ADAVANCED DATABASE SYSTEM

Mid-Lab Project(B)

Submitted By: Muhammad Sawood

Reg No: Fa23-bcs-191

Submitted To: Dr. Talha

Date: 21-10-2025

E-COMMERCE MARKETPLACE BACKEND - PROJECT REPORT

Advanced Database Systems (ADB)

Semester 5 - 2025

Executive Summary

This project focuses on building a complete e-commerce marketplace backend system using modern web technologies. The system demonstrates database optimization, efficient query handling, and robust API development for real-world scenarios.

Developed with **FastAPI** and **MongoDB**, it manages product catalogs, users, orders, and reviews. It includes advanced search functionality, fuzzy matching for misspelled queries, and a hybrid ranking system that blends relevance, popularity, and pricing.

The backend fully meets the project requirements for schema design, index optimization, aggregation queries, and performance efficiency, while adding advanced search and analytics features for a deeper understanding of modern database systems.

Table of Contents

1. Introduction

E-commerce applications rely on strong backend systems capable of managing large datasets of users, products, and orders efficiently. This project was designed to simulate such a backend environment with a focus on **database efficiency**, **API performance**, and **user-oriented features**.

Objectives:

- Design an efficient MongoDB schema for e-commerce data.
- Implement RESTful APIs using FastAPI for all main operations.
- Build an advanced search system with keyword and fuzzy matching.
- Apply aggregation pipelines for analytics and insights.
- Optimize query performance using indexing strategies.

Scope:

The backend supports product management, user profiles, orders, and reviews. It includes more than five functional APIs, over ten database indexes, and optimized queries for search and analytics.

2. Technology Stack

Framework: FastAPI

- High-performance and asynchronous architecture.
- Auto-generated Swagger documentation.
- Type-safe request validation using Pydantic.

Database: MongoDB

- NoSQL document-based structure for scalability.
- Supports text indexing, aggregation, and flexible schemas.

Driver: Motor (Async MongoDB Driver)

- Enables non-blocking I/O with async/await.

Other Tools:

- Pydantic for data validation.
- Uvicorn for ASGI server hosting.
- BSON for handling ObjectIds.

3. System Architecture

The system uses a clean **three-layer structure**:

1. **API Layer**: Handles HTTP requests via FastAPI endpoints.
2. **Logic Layer**: Manages search ranking, filtering, and aggregation logic.
3. **Database Layer**: Handles async interactions with MongoDB using Motor.

Flow:

User → API Request → FastAPI Endpoint → MongoDB Query → JSON Response

4. Database Design

The database uses both **embedding** and **referencing** strategies:

Collections:

- **Products:** Stores product details, category, brand, price, and rating.
- **Users:** Stores user profiles and contact details.
- **Orders:** Stores order data with embedded product snapshots.
- **Reviews:** Stores product reviews linked to users and products.

Each collection follows MongoDB best practices with proper ObjectId usage, denormalization for speed, and embedded ratings for instant access.

5. Indexing Strategy

Indexes play a key role in speeding up queries:

- **Text Index:** For full-text product search (name, brand, category, description).
- **Compound Index:** price and category for efficient range filtering.
- **Rating Index:** For sorting by average ratings.
- **User ID Index:** For quick user order lookups.
- **Timestamp Index:** For sorting recent orders.

These indexes reduce query time significantly and are chosen based on frequent access patterns.

6. API Implementation

Main Endpoints:

1. **/products/search:** Keyword and fuzzy search with filters.
2. **/users/{id}/orders:** Returns detailed user order history.
3. **/products/{id}/reviews:** Shows product reviews with user info.
4. **/orders/{id}:** Retrieves complete order details.
5. **/analytics/top-products:** Finds most purchased products via aggregation.

Key Highlights:

- Advanced hybrid ranking algorithm for search.
- Pagination and filtering support.
- MongoDB aggregation pipelines for analytics.
- Strong validation and error handling.

7. Advanced Features

- **Hybrid Search Algorithm:** Combines relevance, popularity, and price into a single score.
- **Fuzzy Matching:** Supports typo-tolerant search using regex patterns.
- **Automatic Data Seeding:** Loads sample data automatically on startup.

- **Async I/O:** Improves concurrency and scalability.
- **Auto Documentation:** Built-in Swagger UI for testing APIs.

8. Testing & Validation

Testing was performed using Swagger UI, Postman, and cURL commands. Each API was tested for:

- Valid and invalid inputs.
- Search for accuracy and performance.
- Order and review data consistency.
- Error handling for invalid IDs or missing data.

Performance results showed average response times between **50–300 ms**, depending on query complexity.

9. Challenges & Solutions

Challenge 1: ObjectId handling with Pydantic v2

Solution: Created a custom validator class for smooth conversion.

Challenge 2: Optimizing aggregation performance

Solution: Used $lookup pipelines and indexing for faster joins.

Challenge 3: Balancing multiple ranking factors

Solution: Weighted scoring formula (Text 40%, Popularity 40%, Price 20%).

Challenge 4: Handling short search queries

Solution: Conditional logic to switch between text and regex search.

10. Conclusion

This project successfully demonstrates a real-world backend for an e-commerce marketplace. It covers all essential features — from efficient schema design to intelligent search rankings and analytics. Using FastAPI and MongoDB, the system achieves high performance, scalability, and maintainability.

The project not only fulfills academic requirements but also reflects industry-standard practices suitable for production environments.

11. References

1. FastAPI Documentation - https://fastapi.tiangolo.com/
2. MongoDB Manual - https://www.mongodb.com/docs/manual/
3. Motor Async Driver - https://motor.readthedocs.io/
4. Pydantic Docs - https://docs.pydantic.dev/
5. ADB Lecture Notes - Semester 5