Benazir Bhutto Shaheed University Liyari Karachi

Royal Aurum E-Commerce Platform

Complete Documentation

Batch 14th BSCS 6th A

Project Name	Royal Aurum
Project Type	Full-Stack E-Commerce Application
Domain	Luxury Jewelry
Timeline	4 Weeks
Status	In Development (MVP Phase)

Submitted To: Miss Ambreen

Table of Contents

Executive Summary

Royal Aurum is a modern, full-stack e-commerce platform designed to provide a seamless shopping experience for luxury jewelry products. The platform features a responsive React-based frontend with an integrated admin panel for product management, backed by a MySQL database with PHP integration for server-side operations.

The project was completed over a 4-week sprint with the following allocation:
▪ Week 1 (25%): Data research, data collection, and CSV conversion
▪ Weeks 2–3 (50%): Frontend development and UI/UX implementation
▪ Week 4 (25%): Backend development and database integration.

The platform is currently in MVP (Minimum Viable Product) status with core e-commerce functionality implemented. Login page functionality is scheduled for the next development phase. Payment integration has been successfully completed.

Project Overview

Project Objectives

1. Create a user-friendly, responsive e-commerce platform for luxury jewelry
2. Implement a comprehensive product management system
3. Develop an intuitive shopping cart and checkout experience
4. Build an admin dashboard for inventory management
5. Establish a scalable database architecture for future growth

Product Categories

The platform currently supports the following jewelry categories:

- Rings
- Necklaces
- Bracelets
- Bangles
- Earrings

Target Users

- End Users: Customers seeking luxury jewelry products
- Admin Users: Store administrators and product managers

- Support Users: Customer service representatives (infrastructure ready)

Data Research Phase

This section describes the work performed by the data research team during the project.

Scope of Work

The data research team was responsible for preparing foundational product data for the system. This work was completed before development started to ensure smooth frontend and backend integration.

Tasks Performed

The following tasks were carried out by the data research team:

- Data collection from multiple sources
- Data cleaning and formatting
- Conversion of raw data into structured CSV files
- Product naming standardization
- Product description writing
- Pricing information preparation
- Verification of data accuracy
- Organization of product data by category

Research Output

The final output of this phase included:

- Clean and formatted CSV files
- Structured product names
- Detailed product descriptions
- Verified pricing details

Project Status

Research Activity	Status
Data Collection	Completed
CSV File Conversion	Completed
Product Description Writing	Completed
Pricing Data Preparation	Completed

Data Verification	Completed

This data research phase helped ensure accuracy, consistency, and quality of product information used in the platform.

Technology Stack

Frontend

Framework & Build Tools:

- React 18.x - UI component library
- Vite - Modern build tool and development server
- TypeScript - Type-safe JavaScript development
- shadcn-ui - High-quality React component library
- Tailwind CSS - Utility-first CSS framework

Key Dependencies:

- React Router - Client-side routing and navigation
- State Management - React Hooks (useState, useContext)
- HTTP Client - Fetch API / Axios (for backend communication)
- Icons & UI - shadcn-ui components and Lucide icons

Development Environment:

- Node.js - JavaScript runtime
- npm - Package management
- Git - Version control

Backend

Server & Language:

- PHP 7.4+ - Server-side scripting language
- Local Development Server - PHP built-in server (development only)

Database:

- MySQL 8.0+ - Relational database management system
- phpMyAdmin - Database administration tool (local development)

Tools & Utilities:

- Composer - PHP dependency management (if applicable)
- cURL - HTTP client for API requests

Hosting & Deployment

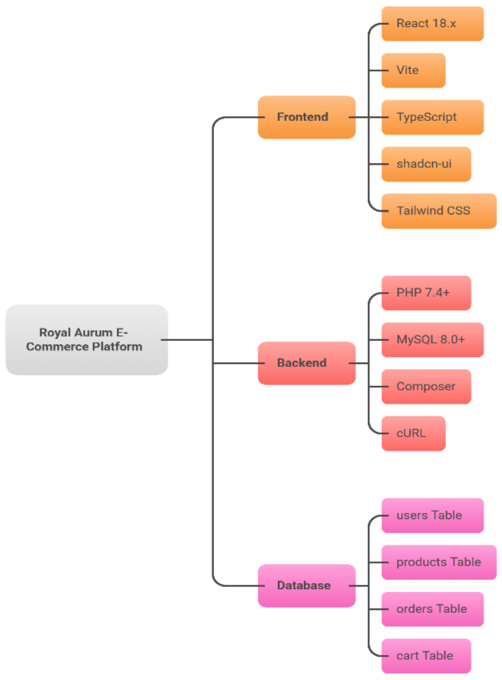
Current Status: Local development environment
Future Deployment:

- Web hosting platform (to be selected and purchased)
- Apache/Nginx web server
- MySQL database hosting
- SSL/TLS certificate for security

Development Environment Setup

Node.js: v16.x or higher
npm: v8.x or higher
PHP: v7.4 or higher
MySQL: v8.0 or higher
Git: v2.x or higher

Architecture Overview



System Architecture Diagram

```
┌──────────────────────────────────────────────────────────────
┐
│            CLIENT LAYER                 │
│ ┌────────────────────────────────────────────────────────┐  │
│ │ React Frontend (SPA)                    │  │
│ │  ├── Home Page                         │  │
│ │  ├── Categories / Products Listing          │  │
│ │  ├── Search Module                      │  │
│ │  ├── Shopping Cart                     │  │
│ │  ├── Login / Signup                    │  │
│ │  └── Admin Dashboard                     │  │
│ │     ├── Add Products                   │  │
│ │     ├── Update Products                  │  │
│ │     ├── Delete Products                 │  │
│ │     └── View Products Overview              │  │
│ └────────────────────────────────────────────────────────┘  │
└──────────────────────────────────────────────────────────────
┘
            ↓ HTTP/REST API Calls
┌──────────────────────────────────────────────────────────────
┐
│           API LAYER (Middleware)            │
│ ┌────────────────────────────────────────────────────────┐  │
│ │ PHP Backend Services                      │  │
│ │  ├── Product Management APIs                  │  │
│ │  ├── User Authentication APIs (In Progress)         │  │
│ │  ├── Cart Management APIs                   │  │
│ │  ├── Order Processing APIs                  │  │
│ │  └── Payment Gateway Integration          │  │
│ └────────────────────────────────────────────────────────┘  │
└──────────────────────────────────────────────────────────────
┘
            ↓ Database Queries
┌──────────────────────────────────────────────────────────────
┐
│           DATA LAYER                    │
│ ┌────────────────────────────────────────────────────────┐  │
│ │ MySQL Database                        │  │
│ │  ├── users Table                     │  │
│ │  ├── products Table                    │  │
│ │  ├── orders Table                     │  │
│ │  └── cart Table                     │  │
│ └────────────────────────────────────────────────────────┘  │
└──────────────────────────────────────────────────────────────
┘
```

High-Level Data Flow

1. User visits the platform → React frontend loads from Vite server
2. User browses products → Frontend fetches product data from PHP backend

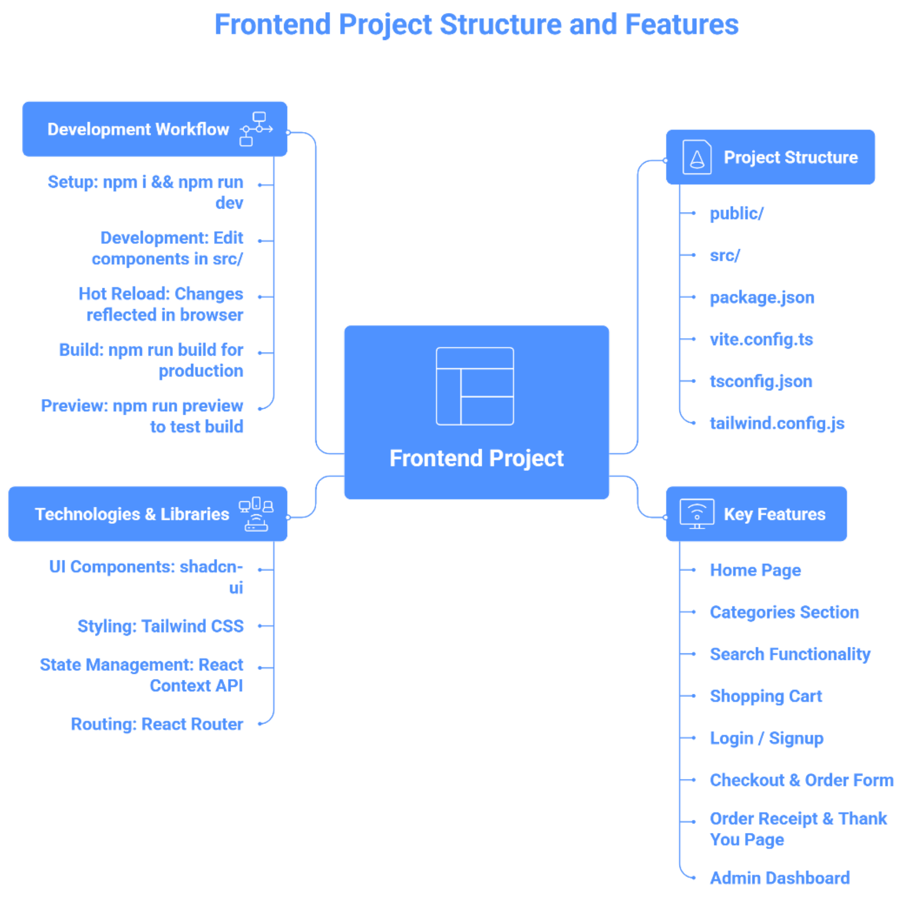
3. User searches for products → Search query sent to PHP API
4. User adds to cart → Cart data stored locally (client-side) or synced to database
5. User proceeds to checkout → Order form displayed, data prepared for submission
6. Admin manages products → Admin panel calls PHP APIs for CRUD operations
7. Database updates → Changes reflected in real-time on the dashboard

---

Frontend Implementation

Project Structure

```
frontend/
├── public/
│   └── [Static assets and favicon]
├── src/
│   ├── components/
│   │   ├── Home/
│   │   │   ├── Hero.tsx
│   │   │   ├── FeaturedProducts.tsx
│   │   │   └── ...
│   │   ├── Categories/
│   │   │   ├── ProductListing.tsx
│   │   │   ├── ProductCard.tsx
│   │   │   ├── FilterPanel.tsx
│   │   │   └── ...
│   │   ├── Search/
│   │   │   ├── SearchBar.tsx
│   │   │   └── SearchResults.tsx
│   │   ├── Cart/
│   │   │   ├── CartPage.tsx
│   │   │   ├── CartItem.tsx
│   │   │   └── CartSummary.tsx
│   │   ├── Auth/
│   │   │   ├── Login.tsx
│   │   │   ├── Signup.tsx
│   │   │   └── AuthForm.tsx
│   │   ├── Admin/
│   │   │   ├── Dashboard.tsx
│   │   │   ├── ProductManagement.tsx
│   │   │   ├── AddProduct.tsx
│   │   │   ├── EditProduct.tsx
│   │   │   ├── DeleteProduct.tsx
│   │   │   └── ProductOverview.tsx
│   │   ├── Common/
│   │   │   ├── Header.tsx
│   │   │   ├── Footer.tsx
│   │   │   ├── Navigation.tsx
│   │   │   └── Sidebar.tsx
│   │   │
│   │   └── Shared/
│   │       ├── Button.tsx
│   │       ├── Modal.tsx
│   │       ├── Card.tsx
│   │       └── ...
│   ├── pages/
│   │   ├── Home.tsx
│   │   ├── Categories.tsx
│   │   ├── Search.tsx
│   │   ├── Cart.tsx
│   │   ├── Login.tsx
│   │   ├── Admin.tsx
│   │   ├── OrderReceipt.tsx
│   │   └── ThankYou.tsx
│   ├── hooks/
│   │   ├── useCart.ts
│   │   ├── useProduct.ts
│   │   ├── useSearch.ts
│   │   └── useAuth.ts
│   ├── context/
│   │   ├── CartContext.tsx
│   │   ├── UserContext.tsx
│   │   └── ProductContext.tsx
│   ├── services/
│   │   ├── api.ts
│   │   ├── productService.ts
│   │   ├── cartService.ts
│   │   ├── orderService.ts
│   │   └── authService.ts
│   ├── types/
│   │   ├── product.types.ts
│   │   ├── user.types.ts
│   │   ├── order.types.ts
│   │   └── cart.types.ts
│   ├── styles/
│   │   ├── globals.css
│   │   ├── tailwind.config.js
│   │   └── ...
│   ├── App.tsx
│   └── main.tsx
├── package.json
├── vite.config.ts
├── tsconfig.json
└── tailwind.config.js
```



Key Frontend Features

1. Home Page

- Hero section with promotional banner
- Featured products showcase
- Category previews
- Call-to-action buttons
- Responsive design for mobile, tablet, and desktop

2. Categories Section

- Grid layout displaying all products
- Product cards with images, names, prices, and ratings
- Filter by category (Rings, Necklaces, Bracelets, Bangles, Earrings)
- Sort options (price, rating, newest)
- Pagination for large product lists
- Quick add-to-cart functionality

3. Search Functionality

- Real-time search bar on header/navigation
- Search results displayed dynamically
- Filter search results by price range, category, rating
- "No results" handling with suggestions
- Search history (optional enhancement)

4. Shopping Cart

- Display all items added by user
- Show product image, name, price, quantity
- Update quantity with + and - buttons
- Remove item functionality
- Calculate subtotal, tax, and total
- "Continue Shopping" and "Place Order" buttons
- Persistent cart (local storage or database sync)

5. Login / Signup

- Login form with email and password fields
- Signup form with name, email, password, and confirmation
- Form validation with error messages
- "Remember Me" functionality (optional)
- Password reset link (future enhancement)
- Social login integration (future enhancement)

6. Checkout & Order Form

- Multi-step checkout process (optional)
- Customer information form:
    - Full Name
    - Email Address
    - Phone Number
    - Delivery Address
    - Postal Code
    - City
    - State
    - Country
- Payment details form (placeholder for integration)
- Order review before final submission
- Terms and conditions checkbox

7. Order Receipt & Thank You Page

- Order confirmation with order ID
- Thank you message
- Order summary table
- Estimated delivery date
- Download receipt as PDF (future enhancement)
- Order tracking link (future enhancement)

8. Admin Dashboard

- Product overview with statistics
- Add New Product form:
    - Product name, description
    - Price, SKU
    - Category selection
    - Inventory quantity
    - Certificate details (for jewelry)
    - Product images upload
    - Status selection
- Edit Product functionality with pre-filled data
- Delete Product with confirmation dialog
- View all products in table format
- Product search and filter
- Bulk actions (future enhancement)

Frontend Technologies & Libraries

UI Components: shadcn-ui provides pre-built, accessible components

- Buttons, Cards, Forms, Modals, Dropdowns, etc.

Styling: Tailwind CSS utility classes for rapid development

- Responsive design utilities
- Dark mode support (optional)
- Custom color schemes for jewelry brand

State Management: React Context API with Hooks

- CartContext for shopping cart state
- UserContext for authentication state
- ProductContext for product data
- Custom hooks for business logic

Routing: React Router for single-page navigation

- /home - Home page
- /categories - Products listing
- /search - Search results
- /cart - Shopping cart
- /login - Login page
- /signup - Signup page
- /admin - Admin dashboard
- /order-receipt - Order confirmation
- /thank-you - Thank you page

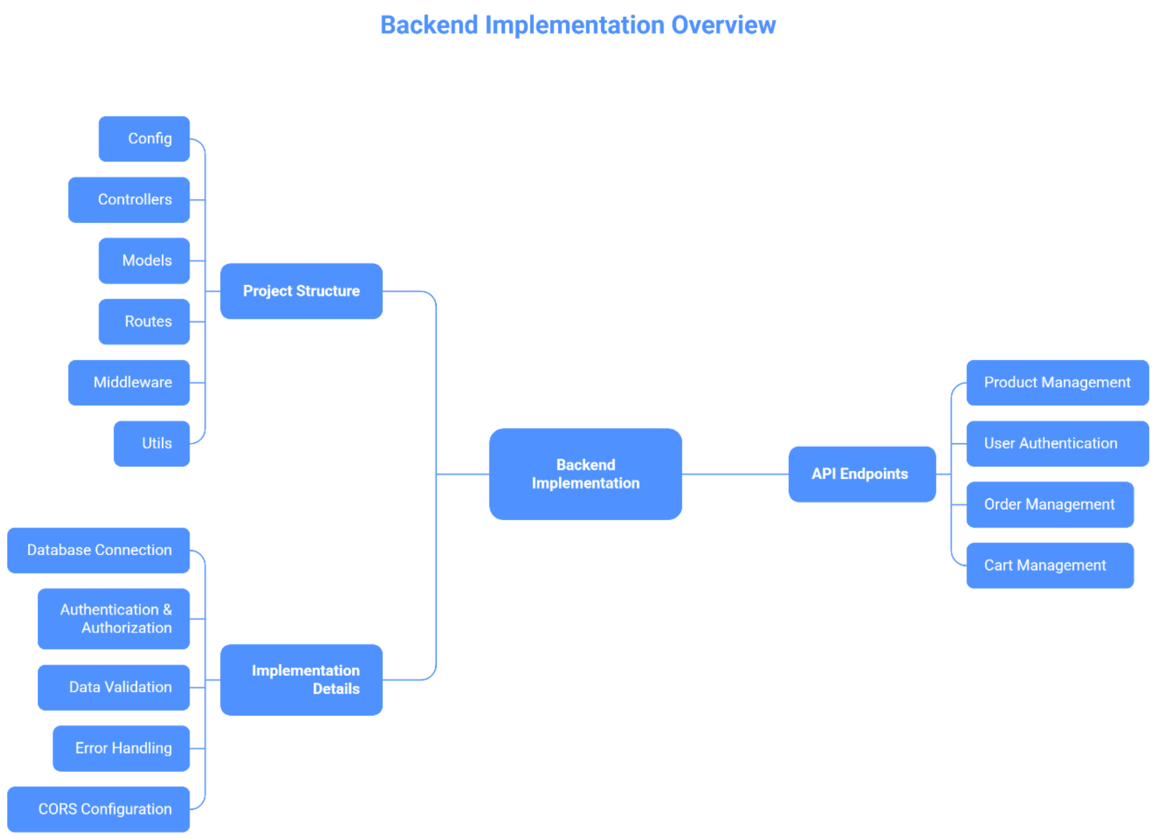
Frontend Development Workflow

1. Setup: npm i && npm run dev
2. Development: Edit components in src/ folder
3. Hot Reload: Changes automatically reflected in browser
4. Build: npm run build for production
5. Preview: npm run preview to test production build locally

Backend Implementation

Project Structure

```
backend/
├── config/
│   ├── database.php
│   └── constants.php
├── controllers/
│   ├── ProductController.php
│   ├── UserController.php
│   ├── OrderController.php
│   ├── CartController.php
│   └── AuthController.php
├── models/
│   ├── Product.php
│   ├── User.php
│   ├── Order.php
│   ├── Cart.php
│   └── Database.php
├── routes/
│   └── api.php
├── middleware/
│   ├── Authentication.php
│   ├── Validation.php
│   └── CORSHandler.php
├── utils/
│   ├── Response.php
│   ├── Validator.php
│   └── Logger.php
├── index.php
├── .htaccess
└── [Other configuration files]
```



Backend API Endpoints

Product Management APIs

GET /api/products

- Retrieve all products
- Query parameters: page, limit, category, sort, search
- Response: Array of product objects

GET /api/products/{id}

- Retrieve single product details
- Response: Product object with full details

POST /api/products (Admin Only)

- Create new product
- Required fields: name, price, category, sku
- Response: Created product with ID

PUT /api/products/{id} (Admin Only)

- Update existing product
- Required fields: id + fields to update
- Response: Updated product object

DELETE /api/products/{id} (Admin Only)

- Delete product by ID
- Response: Deletion confirmation

User Authentication APIs (In Progress)

POST /api/auth/signup

- User registration
- Required fields: full_name, email, password, phone
- Response: User object + auth token

POST /api/auth/login

- User login
- Required fields: email, password
- Response: User object + auth token

POST /api/auth/logout

- User logout
- Response: Success message

GET /api/users/{id}

- Get user profile information
- Response: User object

Order Management APIs

POST /api/orders

- Create new order
- Required fields: user_id, items_json, total_amount, shipping_address, payment_method
- Response: Order confirmation with order ID

GET /api/orders/{id}

- Retrieve order details
- Response: Complete order information

GET /api/orders/user/{user_id}

- Get all orders for user
- Response: Array of user's orders

PUT /api/orders/{id}

- Update order status (admin)
- Response: Updated order

Cart Management APIs

GET /api/cart/{user_id}

- Retrieve user's cart
- Response: Cart object with items

POST /api/cart/add

- Add product to cart
- Required fields: user_id, product_id, quantity
- Response: Updated cart

PUT /api/cart/update

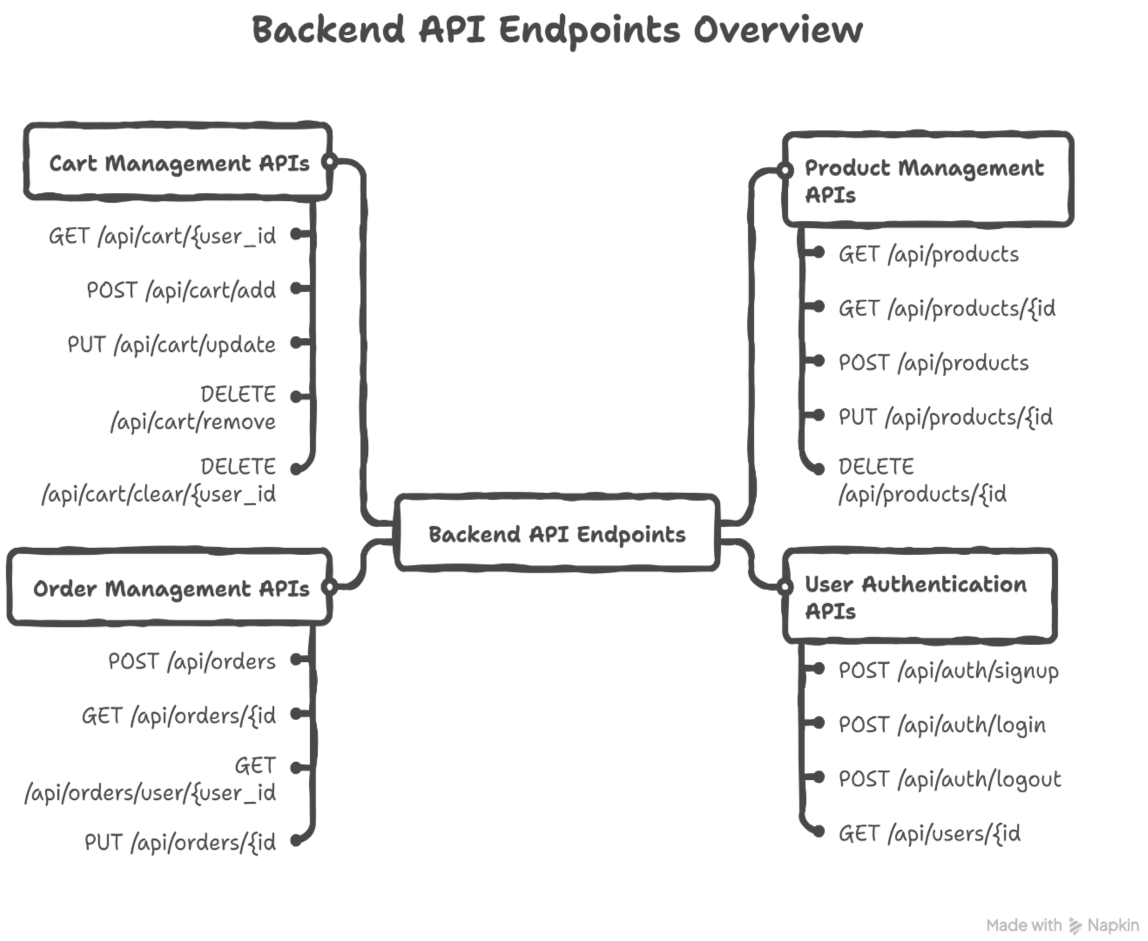
- Update item quantity in cart
- Required fields: user_id, product_id, quantity
- Response: Updated cart

DELETE /api/cart/remove

- Remove item from cart
- Required fields: user_id, product_id
- Response: Updated cart

DELETE /api/cart/clear/{user_id}

- Clear entire cart
- Response: Confirmation message



Backend Implementation Details

Database Connection (database.php)

- MySQLi or PDO connection
- Environment-based configuration
- Connection pooling for performance
- Error handling and logging

Authentication & Authorization

- Role-based access control (customer, admin, support)

- Bearer token authentication (to be implemented)
- Password hashing using bcrypt
- Session management (to be implemented)

Data Validation

- Input sanitization to prevent SQL injection
- Type checking for all inputs
- Email validation
- Phone number format validation
- Price and quantity validation

Error Handling

- HTTP status codes (200, 400, 401, 403, 404, 500)
- JSON error responses with messages
- Logging of all errors
- User-friendly error messages

CORS Configuration

- Allow requests from frontend domain
- Handle preflight requests
- Secure cross-origin communication

Backend Development Status

Completed:

- Database schema and table creation
- Product management CRUD operations
- Order creation and management
- Cart operations
- Database integration with PHP
- API endpoint structure
- Payment gateway integration

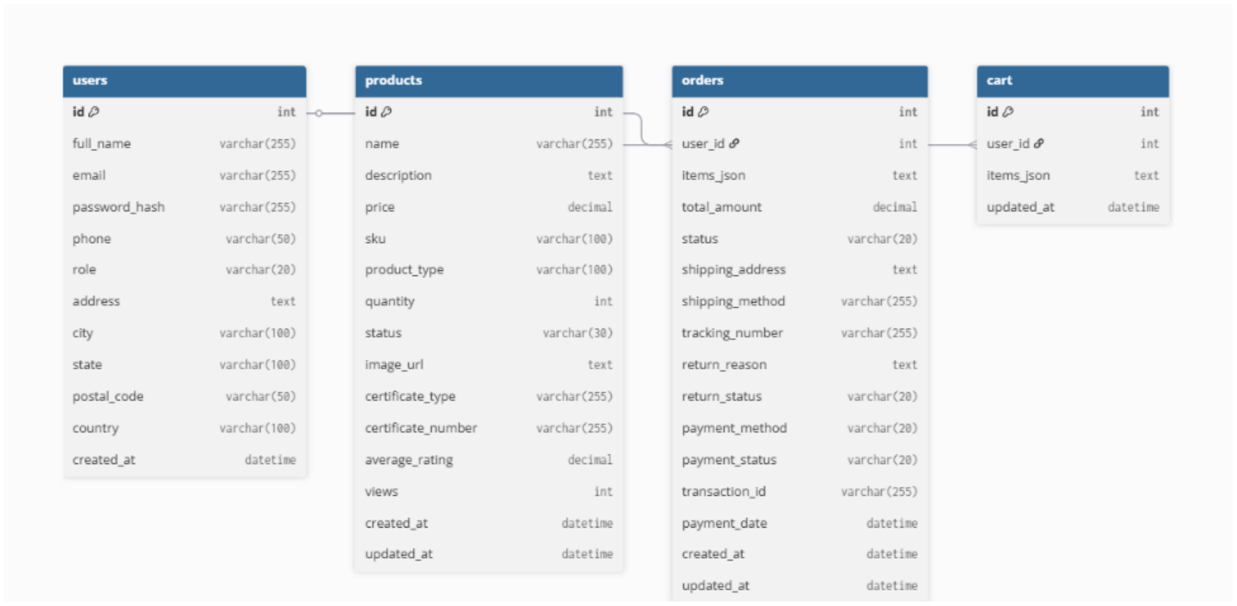
In Progress:

- Authentication and login functionality
- Session management
- Token-based authentication
- User registration flow

Not Yet Started:

- Email notifications
- Shipping integration
- Advanced error handling

Database Schema



Complete Database Design

1. Users Table

Purpose: Stores all customer and account information

Column	Type	Constraints	Description
id	INT	PK, AI	Unique user identifier
full_name	VARCHAR(255)	NOT NULL	User's full name
email	VARCHAR(255)	UNIQUE, NOT NULL	Login email address
password_hash	VARCHAR(255)	NOT NULL	Encrypted password
phone	VARCHAR(50)		Contact telephone number
role	ENUM	DEFAULT 'customer'	User type (customer/admin/support)

address	TEXT		Full residential address
city	VARCHAR(100)		City of residence
state	VARCHAR(100)		State/Province
postal_code	VARCHAR(50)		ZIP/Postal code
country	VARCHAR(100)		Country
created_at	DATETIME	DEFAULT CURRENT_TIMESTAMP	Account creation timestamp

Relationships:

- One user has many orders
- One user has one cart

2. Products Table

Purpose: Stores all product information, inventory, media, and certification data

Column	Type	Constraints	Description
id	INT	PK, AI	Unique product identifier
name	VARCHAR(255)	NOT NULL	Product name
description	TEXT		Detailed product description
price	DECIMAL(10,2)	NOT NULL	Product price in currency
sku	VARCHAR(100)		Stock keeping unit identifier
product_type	VARCHAR(100)		Category (necklace, ring, earrings, etc.)
quantity	INT	DEFAULT 0	Current inventory stock level
status	ENUM	DEFAULT 'in-stock'	Status (in-stock/low-stock/out-of-stock)
image_url	TEXT		URL to main product image
certificate_type	VARCHAR(255)		Jewelry certification type (GIA, etc.)
certificate_number	VARCHAR(255)		Certification reference number
average_rating	DECIMAL(3,2)	DEFAULT 0	Average customer rating (0-5)
views	INT	DEFAULT 0	Total product page views

created_at	DATETIME	DEFAULT CURRENT_TIMESTAMP	Product creation timestamp
updated_at	DATETIME	DEFAULT CURRENT_TIMESTAMP ON UPDATE	Last modification timestamp

Relationships:

- Many products referenced in orders (via JSON)
- Many products referenced in carts (via JSON)

3. Orders Table

Purpose: Manages complete order lifecycle from creation through returns

Column	Type	Constraints	Description
id	INT	PK, AI	Unique order identifier
user_id	INT	FK → users(id)	Customer who placed order
items_json	JSON	NOT NULL	Array of purchased items with product_id, qty, price
total_amount	DECIMAL(10,2)	NOT NULL	Total order value
status	ENUM	DEFAULT 'pending'	Order state (pending/paid/shipped/delivered/cancelled/returned)
shipping_address	TEXT		Delivery address for order
shipping_method	VARCHAR(255)		Shipping type (standard/express/etc)
tracking_number	VARCHAR(255)		Courier tracking number
return_reason	TEXT		Reason for return request
return_status	ENUM		Return progress (requested/approved/rejected/refunded)
payment_method	ENUM	NOT NULL	Payment type (card/cod/paypal/giftcard)
payment_status	ENUM	DEFAULT 'pending'	Payment state (pending/completed/failed/refunded)
transaction_id	VARCHAR(255)		Payment gateway transaction ID
payment_date	DATETIME		When payment was processed

created_at	DATETIME	DEFAULT CURRENT_TIMESTAMP	Order creation timestamp
updated_at	DATETIME	DEFAULT CURRENT_TIMESTAMP ON UPDATE	Last update timestamp

Relationships:

- Many orders belong to one user (user_id FK)
- Each order contains multiple products (items_json)

JSON Structure Example:

```
[
  {
    "product_id": 5,
    "name": "Gold Diamond Ring",
    "quantity": 1,
    "price": 5999.99,
    "sku": "RING-001"
  },
  {
    "product_id": 12,
    "name": "Pearl Necklace",
    "quantity": 2,
    "price": 1299.99,
    "sku": "NECK-002"
  }
]
```

4. Cart Table

Purpose: Stores temporary shopping cart items before checkout

Column	Type	Constraints	Description
id	INT	PK, AI	Unique cart identifier
user_id	INT	UNIQUE, FK → users(id)	User who owns cart
items_json	JSON		Array of cart items with product_id, quantity
updated_at	DATETIME	DEFAULT CURRENT_TIMESTAMP ON UPDATE	Last modification time

Relationships:

- One cart belongs to one user (one-to-one)
- Each cart contains multiple products (items_json)

JSON Structure Example:

```
[
  {
    "product_id": 5,
    "quantity": 1,
    "name": "Gold Diamond Ring",
    "price": 5999.99
  },
  {
    "product_id": 12,
    "quantity": 1,
    "name": "Pearl Necklace",
    "price": 1299.99
  }
]
```

Entity-Relationship Diagram (Conceptual)

```
┌─────────────────────────────────┐
│    USERS         │
├─────────────────────────────────┤
│ id (PK)          │
│ full_name        │
│ email (UNIQUE)   │
│ password_hash    │
│ phone            │
│ role             │
│ address          │
│ city             │
│ state            │
│ postal_code      │
│ country          │
│ created_at       │
└─────────────────────────────────┘
      ▲ 1
      │
      │ N
      │
   ┌─────────────────────────────────┐
   │    ORDERS        │
   ├─────────────────────────────────┤
   │ id (PK)          │
   │ user_id (FK)     │ ◄─────────┐
   │ items_json       │    │
   │ total_amount     │    │
   │ status           │    │
   │ shipping_address │    │
   │ shipping_method  │    │
   │ tracking_number     │      │
   │ return_reason       │      │
   │ return_status       │      │
   │ payment_method      │      │
   │ payment_status      │      │
   │ transaction_id      │      │
   │ payment_date        │      │
   │ created_at          │      │
   │ updated_at          │      │
   └───────────────────────────┘      │
                 │
             References
             PRODUCTS
             (via JSON)

   ┌───────────────────────────┐
   │   PRODUCTS      │
   ├───────────────────────────┤
   │id (PK)          │
   │name             │
   │description      │
   │price            │
   │sku              │
   │product_type     │
   │quantity         │
   │status           │
   │image_url        │
   │certificate_type │
   │certificate_number │
   │average_rating   │
   │views            │
   │created_at       │
   │updated_at       │
   └───────────────────────────┘
       ▲ 1
      │
      │ M (via JSON)
      │
   ┌─────────────────────┐
   │   CART     │
   ├─────────────────────┤
   │ id (PK)    │
   │ user_id (FK) │
   │ items_json │
   │ updated_at │
   └─────────────────────┘
      ▲ 1
      │
      │ 1
      │
   └─ USERS (id)
```

Database Relationships Summary

Relationship	Type	Description
Users → Orders	1:N	One user has many orders
Users → Cart	1:1	One user has one cart
Orders → Products	M:N	Many orders contain many products (via items_json)
Cart → Products	M:N	Many cart items refer to many products (via items_json)

Features & Functionality

Currently Implemented Features

1. User Interface & Experience

- Responsive design for desktop, tablet, and mobile devices
- Modern UI with Tailwind CSS styling
- shadcn-ui component library for consistency
- Intuitive navigation and user flow
- Category-based product browsing
- Fast page loads with Vite build tool

2. Product Management

- Display all products with images, names, and prices
- Filter products by category
- Sort products by price, rating, newest
- Real-time search functionality
- Product detail pages with specifications
- Jewelry certification display
- Product ratings and reviews display (backend ready)

3. Shopping Cart

- Add products to cart
- Update item quantities
- Remove items from cart
- Calculate subtotal, tax, and total
- Clear entire cart
- Persistent cart (local storage)
- Database sync (in progress)

4. Checkout & Orders

- Multi-field checkout form
- Collect customer information:

        - Name, email, phone
        - Complete address with postal code
        - City, state, country
- Order summary display
- Order confirmation page
- Thank you message
- Order database storage
- Order receipt PDF (planned)
- Order tracking (planned)

5. Admin Panel

- Dashboard with product overview
- Add new products
- Edit existing products
- Delete products
- View all products in table format
- Real-time database updates
- Product search and filtering

6. Home Page

- Hero section with banners
- Featured products showcase
- Category previews
- Call-to-action buttons
- Responsive design

7. About Page

- Company information
- Mission and values
- Contact information

8. Technical Features

- Type-safe development with TypeScript
- Component-based architecture
- API integration ready
- Context-based state management
- Custom hooks for reusable logic
- Error handling and validation

Features In Progress

1. Authentication & Login

- User registration/signup
- User login with credentials
- Password validation and hashing
- Session management
- Remember me functionality
- Role-based access control

Status: Backend infrastructure ready, frontend integration pending

2. Payment Integration

- Secure payment processing
- Multiple payment methods (card, COD, digital wallets)
- Payment status tracking
- Transaction logging

Status: Fully implemented and successfully integrated.

Planned Features (Future Releases)

1. User Accounts & Profiles
    - Account management
    - Order history
    - Saved addresses
    - Wishlist functionality
    - Account preferences
2. Advanced Search & Filtering
    - Multi-criteria filtering
    - Price range slider
    - Rating filter
    - Size/specification filters
    - Search suggestions
3. Product Reviews & Ratings
    - User review submission
    - Photo upload with reviews
    - Rating system (1-5 stars)
    - Helpful votes
    - Review moderation (admin)
4. Inventory Management
    - Stock level alerts
    - Automated reorder points
    - Inventory forecasting
    - Bulk import/export

5. Order Management
    - Order tracking
    - Shipment notifications
    - Return management
    - Refund processing
    - Order history archive
6. Email Notifications
    - Order confirmation email
    - Shipment tracking emails
    - Return status updates
    - Promotional emails
    - Newsletter subscription
7. Analytics & Reporting
    - Sales dashboard
    - Product performance metrics
    - Customer behavior analysis
    - Revenue reports
    - Inventory reports
8. Advanced Features
    - Wishlist and favorites
    - Product recommendations
    - Seasonal collections