MINI PROJECT

COURSE CODE - ENSI152

GROUP CODE-Y1-2024-25-G282

Project Report Submitted to

K. R. Mangalam University

Post Graduation

In

Master Of Computer Application

Submitted by

Rishav Kumar Mishra (2401560049)

Adarsh Kumar Jha (2401560035)

Sanskar Saurabh (2401560081)

Ved Prakash Mandal (2401560086)

Under The Supervision Of

Kirti Sharma

School of Engineering & Technology

K. R. MANGALAM UNIVERSITY

Sohna, Haryana 122103, India

Acknowledgement

We are deeply grateful to K. R. Mangalam University for providing the opportunity and resources to complete our project, "Stationary E-commerce Platform".

We sincerely thank our mentor, Ms. Kirti Sharma, for her valuable guidance and support throughout the project. Her advice was crucial to the successful development of the platform.

We also acknowledge the efforts and teamwork of our group members:

- Rishav Kumar Mishra
- Adarsh Kumar Jha
- Sanskar Saurabh
- Ved Prakash Mandal

Their dedication and collaboration ensured the completion of this MERN stack-based project.

Finally, we appreciate the tools and technologies like MongoDB, Express.js, React.js, Node.js, and Razorpay, which played a key role in the project's success.

Project Report: Stationary E-commerce Platform

1. Introduction

In the modern digital era, the demand for online shopping platforms has skyrocketed, covering a wide range of product categories including stationary products. This project report details the planning, development, and implementation of a full-stack e-commerce website dedicated specifically to stationary items. The platform has been developed using the MERN stack — MongoDB, Express.js, React.js, and Node.js — offering a robust, scalable, and high-performance solution for both users and administrators.

The primary objective of this project is to create an intuitive, responsive, and feature-rich platform that allows users to easily browse, search, and purchase various stationary products. The system ensures a smooth user experience through functionalities like real-time product filtering, secure user authentication, seamless payment integration, and order management. Special emphasis has been given to providing a safe and trustworthy online environment by incorporating secure login systems and payment processing via Razorpay, one of India's leading payment gateway solutions.

The platform has two main user roles: Customers and Administrators. Customers can register on the site, log in securely, browse the catalog of available stationary items, add products to their cart, and proceed to checkout. During checkout, users are redirected to a secure Razorpay payment gateway, where transactions are completed safely. After successful payment, users can view their order history and details under their profiles.

On the other hand, administrators have exclusive access to the backend management features of the platform. Admin users can add new products, update existing product details, manage inventory, and monitor customer orders. A separate admin dashboard is designed to enable easy tracking and management of sales, product availability, and order status. This ensures that the website remains updated in real-time with the latest stock and information.

The architecture of the project is designed following the principles of modularity and separation of concerns. The front-end of the application, built using React.js, provides a highly interactive and responsive user interface. React's component-based architecture ensures that the codebase remains organized and scalable for future enhancements. The back-end, developed using Node.js and Express.js, manages the server-side logic, RESTful API routes, and database interactions. MongoDB, a NoSQL database, is used to store user data, product information, and order details efficiently.

Security is a major concern in e-commerce platforms, and appropriate measures have been taken to safeguard the platform. User authentication is implemented using JSON Web Tokens (JWT) and password hashing via bcrypt, ensuring that user credentials are stored and transmitted securely. Additionally, all sensitive data related to payment processing is handled externally by Razorpay’s secure API endpoints, further enhancing the system's trustworthiness.

In conclusion, this project aims not only to deliver a functional e-commerce website for stationary products but also to demonstrate the effective use of modern web development technologies in building real-world applications. It emphasizes creating a seamless experience for users while providing powerful management tools for administrators. Future improvements can include integrating advanced features such as wishlists, product reviews, discount coupons, and mobile app compatibility to further enhance user engagement and business growth.

2. Project Goals and Objectives

• **Develop a Robust and Scalable Platform:**

Create a full-stack e-commerce website specifically for stationary products, capable of handling growing user traffic and product inventory.

• **Implement Secure User Authentication:**

Design secure login, registration, and forgot password functionalities using JWT for token-based authentication and bcrypt for password encryption.

• **Enable Product Filtering:**

Allow users to filter stationary products based on specific categories (e.g., pens, notebooks, art supplies) and price ranges to enhance the shopping experience.

• **Integrate a Shopping Cart System:**

Develop a dynamic shopping cart feature where users can add, update, and remove items before proceeding to checkout.

• **Incorporate a Secure Payment Gateway:**

Integrate Razorpay for safe and seamless transaction processing, ensuring a trusted environment for online payments.

• **Develop an Administrative Panel:**

Create a secure admin dashboard enabling administrators to:

o Manage product categories.

o Create, update, and delete product listings.

o View and manage customer orders efficiently.

• Ensure Data Security and Privacy:

Protect all user and transaction data through secure coding practices and third-party payment handling.

3. Technology Stack

The development of the stationary e-commerce platform relied on a combination of modern web technologies that collectively contributed to a scalable, maintainable, and efficient full-stack application. Each layer of the stack — frontend, backend, database, and deployment — was carefully selected to ensure high performance, security, and an excellent user experience. Below is a detailed explanation of the technology stack used:

Frontend Technologies

The frontend plays a critical role in delivering a responsive and dynamic interface that users interact with directly. The following technologies were used:

- **React.js:**

  React is a popular JavaScript library developed by Facebook for building user interfaces. It allows developers to create reusable UI components, improving code modularity and development efficiency. React's virtual DOM mechanism ensures fast updates and rendering, providing a seamless experience even for dynamic, data-driven applications like e-commerce platforms.

- **Redux or Context API:**

  For state management, Redux or alternatively the built-in Context API was utilized. In an e-commerce platform, where multiple components need access to shared data (such as the shopping cart, user authentication status, and product listings), state management becomes essential. Redux provides a centralized store and predictable state updates, whereas Context API offers a simpler solution for smaller applications.

- **React Router:**

  React Router enables client-side routing, allowing seamless navigation between different pages (e.g., product pages, shopping cart, checkout, admin dashboard) without reloading the page. This enhances performance and provides a smooth, single-page application (SPA) experience.

- **Axios:**
  Axios is a promise-based HTTP client used to communicate with the backend API. It simplifies sending asynchronous requests such as fetching products, submitting orders, user authentication, and payment processing, while offering built-in features like interceptors and error handling.

- **Styled-components or CSS Modules:**

  For styling the application, component-level styling approaches like styled-components or CSS Modules were employed. These techniques ensure that styles are scoped locally to components, preventing conflicts, and improving maintainability of the UI design.

Backend Technologies

The backend of the platform is responsible for handling business logic, data processing, API creation, and interaction with the database. The backend stack includes:

- **Node.js:**

  Node.js is a powerful JavaScript runtime environment that enables running JavaScript on the server side. Its non-blocking, event-driven architecture makes it highly suitable for building scalable and high-performance backend services.

- **Express.js:**

  Built on top of Node.js, Express.js is a lightweight, flexible framework for building web applications and APIs. It simplifies server creation, routing, middleware management, and RESTful API development, significantly speeding up backend development.

- **Mongoose:**

  Mongoose is an Object Data Modeling (ODM) library designed to work in an asynchronous environment. It provides a schema-based solution to model the application data, including validations, queries, and middleware functions. Mongoose ensures structured interaction with MongoDB and reduces the chances of database anomalies.

- **jsonwebtoken (JWT):**

  JWT is used for implementing secure authentication mechanisms. Upon successful login, a JWT token is generated and sent to the client, where it can be used for authenticating subsequent API requests. This ensures stateless authentication and enhances security by avoiding the storage of session information on the server.

- **bcrypt:**

  bcrypt is a password-hashing library used to securely hash and compare user passwords. Passwords are never stored in plain text in the database, significantly reducing the risks associated with data breaches.

- **nodemailer:**

  For functionalities such as password recovery, nodemailer is used to send automated emails to users. It integrates easily with various email services like Gmail or SMTP servers and helps automate communication flows.

- **Razorpay Node.js SDK:**

  The Razorpay Node.js SDK is used to integrate the Razorpay payment gateway into the platform. It handles payment creation, transaction verification, and callbacks, ensuring a secure and seamless payment process for users purchasing stationary products.

Database Technologies

- **MongoDB:**

  MongoDB is a document-oriented NoSQL database that stores data in flexible, JSON-like documents. It is highly scalable, supports powerful querying capabilities, and suits the dynamic schema requirements of e-commerce platforms. MongoDB collections were used to manage users, products, categories, orders, and payment details, ensuring efficient data retrieval and storage.

- **MongoDB Atlas:**

  For cloud hosting the database, MongoDB Atlas was considered. It offers automated scaling, backup, and security features, allowing easy database management and high availability without the overhead of manual server maintenance.

Deployment Technologies

To ensure the platform is accessible to users globally with minimal latency and downtime, reliable hosting services were selected for both frontend and backend deployment:

- **Vercel or Netlify (Frontend Hosting):**

  Vercel and Netlify are popular cloud platforms designed for hosting front-end applications built with frameworks like React. They offer continuous deployment pipelines integrated with GitHub repositories, automatic SSL certification, CDN support for faster load times, and custom domain configurations. These platforms are ideal for quickly deploying React-based web apps with zero server management overhead.
- **Heroku or AWS (Backend Hosting and API Services):**
  - **Heroku:** A platform-as-a-service (PaaS) that allows developers to deploy, manage, and scale applications without worrying about infrastructure. It is particularly friendly for Node.js applications and supports easy database integration with add-ons like MongoDB Atlas.
  - **AWS (Amazon Web Services):** For more extensive scalability and control, AWS services like EC2 (Elastic Compute Cloud) for hosting the Node.js server and S3 for storing static assets were considered. AWS provides enterprise-grade performance, security, and flexibility, making it ideal for production-ready deployments.
  -

- **Environment Variables and Security:**

  Sensitive information like API keys, database URIs, and secret tokens are managed securely using environment variables. Deployment platforms provide built-in mechanisms to inject these variables during runtime, ensuring security best practices.

4. Key Features and Functionality

The stationary e-commerce platform has been designed with a strong emphasis on user experience, security, and operational efficiency. The application provides distinct functionalities for regular users (customers) and administrators (management staff), ensuring smooth operation on both the frontend and backend. Below is an in-depth breakdown of the key features incorporated into the platform:

4.1 User Features

The platform offers a comprehensive range of features for users, ensuring a seamless, secure, and intuitive shopping experience.

User Authentication

- **Registration:**

  New users can register on the platform by providing necessary details such as their username, email address, and password. Validation checks are applied to ensure data correctness and security (e.g., password strength requirements and email format validation).

- **Login:**

  Registered users can securely log into their accounts using their email and password. Upon successful authentication, a token (JWT) is generated, allowing for session management without maintaining sensitive information on the server side.

- **Forgot Password:**

  In case users forget their passwords, the platform offers a password recovery system. This involves sending a password reset link to the registered email address via nodemailer, allowing users to securely create a new password.

Product Browsing and Filtering

- **Product Display:**

  The homepage and product listing pages showcase all available stationary products. Each product displays key details, including name, category, price, description, and thumbnail images, offering users a quick overview.

- **Filtering by Category:**

  Users can refine their product search based on predefined categories such as pens, notebooks, art supplies, office supplies, and more. This categorization helps in improving navigation and enhances the overall shopping experience.

- **Filtering by Price Range:**

  A price range filter allows users to view products within their desired budget. This dynamic filtering updates the displayed product list in real time based on the user's selection.

Product Details

- **Individual Product Pages:**

  Each product has a dedicated page presenting comprehensive details such as a detailed description, multiple product images, price, category tags, and availability status. In future iterations, user-generated reviews and ratings can be integrated to further inform purchasing decisions.

Shopping Cart

- **Adding Items:**

  Users can add multiple products to their shopping cart. The cart updates dynamically, showing the added items without needing a page reload.

- **Managing Items:**

  Users can view all selected items in the cart, adjust product quantities, or remove products entirely.

- **Cart Overview:**

  A running total is automatically calculated based on the contents of the cart, giving users a clear view of their expected expenditure.

Checkout Process

- **Shipping Information:**

  During the checkout process, users provide necessary shipping details, including name, address, contact number, and preferred delivery instructions.

- **Order Review:**

  Before payment, users have the opportunity to review their order, ensuring accuracy in item selection and shipping information.

Payment Integration (Razorpay)

- **Secure Transactions:**

  Payments are securely processed using Razorpay's payment gateway. Users can choose from various payment methods, including credit cards, debit cards, UPI, and net banking.

- **Transaction Handling:**

  After payment, the system verifies transaction success and updates the order status accordingly. Both the user and the admin can track the payment status.

- **Order Confirmation:**

  Upon successful payment, users receive immediate confirmation on the website, along with a detailed summary of their purchase. Optionally, a confirmation email can also be sent.

Order History

- **Past Orders:**

  Users can view a complete history of their previous orders. Each order listing includes details such as order date, product names, quantity, total cost, and shipping status.

- **Order Details:**

  Clicking on a specific order provides an expanded view showing the items purchased, their quantities, prices, shipping address, and payment confirmation.

4.2 Admin Features (Accessible via Secure Login)

A crucial aspect of the platform is the administrative panel, designed to provide a comprehensive suite of management tools. Access to this panel is restricted to authorized personnel only, ensuring the security of sensitive operations.

Category Management

- **Create Category:**

  Administrators can add new product categories to organize the product catalog better. A simple form allows entry of the category name and optional descriptions.

- **Update Category:**

  Admins can modify the details of existing categories, such as renaming or updating descriptions to reflect new business directions or product expansions.

- **Delete Category:**

  Categories can be removed if no longer needed. Before deletion, checks are performed to manage associated products, ensuring they are either reassigned to a different category or appropriately handled to prevent data inconsistencies.

Product Management

- **Create Product:**

  Admins have access to a product creation form where they can add new listings. Inputs include:

  - Product Name
  - Description
  - Price
  - Assigned Category
  - Images (uploaded or linked)
  - Available Stock Quantity
  - SKU (optional, for inventory tracking)

- **Update Product:**

  Existing products can be updated to reflect changes in pricing, description, stock levels, or category. This ensures that the product catalog remains accurate and up-to-date.

- **Delete Product:**

  Admins can permanently remove products from the platform. Confirmation prompts and verification checks are implemented to avoid accidental deletions.

Order Management

- **View All Orders:**

  A centralized dashboard lists all customer orders. Details include:

  - Order ID
  - Customer Name and Contact Information
  - List of Purchased Items
  - Total Amount Paid
  - Payment Status (Pending, Paid, Failed)
  - Shipping Status (Pending, Dispatched, Delivered)

- **Update Shipping Status:**

  Admins can manually update the shipping status of each order as it progresses through fulfillment stages. This helps maintain transparency for customers tracking their purchases.

- **Order Filtering and Search:**

  The admin dashboard allows searching and filtering orders based on criteria such as payment status, date range, or customer name, facilitating faster order management.

5. Challenges and Solutions

During the development of the stationary e-commerce platform, several potential challenges were identified, each requiring thoughtful solutions to ensure the system’s reliability, security, and scalability.

Security

Security was a top priority, given the platform's requirement to handle sensitive user data and financial transactions. Protecting user credentials, preventing unauthorized access, and safeguarding against common web vulnerabilities such as Cross-Site Scripting (XSS) and Injection attacks were critical.

- **Solutions:**
    - **bcrypt** was used to hash user passwords securely before storage in the database, ensuring that even in the event of a data breach, plaintext passwords would not be exposed.
    - All form inputs and API endpoints incorporated strong validation to prevent malicious data entry.
    - JWT tokens were implemented for secure and stateless user authentication.
    - HTTPS protocol was enforced in production environments to encrypt data transmissions between client and server.

Scalability

As user numbers and product listings grow, the application must maintain high performance without compromising the user experience.

- **Solutions:**
    - **Database indexing** on frequently queried fields (such as product categories and user IDs) was employed to speed up query execution times.
    - API endpoints were optimized for minimal response times, using pagination techniques where large datasets are involved.
    - The application architecture is designed to support scaling horizontally, with potential future use of load balancers and distributed server environments.

State Management (Frontend)

Managing complex and interconnected application states across multiple components posed a significant challenge, especially for features like the shopping cart and user authentication.

- **Solutions:**
    - **Redux** or the **Context API** was used to centralize state management, ensuring consistent data flow throughout the application.

    - Middleware solutions like Redux Thunk were considered to handle asynchronous actions, such as API calls for fetching products or submitting orders.

Payment Integration

Integrating a payment gateway brings the challenge of handling a variety of payment states (success, failure, pending) and ensuring the entire process remains secure and user-friendly.

- **Solutions:**
    - **Razorpay's Node.js SDK** was utilized for secure, PCI-compliant payment processing.
    - Robust error handling was implemented to manage transaction failures gracefully, ensuring users receive clear feedback.
    - Server-side verification of payments added an additional layer of security to confirm the authenticity of transactions.

Real-time Updates (Optional Enhancement)

Although not currently implemented, the possibility of adding features like real-time order tracking or admin notifications would require handling real-time communication between client and server.

- **Potential Solutions:**
    - **WebSocket** technologies, or libraries such as **Socket.IO**, could be used to enable bi-directional real-time communication.
    - This would allow dynamic updates without the need for manual page refreshes, significantly enhancing the user experience.

---

6. Conclusion

The development of the stationary e-commerce platform utilizing the MERN stack has successfully resulted in a modern, efficient, and scalable solution tailored for online stationary retail. By combining a dynamic React frontend, a robust Node.js and Express backend, and a flexible MongoDB database, the platform ensures a smooth and secure shopping experience for users.

Key features such as user authentication, product browsing and filtering, shopping cart management, seamless payment integration via Razorpay, and a full-featured admin panel for category, product, and order management, collectively contribute to a highly functional marketplace.

The challenges encountered during development, particularly regarding security, scalability, and state management, were met with robust, best-practice-driven solutions. Security measures such as password hashing, input validation, and HTTPS communication safeguard both users and business operations. The architectural choices, including optimized database indexing and scalable server design, ensure that the platform can grow with increased traffic and data loads.

Looking ahead, several enhancements can be incorporated to further enrich the platform. Adding features such as user reviews, wishlists, advanced product search capabilities, and detailed real-time order tracking would provide greater engagement and convenience to users. Additionally, implementing marketing features like discounts, loyalty programs, and email notifications could enhance customer retention and increase sales.

Overall, the project stands as a strong foundation for a scalable, feature-rich e-commerce platform capable of adapting to evolving business needs and customer expectations in the competitive digital marketplace.