SRI KRISHNA COLLEGE OF TECHNOLOGY
An Autonomous Institution | Accredited by NAAC with 'A' Grade
Affiliated to Anna University | Approved by AICTE
KOVAIPUDUR, COIMBATORE 641042

CRUSH

A PROJECT REPORT

Submitted by

SHAMEER AHAMED S 727822TUEC211

SUDHARSHAN V 727822TUEC211

SUKUMAR M 727822TUEC211

in partial fulfillment for the award of the

degreeof

BACHELOR OF

TECHNOLOGY

IN

DEPARTMENT OF ELECTRONIC AND COMMUNICATION ENGINEERING

AUGUST 2024

SRI KRISHNA COLLEGE OF TECHNOLOGY
An Autonomous Institution | Accredited by NAAC with 'A' Grade
Affiliated to Anna University | Approved by AICTE
KOVAIPUDUR, COIMBATORE 641042

BONAFIDE CERTIFICATE

Certified that this project report **"CRUSH"** is the bonafide work of **SHAMEER AHAMED S, SUKUMAR M, SUDHARSHAN V** who carried out the project work under my supervision.

SIGNATURE

Mr. R Naveen Kumar

Supervisor

Associate Professor,
Department of Electronic and
Communication engineering
Sri Krishna College of Technology,
Coimbatore-641042

SIGNATURE

Dr.k.Muthulakshmi M.E.,Ph.D

HEAD OF THE DEPARTMENT

Associate Professor,
Department of electronic and
communication engineering,
Sri Krishna College of Technology,
coimbatore-641042

Certified that the candidates were examined by us in the Project Work Viva Voce examination held on_____at Sri Krishna College of Technology, Kovaipudur,Coimbatore -641042

INTERNAL EXAMINER

EXTERNAL EXAMINER

ACKNOWLEDGEMENT

ACKNOWLEDGEMENT

First and foremost, we thank the **Almighty** for being our light and for showering his gracious blessings throughout the course of this project.

We are grateful to our beloved Principal **Dr.M.G. Sumithra M.E., Ph.D**. for her tireless and relentless support.

We extend our sincere thanks to our Head of the Department **Dr.Muthu Lakshimi ,M.E., Ph.D.** for her encouragement and inspiration.

We are greatly indebted to our Supervisor **Mr. R Naveen kumar** for his valuable guidance and suggestions in all aspects that aided us to ameliorate our skills.

We are thankful to all those who have directly and indirectly extended their help to us in completing this project work successfully.

ABSTRACT

ABSTRACT

The proposed e-commerce website is a comprehensive digital platform designed to provide users with a seamless online shopping experience. The website will cater to a broad range of products, including electronics, fashion, home appliances, and more, making it a one-stop-shop for consumers. The platform will feature an intuitive user interface, ensuring ease of navigation and quick access to desired products.

Key functionalities of the website include personalized user profiles, advanced search and filtering options, and a secure, multi-step checkout process. To enhance user engagement, the website will incorporate AI-driven product recommendations based on browsing history, preferences, and past purchases. Users will also have access to detailed product descriptions, high-quality images, customer reviews, and ratings to make informed purchasing decisions.

The website will support various payment methods, including credit/debit cards, digital wallets, and cash on delivery, providing flexibility to the customers. Additionally, a robust backend system will manage inventory, order processing, and real-time shipment tracking, ensuring efficient fulfillment and delivery.

The platform will be designed with a responsive layout, ensuring compatibility with both desktop and mobile devices. It will also include features like wishlist management, promotional offers, and loyalty programs to retain customers and encourage repeat purchases.

TABLE OF CONTENTS

TABLE OF CONTENT

CHAPTER	TITLE	PAGE NO
	ABSTRACT	**6**
	LIST OF FIGURES	**10**
	LIST OF ABBREVIATIONS	**12**
1	INTRODUCTION	14
	1.1 PROBLEM STATEMENT	15
	1.2 OVERVIEW	16
	1.3 OBJECTIVE	
2	LITERATURE SURVEY	18
	2.1 RELATIVE WORK	18
3	SYSTEM SPECIFICATIONS	20
	3.1 VS CODE	20
	3.2 REACT	20
	3.3 ROUTERS IN REACT	21
	3.4 LOCAL STORAGE	21
4	PROPOSED SYSTEM	24
	4.1 PROPOSED SYSTEM	24
	4.2 ADVANTAGES	26
5	METHODOLOGIES	28
	5.1 LOGIN PAGE	29
	5.2 SIGN UP PAGE	30
	5.3 HOME PAGE	31
	5.4 5.7 ADD TO CART PAGE	32
6	IMPLEMENTATION AND RESULT	34
	6.1 LOGIN PAGE	34
	6.2 SIGNUP PAGE	40
	6.3 HOME PAGE	48
	6.4 DEALS PAGE	50

	6.5 CATEGORY PAGE	55
	6.6 CART PAGE	57
	6.7 BACKEND	59
7	CONCLUSION AND FUTURE SCOPE	63
	7.1 CONCLUSION	63
	7.2 FUTURE SCOPE	63
8	REFERENCES	66

LIST OF FIGURES

LIST OF FIGURES

Figure No	TITLE	Page No
5.1	Process flow diagram	28
5.2	Login page flowchart	29
5.3	Signup page flowchart	30
5.4	Home page flowchart	31
5.5	Cart flowchart	32
6.1	Login page	34
6.2	Signup page	40
6.3	Home page	48
6.4	Deals	51
6.5	Category page	55
6.6	Cart page	58

LIST OF ABBREVIATIONS

LIST OF ABBREVIATIONS

ABBREVIATIONS	ACRONYMS
HTML	HYPERTEXT MARKUP LANGUAGE
CSS	CASCADING STYLESHEET
JS	JAVASCRIPT

INTRODUCTION

CHAPTER 1

INTRODUCTION

Crush is a specialized e-commerce platform dedicated to offering a curated selection of dresses for both men and women. With a focus on fashion-forward designs, Crush provides a variety of styles that cater to different tastes and occasions, from casual wear to formal attire.

The platform is designed to make shopping for the perfect dress effortless, with detailed product descriptions, size guides, and high-quality images to ensure customers find exactly what they're looking for.

Crush stands out by offering a unique range of dresses from top designers and emerging brands, emphasizing quality and style. The platform is committed to providing an inclusive shopping experience, with a wide range of sizes and styles that appeal to diverse body types and fashion preferences. In addition to its product offerings, Crush ensures a seamless shopping experience with user-friendly navigation, secure payment options, and efficient delivery service

1.1 PROBLEM STATEMENT

Crush aims to address the challenge of providing a specialized online shopping experience for men and women seeking high-quality, stylish dresses. In the crowded e-commerce landscape, many platforms offer a wide array of clothing options, but few focus exclusively on dresses, leading to a diluted shopping experience for those specifically in search of this wardrobe staple. Customers often face difficulties in finding well-curated collections, size inclusivity, and reliable product information, which can result in frustration and a lack of confidence in their purchase decisions.

Crush seeks to solve these issues by creating a dedicated platform that exclusively offers a diverse and carefully curated selection of dresses. The platform's goal is to provide a seamless, user-friendly shopping experience that caters to the specific needs of dress shoppers, ensuring they can easily find stylish, well-fitting dresses with confidence and convenience.

1.2 OVERVIEW

Crush is a niche e-commerce platform dedicated solely to offering a wide range of dresses for men and women. Unlike traditional online retailers that offer a broad spectrum of clothing, Crush focuses exclusively on dresses, ensuring that customers have access to a carefully curated collection that meets diverse fashion needs and preferences. Whether you're looking for casual daywear, elegant evening dresses, or something in between, Crush provides a variety of styles, sizes, and designs to suit every occasion.

The platform is designed with the modern shopper in mind, featuring an intuitive interface that makes browsing, selecting, and purchasing dresses straightforward and enjoyable. Crush places a strong emphasis on quality, partnering with both established brands and emerging designers to bring customers high-quality products that stand out in terms of both style and craftsmanship. Additionally, Crush is committed to inclusivity, offering a wide range of sizes to cater to all body types, and ensuring that every customer can find something that makes them feel confident and stylish.

1.3 OBJECTIVE

The primary objective of Crush is to become the leading online destination for men and women seeking high-quality, stylish dresses. Crush aims to offer a curated selection of dresses that cater to diverse tastes, body types, and occasions, providing customers with a seamless and personalized shopping experience. By focusing exclusively on dresses, Crush seeks to simplify the shopping process, making it easier for customers to find the perfect dress with confidence, while also promoting inclusivity, sustainability, and exceptional customer service.

LITERATURE SURVEY

CHAPTER 2

LITERATURE SURVEY

2.1 RELATED WORKS

In the e-commerce landscape, several platforms offer clothing and fashion items, but few are dedicated exclusively to dresses for both men and women. Some notable competitors in the fashion retail space include platforms like ASOS, which provides a wide range of clothing items including dresses, and H&M, which offers affordable fashion with a focus on trendy styles. These platforms, however, cater to a broad spectrum of clothing categories, often making it challenging for customers specifically looking for dresses to navigate their extensive catalogs.

Other specialized platforms like Rent the Runway focus on dress rentals, allowing customers to rent high-end dresses for specific occasions, while companies like Reformation emphasize sustainable fashion, including dresses made from eco-friendly materials. However, these platforms either limit their offerings to specific categories or have a different business model, such as rental or sustainable fashion, rather than a comprehensive focus on dress retail.

Crush differentiates itself by concentrating exclusively on the dress category, offering a well-curated selection that spans various styles, occasions, and sizes. This focused approach not only simplifies the shopping experience for customers but also allows Crush to position itself as a go-to destination for anyone in search of the perfect dress. By honing in on this niche, Crush is able to provide a more tailored shopping experience, setting itself apart from more generalized fashion retailers.

SYSTEM SPECIFICATION

CHAPTER 3
SYSTEM SPECIFICATION

In this chapter, we are gonna see the softwares that we have used to build the website. This chapter gives you a small description about the softwares used in the project.

3.1 VS CODE

Visual Studio Code is a source code editor developed by Microsoft for Windows, Linux, and macOS. It includes support for debugging, embedded Git control, syntax highlighting, intelligent code completion, snippets, and code refactoring. It is also customizable, so users can change the editor's theme, keyboard shortcuts, and preferences.

VS Code is an excellent code editor for React projects. It is lightweight, customizable, and has a wide range of features that make it ideal for React development. It has built-in support for JavaScript, JSX and enables developers to quickly move between files and view detailed type definitions. It also has a built-in terminal for running tasks.

Additionally, VS Code has an extensive library of extensions that allow developers to quickly add features like code snippets, debugging tools, and linting support to their projects.

3.2.REACT

React is a JavaScript library created by Facebook for building user interfaces. It is a component-based, declarative, and highly efficient library that is used to develop interactive UIs (user interfaces) for single page web applications. React uses a virtual DOM (Document Object Model) that makes it faster and easier to manipulate the DOM

elements. It also provides declarative components that allow developers to write code that is easy to read and maintain. React also offers an extensive library of tools and components that make it easier to develop complex user interfaces.

3.3 ROUTERS IN REACT

Routers are important components in React applications. They provide the ability to navigate between different views or components of the application. React Router is the most popular library to handle routing in React applications. It provides the ability to define routes, set up links, and render components based on the current route. It also provides features like data fetching, code-splitting, and server-side rendering.

3.4 LOCAL STORAGE

Local storage is a type of web storage for storing data on the client side of a web browser. It allows websites to store data on a user's computer, which can then be accessed by the website again when the user returns.

Local storage is a more secure alternative to cookies because it allows websites to store data without having to send it back and forth with each request. Local storage is a key-value pair storage mechanism, meaning it stores data in the form of a key and corresponding value.

It is similar to a database table in that it stores data in columns and rows, except that local storage stores the data in the browser rather than in a database. Local storage is often used to store user information such as preferences and settings, or to store data that is not meant to be shared with other websites. It is also used to cache data to improve the performance of a website. Local storage is supported by all modern web browsers, including Chrome,

Firefox, Safari, and Edge. It is accessible through the browser's JavaScript API. Local storage is a powerful tool for websites to store data on the client side. It is secure, efficient, and can be used to store data that does not need to be shared with other websites.

Local Storage is a great way to improve the performance of a website by caching data. Local storage in web browsers allows website data to be stored locally on the user's computer. It is a way of persistently storing data on the client side, which is not sent to the server with each request. This allows users to store data such as preferences, login information, and form data without needing to send it to a server. It is typically stored in a browser's cookie file, but it can also be stored in other locations such as HTML5 Local Storage. The data stored in local storage is persistent and can be accessedby the website even if the user closes the browser or navigates to another page. It is a great way for websites to store user-specific data, as it is secure, reliable, and fast. It is also a great way for developers to store data that does not need to be sent to the server with each request.

PROPOSED SYSTEM

CHAPTER 4

PROPOSED SYSTEM

This chapter gives a small description about the proposed idea behind the development of our website

4.1 PROPOSED SYSTEM

The proposed system for Crush is a robust, user-friendly e-commerce platform designed exclusively for the sale of dresses for men and women. The system will be developed with a focus on providing a seamless, efficient, and personalized shopping experience, ensuring that customers can easily find and purchase dresses that suit their style and needs. The key components of the proposed system include:

1. **User Interface (UI) and User Experience (UX) Design:**
    - The platform will feature an intuitive, aesthetically pleasing interface with easy navigation to enhance the shopping experience.
    - A responsive design will ensure optimal performance across devices, including desktops and mobile phones.
2. **Product Catalog and Filtering System:**
    - The product catalog will be organized by categories such as style, occasion, color, size, and price range, allowing users to quickly filter and find dresses that meet their preferences.
    - High-quality images, detailed product descriptions, and customer reviews will be provided for each item to aid in decision-making.
3. **Personalization and Recommendation Engine:**
    - The system will include a recommendation engine that uses AI and machine learning to suggest dresses based on the user's browsing history, preferences, and purchase behavior.
    - Users will have the option to create profiles, save favorite items, and receive

personalized recommendations and alerts about new arrivals and discounts.

4. **Shopping Cart and Checkout Process:**
    - The shopping cart will be user-friendly, allowing customers to easily add or remove items, apply discount codes, and view the total cost before checkout.
    - A streamlined, secure checkout process will support multiple payment options, including credit/debit cards, PayPal, and digital wallets.
5. **Inventory Management and Order Fulfillment:**
    - The system will include an inventory management module to track stock levels in real-time, ensuring accurate availability information for customers.
    - Automated order processing and integration with shipping partners will enable efficient order fulfillment, providing customers with tracking information and delivery updates.
6. **Customer Support and Feedback Mechanism:**
    - An integrated customer support system will offer assistance through chatbots, email, and phone support, addressing customer queries and issues promptly.
    - A feedback mechanism will be in place to collect customer reviews and suggestions, allowing Crush to continuously improve its offerings and services.
7. **Marketing and SEO Integration:**
    - The platform will be optimized for search engines (SEO) to increase visibility and attract organic traffic.
    - Integrated marketing tools will enable targeted email campaigns, social media promotions, and special offers to engage customers and drive sales.
8. **Sustainability and Ethical Sourcing:**
    - Crush will partner with suppliers who adhere to ethical practices and offer sustainably sourced materials, providing customers with eco-friendly dress options.
    - Information about the sustainability and ethical standards of products will be highlighted on the platform to promote conscious consumerism.

The proposed system will leverage modern technologies and best practices in

4.2 ADVANTAGES

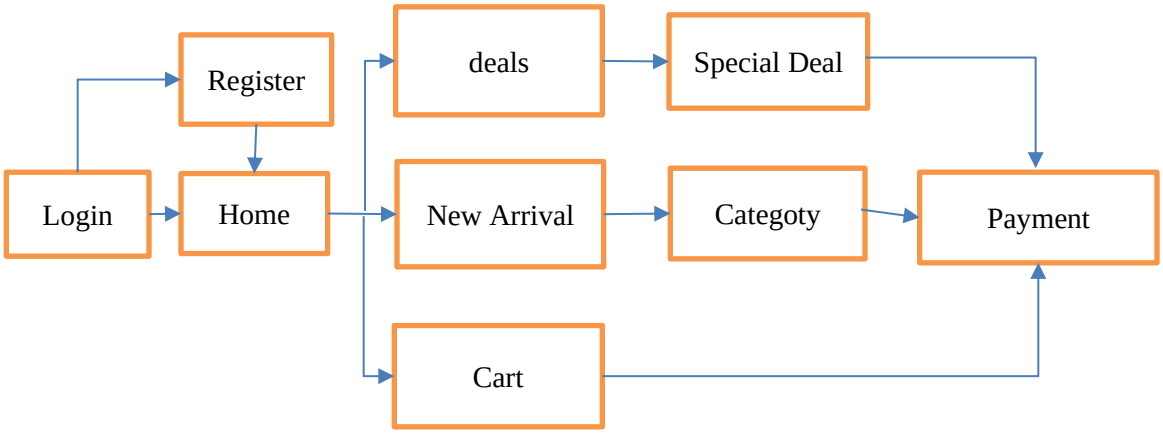
- Customization options: Customers can select specific components and configurations to build their ideal computer.
- Price transparency: Customers can see the cost of each component and the final price of their build, allowing them to make informed decisions.
- Compatibility assurance: The website will typically ensure that the components selected are compatible with each other, reducing the risk of compatibility issues.
- Time-saving: Customers can quickly and easily build their computer online, rather than having to visit multiple stores or websites to find the components they need.
- Quality assurance: The website can offer warranty and quality assurance on all the products and components provided, this can help customers to trust the website.
- Cost savings: Building a computer from individual components can often be less expensive than purchasing a pre-built system.

METHODOLOGIES

CHAPTER 5

METHODOLOGIES

This chapter gives a small description about how our system works.



5.1.LOGIN

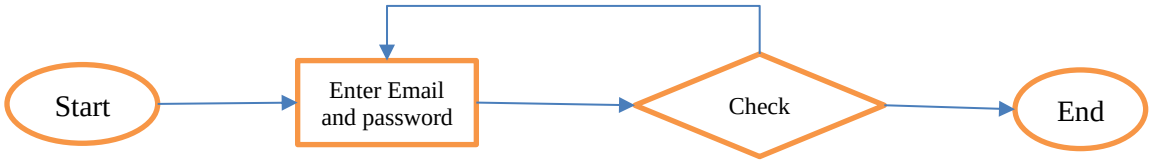


Fig 5.2. LOGIN PAGE FLOWCHART

In this page we will be asking about the username and password of the user. Firstly the website validates the user inputs. It verifies the username and password by checking it with the usernames and passwords stored in the local storage when the user creates an account in the website.

5.2 SIGNUP PAGE

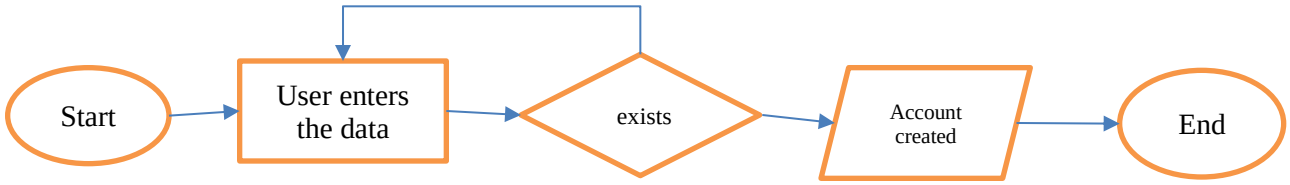


Fig 5.3 SIGNUP PAGE FLOWCHART

This page asks users about the basic details of the user to create an account. This page asks for details like username, password,email id, phone number. After the user enters the details , these details are then validated by our code . If all details are correct then the user is then directed to the login page

5.3. HOME PAGE

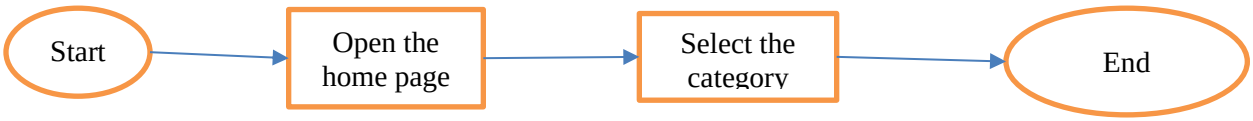


Fig 5.4 HOME PAGE FLOWCHART

This page displays the list of products available for sale. These details have been displayed from the list of products available in the javascript file named data.js. So that ifwe want to add some products to a particular product we can easily do it by adding the details to the home.js file.

5.5 CART PAGE

This page displays the set of products that the user selected by clicking the button “Add to cart” available on the products page. It also displays the products with their names and its price with quantity. This also helps you to increase the quantity of the products you have added using a plus button which is available on the page after every product's name. You can also remove or reduce the quantity by using the minus symbol button which is also available below every product's name. If the cart doesn’t contain anyitem it will display that the cart is empty.

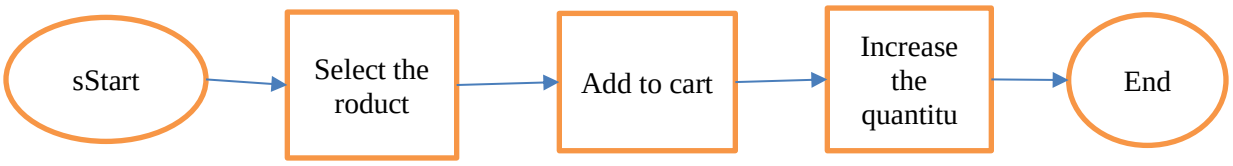


Fig 5.5 CART PAGE FLOWCHART

IMPLEMENTATION AND RESULT

CHAPTER 6
IMPLEMENTATION AND RESULT

This chapter gives a description about the output that we produced by developingthe website of our idea.

6.1 LOGIN

When User enters our website he will be asked about his login details like email idand password. The login details will be verified with the details given while the user creates an account.

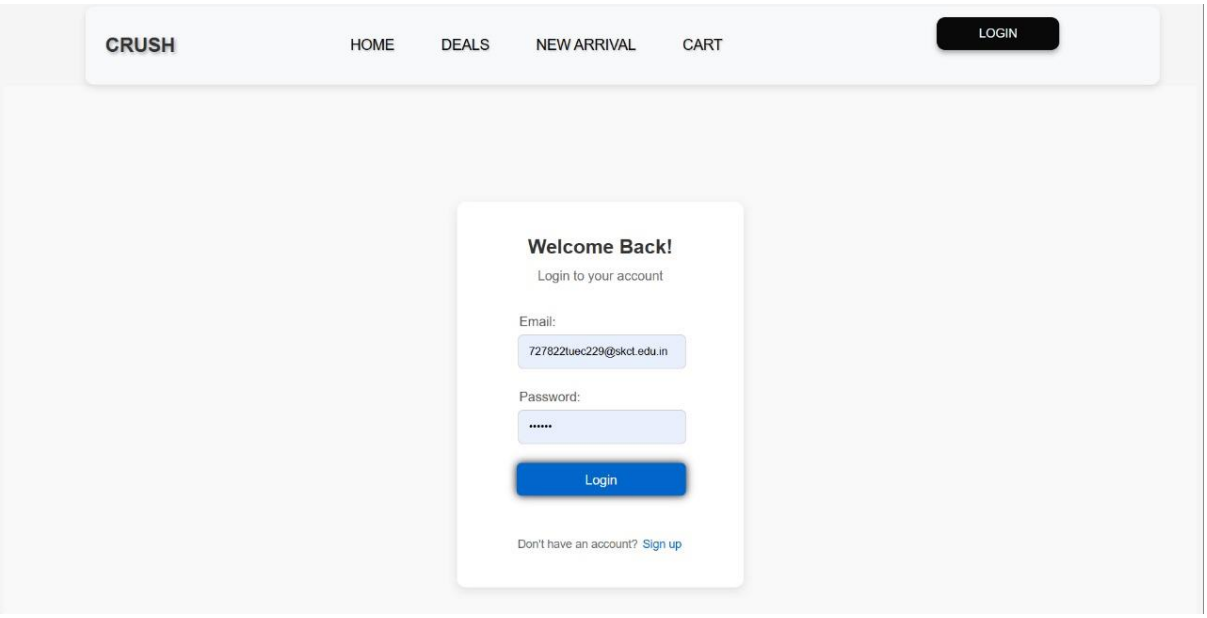


Fig 6.1 LOGIN PAGE

6.1.1 CODING

```
import { useState } from 'react';
import { Link, useNavigate } from 'react-router-dom';

const LoginPage = () => {
```

```
  const [email, setEmail] = useState('');
  const [password, setPassword] = useState('');
  const [inputFocus, setInputFocus] = useState({ email: false, password:
false });
  const [buttonHover, setButtonHover] = useState(false);
  const [error, setError] = useState('');
  const nav = useNavigate();

  const handleLogin = async (e) => {
    e.preventDefault();
    try {
      const response = await fetch('https://retoolapi.dev/5M2qFh/data', {
        method: 'GET',
        headers: {
          'Content-Type': 'application/json',
        },
      });

      const data = await response.json();

      const user = data.find((user) => user.mail === email &&
user.password === password);

      if (user) {

        nav('/');
      } else {
        // Handle errors
        setError('Invalid email or password');
      }
    } catch (err) {
```

```
      setError('An error occurred. Please try again.');
    }
  };

  return (
    <div style={styles.container}>
      <div style={styles.loginBox}>
        <h2 style={styles.header}>Welcome Back!</h2>
        <p style={styles.subHeader}>Login to your account</p>
        {error && <p style={styles.error}>{error}</p>}
        <form onSubmit={handleLogin}>
          <div style={styles.inputGroup}>
            <label style={styles.label} htmlFor="email">Email:</label>
            <input
              style={{ ...styles.input, ...(inputFocus.email &&
styles.inputFocus) }}
              type="email"
              id="email"
              name="email"
              value={email}
              onChange={(e) => setEmail(e.target.value)}
              onFocus={() => setInputFocus({ ...inputFocus, email: true })}
              onBlur={() => setInputFocus({ ...inputFocus, email: false })}
            />
          </div>
          <div style={styles.inputGroup}>
            <label style={styles.label}
htmlFor="password">Password:</label>
            <input
              style={{ ...styles.input, ...(inputFocus.password &&
styles.inputFocus) }}
```

```
            type="password"
            id="password"
            name="password"
            value={password}
            onChange={(e) => setPassword(e.target.value)}
            onFocus={() => setInputFocus({ ...inputFocus, password: true })}
            onBlur={() => setInputFocus({ ...inputFocus, password: false })}
          />
        </div>
        <button
          style={{ ...styles.button, ...(buttonHover && styles.buttonHover)
}}
          type="submit"
          onMouseEnter={() => setButtonHover(true)}
          onMouseLeave={() => setButtonHover(false)}
        >
          Login
        </button>
        <p style={styles.footer}>
          Don't have an account? <Link to="/register"
style={styles.link}>Sign up</Link>
        </p>
      </form>
    </div>
  </div>
 );
};

const styles = {
 container: {
   display: 'flex',
```

```
  justifyContent: 'center',
  alignItems: 'center',
  height: '100vh',
  backgroundColor: '#f9f9f9',
 },
 loginBox: {
  display: 'flex',
  flexDirection: 'column',
  alignItems: 'center',
  padding: '40px',
  backgroundColor: '#ffffff',
  boxShadow: '0 4px 12px rgba(0, 0, 0, 0.1)',
  borderRadius: '10px',
  width: '350px',
 },
 header: {
  marginBottom: '10px',
  color: '#333333',
  fontSize: '24px',
 },
 subHeader: {
  marginBottom: '30px',
  color: '#666666',
  fontSize: '16px',
 },
 error: {
  color: 'red',
  marginBottom: '20px',
 },
 inputGroup: {
  marginBottom: '20px',
```

```
  width: '100%',
  textAlign: 'left',
},
label: {
  display: 'block',
  marginBottom: '5px',
  color: '#555555',
},
input: {
  width: '100%',
  padding: '12px',
  boxSizing: 'border-box',
  border: '1px solid #cccccc',
  borderRadius: '6px',
  outline: 'none',
  transition: 'border-color 0.2s',
},
inputFocus: {
  borderColor: '#0066cc',
},
button: {
  width: '100%',
  padding: '12px',
  backgroundColor: '#0066cc',
  color: '#ffffff',
  border: 'none',
  borderRadius: '6px',
  cursor: 'pointer',
  fontSize: '16px',
  transition: 'background-color 0.2s',
},
```

```
  buttonHover: {
    backgroundColor: '#004d99',
  },
  footer: {
    marginTop: '20px',
    color: '#666666',
    fontSize: '14px',
  },
  link: {
    color: '#0066cc',
    textDecoration: 'none',
  },
};

export default LoginPage;
```

6.2 SIGNUP PAGE

If a user doesn't have an account on the website, User can use a component namedcreate new account available in the login page. When the user clicks on that he will be redirected to the signup page. In sign up he should fill up his email id, password and phone number. These inputs will be validated.

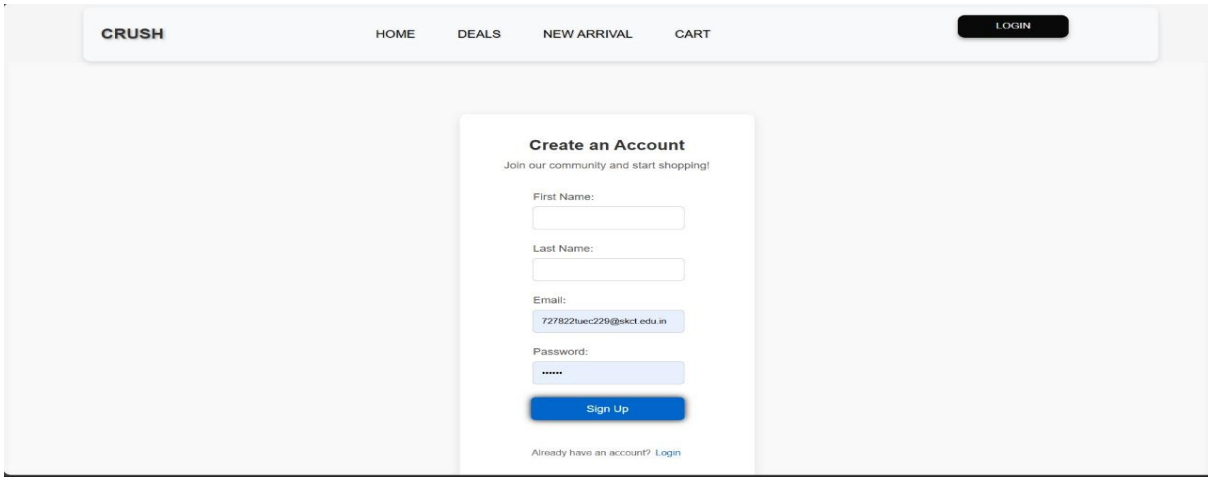


Fig 6.2 SIGNUP PAGE

6.2.1 CODING

```
mport React, { useState } from 'react';
import { Link, useNavigate } from 'react-router-dom';

const Signup = () => {
  const [firstName, setFirstName] = useState('');
  const [lastName, setLastName] = useState('');
  const [email, setEmail] = useState('');
  const [password, setPassword] = useState('');
  const [inputFocus, setInputFocus] = useState('');
  const [buttonHover, setButtonHover] = useState(false);
  const [error, setError] = useState('');
  const navigate = useNavigate();

  const handleRegister = async (e) => {
    e.preventDefault();

    try {
      const response = await fetch('https://retoolapi.dev/5M2qFh/data', {
        method: 'POST',
        headers: {
          'Content-Type': 'application/json',
        },
        body: JSON.stringify({
          firstName,
          lastName,
          email,
          password,
        }),
```

```
      });

      if (response.ok) {
        // Registration successful
        navigate('/countdowntimer');
      } else {
        // Handle errors
        const data = await response.json();
        setError(data.message || 'Registration failed');
      }
    } catch (err) {
      setError('An error occurred. Please try again.');
    }
  };

  return (
    <div style={styles.container}>
      <div style={styles.registrationBox}>
        <h2 style={styles.header}>Create an Account</h2>
        <p style={styles.subHeader}>Join our community and start
shopping!</p>
        {error && <p style={styles.error}>{error}</p>}
        <form onSubmit={handleRegister}>
          <div style={styles.inputGroup}>
            <label style={styles.label} htmlFor="firstName">First
Name:</label>
            <input
              style={{ ...styles.input, ...(inputFocus === 'firstName' &&
styles.inputFocus) }}
              type="text"
              id="firstName"
```

```
            name="firstName"
            value={firstName}
            onChange={(e) => setFirstName(e.target.value)}
            onFocus={() => setInputFocus('firstName')}
            onBlur={() => setInputFocus('')}
          />
        </div>
        <div style={styles.inputGroup}>
          <label style={styles.label} htmlFor="lastName">Last
Name:</label>
          <input
            style={{ ...styles.input, ...(inputFocus === 'lastName' &&
styles.inputFocus) }}
            type="text"
            id="lastName"
            name="lastName"
            value={lastName}
            onChange={(e) => setLastName(e.target.value)}
            onFocus={() => setInputFocus('lastName')}
            onBlur={() => setInputFocus('')}
          />
        </div>
        <div style={styles.inputGroup}>
          <label style={styles.label} htmlFor="email">Email:</label>
          <input
            style={{ ...styles.input, ...(inputFocus === 'email' &&
styles.inputFocus) }}
            type="email"
            id="email"
            name="email"
            value={email}
```

```
            onChange={(e) => setEmail(e.target.value)}
            onFocus={() => setInputFocus('email')}
            onBlur={() => setInputFocus('')}
          />
        </div>
        <div style={styles.inputGroup}>
          <label style={styles.label}
htmlFor="password">Password:</label>
          <input
            style={{ ...styles.input, ...(inputFocus === 'password' &&
styles.inputFocus) }}
            type="password"
            id="password"
            name="password"
            value={password}
            onChange={(e) => setPassword(e.target.value)}
            onFocus={() => setInputFocus('password')}
            onBlur={() => setInputFocus('')}
          />
        </div>
        <button
          style={{ ...styles.button, ...(buttonHover && styles.buttonHover)
}}
          type="submit"
          onMouseEnter={() => setButtonHover(true)}
          onMouseLeave={() => setButtonHover(false)}
        >
          Sign Up
        </button>
        <p style={styles.footer}>
          Already have an account? <Link to="/"
```

```
style={styles.link}>Login</Link>
          </p>
        </form>
      </div>
    </div>
  );
};

const styles = {
  container: {
    display: 'flex',
    justifyContent: 'center',
    alignItems: 'center',
    height: '100vh',
    backgroundColor: '#f9f9f9',
  },
  registrationBox: {
    display: 'flex',
    flexDirection: 'column',
    alignItems: 'center',
    padding: '40px',
    backgroundColor: '#ffffff',
    boxShadow: '0 4px 12px rgba(0, 0, 0, 0.1)',
    borderRadius: '10px',
    width: '400px',
  },
  header: {
    marginBottom: '10px',
    color: '#333333',
    fontSize: '24px',
  },
```

```
subHeader: {
  marginBottom: '30px',
  color: '#666666',
  fontSize: '16px',
},
error: {
  color: 'red',
  marginBottom: '20px',
},
inputGroup: {
  marginBottom: '20px',
  width: '100%',
  textAlign: 'left',
},
label: {
  display: 'block',
  marginBottom: '5px',
  color: '#555555',
},
input: {
  width: '100%',
  padding: '12px',
  boxSizing: 'border-box',
  border: '1px solid #cccccc',
  borderRadius: '6px',
  outline: 'none',
  transition: 'border-color 0.2s',
},
inputFocus: {
  borderColor: '#0066cc',
},
```

```
  button: {
    width: '100%',
    padding: '12px',
    backgroundColor: '#0066cc',
    color: '#ffffff',
    border: 'none',
    borderRadius: '6px',
    cursor: 'pointer',
    fontSize: '16px',
    transition: 'background-color 0.2s',
  },
  buttonHover: {
    backgroundColor: '#004d99',
  },
  footer: {
    marginTop: '20px',
    color: '#666666',
    fontSize: '14px',
  },
  link: {
    color: '#0066cc',
    textDecoration: 'none',
  },
};
export default Signup;
```

6.3 HOME PAGE

When the user enters the home page the user is allowed to select the category of the product they want to select from , the home page is more user-friendly

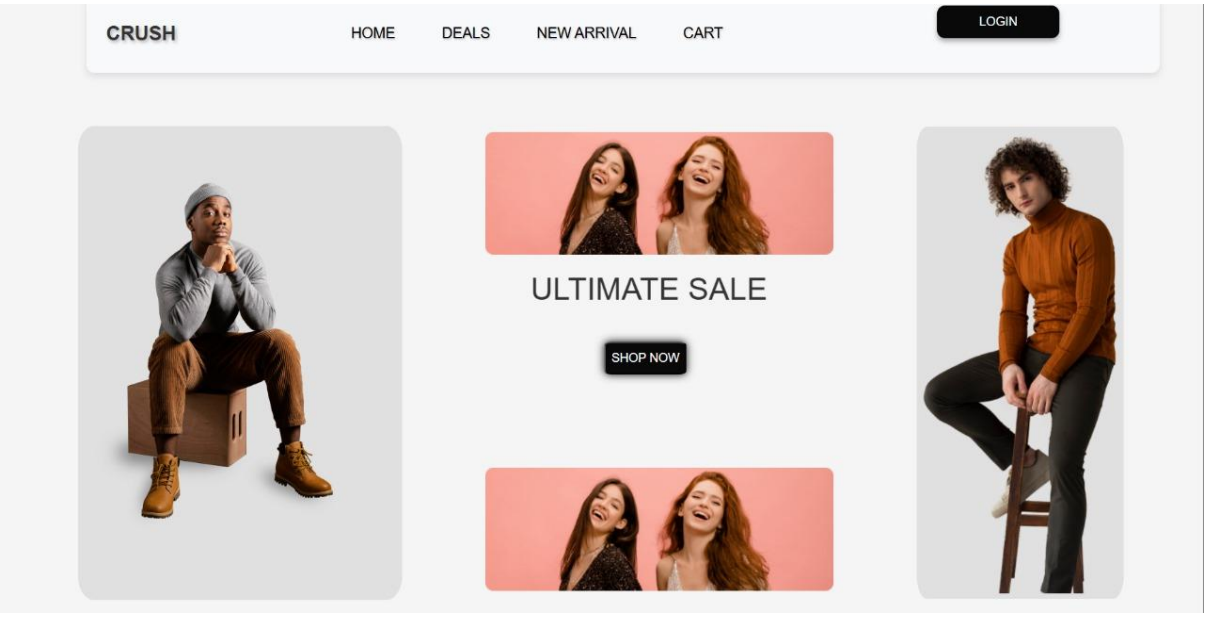


Fig 6.3 HOME PAGE

6.3.1 CODING

```
import React from "react";
import "../Home/Home.css";
import Feedback from "./Feedback";
import img from "../../assets/Images/home1.png";
import img1 from "../../assets/Images/home2.png";
import img2 from "../../assets/Images/images.png";
import img3 from "../../assets/Images/home4.png";
import img4 from "../../assets/Images/home5.png";
import {useNavigate} from "react-router-dom"
const Home = () => {
  const navi = useNavigate();
```

```
  const handleClick = () => {
    navi("/products");
  }
  return (
    <div>
      <main className="imgtotal">
        <div className="img1">
          <img src={img} alt="Home Image 1" />
        </div>
        <div className="img-center-elements">
          <div className="center-img1">
            <img src={img2} alt="Center Image 1" />
          </div>
          <div className="center-heading">
            ULTIMATE SALE<br />
          </div>
          <button className="butt" onClick={handleClick}>SHOP
NOW</button>
          <div className="center-img2">
            <img src={img2} alt="Center Image 2" />
          </div>
        </div>
        <div className="img2">
          <img src={img1} alt="Home Image 2" />
        </div>
      </main>
      <div className="logs">
        <img src={img3} alt="Logs Image" />
      </div>
      <section className="fullfollow">
        <h1 className="follow">Follow Us On Instagram</h1>
```

```
        <div className="insta">
          <img src={img4} alt="Instagram Promotion" />
        </div>
        <div className="promotional-text">
          Get the latest updates and exclusive offers!
        </div>
<br></br>        <a href="#" className="call-to-action">  <b4></b4>Join
Now</a>
      </section>
      <Feedback />
    </div>
  );
};
export default Home;
```

6.4 DELAS PAGE

The deals page shows the special offers that are ongoing on website. In general , deals page shows the deal of the month for the user and it will redirect the user to the available deals.

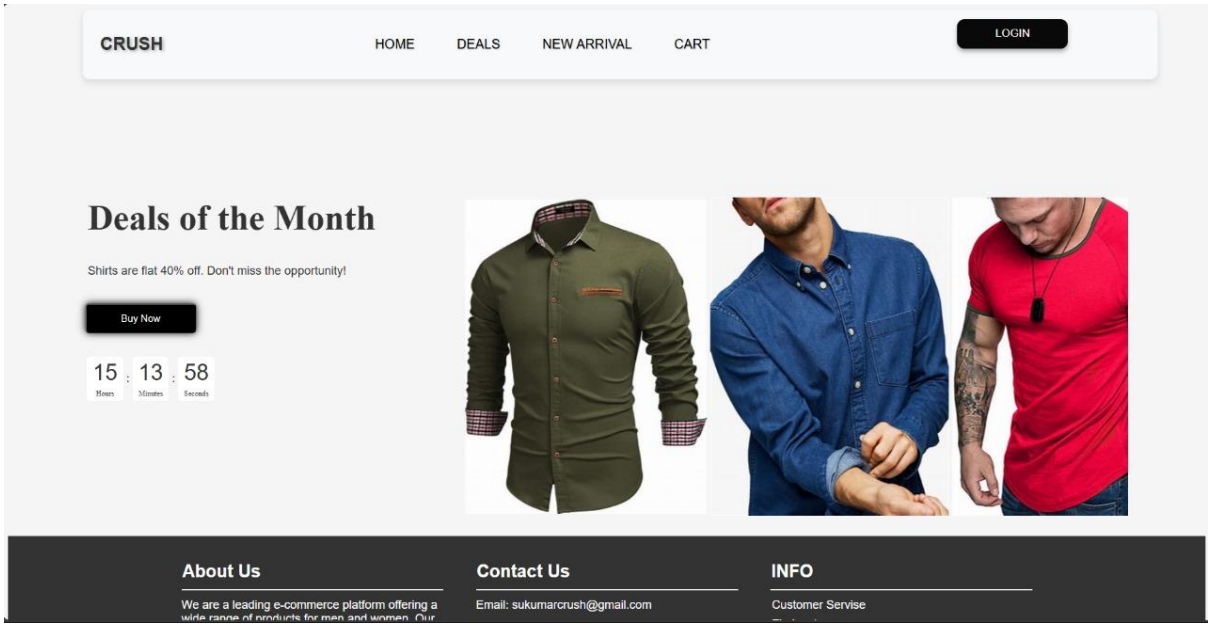


Fig 6.4 DEALS OF THE MONTH PAGE

6.4.1 CODING

```
import React, { useState, useEffect } from "react";
import "./Deals.css";
import { useNavigate } from "react-router-dom";
const images = [
  "/images/image1.jpg",
  "/images/image2.jpg",
  "/images/image3.jpg",
  // Add more image paths as needed
];
const GridDisplay = () => {
  return (
    <div className="grid-container">
      <div className="grid-item">Item 1</div>
      <div className="grid-item">Item 2</div>
      <div className="grid-item">Item 3</div>
      <div className="grid-item">Item 4</div>
```

```
    </div>
  );
};
const CountdownTimer = () => {
  const calculateTimeLeft = () => {
    const difference = +new Date("2024-12-31") - +new Date();
    let timeLeft = {};
    if (difference > 0) {
      timeLeft = {
        hours: Math.floor((difference / (1000 * 60 * 60)) % 24),
        minutes: Math.floor((difference / 1000 / 60) % 60),
        seconds: Math.floor((difference / 1000) % 60),
      };
    }
    return timeLeft;
  };
  const [timeLeft, setTimeLeft] = useState(calculateTimeLeft());
  const [showGrids, setShowGrids] = useState(false);
  useEffect(() => {
    const timer = setTimeout(() => {
      setTimeLeft(calculateTimeLeft());
    }, 1000);
    return () => clearTimeout(timer);
  });
  const navi = useNavigate();
  const handleButtonClick = () => {
    navi("/products");
    setShowGrids(true);
    window.scrollTo({
      top: document.documentElement.scrollHeight,
      behavior: "smooth",
```

```
  });
};
return (
  <div className="full">
    <div className="countdown-slider-container">
      <div className="countdown-timer">
        <div className="header">
          <h1>Deals of the Month</h1>
        </div>
        <div className="sub-header">
          <p>Shirts are flat 40% off. Don't miss the opportunity!</p>
        </div>
        <button onClick={handleButtonClick}>Buy Now</button>
        <div className="timer">
          <div className="time-box">
            <div className="time">
              {String(timeLeft.hours).padStart(2, "0")}
            </div>
            <div className="label">Hours</div>
          </div>
          <div className="colon">:</div>
          <div className="time-box">
            <div className="time">
              {String(timeLeft.minutes).padStart(2, "0")}
            </div>
            <div className="label">Minutes</div>
          </div>
          <div className="colon">:</div>
          <div className="time-box">
            <div className="time">
              {String(timeLeft.seconds).padStart(2, "0")}
```

```
            </div>
            <div className="label">Seconds</div>
          </div>
        </div>
      </div>
      <div className="image-slider">
        <div className="image-container">
          {images.map((item, index) => (
            <img src={item} />
          ))}
        </div>
      </div>
      {showGrids && <GridDisplay />}
    </div>
  </div>
 );
};
export default CountdownTimer;
```

6.5 CATEGORY PAGE

The category page displays the all the products that will be available at our website. Crush offers for both men and women

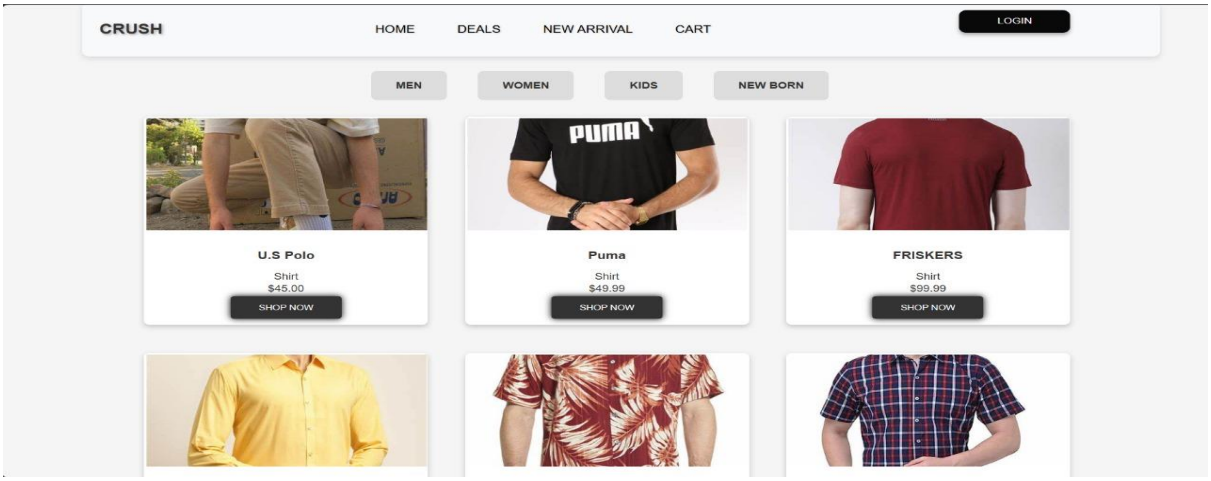


Fig 6.5 CATEGORY PAGE

6.5.1 CODING

```
import React, { useEffect, useState } from "react";
import axios from "axios";
import { useNavigate } from "react-router-dom";
import "../NewArrival/Newarrival.css";
const Layout = () => {
  const [elements, setElements] = useState([]);
  const [category, setCategory] = useState("men");
  const navigate = useNavigate();
  const fetchData = (category) => {
    const apiUrl =
      category === "men"
        ? "http://localhost:3000/api/men"
        : "http://localhost:3000/api/women";
    axios
      .get(apiUrl)
      .then((response) => {
```

```
      setElements(response.data);
    })
    .catch((error) => {
      console.error("There was an error fetching the data!", error);
    });
};
useEffect(() => {
  fetchData(category);
}, [category]);
const handleShopNowClick = (element) => {
  navigate(/product/${element.id}, { state: { product: element } });
};
return (
  <div>
    <div className="new-arrivals-nav">
      <ul>
        <a href="#" onClick={() => setCategory("men")}>
          <li>Men</li>
        </a>
        <a href="#" onClick={() => setCategory("women")}>
          <li>Women</li>
        </a>
        <a href="#">
          <li>Kids</li>
        </a>
        <a href="#">
          <li>New Born</li>
        </a>
      </ul>
    </div>
    <div className="new-arrivals-body">
```

```
      <div className="new-arrivals-box-section">
        <div className="new-arrivals-box">
          {elements.map((element, index) => (
            <div
              key={index}
              className="box"
              style={{ width: "24rem", height: "23rem" }}
            >
              <img
                src={element.image}
                alt={element.Brand}
                className="box-img" width="15rem" height="20rem"
              />
              <div className="box-content">
                <h3>{element.Brand}</h3>
                <p>{element.Category}</p>
                <p>{element.Price}</p>
                <button onClick={() => handleShopNowClick(element)}>
                  SHOP NOW
                </button>
              </div>
            </div>
          ))}
        </div>
      </div>
    </div>
  );
};

export default Layout;
```

6.6 CART PAGE

Cart page shows the products that are selected by the user to buy later or buy in a whole

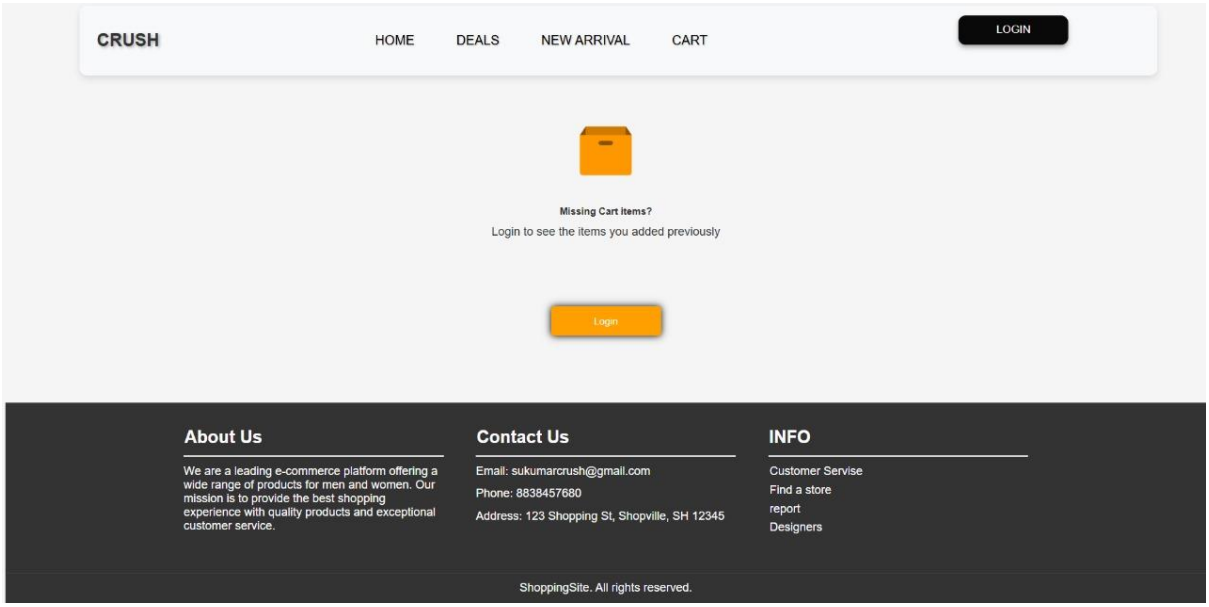


Fig 6.6 CART PAGE

6.6.1 CODING

```
import React from 'react';
import './Cart.css';
import {useNavigate} from "react-router-dom"
const Cart = () => {
  const navi = useNavigate();
  const handleClick = () => {
    navi("/LoginPage");
  }
  return (
    <div className="container-fluid">
```

```
      <div className="main-content">
        <img src="https://img.icons8.com/color/96/000000/empty-box.png"
alt="Empty Cart" className="cart-image"/>
        <h5 className="missing-items">Missing Cart items?</h5>
        <p className="login-prompt">Login to see the items you added
previously</p>
        <button className="login-button-main"
onClick={handleClick}>Login</button>
      </div>
    </div>
  );
};
export default Cart;
```

6.7 PAYMENT PAGE

The payment page that we developed is more user-friendly. We offer both credit and debit card and upi payment.

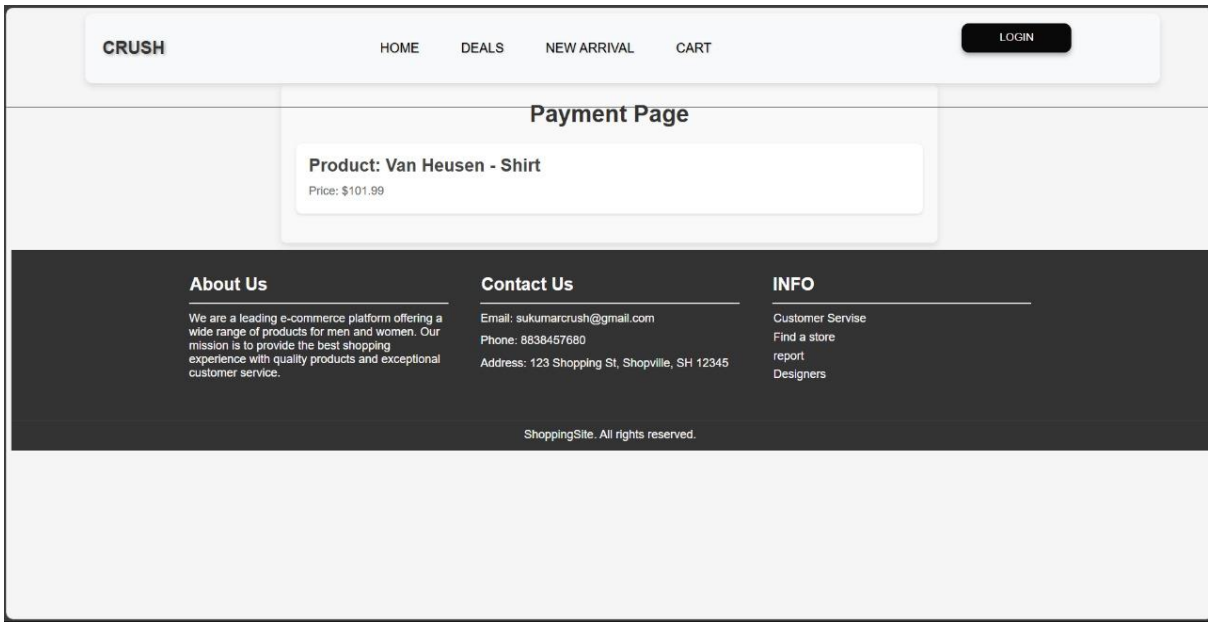


Fig 6.7 PAYMENT PAGE

6.7.1 CODING

```
import React from "react";
import { useLocation } from "react-router-dom";
import "./PaymentPage.css"; // Optional: Add styles as needed
const PaymentPage = () => {
  const location = useLocation();
  const { product } = location.state || {};
  return (
    <div className="payment-page">
      <h1>Payment Page</h1>
      {product ? (
        <div className="product-summary">
          <h2>Product: {product.Brand} - {product.Category}</h2>
          <p>Price: {product.Price}</p>
          {/* Add more product details as needed */}
        </div>
      ) : (
        <p>No product details available.</p>
      )}
      {/* Payment form or integration goes here */}
    </div>
  );
};
export default PaymentPage;
```

6.8 BACKEND CODING

USER.JSX

This provide a structure for the api that will be going to be created

```
import mongoose ,{Schema} from "mongoose";
const scheme = new Schema({
    email:String,
    password:String,
    username:String,
    mobilenumber:String,
});
export const User = mongoose.model('User', scheme);
```

ROUTES.JSx

This file will fetch data from the server and can be used in the application

```
import express from 'express';
import { User } from '../model/User.js';
const router = express.Router();
router.get("/user", async(req, res) => {
    const data = await User.find();
    res.json(data);
});
router.post("/user",async(req,res)=>{
    const newData = await User.insertMany(req.body);
    res.json(newData);
});
export default router;
```

INDEX.JS

```
import express from 'express';
```

```
import mongoose from 'mongoose';
import router from './Router/Route.js';
import cors from 'cors';
const app = express();
app.use(express.json());
app.use(cors());
app.listen(3000);
app.use("/",router);
mongoose.connect("mongodb+srv://Shameer182005:Shameer%2305@clu
ster0.t5s86.mongodb.net/LoginAPI?retryWrites=true&w=majority&appN
ame=Cluster0")
.then(()=>console.log("SUccssfull"))
.catch((err)=>console.log(err));
```

CONCLUSION

CHAPTER 7

CONCLUSION

This chapter tells about the conclusion that anyone can drive from the project andthe learning we learnt by taking over this project.

7.1 CONCLUSION

Crush is poised to make a significant impact in the online fashion retail market by focusing exclusively on dresses for men and women. By offering a curated selection of high-quality dresses, a personalized shopping experience, and a commitment to inclusivity and sustainability, Crush differentiates itself from more generalized fashion platforms. The platform's user-friendly design, efficient search and filtering options, and seamless checkout process all contribute to a superior shopping experience. With its specialized focus, Crush aims to become the go-to destination for anyone looking for stylish, well-crafted dresses, setting a new standard in the e-commerce fashion industry.

7.2 FUTURE SCOPE

The future scope for Crush is expansive, with opportunities to enhance its market presence and customer experience significantly.

Crush could broaden its product lines to include accessories, shoes, and outerwear, creating a comprehensive fashion destination. International expansion is another avenue, allowing Crush to tap into global markets with tailored offerings.

Advancements in AI and machine learning could further personalize the shopping experience, while augmented reality (AR) features could bridge the gap between online and in-store experiences by enabling virtual try-ons.

Sustainability remains a key focus, with potential initiatives such as dress recycling programs and eco-friendly partnerships. Additionally,

Crush could explore subscription services, social commerce integration, and community-building efforts, including influencer collaborations, to foster brand loyalty and engagement.

Custom dress design services could cater to customers seeking unique, made-to-order garments. Finally, investments in logistics could enhance delivery speed, offering same-day or next-day shipping options. These initiatives will position Crush as a forward-thinking leader in the e-commerce fashion industry.

REFERENCES

REFERENCES

1 Geeksforgeeks (https://geeksforgeeks.com)

2 Stockoverflow (https://stackoverflow.com)

3 Ajio (https://Ajio.com)

4 W3schools (https://w3schools.com)

5 Codeacedemy (https://codeacedemy.com)