Project documentation

Giga Store

presented to:

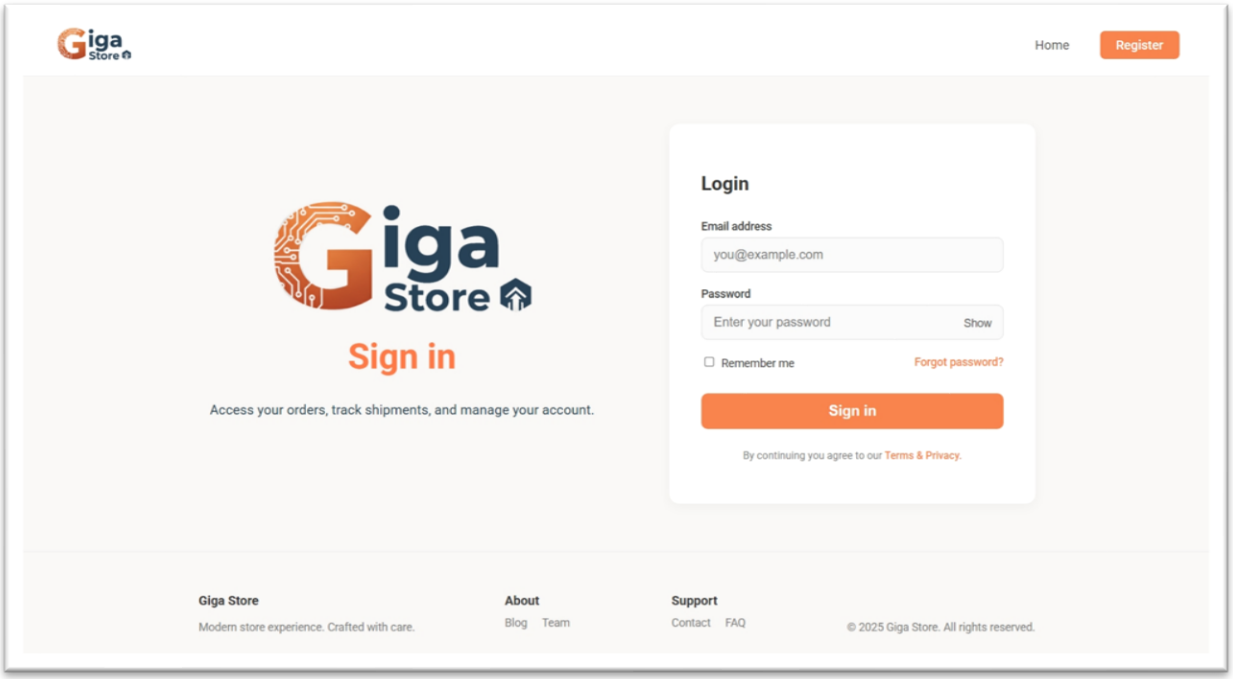
Eng: Abdullah Mohamed

Created by

- احمد محمد ابوبكر عبدالحليم
- احمد محمد احمد عبدالفتاح
- احمد عبدالنبي نصر
- احمد محمد صلاح
- احمد خالد السيد
- احمد محمد زين سعيد

Executive Summary

Giga Store is a full-stack e-commerce web application designed to facilitate the online sale of electronic products. The system provides a seamless shopping experience for customers—ranging from product browsing to secure checkout—while offering a dedicated dashboard for administrators to manage users and monitor system activity.



The primary objective of this project was to build a robust, secure, and scalable application using **Native PHP** and **Vanilla JavaScript**, demonstrating a deep understanding of core web development concepts such as the **MVC Architecture**, **Database Normalization**, and **Asynchronous Data Handling (AJAX)**.

The system features two main interfaces:

1. **Customer Front-Office:** For browsing products, managing a shopping cart, and secure checkout.
2. **Admin Back-Office:** For managing users and monitoring system data.

Technical Stack

Frontend (Client-Side)

- **HTML5 & CSS3:** Responsive design with a custom styling system (Variables for colors/spacing).
- **JavaScript:** Pure JS for dynamic DOM manipulation.
- **LocalStorage:** Client-side state management for the Shopping Cart and Theme preferences (Dark/Light mode).

Backend (Server-Side)

- **Language:** PHP (Hybrid approach: Object-Oriented for Models/Admin & Procedural for simple routing).
- **Architecture:** MVC (Model-View-Controller) Design Pattern.

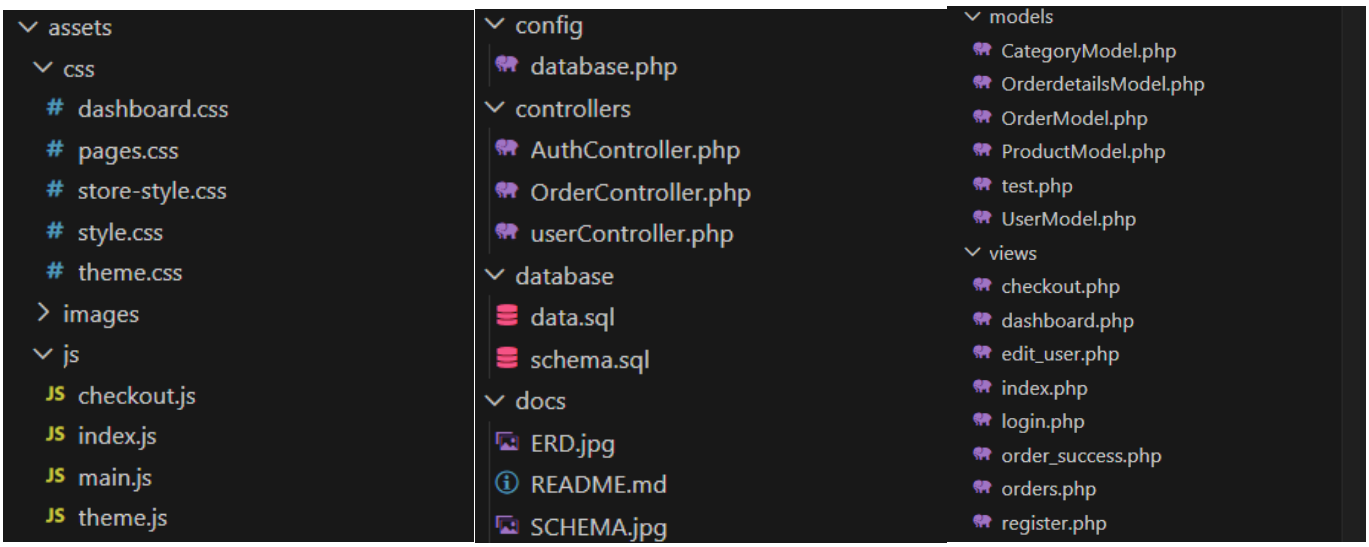
- **Session Management:** Secure handling of user login states **Database**
- **RDBMS:** MySQL.

System Architecture (MVC Pattern)

The project follows a strict **MVC Design Pattern** to ensure separation of concerns. Below is the detailed breakdown of the directory structure based on the project files:

Root Directory (Giga-store)

The root contains the core modules of the application.



- **assets/**: Contains all static resources.
    - **css/**: Modular styling files.
        - store-style.css: Main layout for the storefront.
        - dashboard.css: Specific styles for the admin panel.
        - theme.css: Variables for Dark/Light mode logic.
    - **js/**: Client-side logic.
        - main.js: Handles product filtering, search debouncing, and cart UI updates.
        - checkout.js: Manages the AJAX checkout process and data serialization.
        - theme.js: Handles the toggle state for the User Interface theme.
    - **images/**: Product images and system logos.
- **config/**: System configuration.
    - database.php: Establishes the mysqli connection to the MySQL database. It sets the charset to utf8mb4 and enables strict error reporting.
- **controllers/** (The "Brain"): Handles user requests.

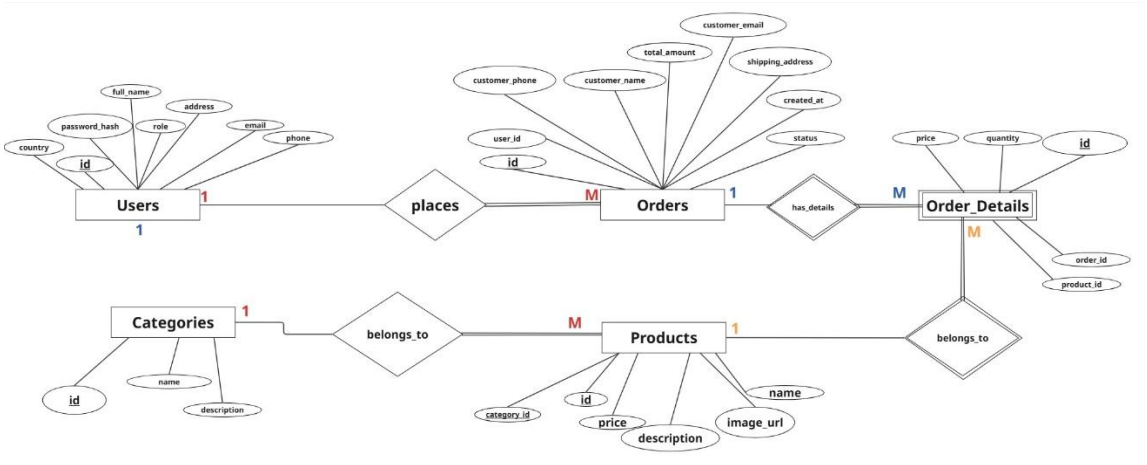
    - AuthController.php: Manages Login, Registration, and Logout logic.
    - OrderController.php: Handles API requests for creating or canceling orders (returns JSON).
    - userController.php: Manages CRUD operations for the Admin Dashboard (Edit/Delete users).
- **models/** (The "Data Layer"): Interacts with the database.
    - UserModel.php: Handles user authentication and data retrieval.
    - ProductModel.php: Fetches product data for the storefront.
    - OrderModel.php: **(Key Component)** Contains complex logic for transaction management (ACID compliance) during order creation.
    - OrderdetailsModel.php: Manages the specific items within an order.
    - CategoryModel.php: Manages product categories.
- **views/** (The "Interface"): The visible pages.
    - index.php: The landing page displaying products.
    - checkout.php: The payment gateway simulation page.
    - dashboard.php: The protected admin panel.
    - orders.php: User's order history.
- **database/**: SQL Scripts.
    - schema.sql: Structure of tables and relationships.
    - data.sql: Seed data for testing.

---

3. Technical Implementation Details

3.1. Database Design (Relational Schema) The database gigastore_db is normalized to the Third Normal Form (3NF).

- **Tables:** Users, Categories, Products, Orders, Order_Details.
- **Relationships:**
    - **One-to-Many:** Categories -> Products.
    - **One-to-Many:** Users -> Orders.
    - **Many-to-Many:** Orders <-> Products (Resolved via Order_Details).
- **Constraints:**
    - ON DELETE RESTRICT on Categories to prevent accidental data loss.
    - ON DELETE CASCADE on Order Details to ensure clean deletions.



Entity Relationship Diagram (ERD) & Database Schema

3.2. Backend Logic (PHP)

- **Session-Based Authentication:** The system uses PHP $_SESSION to track logged-in users and restrict access to pages like dashboard.php.
- **Database Transactions:** In OrderModel.php, the create_order function uses $conn->begin_transaction() to ensure that both the order header and the order items are saved simultaneously. If any part fails, $conn->rollback() is triggered.
- **Object-Oriented Programming (OOP):** Models are implemented as Classes (e.g., class UserModel), promoting code reusability.

3.3. Frontend Logic (JavaScript)

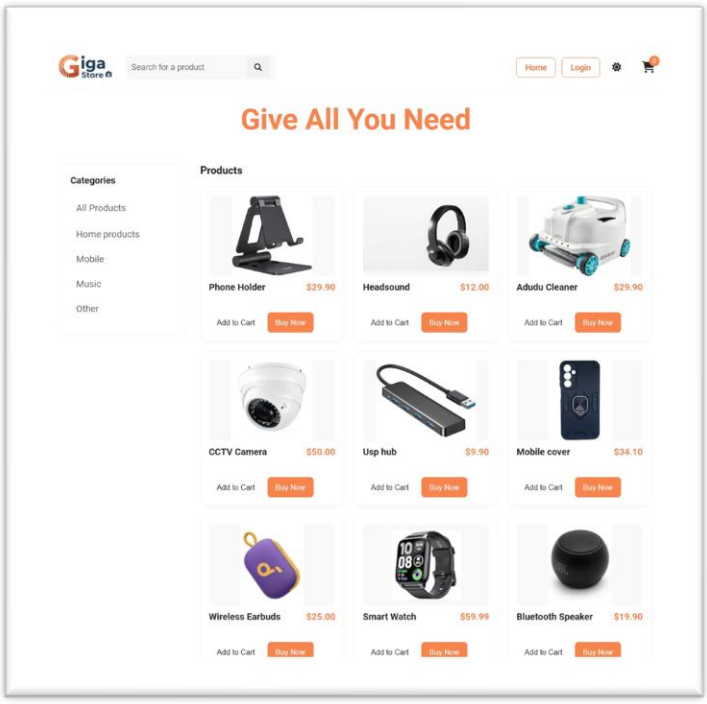
- **Asynchronous Operations:** The checkout process does **not** reload the page. checkout.js collects cart data from localStorage, serializes it into JSON, and sends it via fetch() to OrderController.php.
- **Debounced Search:** The search bar in main.js waits for the user to stop typing (180ms delay) before filtering products, optimizing browser performance.
- **State Persistence:** The Shopping Cart and Theme preference (Dark/Light) are saved in the browser's localStorage, persisting across sessions.

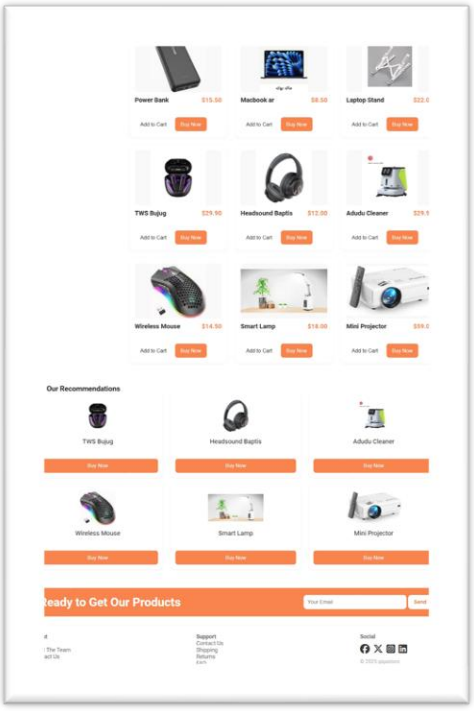
---

4. Features Breakdown

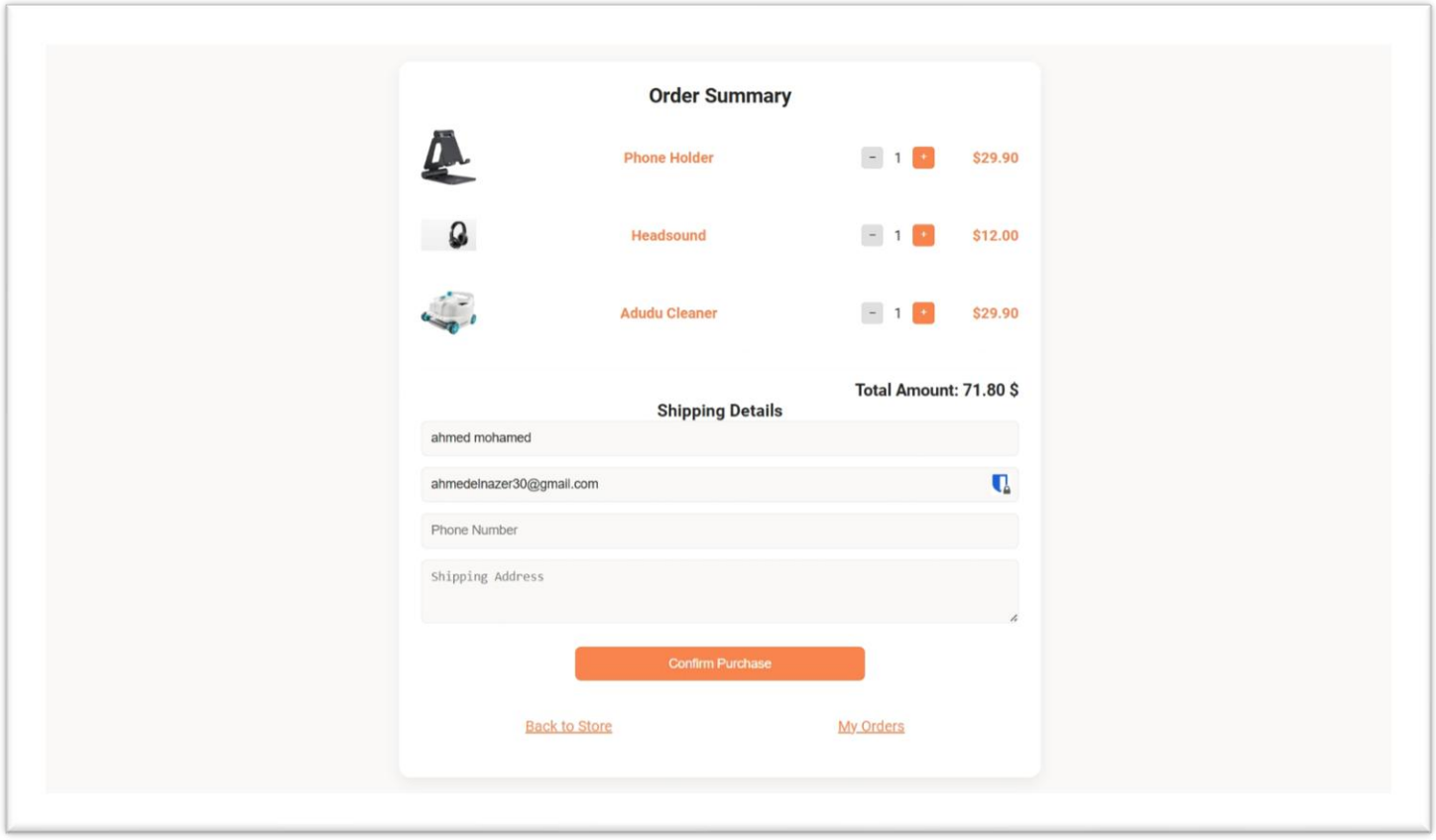
A. User Features

1. **Registration & Login:** Secure account creation with password hashing (bcrypt).
2. **Product Browsing:** Categorized view with a responsive grid layout.





3. **Smart Cart:** Add items, adjust quantities, and view live total calculations without page reloads.

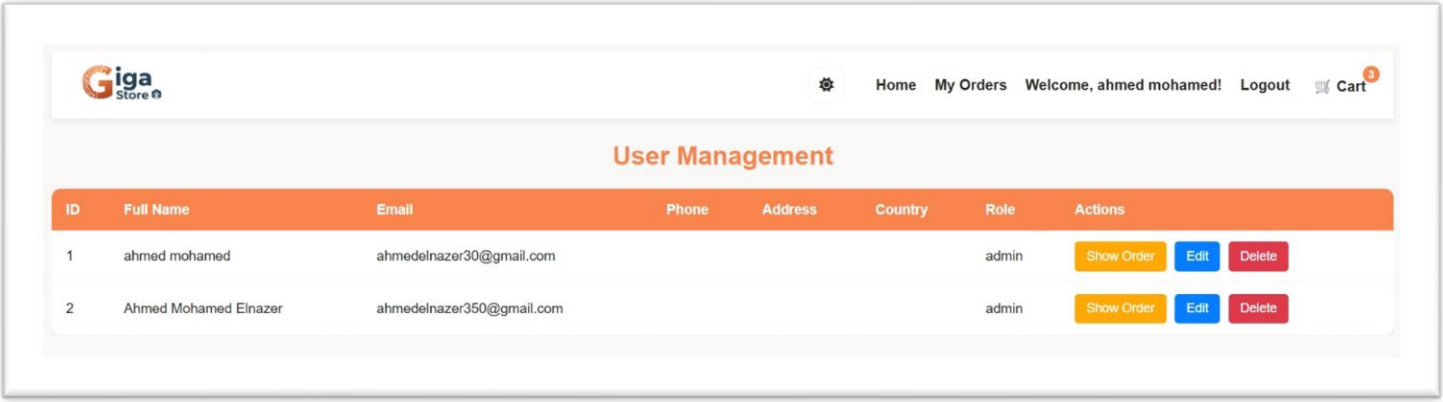


4. **Order Tracking:** Users can view their past orders in orders.php.

5. **Order Cancellation:** Users can cancel "Pending" orders instantly.

B. Admin Features

1. **Dashboard Access:** Protected route (Middleware check for role === 'admin').
2. **User Management:** View all registered users in a tabular format.



3. **Admin Actions:** Ability to Edit user details or Delete users directly from the dashboard.

---

5. Security Measures

- **SQL Injection:** All database queries utilize **Prepared Statements** (e.g., $stmt->bind_param(...)) to sanitize inputs.
- **XSS Protection:** htmlspecialchars() is used when rendering user-generated content (e.g., User Names, Addresses) in Views.
- **Access Control:** Direct access to Controllers or Models via URL is blocked or handled securely; Admin pages check for session roles immediately upon loading.
- **Error Handling:** The database.php config is set to throw Exceptions, which are caught and logged server-side (error_log), preventing sensitive error details from being exposed to the user.

---

6. Setup & Installation

1. **Environment:** Requires XAMPP/WAMP (PHP 7.4+ and MySQL).
2. **Database:** Import database/schema.sql then database/data.sql into phpMyAdmin.
3. **Configuration:** Verify credentials in config/database.php.
4. **Launch:** Access via http://localhost/Giga-store/views/index.php.

---

7. Conclusion

The Giga Store project successfully demonstrates the capability to build a secure, scalable, and user-friendly e-commerce application. By implementing custom MVC logic and handling raw SQL transactions, the project highlights a strong command of fundamental software engineering principles.

Giga Store

THANK YOU