MyShop: A Case Study in Engineering a Scalable E-Commerce Platform with Integrated Hybrid Recommender System

1. Project Overview & Research Perspective

This documentation presents "MyShop," a comprehensive project developed as a practical engineering case study. The primary objective was to investigate, design, and implement a production-aware software architecture that successfully integrates a complex machine learning component—a hybrid recommender system—into a full-stack web application. The project serves as a tangible exploration of the challenges and trade-offs involved in building scalable, intelligent systems, making it a relevant portfolio piece for research-oriented graduate studies in Computer Science and Engineering.

2. System Architecture & Core Design Philosophy

The system is built on a decoupled microservices architecture, a foundational decision to address the core challenge of modularity and independent evolution. This architecture cleanly separates the main application logic from the machine learning service, enabling:

· Independent Scalability: The recommendation engine can be scaled based on AI workload, independent of web traffic.

· Technology Flexibility: The use of Python for ML and Node.js for the web backend allows employing the best tool for each domain.

· Simplified Maintenance & Testing: Each service can be developed, updated, and validated in isolation.

3. Engineering the Hybrid Recommender System: Design Decisions & Trade-offs

The recommender system is the centerpiece of this case study, exemplifying applied AI integration.

· 3.1. Architectural Integration: The primary engineering challenge was seamlessly integrating a Python-based ML model into a JavaScript/Node.js ecosystem. This was solved by deploying the recommender as an independent microservice (FastAPI), communicating with the main backend via well-defined RESTful APIs.

· 3.2. Hybrid Model Design: The system implements a weighted hybrid approach, combining:

  · Content-Based Filtering (70% weight): Utilizes the sentence-transformers/all-MiniLM-L6-v2 model to generate semantic text embeddings from product titles and descriptions.

· Image-Based Recommendations (30% weight): Employs the CLIP model to generate visual embeddings from product images, capturing similarities in appearance, style, and color.

· Rationale for 70/30 Weighting: This weighting reflects a design hypothesis prioritizing textual semantic relevance—crucial for e-commerce—while effectively supplementing it with visual similarity to enhance diversity and user discovery.

· 3.3. Performance Optimization - The Accuracy/Latency Trade-off: To ensure real-time recommendations, we faced the classic trade-off between result accuracy and system latency. The choice of FAISS (Facebook AI Similarity Search) for approximate nearest neighbor search was critical. This library provides sub-second query latency (measured at <500ms on free-tier infrastructure) by trading off exact search for immense speed gains, a necessary compromise for user-facing features.

4. Backend Design: Building for Maintainability and Scale

The backend follows established patterns to ensure robustness and long-term maintainability, crucial for any system with research iteration potential.

· Modular Structure: The codebase is organized into logical layers (models, controllers, routes, middleware), enforcing separation of concerns. For instance, business logic resides in controllers, separate from data schemas (Mongoose models) and API route definitions.

· Key Components:

· Authentication & Security: Implements stateless JWT-based authentication with refresh tokens, managed through dedicated middleware.

· Error Handling: A centralized error-handling middleware ensures consistent API error responses and simplifies debugging.

· Data Modeling: Mongoose schemas for Users, Products, and Orders enforce data integrity and provide a structured interface to MongoDB.

5. Frontend Design: State Management & User Experience

The React-based frontend is designed for a responsive and dynamic user experience, with state complexity managed effectively.

· Component-Based Architecture: Reusable UI components (e.g., Product.jsx, Header.jsx) promote consistency and reduce code duplication.

· State Management with Redux Toolkit: Global application state (user authentication, shopping cart, product listings) is managed centrally using Redux Toolkit. This includes using RTK Query for efficient

data fetching, caching, and synchronization with the backend, ensuring a scalable state management strategy.

6. Deployment & Production Considerations

The entire system is deployed on Render's free tier, demonstrating an understanding of cloud deployment constraints. Key considerations included:

· Configuring environment variables for sensitive data.

· Managing the lifecycle and communication between separate services (web service, ML service, MongoDB Atlas database).

· Understanding the performance implications of resource limits on a free-tier platform.

7. Conclusion, Limitations & Future Research Directions

7.1. Conclusion of the Case Study

This project successfully demonstrates the feasibility and challenges of integrating a production-ready AI component into a modern web application stack. It validates the microservice architecture as an effective pattern for such integrations and provides a hands-on exploration of key engineering trade-offs (modularity vs. complexity, accuracy vs. latency).

7.2. Acknowledged Limitations & Scope

As a focused case study, the current system has intentional and unintentional limitations that define its scope and create opportunities for future work:

· Static Embedding Index: Product recommendations are based on a pre-computed embedding index. Newly added products are not included in recommendations until the index is rebuilt.

· Non-Personalized Recommendations: The hybrid model is session-based and does not yet learn from individual user behavior to provide personalized suggestions.

· Model Optimization: The CLIP model, while powerful, is relatively large. Inference latency and resource consumption could be further optimized for a high-traffic production environment.

7.3. Defined Pathways for Future Research & Engineering

The limitations above naturally lead to concrete, compelling research questions that could form the basis of graduate work:

· Dynamic, Event-Driven Pipeline: Research and implement a system (using message brokers like Apache Kafka) to trigger real-time embedding updates upon product creation/update. Core Research Question: How does recommendation freshness impact user engagement and conversion metrics?

· Personalization Engine: Extend the system to incorporate implicit user feedback (clicks, dwell time, purchase history) and explore collaborative filtering techniques to move from a generic to a personalized hybrid model.

· Model Efficiency for Production: Investigate model compression techniques such as knowledge distillation or pruning on the CLIP model to significantly reduce its size and inference latency while preserving recommendation quality—a critical challenge for scalable deployment.

· Rigorous Benchmarking: Design a comprehensive evaluation framework to quantitatively compare the hybrid model's performance against the individual content-based and image-based models using metrics like Precision@K, Recall@K, and NDCG.

8. Project Artifacts

· Live Demo: https://myshop-76pn.onrender.com/

· Source Code Repository: https://github.com/FatimaAhmadinejad/MyShop

· Technical Documentation: This document.