



VAASAN AMMATTIKORKEAKOULU
UNIVERSITY OF APPLIED SCIENCES

Khoa Bui

APPLICATION FOR BOOKING MOVIE TICKETS ONLINE

Technology

2022

ABSTRACT

Author	Bui Khoa
Title	Application for booking movie tickets online
Year	2022
Language	English
Pages	62 + 2 Appendices
Name of Supervisor	Anna-Kaisa Saari

The objective of this thesis is to design and implement a full-stack application for booking movie tickets. The front-end is based on React framework with the Redux library, the back-end is implemented using NodeJS.

This thesis covers the design and implementation of a front-end web application for a movie booking website and the back-end side handled data by NodeJS. The web is hoisting on surge which is a cloud platform for hosting static website and is extremely simple to use but it also offers customization options for those who need them. Data is given and handled by Swagger to get api and handle basic function like Post, Put, Delete, Get. Swagger is an Interface Description language used to describing Restful Api expressed using Json which are synchronized at the same speed as the server and client, generated a Rest Api document so that I can use it to interact with the Rest API, giving me the clear insights into how the API responds to parameters. This website is a Single Page Application, and the user can interact with the web browser with some function like I describe above. In addition, the app uses Tailwind CSS and Ant-design for styling components to make it good looking and eye-contact with the user.

The research study will provide an insight of how the communication mechanism between client and server in a current RestFul service-oriented web model works and how to communicate across system components by creating and testing the web-based.

Keywords

CONTENTS

ABSTRACT

1	INTRODUCTION	11
1.1	STUDY CASE.....	11
1.2	SCOPE AND OBJECTIVES	11
2	TECHNOLOGIES.....	12
2.1	React	12
2.1.1	Advantages of React.....	12
2.1.2	Disadvantages of React.....	13
2.1.3	JSX.....	13
2.1.4	Props and State.....	14
2.2	React-router.....	14
2.1	Javascript ES6	14
2.2	Redux (Action, Reducer, Store).....	17
2.3	Bootstrap 4.....	19
2.4	TailwindCss, Ant-Design.....	19
2.5	Surge	22
2.6	Nodejs, Swagger	22
3	ENVIRONMENT SETUP.....	24
3.1	INSTALLING VISUAL CODE.....	24
3.2	INSTALLING NODEJS BUNDLES AND NPM	25
3.3	Create-React App	25
3.4	installing some othjer dependencies.....	26
3.5	Architecture of website.....	27
3.6	Use Case Diagram.....	29
4	DESIGN	31
4.1	User Home Page.....	34
4.2	Header and Footer	34
4.3	Detail Movie Page	35
4.4	Booking Page.....	36
4.5	News Page.....	38
4.6	Admin Page.....	39
4.7	LogIn Page.....	41
5	IMPLEMENTATION.....	42
5.1	Server Configuration	42
5.2	Client Configuration	45
5.3	Function Implementation.....	46
a)	User Login.....	46
b)	Carousel.....	47
c)	Booking Carousel.....	49
d)	Advertise Landing Page.....	52
e)	Detail Page.....	53
f)	Booking Page.....	53
g)	Admin Page.....	55

h)	Multilingual Language.....	56
i)	Window Scroll Effect.....	58
6	TESTING AND DEPLOYMENT	59
7	CONCLUSION	61
8	REFERENCE.....	62

LIST OF FIGURES

Figure 1.	Data Flow Redux.....	19
Figure 2.	Swagger UI of the project.....	23
Figure 3.	Bootstrap extension.....	25
Figure 4.	React extension.....	25
Figure 5.	Check version of node and npm package.....	26
Figure 6.	File structure after create new project of react app	27
Figure 7.	Libraries of other dependencies.....	28
Figure 8.	Diagram of application.....	30
Figure 9.	Admin Diagram.....	30
Figure 10.	User Diagram.....	31
Figure 11.	User Page.....	32
Figure 12.	Movie List of Home Page.....	33
Figure 13.	Movie available in each theatre location.....	33
Figure 14.	Movie advertise section.....	34
Figure 15.	Movie banner.....	34
Figure 16.	Header of website.....	35
Figure 17.	Footer of website.....	35
Figure 18.	Detail movie page.....	36
Figure 19.	Different seat can book in theatre.....	37
Figure 20.	Booking ticket page.....	37

		6
Figure 21.	Seat chosen when booking.....	38
Figure 22.	History of all film booking.....	38
Figure 23.	News of movie.....	39
Figure 24.	Admin page.....	40
Figure 25.	User account in admin page.....	40
Figure 26.	Adding new film in admin page.....	40
Figure 27.	Login Page.....	41
Figure 28.	File structure of the project.....	43
Figure 29.	Action rule.....	45
Figure 30.	Reducer file.....	46
Figure 31.	File server of project.....	47
Figure 32.	Unique Id detail.....	54
Figure 33.	Translation language in system.....	58
Figure 34.	Hook used to come to top of the page.....	59

LIST OF CODE SNIPPET

Code snippet 1. Create HTML element	14
Code snippet 2. Create HTML element with JSX	14
Code snippet 3. Example of using React Router.....	15
Code snippet 4. Function Declaration with Javascript	16
Code snippet 5. Using Arrow Function in ES6	16
Code snippet 6. Assign a variable to an obj.....	16
Code snippet 7. Destructuring with ES6 JavaScript	17
Code snippet 8. Rest operator	17
Code snippet 9. Spread Operator	17
Code snippet 10. Bootstrap CDN link	20
Code snippet 11. Install package Ant-design.....	21
Code snippet 12. CDN link of Ant-design.....	21
Code snippet 13 . Import directly to the component.....	21
Code snippet 14. Install package TailwindCSS.....	21
Code snippet 15. Configure file TailwindCSS.....	22
Code snippet 16. CDN Link of TailwindCss	22
Code snippet 17. An example of HTTP server rendering using	24
Code snippet 18. Home Template of Project	44
Code snippet 19. Home Template of components.....	44

	8
Code snippet 20. Store	46
Code snippet 21. Base Service	47
Code snippet 22. Method for log in.....	48
Code snippet 23. Using Formik to create form.....	48
Code snippet 24. Method Carousel Banner.....	49
Code snippet 25. useEffect to get an API	49
Code snippet 26. Reducer of banner feature	50
Code snippet 27. Reducer of List Film in Home.....	51
Code snippet 28. State default of Film	51
Code snippet 29. useEffect to call api from backend	52
Code snippet 30. Action of list films	53
Code snippet 31. News of Landing Page.....	53
Code snippet 32. Detail page Method.....	54
Code snippet 33. Booking Page Method	55
Code snippet 34. Style for each Seat	55
Code snippet 35. Method for counting the total money of booking tickets.....	55
Code snippet 36. Method renderTicketItem	56
Code snippet 37. Method for displaying movie in Admin Section	57
Code snippet 38. Method for Editing film	57
Code snippet 39. Method for AddingMovie.....	58
Code snippet 40. I18N configuration.....	58

LIST OF ABBREVIATIONS

APIs	Application Programming Interfaces
CMS	Content Management System
DOM	Document Object Model
Env	Environment
HTML	Hyper Text Markup Language
IDE	Integrated development environment
ISR	Incremental Static Regeneration
JS	JavaScript
OP	Operating system
SEO	Search Engine Optimization
SPA	Single Page Application
SSR	Server-side rendering
SSG	Static Site Generation
SQL	Structured query language
UI	User Interfaces
URL	Uniform resource locator
CDN	Content delivery network
CSS	Cascading style sheets
SSL	Secure sockets player
CLI	Command-line interface

CI	Confidential informant
HTTP	HyperText transfer protocol
VSC	Vertical service code

1. INTRODUCTION

1.1 Study Case

With the development of technology these days, people can book tickets for watching movies easily without going to the theatre to purchase them. With an application, it is easier and faster for customers to buy tickets online. An online booking system has benefits for both customers and theatre. In addition, the website will be a good place to increase the visibility and availability of the theatre tickets, which improves customer satisfaction. The website can collect and store user information in local storage. This data can be used for providing essential information for each movie but also for establishing a unique background for the user. This project consists only of front-end development and the back-end is used via Swagger from api.

1.2 Objective

The goal of the project is to implement an application for booking movie tickets online. The ticket booking application is an internet-based application, which means that the application can be accessed from anywhere on the network. This online ticket reservation system provides a cinema website that is accessible to all internet users. Users are required to log into the system and process checkout after booking the ticket is done. The purchased tickets can then be picked up at the ticket office or they can be sent to the customer email 's address.

The goal is to implement a web site, which provides information about the current movies at the theatre as well as the details of show times and availability. Ticket reservations are made using a credit card and the reservations can be cancelled if needed. An online ticket reservation system is one of the best applications for people who do not have time to purchase tickets or wait in long lines.

2. TECHNOLOGY

2.1 React

React is a JavaScript framework library that allows developers to create simple, quick, and scalable online applications. React is efficient, versatile, and open-source framework built by Jordan Walke, a software engineer at Facebook. React was originally used on Facebook's news feed in 2011 and then on Instagram in 2012. React allows developers with JavaScript experience to effortlessly create online applications. /1/

React Hooks allows developers to use the state and other features of React. React Hooks are the functions that will connect React state with the lifecycle features from the function components. React Hooks is among the features that are implemented latest in the version React 16.8. React Hooks uses a component-based methodology, which makes it useful when creating complex and reusable user interface (UI) components for mobile and online applications. /1/

The important features of React are server-side rendering support and usage of the virtual Data Object Model, DOM, rather than real DOM. React also follows unidirectional data binding or data flow and it uses reusable or composable UI components for developing the view.

2.1.1 Advantages of using ReactJs?

React uses Virtual DOM to improve efficiency which enhance its performance. Virtual Dom is a virtual representation of the real Dom, as the name implies. A new virtual DOM is built every time the data in a react application changes. It's significantly faster to create a virtual Dom than it is to draw the UI within the browser. As a result of the use of virtual Dom, the efficiency of the application improves.

When compared to frameworks such as Angular, React has a more moderate learning curve. Anyone with a basic understanding of JavaScript can use React to create web applications. React also gives the freedom to choose the tools,

libraries, and architecture for designing the application based on the requirements.

ReactJs is also search engine optimization (SEO) friendly. Developers can utilize React to create compelling user interfaces that are easy to traverse in multiple search engines. ReactJs also enables server-side rendering, which improves the SEO performance of an application.

ReactJs components are reusable. With ReactJs applications are developed using a component-based architecture. Components are self-contained, reusable code chunks. These components can be reused in a variety of similar-functioning apps. The reuse of components speeds up the development process. /2/

2.1.2 Disadvantages of Using React.

React has some limitations, however. Because it is merely a library, React is not a full- fledged framework. The benefits of all React 's components are enormous, but it will take time to properly comprehend them all. Beginner programmers may find it difficult to grasp React. Because inline templating and JSX will be used, coding may become more difficult. /2/

2.1.3 JSX

JavaScript XML is abbreviated as JSX. JSX enables developers to build HTML in JavaScript and insert it into the DOM without the need for basic React operations like `appendChild ()` or `createElement ()`. JSX provides syntax for React. /3/

Code snippet 1 shows an example of how element is created when using basic React operations and Code snippet 2 shows an example how element is created when working with JSX HTML components can be utilized directly inside JavaScript, as seen in Code Snippet 2. It directly uses HTML in a JSX file.

```
const text = React.createElement('p', {}, 'This is a text');
const container = React.createElement('div', '{}', text );
ReactDOM.render(container, rootElement);
```

Code snippet 1. Create HTML element

```
const container = (
  <div>
    <p>This is a text</p>
  </div>
);
ReactDOM.render(container, rootElement);
```

Code snippet 2. Create HTML element with JSX

2.1.4 Props and State in ReactJS.

Props is Reacts abbreviation for Properties. Props are read-only components that must be maintained immutable and pure. Throughout the program, Props are always handed down from the parent to the child components. A prop can never be sent back to the parent component by a child component. This helps in the preservation of unidirectional data flow and is commonly used to show dynamically created data. /4/

React components are built around states. States serve as a data source and should be made as basic as feasible. In a nutshell, states are the objects that define how components are rendered and behave. They, unlike the props, are malleable and produce dynamic and interactive elements. States can be accessed via this. State (). /4/

2.2 React-Router

The ReactJS Router is mostly used when creating Single Page Web Applications. In the application, React Router is utilized to define multiple routes. It is used to define and render component based on the specified path. It will accept components and render to define what should be rendered.

When a user types a specific URL into their browser, and that URL path matches one of the 'routes' in the router file, the user is routed to that route. For example, routes can be used in navigating to each page of the project. The user can be moved to another page using the React-Router on a specified declared path, as seen in Code Snippet 3. Each element specifies the component to which the user wants to be redirected as well as the path to the additional link at the top of the browser. /5/

```
<Route path="/" element={<App />}>
  <Route path="sales" element={<Sales />}>
    <Route path="invoices" element={<Invoices />}>
      <Route path=":invoice" element={<Invoice />} />
    </Route>
  </Route>
</Route>
```

Code snippet 3. Example of using React Router

2.3 JavaScript ES6

JavaScript Es6 is the newer version of JavaScript, released before JavaScript in 2015. JavaScript ES6 offers new functions that can be used for making code clean. Code Snippet 4 shows an example of function declaration with JavaScript and Code Snippet 5 shows how it can be done using ES6 Arrow function. /6/

```
// function expression
let x = function(x, y) {
  return x * y;
}
```

Code snippet 4. Function Declaration with Javascript

```
// using arrow functions
let x = (x, y) => x * y;
```

Code snippet 5. Using Arrow Function in ES6

Another example of ES6 benefits is destructuring. In Code Snippet 6 with JavaScript, the user can declare and get variable information by getting to specific things such as name, age, gender. In Code snippet 7 the object can get the properties variable via destructuring in JavaScript ES6

```
// before you would do something like this
const person = {
  name: 'Sara',
  age: 25,
  gender: 'female'
}

let name = person.name;
let age = person.age;
let gender = person.gender;

console.log(name); // Sara
console.log(age); // 25
console.log(gender); // female
```

Code snippet 6. Assign a variable to an obj

```
const person = {
  name: 'Sara',
  age: 25,
  gender: 'female'
}

let { name, age, gender } = person;

console.log(name); // Sara
console.log(age); // 25
console.log(gender); // female
```


Code snippet 7. Destructuring with ES6 JavaScript

JavaScript ES6 Rest and Spread Operator are used in Code snippet 8 and Code snippet 9 which can be used to express an array with an indefinite number of arguments. In Code snippet 8, the rest of the parameters can be included in the function definition by using three dots (...) followed by the name of the array that will contain them. In Code snippet 9, spread syntax can be used when all elements from an object or array need to be included in a list of some kind. In the example below in Code snippet 9, spread can return the last line as many times, to keep adding an additional one two to the beginning of the array.

```
function show(a, b, ...args) {  
  console.log(a); // one  
  console.log(b); // two  
  console.log(args); // ["three", "four", "five", "six"]  
}  
  
show('one', 'two', 'three', 'four', 'five', 'six')
```

Code snippet 8. Rest operator

```
let arr1 = ['one', 'two'];  
let arr2 = [...arr1, 'three', 'four', 'five'];  
console.log(arr2); // ["one", "two", "three", "four", "five"]
```

Code snippet 9. Spread Operator

2.2 Redux (Action, Store, Reducer)

Redux is a useful solution for organizing application state in JavaScript applications since Redux is a predictable state container. Redux is a common framework for managing state in React projects, but it also works with Angular, Vue.js, and plain old JavaScript. Knowing when to use Redux is something that most people struggle with. The larger and more sophisticated the application becomes, the more likely it will profit from Redux. If the application is expected to be extended later, Redux should be taken into use from the beginning to provide easier maintainability as app changes and scales without changing a lot of the existing code. /7/

Action Creator, Reducer, and Store are the three main components of Redux's operating principle. Action is a pure function that will return an object with two attributes: type, which is the action description, and payload, which is the value of the provided parameter. Action is in charge of gathering data for the state without specifying how it will be used. The Reducer will be in charge of this.

Based on the pure function rule, the reducer accepts two parameters: the previous state and the submitted action and returns a new state without changing the old state. Store is an object that keeps track of the state of the entire application, connecting Redux state to React state, executing Action Creator, listening and updating whenever the application state is changed.

The store is responsible for managing the state of the application. Allowing access to the state using get State method (`getState()`), it also allows state updates to be made via `dispatch(action)`. The state of the entire application is stored in this variable which registers listeners using `subscribe(listener)` and unregisters listeners via the function returned by the `subscribe(listener)`

In short, Reducers are really all that is needed when building a shop. Figure 1 shows the Action flow: the action sends information from the application to the Store, describing what jobs will be done with this store. This information is an object describing what happened. The action consists of two parts, the type which describes the action and the value of the parameter passed */7/*

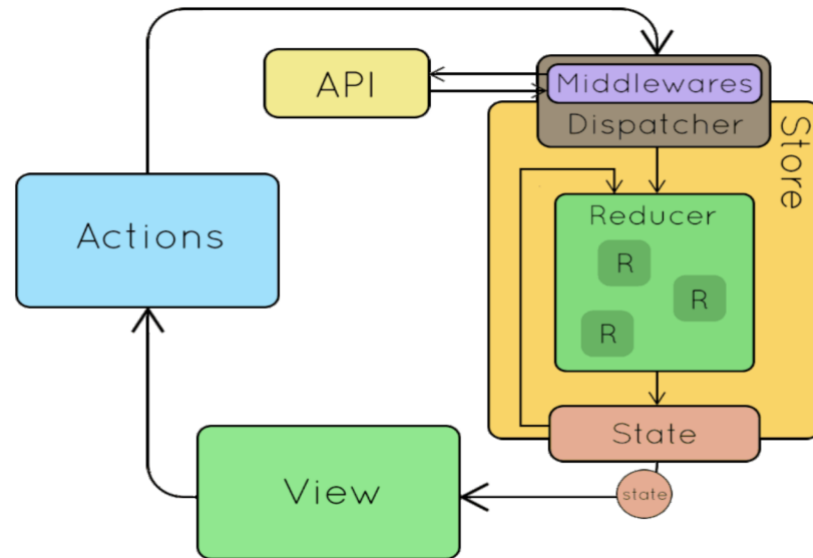


Figure 1. Data Flow Redux

The fact that all data in an application follows the same lifecycle pattern is one of Redux 's numerous advantages. Because Redux architecture maintains a tight unidirectional data flow, the logic of application is more predictable and easier to understand. We will explain shortly the flow of Redux action above. When a user interacts with the application, an action is dispatched. The dispatched action and the current state are passed to the root reducer function. The duty may be divided across smaller reducer functions by the root reducer, which eventually returns to a new state. By performing their callback routines, the store notifies the view. The view can re-render after retrieving updated state. /8/

2.3 Bootstrap 4

Nowadays, webpages are optimized for all browsers (Internet Explorer, Firefox, and Chrome) and screen sizes (Desktop, Tablets, Phablets, and Phones). Bootstrap is a toolkit for building responsive websites and web applications. Bootstrap is free and open source, initially created by Twitter's Mark Otto and Jacob Thornton. Bootstrap is the most widely used HTML, CSS and JavaScript framework for creating mobile-first responsive websites. Bootstrap fixes several issues experienced previously such as cross-browser compatibility. /9/

There are many advantages when using Bootstrap. The first one is that it is faster and easier to apply, the second one is generated web pages that are platform agnostic, the third is that it already makes responsive website, and finally it is completely free. /9/

There are two ways to use Bootstrap on a website, either by including Bootstrap from the CDN link or downloading Bootstrap directly from getbootstrap.com. Code snippet 10 shows how Bootstrap can be taken into use in application. /9/

```
Bootstrap CSS Library  
  
<link rel="stylesheet"  
  href="https://stackpath.bootstrapcdn.com/bootstrap/4.3.1/css/bootstrap.min.css"  
  integrity="sha384-ggOyR0iXCbMQv3Xipma34MD+dH/1fQ784/j6cY/iJTQU0hcWr7x9JvoR  
  xT2MZw1T"crossorigin="anonymous">
```

Code snippet 10. Bootstrap CDN link

2.4 Tailwind CSS and Ant-Design

Tailwind CSS is a utility-first CSS framework for rapidly building custom user interfaces. Ant-design is the UI design language for React implementation with a set of high-quality React components. Combined these two libraries to website it is flexible and easier to apply the CSS because we can change based on the libraries user want to use or change the style of components that apply to the website. It also offers Icon and available components that do not need to spend more time to design. It offers built-in methods to style components, finding the element style more quickly and makes the code more readable. /10/

The Ant-design package is taken into use by installing the package dependencies in project using npm or yarn, as shown in Code Snippet 11.

```
$ npm install antd
```

```
$ yarn add antd
```

Code snippet 11. Install package Ant-design

In Code snippets 12 and 13, after installing the package of ant-design, we can use this library to style for component. The project can import directly the CDN link of the ant-design to use or import to the specified file of the application.

```
https://cdnjs.cloudflare.com/ajax/libs/antd/4.19.2/antd.min.css
```

Code snippet 12. CDN link of Ant-design

```
import 'antd/dist/antd.css'; // or 'antd/dist/antd.less'
```

Code snippet 13 . Import directly to the component

Tailwind CSS is a utility-first CSS framework that allows to create and custom user interfaces quickly(see Code snippet 14). It is a highly configurable, low-level CSS framework that offers all the building blocks needed to create personalized designs. There are many ways to use Tailwind CSS to apply in the project, but the easiest way to integrate in the project is installing Tailwind via npm with the command `npm install Tailwindcss`

Terminal

```
> npm install -D tailwindcss  
> npx tailwindcss init
```

Code snippet 14. Install package TailwindCSS

After installing package in terminal, use the `@tailwind` directive is used to inject Tailwind's base, components, and utilities styles into CSS file (see Code snippet 15).

```
src/input.css

@tailwind base;
@tailwind components;
@tailwind utilities;
```

Code snippet 15. Configure file TailwindCSS

Now we can apply it in the project by using class name from tailwind CSS (see Code snippet 16). In the second method, file can be added through the CDN link from Tailwind, adding it directly to the folder public index.js of the project.

```
src/index.html

<!doctype html>
<html>
<head>
  <meta charset="UTF-8">
  <meta name="viewport" content="width=device-width, initial-scale=1.0">
  <link href="/dist/output.css" rel="stylesheet">
</head>
<body>
  <h1 class="text-3xl font-bold underline">
    Hello world!
  </h1>
</body>
</html>
```

Code snippet 16. CDN Link of TailwindCss

2.5 Surge

Surge is the single-command web publishing. It is used to deploy React Project, to publish html, CSS and Js for free Surge is a front-end deployment. When using surge, we can get some benefits such as:

- Free domain support
- Free SSL for surge.sh
- push State support for single page apps

- Custom 404.html pages
- Barrier-free deployment through CLI

In this project, Surge was used together with NodeJS for deploying the application. Surge can also blend with other build tools like Git Hooks and Nodejs, CI Services. /11/

2.6 Nodejs and Swagger

Swagger is a software tool used for creating, documenting, and accessing Restful Apis. It follows the openAPI specification. Swagger is a tool to handle data API from backend from swagger and render it in front-end-site. /12/

An Api (Application Programming interface) is a communication interface that connects two independent software applications. The Open Api specification is a set of guidelines for establishing Restful Api interfaces for describing, producing, consuming and visualizing them. /12/

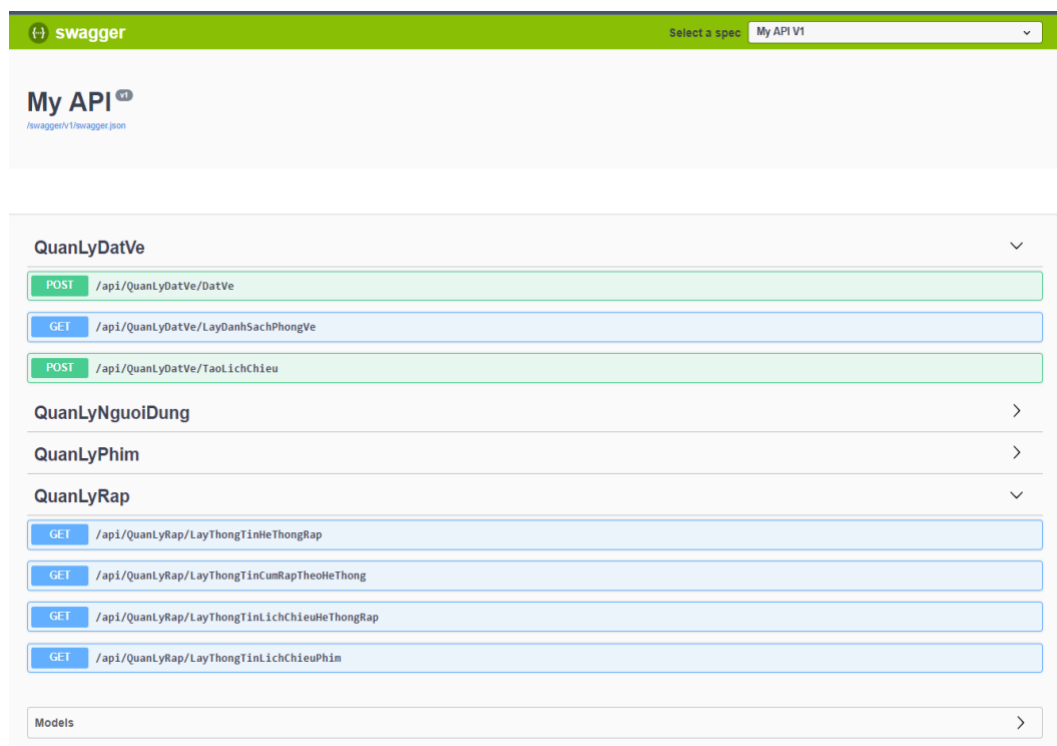


Figure 2. Swagger UI of the Project

With a visual perspective and quicker client-side deployment, Swagger automatically creates API documentation from the Swagger config file. Figure 2 describes all of the movie film APIs. /12/

Node.js is a runtime environment that allows JavaScript code to be executed outside of a browser. It is used to the back-end service of a front-end application, also known as API. Node.js is suited for creating applications that are extremely scalable, quicker, data-intensive, and real-time. Node.js has an efficient and lightweight Input/Output model that is non-blocking and event driven. Many prominent firms utilize it, including Netflix, LinkedIn, Trello, and PayPal. /12/

Code snippet 17 below describes an example of HTTP server created with NodeJS. On the server with hostname '127.0.0.1' and the port '3000', the code in the preceding figure returns the text 'Hello Node.js users'. The server also logs a message in command that says 'Server is operating at **Error! Hyperlink reference not valid.** /12/

```
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 Node.js users');
});

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

Code snippet 17. An example of HTTP server rendering using

3.Environment Setup

3.1 Installing IDE

IDE is an integrated development environment (IDE) which is a software suite that consolidates basic tools required to write and test software. Developers use numerous tools throughout software code creation, building and testing. Development tools often include text editors, code libraries, compilers and test platforms. Without an IDE, a developer must select, deploy, integrate and manage all these tools separately.

In this project, Visual Studio Code is used as IDE. The main reason for VSC is because it offers many features and code snippets which makes implementation faster. It is also easier to check errors by VSC featuring a built-in terminal system that allows developers to perform commands directly on VSC without needing to launch a terminal process. Another reason to choose VSC is that it comes with a built-in GIT command that allows checking diffs stage files and commit directly from the editor.

Furthermore, the Visual Studio market offers a significant number of libraries of extensions and plug-ins that support the coding of today's most popular programming languages. Figure 3 shows a Bootstrap extension for styling components in ReactJS, it is fast and easy to use when applying to the application. Figure 4 illustrates ES7+React/Redux/React-Native snippets which support React syntaxes, developers simply enter the snippets and then click the Tab button to see the full syntax for that snippet. /16/

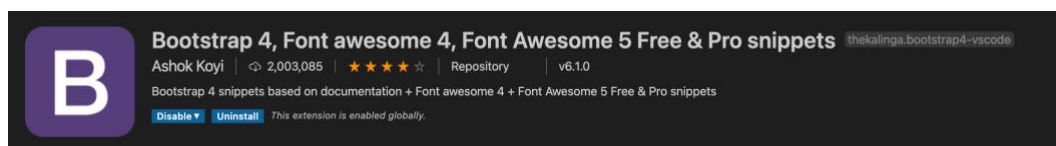


Figure 3. Bootstrap extension

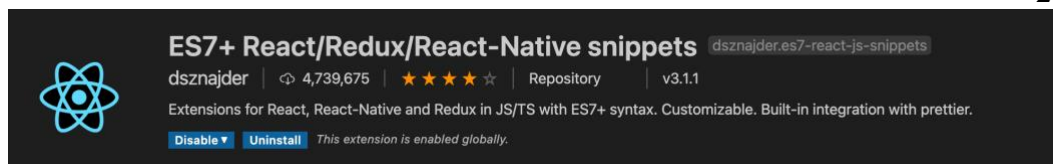


Figure 4. React extension

3.2 Installing NodeJS and NPM

Node version managers allow to install and switch between multiple version of Nodejs and npm so that Node can be tested for application to ensure that its can work on different versions. After downloading, NodeJS is installed directly to the computer and prompting is checked by typing the commands `node -v` and `npm-v`. This command is used to check the current version of node js and npm.

```
buinguyenkhoa@Buis-MBP nextJs-docker % npm -v
7.5.4
buinguyenkhoa@Buis-MBP nextJs-docker % node -v
v14.9.0
```

Figure 5. Check version of node and npm package

3.3 Create React-App

Create-react-app is a Facebook-provided official tool for developing React starting projects. It saves a lot of time from setup and defining settings when there is no need to configure web packs, node modules or dependencies. Installation of the Create-react-app tool and starting React project is done by running one command. The global package was installed first, then the Terminal (or Command prompt in Windows) was opened and the command: `npm install -g create-react-app` was typed. Then, using Terminal, and command 1 was run in the working directory.

We get a project the structure of which is shown in Figure 6 . After running the command, create-react-app generated the files structure like the image below.

/13/

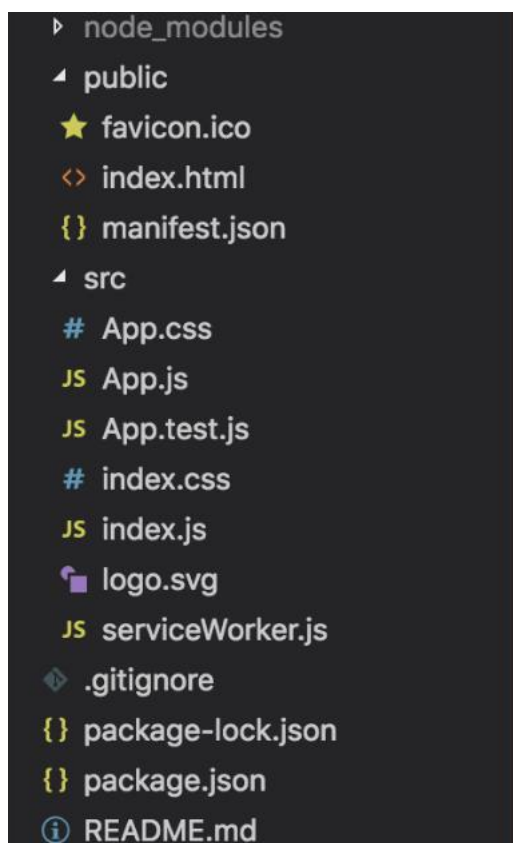


Figure 6. File structure after Create new project React-App

3.4 Installing Other Dependencies

The development process necessitates the use of Webpack. Webpack is a tool that makes it simple to build a project, which is referred to as "dependencies."

Figure 6 shows some dependencies that needed to be installed in the project: `npm install react-router-dom`, `npm install react-redux`, `npm install redux`, `npm install redux-thunk`, `npm install lodash`, `npm install tailwind CSS`. Each dependency has their own functionality for each component. For example, React-router was used to navigate the web page, Redux to manage all state of the component, Redux-thunk is a middleware library, Lodash was used to declare each function easily, and Tailwind for styling components. In addition, Axios was installed, to assist in establishing a connection between the client and server to handle backend services.

```

{
  "name": "react-project-bookingticket-movie",
  "version": "0.1.0",
  "private": true,
  "dependencies": {
    "@ant-design/icons": "^4.7.0",
    "@material-ui/icons": "^4.11.2",
    "@testing-library/jest-dom": "^5.16.1",
    "@testing-library/react": "^12.1.2",
    "@testing-library/user-event": "^13.5.0",
    "@tsamantanis/react-glassmorphism": "^1.1.2",
    "antd": "^4.18.6",
    "axios": "^0.25.0",
    "formik": "^2.2.9",
    "i18next": "^21.6.14",
    "i18next-browser-languagedetector": "^6.1.3",
    "i18next-http-backend": "^1.4.0",
    "lodash": "^4.17.21",
    "moment": "^2.29.1",
    "node-sass": "^7.0.1",
    "react": "^17.0.2",
    "react-dom": "^17.0.2",
    "react-i18next": "^11.15.7",
    "react-redux": "^7.2.6",
    "react-router": "^6.2.1",
    "react-router-dom": "^5.2.0",
    "react-scripts": "5.0.0",
    "react-slick": "^0.28.1",
    "redux": "^4.1.2",
    "redux-thunk": "^2.4.1",
    "sass-loader": "^12.6.0",
    "slick-carousel": "^1.8.1",
    "thunk": "^0.0.1",
    "web-vitals": "^2.1.4",
    "yup": "^0.32.11"
  },
}

```

Figure 7. Libraries of other dependencies

3.5 Architecture of Website

Online tickets booking application is an online project with a web-based component. The main goal of the project is to be able to use the system via the website at any time without having to visit the theatre. They can also acquire all information as a new user, such as available movies and, theatres.

1. Administrator module

The administrator can add and delete the movie and manage the user database in the website. In the administrator dashboard, several template views are needed for the admin users who can access and manage all activities of the association. First, it needs to be made sure that only admin user can access browser home page, so it needs to be checked that every time a user wants to book a ticket, the page redirects to a login screen. If the user is an administrator, the application will let them log in, otherwise the application will deny access.

2. User module

The user at first can submit their personal information to be saved in the local storage. When the user books the movie, they are required to login and reserve the film they already booked. The user model gathers all information, such as movie and theatre information.

3. Ticket reservation module

When the user chooses to book the ticket, the application shows all films and time available online. Figure 8, shows the overall processing of the application with all the phase of sending and receiving actions for the user and administrator.

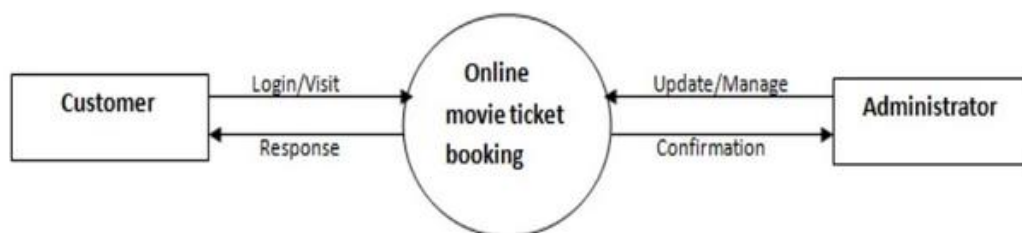


Figure 8. Diagram of application

3.6 Use Case Diagram

The administrator is the place to manage the films shown in the theatre and all the detailed user information logs on to the website. It has some functions such as adding new movies and deleting movies according to the demand of the admin. Admin use cases are shown in Figure 9.

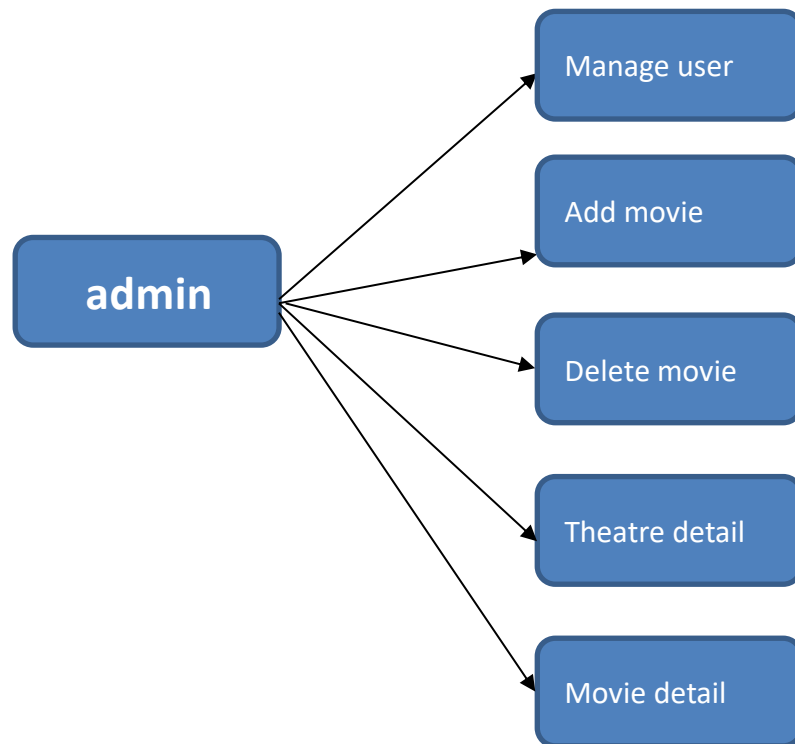


Figure 9. Admin Diagram

Figure 10 illustrates how the user can log in to book a ticket, choose a language, view seats available and reserve a seat or book a ticket.

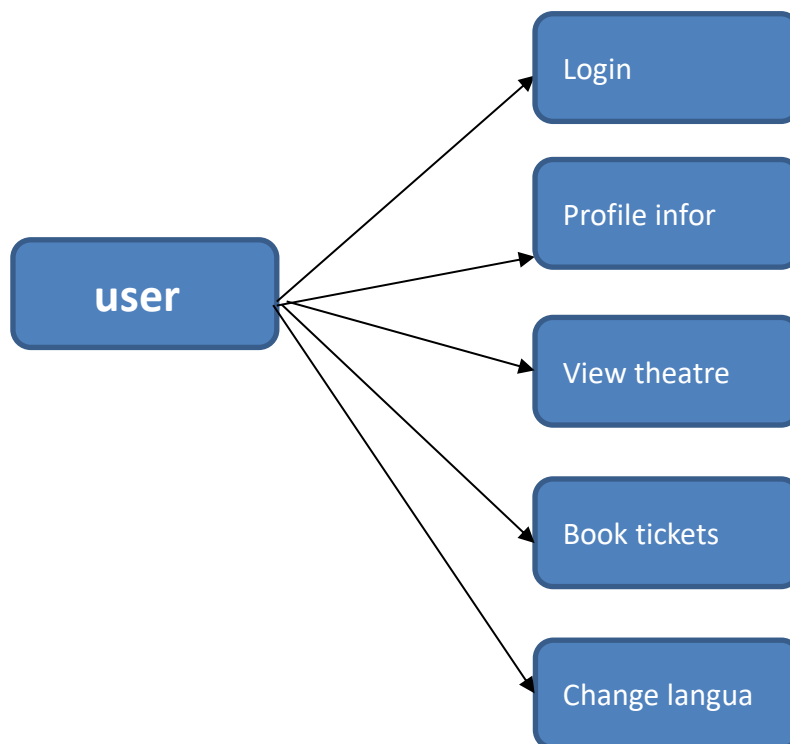


Figure 10. User Diagram

4 DESIGN

The project application is divided into two parts. The first one is User Interface which the user can interact with all the events on the webpage and the second one is the Admin page where the Admin has right to add, delete movies and show all account activities to see who accesses the website.

4.1 User homepage



Figure 11. User Pages

As shown in image Figure 11 : the user interface will have five pages for the user to interact. Web pages include homepage, detail Page showing the detail of each type of movie, Booking Page to view available time and seats, Login Page for the user to login. The last page views the news in 24hours of about each movie.

The Homepage is the site where the user can view all the important parts of the webpage. This page contains all the information needed on the website. The

Homepage contains sections such as Home, Contact, Sign In, Sign Out, Language Choices in Header, when coming to the middle of the homepage. It shows all the movies that are available now (Current Films) and in the future (Upcoming Films). The Advertise Section contains a short description about movies, invitations, downloading the application. It also includes the banner of three films at the beginning of the webpage.

Figure 12 describes a screenshot from the website's movies page. "Filter by Current Movies," "Movie Coming Soon" are the page's filter choices. The sorting by option allows the user to sort the movies depending on their popularity, release dates, and ratings.

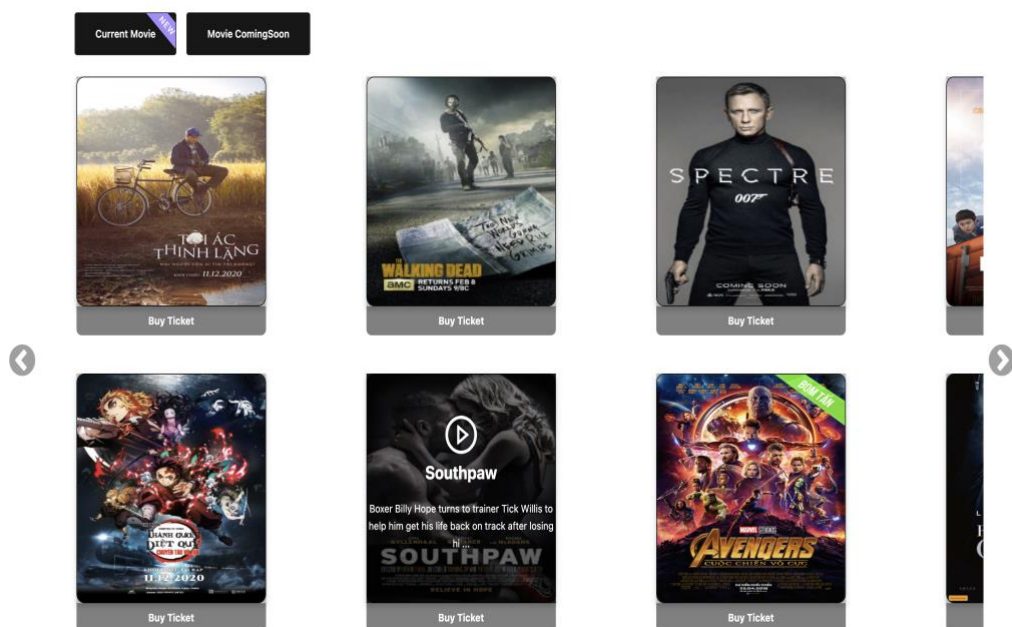


Figure 12. Movie List at Homepage

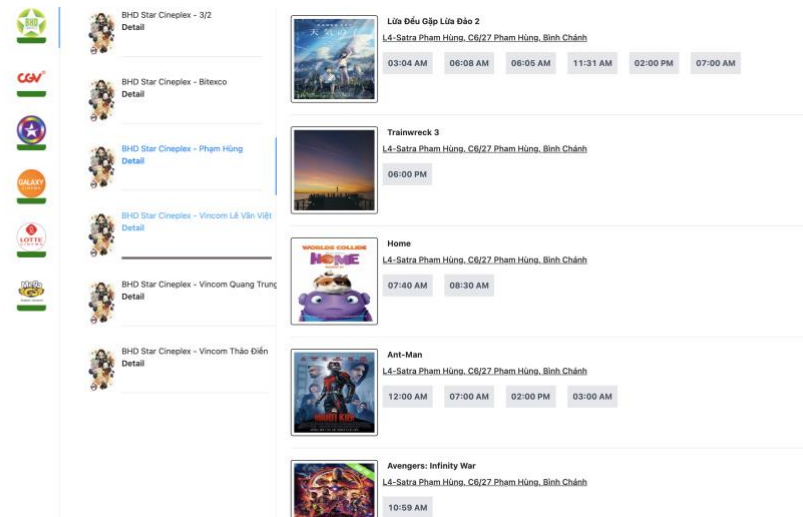


Figure 13. Movie available in each Theatre Location.

As shown in Figure 14, the homepage displays the advertised section where the user can download the application via a mobile phone by clicking on the orange button. Figure 15 shows a banner of a movie with the available times to watch the film.

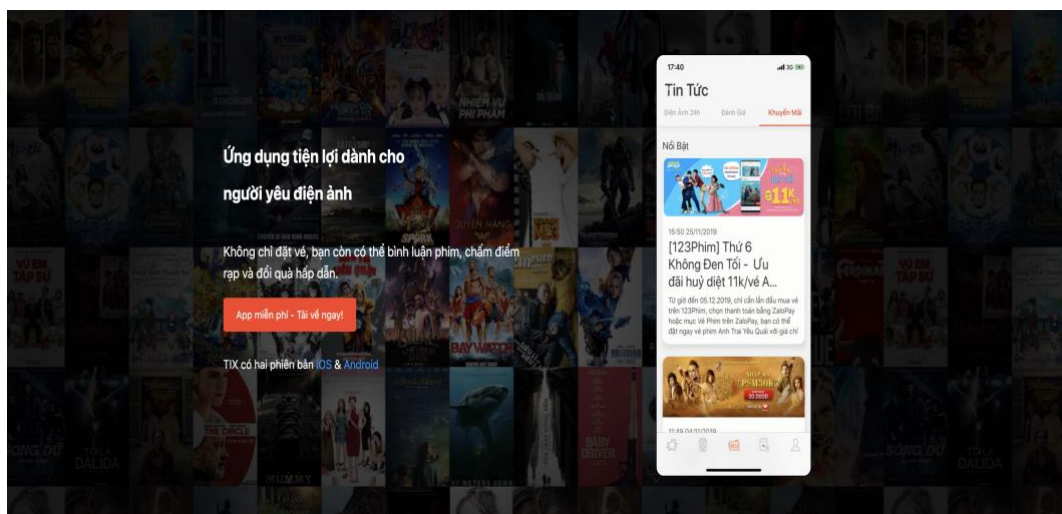


Figure 14. Movie Advertise Section



Figure 15. Movie banner

4.2 Header and Footer

When accessing the website, the first thing the user will see is the Header and the Footer of the page because this place shows all the menu list and services of the page. The navigation is always located on top of the webpage regardless of turning to another site. When the user scroll down, the menu will keep the navigation bar in view so that it is accessible without scrolling back to the top of the page. Figure 16 shows, the menu of the web page with the navigation bar for the user to navigate to other page or log in, log out by toggling the button.



Figure 16. Header of website.

The footer has a dark colour scheme that matches the header to set it out from the rest of the page. It also displays all of the collaborators on the film. An image of the website's footer is shown in Figure 17. The footer offers links to the website's other pages. It also shows the name of the website, as well as the current year and the copyright logo.



Figure 17. Footer of website

4.3 Movie Details Page

The movie details page contains all related information to the film which can be seen by clicking on the showtimes or on top of the movie shown on the home page. This section has a substantial quantity of content, as well as a button or page navigation that directs the user to the web page's sections. It also shows the time available to book a ticket to the movie. When the user clicks on the timetable, the user is redirected to the Booking Ticket Page.

The page, as shown in Figure 18, has the poster of the movie on the left side and other information on the right side. This information includes the title, description, number starts to evaluate depending on each user, average rating, release date, and duration of the film. The status of the film's release is also displayed on the right side. For example, the status indicates if the film has been released or is still in production. After that, scrolling down the website reveals the showtimes of the movies.

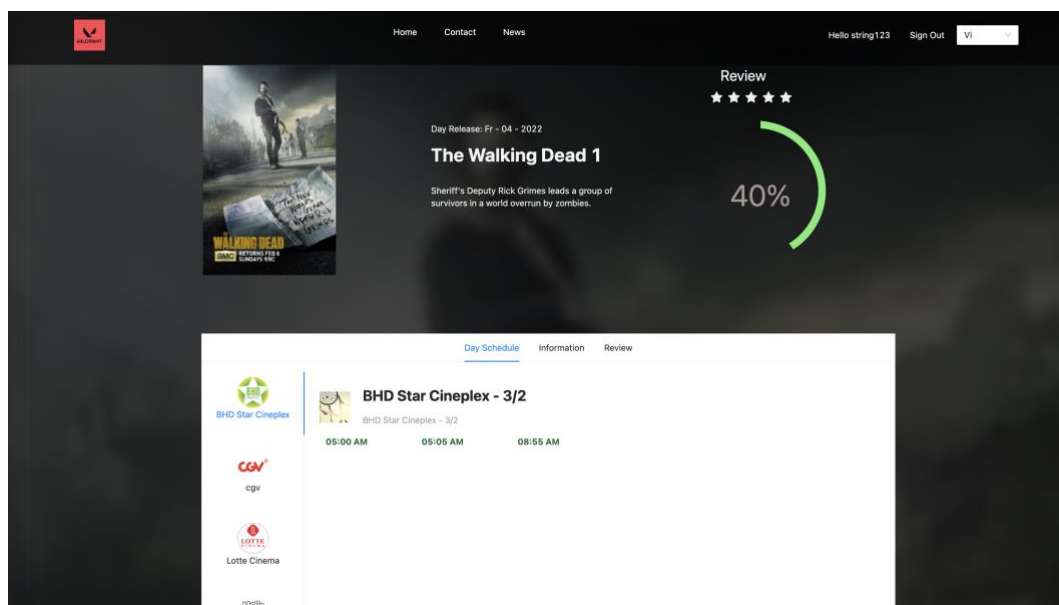
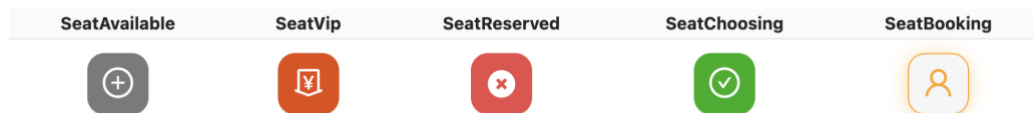



Figure 18. Detail movie Page


4.4 Booking Page


The Booking page is the page which shows all the available sheets. When a user clicks or pushes one of the movies, he/she is redirected to a page with all the necessary information on the film. The navigation bar or the side-navigation bar does not have a link to this page.

As shown in Figure 19, there are five different kinds of seats for the user to book based on the color, gray, orange, red, green and white. The first one, gray, is Seat Available. The seat available is the seat that the user can book. The second one, orange, is Seat VIP. The Seat VIP is the seat that has the higher price than the normal seat because this location provides the best view at the movie theatre. The third one, red, is Seat Reserved. The Seat reserved is the seat that has been already reserved by another person. The fourth one is Seat Choosing. Whenever user clicks on the button Seat Available, it will transform the color from gray to green and it shows that ticket has already been booked successfully. The last one is Seat Booking (after the user has booked a ticket and clicks the button Booking Ticket, the seat will turn yellow and show that the seat has been booked by the user.

**Figure 19.** Different Seats can book in theatre

01 CHOOSE SEAT & CHECKOUT 02 RESULT OF CHECKOUT 






















































 Sign Out
Hello ! string123



The Walking Dead 1

Location: L5-Vincom 3/2, 3C Đường 3/2, Q.10
Theatre: BHD Star Cineplex - 3/2

Screen

01	02	03	04	05	06	07	08	09	10	11	12	13	14	15	
17	18	19	20	21	22	23	24	25	26	27			30	31	
33	34														
49	50														
65	66													80	
81	82														
97	98														
113	114														

0 Euro

The Walking Dead 1
Location: BHD Star Cineplex - 3/2 - Rpp 10
Day Release : 17/03/2022 - 05:03

Seat 0 Euro

Email
string123

Phone

Booking Ticket

Figure 20. Booking Ticket Page.

Figure 20 shows the next page of booking proofs. On the right side, the total price and total number of reserved seats are shown. In addition, the user information is shown, such as email, phone, details of movie, day release and movie theatre location of the film. There is also a Homepage icon in the header; it will rotate to the homepage when the use clicks on that icon. The information of user logging shows on the right side of the header and the user can also login or sign out by clicking on that button.

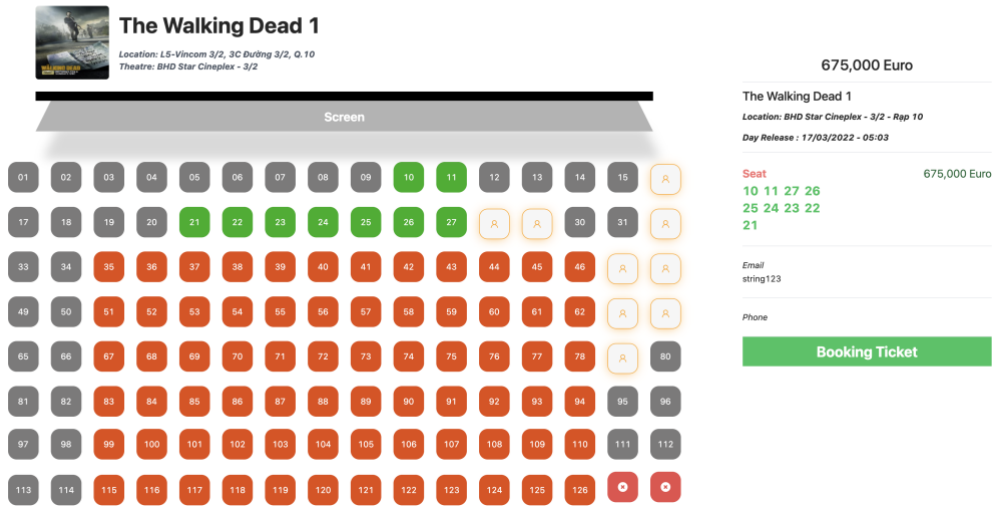


Figure 21. Seat Chosen When Booking

There are two actions on the Booking Page. The first action is when user clicks the button Booking Ticket, the navigation bar will process to the second page, the result of checkout. This navigation bar will show all the number of seats that the user has booked to the movie.

History Booking Ticket Of Customer

Checking out the Location and Time of Movie to have the funny time together



Figure 22. History of all film booking

4.5 News Page

News page(see Figure 23) shows posts and news of the movie related after two hours. On this page, the front-side of the page shows the profile picture of a celebrity. Under that, there are other details, such as the name of that post and

the day released. At the bottom of the post, it shows the quantity of comments and Likes from the user.

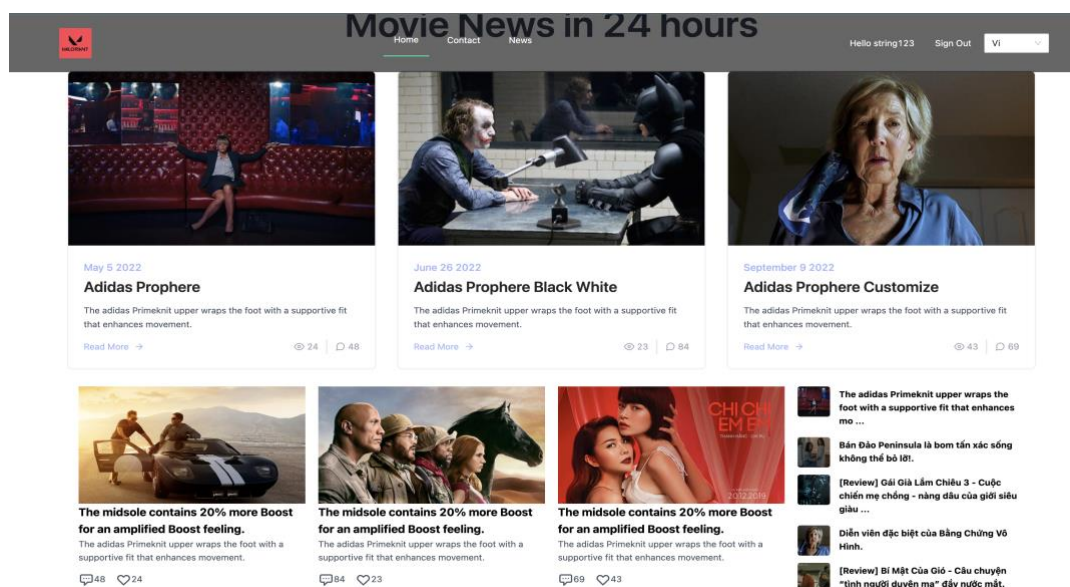


Figure 23. News of movie

4.6 Admin Page

Figure 24 shows the admin page, which allows the admin to handle product data for certain methods such as adding, updating, and removing film items via the website admin. After successfully logging in, the administrator has complete authority over the website's data. The admin has also rights to see all registered users.

At the Films page, the admin can see all the information of films related to the Home Page and he has rights to search, edit or update the specific film. In Figure 25 shows the, the information of the registered users that the admin can see. Moreover, the admin has a possibility to filter in each section by ascending to display the film from the bottom to the top or vice versa. As shown in Figure 26, the admin can also add a description of a new movie including current film, the day of release or trailer of the film.

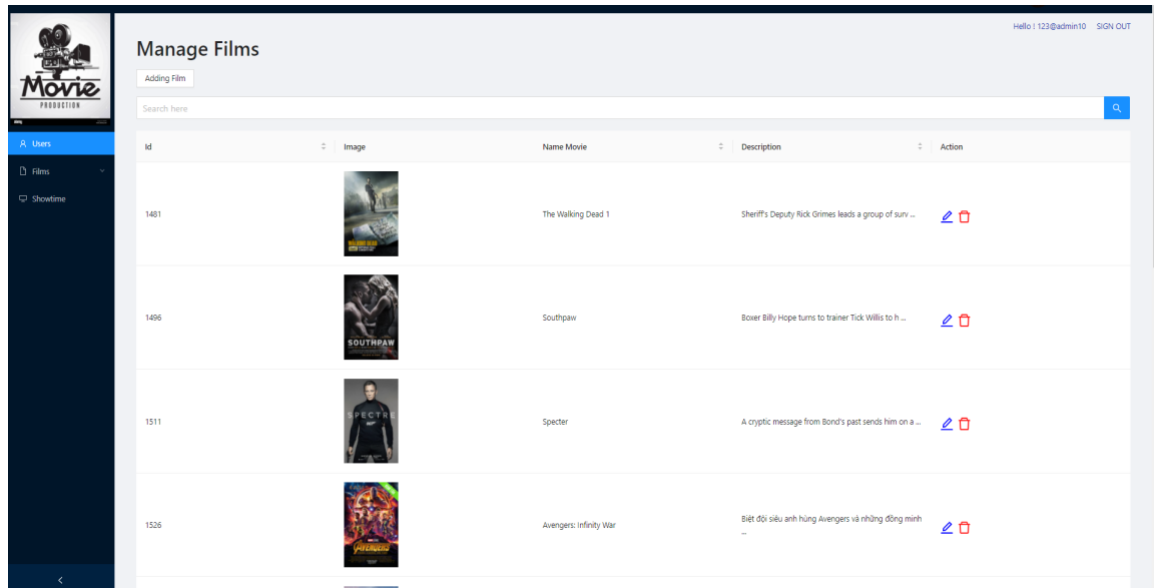


Figure 24. Admin Page

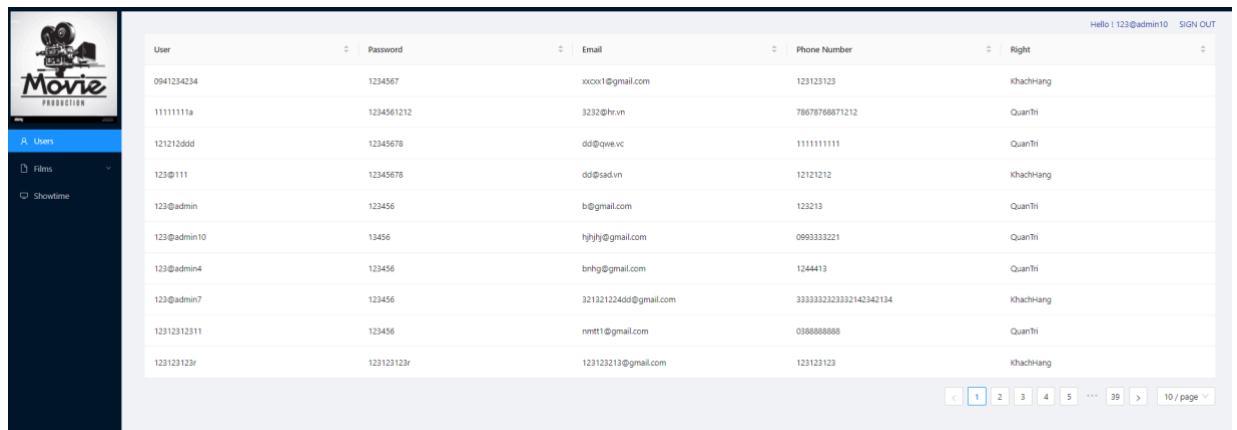


Figure 25. User Account in Admin Page.

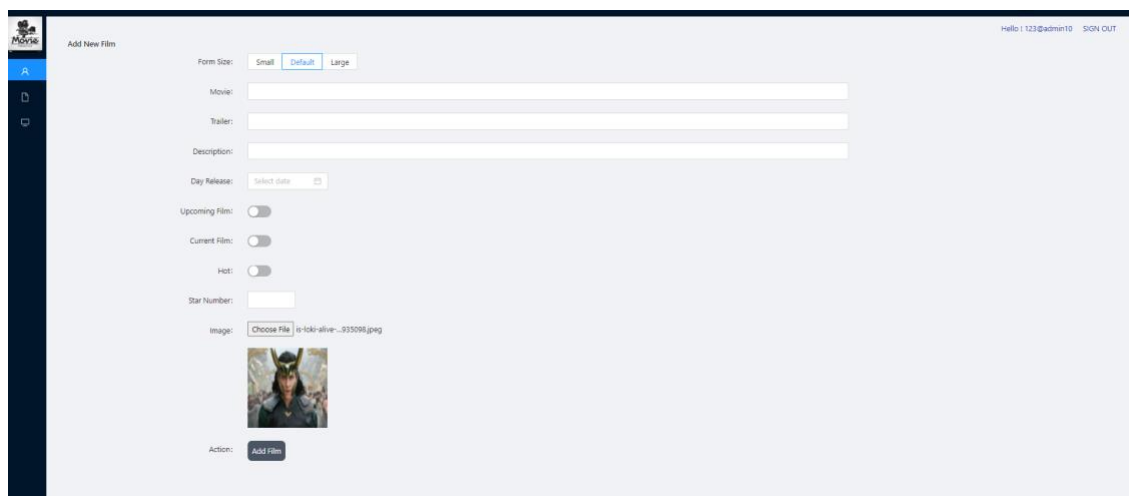


Figure 26. Adding New Film in Admin Page.

4.7 Login Page

The users have the possibility to mark their favorite movies and TV shows, and for that login pages are needed. Each page has a link to the login page. The login and sign out pages are designed to identify whether the input box is filled or not. Both the login and logout buttons on their respective pages are disabled and not clickable at first, but once the user fills out the input boxes, the button becomes enabled and clickable. The "Username" and "Password" input boxes can be seen on the login pages.

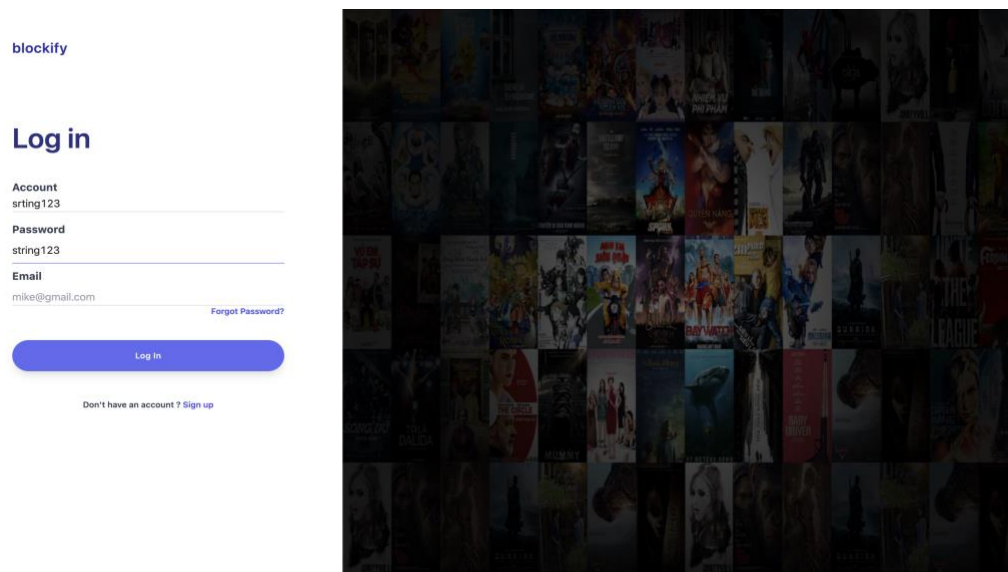


Figure 27. Login Page

After logging in, the user is redirected to the dashboard page. The page includes lists of user's favorite movies and TV shows. If there are no favorite movies or TV shows listed, the web application requires the user to login when booking a ticket. If the user is not logged in, he or she cannot access the booking page.

5. IMPLEMENTATION

5.1 Server Configuration

The development process began with the installation of the necessary dependencies for the development environment. Figure 28 shows how, The "package.json" file was created, which contains a list of all the project's dependencies and dev-dependencies. As the NPM package manager's default behavior, all installed dependency packages were stored in the "node_modules" directory. All the server and web application logic were stored in the "src" directory, which also stored the application's entry point and controller, a "index.js" file. The "public" directory contains all the asset files of the application, such as images and icons. The "build" directory, which is created automatically when the web application is built, stores the build tool's outputs.

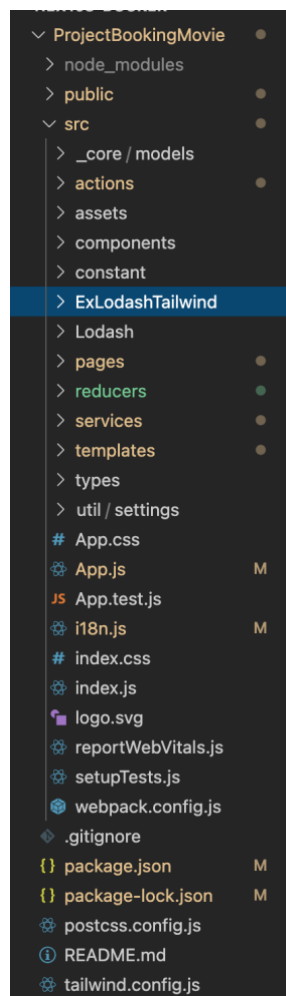


Figure 28. File structure of Project

The Home Template for the project was configured with Router . The Home Template is the place where the directory saved template of the components and is used to rotate to another page by using library history. In Code Snippet 18 and 19 illustrates, how the Home Template has props of each page where this will pass the props of each Component which shows the detail of each page. Use History and Nav link were mainly used to redirect and rotate the user to another page of the project.

```
const HomeTemplate = (props) => {
  // props path exact component
  const { Component, ...restProps } = props; // boc tach' props
  useEffect(() => {
    window.scrollTo(0,0)
  }, [])
  return (
    <Route
      {...restProps}
      render={(propsRoute) => {
        return (
          <Fragment>
            <Header {...propsRoute} />
            <Component {...propsRoute} />
            <hr />
            <Footer />
          </Fragment>
        );
      }}
    />
  );
};
export default HomeTemplate;
```

Code snippet 18. Home Template of Project

```
return (
  <Router history={history}>
    <Loading/>
    <Switch>
      <Suspense fallback={<Loading />}>
        <HomeTemplateLazy path="/" exact Component={Home} />
        <HomeTemplateLazy path="/home" exact Component={Home} />
        <HomeTemplateLazy path="/contact" exact Component={Contact} />
        <HomeTemplateLazy path="/news" exact Component={News} />
        <HomeTemplateLazy path="/detail/:id" exact Component={Detail} />
        <HomeTemplateLazy path="/profile" exact Component={Profile} />
        <Route path="/register" component={Register} />
        <UserTemplate path="/login" exact Component={LoginLazy} />
        <CheckoutTemplate path="/checkout/:id" exact Component={Checkout} />
        { /* <Route path="/login" component={LoginLazy} /> */ }
      </Suspense>
    </Switch>
  </Router>
)
```

Code snippet 19. Home Template of components

All the functions of the application work based on the three main components. These are Reducers, Store and Actions. Figure 29 shows the actions delivering information from the application to the Store, specifying what jobs will be performed using this store. The action is made up of two parts: the type that specifies the action and the value of the argument that was supplied.

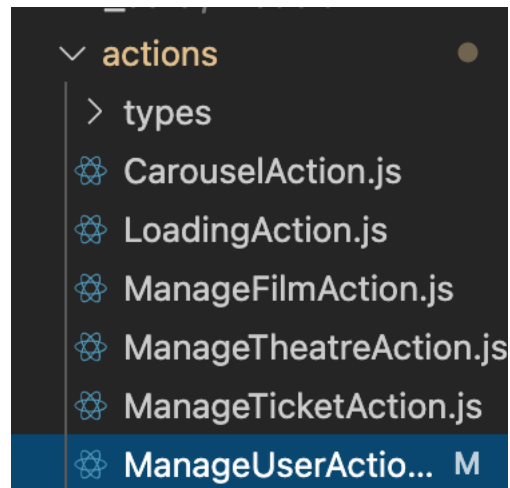


Figure 29. Action File

Figure 30 depicts the Reducer files that were implemented in the project. The action specifies what happened but does not indicate which portion of the response state was changed or how the Reducer will change it. The Reducer receives two parameters: the old state and action information.

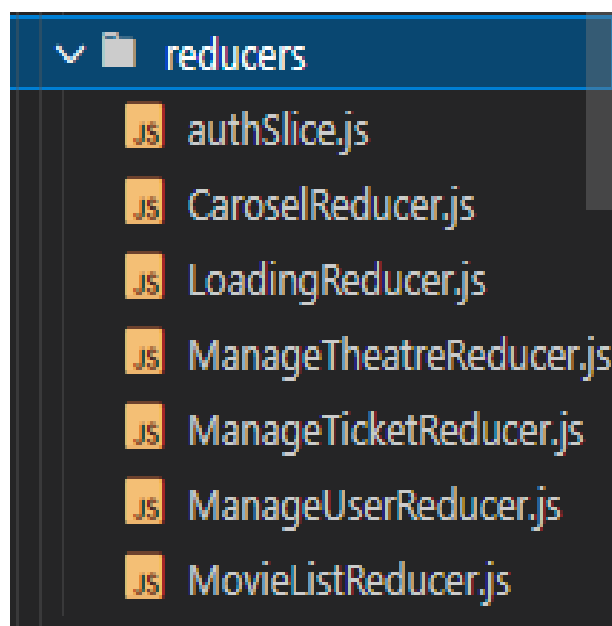


Figure 30. Reducer File

Code snippet 20 demonstrates the store as an object that contains all of states of the application, which can be accessed through `getState ()` and updated using `dispatch (action)`. The store includes a dispatcher, which is in charge of implementing actions inside the reducer; the reducer is in charge of receiving incoming actions. When an action is executed, the dispatcher is completed, and an action is sent to the reducer. The reducer then acts based on the action that was sent. Simultaneously, the value of the new state is saved in the store and that new state is returned. The dispatcher is the middleware manager, and it is typically used to call APIs and logs.

```
1 import { applyMiddleware, combineReducers, createStore } from "redux";
2 import thunk from "redux-thunk";
3 import { CarouselReducer } from "../reducers/CarouselReducer";
4 import { LoadingReducer } from "../reducers/LoadingReducer";
5 import { ManageTheatreReducer } from "../reducers/ManageTheatreReducer";
6 import { ManageTicketReducer } from "../reducers/ManageTicketReducer";
7 import { ManageUserReducer } from "../reducers/ManageUserReducer";
8 import { MovieListReducer } from "../reducers/MovieListReducer";
9 const rootReducer = combineReducers({
10   //state of entire app
11   CarouselReducer,
12   MovieListReducer,
13   ManageTheatreReducer,
14   ManageUserReducer,
15   ManageTicketReducer: ManageTicketReducer,
16   LoadingReducer: LoadingReducer,
17 });
18
19 export const store = createStore(rootReducer, applyMiddleware(thunk));
20
```

Code snippet 20. Store

5.2 Client Configuration

Methods GET, POST, PUST, DELETE are used to call backend from domain and to get API. These methods are defined in Base Service –file. This file is created to manage the link of database. Whenever there is a need to change the link of website or the URL, the developer just needs to update this component and change the link to correct one.

```

import Axios from "axios";
import { DOMAIN, TOKEN } from "../util/settings/config";
export class BaseService {
  //put json về phía backend
  put = (url, model) => {
    return Axios({
      url: `${DOMAIN}/${url}`,
      method: "PUT",
      data: model,
      headers: { Authorization: "Bearer " + localStorage.getItem(TOKEN) }, //JWT
    });
  };

  post = (url, model) => {
    return Axios({
      url: `${DOMAIN}/${url}`,
      method: "POST",
      data: model,
      headers: { Authorization: "Bearer " + localStorage.getItem(TOKEN) }, //JWT
    });
  };

  get = (url) => {
    return Axios({
      url: `${DOMAIN}/${url}`,
      method: "GET",
      headers: { Authorization: "Bearer " + localStorage.getItem(TOKEN) }, //token yêu cầu từ backend chứng minh user đã đăng nhập rồi
    });
  };

  delete = (url) => {
    return Axios({
      url: `${DOMAIN}/${url}`,
      method: "DELETE",
      headers: { Authorization: "Bearer " + localStorage.getItem(TOKEN) }, //token yêu cầu từ backend chứng minh user đã đăng nhập rồi
    });
  };
};

```

Code snippet 21. Base Service

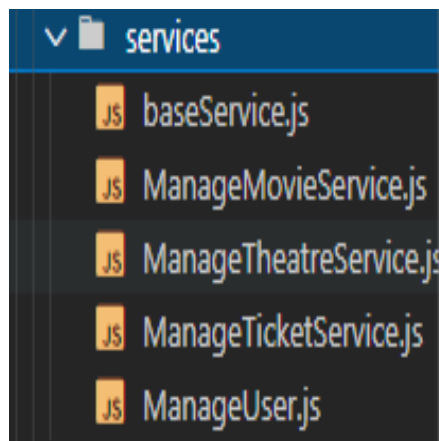


Figure 31. File service of project

Figure 31 shows the services that were implemented in the project. Services is the place to manage all the API http sending from swagger. Post, get, delete, put can be used to get the specific function sending. If there is a need to change the data of movies, they can go in this file to edit or add to the link https.

5.3 Function Implementation

a. User Login

```

export const ManageUserReducer = (state = stateDefault, action) => {
  switch (action.type) {
    case LOG_IN_ACTION: {
      const { inforLogin } = action;
      localStorage.setItem(USER_LOGIN, JSON.stringify(inforLogin));
      localStorage.setItem(TOKEN, inforLogin.accessToken);
      return { ...state, userLogin: inforLogin };
    }
  }
}

```

Code snippet 22. Method for log in

Code snippet 22 shows the code that was implemented for the process by logging into a website using a username and a password. By default, the application saves the information in the local Storage and gets the Token for each user when the user logs in. The filter makes use of the repository to load and save the security context before and after the execution of the remainder of the defined filters in the chain. In addition, Formik which is a good fit for a complete solution including validation, keeps track of visited fields and handling form submission was used to save and handle change whenever the user types in the input field of login. Therefore, whenever the user logs in, it will create the tokens for each user and save the information after login.

```

<form
  onSubmit={(event) => {
    event.preventDefault();
    formik.handleSubmit(event);
  }}

```

Code snippet 23. Using Formik to create form

Code snippet 23 shows that whenever the button Log in is clicked on, the page will prevent the browser from reloading the page by the function prevent Default and handle the Submit form.

b. Carousel

Code snippet 24 method action used to get the data from backend API Swagger and display the banner in the Homepage of web browser, after getting the data it will send to reducer to update the state of the movie list banner. Refer to Code

Snippet 25, we use `useEffect` to dispatch an action to get the API created from an action so that the data can be loaded in the web page.

```
import { manageMovieService } from "../services/ManageMovieService";
import { SET_CAROUSEL } from "../types/CarouselType";

export const getCarouselAction = () => {
  return async (dispatch) => {
    try {
      const result = await manageMovieService.getBannerList();
      dispatch({
        type: SET_CAROUSEL,
        arrImg: result.data.content,
      });
    } catch (errors) {
      console.log("errors", errors);
    }
  };
};
```

Code snippet 24. Method Carousel Banner

```
/* eslint-disable no-unused-vars */
/* eslint-disable no-undef */
import React, { useEffect } from "react";
import { Carousel } from "antd";
import { useSelector, useDispatch } from "react-redux";
import axios from "axios";
import { getCarouselAction } from "../../actions/CarouselAction";
import "../HomeCarousel.css";
export default function HomeCarousel(props) {
  const { arrImg } = useSelector((state) => state.CarouselReducer);
  console.log("arrImg", arrImg);
  const dispatch = useDispatch();
  // eslint-disable-next-line react-hooks/exhaustive-deps
  useEffect(() => {
    dispatch(getCarouselAction());
    // eslint-disable-next-line react-hooks/exhaustive-deps
  }, []);
  const contentStyle = {
    height: "900px",
    color: "#fff",
    lineHeight: "300px",
    textAlign: "center",
    background: "#364d79",
    backgroundSize: "100%",
    backgroundRepeat: "no-repeat",
    backgroundPosition: "center",
  };
};
```

Code snippet 25. UseEffect to get an API

Code snippet 26 shows the code which describes the place to get an API from action method before. After using `useEffect` to get banner of the film, the specified data which related to banner will display on the web browser. Reducer used to update the state of `arrImg` of banner. After it sends the stateDefault of `arrImg` to the store of Redux, the state will automatically update and render on browser.

```
import { SET_CAROUSEL } from "../actions/types/CarouselType";

const stateDefault = {
  arrImg: [
    {
      maBanner: 1,
      maPhim: 1282,
      hinhAnh: "http://movieapi.cyberlearn.vn/hinhanh/ban-tay-diet-quy.png",
    },
  ],
};

export const CarouselReducer = (state = stateDefault, action) => {
  switch (action.type) {
    case SET_CAROUSEL: {
      state.arrImg = action.arrImg;
      return { ...state };
    }

    default:
      return { ...state };
  }
};
```

Code snippet 26. Reducer of banner feature

c. Booking Carousel

Code snippet 27 shows, the MovieListReducer where each case shows the current state of the movie. This reducer updates the state of film based on returning the new state and re-render new state.

```
export const MovieListReducer = (state = stateDefault, action) => {
  switch (action.type) {
    case SET_MOVIE_LIST: {
      state.arrFilm = action.arrFilm;
      state.arrFilmDefault = state.arrFilm;
      return { ...state };
    }

    case SET_CURRENT_FILM: {
      state.dangChieu = !state.dangChieu;
      state.arrFilm = state.arrFilmDefault.filter(
        (movie) => movie.dangChieu === state.dangChieu
      );
      return { ...state };
    }

    case SET_UPCOMING_FILM: {
      state.sapChieu = !state.sapChieu;

      state.arrFilm = state.arrFilmDefault.filter(
        (movie) => movie.sapChieu === state.sapChieu
      );
      return { ...state };
    }
  }
};
```

Code snippet 27. Reducer of List Film in Home

The arrFilm shows all the state Default and default properties of a movie (see Code snippet 28). If he/she wants to update the function of any features, they have to set the stateDefault for that object.

```
const stateDefault = {
  arrFilm: [
    {
      maPhim: 9427,
      tenPhim: "Trạng Tí Phiêu Lưu Ký 121",
      biDanh: "trang-ti-phiêu-luu-ky-121",
      trailer: "https://youtu.be/sx1ROHCmY-4",
      hinhAnh:
        "http://movieapi.cyberlearn.vn/hinhanh/trang-ti-phiêu-luu-ky-121_gp01.png",
      moTa: "Trạng tí phiêu lưu ký là một bộ phim do người Việt sản xuất",
      maNhom: "GP01",
      ngayKhoiChieu: "2022-01-25T13:57:40.603",
      danhGia: 10,
      hot: true,
      dangChieu: true,
      sapChieu: true,
    },
  ],
  filmDetail: {},
  arrFilmDefault: [],
  dangChieu: true,
  sapChieu: true,
  filmInforAdmin: {},
};
```

Code snippet 28. State default of Film

As illustrated in Code Snippet 29, we used useSelector to get the data of movie to display and useEffect used to get an Api from swagger database. After getting data, we passed props to the component where they wanted to use. For instance, we want to use this data to display in SliderSlick and HomeMenu component, so we pass props arrFilm and arrTheatre to display the movie in Landing Page.

```
export default function Home(props) {  
  
  const { arrFilm } = useSelector((state) => state.MovieListReducer);  
  const { arrTheatre } = useSelector((state) => state.ManageTheatreReducer);  
  const dispatch = useDispatch();  
  console.log("propsHome", arrFilm);  
  
  useEffect(() => {  
    const action = getFilmAction();  
    dispatch(action); //dispatch function từ thunk  
    dispatch(getListTheatre());  
  }, []);  
  
  return (  
    <div>  
      <HomeCarousel />  
      <section className="text-gray-600 body-font ">  
        <div className="container px-5 mx-auto">  
          <SliderSlick arrFilm={arrFilm} />  
        </div>  
      </section>  
  
      <div className="mx-44 mt-12" >  
        <HomeMenu arrTheatre={arrTheatre} />  
      </div>  
    </div>  
  );  
}
```

Code snippet 29. useEffect to call api from backend

Code Snippet 30 illustrates how, after getting an Api from manage Movie Service, an action in separate film movie is dispatched, and in each reducer, movies have their own function to return to the new state. After dispatching an action to get API, the getListTheatre will re-render to get the list of movie to display on the web browser.

```

export const getFilmAction = () => {
  return async (dispatch) => {
    try {
      const result = await manageMovieService.getMovieList();
      dispatch({
        type: SET_MOVIE_LIST,
        arrFilm: result.data.content,
      });
    } catch (errors) {
      console.log("errors", errors);
    }
  };
};

export const addFilmUploadImageAction = (formData) => {
  return async (dispatch) => {
    try {
      let result = await manageMovieService.addFilmUploadImage(formData);
      alert("Success add Movie");
      console.log("result", result.data.content);
    } catch (errors) {
      console.log(errors.response?.data);
    }
  };
};

```

Code snippet 30. Action of list films

d. Advertise Landing Page

The fake Api is created in constant and it is called back to display image in component News (see Code snippet 31). We used map to limit the data obtained from Api and marked data name as an item, for the item description of movie if it is larger than 200, the text description will show the remaining text by three dots.

```

<div className="flex flex-wrap -m-4">
  {listData.map((item, index) => {
    return (
      <div className="p-4 sm:w-1/2 lg:w-1/3" key={index}>
        <div className="h-full border-2 border-gray-200 border-opacity-60 rounded-lg overflow-hidden">
          <img
            className="lg:h-72 md:h-48 w-full object-cover object-center"
            src={item.image}
            alt="blog"
          />
          <div className="p-6 hover:bg-indigo-700 hover:text-white transition duration-300 ease-in">
            <h2 className="text-base font-medium text-indigo-300 mb-1">
              {item.day}
            </h2>
            <h1 className="text-2xl font-semibold mb-3">{item.name}</h1>
            <p className="leading-relaxed mb-3">
              {item.description.length > 200 ? (
                <p>{item.description.substring(0, 200)} ...</p>
              ) : (
                <p>{item.description}</p>
              )}
            </p>
          </div>
          <div className="flex items-center flex-wrap ">
            <a
              href="https://movie.zalopay.vn/landing"
              className="text-indigo-300 inline-flex items-center md:mb-2 lg:mb-0"
            >

```

Code snippet 31. News of Landing Page

e. Detail Page

In the detail page, web browser displays the information and time to book the ticket of each film. In addition, the user can also review and evaluate the film based on the number of start in that film.

```
import { useSelector, useDispatch } from "react-redux";
import { getDetailMovie } from "../../actions/ManageTheatreAction";
import moment from "moment";
import { Rate } from "antd";
import "../Detail.css";
import { NavLink } from "react-router-dom";
const { TabPane } = Tabs;
export default function Detail(props) {
  const filmDetail = useSelector((state) => state.MovieListReducer.filmDetail);
  console.log({ filmDetail });
  const dispatch = useDispatch();

  useEffect(() => {
    window.scrollTo(0, 0);
  })
  useEffect(() => {
    let { id } = props.match.params;
    console.log("id", id);
    dispatch(getDetailMovie(id));
    // eslint-disable-next-line react-hooks/exhaustive-deps
  }, []);
}
```

Code snippet 32. Detail page Method

We use useSelector to get data in filmDetail and useEffect to dispatch an action to Reducer to get the detail information of movie (see Code snippet 32). Because each film, has a unique id like the image shown in Figure32, it is 1481 so we could destructure the id from match params to get the id of that film.

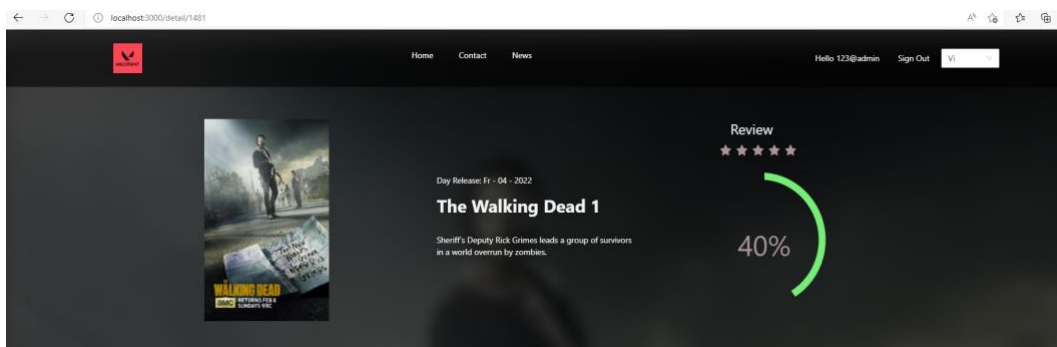


Figure 32. Unique Id of detail page.

f. Booking Page

In code snippet 33, Here is the place where user can display 5 different kinds of seat which are seatVip, seatReserved, seatChoosing, seat Booking and

seatAvailable. Each seat has their own function like we describe before. We use `useEffect` to get the id from Appjs to create async function and send the information by dispatching an action to reducer. After getting the data by dispatching an action, we can map to get the data of different kind of seats to display seats on the web browser.

```
function Checkout(props) {
  const { userLogin } = useSelector((state) => state.ManageUserReducer);
  const { detailTicket, listSeatChosen } = useSelector(
    (state) => state.ManageTicketReducer
  );
  const dispatch = useDispatch();
  console.log("listSeatChoosing", listSeatChosen);
  useEffect(() => {
    window.scrollTo(0, 0);
  })
  // gọi Api
  useEffect(() => {
    // Lay id từ Appjs gọi hàm tạo ra 1 async function
    const action = getDetailTicketAction(props.match.params.id);
    // dispatch function di
    dispatch(action);
    // eslint-disable-next-line react-hooks/exhaustive-deps
  }, []);

  console.log({ detailTicket });
  const { thôngTinPhim, danhSachGhe } = detailTicket;
  const renderSeats = () => {
    return danhSachGhe?.map((seat, index) => {
      let classSeatVip = seat?.loaiGhe === "Vip" ? "seatVip" : "";
      let classSeatReserved = seat?.daDat === true ? "seatReserved" : "";
      let classSeatChosing = "";
    });
  };
}
```

Code snippet 33. Booking Page Method

In Code snippet 34, we use module `classname` to style for each seat like the image below. Here are 5 styling of classname based on different seats.

```
<button
  disabled={seat?.daDat}
  key={index}
  className={`seat ${classSeatVip} ${classSeatReserved} ${classSeatChosing} ${classSeatBookingYourSelf}`}
```

Code snippet 34. Style for each Seat

In Code snippet 35 we use function `reduce` to count the total money of the number ticket user has booked.

```
<div className="col-span-3 my-20">
  <h3 className="text-center text-2xl">
    {listSeatChosen
      .reduce((total, seat, index) => {
        return (total += seat.giaVe);
      }, 0)
      .toLocaleString()}" }
    Euro
  </h3>
</div>
```

Code snippet 35. Method for counting the total money of booking tickets

After finishing booking, the page will process to second page where it shows all the film history of that account booking. In code snippet 36, This method will return all the seats user has booking. To be more specified, we get information from userInfor based on different seat they get where its ticket is marked by user so that the method renderTicketItem will display the image of that movie which is ticket.tenPhim and the name of that movie which is ticket.tenPhim.

```
const renderTicketItem = function () {
  return userInfor.thongTinDatVe?.map((ticket, index) => {
    const seats = _.first(ticket.danhSachGhe);
    return (
      <div className="p-2 lg:w-1/3 md:w-1/2 w-full" key={index}>
        <div className="h-full flex items-center border-gray-200 border p-4 rounded-lg">
          <img
            alt={ticket.tenPhim}
            className="w-24 h-24 bg-gray-100 flex-shrink-0 rounded-full mr-4 object-cover"
            src={ticket.hinhAnh}
          />
          <div className="flex-grow">
            <h2 className="text-gray-900 title-font font-bold">
              {ticket.tenPhim}
            </h2>
            <p className="text-gray-500 text-sm font-bold">
              Day Releases: {moment(ticket.ngayDat).format("hh:mm A")} - Hour
              Starts: {moment(ticket.ngayDat).format("DD-MM-YYYY")}
            </p>
            <p className="text-sm font-bold">
              Location : {seats.tenHeThongRap}
            </p>
            <p className="text-sm font-bold">
              Cinema: {seats.tenCumRap} -{" "}
            </p>
          </div>
        </div>
      </div>
    )
  })
}
```

Code snippet 36. Method renderTicketItem

g. Admin Page

In this section, user have two kind of rights which are admin and basic user. Admin have rights to manage users and movies with those function like Adding, Delete, Edit movie. Basic user, user have has rights to book the ticket and show all the history of that booked movie.

In code snippet 37, useSelector is used to get the data from Reducer and display all the Movie in the admin Page.


```

export default function Films() {
  const { arrFilmDefault } = useSelector((state) => state.MovieListReducer);
  const dispatch = useDispatch();
  console.log("arrFilmDefault", arrFilmDefault);
  useEffect(() => {
    dispatch(getFilmAction());
    // eslint-disable-next-line react-hooks/exhaustive-deps
  }, []);
}

```

Code snippet 37. Method for displaying movie in Admin Section

In Code snippet 38, a `addFilmUploadImageAction`-function is shown. Function is used to show the movie after the button Editing is clicked. With the function admin can change the details of the selected film such as: Movie, trailer, description, day release and the switch button Upcoming or Current Film.

```

export const addFilmUploadImageAction = (formData) => {
  return async (dispatch) => {
    try {
      let result = await manageMovieService.addFilmUploadImage(formData);
      alert("Success add Movie");
      console.log("result", result.data.content);
    } catch (errors) {
      console.log(errors.response?.data);
    }
  };
};

```

Code snippet 38. Method for Editing film

In code snippet 39, this method saves the formik `formData` to handle change when we submit and after that it will call api to send the data to backend to process the action.

```

onSubmit: (values) => {
  console.log("values", values);
  // Create Object fomrdata => Đưa giá trị từ formik vào formData
  values.maNhom = GROUPID;
  let formData = new FormData();
  for (let key in values) {
    if (key !== "hinhAnh") {
      formData.append(key, values[key]);
    } else {
      formData.append("File", values.hinhAnh, values.hinhAnh.name);
    }
  }
  // call api to send formData to backend handle
  dispatch(addFilmUploadImageAction(formData));
  // console.log("formData", formData.get("File"));
},
});

```

Code snippet 39. Method for AddingMovie

h. Multilingual language

In Code snippet 40, The i18next library is used to make the web application multilingual. Before being initialized, the i18next instance requires the i18next settings. I18nextProvider would take the i18next instance as a property after it was created and provide it with the web application environment. The i18next instance is applied differently in the client and server side of the web application.

/14/

```

import Backend from "i18next-http-backend";
import LanguageDetector from "i18next-browser-languagedetector";
// don't want to use this?
// have a look at the Quick start guide
// for passing in lng and translations on init

i18n
  // load translation using http -> see /public/locales (i.e. https://github.com/i18next/react-i18next/tree/master/example/react/public/locales)
  // learn more: https://github.com/i18next/i18next-http-backend
  // want your translations to be loaded from a professional CDN? => https://github.com/locize/react-tutorial#step-2---use-the-locize-cdn
  .use(Backend)
  // detect user language
  // learn more: https://github.com/i18next/i18next-browser-languagedetector
  .use(LanguageDetector)
  // pass the i18n instance to react-i18next.
  .use(initReactI18next)
  // init i18next
  // for all options read: https://www.i18next.com/overview/configuration-options
  .init({
    fallbackLng: "eng",
    debug: true,
    whitelist: ["chi", "eng", "vi"], //Liệt kê các ngôn ngữ
    interpolation: {
      escapeValue: false, // not needed for react as it escapes by default
    },
  });

export default i18n;

```

Code snippet 40. I18N configuration

i18n would take the i18next instance as a property when it was built and supply the web application context to it. In the client and server sides of the web application, the i18next instance is used in a different way. /14/

The JSON format is used to hold key-value pairs in translation files. Languages will use similar translation files and keys, with the difference that the value of the keys will vary depending on the language. In Figure 33, 3 different languages were added in the project, English, Finnish, and Vietnamese. With this, the webpage can be used with these three languages. /14/

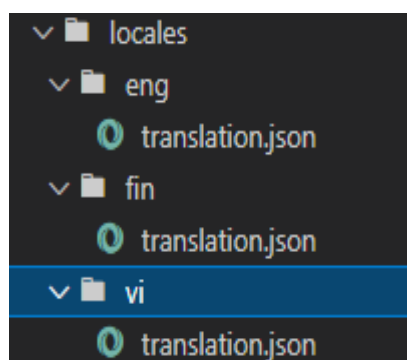


Figure 33. Translation Language in System

i. Window Scroll Effect

In figure 34, hook `useEffect` uses to scroll of the window to a certain location in the page. For example: hook can be used to go to the top of the page after redirect to another page. /15/

```
useEffect(() => {  
  window.scrollTo(0, 0);  
})
```

Figure 34. Hook used to come to top of the page

6. TESTING AND DEPLOYMENT

Table 1 describes an essential aspect of the project in terms of functional verification. This project is currently being updated and will include functionalities and procedures that are appropriate for corporate purposes. This project has been deployed and is running on Surge and can be viewed via a web browser for testing purposes. For the sake of simplicity, the test sample from Table 1 was used.

Table 1. Testing Table

Case	Test	Steps	Response	Status
1.	View current and upcoming film	Click the button to choose category	The list of current and upcoming film	Pass
2.	Checking for logging account	Enter username and password.	The form will process into the website	Pass
3.	Checking booking ticket	Click on the seat user wants to book	the system will change to reserved seat	Pass
4.	Checking history of film booking	Click on the Result of Checkout	The system will show all the history of already booked movies	Pass
5.	Checkout with Payment process	Click on the button Booking Ticket	The application will redirect to the second page which is Result of checkout	Pass
6.	Checking the theatre shown Movie	Click on the Theatre button	The app will show the movie that has available time in each theatre	Pass
7.	Slider of Phone Advertisement	Click on the phone advertisement	The advertise will transfer to another page of the phone advertisement	Pass
8.	Changing language header	Click on the drop-down navigation bar and choose language to change	The web application will translate to the appropriate language user has chosen	Pass
9.	Logging in Admin page with admin account	Click on the http website and call /admin after the link	If don't login with right account, the page will show the alert message and can not rotate to the admin page	Pass

10.	Search Film	Click on the search button at Admin Page	The system will return the right film user wants to look for	Pass
11.	Edit Film	Click on the Edit Icon	The system will return all the film information of that film that user wants to replace	Pass
12.	Delete Film	Click on the Delete Icon	The system will delete the movie and it doesn't show anymore in the Carousel	Pass
13.	Register with existing account	Enter information from existing user in register form	The response will send the modal message that confirms succeed	Failed
14.	Show user	Click on the button user at navigation bar	The admin page will show all the information of the users	Pass
15.	Checking for logging wrong information	Enter username and password	The user will see the pop up modal which shows user type the wrong username and password	Failed
16.	Comment Page	Click on the comment section for a movie	The user will see all comment from that film and give the review to it	Failed
17.	Checking password	Click on the log in navigation bar	The user will see hidden password of the application	Failed

8. CONCLUSIONS

This project was completed effectively, and the results were satisfactory. This project was created to suit the needs of work assignment. It was written in React, and the database was obtained from available api in Swagger Server. This webpage will allow the user to purchase a ticket to the movie online. To complete the ticketing process, the relationship between the admin, user, and customer must be in an excellent working order.

The project was implemented with the goal of making data, theater specifics, and relevant input as accessible as feasible to the user. The user is given a website that can be utilized to book cinema tickets online. React was utilized as the technology to make this web application. Improved performance, scalability, built-in security, and simplicity are just a few of the benefits of React.

The Tailwind component theme is quite popular and has a lot of practical applications. However, because lacking time and experience, the Booking Ticket Website only progresses to the point of meeting the topic's criteria; the processing speed is still lacking. Further study will focus on comprehending the system's management mechanism as well as improving the performance of the application, broadening the scope of project.

Although there are still some flaws in the project, such as Login Page not hiding the password when typing in the blank box, the project is primarily focused on front end with API provided. These bugs will need to be fixed and improved the SEO to create the better platform website for user. In the future, our application for booking movie ticket will include additional features such as registration via Facebook or Gmail, and payment via master debit card. Making the user interface more user friendly in small screen devices is another feature to be developed. We want to add a recommendation system in our project, which will make the application easier for users to find or interact with the developed application.

REFERENCES

/1/ React definition. Accessed 28.2.2022

<https://webmobtechnologies.medium.com/why-use-react-js-a-complete-guide-4863de674d05>

/2/ Advantages and Disadvantages of using ReactJs. Accessed 1.3.2022

<https://www.edureka.co/blog/interview-questions/react-interview-questions/>

/3/ JSX definition. Accessed 3.3.2022

https://www.w3schools.com/react/react_jsx.asp#

/4/ Reactjs Props and state. Accessed 2.3.2022

<https://www.interviewbit.com/react-interview-questions/>

/5/ Reactjs Router. Accessed 3.3.2022

<https://reactrouter.com/>

/6/ Javascript ES6. Accessed 4.3.2022

<https://www.programiz.com/javascript/arrow-function#>

/7/ Redux definition. Accessed 5.3.2022

<https://blog.logrocket.com/why-use-redux-reasons-with-clear-examples-d21bffd5835/#>

/8/ Redux flow. Accessed 6.3.2022

<https://dev.to/oahehc/redux-data-flow-and-react-component-life-cycle-11n>

/9/ Bootstrap 4. Accessed 7.3.2022

[Bootstrap 4 | Introduction - GeeksforGeeks](https://www.geeksforgeeks.org/bootstrap-4-introduction/)

/10/ TailwindCSS, Ant-Design. Accessed 8.3.2022

<https://medium.com/wesionary-team/how-to-implement-ant-design-with-react-7d21b6c> , <https://tailwindcss.com/docs/guides/create-react-app>

/11/ Surge. Accessed 9.3.2022

<https://surge.sh/>

/12/ Swagger, NodeJS definition. Accessed 10.3.2022

<https://www.section.io/engineering-education/documenting-node-js-rest-api-using-swagger/>

<https://en.wikipedia.org/wiki/Node.js>

/13/ Create-react-app. Accessed 11.3.2022

<https://reactjs.org/docs/create-a-new-react-app.html>

/14/ Multilingual Language. Accessed 12.3.2022

<https://medium.com/how-to-react/setup-multilingual-in-react-js-using-i18n-module-33b1bfbb57cd>

/15/ Window scrolls Effect. Accessed 13.3.2022

<https://alligator.io/js/smooth-scrolling/>

/16/ Surge deployment. Accessed 14.3.2022

<https://surge.sh/>

/17/ React-Slick. Accessed 15.3.2022

<https://react-slick.neostack.com/>

