COLLEGE CODE : 8203

COLLEGE NAME : A.V.C College Of Engineering

DEPARTMENT : B.Tech-Information Technology

STUDENT NM_ID : 6758371BD883BE19DBB421292A03628A

ROLL NO : 23IT79

DATE : 22-09-2025

Completed the project named as Phase III

technology project name : E-Commerce Cart System

SUBMITTED BY,

NAME : PRIYADHARSHAN A

MOBILE NO : 8220863295

Project Setup - E-Commerce Cart System

The project setup involves establishing the backend environment using

Node.js and Express. A

MongoDB database is configured to serve as the persistent data store for user information, product catalogs, and shopping carts.

To ensure maintainability and scalability, a clear folder structure is created:

- /models → Contains Mongoose database schemas for Users, Products, Carts, and Orders.
- /routes → Defines the API endpoints for authentication, cart management (CRUD), and the checkout process.
- /controllers → Holds the business logic for handling requests and formulating responses for each API route.
- /middleware → Includes the authentication middleware using **JWT** to protect sensitive routes like the cart and checkout pages.

The frontend can be built with a framework like

React to display product listings and manage the user's shopping cart dynamically.

REST APIs will connect the frontend and backend, enabling seamless data flow for adding items to the cart, updating quantities, and processing checkouts.

Environment variables (

.env) are used to securely store sensitive data, such as the MongoDB connection URI and the JWT secret key, keeping them separate from the main codebase. For version control and collaboration,

GitHub is utilized to manage the project's source code, track changes, and coordinate development efforts.

Core Features Implementation

The MVP for the E-Commerce Cart System is focused on delivering the essential features for secure user management, robust cart functionality, a simple checkout process, and persistent data storage.

- **User Authentication & Authorization (Backend)**
    - Users can register and log in securely. Passwords are encrypted using hashing algorithms, and

JWT tokens are issued to manage user sessions.

    - REST APIs ensure that only authenticated users can access and modify their own shopping carts, preventing unauthorized data access.

- **Cart Management (Backend - MongoDB + APIs)**
    - Shopping carts are stored in MongoDB, linked to a specific user ID. Each cart contains a list of products, including their quantity and price at the time of addition.
    - A full suite of REST APIs enables

Creating (adding items), **Reading** (viewing the cart), **Updating** (changing quantity), and **Deleting** (removing items) cart contents.

    - Backend logic ensures that users can only interact with their own carts, maintaining data integrity and privacy.

- **Checkout Process (Frontend + Backend)**
    - The frontend (e.g., React) dynamically renders the cart's contents by fetching data from the backend APIs.
    - Upon checkout, the cart's state is sent to a secure backend endpoint.
    - The backend processes the checkout by creating an order, storing a permanent record of the transaction in an orders collection, and subsequently clearing the user's cart.
- **Data Persistence & Order History**
    - MongoDB provides permanent storage for all user, product, cart, and order data, ensuring no data is lost between sessions.
    - Each order includes timestamps for creation, allowing for effective order tracking and sales analytics.

Data Storage (Local State / Database)

The system's storage is strategically divided between temporary frontend state management and persistent backend database storage to ensure a responsive user experience and data reliability.

1. **Local State (Frontend - React)**
    - **Temporary Storage:** React's local state (e.g., useState, useContext) is used to temporarily manage the cart's contents in the browser. This provides immediate UI feedback when a user adds an item, without waiting for the server response.
    - **UI Responsiveness:** This approach ensures smooth interactions. For example, updating an item's quantity instantly reflects on the screen while the update request is sent to the backend in the background.
2. **Database Storage (Backend - MongoDB)**

- **Persistent Storage:** MongoDB serves as the single source of truth, storing all critical data, including user accounts, product details, and shopping carts. This guarantees that a user's cart persists across different devices and sessions.
- **Schema Design:** Mongoose schemas define the structure for each data collection. The

Cart schema includes fields for the userId, an array of items (each with a productId and quantity), and timestamps.

- **Data Integrity:** The database maintains proper references between users and their carts, ensuring that data relationships are consistent and secure.

3. **Integration of Local State and Database**
   - Local state acts as a temporary cache for fast UI updates, while the database provides permanent storage. When a user adds an item to their cart, the frontend UI is updated instantly from local state, and an API call is made simultaneously to save the change in the MongoDB database.

Testing Core Features (Backend + Frontend)

Comprehensive testing is conducted to ensure all features of the MVP work correctly and securely.

1. **Backend Testing (APIs)**
   - **Validation:** APIs for user registration, login, and all cart CRUD operations (POST, GET, PUT, DELETE) are rigorously tested.
   - **Tools:** API testing tools like **Postman** or **Insomnia** are used to send requests to each endpoint to verify correct responses, status codes, and data handling.
   - **Security:** Tests are performed to ensure that protected routes return a 401 Unauthorized error if a valid JWT is not provided, and to confirm that a user cannot access or modify another user's cart.
2. **Frontend Testing (React UI)**
   - **Dynamic Rendering:** Checks are performed to ensure the cart correctly displays items, quantities, and totals fetched from the backend.
   - **Interaction Testing:** All user interactions, such as clicking "Add to Cart," updating quantities, removing items, and proceeding to checkout, are tested to confirm they function as expected and update the UI instantly.
3. **End-to-End Testing**

- **User Flow:** The entire user journey is tested: a user registers, logs in, adds multiple items to the cart, updates an item's quantity, removes another, proceeds to checkout, and completes the purchase.
- **Data Consistency:** This end-to-end flow confirms that actions taken on the frontend are accurately and consistently reflected in the backend database.

version Control (GitHub)

To maintain project integrity and facilitate collaboration, version control is managed using

GitHub.

- **Repository:** https://github.com/Priyadharshan79/NM-IBM-AVCCE-PRIYADHARSHAN.git
- **Commits:** Changes are committed regularly with clear, descriptive messages after implementing each feature or fixing a bug, creating a clean and understandable project history.