# Detailed Designed Document : **E-Shop**

- E-Shop boasts an extensive collection of products across various categories, ensuring that users can find everything they need in one convenient place. This website utilize robust encryption protocols to ensure that all transactions are secure. Our seamless checkout process further enhances the overall shopping experience.

## 1. Objectives

- Provide a user-friendly interface for customers to buy product, And goods within their budget.
- Implement robust security measures to protect user data.
- Ensure scalability to accommodate future growth.

## 2. System Overview

### 2.1 System Architecture

The system will follow a client-server architecture with a responsive web-based front end and a back-end server.

### 2.2 Key Features

- User registration and authentication
- Easy product buying and contains admin panel.
- User will get an email when payment is done.
- Secured payment gateway.

## 2.3 User Roles

- Guest
- Registered Users

## 2.4 Technologies Used

- Front-end: HTML, Tailwind CSS/CSS, JavaScript (React)
- Back-end: Node.js
- Database: MongoDB
- Authentication: JWT
- Additional tools/libraries as needed.

# 3. Database Design

### Entity-Relationship Diagram

**User Entity:**

- Username: User's username for identification.
- Email: User's email address for communication.
- Password: Securely stored password for authentication.

**Brands Entity:**

- Label : The label of product from which the product belongs.
- value: The value of the product.

**Cart Entity :**

- Quantity : The quantity of product brought.
- Product : Name of the product.
- User : Name of the user who brought the product.
- Size : size of the product which is brought(If it is clothing product)

# 4. Schema

## 1. Users Schema

**User Schema:**

| Field | Type | Required | Description |
|---|---|---|---|
| _id | ObjectId | Auto | Unique identifier for the user |
| username | String | Yes | Username for the user |
| email | String | Yes | Email address of the user |
| password | String | Yes | Hashed password for user authentication |

## 2. Brands Schema

**Brands Schema:**

| Field | Type | Required | Description |
|---|---|---|---|
| _id | ObjectId | Auto | Unique identifier for the brand |
| name | String | Yes | Name of the brand |
| description | String | Yes | Description or details about the brand |
| createdAt | Date | Auto | Timestamp for when the brand was created |
| updatedAt | Date | Auto | Timestamp for when the brand was last updated |

## 3. Product Schema

**Product Schema:**

| Field | Type | Required | Description |
|---|---|---|---|
| _id | ObjectId | Auto | Unique identifier for the product |
| name | String | Yes | Name of the product |
| description | String | Yes | Description or details about the product |
| price | Number | Yes | Price of the product |
| category | String | Yes | Category to which the product belongs |
| imageUrl | String | Yes | URL of the product image |
| stockQuantity | Number | Yes | Quantity available in stock |
| createdAt | Date | Auto | Timestamp for when the product was created |
| updatedAt | Date | Auto | Timestamp for when the product was last updated |

## 4. Cart Schema

**User Schema:**

| Field | Type | Required | Description |
|---|---|---|---|
| _id | ObjectId | Auto | Unique identifier for the user |
| username | String | Yes | Username for the user |
| email | String | Yes | Email address of the user |
| password | String | Yes | Hashed password for user authentication |

# 5. APIs

## Authentication APIs:

| API | Endpoint | Method | Description |
|---|---|---|---|
| User Registration | `/api/users/register` | POST | Create a new user account. |
| User Login | `/api/users/login` | POST | Authenticate a user and generate a token. |
| User Logout | `/api/users/logout` | POST | Invalidate the user's token. |

**Brand APIs:**

| API Endpoint | HTTP Method | Description |
|---|---|---|
| `/api/brands` | GET | Retrieve a list of all brands |
| `/api/brands/:id` | GET | Retrieve details of a specific brand by ID |
| `/api/brands` | POST | Create a new brand |
| `/api/brands/:id` | PUT | Update details of a specific brand by ID |
| `/api/brands/:id` | DELETE | Delete a specific brand by ID |

### Cart APIs:

| API Endpoint | HTTP Method | Description |
| --- | --- | --- |
| `/api/cart` | GET | Retrieve the current user's shopping cart |
| `/api/cart` | POST | Add a product to the user's shopping cart |
| `/api/cart/:id` | PUT | Update quantity or details of a product in the cart |
| `/api/cart/:id` | DELETE | Remove a product from the user's shopping cart |

### Product APIs:

| API Endpoint | HTTP Method | Description |
| --- | --- | --- |
| `/api/products` | GET | Retrieve a list of all products |
| `/api/products/:id` | GET | Retrieve details of a specific product by ID |
| `/api/products` | POST | Create a new product |
| `/api/products/:id` | PUT | Update details of a specific product by ID |
| `/api/products/:id` | DELETE | Delete a specific product by ID |

### Order APIs:

| API Endpoint | HTTP Method | Description |
| --- | --- | --- |
| `/api/orders` | GET | Retrieve a list of all orders |
| `/api/orders/:id` | GET | Retrieve details of a specific order by ID |
| `/api/orders` | POST | Create a new order |
| `/api/orders/:id` | PUT | Update status or details of a specific order by ID |
| `/api/orders/:id` | DELETE | Cancel a specific order by ID |

# 6. Deployment

This MERN project, named E-shop, is scheduled for deployment on Vercel.