**LAVISH- A FANCY STORE**

# A PROJECT REPORT

***Submitted by***

**AKSHAYA T (920422205009)**

**MAHA SWETHA N (920422205057)**

***in partial fulfillment for the award of the degree of***

**BACHELOR OF TECHNOLOGY**

**IN**

**INFORMATION TECNOLOGY**



**DEPARTMENT OF INFORMATION TECHNOLOGY**

**KAMARAJ COLLEGE OF ENGINEERING AND TECHNOLOGY**

**(An Autonomous Institution - Affiliated to Anna University, Chennai)**

**K.VELLAKULAM, VIRUDHUNAGAR - 625 701**

**NOVEMBER 2024**

**KAMARAJ COLLEGE OF ENGINEERING AND TECHNOLOGY**

**(An Autonomous Institution- Affiliated to Anna University, Chennai)**

**K.VELLAKULAM, VIRUDHUNAGAR - 625 701**

**BONAFIDE CERTIFICATE**

Certified that the project report “**LAVISH-A FANCY STORE”** is the bonafide work **of “AKSHAYA T (920422205009), MAHA SWETHA N (920422205057)”** who carried out the project work under my supervision.

# Dr. E. VAKAIMALAR Ms. P. MAHALAKSHMI

**Head of the Department,**  **SUPERVISOR,**

Associate Professor, Assistant Professor,

Dept. of Information Technology, Dept. of Information Technology,

Kamaraj College of Engg & Tech, Kamaraj College of Engg &Tech, K.Vellakulam, K.Vellakulam,

Virudhunagar - 625 701. Virudhunagar - 625701.

# INTERNAL EXAMINER EXTERNAL EXAMINER

**Abstract**

Title: Lavish: A Full-Stack Fancy Store Website using the MERN Stack

This project is centered around building an e-commerce platform for Lavish, a real fancy store that sells a wide variety of products, including make-up items, skincare products, cleaning essentials, stationery items, toys, gifts, and more. Developed using the MERN stack, which includes MongoDB, Express.js, React, and Node.js, the platform provides users with a clear overview of the store’s offerings.

The website includes a registration system where new users can create accounts by filling in their personal details. Upon successful registration, the users are able to download a PDF catalog that provides detailed information about the store’s products. Registered users also enjoy a 10% discount when they visit the store in person to make a purchase.

The frontend, built with React, ensures smooth navigation and an engaging user experience, while the backend, powered by Node.js and Express.js, handles user authentication and the generation of downloadable PDFs. MongoDB Compass is used for local database interactions, and MongoDB Atlas is utilized for cloud data storage, providing a scalable and secure system for handling user data and product information.

This project demonstrates how the MERN stack can be effectively used to build a dynamic e-commerce website with catalog generation and discount offers for registered users, while maintaining secure and efficient backend operations.

**Acknowledgement**

I would like to express my heartfelt gratitude to all those who contributed to the successful completion of this project, " Using MERN Stack."

First and foremost, I would like to thank **KAMARAJ COLLEGE OF ENGINEERING AND TECHNOLOGY** for providing the resources and environment conducive to learning and innovation.And thankful for **DEPARTMENT OF INFORMATION TECHNOLOGY**.And I am especially grateful to my Head of the Department, **Dr.E.VAKAIMALAR**, whose guidance and support were invaluable throughout this project.

I would also like to extend my gratitude to **Dr.R.ARTHY** and **Ms. P. MAHALAKSHMI** , whose expertise and encouragement helped me navigate the complexities of using the MERN stack effectively.

**Table of Contents**

|  |  |  |  |
| --- | --- | --- | --- |
| **CHAPTER NO.** | | **TITLE** | **PAGE NO.** |
|  | [**ABSTRACT**](#_Abstract)  **[LIST OF FIGURES](#_List_of_Tables_1)** | | iii  vi |
|  |  | |  |
| **1** | | **INTRODUCTION** | **1** |
|  | | 1.1 HTML  1.2 CSS  1.3 JavaScript  1.4 MERN Stack | 1  2  3  4 |
| **2** | | **METHODOLOGY**  2.1 Objective  2.2 Problem Statement  2.3 Block Diagram  2.4 Module Explanation | **5**  **5**  **5**  **6**  **6** |
|  | |  |  |
| **3** | | **RESULTS AND DISCUSSION** | **11** |
|  | |  |  |
| **4** | | **CONCLUSION** | **18** |
|  | |  |  |
| **5** | | **REFERENCES** | **19** |
|  | |  |  |

**List of Figures**

|  |  |  |
| --- | --- | --- |
| **FIGURE NO.**  2.1 | **TITLE**  Process Flow | **PAGE NO.**  **6** |
| 3.1 | Sign-up page | 11 |
| 3.2 | successful registration | 12 |
| 3.3 | Home page | 14 |
| 3.4 | Shop page | 14 |
| 3.5 | Available products | 15 |
| 3.6 | Database storage | 16 |

**CHAPTER 1**

**INTRODUCTION**

**1.1 HTML: The Backbone of Web Content**

HTML, or HyperText Markup Language, is the fundamental building block of the web. It defines the structure of web pages by organizing and formatting content through a system of tags and elements. Each element in HTML plays a specific role, allowing developers to create rich, interactive, and structured documents that browsers can interpret and display.

**Basic Structure of HTML**

At its core, an HTML document consists of a few key elements:

* **<!DOCTYPE html>**: This declaration appears at the very top of an HTML file. It informs the web browser about the HTML version being used, ensuring that the document is rendered correctly.
* **<html>**: This is the root element of any HTML document. Everything contained within the <html> element is part of the web page.
* **<head>**: Within the <html> tag, the <head> element provides meta-information about the document. This section often includes:
  + The document’s title, which appears in the browser tab.
  + Links to stylesheets and other resources.
  + Metadata such as character encoding and viewport settings, essential for responsive design.
* **<body>**: This is where the visible content of the web page resides. All text, images, links, forms, and other elements meant for the user are contained within this section.

HTML, as a markup language, doesn’t control how things look—that’s where CSS comes in—but it lays the structural foundation for web pages, ensuring they’re correctly interpreted by browsers.

**1.2 CSS: The Style and Layout of the Web**

CSS, or Cascading Style Sheets, is the design language that controls how HTML elements are displayed on a web page. CSS enhances HTML by specifying how elements should appear—colors, layouts, fonts, spacing, and more—providing a way to create visually compelling and user-friendly web pages.

**Basic Structure of CSS**

* **Selectors**: The selector determines which HTML element(s) the styles apply to. Common types of selectors include:
  + Tag selectors (e.g., h1) apply styles to all instances of that tag.
  + Class selectors (e.g., .class-name) target elements with a specific class attribute.
  + ID selectors (e.g., #id-name) apply styles to the element with the corresponding ID.
* **Properties**: These specify which aspect of the element you want to style. Common properties include:
  + color: Controls the text color.
  + font-size: Adjusts the size of the text.
  + margin and padding: Define space around and inside an element.
* **Values**: Each property is assigned a value to dictate how the styling is applied. For example, the value blue might be assigned to color, and 16px to font-size.

CSS allows for the separation of content from presentation, which is a key principle in modern web development. This separation improves maintainability and enhances the flexibility of web design.

**1.3 JavaScript: Bringing Interactivity to Web Pages**

JavaScript is a high-level, dynamic, and interpreted programming language that enables interactivity on web pages. From form validation to dynamically loading content, JavaScript enhances the user experience by making web applications more engaging and responsive.

**Key Characteristics of JavaScript**

* **Interpreted Language**: Unlike compiled languages like C++ or Java, JavaScript is executed line-by-line by the browser or JavaScript engine, without needing to be pre-compiled.
* **Dynamic Typing**: JavaScript doesn't require you to define variable types explicitly. For instance, a variable can hold different types of data (numbers, strings, objects) without throwing errors.
* **Prototype-Based**: JavaScript uses a prototype-based inheritance model. Instead of traditional classes and inheritance like in other object-oriented languages, objects in JavaScript can inherit properties and methods directly from other objects.
* **Event-Driven**: One of the most powerful aspects of JavaScript is its ability to handle asynchronous events. It can listen for user actions (like clicks, scrolls, and form submissions) and respond in real-time, enabling highly interactive web experiences.

JavaScript is the backbone of web interactivity. Alongside HTML and CSS, it forms the core of front-end web development. With the addition of frameworks like React, Angular, and Vue.js, JavaScript has become integral to building modern, complex web applications.

**1.4 MERN Stack: Full-Stack JavaScript Development**

The MERN stack is a popular full-stack JavaScript framework used to build web applications. The stack comprises MongoDB, Express.js, React, and Node.js, all working together to streamline the development process for both client-side and server-side applications.

**Components of the MERN Stack**

* **MongoDB**: A NoSQL database that stores data in JSON-like documents. MongoDB offers flexibility and scalability, which are essential for modern web applications that handle large amounts of data.
* **Express.js**: A web application framework for Node.js that simplifies the process of building server-side logic. Express helps manage HTTP requests, routing, and middleware to create robust and scalable back-end applications.
* **React**: A popular front-end JavaScript library developed by Facebook. React uses components and a virtual DOM to efficiently update and render views, making it ideal for building fast, dynamic user interfaces.
* **Node.js**: A runtime environment that allows developers to execute JavaScript on the server. It’s designed to be lightweight and efficient, enabling real-time applications that require a large number of simultaneous connections, such as chat apps or streaming services.

The MERN stack’s greatest advantage is that JavaScript is used throughout, enabling developers to work on the full stack using one language. This simplifies the development process and improves the communication between the front-end and back-end, making it an attractive choice for building modern web applications.

4o

**CHAPTER 2**

**METHODOLOGY**

**2.1. Objective**

The primary objective of the **Lavish** fancy store website is to create an engaging and user-friendly platform that effectively showcases the diverse range of products available in the store.

This website aims to enhance the shopping experience for users by providing the following key features:

* + 1. **Product Showcase**: To display a wide array of products, including make-up items, skincare products, cleaning essentials, stationery items, toys, and gifts, organized into categories for easy navigation. The website will allow users to explore and learn about the products offered by the store.
    2. **User Registration with Incentives**: To implement a user registration system that incentivizes users to register by offering a **10% discount** on in-store purchases. This discount not only encourages user engagement but also fosters customer loyalty and increases the likelihood of repeat visits to the store.
    3. **PDF Catalog Generation**: To provide users with an automatic download of a PDF catalog containing product information upon successful registration. This feature adds value to the user experience by giving registered users immediate access to product details and special offers.

**2.2. Problem Statement**

In the competitive landscape of retail, providing a seamless and attractive shopping experience is essential for driving customer engagement and satisfaction. The **Lavish** store faced several challenges:

* + 1. **Visibility of Products**: Customers often find it challenging to browse through a wide range of products effectively. Without a structured platform, potential customers may overlook items of interest, leading to missed sales opportunities.
    2. **User Engagement**: Attracting and retaining customers in a digital marketplace requires innovative solutions. The lack of a registration incentive can lead to lower conversion rates, as users may not feel compelled to sign up or revisit the site.
    3. **Information Accessibility**: Many customers prefer having tangible information about products to reference. Without a downloadable catalog, users might struggle to recall product details after their visit, affecting their decision-making process.
  1. **Block Diagram**

**Figure 2.1 Process Flow**

* 1. **Module Explanation**

**2.4.1. User Registration Module**

**Objective**: To facilitate user registration and manage user data securely.

* **Components**:
  + **Registration Form**: Collects user information such as name, email, phone number, password, and confirm password. Input validation checks are implemented to ensure data accuracy (e.g., matching passwords and valid email formats).
  + **Backend API**: An Express.js route handles registration requests. It checks for existing users to prevent duplicates, hashes passwords using bcrypt.js for security, and stores user data in the MongoDB database.
  + **PDF Generation**: Upon successful registration, the system generates a PDF catalog using libraries like pdfkit or jsPDF, which is automatically sent to the user for download.
* **Functionality**:
  + Users can register to create an account.
  + Registration process ensures data validation and security.
  + New users receive a downloadable PDF catalog upon successful registration.

**2.4.2 User Authentication Module**

**Objective**: To manage user sessions and secure access to the website features.

* **Components**:
  + **Login Form**: Allows users to enter their credentials (username and password) to log into their accounts.
  + **JWT Authentication**: After successful login, a JSON Web Token (JWT) is issued to the user, allowing secure access to protected routes within the application.
  + **Session Management**: The frontend uses the JWT to maintain user sessions, ensuring users remain logged in until they log out.
* **Functionality**:
  + Users can log in with registered credentials.
  + The system validates user credentials and generates a JWT for session management.
  + JWT ensures secure access to the application, enhancing user data protection.

**2.4.3 Product Catalog Module**

**Objective**: To display a wide range of products available in the store.

* **Components**:
  + **Product Database**: Stores product details, including images, descriptions, prices, and categories, in MongoDB.
  + **Product API**: Express.js routes to fetch product data from the database. The frontend makes GET requests to retrieve and display products dynamically.
  + **Category Navigation**: Allows users to browse products by categories such as make-up, skincare, stationery, etc.
* **Functionality**:
  + Users can view products organized into categories.
  + Products are displayed dynamically based on user interactions.
  + Smooth navigation allows users to explore various product types easily.

**2.4.4 PDF Catalog Module**

**Objective**: To create and provide a downloadable PDF catalog of the store's products.

* **Components**:
  + **PDF Generation Library**: Utilizes libraries such as pdfkit or jsPDF to create PDF files dynamically based on product information.
  + **Catalog API**: An API endpoint that handles PDF generation requests, triggered after user registration.
* **Functionality**:
  + Generates a PDF catalog containing product details for new users upon registration.
  + Ensures easy access to product information in a downloadable format, enhancing user engagement.

**2.4.5 Frontend Module**

**Objective**: To build a responsive and user-friendly interface for the Lavish website.

* **Components**:
  + **React Components**: Individual components for different functionalities such as registration forms, product displays, and navigation bars.
  + **State Management**: Utilizes React’s state management to handle dynamic rendering of product data and user session states.
  + **Responsive Design**: Ensures the website is accessible and visually appealing on various devices using CSS frameworks or media queries.
* **Functionality**:
  + Provides an intuitive and engaging user interface.
  + Supports smooth transitions and interactions, improving overall user experience.
  + Dynamically updates content based on user actions (e.g., viewing products, registering, logging in).

**2.4.6 Backend Module**

**Objective**: To manage the server-side logic and data storage for the Lavish website.

* **Components**:
  + **Node.js Server**: Handles incoming requests from the frontend, processes them, and sends back responses.
  + **Express.js Framework**: Facilitates routing and middleware management for handling API requests.
  + **MongoDB Database**: Stores user and product data, allowing efficient data retrieval and management.
* **Functionality**:
  + Manages API endpoints for user registration, login, product retrieval, and catalog generation.
  + Ensures secure data transactions between the client and server.
  + Facilitates easy data management and retrieval using MongoDB.

**CHAPTER 3**

**RESULTS AND DISCUSSION**

**3.1. User Registration and PDF Catalog Generation**

One of the standout features of the Lavish website is the **user registration system** combined with an automatic PDF download of the store catalog. After successful registration, a PDF file containing the product catalog is generated and made available for download.

* **Registration Form**: The form collects user data such as **name, email, phone number, password, and confirm password**. **Validation** is performed on the frontend to ensure proper data entry, including matching passwords and validating email formats (Figure 3.1).

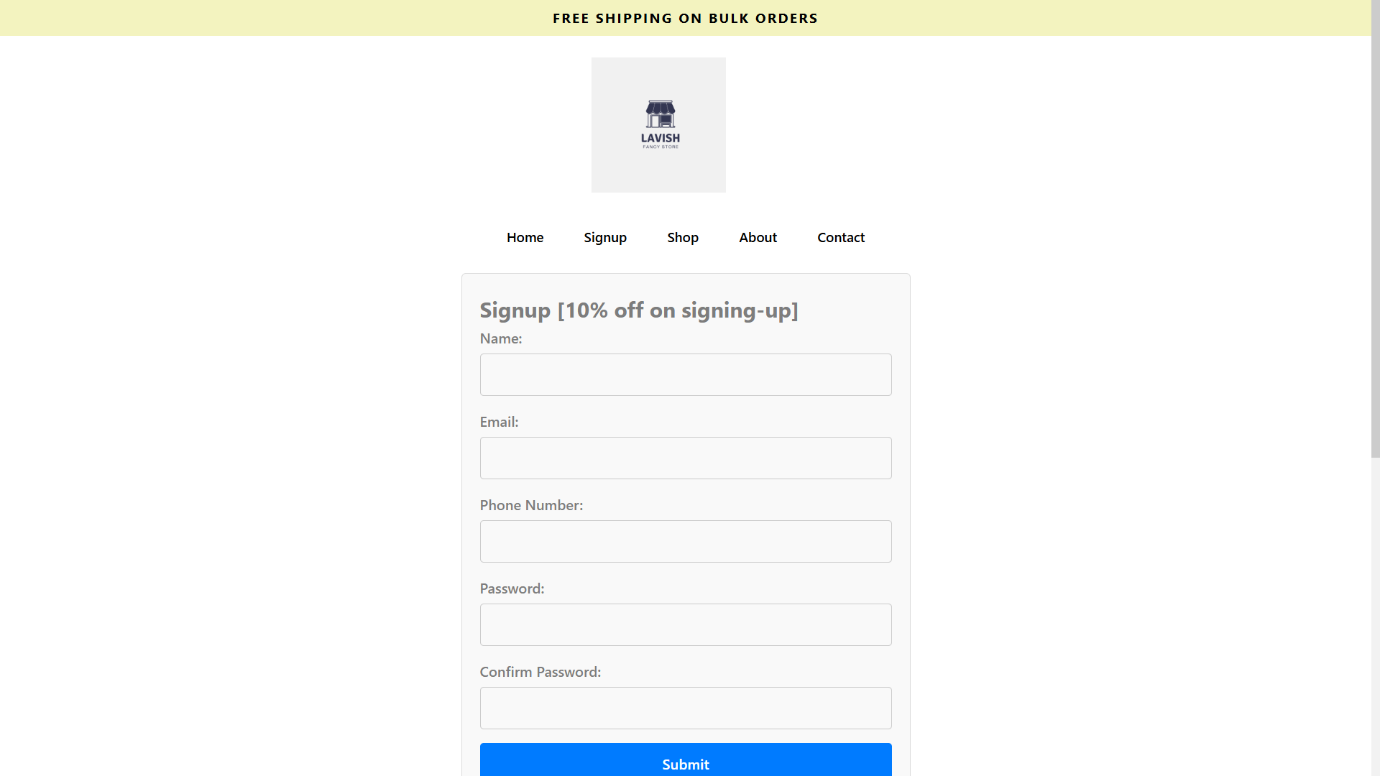


Figure 3.1 Sign-up page

* **PDF Catalog**: Upon successful registration, a **PDF document** is dynamically generated and provided to the user. This PDF includes details about the available products, product categories, and the store's offerings

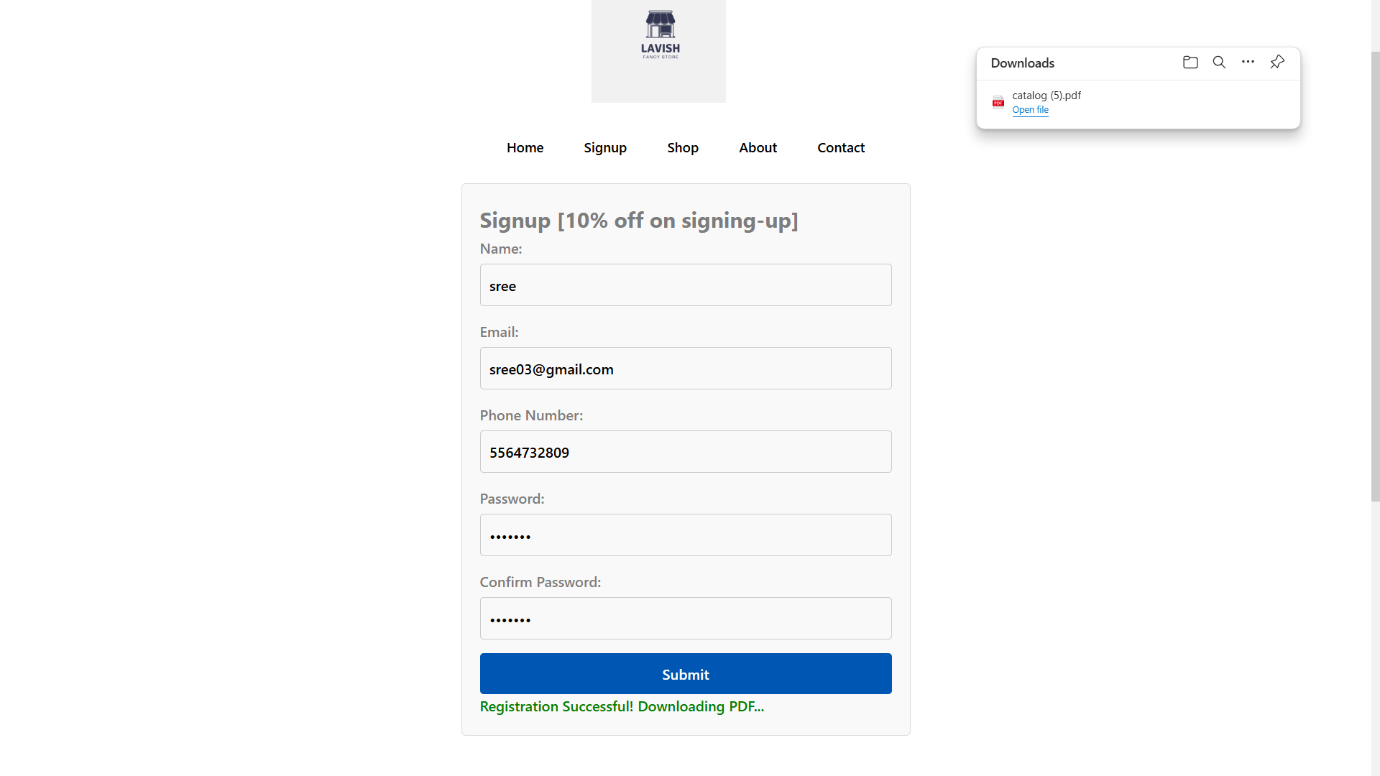


Figure 3.2 Successful registration

**Discussion**: The registration process is smooth, with no noticeable delays. Frontend validation ensures that users do not submit incorrect or incomplete data. The automated PDF catalog generation enhances the user experience by providing immediate access to product details after signing up, promoting user engagement with the store.

**3.2. Discount Mechanism for Registered Users**

Registered users receive a **10% discount** on in-store purchases. While this feature is not fully integrated into the online system, the discount is part of the business logic implemented to track registered users.

* **User Authentication**: Once a user registers, their data is stored in the **MongoDB database**, and they are tagged as eligible for the discount when they visit the store. Future iterations could introduce online shopping, where users can apply their discounts directly at checkout.
* **Discussion**: This feature offers a significant incentive for users to register, encouraging repeat business. The backend system, secured by JWT, guarantees that only valid users receive the discount, ensuring smooth in-store verification.

**3.3. Product Display and Catalog Organization**

The website organizes products into clear categories such as **make-up items, skincare products, cleaning essentials, stationery items, toys, and gifts**. Products are displayed on category pages, allowing users to browse through the items easily.

* **Dynamic Content Rendering**: The products are retrieved from the **MongoDB database** and dynamically displayed on the frontend. The **React.js** frontend handles the rendering of this data efficiently, ensuring a fast and smooth user experience.
* **Real-time Product Data Fetching**: **Express.js** handles the API requests for fetching product data, and the data is served to the frontend via **GET requests**. Users can browse various categories with minimal load times, thanks to the asynchronous nature of the API calls (Figure 3.4).

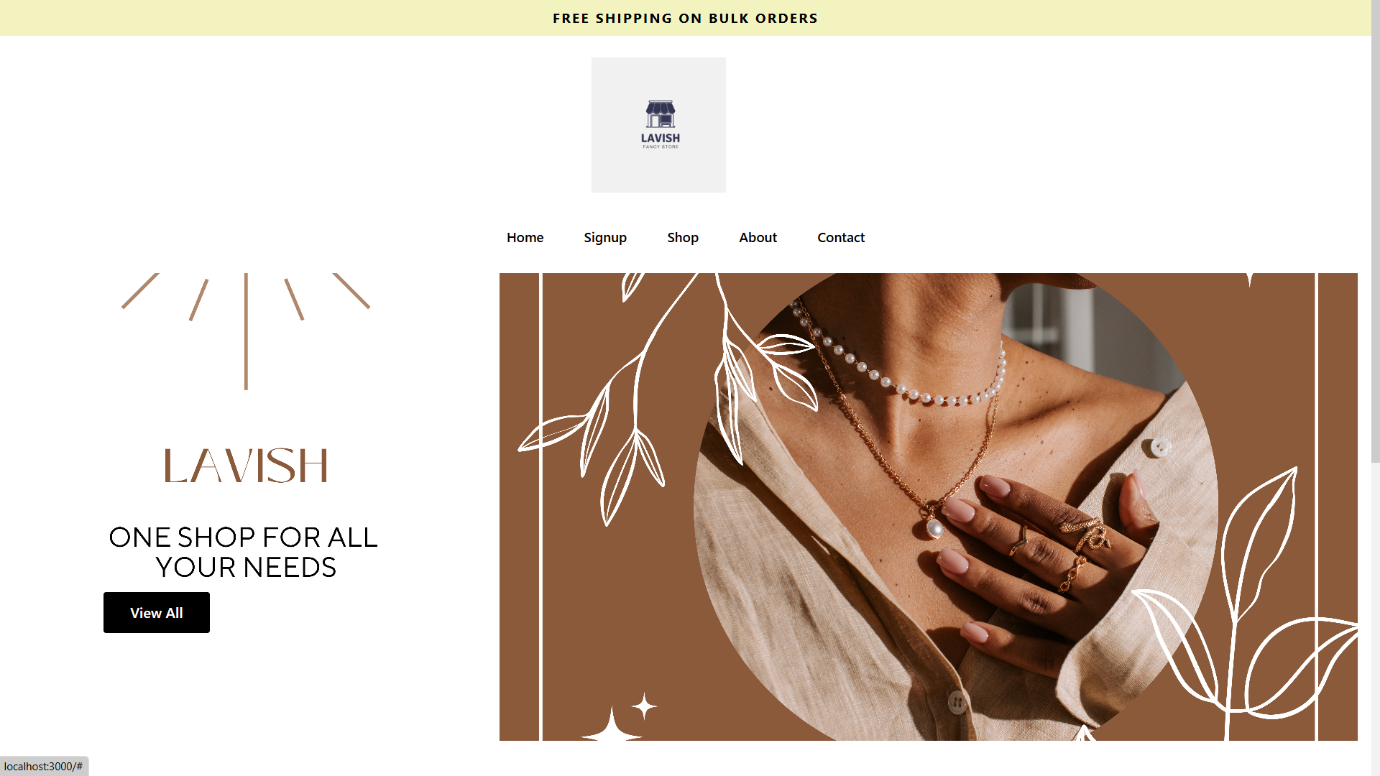


Figure 3.3 Home Page

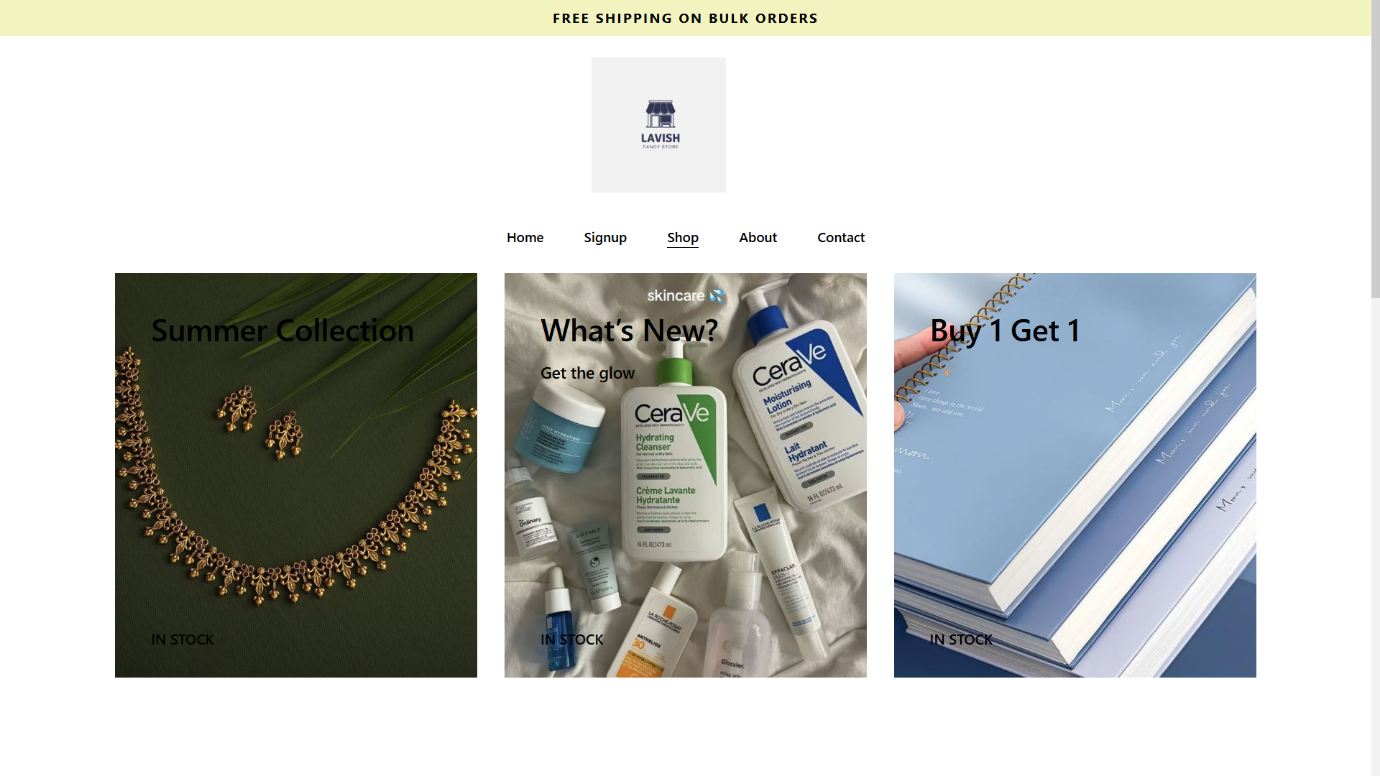


Figure 3.4 Shop page

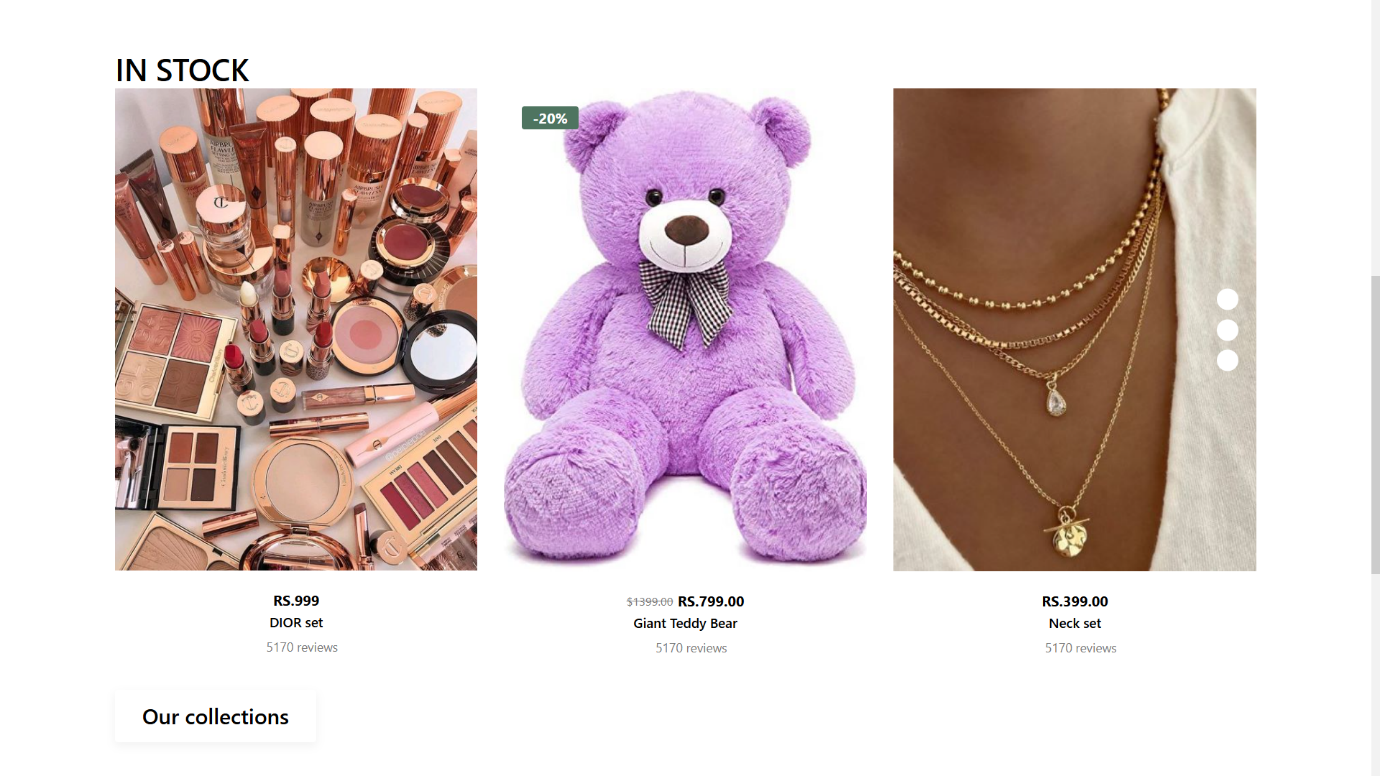


Figure 3.5 Available products

**Discussion**: The clear categorization and fast, dynamic rendering of product data make browsing seamless. Future expansions of the product catalog can easily be managed through MongoDB, making the system scalable for future growth. This also ensures that Lavish can introduce new products or seasonal items without requiring major changes to the codebase.

**3.4. Database Integration and Security**

The Lavish website relies on **MongoDB** for storing user registration data and product details. **MongoDB Compass** was used to visualize and manage the database during development.

* **User Data Security**: User information, especially passwords, is stored securely using **bcrypt.js** to hash passwords before they are saved in MongoDB. This ensures that sensitive user data is not exposed in the event of a data breach.
* **Scalable Database**: Using **MongoDB Atlas** for cloud storage provides flexibility and scalability. The store can grow its user base and product catalog without facing database performance issues(Figure 3.6).

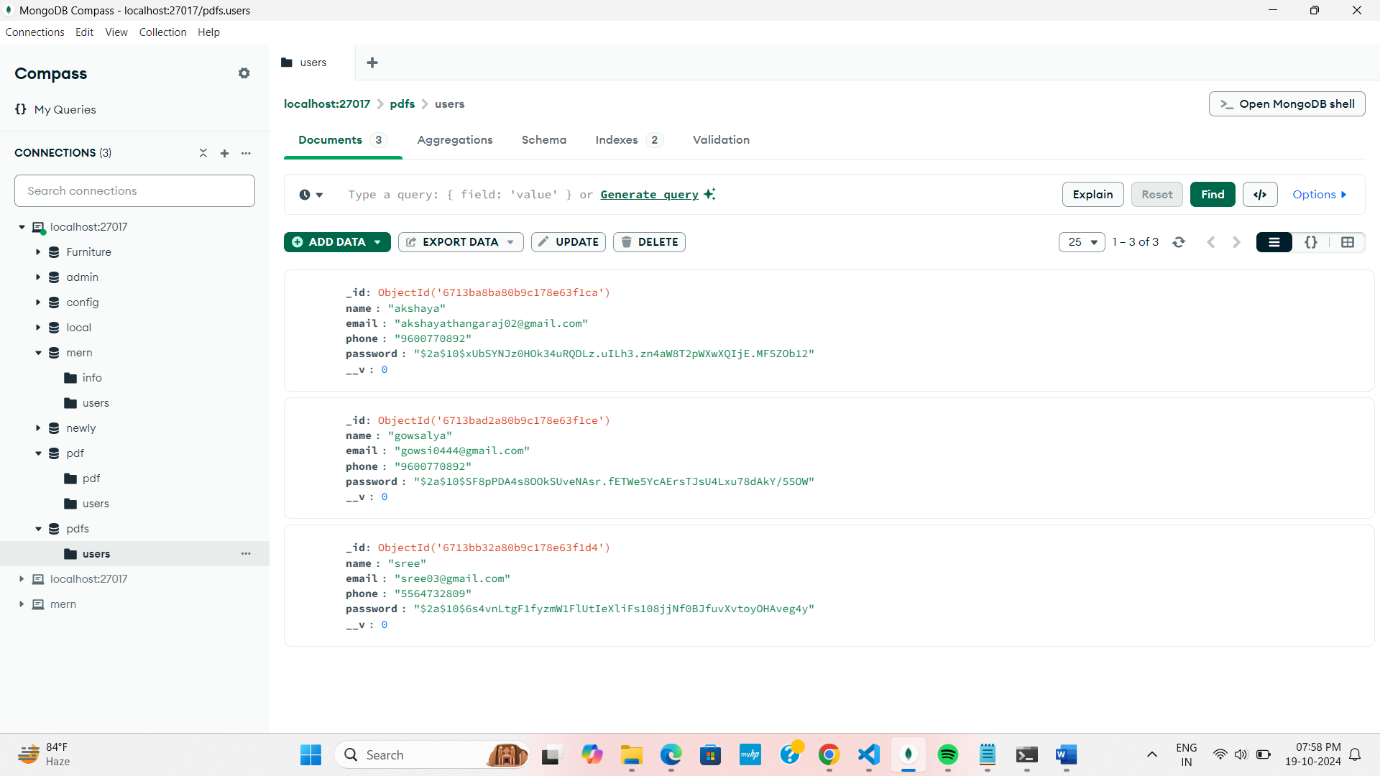


Figure 3.6 Database storage

**Discussion**: The use of bcrypt.js for password hashing ensures security, adhering to modern best practices in web development. MongoDB Atlas allows the application to scale smoothly as user registrations increase, ensuring future-proofing.

**3.5. Performance and User Experience**

React’s ability to manage the user interface dynamically enhances the overall performance of the Lavish website. The following points highlight the results:

* **Fast Rendering**: React’s **Virtual DOM** ensures that only necessary components are updated when data changes, leading to a highly responsive user interface.
* **Seamless User Experience**: The site navigation, from viewing product categories to registering for the discount, is smooth and intuitive.

**Discussion**: The choice of React for the frontend ensures a high-performance web application. Users can browse products and register without experiencing slow load times, making the platform user-friendly.

**3.6. Testing and Debugging**

Several tresting approaches were used to ensure that the website functions reliably:

* **Unit Testing**: Individual React components, such as the registration form and product category display, were tested to ensure proper functionality. This helped catch edge cases, such as invalid user input or missing product data.
* **API Testing**: The **RESTful APIs** were tested using **Postman**, ensuring that data from MongoDB is fetched and delivered to the frontend accurately. The registration endpoint was also tested to verify that user data is stored correctly and that the PDF catalog generation triggers upon registration.
* **End-to-End Testing**: The overall flow, from user registration to catalog download, was tested across multiple devices to ensure compatibility and smooth operation.

**Discussion**: Rigorous testing ensured that the application functions as expected, even under edge cases. Testing APIs with tools like Postman helped identify and resolve potential data issues early, improving the overall stability of the system.

**CHAPTER 4**

**CONCLUSION**

The **Lavish** fancy store website, built using the MERN stack (MongoDB, Express.js, React, and Node.js), successfully provides an engaging and secure platform for users to explore a wide variety of products. The key achievements of the project include:

* **Dynamic and Responsive Frontend**: The React-based frontend provides a seamless user experience, with fast loading times and efficient rendering of product data. The ability to dynamically update the product catalog ensures that users can browse different categories effortlessly.
* **User Registration and PDF Catalog**: The successful implementation of user registration combined with an automatic PDF catalog download enhances the overall user experience. The catalog provides valuable information about the store’s products, encouraging users to visit the store.
* **Secure User Data Handling**: By using bcrypt.js for password hashing and JWT for authentication, the application ensures that sensitive user data is handled securely, following industry best practices.
* **Scalable Backend**: The Node.js and Express.js backend, integrated with MongoDB Atlas, ensures the platform is scalable for future growth, whether that’s more users registering or an expanding product catalog.
* **Discount Mechanism**: The backend system efficiently tracks registered users and applies the 10% discount for in-store purchases, providing a real-world incentive for online engagement.

In conclusion, the Lavish website offers a comprehensive and interactive shopping experience, ensuring that users can easily engage with the store’s product offerings and enjoy exclusive benefits. The MERN stack proved to be an ideal solution for creating a flexible, secure, and scalable e-commerce platform

**REFERENCES**

1. **React.js Documentation**: Official documentation for React.js, covering key concepts like component-based design, hooks, and the Virtual DOM. Available at: https://reactjs.org/docs/getting-started.html.
2. **Node.js Documentation**: Node.js documentation offers insights into backend development using JavaScript, crucial for handling server-side logic. Available at: https://nodejs.org/en/docs/.
3. **Express.js Documentation**: The Express.js framework was used to create the RESTful APIs for the backend. Documentation is available at: <https://expressjs.com/>.
4. **Postman for API Testing**: Postman was used to test the RESTful APIs and ensure smooth data flow between the frontend and backend. Available at: <https://www.postman.com/>.
5. **MongoDB Compass**: Used to manage and visualize the MongoDB database locally. Available at: <https://www.mongodb.com/products/compass>.