PROJECT REPORT

ON

SUSTAINABLE WEBSITE

FOR THE COURSE OF T. Y. B.Sc.

COMPUTER SCIENCE

DESIGNED AND DEVELOPED

BY

RIDDHI SUNIL KASAR

GUIDED BY

Ms. POOJA PATIL

DEPARTMENT OF COMPUTER SCIENCE

PARLE TILAK VIDYALAYA ASSOCIATION’S

MULUND COLLEGE OF COMMERCE (AUTONOMOUS)

*(Affiliated to University of Mumbai)*

2024-25

MULUND (WEST), MUMBAI-400080

MAHARASHTRA, INDIA

**ACKNOWLEDGEMENT**

I have a great pleasure in representing this project report entitled “Sustainable Website” and I grab this opportunity to convey my immense regards towards all the distinguished people who have their valuable contribution in the hour of need.

I like to extend my gratitude to our beloved Principal Dr. Sonali Pednekar for her timely and prestigious guidance.

I take this opportunity to thank Dr. Reena Nagda, Coordinator of the Department and all the professors of the Department of Computer Science of Mulund College Of Commerce, for giving me an opportunity to complete this project and the most needed guidance throughout the duration of the Programme.

I am extremely grateful to my project guide Ms Pooja Patil for her valuable guidance and necessary support during each phase of the project. She was the source of continuous encouragement as each milestone was crossed.

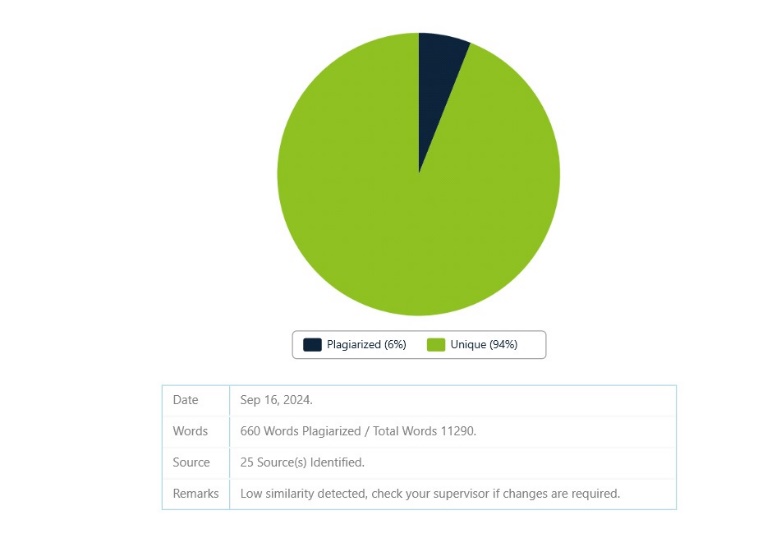
A special thanks to the University Of Mumbai for having prescribed this project work to me as a part of the academic requirement in the Final year of Bachelor of Science in Computer Science.

Finally I also owe to my fellow friends who have been a constant source of help to solve the problems that cropped up during the project development process.

**RIDDHI SUNIL KASAR**

**PLAGIARISM REPORT**

**Plagiarism Checker X Originality Report**



A screenshot of a checker

Description automatically generated

A close-up of a certificate

Description automatically generated

# **INDEX**

**Contents**

[**INDEX** 4](#_Toc177831756)

[**1.Title** 1](#_Toc177831757)

[**1.1 Title of the Project** 1](#_Toc177831758)

[**1.2 Type of the Project** 1](#_Toc177831759)

[**1.3 Developed By** 1](#_Toc177831760)

[**2. Introduction** 2](#_Toc177831761)

[**2.1 Objective of the project** 2](#_Toc177831762)

[**2.2 Description of the proposed system** 3](#_Toc177831763)

[**2.3 Limitation of the current system** 3](#_Toc177831764)

[**2.5 Advantages of the proposed system** 4](#_Toc177831765)

[**3. Requirement Specification** 6](#_Toc177831766)

[**3.1 Software Requirements** 6](#_Toc177831767)

[**3.3 Data Requirements** 6](#_Toc177831768)

[**3.4 Fact Finding Questions:** 7](#_Toc177831769)

[**4. System Design** 9](#_Toc177831770)

[**4.1 Event Table** 9](#_Toc177831771)

[**4.2 Class Diagram** 10](#_Toc177831772)

[**4.3 Use Case Diagram** 11](#_Toc177831773)

[**4.3.1 User** 11](#_Toc177831774)

[**4.3.2 Seller** 12](#_Toc177831775)

[**4.3.3 Transaction** 12](#_Toc177831776)

[**4.3.4 Buying** 13](#_Toc177831777)

[**4.4 Sequence Diagram** 14](#_Toc177831778)

[**4.5 Activity Diagram** 15](#_Toc177831779)

[**4.5.1 Buyer** 15](#_Toc177831780)

[**4.5.2 Seller** 16](#_Toc177831781)

[**4.6 State Diagram** 17](#_Toc177831782)

[**4.6.1 User side** 17](#_Toc177831783)

[**4.6.2 Seller side** 18](#_Toc177831784)

[**4.7 Package Diagram** 19](#_Toc177831785)

[**4.8 Component Diagram** 20](#_Toc177831786)

[**4.9 Deployment Diagram** 21](#_Toc177831787)

[**4.10 Database Design** 21](#_Toc177831788)

[**4.10.1 Database Design- product design** 22](#_Toc177831789)

[**4.10.2 Database Design- sales design** 22](#_Toc177831790)

[**4.10.2 Database Design- users design** 23](#_Toc177831791)

[**5. System Implementation** 24](#_Toc177831792)

[**5.1 aboutus.html** 24](#_Toc177831793)

[**5.2 accessories.html** 26](#_Toc177831794)

[**5.3 buy.html** 27](#_Toc177831795)

[**5.4 cart.html** 28](#_Toc177831796)

[**5.5 Clothing.html** 29](#_Toc177831797)

[**5.5 Donate.html** 32](#_Toc177831798)

[**5.6 Homepage.html** 33](#_Toc177831799)

[**5.7 home\_seller.html** 34](#_Toc177831800)

[**5.8 manage-products.html** 36](#_Toc177831801)

[**5.9 member.html** 40](#_Toc177831802)

[**5.10 polls.html** 42](#_Toc177831803)

[**5.11 profile.html** 45](#_Toc177831804)

[**5.12 sales\_report.html** 49](#_Toc177831805)

[**5.13 index.html** 50](#_Toc177831806)

[**5.14 signup.html** 53](#_Toc177831807)

[**5.15 startpage.html** 56](#_Toc177831808)

[**5.16 sustainable.html** 58](#_Toc177831809)

[**5.17 test.html** 59](#_Toc177831810)

[**5.18 traffic\_details.html** 60](#_Toc177831811)

[**5.19 whatsnew.html** 63](#_Toc177831812)

[**5.20 home\_seller.js** 64](#_Toc177831813)

[**5.21 sales\_report.js** 65](#_Toc177831814)

[**5.22 accessories.js** 67](#_Toc177831815)

[**5.23 buy.js** 70](#_Toc177831816)

[**5.24 cart.js** 72](#_Toc177831817)

[**5.25 clothing.js** 74](#_Toc177831818)

[**5.26 sustainable.js** 76](#_Toc177831819)

[**5.27 startpage.js** 78](#_Toc177831820)

[**5.28 allowedOrigin.js** 78](#_Toc177831821)

[**5.29 corsOption.js** 78](#_Toc177831822)

[**5.30 authController.js** 79](#_Toc177831823)

[**5.31 orderController.js** 81](#_Toc177831824)

[**5.32 productController.js** 82](#_Toc177831825)

[**5.33 userController.js** 84](#_Toc177831826)

[**5.34 authMiddleware.js** 85](#_Toc177831827)

[**5.35 credentials.js** 86](#_Toc177831828)

[**5.36 order.js** 86](#_Toc177831829)

[**5.37 product.js** 87](#_Toc177831830)

[**5.38 user.js** 87](#_Toc177831831)

[**5.39 authRoutes.js** 88](#_Toc177831832)

[**5.40 orderRoutes.js** 88](#_Toc177831833)

[**5.41 productRoutes.js** 88](#_Toc177831834)

[**5.42 userRoutes.js** 89](#_Toc177831835)

[**5.43 server.js** 89](#_Toc177831836)

[**5.44 awarness.html** 90](#_Toc177831837)

[**5.45 forgotpassword.html** 96](#_Toc177831838)

[**6. Results** 102](#_Toc177831839)

[**6.1 Validation and Naming Conventions** 102](#_Toc177831840)

[**6.2 ScreenShots** 103](#_Toc177831841)

[**6.2.1 Login Page** 103](#_Toc177831842)

[**6.2.2 Signup Page** 104](#_Toc177831843)

[**6.2.3 User Home Page** 105](#_Toc177831844)

[**6.3.4 Clothing Page** 105](#_Toc177831845)

[**6.3.5 Accessories Page** 106](#_Toc177831846)

[**6.3.6 Sustainable Items Page** 106](#_Toc177831847)

[**6.3.7 Cart Page** 107](#_Toc177831848)

[**6.3.8 Buy Now Page** 107](#_Toc177831849)

[**6.3.9 Seller Home Page** 108](#_Toc177831850)

[**6.3.10 Manage Products Page** 108](#_Toc177831851)

[**6.3.11 Sales Report Page** 109](#_Toc177831852)

[**6.3.12 Profile Page(Seller and buyer)** 109](#_Toc177831853)

[**6.3.16 About us page** 113](#_Toc177831854)

[**6.3.17 What’s new page** 114](#_Toc177831855)

[**6.3.18 Get involved page** 115](#_Toc177831856)

[**6.3.20 forgotpass token page** 116](#_Toc177831857)

[**6.3.21 successful changepass page** 117](#_Toc177831858)

[**7. Future Enhancements & Conclusion** 118](#_Toc177831859)

[**7.1 Future Enhancement** 118](#_Toc177831860)

[**7.2 Conclusion** 118](#_Toc177831861)

[**8. References** 119](#_Toc177831862)

[**9. Annexures** 120](#_Toc177831863)

[**9.1 Figures list** 120](#_Toc177831864)

[**9.2 Table list** 120](#_Toc177831865)

# **1.Title**

**1.1 Title of the Project:** Sustainable Website

**1.2 Type of the Project:** Website

**1.3 Developed By:** Riddhi Sunil Kasar

# **2. Introduction**

## **2.1 Objective of the project**

* Community Building: Foster a sense of community among local women entrepreneurs through forums, events, and networking opportunities, encouraging collaboration and shared learning.
* Education and Training: Provide resources, workshops, and training programs for sellers to enhance their skills in marketing, product development, and business management, boosting their chances of success.
* Marketing Support: Offer marketing tools and resources to help sellers promote their products effectively, including social media integration and access to promotional campaigns.
* User Feedback and Continuous Improvement: Implement a system for collecting user feedback to continually refine and enhance the platform based on user needs and preferences.
* Mobile Accessibility: Ensure the platform is fully responsive and accessible on mobile devices, catering to a wider audience and improving convenience for both buyers and sellers.
* Secure Transactions: Prioritize secure payment processing and data protection, building trust among users and safeguarding sensitive information.
* Diversity of Product Offerings: Encourage a diverse range of products from various categories to attract a broader customer base and showcase the creativity of local artisans.
* Partnerships with Local Organizations: Collaborate with local NGOs, community groups, and government agencies to provide additional support to women entrepreneurs and promote the platform.
* Analytics and Reporting Tools: Integrate analytics features for sellers to track their performance, sales trends, and customer behavior, enabling informed decision-making.
* Social Responsibility Initiatives: Promote initiatives that give back to the community, such as donating a portion of sales to local causes or engaging in community service projects.
* Environmental Impact Tracking: Implement features that allow sellers to highlight the sustainability of their products, educating consumers on the environmental impact of their purchases.
* Multi-language Support: Offer multi-language options to cater to a diverse customer base and enhance accessibility for sellers and buyers from different backgrounds

## **2.2 Description of the proposed system**

* **Integrated E-Commerce Platform:** The proposed system is a comprehensive e-commerce platform that integrates all key processes, including inventory management, product listings, order processing, and sales tracking, into a single unified interface.
* **Real-Time Data and Inventory Updates:** The system provides real-time updates on inventory levels, sales transactions, and order statuses, ensuring that both sellers and buyers have up-to-date information at all times.
* **Automated Seller and Buyer Workflows:** From order placement to payment processing and shipping notifications, the system automates routine tasks to streamline operations, reduce manual work, and enhance consistency.
* **Centralized Data Management:** All seller and buyer data, including product information, transaction records, and customer feedback, is stored in a centralized database, enabling easy access for analysis, reporting, and decision-making.
* **User-Friendly Interface:** The platform offers a simple and intuitive interface, making it easy for sellers to manage their stores and for buyers to find and purchase products quickly.
* **Secure Payment Processing:** Integrated secure payment gateways ensure safe transactions, protecting both the sellers and buyers from potential fraud.
* **Enhanced Communication Tools:** Built-in communication features facilitate seamless interactions between buyers and sellers, enhancing customer service and reducing response times

## **2.3 Limitation of the current system**

 **Inefficiency and Time-Consuming Processes**: The current reliance on manual management for sales, inventory, and customer communications creates significant inefficiencies that can be detrimental to overall business operations. Sellers often find themselves bogged down by repetitive tasks, such as tracking sales manually, updating inventory records, and responding to customer inquiries individually. This not only leads to wasted time but also detracts from their core activities, such as product design and marketing. The cumulative effect of these inefficiencies can slow down response times to customer requests and diminish the ability to capitalize on emerging market trends.

 **High Potential for Errors**: The manual handling of data inherently increases the risk of human errors, which can have far-reaching consequences. For instance, inaccuracies in inventory counts can lead to stockouts, leaving customers frustrated and resulting in lost sales opportunities. Conversely, overestimating stock can lead to excess inventory that ties up capital and increases storage costs. Additionally, errors in financial tracking can complicate bookkeeping efforts, creating difficulties in reconciling accounts and making it challenging for sellers to gain a clear picture of their financial health. Such inaccuracies not only disrupt day-to-day operations but also undermine trust between sellers and their customers.

 **Limited Scalability**: As the platform expands with more sellers and products, the constraints of manual processes become increasingly evident. The current system lacks the capacity to effectively manage a growing volume of transactions, which can lead to delays and inconsistencies. This limitation not only frustrates existing sellers but also poses a barrier to onboarding new sellers who may require a more streamlined and efficient operational structure. Without scalable solutions, the platform risks stagnation, losing out on potential growth and market share.

 **Inconsistent Customer Experience**: The absence of a standardized system means that buyers may encounter widely varying levels of service quality across different sellers. Some may provide quick responses and timely shipping, while others may be slow to communicate or fulfill orders. This inconsistency can lead to customer dissatisfaction, as buyers might not receive the level of service they expect. Poor experiences can dissuade customers from returning to the platform, ultimately impacting sellers' sales and the platform's reputation in the market. Over time, this can lead to a diminished customer base and lower retention rates.

 **Poor Data Analytics and Insights**: The current lack of a centralized data management system severely hampers the ability to perform effective data analytics. Without comprehensive insights into sales trends, customer preferences, and inventory performance, sellers and platform administrators struggle to make informed decisions. This shortcoming limits their ability to optimize inventory levels, tailor marketing strategies, and enhance customer engagement efforts. As a result, sellers may miss opportunities to target their offerings effectively or identify which products are underperforming. Ultimately, the inability to leverage data insights stifles innovation and growth, putting the platform at a competitive disadvantage in an increasingly data-driven market.

## **2.5 Advantages of the proposed system**

 **Enhanced Efficiency**: The proposed automated system significantly improves efficiency across all operations. By reducing the time and effort required to perform tasks, automation allows sellers to handle a higher volume of transactions and manage their inventory more effectively. This leads to streamlined processes that enhance productivity and accelerate overall business operations, enabling sellers to respond to customer needs more swiftly.

 **Improved Accuracy**: One of the key benefits of the proposed system is the enhancement of accuracy in data management. Automated data entry and processing minimize the risks associated with human errors, ensuring that the information stored is both accurate and reliable. This reduction in errors is crucial for maintaining the integrity of inventory counts, order fulfillment, and financial tracking, fostering greater trust among sellers and their customers.

 **Real-Time Insights**: The system provides immediate access to critical information, specifically showing the quantity of each product sold and the total profit generated by those products. This real-time visibility allows sellers to quickly assess product performance, make informed decisions regarding inventory restocking, and identify profitable items. Such insights enable proactive adjustments to sales strategies and inventory management.

 **Scalability**: An automated system is designed to scale effectively with the growth of the organization. As the number of sellers and products increases, the system can accommodate larger volumes of data and transactions without a decline in performance. This scalability is essential for future-proofing the platform, ensuring it remains effective and efficient as it expands to meet growing demands.

 **Better Customer Experience**: Streamlined operations result in a more seamless customer experience. With real-time data updates and automated workflows, order fulfillment becomes faster and more reliable. Customers benefit from accurate information regarding product availability and timely updates on their orders, enhancing their satisfaction and encouraging repeat business. A consistently positive customer experience is vital for building loyalty and driving long-term success.

 **Centralized Control and Management**: The proposed system centralizes the management of all sales, inventory, and customer data, simplifying the reporting process. This centralized control not only enhances data security but also enables more effective business management. Stakeholders can easily generate reports and gain insights into operations, facilitating better strategic planning and operational efficiency.

 **Increased Revenue Potential**: By optimizing operational processes and enhancing the overall user experience, the platform stands to significantly increase its revenue potential. Streamlined operations help reduce costs, while a better customer experience attracts more buyers and encourages repeat purchases. This combination of factors positions the platform for greater sales success and overall revenue growth.

# **3. Requirement Specification**

## **3.1 Software Requirements**

* Operating System: Microsoft Windows 10 or any other modern operating system (Windows, macOS, Linux).

Application Software:

* Front End HTML: For structuring the web pages,CSS: For styling the website and creating an appealing design,JavaScript: For interactivity and dynamic content.

* Back End: Node.js (optional, if backend features are needed), Express.js (for server-side logic), and MongoDB (for database, if needed).

* Platform: VS Code (or any preferred code editor/IDE).

## **3.3 Data Requirements**

**Product Data:**

* Product ID (\_id): A unique identifier for each product, used to differentiate between products (e.g., 66e6918a3c262b78c8916809).
* Product Name: The name of the product (e.g., "Handwoven Scarf"). This field helps customers identify and search for the product on the platform.
* Product Image: A file path to the product image (e.g., "uploads\1726386570921-scarf.jpg"). This provides a visual representation of the product for potential buyers.
* Product Price: The cost at which the product is sold (e.g., 50). This field is used during transactions and for displaying the price to customers.
* Product Description: A brief description of the product, highlighting key features (e.g., "Soft, handwoven cotton scarf, perfect for all seasons").
* Product Category: The category of the product (e.g., "accessories"). This helps in organizing the product and making it easier for users to browse by category.
* Seller Name: The name of the seller who listed the product (e.g., "Sana's Handmade"). This identifies the creator or vendor of the product.
* Creation Date (createdAt): The date and time when the product was added to the platform (e.g., 2024-09-15T09:35:12.922+00:00). This helps in tracking when the product became available**.**

**Customer Data**:

* Customer ID (\_id): A unique identifier for each customer, used to differentiate between users on the platform.
* Username: The customer’s chosen name for their account, used for identification and login.
* Phone Number: The customer’s contact number, used for communication and verification.
* Email: The email address provided during registration, used for account notifications and login.
* Country: The country of residence for the customer.
* Password: A securely hashed password to protect the customer’s account.
* Role: Defines whether the customer is a buyer or a seller on the platform.
* Creation Date (created At): The timestamp of when the customer’s account was created.
* Version (\_\_v): A field used to track the number of modifications to the customer record in the database.

**Sales Data**:

* Sales ID (\_id): A unique identifier for each sales transaction, used for tracking individual sales.
* Seller ID (sellerId): Identifies the seller associated with the sale, linking the transaction to a specific seller.
* Product Name (productName): The name of the product sold in the transaction.
* Price: The price at which the product was sold during the transaction.
* Category: The category of the product sold (e.g., Clothing, Accessories).
* Quantity: The number of units of the product sold in the transaction.
* Date: The date and time when the transaction took place.
* Version (\_\_v): A field used for version control in the database.3.4 Fact finding questions

## **3.4 Fact Finding Questions:**

* What are the main goals and objectives of implementing NexusBloom’s platform for local women’s businesses?
* How are products currently listed and sold on the platform? (Manual, existing online system, etc.)
* How do you currently manage product inventory and track sales for sellers on NexusBloom?
* What data needs to be shared between the e-commerce platform and other external systems (e.g., payment gateways, shipping services)?
* How do you currently handle customer profiles, product details, and seller data management?
* What product and inventory data need to be tracked to ensure smooth platform operations?
* What customer information do you collect and need to access during the checkout process?
* How flexible does NexusBloom need to be to accommodate future updates, new product categories, or additional sellers?

# **4. System Design**

## **4.1 Event Table**

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| **No** | **Events** | **Trigger** | **Scource** | **Activity** | **Response** | **Destination** |
| **1** | **User Sign-Up** | **Sign-Up button** | **User** | **Entering personal details and creating an account** | **User account will be created and user will be navigated to the login page** | **Login Page** |
| **2** | **User Login** | **Login button** | **User** | **Entering username and password** | **User will be authenticated and navigated to the dashboard** | **Dashboard** |
| **3** | **Browse Products** | **Category Selection** | **User** | **Selecting a product category** | **Display products in the selected category** | **Product Listing Page** |
| **4** | **Add to Cart** | **Add to Cart button** | **User** | **Clicking on the add to cart button** | **Product will be added to the shopping cart** | **Shopping Cart** |
| **5** | **Direct Purchase** | **Buy Now button** | **User** | **Clicking the buy now button** | **Product will be purchased and user will proceed to checkout** | **Checkout Page** |
| **6** | **Enter Personal Details** | **Checkout Form** | **User** | **Filling out name, address, and email** | **Purchase will be processed and transaction details saved** | **Order Confirmation** |
| **7** | **Edit Profile** | **Edit Profile button** | **User** | **Clicking the edit profile button** | **Profile editing interface will be displayed** | **Edit Profile Page** |
| **8** | **Save Profile Changes** | **Save button** | **User** | **Clicking the save button after editing profile** | **Profile details will be updated (email, contact, country, bio, profile picture)** | **Profile Page** |
| **9** | **Home** | **Home button** | **User** | **Clicking the home button** | **User will be navigated to the home page** | **Home Page** |
| **10** | **Back** | **Back button** | **User** | **Clicking the back button** | **User will be navigated back to the previous page** | **Previous Page** |

Table 1 Event Table

## **4.2 Class Diagram**

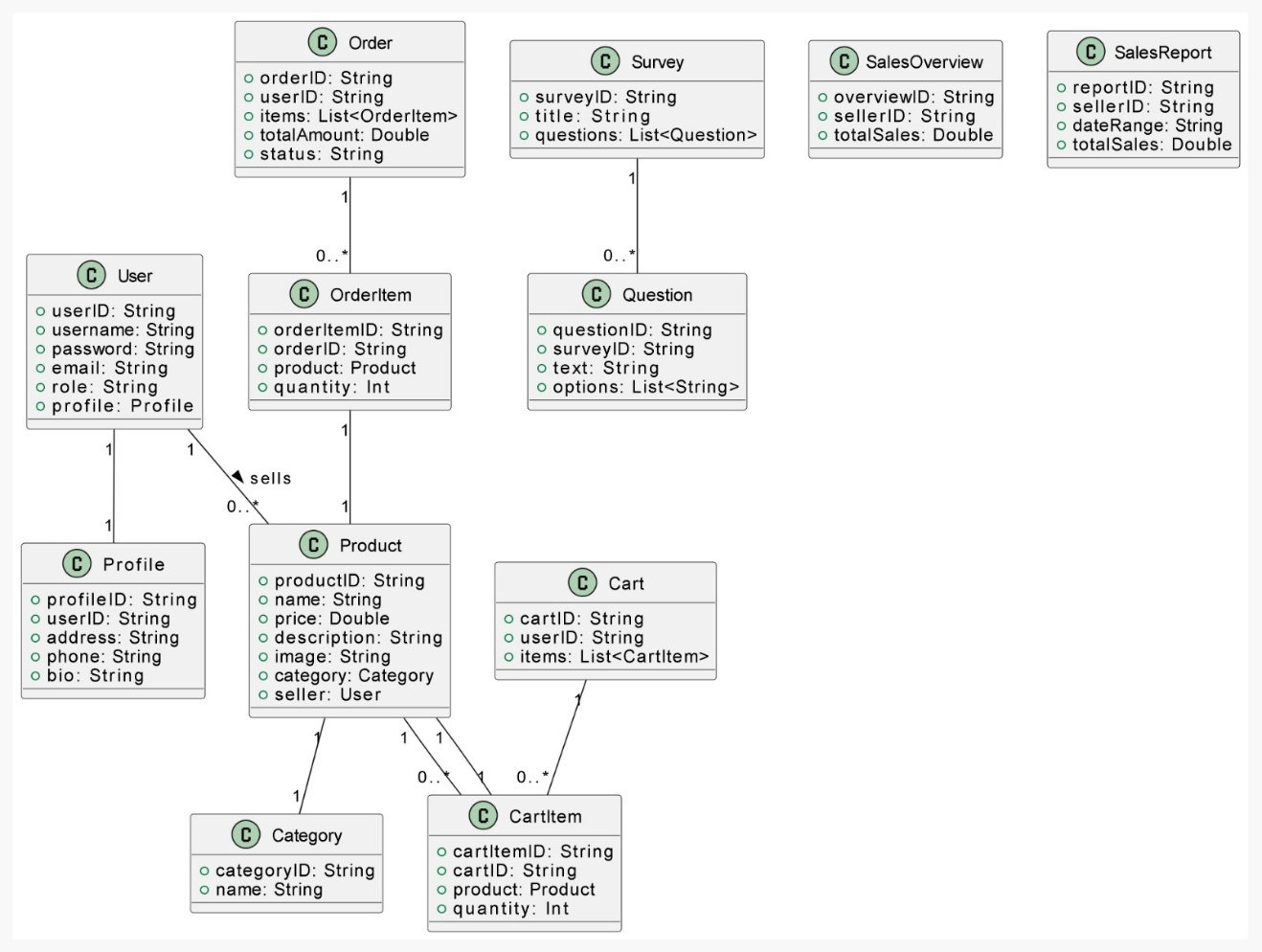


Figure 1 Class Diagram

## **4.3 Use Case Diagram**

### **4.3.1 User**

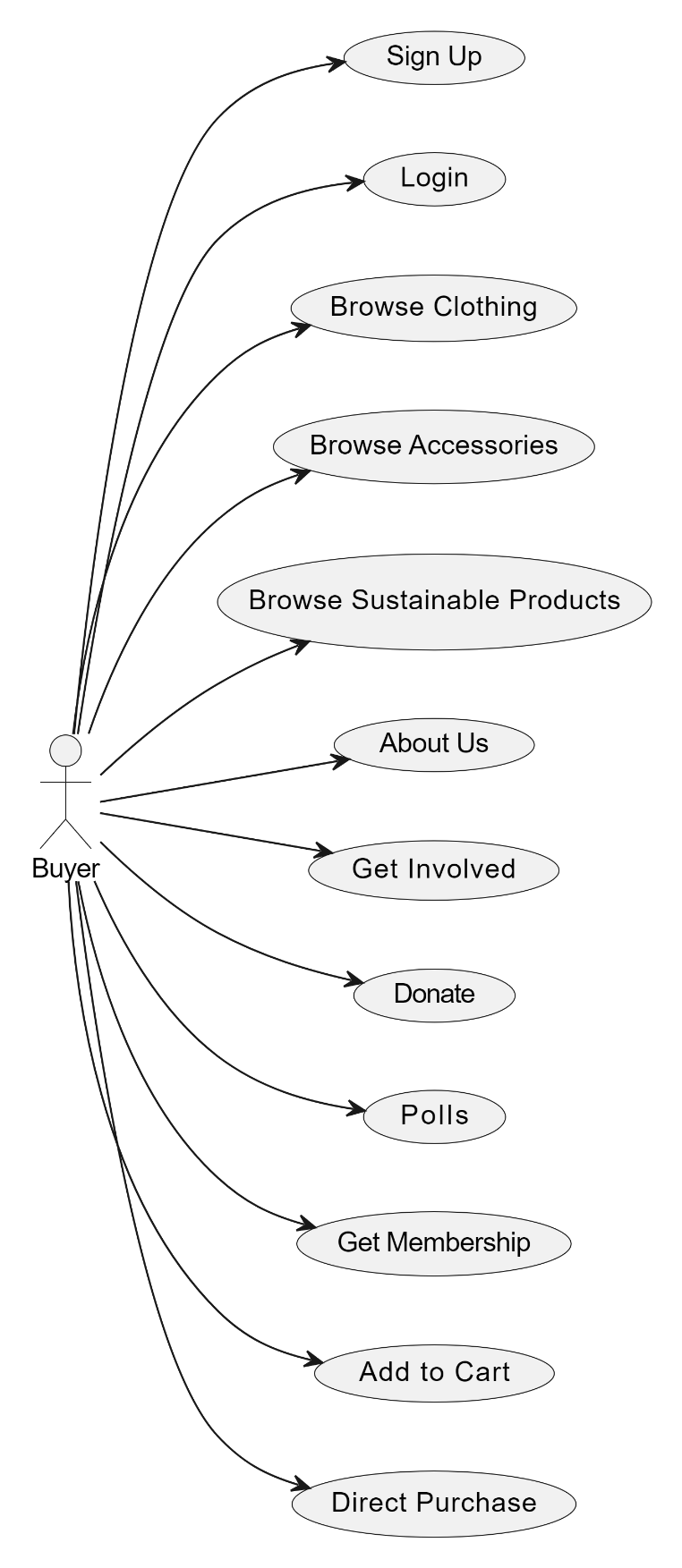


Figure 2 Class Diagram

### **4.3.2 Seller**

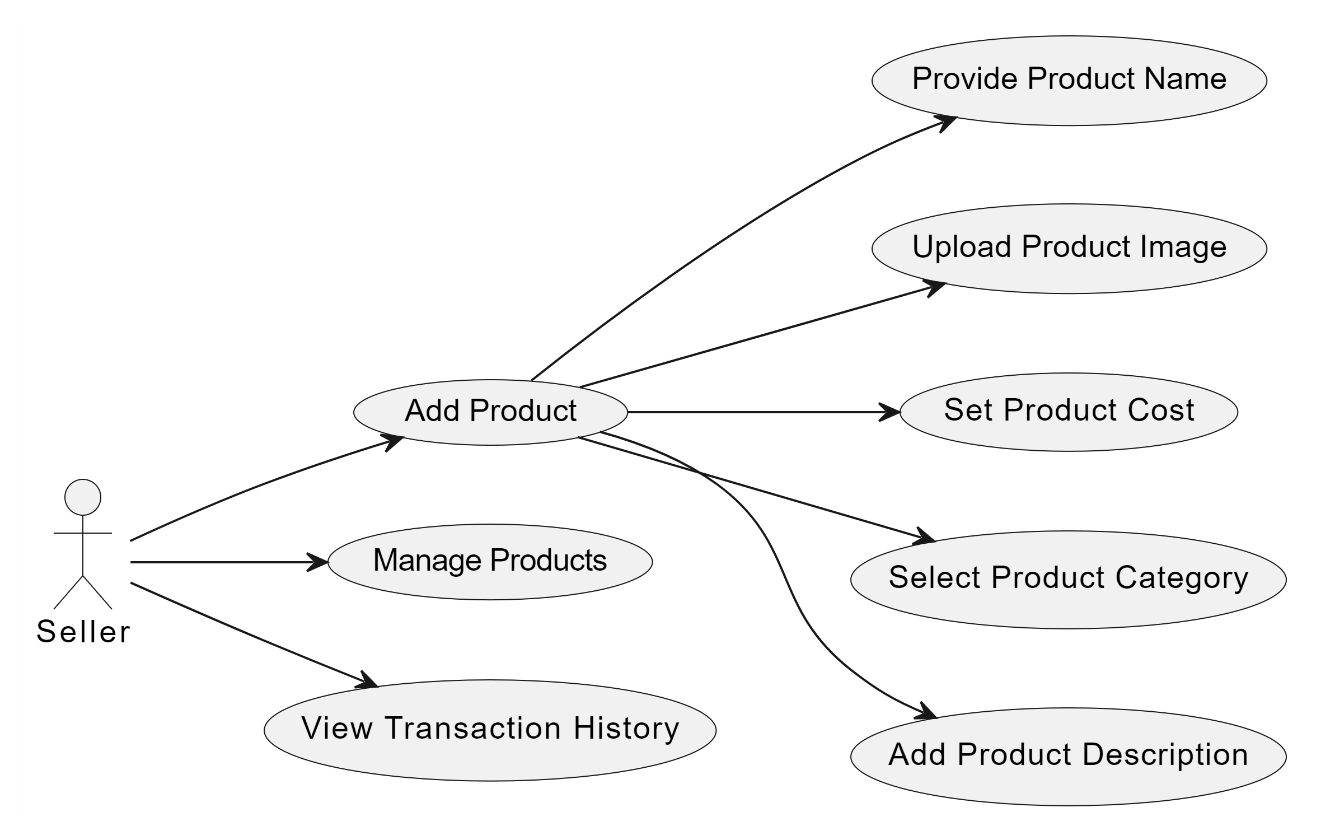


Figure 3 Use case diagram - seller

### **4.3.3 Transaction**

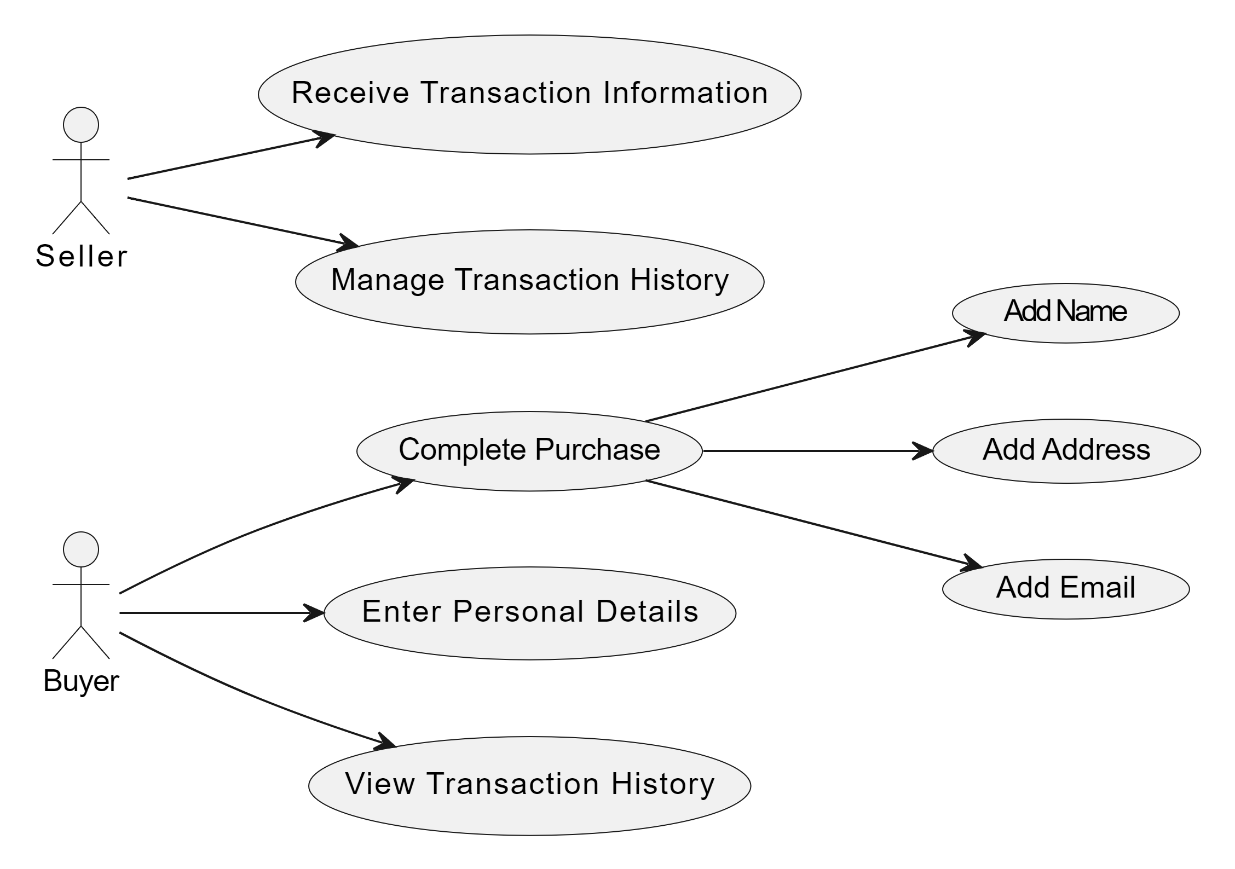


Figure no 4 Use case Diagram – Transaction

### **4.3.4 Buying**

A diagram of a product

Description automatically generated

Figure 5 Use Case Diagram -Buying Process

## **4.4 Sequence Diagram**

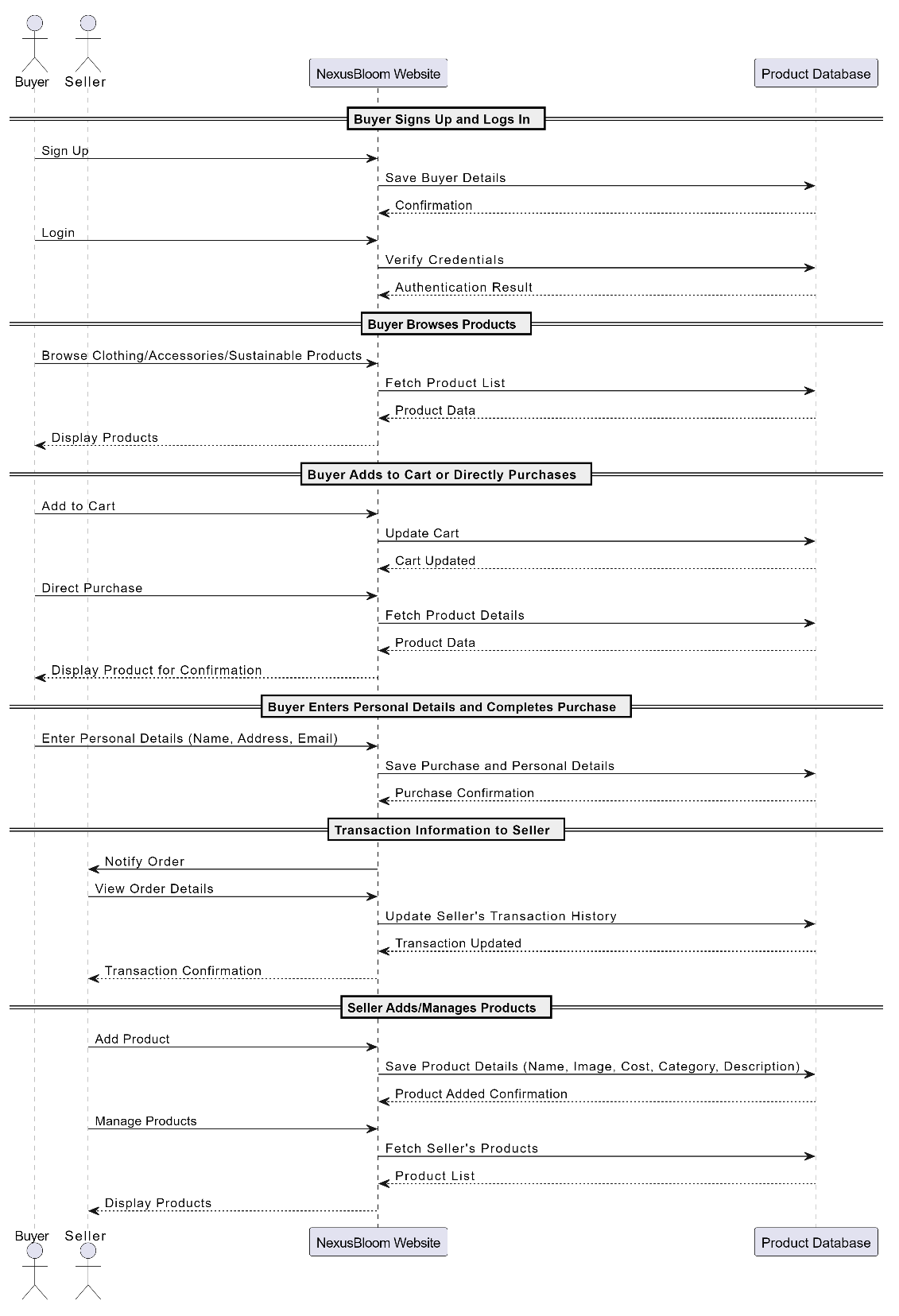


Figure 6 sequence diagram

## **4.5 Activity Diagram**

### **4.5.1 Buyer**

A diagram of a company

Description automatically generated with medium confidence

Figure 7 Activity Diagram – Buyer

### **4.5.2 Seller**

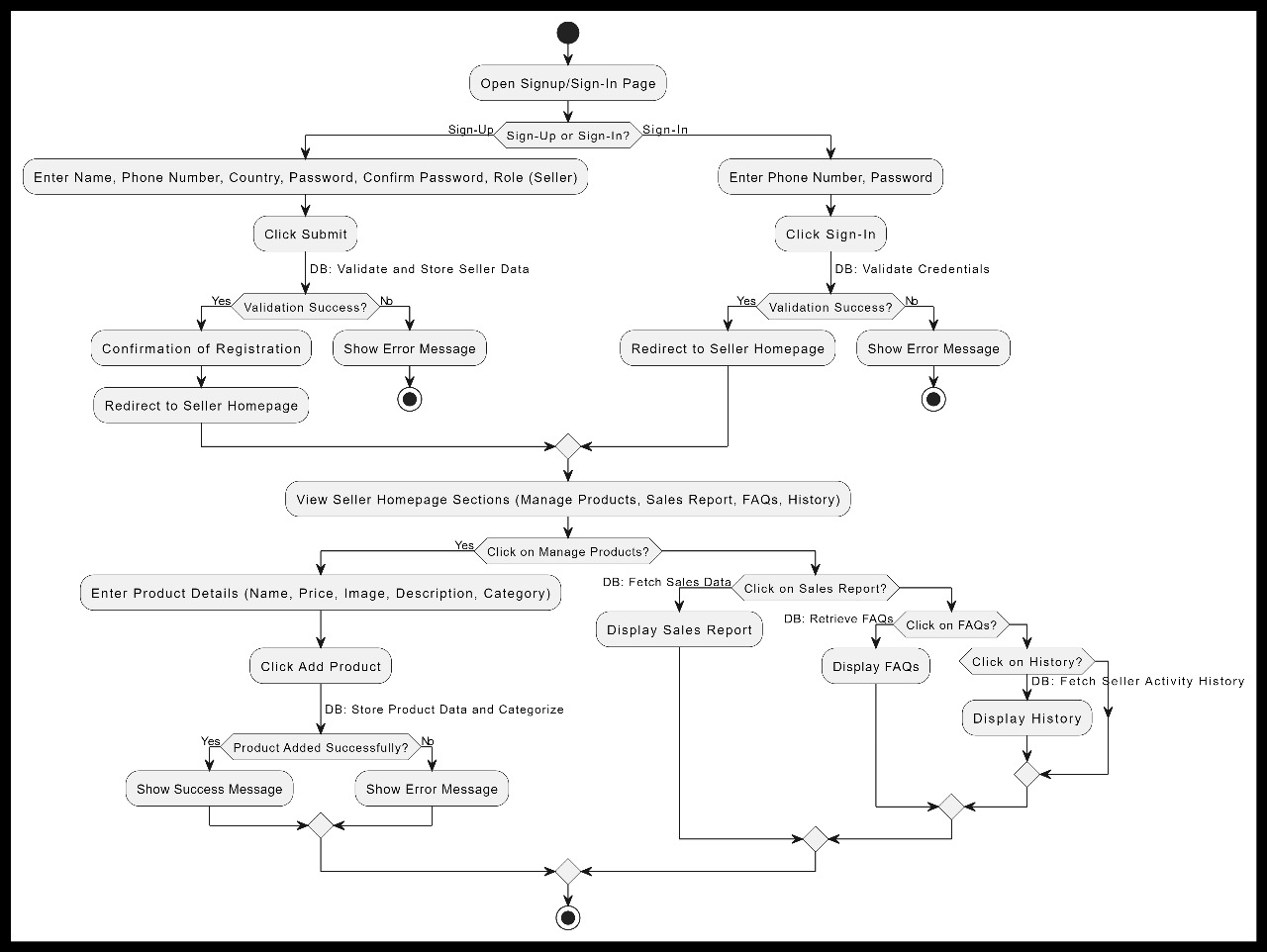


Figure 8 Activity Diagram -Seller

## **4.6 State Diagram**

### **4.6.1 User side**

A diagram of a diagram

Description automatically generated

Figure 9 State Diagram - user

### **4.6.2 Seller side**

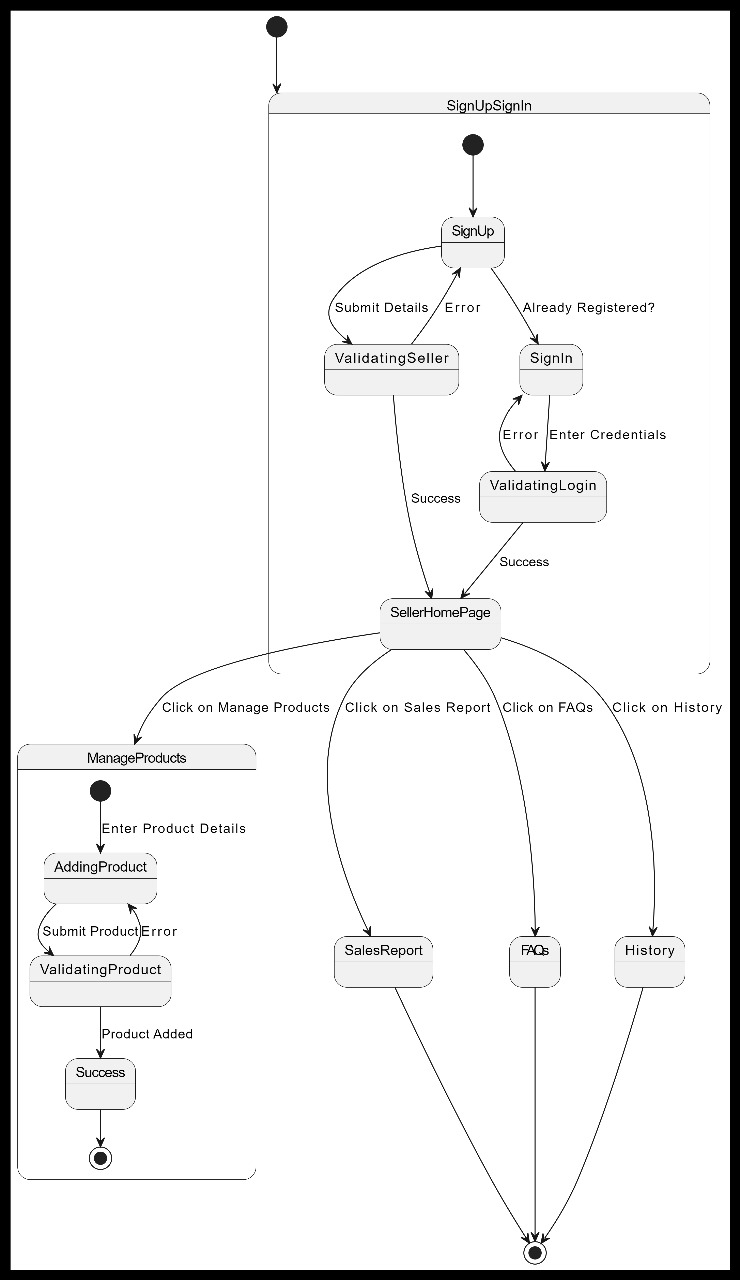


Figure 10 State Diagram- seller side

## **4.7 Package Diagram**

A diagram of a diagram

Description automatically generated

Figure 11 Package Diagram

## **4.8 Component Diagram**

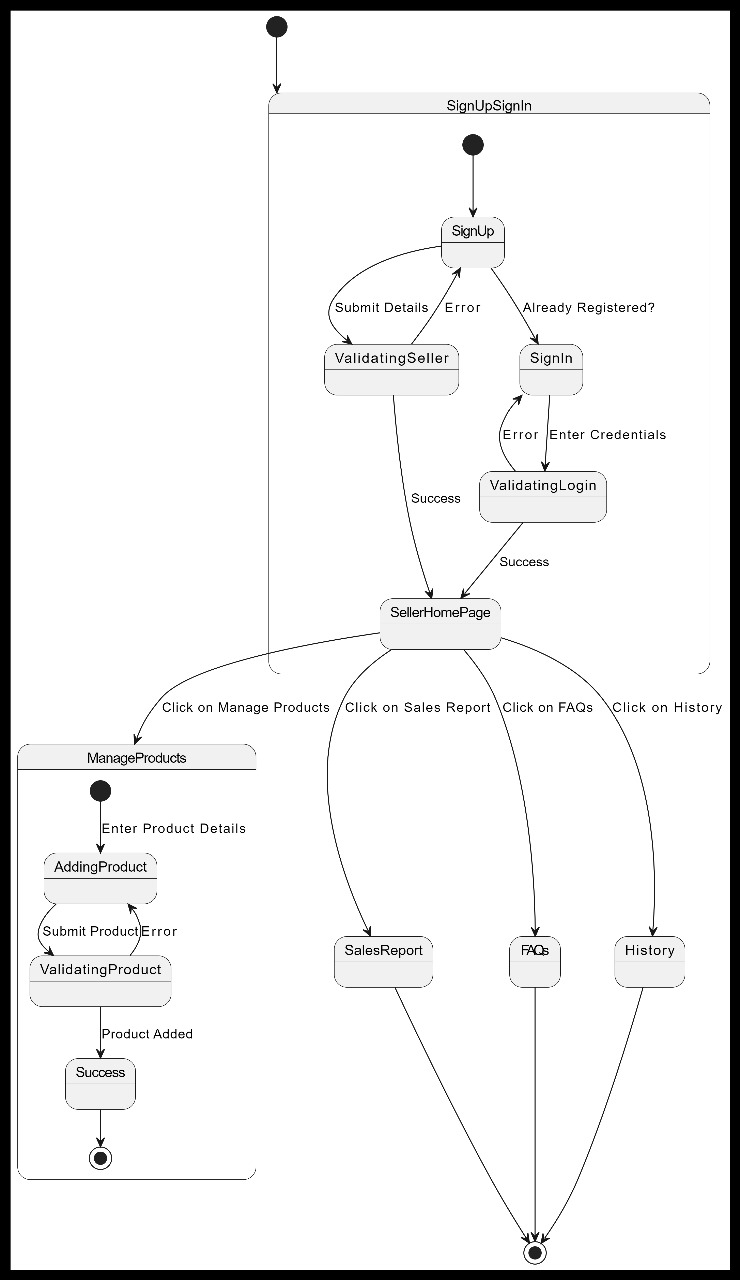


Figure 11 Component diagram

## **4.9 Deployment Diagram**

A diagram of a software flow

Description automatically generated

Figure 12 Deployment Diagram

**4.10 Database Design**A screenshot of a computer

Description automatically generated

Figure 13 Database Design

### **4.10.1 Database Design- product design**

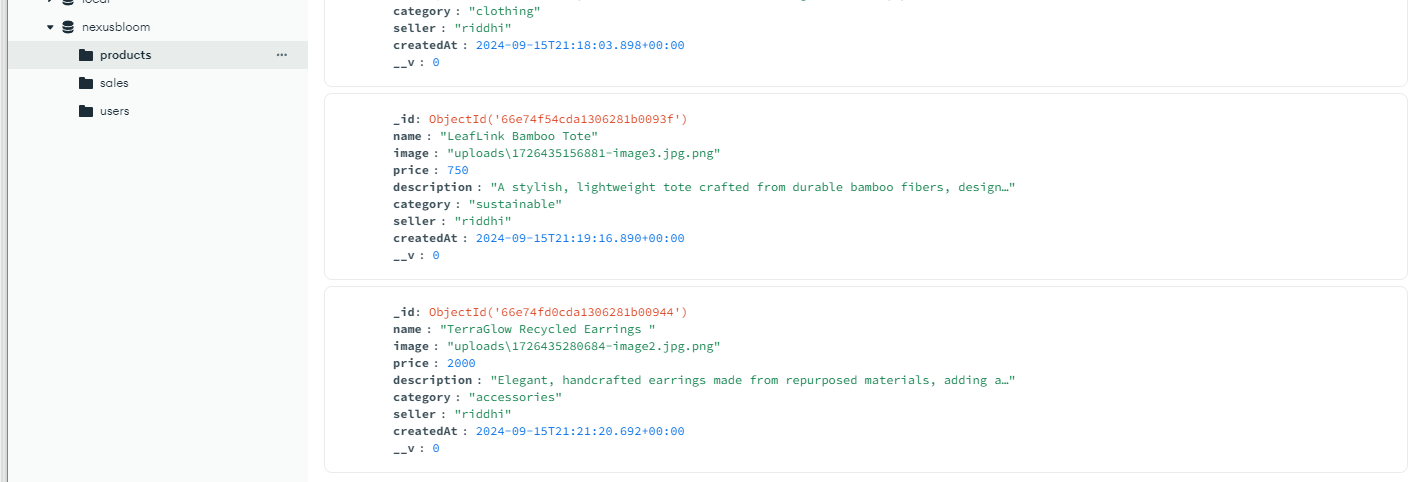


Figure 14 Database Design- Products

### **4.10.2 Database Design- sales design**

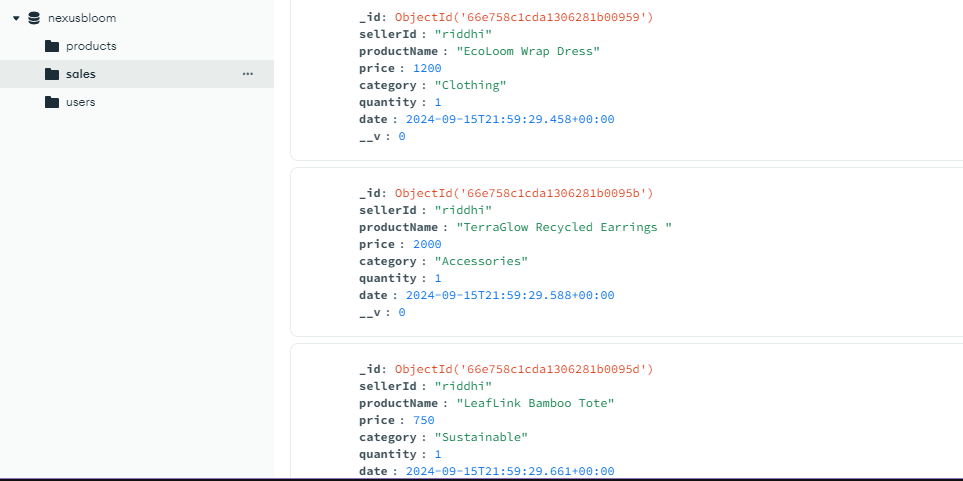


Figure 15 Database Design-sales

### **4.10.2 Database Design- users design**

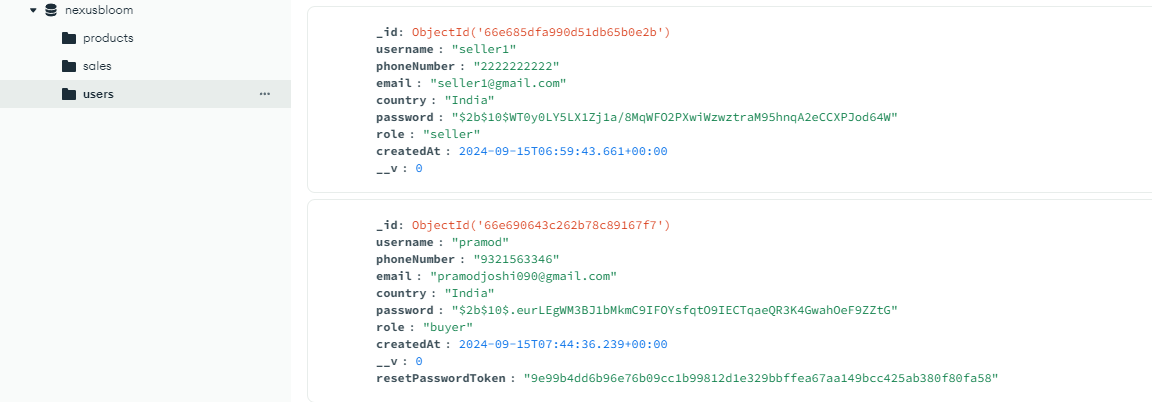


Figure 16 Database Design- users

# **5. System Implementation**

## **5.1 aboutus.html**

<!DOCTYPE html>

<html lang="en">

<head>

    <meta charset="UTF-8">

    <meta name="viewport" content="width=device-width, initial-scale=1.0">

    <title>Nexus Bloom | About Us</title>

    <link rel="stylesheet" href="aboutus.css">

</head>

<body>

    <div class="container">

        <header class="header">

            <h1>About Us</h1>

        </header>

        <section class="intro">

            <p>Welcome to Nexus Bloom, where our mission is to empower and uplift women in all aspects of their lives. Through community, innovation, and unwavering support, we strive to create a world where every woman can thrive.</p>

        </section>

        <section class="mission">

            <h2>Our Mission</h2>

            <p>At Nexus Bloom, our mission is to empower women in the local community by providing them with the tools, resources, and support they need to succeed in business, safety, and health.</p>

        </section>

        <section class="values">

            <h2>Our Values</h2>

            <ul>

                <li>Empowerment: We believe in the strength and potential of every woman.</li>

                <li>Community: We are committed to building a supportive and inclusive community.</li>

                <li>Integrity: We operate with honesty, transparency, and respect.</li>

                <li>Innovation: We strive to provide creative solutions for real-world challenges.</li>

            </ul>

        </section>

        <section class="history">

            <h2>Our Story</h2>

            <p>Nexus Bloom began as a small initiative in 2020, driven by the passion to support and uplift women in our local community. What started as a grassroots movement has grown into a comprehensive platform that addresses the diverse needs of women in business, safety, and health.</p>

        </section>

        <section class="community">

            <h2>Our Community</h2>

            <p>Over the years, we have connected with thousands of women, providing them with opportunities to grow their businesses, enhance their personal safety, and improve their overall well-being. Our community continues to thrive, with over 5 lakhs+ active members today.</p>

        </section>

        <section class="impact">

            <h2>Our Impact</h2>

            <p>Since our inception, Nexus Bloom has positively impacted the lives of countless women. Through our initiatives, we have seen women start and grow their own businesses, take charge of their personal safety, and achieve better health outcomes. We believe in measurable results and are proud to share the stories of success that our community has created.</p>

        </section>

        <section class="team">

            <h2>Meet the Team</h2>

            <p>Our dedicated team is the heart of Nexus Bloom. Comprising passionate individuals from diverse backgrounds, our team works tirelessly to support our mission. Together, we bring a wealth of experience in business development, health, and safety, all aimed at making a lasting impact.</p>

        </section>

        <section class="future">

            <h2>The Road Ahead</h2>

            <p>We are excited about the future and remain committed to expanding our reach, deepening our impact, and continuing to innovate in ways that benefit women everywhere. Together, we can build a brighter, safer, and more prosperous future.</p>

        </section>

        <section class="get-involved">

            <h2>Get Involved</h2>

            <p>Join us in our mission! Whether you want to volunteer, donate, or become a part of our community, there are many ways to get involved. Your contribution, no matter how small, can make a significant difference in the lives of women around you.</p>

            <button onclick="window.location.href='../donate/donate.html'">Donate Now</button>

            <button onclick="window.location.href='volunteer.html'">Volunteer</button>

        </section>

        <div class="back-button">

            <button onclick="window.location.href='../homepage/homepage.html'">Back to Home</button>

        </div>

    </div>

</body>

</html>

## **5.2 accessories.html**

## 

<!DOCTYPE html>

<html lang="en">

<head>

    <meta charset="UTF-8">

    <meta name="viewport" content="width=device-width, initial-scale=1.0">

    <link rel="stylesheet" href="accessories.css">

    <title>Nexus Bloom | Accessories</title>

</head>

<body>

    <header class="header">

        <nav class="nav">

            <ul class="nav-menu">

                <li><a href="../homepage/homepage.html" class="nav-link">Home</a></li>

                <li><a href="../about/aboutus.html" class="nav-link">About Us</a></li>

                <li><a href="../donate/donate.html" class="nav-link">Donate</a></li>

                <li><a href="../profile/profile.html" class="nav-link">Profile</a></li>

                <li><a href="../cart/cart.html" class="nav-link">Cart</a></li> <!-- New Cart Link -->

                <li class="basket">

                    <img src="../assets/basket.png" alt="Basket Icon">

                    <span>Basket: </span><span id="basket-count">0</span> items

                </li>

            </ul>

        </nav>

    </header>

    <div class="main-content">

        <div id="product-grid" class="product-grid">

            <!-- Accessories products will be dynamically inserted here -->

        </div>

    </div>

    <footer class="footer">

        <div class="back-button">

            <button onclick="window.location.href='../homepage/homepage.html'">Back to Home</button>

        </div>

    </footer>

    <script src="../scripts/accessories.js"></script>

</body>

</html>

## **5.3 buy.html**

<!DOCTYPE html>

<html lang="en">

<head>

    <meta charset="UTF-8">

    <meta name="viewport" content="width=device-width, initial-scale=1.0">

    <link rel="stylesheet" href="buy.css">

    <title>Nexus Bloom | Buy Now</title>

</head>

<body>

    <header class="header">

        <nav class="nav">

            <ul class="nav-menu">

                <li><a href="../homepage/homepage.html" class="nav-link">Home</a></li>

                <li><a href="../about/aboutus.html" class="nav-link">About Us</a></li>

                <li><a href="../donate/donate.html" class="nav-link">Donate</a></li>

                <li><a href="../profile/profile.html" class="nav-link">Profile</a></li>

                <li class="basket">

                    <img src="../assets/basket.png" alt="Basket Icon">

                    <span>Basket: </span><span id="basket-count">0</span> items

                </li>

            </ul>

        </nav>

    </header>

    <main class="buy-now-content">

        <div class="order-summary">

            <h1>Order Summary</h1>

            <div id="order-items" class="product-grid">

                <!-- Order items will be dynamically inserted here -->

            </div>

            <div class="order-total">

                <h3>Total: $<span id="total-price">0.00</span></h3>

            </div>

        </div>

        <form id="buy-now-form" class="order-form">

            <h2>Billing Information</h2>

            <label for="full-name">Full Name:</label>

            <input type="text" id="full-name" name="fullName" required>

            <label for="email">Email:</label>

            <input type="email" id="email" name="email" required>

            <label for="address">Shipping Address:</label>

            <textarea id="address" name="address" required></textarea>

            <label for="payment-method">Payment Method:</label>

            <select id="payment-method" name="paymentMethod" required>

                <option value="credit-card">Credit Card</option>

                <option value="paypal">PayPal</option>

                <option value="bank-transfer">Bank Transfer</option>

            </select>

            <!-- Hidden input for seller ID -->

            <input type="hidden" id="seller-id" name="sellerId" value="">

            <button type="submit">Complete Purchase</button>

        </form>

    </main>

    <footer class="footer">

        <div class="back-button">

            <button onclick="window.location.href='../cart/cart.html'">Back to Cart</button>

        </div>

    </footer>

    <script src="../scripts/buy.js"></script>

</body>

</html>

export default AddUser

## **5.4 cart.html**

<!DOCTYPE html>

<html lang="en">

<head>

    <meta charset="UTF-8">

    <meta name="viewport" content="width=device-width, initial-scale=1.0">

    <link rel="stylesheet" href="cart.css">

    <title>Nexus Bloom | Cart</title>

</head>

<body>

    <header class="header">

        <nav class="nav">

            <ul class="nav-menu">

                <li><a href="../homepage/homepage.html" class="nav-link">Home</a></li>

                <li><a href="../about/aboutus.html" class="nav-link">About Us</a></li>

                <li><a href="../donate/donate.html" class="nav-link">Donate</a></li>

                <li><a href="../profile/profile.html" class="nav-link">Profile</a></li>

                <li class="basket">

                    <img src="../assets/basket.png" alt="Basket Icon">

                    <span>Basket: </span><span id="basket-count">0</span> items

                </li>

            </ul>

        </nav>

    </header>

    <main class="cart-content">

        <div class="cart-container">

            <div id="cart-items" class="product-grid">

                <!-- Cart items will be dynamically inserted here -->

            </div>

            <div class="bill-details">

                <h2>Order Summary</h2>

                <div id="bill-items">

                    <!-- Bill items will be dynamically inserted here -->

                </div>

                <div class="total">

                    <h3>Total: $<span id="total-price">0.00</span></h3>

                </div>

                <button class="buy-now-button" onclick="window.location.href='../buy/buy.html'">Buy Now</button>

            </div>

        </div>

    </main>

    <footer class="footer">

        <div class="back-button">

            <button onclick="window.location.href='../homepage/homepage.html'">Back to Home</button>

        </div>

    </footer>

    <script src="../scripts/cart.js"></script>

</body>

</html>

## **5.5 Clothing.html**

<!DOCTYPE html>

<html lang="en">

<head>

    <meta charset="UTF-8">

    <meta name="viewport" content="width=device-width, initial-scale=1.0">

    <link rel="stylesheet" href="clothing.css">

    <title>Nexus Bloom | Clothing Items</title>

</head>

<body>

    <header class="header">

        <nav class="nav">

            <ul class="nav-menu">

                <li><a href="../homepage/homepage.html" class="nav-link">Home</a></li>

                <li><a href="../about/aboutus.html" class="nav-link">About Us</a></li>

                <li><a href="../donate/donate.html" class="nav-link">Donate</a></li>

                <li><a href="../profile/profile.html" class="nav-link">Profile</a></li>

                <li><a href="../cart/cart.html" class="nav-link">Cart</a></li>

                <li class="basket">

                    <img src="../assets/basket.png" alt="Basket Icon">

                    <span>Basket: </span><span id="basket-count">0</span> items

                </li>

            </ul>

        </nav>

    </header>

    <div class="main-content">

       <div class="product-grid" id="product-grid">

            <!-- Clothing products will be dynamically inserted here -->

        </div>

    </div>

    <footer class="footer">

        <div class="back-button">

            <button onclick="window.location.href='../homepage/homepage.html'">Back to Home</button>

        </div>

    </footer>

    <script src="../scripts/clothing.js"></script>

</body>

</html>

const data = await response.json();

console.log(data);

setPurchaseData(data.purchaseData);

setPurchaseChartData(data.aggregatedPurchaseData)

} catch (error) {

console.error('Error fetching purchase data:', error);

}

};

return (

<div className={styles.purchase}>

<Sidebar />

<main className={styles.main}>

<h1>Purchase</h1>

<div className={styles.filter}>

<div className={styles.from}>

<p>From</p>

<input

type="date"

value={startDate}

onChange={(e) => setStartDate(e.target.value)}

/>

</div>

<div className={styles.to}>

<p>To</p>

<input

type="date"

value={endDate}

onChange={(e) => setEndDate(e.target.value)}

/>

</div>

<button onClick={handleFilter}>Filter</button>

</div>

<div className={styles.content}>

<div className={styles.charts}>

<div className={styles.chart}>

<PurchaseChart purchaseChartData={purchaseChartData} />

</div>

</div>

<div className={styles.purchaseTable}>

<div className={styles.productFilter}>

<select

onChange={(e) => {

setSelectedProduct(e.target.value);

}}

>

<option value="all">All</option>

{products.length !== 0 &&

products.map((product, i) => {

return (

<option key={i} value={product.productId}>

{product.productName}

</option>

);

})}

</select>

</div>

<PurchaseTransactionTable transactions={purchaseData} />

</div>

</div>

</main>

</div>

);

};

export default Purchase;

## **5.5 Donate.html**

<!DOCTYPE html>

<html lang="en">

<head>

    <meta charset="UTF-8">

    <meta name="viewport" content="width=device-width, initial-scale=1.0">

    <title>Nexus Bloom | Donate</title>

    <link rel="stylesheet" href="donate.css">

</head>

<body>

    <div class="container">

        <header class="header">

            <h1>Donate to Nexus Bloom</h1>

            <p>Your support helps empower women in our community. Every contribution counts!</p>

        </header>

       <form id="donation-form">

            <div class="input-box">

                <label for="name">Your Name:</label>

                <input type="text" id="name" required>

            </div>

            <div class="input-box">

                <label for="amount">Donation Amount:</label>

                <input type="number" id="amount" placeholder="Enter custom amount" required>

            </div>

            <div class="submit-button">

                <button type="submit">Contribute</button>

            </div>

        </form>

        <div class="thank-you-message" id="thank-you-message" style="display: none;">

            <h2>Thank You!</h2>

            <p>Thank you for donating to us, kind human! Your contribution will help empower women in our community.</p>

        </div>

        <div class="contact-info">

            <h3>Need More Information?</h3>

            <p>Email us at <a href="mailto:info@nexusbloom.org">info@nexusbloom.org</a> or call +91 12345 67890.</p>

        </div>

        <div class="button-container">

            <button onclick="goBack()">Back</button> <!-- Back button -->

        </div>

    </div>

    <script>

        function goBack() {

            window.history.back(); // Navigates to the previous page

        }

        document.getElementById('donation-form').addEventListener('submit', function(event) {

            event.preventDefault();

            document.getElementById('donation-form').style.display = 'none';

            document.getElementById('thank-you-message').style.display = 'block';

        });

    </script>

</body>

</html>

## **5.6 Homepage.html**

<!DOCTYPE html>

<html lang="en">

<head>

    <meta charset="UTF-8">

    <meta name="viewport" content="width=device-width, initial-scale=1.0">

    <link href='https://unpkg.com/boxicons@2.1.4/css/boxicons.min.css' rel='stylesheet'>

    <link rel="stylesheet" href="homepage.css">

    <title>Nexus Bloom | Home</title>

</head>

<body>

    <header class="header">

        <nav class="nav">

            <ul class="nav-menu">

                <li><a href="polls.html" class="nav-link">Home</a></li>

                <li><a href="aboutus.html" class="nav-link">About Us</a></li>

                <li><a href="donate.html" class="nav-link">Donate</a></li>

                <li><a href="../profile/profile.html" class="nav-link">Profile</a></li>

            </ul>

        </nav>

    </header>

    <div class="sidebar">

        <ul class="sidebar-menu">

            <li><a href="../polls/polls.html">Polls & Surveys</a></li>

            <li><a href="#">Forums</a></li>

            <li><a href="../membership/memberb.html">Member Benefits</a></li>

            <li><a href="getinvolved.html">Get Involved</a></li>

        </ul>

    </div>

    <div class="content">

        <section class="info">

            <h2>Welcome to Nexus Bloom</h2>

            <p>Discover our curated collections of sustainable clothing and accessories. At Nexus Bloom, we are committed to quality and style while making a positive impact on the environment. Explore our latest arrivals and join our community of like-minded individuals.</p>

        </section>

        <div class="quarters">

            <a href="../clothing/clothing.html" class="content-bar">OUR CLOTHING</a>

            <a href="../accessories/accessories.html" class="content-bar">ACCESSORY ITEMS</a>

            <a href="../sustainable/sustainable.html" class="content-bar">Sustainable Products</a>

            <a href="../whatsnew/whatsnew.html" class="content-bar">What's New</a>

        </div>

    </div>

</body>

</html>

## **5.7 home\_seller.html**

<!DOCTYPE html>

<html lang="en">

<head>

    <meta charset="UTF-8">

    <meta name="viewport" content="width=device-width, initial-scale=1.0">

    <title>Seller Dashboard - Nexus Bloom</title>

    <link rel="stylesheet" href="home\_seller.css">

</head>

<body>

    <div class="dashboard-wrapper">

        <nav class="dashboard-nav">

            <div class="nav-logo">

                <p>Nexus Bloom</p>

            </div>

            <div class="nav-links">

                <a href="../homeseller/home\_seller.html">Dashboard</a>

                <a href="../manageproduct/manage-products.html">Manage Products</a>

                <a href="history\_seller.html">History</a>

                <a href="messages.html">Messages</a>

            </div>

        </nav>

        <div class="dashboard-content">

            <section class="overview-section">

                <header>Sales Overview</header>

                <div class="overview-cards">

                    <div class="card">

                        <h3>Total Sales</h3>

                        <p id="total-sales"></p>

                    </div>

                    <div class="card">

                        <h3>Products Sold</h3>

                        <p id="qty"></p>

                    </div>

                </div>

            </section>

            <section class="inventory-section">

                <header>Manage Inventory</header>

                <div class="inventory-actions">

                    <a href="../manageproduct/manage-products.html" class="action-button">Manage Products</a>

                </div>

            </section>

            <section class="order-management-section">

                <header>Order Management</header>

                <div class="order-list">

                    <!-- Sample order entry -->

                    <div class="order-item">

                        <p>Order #1234</p>

                        <p>Status: Shipped</p>

                        <p>Total: $50</p>

                    </div>

                    <!-- More orders can be listed here -->

                </div>

            </section>

            <section class="analytics-section">

                <header>Sales & Traffic Insights</header>

                <div class="analytics-cards">

                    <div class="card">

                        <h3>Sales Report</h3>

                        <a href="../salesreport/sales\_report.html">View detailed sales reports</a>

                    </div>

                    <div class="card">

                        <h3>Traffic Insights</h3>

                        <a href="../traficdetail/traffic\_details.html">View website traffic details</a>

                    </div>

                </div>

            </section>

            <section class="resources-section">

                <header>Seller Resources</header>

                <div class="resources-list">

                    <a href="#">Guides & FAQs</a>

                </div>

            </section>

        </div>

    </div>

    <script src="./home\_seller.js"></script>

</body>

</html>

## **5.8 manage-products.html**

<!DOCTYPE html>

<html lang="en">

<head>

    <meta charset="UTF-8">

    <meta name="viewport" content="width=device-width, initial-scale=1.0">

    <link rel="stylesheet" href="manage-products.css">

    <title>Nexus Bloom | Manage Products</title>

</head>

<body>

    <header class="header">

        <nav class="nav">

            <ul class="nav-menu">

                <li><a href="../homeseller/home\_seller.html" class="nav-link">Home</a></li>

                <li><a href="add-product.html" class="nav-link">Add Product</a></li>

                <li><a href="../profile/profile.html" class="nav-link">Profile</a></li>

                <li class="logout">

                    <button onclick="window.location.href='../signin/index.html'"class="nav-link">Logout</button>

                </li>

            </ul>

        </nav>

    </header>

    <div class="main-content">

        <h1>Manage Your Products</h1>

        <section class="add-product-section">

            <h2>Add New Product</h2>

            <form id="add-product-form">

                <div class="form-group">

                    <label for="product-name">Product Name:</label>

                    <input type="text" id="product-name" name="product-name" required>

                </div>

                <div class="form-group">

                    <label for="product-description">Description:</label>

                    <textarea id="product-description" name="product-description" required></textarea>

                </div>

                <div class="form-group">

                    <label for="product-price">Price (₹):</label>

                    <input type="number" id="product-price" name="product-price" required>

                </div>

                <div class="form-group">

                    <label for="product-category">Category:</label>

                    <select id="product-category" name="product-category" required>

                        <option value="clothing">Clothing</option>

                        <option value="accessories">Accessories</option>

                        <option value="sustainable">Sustainable Products</option>

                    </select>

                </div>

                <div class="form-group">

                    <label for="product-image">Upload Product Image:</label>

                    <input type="file" id="product-image" name="product-image" accept="image/\*" required>

                </div>

                <button type="submit" class="add-product-btn">Add Product</button>

            </form>

        </section>

        <section id="product-list" class="product-list">

            <!-- Products will be dynamically inserted here -->

        </section>

    </div>

    <footer class="footer">

        <div class="back-button">

            <button onclick="window.location.href='../homeseller/home\_seller.html'">Back to Home</button>

        </div>

    </footer>

    <script>

        function decodeJWT(token) {

            // Split the token into parts: header, payload, and signature

            const [header, payload, signature] = token.split('.');

            // Decode the base64 URL-encoded payload

            const decodedPayload = JSON.parse(atob(payload.replace(/-/g, '+').replace(/\_/g, '/')));

            // Optionally, you can decode the header in a similar way

            const decodedHeader = JSON.parse(atob(header.replace(/-/g, '+').replace(/\_/g, '/')));

            return { header: decodedHeader, payload: decodedPayload };

        }

    document.addEventListener('DOMContentLoaded', function() {

        const {payload} = decodeJWT(localStorage.getItem('token'))

        console.log(payload.username)

    // Fetch and display products on page load

    fetchProductsBySeller("yeay","ohnooo");

    // Handle form submission

    document.getElementById('add-product-form').addEventListener('submit', async function(event) {

    event.preventDefault();

    // Create a FormData object to handle file uploads

    document.getElementById('add-product-form').addEventListener('submit', async function(event) {

        event.preventDefault();

        // Create a FormData object to handle file uploads

        const selln = localStorage.getItem('uname');

        const formData = new FormData();

        formData.append('name', document.getElementById('product-name').value);

        formData.append('description', document.getElementById('product-description').value);

        formData.append('price', document.getElementById('product-price').value);

        formData.append('category', document.getElementById('product-category').value);

        formData.append('seller', selln);

        formData.append('image', document.getElementById('product-image').files[0]); // Append the file

        // Send the form data to the backend

        const response = await fetch('http://localhost:3000/products', {

            method: 'POST',

            body: formData,

            headers: {

                'Authorization': `Bearer ${localStorage.getItem('token')}` // Authentication token

            }

        });

        const data = await response.json();

        if (data.message === 'Product created successfully') {

            // Clear the form and fetch the updated product list

           // this.reset();

            fetchProductsBySeller();

            alert(data.message)

        } else {

            alert('Error: ' + data.message);

        }

    });

});

    // Event delegation for edit and delete buttons

    /\*document.getElementById('product-list').addEventListener('click', function(event) {

        if (event.target.classList.contains('edit-btn')) {

            // Edit product logic here

            alert('Edit functionality to be implemented.');

        }

        if (event.target.classList.contains('delete-btn')) {

            // Delete product logic here

            const row = event.target.closest('tr');

            row.remove();

        }

    });\*/

    // Fetch and display products for the current seller

    function fetchProductsBySeller() {

    // Retrieve token from local storage

    const token = localStorage.getItem('token');

    if (!token) {

        console.log('No token found in local storage');

        return;

    }

    // Decode the token to extract seller ID (Assuming JWT token structure)

    const payload = JSON.parse(atob(token.split('.')[1]));

    const sellerId = payload.username;

    if (!sellerId) {

        console.log('No seller ID found in token');

        return;

    }

    // Fetch products for the specific seller

  fetch(`http://localhost:3000/products/seller/${sellerId}`)

    .then(response => response.json())

    .then(data => {

        const productList = document.getElementById('product-list');

        if (Array.isArray(data)) {

            console.log(data);

            data.forEach(product => {

                // Create a new product item

                const productItem = document.createElement('div');

                productItem.classList.add('product-item');

                // Set product details

                productItem.innerHTML = `

                    <img src="${product.image.replace('\\', '/')}">

                    <h2>${product.name}</h2>

                    <p>${product.description}</p>

                    <p class="price">$${product.price}</p>

                `;

                // Append the product item to the product list

                productList.appendChild(productItem);

            });

        } else {

            console.log('Invalid data format received');

        }

    })

    .catch(error => console.log('Error fetching products:', error));

}

});

    </script>

**</body>**

**</html>**

## **5.9 member.html**

<!DOCTYPE html>

<html lang="en">

<head>

    <meta charset="UTF-8">

    <meta name="viewport" content="width=device-width, initial-scale=1.0">

    <link rel="stylesheet" href="memberb.css">

    <title>Nexus Bloom | Member Benefits</title>

</head>

<body>

    <header class="header">

        <nav class="nav">

            <ul class="nav-menu">

                <li><a href="../homepage/homepage.html" class="nav-link">Home</a></li>

                <li><a href="../clothing/clothing.html" class="nav-link">Our Clothing</a></li>

                <li><a href="../accessories/accessories.html" class="nav-link">Accessory Items</a></li>

                <li><a href="../sustainable/sustainable.html" class="nav-link">Sustainable Products</a></li>

                <li><a href="../whatsnew/whatsnew.html" class="nav-link">What's New</a></li>

                <li><a href="../membership/memberb.html" class="nav-link">Member Benefits</a></li>

            </ul>

        </nav>

    </header>

    <div class="member-benefits">

        <h1>Member Benefits</h1>

        <p>Join us and unlock a world of exclusive benefits designed to enhance your experience.</p>

        <h2>Membership Tiers</h2>

        <table>

            <tr>

                <th>Benefit</th>

                <th>Basic</th>

                <th>Premium</th>

                <th>VIP</th>

            </tr>

            <tr>

                <td>Discounts</td>

                <td>5%</td>

                <td>10%</td>

                <td>15%</td>

            </tr>

            <tr>

                <td>Access to Events</td>

                <td>No</td>

                <td>Yes</td>

                <td>Yes</td>

            </tr>

            <tr>

                <td>Exclusive Newsletters</td>

                <td>No</td>

                <td>Yes</td>

                <td>Yes</td>

            </tr>

            <tr>

                <td>Personal Consultation</td>

                <td>No</td>

                <td>No</td>

                <td>Yes</td>

            </tr>

        </table>

        <h2>Exclusive Benefits</h2>

        <ul>

            <li>Access to members-only events</li>

            <li>Exclusive discounts on products</li>

            <li>Newsletters with tips and updates</li>

        </ul>

        <h2>Resources and Tools</h2>

        <p>Gain access to educational content, community forums, and more.</p>

        <h2>Networking Opportunities</h2>

        <p>Connect with peers and mentors in our vibrant community.</p>

        <h2>Member Testimonials</h2>

        <blockquote>"Being a member has changed my life!" - Happy Member</blockquote>

        <h2>FAQs</h2>

        <p>Have questions? Check our FAQs or contact us.</p>

        <button class="join-button">Join Now</button>

    </div>

    <footer>

        <p>&copy; 2024 Nexus Bloom. All rights reserved.</p>

    </footer>

</body>

</html>

## **5.10 polls.html**

<!DOCTYPE html>

<html lang="en">

<head>

    <meta charset="UTF-8">

    <meta name="viewport" content="width=device-width, initial-scale=1.0">

    <link href='https://unpkg.com/boxicons@2.1.4/css/boxicons.min.css' rel='stylesheet'>

    <link rel="stylesheet" href="poll.css">

    <title>Nexus Bloom | Polls</title>

</head>

<body>

    <header class="header">

        <div class="nav">

            <ul class="nav-menu">

                <li><a href="../homepage/homepage.html" class="nav-link">Home</a></li>

                <li><a href="../about/aboutus.html" class="nav-link">About Us</a></li>

                <li><a href="../donate/donate.html" class="nav-link">Donate</a></li>

                <li><a href="../profile/profile.html" class="nav-link">Profile</a></li>

                <li><a href="../clothing/clothing.html" class="nav-link">OUR CLOTHING</a></li>

                <li><a href="../accessories/accessories.html" class="nav-link">ACCESSORY ITEMS</a></li>

                <li><a href="../sustainable/sustainable.html" class="nav-link">Sustainable Products</a></li>

                <li><a href="../whatsnew/whatsnew.html" class="nav-link">What's New</a></li>

            </ul>

        </div>

    </header>

    <div class="content">

        <h2>Polls</h2>

        <div class="poll">

            <h3>What type of products would you like to see more of?</h3>

            <form>

                <label>

                    <input type="radio" name="product-type" value="clothing"> Clothing

                </label><br>

                <label>

                    <input type="radio" name="product-type" value="accessories"> Accessories

                </label><br>

                <label>

                    <input type="radio" name="product-type" value="sustainable-products"> Sustainable Products

                </label><br>

                <button type="submit">Vote</button>

            </form>

        </div>

        <div class="poll">

            <h3>How often do you shop for sustainable products?</h3>

            <form>

                <label>

                    <input type="radio" name="shopping-frequency" value="weekly"> Weekly

                </label><br>

                <label>

                    <input type="radio" name="shopping-frequency" value="monthly"> Monthly

                </label><br>

                <label>

                    <input type="radio" name="shopping-frequency" value="occasionally"> Occasionally

                </label><br>

                <button type="submit">Vote</button>

            </form>

        </div>

        <div class="poll">

            <h3>What factors influence your purchasing decisions the most?</h3>

            <form>

                <label>

                    <input type="radio" name="purchasing-factors" value="price"> Price

                </label><br>

                <label>

                    <input type="radio" name="purchasing-factors" value="quality"> Quality

                </label><br>

                <label>

                    <input type="radio" name="purchasing-factors" value="brand-sustainability"> Brand Sustainability

                </label><br>

                <button type="submit">Vote</button>

            </form>

        </div>

        <div class="poll">

            <h3>Would you recommend our products to a friend?</h3>

            <form>

                <label>

                    <input type="radio" name="recommendation" value="yes"> Yes

                </label><br>

                <label>

                    <input type="radio" name="recommendation" value="no"> No

                </label><br>

                <button type="submit">Vote</button>

            </form>

        </div>

        <div class="poll">

            <h3>How did you hear about us?</h3>

            <form>

                <label>

                    <input type="radio" name="referral-source" value="social-media"> Social Media

                </label><br>

                <label>

                    <input type="radio" name="referral-source" value="friend"> Friend

                </label><br>

                <label>

                    <input type="radio" name="referral-source" value="advertisement"> Advertisement

                </label><br>

                <button type="submit">Vote</button>

            </form>

        </div>

    </div>

    <section class="info">

        <h2>Thank You for Participating!</h2>

        <p>Your feedback is essential in helping us improve our offerings and better serve our community. Stay tuned for more updates!</p>

    </section>

</body>

</html>

**5.11 profile.html**

<!DOCTYPE html>

<html lang="en">

<head>

    <meta charset="UTF-8">

    <meta name="viewport" content="width=device-width, initial-scale=1.0">

    <title>Nexus Bloom | Profile</title>

    <link rel="stylesheet" href="profile.css">

</head>

<body>

    <div class="container">

        <header class="header">

            <h1>User Profile</h1>

        </header>

        <div class="profile">

            <!-- Clickable profile picture -->

            <img src="../assets/R.jpg" alt="Profile Picture" class="profile-pic" id="profilePic" onclick="triggerProfilePicInput()">

            <!-- Hidden file input for changing the profile picture -->

            <input type="file" id="profilePicInput" accept="image/\*" style="display: none;" onchange="handleProfilePicChange()"><br>

            <div class="profile-details">

                <label>Username: </label>

                <h2 id="name"></h2>

                <label>Email: </label>

                <p id="email"></p>

                <label>Phone: </label>

                <p id="phone"></p>

                <label>Country: </label>

                <p id="country"></p>

                <label>Bio: </label>

                <p id="bio"></p>

                <!-- Editable Profile Form -->

                <form id="profile-form" style="display: none;">

                    <label for="emailInput">Email:</label>

                    <input type="email" id="emailInput" name="email"><br>

                    <label for="phoneInput">Phone:</label>

                    <input type="tel" id="phoneInput" name="phone"><br>

                    <label for="countryInput">Country:</label>

                    <input type="text" id="countryInput" name="country"><br>

                    <label for="bioInput">Bio:</label>

                    <textarea id="bioInput" name="bio"></textarea><br>

                </form>

            </div>

        </div>

        <div class="social-media">

            <h3>Connect with me</h3>

            <ul>

                <li><a href="https://linkedin.com/in/riddhiK" target="\_blank">LinkedIn</a></li>

                <li><a href="https://twitter.com/riddhiK" target="\_blank">Twitter</a></li>

                <li><a href="https://github.com/riddhiK" target="\_blank">GitHub</a></li>

            </ul>

        </div>

        <div class="button-container">

            <button id="edit-btn" onclick="editProfile()">Edit</button>

            <button id="save-btn" onclick="saveProfile()" style="display: none;">Save</button>

            <button onclick="goBack()">Back</button>

            <button onclick="logout()">Logout</button>

        </div>

    </div>

    <script>

        const username = localStorage.getItem('uname');

        addEventListener("DOMContentLoaded", () => {

            fetchUser().then(data => {

                document.getElementById('name').innerText = data.username || "Username not provided";

                document.getElementById('email').innerText = data.email || 'Email not provided';

                document.getElementById('phone').innerText = data.phoneNumber || "";

                document.getElementById('country').innerText = data.country || "";

                document.getElementById('bio').innerText = data.desc || "";

            });

        });

        function goBack() {

            window.history.back();

        }

        function logout() {

            window.location.href = "../signin/index.html";

            localStorage.clear();

        }

        const fetchUser = async () => {

            const response = await fetch(`http://localhost:3000/user/${username}`);

            const data = await response.json();

            return data;

        };

        function editProfile() {

            document.getElementById('edit-btn').style.display = 'none';

            document.getElementById('save-btn').style.display = 'inline-block';

            document.getElementById('profile-form').style.display = 'block';

            // Populate form fields with current profile data

            document.getElementById('emailInput').value = document.getElementById('email').innerText;

            document.getElementById('phoneInput').value = document.getElementById('phone').innerText;

            document.getElementById('countryInput').value = document.getElementById('country').innerText;

            document.getElementById('bioInput').value = document.getElementById('bio').innerText;

        }

        function triggerProfilePicInput() {

            document.getElementById('profilePicInput').click(); // Trigger the hidden file input click

        }

        function handleProfilePicChange() {

            const profilePicInput = document.getElementById('profilePicInput');

            const profilePicFile = profilePicInput.files[0];

            if (profilePicFile) {

                // Preview the selected image

                const reader = new FileReader();

                reader.onload = function(e) {

                    document.getElementById('profilePic').src = e.target.result; // Set the new profile pic

                };

                reader.readAsDataURL(profilePicFile);

                // Automatically save the new profile picture

                saveProfilePic(profilePicFile);

            }

        }

        function saveProfilePic(profilePicFile) {

            const formData = new FormData();

            formData.append('profilePic', profilePicFile); // Append the selected file

            // Send to the server (adjust the URL to match your server route)

            fetch(`http://localhost:3000/user/profile-pic/${username}`, {

                method: 'PUT',

                body: formData

            }).then(response => response.json())

              .then(data => {

                  console.log('Profile picture updated successfully');

              }).catch(error => console.error('Error updating profile picture:', error));

        }

        function saveProfile() {

            // Retrieve input values

            const email = document.getElementById('emailInput').value;

            const phoneNumber = document.getElementById('phoneInput').value;

            const country = document.getElementById('countryInput').value;

            const desc = document.getElementById('bioInput').value;

            // Update profile details

            document.getElementById('email').innerText = email;

            document.getElementById('phone').innerText = phoneNumber;

            document.getElementById('country').innerText = country;

            document.getElementById('bio').innerText = desc;

            // Hide form and show edit button

            document.getElementById('edit-btn').style.display = 'inline-block';

            document.getElementById('save-btn').style.display = 'none';

            document.getElementById('profile-form').style.display = 'none';

            // Prepare data to send to the server

            const data = {

                email,

                phoneNumber,

                country,

                desc

            };

            // Send the updated profile details to the server

            fetch(`http://localhost:3000/user/${username}`, {

                method: 'PUT',

                body: data // Send as FormData since we have profile details

            }).then(response => response.json())

              .then(data => {

                  console.log('Profile updated successfully');

              }).catch(error => console.error('Error updating profile:', error));

        }

    </script>

</body>

</html>

## **5.12 sales\_report.html**

<!DOCTYPE html>

<html lang="en">

<head>

    <meta charset="UTF-8">

    <meta name="viewport" content="width=device-width, initial-scale=1.0">

    <title>Sales Report - Nexus Bloom</title>

    <link rel="stylesheet" href="sales\_report.css">

</head>

<body>

    <header class="header">

        <nav class="nav">

            <ul class="nav-menu">

                <li><a href="../homeseller/home\_seller.html" class="nav-link">Dashboard</a></li>

                <li><a href="../manageproduct/manage-products.html" class="nav-link">Manage Products</a></li>

                <li><a href="../salesreport/sales\_report.html" class="nav-link">Sales Report</a></li>

            </ul>

        </nav>

    </header>

    <div class="content-wrapper">

        <h1>Sales Report</h1>

        <div class="filter-section">

            <label for="username">Username:</label>

            <input type="text" id="username" placeholder="Enter username">

            <label for="category">Category:</label>

            <select id="category">

                <option value="All">All</option>

                <option value="Clothing">Clothing</option>

                <option value="Accessories">Accessories</option>

                <option value="Sustainable Items">Sustainable Items</option>

            </select>

            <button class="filter-button">Filter</button>

            <button class="refresh-button">Refresh</button>

        </div>

        <table class="sales-table">

            <thead>

                <tr>

                    <th>Seller ID</th>

                    <th>Product Name</th>

                    <th>Price</th>

                    <th>Category</th>

                    <th>Number of Items Sold</th>

                    <th>Date</th>

                    <th>Total Amount</th>

                    <th></th>

                </tr>

            </thead>

            <tbody id="product-grid">

                <!-- Dynamic Data Will Be Inserted Here -->

            </tbody>

        </table>

    </div>

    <footer class="footer">

        <div class="back-button">

            <button onclick="window.location.href='../homeseller/home\_seller.html'">Back to Dashboard</button>

        </div>

    </footer>

    <script src="sales\_report.js"></script>

</body>

</html>

## **5.13 index.html**

<!DOCTYPE html>

<html lang="en">

<head>

    <meta charset="UTF-8">

    <meta http-equiv="X-UA-Compatible" content="IE=edge">

    <meta name="viewport" content="width=device-width, initial-scale=1.0">

    <link href='https://unpkg.com/boxicons@2.1.4/css/boxicons.min.css' rel='stylesheet'>

    <link rel="stylesheet" href="style.css">

    <title>Nexus Bloom | Sign In</title>

</head>

<body>

    <div class="wrapper">

        <nav class="nav">

            <div class="nav-logo">

                <p>Nexus Bloom</p>

            </div>

            <div class="nav-button">

                <button class="btn white-btn" id="loginBtn">Sign In</button>

            </div>

        </nav>

        <div class="form-box">

            <div class="login-container">

                <div class="top">

                    <span>Don't have an account? <a href="../signup/signup.html">Sign Up</a></span>

                    <header>Login</header>

                </div>

                <form id="loginForm" onsubmit="validateForm(event)">

                    <div class="input-box">

                        <input type="text" id="username" class="input-field" placeholder="Username or Email" required>

                        <i class="bx bx-user"></i>

                    </div>

                    <div class="input-box">

                        <input type="password" id="password" class="input-field" placeholder="Password" required>

                        <i class="bx bx-lock-alt"></i>

                    </div>

                    <div class="input-box">

                        <input type="submit" class="submit" value="Sign In">

                    </div>

                    <div class="two-col">

                        <div class="one">

                            <input type="checkbox" id="login-check">

                            <label for="login-check"> Remember Me</label>

                        </div>

                        <div class="two">

                            <label><a href="#">Forgot password?</a></label>

                        </div>

                    </div>

                </form>

            </div>

        </div>

    </div>

    <script>

        async function validateForm(event) {

            event.preventDefault();

            const username = document.getElementById('username').value.trim();

            const password = document.getElementById('password').value.trim();

            localStorage.setItem("uname",username);

            try {

                const response = await fetch('http://localhost:3000/auth/login', {

                    method: 'POST',

                    headers: {

                        'Content-Type': 'application/json',

                    },

                    body: JSON.stringify({ username, password }),

                });

                const result = await response.json();

                if (response.ok) {

                    alert(result.message);

                    localStorage.setItem(

                        'token',

                        result.token

                    )

                    // Redirect user to the appropriate homepage based on role

                    const role = result.role; // Assuming the backend returns the user's role

                    if (role === 'seller') {

                        window.location.href = '../homeseller/home\_seller.html';

                    } else if(role === 'buyer') {

                        window.location.href = '../homepage/homepage.html';

                    }

                } else {

                    alert(result.message || 'Login failed. Please try again.');

                }

            } catch (error) {

                console.error('Error:', error);

                alert('An error occurred. Please try again later.');

            }

        }

    </script>

</body>

</html>

## **5.14 signup.html**

<!DOCTYPE html>

<html lang="en">

<head>

    <meta charset="UTF-8">

    <meta http-equiv="X-UA-Compatible" content="IE=edge">

    <meta name="viewport" content="width=device-width, initial-scale=1.0">

    <link href='https://unpkg.com/boxicons@2.1.4/css/boxicons.min.css' rel='stylesheet'>

    <link rel="stylesheet" href="signup.css">

    <title>Nexus Bloom | Sign Up</title>

</head>

<body>

    <div class="wrapper">

        <nav class="nav">

            <div class="nav-logo">

                <p>Nexus Bloom</p>

            </div>

        </nav>

        <div class="form-box">

            <div class="signup-container">

                <div class="top">

                    <span>Already have an account? <a href="../signin/index.html">Sign In</a></span>

                    <header>Create an Account</header>

                </div>

                <form id="signupForm" onsubmit="return validateForm(event)">

                    <div class="input-box">

                        <input type="text" id="fullName" class="input-field" placeholder="Full Name" required>

                        <i class="bx bx-user"></i>

                    </div>

                    <div class="input-box">

                        <input type="text" id="phone" class="input-field" placeholder="Phone Number" required>

                        <i class="bx bx-phone"></i>

                    </div>

                    <div class="input-box">

                        <input type="text" id="email" class="input-field" placeholder="Email" required>

                        <i class="bx bx-envelope"></i>

                    </div>

                    <div class="input-box">

                        <input type="text" id="country" class="input-field" placeholder="Country" required>

                        <i class="bx bx-globe"></i>

                    </div>

                    <div class="input-box">

                        <input type="password" id="password" class="input-field" placeholder="Password" required>

                        <i class="bx bx-lock-alt"></i>

                    </div>

                    <div class="input-box">

                        <input type="password" id="confirmPassword" class="input-field" placeholder="Confirm Password" required>

                        <i class="bx bx-lock-alt"></i>

                    </div>

                    <div class="input-box">

                        <select id="userType" class="input-field" required>

                            <option value="">Select Account Type</option>

                            <option value="buyer">User</option>

                            <option value="seller">Seller</option>

                        </select>

                        <i class="bx bx-user"></i>

                    </div>

                    <div class="input-box">

                        <input type="submit" class="submit" value="Sign Up">

                    </div>

                </form>

            </div>

        </div>

    </div>

    <script>

        async function validateForm(event) {

            event.preventDefault(); // Prevent the form from submitting the traditional way

            const fullName = document.getElementById('fullName').value.trim();

            const phone = document.getElementById('phone').value.trim();

            const email = document.getElementById('email').value.trim();

            const country = document.getElementById('country').value.trim();

            const password = document.getElementById('password').value.trim();

            const confirmPassword = document.getElementById('confirmPassword').value.trim();

            const userType = document.getElementById('userType').value;

            // Basic form validation

            if (fullName.length < 3) {

                alert("Full Name must be at least 3 characters long.");

                return false;

            }

            const phonePattern = /^[0-9]{10,15}$/;

            const emailPattern = /^[^\s@]+@[^\s@]+\.[^\s@]+$/;

            if (!phonePattern.test(phone)) {

                alert("Please enter a valid phone number (10-15 digits).");

                return false;

            }

            if (!emailPattern.test(email)) {

                alert("Please enter a valid email address.");

                return false;

            }

            if (country.length < 2) {

                alert("Please enter a valid country name.");

                return false;

            }

            if (password.length < 8) {

                alert("Password must be at least 8 characters long.");

                return false;

            }

            if (password !== confirmPassword) {

                alert("Passwords do not match.");

                return false;

            }

            if (!userType) {

                alert("Please select an account type.");

                return false;

            }

            // Data object to be sent to the server

            const formData = {

                username: fullName, // Assuming username is the full name

                password: password,

                email: email,

                role: userType, // The role field should match the backend expectation

                phoneNumber: phone, // Ensure this matches the backend field

                country: country

            };

            try {

                const response = await fetch('http://localhost:3000/auth/register', {

                    method: 'POST',

                    headers: {

                        'Content-Type': 'application/json',

                    },

                    body: JSON.stringify(formData),

                });

                const result = await response.json();

                if (response.ok) {

                    alert(result.message);

                    window.location.href = '../signin/index.html'

                } else {

                    alert(result.message || "Registration failed. Please try again.");

                }

            } catch (error) {

                console.error('Error:', error);

                alert("An error occurred. Please try again later.");

            }

            return false; // Prevent default form submission

        }

    </script>

</body>

</html>

## **5.15 startpage.html**

<!DOCTYPE html>

<html lang="en">

<head>

    <meta charset="UTF-8">

    <meta name="viewport" content="width=device-width, initial-scale=1.0">

    <title>NexusBloom</title>

    <link rel="stylesheet" href="https://stackpath.bootstrapcdn.com/bootstrap/4.5.2/css/bootstrap.min.css">

    <link rel="stylesheet" href="startpage.css">

    <link href="https://fonts.googleapis.com/css2?family=Kalnia+Glaze&display=swap" rel="stylesheet">

</head>

<body>

<div id="carouselExampleIndicators" class="carousel slide" data-ride="carousel">

    <ol class="carousel-indicators">

        <li data-target="#carouselExampleIndicators" data-slide-to="0" class="active"></li>

        <li data-target="#carouselExampleIndicators" data-slide-to="1"></li>

        <li data-target="#carouselExampleIndicators" data-slide-to="2"></li>

    </ol>

    <div class="carousel-inner">

        <div class="carousel-item active">

            <img src="../assests/image1.jpg" class="d-block w-100 carousel-image" alt="Image 1">

            <div class="carousel-caption">

                <h1 class="animate-text">Welcome to NexusBloom</h1>

                <p class="animate-text">Empowering local women’s businesses.</p>

                <p class="animate-text">Buy and sell unique handmade items while supporting your community.</p>

                <button onclick="window.location.href='../signup/signup.html'" class="btn btn-primary">Get Started</button>

            </div>

        </div>

        <div class="carousel-item">

            <img src="../assests/image2.jpg.png" class="d-block w-100 carousel-image" alt="Image 2">

            <div class="carousel-caption">

                <h1 class="animate-text">Welcome to NexusBloom</h1>

                <p class="animate-text">Empowering local women’s businesses.</p>

                <p class="animate-text">Buy and sell unique handmade items while supporting your community.</p>

                <button onclick="window.location.href='index.html'" class="btn btn-primary">Get Started</button>

            </div>

        </div>

        <div class="carousel-item">

            <img src="../assests/image3.jpg.png" class="d-block w-100 carousel-image" alt="Image 3">

            <div class="carousel-caption">

                <h1 class="animate-text">Welcome to NexusBloom</h1>

                <p class="animate-text">Empowering local women’s businesses.</p>

                <p class="animate-text">Buy and sell unique handmade items while supporting your community.</p>

                <button onclick="window.location.href='index.html'" class="btn btn-primary">Get Started</button>

            </div>

        </div>

    </div>

    <a class="carousel-control-prev" href="#carouselExampleIndicators" role="button" data-slide="prev">

        <span class="carousel-control-prev-icon" aria-hidden="true"></span>

        <span class="sr-only">Previous</span>

    </a>

    <a class="carousel-control-next" href="#carouselExampleIndicators" role="button" data-slide="next">

        <span class="carousel-control-next-icon" aria-hidden="true"></span>

        <span class="sr-only">Next</span>

    </a>

</div>

<script src="https://code.jquery.com/jquery-3.5.1.slim.min.js"></script>

<script src="https://cdn.jsdelivr.net/npm/@popperjs/core@2.5.4/dist/umd/popper.min.js"></script>

<script src="https://stackpath.bootstrapcdn.com/bootstrap/4.5.2/js/bootstrap.min.js"></script>

<script src="startpage.js"></script>

</body>

</html>

## **5.16 sustainable.html**

<!DOCTYPE html>

<html lang="en">

<head>

    <meta charset="UTF-8">

    <meta name="viewport" content="width=device-width, initial-scale=1.0">

    <link rel="stylesheet" href="sustainable.css">

    <title>Nexus Bloom | Sustainable Products</title>

</head>

<body>

    <header class="header">

        <nav class="nav">

            <ul class="nav-menu">

                <li><a href="../homepage/homepage.html" class="nav-link">Home</a></li>

                <li><a href="../about/aboutus.html" class="nav-link">About Us</a></li>

                <li><a href="../donate/donate.html" class="nav-link">Donate</a></li>

                <li><a href="../profile/profile.html" class="nav-link">Profile</a></li>

                <li><a href="../cart/cart.html" class="nav-link">Cart</a></li>

                <li class="basket">

                    <img src="../assests/basket.png" alt="Basket Icon">

                    <span>Basket: </span><span id="basket-count">0</span> items

                </li>

            </ul>

        </nav>

    </header>

    <div class="main-content">

        <div class="product-grid">

            <!-- Products will be dynamically inserted here -->

        </div>

    </div>

    <footer class="footer">

        <div class="back-button">

            <button onclick="window.location.href='../homepage/homepage.html'">Back to Home</button>

        </div>

    </footer>

    <script src="../scripts/sustainable.js"></script>

</body>

</html>

## **5.17 test.html**

<!DOCTYPE html>

<html lang="en">

<head>

    <meta charset="UTF-8">

    <meta name="viewport" content="width=device-width, initial-scale=1.0">

    <title>Data Table</title>

    <style>

        body {

            font-family: Arial, sans-serif;

            margin: 20px;

            background-color: #f4f4f4;

        }

        table {

            width: 100%;

            border-collapse: collapse;

            margin-top: 20px;

            background-color: #fff;

        }

        table, th, td {

            border: 1px solid #ddd;

        }

        th, td {

            padding: 8px;

            text-align: left;

        }

        th {

            background-color: #4CAF50;

            color: white;

        }

        tr:nth-child(even) {

            background-color: #f2f2f2;

        }

        tr:hover {

            background-color: #ddd;

        }

    </style>

</head>

<body>

<h1>Data Table</h1>

<table id="data-table">

    <thead>

        <tr>

            <th>ID</th>

            <th>Name</th>

            <th>Email</th>

        </tr>

    </thead>

    <tbody>

        <!-- Data will be inserted here by JavaScript -->

    </tbody>

</table>

<script>

    document.addEventListener('DOMContentLoaded', () => {

        fetch('https://yourapi.com/data')  // Replace with your API endpoint

            .then(response => response.json())

            .then(data => {

                const tableBody = document.querySelector('#data-table tbody');

                // Clear existing data

                tableBody.innerHTML = '';

                // Populate table with data

                data.forEach(item => {

                    const row = document.createElement('tr');

                    row.innerHTML = `

                        <td>${item.id}</td>

                        <td>${item.name}</td>

                        <td>${item.email}</td>

                    `;

                    tableBody.appendChild(row);

                });

            })

            .catch(error => console.error('Error fetching data:', error));

    });

</script>

</body>

</html>

## **5.18 traffic\_details.html**

<!DOCTYPE html>

<html lang="en">

<head>

    <meta charset="UTF-8">

    <meta name="viewport" content="width=device-width, initial-scale=1.0">

    <title>Website Traffic Details - Nexus Bloom</title>

    <link rel="stylesheet" href="traffic\_details.css">

</head>

<body>

    <header class="header">

        <nav class="nav">

            <ul class="nav-menu">

                <li><a href="../homeseller/home\_seller.html" class="nav-link">Dashboard</a></li>

                <li><a href="#" class="nav-link">Add Product</a></li>

                <li><a href="../manageproduct/manage-products.html" class="nav-link">Manage Products</a></li>

                <li><a href="../salesreport/sales\_report.html" class="nav-link">Sales Report</a></li>

                <li><a href="../traficdetail/traffic\_details.html" class="nav-link">Traffic Details</a></li>

            </ul>

        </nav>

    </header>

    <div class="content-wrapper">

        <h1>Website Traffic Details</h1>

        <div class="filter-section">

            <label for="date-range">Date Range:</label>

            <input type="date" id="start-date"> to <input type="date" id="end-date">

            <button class="filter-button">Filter</button>

        </div>

        <div class="traffic-summary">

            <div class="summary-card">

                <h3>Total Visitors</h3>

                <p id="total-visitors">1,200</p>

            </div>

            <div class="summary-card">

                <h3>Page Views</h3>

                <p id="page-views">4,500</p>

            </div>

            <div class="summary-card">

                <h3>Bounce Rate</h3>

                <p id="bounce-rate">45%</p>

            </div>

            <div class="summary-card">

                <h3>Average Session Duration</h3>

                <p id="avg-session-duration">3:15</p>

            </div>

        </div>

        <div class="traffic-chart">

            <h2>Daily Traffic Overview</h2>

            <canvas id="trafficChart"></canvas>

        </div>

    </div>

    <footer class="footer">

        <div class="back-button">

            <button onclick="window.location.href='../homeseller/home\_seller.html'">Back to Dashboard</button>

        </div>

    </footer>

    <script src="https://cdn.jsdelivr.net/npm/chart.js"></script>

    <script>

        // Sample data

        const trafficData = {

            labels: ['2024-08-01', '2024-08-02', '2024-08-03', '2024-08-04', '2024-08-05'],

            datasets: [{

                label: 'Page Views',

                data: [150, 200, 250, 300, 400],

                borderColor: 'rgba(75, 192, 192, 1)',

                backgroundColor: 'rgba(75, 192, 192, 0.2)',

                fill: true,

            }]

        };

        // Render chart

        const ctx = document.getElementById('trafficChart').getContext('2d');

        const trafficChart = new Chart(ctx, {

            type: 'line',

            data: trafficData,

            options: {

                responsive: true,

                plugins: {

                    legend: {

                        position: 'top',

                    },

                    tooltip: {

                        callbacks: {

                            label: function(tooltipItem) {

                                return `Page Views: ${tooltipItem.raw}`;

                            }

                        }

                    }

                },

                scales: {

                    x: {

                        title: {

                            display: true,

                            text: 'Date'

                        }

                    },

                    y: {

                        title: {

                            display: true,

                            text: 'Page Views'

                        },

                        beginAtZero: true

                    }

                }

            }

        });

        // Filter Button Event

        document.querySelector('.filter-button').addEventListener('click', function() {

            const startDate = document.getElementById('start-date').value;

            const endDate = document.getElementById('end-date').value;

            // Implement filtering logic here based on startDate and endDate

            alert(`Filter applied from ${startDate} to ${endDate}`);

        });

    </script>

</body>

</html>

## **5.19 whatsnew.html**

<!DOCTYPE html>

<html lang="en">

<head>

    <meta charset="UTF-8">

    <meta name="viewport" content="width=device-width, initial-scale=1.0">

    <link rel="stylesheet" href="whatsnew.css">

    <title>Nexus Bloom | What’s New</title>

</head>

<body>

    <header class="header">

        <nav class="nav">

            <ul class="nav-menu">

                <li><a href="../homepage/homepage.html" class="nav-link">Home</a></li>

                <li><a href="../about/aboutus.html" class="nav-link">About Us</a></li>

                <li><a href="../donate/donate.html" class="nav-link">Donate</a></li>

                <li><a href="../profile/profile.html" class="nav-link">Profile</a></li>

            </ul>

        </nav>

    </header>

    <div class="main-content">

        <h1>What’s New</h1>

        <section class="update">

            <h2>Community Drives</h2>

            <div class="drive">

                <h3>Beach Cleaning Drive</h3>

                <p>Join us in our beach cleaning drive aimed at keeping our shores pristine. This is a great opportunity to make a positive impact on the environment.</p>

                <button onclick="window.location.href='beach-cleaning.html'">Learn More</button>

            </div>

            <div class="drive">

                <h3>Tree Planting Campaign</h3>

                <p>Participate in our tree planting campaign to help combat deforestation. Every tree planted contributes to a greener planet.</p>

                <button onclick="window.location.href='tree-planting.html'">Get Involved</button>

            </div>

            <div class="drive">

                <h3>Plastic Waste Reduction Initiative</h3>

                <p>Be part of our initiative to reduce plastic waste. Join workshops and community efforts to promote sustainable practices.</p>

                <button onclick="window.location.href='plastic-waste.html'">Learn More</button>

            </div>

        </section>

        <section class="update">

            <h2>Awareness Initiatives</h2>

            <p>We are committed to raising awareness about sustainability and eco-friendly practices. Participate in our workshops, educational events, and community outreach programs to learn more and make a difference.</p>

            <button onclick="window.location.href='awareness.html'">Get Involved</button>

        </section>

    </div>

    <footer class="footer">

        <div class="back-button">

            <button onclick="window.location.href='../homepage/homepage.html'">Back to Home</button>

        </div>

    </footer>

</body>

</html>

## **5.20 home\_seller.js**

document.addEventListener("DOMContentLoaded", () => {

    fetchTotalSales().then(details => {

        document.getElementById('total-sales').innerText = details.total;

        document.getElementById('qty').innerText = details.qty;

    }).catch(error => {

        console.error("Error fetching total sales:", error);

    });

});

const fetchTotalSales = async () => {

    const username = localStorage.getItem('uname');

    const response = await fetch(`http://localhost:3000/api/sales/total/${username}`);

    const data = await response.json();

    return data;

}

## **5.21 sales\_report.js**

document.addEventListener('DOMContentLoaded', () => {

    // Function to fetch products by sellerId

    async function fetchProductsBySeller(sellerId) {

        try {

            const response = await fetch(`http://localhost:3000/api/sales-report/${sellerId}`, {

                method: 'GET',

                headers: {

                    'Content-Type': 'application/json' // Optional for GET requests, but good practice

                }

            });

            if (!response.ok) {

                const errorText = await response.text();

                throw new Error(`Network response was not ok. Status: ${response.status}, Message: ${errorText}`);

            }

            const products = await response.json();

            console.log("hello",products);

            // Process or display the products as needed

            displayProducts(products);

        } catch (e) {

            console.error('Error fetching products:', e);

            // Optionally display a user-friendly message or handle errors appropriately

            displayErrorMessage('Could not load products. Please try again later.');

        }

    }

    // Retrieve the seller ID from localStorage

    const sellerId = localStorage.getItem('uname');

    if (sellerId) {

        fetchProductsBySeller(sellerId); // Fetch products for the current seller

    } else {

        console.log('Seller ID not found');

    }

});

// Function to display products in the UI

function displayProducts(products) {

    const productGrid = document.getElementById('product-grid');

    if (!productGrid) {

        console.error('Product grid element not found');

        return;

    }

    productGrid.innerHTML = ''; // Clear existing content

    products.forEach(product => {

        const { sellerId,date, productName, price, category, quantity } = product;

        const total = parseInt(price)\*parseInt(quantity);

        // Create a new table row

        const productRow = document.createElement('tr');

        productRow.setAttribute('data-product-id', sellerId); // Set data-product-id

        // Set up the row with cells and fallback values

        productRow.innerHTML = `

            <td>${sellerId}</td>

            <td>${productName || 'Unnamed Product'}</td>

            <td>$${price || 'N/A'}</td>

            <td>${category || 'No category available'}</td>

            <td>${quantity || 'N/A'}</td>

            <td>${date || 'N/A'}</td>

            <td>${total || 'N/A'}</td>

            <td>

                <button class="add-to-cart">Add to Cart</button>

                <button class="buy-now">Buy Now</button>

            </td>

        `;

        // Append the row to the tbody

        productGrid.appendChild(productRow);

    });

    // Add event listeners after products are displayed

    addEventListenersToCartButtons();

}

// Function to display an error message

function displayErrorMessage(message) {

    const productGrid = document.getElementById('product-grid');

    if (!productGrid) return;

    // Display an error message inside the grid

    productGrid.innerHTML = `<div class="error">${message}</div>`;

}

// Function to add event listeners to "Add to Cart" buttons

function addEventListenersToCartButtons() {

    const addToCartButtons = document.querySelectorAll('.add-to-cart');

    addToCartButtons.forEach(button => {

        button.addEventListener('click', (event) => {

            const productCard = event.target.closest('.product-card');

            const productId = productCard.getAttribute('data-product-id');

            const productName = productCard.querySelector('h2').textContent;

            const productPrice = productCard.querySelector('.price').textContent;

            const productImage = productCard.querySelector('img').src;

            const sellerId = localStorage.getItem('uname');

            const product = {

                id: productId,

                name: productName,

                price: productPrice,

                image: productImage,

                seller: sellerId

            };

            addProductToLocalStorage(product);

        });

    });

}

// Function to add product to local storage

function addProductToLocalStorage(product) {

    let cart = JSON.parse(localStorage.getItem('cart')) || [];

    cart.push(product);

    localStorage.setItem('cart', JSON.stringify(cart));

    alert('Item added to cart successfully!');

}

## **5.22 accessories.js**

document.addEventListener('DOMContentLoaded', () => {

    fetchProductsByCategory('accessories');

});

async function fetchProductsByCategory(category) {

    try {

        const response = await fetch(`http://localhost:3000/products/category/${category}`);

        if (!response.ok) {

            throw new Error(`Failed to fetch products: ${response.status}`);

        }

        const products = await response.json();

        displayProducts(products);

    } catch (error) {

        console.error('Error fetching products:', error);

        displayErrorMessage('Could not load products. Please try again later.');

    }

}

// Display products

function displayProducts(products) {

    const productGrid = document.getElementById('product-grid');

    if (!productGrid) {

        console.error('Product grid element not found');

        return;

    }

    productGrid.innerHTML = ''; // Clear existing content

    products.forEach(product => {

        const { id, image, name, description, price, seller } = product;

        // Create the product card

        const productCard = document.createElement('div');

        productCard.className = 'product-card';

        productCard.setAttribute('data-product-id', id); // Set data-product-id

        // Set up the inner HTML with fallback values

        productCard.innerHTML = `

            <img src="${image || 'placeholder.jpg'}" alt="${name || 'Product Image'}">

            <h2>${name || 'Unnamed Product'}</h2>

            <p>${description || 'No description available'}</p>

            <div class="price">$${price || 'N/A'}</div>

            <div class="seller">Seller ID: ${seller || 'Unknown'}</div>

            <button class="add-to-cart" data-seller-id="${seller}">Add to Cart</button>

            <button class="buy-now">Buy Now</button>

        `;

        // Append the product card to the grid

        productGrid.appendChild(productCard);

    });

    // Add event listeners after products are displayed

    addEventListenersToCartButtons();

}

function displayErrorMessage(message) {

    const productGrid = document.getElementById('product-grid');

    if (!productGrid) return;

    // Display an error message inside the grid

    productGrid.innerHTML = `<div class="error">${message}</div>`;

}

// Add event listeners to "Add to Cart" buttons

function addEventListenersToCartButtons() {

    const addToCartButtons = document.querySelectorAll('.add-to-cart');

    addToCartButtons.forEach(button => {

        button.addEventListener('click', (event) => {

            const productCard = event.target.closest('.product-card');

            const productId = productCard.getAttribute('data-product-id');

            const productName = productCard.querySelector('h2').textContent;

            const productPrice = productCard.querySelector('.price').textContent;

            const productImage = productCard.querySelector('img').src;

            const sellerId = event.target.getAttribute('data-seller-id'); // Retrieve seller ID from button attribute

            const product = {

                id: productId,

                name: productName,

                price: productPrice,

                image: productImage,

                seller: sellerId,

                category: "Accessories"

            };

            addProductToLocalStorage(product);

        });

    });

}

function addProductToLocalStorage(product) {

    let cart = JSON.parse(localStorage.getItem('cart')) || [];

    cart.push(product);

    localStorage.setItem('cart', JSON.stringify(cart));

    alert('Item added to cart successfully!');

}

## **5.23 buy.js**

document.addEventListener('DOMContentLoaded', () => {

    const orderItemsContainer = document.getElementById('order-items');

    const totalPriceElement = document.getElementById('total-price');

    const sellerIdElement = document.getElementById('seller-id');

    const buyNowForm = document.getElementById('buy-now-form');

    populateOrderSummary();

    function populateOrderSummary() {

        // Get the cart from local storage

        const cart = JSON.parse(localStorage.getItem('cart')) || [];

        const sellerId = localStorage.getItem('seller-id') || ''; // Retrieve seller ID

        console.log(cart);

        let total = 0;

        const productMap = {};

        // Aggregate product quantities and prices

        cart.forEach(product => {

            const productKey = `${product.name}-${product.price}`;

            if (!productMap[productKey]) {

                productMap[productKey] = { ...product, quantity: 0 };

            }

            productMap[productKey].quantity += 1;

        });

        // Clear existing order items

        orderItemsContainer.innerHTML = '';

        // Create and append order items

        Object.values(productMap).forEach(product => {

            const orderItem = document.createElement('div');

            orderItem.className = 'order-item';

            orderItem.innerHTML = `

                <p>${product.name} - $${parseFloat(product.price.replace('$', '')).toFixed(2)} x ${product.quantity}</p>

            `;

            orderItemsContainer.appendChild(orderItem);

            total += parseFloat(product.price.replace('$', '')) \* product.quantity;

        });

        totalPriceElement.textContent = total.toFixed(2);

        sellerIdElement.value = sellerId; // Set the seller ID value

    }

    buyNowForm.addEventListener('submit', async (event) => {

        event.preventDefault();

        const formData = new FormData(buyNowForm);

        const data = Object.fromEntries(formData.entries());

        // Check if the cart has items to process

        const cart = JSON.parse(localStorage.getItem('cart')) || [];

        if (cart.length === 0) {

            alert('Cart is empty. Please add items before proceeding.');

            return;

        }

        // Log the token and cart for debugging

        const token = localStorage.getItem('token'); // Adjust if token retrieval differs

        console.log('Token:', token);

        console.log('Cart data:', cart);

        // Iterate through cart items and send data for each product

        for (let i = 0; i < cart.length; i++) {

            const product = cart[i];

            const orderData = {

                sellerId: product.seller,

                productName: product.name,

                price: parseFloat(product.price.replace('$', '')),

                category: product.category, // Adjust if you have specific categories

                quantity: 1, // Set the desired quantity or modify according to the cart item

            };

            try {

                const response = await fetch('http://localhost:3000/api/checkout', {

                    method: 'POST',

                    headers: {

                        'Content-Type': 'application/json',

                        // Add token to headers if needed for authentication

                        // 'Authorization': `Bearer ${token}`, // Uncomment if authorization is required

                    },

                    body: JSON.stringify(orderData),

                });

                if (response.ok) {

                    console.log(`Order placed successfully for ${product.name}`);

                    cart.splice(i, 1);

                    localStorage.setItem('cart', JSON.stringify(cart));

                    i--; // Decrement index to handle the next item correctly after removal

                } else {

                    console.error(`Error placing order for ${product.name}`);

                    alert(`Error placing order for ${product.name}. Please try again.`);

                }

            } catch (error) {

                console.error('Error:', error);

                alert(`Error placing order for ${product.name}. Please try again.`);

            }

        }

        // Check if all items are processed and then redirect

        if (cart.length === 0) {

            alert('All orders placed successfully!');

            localStorage.removeItem('cart'); // Clear the cart after successful purchases

            window.location.href = '../homepage/homepage.html'; // Redirect to home page

        }

    });

});

## **5.24 cart.js**

document.addEventListener('DOMContentLoaded', () => {

    const cartItemsContainer = document.getElementById('cart-items');

    const billItemsContainer = document.getElementById('bill-items');

    const totalPriceElement = document.getElementById('total-price');

    updateCart();

    updateBasketCount();

    function updateCart() {

        // Get the cart from local storage

        let cart = JSON.parse(localStorage.getItem('cart')) || [];

        console.log(cart);

        // Clear existing cart items

        cartItemsContainer.innerHTML = '';

        billItemsContainer.innerHTML = '';

        let total = 0;

        cart.forEach((product, index) => {

            const productCard = document.createElement('div');

            productCard.className = 'product-card';

            productCard.innerHTML = `

                <img src="${product.image}" alt="${product.name}">

                <div class="product-details">

                    <h2>Name: ${product.name}</h2>

                    <h3>Category: ${product.category}</h3>

                    <p>Price: ${product.price}</p>

                    <p>Seller: ${product.seller}</p>

                </div>

                <button class="remove-from-cart" data-index="${index}">Remove</button>

            `;

            cartItemsContainer.appendChild(productCard);

            // Add to bill details

            const billItem = document.createElement('div');

            billItem.className = 'bill-item';

            billItem.innerHTML = `

                <p>${product.name} - $${parseFloat(product.price.replace('$', '')).toFixed(2)}</p>

            `;

            billItemsContainer.appendChild(billItem);

            // Update total

            total += parseFloat(product.price.replace('$', ''));

        });

        totalPriceElement.textContent = total.toFixed(2);

        // Add event listeners to remove buttons

        const removeButtons = document.querySelectorAll('.remove-from-cart');

        removeButtons.forEach(button => {

            button.addEventListener('click', (event) => {

                const index = event.target.getAttribute('data-index');

                removeProductFromCart(index);

            });

        });

    }

    function updateBasketCount() {

        // Get the cart from local storage

        let cart = JSON.parse(localStorage.getItem('cart')) || [];

        // Update the basket count

        document.getElementById('basket-count').textContent = cart.length;

    }

    function removeProductFromCart(index) {

        // Get the cart from local storage

        let cart = JSON.parse(localStorage.getItem('cart')) || [];

        // Remove the product at the specified index

        cart.splice(index, 1);

        // Save the updated cart back to local storage

        localStorage.setItem('cart', JSON.stringify(cart));

        // Update the cart

        updateCart();

        updateBasketCount();

    }

});

## **5.25 clothing.js**

document.addEventListener('DOMContentLoaded', () => {

    fetchProductsByCategory('clothing');

});

async function fetchProductsByCategory(category) {

    try {

        const response = await fetch(`http://localhost:3000/products/category/${category}`);

        if (!response.ok) {

            throw new Error(`Failed to fetch products: ${response.status}`);

        }

        const products = await response.json();

        console.log(products);

        displayProducts(products);

    } catch (error) {

        console.error('Error fetching products:', error);

        displayErrorMessage('Could not load products. Please try again later.');

    }

}

function displayProducts(products) {

    const productGrid = document.getElementById('product-grid');

    if (!productGrid) {

        console.error('Product grid element not found');

        return;

    }

    productGrid.innerHTML = ''; // Clear existing content

    products.forEach(product => {

        const { id, image, name, description, price, seller } = product;

        console.log(product.name, product.seller);

        // Create the product card

        const productCard = document.createElement('div');

        productCard.className = 'product-card';

        productCard.setAttribute('data-product-id', id); // Set data-product-id

        // Set up the inner HTML with fallback values

        productCard.innerHTML = `

            <img src="${image || 'placeholder.jpg'}" alt="${name || 'Product Image'}">

            <h2>${name || 'Unnamed Product'}</h2>

            <p>${description || 'No description available'}</p>

            <div class="price">$${price || 'N/A'}</div>

            <div class="seller">Seller ID: ${seller || 'Unknown'}</div>

            <button class="add-to-cart">Add to Cart</button>

            <button class="buy-now">Buy Now</button>

        `;

        // Append the product card to the grid

        productGrid.appendChild(productCard);

    });

    // Add event listeners after products are displayed

    addEventListenersToCartButtons();

}

function displayErrorMessage(message) {

    const productGrid = document.getElementById('product-grid');

    if (!productGrid) return;

    // Display an error message inside the grid

    productGrid.innerHTML = `<div class="error">${message}</div>`;

}

// Add event listeners to "Add to Cart" buttons

function addEventListenersToCartButtons() {

    const addToCartButtons = document.querySelectorAll('.add-to-cart');

    addToCartButtons.forEach(button => {

        button.addEventListener('click', (event) => {

            const productCard = event.target.closest('.product-card');

            const productId = productCard.getAttribute('data-product-id');

            const productName = productCard.querySelector('h2').textContent;

            const productPrice = productCard.querySelector('.price').textContent;

            const productImage = productCard.querySelector('img').src;

            const sellerId = productCard.querySelector('.seller').textContent.replace('Seller ID: ', '');

            const product = {

                id: productId,

                name: productName,

                price: productPrice,

                image: productImage,

                seller: sellerId,

                category: "Clothing"

            };

            addProductToLocalStorage(product);

        });

    });

}

function addProductToLocalStorage(product) {

    let cart = JSON.parse(localStorage.getItem('cart')) || [];

    cart.push(product);

    localStorage.setItem('cart', JSON.stringify(cart));

    alert('Item added to cart successfully!');

}

## **5.26 sustainable.js**

document.addEventListener('DOMContentLoaded', () => {

    fetchProductsByCategory('sustainable');

});

async function fetchProductsByCategory(category) {

    try {

        const response = await fetch(`http://localhost:3000/products/category/${category}`);

        if (!response.ok) {

            throw new Error(`Failed to fetch products: ${response.status}`);

        }

        const products = await response.json();

        console.log(products); // For debugging purposes

        displayProducts(products);

    } catch (error) {

        console.error('Error fetching products:', error);

        displayErrorMessage('Could not load products. Please try again later.');

    }

}

function displayProducts(products) {

    const productGrid = document.querySelector('.product-grid');

    if (!productGrid) {

        console.error('Product grid element not found');

        return;

    }

    productGrid.innerHTML = ''; // Clear existing content

    products.forEach(product => {

        const { id, image, name, description, price, seller } = product;

        console.log(product.name, product.seller);

        // Create the product card

        const productCard = document.createElement('div');

        productCard.className = 'product-card';

        productCard.setAttribute('data-product-id', id); // Set data-product-id

        // Set up the inner HTML with fallback values

        productCard.innerHTML = `

            <img src="${image || 'placeholder.jpg'}" alt="${name || 'Product Image'}">

            <h2>${name || 'Unnamed Product'}</h2>

            <p>${description || 'No description available'}</p>

            <div class="price">$${price || 'N/A'}</div>

            <div class="seller">Seller ID: ${seller || 'Unknown'}</div>

            <button class="add-to-cart">Add to Cart</button>

            <button class="buy-now">Buy Now</button>

        `;

        // Append the product card to the grid

        productGrid.appendChild(productCard);

    });

    // Add event listeners after products are displayed

    addEventListenersToCartButtons();

}

function displayErrorMessage(message) {

    const productGrid = document.querySelector('.product-grid');

    if (!productGrid) return;

    // Display an error message inside the grid

    productGrid.innerHTML = `<div class="error">${message}</div>`;

}

// Add event listeners to "Add to Cart" buttons

function addEventListenersToCartButtons() {

    const addToCartButtons = document.querySelectorAll('.add-to-cart');

    addToCartButtons.forEach(button => {

        button.addEventListener('click', (event) => {

            const productCard = event.target.closest('.product-card');

            const productId = productCard.getAttribute('data-product-id');

            const productName = productCard.querySelector('h2').textContent;

            const productPrice = productCard.querySelector('.price').textContent;

            const productImage = productCard.querySelector('img').src;

            const sellerId = productCard.querySelector('.seller').textContent.replace('Seller ID: ', '');

            const product = {

                id: productId,

                name: productName,

                price: productPrice,

                image: productImage,

                seller: sellerId,

                category: "Sustainable"

            };

            addProductToLocalStorage(product);

        });

    });

}

function addProductToLocalStorage(product) {

    let cart = JSON.parse(localStorage.getItem('cart')) || [];

    cart.push(product);

    localStorage.setItem('cart', JSON.stringify(cart));

    alert('Item added to cart successfully!');

}

## **5.27 startpage.js**

document.addEventListener("DOMContentLoaded", function() {

    const animatedElements = document.querySelectorAll('.animate-text');

    animatedElements.forEach((el, index) => {

        el.style.animationDelay = `${index \* 0.5}s`;

    });

    // Keep text visible after animation completes

    animatedElements.forEach(el => {

        el.style.animationFillMode = "forwards";

    });

});

## **5.28 allowedOrigin.js**

const allowedOrigins = [

    '127.0.0.1:5500/'

]

module.exports  = allowedOrigins

## **5.29 corsOption.js**

// config/corsOptions.js

const allowedOrigins = [

  'http://localhost:3000',  // Replace with your frontend URL

  'http://127.0.0.1:5500',   // Replace with your frontend URL, including protocol

  'http://127.0.0.1:5501',

  'http://localhost:5500'

];

const corsOptions = {

  origin: function (origin, callback) {

      if (allowedOrigins.indexOf(origin) !== -1 || !origin) {

          callback(null, true);

      } else {

          callback(new Error('Not allowed by CORS'));

      }

  }

};

module.exports = corsOptions;

## **5.30 authController.js**

const User = require('../models/User');

const bcrypt = require('bcrypt');

const jwt = require('jsonwebtoken');

const { use } = require('../routes/orderRoutes');

const JWT\_SECRET = process.env.JWT\_SECRET || 'your\_secret\_key\_here'; // Replace with a secure key

// Register a new user

exports.register = async (req, res) => {

  const { username, email,phoneNumber, country, password, role } = req.body;

  try {

    // Check if user already exists

    const existingUser1 = await User.findOne({ username });

    if (existingUser1) {

      return res.status(400).json({ message: 'User already exists with given username' });

    }

    const existingUser2 = await User.findOne({ phoneNumber });

    if (existingUser2) {

      return res.status(400).json({ message: 'User already exists with given phone number' });

    }

    const existingUser3 = await User.findOne({ email });

    if (existingUser3) {

      return res.status(400).json({ message: 'User already exists with given email' });

    }

    // Validate role

    if ('seller', 'buyer'].includes(role)) { // Make sure roles match your frontend

      return res.status(400).json({ message: 'Invalid role specified' });

    }

    // Hash the password

    const hashedPassword = await bcrypt.hash(password, 10);

    // Create a new user

    const newUser = new User({ username, phoneNumber,email, country, password: hashedPassword, role });

    await newUser.save();

    res.status(201).json({ message: 'User registered successfully' });

  } catch (error) {

    res.status(500).json({ message: 'Error registering user', error });

    console.error(error);

  }

};

// Login a user

exports.login = async (req, res) => {

  const { username, password } = req.body;

  try {

    // Find the user by username

    const user = await User.findOne({ username });

    if (!user) {

      return res.status(400).json({ message: 'Invalid username or password' });

    }

    // Compare passwords

    const isMatch = await bcrypt.compare(password, user.password);

    if (!isMatch) {

      return res.status(400).json({ message: 'Invalid username or password' });

    }

    // Generate a JWT token

    const token = jwt.sign({ username: user.username, role: user.role }, JWT\_SECRET, { expiresIn: '1h' });

    // Console log the username when login is successful

    console.log(`User logged in: ${user.username}`);

    // Respond with the token, role, and username

    res.status(200).json({

      message: 'Login successful',

      token,

      role: user.role,

      username: user.username

    });

  } catch (error) {

    console.error('Error logging in:', error); // Log the error for debugging purposes

    res.status(500).json({ message: 'Error logging in', error });

  }

};

## **5.31 orderController.js**

const Order = require('../models/order');

exports.placeOrder = async (req, res) => {

    try {

        const { sellerId, productName, price, category, quantity } = req.body;

        // Step 1: Check if an order exists for the same seller and product

            // Step 3: If the order does not exist, create a new order

            const newOrder = new Order({

                sellerId,

                productName,

                price,

                category,

                quantity

            });

            await newOrder.save();

        res.status(200).send('Order placed successfully');

    } catch (error) {

        console.error('Error placing order:', error);

        res.status(500).send('Error placing order');

    }

};

exports.getSalesReport = async (req, res) => {

    try {

        const { sellerId } = req.params;

        // Fetch orders for the specified sellerId

        const details = await Order.find({ sellerId });

        // Format the sales data

        const formattedSalesData = details.map(order => ({

            ...order.\_doc // Spread all fields from the order document

        }));

        // Respond with the formatted sales data

        res.status(200).json(formattedSalesData);

    } catch (error) {

        console.error('Error fetching sales report:', error);

        res.status(500).send('Error fetching sales report');

    }

};

exports.getTotalSales = async (req,res) => {

    try {

        const { username } = req.params;

        let total = 0

        let qty = 0

        // Fetch orders for the specified sellerId

        const products = await Order.find({sellerId:username})

        products.forEach(product => {

            total += parseInt(product.price) \* parseInt(product.quantity)

            qty += parseInt(product.quantity)

        })

        res.status(200).json({total,qty})

    }

    catch(err){

        console.log(err)

        res.send(500).json({message:"Internal Server Error"})

    }

}

## **5.32 productController.js**

const Product = require('../models/Product');

const multer = require('multer');

const path = require('path');

// Set up the Multer middleware

const storage = multer.diskStorage({

  destination: function (req, file, cb) {

    cb(null, 'uploads/');

  },

  filename: function (req, file, cb) {

    cb(null, Date.now() + '-' + file.originalname); // Generate a unique filename

  }

});

const upload = multer({ storage: storage, limits: { fileSize: 5 \* 1024 \* 1024 } }).single('image');

// Controller function to create a new product with image upload

exports.createProduct = async (req, res) => {

  upload(req, res, async (err) => {

    if (err) {

      return res.status(400).json({ message: 'Error uploading image', error: err });

    }

    try {

      console.log(req.body)

      const { name, price, description, category,seller } = req.body;

      const image = req.file ? req.file.path : ''; // Store the image path

      // Assuming req.user.\_id contains the authenticated seller's info

      const newProduct = new Product({ name, image, price, description, category, seller });

      await newProduct.save();

      res.status(201).json({ message: 'Product created successfully', product: newProduct });

    } catch (error) {

      res.status(500).json({ message: 'Error creating product', error });

      console.log(error )

    }

  });

};

// Fetch all products with their image URLs

exports.getAllProducts = async (req, res) => {

  try {

    const products = await Product.find();

    // Map products to include full image URL

    const productsWithImageURL = products.map(product => {

      return {

        ...product.\_doc,

        image: `${req.protocol}://${req.get('host')}/${product.image}` // Full image URL

      };

    });

    res.status(200).json(productsWithImageURL);

  } catch (error) {

    res.status(500).json({ message: 'Error fetching products', error });

  }

};

// Fetch products by category with their image URLs

exports.getProductsByCategory = async (req, res) => {

  try {

    const { category } = req.params;

    const products = await Product.find({ category });

    // Map products to include full image URL

    const productsWithImageURL = products.map(product => {

      return {

        ...product.\_doc,

        image: `${req.protocol}://${req.get('host')}/${product.image}` // Full image URL

      };

    });

    res.status(200).json(productsWithImageURL);

  } catch (error) {

    res.status(500).json({ message: 'Error fetching products by category', error });

  }

};

// Fetch products by a specific seller with their image URLs

exports.getProductsBySeller = async (req, res) => {

  try {

    const { username } = req.params; // Extract sellerId from the URL parameters

    // Fetch products for the specified sellerId

    const products = await Product.find({ seller:username });

    // Map products to include full image URL

    const productsWithImageURL = products.map(product => ({

      ...product.\_doc,

      image: `${req.protocol}://${req.get('host')}/${product.image}` // Full image URL

    }));

    res.status(200).json(productsWithImageURL); // Respond with the formatted products

  } catch (error) {

    console.error('Error fetching products by seller:', error);

    res.status(500).json({ message: 'Error fetching products by seller', error });

  }

};

## **5.33 userController.js**

const User = require('../models/User');

exports.fetchUserData = async (req, res) => {

    try {

        const username = req.params

        const user = await User.findOne(username)

        res.status(200).json(user);

    }

    catch(err){

        console.error('Error fetching user data:', err);

        res.status(500).send('Error fetching user data');

    }

}

exports.updateUserData = async (req,res) => {

    try {

        const username = req.params

        const {phoneNumber,email,country,desc} = req.body

        await User.updateOne(

            username

        ,{

            phoneNumber,

            email,

            country,

            desc

        })

        res.status(200).send('User data updated successfully')

    }

    catch(err){

        console.error('Error updating user data:', err);

        res.status(500).send('Error updating user data');

    }

}

exports.updateUserProfile = async (req,res) => {

    try {

        const username = req.params

        const {phoneNumber,email,country,desc} = req.body

        await User.updateOne(

            username

        ,{

            phoneNumber,

            email,

            country,

            desc

        })

        res.status(200).send('User data updated successfully')

    }

    catch(err){

        console.error('Error updating user data:', err);

        res.status(500).send('Error updating user data');

    }

}

## **5.34 authMiddleware.js**

// middleware/authMiddleware.js

const jwt = require('jsonwebtoken');

const User = require('../models/User');

module.exports = async (req, res, next) => {

  const token = req.header('Authorization')?.replace('Bearer ', '');

  if (!token) return res.status(401).json({ success: false, message: 'Access denied' });

  try {

    const decoded = jwt.verify(token, process.env.JWT\_SECRET);

    req.user = await User.findById(decoded.id);

    next();

  } catch (error) {

    res.status(401).json({ success: false, message: 'Invalid token' });

  }

};

## **5.35 credentials.js**

const allowedOrigins = require('../config/allowedOrigins')

const credentials = (req,res,next) => {

    const origin = req.headers.origin

    if(allowedOrigins.includes(origin)){

        res.header('Access-Control-Allow-Credentials', true)

    }

    next()

}

module.exports = credentials

## **5.36 order.js**

// models/saleModel.js

const mongoose = require('mongoose');

const saleSchema = new mongoose.Schema({

    sellerId: {

        type: String,

        required: true

    },

    productName: {

        type: String,

        required: true

    },

    price: {

        type: Number,

        required: true

    },

    category: {

        type: String,

        required: true

    },

    quantity: {

        type: Number,

        required: true

    },

    date: {

        type: Date,

        default: Date.now

    }

});

const Sale = mongoose.model('Sale', saleSchema);

module.exports = Sale;

## **5.37 product.js**

const mongoose = require('mongoose');

const productSchema = new mongoose.Schema({

  name: { type: String, required: true },

  image: { type: String, required: true }, // URL or file path for the image

  price: { type: Number, required: true },

  description: { type: String, required: true },

  category: {

    type: String,

    enum: ['clothing', 'accessories', 'sustainable'],

    required: true

  },

  seller: { type: String, ref: 'User', required: true }, // Link to seller

  createdAt: { type: Date, default: Date.now }

});

const Product = mongoose.model('Product', productSchema);

module.exports = Product;

## **5.38 user.js**

const mongoose = require('mongoose');

const userSchema = new mongoose.Schema({

  username: { type: String, required: true, unique: true },

  phoneNumber: { type: String, required: true, unique: true },

  email: { type: String, required: true, unique: true },

  country: { type: String, required: true },

  password: { type: String, required: true },

  desc: { type: String },

  role: { type: String, enum: ['seller', 'buyer', 'admin'], required: true }, // User role

  image: {type:String},

  createdAt: { type: Date, default: Date.now }

});

const User = mongoose.model('User', userSchema);

module.exports = User;

## **5.39 authRoutes.js**

// routes/authRoutes.js

const express = require('express');

const router = express.Router();

const authController = require('../controllers/authController');

// Register route

router.post('/register', authController.register);

// Login route

router.post('/login', authController.login);

module.exports = router;

## **5.40 orderRoutes.js**

const express = require('express');

const router = express.Router();

const orderController = require('../controllers/orderController');

router.post('/checkout', orderController.placeOrder);

router.get('/sales/total/:username', orderController.getTotalSales);

router.get('/sales-report/:sellerId', orderController.getSalesReport);

module.exports = router;

## **5.41 productRoutes.js**

const express = require('express');

const router = express.Router();

const productController = require('../controllers/productController');

const authMiddleware = require('../middleware/authMiddleware'); // Middleware for authentication

// Create a new product (only for sellers)

router.post('/', productController.createProduct);

// Get all products (optional)

router.get('/', productController.getAllProducts);

// Get products by category with image URLs

router.get('/category/:category', productController.getProductsByCategory);

// Get products by specific seller

router.get('/seller/:username', productController.getProductsBySeller);

module.exports = router;

## **5.42 userRoutes.js**

// routes/authRoutes.js

const express = require('express');

const router = express.Router();

const userController = require('../controllers/userController');

// Register route

router.get('/:username', userController.fetchUserData);

router.put('/:username', userController.updateUserData);

// Login route

module.exports = router;

## **5.43 server.js**

const express = require('express');

const mongoose = require('mongoose');

const bodyParser = require('body-parser');

const productRoutes = require('./routes/productRoutes');

const authRoutes = require('./routes/authRoutes'); // Import auth routes

const orderRoutes = require('./routes/orderRoutes');

const userRoutes = require('./routes/userRoutes')

const path = require('path');

const cors = require("cors");

const credentials = require('./middleware/credentials')

const corsOptions = require('./config/corsOptions')

const {verifyToken} = require('./middleware/authMiddleware')

require('dotenv').config();

const app = express();

const PORT = process.env.PORT || 3000;

app.use(credentials)

app.use(cors(corsOptions))

// Middleware

app.use(bodyParser.json());

app.use(bodyParser.urlencoded({ extended: true }));

// Serve static files from the 'uploads' directory

app.use('/uploads', express.static(path.join(\_\_dirname, 'uploads')));

// Routes

app.use('/auth', authRoutes); // Use auth routes

// app.use(verifyToken);

app.use('/products', productRoutes);

app.use('/api', orderRoutes);

app.use('/user', userRoutes);

// Connect to MongoDB

mongoose.connect(process.env.MONGODB\_URI, { useNewUrlParser: true, useUnifiedTopology: true })

  .then(() => {

    console.log('Connected to MongoDB');

    app.listen(PORT, () => console.log(`Server running on port ${PORT}`));

  })

.catch(error => console.error('Error connecting to MongoDB:', error));

## **5.44 awarness.html**

<!DOCTYPE html>

<html lang="en">

<head>

<meta charset="UTF-8">

<meta name="viewport" content="width=device-width, initial-scale=1.0">

<link rel="stylesheet" href="awareness.css">

<title>Get Involved | Nexus Bloom</title>

</head>

<body>

<header class="header">

<nav class="nav">

<ul class="nav-menu">

<li><a href="../homepage/homepage.html" class="nav-link">Home</a></li>

<li><a href="../about/aboutus.html" class="nav-link">About Us</a></li>

<li><a href="../donate/donate.html" class="nav-link">Donate</a></li>

<li><a href="../profile/profile.html" class="nav-link">Profile</a></li>

</ul>

</nav>

</header>

<div class="main-content">

<h1>Get Involved</h1>

<form id="involvement-form">

<div class="form-group">

<label for="activity">Choose an Activity</label>

<select id="activity" required>

<option value="" disabled selected>Select an activity</option>

<option value="beach-cleaning">Beach Cleaning Drive</option>

<option value="tree-planting">Tree Planting Campaign</option>

<option value="plastic-reduction">Plastic Waste Reduction Initiative</option>

</select>

</div>

<div class="form-group">

<label for="campaign-details">Select a Campaign</label>

<select id="campaign-details" required>

<option value="" disabled selected>Select campaign details</option>

</select>

</div>

<div class="form-group">

<label for="time-slot">Choose a Time Slot</label>

<select id="time-slot" required>

<option value="" disabled selected>Select a time slot</option>

<option value="morning">Morning Slot: 8 AM - 10 AM</option>

<option value="afternoon">Afternoon Slot: 12 PM - 2 PM</option>

<option value="evening">Evening Slot: 4 PM - 6 PM</option>

</select>

</div>

<div class="form-group">

<label for="name">Your Name</label>

<input type="text" id="name" placeholder="Enter your name" required>

</div>

<button type="submit">Submit</button>

</form>

</div>

<footer class="footer">

<div class="back-button">

<button onclick="window.location.href='../homepage/homepage.html'">Back to Home</button>

</div>

</footer>

<script>

const activitySelect = document.getElementById('activity');

const campaignDetailsSelect = document.getElementById('campaign-details');

const campaigns = {

'beach-cleaning': [

{ name: 'Beach Cleaning Drive 1', location: 'Juhu Beach, Mumbai', date: 'Sep 20, 2024' },

{ name: 'Beach Cleaning Drive 2', location: 'Versova Beach, Mumbai', date: 'Sep 25, 2024' },

{ name: 'Beach Cleaning Drive 3', location: 'Chowpatty Beach, Mumbai', date: 'Oct 5, 2024' }

],

'tree-planting': [

{ name: 'Tree Planting Campaign 1', location: 'Borivali National Park, Mumbai', date: 'Sep 22, 2024' },

{ name: 'Tree Planting Campaign 2', location: 'Aarey Colony, Mumbai', date: 'Sep 29, 2024' },

{ name: 'Tree Planting Campaign 3', location: 'Sanjay Gandhi National Park, Mumbai', date: 'Oct 7, 2024' }

],

'plastic-reduction': [

{ name: 'Plastic Waste Reduction 1', location: 'Marine Drive, Mumbai', date: 'Sep 23, 2024' },

{ name: 'Plastic Waste Reduction 2',…

[4:31 pm, 21/9/2024] Riddhi Kasar~🎐: aboiut us html- <!DOCTYPE html>

<html lang="en">

<head>

<meta charset="UTF-8">

<meta name="viewport" content="width=device-width, initial-scale=1.0">

<title>NexusBloom | About Us</title>

<link rel="stylesheet" href="aboutus.css">

</head>

<body>

<div class="container">

<header class="header">

<h1>About NexusBloom</h1>

</header>

<section class="intro">

<p>At NexusBloom, we envision a world where women can lead, inspire, and thrive. Through our unique platform, we provide a supportive space for women to grow their businesses, enhance their well-being, and build a safer future together.</p>

</section>

<section class="mission">

<h2>Our Mission</h2>

<p>Our mission is simple: to empower women by fostering a community of entrepreneurship, personal safety, and holistic health. By combining digital innovation with a community-centered approach, NexusBloom offers a space where women can not only sell products but also receive the guidance, resources, and support needed to thrive.</p>

</section>

<section class="values">

<h2>Our Core Values</h2>

<ul>

<li><strong>Empowerment:</strong> Every woman has the potential to succeed and flourish.</li>

<li><strong>Innovation:</strong> We leverage technology to build real-world solutions that make a difference.</li>

<li><strong>Community:</strong> Together, we create a network of support, growth, and collective impact.</li>

<li><strong>Sustainability:</strong> Committed to nurturing both individual and environmental well-being.</li>

</ul>

</section>

<section class="history">

<h2>Our Journey</h2>

<p>Founded in 2020, NexusBloom began as a small initiative aimed at helping local women entrepreneurs grow their businesses online. Over time, our platform has evolved into a comprehensive space, addressing women's safety and health alongside business growth. Today, NexusBloom serves as a vital link between women-driven businesses and their communities.</p>

</section>

<section class="community">

<h2>Our Community</h2>

<p>NexusBloom is powered by a dynamic network of women entrepreneurs and changemakers. We connect women with opportunities to sell sustainable and handmade products, while also giving back through local initiatives such as health campaigns and environmental projects. Our community is always growing, now with over 500,000 members actively participating in our movement.</p>

</section>

<section class="impact">

<h2>Our Impact</h2>

<p>From launching businesses to improving personal safety, NexusBloom has transformed lives. Through our platform, women have gained confidence in managing their businesses, staying healthy, and feeling safe in their environments. Our impact continues to grow as we engage with more communities and expand our range of services.</p>

</section>

<section class="get-involved">

<h2>Get Involved</h2>

<p>Want to be part of the movement? Join our community! Whether you are a seller, a volunteer, or simply a supporter, there are many ways to contribute. Explore our marketplace, participate in local activities, or support our environmental and health campaigns. Together, we can make a lasting impact.</p>

<button onclick="window.location.href='../donate/donate.html'">Donate Now</button>

<button onclick="window.location.href='../get-involved/awareness.html'">Volunteer</button>

</section>

<div class="back-button">

<button onclick="window.location.href='../homepage/homepage.html'">Back to Home</button>

</div>

</div>

</body>

</html>

## **5.45 forgotpassword.html**

<!DOCTYPE html>

<html lang="en">

<head>

<meta charset="UTF-8">

<meta name="viewport" content="width=device-width, initial-scale=1.0">

<title>Forgot Password</title>

<style>

body {

font-family: Arial, sans-serif;

display: flex;

justify-content: center;

align-items: center;

height: 100vh;

margin: 0;

}

.container {

width: 300px;

padding: 20px;

border: 1px solid #ccc;

border-radius: 5px;

box-shadow: 0px 0px 10px rgba(0, 0, 0, 0.1);

}

.hidden {

display: none;

}

input {

width: 100%;

padding: 10px;

margin: 10px 0;

box-sizing: border-box;

}

button {

width: 100%;

padding: 10px;

background-color: #28a745;

color: white;

border: none;

cursor: pointer;

}

button:hover {

background-color: #218838;

}

</style>

</head>

<body>

<div class="container">

<h2>Forgot Password</h2>

<!-- Enter token section -->

<div id="tokenSection">

<label for="token">Enter Token</label>

<input type="text" id="token" placeholder="Enter token here">

<button onclick="verifyToken()">Submit Token</button>

</div>

<!-- New password section (hidden initially) -->

<div id="passwordSection" class="hidden">

<label for="newPassword">New Password</label>

<input type="password" id="newPassword" placeholder="Enter new password">

<label for="confirmPassword">Confirm New Password</label>

<input type="password" id="confirmPassword" placeholder="Confirm new password">

<button onclick="submitNewPassword()">Submit New Password</button>

</div>

</div>

<script>

// Function to verify token

async function verifyToken() {

const token = document.getElementById('token').value;

try{

const response= await fetch("http://localhost:3000/auth/check-reset-token",

{

method: 'POST',

headers: {

'Content-Type': 'application/json',

},

body: JSON.stringify({ resetToken: token }),

}

);

const data = await response.json();

if(response.ok){

alert(data.message);

document.getElementById('tokenSection').classList.add('hidden');

document.getElementById('passwordSection').classList.remove('hidden');

}

else{

alert(data.message);

}

}

catch(err){

alert('Please enter a valid token');

}

// You can add your own token validation logic here

if (token) {

document.getElementById('tokenSection').classList.add('hidden');

document.getElementById('passwordSection').classList.remove('hidden');

} else {

alert('Please enter a valid token');

}

}

// Function to submit new password

async function submitNewPassword() {

const newPassword = document.getElementById('newPassword').value;

const confirmPassword = document.getElementById('confirmPassword').value;

const token = document.getElementById('token').value;

console.log(token)

if (newPassword && confirmPassword) {

if (newPassword === confirmPassword) {

// You can add your password reset logic here (e.g., send to server)

try{

const response = await fetch('http://localhost:3000/auth/reset-password', {

method: 'POST',

headers: {

'Content-Type': 'application/json',

},

body: JSON.stringify({ resetToken: token, newPassword: newPassword }),

});

const data = await response.json();

if(response.ok){

alert(data.message);

localStorage.removeItem("uname")

window.location.href = "../signin/index.html"

}

else{

alert(data.message);

}

}

catch(err){

alert(err)

}

} else {

alert('Passwords do not match');

}

} else {

alert('Please fill in both password fields');

}

}

</script>

</body>

</html>

# **6. Results**

## **6.1 Validation and Naming Conventions**

|  |  |  |  |
| --- | --- | --- | --- |
| **Sr No.** | **Control ID** | **Validation Used** | **Reason** |
| **1** | Full Name | Required, string, must be at least 3 characters long | Ensures a valid and complete name is provided |
| **2** | Username | Required, string, can contain special characters | Ensures username is unique |
| **3** | Password | Required, string, must be at least 8 characters long | Ensures password is unique and hard to guess |
| **4** | Phone Number | Required, must be 10 digits | Ensures valid phone number format |
| **5** | Country | Required, must be a valid country name | Ensures accurate location information |
| **6** | Email | Required, must contain @ and follow valid email format | Ensures proper email address is provided |

Table 2 Validation and Naming Conventions

## **6.2 ScreenShots**

### **6.2.1 Login Page**

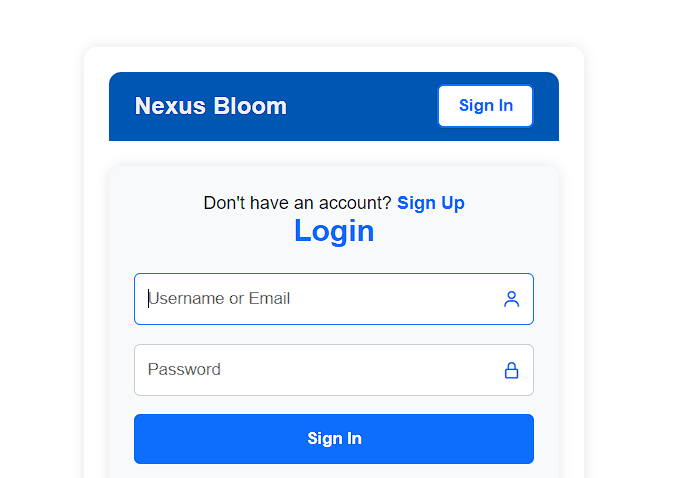


Figure Login Page

### **6.2.2 Signup Page**

![A screenshot of a login form

Description automatically generated

Figure 17 Signup Page

### **6.2.3 User Home Page**

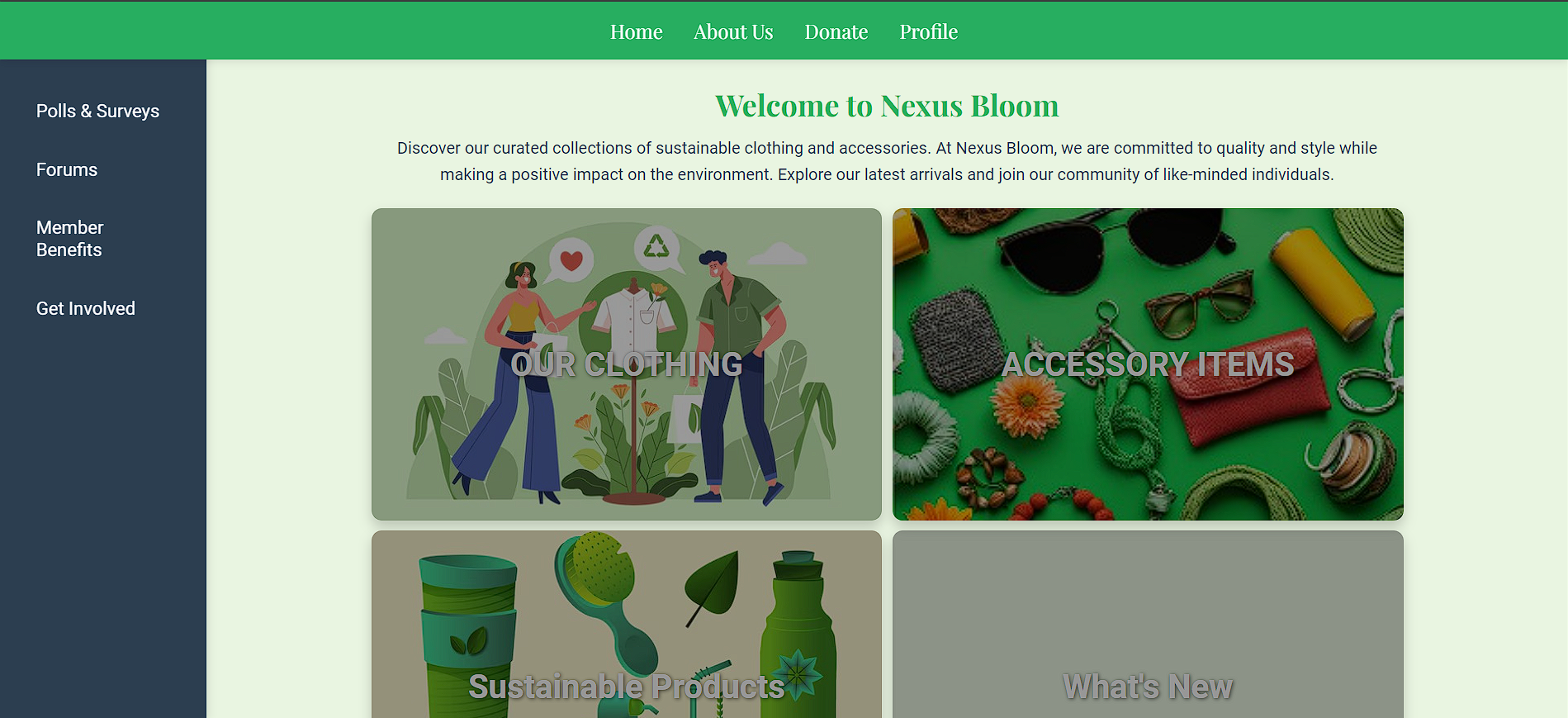


Figure 18 **User Home Page**

## **6.3.4 Clothing Page**

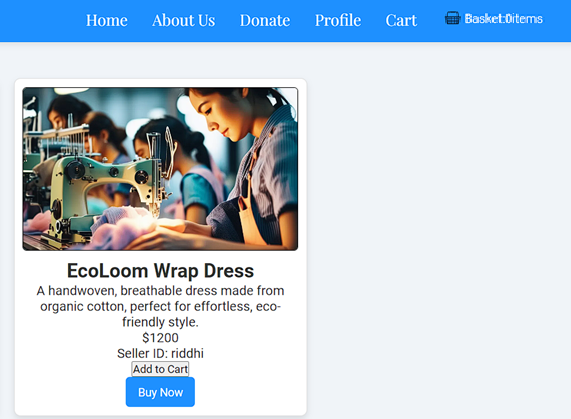
****

Figure 19 Clothing Page

### **6.3.5 Accessories Page**

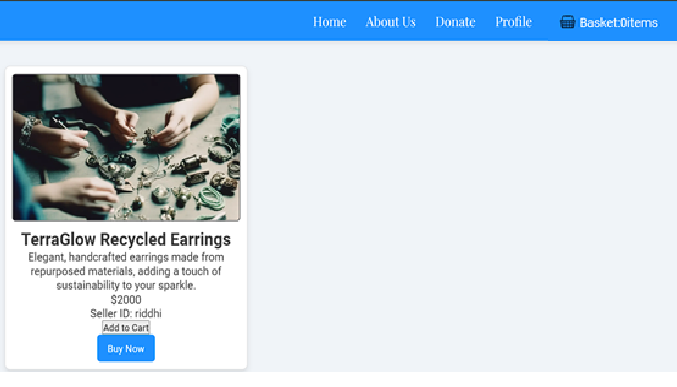


Figure 20 **Accessories** Page

### **6.3.6 Sustainable Items Page**

A screenshot of a computer

Description automatically generated

Figure 21 **Sustainable** Page

**6.3.7 Cart Page**

A screenshot of a computer

Description automatically generated

Figure 22 **Cart Page**

### **6.3.8 Buy Now Page**

A screenshot of a computer

Description automatically generated

Figure 23 **Cart Page**

### **6.3.9 Seller Home Page**

A screenshot of a computer

Description automatically generated

Figure 24 **Seller Home Page**

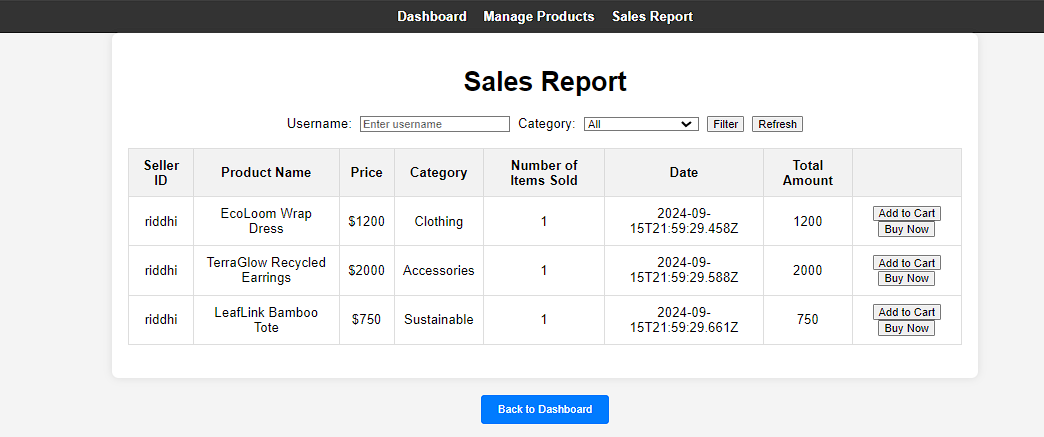
## **6.3.10 Manage Products Page**

A screenshot of a computer

Description automatically generated

Figure 25 **Manage Products Page**

## **6.3.11 Sales Report Page**



## **6.3.12 Profile Page(Seller and buyer)**

A screenshot of a user profile

Description automatically generated

Figure 26 **Page(Seller and buyer)**

**6.3.13 Polls Page**

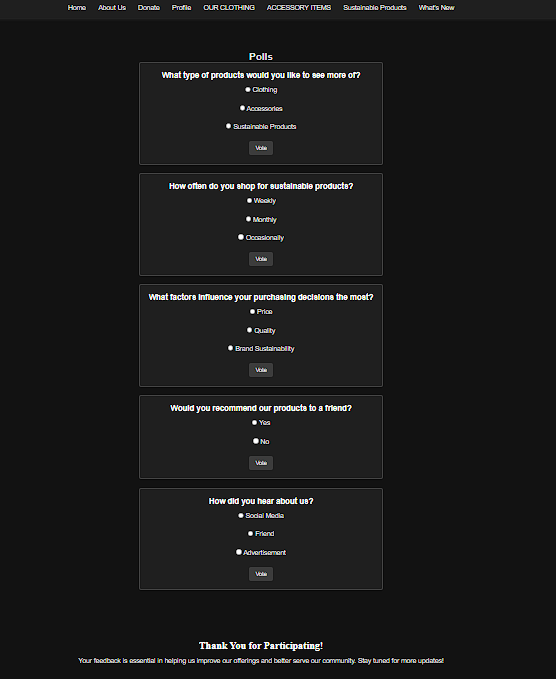


Figure 27 **Polls Page**

**6.3.14 Member Benefits**

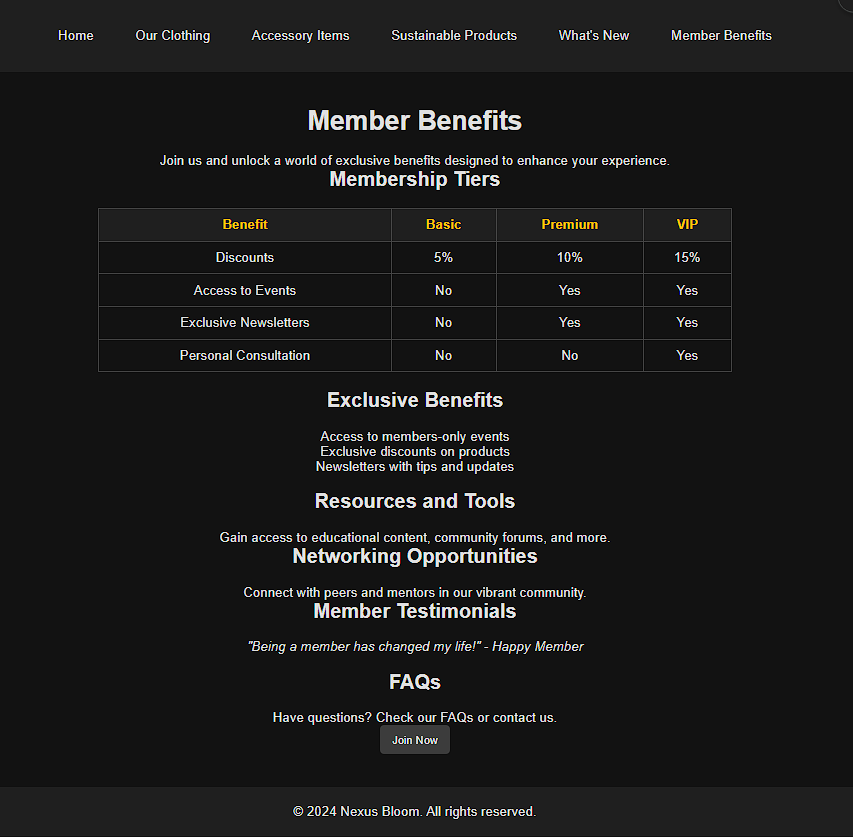


Figure 28 **Member Benefits Page**

**6.3.15 Donate page**

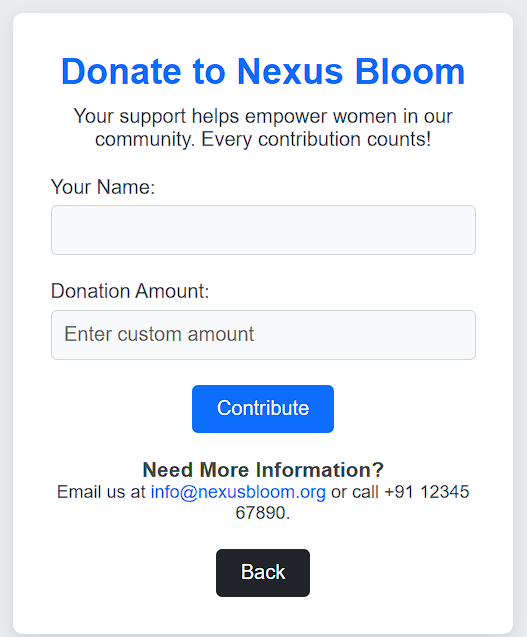


Figure 29 **Donate page**

## **6.3.16 About us page**

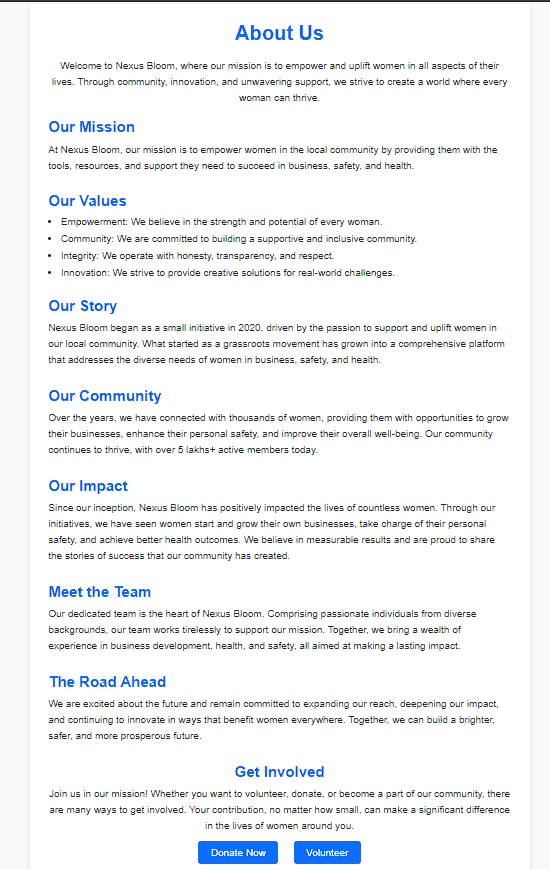


Figure 30 about us

## **6.3.17 What’s new page**

A screenshot of a computer

Description automatically generated

Figure 31 what's new page

## **6.3.18 Get involved page**

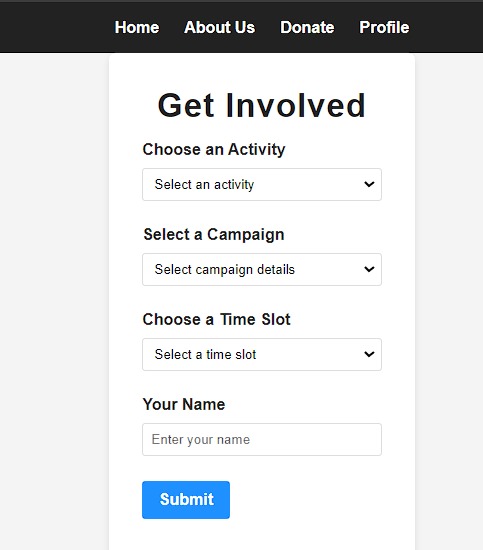
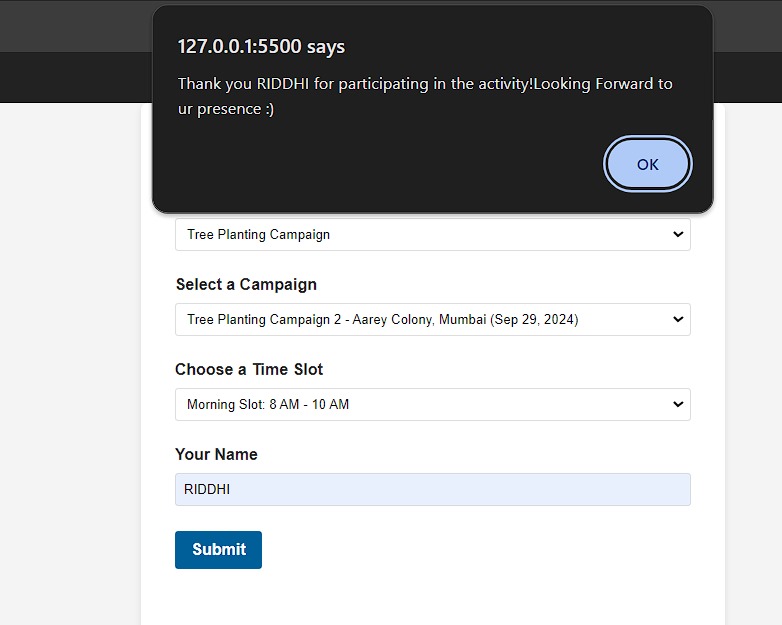


Figure 32 get involved page

6.3.19 getInvolved success page~~~~

## **6.3.20 forgotpass token page**

A login screen with green and black text

Description automatically generated

Figure 33 forgotpass token page

## **6.3.21 successful changepass page**

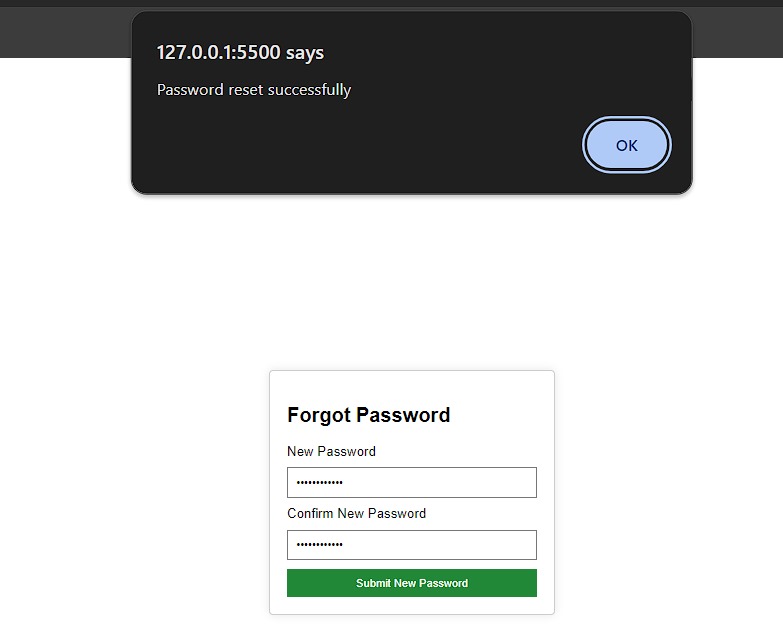


Figure 34 successful pass change page

## **6.3.22 start page**

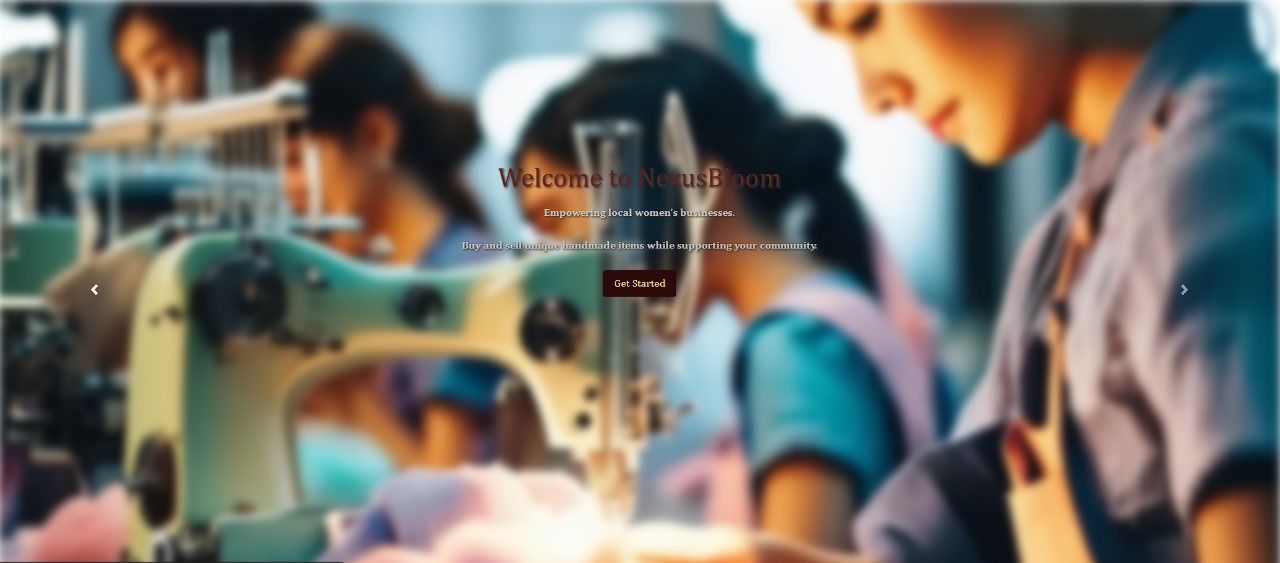


Figure 35 start page

# **7. Future Enhancements & Conclusion**

## **7.1 Future Enhancement**

* Advanced Search and Filtering: Implement advanced search capabilities and filtering options for products based on various attributes such as price range, customer ratings, and product tags. This will enhance the user experience by making it easier to find specific items.
* User Reviews and Ratings: Add a feature for users to leave reviews and ratings for products. This will help build trust and provide valuable feedback to sellers and potential buyers.
* Personalized Recommendations: Integrate machine learning algorithms to offer personalized product recommendations based on users' browsing history and purchase patterns.
* Mobile App Development: Develop a mobile application version of NexusBloom to provide a seamless shopping experience for users on the go and to leverage mobile-specific features like push notifications.
* Enhanced Security Features: Implement additional security measures such as two-factor authentication (2FA) and advanced encryption for sensitive user data and transactions to further protect against unauthorized access.
* Seller Analytics Dashboard: Provide an advanced analytics dashboard for sellers, offering insights into sales performance, customer behavior, and inventory management to help them optimize their operations.
* Internationalization and Localization: Expand the platform to support multiple languages and currencies, enabling international users to browse and make purchases in their preferred language and currency

## **7.2 Conclusion**

NexusBloom is a dynamic e-commerce platform designed to empower local women by providing a marketplace for their products, ranging from clothing and accessories to sustainable goods. With a focus on user-friendly features, secure transactions, and a supportive community, NexusBloom not only facilitates seamless shopping experiences but also promotes local entrepreneurship. As the platform continues to evolve, future enhancements will further enrich user interactions, enhance security, and expand its global reach, solidifying NexusBloom’s commitment to both its users and sellers.

# **8. References**

1. [MongoDB Documentation](https://www.mongodb.com/docs/): Guides on MongoDB setup and management for Sustainable website System database.
2. [NPM(Node Package Manager)](https://www.npmjs.com/): Hosts JavaScript packages needed for backend development
3. [ChatGPT Reference](https://chatgpt.com/c/00ce81f5-3bef-4ecd-a8e1-a53237a477e0): Contains guidance or clarification related to how Sustainable website works.
4. **Node.js with MongoDB – GeeksforGeeks**  
   *Guidance on connectin Node.js with MongoDB for building data-driven web applications.*
5. **HTML, CSS, and JavaScript: HTML5 Guide**:  
   *W3Schools – HTML Tutorial:*Frontend Development References

<https://www.w3schools.com/html>

**9. Annexures**

**9.1 Figures list**

[Figure 1 Class Diagram 9](#_Toc177730368)

[Figure 2 Class Diagram 10](#_Toc177730369)

[Figure 3 Use case diagram - seller 11](#_Toc177730370)

[Figure no 4 Use case Diagram – Transaction 11](#_Toc177730371)

[Figure 5 Use Case Diagram -Buying Process 12](#_Toc177730372)

[Figure 6 sequence diagram 13](#_Toc177730373)

[Figure 7 Activity Diagram – Buyer 14](#_Toc177730374)

[Figure 8 Activity Diagram -Seller 15](#_Toc177730375)

[Figure 9 State Diagram - user 16](#_Toc177730376)

[Figure 10 State Diagram- seller side 17](#_Toc177730377)

[Figure 11 Component diagram 19](#_Toc177730378)

[Figure 12 Deployment Diagram 20](#_Toc177730379)

[Figure 13 Database Design 20](#_Toc177730380)

[Figure 14 Signup Page 90](#_Toc177730381)

[Figure 15 **User Home Page** 91](#_Toc177730382)

[Figure 16 Clothing Page 91](#_Toc177730383)

[Figure 17 **Accessories** Page 92](#_Toc177730384)

[Figure 18 **Sustainable** Page 92](#_Toc177730385)

[Figure 19 **Cart Page** 93](#_Toc177730386)

[Figure 20 **Cart Page** 93](#_Toc177730387)

[Figure 21 **Seller Home Page** 94](#_Toc177730388)

[Figure 22 **Manage Products Page** 94](#_Toc177730389)

[Figure 23 **Profile Page(Seller and buyer)** 95](#_Toc177730390)

[Figure 24 **Polls Page** 96](#_Toc177730391)

[Figure 25 **Member Benefits Page** 97](#_Toc177730392)

[Figure 26 **Donate page** 97](#_Toc177730393)

[Figure 27 about us 98](#_Toc177730394)

**9.2 Table list**

[Table 1 Event Table 8](#_Toc177073046)

[Table 2 Validation and Naming Conventions 97](#_Toc177073047)