Final Project Specification – Laravel Kandura Store

📋 Introduction

This document provides comprehensive and precise specifications for the **Kandura Store** project - an e-commerce system specialized in designing and selling traditional Kanduras. The project requires students to apply advanced Laravel knowledge, including database design, RESTful API development, and role-based access control implementation.

1️⃣ Project Overview

Objective

Develop an integrated e-commerce system specialized in designing and selling customized Kanduras with advanced personalization capabilities.

Target Audience

Laravel course students - final project requiring implementation of best practices and professional standards.

Core Focus

- Professional database design with complex relationships
- Comprehensive role-based access control system
- Secure and efficient RESTful API development
- Clean code following Laravel standards

Scope

Complete system including user interface, admin dashboard, database, and API.

2️⃣ Core Entities

User (User Account)

Field	Type	Notes
id	PK	Unique user identifier
name	string	Full user name
email	string	Email address (unique)
phone	string	Phone number
password	string	Encrypted password
role	enum(‘guest’ , ‘user’ , ‘admin’ , ‘super_admin’)	User role in system
profile_image	string	Profile picture (optional)
is_active	boolean	Account active status
email_verified_at	timestamp	Email verification date
created_at / updated_at	timestamps	Creation and update timestamps

Purpose: Store user account information and authentication credentials.

Relationships:

- `hasMany(Address)` : User has multiple addresses
- `hasMany(Measurement)` : User has multiple measurements
- `hasMany(Design)` : User has multiple designs
- `hasMany(Order)` : User has multiple orders
- `hasOne(Wallet)` : User has one wallet
- `hasMany(Review)` : User can leave multiple reviews

Order (Purchase Order)

Field	Type	Notes
id	PK	Unique order identifier
user_id	FK	Order owner user ID
address_id	FK	Delivery address ID
total	decimal(10,2)	Order total amount
payment_method	enum(‘cash’ , ‘wallet’ , ‘card’)	Payment method
status	enum(‘pending’ , ‘confirmed’ , ‘processing’ , ‘completed’ , ‘cancelled’)	Current order status
coupon_id	FK	Applied coupon ID (optional)
notes	text	Additional notes
created_at / updated_at	timestamps	Creation and update timestamps

Purpose: Store order information and financial transactions.

Relationships:

- `belongsTo(User)` : Order belongs to one user
- `belongsTo(Address)` : Order linked to one address
- `belongsToMany(Design)` : Order contains multiple designs
- `hasOne(Invoice)` : Order has one invoice
- `hasMany(Review)` : Order can have multiple reviews

Address (Delivery Address)

Represents user delivery addresses. Must contain complete delivery information (city, district, street, house number, etc.). Optional geographic information (Geo-location) for map integration.

Purpose: Store multiple delivery addresses for each user.

Relationships:

- `belongsTo(User)` : Address belongs to one user
- `hasMany(Order)` : Address can be linked to multiple orders

Measurement (Body Measurements)

Represents user body measurements (chest, waist, sleeve, shoulder, hip, height). Measurements can come from manual input or AI-based systems.

Purpose: Store user measurements to facilitate design and customization process.

Relationships:

- `belongsTo(User)` : Measurement belongs to one user
- `hasMany(Design)` : Measurement can be linked to multiple designs

Design (Kandura Design)

Represents customized Kandura design for user. Includes fabric type, color, quantity, and embroidery details. Each design can be part of multiple orders.

Purpose: Store details of customized Kandura designs.

Relationships:

- `belongsTo(User)` : Design belongs to one user
- `belongsTo(Measurement)` : Design linked to one measurement
- `belongsToMany(Order)` : Design can be part of multiple orders
- `hasMany(DesignOptionSelection)` : Design has selected options

3️⃣ Advanced Entities

DesignOption (Design Option)

Represents general customization options available when designing Kandura (collar types, sleeve types, pocket styles, etc.). Managed by administration. Each option should have clear name, representative image, and active/inactive status.

Purpose: Provide pre-defined set of customizable options.

DesignOptionSelection (Option Selection)

Represents actual user selections when designing Kandura. Links design to selected option. Can contain free-text value if option allows custom input.

Purpose: Track selected options for each design.

Wallet (Digital Wallet)

User’s digital wallet. Used to store balance for purchases or rewards. Must contain current balance and creation date.

Purpose: Manage user’s digital financial balance.

WalletTransaction (Wallet Transaction)

Records all wallet operations (deposit, withdraw, reward). Must contain operation type, amount, operation description, and timestamp.

Purpose: Maintain detailed transaction log for review and audit purposes.

Coupon (Discount Coupon)

Discount code applicable during checkout. Contains code, discount percentage, expiration date, and activation status.

Purpose: Provide flexible and manageable discount system.

Invoice (Order Invoice)

Detailed order invoice. Contains unique invoice number, total, tax, shipping, and PDF invoice link.

Purpose: Generate and store official invoices for orders.

Notification (System Notification)

System alerts directed to users. Contains title, message, and read status.

Purpose: Notify users of important events (order confirmation, shipment, etc.).

Review (User Review)

User reviews of products or services. Contains rating (1-5), comment, and status (approved/rejected).

Purpose: Collect user feedback and improve service quality.

4️⃣ Roles & Permissions

Guest (Unregistered User)

Unregistered visitor.

Permissions:

- View homepage and products
- Register and login
- View contact information and terms

User (Registered User)

Registered user who makes purchases and designs.

Permissions:

- Create, update, and delete profile
- Manage addresses (add, edit, delete)
- Manage measurements (add, edit, delete)
- Create, update, and delete designs
- View and create orders
- View wallet balance and transactions
- Leave reviews
- View notifications

Admin (Administrator)

Staff member managing orders and content.

Permissions:

- View all users and disable/delete accounts
- View and change status of all orders
- View, edit, and delete all designs
- Manage coupons (create, edit, delete)
- Manage design options
- View and control reviews (approve/reject)
- Send notifications

- Manage wallet (add/withdraw balance)

Super Admin (System Administrator)

Top administrator with full system permissions.

Permissions:

- All Admin permissions
- Manage other administrators (add, edit, delete)
- Manage system settings
- View reports and statistics
- Manage roles and permissions

5️⃣ Laravel Core Components

Service Layer

Use service layer to separate business logic from controllers. Examples: `OrderService`, `DesignService`, `WalletService`.

Events & Observers

Use events and observers for interconnected operations. Example: When order is created, fire `OrderCreated` event to send notification.

Queues & Jobs

Use queues and jobs for time-consuming operations. Examples: Email sending, payment processing.

API Resources

Use API resources to format data when returning from API.

Policies & Gates

Use policies and gates for fine-grained access control. Example: Verify user is design owner before allowing edit.

Query Scopes

Use scopes to reuse repeated queries. Examples: `->active()`, `->completed()`.

Validation Requests

Use request classes to validate data in each Create/Update operation.

Soft Deletes

Use soft deletes to retain data after deletion, especially for users and orders.

Pagination

Display data in organized manner with pagination (e.g., 10 items per page).

6️⃣ API Design

Students must design and implement comprehensive RESTful API following best practices. All endpoints must be secured with authentication and authorization systems.

Main Endpoints

Users

Method	Path	Description	Permissions
POST	`/api/auth/register`	Register new user	Guest
POST	`/api/auth/login`	User login	Guest
POST	`/api/auth/logout`	User logout	User
GET	`/api/users/{id}`	Get user data	User (Self)
PUT	`/api/users/{id}`	Update user data	User (Self)
GET	`/api/users`	List all users	Admin
DELETE	`/api/users/{id}`	Delete user	Super Admin

Addresses

Method	Path	Description	Permissions
GET	`/api/addresses`	List user addresses	User
POST	`/api/addresses`	Create new address	User
PUT	`/api/addresses/{id}`	Update address	User (Owner)
DELETE	`/api/addresses/{id}`	Delete address	User (Owner)
GET	`/api/addresses/{id}`	Get address details	User (Owner)

Measurements

Method	Path	Description	Permissions
GET	`/api/measurements`	List user measurements	User
POST	`/api/measurements`	Create new measurement	User
PUT	`/api/measurements/{id}`	Update measurement	User (Owner)
DELETE	`/api/measurements/{id}`	Delete measurement	User (Owner)
GET	`/api/measurements/{id}`	Get measurement details	User (Owner)

Designs

Method	Path	Description	Permissions
GET	`/api/designs`	List user designs	User
POST	`/api/designs`	Create new design	User
PUT	`/api/designs/{id}`	Update design	User (Owner)
DELETE	`/api/designs/{id}`	Delete design	User (Owner)
GET	`/api/designs/{id}`	Get design details	User (Owner)
POST	`/api/designs/{id}/options`	Add options to design	User (Owner)

Orders

Method	Path	Description	Permissions
GET	`/api/orders`	List user orders	User
POST	`/api/orders`	Create new order	User
GET	`/api/orders/{id}`	Get order details	User (Owner) / Admin
PUT	`/api/orders/{id}/status`	Update order status	Admin
GET	`/api/orders`	List all orders	Admin
DELETE	`/api/orders/{id}`	Cancel order	Admin

Wallet

Method	Path	Description	Permissions
GET	`/api/wallet`	Get wallet balance	User
GET	`/api/wallet/transactions`	List wallet transactions	User
POST	`/api/wallet/deposit`	Add balance	Admin
POST	`/api/wallet/withdraw`	Withdraw balance	Admin

Coupons

Method	Path	Description	Permissions
POST	`/api/coupons/validate`	Validate coupon code	User
GET	`/api/coupons`	List all coupons	Admin
POST	`/api/coupons`	Create new coupon	Admin
PUT	`/api/coupons/{id}`	Update coupon	Admin
DELETE	`/api/coupons/{id}`	Delete coupon	Admin

Invoices

Method	Path	Description	Permissions
GET	`/api/invoices/{id}`	Get invoice	User (Owner) / Admin
GET	`/api/invoices/{id}/pdf`	Download invoice PDF	User (Owner) / Admin
GET	`/api/invoices`	List all invoices	Admin

Notifications

Method	Path	Description	Permissions
GET	`/api/notifications`	List user notifications	User
PUT	`/api/notifications/{id}/read`	Mark notification as read	User
DELETE	`/api/notifications/{id}`	Delete notification	User
POST	`/api/notifications/send`	Send notification	Admin

Response Standards

All responses must follow this format:

```
{
  "success": true,
  "message": "Operation completed successfully",
  "data": {
    // Required data
  },
  "timestamp": "2025-11-02T10:30:00Z"
}
```

Error Handling

Error Code	Description
400	Bad Request - Invalid data
401	Unauthorized - Not authenticated
403	Forbidden - No permission
404	Not Found - Resource not found
422	Unprocessable Entity - Validation error
429	Too Many Requests - Rate limit exceeded
500	Internal Server Error - Server error

Authentication & Authorization

All protected endpoints must require:

- **Bearer Token:** Obtained from `POST /api/auth/login`
- **Header:** `Authorization: Bearer {token}`

- **JWT Expiration:** Set expiration time (e.g., 24 hours)
- **Refresh Token:** For updating token without re-login

Rate Limiting

- **General:** 60 requests per minute per IP
- **Login:** 5 attempts per minute
- **API:** 1000 requests per hour per user

7️⃣ External Integrations

Required Integrations:

Payment Gateway: Implement secure payment system. Can use Stripe, Tabby, or Mada for Saudi Arabia.

Notification System: Use Firebase or Twilio for email or SMS notifications.

Optional Integrations:

Cloud Storage: Use Cloudinary, AWS S3, or Bunny CDN for image and file storage.

Data Import/Export: Support Excel or CSV import and export functionality.

Maps: Support Google Maps or OpenStreetMap for address display on map.

8️⃣ System Enhancements

Caching

Use Redis or Memcached to store repeated data (like options list) to improve performance.

Rate Limiting

Implement request limits per user to protect system from abuse.

Localization

Support Arabic and English languages throughout application.

Responsive Design

Interface viewable on all devices (mobile, tablet, desktop).

Security

- Use HTTPS
- CSRF protection
- Input validation
- Sensitive data encryption

Performance

- Database indexing
- Query optimization
- Eager Loading to avoid N+1 queries
- Image and file compression

9️⃣ Expected Deliverables

Students must submit following outputs:

1. **Database** with all tables and relationships mentioned.
2. **Complete ERD (Entity-Relationship Diagram)** showing relationships between all entities.
3. **Integrated Admin and Super Admin Dashboard** with user-friendly interface.
4. **User/Guest Interface** with professional responsive design.
5. **Complete Roles & Permissions System** implemented using Spatie.
6. **Comprehensive RESTful API** with all mentioned endpoints.
7. **Clean and documented code** following Laravel standards and best practices.
8. **Optional:** Advanced features like Excel import/export, Payment Integration, Cloud Uploads.

Good luck with your project!