E-Commerce - Clothing Website

A PROJECT REPORT

Submitted by

Name : Utkarsh Singh
Reg. No. : 12219342
Roll No. : 31
Section : K22DT

For
Course Code : INT 252

in partial fulfillment for the award of the degree of

B.tech
IN
CSE

Lovely Professional University, Punjab

A PROJECT REPORT
ON

E-Commerce - Clothing Website

Submitted by

Utkarsh Singh

in partial fulfillment for the award of the degree of

B.tech

Lovely Professional University, Punjab

Lovely Professional University, Punjab

BONAFIDE CERTIFICATE

Certified that this project report of **“Clothing Website”** is the bonafide work of “**Utkarsh Singh”** who carried out the project work under my supervision.

Dr. Manjot Kaur
SIGNATURE
Dr. Manjot Kaur
SUPERVISOR

Dr. Manjot Kaur
SIGNATURE

Dr. Manjot Kaur
CLASS MENTOR

Gaurav Pushkarna
SIGNATURE
Gaurav Pushkarna
HEAD OF THE DEPARTMENT

TABLE OF CONTENTS

ABSTRACT

This project is a comprehensive e-commerce website designed specifically for clothing retail, built using modern technologies including React for the front end, Node.js for the back end, and MongoDB as the database. The website provides a seamless and interactive shopping experience for users, allowing them to browse a variety of clothing products, view detailed product descriptions, add items to their cart, and complete secure purchases. The responsive design ensures optimal performance across devices, catering to a wide user base.

On the front end, React enables a dynamic and user-friendly interface, offering a smooth, single-page application experience with real-time updates and an intuitive user interface. State management and component-based architecture facilitate easy scalability and maintenance. The back end, developed with Node.js, manages the application's server logic and handles user requests, payment processing, and secure authentication. MongoDB serves as the database, enabling efficient storage and retrieval of product details, user information, and order histories, which are essential for a personalized shopping experience.

The project emphasizes security, usability, and performance, creating a reliable platform for both users and administrators. Features include user registration, login authentication, product management, order management, and an admin dashboard for overseeing inventory and sales. This project demonstrates the integration of modern web technologies to create a fully functional, scalable, and secure e-commerce platform for the clothing industry.

LIST OF TABLES

1. **Product Perspective**
   The Product Perspective section describes the overall context and role of the e-commerce platform within a larger system. It highlights the platform's purpose, intended usage, and how it integrates with other parts of the business, such as inventory management and customer support systems. This section outlines the project's objectives, target audience, and the competitive advantage it aims to achieve within the e-commerce clothing market.

2. **Product Functions**
   The Product Functions section provides a detailed list of the main features and functionalities the platform offers. This includes essential e-commerce capabilities like product browsing, filtering, searching, adding items to a cart, checking out, and processing payments. It also includes user account functions, such as registration, login, and profile management, as well as admin functions for managing inventory, updating product listings, and monitoring sales. Each function supports the website's goal of delivering a seamless and convenient online shopping experience.

3. **User Characteristics**
   This section defines the types of users expected to interact with the platform and their specific needs. It typically covers two main groups: customers (casual shoppers, returning buyers) and administrators (store owners, managers). Customer characteristics focus on aspects like browsing preferences, familiarity with e-commerce, and device usage (mobile, desktop). For administrators, it includes characteristics such as technical knowledge, system management requirements, and access needs for administrative tools. This section ensures that the platform's features align with user expectations and improve usability across all user types.

4. **General Constraints**
   The General Constraints section outlines limitations or restrictions that could

impact the platform’s performance, scalability, and usability. These constraints may include factors such as hosting and server limitations, budget constraints, and compliance with data privacy regulations (e.g., GDPR). Technical constraints might also involve the capabilities of the chosen tech stack (React, Node.js, MongoDB) and potential limitations in handling high volumes of concurrent users or extensive data storage.

5. **Assumptions and Dependencies**
   This section lists the assumptions made during the development process and external factors that the project depends on. Assumptions might include expectations of user internet speeds, mobile accessibility, and the availability of up-to-date browser versions. Dependencies can include reliance on third-party services, such as payment gateways, shipping APIs, and email notification systems. Understanding these assumptions and dependencies is crucial, as changes in any of these elements could affect project timelines, functionality, or overall system reliability.

LIST OF FIGURES

- **SPECIFIC REQUIREMENTS**

1. External Interface Requirements

This section covers the various interfaces the e-commerce platform requires to interact with external systems, users, hardware, and software. It ensures all necessary connections and integrations are well-defined for smooth operation.

- **1.1 User Interfaces**

  This figure outlines the front-end design elements that users interact with, such as the layout and components of web pages like the homepage, product listings, shopping cart, and checkout. It describes the visual structure, usability aspects, and overall experience designed to be intuitive and visually appealing for users, enhancing engagement and accessibility.

- **1.2 Hardware Interfaces**

  This section defines the hardware components required for the platform to operate effectively. It includes server specifications for hosting the website and handling data requests, as well as the end-user devices (e.g., mobile phones, tablets, desktops) that the platform is optimized for. These requirements help ensure that the platform runs efficiently on the specified hardware.

- **1.3 Software Interfaces**

  The software interfaces define interactions between the e-commerce system and other software components, such as the MongoDB database, Node.js backend, and third-party services like payment processors and shipping APIs. It specifies how these integrations are structured to support core functionality, including data storage, processing, and external service interactions.

- **1.4 Communications Interfaces**

  This section focuses on communication protocols and methods, such as

HTTP/HTTPS, for secure web traffic. It also includes specifications for any web service APIs that allow the platform to communicate with third-party services, handle data transactions securely, and enable interactions between the front-end and back-end services.

2. Functional Requirements

This section details the main functional requirements, or essential features, that the platform needs to deliver to fulfill its purpose as an e-commerce site.

- **2.1 Functional Requirement or Feature**
- **2.1.1 Introduction**

  This section introduces each functional requirement by explaining its purpose within the e-commerce platform. For example, if the requirement is for "Product Browsing," this section would describe why product browsing is essential to the user experience. It provides context on how the feature enhances user engagement, helps with product discovery, and ultimately supports the goal of driving sales. This introduction also outlines any key assumptions, dependencies, or background information relevant to the functionality.

- **2.1.2 Inputs**

  The Inputs section describes the data or actions that trigger this functional requirement. For example, in the "Product Browsing" feature, inputs might include search terms entered by the user, filter selections (like category or price range), or actions such as scrolling through a product list. This section details where these inputs come from, whether directly from user actions or indirectly from other parts of the system (e.g., recommendation algorithms or saved user preferences).

- **2.1.3 Processing**

  Processing outlines the specific actions and calculations performed based on the inputs to deliver the required functionality. For the product browsing feature, processing may involve querying the MongoDB database to retrieve product information that matches the search and filter criteria, sorting the results, and then formatting the data for display. This section breaks down any significant steps or algorithms used to handle the input data, focusing on how it transforms raw input into meaningful results for users.

- **2.1.4 Outputs**

  The Outputs section describes the results produced by the functional requirement after processing is complete. For the “Product Browsing” feature, outputs might include a list of product results that meet the user’s search criteria, each with information like product name, price, description, and image. This section defines the format, structure, and any additional elements (such as pagination controls) that make the output user-friendly and visually appealing.

- **2.1.5 Error Handling**

  Error Handling specifies the measures taken to handle potential issues or incorrect inputs for each functional requirement. For example, if a user enters an invalid search term or no products match the search criteria, error handling may display a “No products found” message. This section also covers how the system handles technical errors, such as database connection issues, and ensures that users receive clear, helpful feedback without the platform crashing or displaying confusing error codes. Proper error handling ensures a resilient and user-friendly experience, even when unexpected issues arise.

  Each of these sub-sections ensures that the functional requirements are implemented in a detailed, user-oriented, and technically robust manner.

3. Non-Functional Requirements

These requirements cover aspects of the platform's quality attributes and ensure it performs well and is reliable, secure, and maintainable.

- **3.1 Performance**

  Performance requirements ensure that the platform responds quickly and can handle a high volume of user requests. It includes page load times, data retrieval speeds, and the ability to scale during peak usage times, ensuring a smooth and efficient shopping experience.

- **3.2 Reliability**

  Reliability focuses on the platform’s ability to operate consistently without failures. It specifies system uptime targets, error-handling mechanisms, and measures to ensure the system remains operational even if minor issues arise.

- **3.3 Availability**

  This requirement specifies the platform’s expected availability and uptime. For an e-commerce site, availability is critical to ensure that users can access the site 24/7. This section includes measures to achieve high availability, such as backup systems, failover servers, and regular maintenance schedules.

- **3.4 Security**

  Security requirements cover aspects like data protection, user authentication, and secure transactions. It specifies measures to safeguard user data, prevent unauthorized access, and ensure that sensitive information, such as payment details, is encrypted and protected against breaches.

- **3.5 Maintainability**

  Maintainability refers to the ease with which developers can update, troubleshoot, and expand the platform. This includes documentation practices, code organization, and modular design principles, all of which help developers make updates and fix issues efficiently.

- **3.6 Portability**

  Portability ensures that the platform can be adapted to various environments, such as different operating systems and devices. This requirement addresses the need for a responsive design compatible with mobile, tablet, and desktop use, ensuring users have a consistent experience across platforms.

List of Symbols, Abbreviations, and Nomenclature

- **API**: Application Programming Interface - A set of protocols and tools for building software applications that allow different software components to communicate with each other.

- **CSS**: Cascading Style Sheets - A style sheet language used for describing the presentation of a document written in HTML or XML, including layout, colors, and fonts.

- **DB**: Database - A structured collection of data that can be accessed and managed.

- **e-commerce**: Electronic Commerce - The buying and selling of goods or services using the internet.

- **HTTP**: Hypertext Transfer Protocol - An application protocol used for transmitting hypermedia documents, such as HTML, on the web.

- **HTTPS**: Hypertext Transfer Protocol Secure - An extension of HTTP that uses encryption to provide a secure communication channel over a computer network.

- **JSON**: JavaScript Object Notation - A lightweight data interchange format that is easy for humans to read and write and easy for machines to parse and generate.

- **MongoDB**: A NoSQL database that stores data in flexible, JSON-like documents, allowing for more complex data structures and easy scalability.

- **Node.js**: An open-source, cross-platform JavaScript runtime environment that executes JavaScript code outside a web browser, primarily used for building server-side applications.

- **React**: A JavaScript library for building user interfaces, especially single-page applications, where a seamless user experience is essential.

- **UI**: User Interface - The space where interactions between humans and machines occur, focusing on the design of user interfaces for software and machines.

- **UX**: User Experience - The overall experience a user has when interacting with a product or service, encompassing aspects like usability, accessibility, and satisfaction.

- **URL**: Uniform Resource Locator - The address used to access resources on the internet.

CHAPTERS

1. Introduction

The rapid growth of the e-commerce industry has transformed the way consumers shop, making online platforms essential for businesses in today’s digital age. This project focuses on developing a comprehensive e-commerce website for clothing, aimed at providing a seamless shopping experience for users. With the rise of mobile and online shopping, customers expect an intuitive and efficient interface that allows them to browse products, make informed purchasing decisions, and complete transactions securely. By leveraging modern web technologies, this project aims to meet these consumer demands while also catering to the specific needs of clothing retailers, ensuring a robust and scalable solution.

The primary objectives of this project include creating an aesthetically pleasing and user-friendly interface, implementing essential functionalities such as user authentication, product categorization, and shopping cart management, and ensuring secure payment processing. The platform will utilize React for the front-end development, enabling dynamic user interactions, while Node.js will power the back-end services, providing a responsive and efficient server environment. Additionally, MongoDB will serve as the database, facilitating effective data storage and retrieval. By clearly defining the project’s scope and objectives, this documentation aims to provide a comprehensive overview of the development process and the final product, illustrating the significant role that a well-designed e-commerce website can play in enhancing the retail experience in the clothing industry.

2. Chapters Developing the Main Theme of the Project Work

This section comprises several crucial chapters that delve into the core functionalities and technical specifications of the e-commerce website, ensuring a comprehensive understanding of its structure, operations, and design principles.

1. System Requirements

This chapter outlines the essential requirements that the e-commerce platform must fulfill to operate effectively and meet user expectations.

- **Functional Requirements**
  Functional requirements describe the specific behaviors and functions the system must exhibit. This includes features like user registration and login, product browsing and search, shopping cart management, order processing, and payment integration. Each functionality is elaborated upon to ensure clarity in what the system is expected to do.
  - **Example**: For product browsing, requirements may specify that users should be able to filter products by category, size, color, and price range, as well as view detailed product descriptions and images.
- **Non-Functional Requirements**
  Non-functional requirements focus on the quality attributes of the system, such as performance, security, usability, and scalability. These requirements ensure that the platform not only meets functional needs but also provides a reliable and efficient user experience.
  - **Example**: The platform should load within three seconds to ensure a smooth user experience and must comply with security standards such as PCI DSS for payment processing.
- **External Interface Requirements**
  This section details how the e-commerce platform interacts with external systems and users. It includes:
  - **User Interfaces**: Design guidelines and user journey mapping to ensure an intuitive navigation experience.
  - **Hardware Interfaces**: Specifications regarding the server and client-side hardware requirements for optimal performance.
  - **Software Interfaces**: Dependencies on third-party services, libraries, and APIs for functionalities like payment processing and shipping.
  - **Communications Interfaces**: Details on protocols and methods for data exchange between client and server, emphasizing security protocols like HTTPS.

2. Functional Requirements

This chapter provides a deeper dive into the specific functionalities of the e-commerce platform, detailing how each feature will work and the interactions between users and the system.

- **Feature Descriptions**
  Each major functionality is broken down into sub-features, outlining the steps users will take to interact with the system and the expected system responses. This detailed approach helps ensure that the development team has a clear understanding of what needs to be built.

    - **Example**: For the shopping cart functionality, descriptions will include how users can add or remove items, view the cart summary, apply discount codes, and proceed to checkout.
- **Inputs, Processing, and Outputs**
  This section specifies the data inputs required for each functionality, how the system will process this information, and what outputs users can expect. This structure is vital for clarity and assists in the development and testing phases.
    - **Example**: When a user searches for a product, inputs could include keywords and filters; processing would involve querying the database for matching items; outputs would include a list of products that meet the search criteria.
- **Error Handling**
  The project must also define how the system will handle errors or exceptions that may occur during user interactions. This includes user-friendly error messages and recovery options, which are crucial for maintaining a positive user experience.
    - **Example**: If a payment fails, the system should provide clear instructions on how to proceed and possible reasons for the failure, allowing users to correct any issues.

3. System Design

This chapter outlines the architectural and design decisions made throughout the project, illustrating how the various components of the e-commerce platform interact.

- **Architectural Design**
  The architectural design provides a high-level overview of the system's structure, including diagrams that represent the different layers of the application (e.g., presentation layer, business logic layer, and data layer). This visual representation helps stakeholders understand how the system is organized.
- **Database Design**
  A critical component of any e-commerce platform is its database, which stores all product, user, and transaction data. This section details the database schema, including tables, fields, data types, and relationships between different entities. It ensures efficient data management and retrieval.
    - **Example**: Tables may include Users, Products, Orders, and Cart, with defined relationships (e.g., each Order must be linked to a User).
- **User Interface Design**
  This section focuses on the design of the user interface (UI), including wireframes and mockups for key pages of the website, such as the homepage,

product listing pages, product detail pages, and the checkout process. It outlines the visual design principles applied to create an attractive and functional user experience.

4. Implementation

This chapter describes the actual coding and development processes used to bring the e-commerce platform to life.

- **Development Process**
  This section outlines the methodologies and practices adopted during the development phase, such as Agile or Scrum, highlighting the importance of iterative development and regular feedback.
- **Front-End Implementation**
  Details the technologies and frameworks used for developing the user interface, such as React. This includes discussions on component architecture, state management, and responsive design considerations.
- **Back-End Implementation**
  Describes the server-side technologies used, such as Node.js, and details how APIs are developed to handle requests from the front end, manage sessions, and interact with the database.
- **Integration**
  This part covers how the front-end and back-end components were integrated to ensure a cohesive user experience. It may include information on testing strategies used to verify that all parts of the system work together seamlessly.

5. Testing and Quality Assurance

This chapter focuses on the strategies and methods employed to ensure the e-commerce platform functions correctly and meets the established requirements.

- **Testing Methodologies**
  Discusses the various testing approaches used, including unit testing, integration testing, system testing, and user acceptance testing. It highlights the importance of thorough testing to identify and resolve issues before deployment.
- **Quality Assurance Practices**
  Outlines the practices in place to maintain high-quality standards throughout development, including code reviews, continuous integration/continuous deployment (CI/CD) pipelines, and automated testing tools.

- **Feedback and Iteration**
  This section emphasizes the importance of user feedback during testing phases and how it informs further development, ensuring that the final product aligns with user needs and expectations.

This comprehensive overview of the chapters developing the main theme of the project work ensures that all aspects of the e-commerce platform are thoroughly addressed, providing clarity and structure to the project documentation. Each chapter builds upon the previous ones, creating a cohesive narrative that outlines the project's goals, implementation strategies, and anticipated outcomes.

3. Conclusion

In conclusion, the development of the e-commerce website for clothing has successfully met its objectives by creating a user-friendly platform that integrates key functionalities such as product browsing, shopping cart management, and secure payment processing. The project has navigated various challenges, including technical hurdles and resource constraints, demonstrating resilience and effective problem-solving strategies. Through the use of modern technologies like React for the front end, Node.js for the back end, and MongoDB for data management, the platform is designed to provide a seamless shopping experience for users. Looking ahead, opportunities for future enhancements, such as advanced analytics and personalized user experiences, could further enrich the platform's capabilities. Overall, this project not only showcases the technical skills and collaboration involved in its development but also highlights the potential impact of a well-designed e-commerce solution in the competitive online retail market.

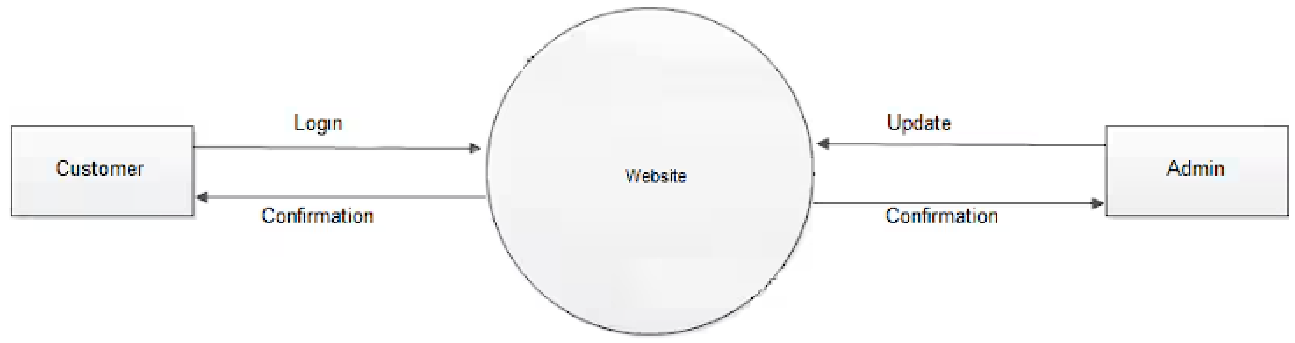
DFD (Data Flow Diagram)

A **Data Flow Diagram (DFD)** is a graphical representation of the flow of data within a system. It visually shows how data moves from one part of a system to another, where it is processed, stored, and outputted. DFDs are especially useful for understanding the structure of information systems and how various components interact. They come in multiple levels, each detailing more about the system's components and data flows.

Before we explore the details of creating a DFD for clothing website, let's first understand the key symbols used in a **Data Flow Diagram for an online shopping system**:

1. **Data Flow**: A Data Flow in a diagram is like a pathway that shows how information moves from one place to another. It could be when you choose a product online and it goes into your shopping cart.
2. **Process:** The Process symbol represents an action or task. In an online shopping diagram, it could be something like when you pay for the items in your cart. It shows what happens to the information.
3. **Entity**: An Entity is something like a group or a person. In an online shopping diagram, it might represent you as a customer. It shows who is involved in the process.
4. **Data Store:** A Data Store is like a storage place where information is kept. In online shopping, this could be where details about products or your order are saved. It's like a virtual shelf where things are kept until they're needed.

Level 0 DFD :



The diagram you've provided is a **Level 0 DFD** (also known as a Context Diagram) for a clothing website. It gives a high-level overview of how the system interacts with external entities, showing only the main process and its connections with outside sources. Here’s a breakdown of the components:

Components in the Level 0 DFD

1. **Customer**:
   - The **Customer** is an external entity that interacts with the website.
   - **Interactions**:
     - **Login**: The Customer sends a login request to access their account or make purchases.
     - **Confirmation**: After logging in or performing certain actions (e.g., making a purchase), the Customer receives a confirmation response from the website.
2. **Admin**:
   - The **Admin** is another external entity that has control over website content and settings.
   - **Interactions**:
     - **Update**: The Admin can send updates to the website, such as adding new products, updating prices, or managing inventory.
     - **Confirmation**: The Admin receives confirmation from the website after a successful update, ensuring the changes have been applied.
3. **Website**:
   - The central process in this DFD represents the entire clothing website, which handles interactions between the **Customer** and **Admin**.
   - The **Website**:
     - Processes login requests from Customers and provides confirmations.

    - Accepts updates from the Admin and sends confirmation once updates are completed.

Explanation of Data Flows

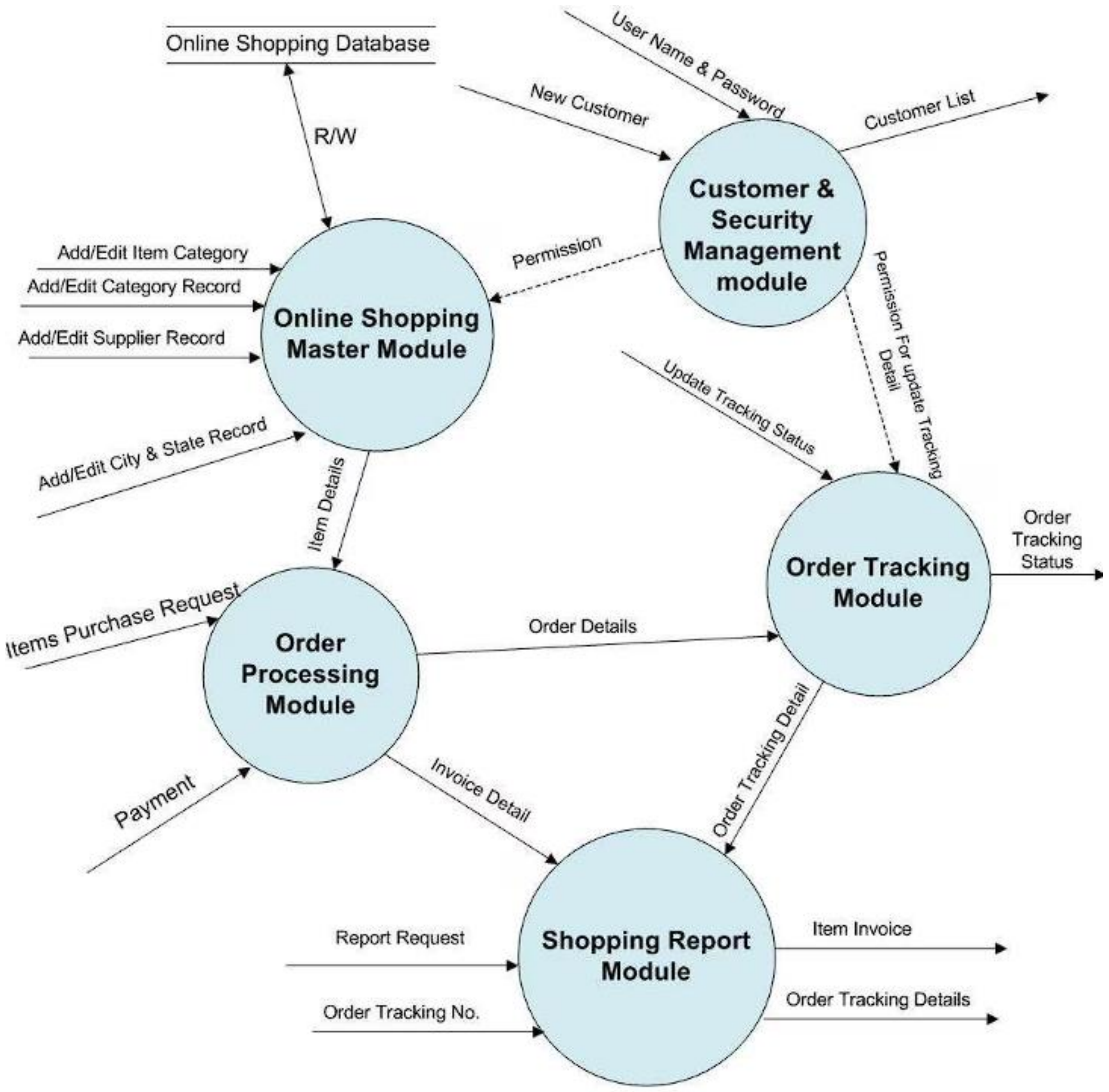
- **Login**: This flow shows the data coming from the Customer when they attempt to log in to the website.
- **Confirmation** (to Customer): Once a Customer logs in or makes a purchase, the website sends a confirmation back to inform them that their action was successful.
- **Update**: This flow indicates that the Admin can update website data, such as product listings, prices, or inventory.
- **Confirmation** (to Admin): Once the Admin's updates are applied, the website sends a confirmation back, verifying the success of the updates.

Purpose of this Level 0 DFD

This Level 0 DFD simplifies the complex interactions of the clothing website, focusing only on high-level interactions with external entities. It shows the basic communication flow without going into detail on specific processes or data manipulation within the system. This is ideal for providing an overview to stakeholders and designers before creating more detailed DFDs (Level 1 and beyond) that show the internal workings of the website.

In a real-world application, subsequent levels would break down the "Website" process into modules like **Product Catalog**, **Order Processing**, **User Management**, etc., to show detailed data handling.

Level 1 DFD :



The Level 1 Data Flow Diagram (DFD) provided represents the main modules and data flows of a **clothing e-commerce website**. This DFD breaks down the system into various core modules, showing how data flows between them and interacts with external entities like the database and customers. Here's an explanation of each component in this DFD:

Key Modules and Data Flows

1. **Customer & Security Management Module**:
    - **Purpose**: This module handles customer account creation, authentication, and permission management.
    - **Data Flows**:
        - **New Customer**: Information flow for registering new customers, which includes storing customer details.
        - **Username & Password**: Customers provide login credentials for authentication.
        - **Customer List**: This list includes all registered customers, which can be accessed or updated.
        - **Permission**: Grants appropriate access permissions for customers.
    - **Interactions**:
        - Communicates with the **Online Shopping Master Module** to manage customer access to other sections of the website.
        - Sends tracking status update permissions to the **Order Tracking Module**.
2. **Online Shopping Master Module**:
    - **Purpose**: This module manages the catalog, categories, and suppliers, along with location records relevant to the clothing products.
    - **Data Flows**:
        - **Add/Edit Item Category**: Allows the addition or modification of item categories (e.g., Men's, Women's, Kids' clothing).
        - **Add/Edit Category Record**: Maintains the list of product categories.
        - **Add/Edit Supplier Record**: Stores information on suppliers providing clothing items.
        - **Add/Edit City & State Record**: Stores location data for tracking delivery zones and managing logistics.
        - **Item Details**: Provides item-specific data (e.g., size, color, price) to the **Order Processing Module**.
    - **Interactions**:
        - Sends data to the **Order Processing Module** to process item-specific purchase requests.
        - Connects with the **Online Shopping Database** to read/write product, category, and supplier information.
3. **Order Processing Module**:
    - **Purpose**: This module handles customer orders, including item selection, purchase requests, and payment processing.
    - **Data Flows**:

        - **Items Purchase Request**: Manages the purchase request initiated by a customer.
        - **Payment**: Processes customer payment through a payment gateway.
        - **Order Details**: Sends order details to the **Order Tracking Module** for shipment updates.
        - **Invoice Detail**: Sends invoice details to the **Shopping Report Module** for report generation and record-keeping.
    - **Interactions**:
        - Communicates with the **Online Shopping Master Module** to retrieve item information for processing orders.
        - Interacts with both the **Order Tracking Module** and **Shopping Report Module** to update order status and generate invoices.

4. **Order Tracking Module**:
    - **Purpose**: This module manages and updates the status of customer orders as they move through different stages (e.g., Processing, Shipped, Delivered).
    - **Data Flows**:
        - **Update Tracking Status**: Allows updates on the current status of an order.
        - **Order Tracking Status**: Provides tracking updates back to customers.
        - **Order Details**: Receives initial order details from the **Order Processing Module** to start tracking.
        - **Order Tracking Detail**: Sends order tracking details to the **Shopping Report Module** for inclusion in order reports.
    - **Interactions**:
        - Works closely with the **Order Processing Module** to ensure each order is tracked correctly.
        - Communicates order tracking updates to the **Shopping Report Module**.
5. **Shopping Report Module**:
    - **Purpose**: This module generates various reports for order tracking, invoicing, and overall sales analysis.
    - **Data Flows**:
        - **Report Request**: Handles requests to generate specific reports, such as sales or inventory reports.
        - **Order Tracking No.**: Assigns a tracking number to each order.
        - **Item Invoice**: Generates invoices for customer orders.

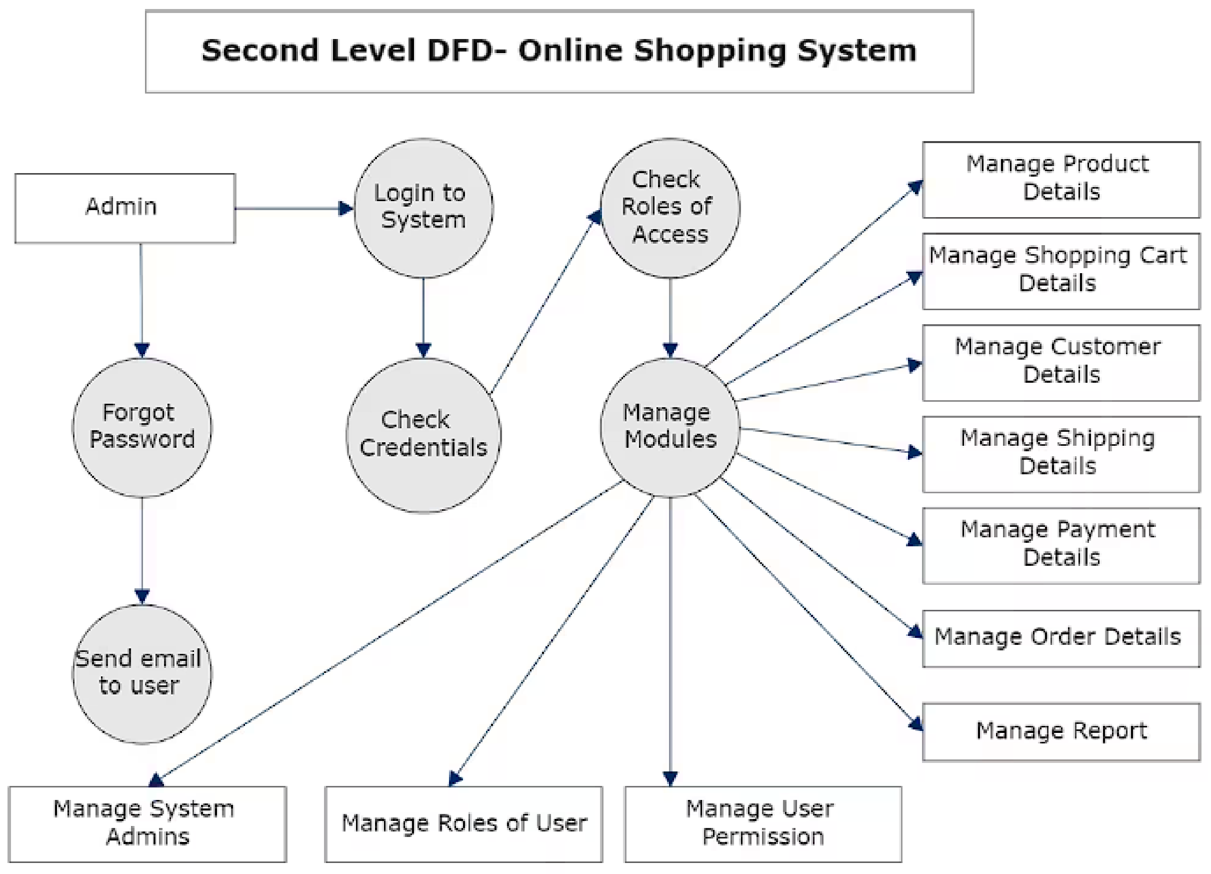
        - **Order Tracking Details**: Includes order tracking data in the reports for accurate order status.
    - **Interactions**:
        - Collects invoice details and tracking information from the **Order Processing Module** and **Order Tracking Module**.
        - Generates comprehensive reports that can be used for inventory management, sales analytics, and customer feedback.
6. **Online Shopping Database**:
    - This central data store holds records related to customers, products, categories, suppliers, and order details.
    - The **Online Shopping Master Module** reads and writes data to/from this database.

Overall Data Flow Summary

- **Customer & Security Management Module** manages customer-related operations and permissions.
- **Online Shopping Master Module** deals with the product catalog, categories, suppliers, and manages basic item information.
- **Order Processing Module** handles the purchase process, including payment and order details.
- **Order Tracking Module** provides order status updates and tracking.
- **Shopping Report Module** generates reports and invoices based on order and tracking details.

This Level 1 DFD provides a clear view of how an e-commerce clothing website's core modules interact with each other and manage customer and product data across the system.

Level 2 DFD :



The diagram shown is a Second-Level Data Flow Diagram (DFD) for an online shopping system, specifically adapted to manage modules and administrative tasks for an online clothing website. Let’s break down the components in this DFD:

1. Admin
The Admin user is the main actor who interacts with the system for management and administrative tasks.

2. Login to System
The Admin starts by logging into the system.
The process checks the Credentials of the user to verify their identity.

3. Forgot Password
If the admin forgets their password, the system has a Forgot Password option.

This triggers a process to Send an email to the user with instructions to reset their password.

4. Check Roles of Access
Once logged in, the system checks the Roles of Access for the admin to determine the level of permissions they have.
Based on the role, the admin can access various system modules.

5. Manage Modules
The main module in the DFD is Manage Modules, which handles the key administrative functionalities of the system.
From here, the admin can access and manage different components of the website, such as product and order details.

6. Manage System Admins
Allows the admin to manage other admins in the system, including adding, updating, and removing their accounts.

7. Manage Roles of User
This module enables the admin to define and assign roles to different users within the system, possibly for other staff or lower-level admins.

8. Manage User Permission
The admin can set permissions for users based on their roles, controlling what each user can access and modify in the system.

9. Manage Product Details
This module allows the admin to add, update, or delete product information for the clothing items on the website.

10. Manage Shopping Cart Details
Admin can review and manage the shopping cart data, including items added by customers.

11. Manage Customer Details
The system allows the admin to manage customer profiles and information.

12. Manage Shipping Details
Admin can manage shipping options, delivery methods, and shipping addresses for customer orders.

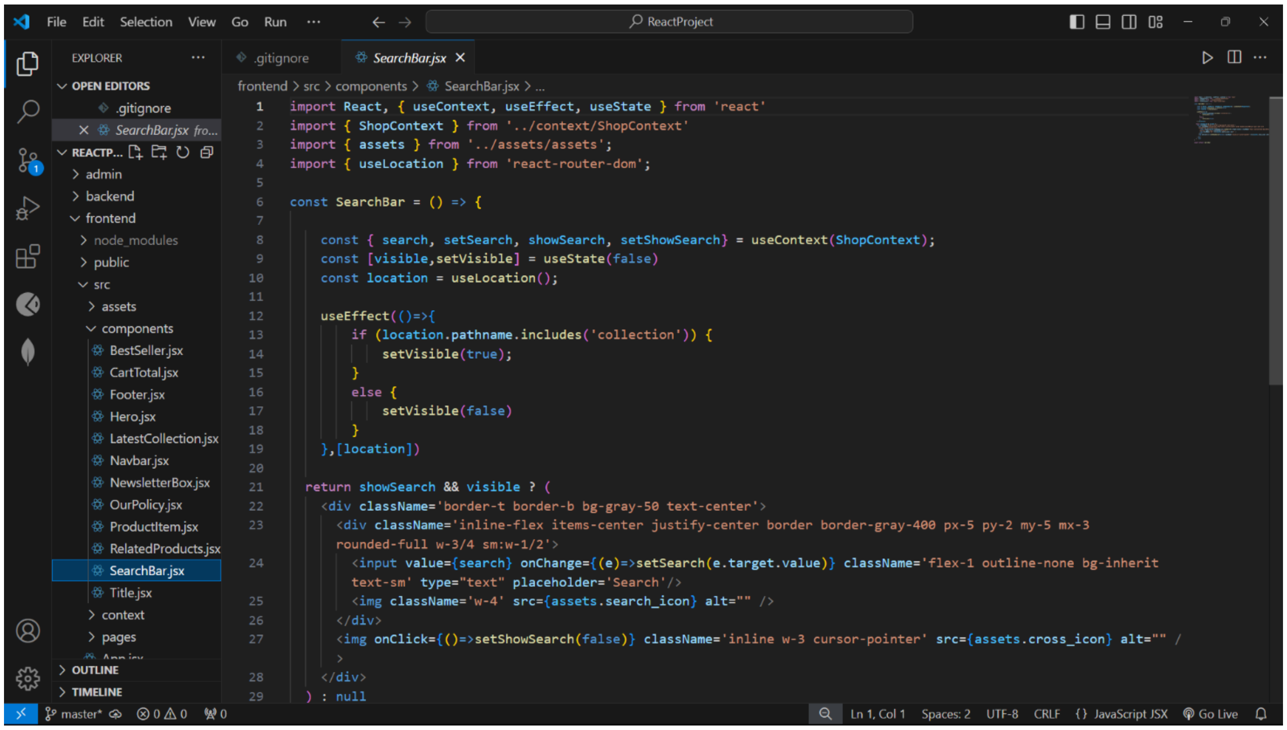
13. Manage Payment Details
This module handles payment-related information, including processing and verifying payments.

14. Manage Order Details
Admin can manage orders placed by customers, tracking order status, and processing returns or cancellations.

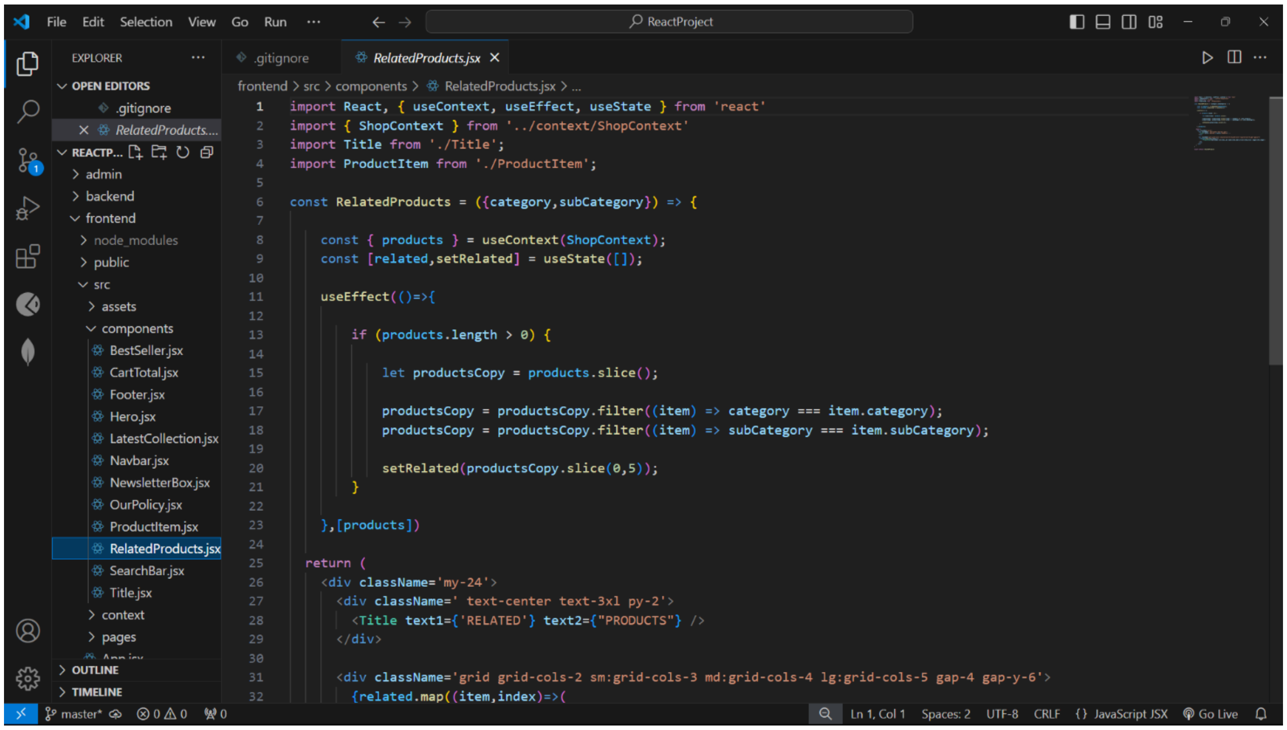
15. Manage Report
This module enables the admin to generate reports related to sales, inventory, and other performance metrics for business analysis.

Summary
This DFD shows how an online shopping system's administrative functions are organized to facilitate a smooth workflow for managing an online clothing store. It outlines the processes and modules for the admin to control product details, user roles, permissions, orders, and reporting, which are essential for maintaining an efficient e-commerce platform.

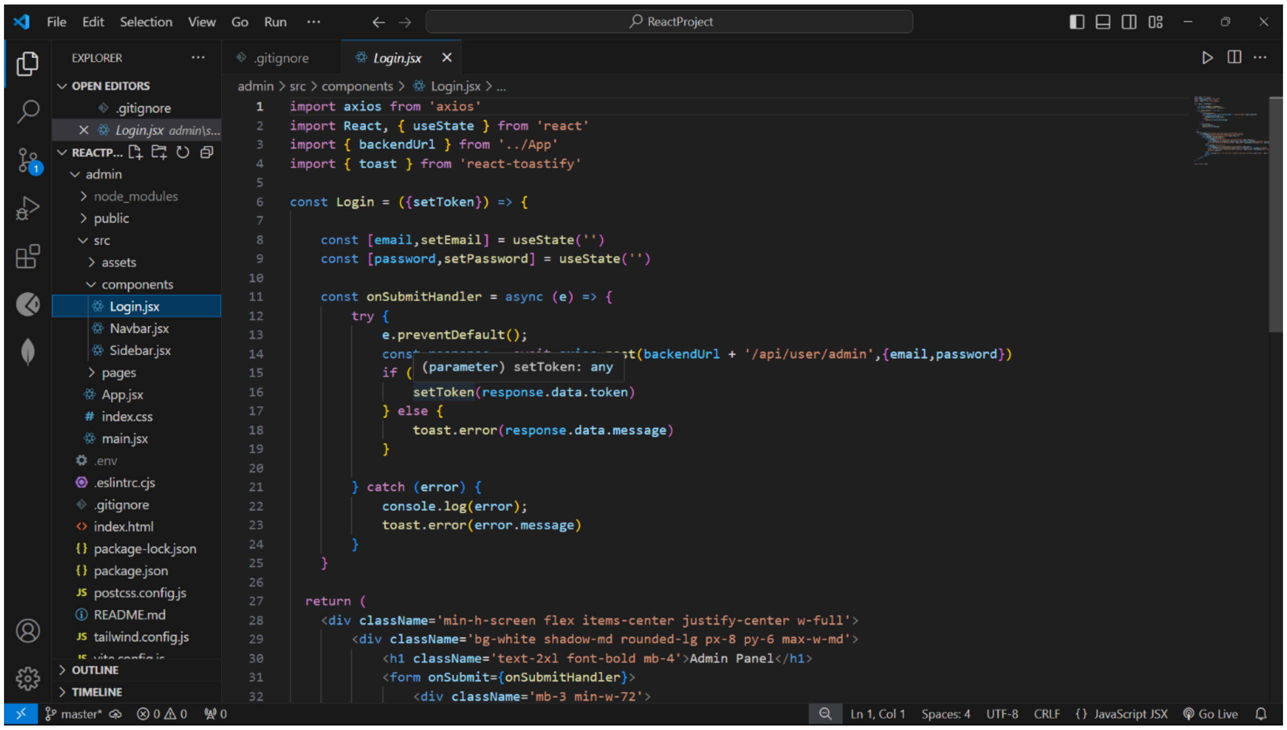
CODE SNAPSHOT



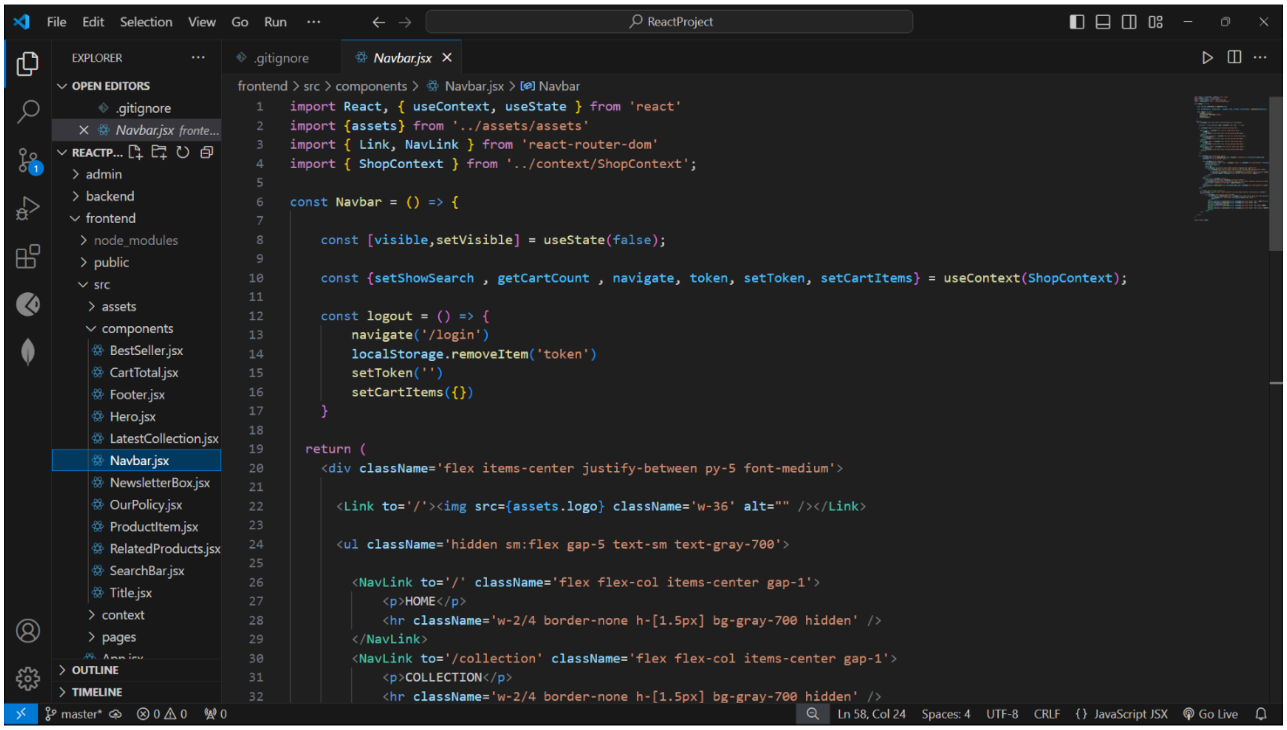
```jsx
import React, { useContext, useEffect, useState } from 'react'
import { ShopContext } from '../context/ShopContext'
import { assets } from '../assets/assets';
import { useLocation } from 'react-router-dom';

const SearchBar = () => {

    const { search, setSearch, showSearch, setShowSearch} = useContext(ShopContext);
    const [visible,setVisible] = useState(false)
    const location = useLocation();

    useEffect(()=>{
        if (location.pathname.includes('collection')) {
            setVisible(true);
        }
        else {
            setVisible(false)
        }
    },[location])

  return showSearch && visible ? (
    <div className='border-t border-b bg-gray-50 text-center'>
      <div className='inline-flex items-center justify-center border border-gray-400 px-5 py-2 my-5 mx-3
      rounded-full w-3/4 sm:w-1/2'>
        <input value={search} onChange={(e)=>setSearch(e.target.value)} className='flex-1 outline-none bg-inherit
        text-sm' type="text" placeholder='Search'/>
        <img className='w-4' src={assets.search_icon} alt="" />
      </div>
      <img onClick={()=>setShowSearch(false)} className='inline w-3 cursor-pointer' src={assets.cross_icon} alt="" /
      >
    </div>
  ) : null
```



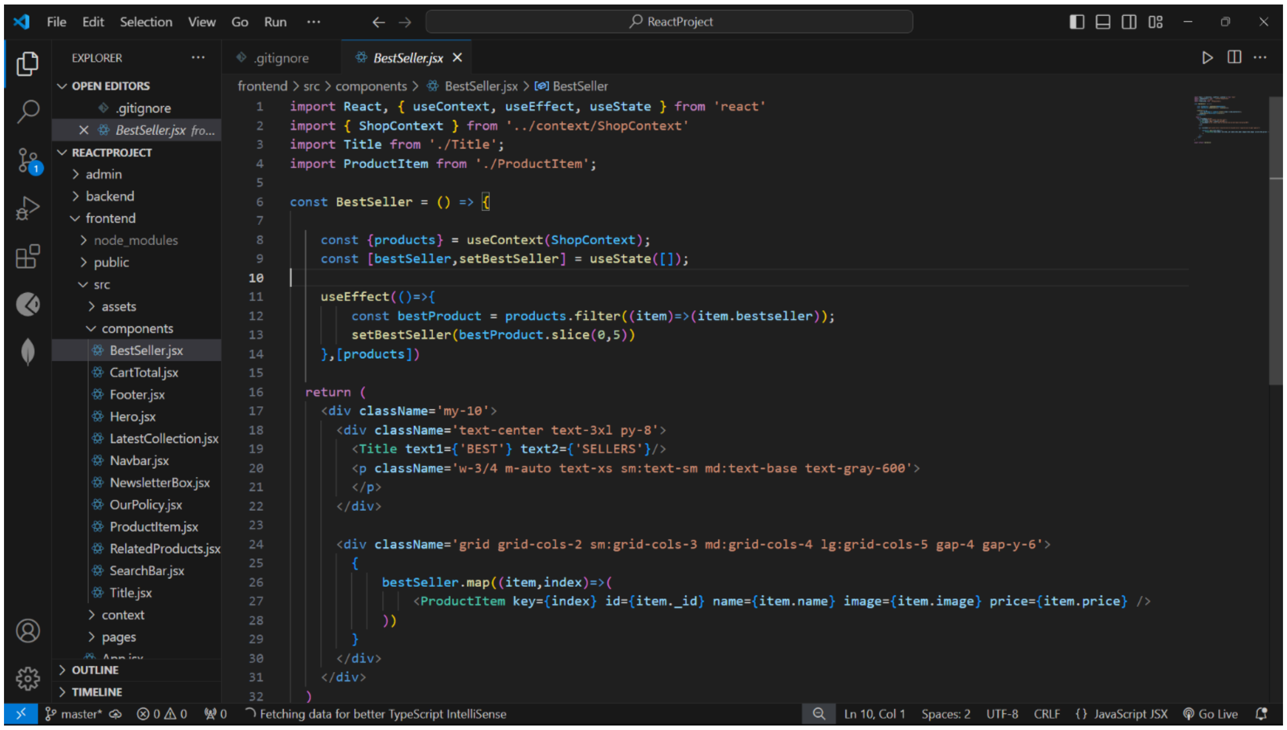
```jsx
import React, { useContext, useEffect, useState } from 'react'
import { ShopContext } from '../context/ShopContext'
import Title from './Title';
import ProductItem from './ProductItem';

const RelatedProducts = ({category,subCategory}) => {

    const { products } = useContext(ShopContext);
    const [related,setRelated] = useState([]);

    useEffect(()=>{

        if (products.length > 0) {

            let productsCopy = products.slice();

            productsCopy = productsCopy.filter((item) => category === item.category);
            productsCopy = productsCopy.filter((item) => subCategory === item.subCategory);

            setRelated(productsCopy.slice(0,5));
        }

    },[products])

  return (
    <div className='my-24'>
      <div className=' text-center text-3xl py-2'>
        <Title text1={'RELATED'} text2={"PRODUCTS"} />
      </div>

      <div className='grid grid-cols-2 sm:grid-cols-3 md:grid-cols-4 lg:grid-cols-5 gap-4 gap-y-6'>
        {related.map((item,index)=>(
```


File Edit Selection View Go Run
ReactProject
EXPLORER
.gitignore
Login.jsx
admin > src > components > Login.jsx > ...
(parameter) setToken: any
master*
Ln 1, Col 1 Spaces: 4 UTF-8 CRLF {} JavaScript JSX Go Live

```jsx
import axios from 'axios'
import React, { useState } from 'react'
import { backendUrl } from '../App'
import { toast } from 'react-toastify'

const Login = ({setToken}) => {

    const [email,setEmail] = useState('')
    const [password,setPassword] = useState('')

    const onSubmitHandler = async (e) => {
        try {
            e.preventDefault();
            const response = await axios.post(backendUrl + '/api/user/admin',{email,password})
            if (
                setToken(response.data.token)
            } else {
                toast.error(response.data.message)
            }

        } catch (error) {
            console.log(error);
            toast.error(error.message)
        }
    }

  return (
    <div className='min-h-screen flex items-center justify-center w-full'>
        <div className='bg-white shadow-md rounded-lg px-8 py-6 max-w-md'>
            <h1 className='text-2xl font-bold mb-4'>Admin Panel</h1>
            <form onSubmit={onSubmitHandler}>
                <div className='mb-3 min-w-72'>
```


File Edit Selection View Go Run
ReactProject
EXPLORER
.gitignore
Navbar.jsx
frontend > src > components > Navbar.jsx > Navbar
master*
Ln 58, Col 24 Spaces: 4 UTF-8 CRLF {} JavaScript JSX Go Live

```jsx
import React, { useContext, useState } from 'react'
import {assets} from '../assets/assets'
import { Link, NavLink } from 'react-router-dom'
import { ShopContext } from '../context/ShopContext';

const Navbar = () => {

    const [visible,setVisible] = useState(false);

    const {setShowSearch , getCartCount , navigate, token, setToken, setCartItems} = useContext(ShopContext);

    const logout = () => {
        navigate('/login')
        localStorage.removeItem('token')
        setToken('')
        setCartItems({})
    }

  return (
    <div className='flex items-center justify-between py-5 font-medium'>

      <Link to='/'><img src={assets.logo} className='w-36' alt="" /></Link>

      <ul className='hidden sm:flex gap-5 text-sm text-gray-700'>

        <NavLink to='/' className='flex flex-col items-center gap-1'>
            <p>HOME</p>
            <hr className='w-2/4 border-none h-[1.5px] bg-gray-700 hidden' />
        </NavLink>
        <NavLink to='/collection' className='flex flex-col items-center gap-1'>
            <p>COLLECTION</p>
            <hr className='w-2/4 border-none h-[1.5px] bg-gray-700 hidden' />
```



```jsx
import React, { useContext, useEffect, useState } from 'react'
import { ShopContext } from '../context/ShopContext'
import Title from './Title';
import ProductItem from './ProductItem';

const BestSeller = () => {

    const {products} = useContext(ShopContext);
    const [bestSeller,setBestSeller] = useState([]);

    useEffect(()=>{
        const bestProduct = products.filter((item)=>(item.bestseller));
        setBestSeller(bestProduct.slice(0,5))
    },[products])

  return (
    <div className='my-10'>
      <div className='text-center text-3xl py-8'>
        <Title text1={'BEST'} text2={'SELLERS'}/>
        <p className='w-3/4 m-auto text-xs sm:text-sm md:text-base text-gray-600'>
        </p>
      </div>

      <div className='grid grid-cols-2 sm:grid-cols-3 md:grid-cols-4 lg:grid-cols-5 gap-4 gap-y-6'>
        {
            bestSeller.map((item,index)=>(
                <ProductItem key={index} id={item._id} name={item.name} image={item.image} price={item.price} />
            ))
        }
      </div>
    </div>
  )
```

LINKS

GitHub :

https://github.com/UtkarshSingh83/Clothing-Website-E-commerce

Deployment Link also in Github :

https://github.com/UtkarshSingh83/Clothing-Website-E-commerce

REFERENCES

- **React Documentation**
React - A JavaScript library for building user interfaces.
Retrieved from https://react.dev/

- **Node.js Documentation**
Node.js - JavaScript runtime built on Chrome's V8 JavaScript engine.
Retrieved from https://nodejs.org/

- **MongoDB Documentation**
MongoDB - Document-oriented NoSQL database program.
Retrieved from https://www.mongodb.com/docs/

w3schools

https://www.w3schools.com/REACT/DEFAULT.ASP