DHARMSINH DESAI UNIVERSITY,NADIAD

FACULTY OF MANAGEMENT AND INFORMATION SCIENCE
BCA DEPARTMENT
SEMESTER-IV
Project Report From 04/12/2024 to 20/12/2024

Name:Sujal Singal PankajKumar
ID No:22BCUOS046
Roll No:BC066

Sr No	Date	Title
1	04/12/2024	Introduction to Basics of ReactJS
2	05/12/2024	Introduction to JavaScript and Its Usage in ReactJS
3	06/12/2024	CSS Fundamentals, Flexbox, Introduction to Responsive Design & Layout
4	09/12/2024	MVC in ReactJS and Working with APIs (Fetch & Axios)
5	10/12/2024	Revisiting MVC Architecture, Integration in Full-Stack Applications
6	11/12/2024	Handling Data for Controllers and Data Passing Techniques
7	12/12/2024	Understanding MVC Basics and Integration in ReactJS
8	13/12/2024	Final Review of MVC Concepts in ReactJS and Full-Stack Application Design
9	16/12/2024	Initialized React app and installed necessary packages. Designed the navbar, homepage layout, and footer. Set up basic routing using **react-router-dom**.
10	17/12/2024	Designed the **About Us**, **Contact Us**, and **Products** pages. Integrated

		fake API for data fetching, and implemented category-based product filtering.
11	18/12/2024	Integrated **Redux** for managing cart state. Designed actions and reducers for adding/removing items in the cart. Calculated total amount based on cart items
12	19/12/2024	Designed **Login** and **Registration** pages, implemented **useNavigate** for redirection, and added **toast notifications** for user feedback on actions.
13	20/12/2024	Made necessary configuration in the project.

Sign of external guide
Guide:

With company seal :
Name:
Professor
Designation:
Name of the Company:

Internal

Name:
Assistant

BCA-DDU

04/12/24: Introduction to Basics of ReactJS

Task Accomplished:

- Learned about the core concepts of ReactJS: components, state, props, and lifecycle methods.
- Created simple components to render dynamic content.
- Explored component reusability and modularity to build scalable applications.
- Understood the working of React's **virtual DOM** for optimized performance.

```jsx
import React, { useState } from 'react';

const WelcomeMessage = () => {
  const [message, setMessage] = useState("Welcome to React!");

  return (
    <div>
      <h1>{message}</h1>
      <button onClick={() => setMessage("React is Awesome!")}>Change Message</button
    </div>
  );
}

export default WelcomeMessage;
```

05/12/24: Introduction to JavaScript and Its Usage in ReactJS

- **Task Accomplished:**
  I revised core **JavaScript** concepts, including closures, promises, asynchronous programming, and event handling, which are essential for ReactJS development. These concepts form the backbone of how ReactJS handles asynchronous operations, such as API calls, user interactions, and state changes. I explored how JavaScript features, like **async/await**, can be integrated with ReactJS for smoother data handling, which ensures that the app doesn't crash while waiting for data from external APIs. This day was critical for deepening my understanding of JavaScript's role in modern web applications and how it integrates with ReactJS.

```jsx
const FetchDataExample = () => {
  const [data, setData] = useState([]);

  const fetchData = async () => {
    const response = await fetch('https://api.example.com/data');
    const result = await response.json();
    setData(result);
  };

  useEffect(() => {
    fetchData();
  }, []);

  return (
    <div>
      <h2>Fetched Data:</h2>
      <ul>
        {data.map(item => <li key={item.id}>{item.name}</li>)}
      </ul>
    </div>
  );
```

06/12/24: CSS Fundamentals, Flexbox, Introduction to Responsive Design & Layout

- **Task Accomplished:**
  I studied key **CSS fundamentals** such as the **box model**, positioning, and CSS Grid. This was followed by an in-depth focus on **Flexbox**, a powerful layout tool that helps in building flexible, responsive layouts. I learned how Flexbox simplifies the creation of complex layouts that adjust seamlessly across various screen sizes. This session included practical exercises to design layouts using Flexbox, allowing me to create mobile-first, responsive web designs. Additionally, I explored the importance of responsive design, ensuring that applications built with ReactJS are optimized for all devices.

```jsx
import React from 'react';
import './App.css';

const App = () => {
  return (
    <div className="container">
      <div className="item">Item 1</div>
      <div className="item">Item 2</div>
      <div className="item">Item 3</div>
    </div>
  );
}

export default App;
```

```css
.container {
  display: flex;
  justify-content: space-around;
  flex-wrap: wrap;
}

.item {
  width: 100px;
  height: 100px;
  background-color: lightblue;
  margin: 10px;
}
```

09/12/24: MVC in ReactJS and Working with APIs (Fetch & Axios)

- **Task Accomplished:**
  On this day, I was introduced to the **Model-View-Controller (MVC)** architecture and how it can be applied in ReactJS applications. I explored the interaction between the **Model** (which handles data), the **View** (which displays the UI), and the **Controller** (which manages logic). I also learned how to integrate ReactJS with backend APIs using two popular methods: **Fetch API** and **Axios**. These methods are used to send and retrieve data from a server, allowing React components to update dynamically based on API responses. Understanding this integration helped me create data-driven applications, where the frontend and backend interact seamlessly.

```jsx
import React, { useEffect, useState } from 'react';
import axios from 'axios';

const AxiosExample = () => {
  const [posts, setPosts] = useState([]);

  useEffect(() => {
    axios.get('https://jsonplaceholder.typicode.com/posts')
      .then(response => {
        setPosts(response.data);
      })
      .catch(error => console.error('Error fetching data:', error));
  }, []);

  return (
    <div>
      <h2>Posts:</h2>
      <ul>
        {posts.map(post => <li key={post.id}>{post.title}</li>)}
      </ul>
    </div>
  );
}

export default AxiosExample;
```

10/12/24: Revisiting MVC Architecture, Integration in Full-Stack Applications

- **Task Accomplished:**
  This session involved revisiting the **MVC architecture**, with a focus on integrating it into **full-stack applications**. I reviewed the role of **controllers** in managing data flow between models and views. Additionally, I explored how controllers in ReactJS can be used to handle user inputs and interact with the backend. This session also covered the practical aspects of building **full-stack applications** by integrating **ReactJS** with backend services. I focused on understanding the flow of data in these applications, which involves the interaction between frontend React components and backend APIs for tasks like data retrieval and user authentication.

```jsx
import React, { useEffect, useState } from 'react';

const App = () => {
  const [data, setData] = useState([]);

  useEffect(() => {
    // Controller fetches data (Model)
    fetch('https://api.example.com/items')
      .then(res => res.json())
      .then(data => setData(data));
  }, []);

  return (
    <div>
      <h1>Items</h1>
      <ul>
        {data.map(item => <li key={item.id}>{item.name}</li>)}
      </ul>
    </div>
  );
}

export default App;
```

11/12/24: Handling Data for Controllers and Data Passing Techniques

- **Task Accomplished:**
  I learned how **controllers** in the MVC architecture interact with **models** to process client requests and handle data. In this session, I explored different techniques for **passing data** between React components, backend controllers, and the database. This included understanding how to pass props between React components, how to fetch data from APIs, and how controllers receive and manage data requests from the frontend. Additionally, I gained hands-on experience in managing data flow in **full-stack applications**, understanding how data is passed through different layers of the application, and how this ensures a smooth user experience.

```jsx
import React, { useState } from 'react';

const TodoApp = () => {
  const [todos, setTodos] = useState([]);
  const [newTodo, setNewTodo] = useState("");

  const addTodo = () => {
    setTodos([...todos, newTodo]);
    setNewTodo("");
  };

  return (
    <div>
      <h1>Todo List</h1>
      <input
        type="text"
        value={newTodo}
        onChange={(e) => setNewTodo(e.target.value)}
      />
      <button onClick={addTodo}>Add Todo</button>
      <ul>
        {todos.map((todo, index) => <li key={index}>{todo}</li>)}
      </ul>
    </div>
  );
};
```

12/12/24: Understanding MVC Basics and Integration in ReactJS

- **Task Accomplished:**
  This day was dedicated to reinforcing my understanding of the **MVC architecture** and its integration with **ReactJS**. I reviewed how React components can represent the **View** in the MVC pattern, how **controllers** manage the business logic, and how **models** interact with the data. I applied these principles by building small projects where I implemented the **MVC** design pattern in ReactJS applications. This hands-on approach helped me see the practical application of MVC in real-world scenarios, enhancing my ability to build structured, maintainable applications.

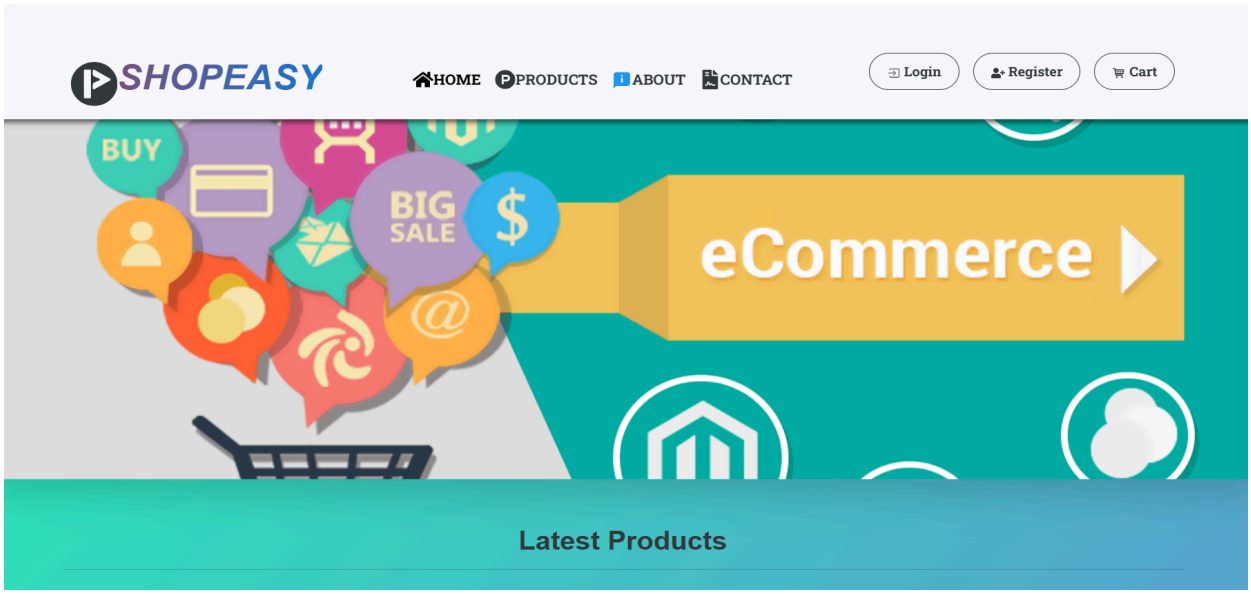
13/12/24: Final Review of MVC Concepts in ReactJS and Full-Stack Application Design

- **Task Accomplished:**
  On the final day of this reporting period, I conducted a **final review** of all the MVC-related concepts that I had learned throughout the week. I worked on integrating all components of a **full-stack application**—Model, View, and Controller—into a single cohesive system. This review helped me consolidate my knowledge of **ReactJS**, the role of **controllers** in managing backend logic, and how **React components** interact with the backend to process data. By the end of this session, I had a clearer understanding of how full-stack applications are structured and the necessary steps to integrate all layers into a working application.

```jsx
import React, { useState, useEffect } from 'react';

const App = () => {
  const [items, setItems] = useState([]);

  useEffect(() => {
    fetch('/api/items')  // Controller API call
      .then(res => res.json())
      .then(data => setItems(data));
  }, []);

  return (
    <div>
      <h1>Items List</h1>
      <ul>
        {items.map(item => <li key={item.id}>{item.name}</li>)}
      </ul>
    </div>
  );
};
```

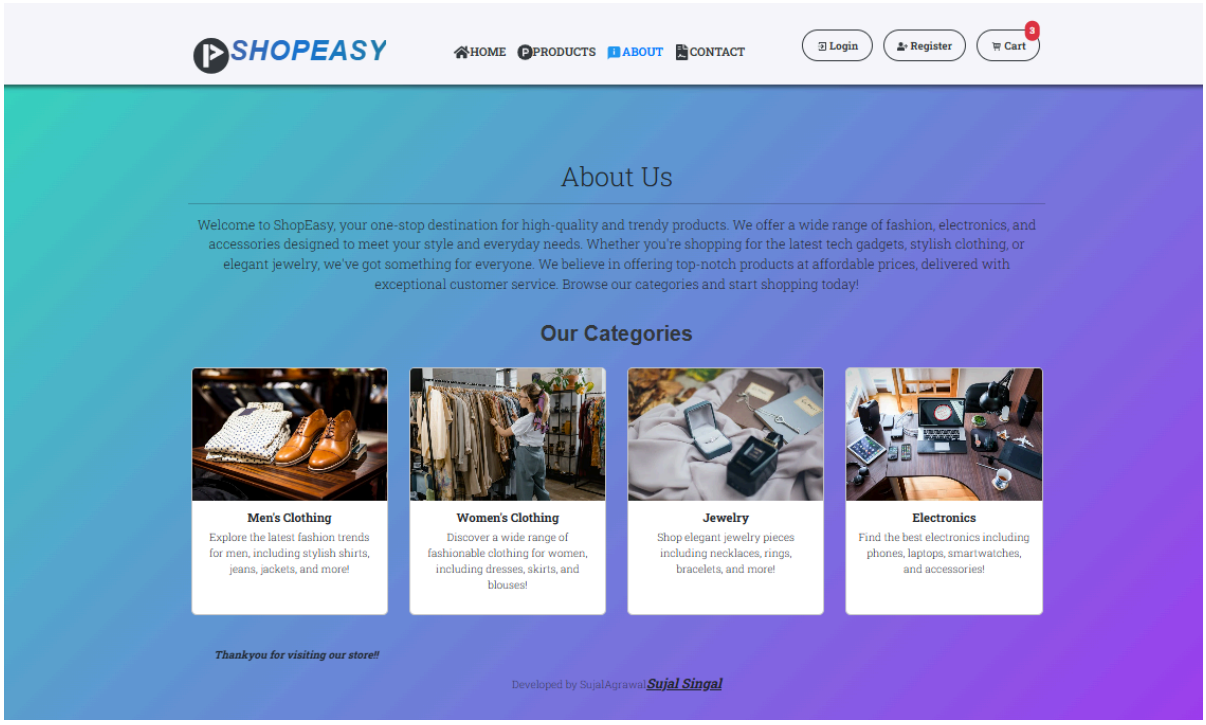
16/12/24 - Initial Setup and Layout Design

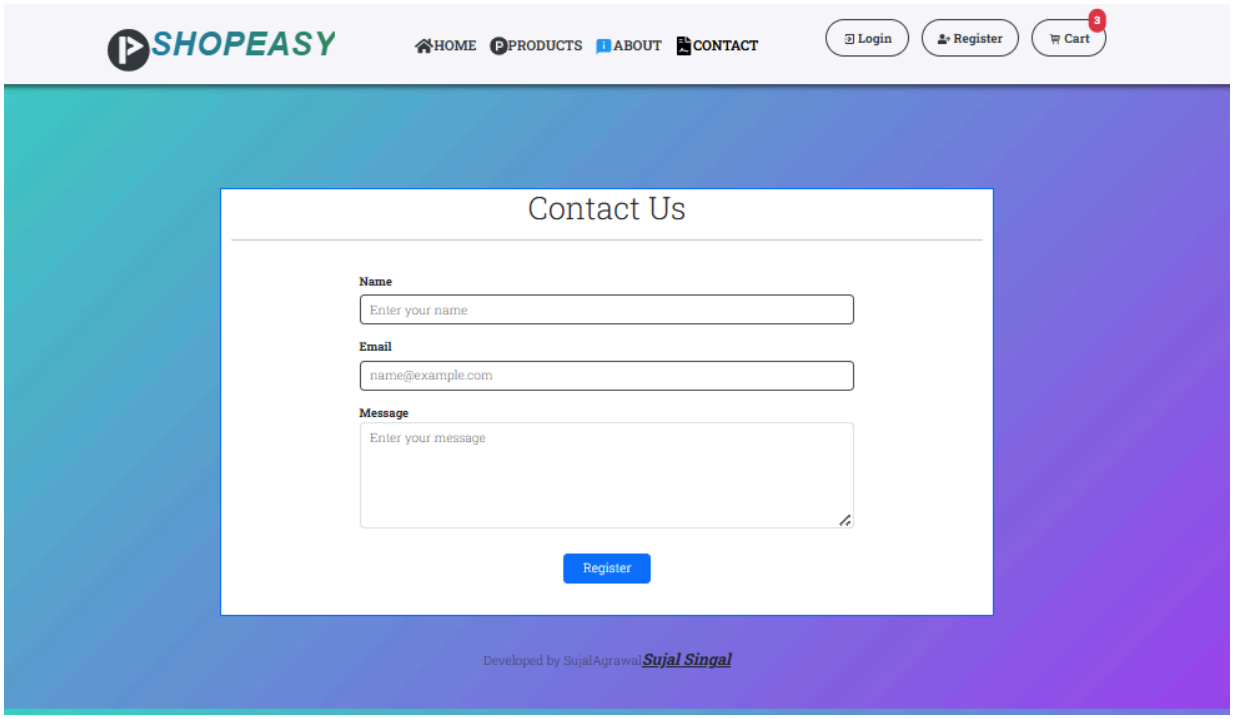
- Initialized React app and installed necessary packages.
- Designed the **navbar** for the e-commerce website, ensuring it is intuitive and user-friendly.
- Developed navigation paths using **react-router-dom**.
    - Utilized **NavLink**, **Link**, **BrowserRouter**, **Routes**, and **Route** to set up page redirection.
- Designed basic layout for the **homepage**, including:
    - Carousel slider (Bootstrap 5.3.3) for featured products.
    - Card layout for showcasing products.
- Designed and implemented **footer layout** with a **marquee effect** for scrolling text.
- Configured card layouts using **Bootstrap v5.3.3** for responsive and stylish product display.



17/12/24- Page Design and Data Fetching

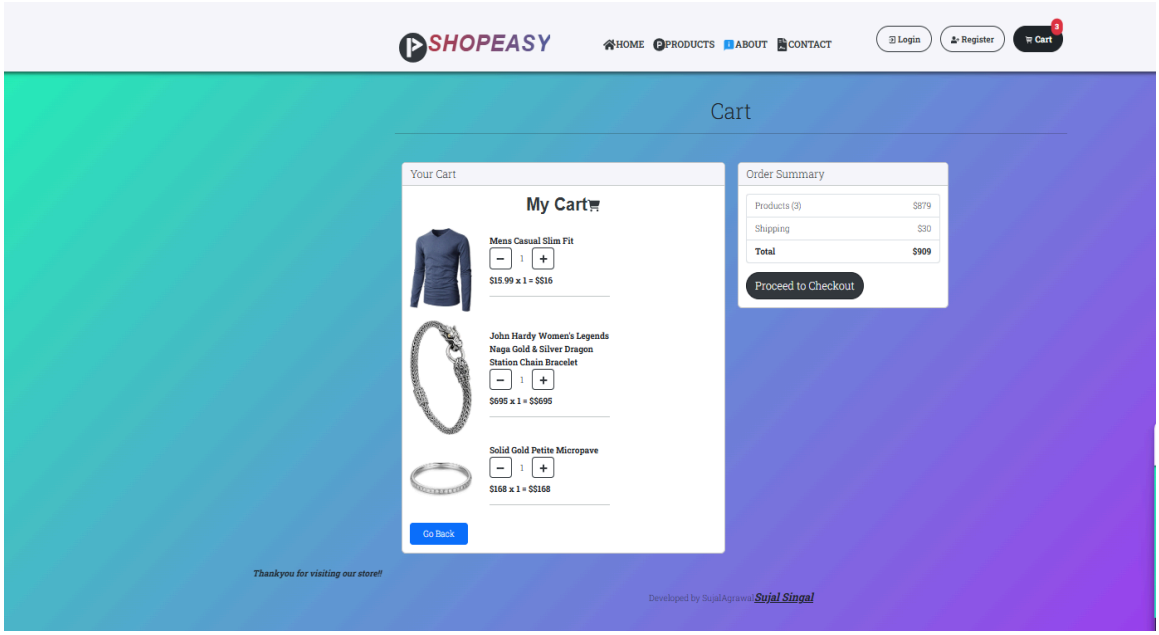
- Designed static pages:
  - **About Us** page.
  - **Contact Us** page.
  - **Products** page.
- Integrated **fake API** to simulate product data:
  - Used **fetch** method to retrieve product data asynchronously.
  - Implemented **useEffect** hook to fetch data based on component dependencies.
  - Applied **async/await** to handle asynchronous operations and prevent crashes from slow data fetching.
- Added product **category filter** functionality:
  - Utilized JavaScript's **filter()** method to dynamically display products based on selected category.
- Enhanced **card hover effects**:
  - Applied **shadow effects** and styled **buttons** on hover.
  - Implemented **clickable buttons** to handle user interactions.

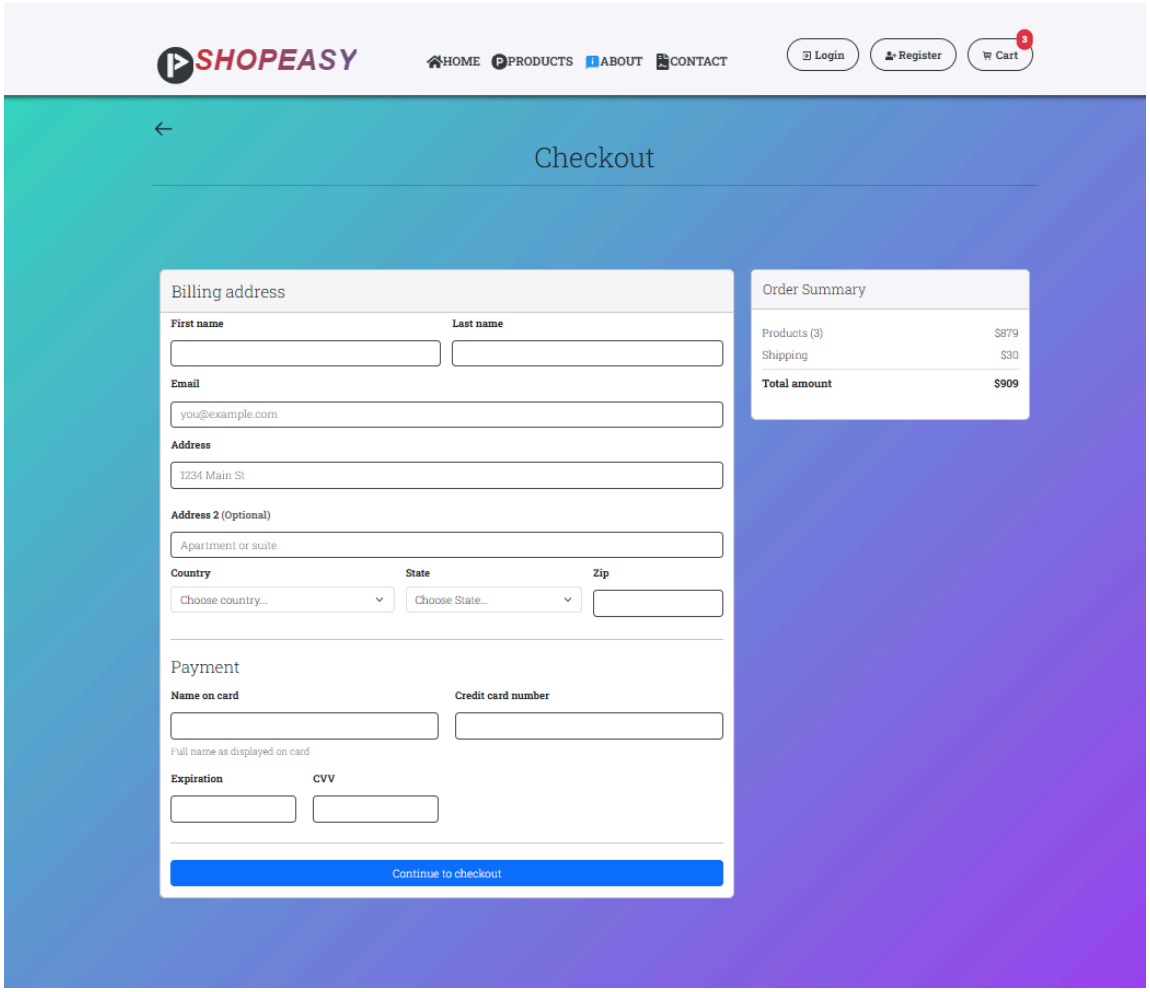




18/12/24 - Redux Integration and Cart Functionality

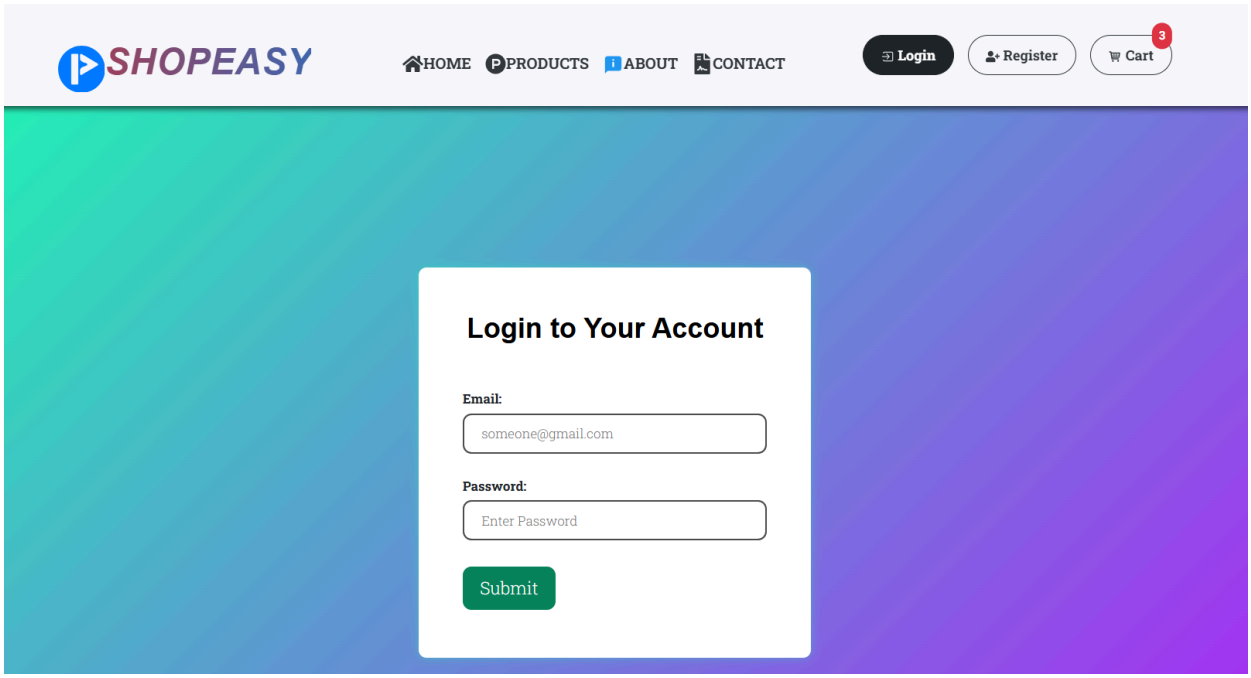
- Set up **Redux** store, actions, and reducers to manage cart events:
    - Defined **addToCart** and **delCart** actions.
    - Used **combineReducers** for better state management in the root reducer.
- Used **dispatch** to trigger actions for adding/removing products from the cart.
- Utilized **useSelector** hook to update the global state based on user interactions.
- Developed dynamic cart functionality:
    - Updated the **cart** button in the **navbar** to reflect the items in the cart.
    - Implemented a calculation for the **total amount** in the cart, which updates in real-time based on item additions/removals.

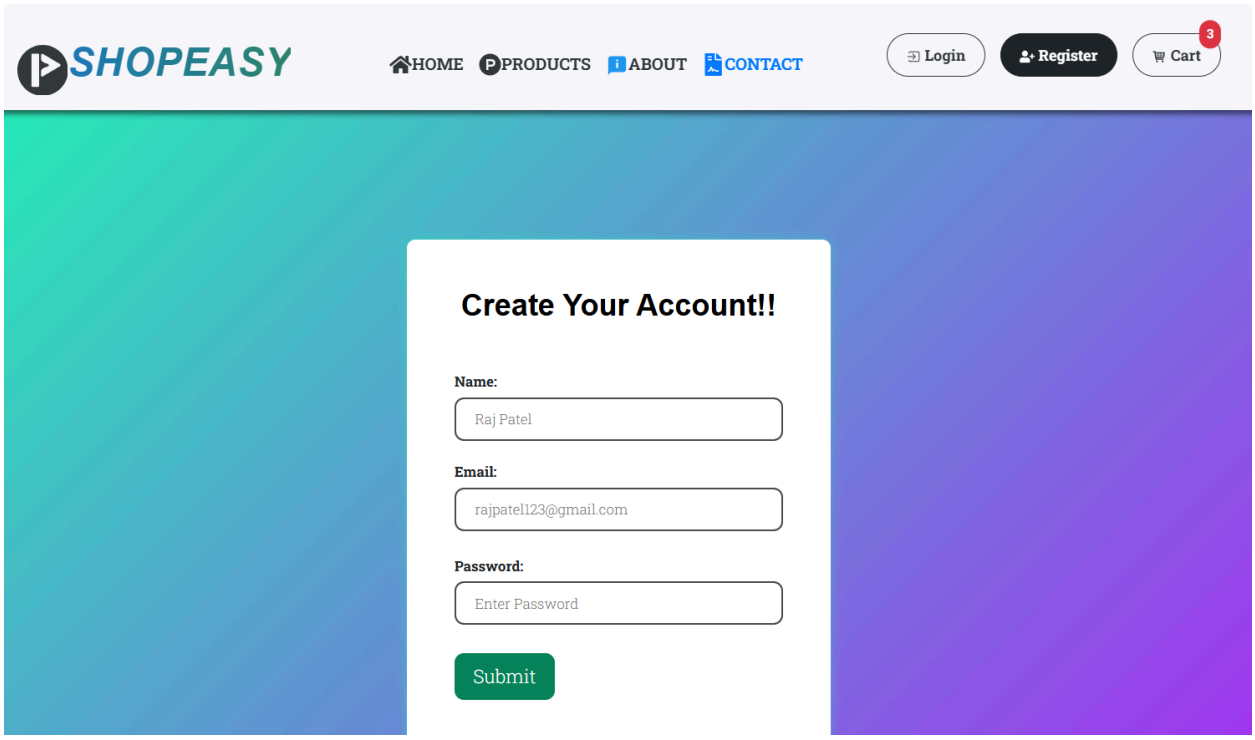




19/12/24 - User Registration and Final Touches

- Designed **Registration Page** and **Login Page** with proper layout and user input validation.
- Integrated **useNavigate** hook to redirect users to appropriate pages after successful login or registration.
- Used **toast notifications** to display user-friendly messages for:
    - Successful login.
    - Addition/removal of items in the cart.
- Made minor design adjustments and improvements across various pages for better UX/UI.


SHOPEASY
HOME
PRODUCTS
ABOUT
CONTACT
Login
Register
Cart
3
Login to Your Account
Email:
someone@gmail.com
Password:
Enter Password
Submit


SHOPEASY
HOME
PRODUCTS
ABOUT
CONTACT
Login
Register
Cart
3
Create Your Account!!
Name:
Raj Patel
Email:
rajpatel123@gmail.com
Password:
Enter Password
Submit

20/12/24

Task Accomplished

Route Handling and Navigation: Configured React Router to ensure smooth navigation between pages like the homepage, about us, products, and contact pages.
State Management with Redux: Final adjustments were made to the Redux store to handle state updates efficiently. Ensured that actions like adding/removing items from the cart were functioning correctly, and that the cart data was persisted globally across different components.
UI Enhancements: A few final tweaks to the UI were made, such as optimizing the layout, ensuring responsiveness on different devices, and adding hover effects to improve user interaction.
Error Handling & Feedback: Implemented toast notifications for better user feedback, showing messages for successful actions like login or adding/removing products from the cart.
Code Clean-Up: Refined the codebase by removing unused variables and functions, ensuring the application was optimized and free from errors.

Tasks Accomplished

1. **Initial Setup:**
    - React app initialization and package installation.
    - Basic layout design with carousel and cards (Bootstrap).
2. **Routing & Navigation:**
    - Setup navigation paths using **react-router-dom**.
    - Designed and configured navbar with links to various pages.
3. **API Integration:**
    - Integrated **fake API** for fetching product data.
    - Applied asynchronous fetching with **async/await** to ensure smooth data retrieval.
4. **Filter Functionality:**
    - Added category-based filtering for products on the homepage.
5. **Cart Management (Redux):**
    - Integrated **Redux** for managing the cart state globally.
    - Designed actions and reducers to handle cart operations (add/remove items).
    - Calculated total amount dynamically based on cart contents.
6. **User Authentication:**
    - Designed **Login** and **Registration** pages.
    - Implemented **useNavigate** for redirection and **toast messages** for user feedback.
