

**DEPARTMENT OF COMPUTER SCIENCE**

**SUBJECT**

**Presented to:**

**Submitted By:** Muhammad Mohsin Asghar, Ibrahim, Rafay

**Registration ID:** 221075, 221051, 221027

**BSIT-F22-A**

# E-Commerce Website Project Report

**1. Introduction**

This project involves building a full-stack e-commerce website using JavaScript (Node.js), HTML, CSS, and MongoDB. The website allows both users and administrators to interact with the platform, offering features such as product browsing, cart management, and user authentication. The backend API is built using Node.js and Express, while MongoDB is used as the database.

**2. Project Objectives**

* **Admin functionalities**:
  + Add, edit, and remove products.
  + View order history.
* **User functionalities**:
  + Browse and view products.
  + Add products to the cart.
  + Checkout and view the cart.
  + Register and login for user authentication.
* **General objectives**:
  + Implement secure authentication with JWT.
  + Set up a MongoDB database for data storage.
  + Ensure the website is user-friendly and responsive.

**3. Project Architecture**

**Frontend:**

The frontend of the project is developed using HTML, CSS, and JavaScript. It includes:

* **Home Page**: Displays a welcome message and navigational links.
* **Products Page**: Displays a list of products fetched from the backend.
* **Cart Page**: Displays the products in the cart and the option to proceed to checkout.
* **Login & Signup Pages**: For user authentication.
* **Admin Pages**: For the admin to manage products and view order history.

**Backend:**

The backend is built using Node.js and Express, connected to a MongoDB database for data storage. The key components are:

* **Controllers**: Contain the logic for handling product management, user authentication, and cart functionalities.
* **Models**: Define the structure for products, users, and orders in the database.
* **Routes**: Handle the API requests for user authentication, product management, and cart operations.
* **Middleware**: Protect routes requiring admin access or authentication.
* **Authentication**: Implemented using JSON Web Tokens (JWT) to secure the login and registration process.

**Database:**

MongoDB is used for data storage. The following collections are present:

* **Users**: Stores user information (name, email, password).
* **Products**: Stores product details (name, price, description, image).
* **Orders**: Stores order details, such as product IDs and user ID.

**4. Functionalities**

**User Side Features**

* **Login & Signup**: Users can register and log in to the website. Passwords are securely hashed before storage.
* **Product Browsing**: Users can browse products, see their details (name, description, price), and add them to the cart.
* **Shopping Cart**: Users can view their cart, update quantities, and remove items. The cart is stored in localStorage for persistent session management.
* **Checkout**: Users can proceed to checkout (currently a placeholder for future payment integration).

**Admin Side Features**

* **Product Management**: The admin can add new products, update product details, and delete existing products.
* **Order Management**: The admin can view all orders placed by users.

**Authentication**

* **JWT Authentication**: The system uses JWT tokens for user authentication. The token is issued upon successful login and is stored in the client's localStorage. The token is required for accessing certain protected routes.

**Security Features**

* **Password Hashing**: User passwords are hashed before being stored in the database, ensuring data security.
* **Authorization Middleware**: Admin-only routes are protected by middleware that checks for valid JWT tokens.

**5. Project Structure**

The project is organized as follows:

diff

Copy code

-config

-db.js

-controllers

-orderController.js

-productController.js

-userController.js

-data

-seeder.js

-middleware

-authMiddleware.js

-models

-orderModel.js

-productModel.js

-userModel.js

-node\_modules

-routes

-orderRoutes.js

-productRoutes.js

-userRoutes.js

.env

package.json

package-lock.json

seed.js

server.js

**Key Files and Their Purpose:**

* **server.js**: The entry point of the backend, where the Express server is initialized and routes are connected.
* **db.js**: Configures the MongoDB connection.
* **models**: Contains Mongoose models for User, Product, and Order.
* **controllers**: Handles the logic for products, users, and orders.
* **routes**: Defines the API endpoints for users, products, and orders.
* **authMiddleware.js**: Middleware to verify JWT tokens and ensure route protection.

**6. Implementation**

**Frontend Implementation:**

The frontend is built using basic HTML, CSS, and JavaScript. It handles:

* **Page Navigation**: Based on URL hash changes, different pages (home, products, cart, etc.) are displayed.
* **Product Display**: Products are fetched from the backend and displayed dynamically.
* **Cart Management**: Products can be added to or removed from the cart, and the cart count is updated.

The main scripts.js file handles the logic for:

* Fetching products from the backend.
* Adding/removing products to/from the cart.
* Handling login and signup forms.

**Backend Implementation:**

The backend is implemented using Node.js and Express:

* **User Registration & Login**: Users can register and log in, with password hashing and JWT authentication.
* **Product Management**: Admins can add, edit, and delete products.
* **Cart Management**: Although not implemented fully on the backend, the cart is stored on the client side for simplicity. Future versions may include saving cart data on the server.

**Database Operations:**

MongoDB is used to store data:

* **User Data**: Name, email, password (hashed).
* **Product Data**: Product details such as name, description, price, and image.
* **Order Data**: User order history, which links products to users.

**Security:**

* **Password Hashing**: Passwords are hashed using bcryptjs to ensure security.
* **JWT Tokens**: Used for secure authentication. Users must provide a token to access protected routes.

**7. Challenges and Solutions**

**Challenges:**

* **Product Management**: Implementing both user and admin functionality required managing different access levels (admin vs. user).
* **JWT Authentication**: Ensuring that JWT tokens are properly verified and passed with each protected request.
* **Frontend Integration**: Ensuring smooth interaction between the frontend and backend, particularly with product fetching and cart management.

**Solutions:**

* **Role-Based Access**: Implemented separate routes and middleware for admin and user roles.
* **JWT Token Handling**: Utilized JWT tokens in combination with localStorage for persistent sessions.
* **Async Requests**: Used async/await in both frontend and backend to handle asynchronous requests (e.g., fetching products, user login).

**8. Future Work**

* **Payment Gateway Integration**: Implementing an actual checkout process with payment gateway integration.
* **User Profile Management**: Allowing users to manage their profiles, change passwords, and view order history.
* **Order Management**: Enabling order placement and tracking.
* **Improved UI/UX**: Enhancing the design and usability of the frontend to improve the user experience.

**9. Conclusion**

This e-commerce project successfully implements the core features necessary for an online store. Users can browse products, add them to their cart, and log in to make purchases, while admins can manage products and view orders. With secure authentication and basic cart functionality, this project is a solid foundation for a fully functional e-commerce platform.