OULU UNIVERSITY OF
APPLIED SCIENCES

Anup Satyal

DESIGNING AND DEVELOPING A WEBSITE WITH REACTJS

Progressive Web Application

DESIGNING AND DEVELOPING A WEBSITE WITH REACTJS

Progressive Web Application

Anup Satyal
Bachelor's Thesis
Spring 2020
Information Technology
Oulu University of Applied Sciences

ABSTRACT

Oulu University of Applied Sciences
Degree Programme in Information Technology,

Author: Anup Satyal
Title of the bachelor's thesis: Designing and Developing a Website with ReactJS
Supervisor: Lasse Haverinen
Term and year of completion: Spring 2020 Number of pages: 44

The main objective of this bachelor's thesis was to design and develop a user-friendly responsive website for a restaurant. Also, this thesis was motivated by an urge to learn and implement the Progressive Web Application.
The project was assigned by "Ravintola Sargam", which is a Nepalese restaurant located in Espoo, Finland.

The Website was built using ReactJS along with Redux state management. Node.js and Express.js were used for back-end programming. The progressive web application was implemented using service worker and a manifest file.

As a result, the website was developed fulfilling all the requirements and handed over to the client. The developed website is online as an official website for "Ravintola Sargam".

Keywords: ReactJS, Node.js, Express.js, Redux, Progressive Web Application, Service Worker, Ravintola Sargam

CONTENTS

VOCABULARY

API - Application Programming Interface

Apps - Applications

CSS - Cascading Style Sheets

DOM - Document Object Model

HTML - Hypertext Markup Language

HTTP - Hypertext Transfer Protocol

JS - JavaScript

JSON - JavaScript Object Notation

JSX - JavaScript Syntax Extension

npm - Node Package Manager

PWA - Progressive Web Application

RestAPI - Representational State Transfer API

UI - User Interface

URL - Uniform Resource Location

VS Code - Visual Studio Code

1 INTRODUCTION

In this modern era of web technology, every day a new technology for web development is being introduced. The web developers need to keep updated on these new technologies and experiment with them. ReactJS being one of those, it is no surprise that it is conquering the codebase. ReactJS is an open-source JavaScript library for the frontend development. Users these days want to use the faster and more dynamic webpages, while the developers want to use the modern and flexible developing environment, without a lot of boilerplate in the package. Thus, ReactJS is gaining its popularity among the community of front-end developers. [1]

The main aim of this thesis was to learn and implement ReactJS to develop a website for a restaurant “Ravintola Sargam”. An urge to learn and implement a Progressive Web Application to the website has added extra motivation.

“Ravintola Sargam” is an average size Nepalese restaurant located in Espoo, Finland. They tend to serve genuine Nepalese dishes with proper service quality and provide a friendly environment to the customers. Lunch is served at a special price and the dinner is served with a wide range of food selection. They provide a selection for starters, desserts, drinks at a reasonable price.

Having a proper website gains a massive audience with online visibility. Customers prefer to know about the food and service quality, which the restaurant provides, before visiting the place. According to the Constant Contact survey, “75% of consumers often choose a restaurant to dine at based on search results“ [2].

This thesis report provides detailed information about the technologies used and the implementation process for the development of the website.

2 WEB DEVELOPMENT TECHNOLOGIES USED

Web development refers to the creation of either static or dynamic web page or web application. Different technologies are used for creating web applications. The process of web development consists of two parts, Front-End and Back-End.

2.1 Front-End

Front-End, also known as the client-side, refers to the user interface of the web page. It is the appearance of the webpage as viewed by the users. The following section describes the primary front-end web technologies.

2.1.1 HTML

HTML stands for Hypertext Markup Language. HTML describes the webpage structure. HTML consists of various elements and is represented with Tags. These elements describe how the content should be displayed on the browser. Example of the HTML tag is “head”, “body”, “table”. These tags are enclosed in <>.

```
<!DOCTYPE html>
<html>
<head>
    <title>HTML File</title>
</head>
<body>
    <h3>This is a HTML File</h3>
</body>
</html>
```

FIGURE 1. The HTML document

The basic HTML document structure is shown in figure 1.

2.1.2 CSS

CSS stands for “Cascading Style Sheets”. It is used to style the HTML document. CSS can be injected into the HTML element, defined inside a HTML file or externally as a (.css) extension.

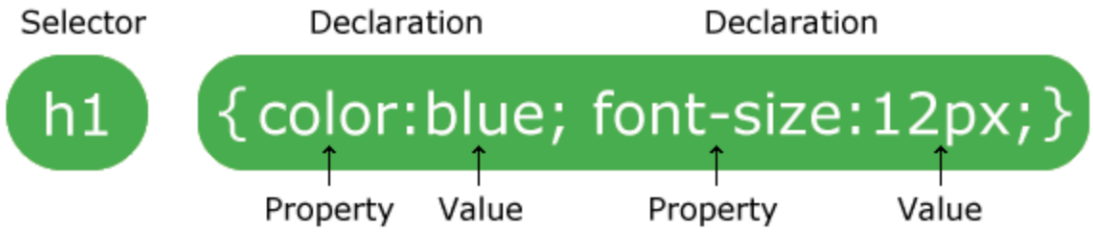

FIGURE 2. The basic CSS Syntax [3]

The basic CSS syntax consists of a selector and a declaration block as shown in figure 2. The selector is the HTML element, which one wishes to style. One or more declarations are separated by semicolons (;), which include a CSS property name and a value separated by a colon (:). CSS declaration blocks are wrapped by curly braces. [3]

2.1.3 JavaScript

JavaScript (JS) is an object-oriented programming language also best known of the scripting languages for web pages. JavaScript is used for both client-side and server-side programming. JavaScript allows the developers to create e.g. dynamically updating content, multimedia control and animations.

JavaScript along with HTML and CSS are the main technologies of the World Wide Web (WWW). These three layers are built on top of one another nicely.

```
<!DOCTYPE html>
<html>
<head>
<script>
function myFunction() {
  document.getElementById("demo").innerHTML = "Paragraph changed.";
}
</script>
</head>
<body>

<h2>JavaScript in Head</h2>

<p id="demo">A Paragraph.</p>

<button type="button" onclick="myFunction()">Try it</button>

</body>
</html>
```

FIGURE 3. The use of JS in HTML [5]

As shown in the figure 3, The JS is injected into the HTML file with a <script> tag. As shown in the figure 4 below, the JS code helps to change the paragraph text when the user presses the “Try it” button.

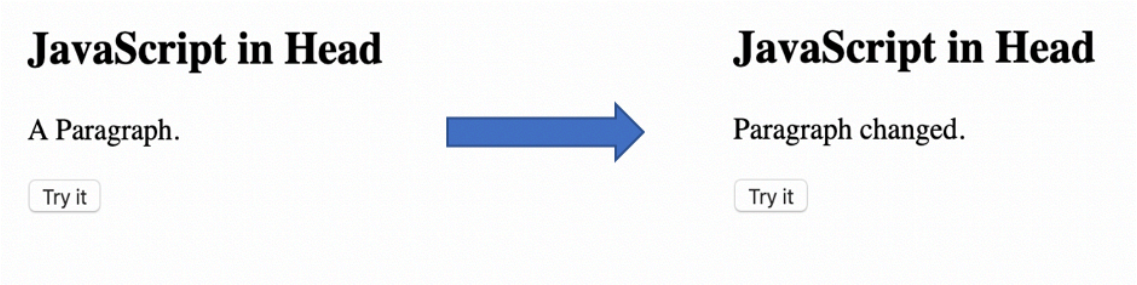

FIGURE 4. The implementation of JavaScript as seen in a browser [5]

2.2 ReactJS

ReactJS, also known as React or React.js, is an open-source JavaScript library for frontend development. ReactJS is used to create an interactive user interface and it is popular among the developer community. Because of its simplicity, native approach, data binding, performance and testability, ReactJS is often chosen over its other competitive frameworks.

History of ReactJS

Compared to other JS libraries and technologies, ReactJS is a new technology on the market. Jordan Walke, a software engineer at Facebook created this library in 2011 [10]. It was open sourced by Facebook and it is currently maintained by Facebook, Instagram and other community developers interested in the library. ReactJS is also influenced by the concept of Component, such as PHP using XHP as an HTML component framework.

ReactJS firstly came into use when the developers implemented it on Facebook's News Feed and later Instagram picked it to use it in their system. Ever since ReactJS has grown to become one of the most used open-source JavaScript libraries. [10]

2.2.1 React Virtual DOM

The most important part of web application development is DOM manipulation. Unfortunately, this process slow and time consuming in most JavaScript operations. The main reason for it is that the JavaScript frameworks try to update the DOM more than needed [11]. For example, if the list of ten items is created and only one of them is updated, then the most JavaScript framework will rebuild the entire list. However, to solve this problem, React Virtual DOM was introduced. [11].

In ReactJS the Virtual DOM is an in-memory representation of a real DOM. Manipulating browser DOM objects is much slower than the React virtual DOM because nothing becomes drawn onscreen in the virtual DOM. [11]

2.2.2 React Components

Components are very important in React. Components are small reusable User Interface elements, that provide data for the view. ReactJS enables one to create smaller components and combine them, nest them inside one another to form an entire User Interface. Components take input, also known as props and return react elements and these elements describe the User Interface contents. React components can also be written as a “function component”. They are similar to JavaScript functions. [12]

```
import React, { Component } from "react";

class Example extends Component {
    render(){
        return(
         <div>
            <h1> This is a react component example. </h1>
            <h3>Example text</h3>
         </div>
        );
    }
}
export default Example;
```

FIGURE 5. The ReactJS Component example

In Figure 5, the code imports React and the Component from the react module. A React Component with class “Example” is created and it renders and returns the with <h1> and <h3> as shown in figure 6.

This is a react component example.

Example text

FIGURE 6. Rendered Component in a browser

Using some sort of state management system, such as “Redux” in ReactJS, provides the benefit of changing components by changing the data in a global store. With Redux, it is easier to inspect changes in the application and states of it, so it helps developers to find the errors easily [13]. The following section explains the concept of Redux and how it is used.

2.2.3 Redux

Redux is an open-source library for managing the state of an application. Redux can be used with any framework, such as ReactJS, AngularJS [36], Ember [37]. However, often ReactJS and Redux are used together.

React Redux is the official Redux user interface binding library for ReactJS. If both ReactJS and Redux are used together, then it is necessary to bind both libraries using React-Redux. React-Redux helps to create a good react architecture by allowing the developers to split large functions or components into the smaller ones, which will do the specific task. React-Redux also helps to optimize performance. For example, if any update is made to the component, then ReactJS by default will re-render the whole component tree. However, React-Redux implements the optimizations system internally and the components are re-rendered only when needed. [14]

Three main foundations of Redux are

1. Store
   The store in Redux holds all the application states. If the application state is located in one place (store). It is easier to read, code, debug and test the application.
2. Actions
   Actions are the plain JavaScript objects that describe what had happened. Actions send the data from the application to the store and are only source that provides the information to the store.

3. Reducers

   Reducers explains how the application's state changes given the actions sent to the store. Reducers return the next state of the application by taking the current state and the actions.

FIGURE 7. The Redux cycle [15]

As shown in the figure 7, when the user dispatches an action, the reducer function returns the next state by taking the dispatched action and the current state. Redux then notifies the view and the update is carried out accordingly. *[15]*

2.3 Back-End

Back-End, also known as the server-side, refers to the internal working part of the web application. It holds and is responsible for all the logic, functions, calculations and databases. Back-end code runs on a server as opposed to the client-side. Python, PHP, C#, and Node.js are some of the well-known back-end technologies.

2.3.1 Node.js

Node.js is an open-source, JavaScript runtime environment used for executing JavaScript code on a server. It is designed to build scalable network applications as it is capable of handling a large number of simultaneous connections with a high throughput. Node.js is used to build an application which requires a connection from the browser to the server persistently [6].

Node.js is often the first choice of software developers as it provides features, such as asynchronous and event-driven, fast code execution, single-threaded model and no buffering. In Node.js, JavaScript is used to build the application as well as RESTAPIs, unifying all the codebase under a single programming language for both front-end and back-end. However, it is not recommended to use Node.js to build CPU intensive applications.

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

const hostname = '127.0.0.1';
const port = 3000;

const server = http.createServer((req, res) => {
  res.statusCode = 200;
  res.setHeader('Content-Type', 'text/plain');
  res.end('Hello World\n');
});

server.listen(port, hostname, () => {
  console.log(`Server running at http://${hostname}:${port}/`);
});
```

FIGURE 8. An example of HTTP server with Node.js [6]

In figure 8 above, the Node.js code returns the text “Hello World” on the local server (port 3000).

2.3.2 Express.js

Express is a flexible Node.js web application framework. Express.js provides additional features that make the web application development fast and easy compared to using just Node.js. It provides a robust set of features for web and mobile applications. With countless HTTP utility methods and middleware, it is easy to create Robust APIs.

```
const express = require('express' 4.17.1 )
const app = express()
const port = 3000

app.get('/', (req, res) => res.send('Hello World!'))

app.listen(port, () => console.log(`Example app listening on port ${port}!`))
```

FIGURE 9. An example of Express App which returns "Hello world" text [7]

2.3.3 RESTful APIs

An API stands for "Application Program Interface". It is a set of protocols, routines and tools for developing the web applications [8]. API describes how the web components should interact with each other. A good API provides all the building blocks for developing the application and a functional application can be developed once the developer puts these blocks together. [8]

REST is the acronym of Representational State Transfer. A RESTful API is an API that uses HTTP request. RESTful API is established on the basis of REST architectural style and its communications approach [9].

3 TYPES OF WEB APPLICATIONS

Web Application types can be categorized into many categories, depending on their visual presentation, functionality, approach or technologies used. Currently, Web Application Development can be divided into three approaches/methods and they are described below.

3.1 Multi Page Applications

Multi Page applications, also shortly known as (MPA), are traditional web applications. When data is exchanged, then a new page is requested from the server to render it in the web browser. So, when a user interacts with the web application, then the whole page is reloaded and a new page is displayed. This process is slow and degrades the user experience as the server takes time to generate the pages, send it to the client and render it on the browser.

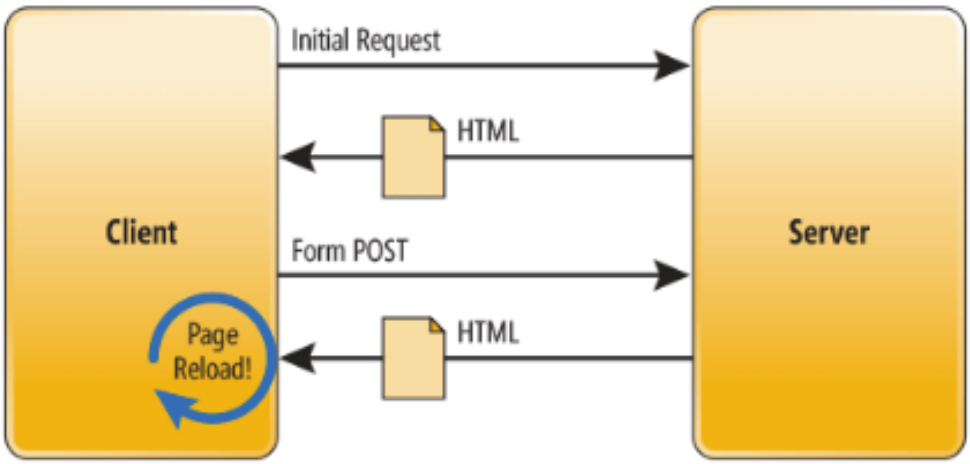

FIGURE 10. The Multi Page Application Lifecycle [16]

As shown in figure 10, Every time the client requests the server, the server generates the new HTML page and renders, which has to reload the webpage again.

3.2 Single Page Applications

Single page applications, also shortly known as (SPA), consists of a single HTML page on the browser which interact with the users dynamically. SPA does not load the entire page when the user makes a specific request. AJAX, HTML5 and JavaScript frameworks are used to build the SPA, which will provide a fluid and responsive webpage without having to reload the page consistently. The whole process is fast compared to MPA as most of the processing takes place on the client-side using JavaScript. [16]

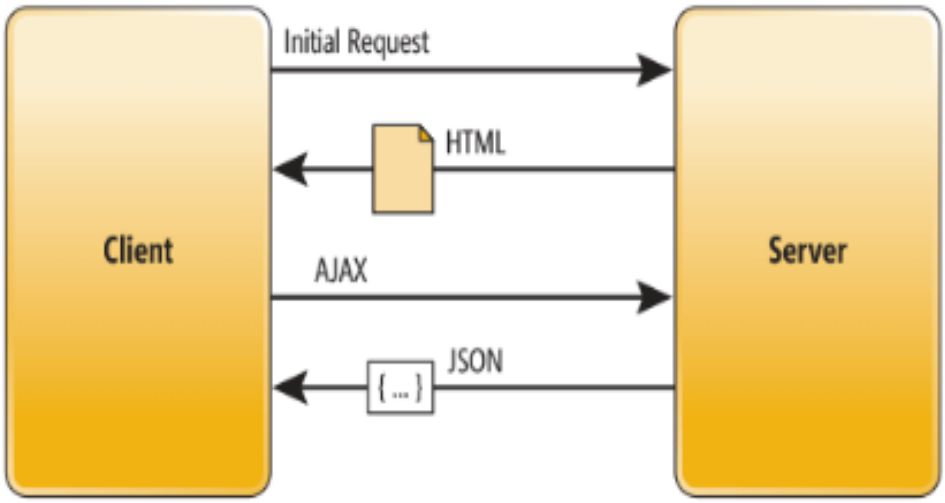

FIGURE 11. The Single Page Application Lifecycle [16]

As shown in figure 11, All the required components of the webpage are loaded when the initial request is made. After that the updates are carried out and these update calls only render the requested data, which is then updated on a webpage without having to reload.

3.3 Progressive Web Application

The Progressive Web Application, also shortly known as (PWA), is a web application, which is well known for its performance, reliability and user engaging capability. Like any other web application, PWA runs in a browser but on top of that, it provides the additional features of notifications, offline use and access to the device hardware.

PWA has been gaining its popularity among developers because of different features. The main characteristic of PWA is its reliability, As soon as the user loads the application from their home screen, it lunches instantly regardless of network connection quality. PWA can be added to the home screen of the user's device without having to download it from the application store. An immersive full-screen experience provides the feel of using native applications. [18]

The main benefits of developing PWA over native applications are listed below.

1. PWA is easy to develop and maintain as it is the same application for all the different devices as opposed to native applications.
2. It is cheaper to build PWA compared to native applications.
3. PWA supports most of the native application capabilities.
4. PWA is easy to publish and it has better conversion as it is easy for users to find and install.
5. PWA can be used offline regardless of the network status.
6. PWA has the device API access.
7. PWA provides an easy update feature.

The core components of PWA are Web App Manifest and Service Worker. They are described below.

3.3.1 Web App Manifest

The Web App Manifest is a JSON file which is responsible for providing PWA its native application interface [19]. The manifest file describes how the application can be launched and how the application should be displayed. It also describes the application name, icon, URL and location.

```
{
  "short_name": "React App",
  "name": "Create React App Sample",
  "icons": [
    {
      "src": "favicon.ico",
      "sizes": "64x64 32x32 24x24 16x16",
      "type": "image/x-icon"
    }
  ],
  "start_url": ".",
  "display": "standalone",
  "theme_color": "#000000",
  "background_color": "#ffffff"
}
```

FIGURE 12. The manifest.json file for PWA.

In figure 12, the manifest.json file describes all the basic elements for PWA and their uses are listed below.

- “short_name”: the text displayed on the home screen next to the app icon
- “name”: is the property used in the application install prompt
- “icons”: describes the image property, added on the user’s screen
- “start_url”: preferred URL loaded when the application is lunched
- “display”: hides the browser UI element including URL bar
- “theme_color”: is used for the toolbar and also during a task switch
- “background_color”: describes the splash screen property

3.3.2 Service Worker

A service worker is the most important component of PWA. It acts as a layer between client and server, and all the browser requests are passed through it. It takes the form of JavaScript file and provides control over websites/pages behavior by intercepting and modifying navigation, resource requests and caching resources. For example, the service worker defines how webpages should behave when there is no Internet connection. Service workers only run over HTTPS to avoid man in the middle attacks. [20]

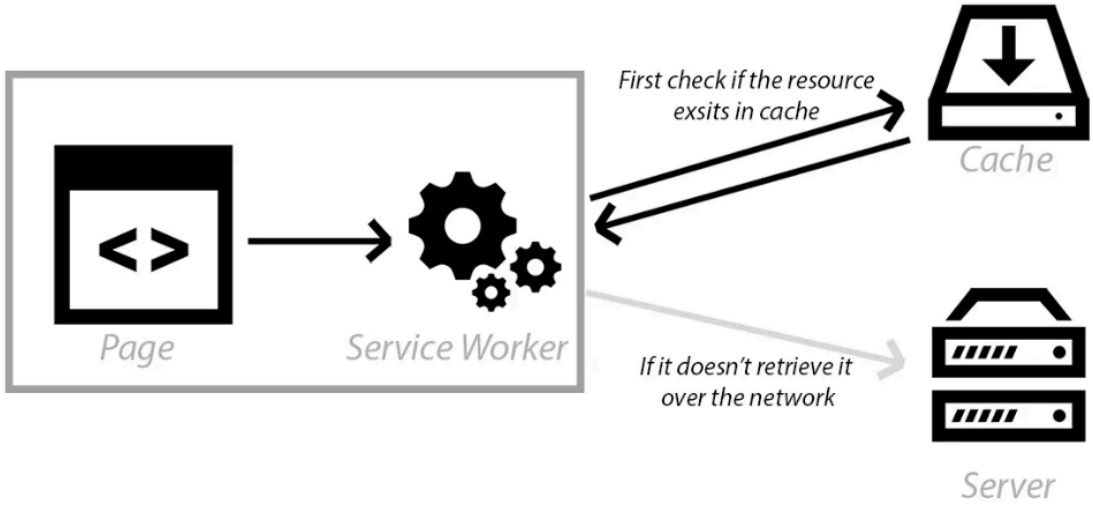

FIGURE 13. The Service worker data flow in PWA [21]

As shown in figure 13, the service worker acts as the middle layer between a client and a server. Firstly, the service worker checks if the resource exists in the cache. If the cache is not available, then the application makes a connection with the server and loads and caches all the data required and shown for the view. If there is no network connection, then the service worker loads all the cached data from the previous session and loads it for the view. This process makes the user experience better even without a network connection.

4 IMPLEMENTATION (WEBSITE FOR RAVINTOLA SARGAM)

In Finland, there are numbers of Nepalese restaurants all over the country and the quality of food and services they provide is well known and appreciated by the customers. Ravintola Sargam being one of them, tends to serve delicious and genuine Nepalese dishes. It is located in Espoo, Finland.

Need for the Website

In this modern time of technology, people prefer to know everything about the place before even visiting there. The case is the same as the restaurants or any food serving places. Customers want to know the menus, services and the quality they provide, before visiting the place itself. So, the restaurants must have a proper website to provide all the necessary information for the customers.

Requirements

The features of the website are listed below.

- In Finnish and the English language
- Responsive (suitable for both desktop and mobile devices view)
- Progressive Web Application (PWA)
- Provides information about the restaurant, opening hours and discounts
- Pictures
- Menu items with a price
    - Separate Lunch menu for every day of the week
    - Starters
    - Mains
    - Group Menu
    - Desserts
    - Drinks
- Contact Information, Map
- Reservation (Customers can make a reservation request to book seats for the dinner)

The website was developed by using the web technologies mentioned above in chapter 2. The proper plan was made to develop the website and the development was done in different phases which are listed below.

- Designing
- Code Editor Selection
- ReactJS Environment
- Website Implementation (Front-End and Back-End)
- PWA Implementation

4.1 Design

The main idea was to create a website with a pleasing visual appearance, easy to navigate and should provide all the information users need. The website should be responsive and the users should be able to use it on both desktop and their mobile phone seamlessly. Before starting to develop an actual application, some mockup designs were made using Balsamiq [22] to make the development easier and smooth.

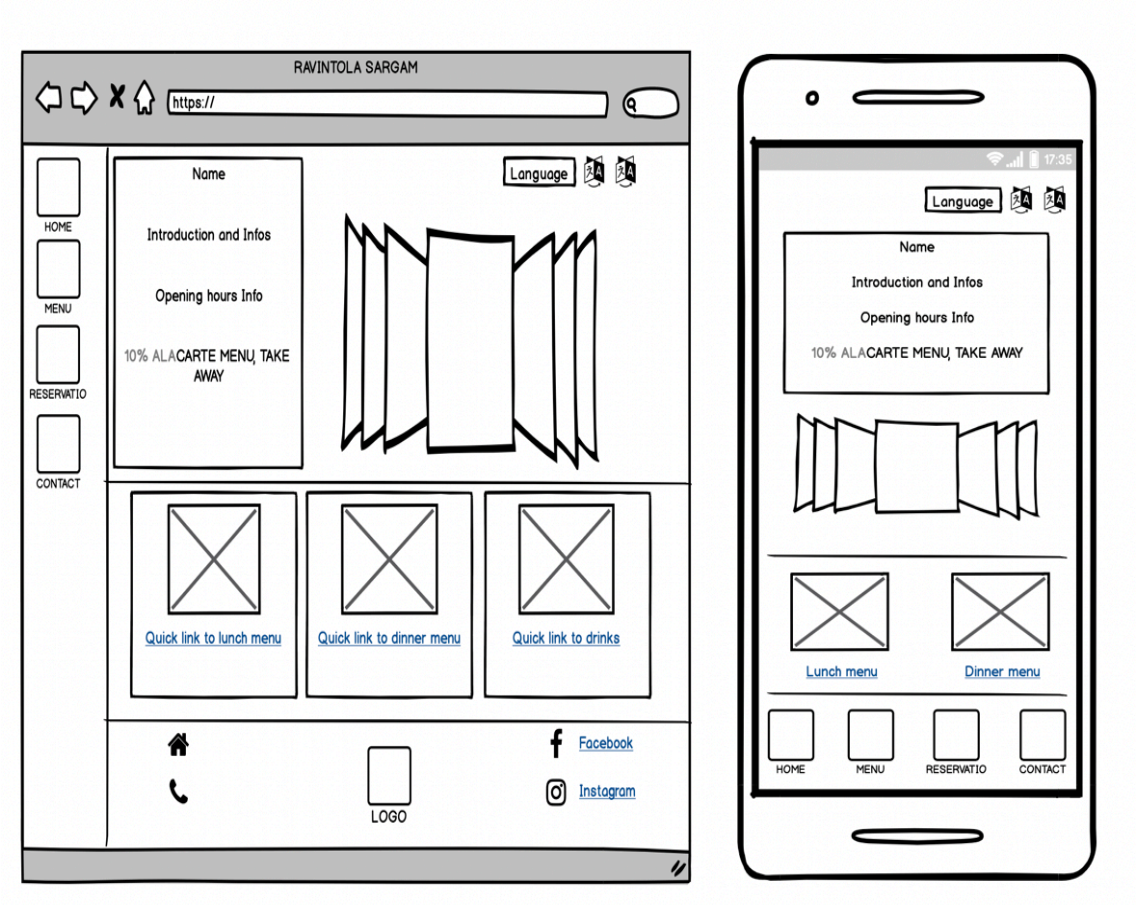

FIGURE 14. The Landing page Mockup design for desktop and mobile view respectively

As shown in figure 14, the basic mockup designs were made before the development, to have a clear idea of how the website is supposed to look and how it should behave on desktop and mobile devices.

4.2 Code Editor

Before starting to develop any application, it is very important to select a proper code editor. It is necessary to choose the code editor that provides all the latest features, saves time, provides proper workflow and maintains code quality. Visual Studio Code [23], Atom [24] and Sublime Text [25] are the few popular and free code editors available.

Visual Studio Code was selected to develop the application as it provides excellent JavaScript support. ES7 React/Redux/GraphQL/React-Native snippets [26], Node.js Modules Intellisense [27], Bracket Pair Colorizer [28] and Prettier – Code Formatter [29] are some useful add-ons/extensions, which were added to the VS code during the development.

4.3 ReactJS Environment

Node.js was installed along with npm. “npm” is the package manager for the Node.js platform which helps to install various packages and resolve their dependencies [30]. Once, the Node.js and npm were installed, the React project was created using the command “`npx create-react-app sargam`”. The created project does not handle backend logic or databases; however, it creates a frontend build pipeline.

All the packages required for the front-end development were installed, as shown in figure 15 below.

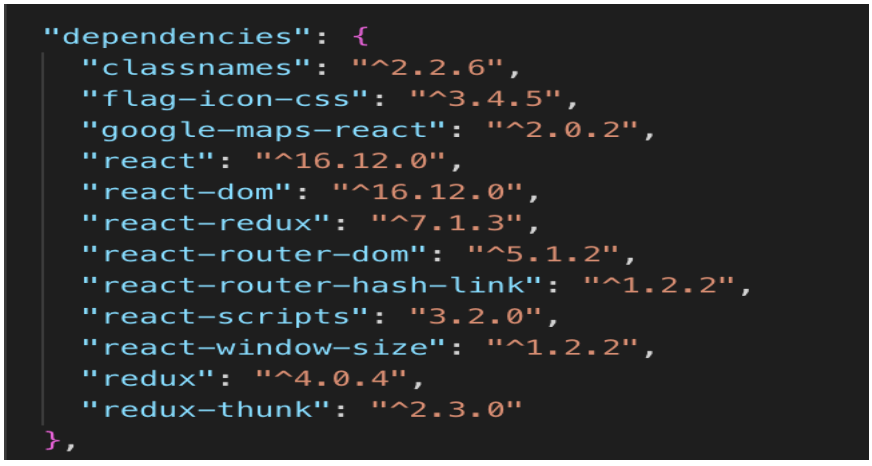

FIGURE 15. The package.json file with dependencies information

4.3.1 Project Structure

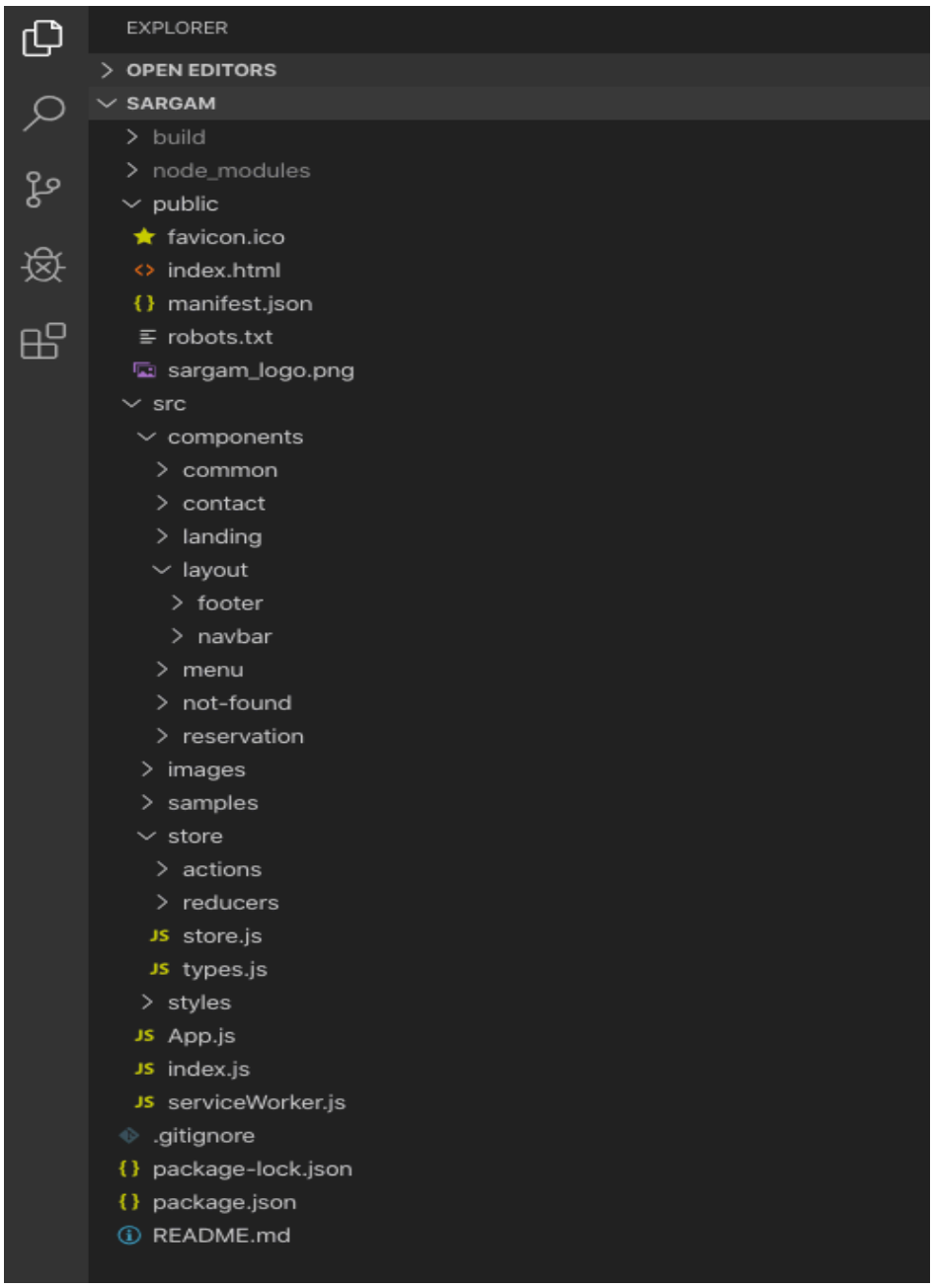

FIGURE 16. The Project Structure

As shown in figure 16, The project structure was created with all the folders and files required for front-end development. They are nested inside one another to provide a proper structure to the project and make it easier to understand.

The contents of the folders and files are described below.

- The “public” folder is an entry point of an application which includes an index.html file, manifest.json file and a logo image used while deploying to PWA.
- The “components” folder includes all the react components of an application. These components are the building blocks of an application describing how the section of User Interface is supposed to look like.
- An “images” folder contains all the images used in the application.
- The “samples” folder includes all the files required for language localization. The files include the contents for Finnish and English languages.
- The “store” folder includes the react-redux elements. It contains action files, reducer files, store.js, and type.js.
- The “styles” folder includes all the .css files required for styling react components.
- An “App.js” file contains the root component of the application [31]. In react, all the components are treated with a hierarchy, where <App /> is the topmost component.
- An “Index.js” file is an actual entry point of all the node applications describing what and where to render.
- The “serviceWorker.js” file contains the logic for the PWA implementation.
- The “package-lock.json” file sets the currently installed versions of a package in stone, where the npm will use those exact versions while running “npm install”.
- The “package.json” file holds the metadata of the project and dependencies information.

4.4 Website

4.4.1 Navigation Bar and Footer

The Navigation bar allows the users to navigate through the different pages of the website. The navigation bar provides access to the Home, Menu, Reservation and the Contact pages of the website.

As shown in figure 17 below, the Navigation bar is placed on the left side of the webpage providing a nice interface for the users. When the webpage is viewed in a mobile device, the navigation bar is moved down providing a feel of native application like.

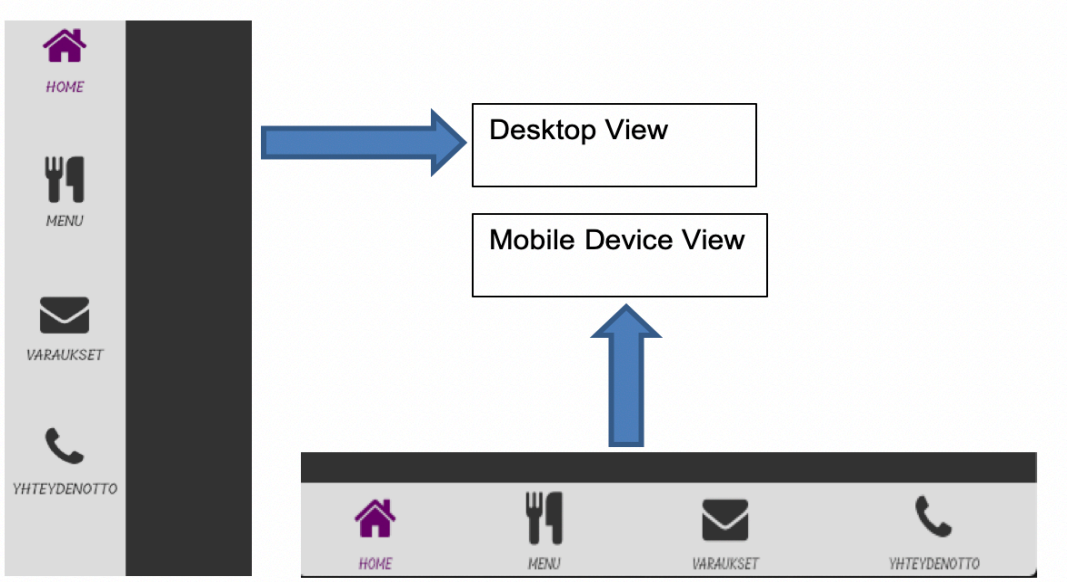

FIGURE 17. The Navigation Bar for desktop and mobile device view respectively

As shown in figure 18 below, The Footer includes the links to other pages of the website, provides access to the social media accounts and includes the logo.

FIGURE 18. The Footer of the Website

4.4.2 Main Page

The main page is the landing page of the website. It includes the general information of the restaurant, opening hours, discounts, image slider, link to menus and language options. As shown in figure 19 below, the main page is composed of different components combined.

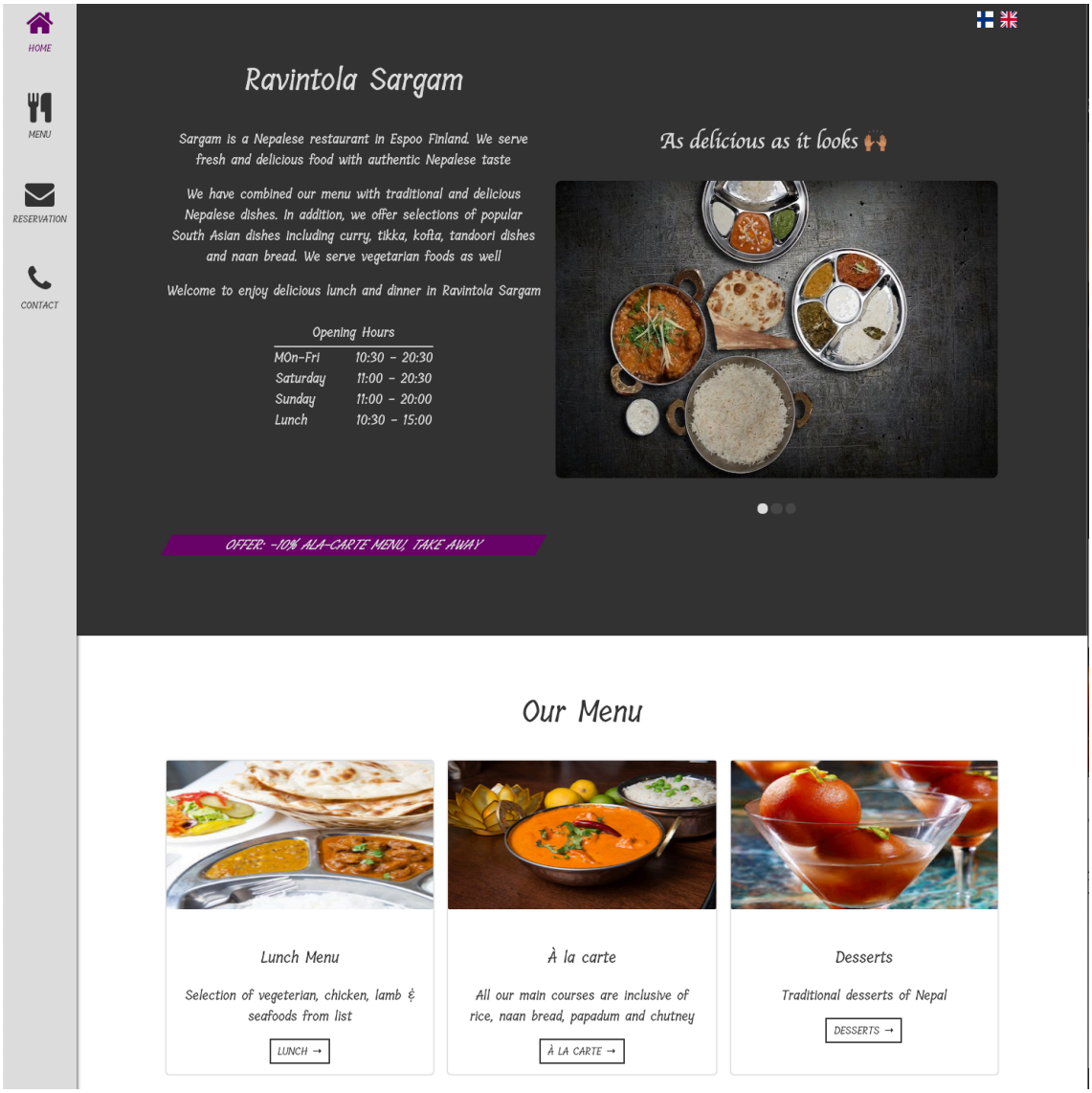

FIGURE 19. The Main Page of the Website

4.4.3 Menu Page

The menu page of the website provides all the necessary information about the available dishes, price and ingredients. As shown in figure 20 below, the dishes are divided into different groups. Menu page includes the navigation to lunch, starters, mains, group menu, desserts and drinks. The dishes are assigned the number, making it easier for customers to place an order.

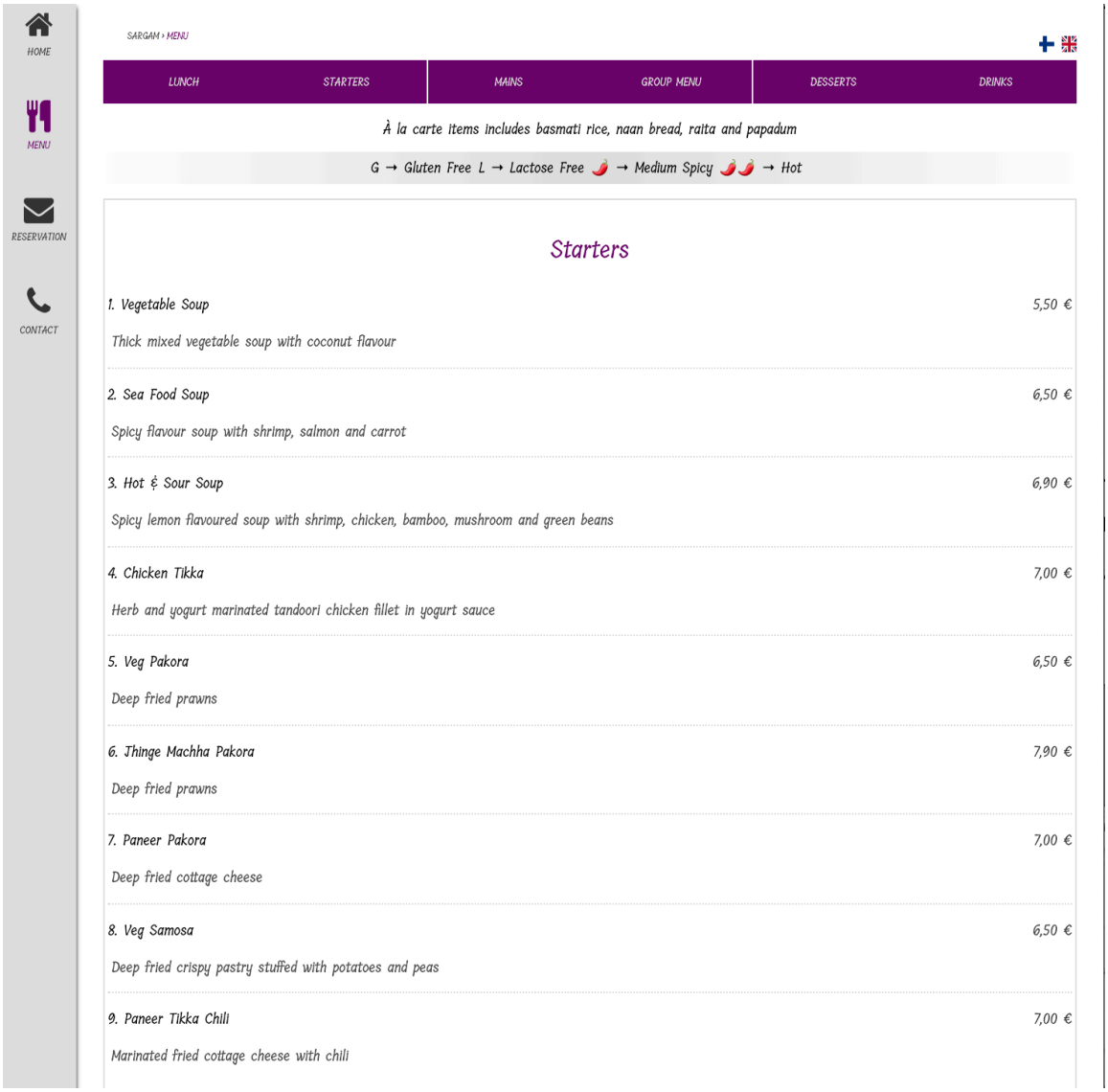

FIGURE 20. The Menu Page of the Website

4.4.4 Contact Page

The contact page provides the contact information of the restaurant. This page allows the customers to send an email for the reservation or any other special requests. As shown in figure 21 below, the Google Maps is also added to this page, providing the exact location of a restaurant.

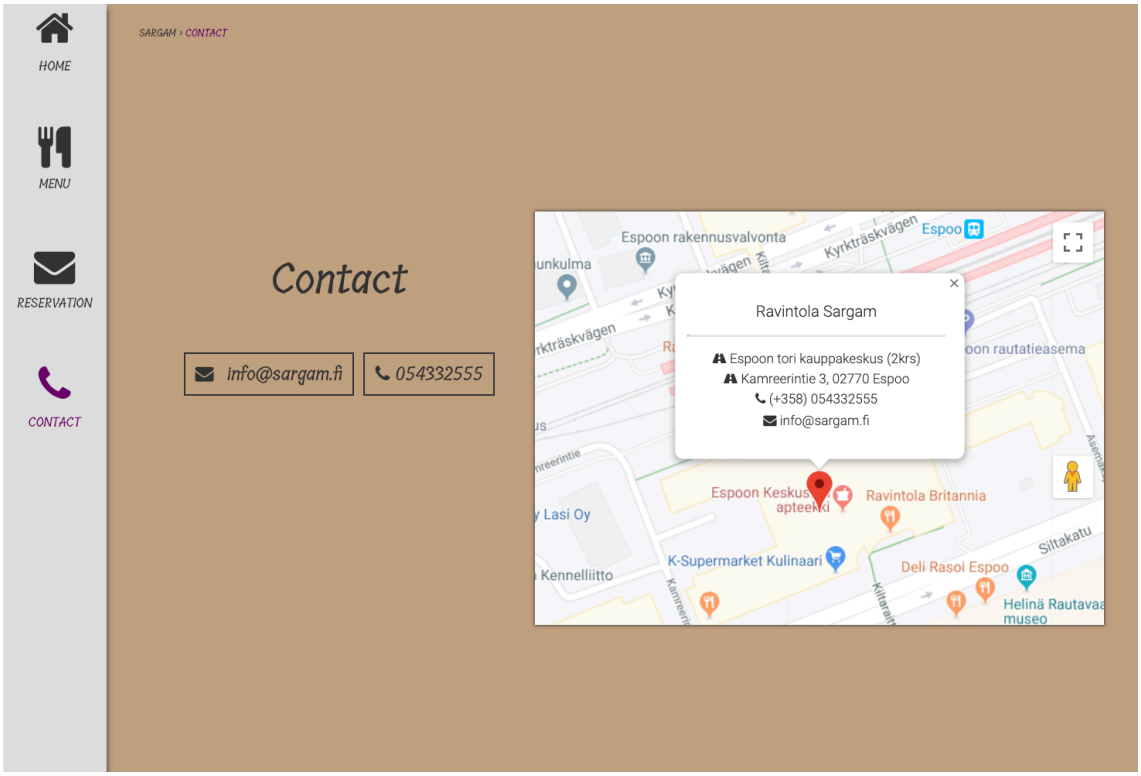

FIGURE 21. The Contact Page of the Website

4.4.5 Reservation Page

The reservation page allows the customers to make a table reservation for a small or a large group. As shown in figure 22 below, the customers have to provide the name, phone number, email, number of people, date and time to make a reservation request. After the reservation request has been made, the information is sent to the restaurant via an email so that they can respond to the email.

The reservation form has also, a form validation and displays text prompt, if either email is sent or not.

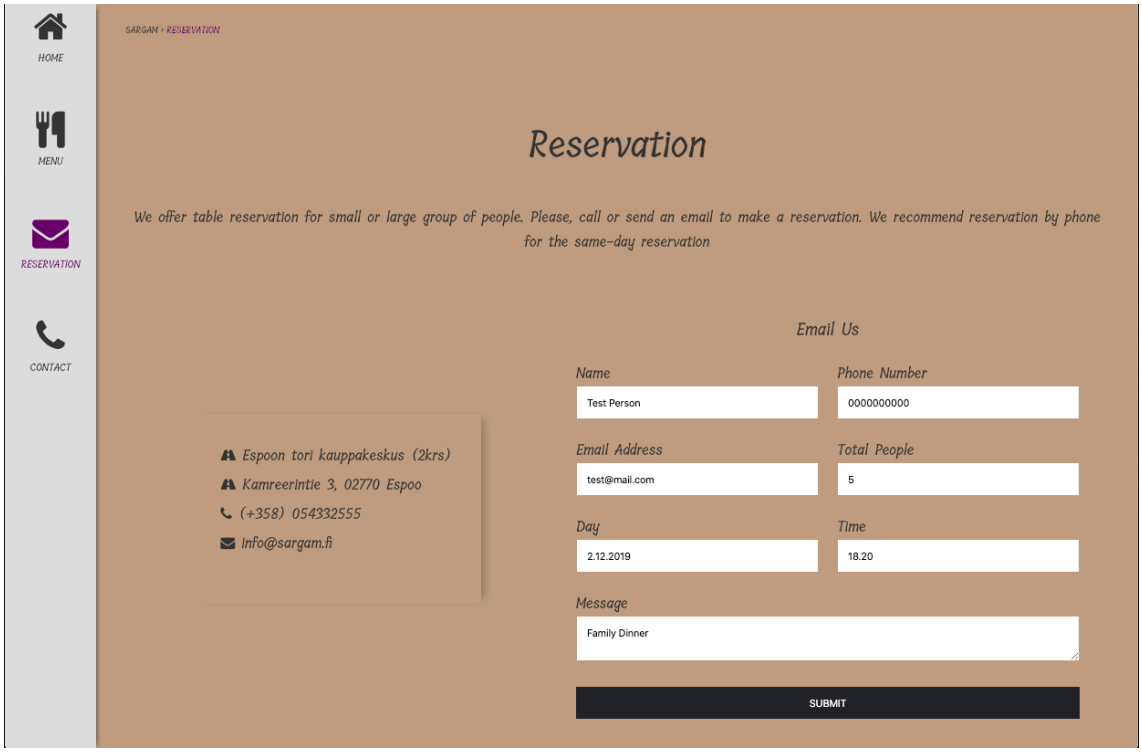

FIGURE 22. The Reservation Page of the Website

Nodemailer and RESTful API

The Nodemailer, which is a module for the Node.js applications, was used to send the email. With the Nodemailer the emails can be sent easily in three steps. [39]

1. Nodemailer transporter is created using either Simple Mail Transfer Protocol (SMTP) or some other mechanism.
2. Email options are set up with the sender and the receiver information.
3. The email is delivered using `sendMail()` method of previously created transporter. [39]

Restful API was used to handle the reservation request, made by the users. During the testing, the Ethereal [35] was used to receive emails. The Ethereal is a fake SMTP service, mostly used by the Nodemailer Users.

```
// create reusable transporter object using the default SMTP transport
let transporter = nodemailer.createTransport({
  host: HOST,
  port: PORT,
  secure: PORT == 465, // true for 465, false for other ports
  auth: {
    user: EMAIL_ID, // generated ethereal user
    pass: EMAIL_PASSWORD // generated ethereal password
  }
});

// setup email data with unicode symbols
let mailOptions = {
  from: `"${req.body.name}" <${req.body.email}>`, // sender address
  to: EMAIL_ID, //receiver
  subject: "Reservation Request", // Subject line
  text: `${output}`, // plain text body
  html: `${output}` // html body
};

// send mail with defined transport object
transporter.sendMail(mailOptions, (error, info) => {
  if (error) {
    console.log(error);
    return res.status(404).json({
      serverError: "Error while sending email. Please call us instead"
    });
  }
  console.log("Message sent: %s", info.messageId);
  res.json({
    success:
      "Sent! Reservations are final once you get the confirmation email.  "
  });
});
});

module.exports = router;
```

FIGURE 23. Nodemailer used to send an email

As shown in figure 23, the Nodemailer transporter is created using the SMPT, email data was setup and the `sendMail()` method was used to deliver the email.

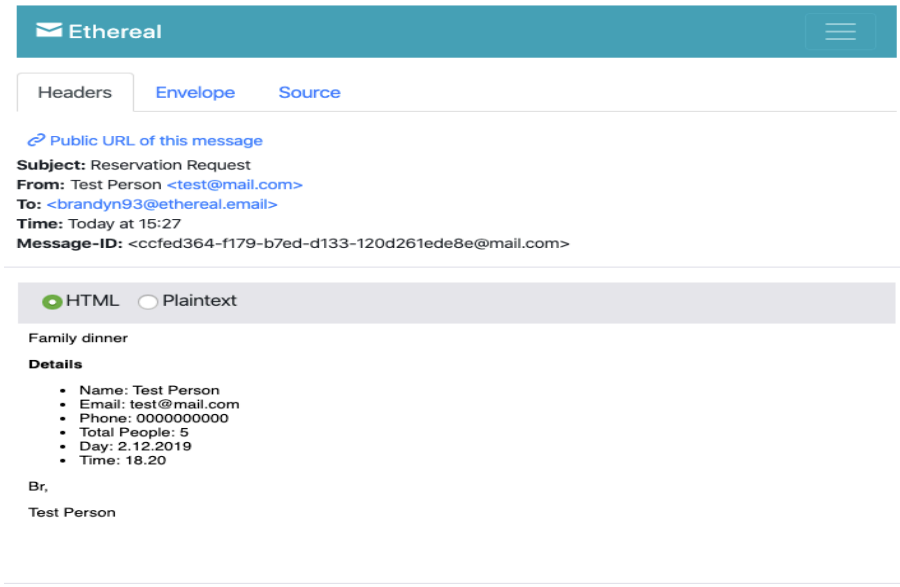

FIGURE 24. An example of an email received in Ethereal during the test

4.4.6 Language Localization

Language Localization plays an important role in web app development. Often referred as the translation of a website, but it is far more than just a text to text translation. Language Localization involves a cultural approach, on top of translating the websites, it reshapes the whole website so that the locals feel as if the website content is built especially for them.[34]

The developed website provides the option of both the English and the Finnish language. Users can choose the language by clicking the flag icons available on the top-right corner. As shown in figure 25 below, when the Finnish language is selected, the contents are changed into Finnish.

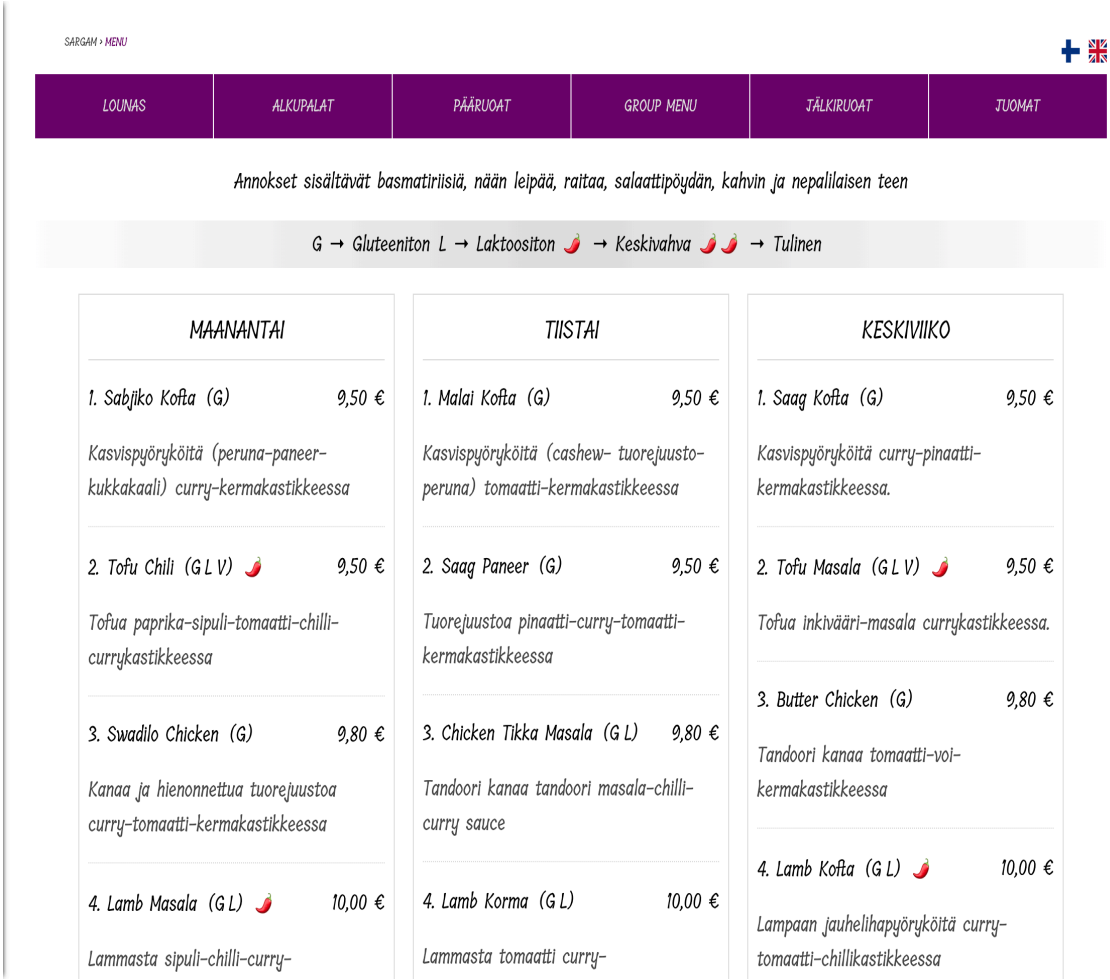

FIGURE 25. The Lunch Menu in the Finnish Language

For language localization implementation, all the language specific data is written in a separate `.js` file, where the contents of the website are written as an object for both English and the Finnish language. The process is managed by redux, where the default language of the website is set in Finnish and all the functions required for changing the language are defined. Figure 26 below shows an example of à la carte items data for the English language and the Finnish language respectively.

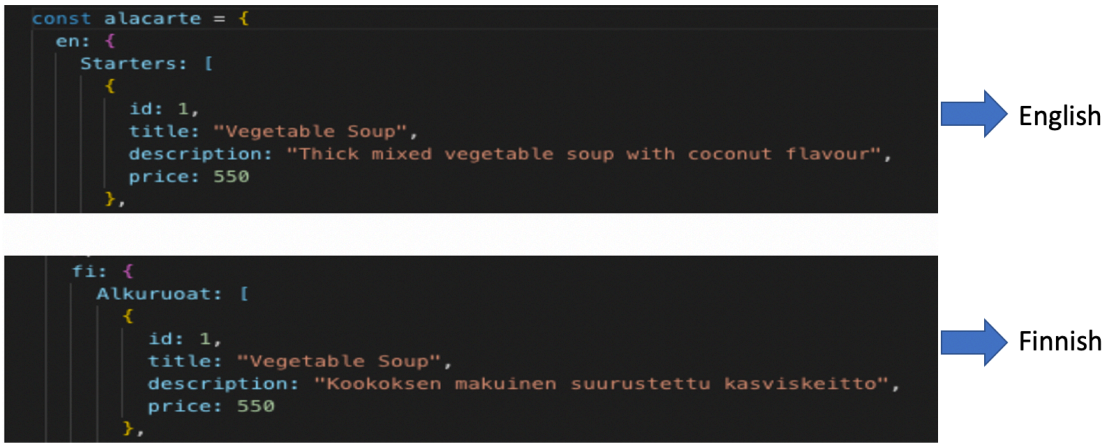

FIGURE 26. An example of à la carte items data

As shown in figure 27 below, once the user opts in to change the language by clicking the available language options, then the `changelanguage` function is called and the contents of the website are changed accordingly.

```
export class LanguageOptions extends Component {
  static propTypes = {
    changeLanguage: PropTypes.func.isRequired
  };

  handleFinnishClick = e => {
    this.props.changeLanguage("fi");
  };

  handleEnglishClick = e => {
    this.props.changeLanguage("en");
  };
```

FIGURE 27. Handling the language selection clicks

4.5 PWA Implementation

To convert the existing react application into Progressive Web application, the manifest.json file and the service worker file is needed. React application creates both of those files by default at the time of the first installation. Small modification in those files ensures the PWA implementation.

```
{
  "short_name": "Ravintola Sargam",
  "name": "Ravintola Sargam Espoo",
  "icons": [
    {
      "src": "sargam_logo.png",
      "sizes": "512x512 64x64 32x32 24x24 16x16",
      "type": "image/png"
    }
  ],
  "start_url": ".",
  "display": "standalone",
  "theme_color": "#000000",
  "background_color": "#ffffff"
}
```

FIGURE 28. The manifest.json file

As shown in figure 28, manifest.json was created, it describes how the PWA application can be launched and how it should be displayed.

A serviceworker.js is also invoked by default when creating a react application. It includes the script which runs in the background of the browser, separate from a webpage. The script includes features like push notifications and the background sync. It also intercepts and handles network requests, such as managing a cache of response [40]. However, the service worker is in an unregistered state and can be registered by changing its state from an “unregister” to “register“ in an index.js file as shown in the figure 29 below.

FIGURE 29. Registering the Service Worker

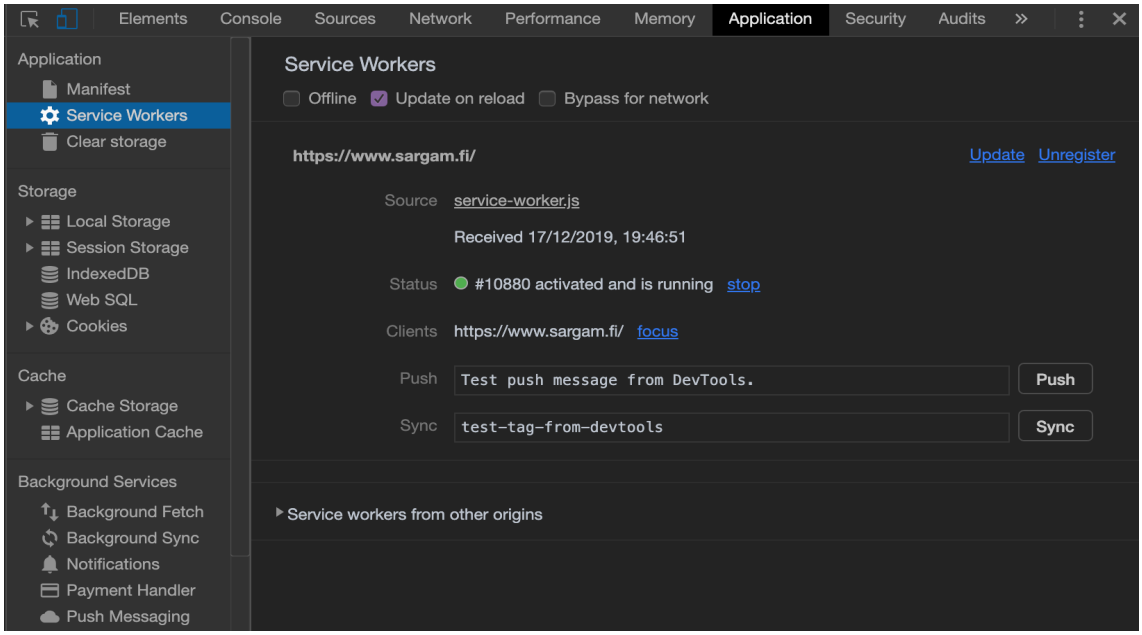

FIGURE 30. The Service worker status in Chrome Dev Tools

As shown in figure 30, the service worker was identified, registered and running without any errors.

Cache-first Strategy

The `workbox-webpack-plugin` is integrated into the production configuration, taking care of generating a service worker file. The service worker file then automatically precached all the local assets and keep them updated, once the updates are deployed. The service worker uses the cache-first strategy for handling all the requests for the local assets, ensuring smooth running of the web app regardless of the network status. This means that all the static assets are fetched from the service worker cache and if that fails, the network request is made. [41]

The service worker lifecycle controls the updated contents after the initial caching. New service worker is installed if there is even a tiny change in the service worker file. To avoid the lazy-loaded contents, the updated service worker is kept in the "waiting" state by default. As a result, the users will end up seeing the older content until the existing open tabs are closed. [41]

5 CONCLUSION

The main motivation to do this project was an urge to learn new skills and put me out of my comfort zone. In the beginning, with very little knowledge and experience of ReactJS, I decided to broaden my skills and try out new technologies, which will be very helpful for my future endeavours.

During the project development, I managed to learn ReactJS along with Redux state management, which was completely new for me. In conclusion, using a state management system, such as Redux in small projects is redundant. Redux is only needed if the application is complex or deals with a lot of data as it allows the developers to manage the application state and solve the forthcoming issues. The decision to combine ReactJS and Node.js for project development has helped me to develop a fast-running application which runs on multiple devices. Using ReactJS and Node.js provides access to a number of open-source libraries and also the developer community of React.js and Node.js provides excellent help.

During this period, I also gained a detailed knowledge of Progressive Web Application (PWA), their benefits and how they are implemented. PWA seems to be a useful technology moving forward as it helps the web to increase its engagement with the users. Because of its limited features, PWA will not replace all the native mobile applications any time soon. However, it provides a nice alternative to native mobile applications.

Overall, the project was successful, I managed to complete all the initial requirements listed in chapter 4. The users can now get access to all the information provided by the restaurant via the website. The language option will help the users to use the website either in Finnish or in English. Also, with the PWA implementation on the website, users can add the website to their mobile devices easily and they can access it even without the Internet connection.

The website was tested, the smooth running of the website was assured and then the website was handed over to the client. The development process and concepts explained in this thesis report can be useful for developing similar projects in the future.

There is still some room for improvement in the project. The next development phase for the website could be the implementation of the Content Management System (CMS) to manage or update the contents of the website. With the implementation of CMS, the client can manage and update the contents of the website even without knowing how to code. Similarly, the website can be made dynamic with the implementation of databases and providing the admin panel to manage the website. The additional features, such as food ordering and table reservation, are possible options as well. I have given all the rights to the client to use and develop the project further if needed.

6 REFERENCES

1 Railsware 2019. Why use ReactJS. Date of retrieval 18.11.2019
https://railsware.com/blog/why-use-react/

2 QSR automations. Emily Wimpsett. Why restaurants need a website. Date of retrieval 18.11.2019
https://www.qsrautomations.com/blog/restaurant-management/why-restaurants-need-website/

3 W3Schools. CSS Syntax. Date of retrieval 22.10.2019
https://www.w3schools.com/css/css_syntax.asp

4 Wikipedia. 2020. JavaScript. Date of retrieval 13.02.2020
https://en.wikipedia.org/wiki/JavaScript

5 W3schools. JS Example. Date of retrieval 22.10.2019
https://www.w3schools.com/js/tryit.asp?filename=tryjs_whereto_head

6 Nodejs.org. Node.js. Date of retrieval 25.10.2019
https://nodejs.org/en/about/

7 Express.js. Hello world example, Date of retrieval 25.10.2019
https://expressjs.com/en/starter/hello-world.html

8 Webopedia. What is an API?. Date of retrieval 25.11.2019
https://www.webopedia.com/TERM/A/API.html

9 SAA 2019. RESTful API. Date of retrieval 25.11.2019
https://searchapparchitecture.techtarget.com/definition/RESTful-API

10 Education Ecosystem 2019. React History. Date of retrieval 25.10.2019 https://www.education-ecosystem.com/guides/programming/react-js/history

11 Codecademy 2019. React Virtual DOM. Date of retrieval 26.10.2019 https://www.codecademy.com/articles/react-virtual-dom

12 Reactjs.org. React component and props. Date of retrieval 26.10.2019 https://reactjs.org/docs/components-and-props.html

13 Sourcetoad 2019. React one-way data flow. Date of retrieval 27.11.2019 https://www.sourcetoad.com/app-development/the-benefits-of-using-react/

14 Redux. Getting started with Redux. Date of retrieval 27.10.2019 https://redux.js.org/introduction/getting-started

15 Tutorialspoint 2019. Redux-data flow, Date of retrieval 27.10.2019 https://www.tutorialspoint.com/redux/redux_data_flow.htm

16 EnterpriceMonkey. MPA vs SPA. Date of retrieval 28.20.2019 https://enterprisemonkey.com.au/blog/single-page-apps-vs-multiple-page-apps/

17 Vaadin 2019. PWA advantages over Native Application. Date of retrieval 30.10.2019 https://vaadin.com/pwa/learn/advantages-over-native-and-hybrid-apps

18 Google Developers. Progressive Web Apps. Date of retrieval 30.10.2019 https://developers.google.com/web/progressive-web-apps

19 Altexsoft 2018. PWA Pros and Cons, Date of retrieval 30.10.2019
https://www.altexsoft.com/blog/engineering/progressive-web-apps/

20 MDN Web Docs. Service worker concepts and usage. Date of retrieval 2.11.2019
https://developer.mozilla.org/en-US/docs/Web/API/Service_Worker_API

21 Dean Hume's Blog 2018. Service worker data flow. Date of retrieval 25.12.2019
https://deanhume.com/service-workers-can-save-the-environment/

22 Balsamiq. Date of retrieval 30.11.2019
https://balsamiq.com/#

23 Visual studio Code. Date of retrieval 30.11.2019
https://code.visualstudio.com

24 Atom. Code editor. Date of retrieval 30.11.2019
https://atom.io

25 Sublime Text. Code editor. Date of retrieval 30.11.2019
https://www.sublimetext.com

26 VS marketplace. ES7 React/Redux/GraphQL/React-Native snippets. Date of retrieval 30.11.2019
https://marketplace.visualstudio.com/items?itemName=dsznajder.es7-react-js-snippets

27 VS Marketplace. Node.js Modules Intellisense. Date of retrieval 30.11.2019
https://marketplace.visualstudio.com/items?itemName=leizongmin.node-module-intellisense

28 VS Marketplace. Bracket Pair Colorizer. Date of retrieval 30.11.2019
https://marketplace.visualstudio.com/items?itemName=CoenraadS.bracket-pair-colorizer

29 VS Marketplace. Prettier – Code Formatter. Date of retrieval 30.11.2019
https://marketplace.visualstudio.com/items?itemName=esbenp.prettier-vscode

30 Npm. JavaScript Package Manager. Date of retrieval 13.11.2019
https://docs.npmjs.com/cli/npm

31 StackOverflow. What are App.js and index.js in react app?. Date of retrieval 24.11.2019
https://stackoverflow.com/questions/50493069/why-does-create-react-app-creates-both-app-js-and-index-js

32 Node. Package-lock.json file. Date of retrieval 25.11.2019
https://nodejs.dev/the-package-lock-json-file

33 Nodejs.org. Package.json. Date of retrieval 25.11.2019
https://nodejs.org/en/knowledge/getting-started/npm/what-is-the-file-package-json/

34 CWT 2017. Difference between localization and translation. Date of retrieval 20.11.2019
http://clearwordstranslations.com/difference-localization-translation/

35 Ethereal. A fake SMTP Service. Date of retrieval 2.12.2019
https://ethereal.email

36 AngularJS. Date of retrieval 3.12.2019
https://angularjs.org

37 Ember. Date of retrieval 3.12.2019
https://emberjs.com

38 Vue.js. Date of retrieval 6.12.2019
https://vuejs.org

39 Nodemailer. Send Message via Nodemailer. Date of retrieval 8.12.2019
https://nodemailer.com/about/

40 Web Fundamentals. Service Workers. Date of retrieval 28.11.2019
https://developers.google.com/web/fundamentals/primers/service-workers/

41 CRA. Making a PWA. Date of retrieval 29.12.2019
https://create-react-app.dev/docs/making-a-progressive-web-app/#offline-first-considerations