E-Commerce Recommendation System

Project Documentation

1. Introduction

A hybrid recommendation system combining:

a. **Django** (Frontend + Backend)
b. **FastAPI** (Recommendation Engine)
c. **TF-IDF Model + Cosine Similarity** (Content-based filtering)

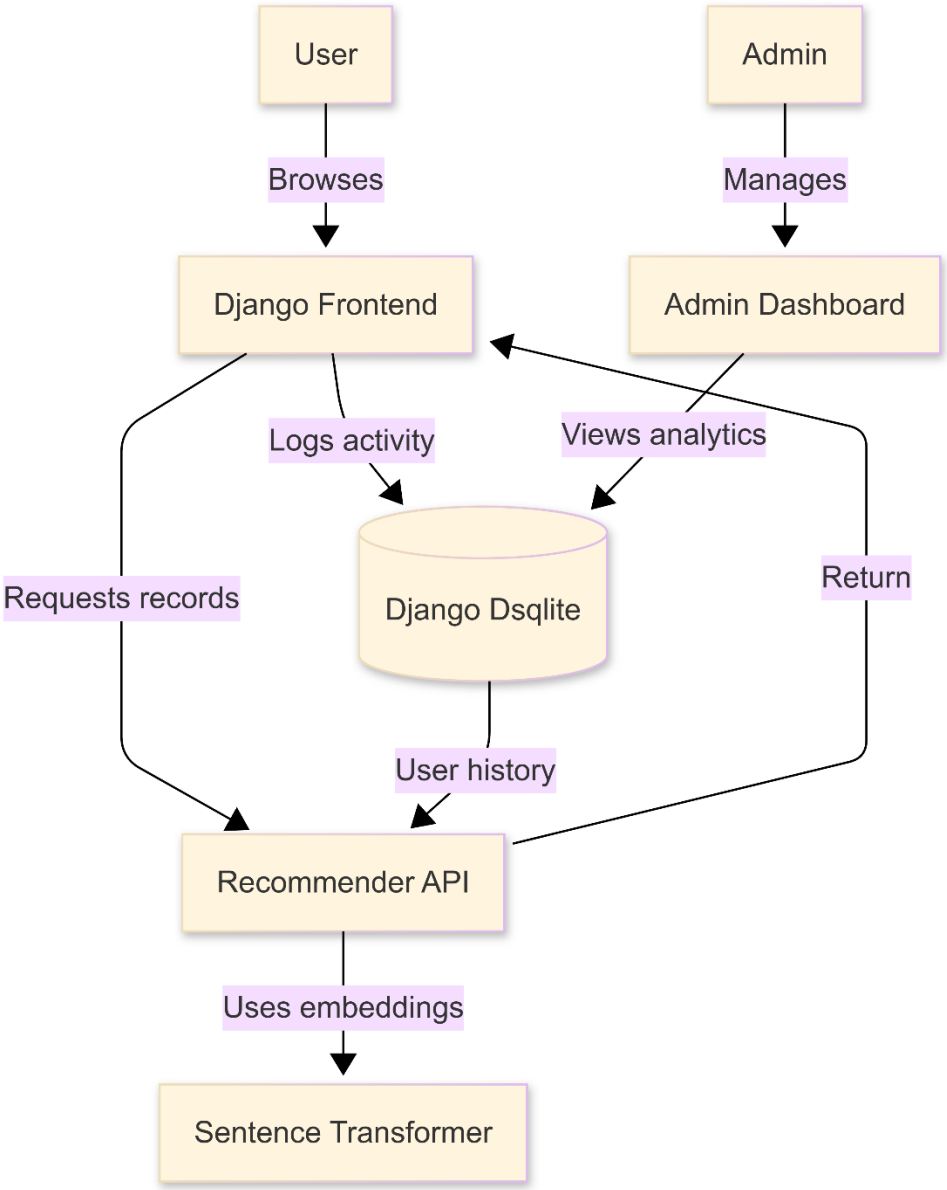


Fig 1: System Architecture Diagram

2. Models Used

a. Core Recommendation Models

Model	Type	Description
TF-IDF Vectorizer	NLP Feature Extraction	Converts product text (name, description) to numerical vectors
Cosine Similarity	Similarity Metric	Measures similarity between product vectors
Rule-Based Filtering	Collaborative	Recommends products from categories the user has interacted with

How They Work Together:

a. TF-IDF analyzes product descriptions
b. Cosine similarity finds similar products
c. User activity data adds personalization

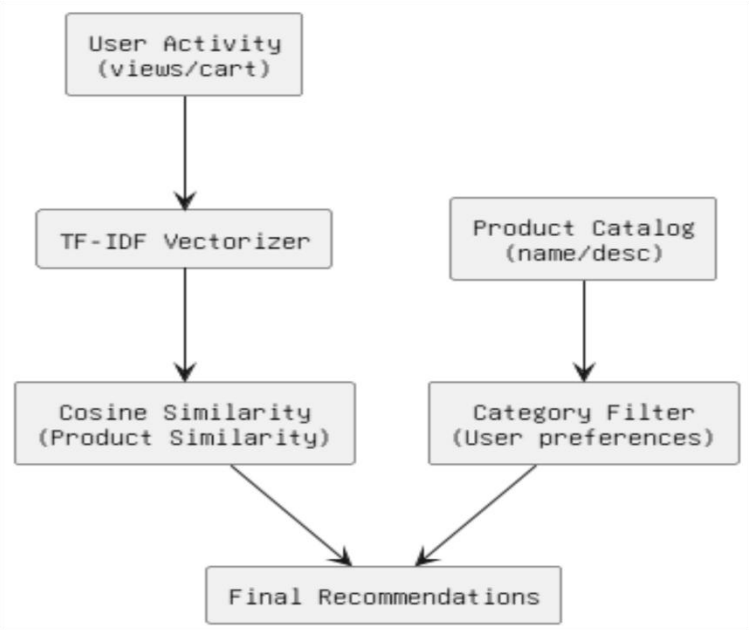
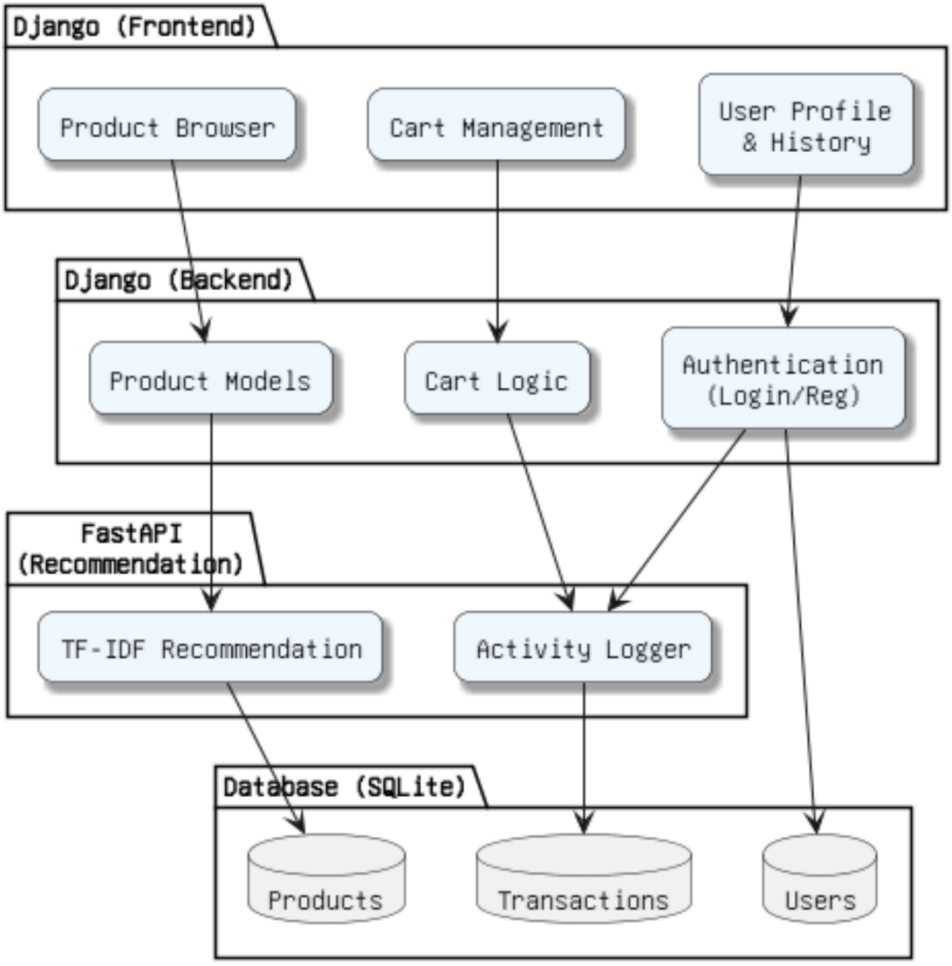


Fig 2: Recommendation Generation Process

3. System Architecture

a. Component Diagram



Key Parts :

a. **Django Frontend**: User interfaces

b. **Django Backend**: Business logic

c. **FastAPI**: Recommendation service

d. **Database**: Stores products/users/transactions in db.sqlite

Data Flow :

a. User interacts with Django
b. Activity sent to FastAPI
c. Recommendations generated
d. Displayed in Django templates

4. Implementation

a. Django Components

Example model (products/models.py)

```
class Product(models.Model):
    name = models.CharField(max_length=255)
    description = models.TextField()
    category = models.ForeignKey(Category, on_delete=models.CASCADE)
```

b. FastAPI Endpoints

```
@app.get("/api/recommend")
async def get_recommendations(user_id: int):
```

i. Get user activity
ii. Generate recommendations
iii. Return product IDs

5. Setup Guide

a. Installation

```
pip install django fastapi scikit-learn
```

Running the System

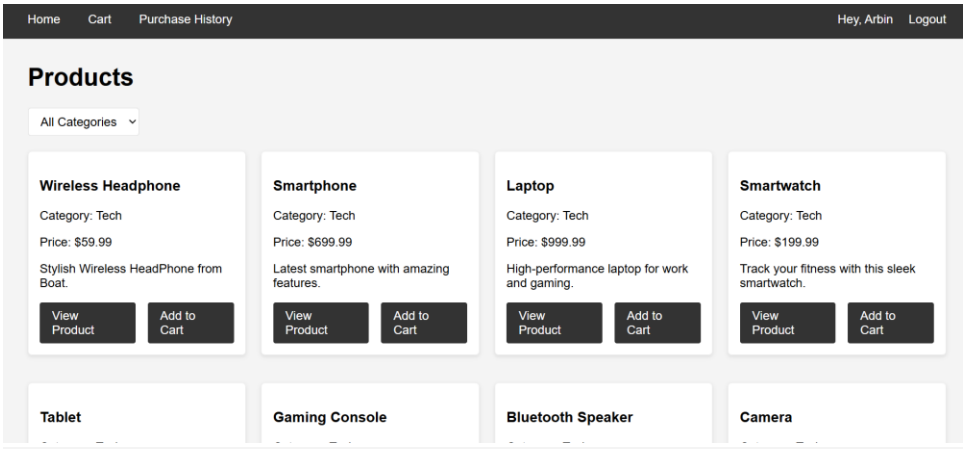
FastAPI (Port 8000)

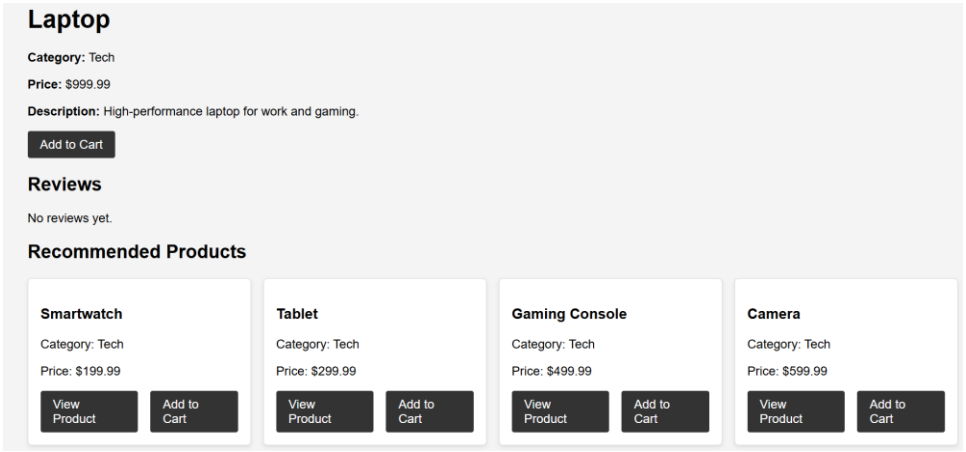
```
uvicorn main:app --reload
```

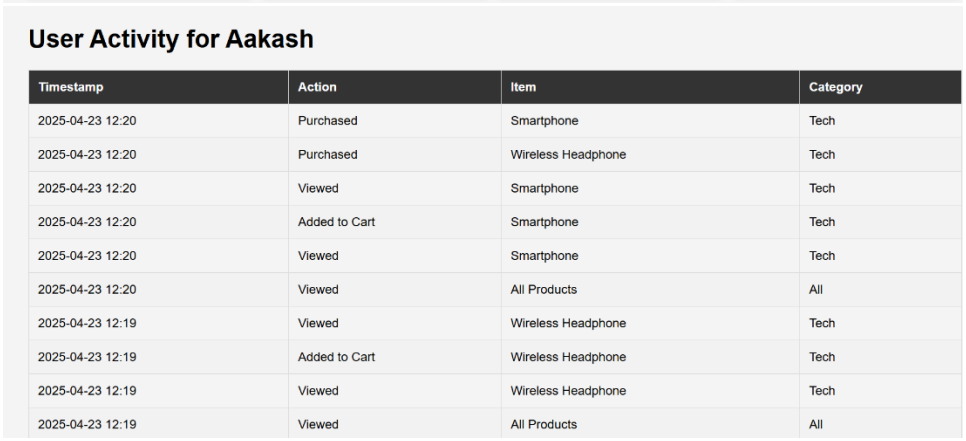
Django (Port 8001)

```
python manage.py runserver 8001
```

6. Screenshots







User Activity for Aakash

Timestamp	Action	Item	Category
2025-04-23 12:20	Purchased	Smartphone	Tech
2025-04-23 12:20	Purchased	Wireless Headphone	Tech
2025-04-23 12:20	Viewed	Smartphone	Tech
2025-04-23 12:20	Added to Cart	Smartphone	Tech
2025-04-23 12:20	Viewed	Smartphone	Tech
2025-04-23 12:20	Viewed	All Products	All
2025-04-23 12:19	Viewed	Wireless Headphone	Tech
2025-04-23 12:19	Added to Cart	Wireless Headphone	Tech
2025-04-23 12:19	Viewed	Wireless Headphone	Tech
2025-04-23 12:19	Viewed	All Products	All

7. Performance Notes

For 50 products(currently):

- First-run latency: ~2s (generates all embeddings)
- Subsequent requests: ~50ms (cached)
- Memory usage: ~150MB (model + embeddings)

8. Limitations & Future Work

Current Limitations:

- Basic text analysis only
- No real-time model updates

Future Improvements:

- Add BERT for semantic understanding
- Implement Redis caching