MULTI-VENDOR ECOMMERCE WEBSITE DOCUMENTATION

TABLE OF CONTENTS

1. INTRODUCTION

This document provides a comprehensive guide for developing a multi-vendor eCommerce platform. The system enables multiple vendors to manage and sell their products while customers can browse and purchase from various vendors in a unified platform.

2. USER ROLES AND RESPONSIBILITIES

2.1. ADMIN

- **Responsibilities:**
    - Manage users, vendors, and site settings.
    - Oversee the approval of vendor applications and content moderation.
    - Generate and analyze reports.
    - Ensure overall platform integrity and functionality.

2.2. VENDOR

- **Responsibilities:**
    - Manage their own products and inventory.
    - Process orders and handle customer service.
    - Create and manage promotions and discounts.
    - View and analyze sales performance.

2.3. CUSTOMER

- **Responsibilities:**
  - Register, login, and manage their account.
  - Browse, search, and filter products.
  - Add products to the cart and complete purchases.
  - Track orders and leave reviews.

3. FEATURES

3.1. ADMIN FEATURES

- **Dashboard:** View key metrics, notifications, and site statistics.
- **User Management:** Add, edit, delete, and manage user roles and permissions.
- **Vendor Management:** Approve or reject vendor applications, manage vendor profiles.
- **Content Moderation:** Monitor and moderate reviews, product listings, and other content.
- **Reporting:** Generate sales, traffic, and user activity reports.

3.2. VENDOR FEATURES

- **Product Management:** Add, update, and delete products; manage inventory.
- **Order Management:** Process and track orders; handle returns and refunds.
- **Promotions:** Create and manage discounts and promotional offers.
- **Profile Management:** Update vendor profile details and settings.
- **Sales Reports:** Access and analyze sales performance data.

3.3. CUSTOMER FEATURES

- **Account Management:** Register, log in, and update account details.
- **Product Search:** Search for products using various filters and keywords.
- **Shopping Cart:** Add products to the cart, view, and edit cart contents.
- **Checkout:** Complete purchases, select shipping options, and enter payment information.
- **Order Tracking:** Track order status and view order history.

- **Reviews and Ratings:** Leave reviews and ratings for products.

4. USE CASES

4.1. ADMIN USE CASES

Use Case: Manage Users

- **Description:** Admin can view, add, edit, and delete user accounts.
- **Actors:** Admin
- **Preconditions:** Admin is logged in.
- **Basic Flow:**
    1. Admin navigates to the user management section.
    2. Admin selects a user to view or edit.
    3. Admin makes necessary changes and saves.
- **Postconditions:** User details are updated in the system.

Use Case: Approve Vendor Applications

- **Description:** Admin reviews and approves or rejects vendor applications.
- **Actors:** Admin
- **Preconditions:** Vendor application is submitted.
- **Basic Flow:**
    1. Admin reviews the vendor application.
    2. Admin approves or rejects the application.
    3. System updates vendor status accordingly.
- **Postconditions:** Vendor is either approved or rejected.

4.2. VENDOR USE CASES

Use Case: Manage Products

- **Description:** Vendors can add, update, or delete their product listings.
- **Actors:** Vendor
- **Preconditions:** Vendor is logged in.
- **Basic Flow:**
    1. Vendor navigates to the product management section.
    2. Vendor selects to add a new product or edit an existing one.
    3. Vendor inputs product details and saves.
- **Postconditions:** Product is added or updated in the system.

Use Case: Process Orders

- **Description:** Vendors process and manage customer orders.
- **Actors:** Vendor
- **Preconditions:** An order is placed.
- **Basic Flow:**
    1. Vendor receives a new order notification.
    2. Vendor processes the order, updates status, and prepares for shipment.
    3. Vendor marks the order as shipped.
- **Postconditions:** Order status is updated and customer is notified.

4.3. CUSTOMER USE CASES

Use Case: Search for Products

- **Description:** Customers search for products using keywords and filters.
- **Actors:** Customer
- **Preconditions:** Customer is on the homepage.
- **Basic Flow:**
    1. Customer enters search terms or applies filters.
    2. System displays matching products.
- **Postconditions:** Customer views a list of products matching the search criteria.

Use Case: Checkout

- **Description:** Customers complete the purchase of products in their cart.
- **Actors:** Customer
- **Preconditions:** Customer has products in their cart.
- **Basic Flow:**
    1. Customer reviews cart contents.
    2. Customer enters shipping and payment details.
    3. Customer confirms the purchase.
- **Postconditions:** Order is placed and confirmation is sent to the customer.

5. FUNCTIONAL REQUIREMENTS

5.1. USER MANAGEMENT

- **Admin:**
  - Create, read, update, and delete (CRUD) operations for user accounts.
  - Assign and manage roles and permissions.
- **Vendor:**
  - Register, verify, and manage their own accounts.
- **Customer:**
  - Register, log in, and update personal information.

5.2. PRODUCT MANAGEMENT

- **Vendor:**
  - Add products with details such as name, description, price, and images.
  - Update and delete products as needed.
  - Manage inventory levels and product variants.
- **Customer:**
  - View product details and images.

5.3. ORDER MANAGEMENT

- **Vendor:**
  - View incoming orders, update status (e.g., processing, shipped).
  - Handle order returns and refunds.
- **Customer:**
  - Place orders, view order status, and request returns.

5.4. PAYMENT GATEWAY INTEGRATION

- **Description:** Integration with payment processors for secure transactions.
- **Requirements:**
  - Support for multiple payment methods (credit cards, PayPal, etc.).
  - Secure handling of payment data.

5.5. REVIEWS AND RATINGS

- **Vendor:**
  - Respond to customer reviews.
- **Customer:**
  - Leave reviews and ratings for products.
  - View product ratings and reviews from other customers.

5.6. SEARCH AND FILTERING

- **Description:** Advanced search and filtering options for products.
- **Features:**
  - Search by keywords, categories, price range, and ratings.
  - Sort results by relevance, price, and newest.

5.7. NOTIFICATIONS

- **Description:** Notifications for order updates, promotions, and other relevant events.
- **Types:**
  - Email and/or SMS notifications.
  - In-app notifications for real-time updates.

6. NON-FUNCTIONAL REQUIREMENTS

6.1. PERFORMANCE

- **Description:** The system must handle high traffic volumes and concurrent transactions.
- **Metrics:**
  - Response times under 2 seconds for key operations.
  - Support for X concurrent users.

6.2. SECURITY

- **Description:** Protect user data and transactions.
- **Features:**
  - SSL/TLS encryption for data transmission.
  - Secure storage of sensitive data (e.g., hashed passwords).

    - Regular security audits.

6.3. USABILITY

- **Description:** User-friendly and accessible design.
- **Features:**
    - Intuitive navigation and clear calls-to-action.
    - Accessibility compliance (e.g., WCAG 2.1).

6.4. SCALABILITY

- **Description:** Ability to scale with growing numbers of users and products.
- **Features:**
    - Load balancing and horizontal scaling options.
    - Modular architecture for easy feature expansion.

6.5. MAINTAINABILITY

- **Description:** Easy to maintain and update.
- **Features:**
    - Well-documented codebase.
    - Modular design with clear separation of concerns.

7. PROJECT PHASES

7.1. PLANNING

- **Activities:**
    - Define project scope and objectives.
    - Identify stakeholders and gather detailed requirements.
    - Develop a project timeline and plan.

7.2. DESIGN

- **Activities:**
    - Create wireframes and mockups for UI/UX design.
    - Design the database schema and system architecture.
    - Define API endpoints and data structures.

7.3. DEVELOPMENT

- **Frontend:** Develop user interfaces using React.
- **Backend:** Implement APIs and business logic using Laravel.
- **Integration:** Connect frontend with backend services.

7.4. TESTING

- Activities:
  - Perform unit testing for individual components.
  - Conduct integration testing to ensure components work together.
  - Execute end-to-end testing for complete workflows.
  - Perform usability and performance testing.

7.5. DEPLOYMENT

- Activities:
  - Set up production environment.
  - Deploy application to live servers.
  - Monitor deployment and resolve any issues.

7.6. MAINTENANCE

- Activities:
  - Regularly update and patch the system.
  - Monitor performance and address user feedback.
  - Implement new features and improvements as needed.

8. TECHNOLOGY STACK

8.1. FRONTEND

- **Framework:** React
- **Libraries:** Redux (state management), React Router (routing), Axios (HTTP requests)
- **Styling:** CSS-in-JS (styled-components), Bootstrap/Tailwind CSS

8.2. BACKEND

- **Framework:** Laravel
- **Database:** MySQL or PostgreSQL
- **APIs:** RESTful APIs
- **Authentication:** Laravel Passport or JWT

8.3. DATABASE

- **Schema Design:** Define tables for users, products, orders, etc.
- **DBMS:** MySQL or PostgreSQL

9. API DOCUMENTATION

9.1. AUTHENTICATION

- **Endpoint:** `/api/auth/login`
- **Method:** POST
- **Request:** `{ "email": "user@example.com", "password": "password" }`
- **Response:** `{ "token": "jwt_token" }`

9.2. USER MANAGEMENT

- **Endpoint:** `/api/users`
- **Method:** GET/POST/PUT/DELETE
- **Request:** `{ "name": "User", "email": "user@example.com" }`
- **Response:** `{ "user": { ... } }`

9.3. PRODUCT MANAGEMENT

- **Endpoint:** `/api/products`
- **Method:** GET/POST/PUT/DELETE
- **Request:** `{ "name": "Product", "price": 100 }`
- **Response:** `{ "product": { ... } }`

9.4. ORDER MANAGEMENT

- **Endpoint:** `/api/orders`
- **Method:** GET/POST/PUT
- **Request:** `{ "order_id": "123", "status": "shipped" }`
- **Response:** `{ "order": { ... } }`

9.5. PAYMENT PROCESSING

- **Endpoint:** `/api/payments`
- **Method:** POST
- **Request:** `{ "amount": 100, "method": "credit_card" }`
- **Response:** `{ "payment": { ... } }`

10. UI/UX DESIGN GUIDELINES

- **Design Principles:** Focus on clarity, simplicity, and responsiveness.
- **Accessibility:** Ensure compliance with WCAG standards for accessibility.
- **Branding:** Maintain consistent use of brand colors, fonts, and logos throughout the site.