**🛒 eCommerce Web Application – Full Project Documentation**

**🔰 1. Project Title: Mini eCommerce Web Application (v4.1.16)**

**🎯 2. Project Objective:**

To build a simple, full-stack online shopping platform where users can:

* Sign up and log in securely.
* Browse, filter, and sort products.
* Add/remove products to/from the cart.
* Place and track orders.
* View/download invoices.
* Manage their profiles.
* Admin can add or delete products.

**🧱 3. Technical Stack**

| **Layer** | **Technology** |
| --- | --- |
| Frontend | HTML5, CSS3, JavaScript |
| Backend | Node.js + Express.js |
| Database | MongoDB (Mongoose ORM) |
| Server | Localhost / Node.js |
| Additional | bcrypt (password hash), pdfkit/html-pdf (invoice), multer (image upload) |

**🧩 4. System Modules**

**➤ 1. Landing Page**

* Displays key visuals.
* Navigation to Login and Signup.

**➤ 2. User Authentication**

* **Signup Page** – New users can register.
* **Login Page** – Users log in with credentials.
* Passwords securely stored (bcrypt).
* Session-based login using cookies or JWT (optional).

**➤ 3. Product Catalog**

* Product listing with:
  + Price range filter
  + Category filter
  + Sort (price ascending/descending)
* **Admin only**: Add/Delete product buttons.

**➤ 4. Cart Module**

* Add to Cart button per product.
* View/Edit cart.
* Live calculation of totals.
* Proceed to checkout.

**➤ 5. Order Module**

* Place an order from the cart.
* Orders stored with:
  + Order ID
  + Date
  + Items
  + Total amount
* View order list (filter by ID/date).
* Only user’s own orders shown.

**➤ 6. Invoice Generation**

* Generate downloadable invoice.
* Print invoice (contains customer, product, total info).

**➤ 7. Profile Page**

* View logged-in user info.
* Access to order history and logout option.

**➤ 8. Logout**

* Clear session, redirect to login.

**🛠️ 5. Backend API Endpoints**

| **Endpoint** | **Method** | **Description** |
| --- | --- | --- |
| /signup | POST | Registers a new user |
| /login | POST | Authenticates user |
| /products | GET | Get product list with filters |
| /add-product | POST | Add a new product (admin) |
| /delete-product/:id | DELETE | Delete a product (admin) |
| /cart/add | POST | Add item to cart |
| /cart/remove/:id | DELETE | Remove item |
| /cart | GET | Get cart items |
| /place-order | POST | Finalize order from cart |
| /orders | GET | View user-specific order list |
| /invoice/:orderId | GET | Generate/Download invoice |
| /profile | GET | Fetch user details |
| /logout | POST | Clear session/token |

**🗃️ 6. MongoDB Schema Design**

**➤ User**

js

CopyEdit

{

userId: String,

name: String,

password: String, // hashed

mobile: String,

dob: Date,

address: String

}

**➤ Product**

js

CopyEdit

{

title: String,

description: String,

category: String,

price: Number,

imageUrl: String

}

**➤ Cart**

js

CopyEdit

{

userId: ObjectId,

items: [

{

productId: ObjectId,

quantity: Number,

price: Number

}

]

}

**➤ Order**

js

CopyEdit

{

userId: ObjectId,

orderItems: [ { title, qty, price } ],

totalAmount: Number,

orderDate: Date,

address: String

}

**🧮 7. Key Workflows**

**➤ Signup & Login**

1. User submits registration form.
2. Backend hashes password and stores user.
3. User logs in → server verifies and returns session.

**➤ Product Browsing**

1. Product list fetched via GET /products.
2. Filter and sort applied client-side or server-side.

**➤ Cart & Order**

1. Add to cart (stores product & quantity).
2. "Place Order" generates unique Order ID, stores in DB.
3. Cart is cleared after order.

**➤ Invoice**

1. After placing order, users can download invoice.
2. Invoice includes:
   * User details
   * Order ID, date
   * List of items and total
3. PDF generated via pdfkit or similar.

**🧪 8. Features to Test**

| **Area** | **Test Cases** |
| --- | --- |
| Auth | Register, login with valid/invalid credentials |
| Product | Load list, apply filters, sort |
| Add Product | Admin can add new product |
| Delete Product | Admin deletes product |
| Cart | Add/remove items, calculate totals |
| Order | Place order, check DB |
| Order List | Filters by ID/date |
| Profile | Correct user info shows |
| Invoice | PDF generated correctly |

**🧰 9. Deployment Guide (Localhost)**

1. **Install MongoDB locally** or use cloud (MongoDB Atlas).
2. Create .env file with DB connection string.
3. Run:

bash

CopyEdit

npm install

npm run dev

1. Frontend can be opened directly via browser.

**📂 10. Project Folder Structure**

pgsql

CopyEdit

project/

├── client/

│ ├── \*.html

│ ├── css/

│ ├── js/

├── server/

│ ├── server.js

│ ├── config/db.js

│ ├── routes/

│ ├── controllers/

│ ├── models/

**🧾 11. Suggested HTML + JS File List**

**📁 HTML Pages**

* index.html (Landing)
* signup.html
* login.html
* product-list.html
* add-product.html
* delete-product.html
* cart.html
* order-list.html
* invoice.html
* profile.html

**📁 JavaScript Files**

* auth.js – Signup/Login
* products.js – Product listing
* cart.js – Add/remove cart
* orders.js – Order placement
* invoice.js – Generate invoice
* profile.js – Profile data
* utils.js – Common utilities

**📆 12. Project Timeline (for Beginner Developer)**

| **Day** | **Task** |
| --- | --- |
| Day 1 | Setup Node + MongoDB + DB schema |
| Day 2 | Signup + Login |
| Day 3 | Product listing + filters |
| Day 4 | Add/Delete Product |
| Day 5 | Cart + Place Order |
| Day 6 | Order list + invoice |
| Day 7 | Profile + Logout + Styling |
| Day 8 | Testing + Debugging + Polishing |

**📝 Project Requirements: Detailed Description**

This section explains each feature/module in detail, including user interactions, business logic, and backend responsibilities.

**🔹 1. Landing Page**

**Objective:**

The homepage serves as the user's first interaction with the application.

**Features:**

* Show eye-catching banner images or product promos.
* Buttons/Links to **Login** and **Signup** pages.
* Optional: Short welcome message or value proposition.

**🔹 2. Signup Page**

**Objective:**

Allow new users to create an account.

**Fields:**

* Full Name
* User ID (unique identifier)
* Password
* Re-enter Password
* Mobile Number
* Date of Birth
* Address

**Functionality:**

* All fields are required with client-side validation.
* Password must match in both fields.
* Mobile number and DOB format validated.
* On submission, a POST /signup API is called to store data in MongoDB.
* Passwords are hashed before storage (use bcrypt).

**🔹 3. Login Page**

**Objective:**

Enable users to access their account securely.

**Fields:**

* User ID
* Password
* Checkbox: “Remember Me” (uses cookies/localStorage)

**Functionality:**

* POST /login API to validate credentials.
* If valid, redirect to product listing page.
* If invalid, show error message.
* If “Remember Me” is selected, store session or token accordingly.

**🔹 4. Product Listing Page**

**Objective:**

Display all products with filtering, sorting, and cart actions.

**Features:**

* List of products with image, title, price, category.
* **Price Range Filter** (e.g., ₹100 – ₹1000)
* **Category Filter** (e.g., Electronics, Clothes)
* **Sorting**: Ascending or descending by price.
* **Add to Cart** button.
* If logged in as admin, also show:
  + **Add New Product**
  + **Delete Product**

**Backend:**

* GET /products with optional query parameters for filtering and sorting.

**🔹 5. Add New Product**

**Objective:**

Admin users can add products to the store.

**Fields:**

* Product Image
* Title
* Description
* Category
* Price

**Functionality:**

* Upload and preview image (can be stored in filesystem or base64).
* Send data to backend using POST /add-product.
* Backend stores product in MongoDB with image URL or file path.

**🔹 6. Delete Product**

**Objective:**

Allow admin to delete products.

**Features:**

* Show list of products with delete buttons.
* Apply same filters and sorting as in product listing.
* On delete, send request to DELETE /delete-product/:id.

**🔹 7. Cart Management**

**Objective:**

Let users add, view, and manage products in their cart.

**Features:**

* Add item to cart from product list.
* View cart on separate page.
* Change quantity, remove item.
* Auto calculate subtotal, taxes (if added), and total.
* “Place Order” button.

**Backend:**

* POST /cart/add, DELETE /cart/remove/:id, GET /cart.

**🔹 8. Place Order**

**Objective:**

Complete purchase of products in cart.

**Features:**

* Finalize cart and store order details.
* Generate order ID, date, items, total amount.
* Clear cart after placing order.
* Redirect to order confirmation screen.

**Backend:**

* POST /place-order to save the order.
* Order data includes: userId, cart items, amount, delivery address.

**🔹 9. Order List Page**

**Objective:**

Show all past orders of the logged-in user.

**Features:**

* List of orders: Order ID, date, total, number of items.
* **Filters**: Order ID or Date range.
* **Buttons**: Download Invoice / Print Invoice.
* Only show the orders of the current logged-in user.

**Backend:**

* GET /orders (authenticated).
* Filter logic on backend using query parameters.

**🔹 10. Invoice Generation**

**Objective:**

Allow users to download or print invoices of orders.

**Features:**

* Show customer info, address, order summary, date, and amount.
* Print-friendly layout.
* Button to download as PDF.

**Backend:**

* GET /invoice/:orderId generates a dynamic HTML/PDF invoice using libraries like pdfkit, html-pdf, etc.

**🔹 11. User Profile Page**

**Objective:**

Display and manage user-specific information.

**Features:**

* View user data: name, contact, DOB, address.
* Link to view all orders.
* Logout button.

**Backend:**

* GET /profile to fetch data.
* Optional: PUT /profile to update info.

**🔹 12. Logout**

**Objective:**

Securely end the user session.

**Features:**

* Clear login tokens, sessions, or cookies.
* Redirect to login or landing page.

**Backend:**

* POST /logout: May just invalidate session/token client-side for simplicity.

**🛡️ Optional Enhancements (Future Scope)**

* Admin dashboard with product/order analytics.
* Password reset via email/OTP.
* Responsive UI using Bootstrap or Tailwind CSS.
* JWT-based auth for mobile support.
* Ratings and reviews per product.

Would you like me to generate one or more of these next?

* Full database schema files?
* REST API design in Swagger/OpenAPI format?
* Wireframe/UX suggestion?
* HTML + JS template pages?