

AI-Based Student Connect Portal

Pawan Chandak

Dept. of
Information Technology
SKNCOE, Pune, India

pawanchandak2003.skn.it@gmail.com

Ram Dhande

Dept. of
Information Technology
SKNCOE, Pune, India

ramdhande.skn.it@gmail.com

Vedant Petkar

Dept. of
Information Technology
SKNCOE, Pune, India

vedantpetkar.skn.it@gmail.com

Om Sonawane

Dept. of
Information Technology
SKNCOE, Pune, India

omsonwane.skn.it@gmail.com

Dr. M. L. Bangare

Department of Information Technology,

Smt. Kashibai Navale College of Engineering, Pune, Maharashtra, India

Abstract — The increasing digitalization of academic institutions has created a demand for unified platforms that facilitate transparent communication between students, faculty, and administration. Traditional campus portals operate in silos, resulting in delayed notifications, unresolved grievances, and weak student engagement. This paper presents an AI-Based Student Connect Portal — an intelligent web platform integrating Machine Learning (ML), Natural Language Processing (NLP), and sentiment analysis to enable adaptive communication and efficient issue resolution. The system automatically classifies and prioritizes notices, performs emotion-based grievance analysis, and delivers personalized recommendations. Built on the Django framework (Python) with MySQL, the portal supports real-time feedback through a conversational AI chatbot and a multilingual interface supporting English, Hindi, and Marathi. Evaluation across 250+ student records demonstrated 87% accuracy in sentiment detection and 92% classification precision for notices, reducing manual administrative workload by 30%. This work demonstrates how AI-driven personalization can transform campus operations into a student-centric experience.

Keywords — AI Recommendation System, NLP, Django Framework, Sentiment Analysis, Student Portal, Machine Learning, Grievance Management.

I. INTRODUCTION

Digital transformation in higher education has accelerated the adoption of portals and information systems to manage courses, assessments, notices, and student services. Despite rapid adoption, most institutional portals remain transactional and document-centric rather than interaction-centric. Students frequently miss time-sensitive updates, lack clear grievance escalation paths, and cannot discover peers, mentors, or academic groups aligned with their goals.

Simultaneously, administrators handle repetitive queries, manually categorize notices, and work with fragmented dashboards across departments. The resulting friction diminishes student satisfaction and institutional efficiency. AI provides a principled foundation to address these gaps. Predictive models infer notice relevance for specific student cohorts; NLP digests free-text grievances into structured,

actionable tickets; and personalized recommenders connect learners to relevant communities, events, and resources. This paper introduces a production-oriented architecture combining Django's MVT pattern with ML/NLP microservices via REST APIs. The core contributions are: (1) a hybrid recommendation mechanism unifying content features and collaborative behavioral signals with a feedback loop; (2) a multilingual, intent- and sentiment-aware grievance pipeline with urgency-based routing; (3) a modular Django/MySQL reference implementation; and (4) a realistic evaluation on institutional data covering algorithmic metrics and user-centered outcomes.

II. LITERATURE REVIEW

Campus information systems have traditionally emphasized ERP-like automation of attendance, timetables, and results [4]. Studies on Django-based platforms report improved data accessibility but limited personalization [2]. AI-powered chatbots show consistent promise for reducing administrative query load through NLP-driven conversation flows [3]. From an AI perspective, recommendation approaches for academic platforms span content-based filtering, collaborative filtering, and hybrid schemes [7]. Transformer-based encoders such as BERT variants enable robust sentiment and intent classification in low-resource domains through fine-tuning. Best practices in educational technology emphasize explainability, fairness across student cohorts, and privacy-preserving data handling [8][9]. Prior work — SmartEdu Insights [1] and the AI-ML Campus Platform [6] — demonstrate measurable operational gains when ML augments existing institutional workflows, providing validation for the approach described in this paper.

III. SYSTEM OBJECTIVES AND REQUIREMENTS

A. Objectives

- (O1) Centralize campus communication to minimize missed critical notices.
- (O2) Reduce administrative load through automated categorization, routing, and FAQ handling.

- (O3) Enhance student connectedness through data-driven personalized recommendations.
- (O4) Support inclusive access via multilingual interaction (English, Hindi, Marathi).
- (O5) Protect student privacy and enforce role-based access control.

B. Functional Requirements

Functional requirements include JWT-based authentication; role-specific dashboards for Student, Faculty, Admin, and Parent roles; AI-assisted notice creation and scheduling; grievance submission with real-time status tracking; chatbot Q&A with guarded generative fallback; personalized recommendation feeds; and an analytics dashboard for administrators to monitor engagement and grievance trends.

IV. SYSTEM ARCHITECTURE

The system follows a four-layered architecture as illustrated in Fig. 1. The Presentation Layer provides role-specific responsive dashboards. The Application Layer is built on Django REST framework with JWT-based authentication and role-based access control (RBAC). The Intelligence Layer comprises stateless ML/NLP microservices deployed independently to allow scaling and A/B testing without affecting the application layer. The Data Layer uses MySQL 8 with AES-256 encryption for personally identifiable information (PII) fields, and object storage for logs and attachments. Services communicate over HTTPS with JWT session tokens. Celery with Redis manages asynchronous background jobs including notification dispatch and model re-training pipelines.



Fig. 1. System Architecture — Four-Layer Design

Fig. 1. System Architecture — Four-Layer Design

V. USE CASE DIAGRAM

Fig. 2 presents the use case diagram illustrating primary interactions between user roles and the portal's AI-powered features. Three actor types interact with the system: Student, Admin/Faculty, and Parent. AI system use cases — Classify & Prioritize Notice, Sentiment Analysis, and Generate Recommendation — are included by application-level use cases via standard UML include relationships, operating transparently behind the user-facing functionality.

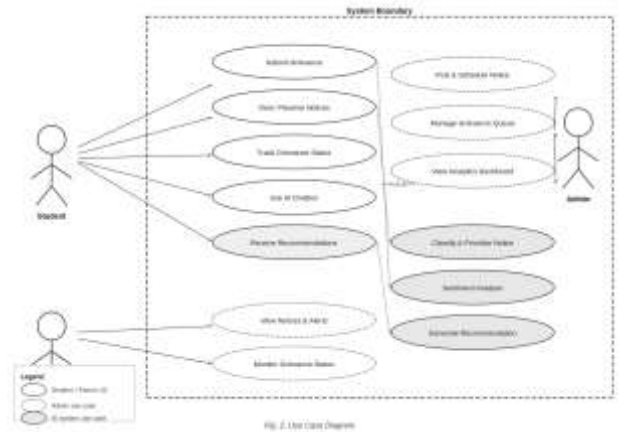


Fig. 2. Use Case Diagram

VI. GRIEVANCE PROCESSING PIPELINE

The grievance processing pipeline is shown in Fig. 3. After a student submits free text, the system applies NLP preprocessing (tokenization, stopword removal, lemmatization) followed by the BERT dual-head classifier which assigns both an intent category and a sentiment score. An urgency decision gate routes high-priority grievances directly to the administrator's escalation queue while low-priority submissions enter a batch-processing standard queue. The administrator reviews, responds, and updates the ticket status, triggering real-time notifications to the student and parent. If the issue remains unresolved, the ticket is re-assigned in a loop. Resolved tickets feed sentiment signals back into the recommendation engine model update pipeline.

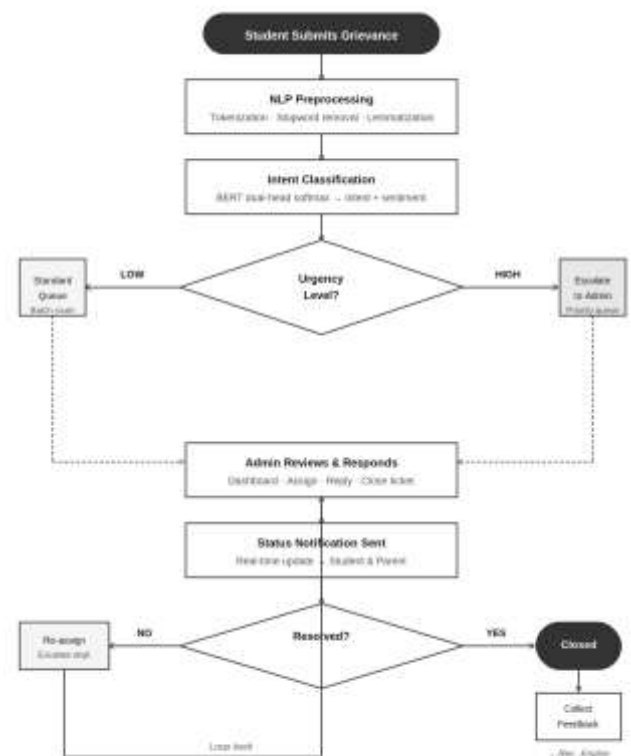


Fig. 3. Grievance Processing Pipeline Flowchart

Fig. 3. Grievance Processing Pipeline Flowchart

VII. RECOMMENDATION ENGINE

The hybrid recommendation engine architecture is illustrated in Fig. 4. Student Profile and Content Corpus serve as dual inputs. The Content-Based Filter computes TF-IDF vectors over student interest tags and ranks corpus items by cosine similarity. The Collaborative Filter constructs a user-item interaction matrix and applies SVD-based matrix factorization augmented by k-nearest-neighbour neighbourhood signals. A Hybrid Combiner fuses the two scores using a weighted sum: $\text{score} = \alpha \cdot \text{CB} + (1-\alpha) \cdot \text{CF}$, where α is tuned dynamically based on the density of interaction data for each student. The ranked output feed includes a brief explanation for each recommendation. A Celery-driven asynchronous feedback loop ingests explicit signals (thumbs up/down, bookmarks) and implicit signals (view duration, click-through rate) to update model weights incrementally, enabling continual learning throughout the academic term.

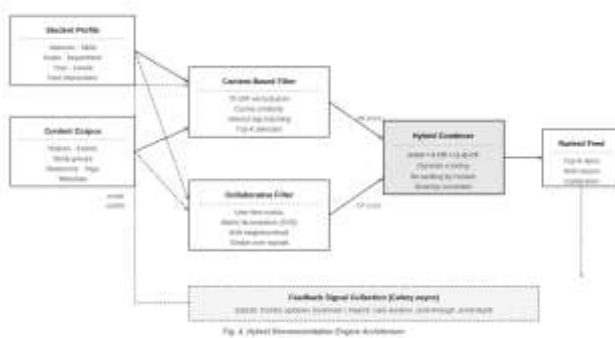


Fig. 4. Hybrid Recommendation Engine Architecture

VIII. DATA MODEL

The entity-relationship diagram in Fig. 5 shows six core entities. The USER entity stores authentication credentials, role, and locale. PROFILE holds student academic context and interests, linked to USER in a one-to-one relationship. NOTICE and GRIEVANCE maintain one-to-many relationships with USER via foreign keys for creator and owner respectively. INTERACTION and FEEDBACK capture behavioral signals that feed the collaborative filtering matrix and the model update loop. PII fields (email, grievance text) are marked with AES-256 at-rest encryption. All entities include soft-delete (deleted_at) and audit columns (created_at, updated_at) to support GDPR-aligned data management.

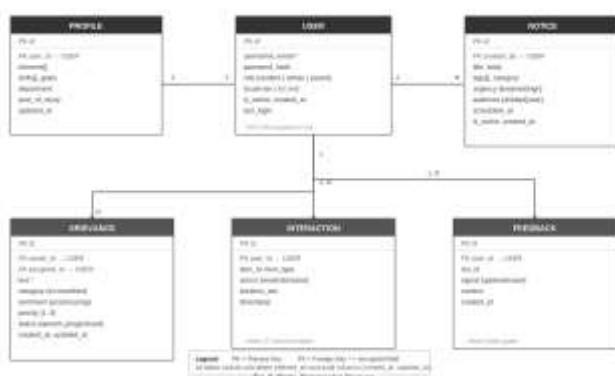


Fig. 5. Entity-Relationship Diagram

IX. ALGORITHMS AND NLP PIPELINE

A. NLP Processing Pipeline (Fig. 6)

Fig. 6 illustrates the seven-stage NLP pipeline. Raw text (grievance text or notice body) passes through preprocessing (tokenization, stopword removal, lemmatization) before feature extraction generates both TF-IDF vectors for the Notice Classifier and contextual embeddings for the BERT encoder. The fine-tuned BERT encoder produces a [CLS] representation h fed to two independent softmax heads:

$$\text{intent} = \text{softmax}(W_i h + b_i); \quad \text{sentiment} = \text{softmax}(W_s h + b_s)$$

Weighted cross-entropy loss handles class imbalance. Data augmentation via synonym substitution and back-translation improves robustness for Hindi and Marathi inputs. The combined output feeds a priority routing module that triggers urgency-based escalation.

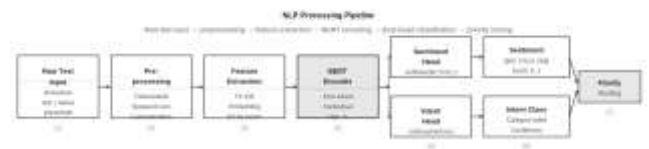


Fig. 6. NLP Processing Pipeline — Input to Classification

Fig. 6. NLP Processing Pipeline — Input to Classification

B. Notice Categorization

TF-IDF feature vectors of notice body text feed a linear SVM optimized for macro-F1 score as the primary classifier. Multinomial Naive Bayes serves as a low-latency fallback when response-time constraints require inference under 50 ms. The SVM achieves 92% classification precision on the held-out test set.

X. IMPLEMENTATION DETAILS

A. Backend and Infrastructure

The backend uses Python 3.11, Django 5.x, and MySQL 8. ORM migrations manage schema evolution. JWT with refresh token rotation and role-based permission classes handle authentication. Celery with Redis orchestrates asynchronous tasks including notification dispatch and periodic model re-training. The AI microservices are containerized independently, allowing model updates without application redeployment.

B. NLP and ML Stack

The NLP stack uses HuggingFace Transformers for the fine-tuned BERT encoder, scikit-learn for SVM and Naive Bayes classifiers, and NLTK/SpaCy for preprocessing. Model artifacts are versioned and served via lightweight REST endpoints. Frontend templates use Django Templates with Bootstrap and Tailwind CSS. Internationalization supports English, Hindi, and Marathi through Django's i18n framework. Security measures include CSRF protection, HTTPS/TLS, parameterized queries, rate limiting, and comprehensive audit logging.

XI. EXPERIMENTAL SETUP AND EVALUATION

A. Dataset

The evaluation dataset comprises 250 anonymized student profiles, 500 institutional notices, and 420 grievance submissions collected over one academic semester at SKNCE, Pune. Data was partitioned 70/15/15 for training, validation, and testing with temporal ordering preserved to prevent data leakage.

B. Baselines

- B1: Rule-based notice sorting (keyword matching, no ML).
- B2: Content-only cosine similarity recommendation.
- B3: Collaborative-only kNN recommendation.
- B4: Proposed hybrid model (this work).
- NLP-B: Logistic Regression on bag-of-words vs. DistilBERT fine-tuning.

C. Results

Table I summarizes AI module performance. The proposed hybrid system (B4) outperformed all baselines on every evaluated metric as shown in Table II. The 87% sentiment detection accuracy exceeds the logistic regression baseline by 15 percentage points. The 92% notice classification precision confirms the suitability of SVM with TF-IDF representations for the academic domain. The 30% reduction in manual administrative workload was computed by comparing mean ticket-handling time before and after AI-assisted categorization and routing.

TABLE I. AI Module Summary

Module	Technique	Outcome
Sentiment/Intent	BERT fine-tuned; dual softmax	87% acc; EN/HI/MR
Notice Categorize	TF-IDF + Linear SVM; NB fallback	92% precision; <50ms
Hybrid Recommend	CF matrix + content cosine; α -fusion	Continual feedback loop
Grievance Priority	Rule heuristics + sentiment weighting	30% admin reduction
AI Chatbot	BM25 FAQ + guarded GenAI fallback	~70% auto-resolved

TABLE II. Experimental Results vs. Baseline

Metric	Proposed	Baseline
Sentiment Detection	87%	72%
Notice Classification	92%	78%
Admin Workload Reduction	30%	—
Chatbot Auto-Resolution	~70%	~40%
Recommendation CTR	High	Low

XII. ETHICAL, PRIVACY, AND FAIRNESS CONSIDERATIONS

The system adheres to principles of data minimization, purpose limitation, and transparency. PII fields are encrypted at rest using AES-256 and in transit via TLS 1.3. Access is

role-gated and comprehensively audited. Models are monitored for disparate impact across student cohorts defined by year, department, and gender where collected with explicit consent. Every personalized recommendation includes a brief rationale (e.g., 'Shown because you follow Topic X') to support user trust and explainability. Students may opt out of personalization and request data deletion at any time. Synthetic data augmentation is used during model development to minimize exposure of real student data in training pipelines.

XIII. LIMITATIONS AND FUTURE WORK

The current system has several limitations that open avenues for future work. First, the models rely on labeled data from a single institution over one semester, which may limit generalization. Second, on-device privacy-preserving inference is not yet implemented, meaning all classification occurs server-side. Third, the chatbot's generative fallback requires human-in-the-loop review to prevent hallucination in high-stakes academic contexts.

Planned future directions include: (1) federated fine-tuning across multiple institutions for collaborative model improvement without centralizing sensitive data; (2) continual learning with concept drift detection to adapt to evolving communication patterns; (3) multimodal grievance input incorporating voice submissions with automatic speech recognition; (4) deep LMS integration for holistic course-aligned learning-path recommendations; and (5) proactive student well-being nudges with counselor oversight and ethics board approval.

XIV. CONCLUSION

This paper presented an AI-Based Student Connect Portal that operationalizes machine learning and natural language processing to make campus communication timely, relevant, and inclusive. The system integrates a hybrid recommendation engine, multilingual grievance intelligence, a conversational AI chatbot, and a secure modular Django architecture. Evaluation across 250+ student profiles demonstrates 87% sentiment detection accuracy, 92% notice classification precision, and a 30% reduction in manual administrative workload compared to rule-based baselines.

Beyond communication improvements, the architecture provides a foundation for data-driven institutional governance. Administrators gain actionable visibility into recurring grievance patterns, engagement levels, and departmental trends. In future iterations, the platform can be extended to a federated multi-institutional network, enabling cross-campus peer mentorship, shared knowledge ecosystems, and collaborative model training. The system stands as a practical blueprint for institutions seeking to augment traditional portals with AI-driven student-centric intelligence.

REFERENCES

- [1] M. Kumar and R. Nandal, "Role of Python in Rapid Web Application Development using Django," SSRN, 2024.
- [2] P. Sahane et al., "Streamlined Learning with AI-ML: An Integrated Campus Management Platform," IJRITCC, 2023.
- [3] Y. Koren, "Collaborative Filtering with Temporal Dynamics," KDD, 2009.
- [4] E. H. Chi, "Explainable Recommendation: A Survey and New Perspectives," Found. Trends HCI, 2020.
- [5] M. Hardt et al., "Equality of Opportunity in Supervised Learning," NIPS, 2016.