**Multivendor E-commerce Project**

**Requirements**

**1. User Roles and Permissions:**

* **Customer:**
  + Registration and login.
  + Browse products.
  + Add/remove products to/from the shopping cart.
  + Place orders.
  + View order history.
  + Leave reviews and ratings.
* **Vendor:**
  + Registration and login.
  + Manage product listings.
  + Add/edit/delete products.
  + Track inventory.
  + View order history.
  + Receive notifications for new orders.
  + Respond to customer reviews.
* **Admin:**
  + Manage users and roles.
  + Monitor transactions and orders.
  + Resolve disputes.
  + View analytics and reports.
  + Manage system settings.

**2. Product Management:**

* Product listing with details (name, description, price, images, etc.).
* Categorization and tagging.
* Inventory tracking and notifications.
* Product variations (sizes, colors, etc.).
* Discounts and promotions.

**3. Order Management:**

* Order placement and tracking.
* Order history and status updates.
* Invoicing and receipts.
* Shipping and delivery information.
* Return and refund processes.

**4. Payment Gateway Integration:**

* Integration with secure and popular payment gateways.
* Support for multiple payment methods (credit/debit cards, digital wallets, etc.).
* Automatic payment distribution for vendors.

**5. Rating and Reviews:**

* Customer reviews and ratings for products.
* Vendor ratings based on performance.
* Moderation tools for inappropriate content.

**6. Search and Filters:**

* Advanced search functionality.
* Filters for price range, category, brand, etc.
* Sort options for search results.

**7. User Authentication and Authorization:**

* Secure user registration and login.
* Two-factor authentication for enhanced security.
* Role-based access control.

**8. Notifications:**

* Email/SMS notifications for order confirmation, shipping updates, etc.
* Subscription options for product updates and promotions.
* Alerts for low inventory.

**9. Reporting and Analytics:**

* Sales analytics for vendors.
* Reports on popular products and customer behavior.
* Financial reports for administrators.

**10. Responsive Design:**

* Mobile-friendly design for a seamless experience on various devices.
* Cross-browser compatibility.

**11. Security Measures:**

* SSL for secure data transmission.
* Encryption of sensitive information.
* Regular security audits and updates.

**12. Customer Support:**

* Live chat, email, or ticketing system.
* Help center with FAQs and guides.

**13. Localization and Internationalization:**

* Support multiple languages and currencies.

**14. Return and Refund Policies:**

* Clearly defined policies communicated to users.

**15. SEO Optimization:**

* SEO-friendly URLs and metadata.
* Sitemap and robots.txt implementation.

**16. Legal and Compliance:**

* Compliance with data protection laws and e-commerce regulations.

**17. Scalability:**

* Design the system for scalability with increasing user and product volumes.

**18. Third-Party Integrations:**

* Integration with third-party services such as shipping carriers, analytics tools, etc.

**19. Backup and Recovery:**

* Regular data backups.
* Robust recovery plan in case of data loss.

**20. Testing:**

* Comprehensive testing of all features, including unit testing, integration testing, and user acceptance testing.

**21. Documentation:**

* Thorough documentation for code, APIs, and user guides.

**22. User Feedback Mechanism:**

* System for collecting and analyzing user feedback for continuous improvement.

**34. Future Enhancements:**

Consideration for future features and technology upgrades

**Data Model**

**User:**

1. \_id
2. userName (unique)
3. password
4. email
5. userType (Enum: customer, vendor, admin)
6. isDeleted
7. createdAt (from schema timestamp)
8. updatedAt (from schema timestamp)

**Customer:**

1. \_id
2. user (objectId of user model)
3. name (object of firstName, middleName and lastName)
4. userName (unique)
5. gender
6. email
7. mobileNo
8. image
9. isDeleted

**Vendor:**

1. \_id
2. user (objectId of user model)
3. name (object of firstName, middleName and lastName)
4. userName (unique)
5. email
6. mobileNo
7. image
8. isDeleted
9. orderHistory (Array of order objectId references)
10. reviews (Array of review objectId references)

**Admin:**

1. \_id
2. user (objectId of user model)
3. userName (unique)
4. Name (object of firstName, middleName and lastName)
5. email
6. isDeleted
7. mobileNo
8. image

**Product:**

1. \_id
2. name
3. description
4. price
5. images (Array of image URLs)
6. category
7. vendor (ObjectId of vendor)
8. inventory
   1. quantity (default 0)
   2. lowStockNotification (default false)
9. discounts (Array of discount codes or offers)
10. review (Array of ObjectId of review)
11. isDeleted
12. createdAt
13. updatedAt

**Order:**

1. \_id
2. products (Array of product objectId references)
3. totalPrice
4. customer (objectId reference to Customer)
5. status (Enum: placed, shipped, delivered, canceled; default: placed)
6. invoice
7. shippingInfo
   1. address
   2. city
   3. state
   4. country
   5. postalCode
8. createdAt
9. updatedAt

**Review:**

1. \_id
2. customer (ObjectId of customer)
3. product (ObjectId of product)
4. rating
5. description
6. createdAt
7. updatedAt

**API Endpoints**

**User:**

1. **/users/create-user (POST for customer)**
2. **/users/create-vendor (POST for vendor)**
3. **/users/create-admin (POST for admin)**

**Customer:**

1. **/users (GET all customer)**
2. **/users/:id (GET single customer)**
3. **/users/:id/ordersv(GET orders of a customer)**
4. **/users/:id (PUT/PATCH single customer)**
5. **/users/:id (DELETE customer)**

**Vendor:**

1. **/vendors (GET all vendors)**
2. **/ vendors /:id (GET single vendor)**
3. **/ vendors /:id (PUT/PATCH single vendor)**
4. **/vendors/:id (DELETE vendor)**

**Admin:**

1. **/admins (GET all admins)**
2. **/ admins/:id (GET single admin)**
3. **/ admins/:id (PUT/PATCH single admin)**
4. **/ admins/:id (DELETE admin)**

**Products:**

1. **/products (GET all products)**
2. **/ products/:id (GET single product)**
3. **/ products/:id (PUT/PATCH single product)**
4. **/ products/:id (DELETE product)**

**Orders:**

1. **/orders (GET all orders)**
2. **/ orders/:id (GET single order)**
3. **/ orders/:id (PUT/PATCH single order)**
4. **/ orders/:id (DELETE order)**

**Auth:**

1. **/auth/login**
2. **/auth/change-password**
3. **/auth/reset-password**
4. **/auth/forget-password**
5. **/auth/refresh-token**

TODO:

1. If no user or customer find then show user not found in response. Now showing fetched successfully.
2. Reduce repetition of name interface in customer, vendor and admin.
3. Reduce repetition of name model in customer, vendor and admin.
4. Reduce repetition of name validation in customer, vendor and admin.
5. Have to show the user, customer, vendor , admins which isDeleted are false.
6. Have to fixed the type of error in service
7. Have to fixed duplicate email for customer, vendor and admin
8. For now mobile no not making unique. Will make unique later.
9. For now save image in cloudinary with multer with customer routes only for practice purpose. After we don’t use this in customer routes. Cause free hosting doesn’t support multer file system.

git config --global --add safe.directory '\*'

git config --global --add safe.directory /some/path