BOOK STORE APPLICATION

A PROJECT REPORT

Submitted by

Goswami Dushyantpuri Lalitpuri

200200107023

In partial fulfillment for the award of the degree of

BACHELOR OF ENGINEERING

in

Computer Engineering Department

Government Engineering College Rajkot

Gujarat Technological University, Ahmedabad

July, 2023

Government Engineering College, Rajkot

Rajkot, Gujarat

CERTIFICATE

This is to certify that the project report submitted along with the project entitled **Book Store Application** has been carried out by **Goswami Dushyantpuri Lalitpuri** under my guidance in partial fulfillment for the degree of Bachelor of Engineering in Computer Engineering, 8th Semester of Gujarat Technological University, Ahmedabad during the academic year 2021-22.

Prof. Jahnvi Doshi

Internal Guide

Prof. (Dr.) Sanjay Shah

Head of the Department

TatvaSoft
sculpting thoughts...

Outsourcing • Custom Software Development • Web Application & eBusiness Solution

Summer Internship Certificate

Date:18/08/2023

This is to certify that Goswami Dushyantpuri Lalitpuri 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

[illegible]

TatvaSoft, Ahmadabad

Government Engineering College, Rajkot

Rajkot, Gujarat

DECLARATION

We hereby declare that the Internship / Project report submitted along with the Internship / Project entitled **Book Store Application** submitted in partial fulfillment 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 / us at TatvaSoft and that no part of this report has been directly copied from any students‟ reports or taken from any other source, without providing due reference.

	Name of the Student	Sign of Student
1.	Goswami Dushyantpuri Lalitpuri	

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 Asst. Prof. Jahnvi Doshi for their guidance and constant supervision as well as for providing necessary information regarding the Summer Internship Project Titled “Book Store Application”. 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.

Goswami Dushyantpuri
(200200107023)

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 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.

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.

List of Figures

List of Tables

Abbreviations

List of Abbreviations:

- API: Application Programming Interface
- CSS: Cascading Style Sheets
- DOM: Document Object Model
- HTML: HyperText Markup Language
- JSX: JavaScript XML
- NPM: Node Package Manager
- UI: User Interface
- URL: Uniform Resource Locator

Table of Contents

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 application 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

I conducted my internship under the guidance and mentorship of TatvaSoft, a renowned IT company with a solid reputation for delivering innovative software solutions. The company's commitment to excellence and its emphasis on embracing the latest technologies provided me with an ideal environment to learn and grow.

1.3 Project Scope: Building a Book Store Application

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 user-friendly interface, real-time search functionality, categorization of books, and a shopping cart system.

Chapter 2 Learning Technologies

2.1 Introduction to React JS & Component-Based Architecture

React JS is a widely used JavaScript library for building user interfaces. It offers a component-based architecture that allows developers to create reusable UI components and efficiently manage the rendering of dynamic data.

The primary objective of this internship is to gain practical experience in building a Book Store application using React JS. Throughout this internship, I will learn various concepts and techniques related to React JS, including component creation, API integration, state management, and user interface design.

React's component-based architecture promotes the modular design of user interfaces. Components are the building blocks of React applications, encapsulating both the UI elements and their behavior. This architecture enhances code reusability and maintainability.

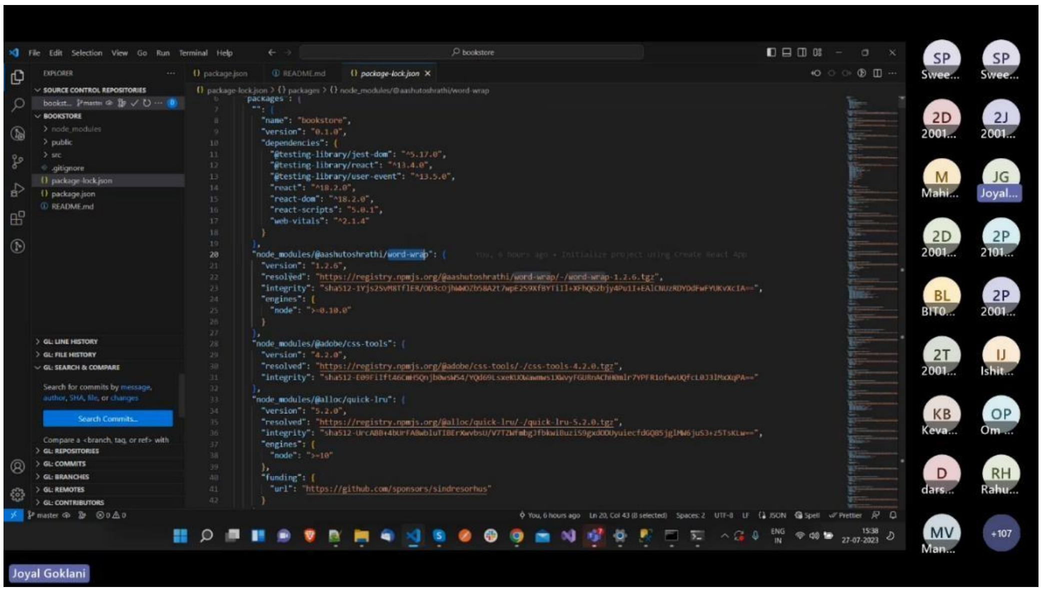
React employs a virtual object model, where it creates a virtual representation of the actual DOM. This virtual DOM allows React to efficiently update the real DOM by calculating the minimal changes required, resulting in improved performance.

Node.js serves as an excellent development environment for React projects. Its package manager, NPM (Node Package Manager), simplifies the process of installing, managing, and sharing JavaScript libraries and dependencies.

To initiate a React project, I used the command npx create-react-app project-name. This command sets up a new project with the necessary files and directory structure.

NPM is used to manage project dependencies. Commands like npm install react-router-dom allow us to install packages for specific functionalities, such as routing in web applications.

The project structure consists of several key directories, including node modules, public, and src. The node modules directory contains installed packages, public houses publicly accessible assets, and src contains the application's source code.



```
packages": {
  "": {
    "name": "bookstore",
    "version": "0.1.0",
    "dependencies": {
      "@testing-library/jest-dom": "^5.17.0",
      "@testing-library/react": "^13.4.0",
      "@testing-library/user-event": "^13.5.0",
      "react": "^18.2.0",
      "react-dom": "^18.2.0",
      "react-scripts": "5.0.1",
      "web-vitals": "^2.1.4"
    }
  },
  "node_modules/@aashutoshrathi/word-wrap": {
    "version": "1.2.6",
    "resolved": "https://registry.npmjs.org/@aashutoshrathi/word-wrap/-/word-wrap-1.2.6.tgz",
    "integrity": "sha512-1Yjs2SvM8TflER/OD3cOjhWWOZb58A2t7wpE2S9XfBYTiIl+XFhQG2bjy4Pu1I+EAlCNUzRDYDdFwFYUKvXcIA==",
    "engines": {
      "node": ">=0.10.0"
    }
  },
  "node_modules/@adobe/css-tools": {
    "version": "4.2.0",
    "resolved": "https://registry.npmjs.org/@adobe/css-tools/-/css-tools-4.2.0.tgz",
    "integrity": "sha512-E09FiIft46CmH5Qnjb0wsW54/YQd69LsxeKUOWawmws1XWvyFGURnAChH0mlr7YPFR1ofwvUQfcL0J3lMxXqPA=="
  },
  "node_modules/@alloc/quick-lru": {
    "version": "5.2.0",
    "resolved": "https://registry.npmjs.org/@alloc/quick-lru/-/quick-lru-5.2.0.tgz",
    "integrity": "sha512-UrcABB+4bUrFABwbluTIBErXwvbsU/V7TZWfmbgJfbkwiBuziS9gxdODUyuiecfdGQ85jglMW6juS3+z5TsKLw==",
    "engines": {
      "node": ">=10"
    },
    "funding": {
      "url": "https://github.com/sponsors/sindresorhus"
    }
```

Fig 2.1 Project Structure

Table 2.1 Commonly Used NPM Commands

Command	**Description**
npm start	Starts the development server.
npm run build	Creates a production-ready build of the app.
npm install package	Installs a package and adds it to dependencies.
npm uninstall package	Removes a package from dependencies.

2.2 Introduction to Components and Basic UIElements

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" />
```

2.2.1 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 theapplication, 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.

2.2.2 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].
```

2.2.3 Props: Passing 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:

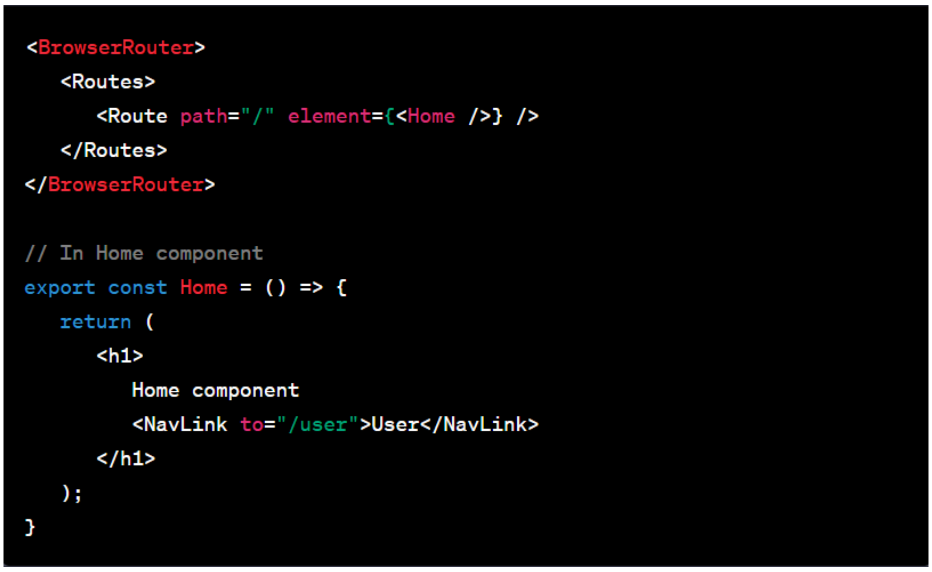
```
// In parent component
 <User name={userName} />

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

2.2.4 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:



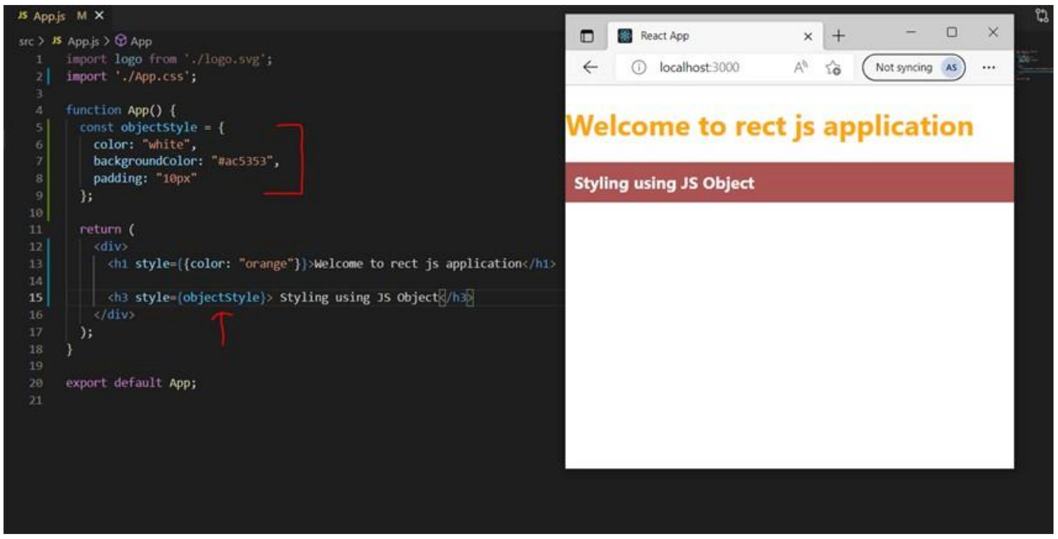
```
<BrowserRouter>
   <Routes>
      <Route path="/" element={<Home />} />
   </Routes>
</BrowserRouter>

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

Fig 2.2 React Router DOM

2.2.5 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.



```
import logo from './logo.svg';
import './App.css';

function App() {
  const objectStyle = {
    color: "white",
    backgroundColor: "#ac5353",
    padding: "10px"
  };

  return (
    <div>
      <h1 style={{color: "orange"}}>Welcome to rect js application</h1>

      <h3 style={objectStyle}> Styling using JS Object</h3>
    </div>
  );
}

export default App;
```

Welcome to rect js application

Styling using JS Object

Figure 2.3 Styling with CSS in React

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.

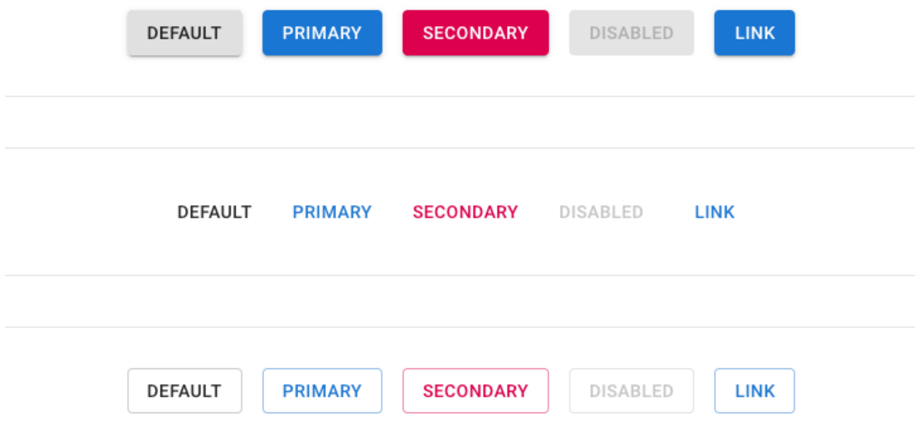


Fig 2.4 Material UI Buttons

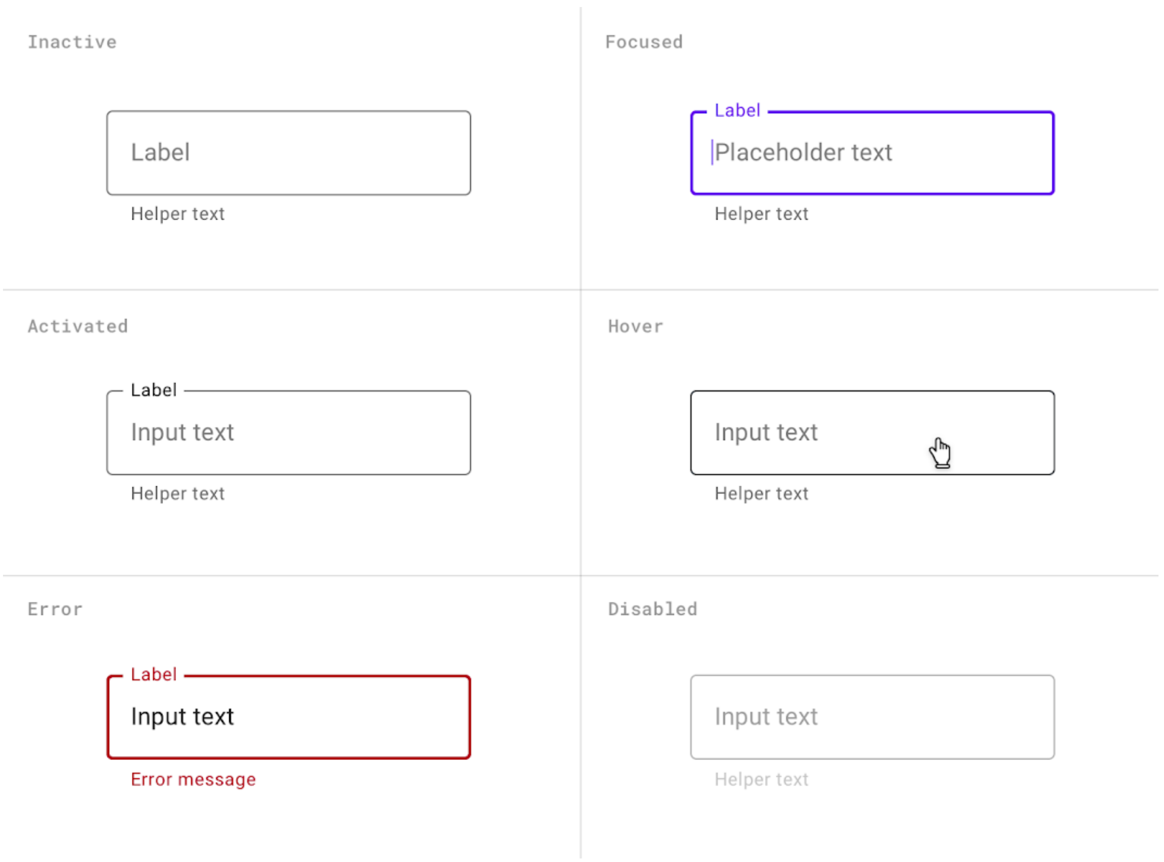


Fig 2.5 Material UI TextFields

2.3.1 Material UI Documentation

Material UI's official documentation provides comprehensive information on available components, customization options, and usage guidelines.

Table 2.2 Basic Material UI Components

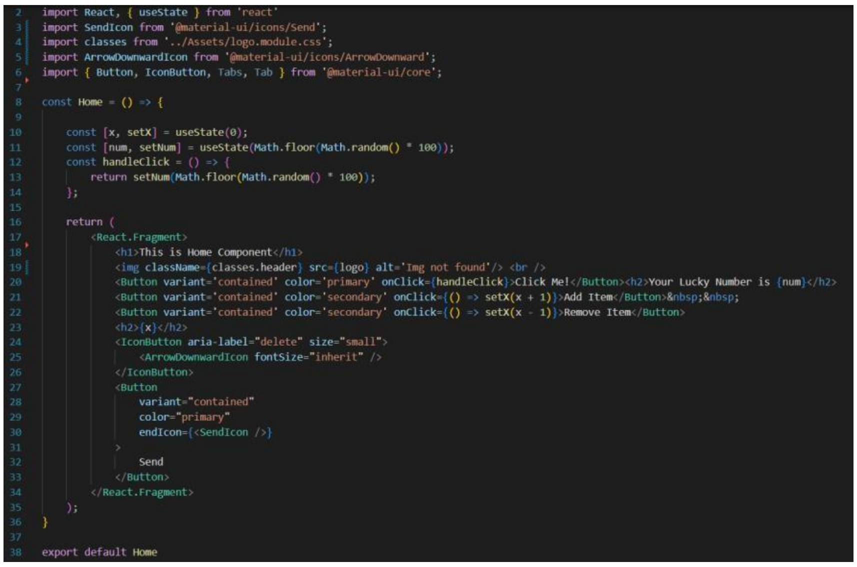
Component	**Description**
Button	Standard button with various styles.
TextField	Input field for text-based user input.
Checkbox	Checkbox input element for selections.
Radio	Radio button input for exclusive choices.
Typography	Component for handling text formatting.

2.4 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.



```
import React, { useState } from 'react'
import SendIcon from '@material-ui/icons/Send';
import classes from '../Assets/logo.module.css';
import ArrowDownwardIcon from '@material-ui/icons/ArrowDownward';
import { Button, IconButton, Tabs, Tab } from '@material-ui/core';

const Home = () => {

    const [x, setX] = useState(0);
    const [num, setNum] = useState(Math.floor(Math.random() * 100));
    const handleClick = () => {
        return setNum(Math.floor(Math.random() * 100));
    };

    return (
        <React.Fragment>
            <h1>This is Home Component</h1>
            <img className={classes.header} src={logo} alt='Img not found'/> <br />
            <Button variant='contained' color='primary' onClick={handleClick}>Click Me!</Button><h2>Your Lucky Number is {num}</h2>
            <Button variant='contained' color='secondary' onClick={() => setX(x + 1)}>Add Item</Button>&nbsp;&nbsp;
            <Button variant='contained' color='secondary' onClick={() => setX(x - 1)}>Remove Item</Button>
            <h2>{x}</h2>
            <IconButton aria-label="delete" size="small">
                <ArrowDownwardIcon fontSize="inherit" />
            </IconButton>
            <Button
                variant="contained"
                color="primary"
                endIcon={<SendIcon />}
            >
                Send
            </Button>
        </React.Fragment>
    );
}

export default Home
```

Figure 2.6 Implementing a Counter using useState

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

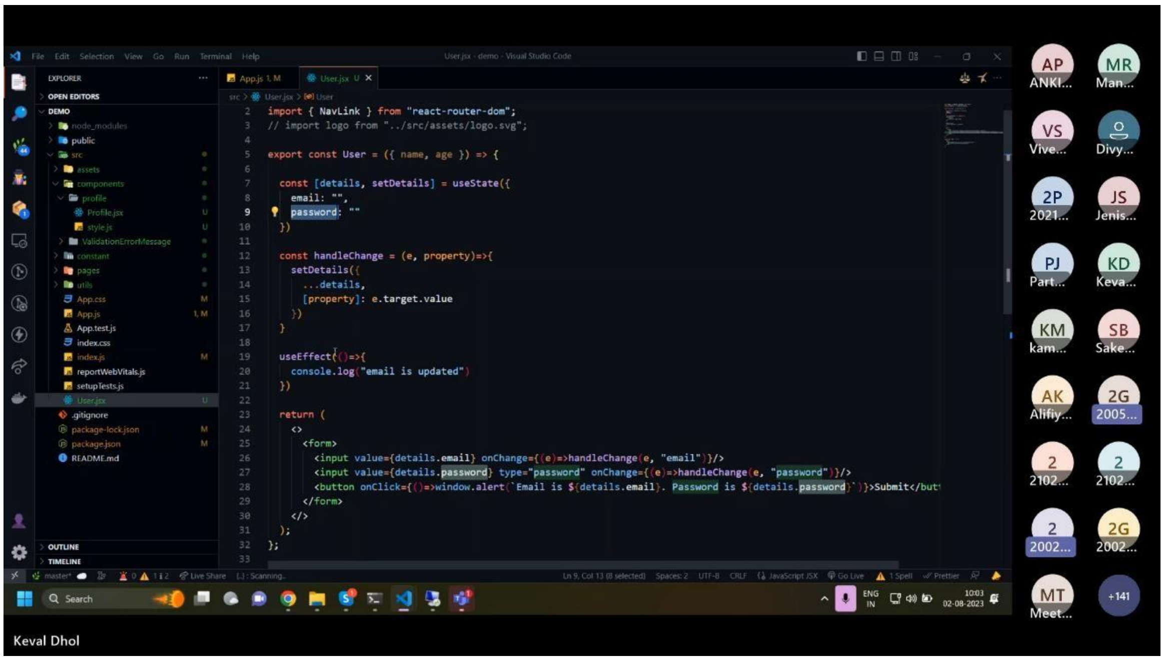
1. Only Call Hooks at the Top Level
2. Only Call Hooks from React Functions
3. Don’t Call Hooks Inside Loops, Conditions, or Nested Functions

2.5 Effects and useEffect Hook

Side effects, such as data fetching, DOM manipulation, and subscriptions, are actions that can occur in React components.

The useEffect hook enables functional components to perform side effects after rendering. It replaces the lifecycle methods in class components.

The useEffect hook takes two arguments: a function that contains the side effect logic and an array of dependencies that trigger the effect.



```
import { NavLink } from "react-router-dom";
// import logo from "../src/assets/logo.svg";

export const User = ({ name, age }) => {

  const [details, setDetails] = useState({
    email: "",
    password: ""
  })

  const handleChange = (e, property)=>{
    setDetails({
      ...details,
      [property]: e.target.value
    })
  }

  useEffect(()=>{
    console.log("email is updated")
  })

  return (
    <>
      <form>
        <input value={details.email} onChange={(e)=>handleChange(e, "email")}/>
        <input value={details.password} type="password" onChange={(e)=>handleChange(e, "password")}/>
        <button onClick={()=>window.alert(`Email is ${details.email}. Password is ${details.password}`)}>Submit</butt
      </form>
    </>
  );
};
```

Fig 2.7 Fetching Data with useEffect

An example of using the useEffect hook to fetch data from an API and update the component's state is demonstrated.

The useEffect hook can also be used to clean up after a component. This is especially useful for unsubscribing from subscriptions or canceling network requests.

Combining the useState and useEffect hooks allows components to fetch and manage data from APIs and update the UI accordingly.

Table 2.3 Dependencies Array in useEffect

Dependencies Array Content	**Effect Behavior**
[]	Runs after the initial render.
[value]	Runs when the value changes.
undefined	Runs after every render.
null	Never runs after any render.
[...props]	Runs when any of the specified props change.
['prop1', 'prop2']	Runs when either prop1 or prop2 changes.

2.6 Introduction to Formik for Form Management

Form handling is a common task in web applications. Formik is a library that simplifies form management and validation in React.

Formik can be installed using NPM: npm install formik.

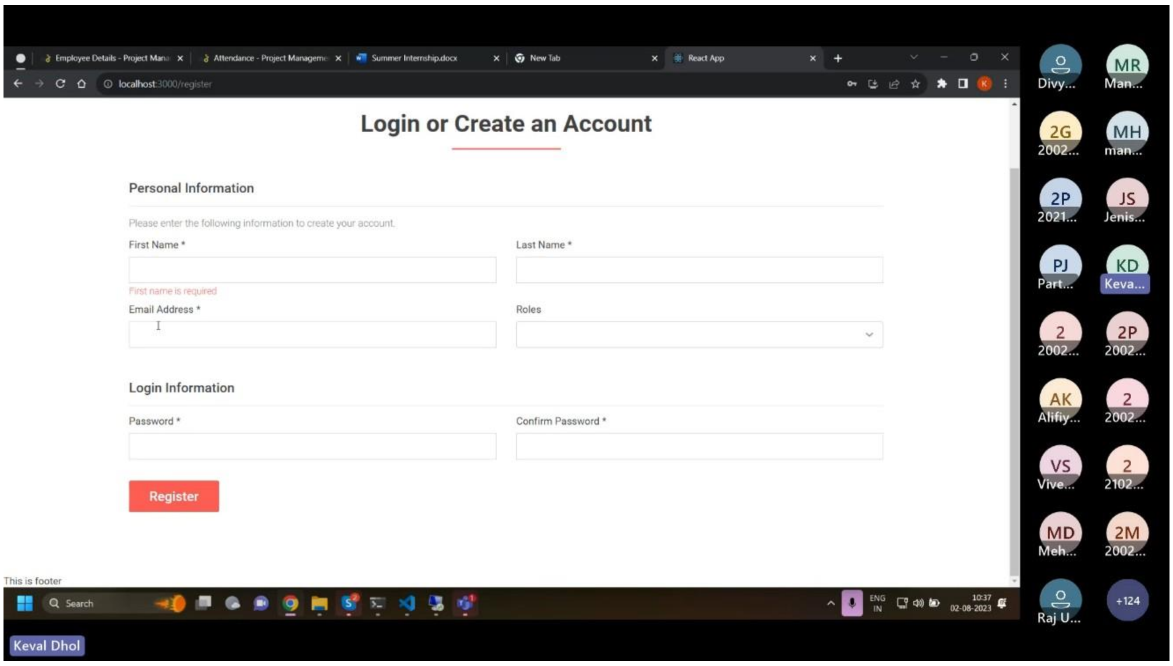
Formik provides a simple and intuitive API for building forms. I created a registration form with fields for username, email, and password.

Formik simplifies form submission by providing a handleSubmit function that encapsulates the form's submission logic.

Formik integrates seamlessly with Yup, a JavaScript library for schema validation.

Yup simplifies form validation and error handling.

Formik's errors object can be used to display validation errors to users, creating a better user experience.



Login or Create an Account

Personal Information

Please enter the following information to create your account.

First Name *

First name is required

Last Name *

Email Address *

Roles

Login Information

Password *

Confirm Password *

Register

This is footer

Fig 2.8 Building a Registration Form using Formik

Table 2.4 Formik's Essential Props

Prop	Description
initialValues	Initial values of form fields.
onSubmit	Function to handle form submission.
validationSchema	Schema for form field validation.
errors	Object containing validation errors.
touched	Object indicating whether fields were touched.

2.7 Making API Calls with Axios

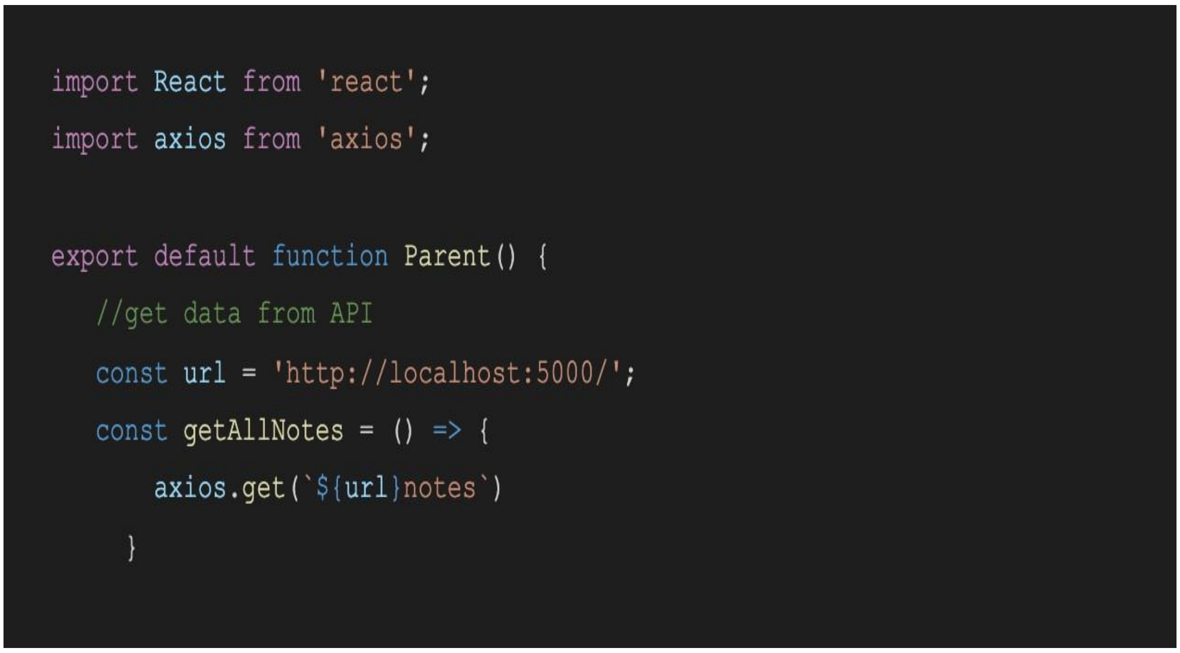
Axios is a popular JavaScript library used to make HTTP requests. It simplifies the process of sending and receiving data from APIs.

Axios can be installed using NPM: npm install axios.

Axios allows us to make GET requests to retrieve data from APIs. The fetched data can be used to update the component's state

POST requests are used to send data to APIs. Axios enables us to send data and receive responses, facilitating interactions with the backend.

.



```
import React from 'react';
import axios from 'axios';

export default function Parent() {
   //get data from API
   const url = 'http://localhost:5000/';
   const getAllNotes = () => {
       axios.get(`${url}notes`)
     }
```

Figure 2.9 Making GET Requests with Axios

Axios responses contain various properties, including data, status, and headers. These properties provide information about the API response.

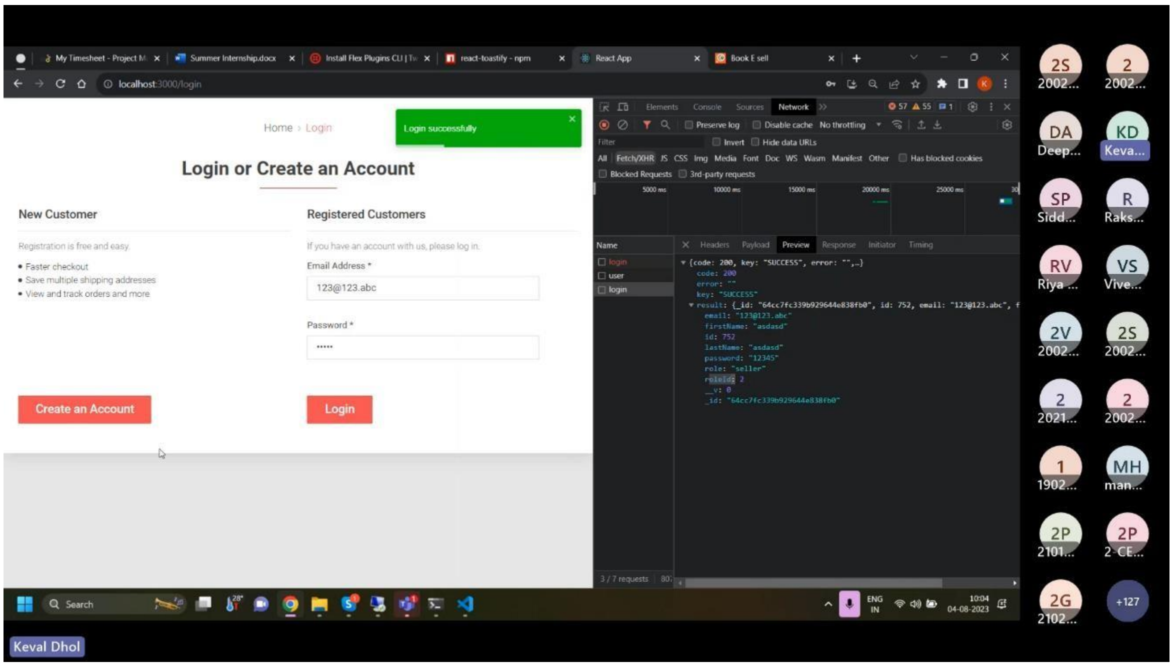
Table 2.5 Common HTTP Status Codes

Status Code	Description
200	OK - Successful request.
201	Created - New resource created.
400	Bad Request - Invalid request format.
401	Unauthorized - Authentication required.
404	Not Found - Resource not found.
500	Internal Server Error - Server issue.

2.8 User Authentication and Protected Routes

User authentication ensures secure access to web applications. It involves verifying user identities and authorizing access to specific resources.

I created a login page where users can enter their credentials. Axios was used to send login requests to the backend for validation.


localhost:3000/login
Home › Login
Login successfully
Login or Create an Account
New Customer
Registration is free and easy.
• Faster checkout
• Save multiple shipping addresses
• View and track orders and more
Create an Account
Registered Customers
If you have an account with us, please log in.
Email Address *
123@123.abc
Password *
Login
Keval Dhol

```
▾ {code: 200, key: "SUCCESS", error: "",…}
    code: 200
    error: ""
    key: "SUCCESS"
  ▾ result: {_id: "64cc7fc339b929644e838fb0", id: 752, email: "123@123.abc", f
      email: "123@123.abc"
      firstName: "asdasd"
      id: 752
      lastName: "asdasd"
      password: "12345"
      role: "seller"
      roleId: 2
      __v: 0
      _id: "64cc7fc339b929644e838fb0"
```

Figure 2.10 Creating a Login Page and Sending Requests

Flow Upon successful login, a token is usually provided by the backend. This token is then stored in local storage or cookies for subsequent requests.

Private routes restrict access to certain pages based on user authentication. React Router's Route component can be used to implement private routes.

Users Unauthenticated users attempting to access private routes are redirected to the login page. This ensures proper user authentication flow.

Table 2.6 Components for User Authentication Flow

Component	**Description**
Login Page	User enters credentials and logs in.
Private Route	Restricts access to specific routes.
useAuth	Custom hook to manage user authentication state.
authReducer	Reducer function for managing authentication state.

Chapter 3 Working on Project

3.1 Building Header, Footer, and Navigation

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

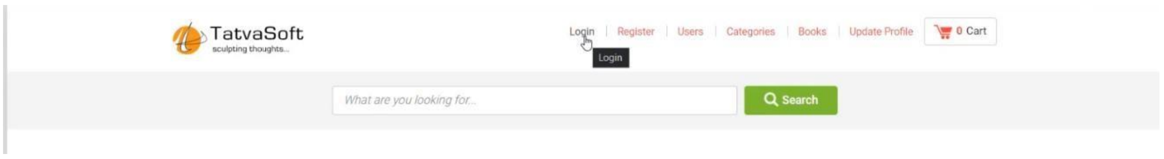


Figure 3.1 Creating a Header Component

A footer component was added to display copyright information and enhance the

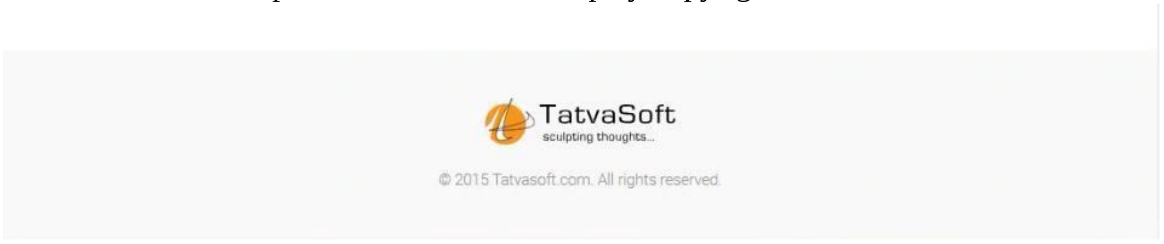


Figure 3.2 Implementing a Footer Component

React Router's Link component was utilized to create navigation links between different pages of the application.

Table 3.1 React Router Navigation Components

Component	Description
BrowserRouter	Provides routing functionality for the app.
Route	Defines a route and its corresponding component.
Link	Creates a navigation link to another route.
NavLink	Similar to Link, with added styling options.

3.2 Building the Book Listing Page

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

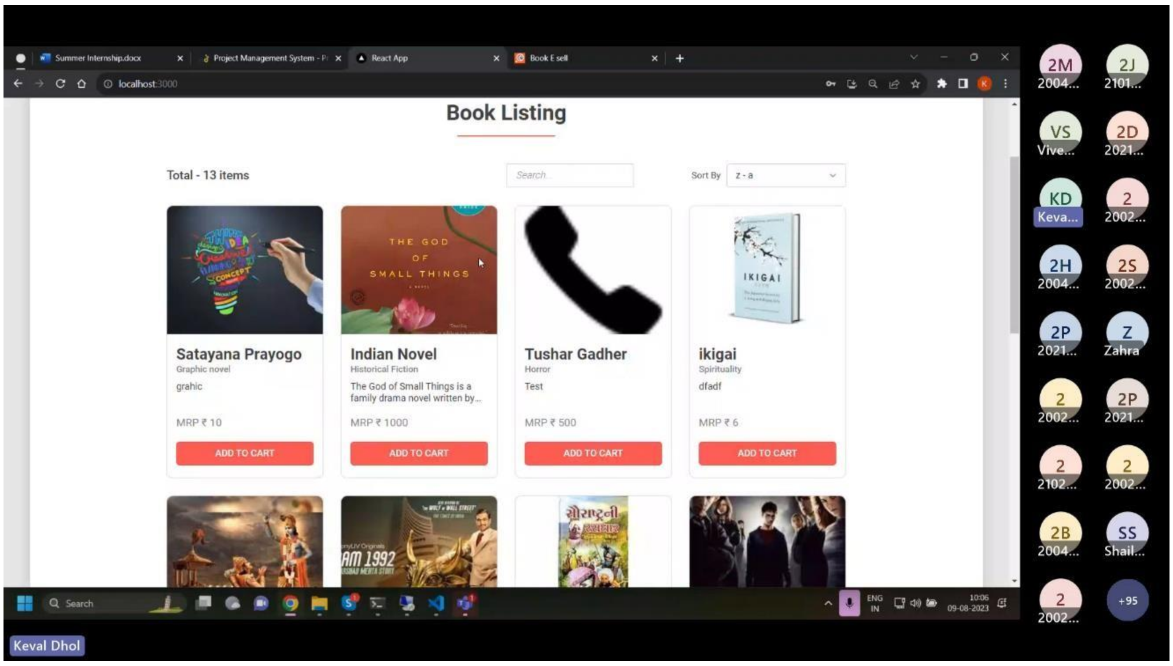


Figure 3.3 Building the Book Listing Page

Material UI's Grid component was utilized to structure and style the layout of the book listings.

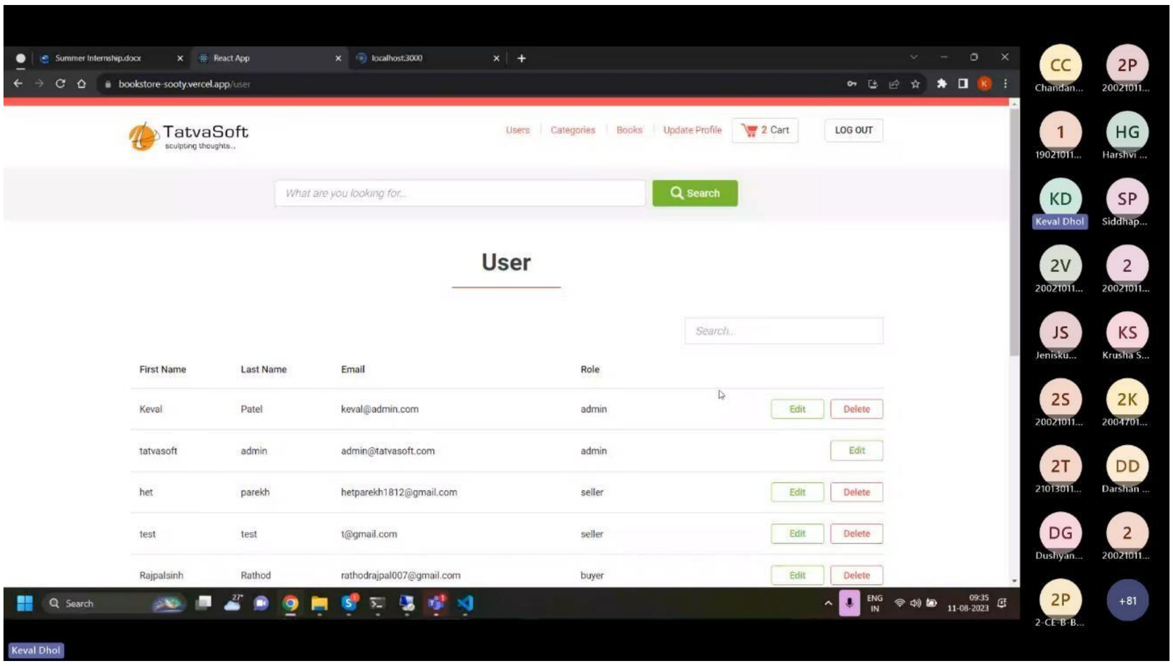
A modal was implemented to display additional details about a selected book when a user clicks on it.

Table 3.2 Material UI Grid Properties

Property	Description
container	Determines whether the grid is a container.
item	Indicates that the grid is an item within a container.
xs, sm, md, lg, xl	Defines the grid behavior for different screen sizes.

3.3 User List Page

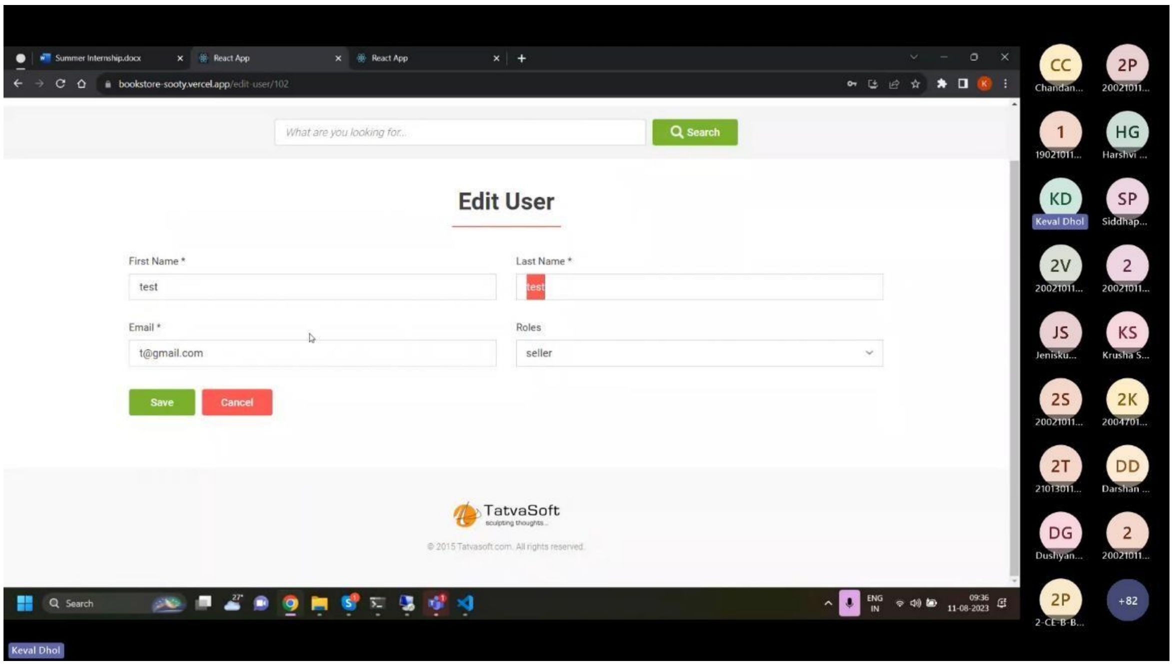
The User List page is a fundamental component of the Book Store application. It provides administrators with the ability to view, manage, and perform CRUD (Create, Read, Update, Delete) operations on user accounts.



First Name	Last Name	Email	Role		
Keval	Patel	keval@admin.com	admin	Edit	Delete
tatvasoft	admin	admin@tatvasoft.com	admin		Edit
het	parekh	hetparekh1812@gmail.com	seller	Edit	Delete
test	test	t@gmail.com	seller	Edit	Delete
Rajpalsinh	Rathod	rathodrajpal007@gmail.com	buyer	Edit	Delete

Fig 3.4 User List Page

Through a user-friendly interface, administrators can add new users by providing essential details, update existing user information, and remove users as needed.


bookstore-sooty.vercel.app/edit-user/102
What are you looking for...
Search
Edit User
Save
Cancel
TatvaSoft
© 2015 Tatvasoft.com. All rights reserved.

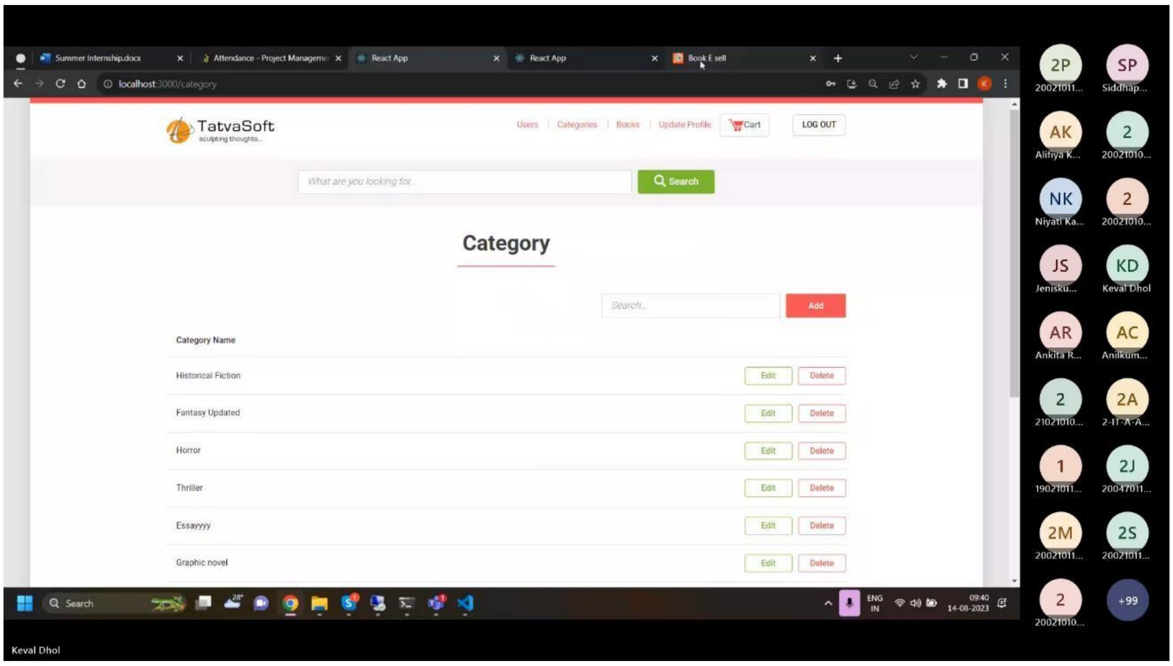
First Name *	test
Last Name *	test
Email *	t@gmail.com
Roles	seller

Figure 3.5 Edit User Page

3.4 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.



Category Name		
Historical Fiction	Edit	Delete
Fantasy Updated	Edit	Delete
Horror	Edit	Delete
Thriller	Edit	Delete
Essayyyy	Edit	Delete
Graphic novel	Edit	Delete

Figure 3.6 User Profile and Category List Page

3.5 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.

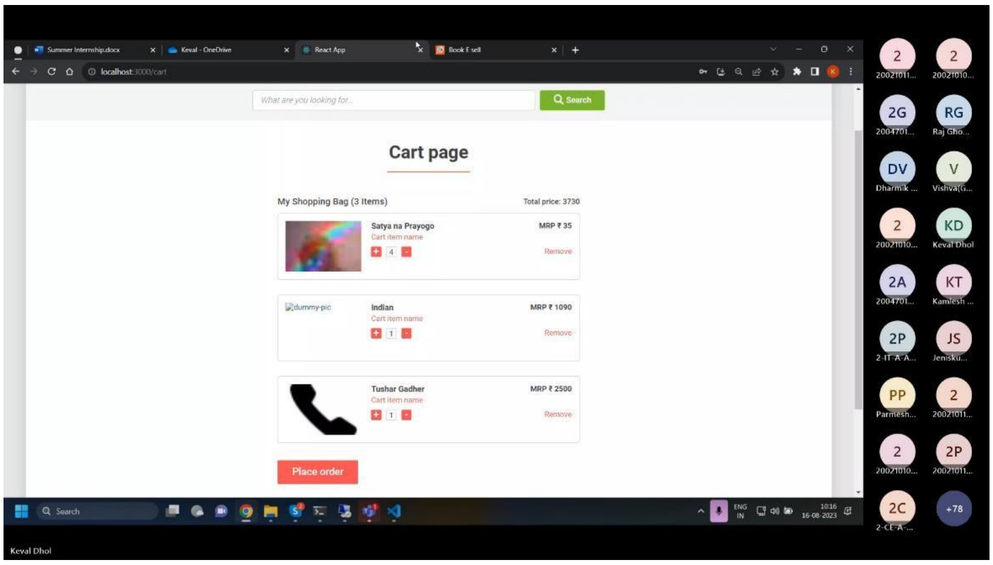


Figure 3.7 Cart Page and Placing Orders

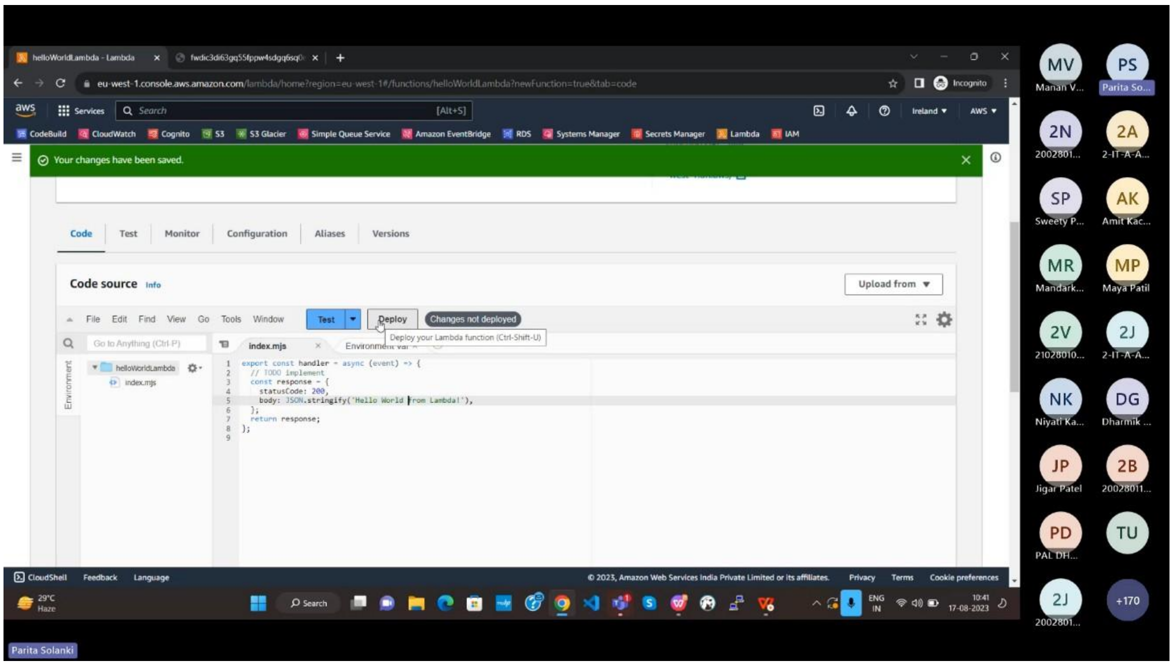
3.6 AWS Deployment

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.



Your changes have been saved.

Code source

```
export const handler = async (event) => {
  // TODO implement
  const response = {
    statusCode: 200,
    body: JSON.stringify('Hello World from Lambda!'),
  };
  return response;
};
```

Figure 3.8 AWS Deployment Pipeline

Chapter 4 Conclusion and Future Work

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

Future work on the application 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.

References

- **W3School:** https://www.w3schools.com/
- **Bootstrap:** https://getbootstrap.com/
- **JavatPoint:** https://www.javatpoint.com/java-tutorial
- **TutorialPoint:** https://www.tutorialspoint.com
- **React Js:** https://legacy.reactjs.org/
- **React Bootstrap:** https://react-bootstrap.netlify.app/
- **GeeksForGeeks:** https://www.geeksforgeeks.org/reactjs-introduction/
- **Api:** https://rapidapi.com/collection/list-of-free-apis
- **Node Js:** https://nodejs.org/en/download/
- **AWS Session Deployment:** https://www.youtube.com/playlist?list=PLC3y8-rFHvwgg3vaYJgHGnModB54rxOk3