MCA Semester – IV

Project – Final Report

Name	**Vatsal Parmar**
USN	**221VMTR02537**
Elective	**Full Stack Development**
Date of Submission	04-06-2024

A study on “E-commerce Web Application”

A Project submitted to Jain Online (Deemed-to-be University)

In partial fulfillment of the requirements for the award of:

Master of Computer Application

Submitted by:

Vatsal Parmar

Under the guidance of:

Mention your Guide’s Name

Mr. Chandra Bhanu

Jain Online (Deemed-to-be University)

Bangalore

DECLARATION

I, *Vatsal Parmar*, hereby declare that the Project Report titled *"E-commerce Web Application"* has been prepared by me under the guidance of the *Chandra Bhanu Sir*. I declare that this Project work is towards the partial fulfilment of the University Regulations for the award of the degree of Master of Computer Application by Jain University, Bengaluru. I have undertaken a project for a period of one semester. I further declare that this Project is based on the original study undertaken by me and has not been submitted for the award of any degree/diploma from any other University / Institution.

Place: Hyderabad

Date: 03-06-2024

_____

Vatsal Parmar
USN: 221VMTR02537

ACKNOWLEDGEMENT

The development of ModernMart, the e-commerce platform, has been a collaborative effort involving the dedication, expertise, and hard work of many individuals and teams. I extend my heartfelt gratitude to everyone who contributed to this project, from the initial planning stages to its successful implementation and launch.

First and foremost, I would like to thank my mentor Mr Chandra Bhanu his tireless efforts in cascading right knowledge for designing, coding, and testing the various components of ModernMart. The technical skills, creativity, and attention to detail have been instrumental in bringing our vision to life.

Furthermore, I would like to express my appreciation to friends for their valuable feedback, support, and collaboration. Their input and insights have been invaluable in shaping the direction and features of ModernMart, making it a truly customer-centric and market-driven platform.

Last but not least, I extend my thanks to my family and loved ones for their patience, understanding, and encouragement during the demanding phases of this project. Their unwavering support has been a source of strength and motivation throughout the journey.

In conclusion, I am immensely proud of the achievements of the ModernMart project and grateful for the opportunity to work on such an exciting and impactful initiative. I look forward to continued success and innovation as I strive to enhance and evolve the ModernMart platform in the ever-changing landscape of e-commerce.

EXECUTIVE SUMMARY

ModernMart set out to revolutionize online shopping by combining state-of-the-art technology with user-friendly design. The main goal was to make shopping enjoyable for customers while helping businesses sell their products more effectively.

I started by studying market trends, how people shop online, and what other companies are doing. This helped me create a website with lots of features that meet the needs of today's shoppers and businesses.

I worked closely using agile methods, which meant I could quickly test ideas and make improvements. This ensured that the website not only looked great but also worked well for everyone who used it.

The heart of ModernMart is its easy-to-use website. We made sure it's simple to find what you're looking for and to buy it without any hassle. Our advanced search and filter tools help you discover products easily, and our secure payment system makes buying safe and easy.

Behind the scenes, ModernMart has a strong system in place to keep everything running smoothly. The presence of strict security measures to protect your information and make sure everything works seamlessly. Plus, I provide businesses with tools to understand how their products are selling and how they can improve.

Keeping your data safe was a top priority for me. I followed industry standards and best practices to make sure your information is always secure. This builds trust and confidence among our users and businesses.

In short, ModernMart brings together the latest technology, smart design, and great functionality to make online shopping better for everyone. I’m dedicated to providing the best experience possible and am always looking for ways to improve and innovate.

TABLE OF CONTENTS

1. Introduction

The capstone project represents the culmination of our journey in full-stack development, serving as a comprehensive showcase of the skills and knowledge acquired throughout our academic endeavours. It stands as a testament to our ability to conceive, design, and implement a fully functional web application, encompassing both front-end and back-end components.

At its core, the project is driven by a set of overarching goals and objectives aimed at demonstrating proficiency in various aspects of web development. These objectives may include building a responsive and visually appealing user interface, implementing robust authentication and authorization mechanisms, designing efficient database schemas, and ensuring seamless integration between different layers of the application.

The problem statement or purpose of the project encapsulates the specific challenge or need that the application aims to address within its chosen domain. This could range from simplifying a complex business process to enhancing user experiences in a particular industry vertical. By clearly defining the problem statement, we can align our development efforts towards creating a solution that delivers tangible value to its intended users or stakeholders.

In the realm of full-stack development, the capstone project holds immense significance as it provides a platform for us to apply theoretical concepts to real-world scenarios and showcase our practical skills to potential employers or collaborators. It serves as a bridge between academic learning and professional practice, allowing us to demonstrate our ability to navigate the complexities of modern web development and deliver high-quality solutions that meet the needs of today's digital landscape.

Moreover, the project serves as a testament to our adaptability and versatility as developers, as we navigate through the myriad challenges and constraints inherent in the development process. From managing project timelines and resources to troubleshooting technical issues and incorporating user feedback, the capstone project offers a holistic learning experience that prepares us for the rigors of professional software development.

In summary, the capstone project represents a culmination of our academic journey, encapsulating our growth and development as full-stack developers. It embodies our passion for innovation, our commitment to excellence, and our readiness to contribute meaningfully to the ever-evolving field of web development. Through diligent planning, meticulous execution, and unwavering dedication, we strive to create a project that not only meets academic requirements but also leaves a lasting impact on the broader community of developers and users alike.

2. System Architecture

- Client

The end-user accesses the ModernMart e-commerce website using a web browser.

- Next.js Frontend

The frontend of ModernMart is built using Next.js, a React framework. It handles the user interface and client-side logic. Requests from the client's browser are sent to the Next.js frontend.

- Node.js Backend

The backend is developed using Node.js and deployed on a VPS (Virtual Private Server) or other cloud hosting service that does not involve AWS.
This backend handles the business logic, processes requests from the frontend, interacts with the database, and manages file storage locally or via another storage service.

- MongoDB

The MongoDB database stores all the application's data, including user information, product details, orders, etc.
The Node.js backend queries the MongoDB database to fetch or store data as needed.

- Local Storage or Alternative Storage Service

Instead of AWS S3, the application uses local storage on the VPS or another cloud storage service to store images and other static assets.
The Node.js backend interacts with this storage solution to upload and retrieve images and other static files.

Interaction Flow

1. Client Interaction
The client accesses the website, and their requests are sent to the Next.js frontend.

2. Frontend Processing
The Next.js frontend processes the client's requests, generating the necessary pages or components.
For data that requires backend interaction (e.g., fetching product details), the frontend sends requests directly to the Node.js backend.
3. Backend Logic

The Node.js backend processes the requests. If data is needed, it queries the MongoDB database.
For any static content or images, the backend interacts with the local storage on the VPS or another storage service.

4. Database Interaction
MongoDB handles queries from the Node.js backend, returning the necessary data for processing.

5. Storage Interaction

Local storage on the VPS or another storage service manages static assets, ensuring efficient retrieval and storage.
This architecture ensures that the application is independent of AWS and PostgreSQL while leveraging MongoDB for database management and a flexible storage solution for static assets.

data flow diagram for our capstone project:

User Interface (Front-end):

User interacts with the web application through the user interface.
Sends requests to the back-end server for various actions such as viewing products, adding items to the cart, or placing orders.
Receives responses from the server and displays them to the user.
Web Server (Back-end):

Receives HTTP requests from the front-end user interface.
Processes incoming requests by routing them to the appropriate handlers or controllers.
Validates user input, performs necessary operations (such as querying the database or invoking external services), and generates responses.
Sends HTTP responses back to the front-end user interface.
Database:

Stores persistent data related to users, products, orders, and other entities.
Receives queries from the back-end server to retrieve, insert, update, or delete data.
Processes database queries and returns results to the back-end server.
External Services (Optional):

Interfaces with third-party services such as payment gateways, shipping providers, or external APIs.
Sends requests to external services for tasks like processing payments or retrieving additional product information.
Receives responses from external services and forwards them to the back-end server for further processing or displaying to the user.

Authentication and Authorization:

Manages user authentication and authorization processes.
Verifies user credentials and issues access tokens or session cookies upon successful authentication.
Enforces access control policies to restrict user access to certain features or resources based on their roles and permissions.

Logging and Monitoring:

Captures log entries for various system events, errors, and user activities.
Monitors system performance metrics such as response times, error rates, and resource utilization.
Provides administrators with insights into system health and usage patterns for troubleshooting and optimization purposes.

This data flow diagram illustrates the flow of information between different components of the application, from user interactions at the front-end interface to data processing in the back-end server and storage in the database. It highlights the key interactions and dependencies within the system, helping to understand how data flows through the application architecture.

3. Technologies Used

Technologies Used

Frontend:

Next.js: A React framework used for building the frontend of the application. It supports server-side rendering and static site generation, providing a fast and SEO-friendly user experience.
React: A JavaScript library for building user interfaces, particularly useful for creating reusable UI components.

Backend:

Node.js: A JavaScript runtime built on Chrome's V8 engine, used for building the backend server that handles business logic, processes requests, and interacts with the database.
Express.js: A minimal and flexible Node.js web application framework that provides a robust set of features for building web and mobile applications.
Database:

MongoDB: A NoSQL database used for storing application data such as user information, product details, and orders. It offers flexibility and scalability, making it suitable for handling diverse data structures.

Storage:

Local Storage on VPS or Alternative Storage Service: Used for storing images and other static assets. This setup allows the application to manage static content efficiently without relying on AWS S3.

Additional Tools and Libraries:

Axios: A promise-based HTTP client for making requests from the frontend to the backend.
Mongoose: An ODM (Object Data Modeling) library for MongoDB and Node.js, providing a straightforward, schema-based solution to model application data.

Webpack: A module bundler used to compile JavaScript modules, ensuring the frontend assets are optimized and loaded efficiently.

Babel: A JavaScript compiler used to convert ECMAScript 2015+ code into a backwards-compatible version of JavaScript, ensuring compatibility across different browsers.
This technology stack provides a robust and scalable solution for building a modern e-commerce application, leveraging Next.js for the frontend, Node.js for the backend, and MongoDB for the database, all hosted on reliable and efficient platforms.

4. Front-end Development

Details About the Front-End Implementation

- Next.js: Next.js was chosen as the core framework for the front end due to its robust support for server-side rendering, static site generation, and dynamic routing capabilities. This framework enables the creation of fast, scalable, and SEO-friendly web applications.

- Next-Auth: For handling user authentication, Next-Auth was implemented to manage user login, registration, and role-based access control. This ensures a secure and efficient authentication process.

- Next.js Middleware: Role-based routing was managed using Next.js middleware, ensuring that users only have access to pages appropriate to their roles. This middleware enforces security and proper user navigation throughout the application.

The codebase adhered to best practices from the Next.js app router documentation, ensuring the application is modular, maintainable, and scalable.

User Interface Design and User Experience Considerations

- Design Principles: The user interface was designed with a focus on simplicity, usability, and user-centricity. Consistency and clarity were key principles, ensuring that users have a smooth and intuitive experience.

- Tailwind CSS: Tailwind CSS was utilized as the primary styling framework. This utility-first approach to CSS allowed for rapid development and easy maintenance of responsive designs.

- Responsiveness: Tailwind's responsive design classes (e.g., lg, md, sm) were used to ensure that the website performs well across various devices and screen sizes. This responsiveness guarantees that users have a seamless experience whether they are on a mobile device, tablet, or desktop computer.

Features and Functionalities of the Front-End

- User Authentication and Role Management: Users can browse the shop and search for products without logging in. However, to add items to the cart, users must log in. Role-based access control ensures that users with ADMIN roles are redirected to the appropriate dashboard if they attempt to access customer-only pages.

- Product Browsing and Search: The front end allows users to easily browse and search for products. This feature is designed to be fast and responsive, providing users with quick access to the products they are interested in.

- Cart Management: Logged-in users can add products to their cart, manage the items,

and proceed to checkout. The cart management system is intuitive and easy to use, ensuring a smooth shopping experience.

- By leveraging these technologies and adhering to best practices, the front-end development of ModernMart provides a robust, user-friendly, and secure e-commerce platform.

5. Back-end Development

Details About the Back-End Implementation

- Node.js and Express: The back end of ModernMart is powered by Node.js and Express. This combination was chosen for its lightweight and efficient performance, as well as its extensive ecosystem of libraries and middleware.

- MongoDB: MongoDB is used as the database, offering a flexible, scalable, and high-performance solution for storing and retrieving data. Its document-oriented model suits the dynamic data structures of an e-commerce platform.

- Authentication and Authorization: JSON Web Tokens (JWT) are employed for user authentication and authorization. This ensures secure communication between the client and server, allowing users to securely log in and access role-based features.

- Server-Side Logic and Database Design

- Database Schema: The database is designed to handle various aspects of the e-commerce platform, including users, products, orders, and roles. Collections in MongoDB are structured to efficiently store and retrieve data, with appropriate indexing to optimize query performance.

- Users Collection: Stores user information, including encrypted passwords, roles, and personal details.

- Products Collection: Contains product details such as name, description, price, and inventory status.
- Orders Collection: Manages order details, including user information, products ordered, status, and timestamps.
- Roles Collection: Defines different user roles and their permissions.
- Server-Side Logic: The server-side logic is implemented to handle business operations such as user registration, product management, order processing, and role-based access control. Middleware functions are used to authenticate users and authorize access to specific routes based on their roles.

APIs and Endpoints Used for Communication with the Front-End:

- RESTful APIs: The back end provides a set of RESTful APIs to facilitate communication with the front end. These APIs are designed to be stateless, allowing for easy integration and interaction between the client and server.

User Authentication APIs:

- POST /api/auth/register: Registers a new user.
- POST /api/auth/login: Authenticates a user and returns a JWT.
- GET /api/auth/profile: Retrieves the authenticated user's profile.
- Product Management APIs:

- GET /api/products: Fetches a list of products.
- GET /api/products/:id: Retrieves details of a specific product.
- POST /api/products: Adds a new product (admin-only).
- PUT /api/products/:id: Updates an existing product (admin-only).
- DELETE /api/products/:id: Deletes a product (admin-only).
- Order Management APIs:

- POST /api/orders: Creates a new order.
- GET /api/orders: Retrieves orders for the authenticated user.
- GET /api/orders/:id: Retrieves details of a specific order.
- PUT /api/orders/:id/status: Updates the status of an order (admin-only).

Middleware for Role-Based Access Control: Middleware functions are implemented to check user roles before allowing access to certain endpoints. This ensures that only authorized users can perform specific actions, such as adding or removing products, or managing orders.

By using these technologies and designing efficient server-side logic and APIs, the back-end implementation of ModernMart ensures a secure, scalable, and high-performance platform for managing e-commerce operations.

6. Database Design

Database Design

- Database Schema and Data Model
- The database schema of ModernMart is designed to handle various entities required for a comprehensive e-commerce platform. Using MongoDB, the schema is structured into collections, each tailored for specific data storage needs.

Users Collection:

The Users Collection stores information about registered users of the ModernMart platform. It includes details such as usernames, emails, encrypted passwords, user roles, and timestamps indicating account creation and updates.

Products Collection:

In the Products Collection, data related to the various products available on ModernMart are stored. This includes details like product names, descriptions, prices, categories, available stock, and timestamps indicating when the product information was created or last updated.

Orders Collection:

The Orders Collection manages information about orders placed by users on ModernMart. It includes data such as the user ID associated with the order, the products ordered along with their quantities, the total price of the order, the order status (e.g., pending, shipped, delivered), and timestamps indicating when the order was created or last updated.

Roles Collection:

Within the Roles Collection, definitions of different user roles and their associated permissions are stored. Each role entry includes a role name and an array of permissions assigned to that role, enabling fine-grained access control throughout the ModernMart platform.

Choice of Database Management System and Justification

- MongoDB:

- Document-Oriented Model: MongoDB's document-oriented model aligns well with the dynamic nature of e-commerce data. Collections can store various fields for each document, which allows for flexibility in data structure.

- Scalability: MongoDB offers horizontal scalability, which is crucial for handling the large volumes of data and high traffic associated with e-commerce platforms. Sharding enables distributed data storage across multiple servers, ensuring performance remains high as the user base grows.

- Performance: MongoDB provides high performance for read and write operations. Indexing capabilities, such as compound and text indexes, optimize query performance, making data retrieval efficient.

- Schema Flexibility: The ability to store nested data structures in BSON format allows MongoDB to represent complex data relationships without the need for JOIN operations, which can be costly in terms of performance in relational databases.

- Ease of Use: The JSON-like structure of MongoDB documents makes it intuitive for developers to use, particularly those familiar with JavaScript and JSON. This reduces the learning curve and speeds up development time.

- Community and Ecosystem: MongoDB has a large and active community, along with comprehensive documentation and a rich ecosystem of tools and integrations. This support network is valuable for troubleshooting, optimizing performance, and finding best practices.

- By leveraging MongoDB, ModernMart benefits from a database system that is both flexible and robust, capable of scaling with the platform's growth while maintaining high performance and ease of use.

7. Authentication and Security

- Authentication and Authorization Mechanisms:

ModernMart employs Next-Auth for authentication, handling user login, registration, and role-based access control efficiently. Users are authenticated using secure sessions and cookies, ensuring that only authorized users can access protected resources. Role-based routing is implemented using Next.js middleware, ensuring that users only access pages appropriate to their roles. Additionally, role-based access control is enforced at the server-side, ensuring that users with ADMIN roles are redirected to their respective dashboards if they attempt to access customer-only pages.

Security Measures:

To protect user data, ModernMart implements several security measures. These include:

- Encryption: Sensitive user data, such as passwords, are encrypted using industry-standard encryption algorithms before being stored in the database. This ensures that even if the database is compromised, the data remains secure.

- Input Validation: All user input is carefully validated and sanitized on both the client and server sides to prevent common security vulnerabilities such as SQL injection and cross-site scripting (XSS) attacks.

- HTTPS: ModernMart employs HTTPS to encrypt data transmitted between the client and server, protecting it from interception or tampering by malicious actors.

- Access Control: Role-based access control mechanisms are implemented to ensure that users can only access resources and perform actions that they are authorized to. This prevents unauthorized access to sensitive data or functionalities.

- Regular Security Audits: Regular security audits and code reviews are conducted to identify and address any potential security vulnerabilities in the application codebase.

By implementing these security measures, ModernMart ensures that user data remains protected and the application is resilient to security threats.

8. Project Management

- Project Timeline, Milestones, and Sprints:

ModernMart followed a structured project management approach, dividing the development process into multiple phases, each with its own set of milestones and sprints.

- Planning Phase: This phase involved defining project requirements, establishing goals, and creating a detailed project plan. Key milestones included finalizing the project scope, defining user stories, and outlining the project timeline.

- Design Phase: During this phase, the focus was on UI/UX design, database schema design, and architectural planning. Milestones included creating wireframes, designing the database schema, and finalizing the technology stack.

- Development Phase: This phase encompassed the actual implementation of the application, including front-end and back-end development, database integration, and testing. Development was divided into multiple sprints, each typically lasting two to four weeks. Milestones included the completion of key features and functionalities, such as user authentication, product browsing, and order management.

- Testing Phase: Once development was complete, the application underwent rigorous testing to identify and address any bugs or issues. Milestones included the completion of unit tests, integration tests, and user acceptance testing (UAT).

- Deployment Phase: In this final phase, the application was deployed to production servers and made available to end users. Milestones included the successful deployment of the application, monitoring for any performance or stability issues, and ensuring a smooth transition to production.

- Project Management Approach:

ModernMart adopted an Agile project management approach, specifically Scrum, to ensure flexibility and adaptability throughout the development process.

- Scrum Framework: The project was divided into a series of sprints, typically lasting two to four weeks, during which specific features and functionalities were developed, tested, and delivered.

- Daily Stand-ups: Daily stand-up meetings were held to keep team members updated on progress, discuss any obstacles or challenges, and plan the day's tasks.

- Sprint Planning: At the beginning of each sprint, a sprint planning meeting was conducted to prioritize tasks, assign responsibilities, and set sprint goals.

- Sprint Review and Retrospective: At the end of each sprint, a sprint review meeting was held to demonstrate completed work to stakeholders and gather feedback. This was followed by a sprint retrospective to reflect on the sprint process, identify areas for improvement, and make adjustments for future sprints.

By following this project management approach, ModernMart was able to efficiently manage the development process, respond to changing requirements, and deliver a high-quality e-commerce platform on schedule.

9. Results and Evaluation

The development of the ModernMart e-commerce platform has yielded substantial results, aligning closely with the project's initial goals. Here's a discussion on the achieved results and an evaluation of the application's performance and efficiency:

Achieved Results:
The ModernMart e-commerce platform successfully met its primary objectives, offering a robust and user-friendly solution for online shopping. Key achievements include:

- Comprehensive Functionality: The platform incorporates essential features such as user authentication, product management, cart functionality, order processing, and role-based access control. This ensures a seamless and secure shopping experience for both customers and administrators.

- Responsive User Interface: The user interface design prioritized simplicity, usability, and responsiveness. Leveraging modern design principles and frameworks like Tailwind CSS, the platform delivers an intuitive browsing experience across various devices and screen sizes.

- Secure Authentication and Authorization: Advanced authentication mechanisms, including token-based authentication and role-based access control, were implemented to safeguard user data and protect sensitive operations. This ensures that only authorized users can access specific functionalities based on their roles.

- Efficient Database Management: The database schema and data model were carefully designed to efficiently manage user information, product details, orders, and roles. The choice of MongoDB as the database management system offers scalability, flexibility, and performance benefits, contributing to the application's overall efficiency.

- Evaluation:

While the project achieved its primary goals, there are areas for further evaluation and improvement:

- Performance Optimization: While the application performs well under normal load conditions, continuous optimization is necessary to enhance performance, especially during peak traffic periods. Strategies such as server-side caching, query optimization, and CDN integration can improve response times and overall user experience.

- Scalability and Robustness: As the user base grows, ensuring the scalability and robustness of the application becomes crucial. Regular stress testing and performance monitoring can identify potential bottlenecks and scalability issues, allowing for proactive adjustments to maintain optimal performance.

- Security Enhancements: While robust security measures were implemented, ongoing vigilance is required to mitigate emerging threats and vulnerabilities. Regular security audits, vulnerability assessments, and compliance checks can identify and address security gaps, safeguarding user data and maintaining trust.

- User Experience Refinement: Continuous refinement of the user experience based on user feedback and usability testing can further enhance the platform's appeal and usability. Iterative improvements to navigation, search functionality, and checkout processes can streamline the user journey and increase customer satisfaction.

- Overall, the ModernMart e-commerce platform has laid a strong foundation for providing a secure, efficient, and user-centric online shopping experience. By prioritizing performance optimization, scalability, security, and user experience refinement, the platform can continue to evolve and meet the dynamic needs of its users.

10. Conclusion

In conclusion, the ModernMart e-commerce platform represents a significant achievement in delivering a robust and user-friendly online shopping experience. Throughout the project, key objectives were met, resulting in a comprehensive solution that meets the needs of both customers and administrators.

Key points of the project and its outcomes include:

Comprehensive Functionality: The platform incorporates essential features such as user authentication, product management, cart functionality, order processing, and role-based access control, ensuring a seamless and secure shopping experience.

Responsive User Interface: Leveraging modern design principles and frameworks like Tailwind CSS, the platform offers a visually appealing and intuitive browsing experience across various devices and screen sizes.

Secure Authentication and Authorization: Advanced authentication mechanisms, including token-based authentication and role-based access control, safeguard user data and protect sensitive operations.

Efficient Database Management: The database schema and choice of MongoDB as the database management system offer scalability, flexibility, and performance benefits, contributing to the application's overall efficiency.

Moving forward, continuous optimization, scalability enhancements, security improvements, and user experience refinements will be essential to maintaining the platform's success and meeting the evolving needs of its users. By prioritizing these areas, ModernMart can continue to provide a secure, efficient, and user-centric online shopping experience for its customers.