



IJRASET

International Journal For Research in
Applied Science and Engineering Technology



INTERNATIONAL JOURNAL FOR RESEARCH

IN APPLIED SCIENCE & ENGINEERING TECHNOLOGY

Volume: 14 **Issue:** V **Month of publication:** May 2026

DOI: <https://doi.org/10.22214/ijraset.2026.82832>

www.ijraset.com

Call:  08813907089

E-mail ID: ijraset@gmail.com

Secure API Gateway Implementation with JWT for E-Commerce

Nujat Najir Mulani¹, Dr. Namrata Ghadgil²

¹Department of Master of Computer Application, JSPM University, Pune, Maharashtra, India

Abstract: In this study, we analyze the designing and development process of a safe e-commerce platform built on microservices architecture. As e-commerce platforms become increasingly important, there is an evident need for web apps which should be effective, safe, and easy to scale. At the same time, monolith applications present some obvious disadvantages due to increasing demand, such as complicated maintenance, difficulty in scaling, and security challenges. In order to cope with these problems, more and more people tend to implement microservices.

Firstly, the paper incorporates a number of new technologies within one architecture. Firstly, an API Gateway and JWT authentication ensure strict and solid security. The frontend is realized through React which ensures high-level interactivity. Moreover, Java Spring Boot works on the backend allowing to separate different services whereas Spring Cloud Gateway acts as a main gateway responsible for routing and security among clients and services. Besides, data storage is provided by MongoDB to ensure flexibility and scalability. In this way, the protection of APIs is performed through JSON Web Token (JWT) authentication ensuring users' credentials. Finally, to deploy this application, Docker is used to control containers, ensuring the work and communication among services.

In general, this study allows us to understand how to create a safe and scalable e-commerce platform with modern technologies.

Index Terms—API Gateway, JWT, Microservices, Spring Boot, React, MongoDB, Docker, E-Commerce Security.

I. INTRODUCTION

Modern e-commerce solutions represent an essential component of the operation of any business organization. Customers demand fast load speed, security, and high availability of the system. This type of software processes very sensitive data, such as personal log details, customer information, payment histories, etc., which means that the main concern associated with building this kind of web application is providing a strong protection mechanism.

Most e-commerce applications have a monolithic architecture that integrates all major components such as users' administration, product catalogue, payment processing, and order processing into one big application. Although this model can work well for small scale applications, it does not allow scaling or modifying the app in future.

However, moving to microservice architecture changes the situation. The software is divided into services responsible for particular tasks. This approach allows better scalability and easier development but also involves the problem of inter-service interaction.

A central entrance point for client requests is API Gateway that handles routing, authorization, logging, and monitoring. In this case, we integrate our API Gateway with JWT to verify a user before accessing other services.

The aim of the present research is to develop a secure and scalable solution for an e-commerce site that uses React for the frontend, Spring Boot microservices on the backend, Spring Cloud Gateway, JWT, MongoDB, and Docker.

II. LITERATURE REVIEW

Numerous studies emphasize the significance of security at the level of the Application Programming Interface (API). With microservices architecture, the use of API Gateways is widely spread as they simplify communication between the front- and back-end servers.

Another interesting topic, that should be mentioned, is JWT authentication. This method of identification of the user is widely used for REST APIs due to its simplicity compared to session-based authentication, as well as absence of necessity in storage of session information on the side of the server.

Speaking of securing enterprise applications, such libraries as Spring Security and Spring Cloud Gateway are commonly used as they provide not only identification via tokens but also route management, filtering of requests, and authorization of the user.

Finally, Docker and other container tools revolutionized application development, as now the developer is guaranteed to have no problems of transferring an application from testing to production environment.

Considering the abovementioned facts, one can suggest that API Gateway, JWT, microservices, and containerization will be a good choice in case of e-commerce development. Such combination has been proven effective by numerous studies.

III. PROBLEM STATEMENT

The typical web-based business application faces a number of problems simultaneously, such as:

- Challenges in scaling when facing high levels of traffic
- Security risks that come with using APIs directly
- Challenges in maintenance due to tight module coupling
- Slow deployment process
- Limited flexibility for future changes

As a result, it is necessary to design a flexible, secure system which will be able to handle traffic effectively while keeping APIs and user data safe.

IV. OBJECTIVES

Objectives of the project include the following:

- To develop an API Gateway with security capabilities for an e-commerce application.
- To create independent microservices by using Spring Boot.
- To integrate authentication and authorisation by using JSON Web Tokens (JWT).
- To develop a dynamic user interface by employing React.
- To use MongoDB to store the application's data.
- To package the application's services using Docker containers.
- To improve security, scalability, and maintainability.

V. SYSTEM ARCHITECTURE

This project consists of four levels of hierarchy, each having different functional responsibilities.

A. Front End Level

This level is designed using React and includes user interface for:

- Home Page
- Product Viewing
- Cart Management
- Registration & Login
- Order History
- Admin Dashboard

B. API Gateway Level

Spring Cloud Gateway acts as the entry point for handling all requests and is responsible for:

- Request Routing
- Token Validation
- Filtering
- Security
- Load Balancing

C. Micro-services Level

The back end layer consists of multiple independent micro-services such as:

- **User Service:** This service takes care of user registration, authentication and their data
- **Product Service:** This service controls the product details and inventory
- **Order Service:** Handles all the customer order processing
- **Payment Service:** Responsible for payment processing

D. Data Layer

MongoDB database contains the data domains of:

- UserDomain
- ProductDomain
- OrdersDomain
- PaymentDomain

E. Execution Layer

All the microservices are packed and deployed as Docker containers.

VI. METHODOLOGY

Step1: Front-end Development

React framework is used to develop a dynamic front end that allows users to view their products, add items to the cart, place orders, and manage account details.

Step2: Back-end Development

A different Spring Boot microservice is developed for each business function.

Step3: Authentication

Users log in using their usernames and passwords to receive JSON Web Tokens on successful authentication.

Step4: API Security

The client sends a JSON web token in the Authorization header in all secure requests, which is verified by the API Gateway before sending a request.

Step5: Database Management

Data is stored in MongoDB, which uses JSON-like documents, thus making e-commerce possible.

Step6: Containers

Containers are created for all components using Docker to enable hassle-free deployment.

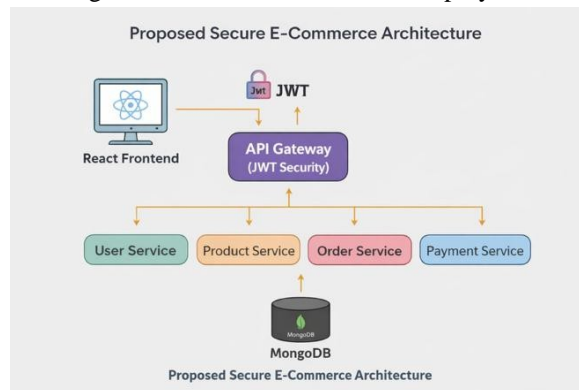


Fig1. Proposed Secure E-Commerce Architecture

VII. IMPLEMENTATION

A. Technologies Used

Components	Technology
Frontend	React.js
Backend	SpringBoot
Security	JWT
Gateway	SpringCloudGateway
Database	MongoDB
Containerization	Docker
APITesting	Postman

B. SecurityProcessFlow

Step1:Usercredentialsubmission

Step2:Authenticationofusercredentialsubmission

Step3:GenerationofJWTtokenincaseofsuccessful authentication

Step4:Storageofthetokenattheclient-side

Step5:Inclusionoftokenbytheclientinall subsequent requests

Step6:Validationoftokenbythegateway

Step7:Requestroutingtocorrespondingservices

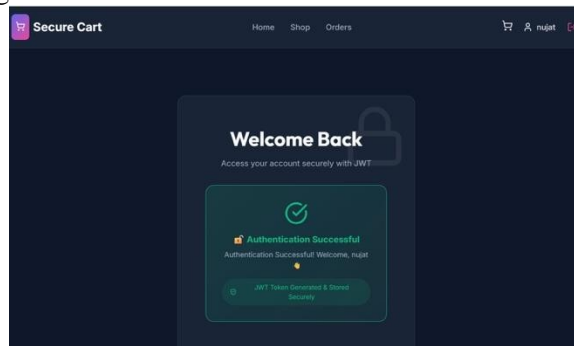


Fig2.UserLoginInterface

VIII. RESULT AND OUTPUT

Capabilitiesofthesysteminclude:

- Userauthentication
- Role-basedaccessforadminsandusers
- Abilitytolistandsearchforproducts
- Shoppingcartmanagement
- Orderingprocessmanagement
- Securityfromanyformofunauthorizedaccess
- ProtectionoftheAPIusingJSONwebtokens(JWT)
- Aresponsivefrontend
- EfficientroutingthroughtheuseofanAPIGateway

Moreover,theadminpanelallowsforthemanagement of products and user rights, whereas the customer-facing platform enables safe shopping and order management.

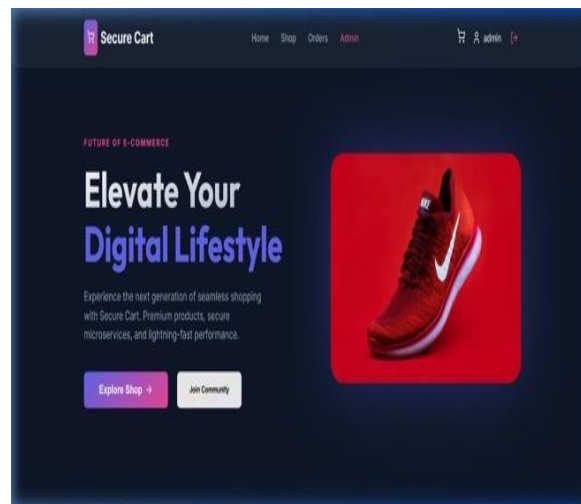


Fig3.HomePage

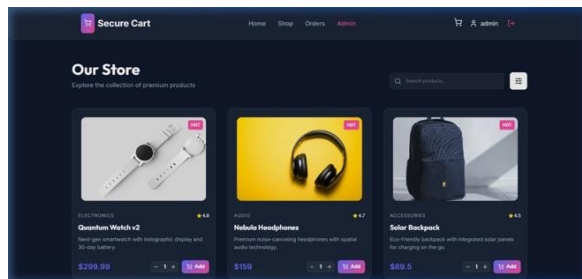


Fig4.ProductListingModule

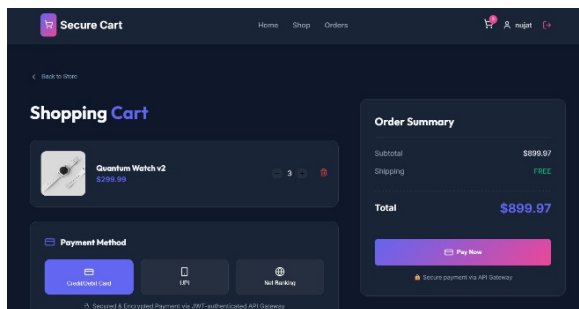


Fig5.Cart&Checkout

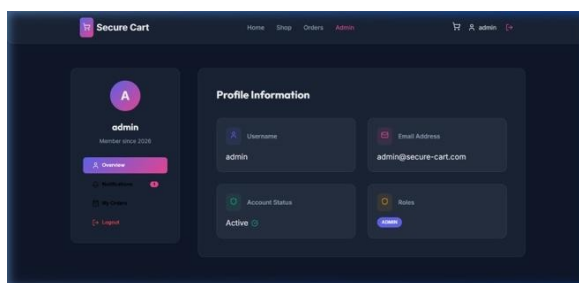


Fig6.AdminDashboard

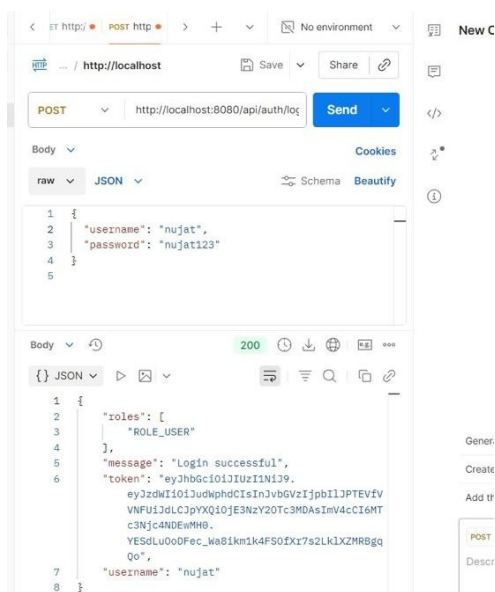


Fig7.APITestingwithJWTToken

IX. ADVANTAGES OF PROPOSED SYSTEM

- High security through JWT authentication
- Centralized API management
- Easy scalability using microservices
- Independent service deployment
- Better fault isolation
- Faster development and updates
- Modern responsive user experience
- Simplified deployment with Docker

X. FUTURE SCOPE

The system can be enhanced further by adding:

- Payment gateway integration
- Recommendation engine using AI
- Real-time notifications
- Kubernetes deployment
- Multi-vendor marketplace support
- Analytics dashboard
- OAuth2 and social login
- Advanced monitoring tools

XI. CONCLUSION

In this research, the successful implementation of an e-commerce application has been illustrated that utilizes the latest web technologies in terms of security and other performance attributes. The use of ReactJS frontend, microservices in Spring Boot, API gateway in Spring Cloud Gateway, MongoDB, JSON Web Token for security of APIs, and Docker make the platform very secure, scalable, and flexible as well.

The API Gateway provides a single place for route management and authentication purposes, whereas JSON Web Tokens ensure that access to protected APIs is secured.

XII. ACKNOWLEDGMENT

I want to say thank you to everyone who helped with the research project called Implementation of Secure API Gateway using JWT Authentication for an E-commerce Portal.

I am really thankful to Dr. Namrata Gadgil because she guided me and gave me suggestions at every stage of the Implementation of Secure API Gateway using JWT Authentication for an E-commerce Portal project.

She helped me a lot with the part and the documentation part of the Implementation of Secure API Gateway using JWT Authentication for an E-commerce Portal project.

I also want to thank the teachers at the department of computer applications at JSPM university in Pune.

They helped me a lot, with my studies. Gave me the support and facilities I needed to do the Implementation of Secure API Gateway using JWT Authentication for an E-commerce Portal project.

My friends and colleagues were also very helpful.

They gave me suggestions when we talked about the Implementation of Secure API Gateway using JWT Authentication for an E-commerce Portal project and that helped me a lot.

I also want to thank my family for encouraging me and motivating me to do the Implementation of Secure API Gateway using JWT Authentication for an E-commerce Portal project.

REFERENCES

- [1] S. Newman, Building Microservices: Designing Fine-Grained Systems. Sebastopol, CA, USA: O'Reilly Media, 2021.
- [2] C. Richardson, Microservices Patterns: With Examples in Java. Shelter Island, NY, USA: Manning Publications, 2018.
- [3] Spring Team, "Spring Boot Documentation," VMware, Inc. [Online]. Available: <https://spring.io/projects/spring-boot>



- [4] Spring Team, "Spring Cloud Gateway Documentation," VMware, Inc. [Online]. Available: <https://spring.io/projects/spring-cloud-gateway>
- [5] Auth0, "Introduction to JSON Web Tokens (JWT)." [Online]. Available: <https://jwt.io/introduction>
- [6] MongoDB Inc., "MongoDB Official Documentation." [Online]. Available: <https://www.mongodb.com/docs/>
- [7] DockerInc., "DockerDocumentation." [Online]. Available: <https://docs.docker.com/>
- [8] Meta Platforms Inc., "React Official Documentation." [Online]. Available: <https://react.dev/>
- [9] O.Tilkov and S.Vinoski, "Node.js: Using JavaScript to Build High-Performance Network Programs," IEEE Internet Computing, vol.14, no.6, pp.80-83, Nov.-Dec.2010.
- [10] M. Fowler and J. Lewis, "Microservices: A Definition of This New Architectural Term," Martin Fowler, 2014. [Online]. Available: <https://martinfowler.com/articles/microservices.html>



10.22214/IJRASET



45.98



IMPACT FACTOR:
7.129



IMPACT FACTOR:
7.429



INTERNATIONAL JOURNAL FOR RESEARCH

IN APPLIED SCIENCE & ENGINEERING TECHNOLOGY

Call : 08813907089  (24*7 Support on Whatsapp)