Journal of Computer and Electrical Sciences (JCES)- Volume-1 Issue-4, April 2026
ISSN: 3049-2602(Online)

E commerce platform with AI Recommendation Engine

I. Raja Devendra[1], Mrs. M. Havila[2]
[1]Student, Department of Computer Science and Engineering
Andhra Loyola Institute of Engineering and Technology, Vijayawada, Andhra Pradesh, India
[2]Assistant Professor, Department of Computer Science and Engineering
Andhra Loyola Institute of Engineering and Technology, Vijayawada, Andhra Pradesh, India

Abstract:*This paper presents the design and implementation of a Cross-Platform Ecommerce Application integrated with an AI-powered Product Recommendation Engine using TF-IDF (Term Frequency-Inverse Document Frequency) and Cosine Similarity algorithms. The system is developed as an Android mobile application using Java, communicating with a Python Flask backend that hosts the recommendation engine. For every product a user views, the engine generates four recommendations — two from the same product category using TF-IDF Cosine Similarity, and two from different categories using a composite cross-category scoring mechanism based on shared product tags, price proximity, and product ratings. Firebase Authentication manages user sessions and Firebase Firestore stores cart, wishlist, and order data in real time. The proposed system addresses the cold start problem inherent in collaborative filtering by relying entirely on product content, making it effective from the first day of deployment without requiring historical user interaction data.*

Keywords: *TF-IDF, Cosine Similarity, Recommendation Engine, Android, Java, Flask, Firebase Firestore, Cross-Category Recommendations, E-Commerce, Machine Learning.*

1. INTRODUCTION

The rapid growth of online commerce has created a significant challenge for users in discovering relevant products across large digital catalogs. Traditional e-commerce platforms typically rely on collaborative filtering recommendation systems that analyze user purchase history and browsing behavior. While effective for large platforms with extensive user data, these systems suffer from the well-known cold start problem — they are unable to generate meaningful recommendations for new users or newly added products without sufficient interaction history.

Content-based recommendation systems offer an effective alternative by analyzing the attributes of items themselves rather than user behavior patterns. TF-IDF vectorization combined with Cosine Similarity is one of the most established content-based approaches, converting product descriptions into numerical vectors and measuring the angular similarity between them to identify related products.

This paper presents a cross-platform ecommerce application that integrates a dual-lane AI recommendation engine. The system provides same-category recommendations using TF-IDF Cosine Similarity and cross-category recommendations using a composite scoring mechanism. The Android frontend communicates with a Python Flask backend over a local network, while Firebase provides cloud authentication and data storage. The system covers 30 products across five categories: Smartphones, Electronics, Lifestyle, Home Appliances, and Fitness.

https://doi.org/10.5281/zenodo.19610386

2. LITERATURE SURVEY

Hu et al. [1] introduced collaborative filtering for implicit feedback datasets using matrix factorization, demonstrating effective personalized recommendations from user interaction data. However, this approach requires substantial historical data and fails for new users and products.

Salton and McGill [2] introduced the TF-IDF weighting scheme for measuring term importance in documents relative to a collection. This technique has been widely applied in content-based recommendation systems and is well-suited for product catalogs with rich textual descriptions.

Lops et al. [3] conducted a comprehensive survey of content-based recommender systems and identified TF-IDF with Cosine Similarity as one of the most effective item-based recommendation approaches for domains where item descriptions are available, such as e-commerce product catalogs.

Zhang et al. [4] reviewed deep learning-based recommendation systems and highlighted that while neural approaches achieve high accuracy on large datasets, they require significant computational resources and large labeled datasets, making traditional NLP-based approaches more practical for lightweight mobile deployments.

From the reviewed literature, the following key observations can be made:

- Collaborative filtering requires large user behavior datasets and fails for new products (cold start problem).
- TF-IDF and Cosine Similarity provide effective content-based recommendations without requiring user history.
- Cross-category recommendations are largely underexplored in existing lightweight mobile systems.
- Firebase integration with Android provides scalable real-time cloud storage suitable for ecommerce applications.
- Existing systems rarely combine same-category and cross-category recommendations in a single unified engine.

3. PROPOSED SYSTEM

The proposed system is a cross-platform ecommerce application with an AI-powered recommendation engine designed to provide intelligent product suggestions without requiring historical user data. The system is developed as an Android mobile application using Java in Android Studio, connected to a Python Flask backend that hosts the TF-IDF recommendation engine.

The architecture follows a three-tier design. Tier 1 is the Android Client which handles all user interface operations including product browsing, cart and wishlist management, checkout, and recommendation display. Tier 2 is the Python Flask Backend which serves product data through the GET /products endpoint and recommendation results through the GET /recommend?product_id endpoint. The TF-IDF matrix and Cosine Similarity scores are pre-computed at server startup for efficiency. Tier 3 is the Firebase Cloud which provides Authentication for secure user session management and Firestore as the cloud NoSQL database storing cart, wishlist, and order data under each user's unique ID.

The recommendation engine implements a dual-lane architecture:

https://doi.org/10.5281/zenodo.19610386

- Lane 1 — Same-Category Recommendations: The top 2 most similar products from the same category are identified using TF-IDF vectorization of product descriptions with Cosine Similarity scoring.
- Lane 2 — Cross-Category Recommendations: The top 2 products from different categories are identified using a composite score: Score = (shared_tags x 0.5) + (price_proximity x 0.3) + (rating/5 x 0.2). This combines brand affinity, price range compatibility, and product quality.
- Final Results: The four recommendations are interleaved as [same1, cross1, same2, cross2] and displayed in two clearly labeled horizontal sections on the product detail screen: 'Similar Products' and 'You May Also Like'.

The product dataset consists of 30 products across five categories with six products per category. Each product includes a name, description, price, rating, category, image reference, and product tags used for cross-category scoring.

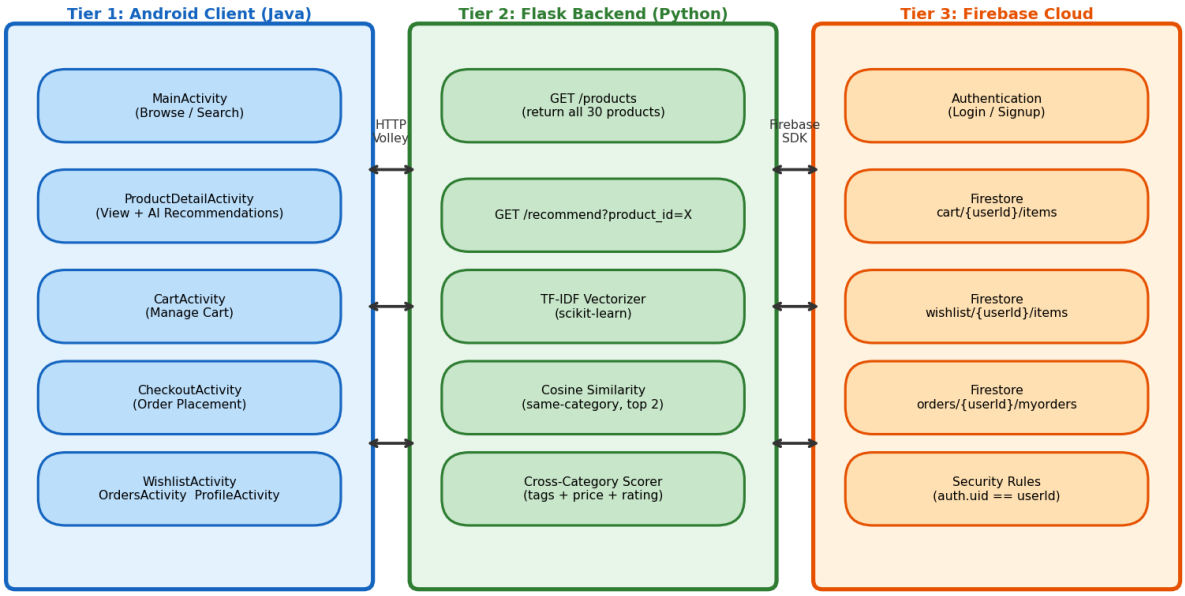

Fig 1: System Architecture — [Three-tier architecture: Android App | Flask API | Firebase Cloud]

4. METHODOLOGY

The methodology is organized into the following steps:

1. TF-IDF Vectorization:

All 30 product descriptions are fed into the TfidfVectorizer from scikit-learn with English stop words removed. Each description is converted into a numerical vector where words that are frequent in a specific product description but rare across all products receive high TF-IDF scores, making them strong identifiers for that product.

2. Cosine Similarity Computation:

The Cosine Similarity between all product pairs is computed from the TF-IDF matrix using the formula: cos(A,B) = (A.B) / (|A| x |B|). A score of 1.0 indicates identical descriptions and 0.0 indicates no shared important terms. This matrix is computed once at server startup and cached for all subsequent recommendation requests.

https://doi.org/10.5281/zenodo.19610386

3. Same-Category Recommendation:

For a given product, all products from the same category are ranked by their Cosine Similarity score in descending order. The top 2 results are returned as same-category recommendations.

4. Cross-Category Scoring:

For each product from a different category, a composite score is computed as: Score = (shared_tags x 0.5) + (max(0, 1 - price_diff/200000) x 0.3) + (rating/5 x 0.2). All different-category products are ranked by this score and the top 2 are returned as cross-category recommendations.

5. Android Frontend Implementation:

The Android application is built in Java using Android Studio. Volley handles asynchronous HTTP requests to the Flask API. RecyclerViews display product lists, cart items, and recommendations. Firebase SDK handles Authentication and Firestore operations for cart, wishlist, and order management.

6. Firebase Data Storage:

User-specific data is stored in Firestore under three subcollection paths: cart/{userId}/items, wishlist/{userId}/items, and orders/{userId}/myorders. Security rules enforce that each user can only access their own data by verifying the request authentication UID matches the document path user ID.

7. Checkout and Order Flow:

The checkout process collects the user's delivery address (name, phone, address, city, state, pincode) and payment method selection (Cash on Delivery, UPI, or Credit/Debit Card). On order confirmation, the complete order document including items, delivery address, payment method, total amount, status, and timestamp is saved to Firestore and the cart is cleared.

5. RESULTS AND DISCUSSION

The proposed cross-platform ecommerce application with AI recommendation engine was successfully developed and tested on Android devices. The system was evaluated across 26 test cases covering all functional requirements including user authentication, product browsing, recommendation accuracy, cart and wishlist operations, checkout flow, and order history.

- The TF-IDF recommendation engine correctly identified same-category products based on description similarity. For example, viewing the iPhone 15 Pro returned Galaxy S24 and Pixel 8 as same-category recommendations, both high-rated smartphones with similar feature descriptions.
- The cross-category scoring mechanism effectively identified relevant products from different categories. Viewing the iPhone 15 Pro returned AirPods Pro 2 (Electronics) and Apple Watch Series 9 (Lifestyle) as cross-category recommendations, reflecting shared Apple brand tags and compatible price ranges.
- Firebase Authentication successfully managed user registration and login with secure session persistence across application restarts.
- Firestore operations for cart, wishlist, and order management completed successfully with proper duplicate detection preventing repeated additions.
- The complete checkout flow — from address entry through payment selection to order confirmation — functioned correctly with orders persisted to Firestore and carts cleared after placement.

https://doi.org/10.5281/zenodo.19610386

- All 26 test cases passed, confirming that all functional and non-functional requirements of the system were met.

The recommendation response time was consistently under 200ms for all test requests, demonstrating the efficiency of pre-computing the TF-IDF matrix at server startup rather than computing it per request.

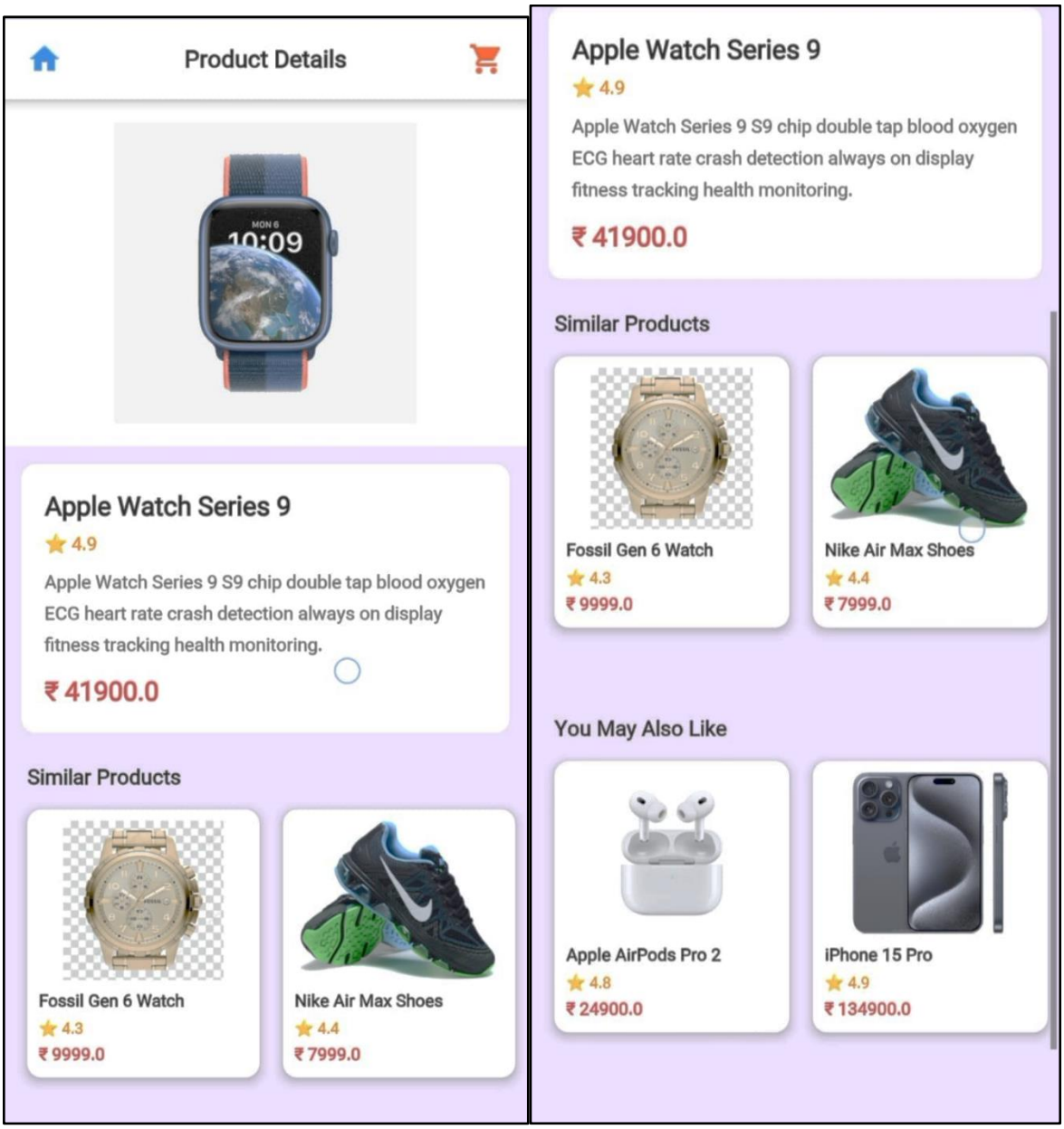

Fig 2: Recommendation Results

https://doi.org/10.5281/zenodo.19610386

Approach	Dataset Required	Cold Start	Cross-Category
Collaborative Filtering	Large user history	Problem exists	No
Deep Learning Recommender	Large labeled data	Problem exists	No
Content-Based (TF-IDF only)	Product descriptions	No problem	No
Proposed System (TF-IDF + Cross-Category)	Product descriptions	No problem	Yes

Table 1: Comparison of Recommendation Approaches

6. CONCLUSION

This work successfully developed a cross-platform ecommerce application integrated with an AI-powered product recommendation engine using TF-IDF and Cosine Similarity. The system effectively addressed the cold start problem by relying on product content rather than user behavior data, making it operational from the first day of deployment.

The dual-lane recommendation architecture provided both same-category recommendations through TF-IDF Cosine Similarity and cross-category recommendations through a composite scoring mechanism combining shared product tags, price proximity, and product ratings. This approach delivered a more diverse and contextually relevant set of recommendations compared to same-category-only systems.

The integration of Firebase Authentication and Firestore provided scalable, secure cloud storage for cart, wishlist, and order data. All 26 test cases passed, confirming the correctness and reliability of the complete shopping flow. The recommendation engine achieved consistent response times under 200ms by pre-computing the TF-IDF matrix at server startup.

Future work includes deploying the Flask API to a cloud platform such as Google Cloud Run for production availability, integrating collaborative filtering as user interaction data accumulates, and implementing real payment gateway processing through Razorpay or Stripe.

REFERENCES

1. Y. Hu, Y. Koren, and C. Volinsky, 'Collaborative Filtering for Implicit Feedback Datasets,' IEEE International Conference on Data Mining (ICDM), pp. 263-272, 2008.
2. G. Salton and M. J. McGill, 'Introduction to Modern Information Retrieval,' McGraw-Hill, New York, 1983.
3. P. Lops, M. de Gemmis, and G. Semeraro, 'Content-based Recommender Systems: State of the Art and Trends,' Recommender Systems Handbook, Springer, pp. 73-105, 2011.
4. S. Zhang, L. Yao, A. Sun, and Y. Tay, 'Deep Learning Based Recommender System: A Survey and New Perspectives,' ACM Computing Surveys, vol. 52, no. 1, pp. 1-38, 2019.
5. Android Developer Documentation, 'Android Studio Developer Guide,' Google Developers, 2024. [Online]. Available: https://developer.android.com/studio
6. Firebase Documentation, 'Firebase Authentication and Firestore Guide,' Google Developers, 2024. [Online]. Available: https://firebase.google.com/docs

https://doi.org/10.5281/zenodo.19610386

7. scikit-learn Documentation, 'TfidfVectorizer and Cosine Similarity,' 2024. [Online]. Available: https://scikit-learn.org
8. Flask Documentation, 'Flask Web Framework,' Pallets Projects, 2024. [Online]. Available: https://flask.palletsprojects.com
9. P. Lops, M. de Gemmis, and G. Semeraro, 'Recommender Systems Handbook,' Springer, New York, 2015.
10. G. Adomavicius and A. Tuzhilin, 'Toward the Next Generation of Recommender Systems: A Survey,' IEEE Transactions on Knowledge and Data Engineering, vol. 17, no. 6, pp. 734-749, 2005.
11. R. Burke, 'Hybrid Recommender Systems: Survey and Experiments,' User Modeling and User-Adapted Interaction, vol. 12, no. 4, pp. 331-370, 2002.
12. M. J. Pazzani and D. Billsus, 'Content-Based Recommendation Systems,' The Adaptive Web, Springer, pp. 325-341, 2007.

https://doi.org/10.5281/zenodo.19610386