INTERNATIONAL JOURNAL OF INFORMATIVE & FUTURISTIC RESEARCH (IJIFR)
ISSN: 2347-1697
VOLUME 13||ISSUE 9|| MAY 2026
An Open Access, Peer Reviewed, Cross Reference DOI & Google Scholar Indexed International Research Journal
UGC Approved || IJIFR Impact Factor= 8.057
Estd Year: 2013 || Email: editors@ijifr.org
Availaible Online at: https://www.ijifr.org/

DOI: https://doi.org/10.64672/IJIFR/26.05.13.09.041 PUBLISHED ON: MAY 31, 2026

AN INTELLIGENT LEARNING RECOMMENDATION FRAMEWORK USING TF-IDF AND COSINE SIMILARITY WITH CAREER PATH PREDICTION AND SENTIMENT ANALYSIS

Mitayi Ganesh Singh[1] , B. Shireesha [2]

[1] Student, Department of Computer Applications, Viswam Engineering College, Andhra Pradesh, India
[2]Assistant Professor, Department of Computer Applications, Viswam Engineering College, Madanapalle, Andhra Pradesh

ABSTRACT

The rapid proliferation of online educational content has created an overwhelming challenge for learners navigating thousands of courses across the internet. Identifying the right course that aligns with an individual's existing skills, career aspirations, and personal interests is no longer trivial. This paper presents an intelligent learning recommendation framework that combines machine learning techniques with a full-featured web application to deliver personalized, real-time course recommendations. The platform implements a recommendation engine powered by Term Frequency–Inverse Document Frequency (TF-IDF) vectorization and cosine similarity computation using Scikit-learn. By analyzing a student's stated interests, career goals, and current skill set, the engine computes semantic similarity scores between the learner's profile and available courses, returning the most relevant recommendations. Beyond course recommendations, the platform incorporates an AI-driven career path prediction module that evaluates the learner's skill profile against predefined career personas—including Data Scientist, Full Stack Developer, Cloud Architect, Cybersecurity Analyst, and AI Engineer—and identifies the most suitable career trajectory with percentage-based match scores. A skill gap analysis module identifies specific skills the learner needs to acquire for their target career, directly linking gaps to relevant courses. The platform also includes a sentiment analysis module that classifies course reviews as Positive, Negative, or Neutral using keyword scoring. The complete system is built using Django 5.2, Bootstrap 5, and SQLite, providing user authentication, profile management, course enrollment tracking, and a responsive web interface.

KEYWORDS *TF-IDF; Cosine Similarity; Learning Recommendation; Career Path Prediction; Skill Gap Analysis; Sentiment Analysis; Django; Content-Based Filtering*

Recommended Citation:
Singh, M G, Shireesha, B : "An AI-Powered Customer Sentiment Analysis Platform Integrating VADER And Rule-Based Emotion Detection", International Journal of Informative & Futuristic Research (IJIFR), Vol. (13) (9), May 2026, pp. 1662-1668. https://doi.org/10.64672/IJIFR/26.05.13.09.041

1. INTRODUCTION

The digital transformation of education has created an abundance of online courses across platforms like Coursera, Udemy, and edX, yet learners face the paradox of choice—too many options impair decision-making. Traditional course discovery relies on keyword search and popularity rankings that treat all learners as equivalent, failing to personalize at the individual level. A student who already knows Python fundamentals receives the same results as a complete beginner, despite radically different needs.

Recommendation systems that learn patterns from user and item data offer a compelling solution, as demonstrated by Netflix, Amazon, and Spotify.

This paper presents an intelligent learning recommendation framework implemented as a Django web application with a custom engine powered by TF-IDF vectorization and cosine similarity. The system maintains detailed student profiles capturing skills, interests, and career goals, computing semantic similarity between profiles and course content. Additionally, the framework integrates career path prediction using persona-based TF-IDF matching against five predefined career roles, skill gap analysis identifying missing competencies linked to relevant courses, and automated sentiment analysis of course reviews. Together, these capabilities transform online learning into an intelligently guided journey toward defined professional goals.

2. LITERATURE SURVEY

Ricci et al. [1] comprehensively treated recommender system approaches including content-based and collaborative filtering. Salton and Buckley [4] established TF-IDF term weighting for information retrieval. Pazzani and Billsus [8] examined content-based systems using feature vectors and similarity computation. Pedregosa et al. [3] introduced Scikit-learn providing the TfidfVectorizer and cosine_similarity implementations used in this work. Devlin et al. [5] demonstrated BERT's superior NLP performance, representing an enhancement path for the sentiment module. Lu et al. [9] surveyed recommender system developments documenting evolution toward hybrid intelligent systems.

Existing platforms predominantly rely on keyword search and popularity metrics treating all learners as equivalent, or collaborative filtering suffering from the cold-start problem. Manual learning path curation is expensive and inflexible. This framework addresses these limitations through content-based TF-IDF recommendation working immediately for new users with profile data, combined with career intelligence contextualizing recommendations within professional trajectories.

3. SYSTEM PROPOSAL

3.1 Problem Statement

The core problem exists at the intersection of information overload and personalization deficiency. Students face thousands of courses varying in level, quality, prerequisites, and relevance, with limited information about alignment with their background. Conventional search mechanisms systematically fail to account for individual learner heterogeneity. A related problem is the absence of career contextualization—learners know their aspirations but lack detailed knowledge of required skills, their current distance from those requirements, and the optimal learning path. Furthermore, accumulated course reviews represent untapped quality intelligence that requires automated analysis at scale.

3.2 Proposed System

The proposed framework is a Django web application with a 'core' module encapsulating all business logic. The database layer defines models for Skill, Course, StudentProfile, Enrollment, and Review. The Recommender class loads all courses, constructs combined feature strings (title + description + category + skills), and builds a TF-IDF matrix using TfidfVectorizer. For recommendations, it concatenates user interests, career goal, and skills into a profile vector, computing cosine similarity against all courses. Career prediction fits a temporary TF-IDF on five persona templates, returning percentage-based match scores. Skill gap analysis identifies courses matching the career goal and returns skills the student lacks. The sentiment module classifies reviews using keyword counting.

3.3 System Architecture

The system follows three-tier Model-View-Template (MVT) architecture. The Data Layer manages five primary models through Django's ORM. The Application Logic Layer encompasses views, the Recommender class, URL routing, and signal handlers. The Presentation Layer uses Django's template engine with Bootstrap 5 styling. Fig. 1 illustrates the complete architecture.

INTERNATIONAL JOURNAL OF INFORMATIVE & FUTURISTIC RESEARCH (IJIFR)
ISSN: 2347-1697
IJIFR /20260530/V13/E9/041
VOLUME 13||ISSUE 9|| MAY 2026
https://www.ijifr.org/

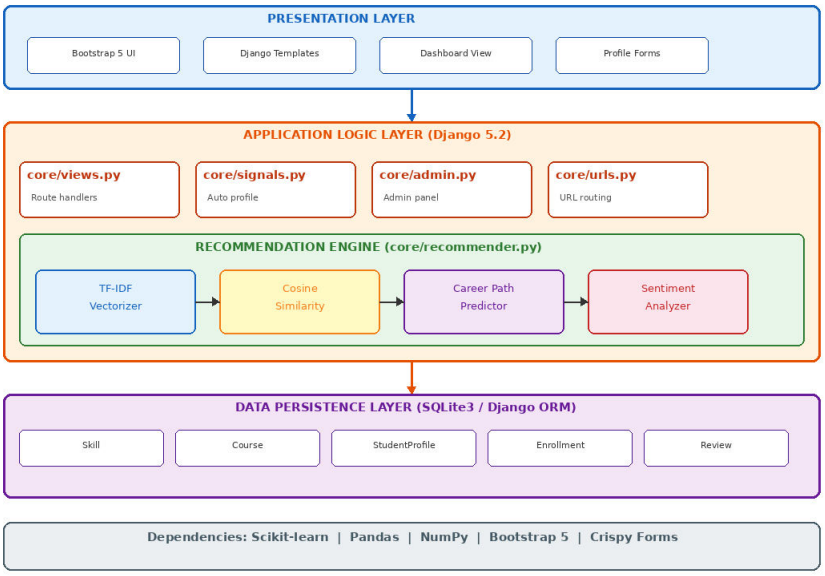

Figure 1: System Architecture of the Learning Recommendation Framework

3.4 System Flow

The operational flow begins with user registration, where a Django signal handler automatically creates a StudentProfile. After profile completion with interests, skills, and career goal, each dashboard visit triggers the recommendation pipeline: Recommender instantiation with TF-IDF fitting, cosine similarity computation for course recommendations, skill gap analysis against the career goal, and career path prediction against five persona templates. Review submissions invoke the sentiment analyzer. Fig. 2 depicts the complete system flow.

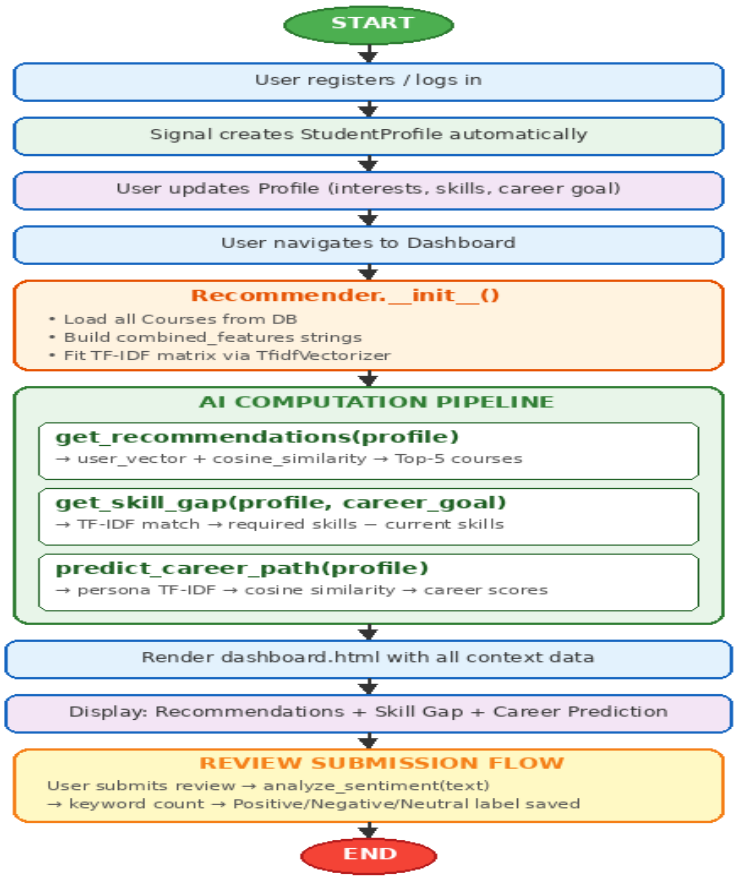

Figure 2: System Flow Diagram of the Learning Recommendation Framework

INTERNATIONAL JOURNAL OF INFORMATIVE & FUTURISTIC RESEARCH (IJIFR)
ISSN: 2347-1697
IJIFR /20260530/V13/E9/041
VOLUME 13||ISSUE 9|| MAY 2026
https://www.ijifr.org/

3.5 UML Diagrams

3.5.1 Use Case Diagram

Two primary actors interact with the system: Students (register, update profile, browse courses, enroll, submit reviews, view recommendations, career analysis, and skill gaps) and Administrators (manage courses, skills, users, and seed data via management commands). The AI Engine acts as a system actor invoked during dashboard rendering and review submission. Fig. 3 presents the use case model.

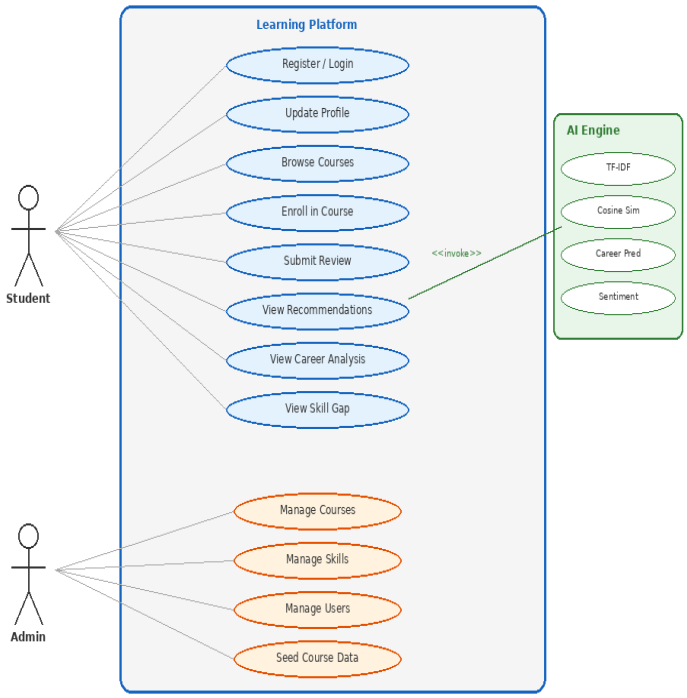

Figure 3: Use Case Diagram of the Learning Recommendation Framework

3.5.2 Sequence Diagram

The sequence diagram traces interactions between Browser, Django Router, Dashboard View, Recommender, Django ORM, and Database during a dashboard request. The flow covers profile retrieval, Recommender initialization with TF-IDF matrix fitting, sequential invocations of get_recommendations(), get_skill_gap(), and predict_career_path(), enrollment query, and template rendering. Fig. 4 illustrates the complete interaction sequence.

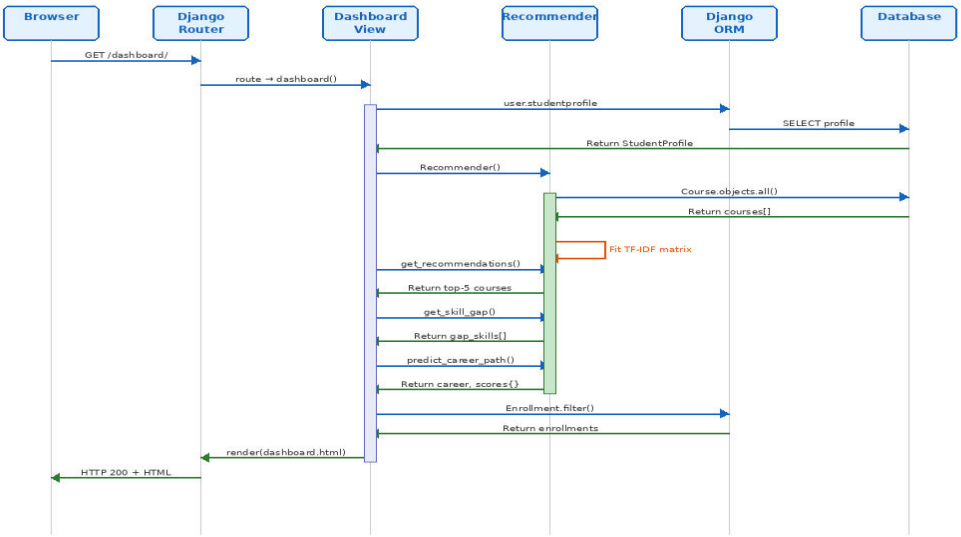

Figure 4: Sequence Diagram for Dashboard Recommendation Request

4. IMPLEMENTATION

4.1 Recommendation Engine

The Recommender class initializes by loading all Course objects, constructing combined_features strings, and fitting a TfidfVectorizer with English stop words removed. The get_recommendations() method concatenates the student's interests, career goal, and skill names into a single text string, transforms it into the TF-IDF space, computes cosine similarity against all course vectors, and returns the top five courses ranked by similarity score. This content-based approach ensures recommendations are driven by semantic alignment rather than popularity metrics, and it avoids the cold-start problem since new users with profile information receive immediate recommendations.

4.2 Career Path Prediction

The predict_career_path() method maintains five career persona dictionaries mapping role titles to characteristic skill strings. A temporary TfidfVectorizer is fitted on the combined corpus of persona descriptions and the student's skill string, ensuring the vocabulary covers both spaces. Cosine similarity between the student vector and each persona vector produces percentage-based match scores. The highest-scoring career is returned as the predicted path alongside the complete score dictionary, enabling students to explore alternative trajectories.

4.3 Skill Gap Analysis

The get_skill_gap() method transforms the career goal into a TF-IDF vector, identifies the three most similar courses in the catalog by cosine similarity, collects all skills taught by those courses into a required_skills set, and returns the set difference between required_skills and the student's current_skills. This directly links career aspirations to specific learning actions, creating a coherent recommendation-to-career pipeline.

4.4 Sentiment Analysis

The analyze_sentiment() method classifies review text using curated positive and negative keyword lists. After lowercasing the input, it counts occurrences of each category's keywords and assigns Positive if positive count exceeds negative, Negative if the reverse, and Neutral if equal. While simpler than transformer-based approaches, this lexicon method is computationally lightweight, fully transparent, and well-suited to the structured vocabulary of course reviews.

4.5 Web Application

The Django application provides user registration with automatic StudentProfile creation via post_save signals, profile management with comma-separated skill parsing, course browsing and enrollment with unique-together constraints, and review submission with integrated sentiment classification. The dashboard template renders all AI insights: recommendations as course cards, skill gaps as lists, career predictions as ranked score tables, and enrollments with completion status, all styled with Bootstrap 5.

5. RESULTS AND DISCUSSION

5.1 Recommendation Quality

The TF-IDF cosine similarity approach produces semantically meaningful recommendations that align course content with learner profiles. Students with Python and statistics interests receive data science courses, while those specifying web development skills receive Django and React courses. The content-based approach ensures transparency—recommendations are driven by explicit textual alignment rather than opaque collaborative signals. For the current dataset scale (tens to hundreds of courses), TF-IDF computation completes in milliseconds on standard hardware. Fig. 5 presents the TF-IDF framework and career persona definitions

5.2 Career Prediction and Performance

The career prediction module correctly identifies dominant career matches for profiles with clear skill concentrations. A student with Python, Pandas, NumPy, and Statistics skills receives the highest match for Data Scientist, while one with Django, React, and JavaScript matches Full Stack Developer. The

complete pipeline (TF-IDF fitting, cosine similarity, career prediction, skill gap) executes in sub-second time for catalogs of up to several hundred courses, with memory requirements under 100 MB for typical catalog sizes.

.

TF-IDF + Cosine Similarity Framework

TF-IDF Vectorization Process:

1. Build combined_features = title + description + category + skills for each course
2. TfidfVectorizer(stop_words='english') → sparse TF-IDF matrix [courses × vocabulary]
3. User profile vector = transform(interests + career_goal + current_skills)

Cosine Similarity: sim(A, B) = (A · B) / (||A|| × ||B||) → Score ∈ [0, 1] → Rank and return Top-5

Table 1: Career Persona Definitions for Path Prediction

Career Role	Skill Keywords	Match
Data Scientist	Python, Pandas, NumPy, Statistics, Machine Learning, SQL	TF-IDF
Full Stack Developer	Django, React, JavaScript, Node.js, SQL, HTML, CSS	Cosine
Cloud Architect	AWS, Azure, Cloud, Docker, Kubernetes, Terraform	Similarity
Cybersecurity Analyst	Network Security, Linux, Ethical Hacking, Cryptography	Score
AI Engineer	Python, PyTorch, TensorFlow, Deep Learning, NLP, CV	× 100%

Table 2: Sentiment Analysis Lexicon Keywords

Category	Keywords	Rule

Figure 5: TF-IDF Cosine Similarity Framework and Career Persona Definitions

6. CONCLUSION

This paper presents an intelligent learning recommendation framework that successfully integrates TF-IDF vectorization and cosine similarity computation with career path prediction, skill gap analysis, and automated sentiment analysis within a complete Django web application. The content-based recommendation engine delivers personalized course suggestions based on semantic alignment between learner profiles and course content, avoiding both the cold-start problem of collaborative filtering and the impersonality of popularity-based approaches. The career intelligence module contextualizes learning decisions within professional development trajectories, providing quantitative match scores against five career personas and identifying specific skill gaps linking current competencies to career requirements. The sentiment analysis module extracts quality intelligence from course reviews at scale. Together, these capabilities transform online learning from a self-directed, hit-or-miss activity into an intelligently guided journey toward defined professional and educational goals.

7. FUTURE ENHANCEMENTS

Key enhancement directions include integration of collaborative filtering to enable serendipitous cross-domain recommendations complementing the content-based engine; replacement of the lexicon-based sentiment analyzer with a fine-tuned BERT or RoBERTa classifier for handling negation, sarcasm, and mixed sentiment; implementation of a learner feedback loop using enrollment decisions and course completion rates to refine recommendations via matrix factorization or reinforcement learning; integration with job posting APIs (LinkedIn, Indeed) for career intelligence grounded in real-time labor market data; implementation of spaced repetition and learning progress tracking; and scalability enhancements including Redis-based TF-IDF matrix caching, PostgreSQL migration, and load-balanced Gunicorn deployment for production-scale operation.

8. REFERENCES

[1] F. Ricci, L. Rokach, and B. Shapira, *Recommender Systems Handbook*, 2nd ed. Springer, 2015.
[2] C. C. Aggarwal, *Recommender Systems: The Textbook*, Springer, 2016.
[3] F. Pedregosa et al., "Scikit-learn: Machine Learning in Python," *JMLR*, vol. 12, pp. 2825–2830, 2011.
[4] G. Salton and C. Buckley, "Term-weighting Approaches in Automatic Text Retrieval," *Information Processing & Management*, vol. 24, no. 5, pp. 513–523, 1988.
[5] J. Devlin et al., "BERT: Pre-training of Deep Bidirectional Transformers," *Proc. NAACL*, pp. 4171–4186, 2019.
[6] Django Software Foundation, "Django Documentation v5.2," *docs.djangoproject.com*, 2024.
[7] W. McKinney, "Data Structures for Statistical Computing in Python," *Proc. 9th Python in Science Conf.*, pp. 51–56, 2010.
[8] M. J. Pazzani and D. Billsus, "Content-based Recommendation Systems," in *The Adaptive Web*, Springer, pp. 325–341, 2007.
[9] J. Lu et al., "Recommender System Application Developments: A Survey," *Decision Support Systems*, vol. 74, pp. 12–32, 2015.
[10] J. S. Breese, D. Heckerman, and C. Kadie, "Empirical Analysis of Predictive Algorithms for Collaborative Filtering," *Proc. UAI*, pp. 43–52, 1998.
[11] Bootstrap Team, "Bootstrap 5 Documentation," *getbootstrap.com*, 2024.
[12] Python Software Foundation, "Python 3.12 Documentation," *docs.python.org*, 2024.