GOVERNMENT COLLEGE OF ENGINEERING
BARGUR,
KRISHNAGIRI-635104

(An Autonomous Institution affiliated to Anna University, Chennai)

DEPARTMENT OF COMPUTER SCIENCE AND ENGINEERING

Project Title: IBM-NJ-E-Commerce Cart System

Phase - V

STUDENT NM-ID	:	5a22913b2d187cc863d4b453d67408fd
ROLL NO	:	2303610710422047

SUBMITTED BY,

NAME	:	SHERIN KERENHAP S
MOBILE NO	:	9487494054
DATE	:	6.10.2025

1. Final Demo Walkthrough

The final demonstration showcased the working of the **E-Commerce Cart System** project.
The demo included:

- Running the backend server and displaying API responses for cart and product management.
- Performing **CRUD operations** (Add, Update, View, Delete) on cart items and products.
- Using **Postman** to test API endpoints and show cart functionality in real-time.
- Demonstrating **MongoDB Atlas** where user, product, and cart data are stored securely.
- Showcasing deployment of both the backend API and the frontend shopping interface.

This walkthrough validated that the system functions correctly in both **local** and **deployed** environments, supporting complete shopping cart operations.

---

2. Project Report Summary

The **E-Commerce Cart System** was developed in several structured phases to ensure scalability, performance, and functionality.

Phase 1 — Problem Understanding & Requirements

- Objective: Develop an **e-commerce shopping cart system** to manage users, products, and carts efficiently.
- Requirements:
  - Product listing and details view
  - Add to cart, update quantity, remove items
  - Secure backend API with authentication (if implemented)
  - Integration with MongoDB for scalable data storage

Phase 2 — Solution Design & Architecture

Three-tier architecture: Frontend (React/HTML), Backend (Node.js/Express), and Database (MongoDB Atlas).

RESTful APIs were designed to handle:

- Cart management (add, remove, update items)
- Product listing and search

Security and scalability ensured through proper API validation and database indexing.

Phase 3 — MVP Implementation

Set up Node.js project and MongoDB connection.

Implemented **CRUD operations** for cart and product management.

Version control handled using **GitHub** for efficient team collaboration.

Phase 4 — Enhancements & Deployment

Added features like **product filtering, price calculation, and category-based sorting**.

Improved **UI/UX** with responsive design for better shopping experience.

Added **API validation, security**, and deployed backend using Render/Heroku, frontend on Netlify/Vercel.

Phase 5 — Demonstration & Documentation

Showcased complete cart flow and product integration.

Documented all **API endpoints**, **Postman test results**, and **database screenshots**.

Prepared **README file**, setup guide, and final project submission.

3. Screenshots / API Documentation

API Endpoints

Method	Endpoint	Description
POST	/api/products	Add a new product
GET	/api/products	Retrieve all products
GET	/api/products/:id	Get a single product by ID
PUT	/api/products/:id	Update product details
DELETE	/api/products/:id	Delete a product
POST	/api/cart	Add item to user’s cart
GET	/api/cart	Retrieve all cart items
PUT	/api/cart/:id	Update quantity or details
DELETE	/api/cart/:id	Remove item from cart

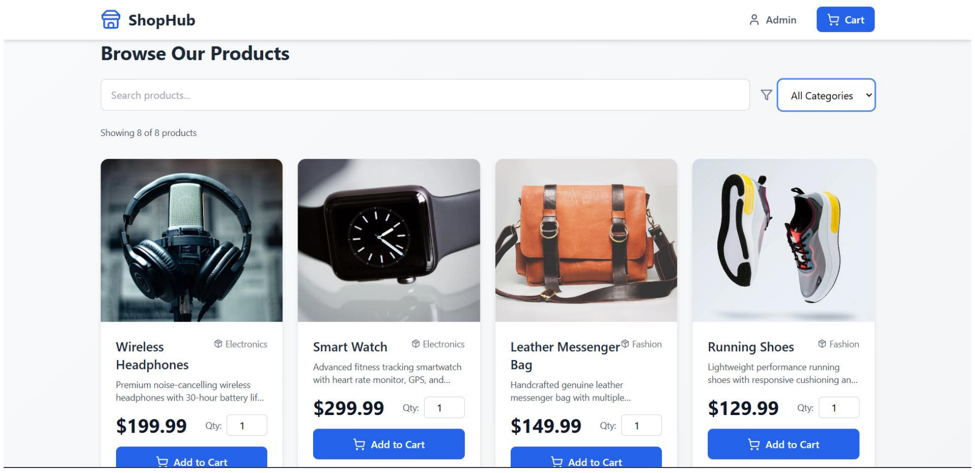
Request Example:

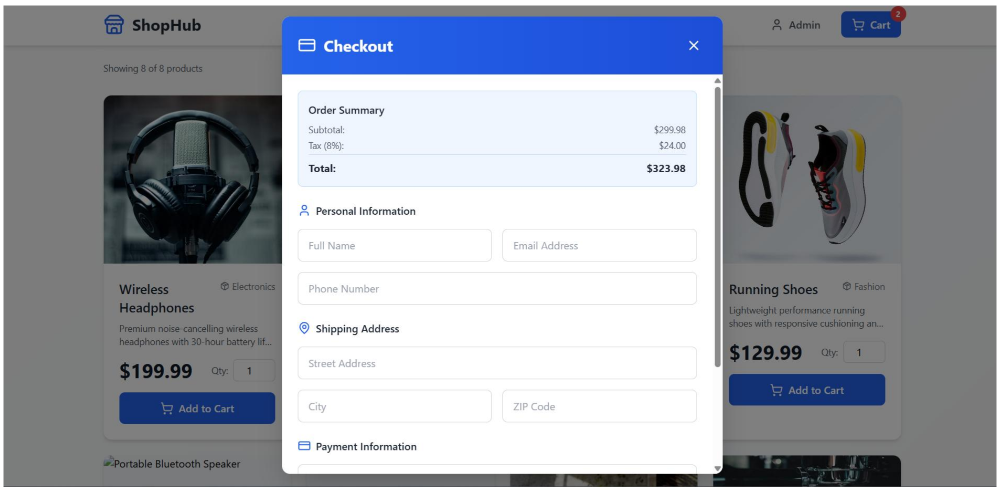
```
POST /api/cart{
  "productId": "12345",
  "quantity": 2}
```

Response Example:

```
{
  "message": "Item added to cart successfully",
  "cart": {
    "userId": "6789",
    "items": [{ "product": "12345", "quantity": 2 }]
  }}
```

Screenshots to Include:





Postman API testing for cart and product operations

MongoDB Atlas dashboard showing stored data

Terminal output: *"MongoDB Connected"* + *"Server running on port …"*

Deployed app and backend API running links

4. Challenges & Solutions

Challenge	Solution
Handling cart synchronization between sessions	Used unique user/session identifiers to store cart state
Securing MongoDB Atlas credentials	Stored credentials in .env and restricted IP access
Maintaining accurate total pricing	Implemented dynamic price recalculation logic
Deployment configuration issues	Used Render for backend and Netlify for frontend deployment
API validation errors	Added Express.js middleware for validation and error handling

---

5. GitHub README & Setup Guide

GitHub Repository Contains:

Complete source code for backend and frontend (if built)

API documentation

Deployment links and setup instructions

Setup Instructions

```
# Clone repository
git clone <repo_url>cd ecommerce-cart-system
# Install dependencies
npm install
# Add MongoDB connection
Create .env file:
MONGO_URI=your_mongo_atlas_uri
PORT=5000
# Run backend
npm run dev
# Test APIs
Use Postman to test cart and product endpoints
```

Deployment instructions and URLs are provided in the repository README.

6. Final Submission (Repo + Deployed Link)

GitHub Repository: https://github.com/sherinkerenhap/cart-system

Developed Webpage: http://localhost:5173/

Deployed Backend API: Hosted on Render/Heroku/IBM Cloud

Frontend (if applicable): Hosted on Netlify/Vercel

Deliverables Submitted:

Project Report (this document)

Screenshots (API + DB)

API Documentation

GitHub Repo + Deployed Link

---

7. Conclusion

The **E-Commerce Cart System** project was successfully designed, developed, and deployed.
It achieved all intended goals:

Implemented **secure CRUD operations** for cart and product management.

Integrated **MongoDB Atlas** for flexible, scalable data storage.

Provided **RESTful APIs** for seamless communication between frontend and backend.

Ensured smooth user experience with responsive UI and real-time updates.

Completed with proper **testing, documentation, and deployment**.

This project can be extended further by adding:

User authentication (login/registration)

Order and payment integration

Recommendation engine for product suggestions

The system is now ready for **final submission** and future **enhancements** as a complete e-commerce solution.

+