order number: . . . . .

Project report submitted to the

UNIVERSITY OF MOHAMED BOUDIAF – MSILA

FACULTY OF MATHMATICS AND COMPUTER SCIENCE

DEPARTEMENT OF COMPUTER SCIENCE

In partial fulfillment of the requirements for the degree of

Master in **Computer science**

Specialty: Information Systems and Software Engineering

By

Tharrafi, Abdallah

Aziz, Salah Eddine

Title of the project

Development of an Online Store for Product and Order Management

---

Under the supervision of

Dr. Loucif Hamza

june, 2025

Dedications

"To my mother, father, siblings, and friends, you have been my pillars of support throughout my education journey. I must also acknowledge my professors, whose invaluable guidance and knowledge were crucial. I dedicate my graduation thesis to you, praying that the Almighty may grant you long life and bless you with goodness"

Aziz Salah Eddine

"To my beloved mother and father, Your endless love, unwavering support, and constant prayers have been my greatest strength throughout this journey. Thank you for your sacrifices, your guidance, and your faith in me. This achievement is as much yours as it is mine. With all my love and gratitude, I dedicate this work to you"

Tharrfai Abdallah

Acknowledgements

We are deeply grateful to our thesis supervisor, Dr. Loucif Hamza, for his constant support, guidance, and encouragement throughout the development of this project. His insights and dedication played a crucial role in shaping our work and helping us overcome many challenges.

We also extend our sincere thanks to our families and friends for their patience, motivation, and understanding during the demanding periods of this research journey.

Table of Contents

List of figures:

List of tables:

General Introduction

With the rapid advancement of information and communication technologies, e-commerce has become a vital component in modern business practices. It enables individuals and organizations to buy and sell goods and services over the internet with efficiency, flexibility, and global reach. In recent years, The COVID-19 pandemic accelerated the global shift towards digital commerce, making e-commerce not just a convenience but a necessity for many businesses [1].

This project focuses on the design and implementation of a flexible and functional ecommerce platform using the Laravel framework. The system allows users to browse products, add them to a shopping cart, create accounts, and place orders. Additionally, administrators are provided with a dashboard to manage products, users, and order processing. Although payment integration is simulated in this academic version, the system is designed to support real-world expansion in the future.

The methodology followed in this project includes requirement analysis, architectural and database design using UML diagrams, system implementation using modern web technologies, and testing of core functionalities. The solution is based on the MVC architecture, with Laravel at the back-end, Blade for templating, and Bootstrap for responsive UI design.

This report is organized into five main chapters:

- The first chapter presents the theoretical background of e-commerce and its key components.
- The second chapter focuses on analyzing system requirements and design.
- The third chapter introduces the tools and technologies used.
- The fourth chapter presents the implementation details and testing.
- The final chapter evaluates the work, discusses its strengths and limitations, and outlines future improvements.

We hope that this work reflects our technical understanding and practical abilities, and that it can serve as a foundation for future developments or real-world deployment.

CHAPTER 1

Literature Review

1.1 Introduction

Electronic commerce, commonly referred to as e-commerce, has become a central pillar in today's digital economy. It allows businesses and individuals to conduct commercial transactions online, reshaping traditional business models and offering new opportunities for growth and innovation.

Over the past two decades, e-commerce has evolved significantly—moving from basic websites offering limited functionality to complex platforms that integrate inventory management, customer relationship tools, data analytics, and secure payment systems. This shift has been accelerated by advances in internet infrastructure, mobile technologies, and global digital adoption.

For developers, building an e-commerce platform presents both technical and strategic challenges. These include creating responsive and intuitive user interfaces, maintaining secure transactions, optimizing performance for high traffic, and ensuring scalability. To address these requirements, modern frameworks such as Laravel offer developers a robust environment to build modular and secure web applications efficiently.

This chapter presents the conceptual background necessary to understand the nature of e-commerce systems. It begins with a definition of e-commerce, explores its main types, then discusses the general components of any e-commerce platform, the main development challenges, and concludes with a comparison between Laravel and other development frameworks.

1.2 Definition of E-Commerce

E-commerce refers to the process of buying and selling goods and services using electronic networks, primarily the Internet. It has evolved from simple online catalogs to complex platforms supporting analytics, secure payments, and customer engagement [2].

E-commerce is not limited to retail transactions between businesses and consumers. It also includes business-to-business (B2B) exchanges, government procurement systems, and peer-to-peer marketplaces. The process typically involves multiple stages, such as product selection, order placement, payment, and delivery.

Modern e-commerce platforms combine front-end interfaces that users interact with, and back-end systems that manage products, orders, payments, and logistics. They may also integrate analytics tools to track user behavior and improve sales strategies.

1.3 Types of E-Commerce Systems

E-commerce systems are categorized based on the nature of interactions between the parties involved in the transaction. Each type serves a specific purpose and target audience. The major types of e-commerce are:

- **Business-to-Consumer(B2C):**

This is the most common model, where businesses sell products or services directly to individual consumers through online platforms. Examples include Amazon, eBay, and local marketplaces like Jumia or Noon. These platforms focus on user experience, ease of browsing, and efficient payment systems.

- **Business-to-Business(B2B):**

In this model, transactions occur between companies. A manufacturer may sell goods to a wholesaler, or a wholesaler may sell to a retailer. B2B platforms require advanced features such as bulk ordering, invoicing, and multi-user company accounts.

- **Consumer-to-Consumer(C2C):**

C2C platforms allow individuals to sell goods or offer services to other individuals. These are typically marketplace-based platforms such as OLX, Facebook Marketplace, or eBay. They include features like seller ratings, direct messaging, and secure payment handling.

- **Consumer-to-Business(C2B):**

This less common model involves individuals offering products or services to companies. An example is a freelancer providing graphic design or writing services to a business through platforms like Fiverr or Upwork.

- **Government-to-Consumer/Business(G2C/G2B):**

Governments offer services and transactions to citizens (G2C) and businesses (G2B) through e-commerce systems. These include tax payments, license renewals, online bill payments, and business registration portals.

1.4 General Components of an E-Commerce Platform

A complete e-commerce platform is composed of various components that ensure smooth operation for both users and administrators. These components can be divided into front-end and back-end layers.

1.4.1 User Interface (Front-End)

The user interface includes all the visible parts of the system that users interact with while browsing the store or placing orders.

1.5 Challenges in Developing E-Commerce Platforms

A smooth user experience is a key success factor in e-commerce platforms, influencing customer retention, conversion rates, and brand loyalty [3].

1.5.1 Security

E-commerce platforms handle sensitive user information, including login credentials, personal details, and payment data. Security is therefore a top priority. Developers must:

- Server response time
- Page load speed
- Database optimization
- Caching strategies and asset compression

These measures help maintain performance during high-traffic periods like sales events.

1.5.2 Performance

Performance affects user satisfaction and retention. A slow-loading platform may lead users to abandon the website. Developers must consider:

- Clear navigation structure
- Responsive design for mobile devices
- Minimal click steps to complete purchases
- Feedback for user actions (e.g., confirmation messages)

Improving UX often has a direct positive impact on conversion rates and customer satisfaction.

1.6 Comparison Between Laravel and Other Frameworks

Laravel is widely adopted due to its expressive syntax, built-in tools for routing, authentication, and security, making it ideal for rapid development of modern web applications [4].

Why Laravel is preferred:

- Easier to learn and set up

- Offers pre-built tools for routing, authentication, and templating
- Suitable for rapid development
- More active community and modern practices
- Better documentation and support
- Enhanced security features
- PHP compatibility with hosting services
- Team familiarity with PHP

1.7 Conclusion

This chapter introduced the core theoretical foundations of e-commerce platforms. It began with a definition and classification of e-commerce systems based on user interaction models such as B2C, B2B, C2C, and others. Then, it explored the essential components required to build an online store, such as the user interface, dashboard, and admin panel. We also addressed key challenges like security, performance, and user experience.

These foundational insights provide the context for understanding the system architecture, design process, and development approach that will be presented in the following chapters.

CHAPTER 2

System Analysis and Design

2.1 Introduction

Before building any software system, it is crucial to perform thorough analysis and design to ensure that the final product meets user needs and functions efficiently. This chapter focuses on the planning and structural preparation for the e-commerce platform presented in this project.

The process starts by identifying the **functional and non-functional requirements** of the system, followed by designing the **architecture**, **database structure**, and **UML diagrams** that describe how different parts of the system interact with each other.

By laying this groundwork, the development phase becomes more organized, with clear objectives and a roadmap that minimizes errors and ensures system scalability and maintainability.

2.2 Functional Requirements

Functional requirements define the specific features and behaviors that the system must implement to meet user needs and business objectives. These requirements describe what the system must do in terms of functionality.

For this e-commerce platform, functional requirements include user registration, product browsing, shopping cart management, order placement, and administrative actions such as managing products, categories, and customer orders.

These requirements serve as a foundation for system design, guiding the development team on what functionalities must be implemented.

2.3 Non-Functional Requirements

Non-functional requirements describe how the system performs rather than what it does. These requirements define quality attributes such as performance, security, usability, scalability, and maintainability. While functional requirements define actions, non-functional requirements define constraints and standards.

For this e-commerce platform, the non-functional requirements include:

- **Performance:** Pages should load in less than 2 seconds under normal conditions.
- **Scalability:** The system should support an increasing number of users and products without major performance issues.

- **Security:** User data, especially login credentials and order information, must be securely stored and transmitted using encryption (HTTPS).
- **Usability:** The interface should be user-friendly, with a clean design that works on both desktop and mobile devices.
- **Availability:** The system should maintain high uptime and be available at least 99% of the time.
- **Maintainability:** The code should be modular and documented to allow easy updates and debugging.

These requirements ensure that the system remains reliable, secure, and efficient in real-world usage.

2.4 System Architecture

The MVC architecture separates concerns by organizing code into models, views, and controllers, improving maintainability, testability, and scalability of web applications[5].

In this architecture:

- The **Model** represents the data layer and business logic. It handles database queries, data validation, and relationships between different entities (e.g., products, users, orders).
- The **View** is responsible for displaying data to the user. It consists of the front-end pages (HTML/Blade in Laravel) where users can interact with the application.
- The **Controller** receives user input from the view, processes it, and sends the appropriate commands to the model. It also prepares the response and sends it back to the view.

Laravel follows the MVC pattern natively, which improves project structure and allows developers to work on each layer independently.

APIs (Application Programming Interfaces) are essential components in modern web applications, especially in systems that require interaction between the front-end and back-end in a flexible and asynchronous manner.

In this e-commerce platform, RESTful APIs are used to:

- Fetch product information dynamically
- Add or remove items from the shopping cart without reloading the page

- Submit orders asynchronously

Using APIs allows the system to be more modular and compatible with future enhancements, such as mobile applications or integration with third-party services like payment gateways or delivery APIs.

Laravel supports API development natively, offering features like route grouping, middleware, request validation, and resource controllers that simplify building secure and efficient APIs.

2.5 Database Design

A well-structured database is fundamental to the success of any e-commerce platform. It ensures the consistency, integrity, and accessibility of all data related to users, products, orders, and other entities.

For this project, the database was designed using **MySQL**, and follows a relational model. The design includes multiple interconnected tables that reflect the logical relationships between key components of the system.

The main tables include:

- **Users:** Stores user information such as name, email, role (admin/user), and authentication data.
- **Products:** Contains product details like title, description, price, stock quantity, and image URL.
- **Categories:** Used to group products into logical sections for easier navigation.
- **Orders:** Keeps records of each purchase, linking users with their orders and timestamps.
- **Order Items:** Stores the individual items within each order, with references to products and quantities.
- **Cart:** Temporarily holds selected items before the user places an order.

Relationships such as **one-to-many** and **many-to-many** are used throughout the schema, especially between users and orders, products and categories, and orders and items.

2.6 UML Diagrams

Unified Modeling Language (UML) diagrams are a standard way to visually represent the structure and behavior of a system. They help in planning, understanding, and communicating how various components interact.

In this project, the following UML diagrams were used to illustrate the design of the e-commerce platform:

2.6.1 Use Case Diagram

The use case diagram shows the interactions between the system and its users (actors), including customers and administrators.

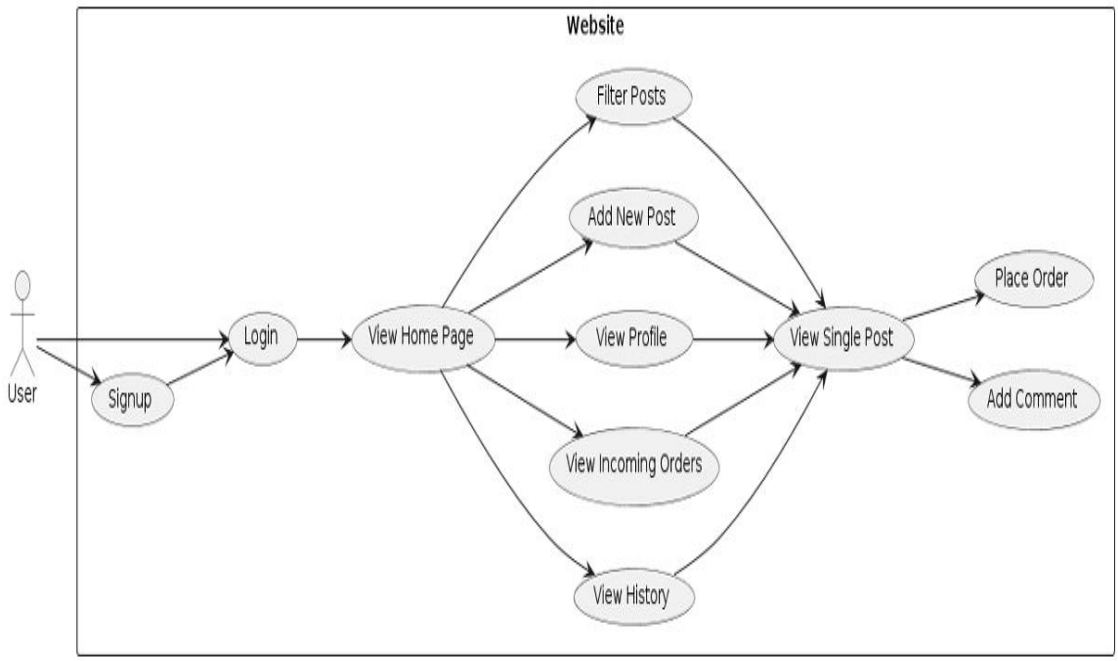


Figure 1: : UML use-case diagram

Actor	Description	Use Case Name	Use Case ID
User	Allows users to create an account or log in	Register/Login	UC01
User	Users can view and search available products	Browse Products	UC02
User	Users can add selected items to their cart	Add to Cart	UC03
User	Users can confirm purchases and place orders	Place Order	UC04
Admin	Admin can add, update, or delete products	Manage Products	UC05
Admin	Admin can view and manage customer orders	View Orders	UC06

Tableau 1 : Description of Main Use Cases

2.6.2 Sequence Diagram

The sequence diagram illustrates the flow of messages and interactions between objects over time for specific scenarios.

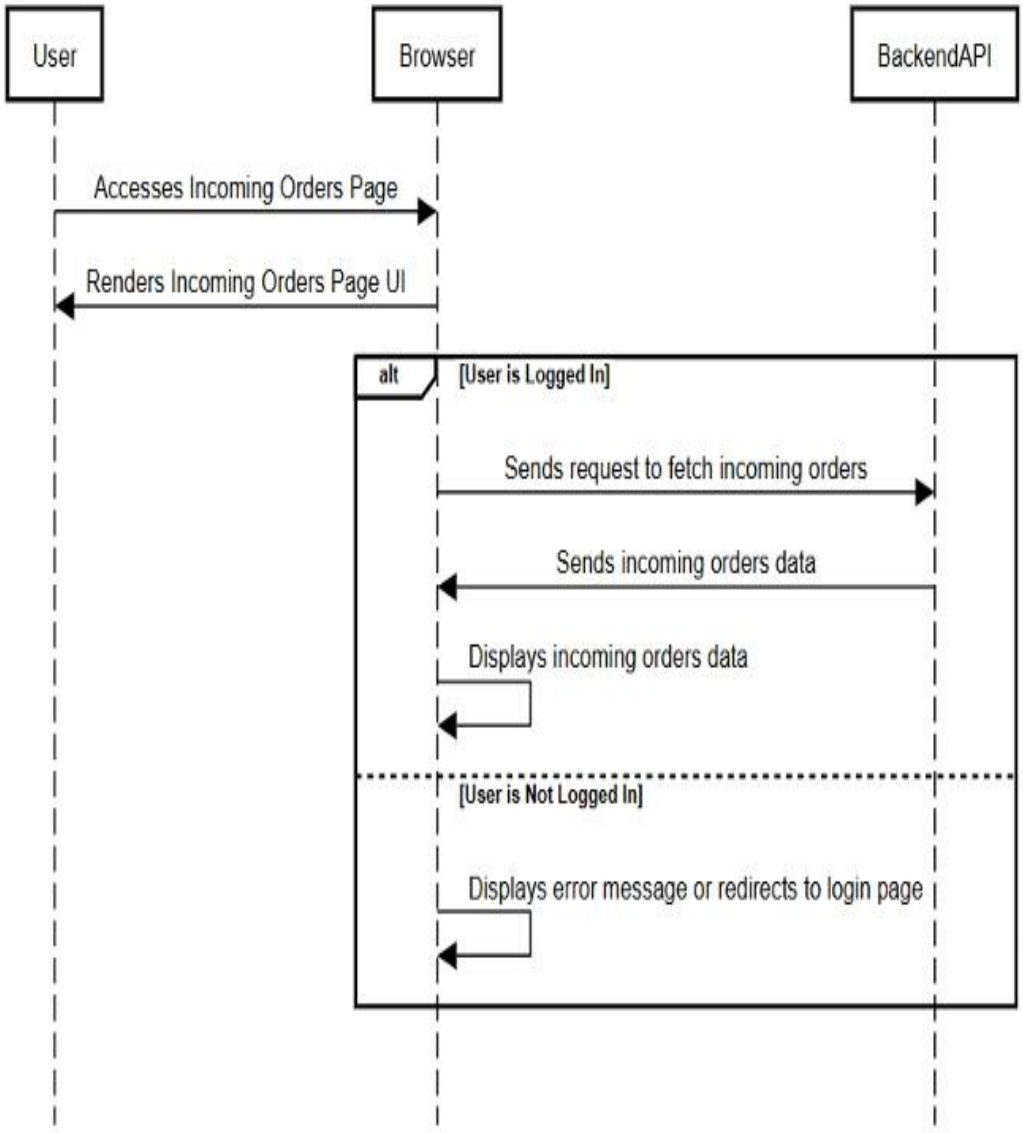


Figure 2: Sequence diagram of the 'Order' page

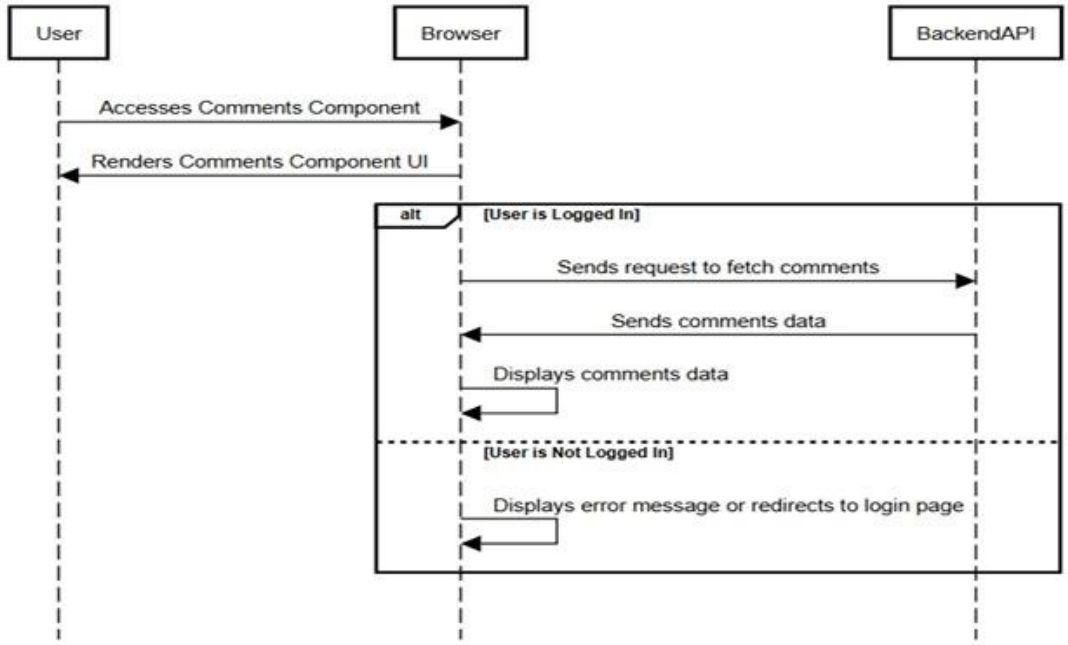


Figure 3: Sequence diagram of the 'Comment' page

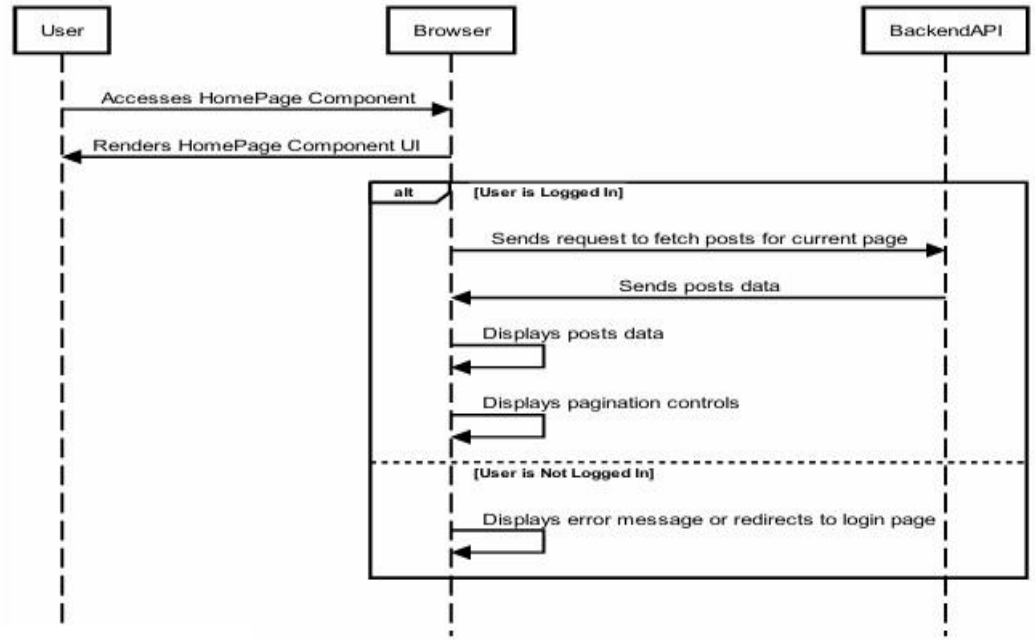


Figure 4: Sequence diagram of the 'Home' page

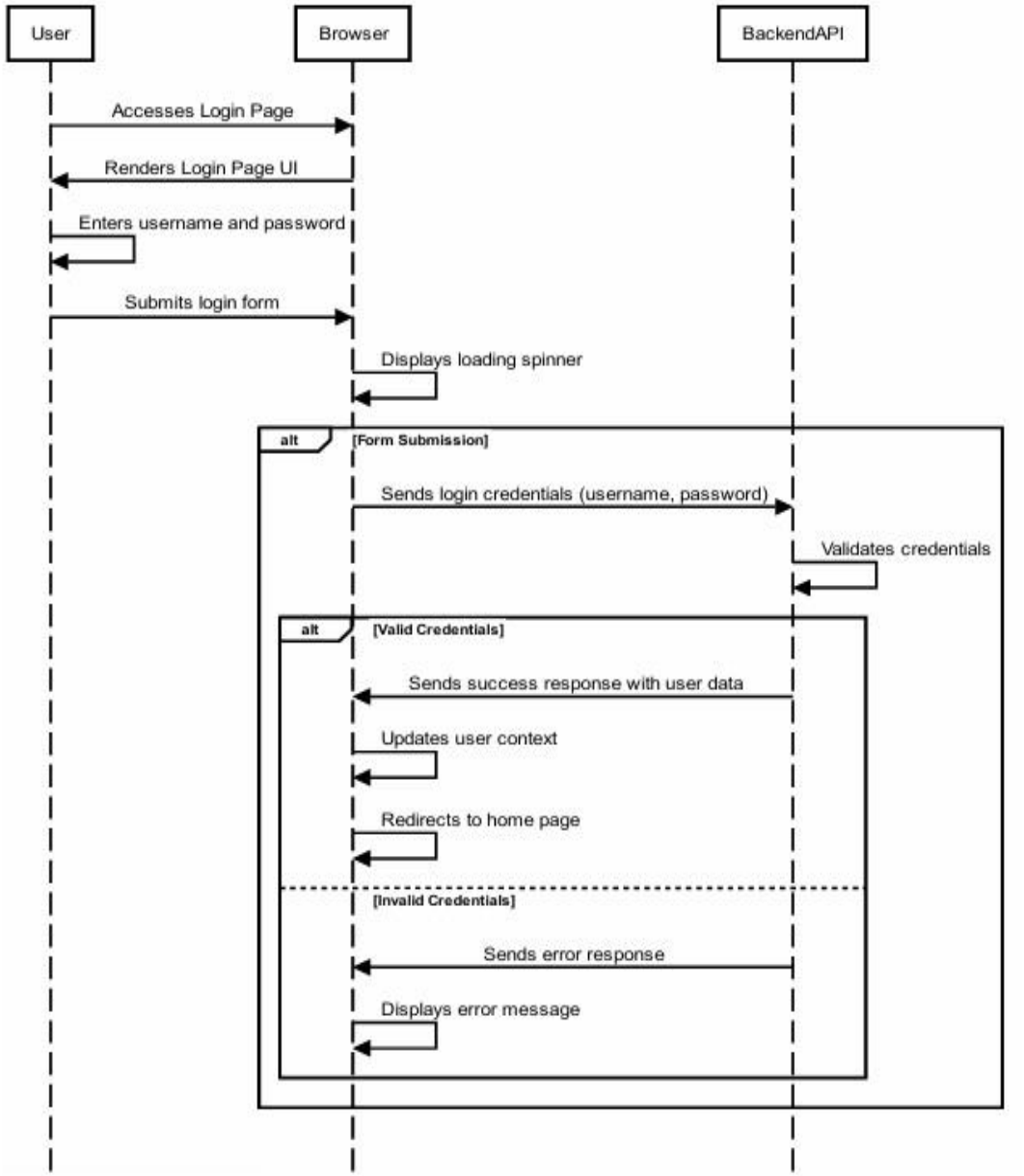


Figure 5: Sequence diagram of the 'Login' page

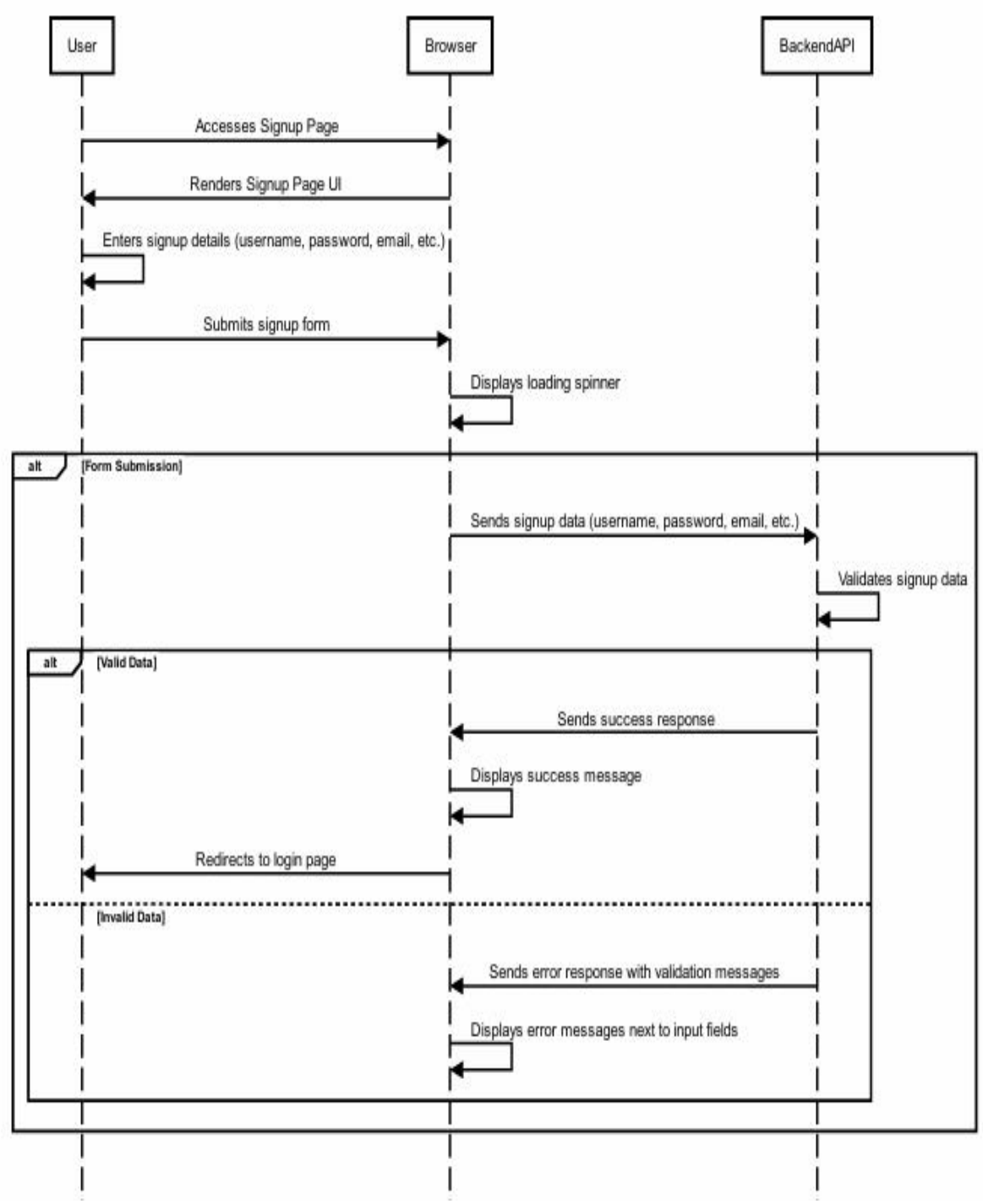


Figure 6: Sequence diagram of the 'Sign Up' page

2.6.3 Class Diagram

The class diagram displays the structure of the system by showing classes, their attributes, methods, and relationships.

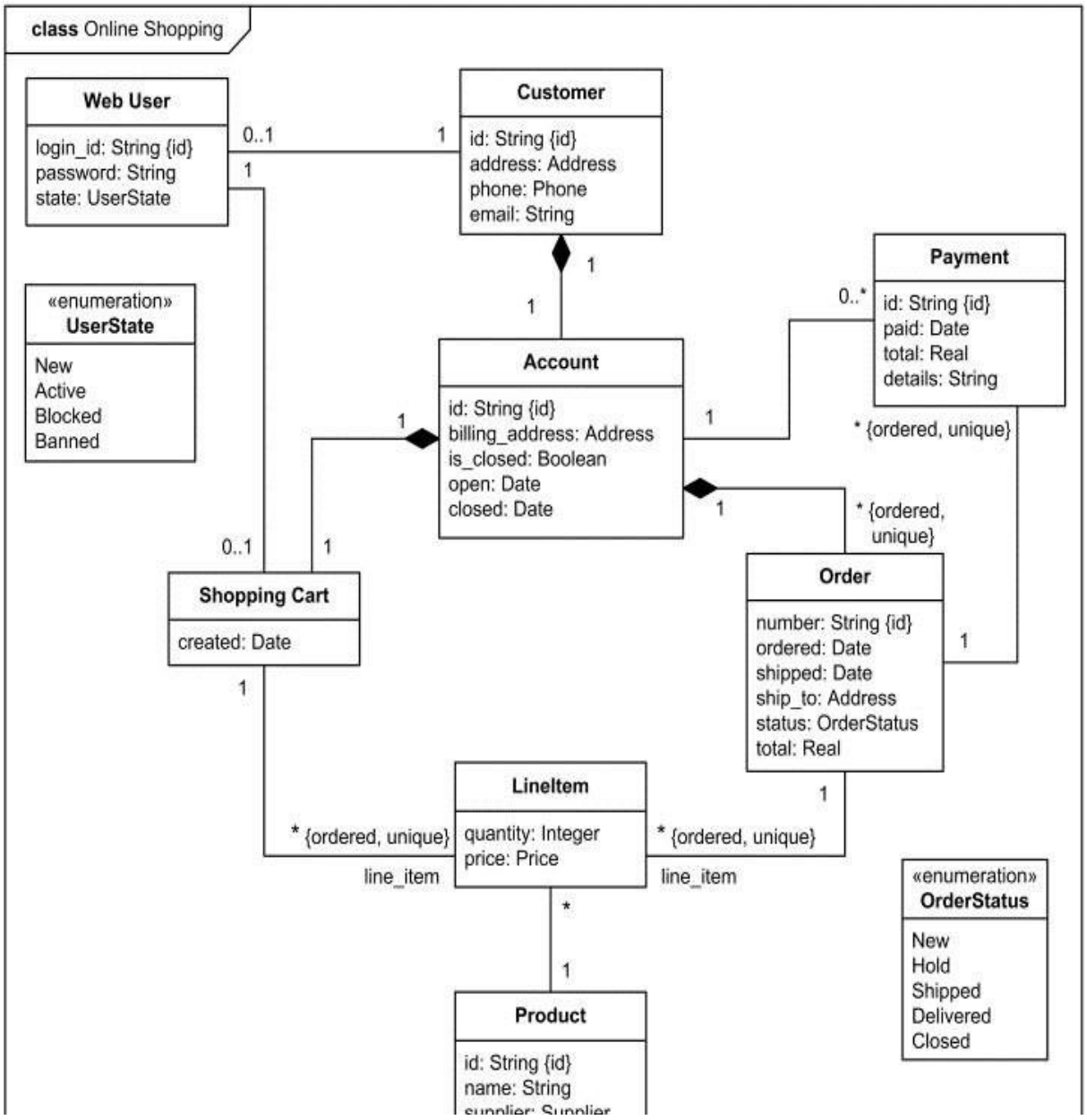


Figure 7: class diagram for an Online Shopping System

CHAPTER 3

Tools and Technologies

3.1 Introduction

In the development of any software project, the selection of the appropriate tools and technologies plays a critical role in determining the project's efficiency, scalability, and overall success. This chapter explores in detail the different tools, languages, libraries, frameworks, and environments that were used during the creation of our e-commerce platform. We also reflect on the reasons behind each choice, highlight their advantages, and suggest alternative solutions that could be used in future versions.

3.2 Development Environment

The environment in which the development takes place strongly influences the productivity and stability of the system. For this project, we used:

- **Operating System:** Windows 11 — offers wide compatibility with development tools.
- **Code Editor:** Visual Studio Code — lightweight, fast, and highly customizable with a rich ecosystem of extensions.
- **Local Server:** XAMPP — provides an Apache server and MySQL database environment for PHP development.
- **Version Control:** Git — used for tracking changes in the source code and collaborating through GitHub.

Additionally, extensions like Laravel Blade snippets, Live Server, and Prettier were used within VS Code to speed up development and improve code formatting.

3.3 Programming Languages and Frameworks

The following programming languages and frameworks formed the technical core of the project:

- **PHP 8.x:** A widely used server-side scripting language.
- **Laravel 10:** The main backend framework, known for its elegant syntax and robust features such as routing, middleware, authentication, and Eloquent ORM.

- **Blade Templating Engine:** Laravel's built-in template engine which enables writing dynamic views using PHP within HTML syntax.
- **HTML5 & CSS3:** For creating structured and styled web content.
- **JavaScript (ES6):** Enhances client-side interactivity.
- **AJAX (Asynchronous JavaScript and XML):** Used to update parts of the webpage asynchronously without reloading it.
- **Bootstrap 5:** A popular CSS framework used to build responsive and mobile-first designs.

Why Laravel?

Laravel was chosen due to its modern features, clear documentation, support for MVC, and built-in tools like Artisan CLI, Eloquent ORM, and Sanctum for authentication. It promotes clean, readable, and maintainable code.

3.4 Database

- **Database System:** MySQL was used for its reliability and ease of integration with Laravel.
- **Management Tool:** phpMyAdmin for database browsing and management.

The relational schema includes tables such as users, products, categories, orders, order_items, and carts, with foreign key constraints to ensure data integrity.

Laravel migrations were used to manage the database structure and make schema versioning easier across development stages.

3.5 Design Tools

To design the visual layout and structure of the platform:

- **Figma:** Used to create wireframes, mockups, and high-fidelity prototypes of the interface.
- **StarUML:** Used for creating UML diagrams (Use Case, Class, Sequence) to represent the system's logic visually.

These tools helped in planning the front-end experience and defining system logic before implementation.

3.6 APIs and Libraries

Modern applications require asynchronous communication. Our platform used:

- **Laravel RESTful APIs:** Built-in support in Laravel to create REST endpoints.
- **Composer:** Dependency manager for PHP packages.
- **npm (Node Package Manager):** Used for installing JavaScript dependencies like Bootstrap and jQuery.
- **Axios:** Lightweight HTTP client used with AJAX for sending API requests.

APIs were created for:

- Fetching product details dynamically.
- Adding/removing items from the cart.
- Submitting orders without reloading pages.

This API-first approach ensures that the backend is decoupled from the front-end and can be reused across different platforms such as mobile apps.

3.7 Testing Tools

- **PHPUnit (Laravel Testing):** Laravel provides an integrated testing environment with PHPUnit to create unit and feature tests.
- **Browser Developer Tools:** Used to inspect HTML/CSS, debug JavaScript, and monitor API requests.
- **Laravel Log Files:** Helped trace errors and issues in back-end logic.

We tested authentication workflows, product addition, order placement, and admin access restrictions.

Deployment and Hosting Considerations

Though the project was developed locally, we considered future deployment. Possible deployment options include:

- **Shared Hosting Providers:** Such as Hostinger or Namecheap, for small-scale deployments.
- **VPS or Cloud Platforms:** Like DigitalOcean or Laravel Forge, for scalable production use.
- **Database Hosting:** Using external MySQL services or managed databases like PlanetScale.

We also explored the use of **.env** files for configuration and environment switching.

Alternative Tools Considered

During our planning phase, we compared several tools:

- **Symfony** instead of Laravel: More complex, but offers high flexibility.
- **Vue.js or React:** As full front-end frameworks for SPA development.
- **MongoDB:** For NoSQL structure, but was not suitable due to the nature of our relational data.
- **Tailwind CSS:** Considered for utility-first design but required more setup.

Conclusion :

The tools and technologies selected for this project were crucial to achieving a secure, scalable, and maintainable e-commerce application. Laravel's ecosystem, combined with front-end technologies like Bootstrap and JavaScript, allowed for rapid development. With structured planning and the help of modern design and debugging tools, the platform was successfully implemented and can easily be extended in the future.

Future upgrades in the tech stack may include: Progressive Web App support, API rate limiting, queue management using Laravel Queues, or real-time updates with Laravel Echo and WebSockets.

CHAPTER 4

Implementation and Testing

4.1 Home Page

- The **Home Page** introduces the platform with featured products and categories to

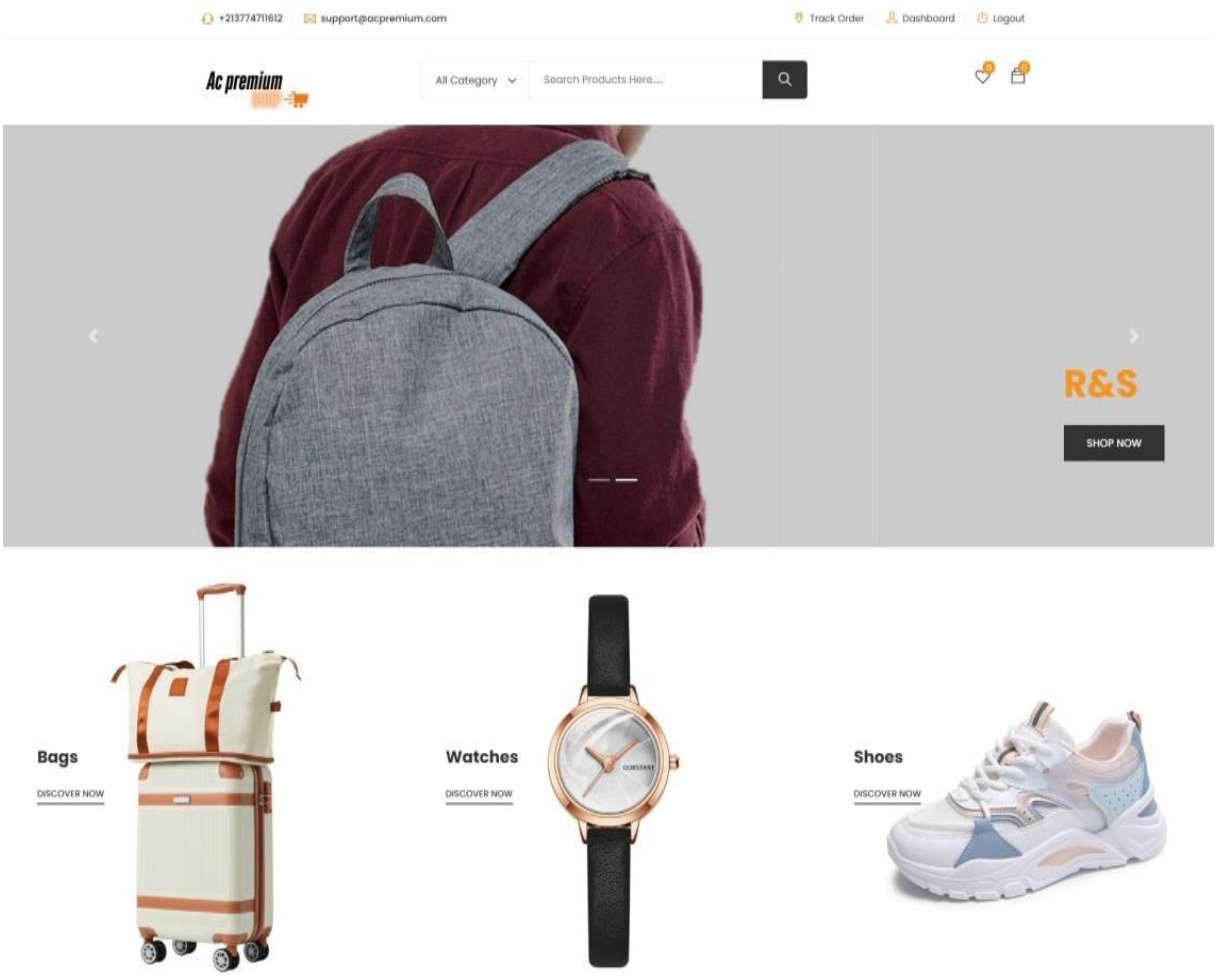
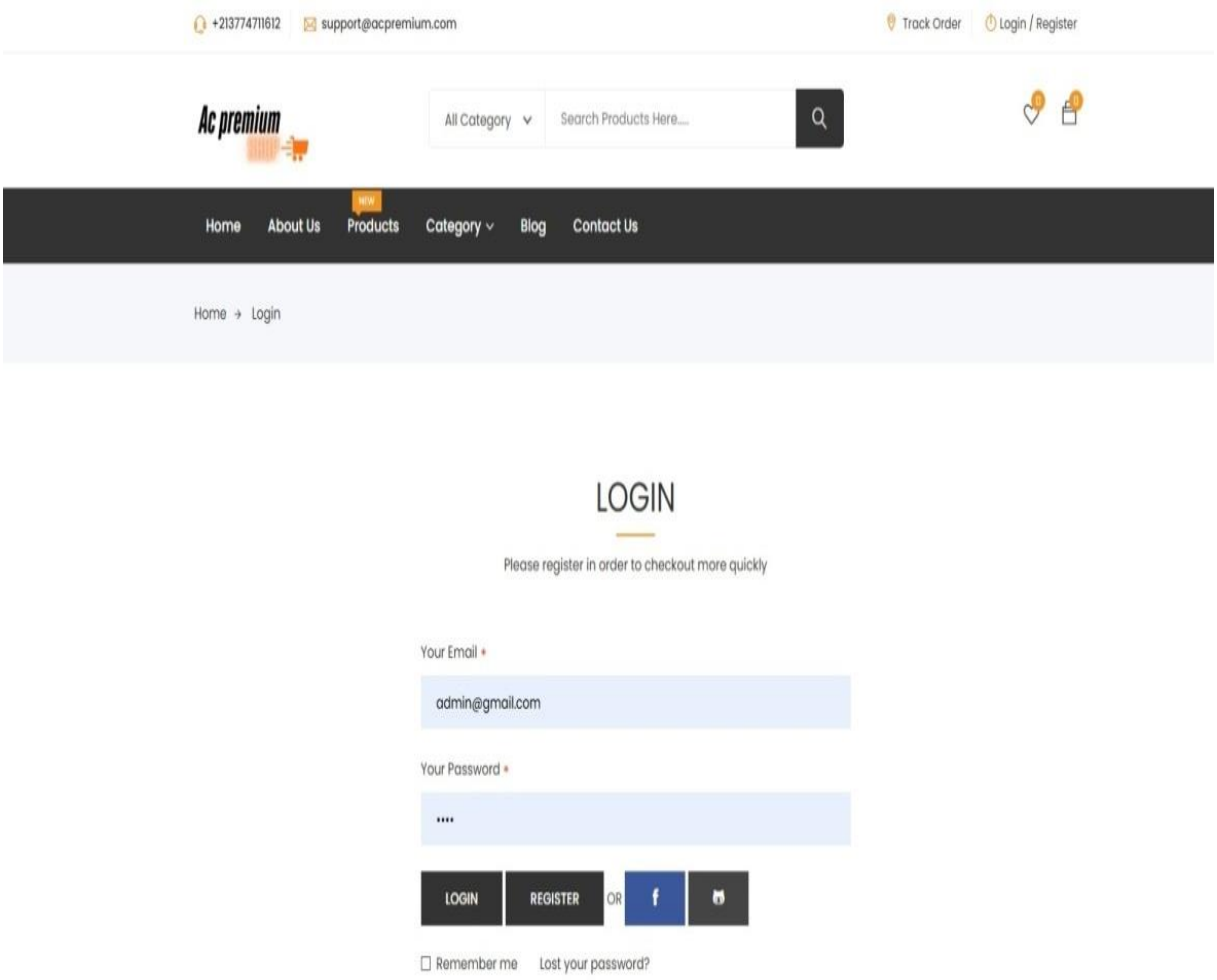


Figure *8:* Home Page

attract attention.

4.2 Login Page



- The **Login Page** allows users to securely access their accounts.

Figure *9:* Login Page

4.3 Product Details Page

- The **Product Details Page** provides information about each product, including

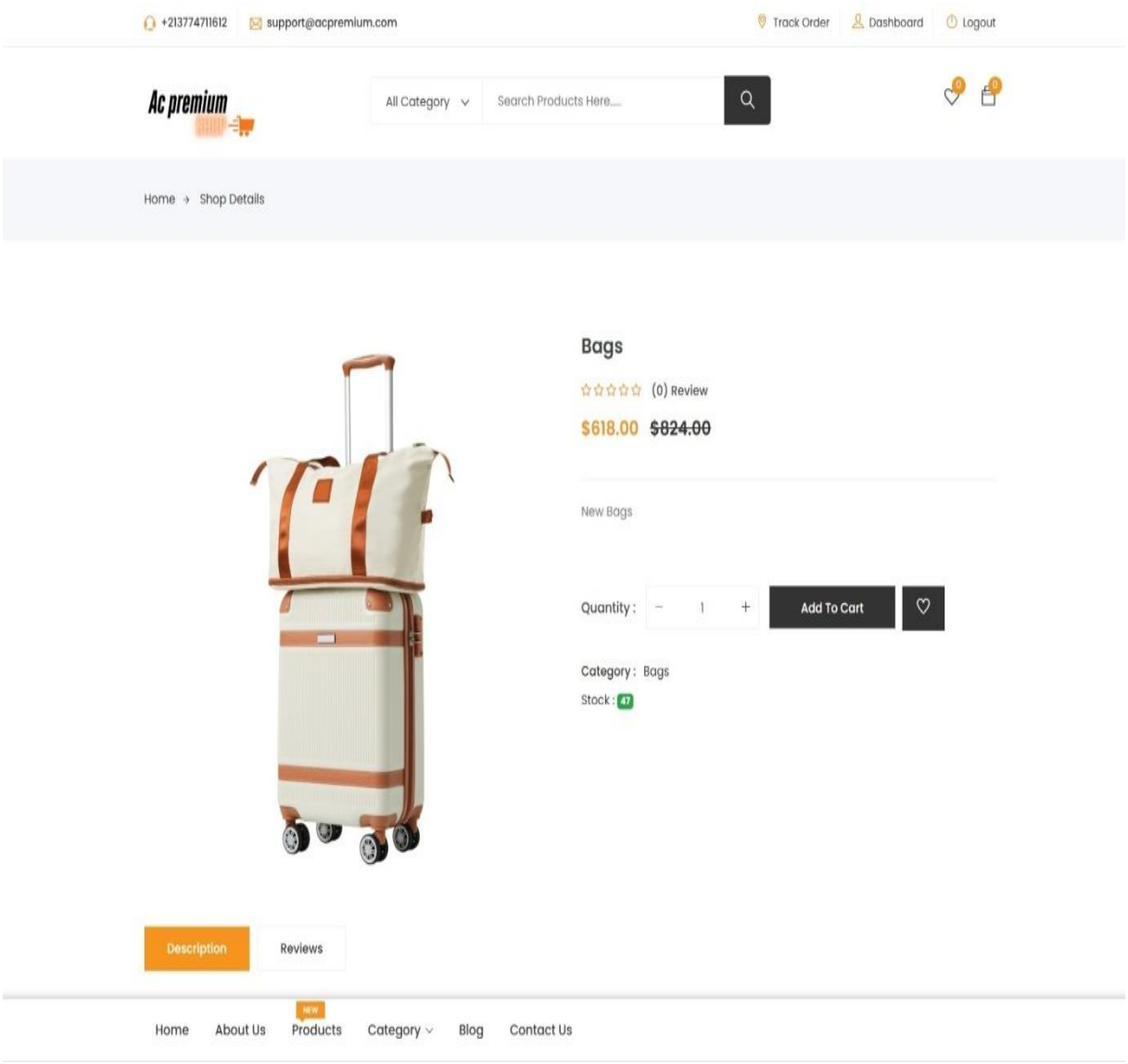


Figure *10:* Product Details Page

name, price, and description.

4.4 Contact Page

- The **Contact Page** offers a form or contact section where users can send messages, or ask questions.

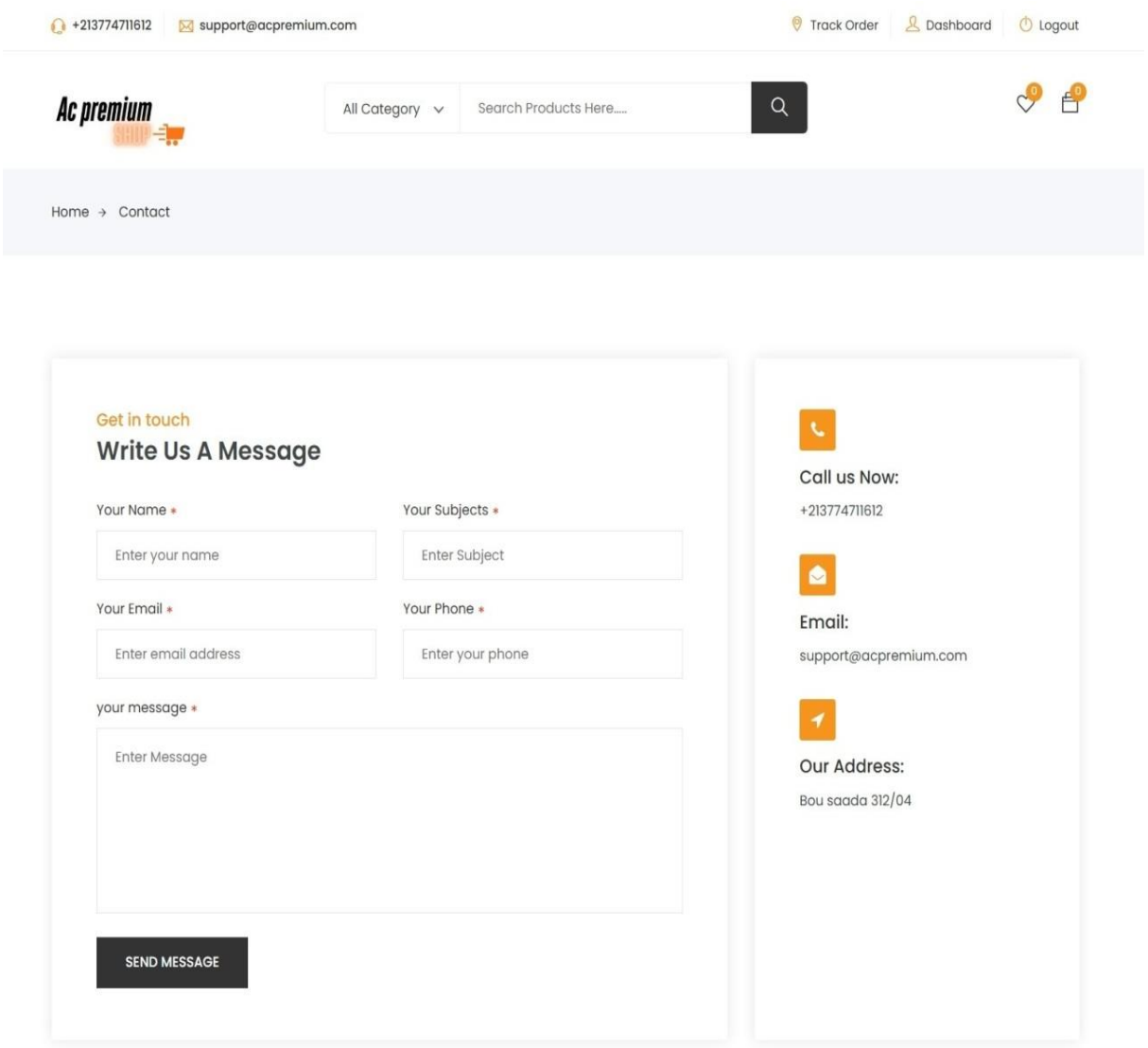


Figure *11:* Contact Page

4.5 Shopping Cart Page

- The **Shopping Cart Page** summarizes the selected products, their quantities, and total cost before proceeding to checkout.

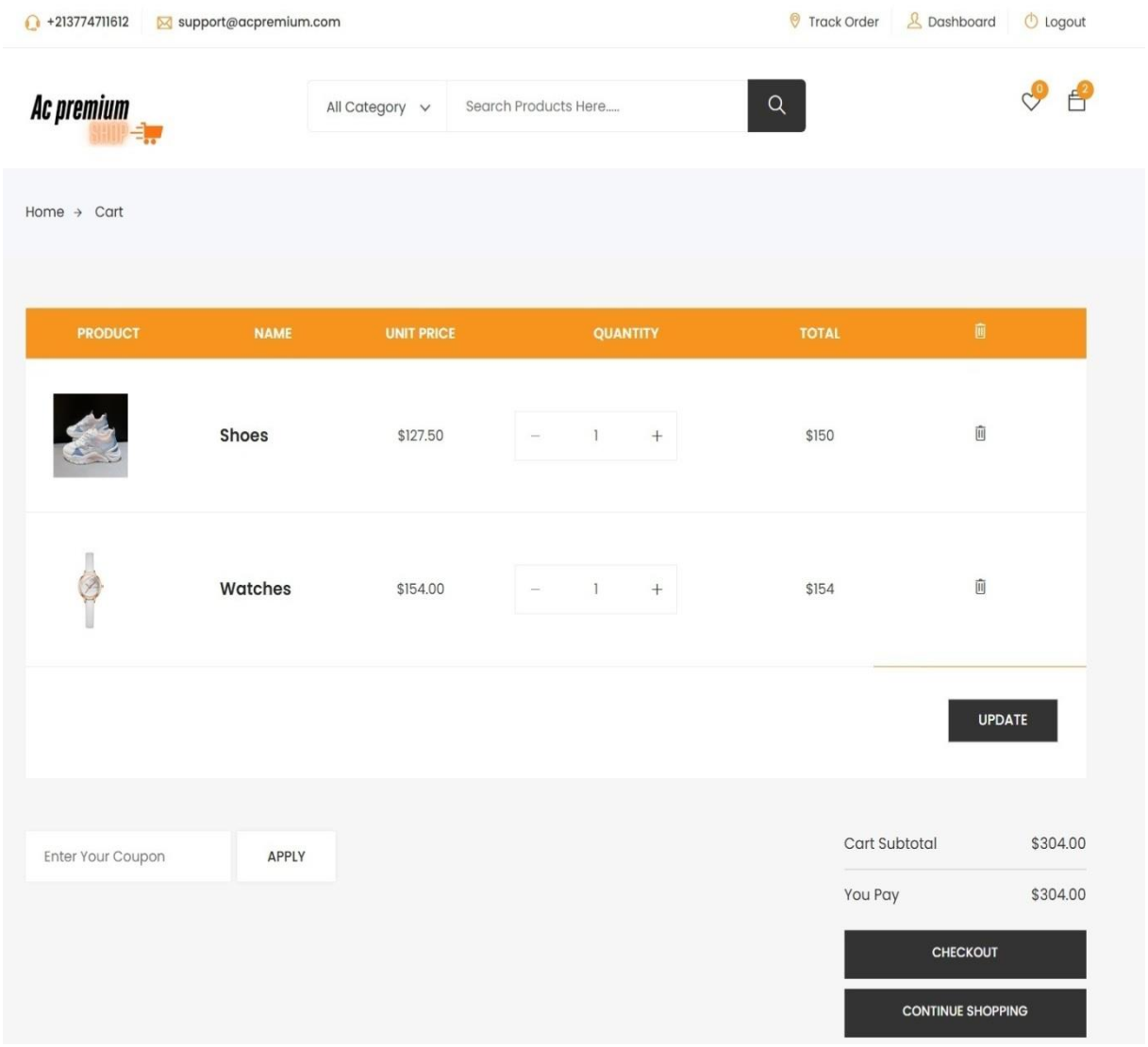


Figure *12:* Shopping Cart Page

4.6 Checkout Page

- The **Checkout Page** is where the user enters delivery information and confirms the order

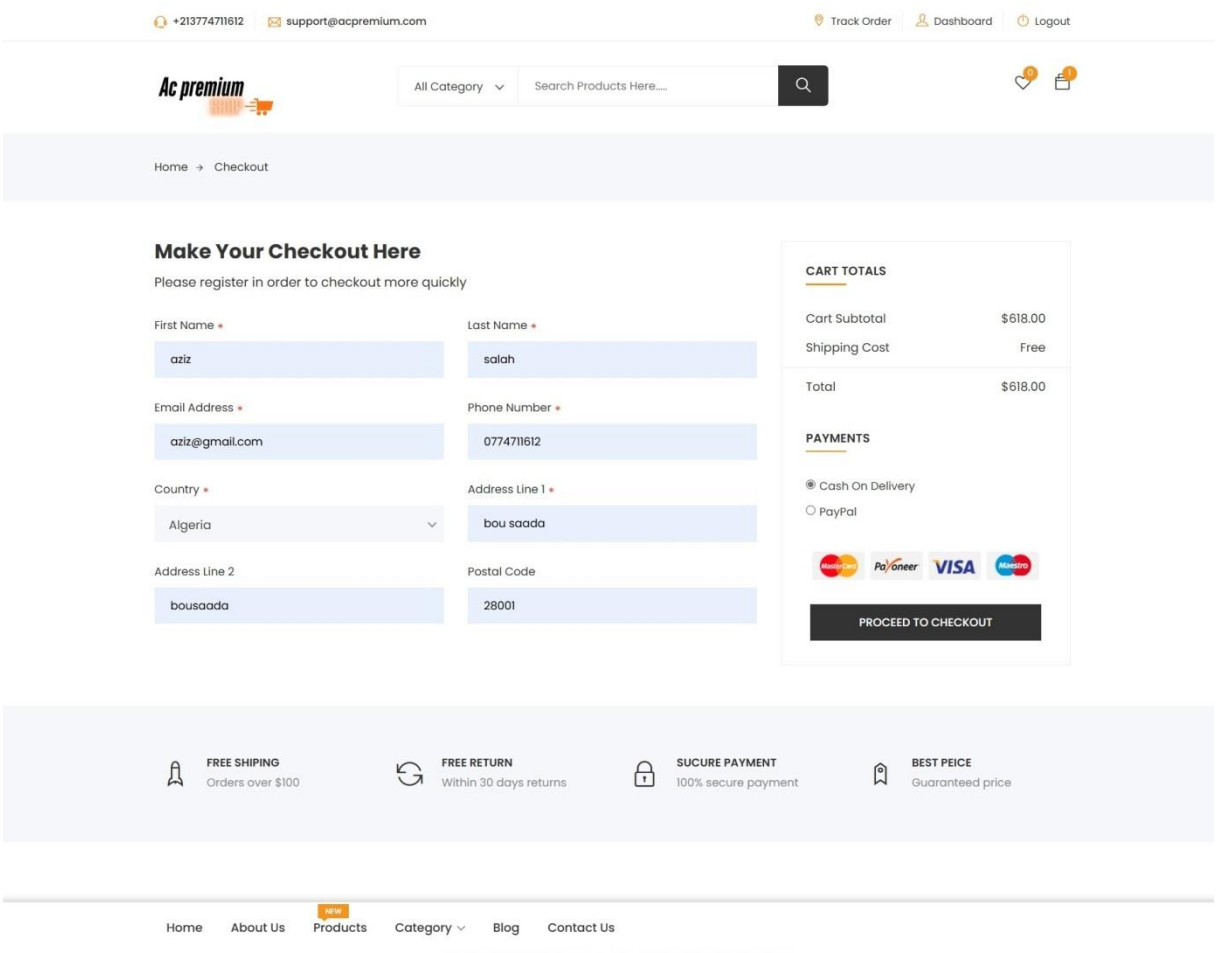


Figure 13: Checkout Page

4.7 User Dashboard

- User Dashboard Once logged in, users can access a private dashboard where they can:

1. View their order history
2. Update their account details
3. Track current orders

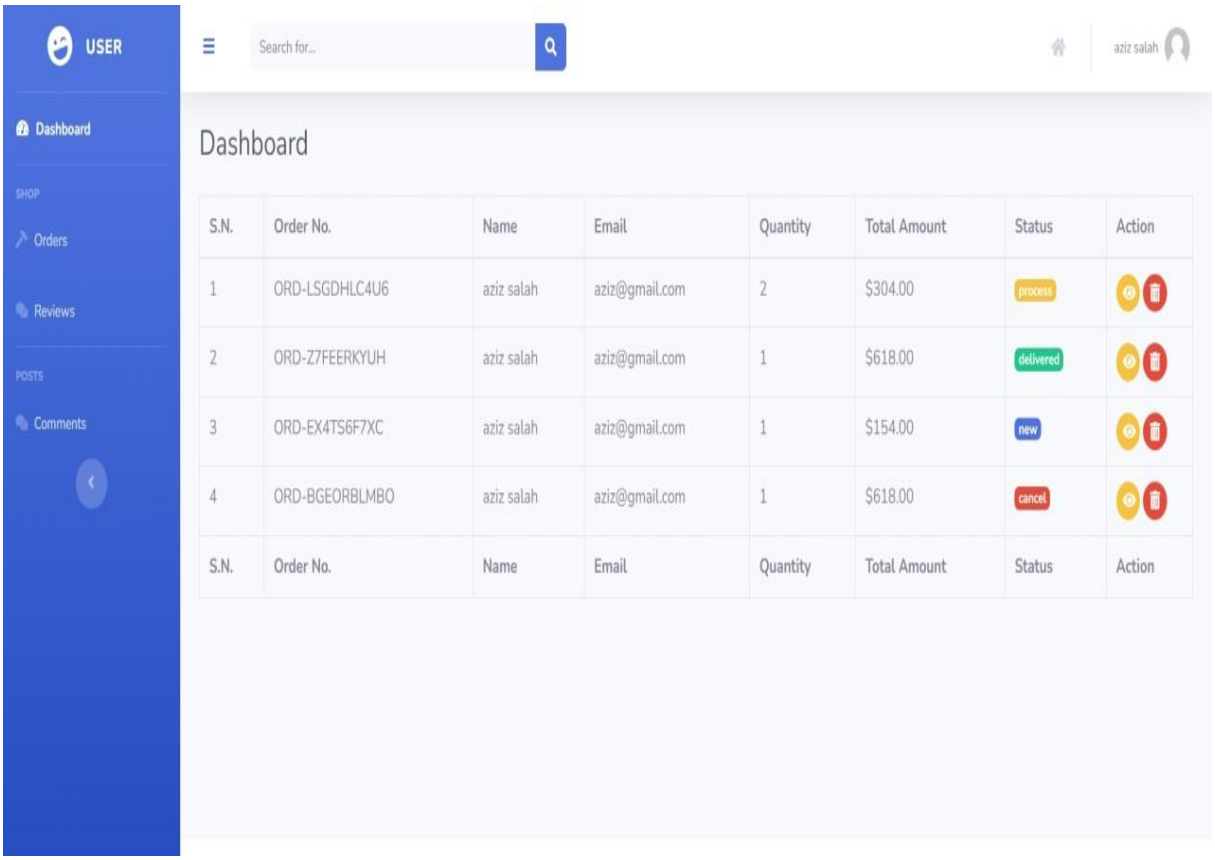


Figure *14:* User Dashboard

4.8 Admin Dashboard

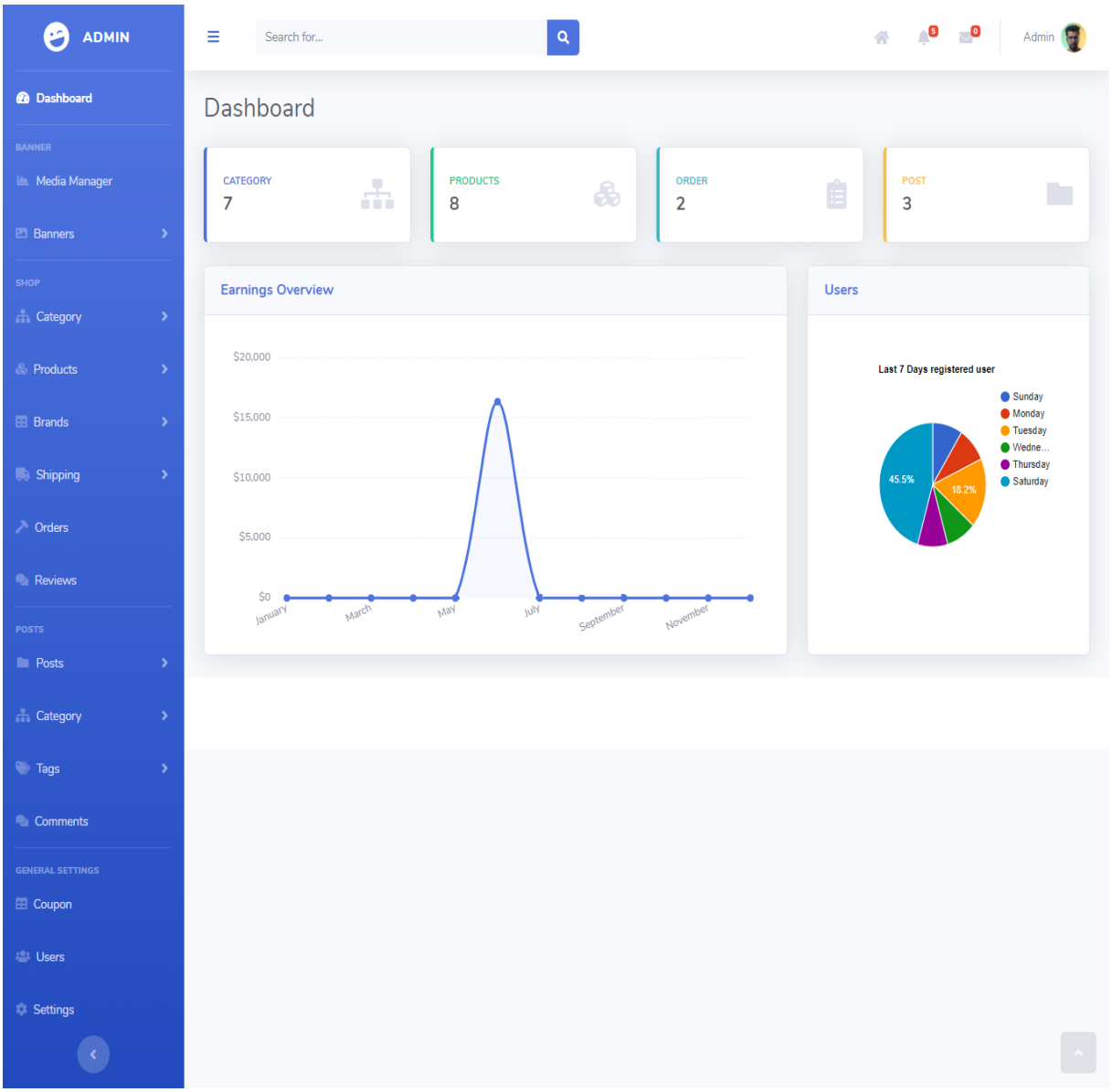


Figure *15:* Admin Dashboard

The Admin Dashboard provides authorized administrators with a centralized interface to manage and monitor all system operations. Once logged in, administrators can:

- **Manage Products** – Add, update, or delete product listings, including titles, descriptions, prices, images, and stock levels.
- **Manage Categories** – Create and organize product categories to improve browsing and navigation.
- **View and Process Orders** – Access all customer orders, update order statuses (e.g., pending, shipped, delivered), and generate order reports.
- **Manage Users** – View registered users, update roles or status (e.g., active, banned), and respond to customer inquiries.
- **Access Analytics** – Review sales data, product performance, and customer behavior to support decision-making.
- **System Settings** – Configure general system parameters such as currency, tax, shipping methods, and application settings.

This dashboard enhances administrative efficiency by streamlining backend operations hrough a user-friendly and secure interface.

4.9 Admin Panel (Back-End)

- The admin panel is used by the store manager to oversee and control the system's data. It **Product Management**: Add, edit, or delete products.

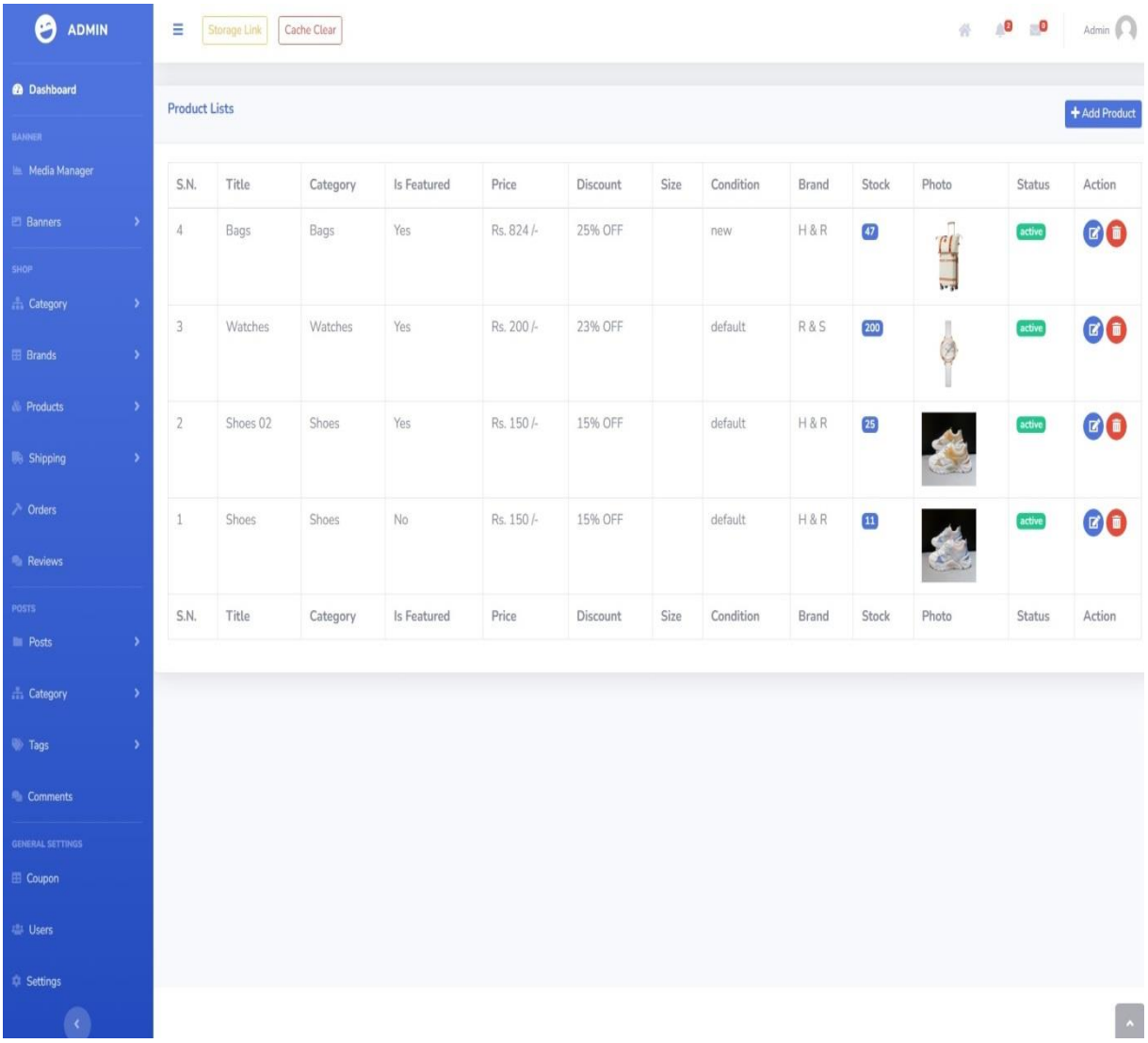


Figure ***16:*** Product List (Admin)

4.10 Order Management

- **Order Management**: View all orders, approve, or cancel them.

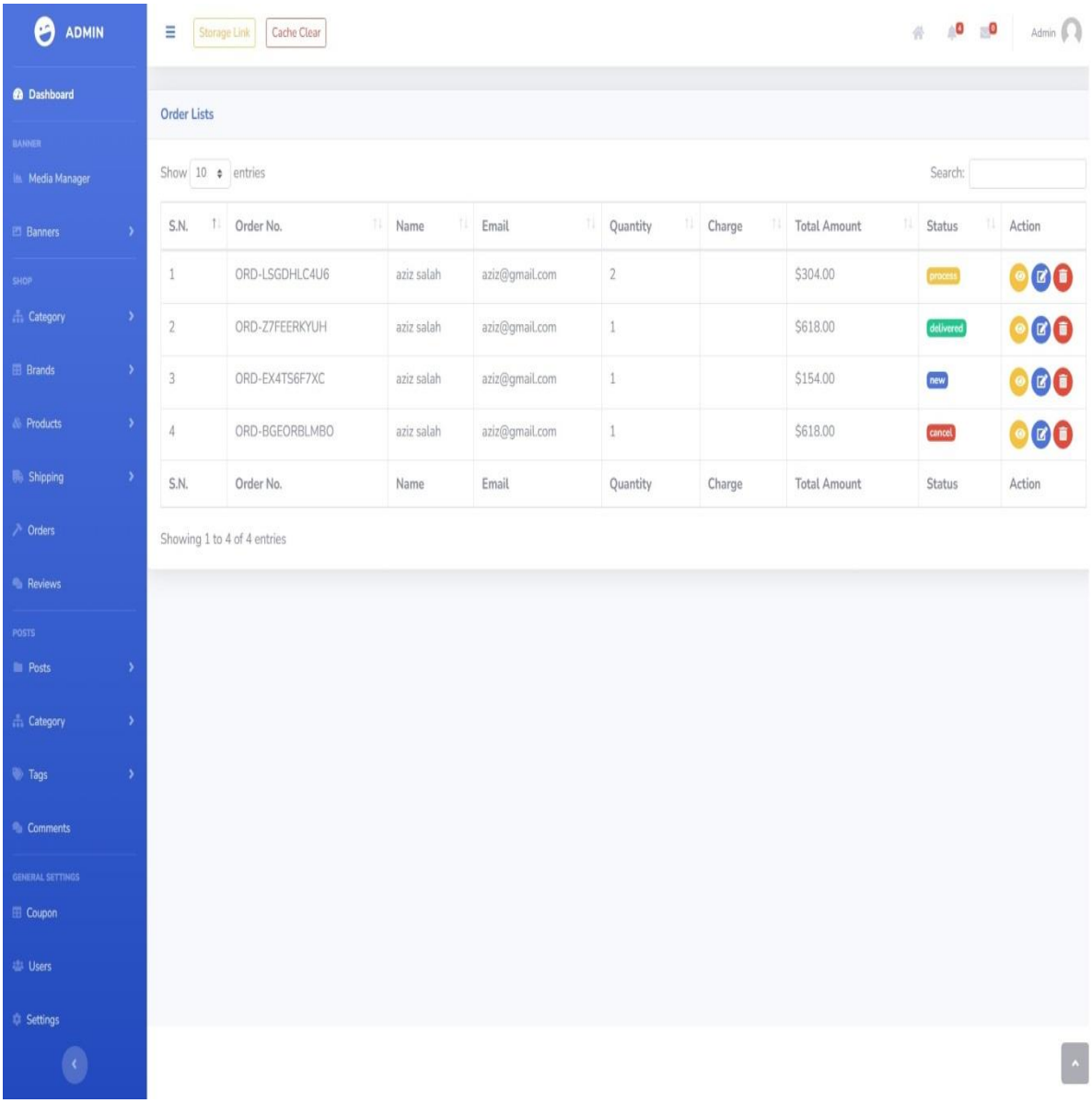


Figure ***17:*** Order List (Admin)

4.11 Category Management

- **Category Management**: Organize and edit product categories.

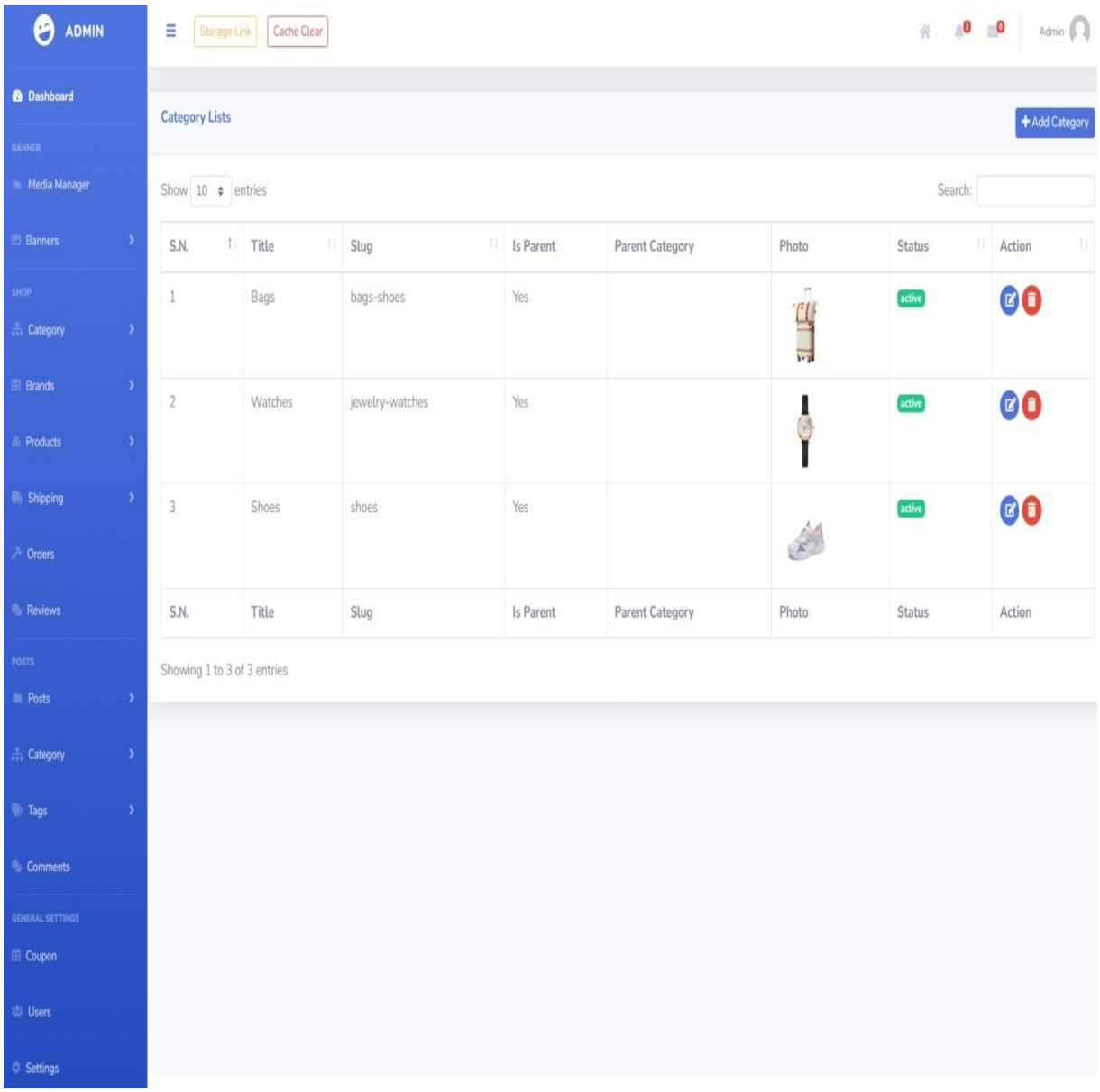


Figure *18:* Category List (Admin)

CHAPTER 5

Evaluation and Perspectives

5.1 Introduction

This chapter is dedicated to critically evaluating the work carried out during the development of the e-commerce platform. It discusses the system's strengths and weaknesses, offers reflections on the outcomes, and outlines suggestions for future improvements and perspectives. This self-assessment aims to identify what has been successfully achieved and what areas remain to be improved in the context of a real-world deployment.

5.2 Strengths

The developed system possesses several strengths that contribute to its functionality and effectiveness:

- **Core Functionalities:** The platform successfully includes essential features such as product listing, shopping cart, user authentication, and order placement.
- **Admin Dashboard:** A well-structured and secure administrative interface allows product, category, and order management efficiently.
- **Security Measures:** The use of Laravel's built-in authentication ensures a secure login system. Passwords are hashed, and user data is transmitted over HTTPS (simulated environment).
- **Responsiveness:** With Bootstrap and responsive design principles, the interface works across various devices.
- **RESTful APIs:** Enable asynchronous actions like adding to cart or submitting orders without refreshing pages.
- **MVC Architecture:** Clear separation of concerns, which improves maintainability, scalability, and collaborative development.
- **User Experience:** The layout is intuitive, navigation is simple, and the platform offers a smooth shopping experience.

5.3 Weaknesses

Despite its functional completeness, the system has some limitations:

- **No Real Payment Integration:** While the order workflow is complete, real payment gateways (e.g., PayPal, Stripe) have not been implemented.
- **Basic Search System:** The search is limited to product titles. It lacks smart suggestions, category filtering, and typo tolerance.
- **Limited Error Handling:** Certain exceptions and errors (e.g., invalid inputs, server issues) are not adequately managed.
- **No Multi-language Support:** The interface is only in English; users cannot choose another language.
- **Lack of Notifications:** There is no real-time feedback for order confirmation, shipping updates, or admin actions.
- **Basic Analytics:** While the admin dashboard exists, advanced sales statistics and user behavior tracking are minimal.

5.4 Future Perspectives

To improve the project and make it ready for a production environment, we suggest the following future developments:

- **Payment Gateway Integration:** Implement secure payment methods using APIs from Stripe, PayPal, or CIB (Algeria).
- **Advanced Search and Filters:** Add full-text search with Laravel Scout or MeiliSearch, including filtering by price, category, rating, and sorting.
- **Multilingual Support:** Enable the platform to support multiple languages using Laravel localization features.
- **Mobile App Version:** Develop a native or hybrid mobile application using Flutter or React Native connected to the same backend APIs.
- **Order Tracking and Notifications:** Allow users to track order status and receive email or SMS notifications for order updates.
- **Review and Rating System:** Allow users to leave reviews and rate products.
- **Better Error Handling:** Implement global exception handling and user-friendly error messages.
- **SEO Optimization:** Improve visibility on search engines with metadata, optimized URLs, and sitemap.

- **Analytics Dashboard:** Include charts and reports showing sales trends, user activity, and product performance.

Technical Lessons Learned

Through this project, several technical skills and practices were strengthened:

- **Database Normalization:** Understanding how to model efficient relational databases.
- **REST API Design:** Structuring endpoints and handling asynchronous front-end interactions.
- **Testing with PHPUnit:** Writing and executing unit and feature tests in Laravel.
- **Code Modularity:** Applying separation of logic using MVC.
- **Version Control:** Using Git for tracking changes and collaboration.

Team Collaboration and Project Management

The collaboration between the team members played a crucial role in the project's success. Work was divided fairly based on strengths:

- Salah Eddine focused on front-end design, Blade templating, and user experience.
- Abdallah handled database design, admin panel logic, and API integration.

Regular communication and task distribution using a shared GitHub repository ensured consistent progress.

Conclusion

This evaluation demonstrates that the project met its primary goals and provided a working e-commerce platform prototype. While limitations exist, they are well-identified and serve as opportunities for improvement. With some additional features and optimizations, the system could evolve into a robust solution ready for real-world application.

The experience has strengthened our understanding of full-stack web development, Laravel framework capabilities, and the complexities of building production-ready systems.

General Conclusion

This project aimed to design and implement a functional e-commerce platform using Laravel. Throughout the development process, we covered essential steps including requirement analysis, system design, implementation, and evaluation. The result is a modular and responsive web application that supports product browsing, cart management, and order placement.

Although the project achieved its main objectives, there remains room for enhancements. Integrating real-world features such as payment gateways, multi-language support, and mobile compatibility will make the system more practical and ready for deployment. This work has improved our technical and teamwork skills and serves as a foundation for future software development projects.

Abstract:

This project presents the development of an e-commerce web application using the Laravel framework. It includes essential functionalities such as product browsing, shopping cart, user authentication, and order management. The system also features an admin dashboard for content control. The work followed a structured methodology from analysis to testing, resulting in a scalable and user-friendly platform. Recommendations for future work include payment gateway integration, mobile application support, and enhanced security.

Key words: E-commerce ; Web Development

الملخص:

يهدف هذا المشروع إلى تطوير منصة تجارة إلكترونية باستعمال لارفيل، يشمل النظام وظائف أساسية مثل عرض المنتجات، إدارة سلة التسوق، وإنشاء الحسابات وإجراء الطلبات. كما يتضمن نظاما للإدارة للتحكم في محتوى الموقع اعتمدنا منهجية من التحليل إلى الاختبار، ونقدم في الختام مقترحات لتطوير النظام في المستقبل.

الكلمات المفتاحية: التجارة الالكترونية ؛ تطوير الويب

Bibliography

[1] UNCTAD. (2021). *COVID-19 and E-commerce: A Global Review*. United Nations.

[2] Laudon, K. C., & Traver, C. G. (2021). *E-commerce 2021: Business, Technology and Society*. Pearson.

[3] Nielsen, J. (2012). *Usability Engineering*. Morgan Kaufmann.

[4] Laravel Documentation. (2024). https://laravel.com/docs

[5] Sommerville, I. (2015). *Software Engineering* (10th ed.). Pearson.

[6] Bootstrap Documentation. (2024). https://getbootstrap.com/docs

[7] Fielding, R. T. (2000). *Architectural Styles and the Design of Network-based Software Architectures*. Doctoral Dissertation, University of California.