Journal Publication of International Research for Engineering and Management (JOIREM)
Volume: 04 Issue: 4| April 2026
ISSN (O) 3107-6696

EazyStore: Design and Implementation of a Secure Full-Stack E-Commerce Web Application

"Integrating JWT Authentication, Role-Based Access Control, and Stripe Payment Processing in a Modern Web Platform"

[1]Dhanushraj G (23CU0310381), [2]Ms. P. Chandralekha

[2]Assistant Professor
Computer Science and Engineering, Hindustan Institute of Technology and Science

---

Abstract – The rapid growth of online commerce demands e-commerce platforms that are simultaneously secure, scalable, and user-friendly. This paper presents EazyStore, a full-stack web application that integrates JWT-based stateless authentication, BCrypt credential hashing, role-based access control, and Stripe payment processing within a Spring Boot and React architecture. The system supports the complete e-commerce lifecycle — user registration and login, product browsing, cart management, secure checkout, and admin order management — while enforcing application-level security at every layer. A modular RESTful API design separates concerns across authentication, product, cart, order, and admin domains. Experimental validation using Stripe test transactions and role-scoped endpoint testing confirms correct security enforcement and end-to-end functional completeness. EazyStore provides an accessible, developer-transparent reference implementation for secure full-stack e-commerce development.

Keywords— E-Commerce Security, JSON Web Token (JWT), Spring Boot, React, Role-Based Access Control, BCrypt, Stripe Payments, RESTful API, Spring Security, Full-Stack Web Development.

I. INTRODUCTION

The rapid expansion of digital commerce has made secure, scalable e-commerce platforms indispensable. As consumers increasingly rely on online channels for transactions, the demand for robust authentication, safe payment handling, and seamless user experience has never been greater. Poorly secured platforms expose users to risks such as credential theft, unauthorized API access, and payment fraud, all of which erode trust and invite regulatory scrutiny.

EazyStore addresses these challenges by providing a full-stack e-commerce web application built on Spring Boot and React, with JWT-based stateless authentication, role-based access control, BCrypt password hashing, and Stripe payment integration. The platform delivers a cohesive shopping experience covering product browsing, cart management, checkout, and administrative oversight, while enforcing security at every layer.

The specific problem this work addresses is the absence of an accessible, developer-transparent, and production-ready e-commerce reference that demonstrates secure authentication, RESTful API design, and payment integration in a single cohesive system. EazyStore fills this gap, offering a clearly architected, extensible platform suitable for academic study and real-world deployment.

II. BACKGROUND AND RELATED WORK

A. Traditional E-Commerce Platforms

Early platforms such as OpenCart and legacy Magento employ monolithic architectures with session-based authentication. They suffer from tight coupling between frontend and backend, limited scalability under high concurrent loads, and inadequate support for modern security standards. These shortcomings make them unsuitable as a basis for a secure, modern e-commerce implementation [1].

B. JWT-Based Authentication Systems

JSON Web Tokens have emerged as the standard for stateless authentication in distributed web applications. Research by Rana and Pandey (2024) demonstrates that HMAC-SHA-256 signed JWTs significantly reduce session management overhead while improving resistance to session hijacking [2]. Gbenle (2025) further establishes best practices for applying OAuth2 and JWT in securing distributed API gateways [3]. EazyStore builds upon these principles, issuing signed JWTs upon login and validating them on every protected endpoint.

C. Spring Boot Security Frameworks

Spring Security provides a comprehensive authentication and authorization framework for Java-based applications. Kokila et al. (2025) highlight that Spring Security's filter chain, combined with stateless JWT validation, effectively prevents unauthorized access in distributed systems [4]. Aldea et al. (2025) extend this analysis to microservice architectures, confirming the robustness of token-based authentication mechanisms [5]. EazyStore leverages Spring Security to enforce BCrypt hashing and ROLE_USER / ROLE_ADMIN separation.

D. RESTful API Security

Tanveer et al. (2025) survey common RESTful API vulnerabilities including injection attacks, broken object-level authorization, and insufficient CORS configuration, proposing input validation and strict CORS policies as primary mitigations [6]. Mao et al. (2025) extend this to multi-tenant API gateways, reinforcing the need for role-scoped data access [7]. EazyStore enforces CORS configuration and input validation across all endpoints to mitigate these risks.

E. Research Gap

Existing academic implementations typically address authentication or payment integration in isolation. None provides a unified, transparently documented Spring Boot + React + Stripe reference that cohesively demonstrates JWT authentication, RBAC, BCrypt storage, CORS hardening, and admin management in a single deployable system. EazyStore addresses this gap.

III. SYSTEM ARCHITECTURE

A. Layered Architecture

EazyStore employs a three-tier architecture. The Presentation Layer is implemented in React 18 with Vite, providing a responsive single-page application. The Business Logic Layer is a Spring Boot REST API enforcing authentication, authorization, and Stripe payment orchestration. The Data Layer is a MySQL 8.0 database managed via Spring Data JPA, storing users, products, orders, addresses, roles, and contact messages.

B. Security Architecture

A dedicated security sub-layer underpins the application. JWT tokens are signed with HMAC-SHA-256 and validated by a Spring Security filter on every request. BCrypt (strength 12) is used for password hashing, ensuring that stored credentials resist brute-force attacks even if the database is compromised. CORS configuration restricts cross-origin access to trusted origins, and global exception handlers prevent stack trace leakage.

C. Payment Architecture

Stripe payment intents are created server-side within the Spring Boot backend, ensuring that secret keys never reach the client. The React frontend uses Stripe Elements to collect card details directly with Stripe's servers, eliminating PCI scope from the EazyStore backend. Upon payment confirmation, the backend creates and persists the order.

IV. MODULE DESCRIPTION

A. Authentication Module

The authentication module handles user registration and login via Spring Security. Passwords are hashed using BCrypt before storage. On successful login, a signed JWT is issued containing the user's subject and role claims. Subsequent requests include the token in the Authorization header; a custom JwtAuthFilter validates the token, extracts claims, and populates the Spring SecurityContext for downstream authorization.

B. Product Catalog Module

Products are persisted in the MySQL products table with fields for name, description, price, stock, and category. The React frontend fetches products via a public GET endpoint, supporting keyword search and multi-field sorting. A detailed product page displays full specifications with an Add to Cart action. Only ROLE_ADMIN users can create, update, or delete products via protected endpoints.

C. Cart Module

Cart state is managed on the React frontend using the Context API and persisted in local Storage for session continuity. Users can add items, adjust quantities, and remove entries, with real-time subtotal recalculation. On checkout initiation, cart contents are submitted to the backend along with the JWT for validation before Stripe payment intent creation.

D. Checkout and Payment Module

Checkout integrates Stripe Elements for secure card collection. The backend creates a Payment Intent via the Stripe SDK and returns a client secret to the frontend. The React Stripe.js library confirms the payment against Stripe servers. On success, the backend persists the order and order items, calculates totals, and returns an order confirmation to the user.

E. Order Management Module

Users can view their own order history through a ROLE_USER endpoint. Administrators access a ROLE_ADMIN endpoint that lists all orders with customer details. Order status can be updated by admins, supporting a basic fulfilment workflow. Contact messages submitted via the contact form are similarly restricted — customers submit, admins view.

V. IMPLEMENTATION

A. Technology Stack

The backend is developed in Java 21 with Spring Boot 3, Spring Security, Spring Data JPA, and the Stripe Java SDK. The frontend uses React 18 with Vite, Tailwind CSS, Axios, and Stripe.js. MySQL 8.0 (port 3307) serves as the relational store. Maven 3.9+ manages backend dependencies and build lifecycle. Development tools include IntelliJ IDEA, VS Code, Postman, Chrome DevTools, and Draw.io.

B. Authentication Flow

Registration accepts username, email, and password. The service layer validates uniqueness, hashes the password via BCryptPasswordEncoder, assigns ROLE_USER, and persists the customer entity. Login invokes Spring Security's AuthenticationManager; on success, JwtUtil generates a token signed with a secret key, expiring after 24 hours. All protected routes pass through JwtAuthFilter before reaching controllers.

C. Database Design

Core entities include Customer (id, email, password_hash, role), Product (id, name, description, price, stock), Order (id, customer_fk, total, status, created_at), OrderItem (id, order_fk, product_fk, quantity, unit_price), Address (id, customer_fk, street, city, state, postal_code, country), and Contact (id, name, email, message). Foreign key constraints enforce referential integrity.

D. Testing and Validation

Functional testing was conducted using Postman for all REST endpoints, verifying correct HTTP status codes, payload structures, and error responses. Security testing confirmed that endpoints return 401 Unauthorized without a valid JWT and 403 Forbidden when a ROLE_USER attempts an admin-scoped operation. Stripe test cards (4242 4242 4242 4242) validated end-to-end payment and order creation.

VI. RESULTS

The EazyStore platform was evaluated against its core objectives. JWT-based authentication and registration function correctly, with tokens validated on all protected routes. BCrypt hashing was confirmed via database inspection — no plaintext passwords are stored. Product CRUD operations are correctly restricted to admin roles; product browsing remains publicly accessible.

Cart operations (add, update, remove) perform with real-time total recalculation. Stripe payment integration processes test transactions successfully; orders are created and persisted upon payment confirmation. The admin panel correctly surfaces all orders and contact messages under role-restricted endpoints. System responsiveness is satisfactory for single-instance academic deployments.

VII. CONCLUSION AND FUTURE WORK

This paper has presented EazyStore, a secure full-stack e-commerce web application demonstrating cohesive integration of JWT authentication, BCrypt credential protection, RBAC, RESTful API design, and Stripe payment processing using Spring Boot and React. The system fulfils its objectives of security, usability, and modularity.

Future enhancements include a product review and ratings system, an AI-based recommendation engine, multi-vendor seller support, OAuth2 social login integration, Redis-based JWT token blacklisting for secure logout, and a React Native mobile application to extend the platform to iOS and Android users.

References.

[1] R. Wieruch, The Road to React. Self-published, 2023.

[2] M. Rana and A. Pandey, "Enhancing Data Security: A Study on JSON Web Token (JWT) and HMAC SHA-256 for Web Application Security," Int. J. Recent Innovation Trends Comput. Commun., 2024.

[3] T. P. Gbenle, "Applying OAuth2 and JWT Protocols in Securing Distributed API Gateways," Int. J. Mod. Res. Eng., 2025.

[4] M. Kokila et al., "Authentication, Access Control and Security Challenges in Modern Distributed Systems," ScienceDirect J. Inf. Secur., 2025.

[5] C. L. Aldea et al., "Authentication Challenges and Security Solutions in Microservice Architectures," MDPI Appl. Sci., 2025.

[6] F. Tanveer et al., "A Survey on RESTful API Vulnerability Detection and Security Mechanisms," ScienceDirect J. Syst. Softw., 2025.

[7] Y. Mao et al., "Research on API Security Gateway and Data Access Control in Multi-Tenant Systems," Preprints J., 2025.

[8] P. Rujichaikul, "Token-Based Authentication Monitoring System Using JWT and OAuth 2.0," J. Cyber Secur. Mobility, 2025.

[9] S. Dalimunthe, "RESTful API Security Using JSON Web Token (JWT) with HMAC-SHA512 Algorithm," Int. J. Web Eng., 2026.

[10] C. Walls, Spring Boot in Action, 2nd ed. Manning Publications, 2022.