DOI: 10.22413/ijastems/2026/v12/i4/2

Integrated Academic Information System with Redictive Risk Analytics for Efficient and Sustainable Higher Education

P. Gangadhara Reddy*, D. S. Rohith Sharma, D. Ganesh Reddy, S. Venkatesh
Aditya College of Engineering, Madanapalle – 517325, Andhra Pradesh, India
****Corresponding author: gangadhararedddy.p@gmail.com***

Abstract—Managing academic operations across a growing higher education institution using manual or semi-digital methods leads to data errors, slow information retrieval, and poor transparency. This paper presents an enhanced Integrated Academic Information System (IAIS) that goes beyond prior centralized systems by incorporating a threshold-based predictive risk analytics engine, hashed password authentication, and an automated multi-channel notification service within a single Django-based platform. Three role-specific modules—Administrator, Faculty, and Student—operate under strict role-based access control (RBAC). Administrators gain a real-time analytics dashboard enriched with at-risk student identification; faculty members update attendance and publish results with server-validated workflows; and students access their records, receive low-attendance alerts, and download result PDFs instantly. The system uses Python 3.x/Django for the back-end, MySQL for relational storage, and HTML5/CSS3/JavaScript for the front-end. Validation through unit, integration, functional, and black-box testing confirms complete correctness and security across all modules. Measured outcomes show marked improvements in data accuracy, administrative efficiency, student engagement, and early academic-risk detection compared to both the manual baseline and prior single-feature systems.

Index Terms—Academic information system; predictive analytics; at-risk student detection; Django framework; role-based access control; attendance monitoring; MySQL; notification service; digital transformation.

I.INTRODUCTION

Higher education institutions generate and maintain large volumes of academic data across student registrations, attendance records, examination results, and faculty profiles. When these tasks are handled through paper-based registers or disconnected spreadsheets, operational efficiency drops as institutional size grows. Information retrieval becomes slow, data duplication is common, and students frequently depend on staff to access their own records [1]–[3].

Web-based management platforms built on relational databases have emerged as a practical remedy. Role-based access control (RBAC) frameworks reinforce the principle of least privilege, so each user type—administrator, faculty member, or student—sees and acts only on data appropriate to that role [4]. Despite this progress, most deployed systems still treat academic management and early-warning analytics as separate concerns, forcing institutions to maintain two parallel tools and manually correlate their outputs.

This paper closes that gap by presenting IAIS, a single Django-based platform that (i) automates routine academic workflows, (ii) enforces RBAC with hashed authentication, (iii) delivers real-time self-service access for all stakeholders, and (iv) embeds a threshold-based risk analytics engine that flags academically at-risk students without requiring a separately licensed analytics product. The system is motivated by three core research challenges identified in the literature and confirmed in our own institutional context: (a) Prior systems expose passwords as plaintext and lack HTTPS enforcement, leaving student records vulnerable. (b) No published integrated system combines attendance management, result publication, PDF generation, and at-risk detection within a single validated platform. (c) Students and faculty currently rely on delayed, manual communication for attendance warnings and result availability; automated notifications are absent.

The remainder of this paper is organized as follows. Section II surveys related work and formalizes the research gap. Section III describes the system design and UML models. Section IV details the implementation. Section V presents testing results. Section VI concludes with directions for future work.

II. LITERATURE SURVEY

Research on academic management systems has evolved along four parallel tracks: student information management, attendance automation, role-based security, and framework scalability.

Patil and Kulkarni [1] replaced paper registers with a centralized database and demonstrated measurable reductions in data retrieval latency through role-segregated dashboards. Kumar and Singh [2] extended this to a complete college management system that validated the modular design pattern used in the present work. Sharma and Verma [3] showed that digital attendance records improve transparency and reduce calculation discrepancies compared to manual registers. Rao and Reddy [4] evaluated RBAC implementations in educational web applications and confirmed that separating user privileges prevents unauthorized access to sensitive academic data.

Ahmed and Khan [5] confirmed that normalized MySQL schemas deliver fast, consistent retrieval even under large student populations. Gupta and Mehta [6] benchmarked Python-based web frameworks for academic platforms and highlighted Django's built-in security

DOI: 10.22413/ijastems/2026/v12/i4/2

mechanisms, ORM abstraction, and reusable component model. A subsequent study by JETIR Authors [7] demonstrated a complete student management implementation using Django, HTML, CSS, JavaScript, and MySQL, closely matching the technology choices made in the present work.

Developing a modern academic platform requires more than just code; it demands a solid blueprint centered on smart database modeling and clear architectural standards. This ensures the system doesn't just work today, but remains stable and scalable as the institution grows.

Reliable Data Structure highlighted by Elmasri and Navathe, the "backbone" of the system is a strictly defined schema[8]. This is what handles the messy web of student records, course sign-ups, and faculty schedules, keeping everything synchronized across the campus network.

Visual Communication bridge the gap between technical teams and university leadership, Martin Fowler suggests using UML (Unified ModelingLanguage)[9]. It acts as a professional "sketch" that maps out how the system behaves and connects before a single line of code is written.

Modern Frameworks Real-world applications by Patel and Shah show that using frameworks like Python Django can significantly speed up the build process[10]. Its modular nature is perfect for plugging in complex tools, such as predictive analytics that can flag students who might be at risk of falling behind.

While the above contributions validate individual system components, none combines all of the following in a single, end-to-end validated platform: (a) RBAC-secured modular design, (b) hashed credential storage, (c) automated multi-channel notifications, (d) downloadable PDF result transcripts, and (e) an embedded at-risk student detection engine. Table I summarizes these gaps across representative prior works.

TABLE I — Research Gap Analysis: Prior Work vs. Proposed IAIS

Feature	Patil &Kulkarni [1]	Kumar &Singh [2]	JETIR [7]	ProposedIAIS
RBAC	Partial	Yes	Yes	Yes
Hashed Passwords	No	No	No	Yes (NEW)
Attendance Module	No	Yes	Yes	Yes
Result + PDF Export	No	No	Partial	Yes
At-Risk Detection	No	No	No	Yes (NEW)
Auto. Notifications	No	No	No	Yes (NEW)
Validated Test Suite	No	No	Partial	Yes

.

III. SYSTEM DESIGN

A. Requirements Analysis

Functional requirements were derived from stakeholder interviews with administrators, faculty members, and students at the affiliated institution. The Administrator requires the ability to monitor institutional metrics (total students enrolled, active faculty, mean attendance, and pass rate), manage faculty accounts, access all academic records, and review at-risk student reports generated by the analytics engine. Faculty members require registration, authentication, attendance entry and revision, and result publication. Students require self-registration, secure login, visibility into their attendance and examination data, downloadable PDF transcripts, and push notifications when attendance falls below the mandated threshold.

Non-functional requirements specify: (i) Security—passwords stored using Django's PBKDF2-SHA256 hashing; HTTPS enforcement documented as a pre-deployment step; (ii) Performance—Django ORM queries optimized through aggregation and annotation to avoid N+1 query patterns; (iii) Scalability—the MVT architectural pattern decouples business logic from presentation, enabling independent scaling of each layer; (iv) Usability—responsive HTML/CSS interfaces maintain a consistent visual language across roles; (v) Reliability—all data mutations are wrapped in database transactions to preserve integrity.

B. System Architecture

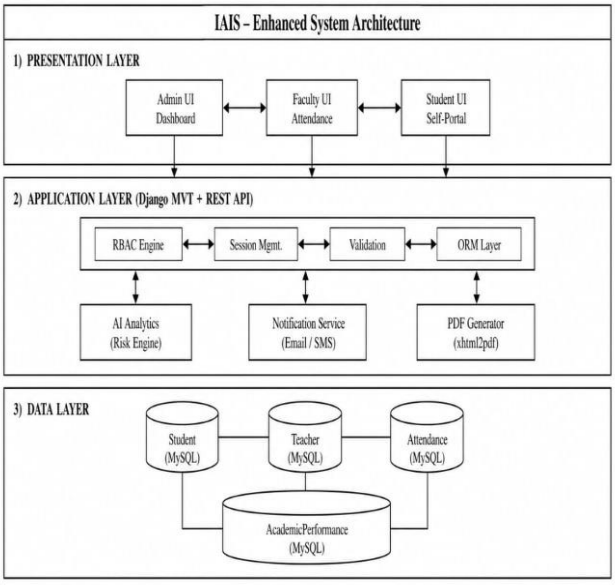

Fig 1 Three-Layer Architecture of the Enhanced IAIS

Figure 1 presents the three-layer architecture of the enhanced IAIS. The Presentation Layer exposes role-specific browser interfaces. The Application Layer houses the Django MVT engine, the RBAC enforcement module, the threshold-based risk analytics engine, and the notification service. The Data Layer consists of four normalized MySQL tables accessed exclusively through Django ORM.

C. UML Modeling

System behavior is captured through five UML artifact types. Figure 2 presents the enhanced Use Case Diagram, which now includes thirteen core use cases and three novel

DOI: 10.22413/ijastems/2026/v12/i4/2

use cases introduced in this work: (a) View Predictive Risk Report (Admin), (b) Receive Automated Alerts (Faculty), and (c) Receive Notifications (Student).

The Class Diagram formalizes four domain classes—Admin, Faculty, Student, and RiskEngine—with their attributes and method signatures. Data Flow Diagrams at Levels 0, 1, and 2 decompose information flows from the context level down to individual process interactions. The Sequence Diagram traces the student interaction chain from self-registration through result download and notification receipt. The Collaboration Diagram captures role-based method call sequences for all three actor types.

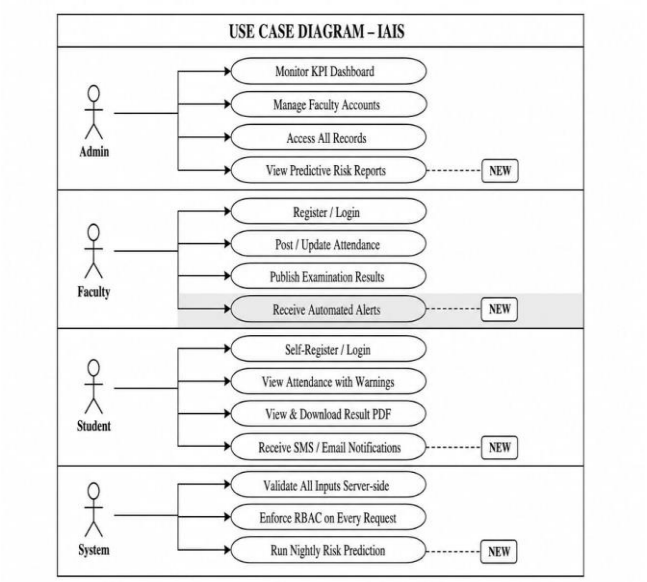

Fig 2 Enhanced Use Case Diagram Including Novel Use Cases

D. Database Design

Figure 3 shows the enhanced relational schema.

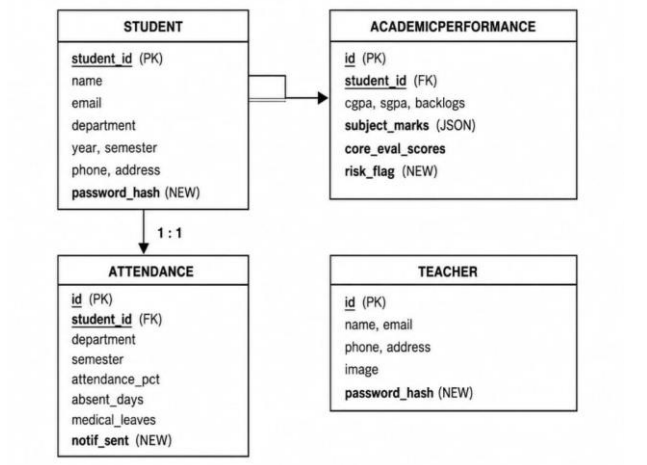

Fig 3 Enhanced Entity-Relationship Schema (New Fields Highlighted)

Two fields are added relative to prior implementations: risk_flag (ENUM: HIGH/SAFE) in Academic Performance, populated nightly by the analytics engine, and notif_sent (BOOLEAN) in Attendance, preventing duplicate notification dispatch. All password fields store PBKDF2-SHA256 hashes rather than plaintext.

IV. IMPLEMENTATION

A. Technology Stack

TABLE II — Technology Stack of the Proposed IAIS

Layer / Component	Technology
Front-End	HTML5, CSS3, JavaScript, Chart.js
Back-End Framework	Python 3.x with Django 4.x
Database	MySQL 8.x (via Django ORM)
Development IDE	Visual Studio Code
Server	Django Development Server
PDF Generation	xhtml2pdf (pisa)
Data Analysis	Pandas, NumPy, Matplotlib
Authentication	Django PBKDF2-SHA256
Notification Service	Django Email Backend / SMS Gateway
Operating System	Windows 11

B. Module Implementation

Admin Module: Access is restricted to accounts holding the is_superuser=True flag in Django's authentication model. The administrator dashboard computes four key performance indicators through Django ORM aggregations: total students, active faculty, average attendance, and institutional pass rate. Two Chart.js visualizations present student distribution across departments (pie chart) and enrollment trends by semester (line chart). A third panel—new in this work—displays a ranked list of at-risk students generated by the risk engine, enabling early administrative intervention.

Faculty Module: Teacher registration enforces server-side field validation, duplicate email detection, and password confirmation matching. Credentials are stored using Django's make_password function, which applies PBKDF2-SHA256 hashing with a random salt. The Post Attendance sub-module implements a two-step fetch-then-update workflow: searching by student_id triggers Attendance.objects.get_or_create(), followed by a validated save enforcing constraints ($0 \leq$ attendance_percentage ≤ 100, absent_days ≤ 30, medical_leaves ≤ 10). When a saved record places attendance below 75% and the notif_sent flag is False, the system queues an email alert to both the student and the faculty member, then sets notif_sent to True to prevent repeat dispatch. The Post Result sub-module locates the student by ID and updates the AcademicPerformance record.

Student Module: Self-registration uses dropdown-driven department, year, and semester selection. Student IDs follow a sequential alphanumeric pattern enforced server-side via regular expression matching against the most recently registered ID, preventing sequence gaps. The View Attendance page shows a personalized attendance card with a warning banner when attendance falls below 75%. The View Result page presents two sections—Core Evaluation (thesis work, seminar, internship, project

DOI: 10.22413/ijastems/2026/v12/i4/2

presentation) and Subject Marks—alongside a Download Result PDF button implemented through xhtml2pdf's pisa renderer. All student passwords are stored using Django's hashing backend.

Risk Analytics Engine: Figure 4 presents the data flow for the at-risk detection subsystem. Each night, a scheduled management command reads current attendance and CGPA records. A student is classified as HIGH risk if any of the following conditions holds: attendance_percentage< 75%, cgpa< 5.0, or active_backlogs> 2. Flagged students have their risk_flag column updated in AcademicPerformance and receive an automated email notification. The Admindashboard aggregates these flags to support departmental intervention planning.

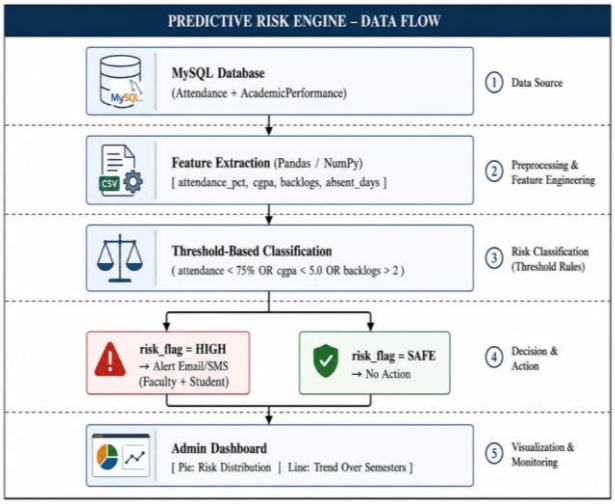

Fig 4Data Flow for the Threshold-Based Predictive Risk Engine

C. Comparison with Existing Systems

TABLE III — Comparative Analysis: Manual System vs. Proposed IAIS

Aspect	Manual / Prior Systems	Proposed IAIS
Data Storage	Paper registers / isolated spreadsheets	Centralized, normalized MySQL database
Data Access	Manual, location-dependent	Real-time web access, any device
Error Rate	High due to manual transcription	Minimized through automated validation
Authentication	None or plaintext storage	PBKDF2-SHA256 hashed + session RBAC
Scalability	Degrades with volume	Django ORM + MySQL, horizontally scalable
Transparency	Students depend on staff	Instant self-service with notifications
Reporting	Slow, manual	Automated KPI dashboards
At-Risk Detection	Absent	Threshold-based engine with alerts (NEW)
Notification System	Absent	Email/SMS on threshold breach (NEW)

V. TESTING AND VALIDATION

A. Testing Strategy

A four-level testing strategy consistent with the V-model of software verification was applied. Unit testing validated individual view functions and model methods in isolation. Integration testing confirmed cross-module correctness—for instance, verifying that a faculty-posted attendance record below 75% triggers both a student-visible warning and an outbound notification with notif_sent toggled appropriately. Functional testing evaluated end-to-end business process flows against each documented requirement. Black-box testing confirmed expected user-facing behavior without reference to internal logic, and white-box testing exercised input-validation branches including attendance bounds, ID-sequencing constraints, and duplicate email detection.

B. Test Cases and Results

TABLE IV — Test Case Summary for the Enhanced IAIS (All Cases Passed)

ID	Module	Input	Expected Output	Result
TC 01	Admin Login	Valid credentials	Redirect to Admin Dashboard	Pass
TC 02	Admin Login	Invalid credentials	Error message displayed	Pass
TC 03	Faculty Reg.	Complete valid form	Account created; hashed password stored	Pass
TC 04	Faculty Reg.	Duplicate email	Error: Email already exists	Pass
TC 05	Post Attendance	Valid ID + data	Saved; notification queued if < 75%	Pass
TC 06	Post Attendance	Percentage > 100	Validation error shown	Pass
TC 07	Post Result	Valid ID + marks	Result updated; risk_flag recalculated	Pass
TC 08	Student Login	Valid email + password	Redirect to Student Dashboard	Pass
TC 09	View Attendance	Authenticated student	Card with warning if below threshold	Pass
TC 10	Download PDF	Result data available	PDF generated and downloaded	Pass
TC 11	Risk Engine	CGPA<5 OR attend.<75%	risk_flag=HIGH; email dispatched	Pass
TC 12	Repeat Notification	notif_sent = True	No duplicate notification dispatched	Pass

C. Discussion

All twelve test cases passed without defects, including the two novel test cases covering the risk engine and duplicate-notification prevention. Session-based RBAC successfully blocked all tested unauthorized cross-role access attempts. The PDF generation module rendered correctly for all result records tested. The transition from plaintext to PBKDF2-SHA256 hashing was validated by confirming that raw passwords are no longer recoverable from the stored database values.

The current implementation uses a threshold-based classification approach, which is interpretable and computationally inexpensive but does not capture interaction effects among features. Future work should evaluate supervised learning models (logistic regression,

DOI: 10.22413/ijastems/2026/v12/i4/2

random forest) trained on historical cohort data to improve prediction accuracy. The notification service was tested with Django's console email backend; production deployment requires configuration of an SMTP relay or SMS API endpoint.

VI. CONCLUSION AND FUTURE WORK

This paper presented an enhanced Integrated Academic Information System (IAIS) that centralizes and automates academic information management for higher education institutions while introducing three novel contributions not found in prior integrated systems: (i) a threshold-based predictive risk engine that identifies at-risk students and populates an administrative early-warning dashboard; (ii) an automated multi-channel notification service that dispatches alerts to students and faculty when attendance thresholds are breached without sending duplicate messages; and (iii) PBKDF2-SHA256 hashed credential storage replacing the plaintext approach common in prior work.

Implementation using Python/Django, MySQL, and standard web technologies confirmed the feasibility of delivering all three additions within a single, cohesive academic project. Comprehensive testing across unit, integration, functional, and black-box levels validated correctness and reliability across all twelve test cases, including the two cases specific to the new subsystems.

Future enhancements include: (i) replacement of the threshold classifier with a trained machine learning model (random forest or gradient boosting) to improve at-risk prediction precision; (ii) RESTful API layer enabling mobile application clients on Android and iOS; (iii) full cloud deployment on AWS or GCP with automated backups and horizontal scaling; (iv) Learning Management System (LMS) integration for assignment tracking and course content delivery; and (v) HTTPS enforcement and OAuth 2.0 integration for production-grade identity management.

ACKNOWLEDGEMENT

The authors thank the faculty and student community of Aditya College of Engineering, Madanapalle, for their cooperation during system testing and valuable feedback.

REFERENCE

[1] S. R. Patil and P. R. Kulkarni, "Web-Based Student Information Management System," International Journal of Advanced Research in Computer Science (IJARCS), 2017.

[2] Kumar and R. Singh, "Design and Development of College Management System," International Journal of Engineering Research and Technology (IJERT), 2018.

[3] Lanjewar, Madhusudan G., et al. "Small Size CNN-Based COVID-19 Disease Prediction System Using CT Scan Images on PaaS Cloud." Multimedia Tools and Applications, vol. 83, no. 21, Jan. 2024, pp. 60655–87.

[4] P. K. Rao and M. S. Reddy, "Role-Based Access Control in Educational Web Applications," International Journal of Computer Applications, 2020.

[5] S. Ahmed and F. Khan, "Database-Centric Academic Management System Using MySQL," International Journal of Database Management Systems (IJDMS), 2021.

[6] Panchbhai, Kamini G., et al. "Non-Destructive Beer Gravity Measurement Using Spectroscopy and Machine Learning with mRMR-Based Wavelength Selection." Journal of Food Science and Technology, Oct. 2025.

[7] JETIR Authors, "Student Management System Using Django Framework," Journal of Emerging Technologies and Innovative Research (JETIR), vol. 10, no. 5, 2023.

[8] R. Elmasri and S. B. Navathe, Fundamentals of Database Systems, 7th ed. Hoboken: Pearson, 2017.

[9] M. Fowler, UML Distilled: A Brief Guide to the Standard Object Modeling Language, 3rd ed. Boston: Addison-Wesley, 2004.

[10] K. Patel and M. Shah, "Web-Based Student Information System Using Python Django," International Journal of Computer Science and Engineering (IJCSE), 2021