VAASAN AMMATTIKORKEAKOULU
UNIVERSITY OF APPLIED SCIENCES

Quang Luong

WEB APPLICATION DEVELOPMENT WITH REACTJS FRAMEWORK

CASE: WEB APPLICATION FOR AN ASSOCIATION

School of Technology
2019

VAASAN AMMATTIKORKEAKOULU
UNIVERSITY OF APPLIED SCIENCES
Information Technology

ABSTRACT

Author	Quang Luong
Title	Web Application Development with ReactJS Framework
Year	2019
Language	English
Pages	41
Name of Supervisor	Pirjo Prosi

The objective of this thesis was to study ReactJS framework with Redux library then apply it as an advanced technique to building up a web application. Furthermore, Firebase service from Google, a server-less service which offered many features for web development in convenient way, was also studied.

The most important reason for choosing ReactJS was this framework is known to be one of the most widely used ones. The motivation for the project done in this thesis was to create a homepage for an association, they want to have an application to publish their news, events and achievements. Besides the ReactJS, Bootstrap 4 framework also was used for user interface and PHP mail function was used for back-end service.

Summarizing this project work, the study showed that React is a strong framework for building modern web pages with the help from Google Firebase service. However, it is not the best solution for all the web development problems.

Keywords: Web app, web design, data, programming, new technology, ReactJS, Bootstrap, Firebase

CONTENTS

LIST OF FIGURES

LIST OF ABBREVIATIONS

API – Application Program Interface

CMS – Content Management System.

CSS – Cascading Style Sheets

DOM – Document Object Model

HTML – Hypertext Markup Language

HTTP – HyperText Transfer Protocol

IDE – Integrated Development Environment

JS – JavaScript

JSX – JavaScript XML

PHP – Hypertext Preprocessor

SCM – Source Code Management

SPA – Single Page Application

UI – User Interface

URL – Uniform Resource Locator

UX – User Experience

VSC – Visual Studio Code

XML – eXtensible Markup Language

1 INTRODUCTION

Nowadays, ReactJS has become a helpful framework for building web-based application and doing different tasks. According to SimilarTech website, it is now being used on more than 682,000 websites /1/. With the supporting of Redux library, ReactJS are more likely used in the development of an application standard of dynamic perspectives and responsiveness.

1.1 Study Case

The team leader of Chubby Kitchen wants to have a homepage for their team to publish information about their events in the future and the events they archived in the pass. In this way, the customer can connect with them in a convenient way and have a general look at their team activities.

Furthermore, the webpage application should be compact and easy to use for all customers when they visit the website. So, the problem can be solved by a single page application, which is a combination of ReactJS framework and a severless service called Firebase from Google. By developing this application, we can study more about how ReactJS and Firebase service works, we also know how ReactJS and Redux are compatible with each other.

However, a back-end server is also important when develop a website, so we try to develop a server-side service which can send an email with an attachment to the users of the application. This can be done with PHP classes, which will help us understand how PHP can work with ReactJS.

1.2 Scope and objectives

The main objective of this thesis project is to create a dynamic web application for the Chubby Kitchen team. The new web application interface should be divided into several sections, where, the basics information of the association will be shown on the homepage where all users can see it. Besides that, the information about upcoming events and the updates of latest achievements will also appear in the homepage. These next events and achievement have their own pages.

Users can access to the homepage via an online address and they can register a new account with an email address. With that account, users can edit their basic information (First name, Last name, address), in addition, they can also enroll to events.

Admin users, who are leaders of Chubby Kitchen, can login to the dashboard (a CMS only accepts login action by admin account). With this CMS, the admin can add, edit and remove any events, achievement and user accounts. The CMS also shows the stats of events and users on the screen.

Whenever a user enrolls to an event, the system will automatically send a confirmation email to the user's email address. The email also attaches the invoice and confirmation letter as a pdf file.

1.3 Approach and method

This thesis project uses waterfall development as an approach, in which development is performed step by step in a downward direction (for a Waterfall see Figure 1) through several phases, including Requirement analysis, Design, Implementation, Testing and Deployment. The first formal description of the method is often cited as an article published by Winston W. Royce in 1970 although Royce did not use the term "waterfall" in this article /2/.

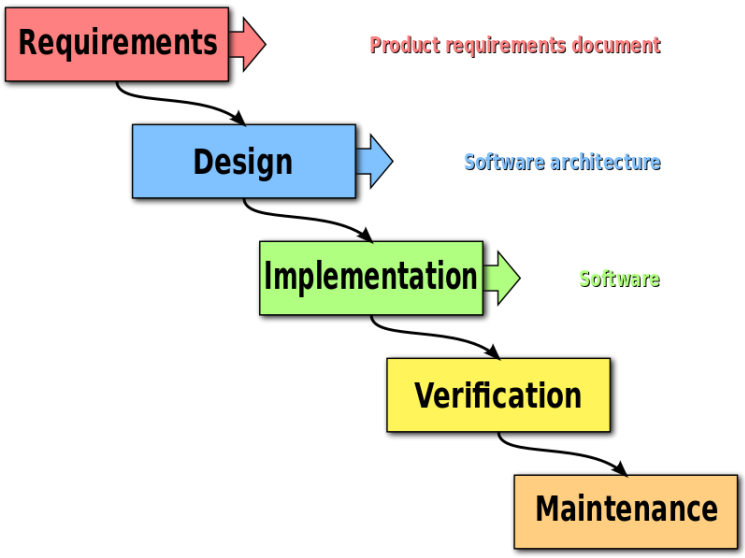

Figure 1. The waterfall development process /2/.

This report is written in six sections. In the beginning, there is an introduction section and then, the second section shows theoretical information on those technologies used in the whole application. Next, in the third section, the instructions for environment and other support tools installation will be provided. After that, the development of some main functions will be shown in several steps in the fourth section. Then, some words about testing and debugging are presented in the fifth section. Finally, the outcomes and review of this thesis project will be discussed in the last section.

2 TECHNOLOGIES

As mentioned before, this project is a Single Page Application conducted with a combination of HTML, Bootstrap 4, JavaScript and ReactJS (also Redux) for the front-end. For the back-end, React-redux-firebase and Google Firebase service are used. On the other hand, some PHP functions (mail, pdf maker) are also applied for the automatic mail system.

2.1 Single page application

Single page Application (SPA) is a web application that enhances the user experience by working inside a browser and it does not require page reloading during working tasks. First, when downloading any web page, SPA will load a single HTML page, then based on the user's request, SPA will continue to load other HTML onto the same page. Currently, there are several big companies that are using SPA technology for their web applications, for instance Gmail, Google Maps, Facebook or GitHub. /3/

In a simpler way, the entire web resources include CSS, JavaScript, master layout files or web page structure that will be loaded for the first time when we start browsing a website. During the browsing, when transferring to another page, the client will send the request to get the necessary data (usually the content). In Figure 2, we can see the difference between SPA and traditional web application when they are working /4/.

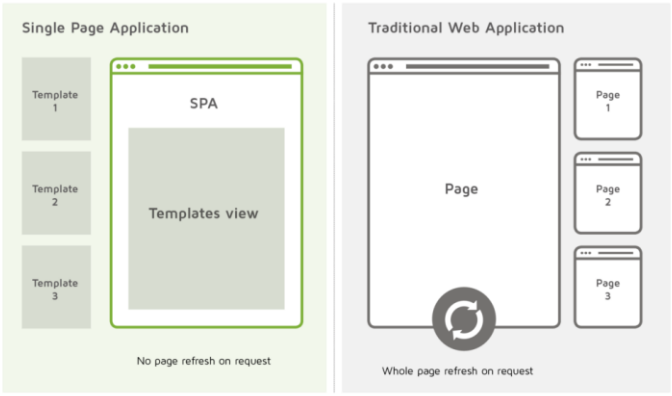

Figure 2. Single Page Application vs. Traditional Web Application /4/.

2.2 HTML5

HTML (HyperText Markup Language) is a text markup language designed to create web pages with information snippets presented on World Wide Web (www). HTML is an indispensable part of the Internet. HTML5 is the 5th revised version of HTML. It is considered the latest standard language for HTML, replacing HTML4, XHTML and HTML DOM Level 2; Designed specifically to provide rich content without the need of additional plugins. It allows a new layer of web applications to be created. Current versions support multimedia content and offline functions without the need for proprietary copyright technologies. They also provide almost everything from animation, graphics, music to movies, and use to build complex web applications /5/.

HTML5 gives users a completely new surfing experience, faster, more stable and more secure. It will help the content on the website work better without depending on any other auxiliary applications. With HTML5 a user can access the web with every device from a PC, a laptop, to mobile devices without any difference. However, it is imperative that the browser used must support HTML5. Current browsers like Chrome, Firefox, IE, Safari, Opera, for example, are making improvements to support and compatible with HTML5.

HTML5 has added a lot of new markups, for example <header> and <footer> tags help in separating the upper and lower parts of the content blocks and to be used multiple times on a single page. The <article> tag helps identify a specific section of content, for example, a blog post or a reader comment. The <nav> tag identifies which parts are considered to be navigation blocks. The <section> tag allows you to specify a piece of content; similar to the current tags. They are more semantic and easier for SEO engines. By improving the ability of searching content on a page, those tags help search engines pay more attention to different parts of the content in individual sections.

HTML5 has also added new useful feature, such as new types of input, local storage, form validation, for example. In the past, those features were handled by JavaScript add-on but now they can be done by HTML5. These changes make the

development easier for developers and make the website work faster than before. Figure 3 shows differences between page structure when using HTML4 and HTML5 elements.

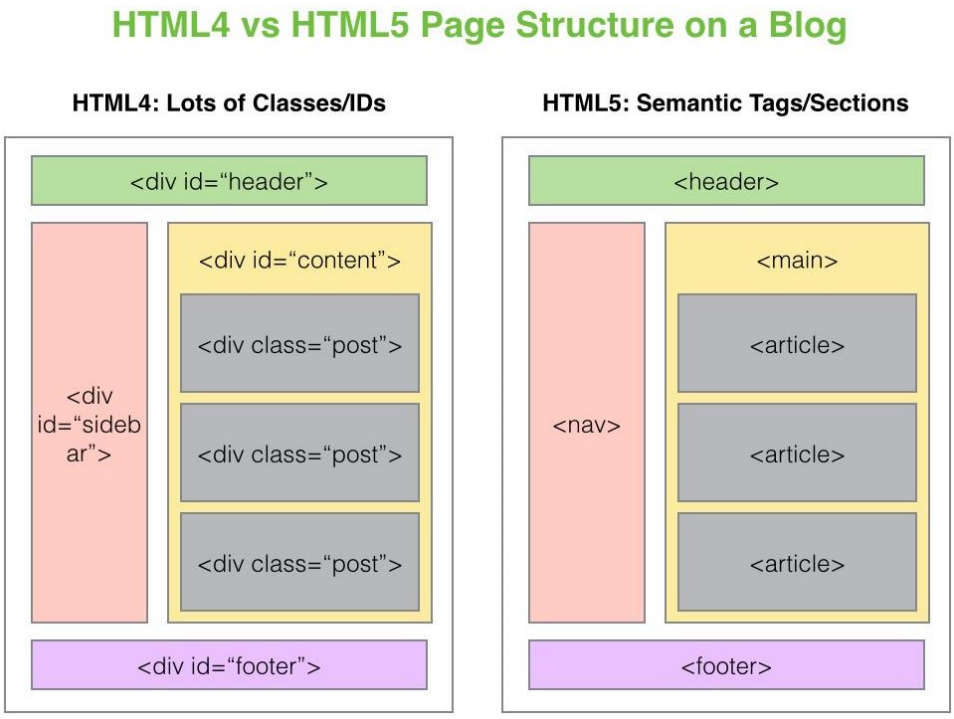

Figure 3. Differences between structure of HTML4 and HTML5 /6/.

2.3 JavaScript

JavaScript is the most popular programming language, which is used by 95% of websites on the internet, according to Web3Techs (Web Technology Surveys) /7/. It is also one of the three main languages of web programming. Together with HTML it helps in adding content to the website, the CSS part designs format, lay-out, style, alignment of web pages, whereas JavaScript will improve how the site works. JavaScript can be learned quickly and easily for a variety of purposes, from improving website features to running games and creating web-based application.

JavaScript was created in ten days by Brandan Eich, a Netscape employee, in September 1995. At the very first moment, it was named Mocha, but its name was changed to Mona and LiveScript before becoming the popular JavaScript now. The first version of this language was restricted exclusively by Netscape and had only

limited features, but it continued to evolve over time, thanks in part to the community of developers who continually worked with it /8/.

In 1996, JavaScript was officially named ECMAScript. ECMAScript 2 was released in 1998 and ECMAScript 3 continued to be released in 1999. It has continued to evolve into JavaScript today, now operating across all browsers and across mobile devices to desktop computers.

2.4 ECMAScript 6 – ES6

ES6 stands for ECMAScript 6, which is considered a set of advanced techniques of JavaScript and is the latest version of the ECMAScript standard.

ECMAScript is proposed by the European Computer Manufacturers Association as the standard of JavaScript language. Considering that there are a lot of browsers available today and if each browser has a different way of running JavaScript, the websites cannot work on all those browsers, so there should be a common standard forcing browsers to develop based on that standard. An example of different from pure JS function of ES6 can be seen in Figure 4.

```
1  //pure JavaScript
2  function sum(a, b) {
3    return a + b;
4  }
5
6  //ES6
7  (a, b) => ( a + b );
```

Figure 4. Pure JavaScript function vs. ES6.

2.5 Bootstrap 4

“Bootstrap is a free and open source front end development framework for the creation of websites and web apps” /9/. It includes HTML templates, CSS templates and JavaScript to create the basic ones available: typography, forms, buttons, tables, navigation, modals, image carousels and more. In bootstrap there are additional JavaScript plugins in it, helping to design responsiveness easier and faster.

Bootstrap can be installed by copying these links into a corresponding position in an html file. Full instructions of installation can be found at https://getbootstrap.com/.

Paste the stylesheet <link> into <head> before all other stylesheets to load CSS.

```
<link rel="stylesheet" href="https://stackpath.bootstrapcdn.com/bootstrap/4.3.1/css/boot-
strap.min.css"
integrity="sha384-
ggOyR0iXCbMQv3Xipma34MD+dH/1fQ784/j6cY/iJTQUOhcWr7x9JvoRxT2MZw1T"
crossorigin="anonymous">
```

Place the following <script>s near the end of pages, right before the closing </body> tag, to enable them. jQuery must come first, then Popper.js, and then JavaScript plugins.

```
<script src="https://code.jquery.com/jquery-3.3.1.slim.min.js"
integrity="sha384-q8i/X+965DzO0rT7abK41JStQIAqVgRVzpbzo5smXKp4YfRvH+8abtTE1Pi6jizo"
crossorigin="anonymous"></script>
<script src="https://cdnjs.cloudflare.com/ajax/libs/popper.js/1.14.7/umd/popper.min.js"
integrity="sha384-UO2eT0CpHqdSJQ6hJty5KVphtPhzWj9WO1clHT-
MGa3JDZwrnQq4sF86dIHNDz0W1"
crossorigin="anonymous"></script>
<script src="https://stackpath.bootstrapcdn.com/bootstrap/4.3.1/js/bootstrap.min.js"
integrity="sha384-JjSmVgyd0p3pXB1rRibZUAYoIIy6OrQ6VrjIEaFf/nJGzIxFDsf4x0xIM+B07jRM"
crossorigin="anonymous"></script>
```

2.6 ReactJS

React (or ReactJS, React.js) is an open-source JavaScript library for building reusable interface components. It was created by Jordan Walke, a software engineer at Facebook. React was first deployed to Facebook's Newsfeed application in 2011, then deployed to Instagram.com in 2012. It was open-sourced at JSConf US in May 2013. There are a lot of big companies using React for their projects, such as Netflix, Airbnb, Twitter, for example /10/.

ReactJS allows a developer to break complex UI constructs into independent components and this idea makes ReactJS unique. Instead of worrying about the overall web application, developers now easily break down complex UI/UX structures into simpler components. This makes everything intuitive and easy to imagine compared to how it was before.

It also allows a developer to write applications directly on JavaScript. And JSX is one of the features that not only makes ReactJS easy but also more interesting. Developer can now create a new feature and can see it appear in real time, enabling Developer to directly embed HTML fragments into JavaScript /11/.

2.6.1 JSX

JSX is short form for JavaScript XML. It is a type of extension syntax for JavaScript language written in XML style. JSX provides a syntactic way for replacing the statement React.createElement() in React. The code written in JSX will be converted to JavaScript so that the browser can understand the code.

JSX has a syntax that looks like both of JavaScript and HTML. The JSX structure requires HTML tags to have a closing tag. An empty tag (no attribute) can be self-closing with ‘/>’. All HTML-like tag in JSX should be nested in a tag. The JSX uses ‘camelCase’ for the attribute in HTML-like tag, for example, in HTML there is ‘class’, in JSX we must write ‘className’. Both code blocks in Figure 5 are identical, however, the one is written with JSX is easier to read and understand.

```jsx
const element = React.createElement(
  'h1',
  { className: 'greeting' },
  'Hello, Quang Luong!'
);

//The code block above is identical with the one below

const element = (
  <h1 className="greeting">
    Hello, Quang Luong!
  </h1>
);
```

Figure 5. JSX example

2.6.2 Virtual DOM

DOM is the abbreviation for Document Object Model, which is a standard defined by W3C to access and manipulate HTML or XML code in scripting languages, JavaScript is an example. DOM helps manipulate data in an object-oriented model because elements of the DOM have defined structure into objects, methods or properties to be easily accessed. They are treated as nodes and represented as a DOM Tree.

Virtual DOM is built as an abstraction class located on the DOM. Virtual DOM provides an additional means for the DOM and events system, instead of directly impacting the DOM. More specifically, Virtual DOM is used to re-render the DOM effectively using an algorithm to find differences and only re-represent changed components. This makes the implementation extremely faster.

Instead of creating a direct change to the browser DOM model, ReactJS creates changes on a virtual DOM model (Virtual DOM). It then calculates the difference between the two DOM models, and only updates the differences for the browser DOM. This process can be seen in Figure 6, where the change was applied to an *img* element /12/.

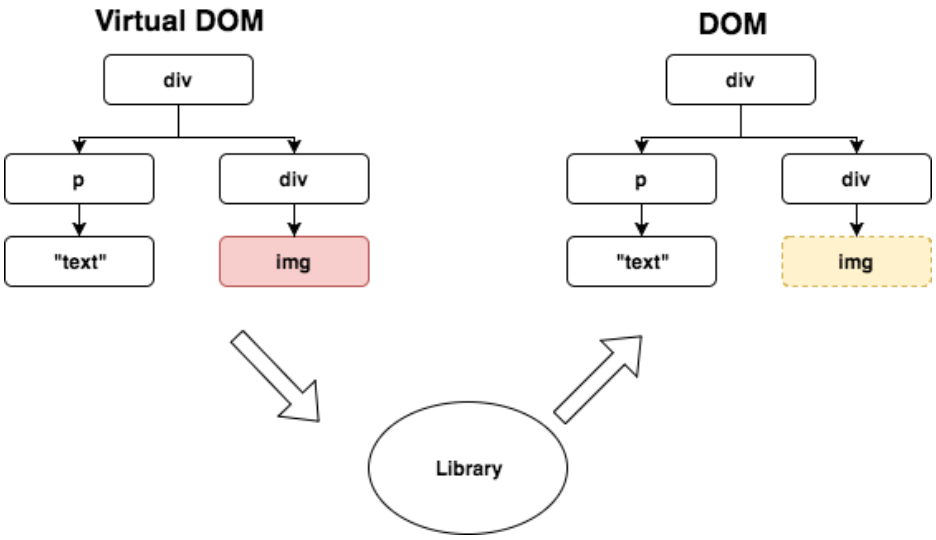

Figure 6. Updating object in ReactJS with virtual DOM

2.6.3 Components in ReactJS

Component is a very important concept in ReactJS. Components can be broken down into non-interdependent, inherited or reused. Separating components helps to manage code better, as it is easier to maintain. As shown in Figure 7, a shopping cart can be broken down into many components, which help developers reuse it. Also, it is relatively easy to fix any problems that may occur.

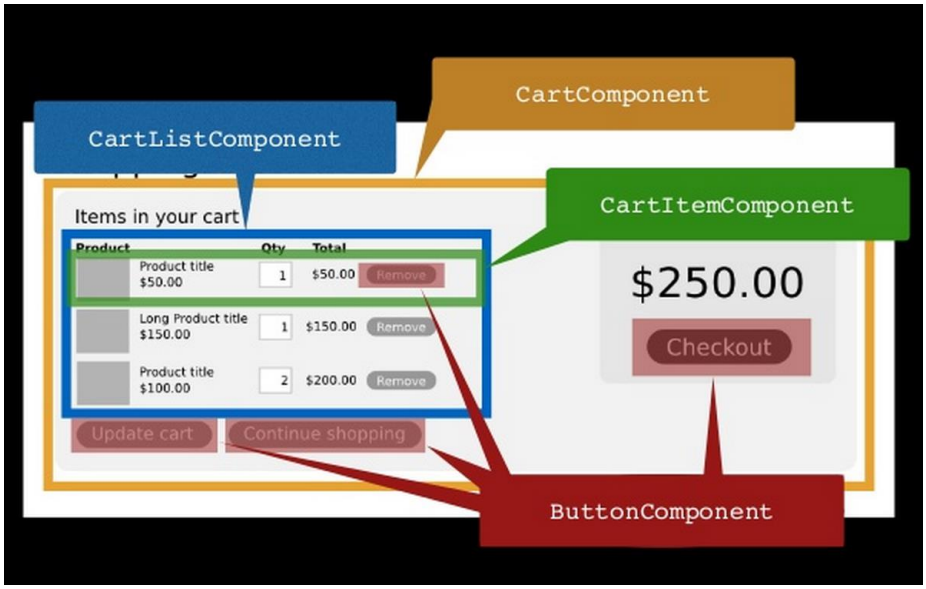

Figure 7. Components in ReactJS /13/.

In this project, header, content, footer and the navigation bar will be split into components and reuses for different template views. For example, Figure 8 shows a component in the project which is named ‘App’.

```
class App extends Component {
  render() {
    return (
      <Router>
        <div className="App">
          <Nav />

          <AppRouter />

          <Footer />
        </div>
      </Router>
    );
  }
}

export default App
```

Figure 8. Example of a component

2.6.4 Props

The props are the properties of a component. It is possible to change the component's props by transferring data from the outside (as Figure 9). When a props is transferred into component, it is invariant, which means its data cannot be changed

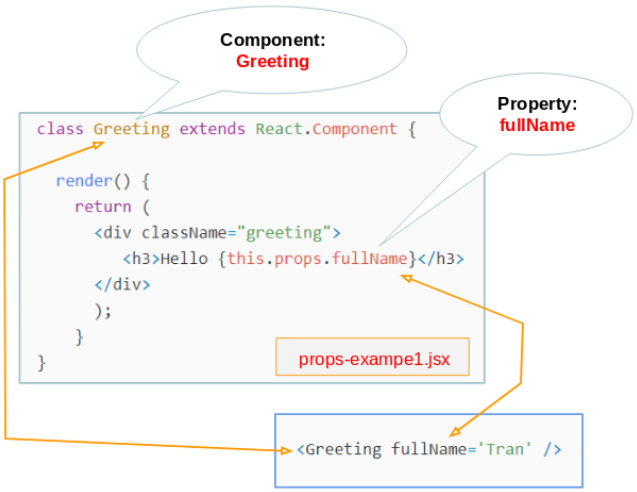

Figure 9. Props of component.

2.6.5 State

The state represents the state of the component. The state is private and can only change within the component itself. It is possible to change states by calling function this.setState(). The code in Figure 10 will create the value for MainApp component's state and use those value inside the component.

```
class MainApp extends React.Component {
  constructor(props) {
    super(props);
    this.State = {
      title: 'This is the title',
      subtitle: 'This is the subtitle'
    }
  }
  render() {
    return (
      <div>
        <h1> {this.state.title} </h1>
        <h2> {this.state.subtitle} </h2>
      </div>
    );
  }
}
export default MainApp;
```

Figure 10. State of component

2.6.6 Comparing between prop and state

Like props, state also holds information about components. However, the type of information and how to handle it are different. Figure 11 shows a comparison between prop and state.

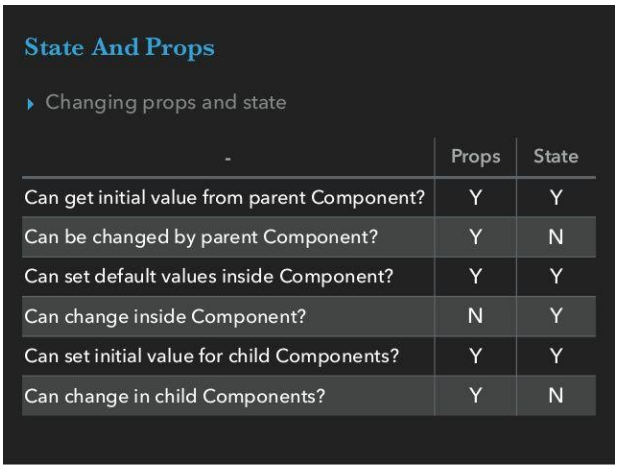

Figure 11. Prop vs. State /14/.

2.7 Redux

Redux is a predictable state management tool for JS applications. It helps you write applications that work consistently, runs in different environments (client, server, and native) and is easy to test. With Redux, the state of the application is kept in a place called store and each component can access any state they need from 'Store' instead of transfer state change through levels like ReactJS (as Figure 12).

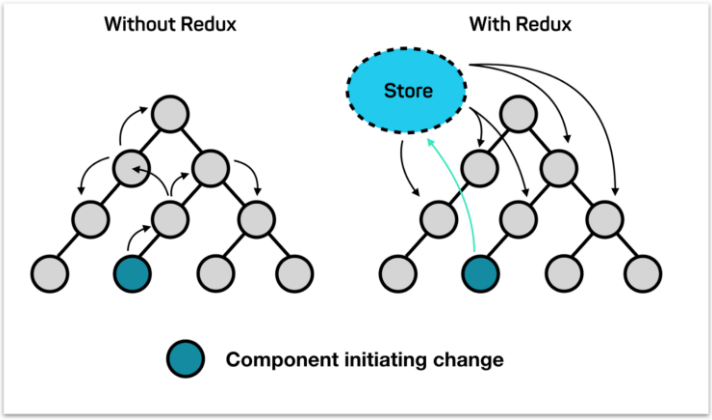

Figure 12. State changes in Redux vs. pure ReactJS /15/.

The way Redux works is quite simple. Each component can access directly to the stored state instead of having to send drop down props from one component to another. There are three components of Redux: Actions, Store, Reducers. They connect and handle the data in the whole application (Figure 13) /16/.

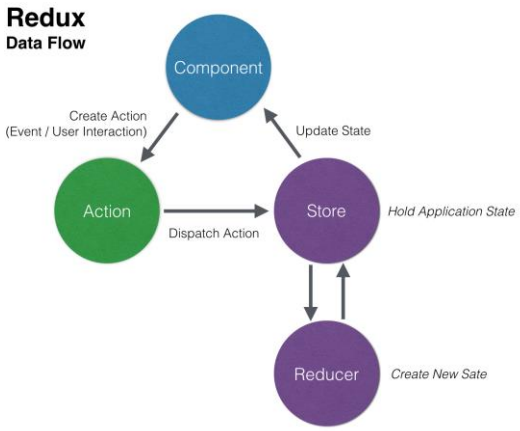

Figure 13. Data flow in Redux application /16/.

2.7.1 Actions

Actions can be explained with the word “events”. They are the way we send data from the app to Redux store. These data may be user vs. app interaction, API calls, or it may also be from the submission form.

Actions are sent using the store.dispatch() method, they must have a type property to represent the type of action to perform. They must also have a payload containing information. Actions are created through an action creator (as Figure 14).

```
const setLoginStatus = (name, password) => {
  return {
    type: "LOGIN",
    payload: {
        username: "foo",
        password: "bar"
    }
  }
}
```

Figure 14. Login Action component

2.7.2 Reducers

Reducers are primitive functions that take the current state of the app, execute an action and return a new state (as Figure 15). These states are stored as objects and they specify how the state of an application changes in responding to an action sent to the store.

```
const LoginComponent = (state = initialState, action) => {
  switch (action.type) {
    case "LOGIN":
      return state.map(user => {
        if (user.username !== action.username) {
          return user;
        }

        if (user.password == action.password) {
          return {
            ...user,
            login_status: "LOGGED IN"
          }
        }
      });

    default:
      return state;
  }
};
```

Figure 15. Login reducer component

2.7.3 Store

Store keeps application states and it is unique in any Redux application. We can access states saved with getState(), update state with dispatch(action), and register or unsubscribe listeners through subscribe(listener).

It is possible to create a store for our application with the code below

```
1 const store = createStore(LoginComponent);
```

2.8 Google Firebase

Firebase is a Google-provided backend system service for applications from mobile to Web platform. With Firebase it is possible to shorten the development time, deployment, and scaling time of our application. Firebase services give a powerful supporting to all platform with millions of users /17/.

In this project, Cloud Firestore is used as a database for the web application. Cloud Firestore is a flexible and extensible database for a mobile or web application. It is easy to synchronize data between client-side applications (Realtime) and offline data support in the application. Cloud storage is a noSQL database, all data will be given in JSON format and with this it is possible to easily access and manipulate data to the application.

Additionally, Firebase Auth service is also used, which manages users in a simple and safe way. Firebase Auth provides many methods for authentication, including email and password, third-party providers such as Google or Facebook. Our own interface can be built and this service can be implemented into our UI.

Another service that used in this project is Firebase Storage. It is a service built for the purpose of storing and managing content created by application users such as photos, videos or file data. Firebase Storage provides APIs for securely uploading and downloading files from our app.

Finally, when the development is completed, a host is needed for the deployment. Therefore, firebase hosting is used, a free hosting and domain from Firebase service. One thing needed is upload "build version" of our application to that host, then the application can be accessed by the provided domain from the Firebase service.

In this project, a node package called react-redux-firebase is also used. It offers an easy and simple method to connect the database in Firebase.

2.9 PHP function

In the created application, every time a user enrolls an event, the system will automatically send an email and a pdf file of confirmation to the user. Therefore, those features were developed with PHP mail() function for backend server.

In addition, the FPDF library was used. It is a PHP class, which allows generating dynamical content of pdf base on an event detail in our application. This file will be sent via email with an attachment to users (Figure 16). This library can be found at http://www.fpdf.org/

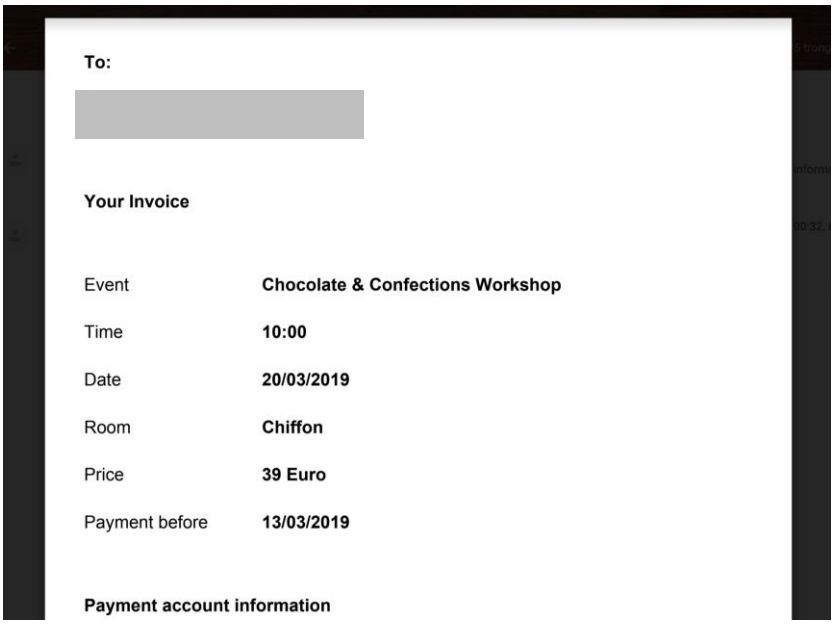

Figure 16. PDF file made by FPDF library

3 ENVIRONMENT SETUP

3.1 Installing IDE

IDE stands for Integrated Development Environment and it is a software that provides programmers with an integrated environment that includes various tools such as code writing programs or code editors as well as debug programs. In other words, IDE is software that includes other software packages to help develop software applications.

At this moment, there are many IDEs that give developers a nice environment to program and build their software. Most of them are free of cost and open source. One of those good choice for completing this thesis is Visual Studio Code, which is a software commonly used on popular operating systems today. In this project's case it is Mac OS. VSC can be installed directly by downloading setup file on website https://code.visualstudio.com/

The main reason for using VSC is the several support features it offers for the developers, including the built-in terminal system, helping developers run commands directly on VSC without having to run a terminal task on the system. Another reason for using VSC is that it has built-in GIT command, with which it is possible to review diffs, stage files, and make commits right from the editor. It is also possible to push and pull project's code from any hosted SCM service.

In addition, the VS market provides a large number of libraries of extensions and plug-ins that support coding of most programming languages today. For this project, the extensions are recommended in Figure 17 and Figure 18.

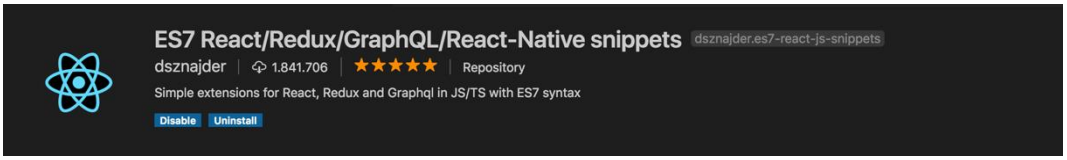

Figure 17. ReactJS snippets.

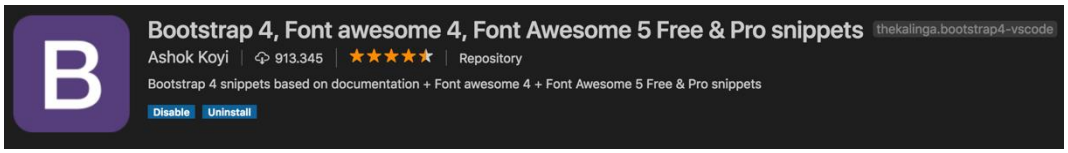

Figure 18. Bootstrap snippets.

3.2 Installing NodeJs Bundles and NPM

NodeJS is a source code based on JavaScript V8 platform and it is used to build web applications such as video clip pages, forums and especially narrow-range social networking sites. NodeJS is an open source library widely used by thousands of developers worldwide. It provides rich libraries in the form of JavaScript modules for simple programming and it reduces time used at the lowest level. One of the most important things is that NodeJS allows creating a web server so that React components can be used locally and deployed to the web directly.

NPM stands for Node Package Manager and it is a tool (program) that manages JavaScript programming libraries for Node.js. This tool is necessary for the open source world. In the JavaScript community, developers share hundreds of thousands of codes that help new projects avoid having to rewrite basic components, programming libraries, or even frameworks.

To install both, it is possible to download directly from the homepage website of NodeJS - https://nodejs.org. Once NodeJS is installed, NPM also comes with NodeJS. We can check them with these commands from Terminal (Figure 19).

```
1 > node -v
2 > npm -v
```

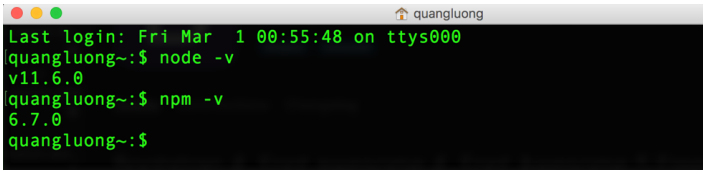

Figure 19. Screenshot of checking current nodeJS and npm version.

3.3 Create React App

Create-react-app is an official tool for generating React starter project provided by Facebook developers. It helps to save a lot of time consumption from setup and configuration (no need to configure web packs, node modules, dependencies, etc.). The command only needs to be run to install the tool start our React project.

First, install the global package, open the Terminal (or Command line in Windows) and run the following command

```
1 npm install -g create-react-app
```

Then move to working directory with Terminal and run the command

```
1 create-react-app my-app
```

Wait a couple of minutes while the tool generates everything needed for React project. After executing the command, we get a project with the structure in Figure 20. my-app's files structure created by create-react-app. Then the project can be started by changing code and structure inside of a “src” folder.

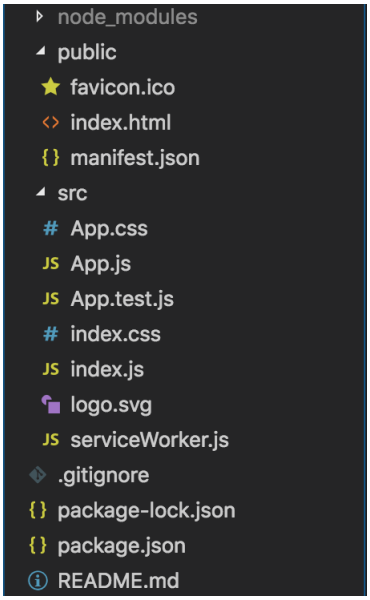

Figure 20. my-app's files structure created by create-react-app

3.4 Installing dependencies

Some webpack is needed for the development process. Webpack is a tool to help build the project easily, which can be called as “dependencies”. For this project, these dependencies are used:

```
"axios": "^0.18.0",
"firebase": "^5.3.0",
"react": "^16.4.1",
"react-dom": "^16.4.1",
"react-modal": "^3.8.1",
"react-redux": "^5.0.7",
"react-redux-firebase": "^2.1.6",
"react-router": "^4.3.1",
"react-router-dom": "^4.3.1",
"react-scripts": "^2.1.5",
"redux": "^4.0.0",
"redux-firestore": "^0.5.7",
"redux-thunk": "^2.3.0"
```

The application needs to be navigated by links for corresponding parts, we can define our route with *react-router* and *react-router-dom*. As mentioned before, a tool for managing all states of our component is needed, so *redux* and *react-redux* are used in this case. For accessing and using data from Firebase, *firebase*, *react-redux-firebase* and *redux-firestore* need to be installed. Finally, *redux-thunk* package also installed for supporting the usage of redux in the application.

One of helpful packages which needs to be installed is *axios*. This will help make a connection between the client and server for handling backend services.

Those can be installed with the following code:

```
1 npm install -save {name of dependencies}
```

4 IMPLEMENTATION

The application will be divided into two smaller parts, one of them is the homepage, which will show information to the user and the other one is the admin dashboard which provides a CMS for managing the activities of team.

4.1 Homepage

4.1.1 User Interface

For the homepage, different templates, which show corresponding content to users are needed. First of all, a homepage (Figure 21) which will show information about the team, and upcoming events and archive events in the past is needed.

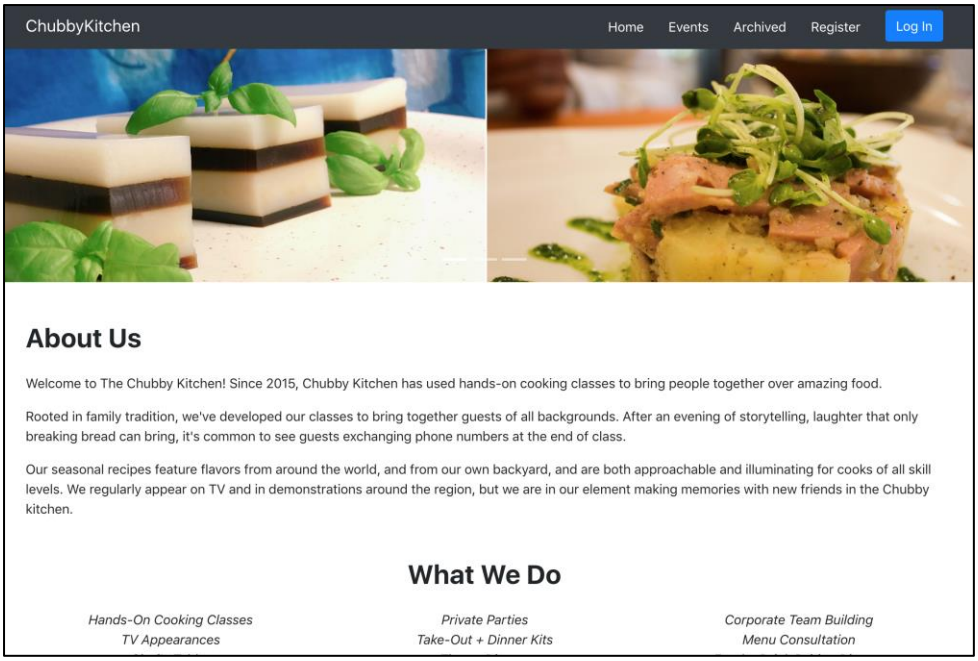

Figure 21. Application's homepage

Then, Events templates which shows a list of all event in the future is needed (Figure 22). An Archived template, which shows all archive event the team held in the past is needed.

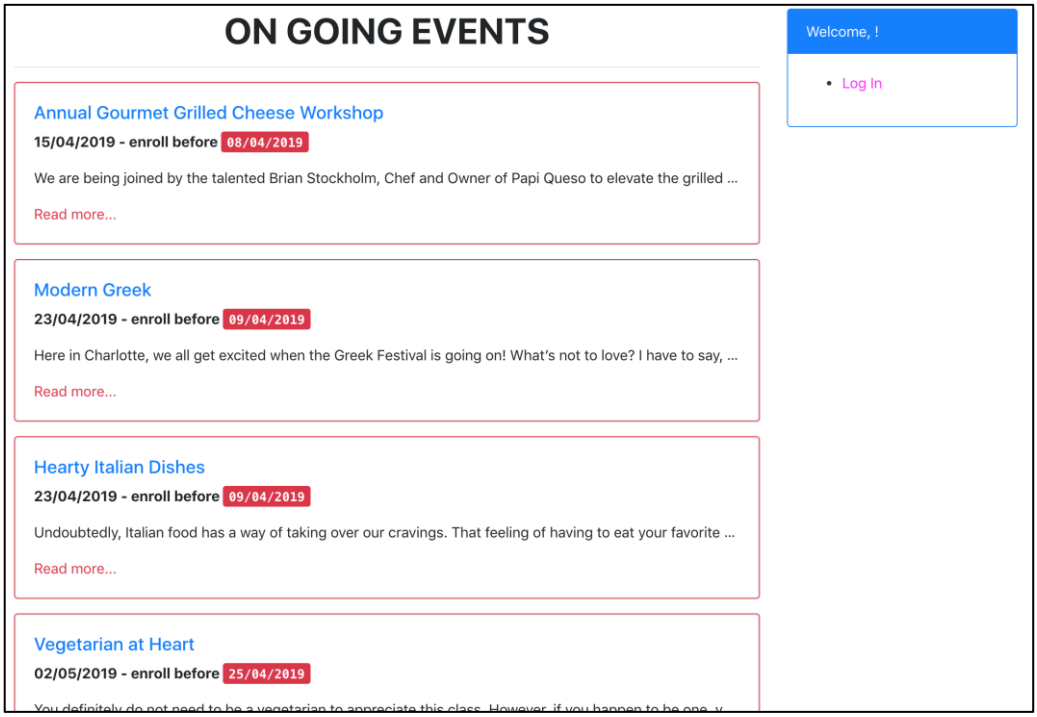

Figure 22. Events list

For each event, there is a need to list all of details to user and therefore another template which shows details dynamically based on the event as shown in Figure 23 is needed.

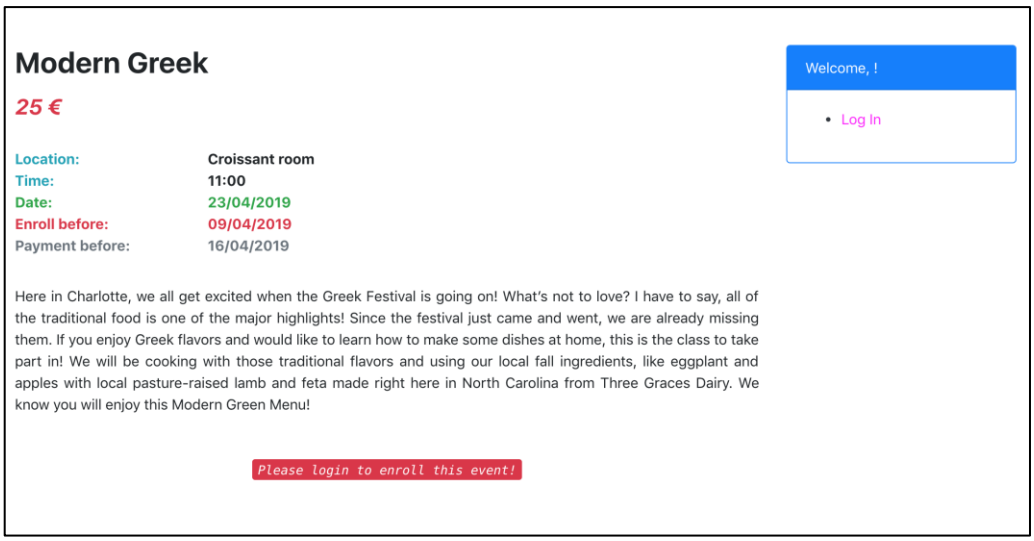

Figure 23. Event's detail view

4.1.2 File structure

In this chapter, the file structure of the conducted project will be defined by dividing component source files into multiple parts and placing them in corresponding folders. To do this a ReactJS project needs to be created (see section 3.3). Once the initial project is created, the "src" folder then created will be moved with some more subfolders for storing app components.

First, a subfolder named "components" is used for storing all the components in the application. The application can be divided into several small sections which are Header, Footer, Event, etc. In other words, they are the main UI of the application.

Then, a new JS file is created in which how to connect to the firestore database and fetch the data from there is defined. This file will be placed in a subfolder named "firebaseDB" for easier to maintain in the future.

As mentioned before, the created application is a single page application, and which template view is shown to the user by route them in exactly URL must be defined (see more in 4.1.3). Similarly to the database connection, this router file is placed in a subfolder named "RouterURL" so it can be managed or edited more conveniently.

Finally, the application used Redux library, so three main components, which are Store, Action and Reducer, in a subfolder named "Store" need to be defined. As a result, there is a project with a file structure shown in Figure 24.

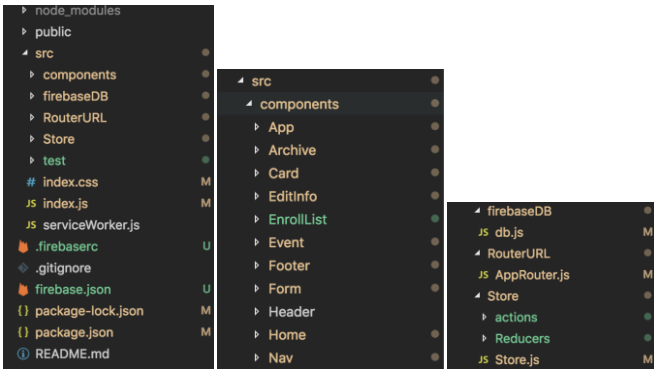

Figure 24. Homepage project file structure

4.1.3 Router

In this file, it is defined how the app created is divided into separate sections by route (Figure 25). This mean that the component views will be shown to users by the matching route they enter in the address bar. Each time user enters a route, they will be connected to the content they want to see.

```
<Switch>
  <Route exact path="/" component={Home} />
  <Route exact path="/events" component={Event} />
  <Route exact path="/archived" component={Archive} />
  <Route path="/log-in" component={LogIn} />
  <Route path="/register" component={Register} />
  <Route path="/edit-user-info" component={EditInfo} />
  <Route path="/events/:id/:slug" component={EventDetail} />
  <Route path="/archived/:id/:slug" component={ArchiveDetail} />
  <Route path="/enrollment" component={EnrollList} />
  <Route component={Home} />
</Switch>
```

Figure 25. App's route list

4.2 Admin Dashboard

4.2.1 User Interface

In the admin dashboard, several template views are needed for the admin users who can access and manage all activities of the association. First, it needs to be made sure that only admin user can access this page, so it needs to check that every time a user enters the site by a login screen (Figure 26). If he or she is an admin, the app will let he/she log in, otherwise the app will deny access.

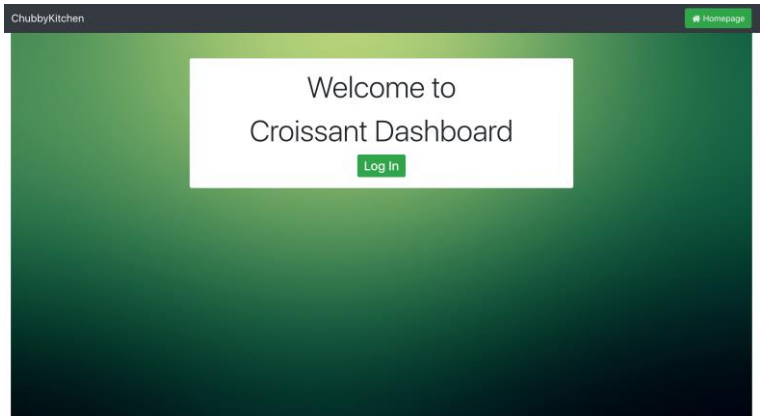

Figure 26. Admin dashboard welcome page

The CMS will be divided into three main sections which will let the admin manage the events, users and archives list, get stats of the app activities, and add, edit or delete anything. There will be three templates for each (an example in Figure 27).

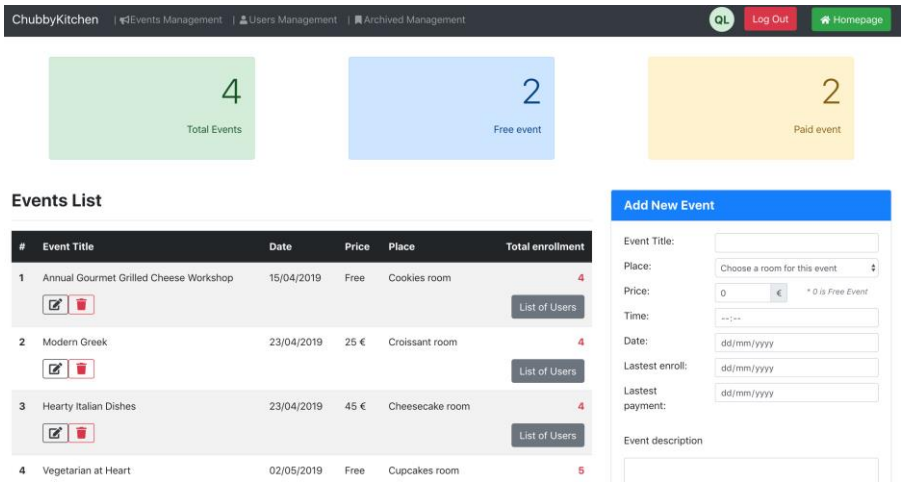

Figure 27. Events management views

Admin users also add, update or delete a user in the "User management". With this tool, the admin can update a user type account to an admin account or edit information (Figure 28) for any account.

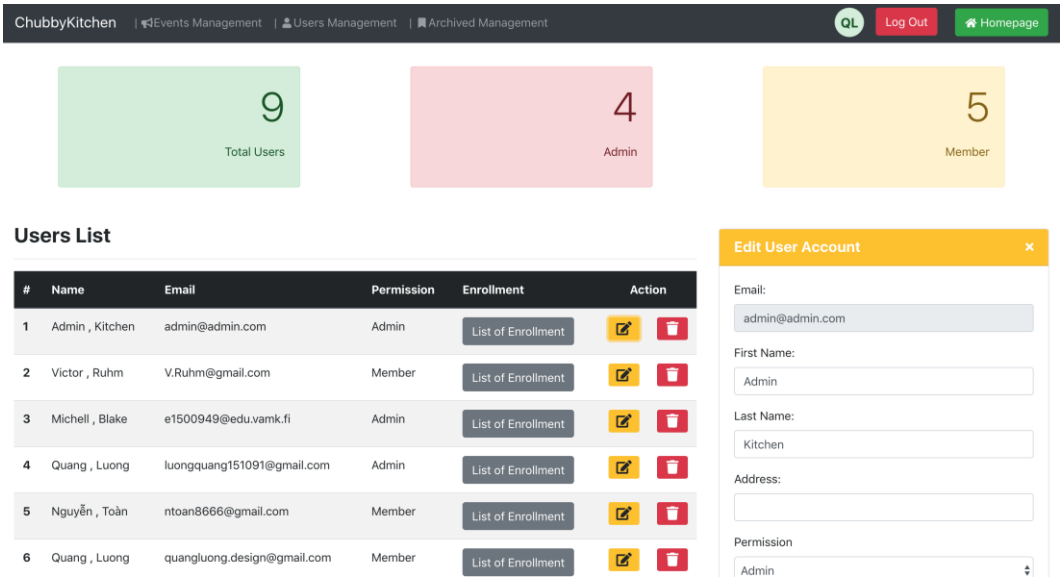

Figure 28. User management

For each event, the admin wants to get a list of all users who enrolled an event, so a page for that task needs to be built (as Figure 29).

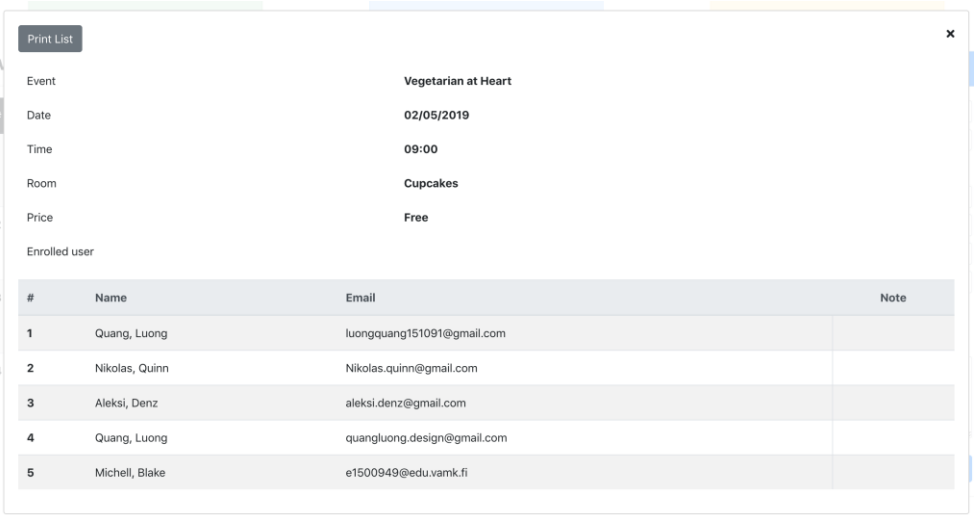

Figure 29. Example of an event's list users

4.2.2 File structure

In the manner of Homepage, a ReactJS project was also created and divided into smaller files and subfolder. This structure is shown in Figure 30.

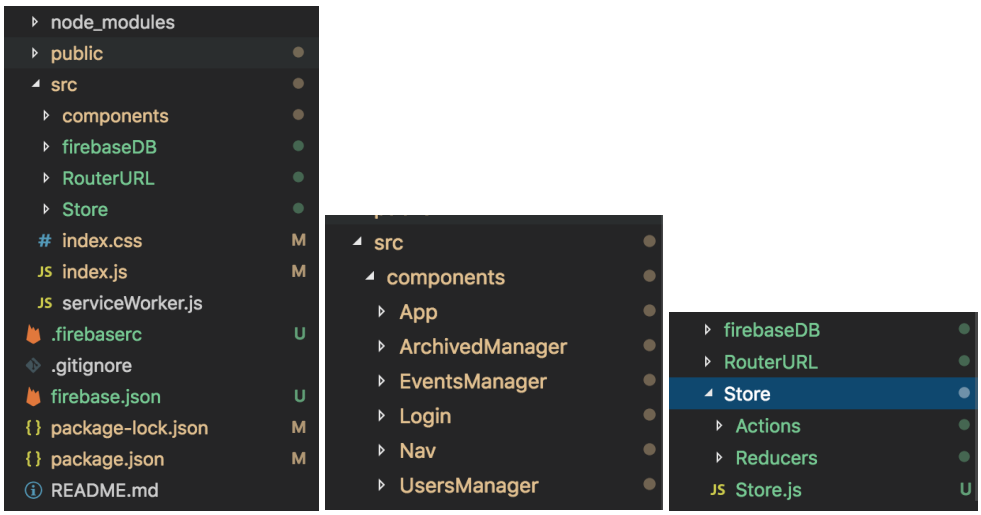

Figure 30. Dashboard project file structure

4.2.3 Router

In the dashboard, its feature needs to be navigated to the corresponding URL as in Figure 31. This is similar to the Router in the Homepage

```
<Switch>
  <Route exact path="/" component={WelcomePage} />
  <Route path="/log-in" component={Login} />
  <Route path="/event-manager" component={EventsManager} />
  <Route path="/user-manager" component={UserManager} />
  <Route path="/archive-manager" component={ArchivedManager} />
  <Route component={WelcomePage} />
</Switch>
```

Figure 31. Dashboard's route list

4.3 User registration

The admin of Chubby Kitchen team wants to let their customers register as users when they use the app, so a register function for the application is possible to build by using Google Firebase Auth function. First of all, an UI for user who want to register a new account in our app is needed (Figure 32). The register form is quite simple with some input fields and a submit button, this form can be created with a bootstrap 4 form template.

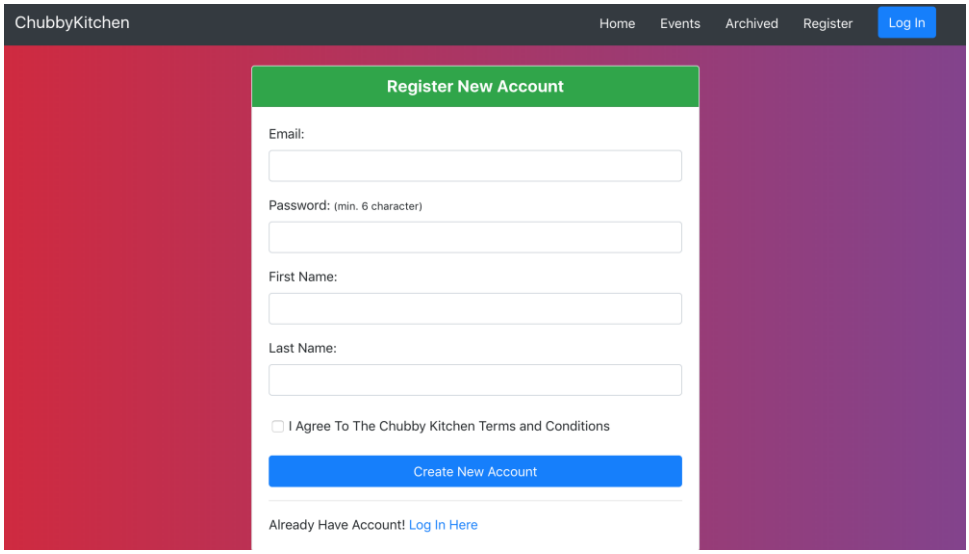

Figure 32. User registration form

Next, we need to connect this form with redux action so that every time a new user submits his/her information, the application will send their input to the database for creating a new account and store in the user database. The following firebase's API will help in creating a new account then add the new user info to database

```
firebase.auth().createUserWithEmailAndPassword(
    newUser.email,
    newUser.password
  )
```

```
firestore.collection('userList').doc(resp.user.uid).set({
      firstName: '',
      lastName: '',
      email: newUser.email,
      initials: '',
      address: '',
      role: newUser.role
    })
```

4.4 User Login / Logout

The following Firebase's API will get the user's email and password from a login form when the user types in there and sends the request directly to our Firebase backend server, then the server will check and let the user log in.

```
firebase.auth().signInWithEmailAndPassword(
    credential.email,
    credential.password
  )
```

The log out function can be done with the following API

```
firebase.auth().signOut().
```

The Admin Dashboard contains much sensitive information and for security reasons, every time the admin leaves the dashboard (closes the tab or clicks an external link to another page), the CMS will automatically log the admin out, this can be

done by setting a SESSION for admin login in the dashboard. Little change in the code of dashboard login had to be made:

```
firebase.auth()
          .setPersistence(firebase.auth.Auth.Persistence.SESSION)
    .then(function() {
      return firebase.auth().signInWithEmailAndPassword(
        credential.email,
        credential.password
      )
    }
```

Another thing that has to be done is to prevent any user account from accessing to admin dashboard without permission. We want to make sure that only admin account can access the dashboard, so after login and before rendering any template view, we have to check if the account is an admin by the code.

```
const { auth } = this.props;
  if(!auth.uid) {
    return <Redirect to="/" />
  }
```

If it is not an admin account, the dashboard will redirect the user to the welcome page and user cannot interact with any dashboard feature. The result for this can be seen in Figure 33.

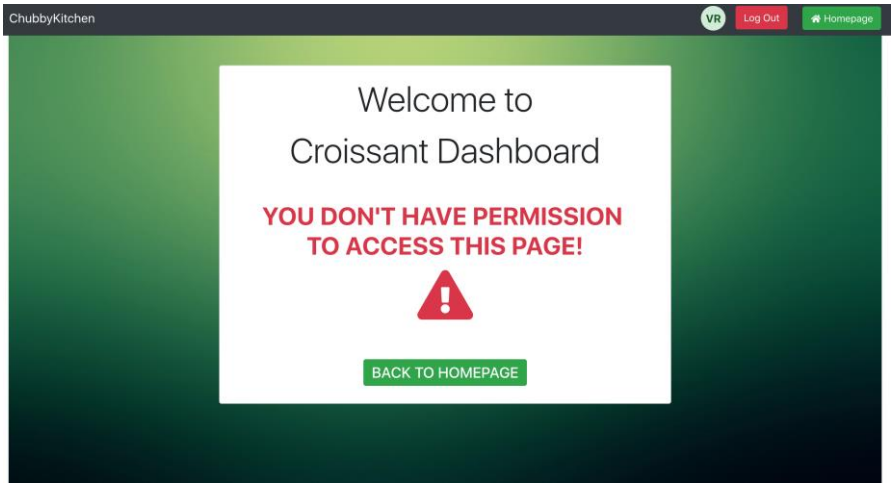

Figure 33. Dashboard's login error with an 'user' account

4.5 Manipulate data from database

With the helping of *react-redux-firebase*, the data from database can be get and then show to the user easily by the code. Here the code is placed within *export* function so that the data fetched from database will always render with the component.

```
export default compose(
  connect(mapStateToProps),
  firestoreConnect([
    { collection : 'eventList',
      orderBy: ['date', 'asc'],
      startAt: (new Date()).toISOString()
    }
  ])
)(EventList);
```

The function *firestoreConnect* is used for establishing a connection to the database then point out which database we want to get the data from, in this case it is "eventList", and data is sorted in order and other conditions.

4.6 Sending confirmation email

The created application needs a connection to the server for handling the email sending. In section 3.4, *"axios"* package was installed, this package will help in making a connection between the app and server side, it also sends all needed data to the sever for handling email content and making a pdf file of confirmation invoice.

To do this, the *"axios"* package first need to be imported, then the API path for email sending function is defined before creating the sending method.

```
import axios from 'axios';

const API_PATH = 'https://www.some-server.fi/php
                          /chubbykitchen/enrollConfirm.php';
```

Then a method is called to let *axios* know what to do every time a user clicks the enroll button to any event by this code so the data will be sent to server by POST method with type of JSON data. Then the server can get all information and handle the email sending request with PHP mail function and PDF maker library

```
axios({
    method: 'post',
    url: `${API_PATH}`,
    headers: { 'content-type': 'application/json' },
    data: enrollment
  })
```

4.6.1 PDF file creation

Here the library was used to help make the pdf file of the invoice. The content of the pdf file is based on information axios send to sever. The pdf function executed the code below to make a file and then attach it to email as attachment before sending it to the user.

```
require('fpdf/html_table.php');
$pdf=new PDF();
$pdf->AddPage();
$pdf->SetFont('Arial','',14);
//here we set the content of file
$html='<table border="0">
<tr>
<td colspan="2" height="30"><b>To:</b></td>
</tr>
<tr>
<td colspan="2" height="10"></td>
</tr>
...
$pdf->WriteHTML($html); //here we write the $html to file
$pdf->Output();
$pdfdoc = $pdf->Output("", "S"); //output as a file
$filename = "Invoice.pdf"; //give file name for our pdf file
//attach pdf file with base 64 encode
$attachment = chunk_split(base64_encode($pdfdoc));
```

4.6.2 PHP mail() function

The email of confirmation can be sent to a user with the function mail() of PHP language. In this function, the email we want to send to must be defined, the subject of email, the email content and, hidden header field for email must be also defined.

```
mail($to, $subject, $body, $headers);
```

Those data can be obtained with a JSON object which was be sent by axios before, to get the data we executed the code:

```
$rest_json = file_get_contents("php://input");
$_POST = json_decode($rest_json, true);
```

Until now, all data we need is put in $_POST variable, a multidimensional array, we just grab them from there and use in the code. For example:

```
$email = $_POST['userInfo']['email'];
$firstName = $_POST['userInfo']['firstName'];
$lastName = $_POST['userInfo']['lastName'];
$eventTitle = $_POST['eventInfo']['title'];
```

4.6.3 Setup CORS for PHP API

Cross origin resources sharing is a mechanism that uses HTTP headers to interact with the browser, allowing a web application to run from a domain A to have access to domain B (different origin) resources.

A request to a resource outside the scope of that Website is called a "*Cross origin request*".

CORS requires servers to implement and grant access resources for specific domains and resources corresponding to that domain. If access is outside the scope specified by the server, that request will be blocked.

Therefore, CORS for our API must be setup to allow the app to use a PHP function for handling email sending. CORS can be set up by the code below in PHP file.

```
header("Access-Control-Allow-Origin: *");
header("Access-Control-Allow-Headers: Content-Type, origin");
```

The symbol “*” means that our API accepts any access request from our application.

5 DEPLOYMENT AND TESTING

When the implementation phase is completed, the created application was deployed to firebase hosting and how it works with different browsers was tested.

The app was tested with different browsers and the app worked perfectly well with Internet Explorer, Firefox and Google Chrome. It responded quickly and did not delay any action. The user interface was displayed nicely on all the browsers and all features worked well.

6 CONCLUSIONS

JavaScript is now the most popular and common programming language all over the world. One of main advantages is that JS offers versatile and speedy performance in any application. Furthermore, the compatibility with many modern browsers and devices makes JS is the good choice for many projects.

In addition, many frameworks of JS are developing day by day; they have created a revolution in the field of app and web-based app development. When a framework comes with additional functions support for associate with back-end, it will become more powerful. This also gives ReactJS a role to play among all frameworks available in the market.

Nowadays, ReactJS has been used much due to being compact and easy to learn and use. After developing the application, it is evident that ReactJS quite simple, everyone can learn it easily. Due to its compact and good performance, it was a good choice for the created application. Everything needed is run a command to initialize a project then start by changing the code.

The main purpose of this thesis was to apply the ReactJS framework to give an advanced technique building up a web application. Furthermore, some of modern technologies were used to help the development in a convenient way. Redux is a wonderful library to work with when using ReactJS. It is also quite simple and the work flow is easy to understand. In addition, Firebase from Google is a good choice among variety of server-side technologies that are on the market today. It provided multi-platform support to help save time in management as well its stability.

The application in this thesis is just simple project, which can be developed more in the future if there is a chance to work with it again.

REFERENCES

/1/ Market share & web usage statistics ReactJS. Accessed 27 March 2019 https://www.similartech.com/technologies/react-js

/2/ Software development process. Accessed 01 March 2019. https://en.wikipedia.org/wiki/Software_development_process

/3/ Single-page vs. multiple-page application. Accessed 01 March 2019. https://medium.com/@NeotericEU/single-page-application-vs-multiple-page-application-2591588efe58

/4/ Single Page Applications – Why they make sense. Accessed 02 March 2019. http://www.digitalclaritygroup.com/single-page-application-make-sense/

/5/ HTML5. Accessed 02 March 2019. https://en.wikipedia.org/wiki/HTML5

/6/ HTML5 Semantic Tags. Accessed 02 March 2019. https://www.vikingcodeschool.com/html5-and-css3/html5-semantic-tags

/7/ Usage of client-side programming languages for websites. Access 27 March 2019. https://w3techs.com/technologies/overview/client_side_language/all

/8/ JavaScript. Accessed 05 March 2019. https://en.wikipedia.org/wiki/JavaScript

/9/ What is Bootstrap. Accessed 05 March 2019. https://whatis.techtarget.com/definition/bootstrap

/10/ React (JavaScript library). Accessed 07 March 2019. https://en.wikipedia.org/wiki/React_(JavaScript_library)

/11/ What is React.js and how does it work?. Accessed 27 March 2019. https://hub.packtpub.com/what-is-react-js-how-does-it-work/

/12/ React virtual DOM explained in simple English. Accessed 27 March 2019. https://programmingwithmosh.com/react/react-virtual-dom-explained/

/13/ React JS: The new kid on the block. Accessed 27 March 2019. https://www.tivix.com/blog/react-js-the-new-kid-on-the-block

/14/ React for Dummies. Accessed 27 March 2019. https://www.slideshare.net/mitchbox/react-for-dummies

/15/ Restate—the story of Redux Tree. Accessed 10 March 2019. https://hackernoon.com/restate-the-story-of-redux-tree-27d8c5d1040a

/16/ Introduction To State Management With React. Accessed 10 March 2019. https://medium.com/codingthesmartway-com-blog/learn-redux-introduction-to-state-management-with-react-b87bc570b12a

/17/ Introduction to Firebase. Accessed 12 March 2019. https://hackernoon.com/introduction-to-firebase-218a23186cd7