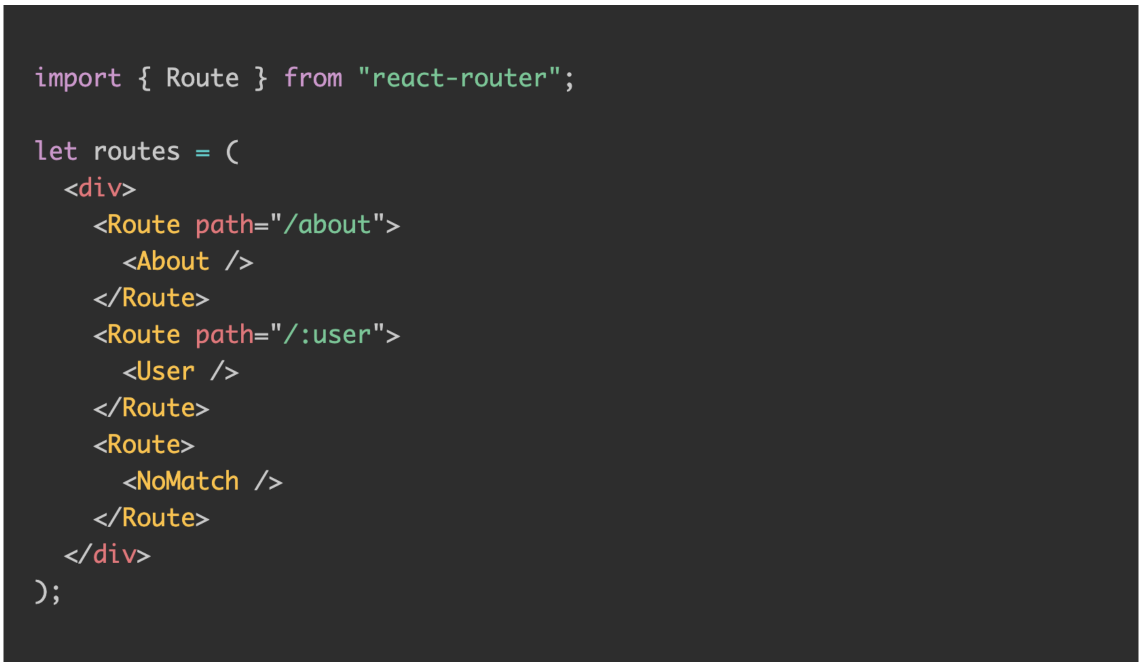
Another important component in React Router is <Switch>, it manages to render only the first matching route.



```
import { Route, Switch } from "react-router";

let routes = (
  <Switch>
    <Route exact path="/">
      <Home />
    </Route>
    <Route path="/about">
      <About />
    </Route>
    <Route path="/:user">
      <User />
    </Route>
    <Route>
      <NoMatch />
    </Route>
  </Switch>
);
```

Figure 8. Switch component (reactrouter, Switch, 2021 d).

In Figure 8, Switch will look for the route matching to URL, if the route “/about” is matched, it will stop the process of searching and render only the About component. In another case, if Switch is not supposed to be used as the figure 9 below, it will not stop the process of looking for matching routes till the end, as a consequence, it will render all the components About, User, and NoMatch, since all the routes starting with “/” and it will get a match to the URL.



```
import { Route } from "react-router";

let routes = (
  <div>
    <Route path="/about">
      <About />
    </Route>
    <Route path="/:user">
      <User />
    </Route>
    <Route>
      <NoMatch />
    </Route>
  </div>
);
```

Figure 9. Route components without Switch (reactrouter, Route, 2021 c).

In the Portfolio of this thesis, Browser Router, Route, Link, and Switch are the main components used to manage routes and navigation to each page of the application.

2.4 SASS

SASS stands for Syntactically Awesome Style Sheets, which is a stylesheet language compiled to CSS. It allows applying variables, nested rules, mixins, and functions, etc with a fully CSS-compatible syntax (sass-lang, documentation, 2021 a).

SASS comprises two type of syntaxes. The primary syntax is named “the indented syntax” with the extension .sass. In this syntax, indentation is used to divide blocks of code, and a newline is necessary between rules. Furthermore, the newest syntax is SCSS – Sassy, it is having an extension .scss, using semicolons and curly braces to format the document. Below are the examples of these syntaxes (Figure 10 and Figure 11) (sass-lang, syntax, 2021 b).

```
@mixin button-base() {
  @include typography(button);
  @include ripple-surface;
  @include ripple-radius-bounded;

  display: inline-flex;
  position: relative;
  height: $button-height;
  border: none;
  vertical-align: middle;

  &:hover { cursor: pointer; }

  &:disabled {
    color: $mdc-button-disabled-ink-color;
    cursor: default;
    pointer-events: none;
  }
}
```

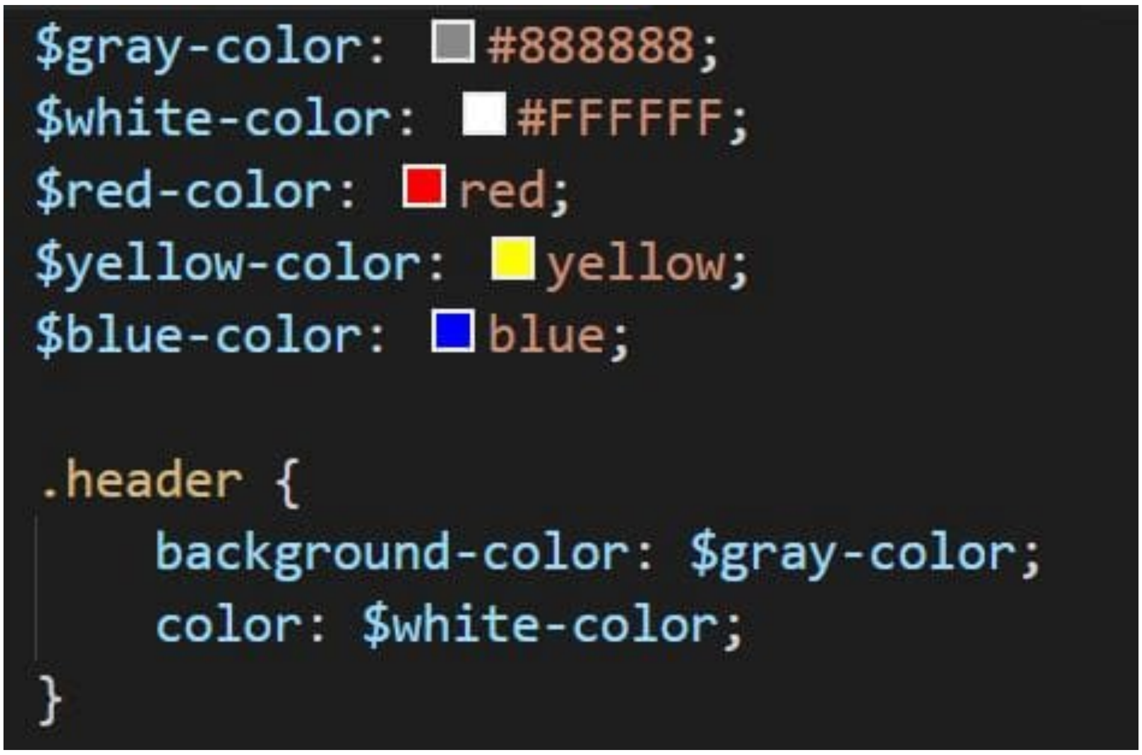
Figure 10. SCSS syntax (sass-lang, syntax, 2021 b).

```
@mixin button-base()
  @include typography(button)
  @include ripple-surface
  @include ripple-radius-bounded

  display: inline-flex
  position: relative
  height: $button-height
  border: none
  vertical-align: middle

  &:hover
    cursor: pointer

  &:disabled
    color: $mdc-button-disabled-ink-color
    cursor: default
    pointer-events: none
```

Figure 11. Indented syntax (sass-lang, syntax, 2021 b).

SCSS is used for the Portfolio since it offers various advantages. First, it is friendly to use since its syntax is similar to CSS. Second, It lets programmers use variables to store a style that could be used over and over around the document, consequently, developers can shorten the code by utilizing the variable. Furthermore, this way, it helps the editing code process to be in control, as developers only need to change the definition of variables.

Below is an example of how to define variables and the way to use them (Figure 12).



```scss
$gray-color: #888888;
$white-color: #FFFFFF;
$red-color: red;
$yellow-color: yellow;
$blue-color: blue;

.header {
    background-color: $gray-color;
    color: $white-color;
}
```

Figure 12. Variables in SCSS (sass-lang, documentation, 2021 a).

The next feature to make SCSS popular is that it applies nested syntax, which means a block of code can be nested in another one. In this way, it makes code in an organized structure, and easy to manage later, it also avoids rewriting selectors unnecessary. Below is an illustration of a piece of code following nested syntax (Figure 13).

```scss
.navbar {
  font: $ubuntu-font;
  color: $blue;
  li {
    margin-left: 1rem;
    a {
      padding: 5px;
      font-size: 1.5rem;
      span {
        font-weight: 600;
      }
    }
  }
}
```

Figure 13. Nested SCSS syntax (sass-lang, documentation, 2021 a).

Another outstanding aspect of SCSS is the @import rule, it lets humans separate code into numerous smaller files, which helps developers to maintain, modify and manage code comfortably. Afterward, they just need to import all the component files to the main file by the @import rule. Below is an example of this rule (Figure 14).

```scss
@import "source/font-awesome";
@import "source/slick";
@import "framework/bootstrap";
@import "my-custom-theme";
```

Figure 14. Example of @import rule in SCSS (sass-lang, documentation, 2021 a).

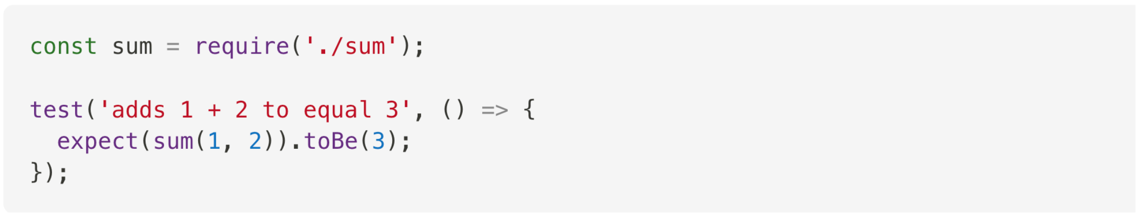
2.5 Testing

Front-end testing is performing of the usability and functionality examination of a GUI of an application. It is comprehended to validate buttons, forms, menus, as well as other elements in the application that are conspicuous to the users. The two fundamental testing tools are investigated below to clarify its testing method.

2.5.1 Jest

Jest is a JavaScript open-source framework for unit testing, preserved by Facebook, offered essentially to test elements in React Native and React web applications. It is designed to verify the accuracy of each and every Javascript codebase. It offers a feature-rich, familiar and approachable API for testing that will give results rapidly (jestjs, philosophy, 2021 b).

In every test file, its name has to be in the correct form to ensure Jest can find and run it, it should include the “test” term in its file name, such as “math.test.js”. The most common jest test is used with “expect().toBe()”, the function needs to test that is put in the “expect” bracket and the expected result will be put in the “toBe” bracket. An example of the Jest test with “expect().toBe()” is illustrated in Figure 15.



```
const sum = require('./sum');

test('adds 1 + 2 to equal 3', () => {
  expect(sum(1, 2)).toBe(3);
});
```

Figure 15. Jest test example (jestjs, Using Matchers, 2021 a).

In addition, there are more defined matchers with expect() (jestjs, Using Matchers, 2021 a):

- For truthiness: tobeNull(), tobeUndefined(), tobeDefined(), tobeTruthy(), tobeFalsy().
- For numbers: tobeGreaterThan(), tobeGreaterThanOrEqual(), tobeLessThan(), tobeLessThanOrEqual(), toEqual().
- For strings: toMatch(/expected strings/).
- For arrays and iterables: toContain().
- For exceptions: toThrow().

In fact, Jest become a popular testing framework since it is speedy to execute, relatively easy to install an setup and does not require complex configuration.

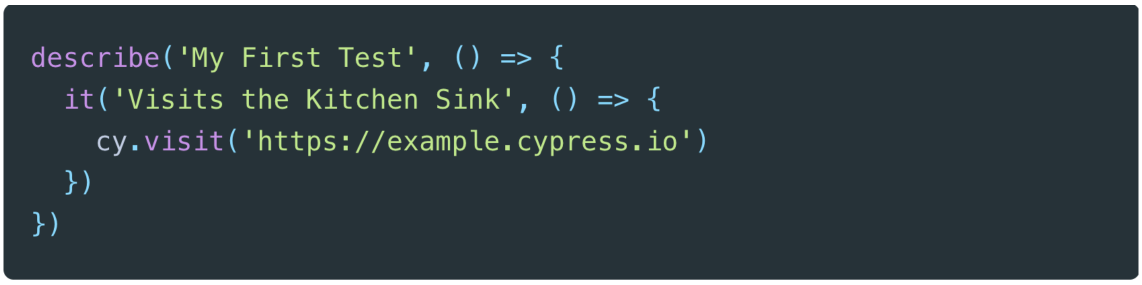
Many large organizations including Instagram, Airbnb, Twitter, and Pinterest use Jest for their React testing (testim, 2021).

2.5.2 Cypress

Cypress is a new standard framework for End to End Testing, it enables web automation testing in JavaScript including unit tests, end-to-end tests, and integration tests.

Cypress's outstanding features help it become the best choice for developers in end-to-end testing. The most of important factor is that it uses modern tool since it is a JavaScript framework based on Mocha and Chai, also utilize Nodejs to operate on browser, therefore it is trustworthy and fast for testing on most of the websites, as well as a perfect match for React and Angular application in testing field. In addition, developers might also operate cross-browser tests, which is non-functional testing to lets developers inspect whether the website runs as intended when it is accessed, with Cypress testing's help. Different from other testing methods, Cypress is not allowed to run outside of the browser executing independent commands over a network. Another highlight characteristic that helps Cypress reach most of the developers is its straightforward and rapid setup since it is based upon Nodejs, they only need to install npm Cypress, then everything is ready to go. Furthermore, Cypress offers the ability to implement and debug extremely faster. Developers can debug the applications with development tool on Chrome while tests are being run in the browser, and if any test fails, it will provide a readable and straightforward issue text, as well as may include a few suggestions to reconstruct the execution, by this ways, it is extremely helpful for a developer in debugging. Additionally, Cypress provides some screenshots of test failures, that help the process of finding errors and debugging to be light and smoother (Chakma, 2021).

There are a few steps in implementing Cypress tests. First, Cypress need to be installed by npm packages, after installing, test files should be created in folder "cypress/integration", then developer need to be careful with the file extension when creating the file, they have to ensure the file name is in the right structure, which is “name.spec.js”, otherwise Cypress can not find test files to execute. The Cypress test example is illustrated in Figure 16.



```
describe('My First Test', () => {
  it('Visits the Kitchen Sink', () => {
    cy.visit('https://example.cypress.io')
  })
})
```

Figure 16. Cypress test example (cypress, 2021).

There are numerous methos that Cypress offer to test including cy.visit(), cy.get().should(), cy.get().click(), etc. Bases on its usability, Cypress is chosen the most to implement E2E test for front-end in an application.

3 CHARACTERISTICS OF REACT

The React appearance significantly broadened the golden opportunities for developers to create a user-friendly interface. Its growth is inspected below to indicate every stage of its development from its first release till the time being.

- Pre-2011: when Facebook is growing bigger, its code base and features are increasingly implemented. Since this kind of expansion, it was exceedingly difficult to maintain the code and application's features, as a consequence, the system slowed down quickly. In this urgent circumstance, "Jordan Walke" came up with a necessary solution for Facebook, a prototype named FaxJS was built to make the code maintenance become lighter, and this is where React started to be born.
- 2013: React is introduced to the world on May 29-31 in a conference of JavaScript Community by "Tom Occhino" and "Jordan Walke". In Jordan's terms, "One of the things we strived for when we were building our component framework, is that we want to minimize the number of developer-facing mutations that the developer is exposed to." (Shaleynikov, 2018).
- 2014: "ReactJsWorldTour" was organized since more and more developers endorsed React library. Furthermore, "Chrome Developer Tools" extensions had "React Developer Tools" join the team, as well as "React Hot Loader" plugin was added.
- 2015: React released v0.13 in March, which had a valuable new feature that supports for ES6 classes, and the function "setState" was now asynchronous instead of synchronous in the former version. Meanwhile, a React mobile version was announced over the world as React Native. Later, React v0.14 was released on October, the major update in this new version is React library was split into two segments, React DOM and React. The last event in this year is Babel is used as the compiler for React.

- 2016: this year, React made a big movement from v0.14 to v15.0. The major changes are that it brought for all SVG attributes new support by browser, as well as the “document.createElement” was utilized instead of “innerHTML” for the components’ mounting, accordingly React achieved more recognition.
- 2017: v16.0 was released in September, new highlight features are added in this version including components return strings and arrays from render, error-handling is improved, as well as server renderer was entirely rebuilt. (Shaleynikov, 2018)
- 2018: the releases of v16.4 to v16.7 were noticed, with further improvements of existing features, such as regression of Nextjs and react-native-web are fixed, reconstructing React DOM and “React core library”, etc.
- 2021: over the period from 2019 to the present, React has been updated to several versions, the latest is v17.0.2, which has most of the components and structure improved, for instance, a crash in React DOM is fixed, unnecessary dependency is removed, etc.

Why React becomes the most used Front-end library over other famous frameworks, this is the most asked question over the years, to respond to this question, an analysis of React is conducted to indicate React’s advantages and disadvantages, as well as make a comparison of React and others.

3.1 Advantages of React

React has brought numerous positive aspects for the Front-end development industry. Below are the most outstanding advantages of React.

- The first elementary good thing is that it is not difficult to learn and utilize, since it has a good supply of tutorials, training resources, and documentation over the Internet. Furthermore, it is built with a JavaScript base, which is one of the most common-used programming languages, therefore, any developers who are familiar with JavaScript can easily figure out how React works and apply it to practical work.

- The second positive feature is that it enhances web applications technology to a new level by using JSX – JavaScript syntax extension, which allows developers to write HTML directly in React (within JavaScript code), by that way it provides shortcuts to reduce a bunch of unnecessary code but still maintain more functionalities. Accordingly, application development moves faster.

- The third important benefit is the components' reusability. React lets developers build applications from multiple components, each component has the responsibility of rendering each part of UI, such as buttons, sections, etc. Based on that, a component can be used again where it is needed without creating the same code with the same functionality.

- Furthermore, it uses Virtual DOM for updating, hence DOM does not need to render completely every time it changes. It helps to enhance performance to be smoother and faster. The effectiveness of Virtual DOM was indicated in React section of Chapter 2.

- In addition, it is supported by Chrome and Firefox developer tools extension. These tools are giving considerable assistance to developers in examining and editing the properties by allowing them to select the particular component on the website.

- Another good characteristic of React is that it is friendly to SEO. Standard JavaScript framework has difficulty in concerning with SEO since the search engines are basically having problems in communicating with JavaScript-heavy applications. However, React solves this issue and it helps applications be comfortably navigated on various search engines, since React applications can be deployed on the server, and Virtual DOM is rendered as a normal web page running in the browser. (javapoint, 2021 a)

Generally, React and its community is praiseworthy and it deserves to be the world's most reputed Front-end JavaScript library by its support to the web development industry.

3.2 Disadvantages of React

Everything always has a positive view come together with negative aspects, and React is not an exception. Besides significant advantages, React also brings some adverse consequences.

- The most instantly recognizable fact is the rapid pace of React's development. It can be concerned as a benefit or a drawback depending on the view and ability of every developer. Some consider this development's explosion as an advantage would declare that React's improvement has made their job become comfortable and straightforward. However, from the view of developers concern this issue has brought several obstacles, they would argue that the technology continually updates and changes extremely fast, they have to continue to relearn a new method of working with React, and it might be difficult to keep updated and get adapt to all of these changes (Insignares, 2021).

- The next feature that could be a negative point of React is JSX. JSX is the syntax extension that gives permission to JavaScript and HTML mixed together, obviously, it has huge benefits for building web applications. Nevertheless, some members of the web development community consider it as a large barrier, particularly new developers or people who do not have knowledge about JavaScript before. They complain about the complexity and difficulty in learning React. (javapoint, 2021 a)

- In addition, poor documentation could be judged as a problem for developers. This is a consequence of React’s impressive development since the technology is improved and changed extremely fast, hence, resources and documentation can not cover all of the latest updates. Furthermore, it is an open-source library, anyone can create resources, it might be difficult and time-consuming to find a good resource with all information needed. (Insignares, 2021)

Overall, every aspect of React brings both good and bad effects, the important thing is whether developers are able to take these features and make them to be an advantage for themselves.

3.3 The position of React in the marketplace

Due to the above-mentioned advantages and its other good features, React has been one of the most crucial and demanding frameworks in the marketplace nowadays.

Among other frameworks, the utilization of React by specialists worldwide accounts for 40.14% overtime of 2021. In contrast, AngularJS only takes 11.49%, and VueJS is holding 18.97%. React is used approximately 4 times more than AngularJS, and double VueJS. The illustration of React in market share is shown below in Figure 17.

By the above indicators, it is conspicuous that React maintains a desirable position in the market since it is holding nearly half of the market share.

Therefore, with the incredible development speed, React is expected to have a promising future, and bring more new features in the incoming releases.

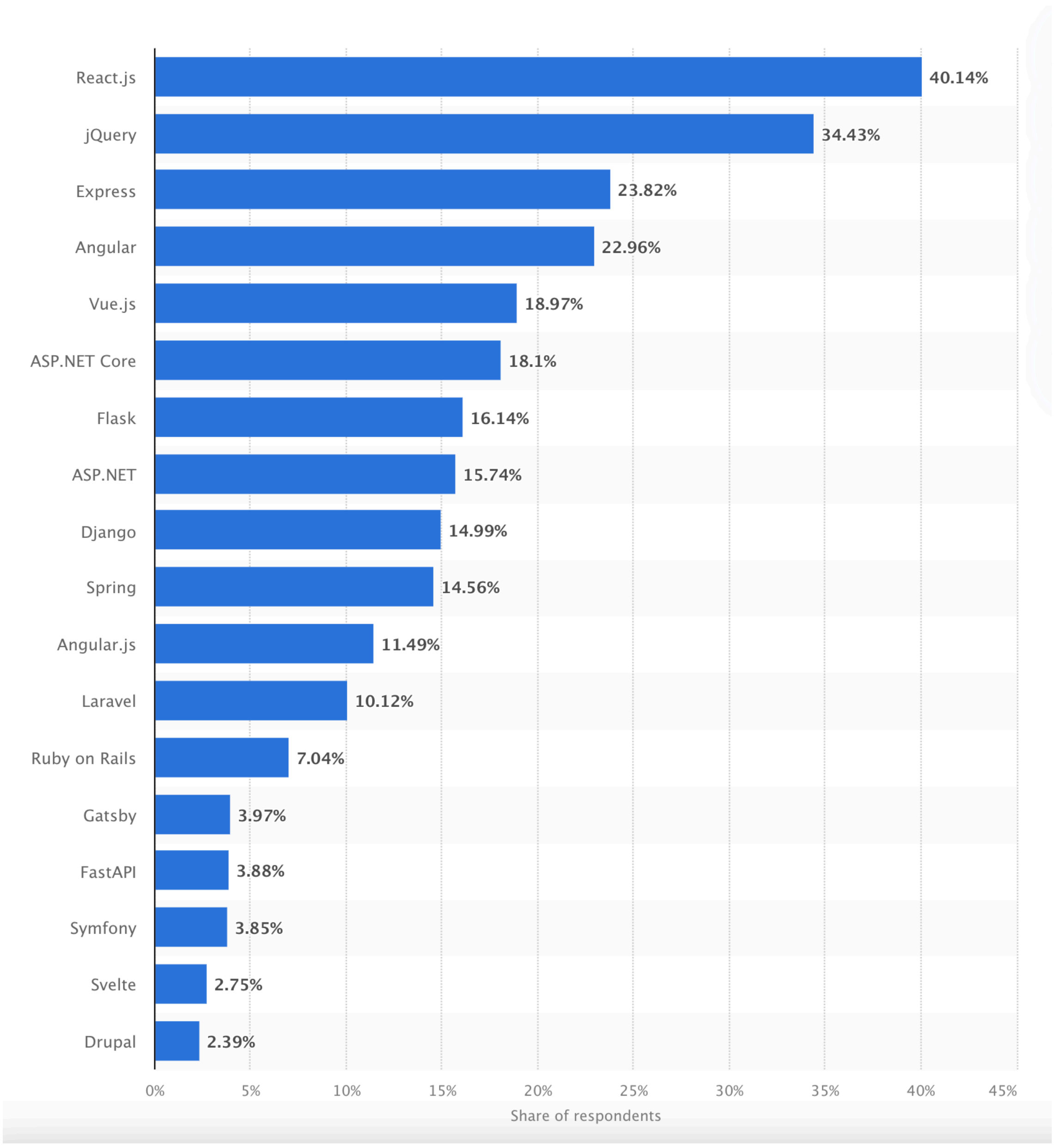


Figure 17. Most utilized web frameworks by developers worldwide, as of 2021 (statista, 2021).

3.4 Comparing React to AngularJS

In order to have a better perspective of React and its unique features in the marketplace, a comparison is examined between React and another Front-end framework, which is AngularJS, which is holding a good rate in market share.

Similar to React, AngularJS is also one of the most famous frameworks for building UI nowadays, it is first released on October 20, 2010, maintained by Google and the society of corporations and individuals.

Furthermore, It is not only built with JavaScript as React, however, it is also a JavaScript MVC (Model View Controller) framework for implementing a powerful web application, it supports models, controllers, and views. In addition, it offers developers to utilize HTML as a template language, and grant permission to the extension of HTML's syntax is to express components of applications succinctly and clearly (docs, 2021).

Table 1 is a brief of general facts about React and AngularJS.

Table 1. General facts of React and AngularJS.

	React	AngularJS
Original author	Jordan Walke	Misko Hevery
Developer(s)	Facebook, as well as community	Google, together with community
Type	Open-source JavaScript library	Fully-featured JavaScript MVC framework
Initial release	May 29, 2013	October 20, 2010
Steady release	v17.0.2 / 22 March 2021	v1.8.2 / 21 October 2020
Language	JavaScript + JSX	JavaScript + HTML
DOM	Virtual DOM	HTML DOM

Besides the above-mentioned objective, there are numerous different fields between these two frameworks. Below are some main different fields between React and AngularJS.

- In terms of concept, React allows bringing HTML into JavaScript works with the server-side rendered Virtual DOM. However, AngularJS offers developers to write JavaScript into HTML works with the client-side rendered real DOM.

- Regarding data binding, React is using one-way data binding. Since it is designed in a form of nested components, these components can receive data from their parents via the “props” attribute. Using one-way data binding, React only allows to direct data in one direction, which means data only is passed down to their children components. With this flow, it assists to control the structure complexity, as well as debugging process, etc of enormous React applications. (Khirale, 2018)

  In contrast, AngularJS is applying two-way data binding. Its templates work differently compared to React. AngularJS treats data-binding as the automatic data synchronization between the view components and model. In Angular application, the model is regarded as the single-source-of-truth for the application, and the view is an instantaneous projection of the model. Any changes to the model are instantly reflected in the view and vice versa. It implies model and view are tied together in a sync way. However, this two-way data binding approach can bring a negative effect on performance. (Khirale, 2018)

  Accordingly, React with one-way data binding has more advantages than AngularJS in a complex application.

- Concerning implementation, since React is open-source, it is necessary to add external libraries for having further features, such as calling APIs, handling dependencies, setting up tests. Therefore, React applications usually have many external dependencies installed.

  Meanwhile, AngularJS provides a huge number of endemic features and options, developers benefit from these features directly, hence this helps the setup process go faster without the confusion of selecting external options (rishabhsoft, 2019).

It might be concluded as AngularJS might save more time and effort than React in starting up an application due to its fully-features provided.

- In view of performance, due to React's own virtual DOM utilization, it makes navigation within a website become comfortable and smooth as data can get displayed without reloading the page. Accordingly, React can achieve high performance even with dynamic and complex applications.

  In the interim, MVVM (Model View ViewModel) of AngularJS offers valuable support of reducing the web page's loading speed appreciably. Furthermore, it helps to minimize the access amount to the server due to the asynchronous mode. However, with the sizeable and dynamic application, Angular brings a lower performance compared to React (rishabhsoft, 2019).

  Hence, it is recognized that in the same size application, Virtual DOM of React enables to give a faster performance than AngularJS.

- Regarding dependencies injection, this term means it is a "software design pattern" which supports making components become reusable, testable, and maintainable (rishabhsoft, 2019). However, React does not allow any notion of a built-in container for dependencies injection, but it can make injection becomes possible by using several instrument modules, for instance, ECMAScript6, Browserify, RequireJS. In contrast, AngularJS is managed to automatically detect the appropriate objects which are injected with parameters such as $filter, store, $routeParams, and $scope. Furthermore, these two functions $inject and $provide are the main core to make the dependencies injection feasible in AngularJS framework (Khirale, 2018).

In general, although the structure of these two framework is different, both desire to develop a faster application and experience a better UI. Below is popular use-cases of AngularJS and React (Figure 18).

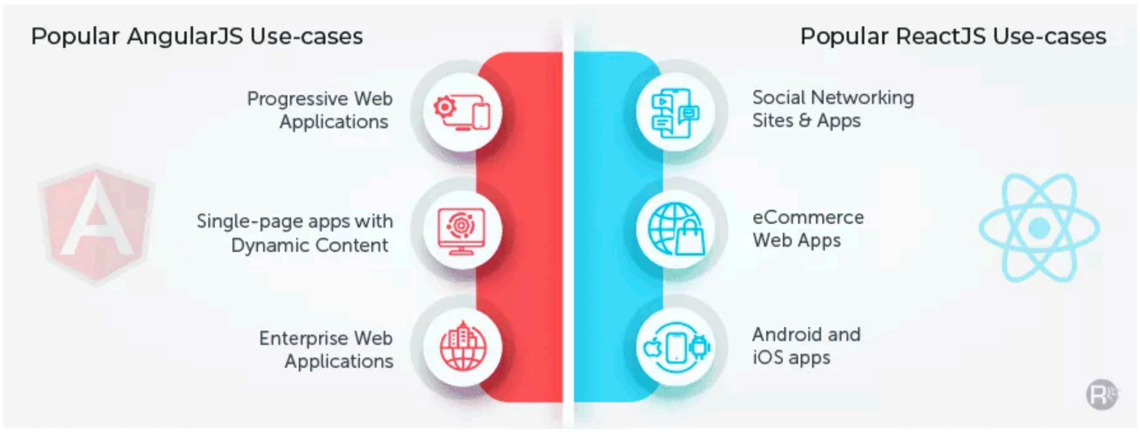

Popular AngularJS Use-cases
Progressive Web Applications
Single-page apps with Dynamic Content
Enterprise Web Applications
Popular ReactJS Use-cases
Social Networking Sites & Apps
eCommerce Web Apps
Android and iOS apps

Figure 18. AngularJS and React Uses-cases (rishabhsoft, 2019).

4 FRONT-END TESTING

4.1 Concepts of Front-end testing

Front-end testing is a process of testing the performance and functions of a website or a web application. Before putting a website in operation, developers need to investigate what is displayed on the screen and its functions run as expected

The main duties of Front-end testing are:

- Detecting client-side performance matters.
- Attesting the GUI’s functionalities in cross-device and cross-browser.
- Examining the third-party services’ integration.

Front-end testing can be separated into three main genres: E2E testing (UI testing), Integration testing, and Unit testing. In a short term, unit tests can examine how a piece of code works, integration tests verify the interaction between two pieces of code, and E2E tests can check how the entire application run under the eyes of visitors. A testing pyramid is illustrated in Figure 19 (Arafa, 2021).

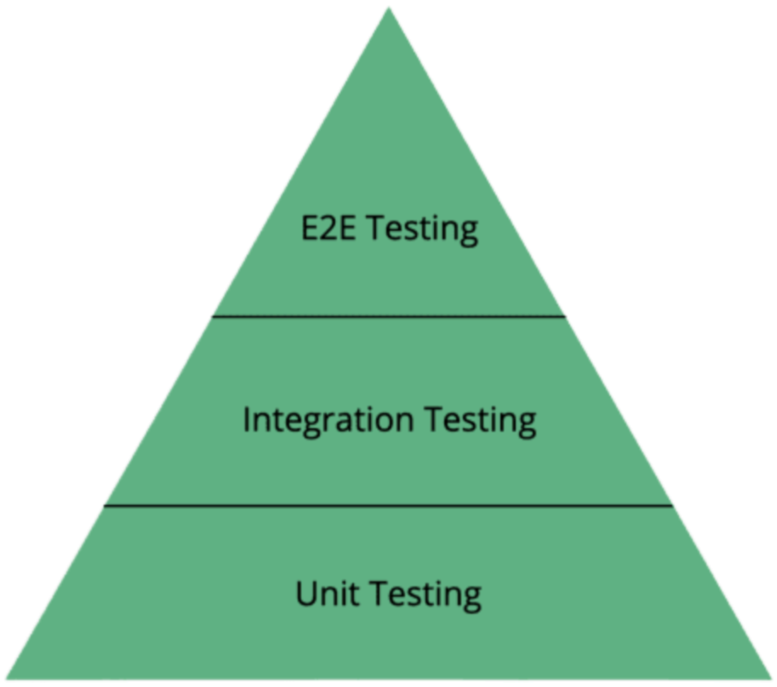


Figure 19. Types of Front-end testing (Arafa, 2021).

Three fundamental tests will be analyzed below to bring a more specific view of each Front-end testing method, it is starting from the bottom of the triangle. Furthermore, in the Portfolio, unit testing and E2E testing are set up to check elements' functionalities and the flow of the whole application run, with Jest and Cypress.

4.2 Unit testing

Unit testing, also called component testing, is implemented for testing secluded modules, as units, of the codebase. It is the first move of the testing process. The goal is to inspect the operation of isolated code pieces.

Unit testing brings many good effects to developers (pp_pankaj, 2019):

- Supporting developers achieve a deep understanding of each component's functionalities.

- Allows verifying the correctness of the module's operation and amending code.
- Enable to examine the individual part of the application without concerning whether others are accomplished, help to fix bugs in a very early stage of development.

Below is an illustration of a component or unit in an application that can be tested in the Unit testing (Figure 20) (javatpoint, Unit Testing, 2021 d).

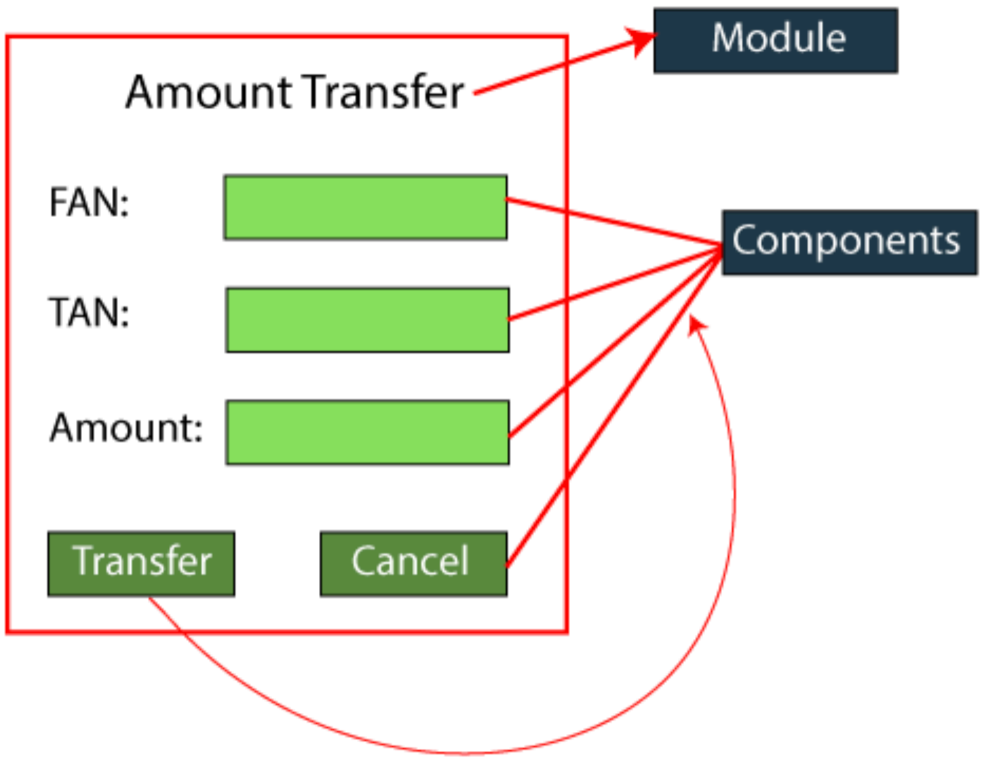


Figure 20. Example of unit or component in Unit testing (javatpoint, Unit Testing, 2021 d).

There are many tools available in the market for Unit testing, including Jest, NUnit, JUnit, PHPunit, Parasoft Jtest, EMMA, etc. Among these tools, Jest is the most common-used unit testing method, and it is chosen for testing the case study of this thesis.

4.3 Integration testing

Integration testing, also known as group components testing, is the method to verify the interconnection between various elements or validate the interfaces between units. This integration testing is normally run after Unit testing.

Many Integration testing tools are feasible for the time being, such as Citrus, FitNesse, TESSY, Protractor, Rational Integration tester, etc.

The Integration testing is demonstrated in Figure 21 (javatpoint, Integration testing, 2021 b).

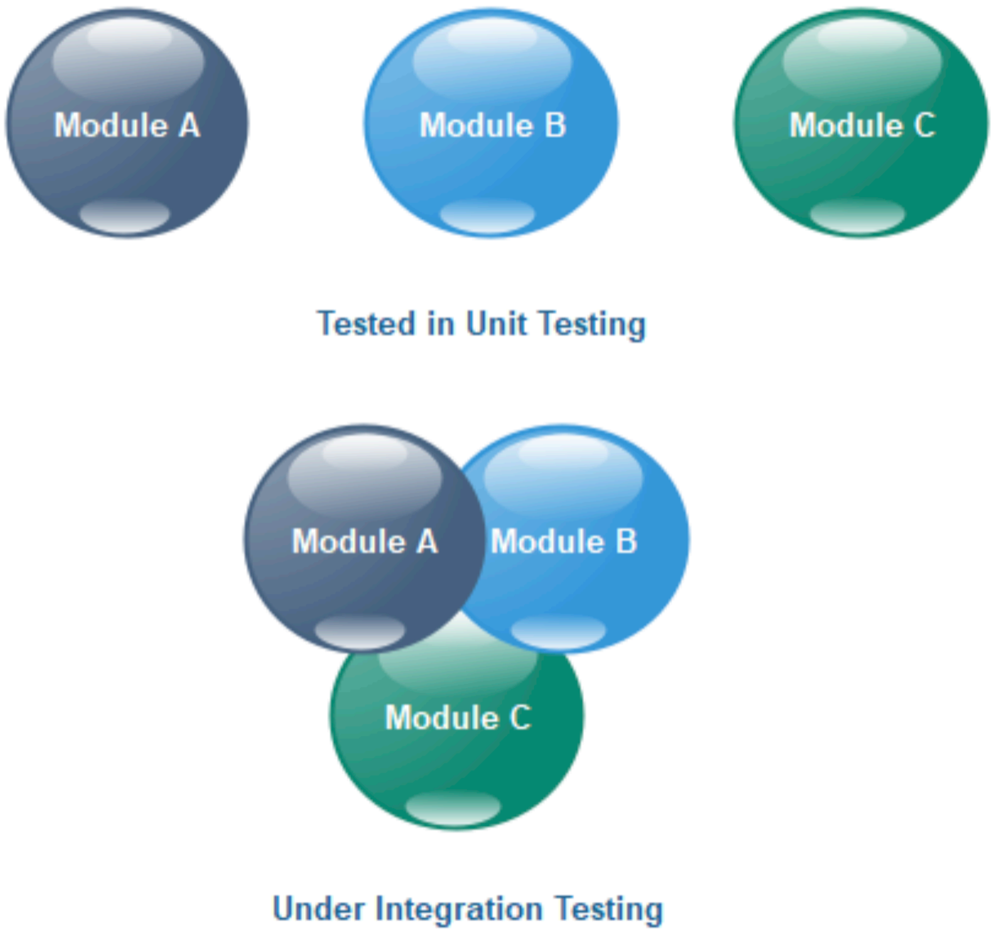


Figure 21. Integration testing (javatpoint, Integration testing, 2021 b).

4.4 E2E testing

E2E test is the final stage of the testing flow. It is defined as software experimentation to assure the whole application operates as intended. In order to check different paths that people may take during all the time they are accessing the website, the test simulates the possible scenario of the end-user from the very first start to end (smartbear, 2021).

E2E tests offer numerous advantages to an application (katalon, 2019):

- Broaden the test scope.
- Verify the application’s accuracy and health.
- Optimize the functionality of the application to bring the best experience to customers.
- Shorten “time to market”.
- Lower cost and reveal bugs.

E2E test usually has 4 main stages in its lifecycle. The lifecycle can help to explain the operation method of the tests. An illustration of its lifecycle is in Figure 22.

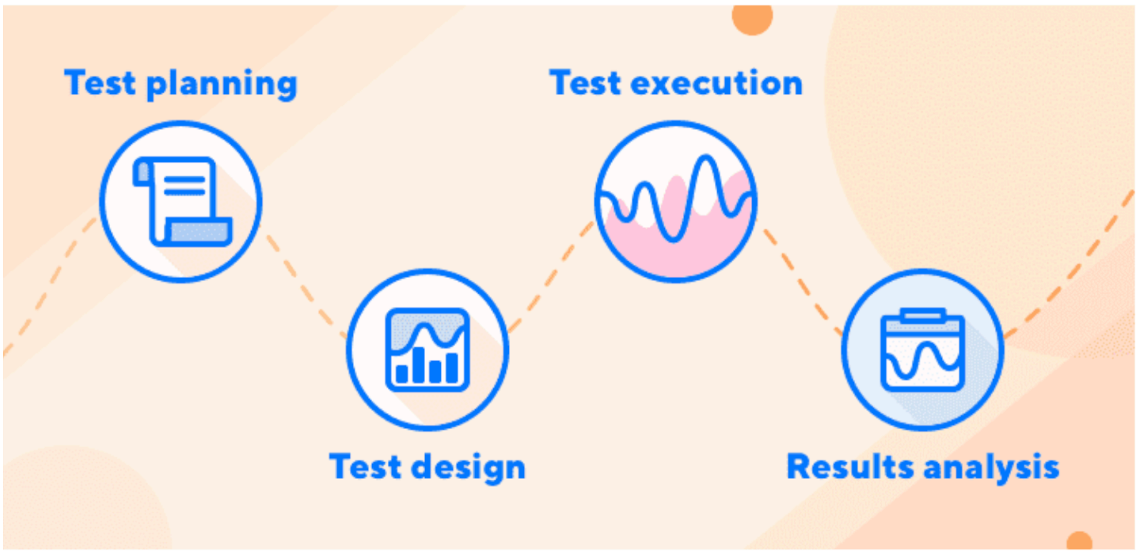


Figure 22. E2E testing lifecycle (katalon, 2019).

In the first stage of the lifecycle, test planning, developers have to determine crucial tasks, associated workflow, and resources. After the key tasks are pointed out, it moves to the second stage, test design, in this level, test specifications need to be clarified, then a test case is created, moreover, developers have to analyze the risk and test usage before arranging test. When the preparation is ready, these test cases are executed in stage 3, test execution, then testing results are documented. In the last stage, developers will analyze the test results and evaluate testing, furthermore, implement additional tests if necessary.

Cypress is one of the most popular E2E testing frameworks, therefore, it is chosen for testing the case study in this thesis.

5 PORTFOLIO AND ITS TESTING METHODS

5.1 Importance of Portfolio in Job hunting

In society nowadays, humans have to face immense competition during job hunting. They have to find a way to stand out among others candidates, and impress the recruiter. The CV of a candidate is a key to deciding whether they can be shortlisted. Furthermore, if a candidate can make a portfolio besides their CV, it is giving a plus to their profile, especially for web developers, the portfolio is indispensable. A Portfolio can be understood as a collection of an applicant's education, skillsets, belief, experiences, achievements, etc, in a detailed and live form. It can be seen as a modern-day CV, also a showcase for web developers to prove their skills and abilities in practice, to convince the employer.

There are a few reasons that people need a portfolio in their job hunt:

- Providing a detailed view of the applicant's work history, belief, qualification, etc. It is not possible with a CV since a CV is a brief overview of a job seeker, the recruiter can not have a comprehensive look through a few bullet points or descriptions on the CV.
- Visuals attract attention, the employer can be impressed or be attracted by images, graphics, or animation presented in the portfolio.
- Saving the HR time, the HR may receive hundreds of applications a day, and with the portfolio, job seekers can make themselves being selected among others by giving concrete evidence of their skills set through the portfolio.
- Sharing the goals and current progress, the candidates can share the current goals, the current project they are working on in their portfolio, by this way, they can show the improvement day-by-day and tell the HR what kind of people they are.

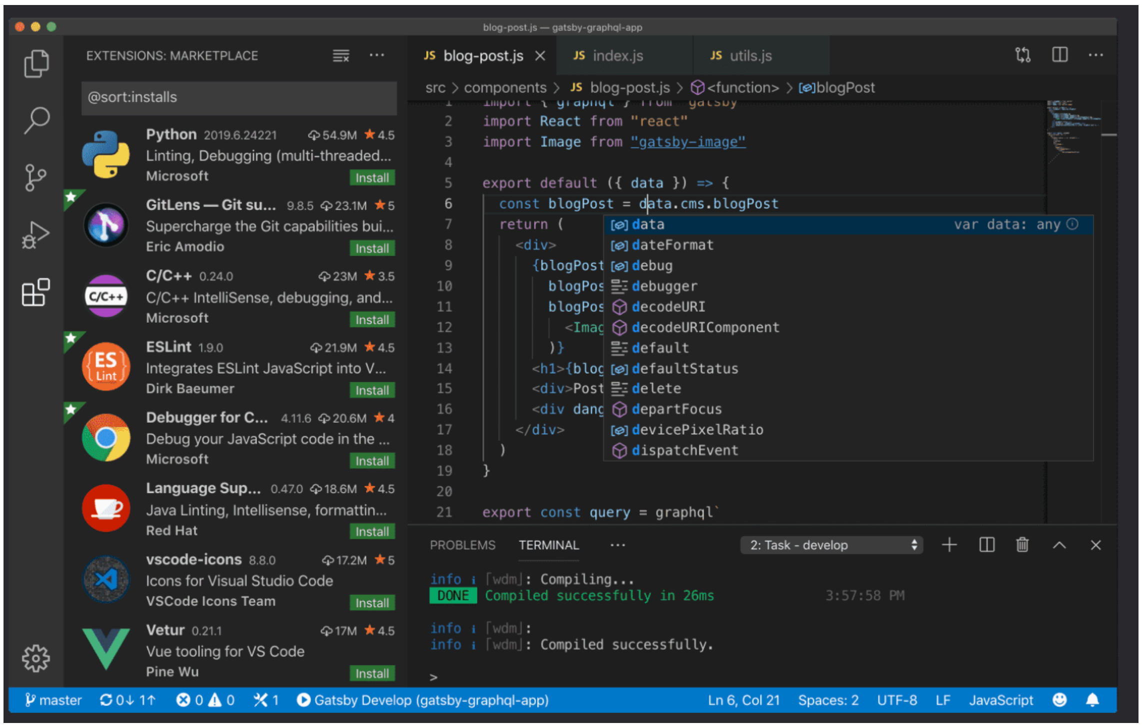
- Personal evaluation, with the portfolio, helps the candidates keep track of what they are doing, and notice if their work is in the right direction to reach the goal. (Recruiter, 2021)

Due to the above-mentioned aspects, a Portfolio can be considered as an important and necessary part of the candidate in the job hunt. That is also the reason to choose Portfolio to become a case study built along with React in this thesis.

5.2 Code environment setup

5.2.1 Code editor

In order to build an application, the first thing developers need is a code editor. Visual Studio Code (VSC) is one of the best code editors at the present, and it is used to build the Portfolio in this thesis. VSC can be utilized for Linux, macOS, and Windows, managed by Microsoft. Below is a picture of VSC (Figure 23).



Python 2019.6.24221 54.9M ★4.5
Linting, Debugging (multi-threaded...
Microsoft Install

GitLens — Git su... 9.8.5 23.1M ★5
Supercharge the Git capabilities bui...
Eric Amodio Install

C/C++ 0.24.0 23M ★3.5
C/C++ IntelliSense, debugging, and...
Microsoft Install

ESLint 1.9.0 21.9M ★4.5
Integrates ESLint JavaScript into V...
Dirk Baeumer Install

Debugger for C... 4.11.6 20.6M ★4
Debug your JavaScript code in the ...
Microsoft Install

Language Sup... 0.47.0 18.6M ★4.5
Java Linting, Intellisense, formattin...
Red Hat Install

vscode-icons 8.8.0 17.2M ★5
Icons for Visual Studio Code
VSCode Icons Team Install

Vetur 0.21.1 17M ★4.5
Vue tooling for VS Code
Pine Wu Install

```
import { graphql } from "gatsby"
import React from "react"
import Image from "gatsby-image"

export default ({ data }) => {
  const blogPost = data.cms.blogPost
  return (        data                      var data: any
    <div>         dateFormat
      {blogPost   debug
        blogPos   debugger
        blogPos   decodeURI
          <Imag   decodeURIComponent
        )}        default
      <h1>{blog   defaultStatus
      <div>Post   delete
      <div dang   departFocus
    </div>        devicePixelRatio
  )               dispatchEvent
}

export const query = graphql`
```

```
info ℹ ｢wdm｣: Compiling...
DONE  Compiled successfully in 26ms                    3:57:58 PM

info ℹ ｢wdm｣:
info ℹ ｢wdm｣: Compiled successfully.

>
```

master 0↓ 1↑ 0 0 1 Gatsby Develop (gatsby-graphql-app) Ln 6, Col 21 Spaces: 2 UTF-8 LF JavaScript

Figure 23. example of VSC.

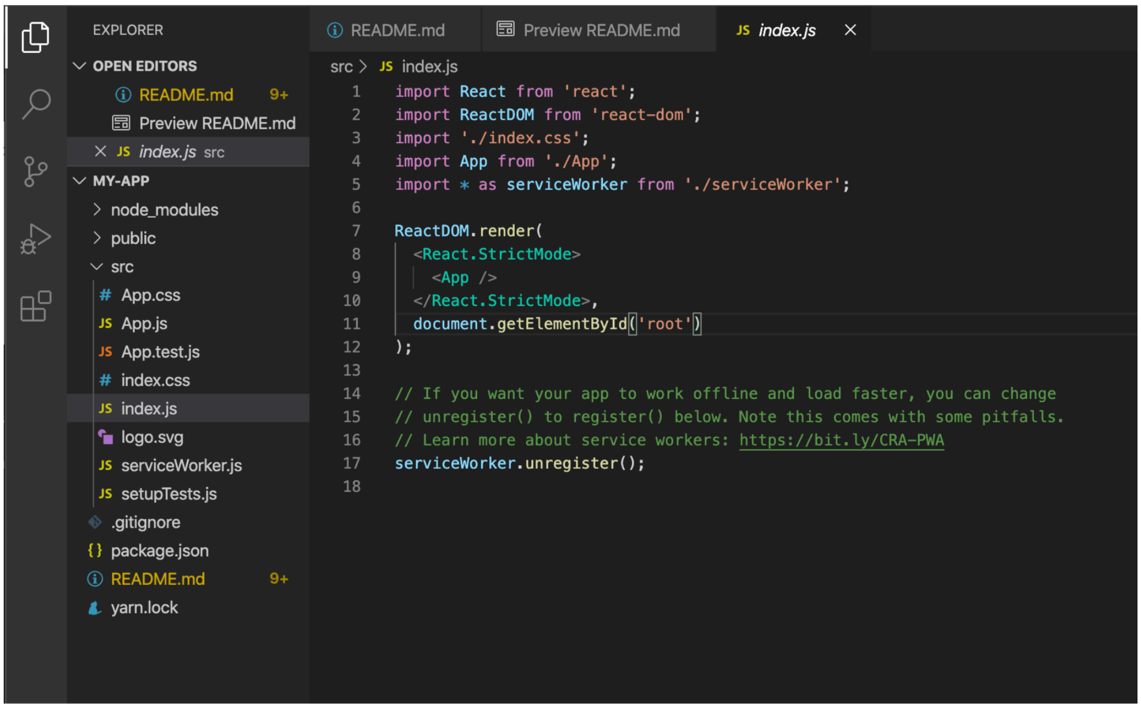
5.2.2 Application initialization

The Portfolio is initialized by the Create-React-App command. The command is shown in Figure 24 below.

```
npx create-react-app my-app
cd my-app
npm start
```

Figure 24. Create-React-App command (reactjs, Create a New React App, 2021 a).

The Create-React-App command sets up all the things needed to build an application, including *node-modules* folder, *src* folder, *package.json*. The folder structure is described below in Figure 25.

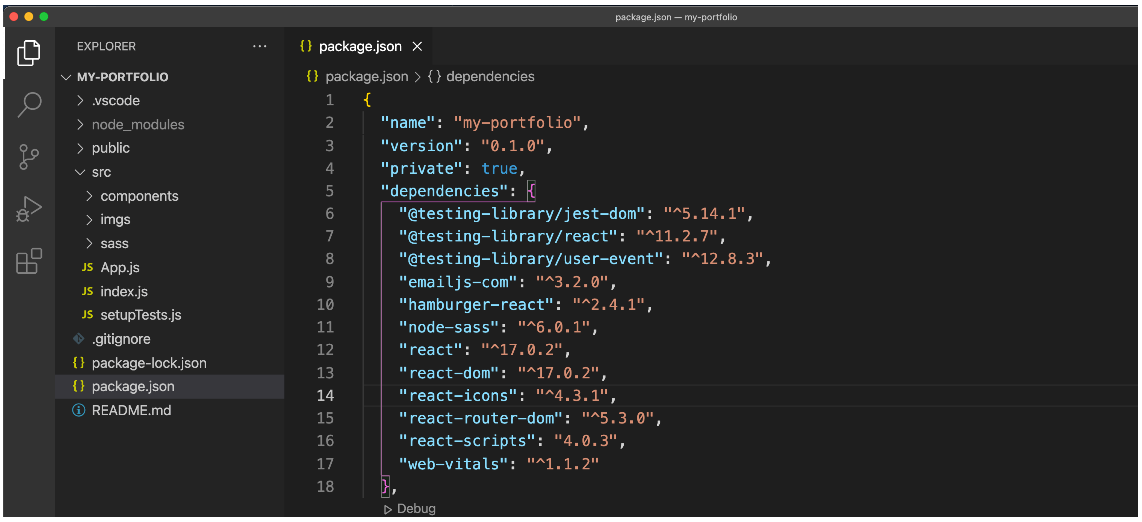


```
import React from 'react';
import ReactDOM from 'react-dom';
import './index.css';
import App from './App';
import * as serviceWorker from './serviceWorker';

ReactDOM.render(
  <React.StrictMode>
    <App />
  </React.StrictMode>,
  document.getElementById('root')
);

// If you want your app to work offline and load faster, you can change
// unregister() to register() below. Note this comes with some pitfalls.
// Learn more about service workers: https://bit.ly/CRA-PWA
serviceWorker.unregister();

```

Figure 25. React app folder structure (code, 2021).

5.2.3 Dependencies package installation

This Portfolio uses SCSS for styling, as well as React Router for handling the navigation between routes. Therefore, it is necessary to install node-sass and react-router as dependencies in the app. Furthermore, there are some other dependencies needed to provide more functions for the app, including emailjs-com, hamburger-react. All installed dependencies can be seen in the *package.json* file in Figure 26 below.



```json
{
  "name": "my-portfolio",
  "version": "0.1.0",
  "private": true,
  "dependencies": {
    "@testing-library/jest-dom": "^5.14.1",
    "@testing-library/react": "^11.2.7",
    "@testing-library/user-event": "^12.8.3",
    "emailjs-com": "^3.2.0",
    "hamburger-react": "^2.4.1",
    "node-sass": "^6.0.1",
    "react": "^17.0.2",
    "react-dom": "^17.0.2",
    "react-icons": "^4.3.1",
    "react-router-dom": "^5.3.0",
    "react-scripts": "4.0.3",
    "web-vitals": "^1.1.2"
  },
```

Figure 26. Package.json file.

5.3 Implementation of components

The Portfolio is divided into 4 main categories by the name of each section, such as home, about, projects, contact, and 3 other files which are NavBar (navigation bar), Footer, Particles (for the background animation). Below is the picture of the component structure in the Portfolio application (Figure 27).

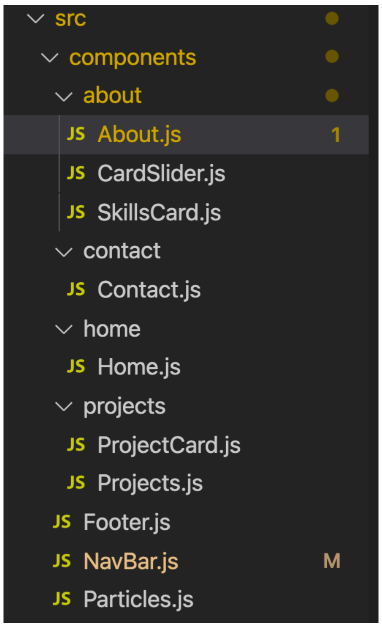


Figure 27. Components structure.

5.3.1 Navigation Bar

The navigation bar consists of 4 routes leading to 4 pages (components) of the application that are home page, the about page, the projects page, and the contact page. The illustration of the navigation bar is in Figure 28.

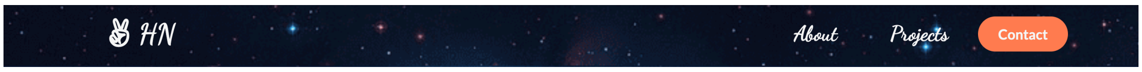
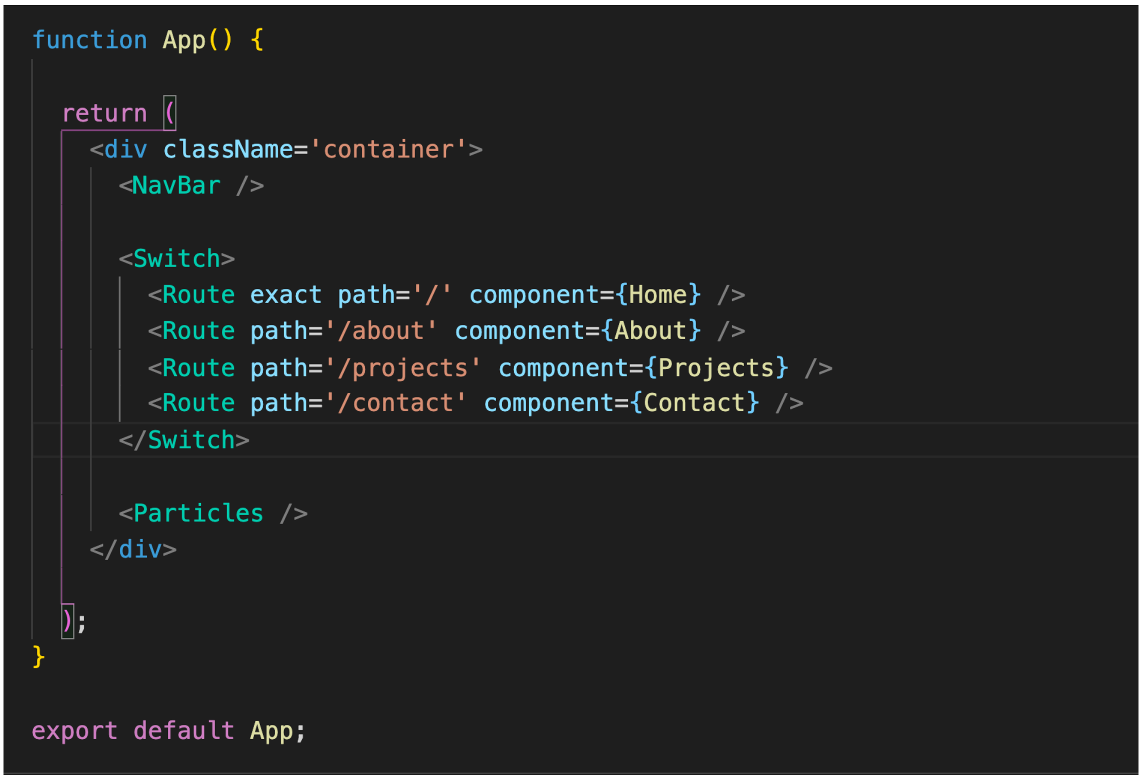


Figure 28. Navigation Bar.

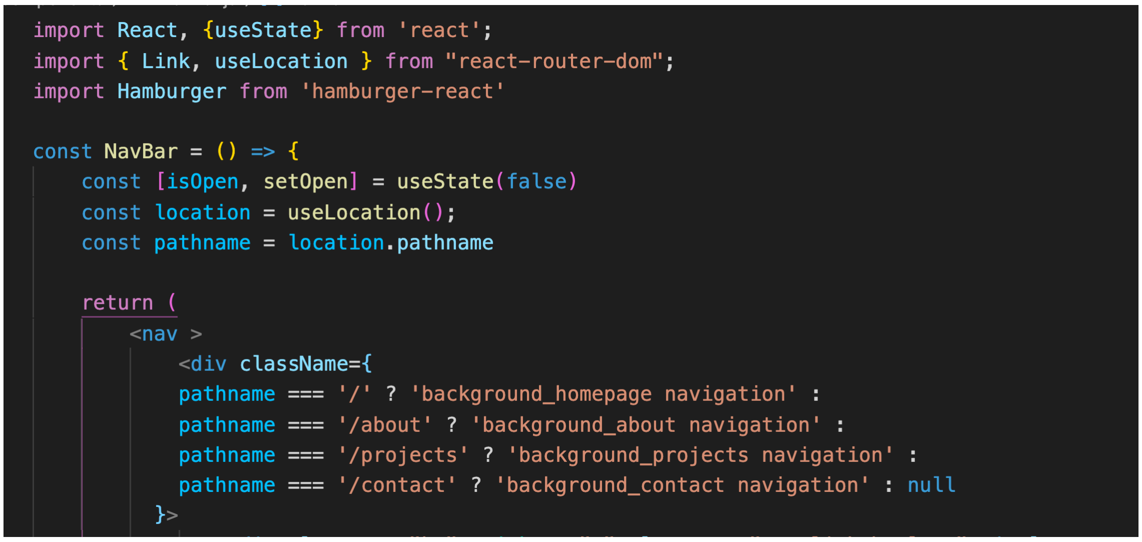
All these components are defined with appropriate routes by using <Switch> and <Route> from react-router-dom in App.js. It is shown in Figure 29 below.



```
function App() {

  return (
    <div className='container'>
      <NavBar />

      <Switch>
        <Route exact path='/' component={Home} />
        <Route path='/about' component={About} />
        <Route path='/projects' component={Projects} />
        <Route path='/contact' component={Contact} />
      </Switch>

      <Particles />
    </div>

  );
}

export default App;
```

Figure 29. App.js structure.

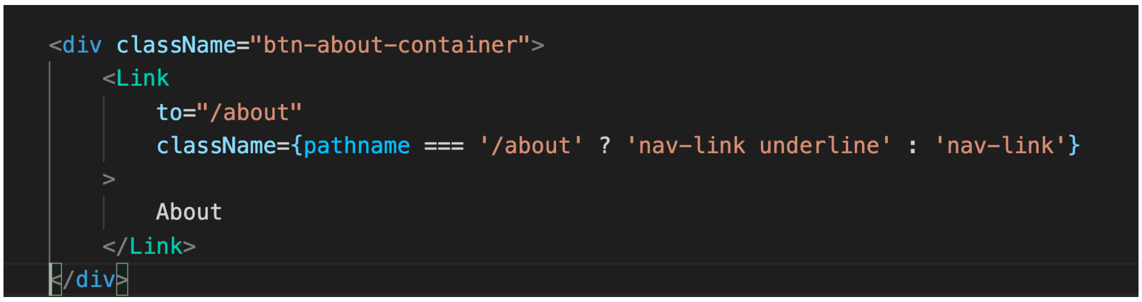
In NavBar component, <Link> and <*useLocation*> need to be imported from react-router-dom in order to be able to use, <Link> is to create links in the application, and the <useLocation> is to check the current location's pathname. The aim of checking the current location is to change the background of the navigation bar since it will have a different background based on which page is presented on the screen. Below is the piece of code utilizing the <useLocation> to get data of the current location (Figure 30).



```
import React, {useState} from 'react';
import { Link, useLocation } from "react-router-dom";
import Hamburger from 'hamburger-react'

const NavBar = () => {
    const [isOpen, setOpen] = useState(false)
    const location = useLocation();
    const pathname = location.pathname

    return (
        <nav >
            <div className={
            pathname === '/' ? 'background_homepage navigation' :
            pathname === '/about' ? 'background_about navigation' :
            pathname === '/projects' ? 'background_projects navigation' :
            pathname === '/contact' ? 'background_contact navigation' : null
          }>
```

Figure 28. <useLocation> from react-router-dom.

In Figure 30, useLocation() hook is used to access to location object that defines the current URL, its data is assigned to variable “location”, and the current pathname of URL can be filtered from variable “location.pathname” and it is assigned to variable “pathname”. In <div>, className is used for calling the class in SCSS – used for styling, it is using the conditional operator to implement the appropriate background class name according to data of the variable “pathname”.

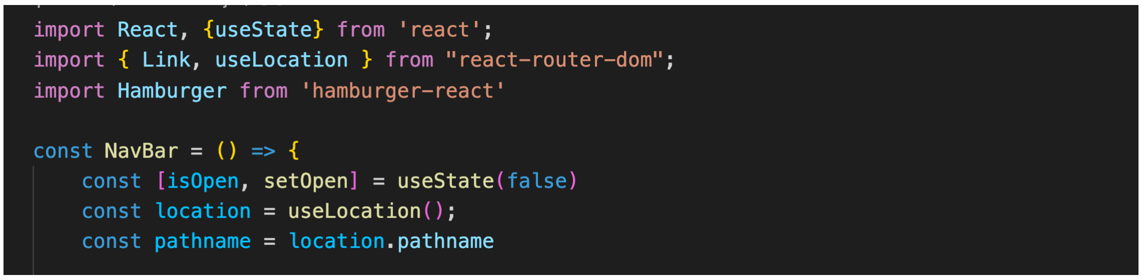


```
<div className="btn-about-container">
    <Link
        to="/about"
        className={pathname === '/about' ? 'nav-link underline' : 'nav-link'}
    >
        About
    </Link>
</div>
```

Figure 29. the usage of <Link>.

In Figure 31, it illustrates how to implement a <Link> component, the link is established by property “to”, in this case, “to” is assigned to value “/about”. After users click on “About” in the navigation bar, the URL will be changed to link “/about”.

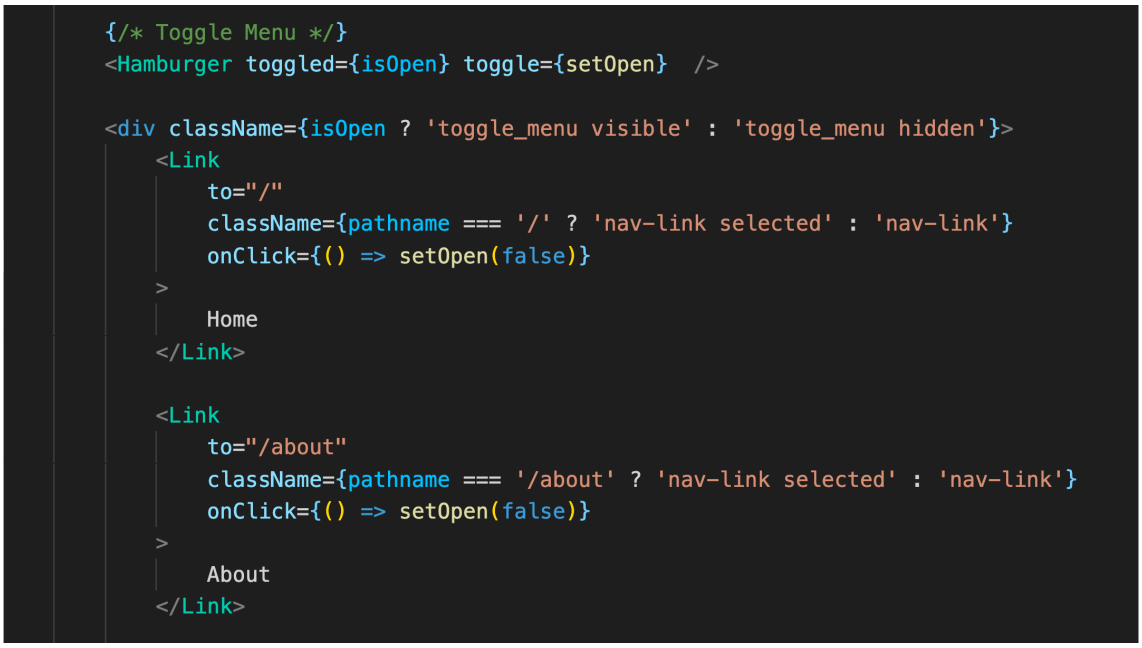
Furthermore, this Navigation Bar component also uses Hamburger from the “hamburger-react” library to create a hamburger button and toggle menu, which replace for navigation bar in the mobile version. In addition, it uses the state hook to create a state for controlling the open-close function of the hamburger button and toggle menu.



```
import React, {useState} from 'react';
import { Link, useLocation } from "react-router-dom";
import Hamburger from 'hamburger-react'

const NavBar = () => {
    const [isOpen, setOpen] = useState(false)
    const location = useLocation();
    const pathname = location.pathname
```

Figure 30. useState hook.

In Figure 32, the hook “useState” to create a state and assign it to variable isOpen, then “setOpen” is a function to manage the state. Below is a piece of code of how to apply the function setOpen into the toggle menu (Figure 33).



```
{/* Toggle Menu */}
<Hamburger toggled={isOpen} toggle={setOpen}  />

<div className={isOpen ? 'toggle_menu visible' : 'toggle_menu hidden'}>
    <Link
        to="/"
        className={pathname === '/' ? 'nav-link selected' : 'nav-link'}
        onClick={() => setOpen(false)}
    >
        Home
    </Link>

    <Link
        to="/about"
        className={pathname === '/about' ? 'nav-link selected' : 'nav-link'}
        onClick={() => setOpen(false)}
    >
        About
    </Link>
```

Figure 31. Toggle menu.

In this Figure 33, the function “setOpen” is assigned to the toggle property of Hamburger to control the open-close status of the hamburger button. Moreover, this function is also utilized in the onClick property of <Link>, when onClick is triggered, setOpen is activated to change the state “isOpen” to be false, leading to closing toggle menu.

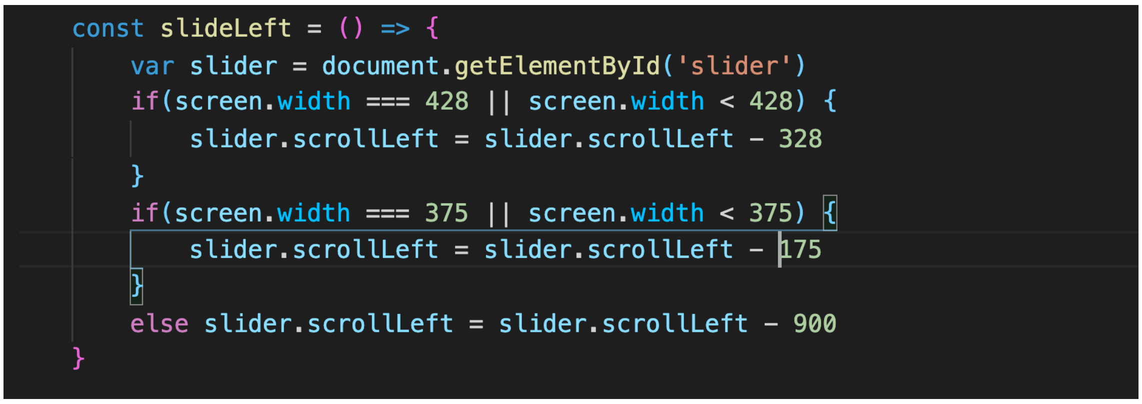
5.3.2 Home page

The homepage is the very first page when people access the application, this page is to introduce the name and title of the Portfolio’s owner. Furthermore, a profile picture is also displayed on this page.

Besides name and title, a button Explore me attach a link to about page is also presented underneath the title to attract people’s attention.

5.3.3 About page

This page provides the background and the professional skillset of the Portfolio’s owner. The skillsets are divided into 4 main cards, and the CardSlider component is created to handle the motion between these cards. Furthermore, it uses functions “slideLeft” and “slideRight” to scroll the cards. The function “slideLeft” is demonstrated in Figure 34.

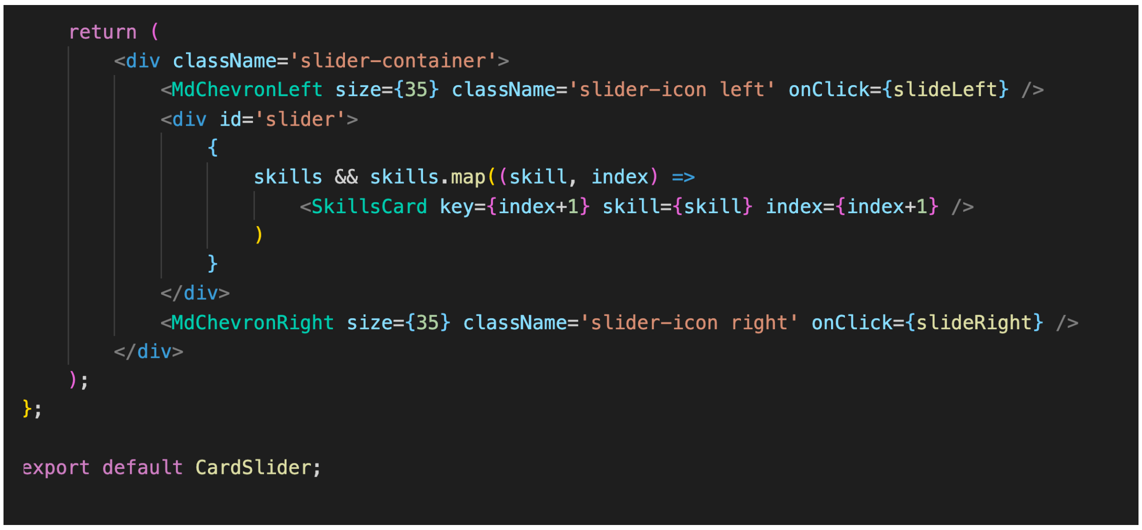


```
const slideLeft = () => {
    var slider = document.getElementById('slider')
    if(screen.width === 428 || screen.width < 428) {
        slider.scrollLeft = slider.scrollLeft - 328
    }
    if(screen.width === 375 || screen.width < 375) {
        slider.scrollLeft = slider.scrollLeft - 175
    }
    else slider.scrollLeft = slider.scrollLeft - 900
}
```

Figure 32. Function slideLeft.

The functions “slideLeft” and “slideRight” are implemented in the arrow icon MdchevronLeft and MdchevronLeft from react-icon.

In addition, the JavaScript array map function is used to take data of each object skill in array skills, and send this information to the “SkillsCard” component as props including skill and index. This structure of code is illustrated in Figure 35 below.

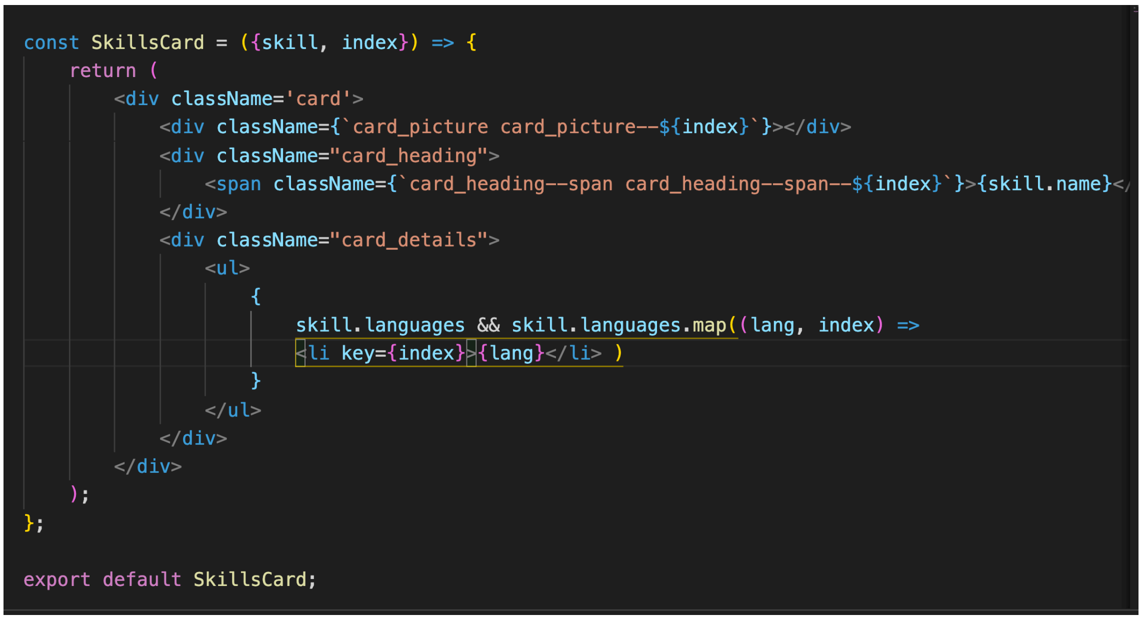


```
    return (
        <div className='slider-container'>
            <MdChevronLeft size={35} className='slider-icon left' onClick={slideLeft} />
            <div id='slider'>
                {
                    skills && skills.map((skill, index) =>
                        <SkillsCard key={index+1} skill={skill} index={index+1} />
                    )
                }
            </div>
            <MdChevronRight size={35} className='slider-icon right' onClick={slideRight} />
        </div>
    );
};

export default CardSlider;
```

Figure 33. CardSlider component.

After receiving data as props from the parent component, SkillsCard renders this information in the shape of a card.

The code base is shown below in Figure 36.

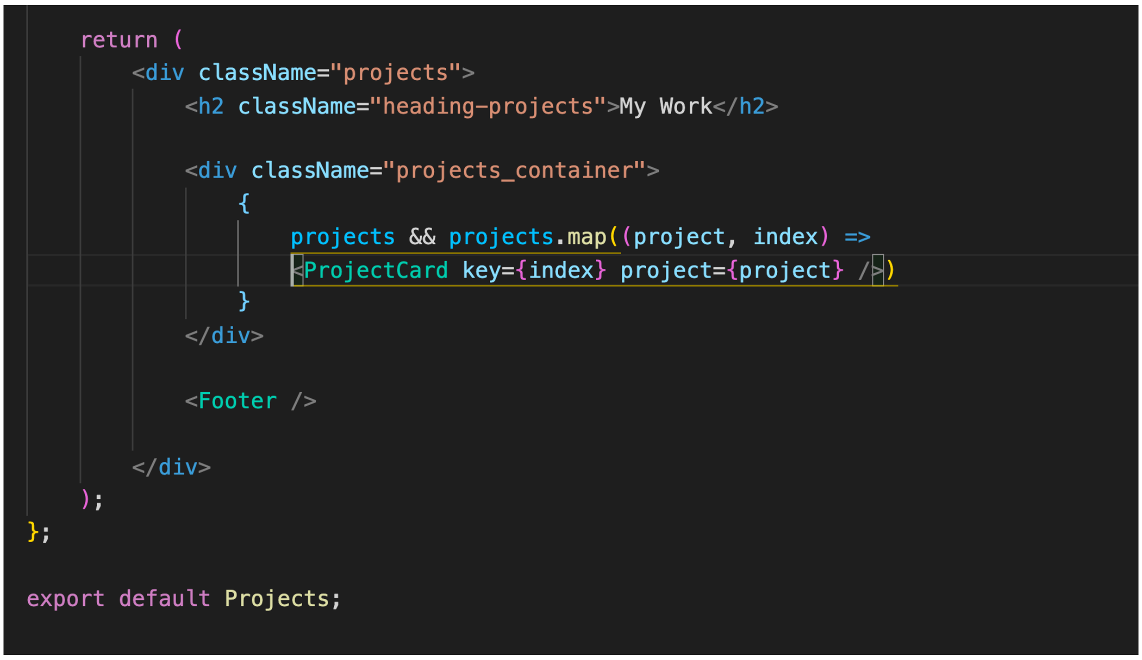


```
const SkillsCard = ({skill, index}) => {
    return (
        <div className='card'>
            <div className={`card_picture card_picture--${index}`}></div>
            <div className="card_heading">
                <span className={`card_heading--span card_heading--span--${index}`}>{skill.name}</
            </div>
            <div className="card_details">
                <ul>
                    {
                        skill.languages && skill.languages.map((lang, index) =>
                        <li key={index}>{lang}</li> )
                    }
                </ul>
            </div>
        </div>
    );
};

export default SkillsCard;
```

Figure 34. SkillsCard component.

5.3.4 Projects page

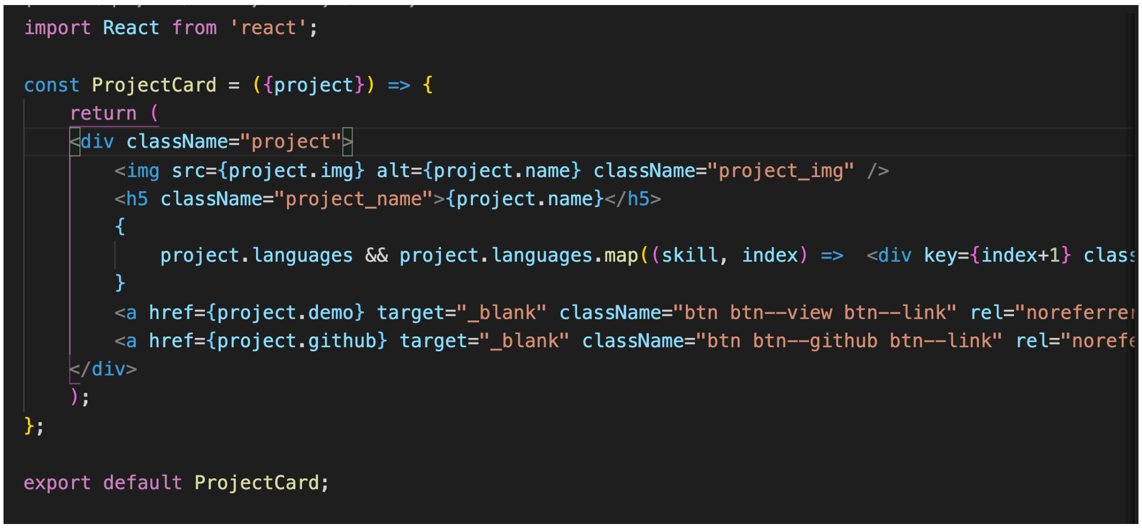
The goal of the Project page is to demonstrate the author’s work and her own projects. Similar to the skillsets on the About page, each project is arranged and presented on screen in the shape of a card. The map function is used to process each item in an array of projects, and pass the data to children’s components as props project, each item is performed by displaying a ProjectCard component. Below is the codebase of the Projects component (Figure 37).



```
    return (
        <div className="projects">
            <h2 className="heading-projects">My Work</h2>

            <div className="projects_container">
                {
                    projects && projects.map((project, index) =>
                    <ProjectCard key={index} project={project} />)
                }
            </div>

            <Footer />

        </div>
    );
};

export default Projects;
```

Figure 35. Projects component codebase.

The ProjectCard component receives the data passed as props, and renders this data in HTML element to put it into the shape of the card. Figure 38 below shows the code base of the ProjectCard component.

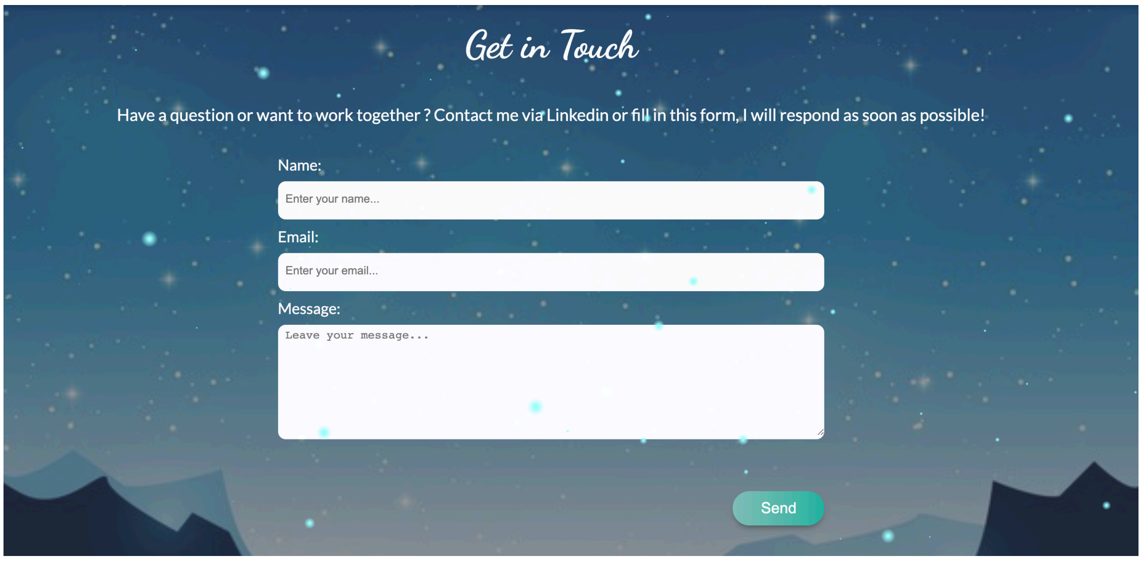


```
import React from 'react';

const ProjectCard = ({project}) => {
    return (
    <div className="project">
        <img src={project.img} alt={project.name} className="project_img" />
        <h5 className="project_name">{project.name}</h5>
        {
            project.languages && project.languages.map((skill, index) =>  <div key={index+1} class
        }
        <a href={project.demo} target="_blank" className="btn btn--view btn--link" rel="noreferre
        <a href={project.github} target="_blank" className="btn btn--github btn--link" rel="norefe
    </div>
    );
};

export default ProjectCard;
```

Figure 36. Projectcard component codebase.

5.3.5 Contact page

Contact page displays a form to ask visitors to leave their information, such as name, email, and message if they desire to contact the author. The form is shown below in Figure 39.



Get in Touch

Have a question or want to work together ? Contact me via Linkedin or fill in this form, I will respond as soon as possible!

Name:

Enter your name...

Email:

Enter your email...

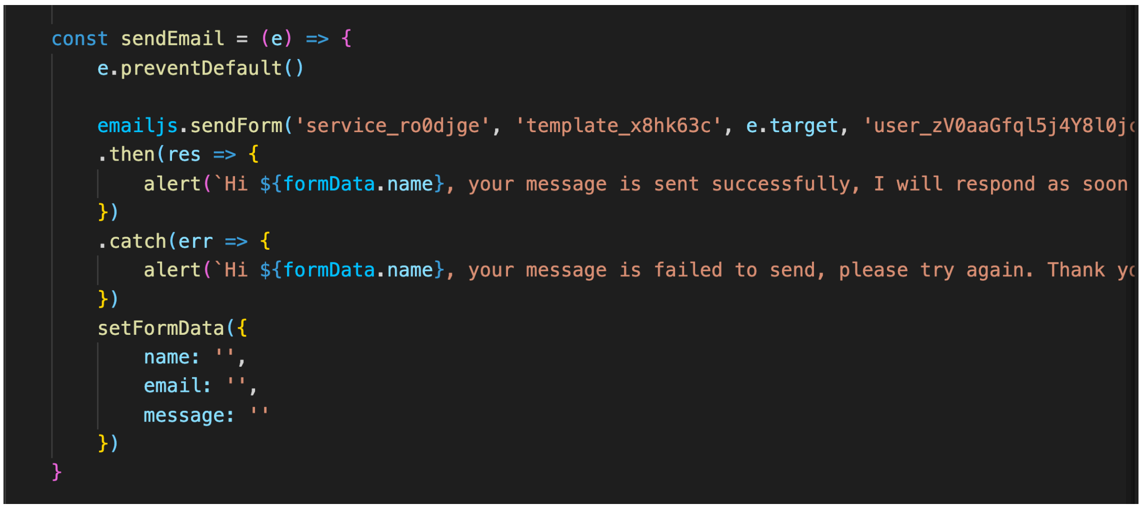
Message:

Leave your message...

Send

Figure 37. Contact page.

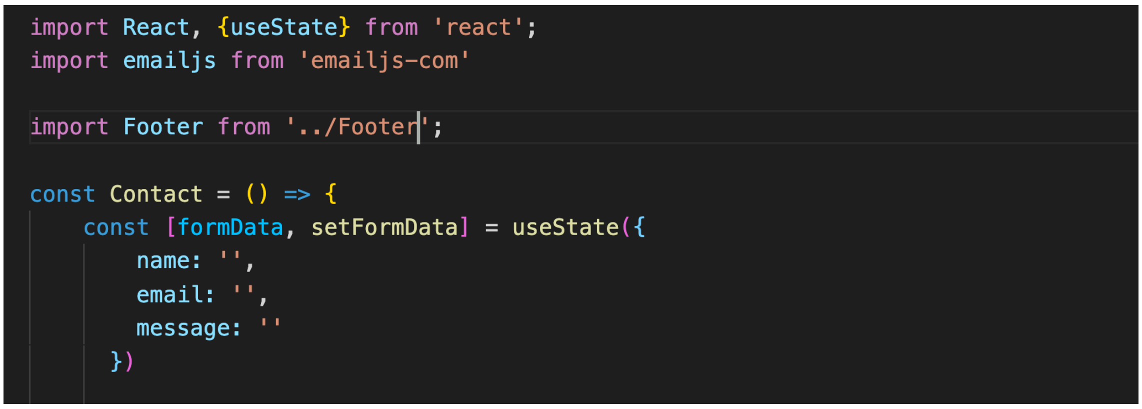
In this Contact component, the “emailjs” from the “emailjs-com” library is used to handle the function submit the form. Emailjs offers the method to send form data to the author’s email, it works as an instant back-end sending an email immediately from JavaScript. After sending information, there is an alert to let people know whether the message is sent successfully. Below is the “sendEmail” function handling the form submission (Figure 40).



```
const sendEmail = (e) => {
    e.preventDefault()

    emailjs.sendForm('service_ro0djge', 'template_x8hk63c', e.target, 'user_zV0aaGfql5j4Y8l0j
    .then(res => {
        alert(`Hi ${formData.name}, your message is sent successfully, I will respond as soon
    })
    .catch(err => {
        alert(`Hi ${formData.name}, your message is failed to send, please try again. Thank yo
    })
    setFormData({
        name: '',
        email: '',
        message: ''
    })
}
```

Figure 38. Submit form function.

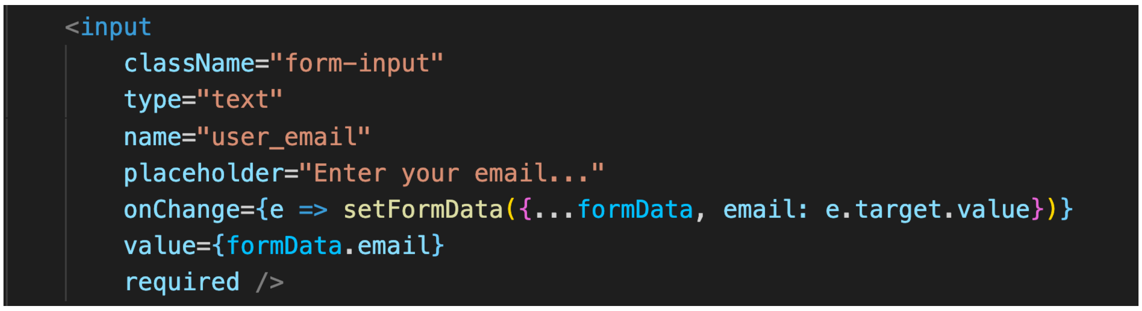
Furthermore, this component also “useState” hook to manage the form’s data. The form’s data is identified as an object which has properties including name, email, and message. Below is Figure 41 describes the state object “formData”, as well as function “setFormData” that modifies the state, which both are created from “useState” hook.



```
import React, {useState} from 'react';
import emailjs from 'emailjs-com'

import Footer from '../Footer';

const Contact = () => {
    const [formData, setFormData] = useState({
        name: '',
        email: '',
        message: ''
      })
```

Figure 39. State in Contact component.

The value of each property in object formData is assigned to the value of appropriate input. This makes a connection between the component state and the form. Moreover, the function onChange handling the “setFormData” allows modifying the state every moment the input value changes. Below is an illustration of an email input (Figure 42).



```
<input
    className="form-input"
    type="text"
    name="user_email"
    placeholder="Enter your email..."
    onChange={e => setFormData({...formData, email: e.target.value})}
    value={formData.email}
    required />
```

Figure 40. Form input in the Contact component.

5.4 Front-end testing

In this Portfolio, the Jest and React testing libraries are chosen for unit tests, furthermore, in order to test from the visitor’s experience, Cypress is implemented to satisfy this goal.

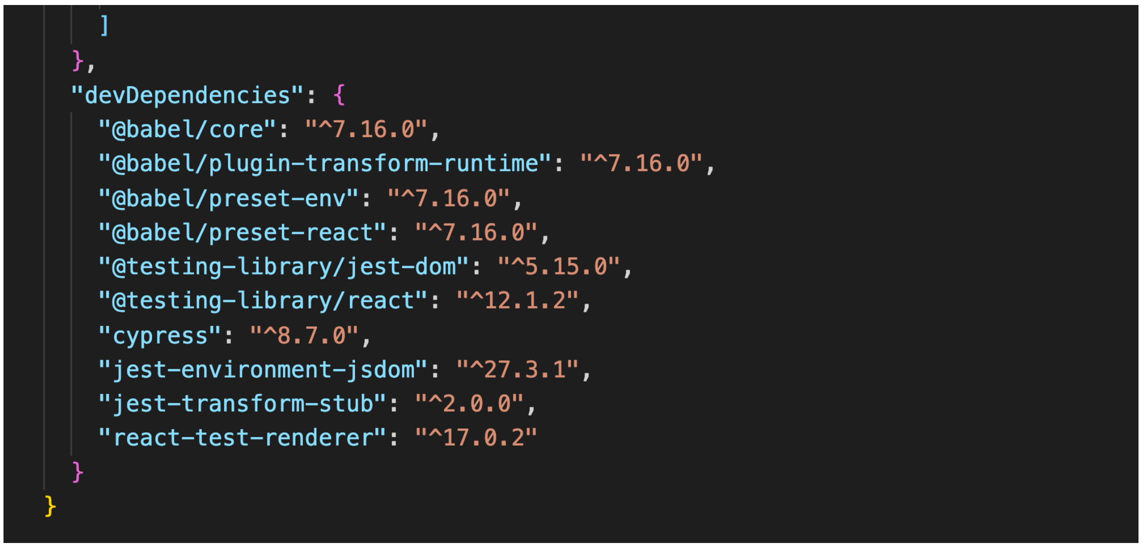
5.4.1 Implementing tests

a. Unit tests

Before executing tests, there is some dependencies need to be installed to support running the tests, and these packages will be placed in development mode. However, Jest is not necessary to install manually, since “create-react-app” command uses react-testing-library as the test runner, and Jest is already included in the react-script.

DevDependencies that are mandatory to run the tests are demonstrated in package.json in Figure 43 below.

These packages are installed with the command “npm install –save-dev *package_name*”



```
      ]
    },
    "devDependencies": {
      "@babel/core": "^7.16.0",
      "@babel/plugin-transform-runtime": "^7.16.0",
      "@babel/preset-env": "^7.16.0",
      "@babel/preset-react": "^7.16.0",
      "@testing-library/jest-dom": "^5.15.0",
      "@testing-library/react": "^12.1.2",
      "cypress": "^8.7.0",
      "jest-environment-jsdom": "^27.3.1",
      "jest-transform-stub": "^2.0.0",
      "react-test-renderer": "^17.0.2"
    }
  }
```

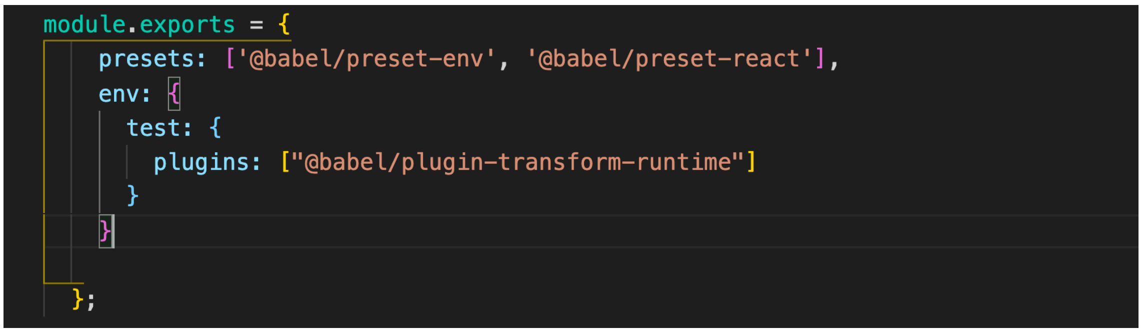
Figure 41. devDependencies installed.

After installing the required “devDependencies”, babel.config.js and jest.config.json are created in the root folder to set up the environment for the tests.

The Babel is a JavaScript compiler, and it is needed to convert code and transform syntax in browsers and target environments. Its presets, such as @babel/preset-env or @babel/preset-react, are the smart presets that allow the system to manage JavaScript bundles easier.

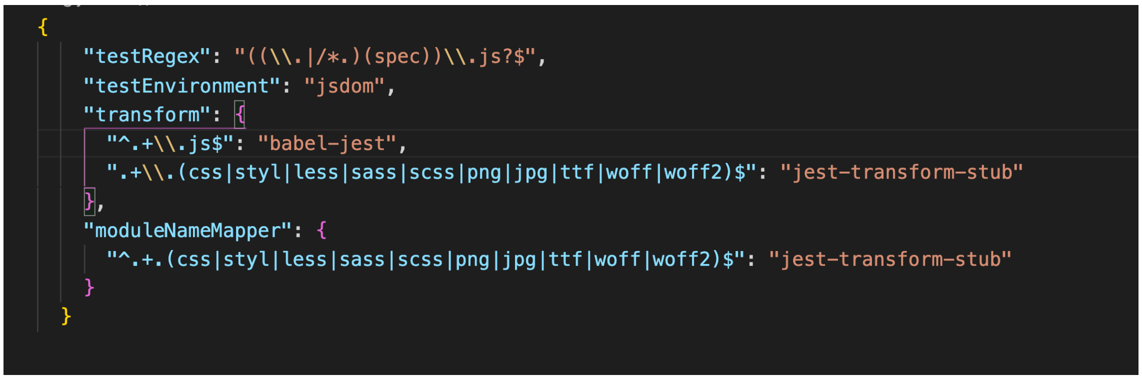
Jest.config.json file includes the commands that help to set up the Jest testing environment, such as transforming the image extension and SCSS styling extension to be readable for Jest when executing tests over the entire code.

The code and commands of these two files are displayed below in Figure 44 and Figure 45.



```
module.exports = {
    presets: ['@babel/preset-env', '@babel/preset-react'],
    env: {
      test: {
        plugins: ["@babel/plugin-transform-runtime"]
      }
    }

  };
```

Figure 42. babel.config.js desmontration.



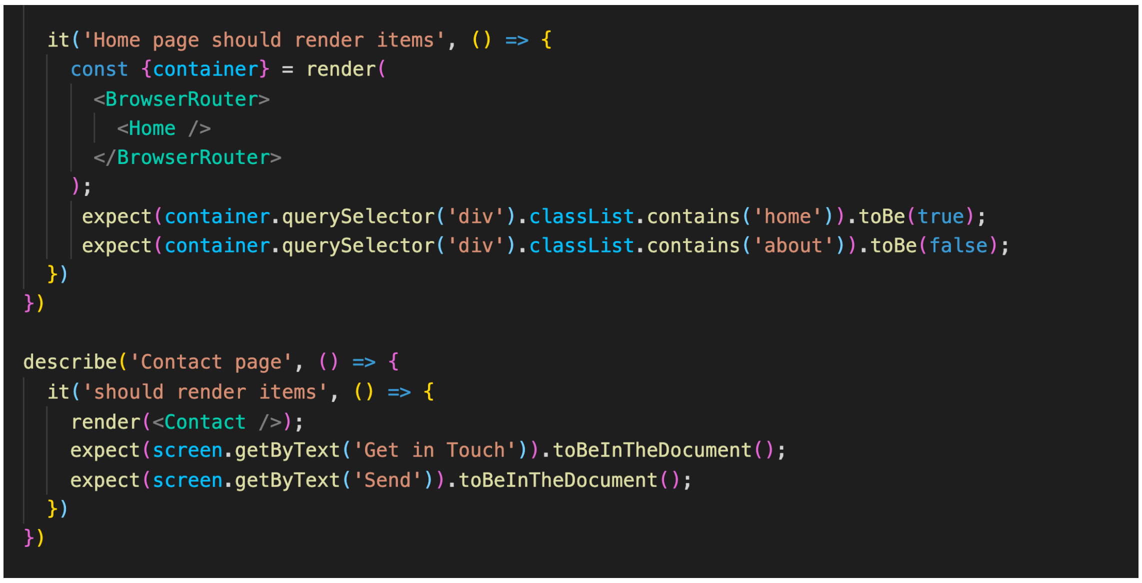
```
{
    "testRegex": "((\\.|/*.)(spec))\\.js?$",
    "testEnvironment": "jsdom",
    "transform": {
      "^.+\\.js$": "babel-jest",
      ".+\\.(css|styl|less|sass|scss|png|jpg|ttf|woff|woff2)$": "jest-transform-stub"
    },
    "moduleNameMapper": {
      "^.+.(css|styl|less|sass|scss|png|jpg|ttf|woff|woff2)$": "jest-transform-stub"
    }
  }
```

Figure 43. jest.config.json demonstration.

When the testing environment is ready, the next step is implementing tests. In this Unit test, components and routes are two main objectives that are examined. Tests folder is created in the src folder, it contains two test files including “Component.spec.js” and “Routes.spec.js”, and the test file extension should be in type “spec.js” or “test.js” to notify Jest performing these test files.

The main goal of component tests is to investigate whether the render of the components is processed as expected and contains all needed elements. The first test is rendering the Home component and checking if it has the element class “home”, as well as it should not contain the class “about” of About page. Similarly, the second test is rendering the Contact component, and examining if it concludes the text “Get in Touch” and “send”, which should be presented on the Contact page.

The source code of component tests is illustrated in Figure 46.

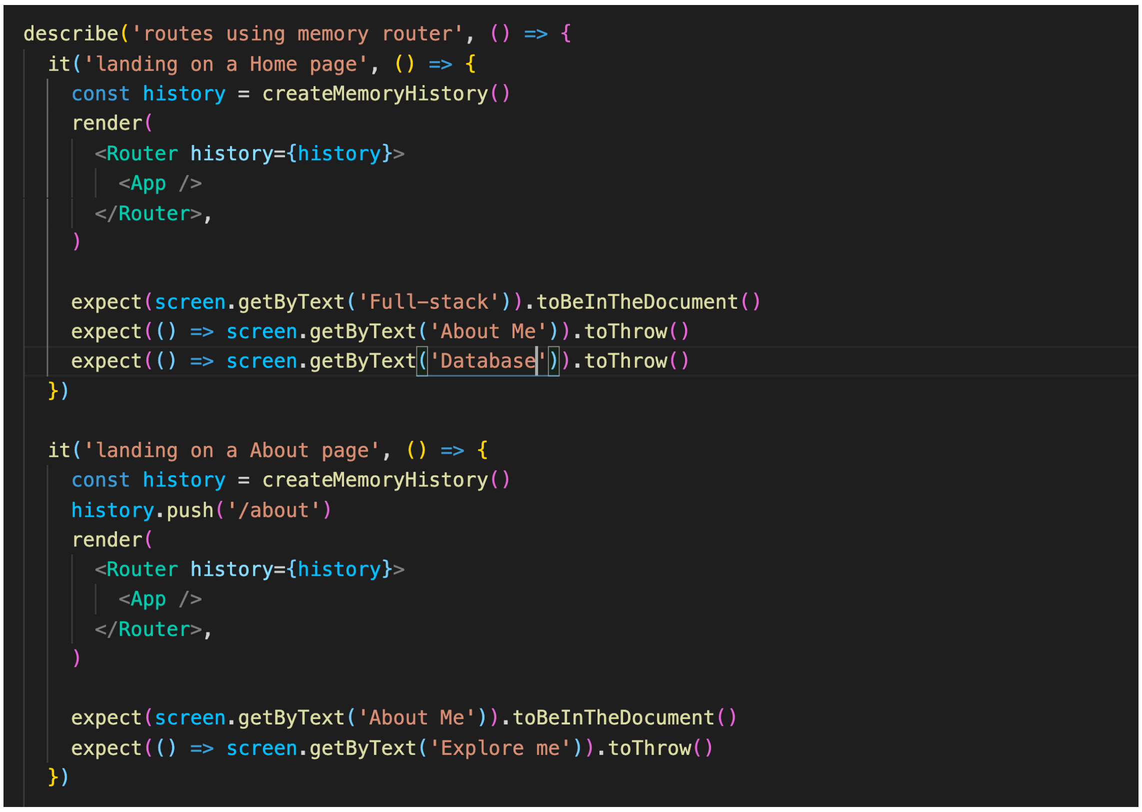


```
  it('Home page should render items', () => {
    const {container} = render(
      <BrowserRouter>
        <Home />
      </BrowserRouter>
    );
     expect(container.querySelector('div').classList.contains('home')).toBe(true);
     expect(container.querySelector('div').classList.contains('about')).toBe(false);
  })
})

describe('Contact page', () => {
  it('should render items', () => {
    render(<Contact />);
    expect(screen.getByText('Get in Touch')).toBeInTheDocument();
    expect(screen.getByText('Send')).toBeInTheDocument();
  })
})
```

Figure 44. Components test.

For the routes tests, the createMemoryHistory(), as well as the history of react-router-dom and history library is used to create variable “history” and generate the route in <Router>. By using “history.push(‘/about’)” command, the test will check if the appropriate component is rendered corresponding to the route. In this case, the About component should be rendered since the route is “/about”, if the component is presented correctly, the screen should contain the text “About Me” on the page, meanwhile, the text “Explore me” should not be on the page, since it belongs to Home page.

The source code of routes tests is demonstrated below in Figure 47.



```
describe('routes using memory router', () => {
  it('landing on a Home page', () => {
    const history = createMemoryHistory()
    render(
      <Router history={history}>
        <App />
      </Router>,
    )

    expect(screen.getByText('Full-stack')).toBeInTheDocument()
    expect(() => screen.getByText('About Me')).toThrow()
    expect(() => screen.getByText('Database')).toThrow()
  })

  it('landing on a About page', () => {
    const history = createMemoryHistory()
    history.push('/about')
    render(
      <Router history={history}>
        <App />
      </Router>,
    )

    expect(screen.getByText('About Me')).toBeInTheDocument()
    expect(() => screen.getByText('Explore me')).toThrow()
  })
```

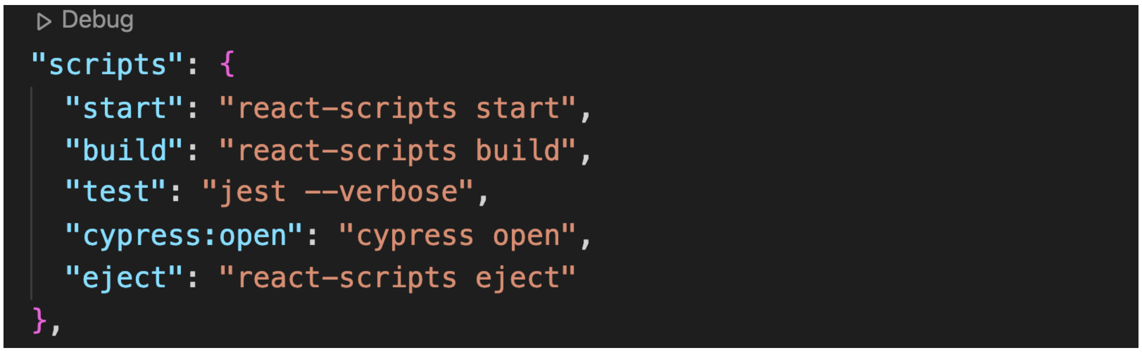
Figure 45. Routes tests.

b. E2E tests

In E2E tests, Cypress is chosen to execute the tests. Similar to unit tests, before implementing the tests, the required “devDependency” installed is Cypress. The command to install is “npm install –save-dev cypress”.

After Cypress is installed, the test script should be established in the package.json file to run the cypress test.

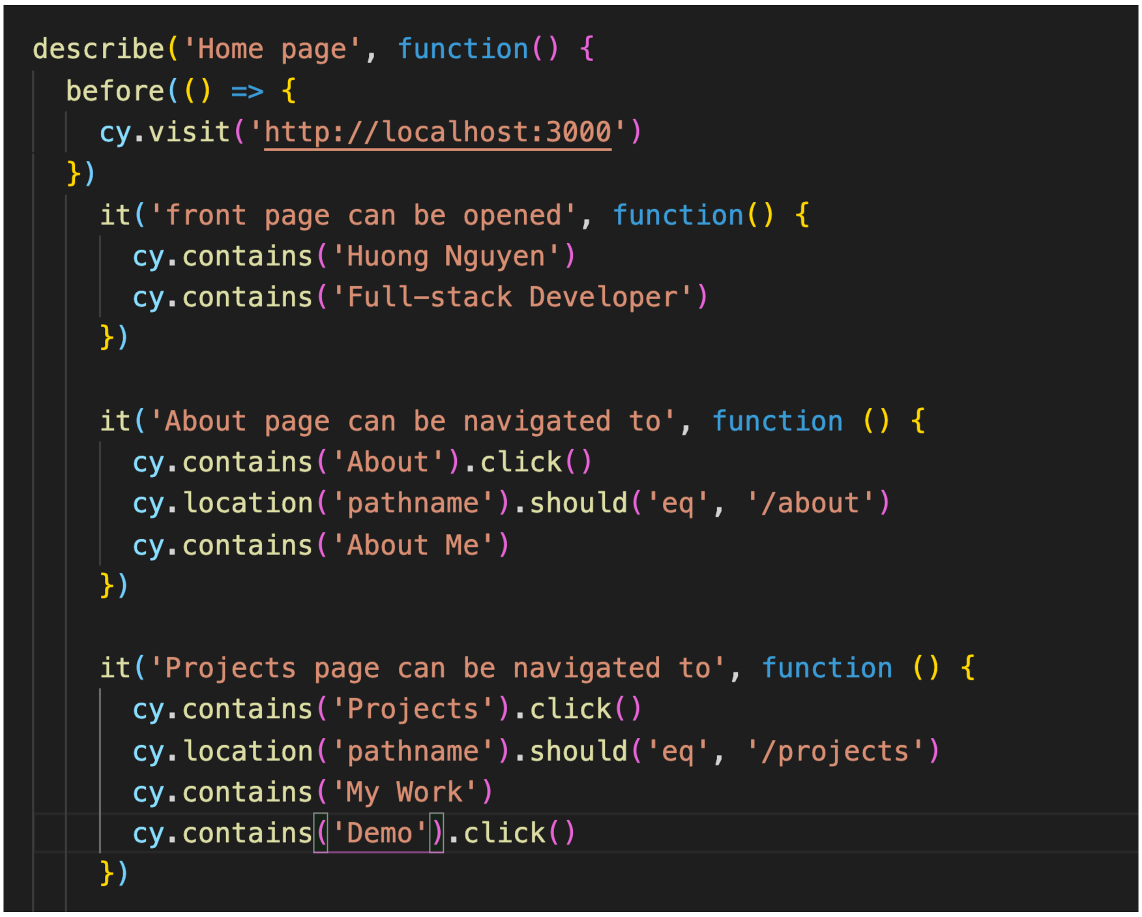
The script for Jest tests and Cypress tests are illustrated below (Figure 48).



```
▷ Debug
"scripts": {
  "start": "react-scripts start",
  "build": "react-scripts build",
  "test": "jest --verbose",
  "cypress:open": "cypress open",
  "eject": "react-scripts eject"
},
```

Figure 46. Test script.

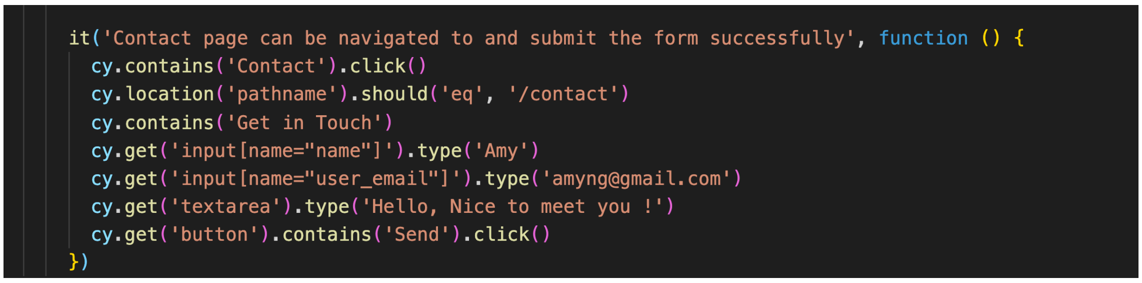
A Cypress folder is created in the root tree after running the cypress test script, the tests can be placed in the sub-folder “integration” of the “cypress” folder.



```
describe('Home page', function() {
  before(() => {
    cy.visit('http://localhost:3000')
  })
    it('front page can be opened', function() {
      cy.contains('Huong Nguyen')
      cy.contains('Full-stack Developer')
    })

    it('About page can be navigated to', function () {
      cy.contains('About').click()
      cy.location('pathname').should('eq', '/about')
      cy.contains('About Me')
    })

    it('Projects page can be navigated to', function () {
      cy.contains('Projects').click()
      cy.location('pathname').should('eq', '/projects')
      cy.contains('My Work')
      cy.contains('Demo').click()
    })
```

Figure 47. Cypress test.

In Figure 49, the tests describe the path when the visitor first accesses the application, then click to button on the navigation bar to navigate to the About page and Projects page. The aim of the test is to check if the page’s content is correct.



```
it('Contact page can be navigated to and submit the form successfully', function () {
  cy.contains('Contact').click()
  cy.location('pathname').should('eq', '/contact')
  cy.contains('Get in Touch')
  cy.get('input[name="name"]').type('Amy')
  cy.get('input[name="user_email"]').type('amyng@gmail.com')
  cy.get('textarea').type('Hello, Nice to meet you !')
  cy.get('button').contains('Send').click()
})
```

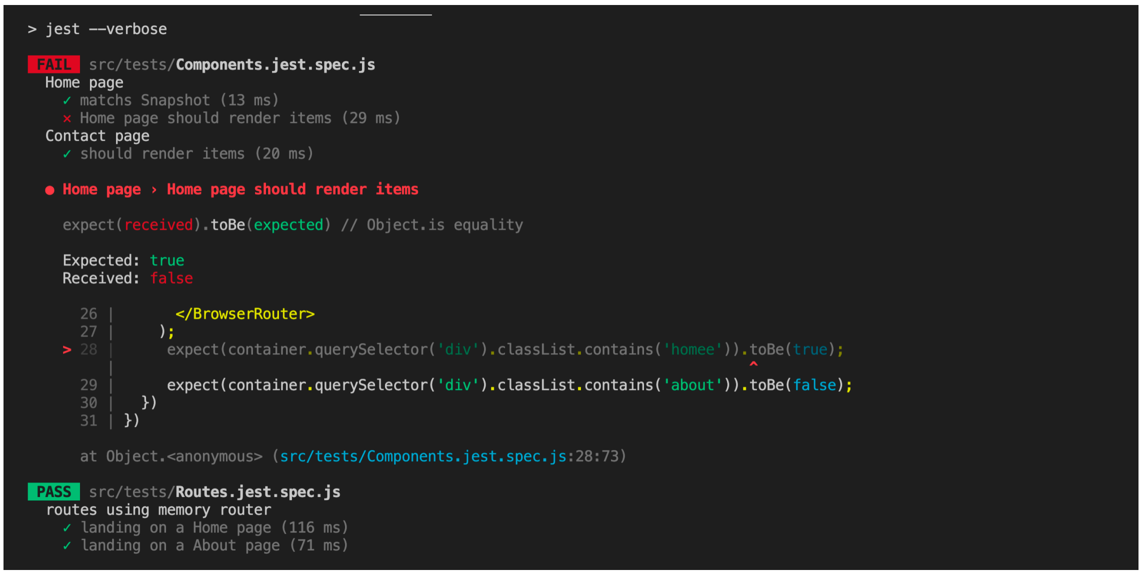
Figure 48. Form check Cypress test.

The test in Figure 50 is to check if the form on the Contact page work as intended. It follows the steps of the end-user go normally, including filling the form and clicking to submit the form.

All the above tests, such as unit tests and E2E tests, aim is to ensure the application works correctly and guarantee to bring the best user-friendly experience to visitors.

5.4.2 Result

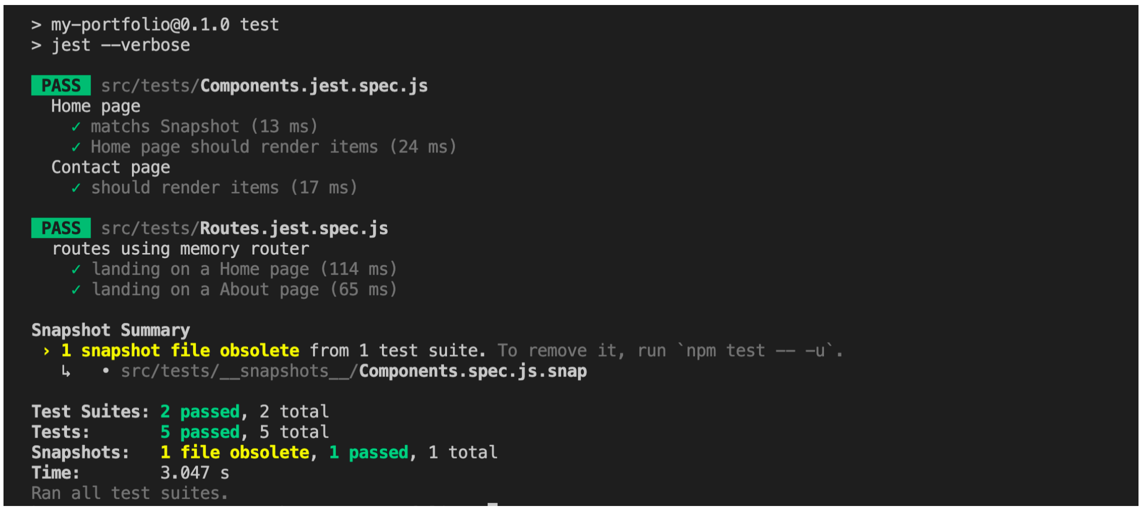
During the execution of the tests, most of the tests passed at the first time, moreover, there were a few steps that did not pass. The errors and their relevant issues were shown on the terminal to help developers figure out the bugs and fix them.



```
> jest --verbose

 FAIL  src/tests/Components.jest.spec.js
  Home page
    ✓ matchs Snapshot (13 ms)
    × Home page should render items (29 ms)
  Contact page
    ✓ should render items (20 ms)

  ● Home page › Home page should render items

    expect(received).toBe(expected) // Object.is equality

    Expected: true
    Received: false

      26 |         </BrowserRouter>
      27 |       );
    > 28 |       expect(container.querySelector('div').classList.contains('homee')).toBe(true);
         |                                                                          ^
      29 |       expect(container.querySelector('div').classList.contains('about')).toBe(false);
      30 |   })
      31 | })

      at Object.<anonymous> (src/tests/Components.jest.spec.js:28:73)

 PASS  src/tests/Routes.jest.spec.js
  routes using memory router
    ✓ landing on a Home page (116 ms)
    ✓ landing on a About page (71 ms)
```

Figure 49. Passed tests and the failed test by Jest.

Figure 51 illustrates an example of the result of passed and failed tests. It shows one of the tests in the “Components.jest.spec.js” file was failed, and the reason was displayed clearly on the screen. It expected the result which is “true”, however it received a “false” answer, therefore the test failed. Furthermore, it also indicates the line of the errors, by this way, developers specify the bugs area quickly and find the solution in a very shorter time.



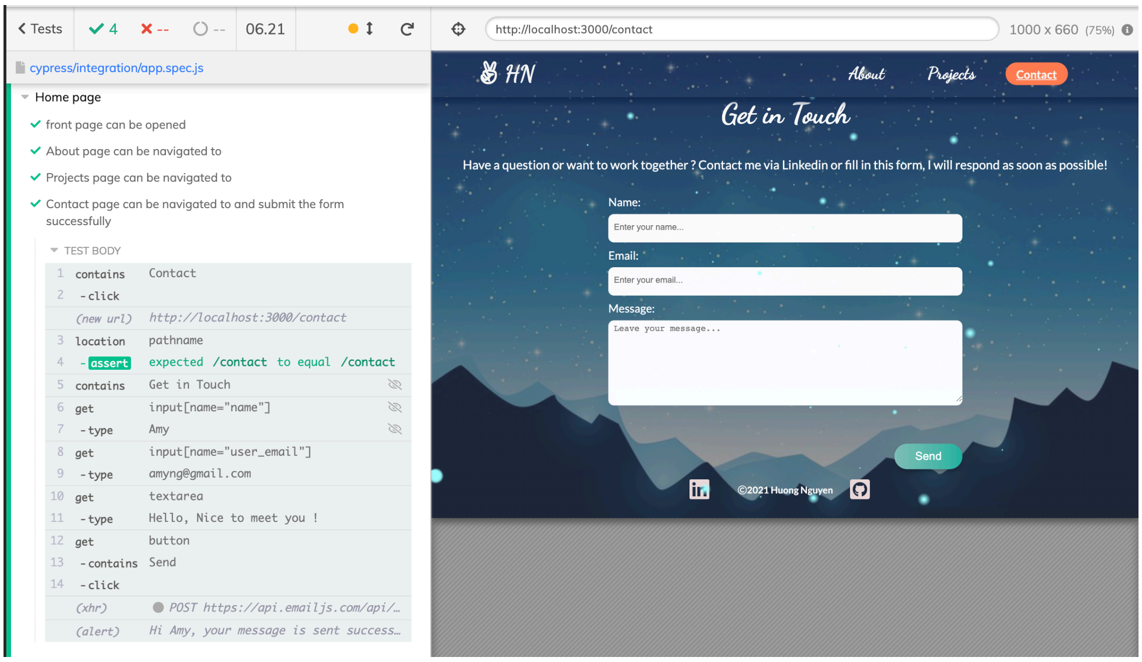
```
> my-portfolio@0.1.0 test
> jest --verbose

 PASS  src/tests/Components.jest.spec.js
  Home page
    ✓ matchs Snapshot (13 ms)
    ✓ Home page should render items (24 ms)
  Contact page
    ✓ should render items (17 ms)

 PASS  src/tests/Routes.jest.spec.js
  routes using memory router
    ✓ landing on a Home page (114 ms)
    ✓ landing on a About page (65 ms)

Snapshot Summary
 › 1 snapshot file obsolete from 1 test suite. To remove it, run `npm test -- -u`.
   ↳   • src/tests/__snapshots__/Components.spec.js.snap

Test Suites: 2 passed, 2 total
Tests:       5 passed, 5 total
Snapshots:   1 file obsolete, 1 passed, 1 total
Time:        3.047 s
Ran all test suites.
```

Figure 50. Passed Jest tests.

After fixing the issues and running the tests again, it achieves the passed results in all Jest tests, it is shown in Figure 52.

For the Cypress tests, after running the test script, it opens a Chrome browser to simulate the workflow of the end-user. All the test was displayed on the left, and the browser opened on the right. In every test, the stage is presented understandably to assist developers to keep track of the tests and easily detect issues if it exists.

It is illustrated obviously in Figure 53. All the Cypress tests passed, especially the form test, it shows the method POST and the alert to notify the form submitted successfully after finishing the last step of the testing flow, which is clicking the "Send" button.



cypress/integration/app.spec.js

Home page

- ✓ front page can be opened
- ✓ About page can be navigated to
- ✓ Projects page can be navigated to
- ✓ Contact page can be navigated to and submit the form successfully

TEST BODY

```
1  contains     Contact
2   -click
   (new url)    http://localhost:3000/contact
3  location     pathname
4   -assert     expected /contact to equal /contact
5  contains     Get in Touch
6  get          input[name="name"]
7   -type       Amy
8  get          input[name="user_email"]
9   -type       amyng@gmail.com
10 get          textarea
11  -type       Hello, Nice to meet you !
12 get          button
13  -contains   Send
14  -click
   (xhr)        ● POST https://api.emailjs.com/api/…
   (alert)      Hi Amy, your message is sent success…
```

Have a question or want to work together ? Contact me via Linkedin or fill in this form, I will respond as soon as possible!

Figure 51. Cypress tests.

Overall, there are different operating methods in testing frameworks, however, its general objective is to detect errors, examine functions and components over the source code. Depending on the programming language, the goal of testing, and operating method, developers need to choose the suitable testing frameworks for their application.

6 CONCLUSION

With the rapid development in technologies, the web industry become more dynamic and diverse as there are many frameworks and libraries released to help to build a web application with faster and more efficient performance. Standing out among them is React library with the Virtual DOM feature to boost the performance of an application.

This thesis focused on how to build a Single-page application with React and React Router. Furthermore, it also studied Front-end testing methods and the developed application was tested with Jest and Cypress to check the functionalities of each element and how the entire application works.

In the thesis, React was examined thoroughly, including its general definition, operating method. Furthermore, its advantages and disadvantages indicate that it is one of the best choices for Front-end developers and it is projected to hold a strong position in the marketplace. Additionally, an analysis of React development over time, its position in the marketplace for the time being, and a comparison between React and AngularJS were investigated to validate its reputation and unique functional characteristics.

Chapter 4 introduced the concept of Front-end testing indicated how important it is in the development workflow, and presented three types of testing methods and the current popular testing frameworks. It provided the definition of 3 main types of testing: unit tests, integration tests, and E2E tests. It gave a more specific view of testing to help developers choose the proper testing method based on the aim of testing.

Furthermore, the thesis also discussed the necessity of a Portfolio in job hunting and its various benefits for job seekers. The Portfolio was demonstrated clearly, and its testing segment was implemented and analyzed thoroughly by using Jest, React testing library, and Cypress.

REFERENCES

Arafa, A. (2021, 6 24). Retrieved 11 2021, from Front End Testing Types And Tools: https://dev.to/amiraarafa/front-end-testing-types-and-tools-4a0l

Chakma, J. (2021, 7 3). Retrieved 10 2021, from Benefits of Cypress Web testing tool: https://techbullion.com/7-benefits-of-cypress-web-automation-testing/

code. (2021). Retrieved 11 2021, from Using React in Visual Studio Code: https://code.visualstudio.com/docs/nodejs/reactjs-tutorial

Codecademy. (2021). Retrieved 10 2021, from React: The Virtual DOM: https://www.codecademy.com/articles/react-virtual-dom

cypress. (2021). Retrieved 11 2021, from cypress visit: https://docs.cypress.io/api/commands/visit#Syntax

Docs, M. W. (2022). Retrieved 01 2022, from mozilla: https://developer.mozilla.org/en-US/docs/Web/API/Document_Object_Model/Introduction

docs. (2021). Retrieved 10 2021, from What Is AngularJS?: https://docs.angularjs.org/guide/introduction

Hamedani, M. (2018, 12 3). Retrieved 10 2021, from React Virtual DOM Explained in Simple English: https://programmingwithmosh.com/react/react-virtual-dom-explained/

Insignares, A. (2021, 3 10). Retrieved 10 2021, from What are the Cons of React?: https://www.koombea.com/blog/react-pros-and-cons-what-are-the-advantages-and-disadvantages-of-reactjs/

javapoint. (2021 a). Retrieved 10 2021, from Pros and Cons of ReactJS: https://www.javatpoint.com/pros-and-cons-of-react javatpoint. (2021 b). Retrieved 11 2021, from Integration testing: https://www.javatpoint.com/integration-testing

javatpoint. (2021 c). Retrieved 10 2021, from React Router: https://www.javatpoint.com/react-router

javatpoint. (2021 d). Retrieved 11 2021, from Unit Testing: https://www.javatpoint.com/unit-testing

jestjs. (2021 a). Retrieved 10 2021, from Using Matchers: https://jestjs.io/docs/using-matchers

jestjs. (2021 b). Retrieved 10 2021, from philosophy: https://jestjs.io/

katalon. (2019, 6 10). Retrieved 11 2021, from Benefits of End-to-End Testing: https://www.katalon.com/resources-center/blog/end-to-end-e2e-testing/

Khirale, A. (2018, 4 2). Retrieved 10 2021, from Comparison Between AngularJS VS React in 2018: https://dev.to/amitkhirale/comparison-between-angularjs-vs-react-in-2018-1i84

Lin, P. (2019, 12 2). Retrieved 10 2021, from Types of React Components: https://levelup.gitconnected.com/types-of-react-components-a38ce18e35ab

Lindley, C. (2018). Retrieved 10 2021, from What is a Front-end Developer?: https://frontendmasters.com/guides/front-end-handbook/2018/what-is-a-FD.html

Nassi, T. (16. 2 2021). Retrieved 1 2022, from split: https://www.split.io/blog/react-router-feature-flags/

nodejs. (2021). Retrieved 11 2021, from NodeJS Download: https://nodejs.org/en/download/

Pandit, N. (2021). Retrieved 10 2021, from What is React.js?: https://www.c-sharpcorner.com/article/what-and-why-reactjs/

pp_pankaj. (2019, 4 30). Retrieved 11 2021, from Unit Testing | Software Testing: https://www.geeksforgeeks.org/unit-testing-software-testing/

reactjs. (2021 a). Retrieved 11 2021, from Create a New React App: https://reactjs.org/docs/create-a-new-react-app.html

reactjs. (2021 b). Retrieved 10 2021, from Reactjs: https://reactjs.org/

reactrouter. (2021 a). Retrieved 10 2021, from BrowserRouter: https://reactrouter.com/web/api/BrowserRouter

reactrouter. (2021 b). Retrieved 10 2021, from HashRouter: https://reactrouter.com/web/api/HashRouter

reactrouter. (2021 c). Retrieved 10 2021, from Route: https://reactrouter.com/web/api/Route

reactrouter. (2021 d). Retrieved 10 2021, from Switch: https://reactrouter.com/web/api/Switch

Recruiter, U. (2021). Retrieved 11 2021, from 5 Reasons Why All Jobseekers Need to Have a Work Portfolio: https://theundercoverrecruiter.com/5-reasons-why-all-job-seekers-need-have-work-portfolio/

rishabhsoft. (2019, 2 26). Retrieved 10 2021, from AngularJS Vs ReactJS: Know Which is the Best for Your Project: https://www.rishabhsoft.com/blog/reactjs-vs-angularjs

sass-lang. (2021 a). Retrieved 10 2021, from documentation: https://sass-lang.com/documentation

sass-lang. (2021 b). Retrieved 10 2021, from syntax: https://sass-lang.com/documentation/syntax

Shaleynikov, A. (2018, 8 31). Retrieved 10 2021, from Evolution of React on a Timeline: https://dzone.com/articles/evolution-of-react-on-a-timeline

Shruti. (2021, 10 21). Retrieved 10 2021, from Reactjs Router: https://www.geeksforgeeks.org/reactjs-router/

smartbear. (2021). Retrieved 11 2021, from What Is End-To-End Testing?: https://smartbear.com/solutions/end-to-end-testing/

statista. (2021). Retrieved 10 2021, from Most used web frameworks among developers worldwide, as of 2021: https://www.statista.com/statistics/1124699/worldwide-developer-survey-most-used-frameworks-web/

Stefanuk, A. (2021). Retrieved 10 2021, from React JS Developer Salary Comparison in Different Countries: https://mobilunity.com/blog/react-js-developer-salary-in-different-countries/

Suthwal, D. (2020, 11 30). Retrieved 10 2021, from React Virtual DOM vs Real DOM: https://medium.com/devinder/react-virtual-dom-vs-real-dom-23749ff7adc9

testim. (27. 8 2021). Retrieved 10 2021, from Jest Testing: A Helpful, Introductory Tutorial: https://www.testim.io/blog/jest-testing-a-helpful-introductory-tutorial/

Wiki. (2021). Retrieved 10 2021, from React (JavaScript library): https://en.wikipedia.org/wiki/React_(JavaScript_library)

Z, A. (25. 06 2018). *rubygarage*. Retrieved 01 2022, from https://rubygarage.org/blog/single-page-app-vs-multi-page-app#article_title_1