Building a Scalable Multi-Vendor E-Commerce Platform with AWS and Modern Web Technologies

Project Report

Ayush Sahu (e22cseu1049@bennett.edu.in)

A comprehensive overview of a multi-vendor e-commerce platform leveraging AWS, React, and Node.js for scalability and performance.

1 Introduction

In this write-up, I share insights from my recent work on creating a dynamic and scalable e-commerce platform. Designed for multi-vendor functionality, the platform brings buyers and sellers together in a unified digital marketplace. By leveraging various AWS services, I ensured reliability, performance, and secure operations.

2 Project Overview

The goal was to develop a full-fledged e-commerce system capable of handling user roles, managing products, processing orders, and visualizing analytics. Below are the major components and functionalities integrated into the platform:

- User authentication with buyer/seller roles
- Product listings with search, filter, and categorization
- Cart, checkout, and order tracking
- Seller management dashboard
- Integration with payment systems
- Business intelligence and reporting

3 Technology Stack

Frontend: React.js + TypeScript, Tailwind CSS, Context API, Framer Motion

Backend: Node.js + Express, PostgreSQL with Prisma ORM, AWS Cognito **Infra:**

AWS Cloud, Docker, GitHub Actions, CloudFormation

4 Architecture Highlights

The architecture follows a microservices model with AWS integration:

- Elastic Beanstalk for deployment
- RDS for PostgreSQL
- Cognito for user access
- S3 and CloudFront for assets
- EventBridge, SQS for async messaging
- Redshift and QuickSight for analytics

5 CI/CD Pipeline

A robust CI/CD pipeline ensures smooth development-to-deployment workflows. Automated testing, code quality analysis, and blue/green deployments help minimize downtime and maintain consistency across environments.

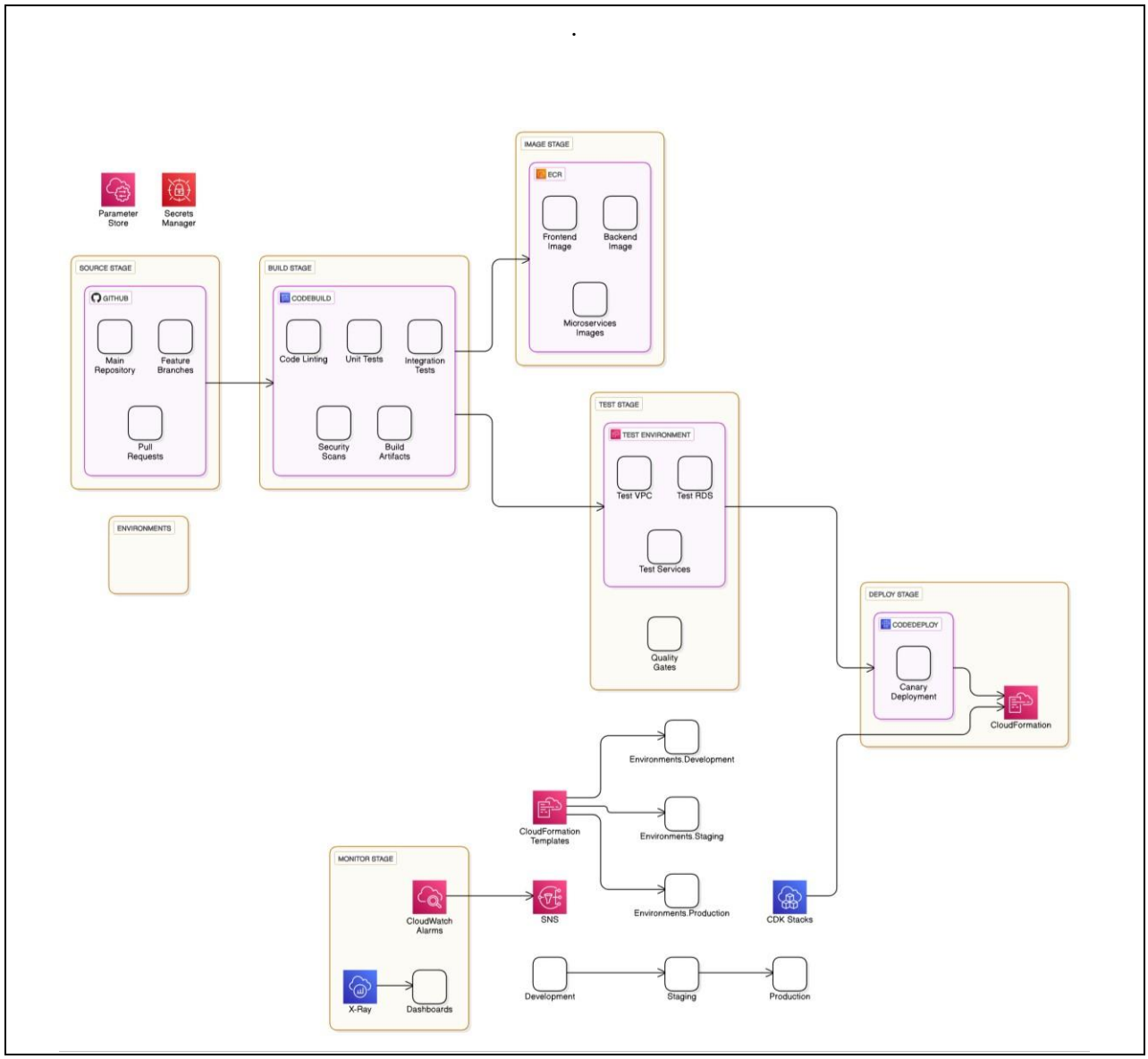

Figure 1: CI/CD Pipeline Configuration

6 Deployment Strategy

Deployment is containerized using Docker and managed via Elastic Beanstalk. The strategy supports horizontal scaling and reliable version rollbacks.

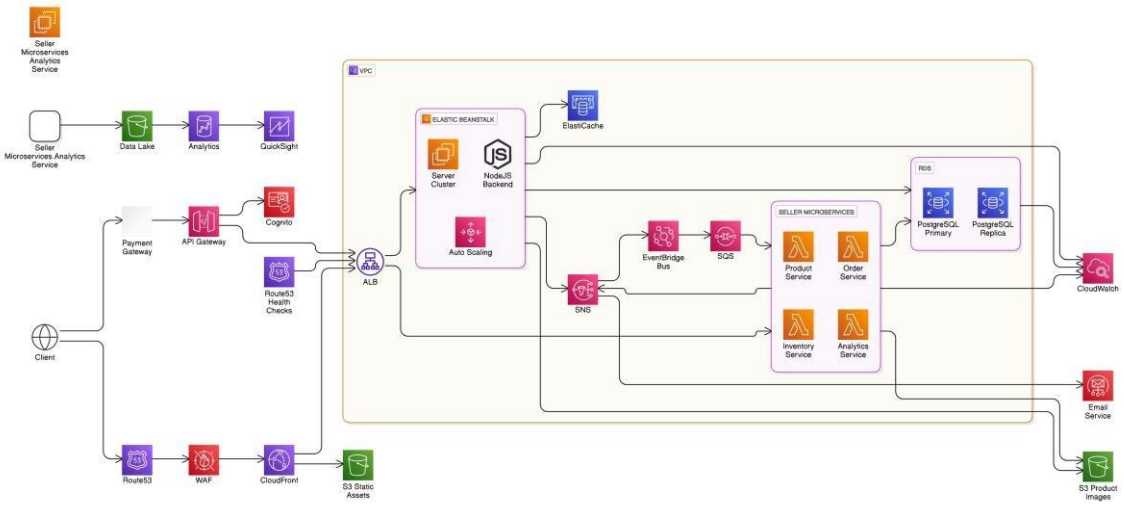

Figure 2: Deployment Strategy Diagram

7 Database Schema

The PostgreSQL database uses a normalized relational structure, mapping out users, products, orders, and transactional relationships efficiently. Optimized queries support the platform's responsiveness and reporting capabilities.

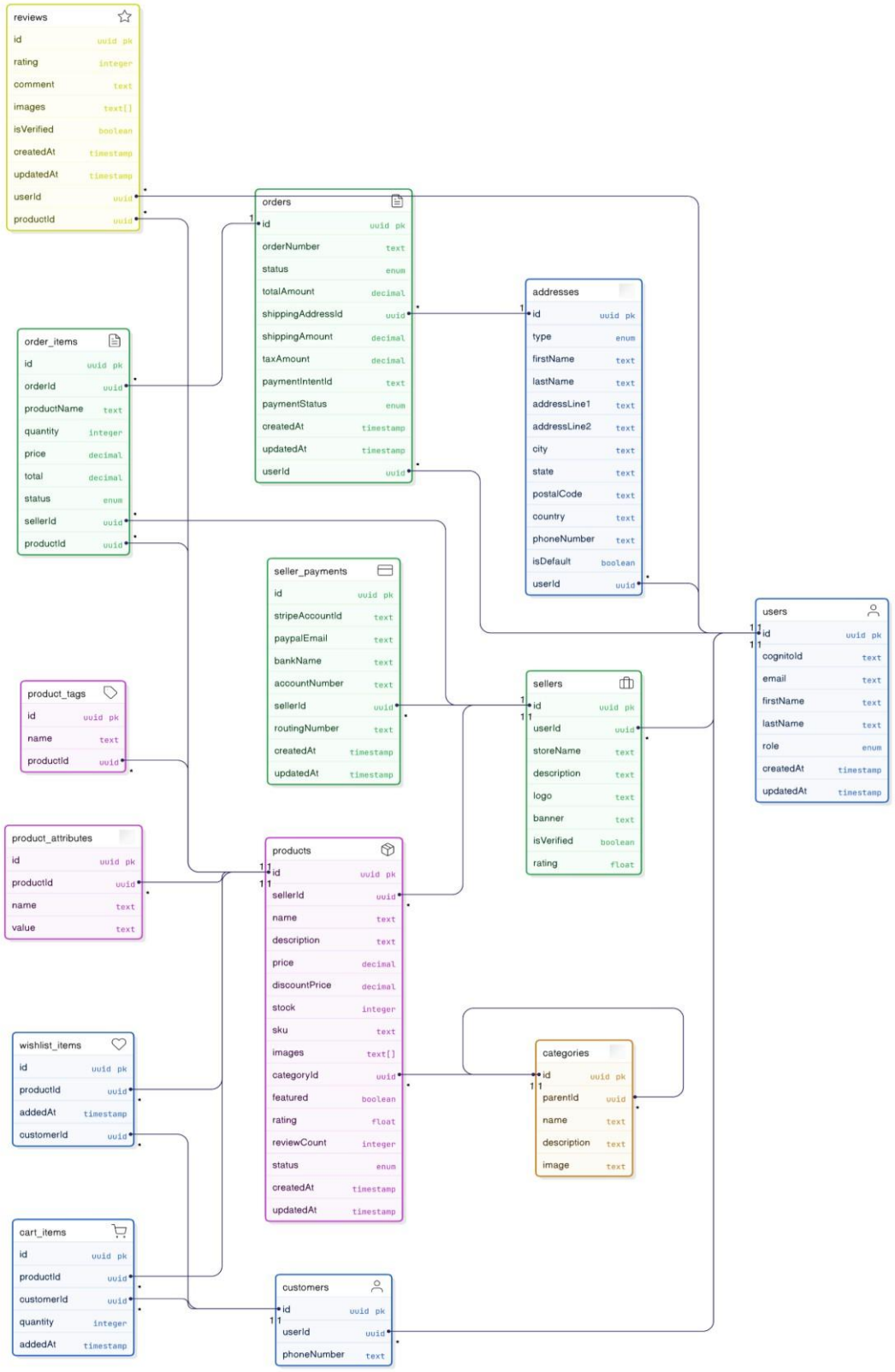

Figure 3: Database Schema

8 Challenges & Solutions

- Integrating AWS Cognitos complex authentication flows
    - Coordinating order flows between multiple vendors
    - Synchronizing deployments across staging and production

9 Project Outcomes

- Live deployment of a scalable, feature-rich platform
- Smooth UX across mobile and desktop
- Secured payments and multi-vendor support

10 Future Enhancements

- Machine learning for product recommendations
- Advanced analytics dashboard
- Mobile application development
- International shipping and multi-currency support

11 Conclusion

This multi-vendor e-commerce platform demonstrates the implementation of modern web technologies and cloud architecture to create a scalable, secure, and user-friendly online marketplace. The project showcases expertise in full-stack development, cloud infrastructure, and best practices in software engineering.

12 Deployment and Application Screenshots

This section provides visual insights into the deployment environment, configurations, and application interfaces of the e-commerce platform.

12.1 Elastic Beanstalk Deployment Environment

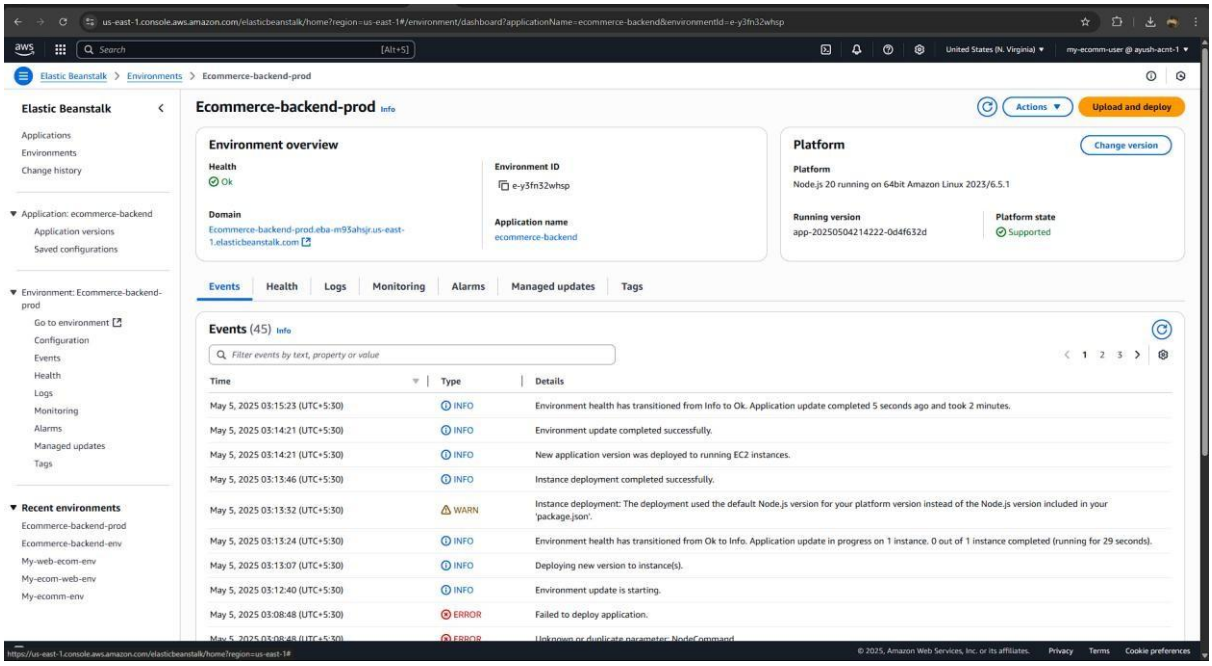

Figure 4: Elastic Beanstalk deployment environment showing health status and configuration

12.2 PostgreSQL Database Instance

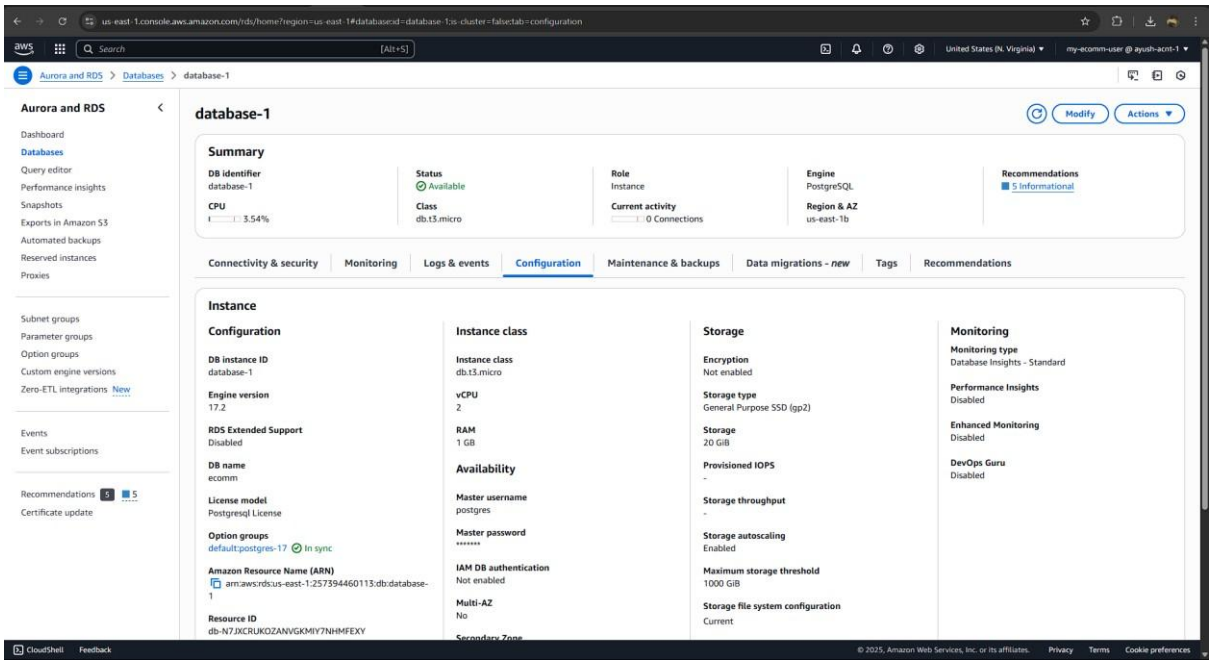

Figure 5: PostgreSQL database instance with multi-AZ configuration

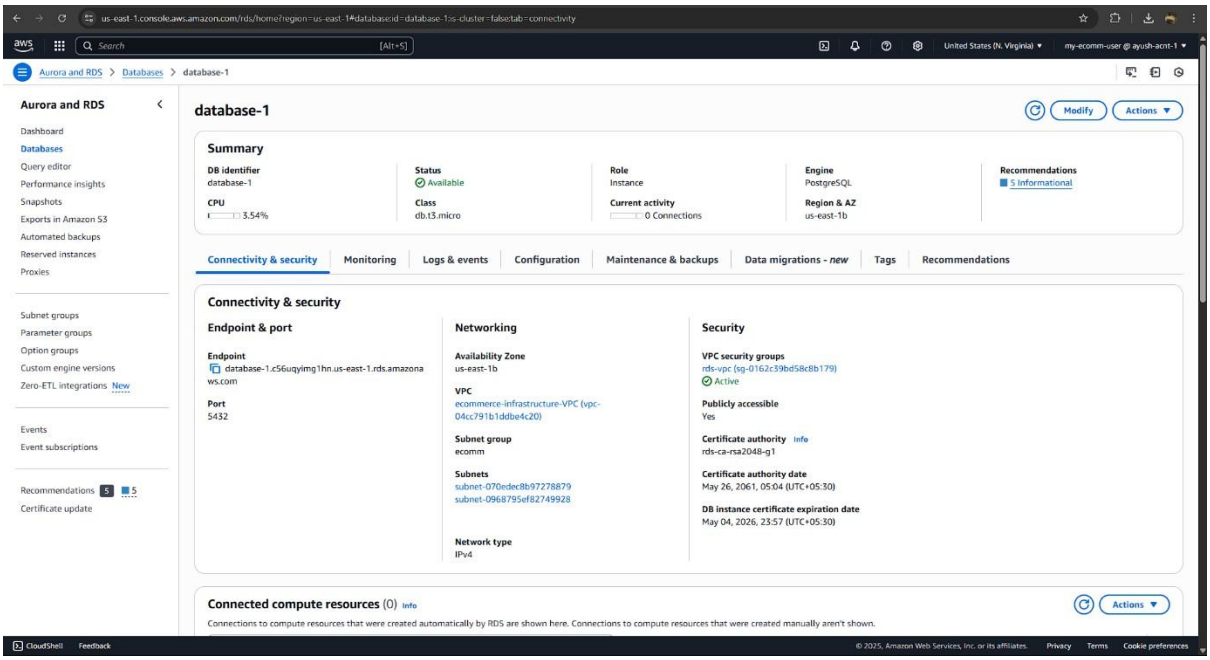

Figure 6: Additional PostgreSQL configuration details

12.3 Cognito User Authentication Service

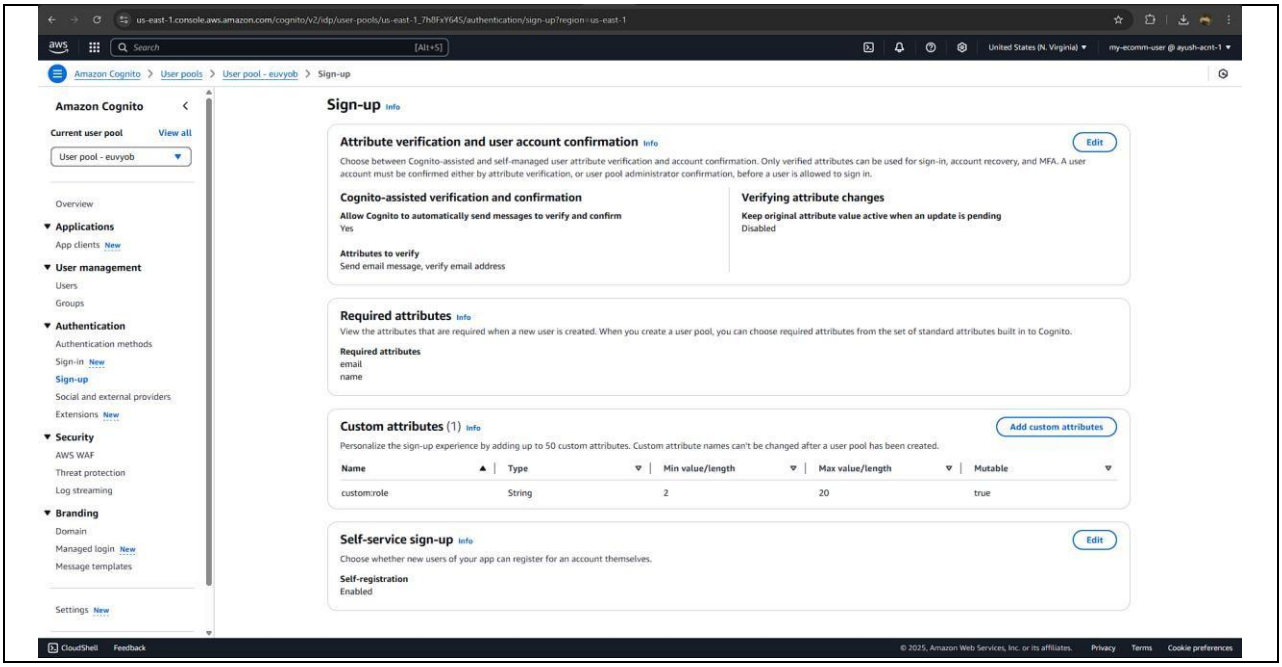

Figure 7: Cognito user authentication service with OAuth2 configuration

12.4 S3 Storage

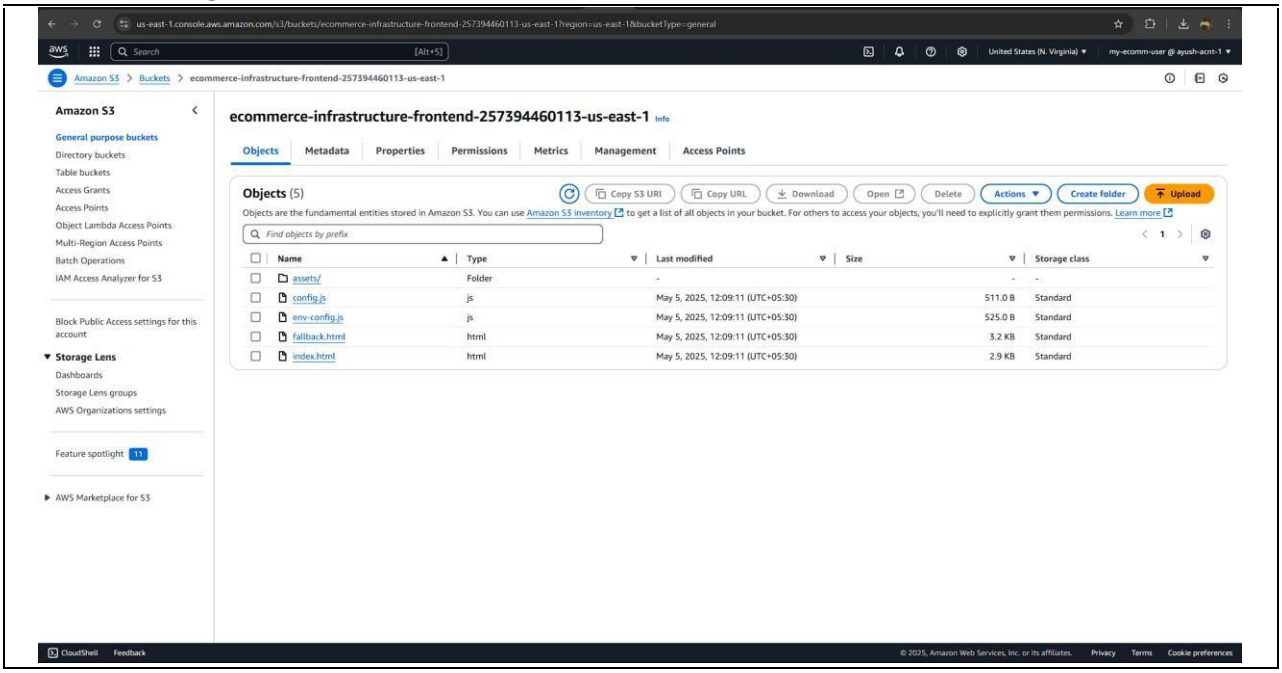

Figure 8: S3 storage for static assets and product images

12.5 Authentication Flow

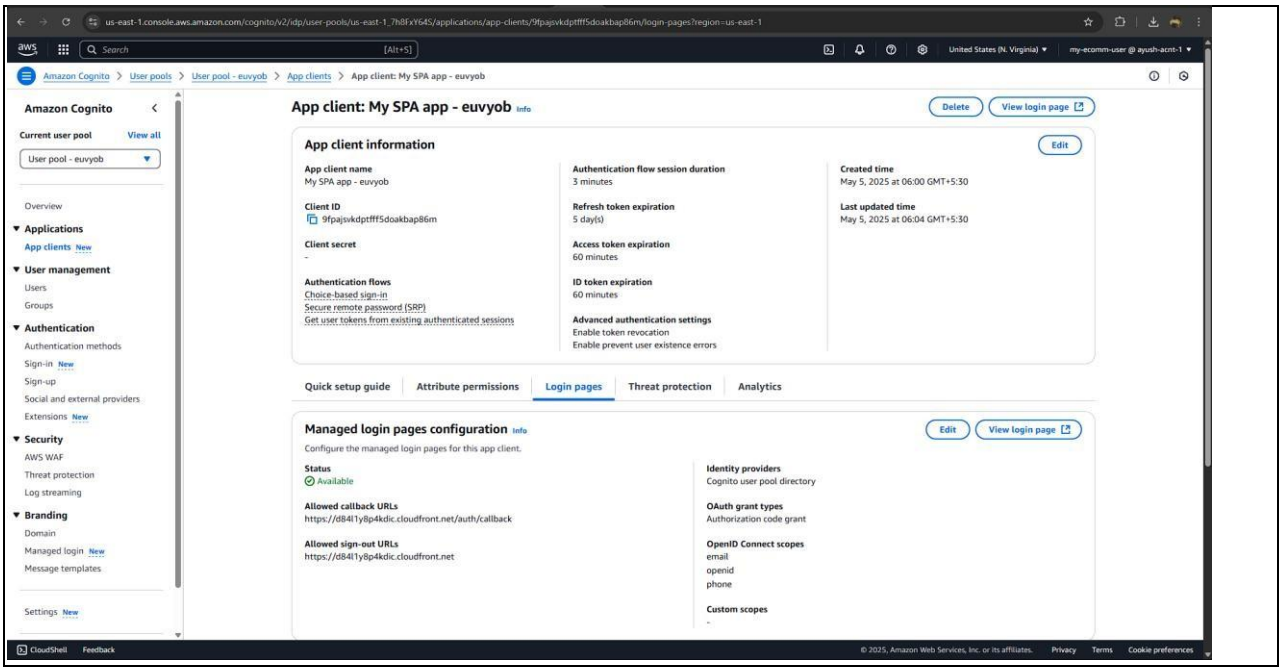

Figure 9: Authentication flow diagram

12.6 CDN Configuration

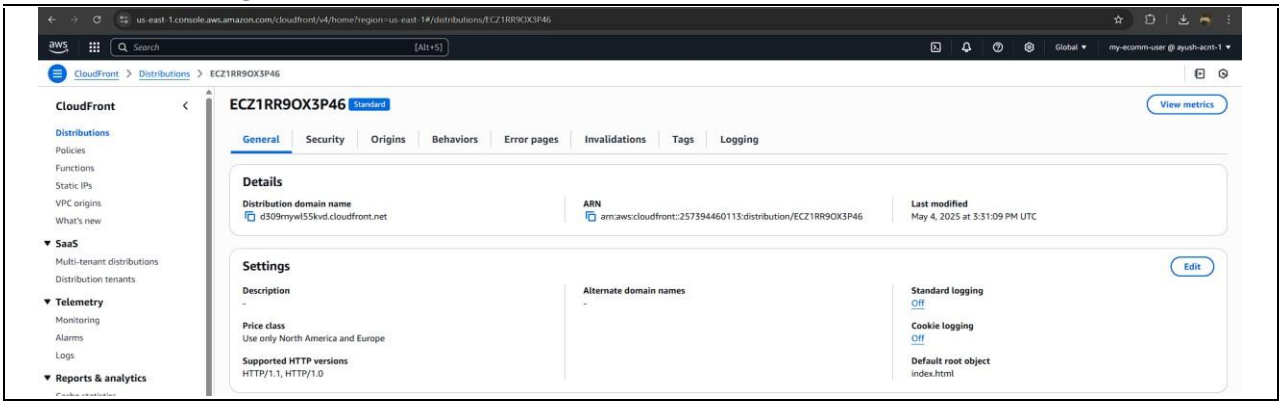

Figure 10: CDN configuration for global content delivery

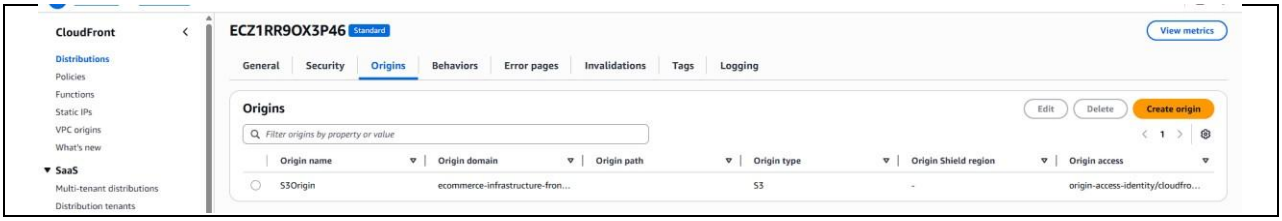

Figure 11: Additional CDN configuration details

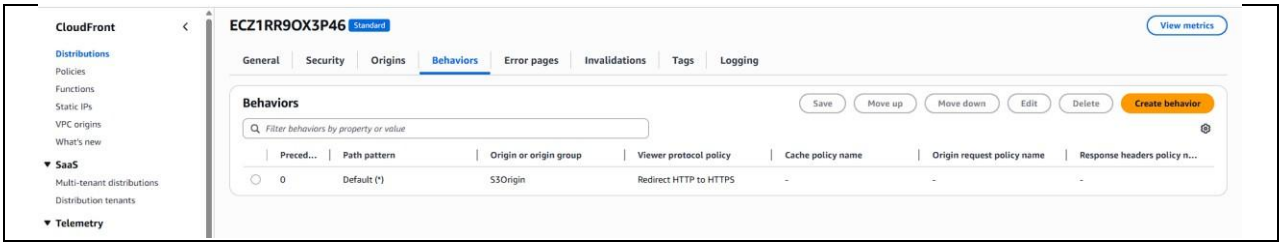

Figure 12: Further CDN configuration insights

12.7 Container Registry

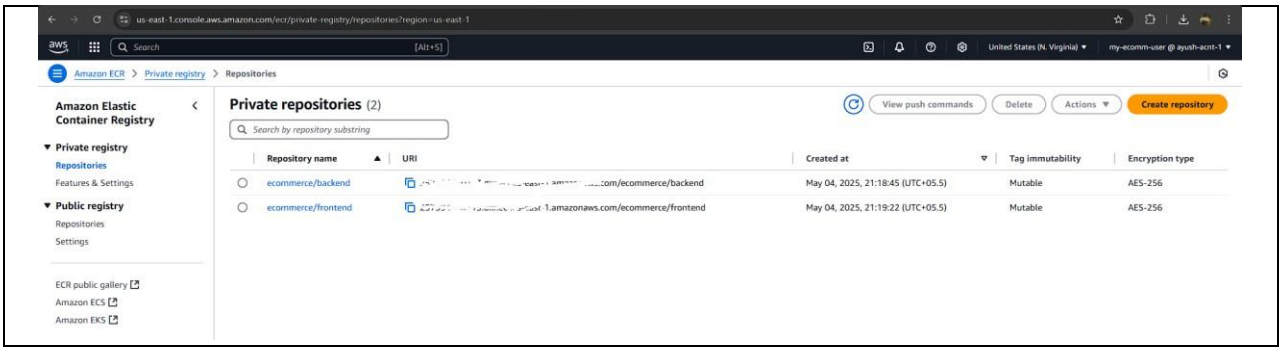

Figure 13: Container registry storing Docker images for deployment

12.8 CloudFormation (IaC)

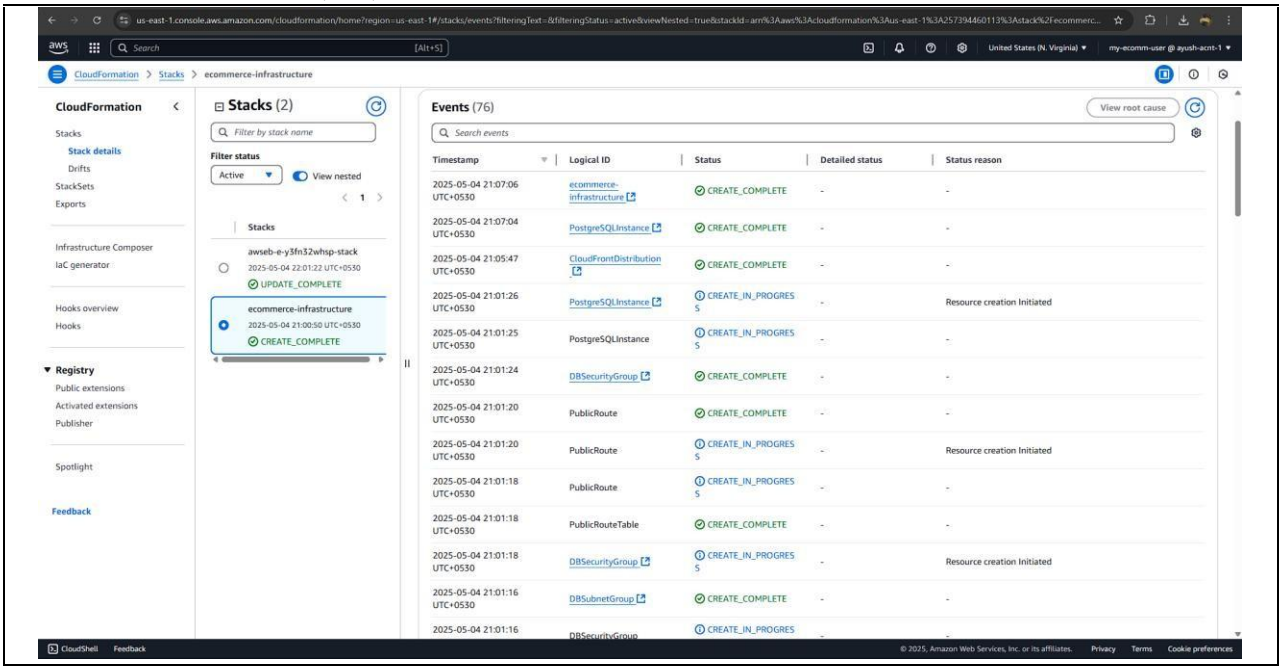

Figure 14: CloudFormation infrastructure as code configuration

12.9 CI/CD Pipeline

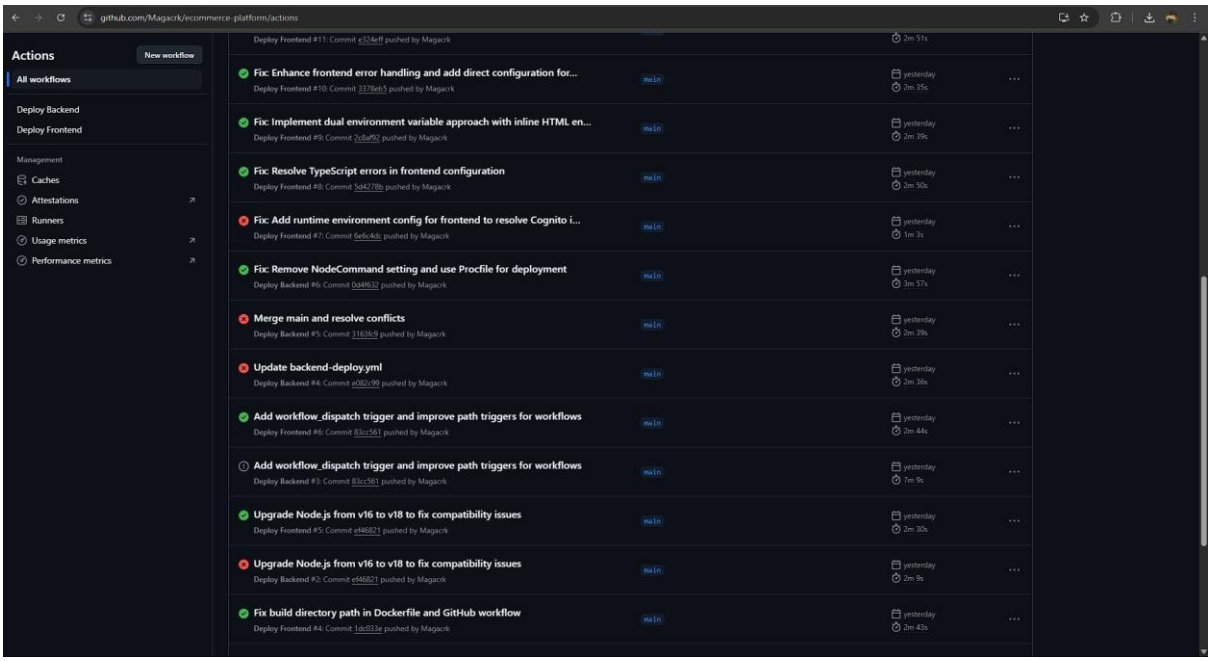

Figure 15: CI/CD pipeline configuration for automated testing and deployment

12.10 Application Screenshots

12.10.1 Dashboard

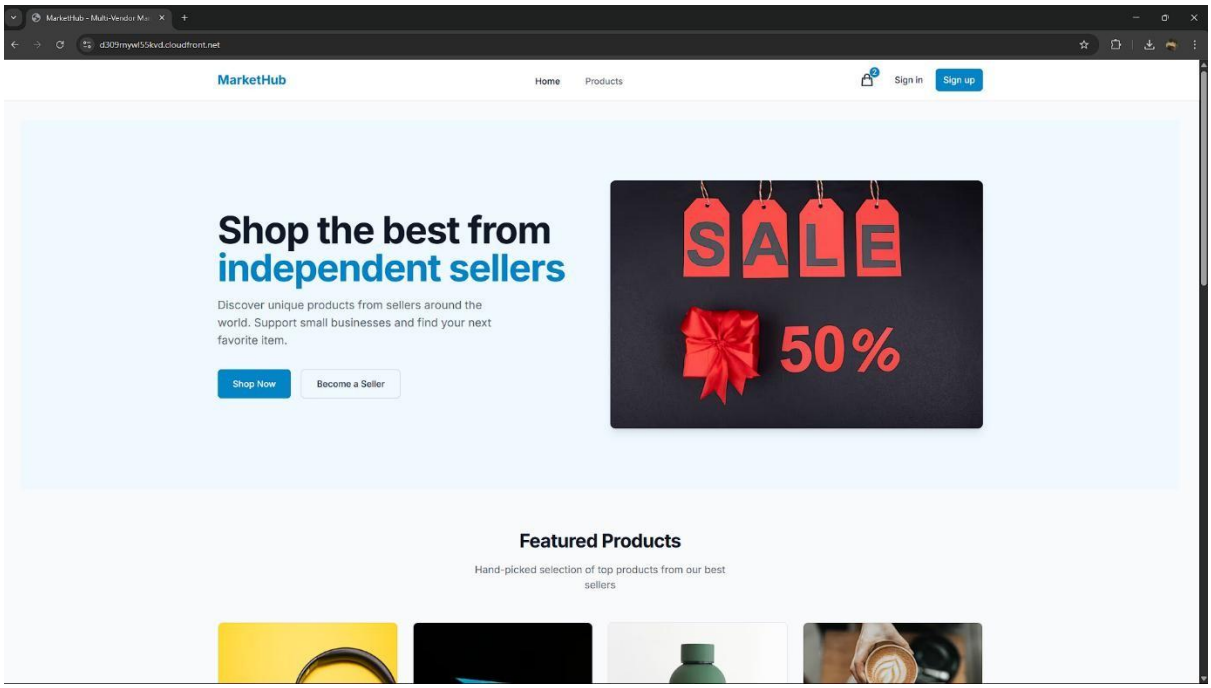

Figure 16: Dashboard page for users

12.10.2 Seller Sign-in

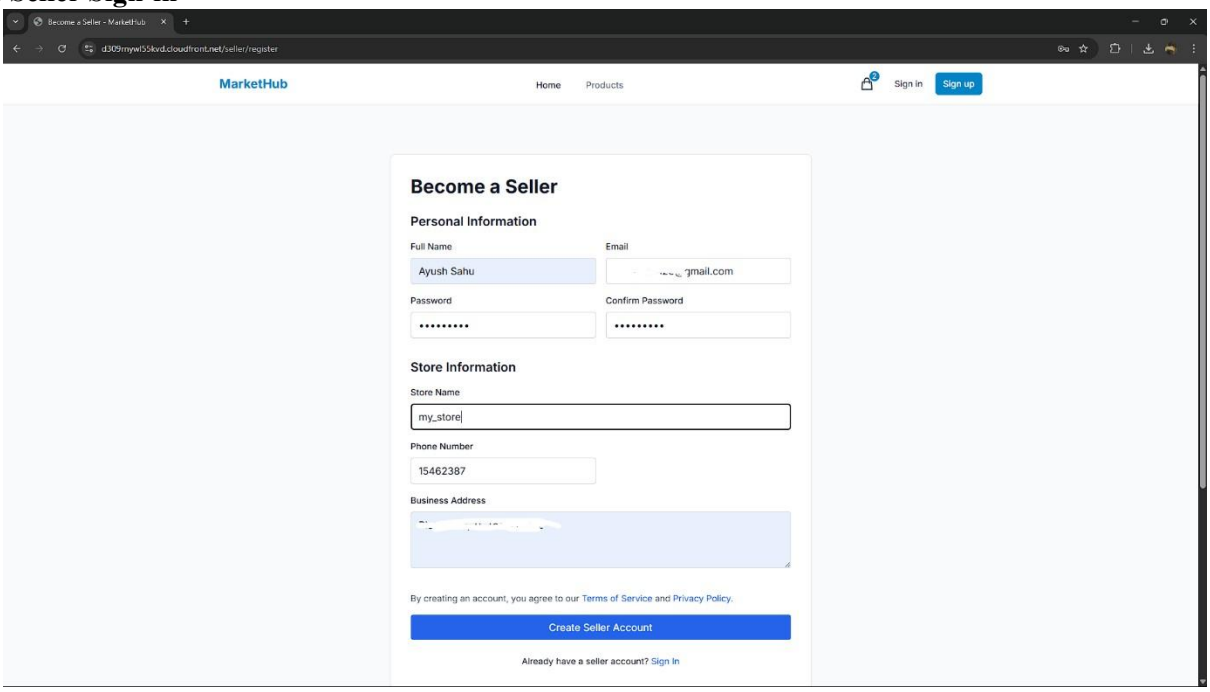

Figure 17: Seller sign-in interface

12.10.3 Seller Dashboard

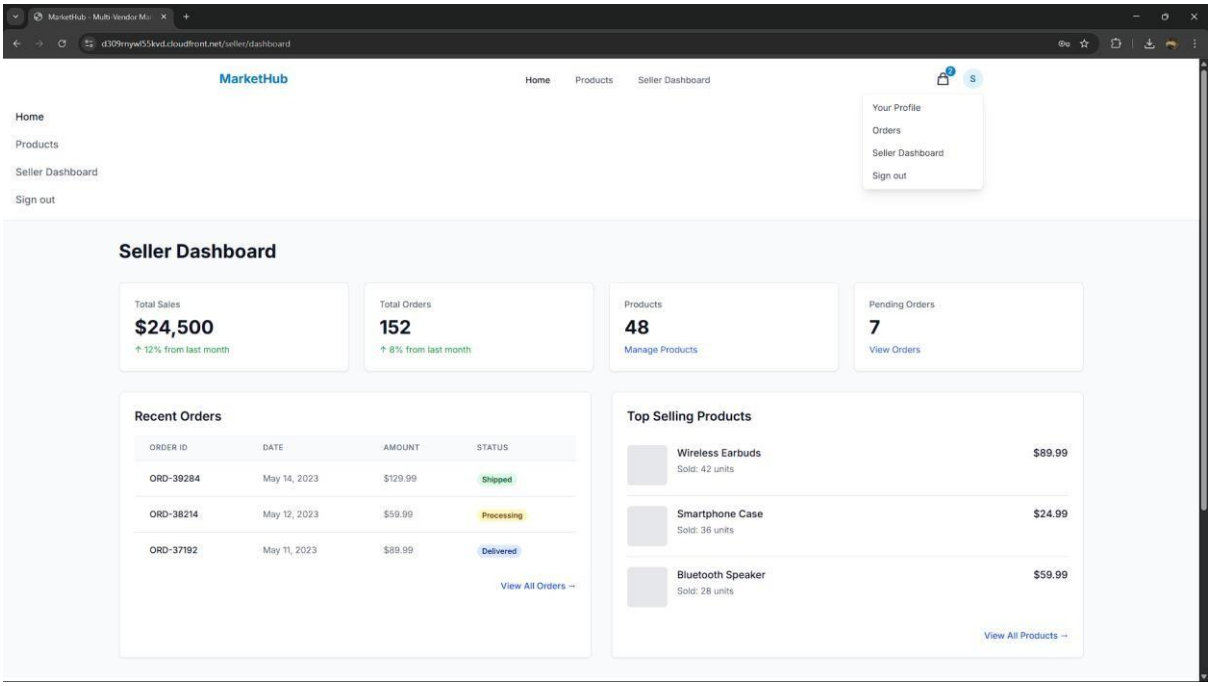

Figure 18: Seller management dashboard

12.10.4 Buyer Dashboard

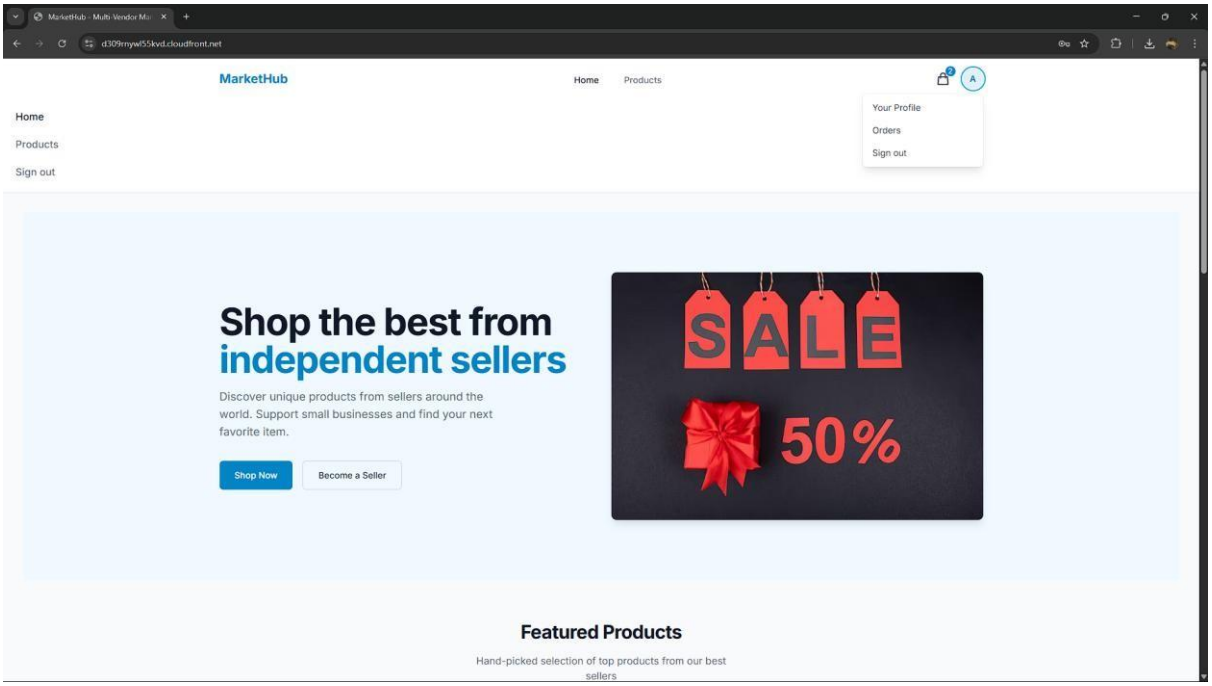

Figure 19: Buyer dashboard interface

12.10.5 Cart and Checkout

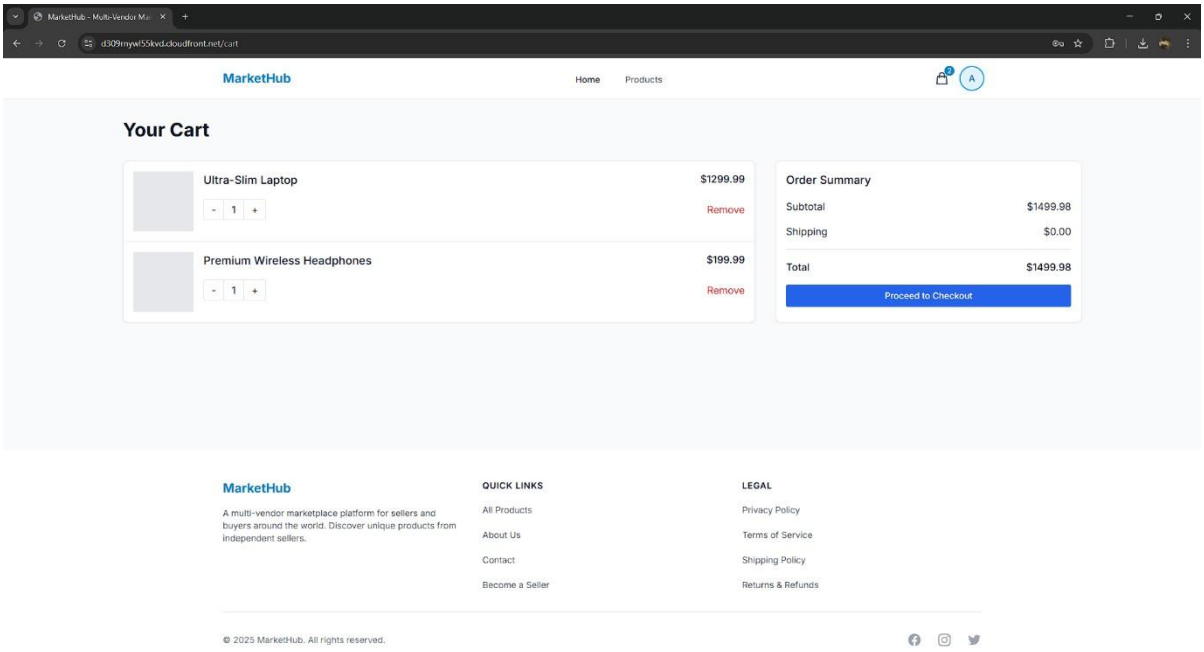

Figure 20: Cart interface

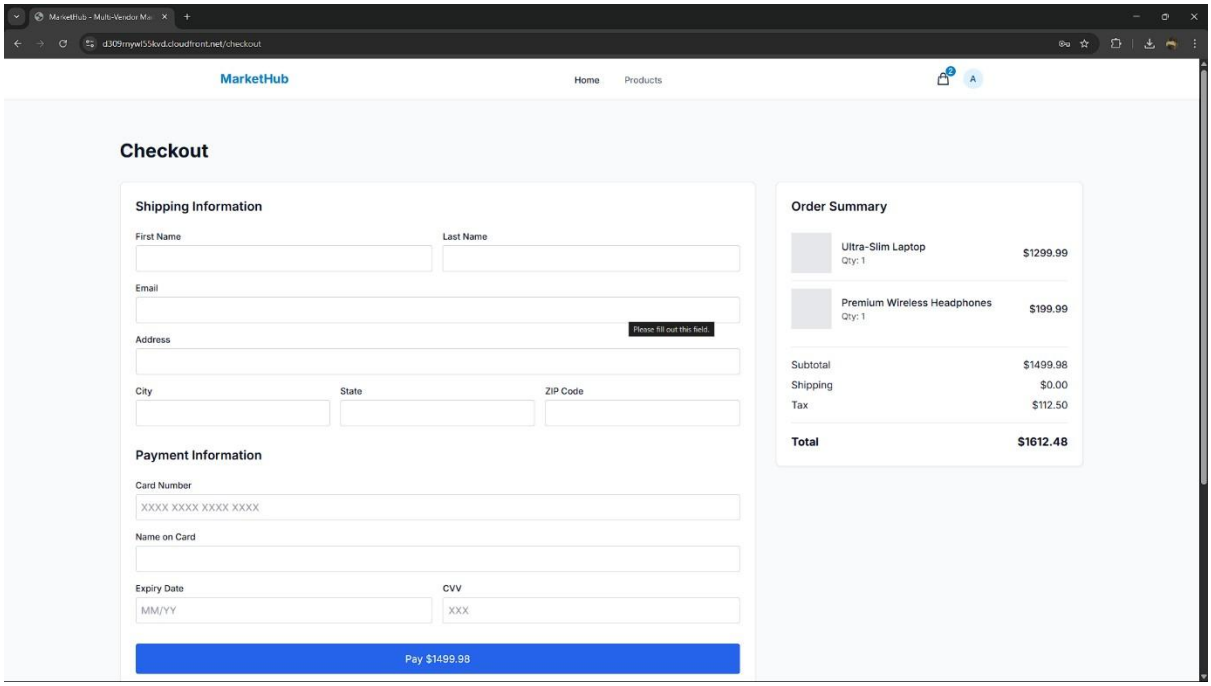

Figure 21: Checkout interface