Surendra Pandey

Build client-side web applications of Shift Manager using React and Redux

Helsinki Metropolia University of Applied Sciences

Bachelor’s Degree

Information and Communication Technology

Innovation Project

07/03/2019

Authors Title Number of pages Date	Surendra Pandey Build client-side web applications of Shift Manager using React and Redux 31 07/03/2019
Type of Project	Bachelor’s Thesis
Degree Programme	ICT
Instructors	Janne Salonen, Head of School (ICT) Sonja Holappa, Language Supervisor

The thesis focused on the creation of client-side web applications for Shift Manager. Shift Manager is a tool created for supervisors to help them in creation and publication of work shifts and to eliminate the human errors encountered during hours calculations. Focusing on the creation of the whole solution seemed to be a broad scope so, this thesis was focused only on the creation of the client-side of the solution.

As the outcome of the thesis two web applications, Workplace App and User App, were created using React and Redux. Supervisors can use Workplace App to generate employees’ work shifts and publish them to their employees by a couple of button clicks. Then employees can see their published schedules by easily logging in to the User App. The shift hours are calculated automatically leaving no room for human errors.

The built product was then tested by the case company, a chain restaurant in Helsinki, implementing it in two different branches; they were satisfied with the services provided. The work in the product will be continued in the future as there are improvements needed to be done. Although the solution was built visualizing the problems faced in the case company, this solution is built in such a way that it can be used in any other company implementing a shift-based work system.

Keywords	React, Redux, JavaScript, Shift Manager

Contents

Abbreviations

PHP	Personal Home Page
ASP	Active Server Pages
HTML	HyperText Markup Language
CSS	Cascading Style Sheets
ERP	Enterprise Resource Planning
API	Application Programming Interface
KPI	Key Performance Indicator
IDE	Integrated Development Environment
DOM	Document Object Model
PaaS	Platform as a Service
SaaS	Software as a Service
IaaS	Infrastructure as a Service
IDE	Integrated Development Environment

1 Introduction

When we look at the current business world, we can see that technology is on the hype. Every business relies on technology to boost performance and generate greater income. For instance, super markets are using self service cashiers where customers interact with software with audio directions for buying goods. Use of self-service cashiers positively impacts on the revenue of the company by cutting off the number of workers required on the daily basis. Although few businesses are using the latest technologies and performing in their best possible way, there are many businesses which are still lagging decades behind. If we look in the Human Resource sector, many business running hour-based shifts systems are still generating the shifts in google sheets and distributing the shifts in papers. Although shifts can be generated in google sheets, it is extremely difficult to manage and becomes time consuming when the number of employees grows. Since time is money, businesses are losing money in this process of shift generation.

The study was conducted visualizing the difficulties faced by the supervisors and employees of Ravintotalot Oy. Ravintotalot Oy is a medium scale restaurant chain company having 15 branches located in the Helsinki and Espoo region. The company has around 200 employees all together. Since the company is implementing a shift-based work system, the supervisor of each branch must dedicate many hours in shift creation and shift publication. As they are creating the shifts in google sheets, it has been very difficult for them to manage, create and publish the shift data. In addition, since the total hours are calculated manually, most of the time the salary paid to the employees does not match the actual hours worked by the employees. Although the study is based on the issues faced by the employees of Ravitotalot Oy, the application can be used by any company implementing the shift-based work system.

The goal of the study was to make the creation of the work shifts more manageable, eradication of the human errors during hours calculation and shift publication very easily. The objective of the study was to create a cloud-based web application where supervisors could easily create the work shifts, the hours are calculated automatically, and the shifts are published to the employees with a couple of button clicks.

Since the scope of this study is very broad, this report is going to be about the creation of the front-end applications required to accomplish the goals of the study. The report is divided into seven sections. First section is about the general introduction of the study. Second section walks through the background information about the technologies used to accomplish the goals of the project. Third section is about the overview of the solution going to be built and the set-up steps of the application from the customer's point of view.

The fourth section is about the development of the applications. It talks about the data flow process, the environment set up required for the development of the applications, and the folder structure of the applications. The fifth section talks about the outcome of the study explaining about the views in the applications where the actual issues are addressed. The sixth section provides a glimpse to the future of the project. Finally, the study is concluded in the sixth section by re-addressing the issues and providing the solution to those issues. In addition, this section talks about the benefits seen during the development process due to the use of React and Redux.

2 Background

2.1 Web Application

Web Application is a client-server computer program that performs certain tasks by running on web browsers. The complexity of the web application varies depending on the functionality of the program. For example, message board application, todo application, etc. are among the simplest web applications whereas online games, online trading platforms, etc. are among the most complex ones.[1]

Since, the Internet is profitable or cost effective for communication purposes, and web applications utilize the internet for the exchange of data, almost all the businesses use web applications to communicate with their target customers.[2]

Web applications are created with the help of server-side and client-side programming languages. PHP, ASP and Node.js are few examples of server-side languages whereas JavaScript, Html and CSS are examples of client-side languages. Server-side languages are used to store data, provided from client-side, to the database and Client-side languages are used to collect data from the users using online forms, shopping carts, etc.[2]

As the web applications reside in some servers on the internet and they can be reached through web browsers, web applications are platform independent. Meaning, such applications are available to everyone having the internet connected devices compatible with running web browsers. Chrome, Firefox, Explorer, Edge, etc are some of the commonly used web browsers. In addition, web applications are not needed to be installed in any devices before using them. Thus, they do not need a huge amount of compute power and storage capacity to run.[2]

2.2 Progressive Web App

Progressive Web Apps (PWAs) are a kind of web application built keeping user experience in mind. In other words, they are created to provide native like experience while using a web application. Users can even add a shortcut icon in their home screen like in native applications.[3]

They are created using normal web programming languages like HTML, CSS and JavaScript. In addition, manifest file and service workers play vital roles in making web applications PWAs. Manifest file is related more with appearance of the application whereas service workers are related with the communication purpose of the application and the platform. For instance, the Home screen icon, full screen display and the decision either to display the browser is served with the help of a manifest file.
On the other hand, features like push notifications are served by Service workers.[3]

Progressive web applications provide native like experience without any need to be installed locally before using. This eliminates the need of high performing devices with higher storage capacity. As they are deployed in some servers on the internet and are served through links, PWAs are highly reachable. Moreover, users can access the features offline and receive push notifications like in native applications. When looked at from a developer's perspective, the major advantage of progressive web applications over native applications is, PWAs can run on any platform with a single code base. There is no need for separate applications for Android and IOS.[3]

Thus, progressive web applications (PWAs) are web applications created using web programming languages and provide experience like native applications. When the required conditions are met, the manifest file adds a shortcut icon on the home screen. No matter the internet quality, the application is loaded without any delay when launched using a shortcut icon.

2.3 Cloud Computing

In the past, the software development process was extremely expensive due to the huge upfront cost of the servers needed to be set up for the software deployment. The first thing software engineers had to do was to predict the number of users of the application and buy the servers needed. This step being expensive and time consuming, the concept of cloud computing was introduced. The general concept of cloud computing means providing computing services such as servers, storage capacity, databases, networking and many more over the internet. Amazon Web Services, Microsoft Azure, Google Cloud, etc are some of the top companies providing cloud computing services.

Generally, cloud computing services are categorized as Infrastructure as a Service (IaaS), Platform as a Service (PaaS), Software as a Service (SaaS) and Serverless computing. When full control over the server is needed, IaaS service is recommended. In this case, vendors provide maintained hardware components over the internet however, the actual setup of the environment is left to the buyer. When an environment is needed to develop, test, deliver and manage software applications, PaaS is recommended. In this case, cloud companies will maintain the hardware components and will set up and update the environment whenever needed. In SaaS, actual software is provided over the internet to the user. In this case, hardware components, server environment and the software are maintained by the cloud computing vendors. Anyone having an internet connected device can subscribe to the services. Serverless computing services are extended over PaaS services. These services are used to build app functionalities without any need to manage servers and the environment. Moreover, the services in this category run only when an event is triggered by any means and gets shut down after the task is completed.[4]

These services are cost efficient, reliable, and highly secured. Since, the cloud vendors have huge data centres at different locations globally and their data centres can be rented whenever needed, the huge upfront cost of buying the servers has been eliminated. The cost would be for only the servers that have been used. Moreover, any number of servers can be run instantly, hence the performance of the application is not affected even when the number of users increases. As the data centres are set up and maintained by professionals, the services provided are highly secured.[4]

2.4 Dashboard

Dashboard is a digital interface in software where data is collected, analysed and displayed from multiple raw sources like files, attachments, services and APIs in a format that is easy to visualize. Since, the data displayed in the dashboard could range from simplest to most complex, the data is displayed in the form of tables, line charts, bar charts and gauges. Moreover, data centralization being a key feature, dashboard helps businesses to keep track of their performance from a single location.[5]

However, all businesses do not have dashboards serving the same purpose. Businesses need different data for different purposes; hence, they collect a huge amount of complex data which could be both related and unrelated to each other. So, depending on the needs of the business, dashboards can be divided into following three categories: Operational Dashboard, Strategic Dashboard and Analytical Dashboard.

Operational dashboard is designed to track the data which changes frequently, and the updated information is checked multiple times a day. One of the examples of operational dashboards is a Daily Web Overview Dashboard of Google Analytics where web performance is tracked hourly against specified objectives. The collected data is later used by the digital marketing team for various purposes. [6]

Strategic Dashboard is designed to track the status of Key Performance Indicators (KPI). Such a dashboard is used by business executives to check their performance in business. As the data tracked by the strategic dashboard is changed less frequently compared to the Operational Dashboard, it is viewed less frequently.[6]

Analytical Dashboard is designed to investigate trends, predict outcomes and discover insights by analysing large volumes of data. Such a dashboard is commonly found in business intelligence tools. The reason behind analytical dashboards appearing in Business intelligence tools is that they are developed by data analysts.[6]

One of the use cases of dashboard could be businesses keeping track of their sales data. They must analyse the sales data every month to know their sales performance. Analysing the data manually is a very difficult and time-consuming job for businesses and this is where dashboard applications come very handy. The business world is full of such scenarios where the dashboard applications can be used.

3 Shift Manager

3.1 Background

Shift Manager is a tool developed for supervisors of companies implementing the shift-based work system. It is a cloud-based web application which allows supervisors to generate work schedules for employees and helps to deliver schedules to the respective employees seamlessly. Also, the application notifies employees about newly created shifts instantly which ensures that the employee has received the new schedules.

When a shift is created using excel sheets or any other software meant for some other purposes, supervisors need to calculate the shift hours for individual employees manually. This increases the risk of human error in hours calculation which ultimately leads to the loss of money. Shift Manager completely eradicates this problem since the shift hours are automatically calculated by the application itself. In addition, when the shifts are distributed in papers, risk of losing the shifts increases and the employee might not appear at work in the specified time. This risk also disappears as the shift is always available in the employee's smartphone.

Since, Shift Manager is not the first product that has solved the problems, there exists many other ERP software that can do the same job. They also contain varieties of other features that might be useful for companies in varieties of scenarios. However, the existence of so many features adds complicacy and hampers the usability of the software. Also, it is extremely expensive for small and even for medium scale companies to buy the whole software. In addition to the price of the actual software, the training required for the employees to use the software adds up some extra cost.

In contrast to the full-fledged ERP software's usability, Shift Manager is extremely easy to use since many features are not cluttered in single software. It is built keeping simplicity in mind and focusing just on creating and communicating employee schedules. In addition to the usability, the price of the application is also minimum compared to the ERP software, so it is affordable for companies of any scale.
Shift Manager provides its services in the SAAS model. Companies can subscribe to the services on a monthly basis with a free trial period of at least one month. Since, there is a trial period, the software could be tested without paying any money. Furthermore, if due to any reason, companies dislike the product after using it for a certain time interval, the services could be unsubscribed whenever intended.

3.2 Application setup from customers' perspective

From the customers' perspective, Shift Manager consists of two different applications, Workplace App and User App. Workplace App is created for workplaces and is operated by the supervisors assigned to create schedules. On the other hand, the User App is created for employees to check work schedules assigned to them. Both the applications are accessible from web browsers with the help of HTTP URL or Link. However, there are two separate links to access individual applications.

Since, there are two different applications for workplaces and employees, there are separate signup and login portals for each application. Initially, everyone should get registered to the application using the correct portal. Meaning, Workplace should be registered using workplace application's signup portal and Employees should be registered using the user application's signup portal.

After the registration is completed, everyone should be able to login to the application using the right login portal with correct credentials. However, at this point neither supervisors could create shifts for employees nor employees could receive any schedules since there are no employees added to the newly created workplace. Even before adding any employee, roles should be created in the workplace so that employees could be assigned to certain roles. Therefore, after the completion of the registration process, supervisors need to create roles in the workplace and then add the registered employees to the workplace. After this step, the workplace setup is complete, and the application is ready to create and publish the employees' schedules.

4 Development Tools, Technologies and Approach

When looking from the development perspective, architecture of the application seems to be the first and foremost thing needed to be explained since it gives the overall glimpse of the application. Being a web application, Shift Manager is based on client server architecture which is also known as multi-layered architecture or tier architecture. In multi-layered architecture, each layer has a separate responsibility. An example of most common multi-layered architecture is a 3 layered or 3-tier architecture.

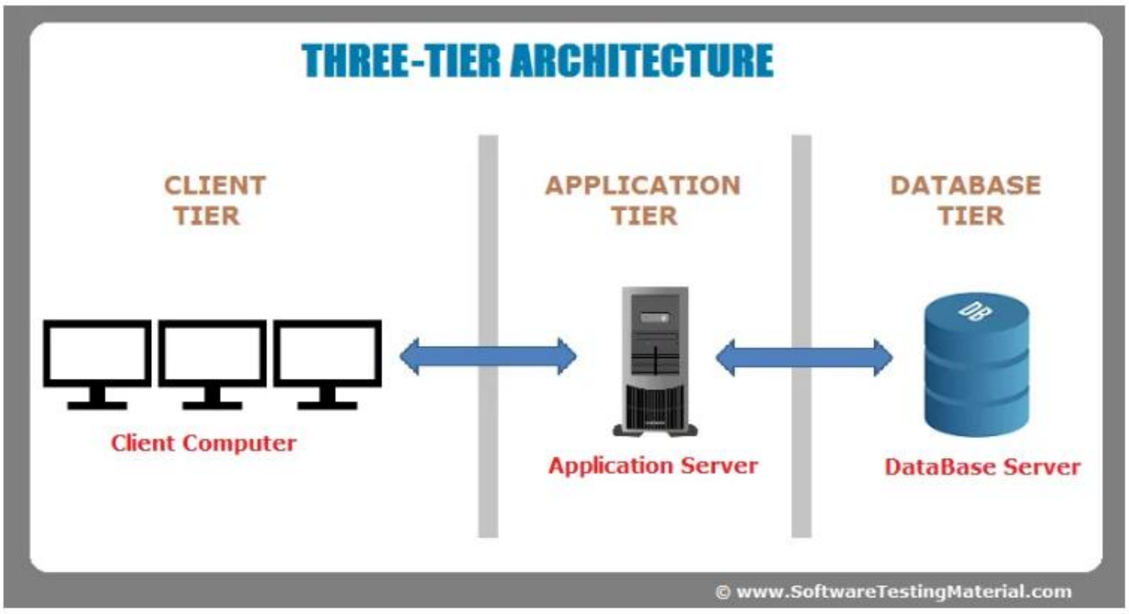


Fig: 1 Three-Tier Architecture [7]

As displayed in the Fig: 1, the 3-layer architecture consists of a data layer, application layer and client layer. Data layer is responsible for running the database server from where the database application is hosted, whereas the Application layer is responsible for running the application server from where the backend application is hosted. Similarly, the Client layer is responsible for running servers which receive requests from the client computers and serve client-side applications.

Shift Manager being a web application has a 3 layered architecture. It consists of a Postgresql database in the data layer, a Node.js application in the application layer and two React.js applications in the client layer. To make the services available to the customers, Shift Manager is hosted on the servers provided by Digital Ocean (https://www.digitalocean.com/).

Database is responsible for storing and retrieving the data based on the request made to it. Whereas, a backend application is responsible for analysing the request, performing the action based on the request and responding with the success or failure status received while performing the action. Finally, frontend applications are responsible for collecting and validating data from the users and requesting for some action to the backend application based on the collected data. When the response from the backend application is received, frontend applications display the updated data to the users.

Since, there are 3 different applications and a database, talking about the development process of all the applications would make the scope of the report huge. Thus, to narrow down the scope, only the development process of client-side applications is going to be discussed.

4.1 Tools used in client-side Applications

Development of client-side applications mainly includes designing User Interfaces and writing code to create the designed Interfaces. Making design of User Interfaces can be considered as the easy part since it only shows the appearance of the application. There are many tools available which makes it very easy to create and manage the design. However, when it comes to the implementation of the design, there are thousands of lines of code which makes it very difficult to manage if a normal text editor is used. So, proper tools are required to add, update, track and store the written code.

The list of tools that are used in Shift Manager to design User Interfaces and to write and manage code are: FIGMA, Visual Studio Code, Git and Chrome Dev-Tools.

4.1.1 FIGMA

As mentioned on the official website of FIGMA, it is a cloud-based user interface design and prototyping tool. Being a web application, it runs in a browser and does not need to be installed in individual devices. It is very easy to use, collaborate and is free of cost when used by a single person. However, to share the same account between multiple users, it needs to be subscribed. Also, it has a powerful feature which is multiplayer editing. Since, FIGMA is easy to learn and use, free of cost and has multiplayer editing features, it is used to design user interfaces for Shift Manager.

4.1.2 Visual Studio Code

Visual Studio Code is a tool used to add and edit code. It is an IDE (Integrated Development Environment) created by Microsoft and is compatible in Linux, Windows and macOS.[9] Like other IDEs, it contains a text editor, a file tracking system which makes it easy to divide and manage code in different modules, a CLI (Command Line Interface) which is very useful for developers in varieties of scenarios and many other useful features. Some of the features worth mentioning are built in Git and Extensions. Git is a version control system and is extremely useful during software development. There is a separate section talking about Git in detail below. Apart from built in features, Extensions provide the possibility to add additional features when needed. For instance, Visual Studio Code natively supports JavaScript code however, it is possible to add an extension to make it support other languages like Python.

4.1.3 Git as Version Control System

As mentioned earlier, Git is a version control system which helps developers to keep track of the changes made in their source code in a well-managed way. Software development is a lengthy process and involves multiple developers working on the same source code every day. Since changing a source code provides room for bugs, if the official source code is changed directly by every developer, many of the features might break. This helps to imagine how inefficient it could be to code without a version control system.

Version control system helps developers by giving the ability to clone the source code, create branches to work on different features without affecting the source code and finally merge the branch to the source code when the feature is ready. This way multiple developers could collaborate in a huge project without breaking each-other's work. Also, since all the code is tracked by the version control system, it is very easy to either change or remove any features when needed.[11]

Talking more about Git, being a distributed version control system, it maintains the whole codebase including the history for every developer on their computer. Since the source codebase is available on every developers' local machine, it is very easy for developers to create branches, make changes to the code and merge the changes.[11]

Git maintains the code distribution with the help of Git Bash and GitHub. Git Bash is a command line interface created for developers to keep track of their codebase. GitHub is a tool which provides repository hosting service. Using this service, a team of developers can easily collaborate, and version control their software. The user-friendly interface of GitHub helps everyone including the very beginners to take advantage of Git without using the command line. Even more fascinating is the free service to host public repositories.[11]

4.1.4 Chrome Dev-Tools

Google Chrome contains a collection of built in tools, useful for developers during the development process of any web application, known as Chrome Dev-Tools. These tools come handy while debugging the problems in layouts, debugging the JavaScript errors with the help of JavaScript console and tracking other meta information related to the web application.[12]

The layout of the web application is generated by HTML code and the styles are given by CSS code. If any problem is seen in the layout, by navigating to the elements tab of Dev-Tools, the code can be changed on the fly and can be fixed from the browser itself. However, the code is not saved so the changes must be copied before refreshing or closing the browser.[12]

JavaScript is used to add the functionality to the interactive layouts of the application and while making the application functional many errors are faced. These errors appear on the JavaScript console and contain the details about the errors. Also, the variables and the functions defined in the code while building the application can be accessed from the console. The availability of code in the console eases up the pain faced during the debugging process.[12]

Finally, the meta data related to the web application can be seen on the Network tab. Network tab keeps track of the network requests made to the server and displays the information about any request.[12] Since the performance of the application is dependent on the status of the network requests, requests that take a long time to resolve impacts negatively. When there is degrade in performance it is easy to track and solve the problem with the help of the information provided by the Network tab.

4.2 Technologies

For the development of any client-side application, HTML (Hypertext Mark-up Language), CSS (Cascading Style Sheet) and JavaScript are used. Browsers understand only these languages; however, as technologies are evolving at a rapid pace, there are many libraries and frameworks built on top of these technologies which helps in making the development process easier and faster. Some of the most popular libraries and frameworks are React, Redux, Angular, Vue, jQuery, etc.

Client-side applications of Shift Manager are built using React, Redux, SASS, and AXIOS. These technologies are discussed below in detail.

4.2.1 React

React is a frontend JavaScript library created by Facebook to create responsive and reusable UIs. It is one of the most popular libraries existing in the present context. Since, react is a library not a framework, it is not a complete solution for the creation of full frontend applications. Since it is used to create User interfaces only, other libraries are needed to manage the application state. For instance, either Redux or Flux can be used as state containers with React. However, these are not the only libraries that exist. The main reasons behind React's increasing popularity are – designing UI as stateful components which are composable and reusable, *the nature of reactive updates* and the virtual DOM (Document Object Model).[13]

Components are the foundation of User Interfaces in react. It is easy to visualise components when thinking about functions in any programming languages. Functions contain reusable blocks of code which perform some action on the inputs and return the processed output. A function to add two numbers and return sum is the most common example. This function can be now used wherever needed. In addition, this function could be used to compose another function which returns the average of two numbers. Similarly, the inputs in components are represented by "properties" and "state" and the output is an UI displaying the data contained in its properties and state. As functions, bigger components can be composed with the help of smaller components with individual responsibility. In addition, components can be used in multiple places in multiple projects.[13]

The state is the source of data represented in the user interface returned by a component. Whenever the data in the state changes, the react components automatically updates the data displayed in the user interface. This way the user interface is always representing the right data and the developer does not need to think about how and when to update the user interface. The only thing needed to be taken care of is the supply of data to the state.

Virtual DOM (Document Object Model) is an in-memory HTML tree created using JavaScript. React separates the real DOM from a developer with the help of this virtual DOM which is more user friendly. It only updates the real DOM whenever the state of the virtual DOM changes. In addition, react updates only the part of the HTML tree or real DOM where the data has been changed. Due to this reason, applications built using React are better in performance compared to applications built on traditional methods.[13]

The above-mentioned reasons make developers easy to choose React for developing fast and efficient applications that involve handling huge amounts of data. Also, because of the same reasons, react has been used to build the user interfaces in Shift Manager.

4.2.2 Redux

Redux is a state management library used to build consistently behaving JavaScript applications. Redux can be used either with plain JavaScript or with other UI libraries or frameworks like Angular, React, Vue, etc.

With the increase in functionalities in an application, increases the size of data or state handled in the application. Since the data rendered in different parts of the application are co-related, the change in data at one part affects the whole application. In such scenarios when an error is encountered, it becomes very tough to solve it. To solve such issues, it is very important to know from where and how the data has changed. However, in many cases there is no defined rule to store and update the application state as the state management library is not used.[15]

This is where Redux comes into play. As defined in the official documentation, redux is a predictable state container for JavaScript applications.[14] Redux provides a centralized state container which stores the data used in the application keeping it separated from the UI. In addition, to keep it predictable, redux provides a set of rules needed to be followed to update the data contained in the state container. This way any issues encountered in the application with the change in the state are found and resolved easily.[15]

Single source of truth, read-only state and making changes with pure functions are the three fundamental principles based on which Redux can be described. Redux saves the state of the application in object tree format inside redux store. The data stored in the store is easily accessible to the UI components however, it cannot be changed directly. To change the state, actions should be emitted. Actions are plain JavaScript objects which contain type as the mandatory item. These emitted actions including the application state are then passed through the pure JavaScript functions which are responsible for updating the state based on the action type. These functions are known as reducers and they return the updated state of the application in the object tree format as required by the redux store. Finally, when the application state is updated, the UI components update the rendered data automatically.[15]

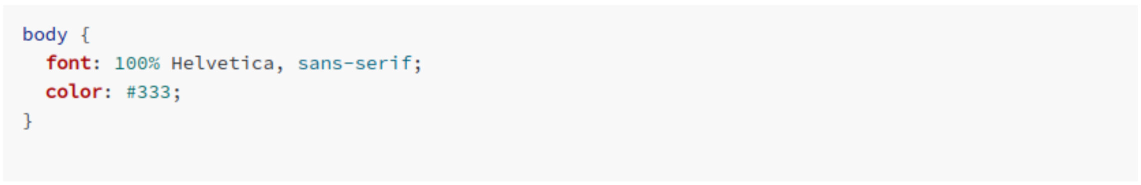
4.2.3 SASS

SASS, Syntactically Awesome Style Sheets, is a CSS pre-processor created for developers to help them code in a faster and more efficient way. CSS pre-processors are scripting languages, extended from CSS, which contains all the features of CSS and additionally include extra features such as variables, mixins, inline styles, Inheritance and nested rules. According to the official website of SASS, it is one of the most popular and widely used CSS pre-processors existing currently in the world. Some of the most common examples of CSS pre-processors other than SASS are Less, Stylus, etc.

SASS provides two different syntax options which are listed below:

1. Indented
2. SCSS

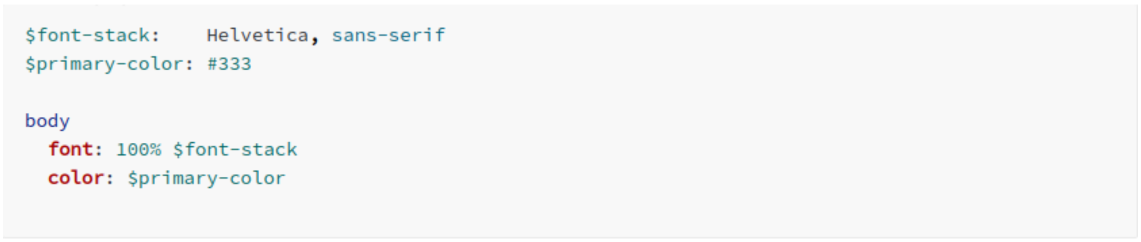
Indented SASS is written in file with extension .sass. Coding using this syntax is more concise since extra space is not used by semi-colon and curly braces. Below is the sample code written using CSS and indented syntax of sass:



```
body {
  font: 100% Helvetica, sans-serif;
  color: #333;
}
```

Fig 2: CSS code [16]

Sample code in Fig 2, defines the styles for the body HTML element. “font” property defines the styles for texts written inside the body element whereas the “color” property defines the colour of the text.

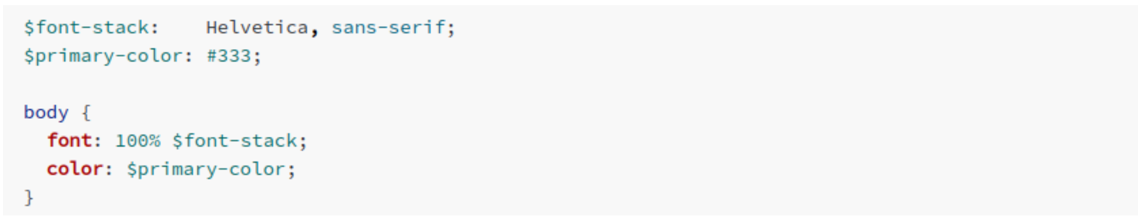


```
$font-stack:    Helvetica, sans-serif
$primary-color: #333

body
  font: 100% $font-stack
  color: $primary-color
```

Fig 3: SASS code[16]

Sample code in Fig 3 does the same task as done by the code in Fig 2. However, in Fig 3, variables, $font-stack and $primary-color, are used to store the styles at first and then they are supplied to the font and “color” properties inside the body element. These variables could be used to style any HTML elements, such as div, p, section, etc., inside the same file. Fig 3 also depicts the elimination of semicolons and curly braces which makes the code clearer and more concise.

SCSS is written in the file with extension .scss. This syntax is similar to the syntax of CSS as semicolons and curly braces are used in SCSS. This makes SCSS superset of CSS and makes it easier to learn. Due to this reason SCSS is preferred more by developers compared to indented SASS. Below is the sample code written using SCSS:



```
$font-stack:    Helvetica, sans-serif;
$primary-color: #333;

body {
  font: 100% $font-stack;
  color: $primary-color;
}
```

Fig 4: SASS code[16]

Similarly, sample code in Fig 4 performs the same task as done by code in Fig 4 and Fig 3. The syntax in Fig 4 looks like the syntax in Fig 2 as both consists of semicolons and curly braces. However, variables are supported only in SCSS code, Fig 4, not in CSS code, Fig 2.

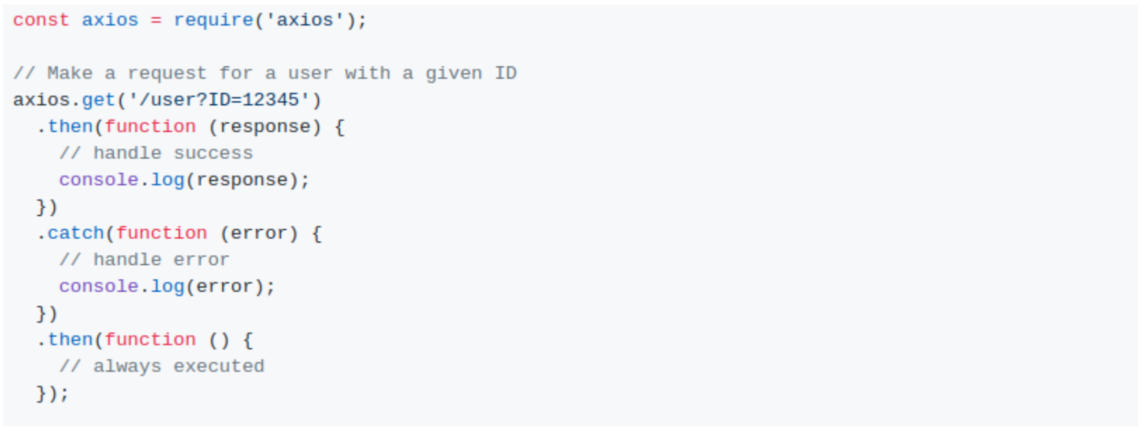
4.2.4 Axios

When an URL is entered in a web browser, the web browser sends a request to the server and the server responds with the client-side application. The user interacts with the application to update some data in the server. To update the data in the server, the application needs to make XMLHTTPRequest from the browser. Axios is a small but powerful library which helps applications to make XMLHTTRequest from the browser.

Every interaction of a client with a server consists of a request and a response. Since, the server and the client are in different locations, it takes some time for a request to resolve. And there might come some scenarios, such as network failure, where the request that is already made needs to be modified or cancelled. Axios allows developers to intercept, transform or cancel any requests that are being made and are not resolved yet. It also provides automatic transformation of other forms of data to JSON format.

Whenever a request is made using Axios, as it takes certain time for a request to resolve, a Promise is returned. Promise is not the actual result received after processing a request. It is a JavaScript object which provides a way to handle the result of the asynchronous requests without obstructing the flow of the application.

Below is an example which illustrates the way to make a http GET request using Axios in JavaScript applications:



```
const axios = require('axios');

// Make a request for a user with a given ID
axios.get('/user?ID=12345')
  .then(function (response) {
    // handle success
    console.log(response);
  })
  .catch(function (error) {
    // handle error
    console.log(error);
  })
  .then(function () {
    // always executed
  });
```

Fig 5. JavaScript code[17]

In Fig 5, line 1 imports axios in the code. After that “axios.get(‘/user?ID=12345’)” make a GET request to the URL “/user?ID=12345”. Whenever the request is successfully executed in the server, the request’s status is either resolved or rejected. If the request is resolved and the result is received, code inside the “then” function is executed. However, if the request is rejected, code inside the “catch” function is executed. Depending on the result of the request, the application either updates the data displayed or an error message is displayed.

Similarly, depending on the need, all kinds of HTTP requests, such as GET, POST, PATCH, PUT, UPDATE and DELETE, can be made seamlessly using Axios.

4.3 Development Approach of client-side Applications

4.3.1 Data flow in client-side Applications

Although there are two different client-side applications in Shift Manager, the architecture of both the applications is the same. Both the applications are API driven and they are built using React and Redux.

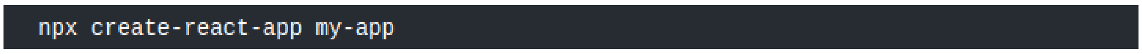
Talking about the flow of data, what happens is, when a user interacts with the application views created using React, events like button clicks, form data submission, etc. are generated. As the application state is managed by Redux, the events generated in react views trigger action creators of the redux application. These action creators are responsible for dispatching actions, JavaScript objects, which are eventually passed through the redux middlewares. This is the place where the actual requests to the APIs are made. After making the API requests, middlewares dispatches the same action however now with the actual data received from the API. Then the action is passed through all the reducers. Although the action is passed through all the reducers, the action is only processed by the responsible reducer assigned for the task. After processing the action, a new state is returned which is then updated in redux store. Whenever React realizes the presence of a new state in a redux store, it automatically re-renders the views with the new data. The application repeats this cycle and ensures the user is displayed with the correct information every time the user triggers any event on the views.

4.3.2 Development Environment and Folder structure

Since, both the client-side applications are designed with the same data flow architecture and as the same technologies are used, the development process is also the same.

Being a React Redux application, the development environment is set up by installing Node.js. Similarly, VS code or Visual Studio code is installed for coding, npm (node package manager) is installed for managing the packages used in the application and Git is installed for the version controlling purpose. As VS code provides customization features, Git bash terminal is used as the default terminal instead of the inbuilt VS code terminal. Since developers are more familiar with Git bash terminal compared to VS code terminal, Git bash is used.

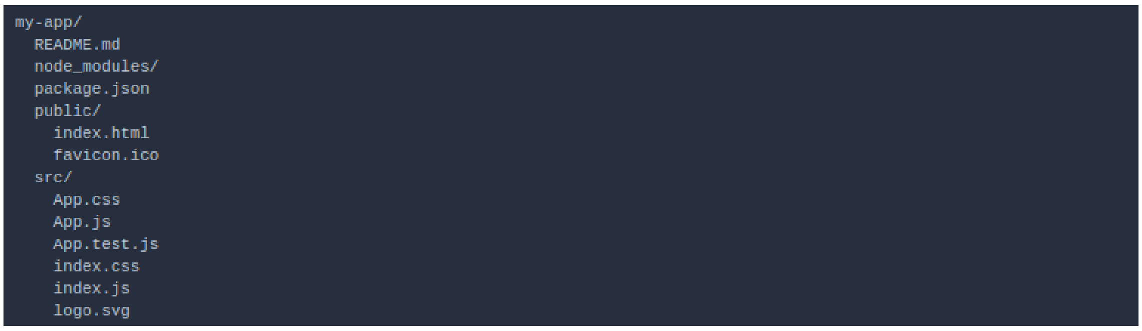
After setting up the development environment, the project is initiated with the help of the command, create-react-app, provided by the React community. This command is entered in the terminal of VS code.



```
npx create-react-app my-app
```

Fig 6.[17]

The command in Fig 6 initiates the React project with the name “my-app”. Entering this command generates initial folder structure as depicted below in Fig 7.



```
my-app/
  README.md
  node_modules/
  package.json
  public/
    index.html
    favicon.ico
  src/
    App.css
    App.js
    App.test.js
    index.css
    index.js
    logo.svg
```

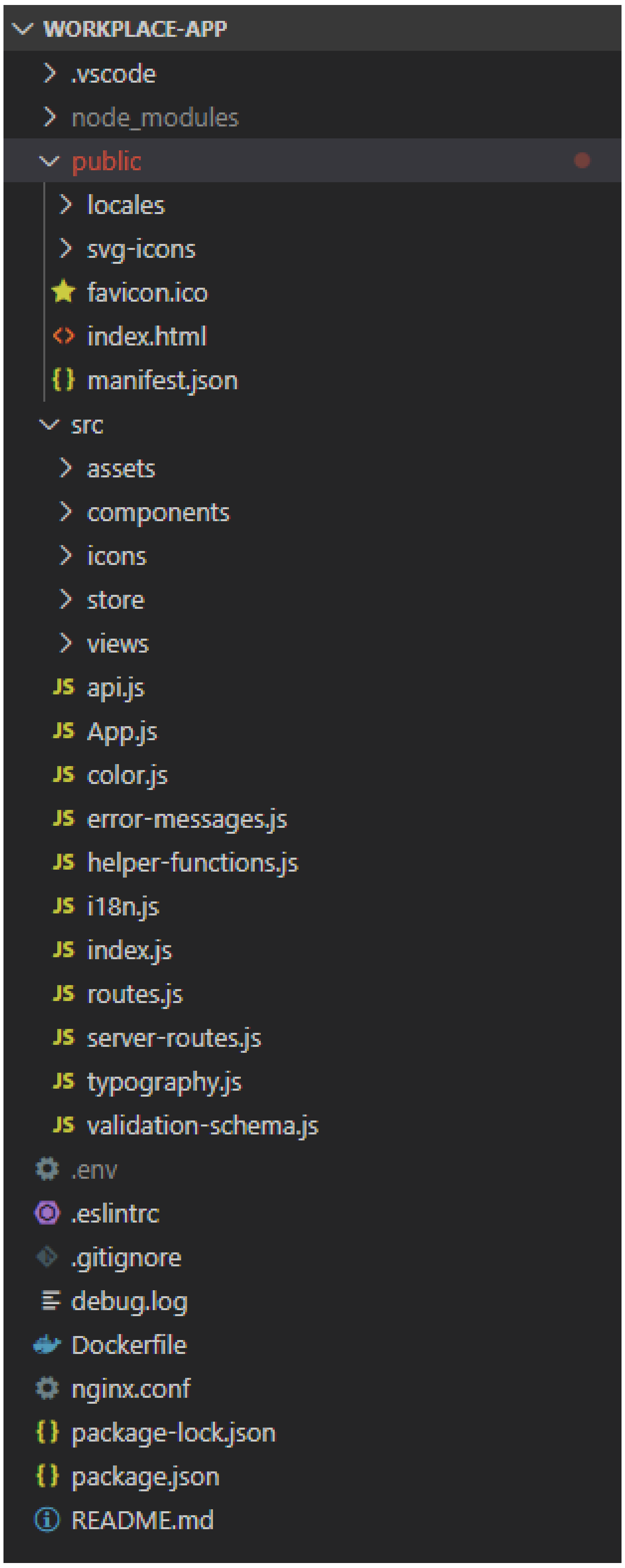
Fig 7.[18]

It is seen that inside the “my-app” folder there exists two files README.md and package.json. README.md is a file used by Git and package.json is used in the project to keep track of all the code packages used in the project. Beside these two files there exist three folders: node_modules, public, and src. node_modules is there to hold actual packages mentioned in the package.json file, the public folder contains the html page and other static files, such as icons, served in the html page and the src folder contains all the code that is written by the developers.

At this stage, the project supports React, JavaScript and CSS. Since technologies like SCSS, Redux and Axois are used in Shift Manager, the content inside the public and src folders is modified to meet the requirements of the project. However, there are two files that must exist with the same name so that the project builds successfully. These files are “public/index.html” and “src/index.js”.

Fig 8 is the screenshot of the folder structure created in the client-side applications of shift manager. When looked at carefully, node_modules, public, src, package.json and README.md still exists. Also, there exists “public/index.html” and “src/index.js” files.

In addition, there exists many other files and folders which makes the folders structure seem complex. In this structure there exists few which are automatically generated by vs code since they are needed for the code management purpose. These automatically generated files and folders are: vscode, package-lock.json and debug.log.



```
WORKPLACE-APP
  .vscode
  node_modules
  public
    locales
    svg-icons
    favicon.ico
    index.html
    manifest.json
  src
    assets
    components
    icons
    store
    views
    api.js
    App.js
    color.js
    error-messages.js
    helper-functions.js
    i18n.js
    index.js
    routes.js
    server-routes.js
    typography.js
    validation-schema.js
  .env
  .eslintrc
  .gitignore
  debug.log
  Dockerfile
  nginx.conf
  package-lock.json
  package.json
  README.md
```

Fig 8. Screenshot of folder structure of Workplace application taken in VS code.

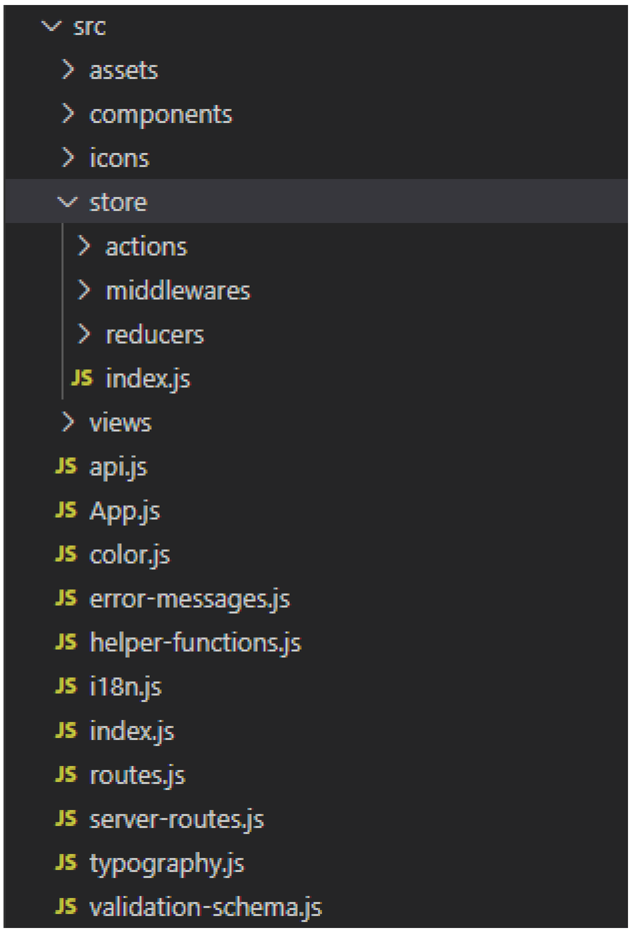
Also, there exist files which are manually generated that are not used for development purposes rather they are used while deployment of the applications. These files are .env, Dockerfile and nginx.conf.

Similarly, .gitignore is used by Git to ignore the files and folders that are not required to be saved in the remote repository in GitHub. For instance, the node_modules folder and the .env file are not meant to be saved in the remote repository. The reason behind this is, the node_modules folder is huge, and it can be generated automatically with the help of package.json file. Similarly, .env file is not saved in the remote repository for security reasons since it can contain critical information like usernames and passwords used in the application.

Until this point all other files and folders except public and src are explained. So, moving forward to the public folder locales, svg-icons, and manifest.json seems to be the new files and folders included. The purpose of the locales folder is to hold the language translations that are used in the application. Depending on the user's language preference suitable translation is rendered from this folder. "svg-icons" folder on the other hand holds the icons that are used in the application but not directly. They need to be changed to React components first. Finally, the purpose of manifest.json file is to make the application downloadable. The browser creates a link from this manifest.json file and loads at the top of the URL bar. Clicking the link sets the application icon at the home screen. As mentioned earlier, since these files and folders are not meant to be compiled during the build process, they are kept here.

Now the last and most important folder remaining is the src folder. This is the place where all the code written to make the application resides. Since the application involves thousands of lines of code, for the better management of the code, code related to different technologies are managed in different folders.

Below is the screenshot of the folder structure of the "src" folder of the workplace application.



```
src
  assets
  components
  icons
  store
    actions
    middlewares
    reducers
    index.js
  views
  api.js
  App.js
  color.js
  error-messages.js
  helper-functions.js
  i18n.js
  index.js
  routes.js
  server-routes.js
  typography.js
  validation-schema.js
```

Fig 9. Screenshot of src folder structure of Workplace application taken in VS code.

Both the client-side applications consist of multiple views. Each of the views is created as a separate React component. Since, these views are complex, for the sake of simplicity, they are further broken down into smaller React components. These view components and the smaller components are located inside “src/views” and “src/components” respectively. “src/icons” folder also contains the React component; however, these components represent the icons present inside “public/svg-icons” and they are automatically generated.

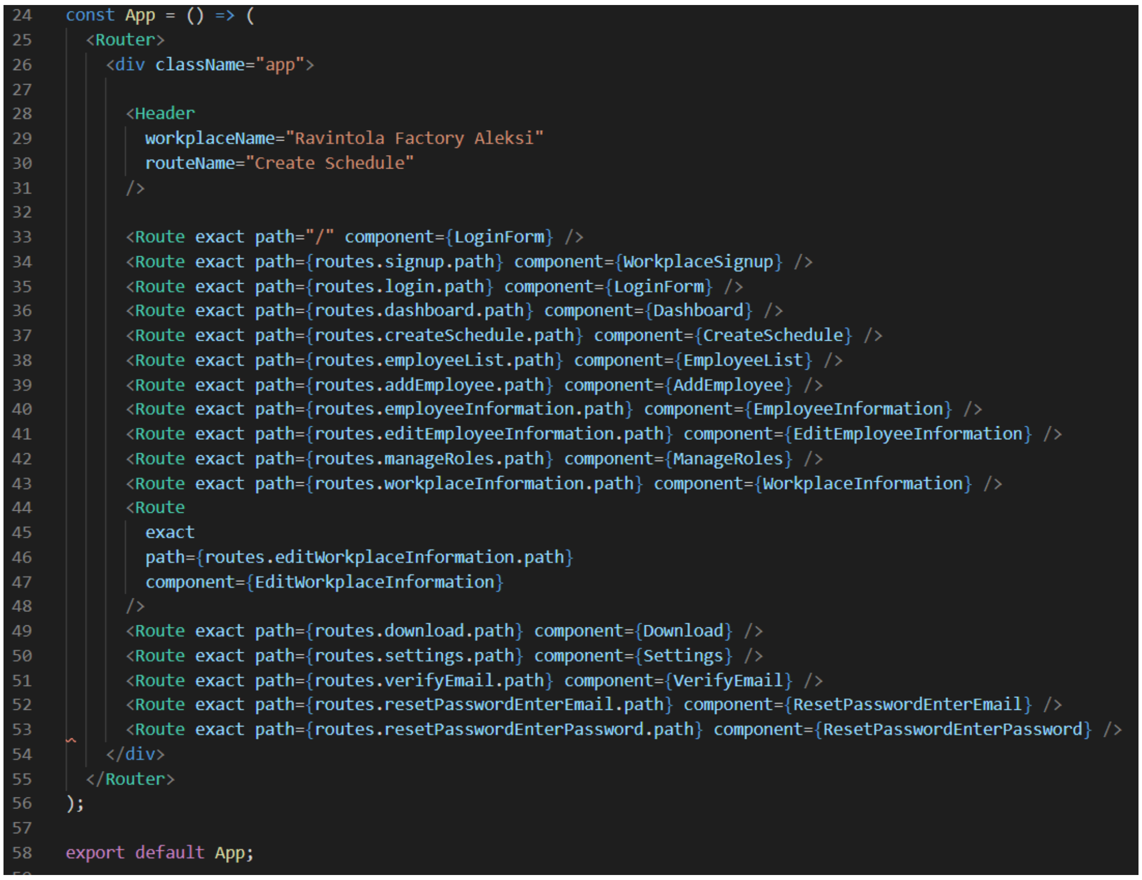
These views and the components have their own styles created using SCSS. The SCSS code written for styling the components is placed inside the “src/assets/styles” folder.

Since, the responsibility of the React components is to interact with the user and fetch the required data from the Redux store, a decent amount of code is needed to set up the flow of the data in and out of the store. The code required for this set up is situated inside the “src/store” folder.

Explaining the remaining files inside the “src” folder; all other files except App.js and index.js are the helper JavaScript files serving different purposes such as making API requests to the backend application and holding informations for instance colour codes, typography, API list, application routes list, error messages, input fields validation schema, etc. App.js file is used for connecting React and Redux applications together with the help of connecting libraries and making the application whole. The formed application is then injected to the html template file “public/index.html” from “src/index.js” file.

4.3.3 Connection between Application Views and Application Store

Until this point it is known that separate views of the applications are created as separate React components and they are placed in the “src/views” folder. But to make a complete solution, they need to be glued together which is done with the help of a connector package called react-router-dom. This connector provides multiple React components to do the job, out of which following components have been used: Router, Route and Link. Below is the snapshot of a code section, inside “src/app.js” file, where the views have been tied together:

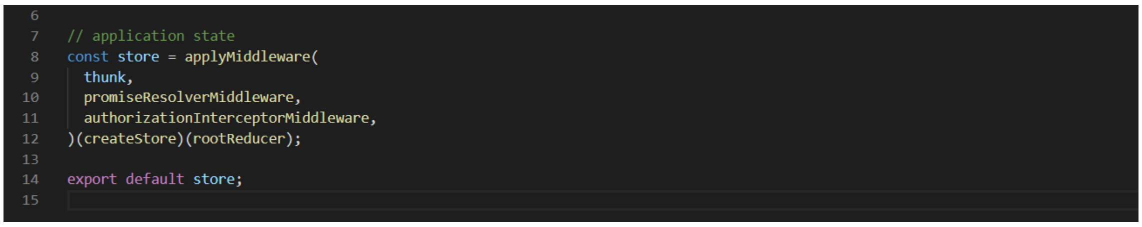


```
const App = () => (
  <Router>
    <div className="app">

      <Header
        workplaceName="Ravintola Factory Aleksi"
        routeName="Create Schedule"
      />

      <Route exact path="/" component={LoginForm} />
      <Route exact path={routes.signup.path} component={WorkplaceSignup} />
      <Route exact path={routes.login.path} component={LoginForm} />
      <Route exact path={routes.dashboard.path} component={Dashboard} />
      <Route exact path={routes.createSchedule.path} component={CreateSchedule} />
      <Route exact path={routes.employeeList.path} component={EmployeeList} />
      <Route exact path={routes.addEmployee.path} component={AddEmployee} />
      <Route exact path={routes.employeeInformation.path} component={EmployeeInformation} />
      <Route exact path={routes.editEmployeeInformation.path} component={EditEmployeeInformation} />
      <Route exact path={routes.manageRoles.path} component={ManageRoles} />
      <Route exact path={routes.workplaceInformation.path} component={WorkplaceInformation} />
      <Route
        exact
        path={routes.editWorkplaceInformation.path}
        component={EditWorkplaceInformation}
      />
      <Route exact path={routes.download.path} component={Download} />
      <Route exact path={routes.settings.path} component={Settings} />
      <Route exact path={routes.verifyEmail.path} component={VerifyEmail} />
      <Route exact path={routes.resetPasswordEnterEmail.path} component={ResetPasswordEnterEmail} />
      <Route exact path={routes.resetPasswordEnterPassword.path} component={ResetPasswordEnterPassword} />
    </div>
  </Router>
);

export default App;
```

Fig 10. Screenshot of a code section inside “src/app.js” file of Workplace application taken in VS code.

As shown in Fig 10, the Router being a parent component wraps all the views between line numbers 25 to 55. It can also be seen that there are multiple Route components listed one after another between line numbers 32 and 54. The way it works is, the Router component renders only one Route at a time depending on the URL selected by the user. The Router component compares the URL, present in the URL bar of the browser, to the “path” property supplied in the Route component; the Route with the matching “path” is then rendered in the application. In this way, all the views are combined together to form a complete react application detached from the Redux store.

Since the state of the application is in the Redux store, it should be supplied to the React application. In this case, the Redux store is created by combining multiple reducers and these reducers are created such that each view of the application has a separate reducer. These reducers are connected, to form a root reducer, with the help of a connector known as Combined Reducer. Thus, the formed root reducer is then used to create the Redux store inside the file “src/store/index.js”.

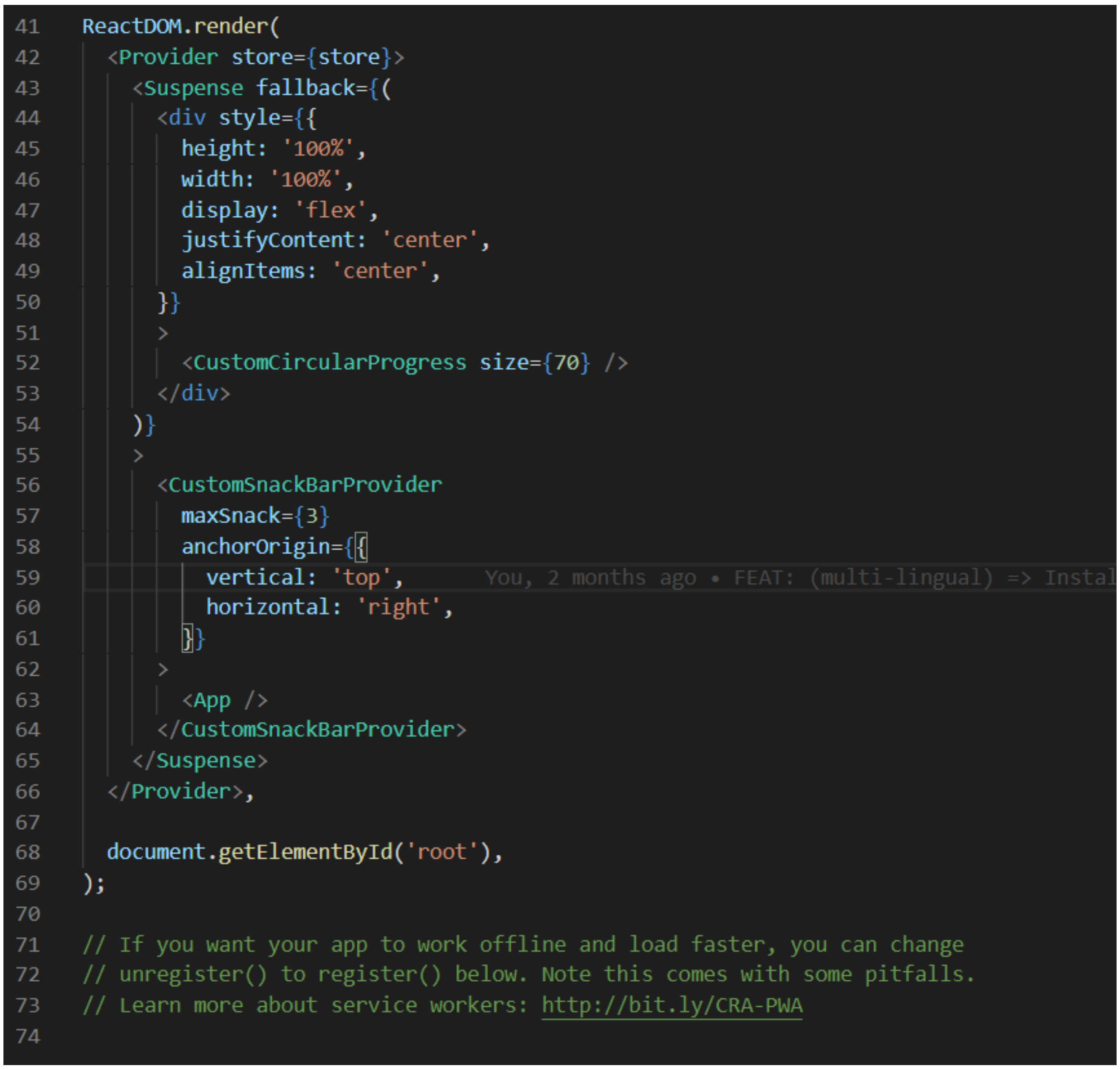


```
6
7   // application state
8   const store = applyMiddleware(
9     thunk,
10    promiseResolverMiddleware,
11    authorizationInterceptorMiddleware,
12  )(createStore)(rootReducer);
13
14  export default store;
15
```

Fig 11. Screenshot of a code section inside “src/store/index.js” file of Workplace application taken in VS code.

The code in Fig 11 generates a store for the application, attaching “rootReducer” and three different middlewares, with the help of the “createStore” module and “applyMiddleware” function that are used in line number 12 and line number 8 respectively. The created store is then exported from the file, “src/store/index.js”, so that it can be supplied to the React application.

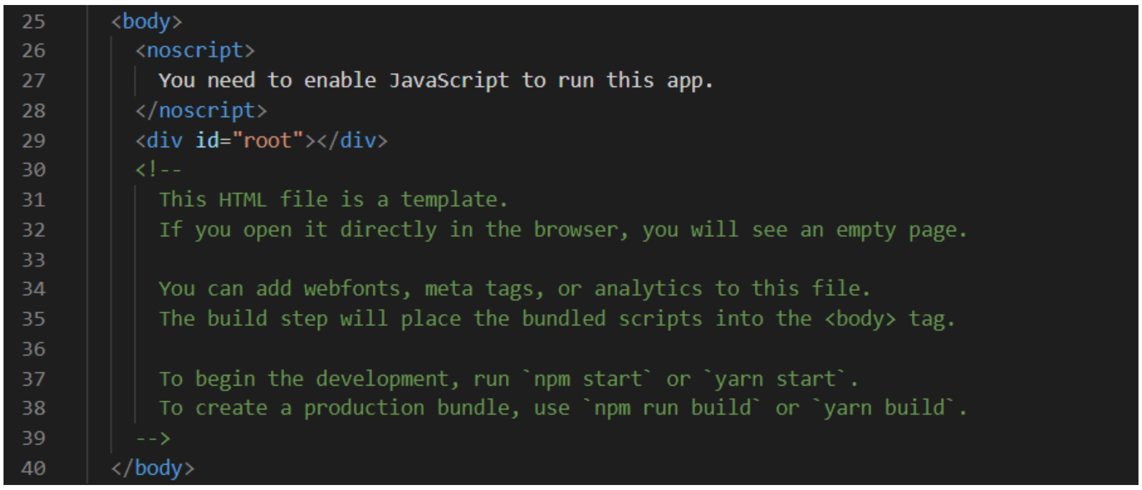
Although the Redux store and the React application are created, they need to be glued together and the whole application needs to be injected in the html template. This work is done inside the file “src/index.js” with the help of the Provider component provided by React itself.



```
41  ReactDOM.render(
42    <Provider store={store}>
43      <Suspense fallback={(
44        <div style={{
45          height: '100%',
46          width: '100%',
47          display: 'flex',
48          justifyContent: 'center',
49          alignItems: 'center',
50        }}
51        >
52          <CustomCircularProgress size={70} />
53        </div>
54      )}
55      >
56        <CustomSnackBarProvider
57          maxSnack={3}
58          anchorOrigin={{
59            vertical: 'top',       You, 2 months ago • FEAT: (multi-lingual) => Instal
60            horizontal: 'right',
61          }}
62        >
63          <App />
64        </CustomSnackBarProvider>
65      </Suspense>
66    </Provider>,
67
68    document.getElementById('root'),
69  );
70
71  // If you want your app to work offline and load faster, you can change
72  // unregister() to register() below. Note this comes with some pitfalls.
73  // Learn more about service workers: http://bit.ly/CRA-PWA
74
```

Fig 12. Screenshot of a code section inside “src/index.js” file of Workplace application taken in VS code.

Although the code in Fig 12 seems complicated, the focus here is the code segment that connects the Redux store with the React application and the code segment that injects the whole application in the html template. The Provider component, being the parent component, wraps all the code in between line 42 to 66 including the “App” component in line 63. Here, the purpose of this component is to supply the application state, from the Redux store and supplied to it as a property “store” (as depicted in line 42), throughout the application. Now the application is complete and is ready to be injected in the html template. ReactDom renders the whole application inside the template with the help of the “render” function as shown in line number 42. The location in the template where the application should be injected is mentioned at line 68. Fig 13, below, is the screenshot of a code section inside the template file where the application is injected in the element “div” with id “root” in line 29:



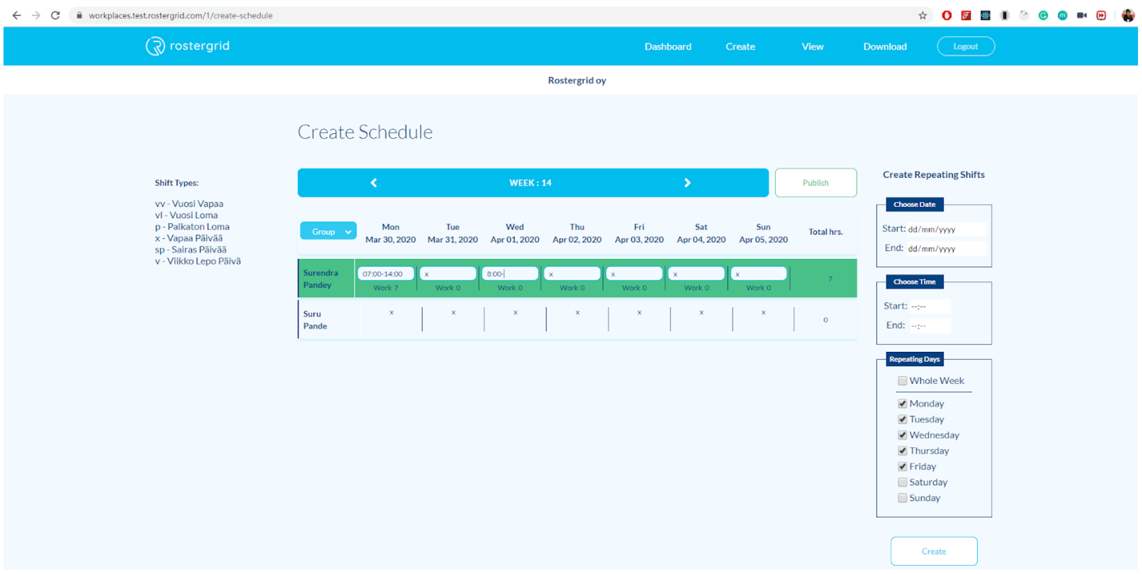
```html
  <body>
    <noscript>
      You need to enable JavaScript to run this app.
    </noscript>
    <div id="root"></div>
    <!--
      This HTML file is a template.
      If you open it directly in the browser, you will see an empty page.

      You can add webfonts, meta tags, or analytics to this file.
      The build step will place the bundled scripts into the <body> tag.

      To begin the development, run `npm start` or `yarn start`.
      To create a production bundle, use `npm run build` or `yarn build`.
    -->
  </body>
```

Fig 13. Screenshot of a code section inside “public/index.html” file of Workplace application taken in VS code.

5 Outcome

The outcome of the project are two different web applications namely Workplace App and User App created using React and Redux. Since the Workplace application is for supervisors, features like work shifts creation, work shifts publication and hours calculation exist in this application. And as the User App is for employees, features like work shifts delivery and employees' work history exist in this application.

Among multiple views in the Workplace application, tasks related to work shifts are done in Create Schedule view. Fig 14, below, is the screenshot of the view taken from the Workplace application deployed in the cloud. The name of the workplace is “Rostergrid Oy” which is displayed below the navigation bar at the top of the view. There is a list of employees in the mid-section of the view which contains two employees. Shifts can be given to these employees by selecting each employee and assigning them shifts for a date. For instance, Fig 14. depicts the creation of work shifts for “Surendra Pandey”; when the employee is selected, a green background appears with the form fields for each date. In addition, as shown in the Fig 14 the hours are calculated automatically and displayed at the end of the employee bar and in case of “Surendra Pandey” total hours for week 14 is 7 hours.



Rostergrid oy

Create Schedule

Shift Types:

vv - Vuosi Vapaa
vl - Vuosi Loma
p - Palkaton Loma
x - Vapaa Päivää
sp - Sairas Päivää
v - Viikko Lepo Päivä

WEEK : 14

Group	Mon Mar 30, 2020	Tue Mar 31, 2020	Wed Apr 01, 2020	Thu Apr 02, 2020	Fri Apr 03, 2020	Sat Apr 04, 2020	Sun Apr 05, 2020	Total hrs.
Surendra Pandey	07:00-14:00 Work 7	x Work 0	8:00- Work 0	x Work 0	x Work 0	x Work 0	x Work 0	7
Suru Pande	x	x	x	x	x	x	x	0

Create Repeating Shifts

Choose Date
Start: dd/mm/yyyy
End: dd/mm/yyyy

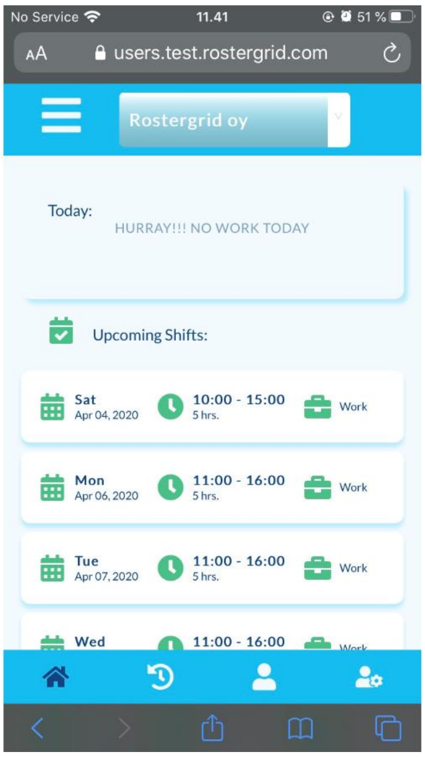
Choose Time
Start: --:--
End: --:--

Repeating Days
☐ Whole Week
☑ Monday
☑ Tuesday
☑ Wednesday
☑ Thursday
☑ Friday
☐ Saturday
☐ Sunday

Fig 14. Screenshot of Create Schedule view inside Workplace App deployed in Cloud.

After the shifts are successfully created for all the employees clicking the publish button publishes the work schedules.

Thus, published work shifts are now visible in the dashboard of each employee inside the User App. However, the employees can see their work shifts only, not the entire schedule of the workplace. Fig 15 and Fig 16, below, are the screenshot of the dashboard view and the history view taken from the User App deployed in the cloud. In the dashboard view only today's and upcoming shifts of an employee are seen. However, in the history view work shifts of the past four months are seen. As shown in Fig 15, the employee does not have any shift for today however, there are plenty of upcoming shifts for him. Similarly, Fig 16 shows that the employee had two work shifts assigned in the month of April. Since the screenshot is taken in the month of April, it is mentioned as "This month" in Fig 16.

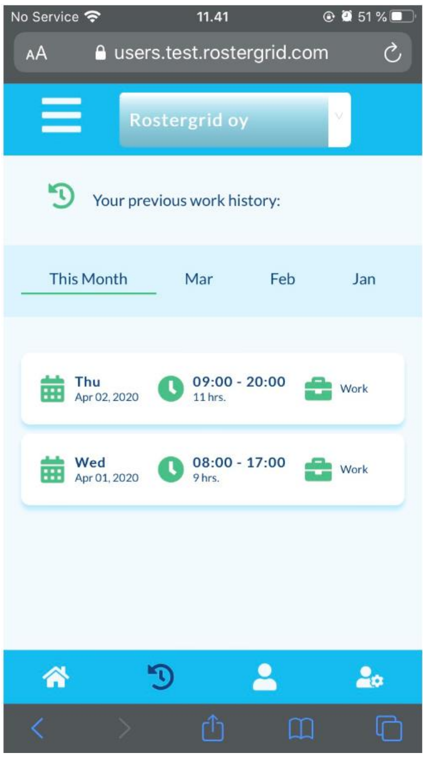


Today:
HURRAY!!! NO WORK TODAY

Upcoming Shifts:

Sat Apr 04, 2020	10:00 - 15:00 5 hrs.	Work
Mon Apr 06, 2020	11:00 - 16:00 5 hrs.	Work
Tue Apr 07, 2020	11:00 - 16:00 5 hrs.	Work
Wed	11:00 - 16:00	Work

Fig 15. Screenshot of the Dashboard view taken from the User App deployed in Cloud.



Thu Apr 02, 2020	09:00 - 20:00 11 hrs.	Work
Wed Apr 01, 2020	08:00 - 17:00 9 hrs.	Work

Fig 16. Screenshot of the History view taken from the User App deployed in Cloud.

Finally, the Download view in the Workplace App allows supervisors to download the employee schedules for a selected date range. Providing start date, end date and file name and clicking the download button downloads an excel file containing the information useful during salary payment. In the downloaded file total hours, evening hours, night hours, Sunday hours and information related to holidays are present. All this information is calculated by the application itself so that there is no room for human errors.

6 Future of the project

At this point MVP of the project is deployed in the cloud infrastructure of Digital Ocean and the client company has tested the product in two different branches located in Helsinki and Espoo. Since the addressed issues have been solved, they are satisfied with the services provided. Now they are looking forward to implementing the solution in their other branches. In addition, they have requested for new features that could make them more productive.

Making the MVP of the product is ready is not the final task as there are many improvements needed to be done. So, the development of the product will be prolonged mainly focusing on the improvements, fulfilling the demand of new features and maintaining the product in the long run.

7 Conclusions

Thus, the goal of aiding supervisors in creating and publishing the work shifts seamlessly and eradicating the human errors during hours calculation is achieved by creating a cloud-based web application, Shift Manager. Shift Manager consists of two client-side applications, Workplace App and User App, working together to provide the services. Among these applications, the Workplace App is created for supervisors to generate work shifts and the User App is created for employees to view their upcoming work shifts and their work history.

These client-side applications are created using React and Redux. As they have multiple views, it becomes very difficult to manage the data flowing in the application and becomes complicated and time consuming to create the views. However, understanding the data handling process of Redux and learning the way of visualising the views in terms of React components made the development of the applications easy and interesting. In addition, since the architecture of the React Redux application maintains the one-way flow of data it became very easy to track bugs in the applications.

Finally, since the client-side applications are run in browsers, they are accessible from most of the devices all over the world making them available to maximum number of users.

References

1. Lifewire. 2019. *What Exactly Is A Web Application?* [online] Available at: <https://www.lifewire.com/what-is-a-web-application-3486637> [Accessed 13 April 2019].

2. Gibb, R., 2016. *What Is A Web Application? | How A Web Application Works*. [online] Articles for Developers Building High Performance Systems. Available at: <https://www.maxcdn.com/one/visual-glossary/web-application/> [Accessed 13 April 2019].

3. web.dev. 2020. *Progressive Web Apps*. [online] Available at: <https://developers.google.com/web/progressive-web-apps/> [Accessed 20 April 2019].

4. Azure.microsoft.com. 2020. *What Is Cloud Computing? A Beginner'S Guide | Microsoft Azure*. [online] Available at: <https://azure.microsoft.com/en-us/overview/what-is-cloud-computing/> [Accessed 26 April 2019].

5. Klipfolio.com. 2020. *Data Dashboards. Definition, Design Ideas Plus 3 Examples*. [online] Available at: <https://www.klipfolio.com/resources/articles/what-is-data-dashboard> [Accessed 1 May 2019].

6. Klipfolio.com. 2020. *Choosing The Right Dashboard Report*. [online] Available at: <https://www.klipfolio.com/choosing-the-right-dashboard-report> [Accessed 1 May 2019].

7. Softwaretestingmaterial.com. 2020. *Software Architecture: One-Tier, Two-Tier, Three Tier, N Tier*. [online] Available at: <https://www.softwaretestingmaterial.com/software-architecture/> [Accessed 14 November 2019].

8. Figma. 2020. *A Free, Online UI Design Tool For Teams | Figma*. [online] Available at: <https://www.figma.com/ui-design-tool/> [Accessed 20 November 2019].

9. En.wikipedia.org. 2020. *Visual Studio Code*. [online] Available at: <https://en.wikipedia.org/wiki/Visual_Studio_Code> [Accessed 25 November 2019].

10. Atlassian. 2020. *What Is Version Control | Atlassian Git Tutorial*. [online] Available at: <https://www.atlassian.com/git/tutorials/what-is-version-control> [Accessed 6 December 2019].

11. Kinsta Managed WordPress Hosting. 2020. *What Is Github? A Beginner's Introduction To Github*. [online] Available at: <https://kinsta.com/knowledgebase/what-is-github/> [Accessed 10 December 2019].

12. Codecademy. 2020. *Use Devtools | Codecademy*. [online] Available at: <https://www.codecademy.com/articles/use-devtools> [Accessed 1 January 2020].

13. Buna, S., 2020. *Yes, React Is Taking Over Front-End Development. The Question Is Why.* [online] freeCodeCamp.org. Available at: <https://www.freecodecamp.org/news/yes-react-is-taking-over-front-end-development-the-question-is-why-40837af8ab76/> [Accessed 6 January 2020].

14. Redux.js.org. 2020. *Redux - A Predictable State Container For Javascript Apps. | Redux*. [online] Available at: <https://redux.js.org/> [Accessed 15 January 2020].

15. Batool, S., 2020. *An Intro To Redux And How State Is Updated In A Redux Application*. [online] freeCodeCamp.org. Available at: <https://www.freecodecamp.org/news/an-intro-to-redux-and-how-state-is-updated-in-a-redux-application-839c8334d1b1/> [Accessed 25 January 2020].

16. Sass-lang.com. 2020. *Sass: Sass Basics*. [online] Available at: <https://sass-lang.com/guide> [Accessed 28 January 2020].

17. GitHub. 2020. *Axios/Axios*. [online] Available at: <https://github.com/axios/axios> [Accessed 2 February 2020].

18. Reactjs.org. 2020. *Create A New React App – React*. [online] Available at: <https://reactjs.org/docs/create-a-new-react-app.html> [Accessed 9 February 2020].

19. Create-react-app.dev. 2020. *Create React App · Set Up A Modern Web App By Running One Command*. [online] Available at: <https://create-react-app.dev/docs/folder-structure/> [Accessed 25 March 2020].