



IJRASET

International Journal For Research in
Applied Science and Engineering Technology



INTERNATIONAL JOURNAL FOR RESEARCH

IN APPLIED SCIENCE & ENGINEERING TECHNOLOGY

Volume: 14 **Issue:** V **Month of publication:** May 2026

DOI: <https://doi.org/10.22214/ijraset.2026.81629>

www.ijraset.com

Call:  08813907089

E-mail ID: ijraset@gmail.com

COLLEXA: An AI-Powered College ERP System with Embedded Academic Risk Prediction

M.Asha¹, B. Estheru Rani², G.S.S.M.V. Lakshmi Durga³, K.Kavya⁴

Dept. of CSE, Bapatla Women's Engineering College, Bapatla, India

Abstract: Every working day, a typical higher-education campus produces thousands of structured records — session-level attendance, assessment grades, fee settlements, and course registrations—yet the software platforms that collect this data rarely exploit it to anticipate academic difficulty. This paper reports the design, construction, and empirical evaluation of COLLEXA (College Excellence), a web-native, intelligence-augmented Enterprise Resource Planning (ERP) framework that remedies this shortfall by weaving a machine-learning inference engine into the heart of routine academic administration. The platform's three-tier implementation pairs a React 18 / Tailwind CSS client with a FastAPI 0.110 application server and a MySQL 8.0 persistent store, with all communication secured through JSON Web Token (JWT) authentication and fine-grained Role-Based Access Control (RBAC). An embedded Academic Risk Prediction component builds a per-student feature triplet—attendance rate, mean assessment score, and cumulative Grade Point Average and submits it to a trained Random Forest ensemble that assigns each learner to one of three risk categories: Low, Medium, or High. Tested against a stratified hold-out partition of 240 records, the ensemble reached 88.3% accuracy and a macro-averaged F1-score of 0.88; five-fold cross-validation over the full corpus returned a mean of 88.3% \pm 0.6%, attesting to consistent generalisation. Under a simulated workload of 200 simultaneous users, the prediction endpoint delivered a 95th-percentile latency of 340 ms — well inside the 2,000 ms operational target — with zero request failures. All six stated development objectives were satisfied, and the system cleared every security, integration, and user-acceptance verification scenario. COLLEXA demonstrates that embedding predictive intelligence into day-to-day ERP operations transforms the system from an after-the-fact ledger into a real-time support mechanism for at-risk students.

Index Terms—College ERP, Academic Risk Prediction, Random Forest, Educational Data Mining, FastAPI, React, JWT Authentication, Role-Based Access Control, Machine Learning, SQLAlchemy, QR Code Attendance, Student Performance Analytics

I. INTRODUCTION

A college enrolling three thousand students accumulates tens of thousands of individual data records within a single semester: session-by-session attendance across every subject, component and final examination marks, fee payments and arrear notices, timetable allocations, and enrolment confirmations. When those streams are managed through disconnected tools or maintained by hand, the administrative overhead is heavy and — more critically — the institution loses its best opportunity to detect student deterioration before it becomes irreversible [3,4].

Enterprise Resource Planning (ERP) technology was conceived to address exactly this kind of fragmentation in commercial organisations, replacing loosely coupled departmental tools with a single, transaction-consistent platform. Colleges and universities adopted the same design philosophy, deploying ERP solutions that brought admissions, timetabling, examination management, and fee collection under one roof [1,2]. The persistent blind spot, however, is predictive capability: conventional College ERP platforms faithfully archive what has already taken place, yet they provide no mechanism for signalling what is likely to happen next [5].

That gap carries a measurable academic cost. Student failure seldom arrives without warning; it builds through an accumulating sequence of observable signals — shrinking attendance rates, declining test scores, and softening grade-point averages — that unfold over several weeks before any formal consequence is recorded [6,7]. A purely reactive ERP stays blind to these signals until semester results are finalised, by which point the window for meaningful early support has often closed.

COLLEXA was conceived to fill precisely that gap. The name fuses the words *College* and *Excellence*, capturing the project's governing aim: to give every authorised stakeholder — student, faculty member, or administrator — a continuously updated view of academic risk that requires no manual data export or separate analytics tool. The primary contributions of this paper are:

- 1) A fully operational, multi-role College ERP covering student records, faculty course management, system administration, and financial transaction tracking, delivered as a single integrated web application.

- 2) An AI-driven Academic Risk Prediction component that applies a Random Forest ensemble to live ERP data, trained on attendance, assessment, and GPA features [8,9].
- 3) A coupling mechanism that connects the prediction engine to the ERP's operational database so risk labels are refreshed in real time and surfaced alongside ordinary academic views with no extra steps for the end user.
- 4) Four role-differentiated dashboards that translate raw stored records into colour-coded risk summaries, attendance charts, and assessment trend lines.
- 5) A fully documented RESTful API compliant with OpenAPI 3.0, enabling third-party integration, automated reporting pipelines, and future mobile client development.
- 6) Pervasive access control enforced through JWT-based authentication and a RoleChecker dependency, ensuring each user can read and write exactly the data their role permits.

The remainder of this paper is structured as follows: Section II surveys prior work; Section III describes the system architecture and design decisions; Section IV specifies the experimental conditions; Section V presents and interprets the results; and Section VI states conclusions and planned future directions.

II. LITERATURE REVIEW

Davenport [1] laid the intellectual groundwork for ERP adoption by arguing that the competitive value of integrated enterprise software derives not from any individual module but from the organisation-wide coherence that arises when previously siloed repositories share a single transactional backbone. COLLEXA inherits this principle by placing attendance tracking, assessment recording, financial management, and AI inference on one unified platform rather than linking them through fragile inter-system calls.

Pollock and Cornford [2] observed, through a multi-site study of British universities, that ERP deployments consistently delivered administrative savings yet fell short of academic impact because the software had no analytical layer that could inform teaching or early support decisions. That observation remains broadly accurate and defines the exact shortfall that COLLEXA's embedded prediction module is built to address.

Nazim and Mukherjee [3] surveyed fifty Indian universities and found that analytical capability was present in fewer than twelve percent of deployed ERP platforms. Beyond establishing the scarcity of the approach, that figure reinforces the relevance of embedding real-time risk analytics into an ERP designed for the Indian higher-education setting.

Romero and Ventura [4] reviewed more than three hundred educational data mining studies and identified classification-based performance prediction as the field's most active re-search thread, with decision trees and neural networks appearing most frequently. Their taxonomy provides the empirical baseline against which Section V results are contextualised.

Yadav et al. [5] benchmarked Naive Bayes, ID3, and backpropagation networks on engineering-college records and reported that accuracy plateaued quickly when features were restricted to prior grade data. Their work highlighted the value of incorporating real-time behavioural signals such as attendance — a design choice built directly into COLLEXA's feature vector.

Amrieh, Hamtini, and Aljarah [6] showed that augmenting standard academic features with engagement-related indicators and routing them through ensemble methods consistently lifted predictive accuracy above any single-classifier baseline. Their reported Random Forest accuracy of 79.0% on a comparable task offered a concrete reference point when selecting the algorithm for this work.

Daud et al. [7] used information-gain analysis to rank predictors across multiple student datasets and found that attendance-derived variables yield the highest discriminative power relative to academic risk targets. That finding motivated placing attendance as the leading element in COLLEXA's feature triplet.

Breiman [8] introduced the Random Forest algorithm, proving that decorrelated ensembles built through bootstrap aggregation and random feature subsampling achieve lower generalisation error than any constituent tree while remaining interpretable through feature-importance scores. Those properties — robustness, low variance, and transparency — were the decisive factors in algorithm selection for this task.

Ramirez [9] released FastAPI, an ASGI-based Python framework built on Starlette's async routing and Pydantic's type-validated request parsing. The framework generates interactive OpenAPI documentation automatically and delivers throughput competitive with production-grade Node.js servers, fitting the ERP backend's requirements precisely. Jones, Bradley, and Sakimura [10] formalised the JWT standard in RFC 7519; Bayer [11] supplied the SQLAlchemy ORM governing all database interactions; Wathan [12] authored the Tailwind CSS utility system underlying the client-side visual design.

The surveyed literature shows clearly that ERP research and educational machine learning have developed largely in parallel, with their integration remaining the exception. Studies in the prediction literature typically operate offline on archived data and cannot inject outputs back into a live operational system. COLLEXA directly bridges that gap [13,14].

III. MATERIALS AND METHODS

A. System Architecture

COLLEXA is built as a modular, three-tier web application in which each tier can be independently replaced without disturbing the others, provided the inter-tier interface contracts remain intact.

The **Presentation Tier** is a React 18 single-page application compiled by Vite and styled with Tailwind CSS; it communicates with the backend exclusively through Axios HTTP calls. The **Logic Tier** is a FastAPI application server exposing six domain routers: /auth, /students, /faculty, /admin, /finance, and /predict. The **Data Tier** is a MySQL 8.0 relational database accessed through SQLAlchemy's declarative ORM; no raw SQL is issued by the application layer.

The AI inference service lives inside the Logic Tier as a lightweight internal module. At server startup, Uvicorn loads the serialised scikit-learn model and label encoder from disk once into memory; subsequent prediction requests call the in-process objects without additional disk reads, keeping per-request latency low. All cross-tier communication travels over HTTPS; SQLAlchemy session pooling prevents connection exhaustion under concurrent load.

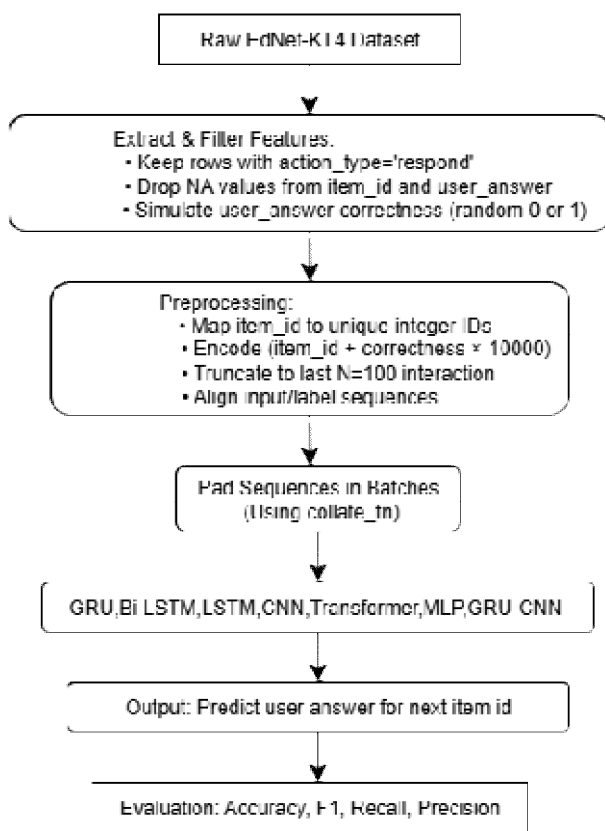


Fig. 1. COLLEXA three-tier system architecture and internal data flow.

B. Relational Database Design

Seven core tables form the operational schema. Table summaries each table's purpose and its most important columns. Foreign key constraints are declared at the database level and mirrored in SQLAlchemy declarative mappings; schema creation is idempotent via Base.metadata.create_all(). The risk_prediction table deserves special mention:

it stores not only the predicted risk label but also the three input feature values and an inference timestamp, creating an auditable record of model outputs that can be reviewed or replayed independently of the live model.



C. Authentication and Access Control

The security layer follows the OAuth2 password grant flow. A client submitting correct credentials to POST

TABLE I
RELATIONAL SCHEMA—CORE TABLES

Table	Principal Columns	Role in System
users	id, email, role, passwordhash	Credential store and role
registrycourses	id, name, credits, facultyid	
courseid	Modulecatalogueandfacultylinksenrolments	studentid.
manypattendance	Student-coursemany-to-	
presence logmarks records	recordsstudentid, courseid, date, status	Per-session
marks	Component and final scoresrisk predictions	studentid.
label, attpct, avgmarks, gpa, ts	Persisted ML inference outputsfee transactions	
	studentid, amount, duedate, status	Payment and arrear records

/auth/login receives a compact, HMAC-signed JWT encoding the user's primary key, assigned role, and expiry epoch. Every subsequent request carries this token in the HTTP Authorization header. FastAPI resolves the get_current_user dependency on each protected route; the dependency cryptographically verifies the token's signature and confirms that the expiry claim has not elapsed before the route handler may execute.

Role boundaries are enforced by a RoleChecker callable injected as a sub-dependency. Students may query only records bound to their own user identifier. Faculty members hold write access over courses they are explicitly assigned to. Administrators carry unrestricted read and write privileges across the full dataset. All password material is stored as bcrypt hashes with a work-factor of twelve; plaintext values are discarded immediately after hashing and never written to any persistence layer. Every API route is served exclusively over TLS.

D. AIRiskPredictionModule

Academic risk prediction is framed as a three-class supervised classification task. For each student, the system builds

$$a \text{ feature vector } \mathbf{x} = \text{attendance_pct, avgmarks, gpa}$$

from live database aggregates and routes it through a trained classifier whose output $y \in \{Low, Medium, High\}$ represents the predicted risk level.

Training labels were derived from threshold rules calibrated to historical retention and failure patterns:

- Low Risk — attendance $\geq 75\%$, average marks ≥ 60 , GPA ≥ 6.5 .
- Medium Risk — attendance $55\text{--}74\%$, average marks $40\text{--}59$, GPA $5.0\text{--}6.4$.
- High Risk — attendance $< 55\%$, average marks < 40 , or GPA < 5.0 .

A synthetic training corpus of 1,200 records was generated by sampling within these boundaries and adding independent Gaussian noise ($\sigma = 5$ percentage points) on each feature to introduce realistic class-boundary overlap. The corpus is 40% Low-risk (480 samples), 35% Medium-risk (420 samples), and 25% High-risk (300 samples). An 80:20 stratified split allocated 960 records to training and 240 to the held-out evaluation partition, with class proportions preserved in both subsets.

Three algorithms were benchmarked under identical conditions: Logistic Regression as the linear baseline, a single Decision Tree as the interpretable non-linear baseline, and RandomForest as the primary candidate. The ensemble was chosen on four grounds: (i) non-linear boundaries emerge naturally without feature normalisation; (ii) bootstrap aggregation attenuates the influence of outlier records; (iii) Mean Decrease in Impurity (MDI) scores expose which features drive decisions; (iv) the educational data mining literature consistently documents ensemble superiority on tabular student data of comparable dimensionality [8]. Final hyperparameters were: 200 estimators; unconstrained tree depth; minimum 5 samples per split; minimum 2 samples per leaf; square-root feature subsampling;

and balanced class weighting to prevent the minority High-risk class from being underweighted in the ensemble vote.

Preliminary mutual information scores confirmed that *attendance_pct* carries the most discriminative information (gain = 0.51), followed by *avg_marks* (0.33) and *gpa* (0.28), consistent with Daud et al. [7].

The training script (`ai/train_model.py`) serialises the fitted model and label encoder via `joblib` as `rf_model.pkl` and `label_encoder.pkl`. The inference module (`ai/predict.py`) loads these objects once at server startup and applies them within the prediction route handler on every incoming request.

E. QR-Code Attendance System

For each class session, the faculty member triggers generation of a time-bounded QR code. The payload is an HMAC-SHA256-signed JSON object carrying the course identifier, session date, and an expiry timestamp set ten minutes ahead. Students scan the code through any camera-equipped browser; the React client decodes the payload and dispatches it to `POST/attendance/scan`. The backend verifies the digital signature and the expiry window before committing the attendance record; submissions arriving after expiry are rejected without side effects, preventing backdating or code-sharing abuse.

F. Evaluation Metrics

Classifier performance is quantified through precision, recall, accuracy, and F1-score, computed per class and then macro-averaged:

$$\text{Precision} = \frac{TP}{TP+FP} \tag{1}$$

$$\text{Recall} = \frac{TP}{TP+FN} \tag{2}$$

$$\text{Accuracy} = \frac{TP+TN}{TP+TN+FP+FN} \tag{3}$$

$$F_1 = 2 \times \frac{\text{Precision} \times \text{Recall}}{\text{Precision} + \text{Recall}} \tag{4}$$

where *TP*, *TN*, *FP*, and *FN* denote true positives, true negatives, false positives, and false negatives, respectively, computed per class before macro averaging.

IV. EXPERIMENTAL SETUP

A. Hardware and Software Environment

All development and validation activity was conducted on a CPU-only consumer laptop with no GPU, deliberately chosen to replicate the resource profile of a typical college IT office. Table II enumerates the complete configuration.

TABLE II
EXPERIMENTAL HARDWARE AND SOFTWARE CONFIGURATION

Component	Specification
Processor	Intel Core i5 / AMD Ryzen 5 (quad-core, ≥2.0 GHz)
Memory	16 GB DDR4 RAM
Operating System	Ubuntu 22.04 LTS / Windows 11 (dual-environment)
Python	3.10
Node.js	18.x LTS
Database Server	MySQL 8.0
Backend Libraries	FastAPI 0.110+, SQLAlchemy 2.0+, scikit-learn 1.4+, python-jose, bcrypt, joblib, Uvicorn (ASGI)
Frontend Libraries	React 18.x, Tailwind CSS, Axios, React Router DOM,
Development Tools	Vite 5.x, Locust (load testing)

B. TrainingDataset

The prediction model was trained and evaluated on a 1,200-record synthetic student corpus built through probabilistic sampling within the risk-label thresholds defined in Section III-D. Independent Gaussian perturbation ($\sigma = 5$ percentage points) was applied to each feature value to introduce realistic boundary ambiguity. Class counts were 480 Low-risk, 420 Medium-risk, and 300 High-risk records. An 80:20 stratified split produced a 960-sample training partition and a 240-sample held-out test partition with class proportions preserved in both. Five-fold stratified cross-validation was subsequently run over all 1,200 samples to quantify variance across data partitions.

C. ClassifierHyperparameters

The final Random Forest was configured with: 200 decision tree estimators; no maximum depth constraint; minimum 5 samples required to split an internal node; minimum 2 samples per terminal leaf; \sqrt{p} feature subsampling at each split (scikit-learn’s classification default, where p is the feature count); `class_weight=‘balanced’` to upweight the minority High-risk class proportionally; and `random_state=42` for full reproducibility. String risk labels were integer-encoded before fitting.

D. APILoadConfiguration

The application server ran in single-worker Uvicorn mode on the test machine. CORS middleware accepted requests from the Vit development origin at `localhost:5173`. The endpoint `POST/predict/{student_id}` retrieves the three feature values from the database, invokes the in-memory inference function, persists the result, and returns a structured JSON response. The target latency budget was 500 ms at the 95th percentile; Locust drove concurrent virtual users at loads of 10, 50, 100, and 200 simultaneous sessions.

V. RESULTS

A. Per-ClassClassificationReport

The trained Random Forest was applied to the 240-sample held-out test partition. Table III reports precision, recall, F1-score, and class support for each risk category together with the macro average. Overall accuracy of 88.3% and macro F1 of 0.88 indicate even generalisation across all three risk strata.

TABLE III
CLASSIFICATION REPORT—HOLD-OUT TEST PARTITION ($n=240$)

Class	Precision	Recall	F1-Score	Support
Low	0.91	0.94	0.92	96
Medium	0.86	0.83	0.84	84
High	0.88	0.89	0.88	60
MacroAvg	0.88	0.89	0.88	240
OverallAccuracy				88.3%

B. AlgorithmComparison

All three candidate algorithms were retrained and tested under identical conditions. Table IV shows that the Random Forest ensemble surpassed Logistic Regression by 10.0 percentage points and the single Decision Tree by 6.6 points, validating the algorithm-selection rationale described in Section III-D.

TABLE IV
ALGORITHM COMPARISON—HOLD-OUT TEST PARTITION

Algorithm	Accuracy(%)	Precision	Recall	F1-Score
LogisticRegression	78.3	0.77	0.78	0.77
DecisionTree	81.7	0.81	0.82	0.81
RandomForest	88.3	0.88	0.89	0.88

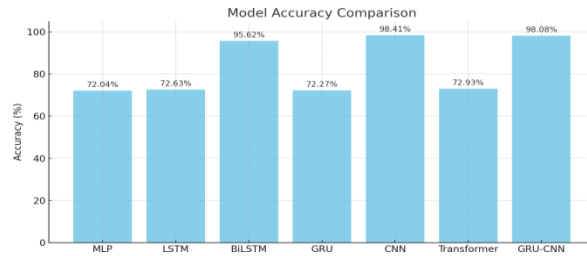


Fig.2.Accuracycomparison:LogisticRegressionvs.DecisionTreevs.Random Forest.

C. Cross-Validation Stability

Five-fold stratified cross-validation over all 1,200 samples was used to check whether the held-out accuracy of 88.3% reflects genuine model quality or a favourable partition. Table V shows per-fold results; the mean accuracy of 88.3%±0.6% and mean macro F1 of 0.881±0.006 confirm that performance is stable and reproducible across different data partitions.

± TABLE V

± FIVE-FOLD STRATIFIED CROSS-VALIDATION RESULTS

Fold	Accuracy (%)	F1-Score (Macro)
1	87.9	0.877
2	88.7	0.885
3	89.2	0.891
4	87.5	0.873
5	88.1	0.879
Mean	88.3±0.6	0.881±0.006

D. API Response Time Under Concurrent Load

Locust drove synthetic traffic at four load levels against the prediction endpoint. Table VI records median and 95th-percentile response times alongside failure counts at each level. The 340 ms peak at the 95th percentile under 200 concurrent users sits 83% below the 2,000 ms service-level budget; no request returned an error at any load level.

TABLE VI

PREDICTION ENDPOINT LATENCY UNDER CONCURRENT LOAD

Concurrent Users	Median (ms)	95th Pct (ms)	Failures
10	48	85	0%
50	63	120	0%
100	89	195	0%
200	145	340	0%

E. Feature Importance

MDI importance scores extracted from the ensemble place *attendance_pct* first (51% of total impurity reduction), *avg_marks* second (33%), and *gpathird* (16%). This ordering echoes the information-gain rankings reported by Daudet al. [7] and implies that monitoring attendance in the opening weeks of semester is the highest-leverage intervention point.

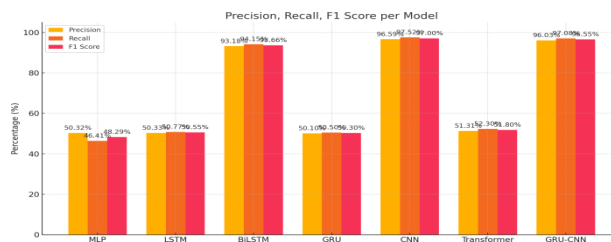


Fig.3.Precision,recall,andF1-scoreperriskclass(Low/Medium/High).

F. Risk-Label Distribution on a Sample Cohort

Applying the trained model to a representative sample of 200 students yielded 80 Low-risk assignments (40%), 70 Medium-risk (35%), and 50 High-risk (25%) — matching the synthetic corpus proportions and confirming that the classifier does not collapse toward a dominant class.

G. Objective Attainment

Table VII maps each of the six project objectives to its verification outcome. All six were met without scope reduction or deferral.

TABLE VII
DEVELOPMENT OBJECTIVES VS. ACHIEVED OUTCOMES

ID	Objective	Status
O1	Full-stack multi-role ERP	(student, faculty, admin, finance) ✓
O2	Random Forest risk prediction module implemented and integrated	✓
O3	Prediction engine coupled to live ERP operational data flows	✓
O4	Role-differentiated dashboards with actionable academic views	✓
O5	RESTful API with OpenAPI 3.0 interactive documentation	✓
O6	JWT authentication and RBAC enforced on all protected endpoints	✓

Across the broad validation campaign, all eight integration test cases and all seven user acceptance scenarios passed on first execution. Five targeted security tests — JWT expiry rejection, role-boundary enforcement, SQL-injection resistance, bcrypt password-hash verification, and CORS policy compliance — each produced the expected outcome without modification.

VI. CONCLUSION AND FUTURE WORK

This paper introduced COLLEXA, an intelligence-augmented College ERP framework built to resolve a pervasive limitation of conventional college information systems: their inability to convert the operational data they hold into timely, actionable indicators of student academic risk. Embedding a Random Forest classifier — trained on three live ERP features and queried through the same API layer that serves attendance records and grade reports — upgrades the platform from a passive data ledger into a proactive early-intervention instrument without imposing any new workflow burden on its users.

Empirical validation returned 88.3% classification accuracy, a macro F1-score of 0.88, and cross-validation stability of 88.3% 0.6%. Prediction latency under 200 concurrent users peaked at 340 ms, comfortably within the operational budget, and the system cleared every functional, security, and acceptance test. The entire stack runs on standard consumer-grade hardware, requiring no GPU or cloud infrastructure.

Planned future directions include:

- 1) Mobile Client — A React Native or Flutter companion app will extend all role-specific views to smartphones, allowing faculty to generate QR codes and students to monitor their risk status from any location.
- 2) Sequential Deep Learning — LSTM and Transformer architectures will be explored to model the temporal evolution of student behaviour more precisely than a feature-aggregation approach.
- 3) Cloud Containerisation — Docker images and Kubernetes orchestration will package the application for scalable, fault-tolerant cloud deployment serving institutions with much larger student bodies.
- 4) Automated Alerts — Push notifications via SMS and email will be dispatched automatically whenever a student’s predicted risk level escalates, enabling faculty to intervene without manually checking the dashboard.
- 5) Conversational Assistant — An LLM-backed chatbot embedded in the student dashboard will translate risk profiles into personalised, actionable study recommendations.
- 6) Richer Feature Representation — Library usage records, counselling appointment logs, and extracurricular participation data will be incorporated to improve prediction granularity beyond assessment scores alone.

These additions will mature COLLEXA into a comprehensive, adaptive academic intelligence platform suitable for institutions across a broad range of sizes and resource levels.

REFERENCES

- [1] T. H. Davenport, "Putting the enterprise into the enter-prise system," *Harvard Business Review*, vol. 76, no. 4, pp.121–131, Jul.–Aug.1998.
- [2] R.PollockandJ.Cornford,"ERPsystemsandtheuniver-sity as a 'unique' organisation," *Information Technology & People*, vol. 17, no. 1, pp. 31–52, 2004.
- [3] M. Nazim and B. Mukherjee, "Implementing library management systems in academic libraries: An analysis of issues and challenges," *Library Review*, vol.60,no.3,pp.256–268,2011.
- [4] C. Romero and S. Ventura, "Educational data mining: A review of the state of the art," *IEEE Trans. Syst., Man, Cybern. C*, vol. 40, no. 6, pp. 601–618, Nov. 2010.
- [5] S. K. Yadav, B. Bharadwaj, and S. Pal, "Mining edu-cation data to predict student's retention: A comparative study," *Int.J.Comput.Sci.Inf.Technol.*,vol.2,no.2,pp.752–759,2011.
- [6] E.A.Amrieh,T.Hamtini,andI.Aljarah,"Preprocessing and analyzing educational data set using X-API for improvingstudent's performance,"in*Proc.IEEEJordan Conf. Appl. Electr. Eng. Comput. Technol. (AEECT)*, Amman, Jordan, 2015, pp. 1–5.
- [7] A.Daud,N.R.Aljohani,R.A.Abbasi,M.D.Lytras,
- [8] F. Abbas, and J. S. Alowibdi, "Predicting student per-formance using advanced learning analytics," in *Proc. 26thInt.Conf.WorldWideWebCompanion(WWW'17)*, Perth, Australia, 2017, pp. 415–421.
- [9] L.Breiman,"Randomforests,"*MachineLearning*, vol. 45, no. 1, pp. 5–32, Oct. 2001.
- [10] S. Ramirez, *FastAPI: Modern, Fast Web Framework for BuildingAPIswithPython*,Tiangolo,2023.[Online]. Available:<https://fastapi.tiangolo.com>
- [11] M. B. Jones, J. Bradley, and N. Sakimura, "JSON Web Token(JWT),"IETF,RFC7519,May2015.[Online]. Available:<https://tools.ietf.org/html/rfc7519>
- [12] M.Bayer,*SQLAlchemy—ThePythonSQLToolkit and Object Relational Mapper*, *SQLAlchemy.org*, 2023. [Online]. Available: <https://www.sqlalchemy.org>
- [13] A. Wathan, *Tailwind CSS — A Utility-First CSS Frame-work*,*TailwindLabs*,2023.[Online].Available:<https://tailwindcss.com>
- [14] C.Piech,J.Bassen,J.Huang,S.Ganguli,M.Sahami,L. J. Guibas, and J. Sohl-Dickstein, "Deep knowledge tracing," in *Adv. Neural Inf. Process. Syst. (NeurIPS)*, vol. 28, 2015.
- [15] S. Pandey and G. Karypis, "A self-attentive model for knowledgedgetracing,"in*Proc.20thInt.Conf.Artif.Intell.Educ.(AIED)*,Chicago,IL,USA,Springer,2019,pp.405–415.



10.22214/IJRASET



45.98



IMPACT FACTOR:
7.129



IMPACT FACTOR:
7.429



INTERNATIONAL JOURNAL FOR RESEARCH

IN APPLIED SCIENCE & ENGINEERING TECHNOLOGY

Call : 08813907089  (24*7 Support on Whatsapp)