

BOOKSTORE USING REACT JS

SUMMER INTERNSHIP PROJECT REPORT

Submitted by

VIJAY PREMAJIBHAI ANIYALIYA

200200107072

In partial fulfilment for the award of the degree of

BACHELOR OF ENGINEERING

in

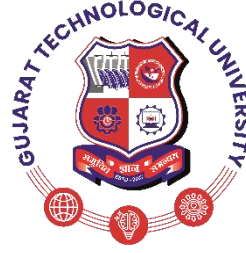
Computer Engineering

Government Engineering College, Rajkot



Gujarat Technological University, Ahmedabad

July, 2023



Government Engineering College, Rajkot

Mavadi-Kankot Road, Near Hanuman Mandir, Rajkot

CERTIFICATE

This is to certify that the project report submitted along with the project entitled **Bookstore using React JS** has been carried out by **Vijay Premajibhai Aniyaliya (200200107072)** under my guidance in partial fulfilment for the degree of Bachelor of Engineering in Computer Engineering Branch, 7th Semester of Gujarat Technological University, Ahmedabad during the academic year 2023-24.

Prof. Sanjay Bhanderi

Internal Guide

Prof. Sanjay Shah

Head of Department

Summer Internship Certificate

Date: 18/08/2023


This is to certify that Aniyaliya Vijay Premajibhai undergone summer internship from 27th July 2023 to 10th August 2023. Details of the project is as under

Name of Project: Book Store Application

Technology: ReactJS

We wish him/her grand success for the future.

Authorized Signatory



TatvaSoft, Ahmadabad



Government Engineering College, Rajkot

Mavadi-Kankot Road, Near Hanuman Mandir, Rajkot

DECLARATION

I hereby declare that the Internship report submitted along with the Internship entitled **Bookstore using React JS** submitted in partial fulfilment for the degree of Bachelor of Engineering in **Computer Engineering** to Gujarat Technological University, Ahmedabad, is a bonafide record of original project work carried out by me at **TatvaSoft.** under the supervision of **Mr. Keval Dhol** and that no part of this report has been directly copied from any student reports or taken from any other source, without providing due reference.

Name of the Student

Sign of Student

Vijay Premajibhai Aniyaliya

Acknowledgement

I have taken many efforts in this project. I faced lots of problems and tried my best to solve it. However, it would not have been possible without the kind support and help of many individuals and organizations. I would like to extend my sincere thanks to all of them. I am highly indebted to Prof. Sanjay Bhandari for their guidance and constant supervision as well as for providing necessary information regarding the Summer Internship Project Titled “BookStore using Reactjs”. I would like to express my gratitude towards our Head of the Department and other staff members of Computer Engineering Department. I even thank and appreciate to my colleague in developing the project and people who have willingly helped me out with their abilities.

Abstract

This report outlines my summer internship experience in the field of computer engineering, conducted as part of the curriculum at GTU. I reflect on my immersive summer internship experience where I had the incredible opportunity to create a fully functional BookStore using React JS. The internship was conducted remotely in collaboration with **TatvaSoft**, a reputable IT company known for its expertise in software solutions.

Over the course of my internship, I immersed myself in web development, focusing on React, and associated technologies. I covered a wide range of topics, including setting up the environment, creating React components, using Material-UI for design, implementing forms with Formik, making API calls with Axios, managing state with hooks like useState and useContext, and deploying applications on AWS. This abstract provides a concise overview of my journey, highlighting the key concepts and skills I gained during this valuable learning experience.

Table of Figures

Sr No.	Figure Name	Page No.
2.1.1	Creating react project	4
2.1.2	Project files	5
2.2.1	React Router DOM	8
2.2.2	Styling with CSS in React	8
2.3.1	Material UI Buttons	9
2.3.2	Material UI TextFields	10
2.5.1	Implementing a Counter using useState	12
3.1.1	Registration page	14
3.3.1	Login page	16
3.4.1	Header	18
3.4.2	Global book search	18
3.6.1	Book listing	20
3.6.2	Sorting options	21
3.7.1	Book page	22
3.8.1	User list page	23
3.9.1	User profile page	24
3.9.2	Category list page	25
3.10.1	Cart page	26
3.11.1	AWS Deployment Pipeline	27

Abbreviations

Abbreviation	Full Form
NPM	Node Package Manager
HTML	Hypertext Markup Language
CSS	Cascading Style Sheets
DOM	Document Object Model
ES6	ECMAScript 6
URL	Uniform Resource Locator
JSX	JavaScript XML
API	Application Programming Interface
HTTP	Hypertext Transfer Protocol
JWT	JSON Web Tokens
JSON	JavaScript Object Notation
AWS	Amazon Web Services

Table of Contents

Declaration	i
Acknowledgement	ii
Abstract	iii
List of Figures	iv
List of Abbreviations	v
Table of Contents	vi
Chapter 1: Introduction	1
1.1 Internship Overview	1
1.2 Host Company: TatvaSoft	1
1.3 Project Scope “Building a BookStore project”	2
Chapter 2: Learning React JS	3
2.1 intro to react	3
2.2 Introduction to Component and Basics UIElements	5
2.3 Introduction to Material UI and Basic UI Elements	8
2.4 Concept of useState	9
Chapter 3: Implementation of project	11
3.1 Registration Page UI	11
3.2 Registration Page API integration	13
3.3 Login Page UI & API integration	14
3.4 Building Header, Footer	16
3.5 AuthContext	18

3.6 Building the Book Listing Page	19
3.7 Product CRUD	21
3.8 User List Page	23
3.9 User Profile and Category List Page	24
3.10 Cart Page	27
3.11 AWS for Deployment	29
Chapter 4: Conclusion and Future Work	31
Chapter 5: References	32

Chapter 1: Introduction

This report outlines my summer internship experience in the field of computer engineering, conducted as part of the curriculum at GTU. For my internship project, I chose to work on the development of an online Book Store application using React JS. The internship was conducted remotely in collaboration with TatvaSoft, a reputable IT company known for its expertise in software solutions.

1.1 Internship Overview:

Throughout the internship, I had the opportunity to delve into the world of web development and gain practical experience in creating a dynamic and user-friendly application. The chosen topic of developing a Book Store Project aligned well with my interests and allowed me to explore React JS, a popular JavaScript library for building interactive user interfaces.

1.2 Host Company: TatvaSoft

TatvaSoft is a CMMi Level 3 & Microsoft Solutions Partner Software Development Company offering software development services on diverse technology platforms like Microsoft, Angular, React, NodeJS, Java, PHP, SharePoint, Open Source, BI, & Mobile.

With rich and varied experience of 18 years in software development and stringent quality standards, the company offers utmost qualitative, on-time and cost-effective software solutions. The company serves clientele across the industries and globe with offices in US, Canada, UK, Australia, and Development Center in India with a workforce of 810 IT professionals.

Company has successfully completed 1800+ projects with 700+ SMEs and Fortune 500 companies.

My Experience @TatvaSoft:

As per the curriculum of GTU, I got a chance to do an Internship Training at TatvaSoft. Let me Introduce TatvaSoft. in my words, It provides a very Good Employee friendly environment, Flexible working hours, and Good Working Support. The Goal of the company is for each employee of the company to flexibly contribute to the company and solve its challenges as their challenge.

During the Internship I learned lots of new things, If I tell, then more than 90% of things that I learned are new to me, they provide me with various resources so that I can learn things in a better way, and they provided me with a guide who guides me how to achieve the particular goal at each day and how to work in big and giant companies. I learnt how Projects work in Industries and how we have to change our mindset from a college mindset to an Industries mindset.

The Mode of Internship was online. One Guide provided to us to help us during the Internship. Each day they train us on some concepts and then check previous concept outcomes on a daily basis. So that they can teach further concepts. They taught us React js during the Internship, They Started from scratch and moved further to Ecommerce project . We will discuss it more in the next chapter.

1.2 Project Scope: Building a Book Store Project

The objective of my internship project was to design and develop a web-based Book Store application using React JS. The application aimed to simulate an online platform where users could browse, search, and purchase books. Key features included a userfriendly interface, real-time search functionality, categorization of books, and a shopping cart system.

Chapter 2: Learning React Js

During the initial phase of my internship, I focused on learning the fundamentals of React.js and grasping the concepts of Material-UI. This encompassed various aspects, ranging from installing npm (Node Package Manager) to crafting components within the React framework. Additionally, I delved into the application of Material-UI to enhance the visual aesthetics and user experience of websites.

In this learning phase, I gained proficiency in essential tasks such as setting up npm for project dependencies and managing packages. I also acquired the skills to create and structure components in React, enabling the development of modular and reusable sections of the application. Through the utilization of Material-UI, I explored methods to elevate the overall appearance and usability of websites, effectively utilizing pre-designed components and styles to achieve a polished and appealing look.

Overall, this initial learning period equipped me with a solid foundation in React.js and a practical understanding of integrating Material-UI to optimize the visual and interactive aspects of web interfaces.

2.1 Intro to react

On my first day of the internship, I began by downloading Node.js, an essential runtime environment that allows us to execute JavaScript code outside of a browser. This step was crucial as Node.js provides the tools necessary for various development tasks, including managing dependencies and running servers.

One of the key tools that comes bundled with Node.js is NPM, which stands for Node Package Manager. NPM is a command-line tool that facilitates the installation, management, and sharing of packages (libraries and modules) written in JavaScript. These packages can be integrated into projects to add functionalities, libraries, or tools,

streamlining the development process. NPM also allows us to manage project-specific dependencies efficiently.

After setting up Node.js and NPM, I proceeded to create a new React.js application using the `create-react-app` command. This command is a convenient way to generate a boilerplate React project with a preconfigured development environment. It includes essential files, configurations, and a basic folder structure, enabling a smooth start to the development process.

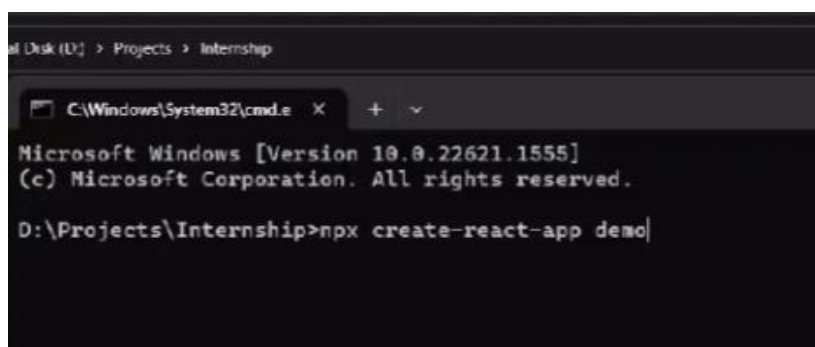


Fig. 2.1.1

Once the project was created, I learned how to run the React application. By using the `npm start` command, I launched a development server that enabled me to preview and interact with the application in a web browser. This server also facilitates live reloading, which means any changes I make to the code are immediately reflected in the browser without the need for manual refreshing.

In the process of exploring the project's files and structure, I gained a basic understanding of how a React application is organized. The project's root directory contained important files such as `package.json`, which stores project information and dependencies, and `src` directory, which houses the main source code of the application. Inside the `src` directory, I encountered components, styles, and other assets that collectively build the user interface

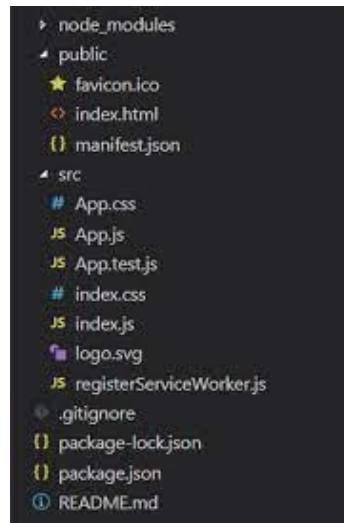


Fig. 2.1.2

In summary, my first day of the internship involved setting up the development environment by downloading Node.js, introducing NPM for managing packages, creating a React project using `create-react-app`, running the project's development server, and familiarizing myself with the initial files and structure of a React application. This foundation served as a launchpad for my journey into React.js development during the internship.

2.2 Introduction to Component and Basics UI Elements

React JS has revolutionized web development by offering a powerful and efficient way to create interactive user interfaces. As a widely used JavaScript library, react utilizes a component-based architecture to build dynamic and reusable UI components. This report delves into the core concepts of React JS, including JSX syntax, components, array methods, and the basics of React Router for navigation

In React, JSX (JavaScript XML) is a syntax extension that allows you to write HTML-like code within JavaScript. This blend of JavaScript and HTML enhances the readability and maintainability of code. However, one notable caveat is that the keyword "class" is

reserved in JavaScript. As a result, react uses the "className" attribute instead of "class" to define CSS classes for elements.

Additionally, react facilitates the seamless integration of images within components. By importing images and referencing them in the JSX code, developers can easily include visuals within their applications. For instance:

```
import logo from './logo.svg';  
  
<img src={logo} className="App-logo" alt="logo" />
```

Components in React:

Central to React's design is its component-based architecture. A component is a modular building block that encapsulates a specific piece of UI logic and rendering.

Components can be functions that return JSX elements. They can be reused throughout the application, which enhances modularity and code reusability.

Components can be utilized within other elements, including self-closing tags like `<User />`. Creating components involves conventional function syntax or using arrow functions, allowing for succinct and expressive code.

To organize and reuse components effectively, react supports exporting and importing mechanisms. Components can be exported using `export` or `export default`. Importing components is achieved through the `import` statement.

While exporting components is straightforward, default exports come with limitations when attempting to export multiple variables. This distinction prompts developers to choose the most suitable export method based on their project's structure.

Array Methods in React:

React applications often deal with data manipulation. Understanding array methods like `filter` and `map` is crucial for effective data handling. For instance, given an array `array = [10, 20, 30, 40, 50]`


```
array.filter((item) => item !== 30)  returns [10, 20, 40, 50].  
array.map((item) => item !== 30)    returns [true, true, false, true, true].  
array.map((item) => item * 5)       returns [50, 100, 150, 200, 250].
```

Props: Parsing Data Between Components

Props (short for "properties") are a fundamental concept in React for passing data from parent to child components. Props are read-only, similar to function arguments. They enable components to be dynamic by accepting external data

Demonstrating this concept, we can pass the username to the User component as follows:

```
// In parent component  
<User name={userName} />  
  
// In User.jsx  
export const User = ({ name }) => { return <h1>This is user component {name}</h1>;  
}
```

Basics of React Router DOM:

React Router is crucial for enabling navigation and routing in React applications. By implementing the BrowserRouter and defining routes using Routes and Route components, developers can create dynamic single-page applications. Navigation links are simplified using NavLink.

A practical example:

```
<BrowserRouter>
  <Routes>
    <Route path="/" element={<Home />} />
  </Routes>
</BrowserRouter>

// In Home component
export const Home = () => {
  return (
    <h1>
      Home component
      <NavLink to="/user">User</NavLink>
    </h1>
  );
}
```

Fig. 2.2.1

Styling with CSS

Styling in React can be achieved using CSS. The `className` attribute is used instead of `class` to avoid conflicts with JavaScript's `class` keyword

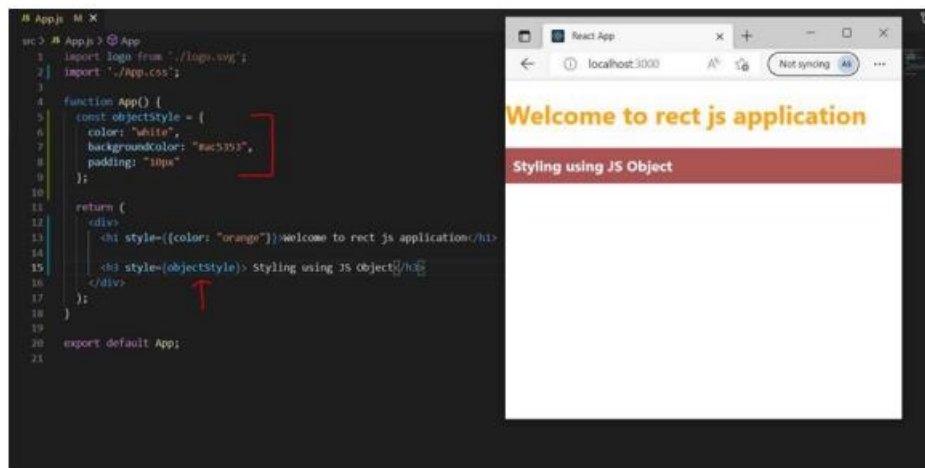


Fig. 2.2.2

2.3 Introduction to Material UI and Basic UI Elements

Material UI is an open-source React component library that implements Google's Material Design. It provides prebuilt, customizable components that adhere to Material Design principles.

Material UI can be installed using NPM: `npm install @material-ui/core`. This package includes a wide range of UI components and styles.

Basic Material UI components, such as buttons and text fields, can be easily integrated into React applications.

Material UI allows customization of themes, including color palettes, spacing, typography, and shadows. This ensures a consistent and visually appealing design.

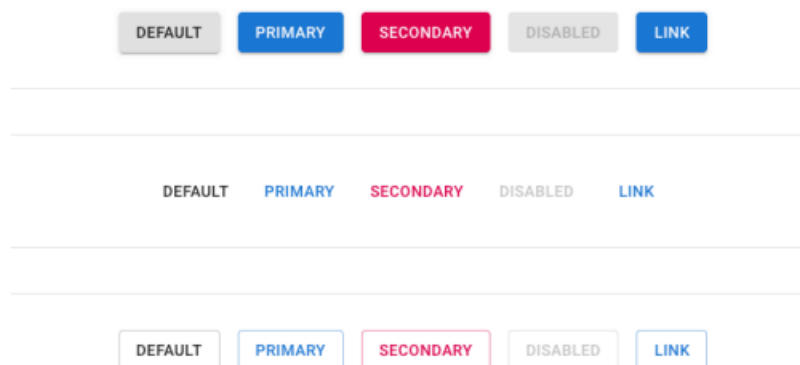


Fig. 2.3.1

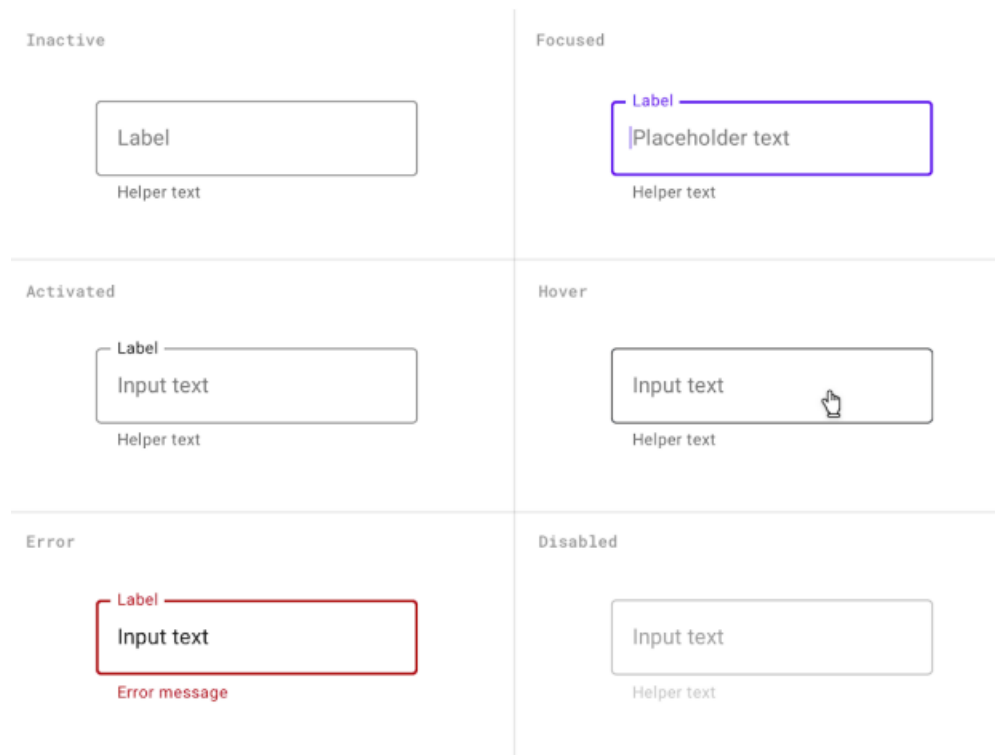


Fig. 2.3.2

2.4 Concept of useState

On my 15 day of the internship, I continued my exploration of Material-UI version 4, delving deeper into its capabilities and features. I covered topics such as Mui Theme and theme customization, as well as implementing more advanced Material-UI components while incorporating the basics of the `useState` hook.

Mui Theme and Theme Customization:

I learned about the concept of the Mui Theme in Material-UI, which allows for consistent theming across the entire application. The Mui Theme provides a central place to customize colors, typography, spacing, and other design-related attributes. I grasped how to create a customized theme to match the desired visual identity of the application, providing a consistent and cohesive user experience.

Implementing Advanced Material-UI Components:

Building on the basic components I implemented earlier, I delved into more advanced Material-UI components. This included components like menus, modals (dialogs), and responsive layouts. I witnessed how these components could be seamlessly integrated into my React application, enhancing its functionality and visual appeal.

Introduction to `useState`:

I was introduced to the `useState` hook, a fundamental part of React's state management. `useState` allows components to hold and manage their own state, enabling dynamic updates to the UI based on changes in state. I practiced implementing `useState` to create interactive components that respond to user interactions and trigger re-render

2.5 State Management and useState Hook:

State management is crucial for managing dynamic data and UI changes. React's useState hook allows functional components to manage state without using class components.

Hook The useState hook is used to add state to functional components. It returns an array containing the current state value and a function to update it.

The setState function returned by useState can be used to update the state. When the state is updated, react re-renders the component.

```

1 import React, { useState } from 'react';
2 import SendIcon from 'material-ui/icons/send';
3 import classes from './Assets/logo.module.css';
4 import ArrowDownwardIcon from 'material-ui/icons/arrowdownward';
5 import { Button, IconButton, Tabs, Tab } from 'material-ui/core';
6
7
8 const Home = () => {
9
10   const [x, setX] = useState(0);
11   const [num, setNum] = useState(Math.floor(Math.random() * 100));
12   const handleClick = () => {
13     return setNum(Math.floor(Math.random() * 100));
14   };
15
16   return (
17     <React.Fragment>
18       <div>This is Home Component</div>
19       <img className={classes.header} src={logo} alt="img not found" /> <div />
20       <button variant="contained" color="primary" onClick={handleClick}>Click Me</button><div>Your lucky number is {num}</div>
21       <button variant="contained" color="secondary" onClick={() => setX(x + 1)}>Add Item</button><div>Kohli</div>
22       <button variant="contained" color="secondary" onClick={() => setX(x - 1)}>Remove Item</button>
23       <div>Kohli</div>
24       <IconButton aria-label="delete" size="small">
25         <ArrowDownwardIcon fontSize="inherit" />
26       </IconButton>
27       <button
28         variant="contained"
29         color="primary"
30         onClick={handleClick} />
31     </div>
32     <Send
33     />
34     </React.Fragment>
35   );
36 }
37
38 export default Home

```

Fig. 2.5.1

A simple counter application demonstrates the usage of the useState hook to create dynamic and interactive UI elements.

The useState hook can also manage more complex state structures, such as objects and arrays.

It's important to follow best practices when using the useState hook, such as initializing state, updating state immutably, and considering performance implications.

❖ Rules of Hooks

1. Only Call Hooks at the Top Level
2. Only Call Hooks from React Functions
3. Don't Call Hooks Inside Loops, Conditions, or Nested Functi

Chapter 3: Implementation of project

Now, we are moving forward with the implementation of all the concepts we have learned in the previous four days in the Book-E-Store project. We will apply these concepts to enhance the functionality and user experience of the application.

3.1 Registration Page UI

On my fifth day of the internship, I delved into more advanced concepts and tools within React.js, including the `useEffect` hook and the Formik library. I applied this knowledge to create a Registration Page UI and implemented form validation using Formik.


Exploring Formik for Form Handling:


I delved into Formik, a library that simplifies form management in React applications. Formik streamlines tasks such as form state management, validation, and handling form submission. I discovered how Formik provides a structured approach to building forms that are both user-friendly and easy to maintain.

Creating a Registration Page UI:

Using the concepts, I learned about `useEffect` and Formik, I designed and implemented a Registration Page UI for the application. I utilized Material-UI components to create an intuitive and visually appealing form interface, which included fields for users to input registration details.

BOOKSTORE

Login | Register  0 Cart

What are you looking for...  Search

Home > [Create an Account](#)

Register your Account

Personal Information

Enter the following details to create your account.

First Name *

Last Name *

Email Address *

Roles

Login Information

Password *

Confirm Password *

[Register](#)

Fig. 3.1.1

Form Validation with Formik:

One of the highlights of the day was learning how to perform form validation using Formik. I implemented validation rules for various form fields, ensuring that user inputs met the required criteria before submission. This included validating fields such as email addresses, passwords, and other user information.

In summary, my fifth day of the internship marked a deeper dive into advanced React.js concepts and tools. I learned how to use the `useEffect` hook to manage side effects and lifecycle actions, explored the Formik library for form handling, and applied this knowledge to design and develop a Registration Page UI. Implementing form validation using Formik further enhanced the functionality and usability of the application's

registration process. This day's activities added valuable skills to my React.js toolkit and expanded my ability to create interactive and user-centric interfaces.

3.2 Registration Page API integration

On my sixth day of the internship, I ventured into the realm of asynchronous operations, API calls, and data handling within a React.js application. I covered concepts such as Promises, JSON, and employed the Axios library to make API calls. Additionally, I learned how to use toast notifications and seamlessly integrated API calls into the Registration Page.

Understanding Promises and JSON:

I delved into Promises, a mechanism for handling asynchronous operations in JavaScript. Promises provide a structured way to work with asynchronous code, enabling more organized and readable handling of tasks like fetching data from APIs.

JSON (JavaScript Object Notation) was also a focus. I learned how to serialize and deserialize data in JSON format, which is commonly used for exchanging data between a server and a web application.

Using Axios for API Calls:

I learned how to use the Axios library to make API calls in a React application. Axios simplifies the process of making HTTP requests and handling responses. It provides a concise syntax and supports features like request cancellation, interceptors, and automatic transformation of response data.

Handling API Response and Toast Notifications:

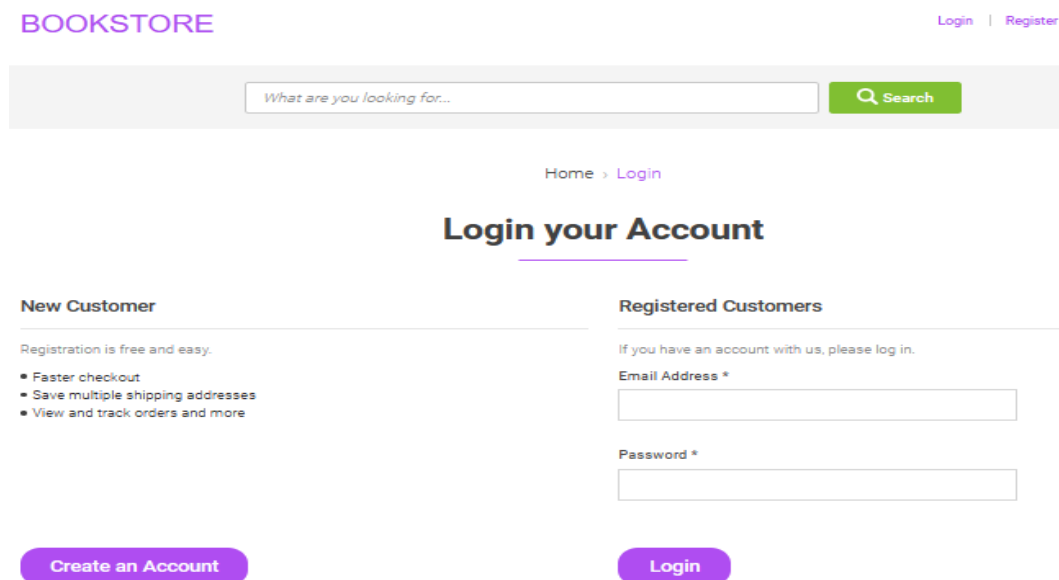
Understanding how to handle API responses is crucial. I learned how to interpret response status codes and response data, distinguishing between successful and erroneous responses

Integration of API Calls into Registration Page:

Putting the learned concepts into practice, I integrated API calls into the Registration Page I had previously designed. I used Axios to send registration data to the server and handle the response. This process involved sending user inputs, validating them, and displaying relevant toast notifications based on the API response.

3.3 Login Page UI & API integration

On my seventh day of the internship, I continued my journey by focusing on the creation of a Login Page UI and integrating API calls for the login functionality. This allowed me to build a complete user authentication process within the React.js application.



The screenshot displays the 'BOOKSTORE' login interface. At the top, the 'BOOKSTORE' logo is on the left, and 'Login | Register' links are on the right. Below this is a search bar with the placeholder text 'What are you looking for...' and a green 'Search' button. The main heading is 'Login your Account', preceded by a breadcrumb 'Home > Login'. The page is divided into two columns. The left column, titled 'New Customer', states 'Registration is free and easy.' and lists three benefits: 'Faster checkout', 'Save multiple shipping addresses', and 'View and track orders and more'. It features a blue 'Create an Account' button. The right column, titled 'Registered Customers', states 'If you have an account with us, please log in.' and contains two input fields: 'Email Address *' and 'Password *'. It features a blue 'Login' button.

Fig. 3.3.1

Creating the Login Page UI:

I started by designing and implementing a Login Page UI. I utilized Material-UI components to craft an intuitive and visually appealing interface for users to input their login credentials.

Integrating Login API Calls:

Building on the knowledge gained from previous API integration tasks, I extended my skills to integrate API calls for the login functionality. I used Axios to send user login credentials to the server and manage the response. This involved handling different response scenarios, such as successful login, incorrect credentials, or server errors.

User Authentication and State Management:

As part of the login process, I implemented user authentication. Upon successful login, I learned how to manage user authentication state, typically using techniques like JWT (JSON Web Tokens) to securely verify user identities and grant access to authorized resources.

Error Handling:

I also learned about effective error handling. In the case of incorrect login credentials or server errors, I implemented mechanisms to provide users with error toast . This ensured a user-friendly experience, even in scenarios where login attempts were unsuccessful.

Completing the Authentication Flow:

By integrating the Login Page UI with API calls, state management, and error handling, I successfully completed the user authentication flow. This allowed users to log in, receive appropriate toast based on their inputs, and gain access to protected areas of the application.

3.4 Building Header, Footer:

A header component was developed to provide navigation links and enhance the

Creating the Header and Footer:

I began by designing and implementing a Header and Footer for the application. The Header typically includes navigation links, logo, and user-related components, while the Footer provides important information and links that remain accessible throughout the app. By adding these elements, I ensured a consistent and organized layout, making navigation and access to essential information seamless for users.

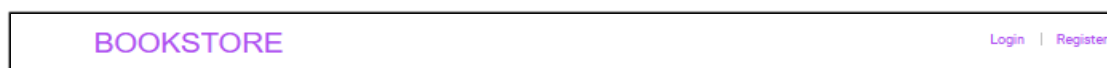


Fig. 3.4.1

Implementing Global Book Search:

To enhance user functionality, I introduced a global book search feature. This feature allowed users to search for books across the entire application, enabling quick access to relevant content. I integrated a search bar that allowed users to input keywords, and upon submission, I used Axios to make API calls to retrieve search results from the server.



Fig. 3.4.2

User Feedback and UI Refinement:

As I implemented the Header, Footer, and global book search, I ensured that user feedback and interactions were accounted for. For instance, I provided loading indicators during API calls to inform users about ongoing processes. I also refined the UI by incorporating responsive design principles to ensure optimal user experiences across various devices and screen sizes.

Bringing It All Together:

By implementing the Header and Footer, and enabling global book search, I contributed to creating a more intuitive and user-friendly application. These enhancements facilitated smoother navigation, consistent design, and quicker access to relevant content. This day's work underscored the importance of user-centric design and the impact of well-structured UI elements on overall user satisfaction.

In summary, my eighth day of the internship was focused on enhancing navigation and user functionality within the application. By creating a Header and Footer, and implementing a global book search feature, I aimed to provide users with a more seamless and efficient experience. This day's tasks aligned with the overarching goal of creating a user-centric and feature-rich React.js application.

3.5 AuthContext

On my 15 day of the internship, I delved into advanced React.js concepts to manage user authentication, implement private routes, and make use of the ``useContext`` hook for efficient state management. These concepts helped enhance security and user experience within the application

Implementing Private Routes:

Building on my knowledge of user authentication, I tackled the concept of private routes. Private routes are accessible only to authenticated users, preventing unauthorized access to certain areas of the application. I implemented a mechanism that redirects users to a login page if they attempted to access a private route without being authenticated.

Creating an AuthContext Using ``useContext``:

I applied the ``useContext`` hook to create an AuthContext, which stored user authentication state and provided methods to update that state. This context served as a central hub for handling user authentication data and actions. By utilizing the ``useContext`` hook, I was able to efficiently manage and share authentication-related information across different components.

3.6 Building the Book Listing Page:

The Book Listing page was implemented to display a list of books fetched from an API using Axios

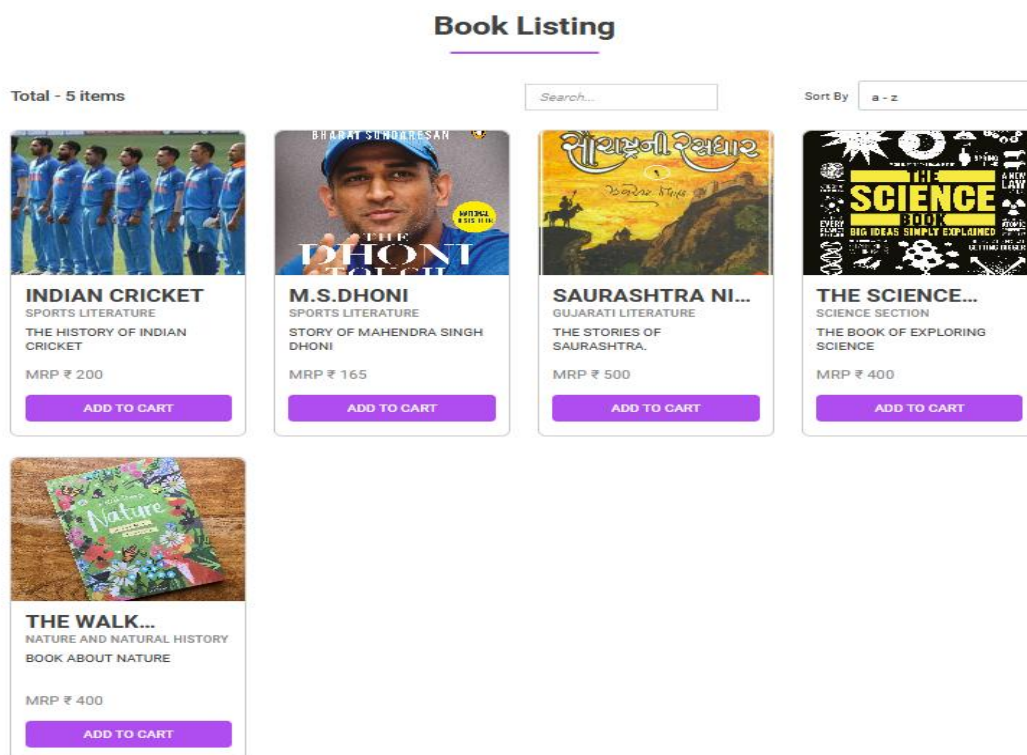


Fig. 3.6.1

Implementing Local Search:

I introduced local search functionality, enabling users to search for specific books based on keywords or titles. I created a search bar where users could enter their search queries, and I implemented a mechanism to filter and display matching books in real-time as users typed.

Adding Sorting Options:

To enhance the user experience further, I implemented sorting options. Users could choose how the book listings were sorted, whether by title, author, publication date, or other relevant criteria. This allowed users to customize the order in which they viewed the book listings.

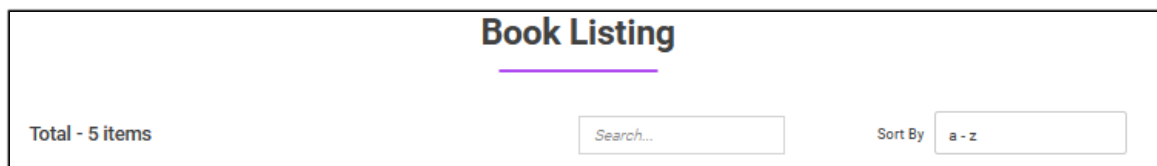


Fig. 3.6.2

3.7 Product CRUD

On my eleventh day of the internship, I extended the functionality of the Book Listing Page to cater to admins and sellers by implementing features to add, update, and delete books. These additions empowered admins and sellers to manage the book inventory effectively within the application.

Book Listing Page for Admins and Sellers:

I started by refining the Book Listing Page to differentiate between user roles. Admins and sellers needed additional functionalities beyond what regular users had access to. I designed the page to display a list of books specifically targeted for admins and sellers, providing them with a comprehensive overview of their inventory.

Implementing Add Book Functionality:

For admins and sellers, I introduced the ability to add new books to the application. I created a form that allowed users to input book details such as title, author, genre, and more. Upon submission, the data was sent to the server via an API call, and the book was added to the inventory.

Book Page			
		Search...	Add
Book Name	Price	Category	
SAURASHTRA NI RASDHAR	500	GUJARATI LITERATURE	Edit Delete
M.S.DHONI	165	SPORTS LITERATURE	Edit Delete
THE WALK THROUGH NATURE	400	NATURE AND NATURAL HISTORY	Edit Delete
THE SCIENCE BOOK	400	SCIENCE SECTION	Edit Delete
INDIAN CRICKET	200	SPORTS LITERATURE	Edit Delete
Rows per page: 10 1-5 of 5 < >			

Fig. 3.7.1

Enabling Update Book Functionality:

I implemented the functionality for admins and sellers to update existing book information..

Introducing Delete Book Feature:

To complete the management capabilities, I enabled the option to delete books from the inventory. Each book listing had a "Delete" button, which, when clicked, triggered an API call to remove the book from the database.

3.8 User List Page:

The User List page is a fundamental component of the Book Store project. It provides administrators with the ability to view, manage, and perform CRUD (Create, Read, Update, Delete) operations on user accounts.

User			
<div>Search...</div>			
First Name	Last Name	Email	Role
vijay	aniyariya	xyz@gmail.com	seller
aniyariya	vijay	abc@gmail.com	buyer
sandip	sandip	sandip@gmail.com	buyer
vijay	vijay	vijay@gmail.com	seller
admin	admin	admin@gmail.com	admin
firstname	lastname	newemail@gmail.com	seller
fname	lname	e@gmail.com	seller
f_name	l_name	Email@gmail.com	seller

Fig. 3.8.1

Implementing Add User Functionality:

For administrators, I introduced the capability to add new users to the application. I created a form where administrators could input user details such as username, email, role, and more. Upon submission, the data was sent to the server through an API call, and the new user account was created.

Enabling Update User Functionality:

I extended the functionality to allow administrators to update user information. Each user listing included an "Edit" button, which, when clicked, opened a form allowing administrators to modify user details. The updated information was then sent to the server, reflecting the changes in real-time on the User List Page.

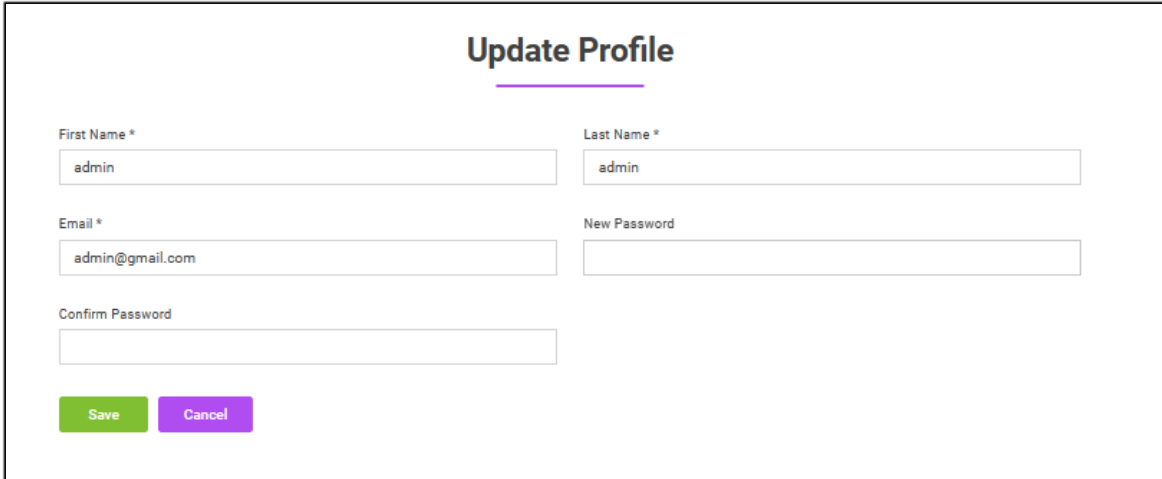
Introducing Delete User Feature:

To provide administrators with complete control, I added the ability to delete user accounts. Similar to the book management features, each user listing had a "Delete" button. Clicking this button triggered an API call to remove the user from the database, ensuring effective user account management.

3.9 User Profile and Category List Page

The User Profile page allows users to manage their individual profiles. Users can view and modify their account information, such as their username, email, and password.

The Category List page plays a pivotal role in organizing the Book Store's inventory. Administrators can manage categories by adding, updating, and deleting them.



Update Profile

First Name *
admin

Last Name *
admin

Email *
admin@gmail.com

New Password

Confirm Password

Save Cancel

Fig. 3.9.1

Creating the Category List Page:

For administrators and sellers, I designed and implemented a Category List Page. This page displayed a list of product categories available within the application. Categories are vital for organizing and managing products effectively

Category

Search... Add

Category Name	
VIJAY	Edit Delete
NATURE AND NATURAL HISTORY	Edit Delete
GUJARATI LITERATURE	Edit Delete
SPORTS LITERATURE	Edit Delete
SCIENCE SECTION	Edit Delete

Rows per page: 10
 1-5 of 5
 <
>

Fig. 3.9.2

Adding, Updating, and Deleting Categories:

I introduced features that allowed administrators to add, update, and delete categories. For adding categories, administrators could input category details such as name and description through a form. Similarly, I enabled category updating by providing an "Edit" button that opened a form for modifying category details. Admins could also delete categories using a "Delete" button, which triggered an API call to remove the category from the database.

3.10 Cart Page

The Cart Page is a critical component of the Book Store application, enabling users to manage their selected items before proceeding to the checkout process. It offers a seamless shopping experience by allowing users to review their chosen books, adjust quantities, and remove items.

The Cart Page leverages the Cart Context to efficiently manage the user's shopping cart state across different components. The context ensures that items persist as users navigate through the application.

The checkout process culminates in the Placing Order functionality. Once users are satisfied with their selections, they can proceed to place their orders. This involves confirming their cart contents, entering shipping details, and finalizing the purchase.

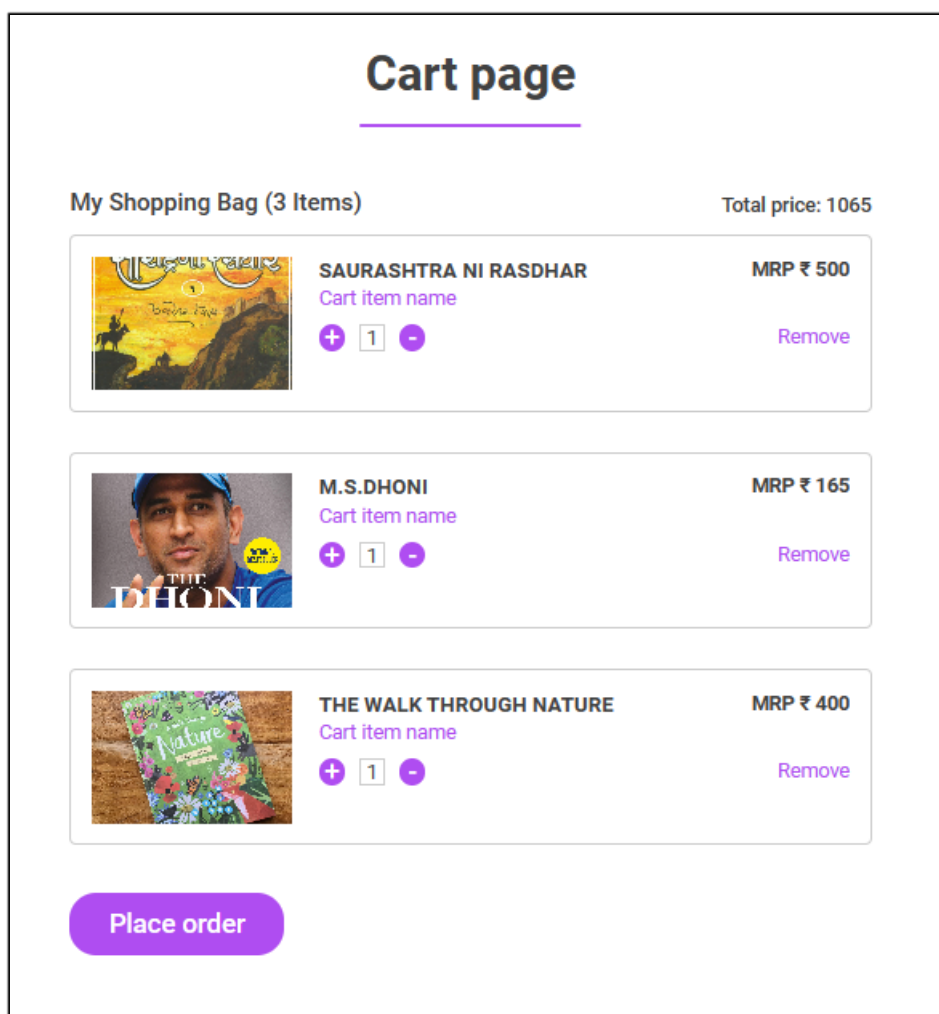


Fig. 3.10.1

3.11 AWS for Deployment

Deploying the Book Store application to the cloud is a pivotal step towards making it accessible to users globally. Amazon Web Services (AWS) provides a robust platform for hosting and scaling applications.

The deployment process involves utilizing various AWS services, including Amazon S3 for hosting static assets, Amazon CloudFront for content delivery, and Amazon Amplify for managing the deployment pipeline.

Amazon Amplify simplifies the deployment process by providing an intuitive interface for setting up deployment pipelines. This includes linking the application repository, configuring build settings, and automating deployment triggers.

With the deployment pipeline in place, continuous deployment is achieved. Any changes pushed to the repository trigger an automated build and deployment process, ensuring that the Book Store application is always up to date.

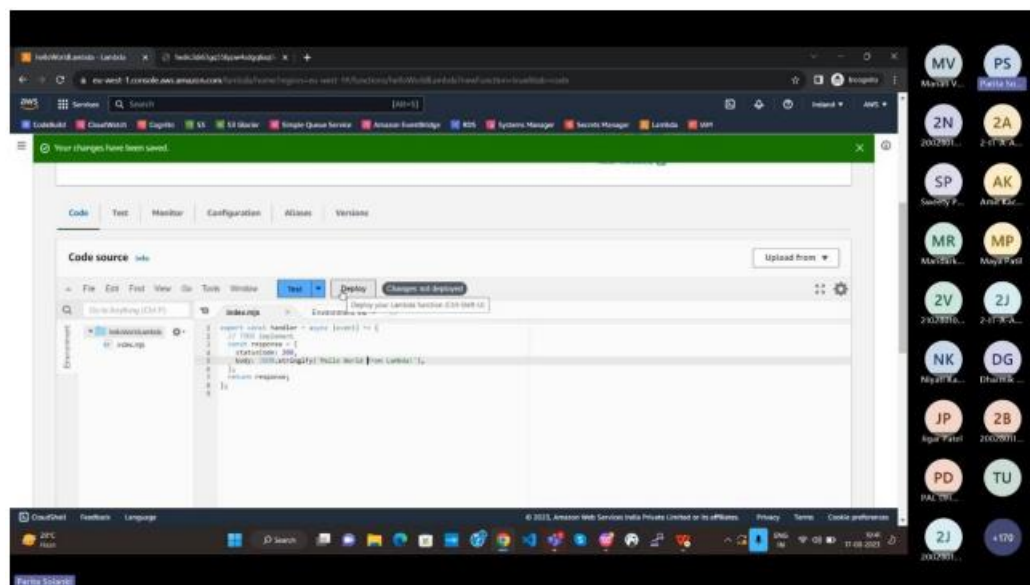


Fig. 3.11.1

Chapter 4: Conclusion and Future Work

Throughout this internship, I successfully built a Book Store Project using React JS. I gained hands-on experience in component creation, state management, API integration, and user authentication.

Future work on the project could include implementing user profiles, a shopping cart, and order processing. Additionally, optimizing the application for performance and scalability would be a priority.

I learned the importance of modular design, code organization, and efficient state management. The project also provided insights into handling asynchronous operations and user interactions.

I would like to express my gratitude to my mentors and the entire development team for their guidance and support during this internship.

Chapter 5: References

- <https://www.youtube.com/playlist?list=PLC3y8-rFHvwgg3vaYJgHGnModB54rxOk3>
- <https://www.youtube.com/playlist?list=PLwGdqUZWnOp3aROg4wypcRhZqJG3ajZWJ>
- <https://www.w3schools.com/REACT/DEFAULT.ASP>
- <https://legacy.reactjs.org/docs/getting-started.html>
- <https://www.javatpoint.com/reactjs-tutorial>
- <https://www.geeksforgeeks.org/reactjs-tutorials>