Expertise
and insight
for the future

Dhiraj Koirala

Web application for professional union using React and Node.js

Metropolia University of Applied Sciences

Bachelor of Engineering

Information Technology

Bachelor's Thesis

8 November 2020

Author Title Number of Pages Date	Dhiraj Koirala Web application for professional union using React and Node.js 39 pages 7 November 2020
Degree	Bachelor of Engineering
Degree Programme	Information Technology
Professional Major	Software Engineering
Instructors	Janne Salonen, Head of Department (ICT) at Helsinki Metropolia University of Applied Sciences

The main purpose of the project described in this thesis was to develop an application that assists a union of professional workers in converting their daily operations into a digital platform. The application is a demo application which includes user profile management, income expenses recording, event management tasks, access management, meeting management and committee workflow tracking.

An additional target of the project was to get familiar with modern technology that was implemented during the development of the application. The application was built in the Windows environment using ReactJS, JavaScript, and Redux in the frontend development. Node.js was used for the backend development. For storing purposes, MongoDB was chosen. It includes additional libraries which were used to support the major stack to enhance the performance in terms of security and user interface.

This thesis can be useful in various aspects of business management. It gives information that helps to reduce the cost of an organization by reducing the labor required by manual management. The developed application can reduce the communication gap with the customer and the organization. In general, several resources are under-utilized when records are not managed properly. The issue can be mitigated with the developed application. Redundancy errors and financial errors can also be rectified. Information can also be kept more secure as it can be recorded digitally.

This demo application was successfully presented to some selected members of an architect organization. It received compliments for the work and the expectations of the client were met. Some members of the organization gave feedback whereas some mentioned some extra features, such as email service and group discussion portal, which increased the usefulness of the application. This application opens the door of digitization to the union who was running their business manually. This thesis explains how the organization can utilize its resources with technological assistance.

Keywords	Single Page Application, ReactJS, JavaScript, Redux, Node.js, MongoDB

Contents

List of Abbreviations

SQL	Structured Query Language
NPM	Node Package Manager
HTML	Hyper Text Markup Language
CSS	Cascading Style Sheets
SCSS	Sassy Cascading Style Sheets
XML	Extensible Markup Language
JS	JavaScript
IOT	Internet of Things
IR	Intermediate Representation
UI	User Interface
API	Application Programming Interface
DB	Database
AI	Artificial Intelligence
CRUD	Create, Read, Update, Delete
URL	Uniform Resource Locator
JSON	JavaScript Object Notation
JWT	JSON Web Toke
HTTP	Hypertext Transfer Protocol

CORS	Cross-Origin Resource Sharing
XSS	Cross-Site Scripting
NAV	Navigation
NPX	NPM Package Runner

1 Introduction

In the current digital world, the Internet has become a part of every person's life. Business organizations are competing in the digital platform to upgrade themselves. Multiple platforms are available to enter in the modern market, such as mobile technology, social media, cloud platform, and the web. Web technology has always been a top choice for digitalization. When the internet went live to the world on 6th August 1991 [1], the web was only limited to a small number of users. JavaScript was introduced by Brendan Eich in 1995 [2]. Since then web technology has been increasing its flexibility, scope, and efficiency. Multiple frameworks and libraries are introduced to assist developers in creating new functionality and performance for the web. In the same race, Jordan Walke, a software engineer at Facebook, released the first prototype of React. Subsequently, React has been upgraded and version 17.0.0 was released on 20th October 2020. [3.]

The purpose of this project was to get familiar with ReactJS and Node.js. This application was developed for a union of professional architects to enable the organization function smoothly.

Frontend development was done by using ReactJS alongside Redux to show the retrieved information on the frontend with the UI (User Interface) library. MDBreact and Ant design library were used to design the site. Bootstrap and SCSS (Sassy Cascading Style Sheets) were used to animate the site and to make it interactive, responsive, and appealing. Additionally, the site utilizes libraries, such as react-router-dom, yup, redux-thunk, and react-select-search to make the substance on the site more powerful. There are many reusable components. Furthermore, the site has a responsive view which can be seen in large, medium, and small screen gadgets as required by developers and clients. Besides, the site has an authentication system.

Backend development was done using Node.js along with the Express server. NPM (Node Package Manager) was used to add the necessary package for the development of the application. For the server, Express.js was used. Similarly, storage was required to store the data of the application. Since, NoSQL was deemed suitable for the project, MongoDB was used in the project. To manage the data query and the model, the Mongoose library was used.

There were two ways of creating the request to the server from the frontend. The first one was using the native fetch function of JavaScript and another was using a library found in the NPM store. Axios is the most popular library among the existing libraries. It has multiple features, such as interceptor, automatic json transform, cancel request, and http request. As a result, Axios was used over the native fetch function.

In short, this thesis describes the process of implementing ReactJS, Redux, and Node.js for creating a real world application for the organization of architectures. It also has sections where features of the libraries used in the application are described.

2 Frontend

Frontend web development, also called client-side advancement is the act of creating HTML (Hyper Text Markup Language), CSS (Cascading Style Sheets) and JavaScript for a site or a web application. It is a part that a client can see and interact with.

The final year project documented in this thesis was designed in a user-friendly way which ensures the best functionality and appearance of the website. The frontend includes engagement of the user with the system. It is an interface where the general user interacts with the application for certain performance.

The frontend comprises various tools to work together for the overall performance. Some of the frontend tools that were used in the project are described in the next sections.

2.1 Hypertext Markup Language (HTML)

HTML is a markup language which gives the structure to the given content. It is not a programming language. It contains a number of elements which wrap the content to give the defined structure. The UI that is displayed on a website is mainly a result of HTML tags that are used in the development. A tag consists of various components. The anatomy of HTML is described in figure 1 below.

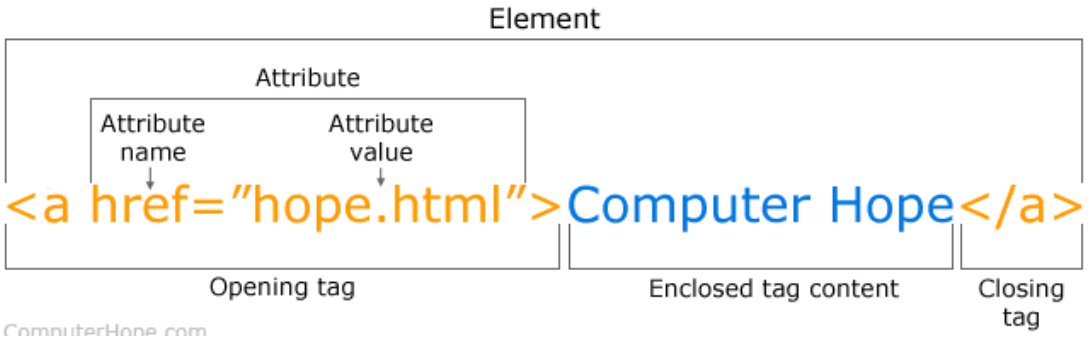

Figure 1: Anatomy of HTML tag [4]

Figure 1 demonstrates the structure of an HTML tag. This is an anchor tag which is one of the widely used tag in HTML for links. An element consists of opening tags and closing tags which define what form of display is to be given. An attribute defines the property of

the tag and the value is given within double quotes (“…”). Then the content is placed between the opening tag and the closing tag.

2.2 Cascading Style Sheet

CSS is a stylesheet to provide the styling to the content written in HTML or XML (Extensible Markup Language). It explains how the element should be presented on a screen. It uses “.css”as a file extension to save the file.

```
body {
  color: #fff;
  background-color: rgba(0,0,0,1);
  margin: 2px;
}
.paragraph {
  font-size: 15px;
}
#main {
  background-color: aqua;
}
```

Figure 2: Basic CSS syntax.

Figure 2 shown above is an example of basic CSS syntax which includes selector, property, variable, value, declaration. The body, class, and id are used as selector.

2.3 JavaScript

JavaScript is a very popular programming language developed by Brendan Eich in 1995. It was originally developed to enhance the frontend part of web development by adding functionality and user interaction on a website. Later, Google introduced ‘v8 engine’, which took JavaScript to the next level. A different frontend framework is developed on top of JavaScript, and further development of Nodejs took the use to the server side. Currently, JavaScript is used in various branches of technology such as frontend, backend, Internet of Things (IoT), game development, and mobile development. [5.]

JavaScript is a single threaded programming language. It can execute only one task at a time. Every browser consists of a JavaScript engine that executes JavaScript code. A

machine can only understand binary language, whereas JavaScript is a high-level language. As a result, certain steps need to be followed to make written language understandable by a machine [5].

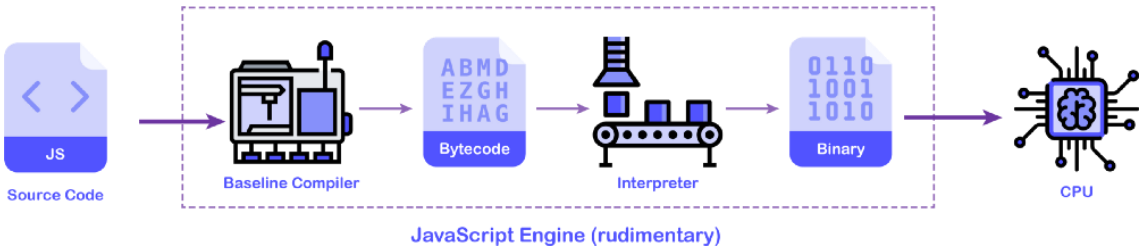

Figure 3: Execution steps of the JavaScript language in a browser [5]

Figure 3 shows the steps where the high-level JavaScript language is changed to machine readable code. Every JavaScript engine contains a baseline compiler which takes source code form the user and converts it to intermediate representation (IR). The converted code is called byte code. An interpreter now exists that converts bytecode to machine code. The output of the interpreter is binary code. Binary code consists of 0 and 1, and it can be directly understood by a machine. This redirect what action to be performed by the machine. [5]

2.4 React

React is a JavaScript library for the development of user interfaces. React is used for developing a single page application which is efficient and flexible. It was developed by Jordan Walke, a software engineer at Facebook, in March 2011 [6]. The developers can divide the application into small pieces which are later combined to form a complete workable application. The small pieces are termed ‘component’. The components are reusable code. There are other properties that are used in React. They include props, state, render, and update. Older version of React uses class-based components which are now converted into function-based components and hooks in current version.

To understand how React works, it is necessary to analyze its root level. React needs JavaScript to interact with HTML documents that are used to create a UI. React works with the help of DOM (Document Object Model) API (Application Programming Inter-

face). Updating the DOM becomes challenging when multiple changes and event triggers occur at the same time. As a result, React creates a new version of its DOM which is known as React virtual DOM. Then it compares the real DOM with React virtual DOM and updates the real DOM to match with the virtual. [7.]

There are different stages during the lifespan of a component while rendering. The following Figure 4 shows the different stages.

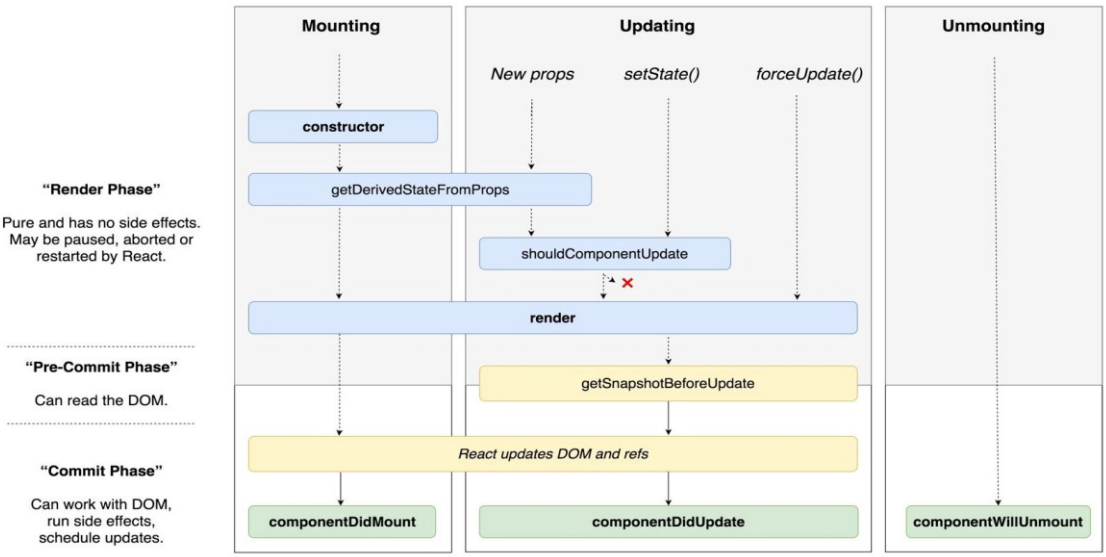

Figure 4: Life cycle of React component [8]

Before mounting the component, there is a pre-mounting phase where constructor of the component is called in a class-based component. Once it is initialized, it is mounted on DOM, it invokes the componentDidMount() method. This is called just after the component is rendedred. It is called once in a lifecycle. Now, DOM has been accessed because rendering has already been called. Now there are a number of methods that are called when there is any update in the component. They are listed below:

- shouldComponentUpdate(): prevents unnecessary re-render
- componentDidUpdate(): is used when previous and current state is changed

The last method is called before the component is destroyed. This method cleans up the component and dumps the data that are used in this component. [8.]

In function-based component hooks are used. 'useEffect' function (hook) has replaced three methods of component lifecycle (componentDidMount, shouldComponentUpdate, and componentDidUpdate). These three replaced methods are called according to the second argument passed in the useEffect hook. Figure 5 below illustrates what hooks look like.

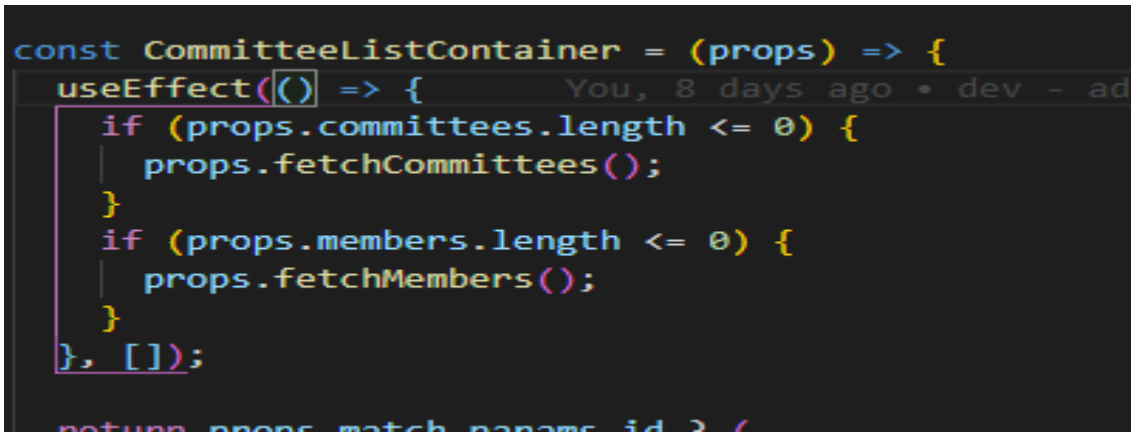

Figure 5: React useEffect hook

Figure 5 has useEffect hooks which take two arguments to execute. The first argument is a function type and the second argument is an array type which means it takes different content inside it. the first argument function contains the code that is to be run when the component is loaded. If the second argument, i.e. array argument is empty, then useEffect is only run once. If there is data, then it compares the previous data with the latest and runs the useEffect function if there is any difference.

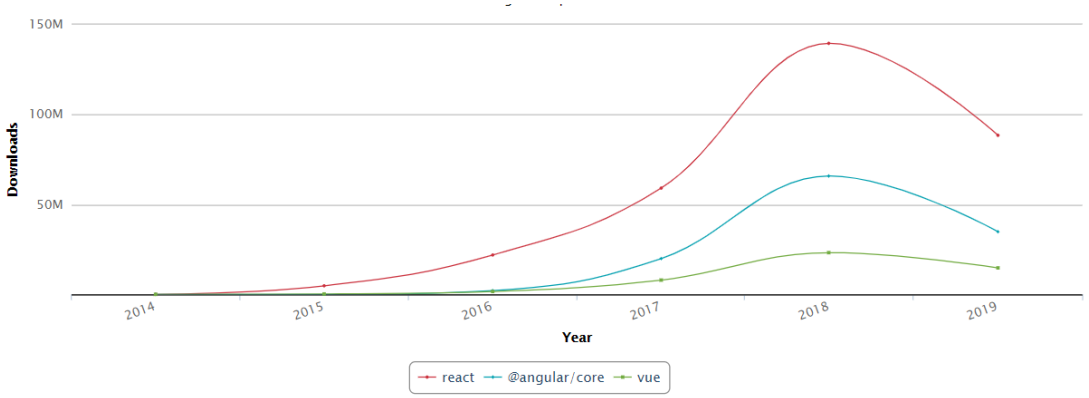

Figure 6: Comparison of various frameworks with React [9]

The above Figure 6 shows statistics of downloads of different frontend libraries/ frameworks. Above figure shows that React has been the top choice of development tool. The reasons for the popularity of react are listed below:

- Reusable stateful components.
- It is simple to learn and implement.
- React compares and modifies the real DOM with virtual DOM, which makes it faster.
- React allows dividing the application into different components, so any changes in one component do not affect other components.
- React is search engine friendly.
- React has a wide community idea are shared. The community has been increasing. There are also blogs, videos, articles and many more channels for support.

2.5 Redux

Redux is a popular JavaScript library to manage the state of an application. There are several states dispersed in different containers or components. These states can be

passed from one component to another. However, if the state has to be passed onto a different level component or to a branch of a different sibling, then it is very difficult. To solve this problem, the state is kept in a single store. Then each component is allowed access to read the value form the store. For the final year project, React Redux library was used.

There are three main parts in Redux. They are listed below:

- Store: Store is an immutable object tree in Redux. The store holds all the states of the application. There is only one store in redux and it is necessary to specify the reducer for the store. There is a create store function available in Redux.

  ```
  createStore(reducer, [preloadedState], [enhancer])
  ```

  Additional argument can also be passed in the createStore function. The second argument acts as a preload state for store and the third is the enhancer of the array. Then the provider is used to wrap the root component of an application and the store is passed as props from the provider.

- Actions: Actions are plain JavaScript objects that have a type and a payload attribute. This type defines what action has is performed. The action type is a simple constant string.

- Reducer: Reducer is a pure function used for updating the state of the store. Reducer creates an operation according to the type mentioned in the action. If there is more than one reducer, then the root reducer is created to combine the reducers into one and the root reducer is passed to the store.

```
import {
  EVENTS_FETCH_ERROR,
  EVENTS_FETCH_SUCCESS,
  EVENTS_FETCH_PENDING,
} from "../constants/actionTypes";

const initialState = {
  pending: false,
  events: [],
  error: null,
};

export function eventsReducer(state = initialState, action) {
  switch (action.type) {
    case EVENTS_FETCH_PENDING:
      return {
        ...state,       You, 2 months ago • dev - sona frontend f
        pending: true,
      };
    case EVENTS_FETCH_SUCCESS:
      return {
        ...state,
        pending: false,
        events: action.payload,
      };
    case EVENTS_FETCH_ERROR:
      return {
        ...state,
        pending: false,
        error: action.error,
      };
    default:
      return state;
  }
}
```

Figure 7: Sample code of reducer

Figure 7 shows the reducer for the event. The different type of action has been imported from the action file. Then the initial state of the event state has been given. The reducer function takes two arguments. The first argument is state and the second argument is action for the reducer. Then the switch takes the action type passed and compares it with the imported action type and updates state value with the respective operation which match the defined action type.

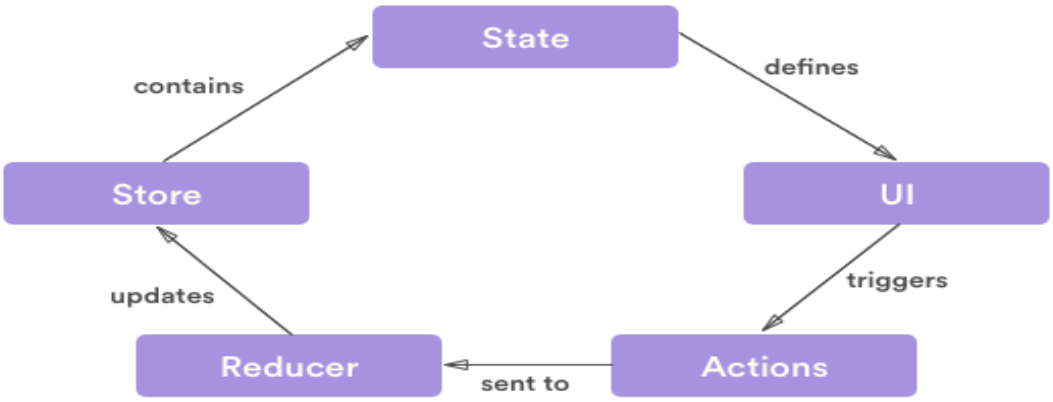

Figure 8: Redux lifecycle [10]

Figure 8 shows the cycle of the redux. Here the store holds all the data. The date is retrieved by the component and displayed in the UI. When there are any event triggers in the UI, then the action is dispatched. This action is sent to the reducer. The reducer then updates the store. Then the data in the store is changed and that changed information is displayed in the UI. [10.]

3 Client Business

The final year project was designed especially for an organization of professional employees of an employee union. This kind of an organization provides professional services, such as occupational training, events, seminars, and suggestions for policies. In this project, the client was an association of professional architects.

3.1 Organization Structure

This organization is basically on flat structure. There are members and a management committee. There are also a number of sub-committees which assist the management committee. There is one committee in this organization which acts as executive committee. The committee members are elected by an active member of organization each year. This executive committee creates other sub-committees which provides necessary assistance in the field of professional practice, academic field, green and sustainable architecture, youth architecture and social responsibility.

3.2 Organization Operation

The operation of business starts with a membership request. A member who has completed his/her professional degree applies for the membership of the organization. This application is reviewed by the secretary and an authorized person. Decisions are made considering the information provided and document attached. Decisions are either approved or rejected. If the application is approved, then these members are counted as active members. Each member pays yearly subscription to renew their membership. Similarly, there are a number of contracts done with the commercial sector for mutual growth. For example, there is a contract between a cement company and this organization. According to the contract, the cement company pays a sum of money to the organization for its operation costs. In return, the organization recommends its members to use the high standard cement of the cement company. The management (executive) committee is elected once a year. Active members are eligible to vote in the election. Other sub-committees are formed by the management committee in a different area. These committees are elected from a group of experienced members such as professors of an academic committee and practitioners of professional committee. Another main

task of a committee is to provide a suggestion and prepare bylaws on the behalf of their professional field. There are also a number of seminars and workshops organized with collaboration with another organization for the development of members.

4 Application Overview

The application was pre-designed on a white board. Initially, the backend structure was designed. Node.js was chosen for the backend, ReactJS for the frontend, and MongoDB for the database. The database structure of the application can be described more with the following diagram.

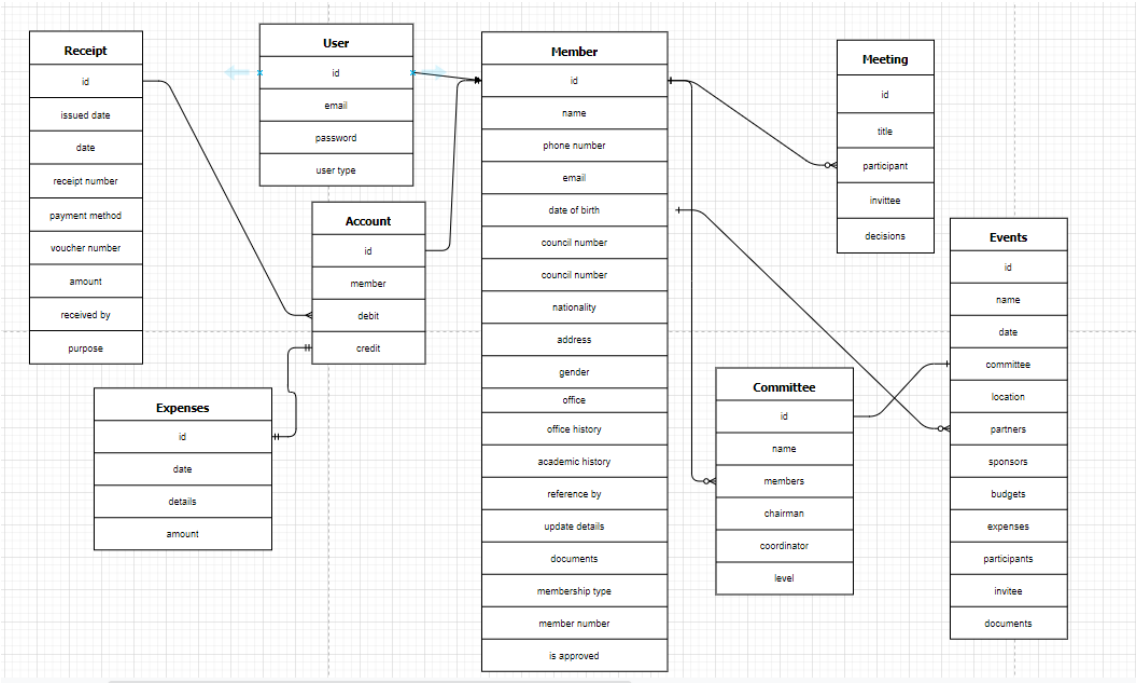

Figure 9: Data model design of application

In the above figure 9, a link can be seen between different collections. At first, there is an admins collection, which is used to keep track of users who login logged in and records the email, password and userType. Similarly, there is a member collection which records the details of members. This works as a central and most important collection because most of the other collections have the dependency on this collection. Another collection is Committee, which has arrays where members are selected. There is also an attribute for chairman, coordinator, name, and type for the committee. Similarly, there is another collection for meetings. This records the date, present members, invitees, and decisions. The ID of these meetings is recorded to the committee. There is also a collection for the event that records the events organized by the committee. It has a field for date, location, and committee before the event occurred. During the event day, they have an array for recording the invitees and the participants. The account collection records the financial data of the organization.

4.1 Backend Overview

This application has the backend that currently runs on port 8000 at localhost. The express library is used to create a server. The following figure shows the folder structure of the backend.

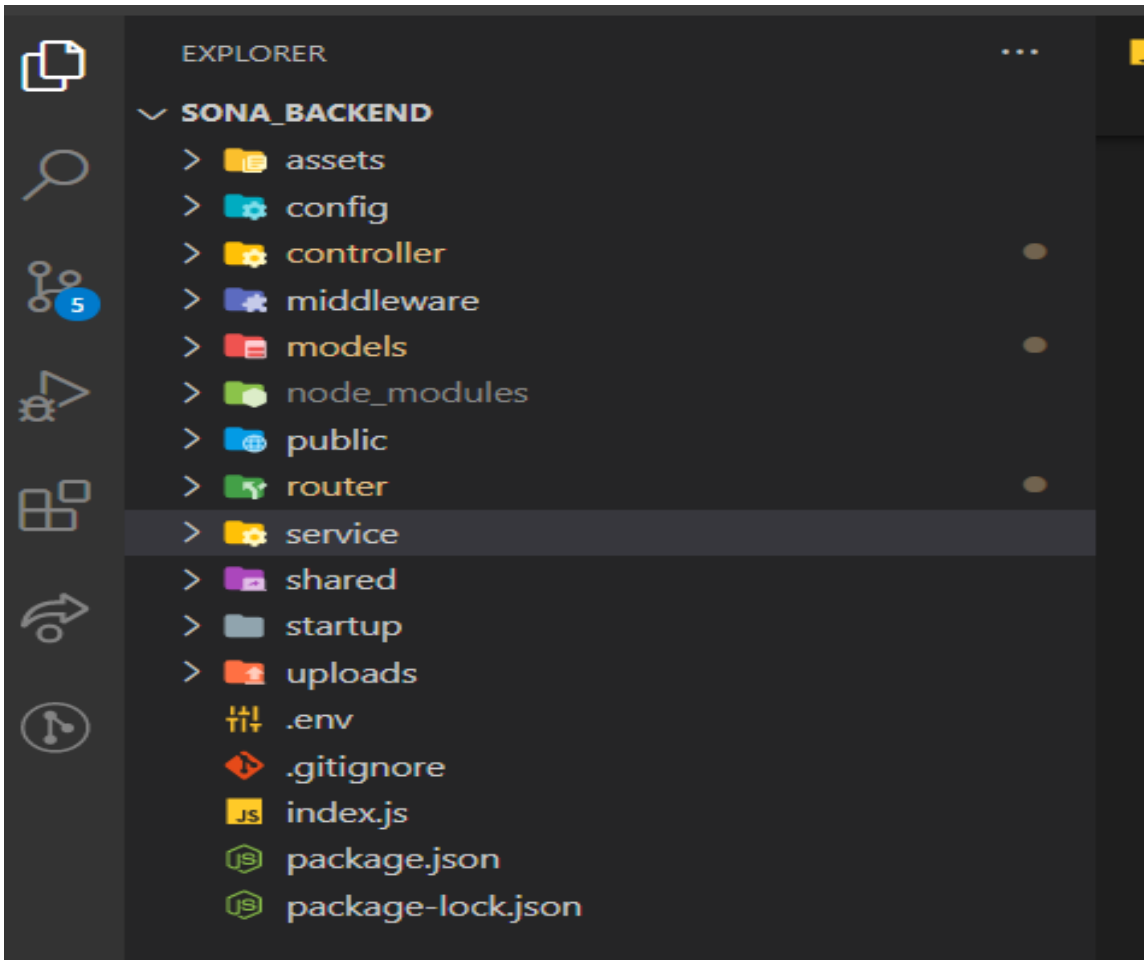

Figure 10: Folder structure of backend

The backend design consisted of the above major folder. The design is briefly described below:

- Assets folder includes all the resources that are required for development. Basically, it includes assets, such as json and some static files.

- The config folder includes the configuration files that are setup for the application. It also includes the configuration for the database which is connected to the application.

- The third folder, controller, includes the supporting function for the router folder. This includes basically the Create, Read, Update, and Delete (CRUD) operation function logic for each model.

- The middleware folder includes the middleware that is used in different routes. Auth middleware, a file uploader, and file filter middleware were used for this application.

- The model folder includes the schema of the database collections that is described in figure 9.

- The router folder includes the file for the different endpoint of the routes.

- The service folder includes the additional functions that are comprised by the CRUD function. Sometimes there is concurrent functionality or sequential CRUD operations to be done at once. Thus, this folder includes a combination of multiple operations.

- The shared folder consists of the data that are shared with multiple parts of the backend. This folder mostly consists of a constant value, default value and some static functions which return special cases such as error or success.

- Moving to file, .env file includes the pre-defined environment variable for the application. This includes the port number, database URL, and API keys. While deploying, the application environment values are taken from a different source OR from different sources (?). If the taken values are pre-defined in the .env file, the application uses them (?). If some variables are not found in the .env file, then the application takes value from the deployed environment. The variables OR values (?) are mostly hidden to maintain privacy.

- The index file is the entry point of the application. While running the application, the file is run across all the import and configuration in this index file.

- Package.json is the root file of Node.js are used for Node Package Manager (NPM). It includes all the metadata needed for the development of the project. It includes and manages the dependencies that are used in development of this application, and versions and scripts that run the project.

4.2 Features of Backend

The project backend consists of numerous features. These features are combined together to work a stable and flexible program. Some of the features are listed in the next section.

4.2.1 Flexible Model

The model of the application is designed in such a way that it can be exported and utilized multiple times. The Mongoose library is used to create the schema of the collection. This library consists of multiple in-built functions for querying and updating the data. Sample code for the model can be seen below.

```
You, a few seconds ago | 1 author (You)
const mongoose = require("mongoose");
const joi = require("joi");
const joiObjectid = require("joi-objectid");
const objectId = require("joi-objectid")(joi);
      You, 2 months ago • develop - all model added, login , auth and admin user (route) created
const eventSchema = new mongoose.Schema({
  name: { type: String, trim: true, uppercase: true },
  date: { type: Date },
  committee: {
    type: mongoose.Schema.ObjectId,
    ref: "Committee",
  },
  location: { type: String, trim: true },
  partners: [{ type: String, trim: true }],
  sponsor: [{ type: String, trim: true }],
  budget: { type: Number },
  invitee: [{ type: String, trim: true }],
  expenses: [
    {
      type: mongoose.Schema.ObjectId,
      ref: "Expenses",
    },
  ],
  participant: [
    {
      type: mongoose.Schema.ObjectId,
      ref: "Member",
    },
  ],
  documents: [
    {
      filename: { type: String, trim: true },
      preparedBy: { type: String, trim: true },
      link: { type: String, trim: true },
    },
  ],
});

const Event = mongoose.model("Event", eventSchema);
```

Figure 11: Code of event model

The above figure shows the model of event collection. The collection of the event has multiple fields which are defined by eventSchema. The field name and location are a type string whereas the partners and sponsors fields are an array of strings. Similarly, the application has the date field with type Date. There are also two ways of including objects in the Mongoose schema. The first type is in the field documents where the array of objects is directly passed to the field. The second type is creating a new model for the object. The new model can be formed by creating a schema that has the same structure as the object. When the data is inserted in that particular collection, MongoDB automat-

ically generates an ID for it. This ID can be passed to other models as type 'mongoose.Schema.ObjectId' and the model reference is to be provided. In above figure 11, committee field has ID passed to it. Similarly, the array of such IDs can also be passed in the schema, for instance in the field expenses and participants. Later while performing a query, the ID can be populated and it can retrieve the data from the connected collection.

4.2.2 Multi-level User

This application is designed for a multi-level user. According to the client, the business structure is currently designed for three levels. The top-level users are administrators. The user of this level include the developer and members of the management committee. They are given authority and access to every level of management of the application. The second level users are the staff of the organization. They can update the details of any member, update the accounts of every member, and add the data in collections, such as event details, meeting decisions, and committees. Lower level user are members. They can only have the right to read the data. They also have authorization to update their personal data and know only their account balance.

4.2.3 JWT Token Authorization and Bcrypt Hash

Security of the user data is a vital part of digital work. Several laws have been passed to protect the user information. Thus, the authorization should always be taken seriously. For this application, a JWT (JSON web token) have been used to identify the right user and restrict the limitation of access. While logging in the application, if the user provides a correct user email and password, the required data is fetched from the database and injected in the token as payload. This token is sent as a response which is saved in the local storage of the web browser. Whenever the request is received from the client-side, first it runs the middleware. Middleware is the piece of code that is between the request response cycle to do a particular task.

```
const adminAuth = (req, res, next) => {
  const token = req.header("x-auth-token");
  if (!token)
    return res.status(401).send({
      status_code: 401,
      message: "FAILED",
      data: "Assess denied. No token provied",
    });
      You, 2 months ago • develop - all model added, login , auth and a
  try {
    const decoded = jwt.verify(token, process.env.JWT_PRIVATE_KEY);
    if (decoded.userType === "ADMIN") {
      req.user = decoded;
      next();
    } else
      return res.status(401).send({
        status_code: 401,
        message: "FAILED",
        data: "Assess denied. Unauthorized",
      });
  } catch (error) {
    return res.status(400).send({
      status_code: 400,
      message: "FAILED",
      data: "Invalid Token",
    });
  }
};
```

Figure 12: Authentication middleware code for admin user

Figure 12 shown above shows a simple middleware function which is used to authorize a user as an administrator. Middleware has access to the request and response request. It also has a callback function. In this function, it first checks if the request has a token or not in its header. If it does not have a token, it sends a failure response. Similarly, it also checks if the token sent is valid with the secret decoding env variable. If anything fails, it sends the failed response with respective status code and a message. If all the pre-conditions match, then it calls the call back function. In this particular case, it calls the next function. On the other hand, Bycrypt is used to hash the password of the user. The admin user and top-level user has direct access to the database. Thus, if the password is saved in simple string format, it can be easily misused. Thus, before saving, the

password is encrypted. To encrypt the password, salt is created specifying the length. It is a one-way function which turns any amount of data into a fingerprint.

4.2.4 Data Validation

There are numerous attacks in the server in real world. Hackers are always hungry to consume data. To protect from the tentative risk of an attack, each data request is to be validated before processing in the server. There are also numerous reasons for validation. It minimizes the server breakdown risk by blocking an unprocessable entities. The false attachments are filtered. Excess payload or missing fields can be easily evaluated before processing. 'Joi' has been used for server-side validation. An example of joi validation can be seen in figure 13 below.

```
const meetingValidator = (validData) => {
  const schema = {       You, 2 months ago • develop - all model adde
    title: joi.string().required(),
    date: joi.date(),
    participant: joi.array().items(objectId()),
    invitee: joi.array().items(joi.string()),
    decisions: joi.array().items(joi.string()),
  };
  return joi.validate(validData, schema);
};
```

Figure 13: Validation code for meeting schema using Joi

Figure 13 shown above shows how the meeting data is validated. The data is passed as an argument. Inside the function, the validator schema is defined. The schema field should be similar to the mongoose collection filed. Then each field is evaluated with the Joi function for each type of data.

4.2.5 Secure Header with Helmet

There is also another well-known vulnerability in the header. This is the major point of threats in a web application. To protect the header from leaking, the helmet library has been used in this application. This library is a group of middleware that deals with the security of HTTP response headers. [11.] Helmet sets the following security middleware:

1. Content-Security-Policy
2. Strict-Transport-Security
3. X-Download-Options (for IE8+)
4. Cache-Control
5. X-Content-type-Options
6. X-Frame-Options
7. X-XSS-Protection

4.3 UI Overview

The User Interface (UI) is the presentation part of application. The design has to be done in such a way that the clients are able to experience the most satisfaction with full functionality. It is important to ensure all the access rights are given to the right person. It is important to make sure that all the component structures are correctly wire framed. In this application, the UI is classified into two parts.

4.3.1 Public Routes

Public routes are those paths which can be accessed by any user. The users do not have to sign in to visit these routes. The routes navigate the user different pages which display the information on respective topics according to the URL. They do not have access to change or update data from the public routes. The routes handle the informative part of the application. Basically, there are three pages which are fully public. The next section describes each page.

4.3.2 Private Routes

Private routes are secured routes for authorizing the user which has to be authenticated before access. This requires login. When the user sends the login credentials to the server, the server processes the data and indicates if the user login is a success or not. When the request is a success, the server sends some user details which are stored in the local storage of the browser. In the frontend part, the context API of React was used to provide the data to the component. The router consumes the data provided by the provider to classify the routes. If the user is logged in, the page is opened. If not, the path is redirected to the sign in page.

Figure 14 shown below shows the structure of the frontend of the project.

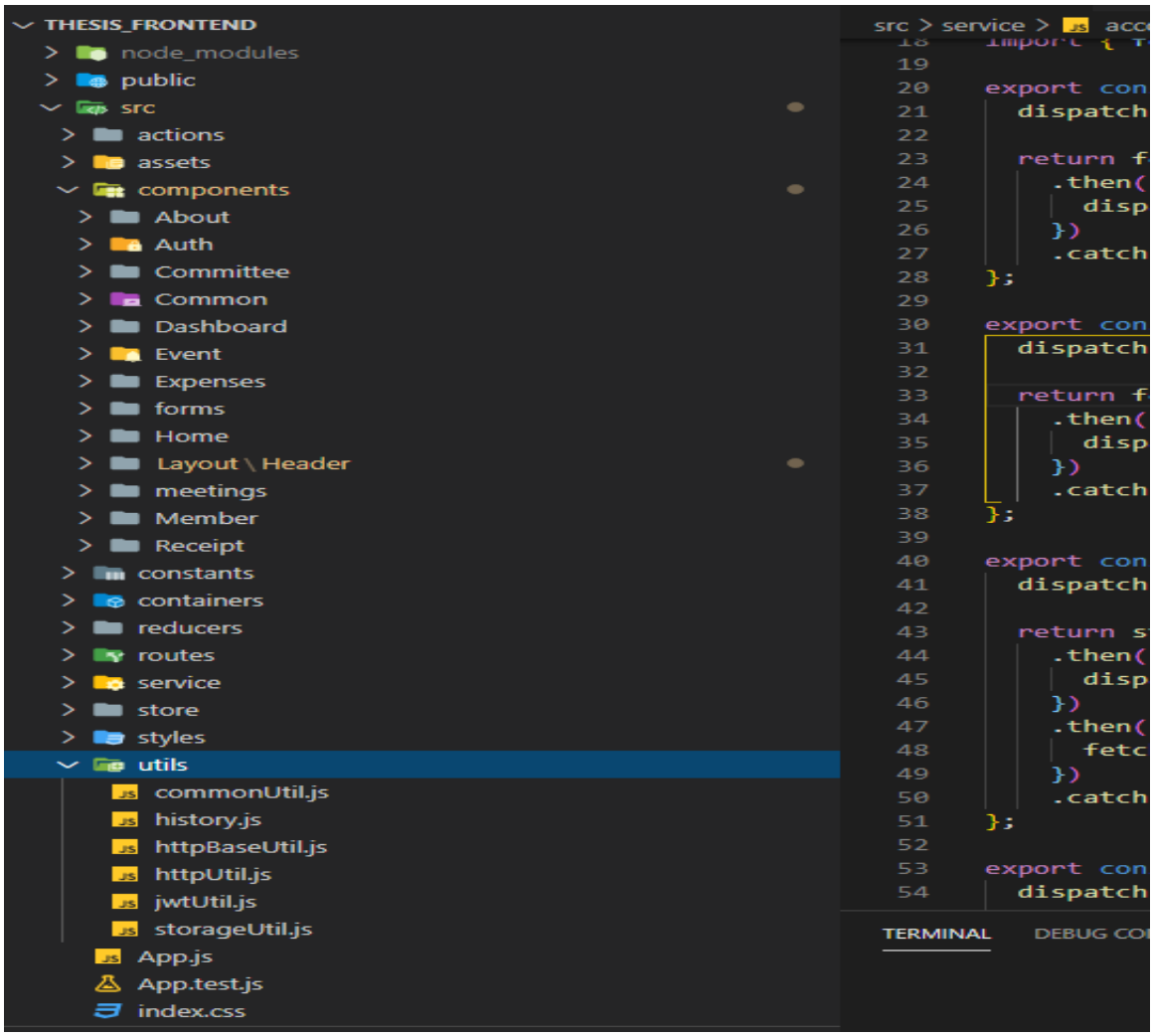

Figure 14: Folder structure of frontend

The structural part of the frontend of the project is explained below.

- Actions: This folder includes the function that returns the action type and the payload for dispatched actions.

- Assets: The assets folder includes the resources and static files required for running the project. It includes icons, images, fonts, CSS and the JSON files.

- Components: This is the most important folder for the frontend. It includes all the UI files that are displayed in the application. Components folder has a sub-folder for each page displayed on the screen.

- Constant: Application configuration constant files are found in this folder. Application constant file include constant variables such as API_URL, JWT_TOKEN, and LOGGED_USER_EMAIL. In addition, there is another file which exports all the action values for the reducer. This also has the field for forms.

- Container: This folder contains the business logic that is passed as props to the component.

- Reducer: The reducer folder contains the file for the root reducer and the reducer for each field of state.

- Routes: The routes folder contains the router for each page. The routes import the container file for rendering in a respective location.

- Service: This service folder includes the service file for each component. This file has the https requests for different events. The different actions are dispatched in steps according to server response.

- Store: This folder has the configuration file for the store configuration for different environments.

- Utils: This folder contains various setup files that are used in different parts of the project. First it contains configuration for the history library. The next file has the configuration setup for the http request. Other files have the utility function for local storage.

4.4 UI Components

UI stands for user interface. User Interface is designed by designer before the real application goes under development. UI of this application was first sketch in paper by pencil. Then later it was extended using photoshop and online tools. After design was ready, it was implemented on this application. Some user interface components are explained below.

4.4.1 Navigation Bar

The navigation bar is the interface that is displayed on top of the screen. This component allows us to navigate through different screen. React is used to create single page application. There is only one page, so the navigation bar only updates the URL that the clients visit which creates the effect on the components of the page. The navigation bar of the application is shown below.

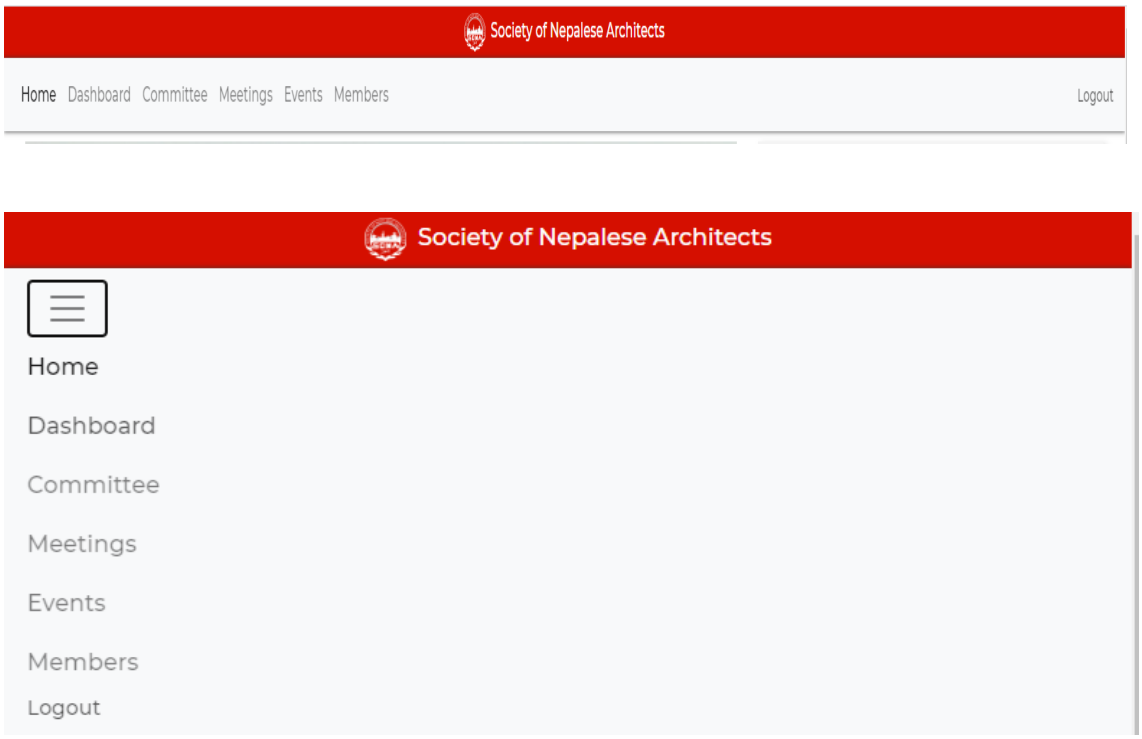

Figure 15: Navigation bar for desktop and mobile view

The navigation bar shown in the above figure 15 shows the different views on different devices. There is a dashboard in the navigation bar when the user is signed in. If the

user has not logged in, then its hidden. Similarly, the right-hand-side of the navigation bar has a link to sign in or a registration page or logout. If user click on logout, then the user is sign out and the page is redirected to the login route.

4.4.2 Homepage

The homepage is the default page for the application. It has introductory contents and it works as an entry point. The URL of this page is public route.

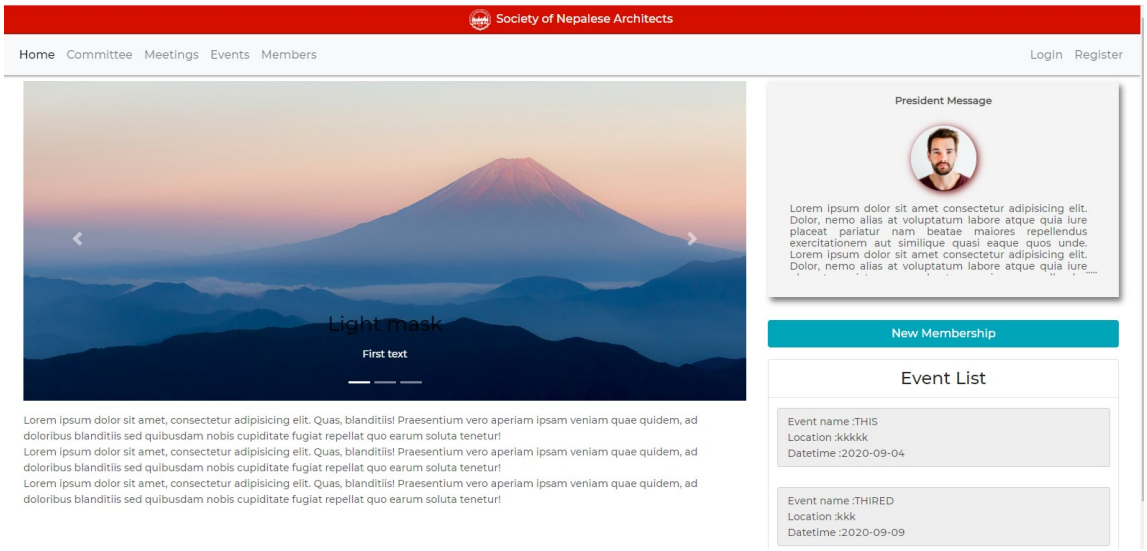

Figure 16: Homepage view of application

Figure 16 shown above is an image of the home page of the application. It is the landing page of the website, also known as the front page or the main page. When the client inserts the URL of the website, they are directed to this page.

There is a carousel in the main section. This highlights the picture and media that provide the information about the organization. Below this carousel, there is a text section. This is mainly used to give overall introduction to the organization at one glance. Similarly, there are other details on the right side. The topmost part includes a message from current president of the organization and his image. Below that there is a button which links the user to the member registration page. This link can also be opened by clicking at Register at the navigation bar. Below the new member button, there is a brief description of upcoming events. Only the most recent two events are shown on this page.

4.4.3 Login, Registration and Dashboard

The authentication is a major part of the application which decides how much visibility the user has in the UI. Figure 17 illustrates what the registration page looks like. This page is used to apply for new membership.

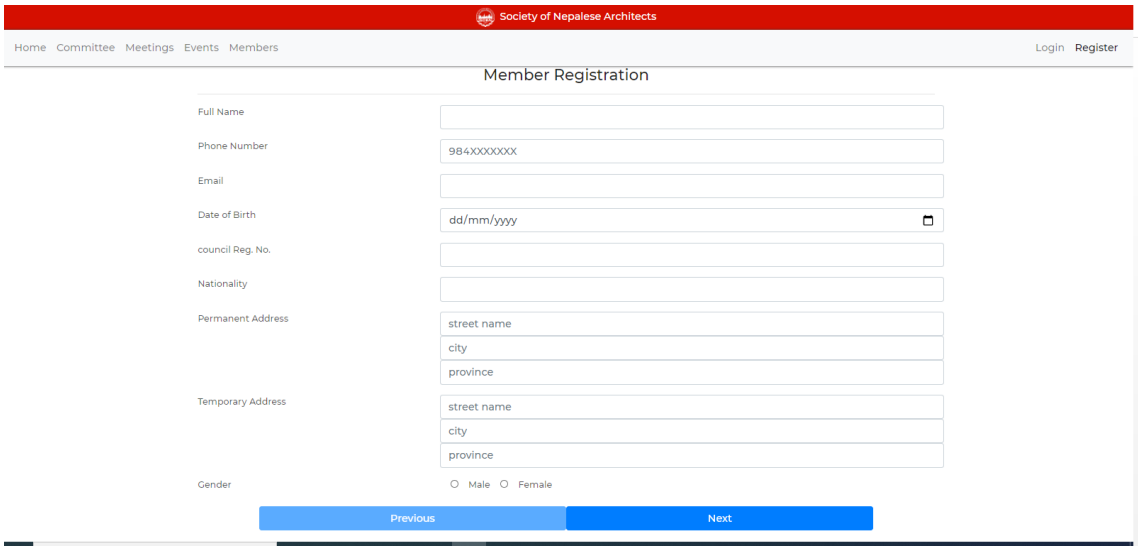

Figure 17: Membership registration request form

Figure 17 above shows how the registration form is displayed. For membership registration, a step form has been used in this application. The first step of the form is used for personal details. It includes information such as the name of the user who wants to register, contact details, and date of birth. The second step of the form is used to collect official information. It records the office name, the position of the user in the company, and the contact details of the workplace. There is also section in the form which is used for collecting information about the past working history of the user. The next step includes collecting information about the academic records of the user (?). In the next step there is a field to collect the required files of registration. This step has a field for academic documents, council certificates, national cards, and profile pictures. The final steps add the recommendation from any member.

After the member request is sent by the applicant, it is approved by a user with administration rights. This action automatically creates a user account for the member. Once the request is approved, the user can be directed to the sign in page by the administrator. On the login page there are the fields for email, password, and user type. In this model, the email was used as the primary key to recognize the user is unique. Since the user

can have multiple emails to create a fake account, it is planned to replace the unique ID by a council certificate number. After the user has been able to login successfully, the user is redirected to the dashboard. If not, a notification about failure is displayed. Figure 18 shows more details.

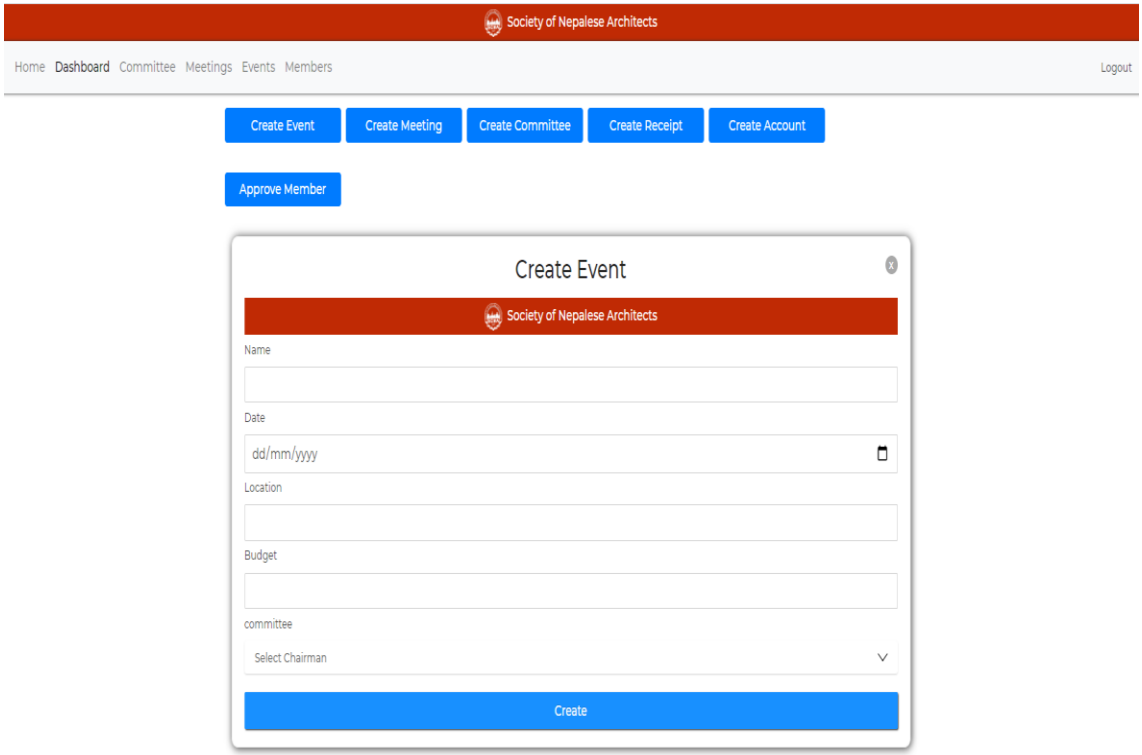

Figure 18: Event form view

The figure shown above is an image of the dashboard of the application. This dashboard consists of various operating tools at one point. It acts as a central system of the application operation. The content displayed in the above figure is for administrator user. Different buttons are present for creating events, meetings and committees. Collapse is used to display the respective item when the button is clicked. The display tabs are recognized using the ID given in the component. If the user is logged in as a member, then some of the elements are filtered.

4.4.4 Member

Member management is one of the most crucial parts of an organization. Member information is divided into three parts. First part is the information which can be shared publicly such as membership status, membership type and membership beginning date of

the member. The second type of information is semiprivate which can only be circulated within member of this organization such as current office address, academic history of member and current professional status of member. Third type of information is strictly private and this type of information are only used for official work or to the admin user. For filtering the user type, the application uses the data stored in the local storage. If the storage contains the email, the user ID or token, then all the information is displayed. The first member page can be reached via 'base_url/members'. This will take the client to the place where all the approved members of the organization are listed. They are arranged according to the membership ID. Similarly, if the user is an administrator, then there is a toggle component above the member list table.

Figure 19: Membership list page view

The above figure 19 shows the interface used for the membership list. The first column is for a serial number; the second column displays the full name of the member; the third column shows their membership number; and the last column displays the anchor tag for the link. This tag is hidden if the user is not logged in. When the link is clicked, the route is redirected to the individual member details. This individual has the same member URL and ID attached at the end. An example of the individual page of a member can be seen in figure 19.

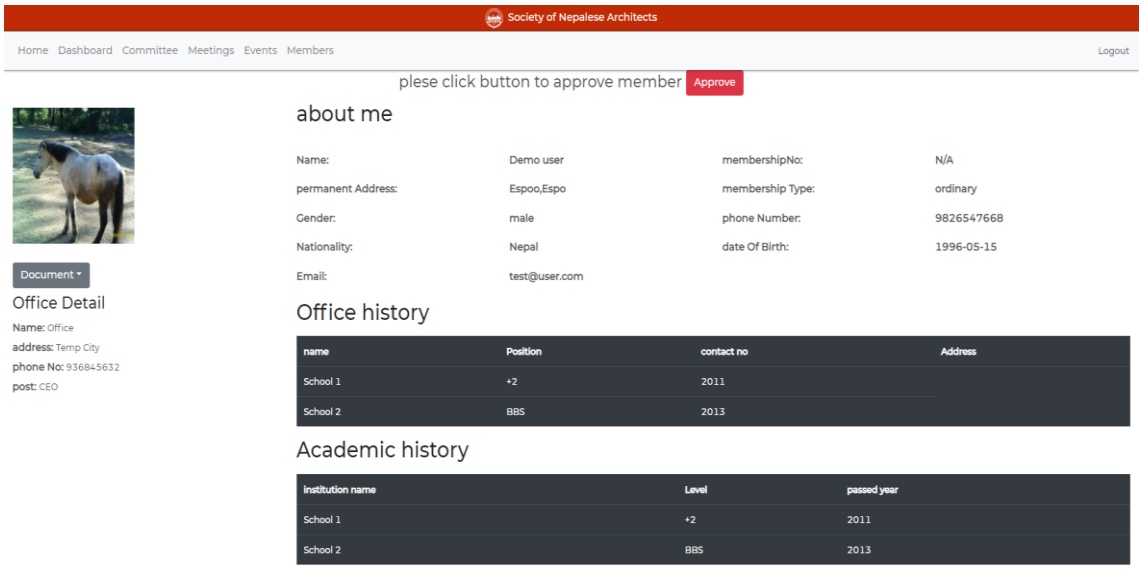

Figure 20: Member profile page for an unapproved member

Figure 20 shown above describes the interface for member details. This interface shows all the information of the selected members. The above figure is a sample of the application request for membership. The left-hand side of the body shows the profile picture of the member followed by the document links that are submitted while applying. Similarly, the topmost part of the body shows the approval information with the button which dispatches the approval request. This is only displayed for an unapproved member. Below this information, personal information is displayed. This application has also a table for displaying the office history and academic history of the member.

4.4.5 Meeting, Committee and Event

This application different pages for meetings, committees and events. These pages provide a list of their respective information. When the link attached to the list is clicked, then URL is changed and display is navigated to the individual page. There it shows the individual information. Figure 21 illustrates the event page view.

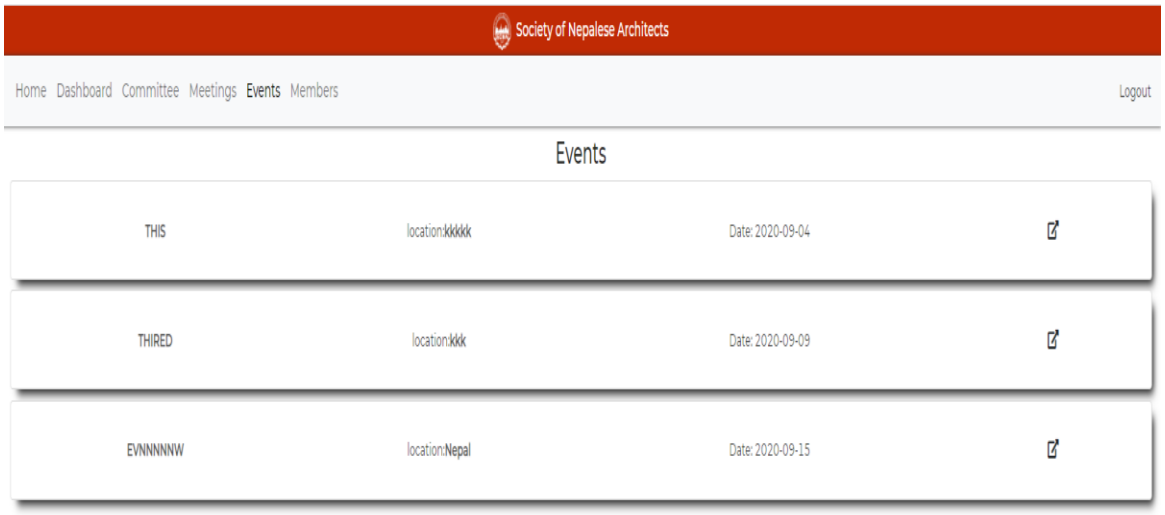

Figure 21: Event list page

Figure 21 shows a list of events that are organized the organization. The events are described with various color codes to give the status to the user at a single glance. When the user clicks the redirect, icon attached to end of each card, the interface redirects the user to event individual page where detail information of the event is shown.

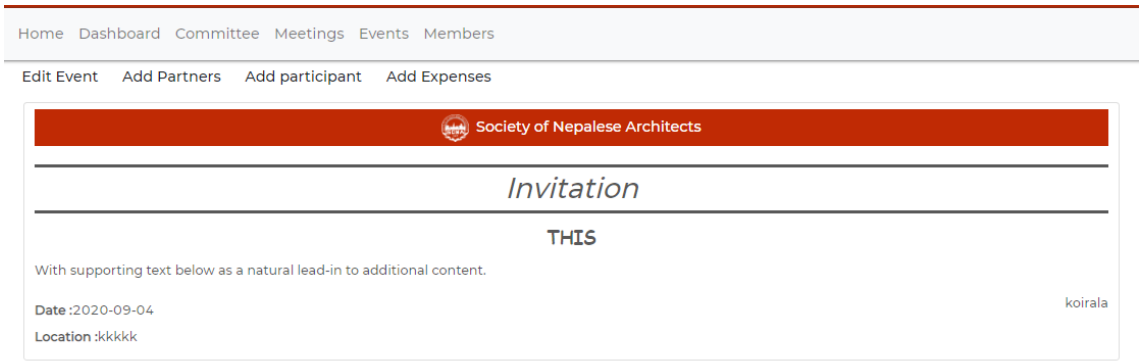

Figure 22: Event page

Once the user is redirected, the page shows details of the particular page. Figure 22 also shows event details and how the data is described by the schema of the event. The above figure shows the date and location of the event. It also shows the components to edit the event such as an edit event, add partners, add participants and add expenses. This functionality is shown only if the user is an administrator.

4.4.6 Income and Expenses UI

As an additional feature of the application, there is a small UI for tracking the financial condition of the organization. There is a form which records the expenses made by the organization. This field has a date, details of expenses and amount of expenses. This UI is also added to the event as expenses for that particular event.

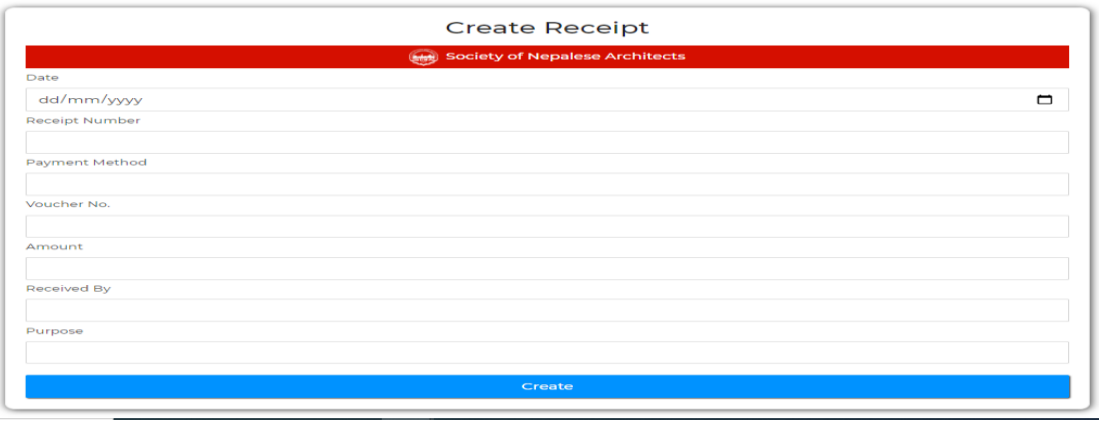

Figure 23: Receipt form view on the account page

Similarly, it also has a UI for recording incomes. This is recorded as a receipt for all income. The income has other fields besides expenses. It has a payment method which has some select value. There is also a field for the bank voucher number if the payment is done through the bank. It also has a field for recording the ID of the user who is creating the receipt. Generally, the receipt is issued to individuals or organizations. There is also a field to track the records of the recipient.

5 Application Programming Interface (API)

Application Programming Interface (API) in brief is an interface or communication protocol between a client and a server. A freely accessible web-based API returns information, likely in JSON (JavaScript Object Notation) or XML (Extensible Markup Language). An API is not the database or the server; it is basically a code that oversees the get-to-point for the server. The API gives a quick, steady, and reliable way to induce third party information. [12.]

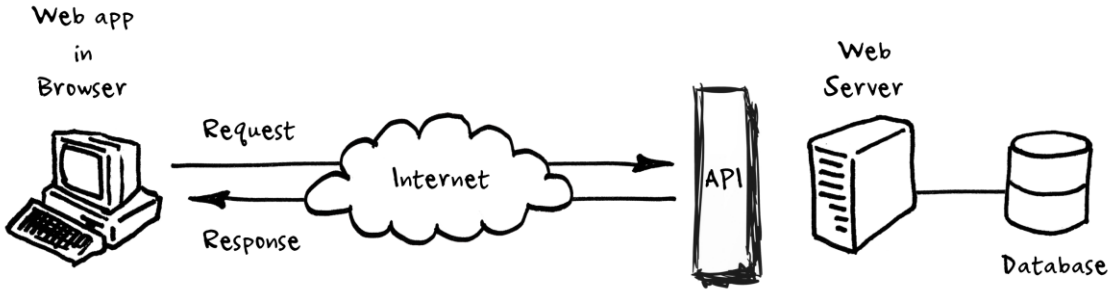

Figure 24: API accessing the database through an access point [12]

The above figure 24 shows the process where the API receives the request and sends back the response in JSON or XML format.

For this project, Base Uniform Resource Locator (URL) was set in the backend at environment. This base URL depends upon the platform where the backend is deployed, and additional routes are added above it. There is also base configuration given to the frontend. There were ten API endpoints used in the creation of the application which exchange data in the JSON format.

The list API used in the application is described in table 1.

Table 1: API endpoint of the application

End Points	Use
/api/admin	To create and delete admin
/api/login	To login different user
/api/members	To perform CRUD operation for member

/api/account	To handle the CRUD operation of individual member
/api/event	To perform CRUD operation for event
/api/meeting	To perform CRUD operation for meeting
/api/receipt	To perform CRUD operation for receipt
/api/expenses	To perform CRUD operation for expenses

From the APIs listed in table 1, creating the administrator, login and member does not require any headers other than a header of the content type. However, for creating events, meetings, receipts and expenses, an administrator or staff user token is required. There is also an additional API for membership approval which can only be accessed by an administrative person with authentication token.

6 Cost Estimation and Budget for Project

Like in any project, some costs occurred in the development of the application created in this project. The costs have been listed in table 2. What is more, the project is not finalized yet, so there are still some future costs which have to be estimated.

Table 2: Cost estimation of the project (until now)

Title	Amount (€)	Remarks
Physical resources (laptop, office space, etc.)	3,000	
Development resources (code editor, license, etc.)	0	Using opensource development tools
Labor costs (approximately 800 hours)	14,400	Estimated cost
Miscellaneous overheads	2,000	Estimated cost
Total	**19,800**	

The above table shows the estimated costs for the development of the application. There is always unseen overhead expenses while continuing the project to final phase which is not included in above table.

Until now there were no big threats in the development process, but in the current situation the COVID-19 pandemic has had a negative impact on the development. Developers are not able to focus on the product and there have been some mental and psychological hindrances. This may increase the development time and increase the labor costs.

7 Conclusion

The main goal of this thesis was to implement React and Node.js along with other supporting technologies in real life implementations. This project has become a good example on how to help a manually running organization to adopt digital services to enhance the business operation of the organization. The service provided by the developed application helps to bring the customer and the service provider closer. This a small step to change the structure of the organization, making it digital.

While looking at the application in detail, various difficulties have been identified in implementing React, Redux and Node.js. Detailed analysis of the application shows that various difficulties occurred in implementing the technology in the real-world application. Overall, the development process required learning new skills. It was difficult to determine how to create and actualize APIs in ReactJS and to oversee states with the Redux state development tools. Reducer, Action, and Store are three rules that were followed while utilizing Redux. Consequently, overseeing and connecting with the state tree was made simple and proficient. Moreover, a functional component of ReactJS was utilized for the UI part, and a class component of ReactJS was utilized for calling the activity and recovering information from the Redux state into the application. Subsequently, it was inferred that the utilization of the state in web application is the board framework. In conclusion, the combination of React, Redux, MongoDB, Express, and Node.js has given a small and powerful application.

This project has shown that digitalization can be implemented not only in entertainment and technological world but also in an organization which is not operating in the field of IT (Information Technology). This also shows the knowledge and implementation of technologies is expanding rapidly in every sector.

To summarize, this thesis gives information about the integration of React and Node.js with supporting libraries in modern development practice. It also demonstrates the benefits of using a web application over the manual work methodology.

The initial version of the application is ready for deploying. The future plan is to expand additional features that are not available in the current version of the application. Some plans have already been implemented and some are in development.

References

1. Bryant, Martin (6 August 2011). 20 Years Ago Today, the World Wide Web Opened to the Public [online]. TheNextWeb. Available at: https://thenextweb.com/insider/2011/08/06/20-years-ago-today-the-world-wide-web-opened-to-the-public/ [Accessed 2 November 2020].

2. Aston, Ben (1 April 2015). A Brief History of JavaScript [online]. Medium. Available at: https://medium.com/@_benaston/lesson-1a-the-history-of-javascript-8c1ce3bffb17 [Accessed 2 November 2020].

3. Rottinger, Jim (4 June 2019). The Role of JavaScript in the Modern Web [online]. Medium. Available at: https://medium.com/better-programming/the-role-of-javascript-in-the-modern-web-ff0f6961829a [Accessed 2 November 2020]

4. Computer Hope (2020). HTML Documentation [online]. Available at: https://www.computerhope.com/jargon/h/html.htm [Accessed 24 June 2020].

5. Hiwarale, Uday (23 April 2018). How Does JavaScript and JavaScript Engine Work in the Browser and Node [online]? Medium. Available at: https://medium.com/jspoint/how-javascript-works-in-browser-and-node-ab7d0d09ac2f [Accessed 24 June 2020].

6. Wikipedia (no date). React (web framework) [online]. Available at: https://en.wikipedia.org/wiki/React_(web_framework) [Accessed 2 November 2020]

7. Ivanov, Alexey and Andy Branov (28 March 2018). Optimizing React: Virtual DOM Explained [online]. Available at: https://evilmartians.com/chronicles/optimizing-react-virtual-dom-explained [Accessed 2 November 2020]

8. Hamedani, Mosh (19 November 2018). React Lifecycle Methods-A Deep Dive [online]. Available at: https://programmingwithmosh.com/javascript/react-lifecycle-methods/ [Accessed 2 November 2020].

9. Hlebowitsh, Nadia (10 June 2019). Stats on Top JS Frameworks: React, Angular & Vue [online]. Available at: https://www.tecla.io/blog/2019-stats-on-top-js-frameworks-react-angular-and-vue [Accessed 24 June 2020].

10. Sharma, Radium (21 March 2018). ABC of Redux [online]. Available at: https://dev.to/radiumsharma06/abc-of-redux-5461 [Accessed 20 July 2020]

11. StrongLoop/IBM (no date). Express Documentation [online] Available at: https://expressjs.com/en/starter/installing.html [Accessed 24 June 2020].

12. Shrestha, Ujwaj (2020). Implementing API in React [online]. Avalilable at: https://www.theseus.fi/handle/10024/345264 [Accessed 28 June 2020].