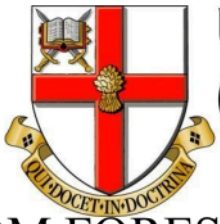
J119811-Report

by MD TANVIR SARDARR

Submission date: 16-Oct-2025 11:28AM (UTC+0100)
Submission ID: 265564203
File name: 134340_MD_TANVIR_SARDARR_J119811-Report_1231640_1559418538.pdf (8.91M)
Word count: 17452
Character count: 114328



University of Chester

A RANDOM FOREST-BASED SYSTEM FOR PREDICTING STUDENTS' DROPOUT RISK

Md Tanvir Sardarr
M.Sc. Data Science

16 October 2025

Supervisor:
Nkosi Mpofu

School of Computer and Engineering Sciences Faculty of Science, Business and Enterprise

Abstract

In the context of higher education, there is a significant need to focus on student retention, which imposes a necessity to find early-warning systems that are both efficient and clear enough to be implemented into the current advising processes. This dissertation seeks to fill this requirement by creating a decision-support system aimed at students dropout risk prediction and the facilitation of time-sensitive intervention. The central component of the system is a well-tuned Random Forest model, which is trained with a nested cross-validation scheme in order to guarantee high performance. Predictions are converted into actionable Low/Medium/High-risk bands. System is implemented as a web-based lightweight system (developed with Flask and SQLite) with administrator, staff, and student-specific dashboards with messaging services. To encourage transparency and trust, every risk prediction includes a SHAP explanation with each advisor and student being given clear and personalized explanations as to why the score was obtained. The model performs well with a held-out test set on a Monte Carlo simulated dataset and a macro AUROC of 0.934 and a balanced accuracy of 0.731. The model has probabilities that are well-calibrated (Brier score: 0.082; ECE: 0.048). Altogether, this dissertation adds a complete implementation of the early-warning service that manages to combine predictive precision with model control, explainability, and intervention procedures. Although the initial validation was conducted on synthetic data, future research will involve external validation using institutional data and continuous monitoring for long-term deployment.

Disclaimer

This dissertation is presented in compliance with MSc Data science degree at the university of Chester. In the absence of certain recognitions, it is regarded as the original work of the author, and has referenced the sources accordingly. This is a research prototype that is a predictor of a student likelihood of dropping out. Outputs are stochastic and can be inaccurate. They should not be relied on as the sole criterion for making decisions that involve individuals, nor should they replace professional judgment, institutional policy, or the law. High, Medium, and Low bands together with the probability levels behind them are specified in this dissertation just to be used in research and demonstration purposes. They demonstrate the trade-offs in calibration and decision in an experimental context. These bands need to be reconsidered and can be modified (including threshold values, bands, and actions related to them) after external validation in any real-world production deployment, and periodically monitored, with band recalibration.

Dedication

To my parents and family, whose unwavering love and sacrifices made this journey possible. To all of my teachers and supervisor (Nkosi Mpofu) for their guidance and belief. Above all, every student who has struggled in silence, may this work help bring timely support and a fairer chance to thrive.

Acknowledgments

This dissertation represents the culmination of my own independent work and research. I owe a huge debt of gratitude to my supervisor, Nkosi Mpofu, who has been with me through this rigorous journey with his guiding hand, corrective criticism, and encouragement. I also appreciate the invaluable online library services that provide essential resources. To achieve a clear and consistent language, I used Grammarly as a proofreading tool during the writing and editing process. Any other mistake is, of course, my responsibility.

Table of Contents

Table of figures

List of Tables

Chapter 1 - Introduction

1.1 Background and motivation

Student retention is a challenge that all universities face worldwide. Comparisons across countries, especially between Organization for Economic Co-operation and Development (OECD) systems, indicate the presence of large disparities in completion and early attrition rates (Aina et al., 2022). These are not coincidental differences but are quite significantly predetermined by the intricate system of socio-economic, demographic, and institutional aspects. In practice, this implies that there are large variance rates in countries and institutions with reference to first-year dropout rates and graduation rates within the conventional time ranges. This macro level of heterogeneity sets a minimum level of risk that differs in various national environments (Aina et al., 2022). Simultaneously, online learning spaces abandon the rich behavioral imprints of patterns of learning-management-system activities and engagement. Sudden fluctuations of these traces are usually predetermined by withdrawal. This allows building early-warning systems which integrate effective academic and socio-economic predictors and time-dependent indicators of engagement (Cheng et al., 2025; Marcolino et al., 2025). To enable fair and timely intervention in such systems, the predicted risks should be informed. It implies that they should correspond to the average of observed outcomes, and the level of confidence that a model assigns to each estimate should be reported (Silva Filho et al., 2023).

1.2 Problem statement

The issue of student retention has been a center of concern in the higher education sector and the rates of student dropout differ significantly across all nations. The completion rates in the UK, the Republic of Israel, Switzerland, and Ireland are over 80% in the 3rd year, and 1st-year dropout rates range between 6% and 20%. In contrast, the non-completion rates in other OECD members, including Brazil, Slovenia, and Italy, are more than 40% (Aina et al., 2022). These disparities represent the joint impact of socio-economic, demographic, and institutional variables on student persistence (Aina et al., 2022). The decision to remain or quit can be formulated as a cost-benefit analysis that will depend on student traits (e.g., age, gender, minority status), family background (e.g., financial means), and the level of academic and social inclusion. All these factors together change the expected returns and the actual cost of schooling. It is important to note that the level of cognitive ability not only raises opportunity costs and expected returns (Aina et al., 2022). Although the complete data regarding the students is available, the universities do not have practical tools that effectively complement predictive analytics with advising actionable workflows. It has been suggested that learning platforms can be used to obtain a large quantity of behavioral cues that can be used to predict withdrawal. However, there is a major difficulty in converting the data into effective interventions (Cheng et al., 2025). The main issue has been in the lack of connection between advanced risk identification tools and their implementation by support personnel. Current predictive models also tend to produce risk

scores that cannot be used to explain the context of constructive advisor-student discussions (Silva Filho et al., 2023). This lack of transparency negatively affects staff trust and impairs the applicability of predictive analytics in the real-world advising environment. Since the withdrawal rates are dependent on a complicated interplay of socio-economic, demographic, and institutional factors, the interventions will need a combination of academic affairs, financial aid, and mental health services, and each intervention will require presenting the risk information with specific lenses and proper access controls (Aina et al., 2022).

1.3 Aim

To design and implement an explainable and web-based system for predicting student dropout risk.

1.4 Objectives

- To develop an AI-based system using Random Forest for accurately predicting student dropout risk based on academic, behavioral, financial, and psychological features.
- To integrate explainability techniques (SHAP) that provide transparent insights into the factors influencing each student's dropout risk.
- To implement Monte Carlo-inspired scheduled and probabilistic confidence estimation to improve model reliability and decision support.
- To incorporate real-time dashboard that visualizes risk levels and supports admin-led interventions.

1.5 Research Questions

- How effective is the Random Forest algorithm in predicting student dropout using academic, financial, behavioral, and psychological features in higher education?
- How early in a course can dropout be predicted accurately enough to allow timely interventions without compromising prediction reliability?
- How can SHAP improve transparency and stakeholder trust in AI-driven dropout prediction systems?
- What is the added value of using Monte Carlo simulation to estimate confidence intervals in dropout predictions generated by machine learning models?

1.6 Scope

This project develops a web-based application using Random Forest to identify students who might be at risk of withdrawal in their early stages. This will also support staff management for 3 departments such as academic affairs, financial aid, and mental health services for any institution. The main risk assessment engine takes formatted student data, such as attendance data, academic achievement (GPA, assignment averages, quiz scores), activity data (engagement indicators), financial data, mental health indicators, and extracurricular activity data and executes a probabilistic risk analysis (High, Medium, Low) with associated

composite risk scores. Every prediction incorporates SHAP-based feature attributions that will facilitate clear and understandable advising discussions. The user-facing features include detailed prediction workflows, including a dashboard overview, risk assessment with progress tracking, and historical prediction management (eg, delete history). There is also a central messaging system that makes communication easy. Administrative interfaces provide cross-system analytics, including model performance indicators, risk distribution plots, feature importance, confusion matrices, and the ability to view users and check predictions.

1.7 Significance

The system transforms regularly gathered academic, engagement, financial, and well-being indicators into risk estimates that are calibrated to compare the predicted probabilities with actual outcomes. This, in turn, enhances the readiness of advisors to make informed decisions (Silva Filho et al., 2023; Richardson et al., 2024). It enhances precision by incorporating uncertainty information through Monte Carlo simulation, ensuring that limited support resources are prioritized when cases approach critical operating limits (Dega et al., 2023; Patel et al., 2021). SHAP-based explanations and reliability diagrams increase transparency by making the behavior of the model understandable to oversee and communicate with students. The implementation is lightweight and provides a Flask and SQLite application with dominating integrity rules and auditable logs that can enable universities to pilot, assess, and refine an evidence-based early warning facility as part of current processes and governance (Suraya and Sholeh, 2021).

1.8 Organization of the rest of the chapters

The remainder of the dissertation is structured as follows:

- **Chapter 2**: Reviews literature on ML model comparisons on dropout risk, Monte Carlo Simulation, SHAP, Flask, SQL Lite, Methodological Analysis, and research gaps.
- **Chapter 3:** Details the research design, data simulation and processing, technology stack, ML pipeline, procedures, Ethical and Privacy Considerations, and limitations.
- **Chapter 4**: Describes the system architecture, system workflow, database design, and user interface design
- **Chapter 5:** Provides a technical deep dive into the system's implementation
- **Chapter 6:** Discusses findings in relation to the objectives.
- **Chapter 7**: Concludes with key contributions and proposes future work.

Chapter 2: Literature Review

2.1 Introduction

The literature review provides the scholarly setting by summarizing the existing evidence on predicting student-dropout risk and the related decision-support practices. It explicates what has been known regarding predictors of machine-learning (ML) model performance, which uncovers unsolved challenges related to probability calibration, quantification of uncertainty, imbalance of classes, time disengagement, as well as explainability. . The review focuses on peer-reviewed work (primarily 2019–2025) and key foundational sources relevant to:

(i) Comparing ML Models for Dropout Prediction
(ii) Key factors for risk prediction
(iii) Monte Carlo approaches generating a synthetic dataset
(iv) Explainable AI techniques (SHAP)
(v) Flask in education-facing information systems
(vi) SQLite as an embedded relational store
(vii) Methodological Analysis (Hyperparameter tuning, Probability calibration, isotonic and sigmoid mapping, and Class imbalance handling).
(viii) Research Gaps
(ix) Conclusion

2.2 Comparing ML Models for Dropout Prediction

Random Forest consistently performs strongly in comparative evaluations. On a dataset of 4,424 students with 34 attributes that included grades, financial aid status, engagement metrics, and self-reported variables, Random Forest achieved 80.56% accuracy with specificity of 76.41% and sensitivity of 72.42%, outperforming XGBoost on the same data (Putra et al., 2025). In a larger cohort of 15,084 students, Random Forest achieved 85% recall and demonstrated stable performance across cross-validation folds (Da Conceição Silva et al., 2025). Studies using learning-management-system logs also report competitive results. CatBoost reached an average F1 of about 0.80 on Moodle activity traces, with strength on the minority class (Marcolino et al., 2025). A comparison of these, Random Forest, Support Vector Machines, Gradient Boosting, and LightGBM, found Random Forest delivered the highest F1 score (0.87), ahead of Gradient Boosting (0.85), LightGBM (0.83), Decision Trees (0.78), and SVM (0.76) (Villar and de Andrade, 2024). Together, these findings support the selection of Random Forest for dropout risk prediction due to its capacity to model nonlinear relationships, its robustness through assembling, and its provision of interpretable importance measures.

2.3. Risk factors for risk prediction

Based on empirically validated, literature-supported educational research, the continuous variables can be discretized into low-risk, medium-risk, and high-risk bands at empirically set thresholds at which a significant risk of student vulnerability is known to exist (Andersen et al., 2021). This binning process serves multiple purposes. It aids educational practitioners in triage, facilitates the interpretation of complex model results, and provides a common language for intervention. Besides, this methodology can be considered methodologically rigorous, since it can be used together with robust ensemble classifiers, such as Random Forest (Lindhardt et al., 2022; Cheng et al., 2025). The choice of thresholds is strictly based on literature in education. In the case of attendance, risk is as low (less than 80%), medium (70-79%), or high (less than 70%), based on the results that students in the 8th grade with attendance of 80% or less had a risk of dropping out of high school of 78%, which corresponds with the standard definition of absence, which includes missing 10% of the school year (Liza, 2020). Academic performance in terms of GPA falls under three risk groups namely low risk (below 3.0), medium risk (2.5-2.99) and high risk (below 2.5). This classification is determined by the fact that those students with a GPA of less than 2.0 are 5 times more likely to drop out (Liza, 2020). Mental well-being, represented by a mental health score of less than 7, is regarded as low-risk, medium-risk (5-6.9), or high-risk if less than 5 (Andersen et al., 2021). The inactivity of engagement in days is classified as a low-risk (less than 7 days), medium-risk (between 8 and 30 days), and high-risk if more than 30 days (Cheng et al., 2025). A binary financial instability indicator is considered a high-risk status when it is true (1), indicating the much higher probability of dropout of students with low-income backgrounds (Education at a Glance, 2023). Finally, the condition of extracurricular activity participation is a dichotomous indicator. Participation in extracurricular activity may enhance a student's sense of loyalty to the institution (NCES, n.d.).

2.4 Monte Carlo approaches to generate synthetic dataset

Monte Carlo simulation offers a critical approach to quantifying uncertainty in the prediction model of dropout with the calibration of probabilities by giving instance-level confidence. Meanwhile, calibration ensures that the predicted probabilities accurately represent the observed frequencies on average. It fails to render the accuracy of personal forecasts, especially around the critical decision levels (Kummaraka & Srisuradetchai, 2025). This uncertainty quantification is fundamental to effectively prioritizing interventions in the constantly shifting environment of student retention, where engagement can change quickly. Monte Carlo Dropout (MCDO) uses a series of stochastic forward makings of a neural network to approximate a predictive distribution during inference. It has been shown that MCDO has been proven to be more accurate in forecast and estimating uncertainty in time-series data and have been shown to be flexible to irregular patterns (Kummaraka et al., 2025). On the same note, tree ensembles such as the Random Forests can be combined with Monte Carlo techniques. As an example, bootstrap-based simulations will propagate input

uncertainty, producing predictive intervals capable of more accurately and reflecting outcome variability compared to single-point estimates (Dega et al., 2023). This method is directly applicable to the risk scoring, where the confidence of a probability close to an involvement threshold is as imperative as the probability itself. These studies can be synthesized in ways that highlight three major strengths of Monte Carlo techniques in predicting dropouts. They are simple to use with neural networks as well as tree ensembles (Kummaraka & Srisuradetchai, 2025; Dega et al., 2023).

2.5 SHAP Explainable Technique

The previous studies in the field of student withdrawal modeling create an understanding that feature attribution procedures, including SHAP (SHapley Additive exPlanations), play a crucial role in defining prevalent risk variables (Da Conceição Silva et al., 2025). Such interpretability is also evident in the extended responsibility of educational practice which further stresses that such responsibility underpins the development of trust, responsibility, and reasonable actions (Zhu et al., 2025). On a case-by-case basis, behavioral indicators provided by digital platforms are highly effective predictors (Marcolino et al., 2025). However, SHAP offers a conceptually sound, game-theoretic model to connect these indicators with the risk assessment of individuals in a testable manner. Local SHAP-based profiles can contextualize the increased risk around temporal discontinuities, enabling timely outreach (Cheng et al., 2025). To reduce misinterpretation, these descriptions are to be supplemented by probabilities calibrated and clear of confidence, such that a common form of communication on the level of risk magnitude and reliability is guaranteed across the various student subgroups (Nikolić et al., 2025).

2.6 Flask in education-facing information systems

Flask has been used to implement complete academic information systems with coherent data models and browser-based interfaces. A database-driven thesis management application demonstrates a Flask architecture with clearly defined tables for lecturers, students, thesis topics, and supervision histories, showing that Flask can structure CRUD workflows, forms, and reporting around academic processes (Suraya & Sholeh, 2021). In a separate deployment context, a volunteer-management system documents the use of Flask to provide user logins, role-specific views, and record maintenance suitable for small organisations (Anand et al., 2022). Together these examples indicate that Flask can deliver the web endpoints and templated pages needed for risk dashboards, intervention logs, and explanation views that integrate model outputs with student records. They also illustrate the practical value of route-based modularity and template rendering that can surface prediction details (for example, feature importances or SHAP plots) alongside case notes drawn from a relational store.

2.7 SQLite as an embedded relational store

SQLite appears in practitioner-oriented academic systems where simplicity of deployment and predictable relational semantics are required. The volunteer-management study reports a Flask–SQLite pairing that

supports user accounts and operational records without the overhead of a separate database service (Anand et al., 2022). For dropout-risk research, these precedents justify SQLite as a workable store for student profiles, engineered features, model outputs, and intervention histories when concurrency demands are modest, such as in departmental pilots or research prototypes that are primarily read-heavy for dashboards and write-light for periodic scoring.

2.8 Methodological Analysis

Machine learning models can affect their practical utility in high-stakes areas, such as causing poor performance due to default hyperparameters, incorrect probability estimates, or biases caused by unbalanced data. To counter these risks, a strict protocol of analysis is necessary. The sub-section provides background to such a protocol by synthesizing the research in hyperparameter tuning, probability calibration, and recalibration methods that are aimed at imbalanced data sets.

2.8.1 Hyperparameter Tuning

Recent work shows that model performance in tabular prediction depends strongly on structured hyperparameter search. In Random Forests, predictive accuracy and stability vary with the number of trees, maximum depth, and split criteria, so explicit tuning is required rather than adopting defaults (Contreras et al., 2021). Scaling choices also alter the decision boundaries for distance and margin-based learners. Therefore, it must be validated within the pipeline rather than outside (de Amorim et al., 2023). In education analytics, optimization studies on learning management system logs and institutional data report gains from systematic search strategies for tree ensembles and gradient boosting, reinforcing the importance of protocolized tuning with cross-validation (Marcolino et al., 2025; Vaarma & Li, 2024). For model selection, fully nested cross-validation can be unnecessarily expensive in many practical applications; repeated stratified cross-validation with a separate test split offers a defensible balance of bias and variance when resources are limited (Wainer and Cawley, 2021).

2.8.2 Probability calibration

Discrimination and probability fidelity are distinct properties and should be reported together in risk stratification research. A calibrated model outputs probabilities that match empirical frequencies, which can be examined with reliability curves, the Brier family of measures, and expected calibration error (ECE), alongside class-wise diagnostics (Silva Filho et al., 2023). Recent studies emphasize practical caveats in interpreting ECE, such as binning sensitivity and small-sample effects, and advocate complementary visual and quantitative summaries to avoid misleading conclusions (Pavlovic, 2025; Blog, 2025). Because the receiver operating characteristic can remain informative under class imbalance, it should be complemented with calibration reporting to avoid overestimating utility when scores are poorly aligned with risks (Richardson et al., 2024). In problems with multiple classes or grouped outputs, confidence adjustment becomes more challenging and may require specialized strategies, evaluated with care, to prevent inflation

of predicted confidence (LeCoz, Herbin, & Adjed, 2024). For tree ensembles, uncertainty in predictions can be propagated using Monte Carlo simulation so that probability summaries better reflect variability arising from data and model components (Dega, Dietrich, Schrön, & Paasche, 2023).

2.8.3 isotonic and sigmoid mapping

Isotonic regression is a nonparametric monotone fit that can capture complex misalignment patterns but may overfit when the calibration set is small. Sigmoid mapping imposes a parametric form that tends to be more stable under limited data but can underfit when misalignment is nonlinear (Silva Filho et al., 2023). Recent tutorials clarify how visual reliability analysis and sample-aware validation can reveal stepwise artifacts from isotonic fits or over-smoothing from sigmoid mappings (Pavlovic, 2025; Blog, 2025). In multiclass contexts, standard binary mappers must be extended or replaced with multiclass strategies, and emerging work highlights the risk of confidence inflation without careful evaluation design (LeCoz et al., 2024). Calibration also interacts with downstream equity. Probability alignment can mitigate systematic confidence distortions across subpopulations, reinforcing the need to assess fidelity within relevant groups (Nikolić et al., 2025).

2.8.4 Class imbalance and recalibration

Many student-outcome data sets have an imbalanced character, and literature characterizes its effect on dynamics of learning and probability faithfulness. Fold-aware oversampling is still significant, and flexible variants that attempt to form the minority-matching neighborhoods instead of enforcing uniform neighborhoods, without affecting the cross-validation integrity, are still present (Bunkhumpornpat et al., 2024). In the education field, in particular, synthesis sampling with dropout prediction, the combination of synthetic sampling and ensemble optimization yields significant improvements, which proves that sampling and tuning are inseparable decisions (Jain et al., 2025). Researchers also observe that probability scales can be distorted by any intervention that changes the distributions of scores, oversampling or weighting of classes by them, so calibration assessment of unperturbed data is frequently seen as a protection in studies that are of interest in probability fidelity (Silva Filho et al., 2023). Even though ROC analysis is not affected by changes in prevalence, it cannot answer the question of the observed risks being reflected by the predicted risks, and it is this difference that pushes the reporting of discrimination and calibration of the imbalanced regimes (Richardson et al., 2024).

2.9 Research gaps

Comparative studies continue to position ensemble methods as strong baselines for dropout prediction, with evidence from Indonesian and Brazilian cohorts illustrating the point (Putra et al., 2025; Villar & de Andrade, 2024). Yet most evaluations remain single-site and cross-sectional, which limits claims about transportability to new terms or institutions (Vaarma and Li, 2024). The difficulty is amplified when hyperparameters are tuned on static snapshots. Contreras et al. (2021) demonstrate that Random Forest

performance varies with depth, tree count, and split criteria, whereas LMS-log optimization rarely examines temporal generalization (Marcolino et al., 2025). Probability fidelity is underreported relative to ranking. The calibration survey by Silva Filho et al. (2023) describes mature tools' reliability curves, Brier decompositions, and ECE, yet practice still defaults to ROC summaries that remain stable under imbalance while obscuring whether scores reflect empirical risk (Richardson et al., 2024). Where mapping is attempted, choices between isotonic and sigmoid functions are rarely justified by sample size, even though small calibration sets can induce stepwise artifacts or over-smoothing, as recent tutorials explicitly state (Pavlovic, 2025; ICLR Blogposts, 2025). Imbalance handling is widespread, but its interaction with probability scales is inconsistently audited. Flexible oversampling methods, such as SMOTE, enhance minority sensitivity within folds (Bunkhumpornpat et al., 2024), and studies combining SMOTE with tuned ensembles, including student dropout, report significant gains (Jain et al., 2025). However, the calibration survey cautions that resampling and class weights shift score distributions, implying a need for recalibration on untouched data that many education papers omit (Silva Filho et al., 2023). The same is presented in decision policy. The theory of threshold under uncertainties cost-related uncertainty has an extended history (Ferri et al., 2019), and principled threshold adjustment is shown to be effective in applications to domains (Esposito et al., 2021). However, manually-selected cut-points tend to prevail despite the shown that data-driven thresholds can resemble the behavior of practitioners (Patel et al., 2021). The communication of uncertainty is still limited in tabular dropout models, although Monte Carlo propagation on ensembles can provide informative intervals in geospatial tasks (Dega et al., 2023) and nonstationary series can be better forecasted using stochastic inference (Kummaraka & Srisuradetchai, 2025). The trend can be echoed in dual-modal studies of sudden behavior change prior to withdrawal, where volatility is a common occurrence, and not an anomaly (Cheng et al., 2025).

2.10 Conclusion

This review of the literature confirms that Random Forest has been a strong and interpretable algorithm that is highly suitable in this field and has shown a stable competitive edge over existing ensembles (Putra et al., 2025; Villar & de Andrade, 2024). It emphasized that predictive accuracy is not enough but that probability calibration (Silva Filho et al., 2023) and quantification of uncertainty with Monte Carlo methods (Dega et al., 2023; Kummaraka & Srisuradetchai, 2025) should be adopted. Moreover, the chapter accentuated the value of explainable methods of AI, especially SHAP, to achieve model transparency and actionability (Padmasiri and Kasthuriarachchi, 2024). The analysis of Flask and SQLite (Suraya & Sholeh, 2021; Anand et al., 2022) proved them to be the right choice to prototype the end-to-end system that is planned to be realized in this study and incorporates predictive modeling and an available decision-support interface.

Chapter 3 Methodology

3.1 Research design

The research is designed as a design and development research (DDR) to develop a web-based decision support system for predicting the risk of student dropouts. The research method employs a quantitative, experimental approach to machine learning. A pipeline in a real-world Flask and SQLite system and follows the patterns of educational programs (Anand et al., 2022; Suraya & Sholeh, 2021). Random Forest classifier, which was chosen due to its good performance when applied in educational prediction tasks and its nature. interpretability(Contreras et al., 2021). The system is designed to produce calibrated probabilities instead of simple classifications that are translated to actionable Low, Medium, and High empirically derived risk bands. The development of models is completed in a rigorous nested cross-validation and hyperparameter search to measure robustness (Contreras et al., 2021).

3.2 Data Simulation and Processing

The Monte Carlo routine synthesizes the dataset, using the Python Faker library to create realistic student identities, and compounds this with parameterized statistical distributions on academic as well as behavioral characteristics (Kummaraka and Srisuradetchai, 2025). The simulation employs the random number generator of NumPy with a constant seed of 42 to guarantee the ability to replicate (Megalooikonomou et al., 2023). Faker creates fake email addresses and names that resemble real ones and builds student profiles that follow the patterns in the institution (Dega et al., 2023). Sampled academic and behavioral attributes like attendance follow a normal distribution with means of 84 and a σ=10, but with a maximum of 40 and a minimum of 100 to fit the realistic range of 40-100. An attendance below 90 is a critical threshold of academic success (Liza, 2020). The normal distribution used to produce GPA has the means of 3.0 and standard deviation of 0.5 within the range of 0.5 to 4.0. The average of the assignments and the quizzes are sampled to get normal distributions with mean equal to 70 and 72, and σ=12 and 15, respectively, but both limited to the scale 0-100%. The behavior is temporal, with dates of last activity, and the psychosocial risk factors are represented by mental health scores (Andersen et al., 2021). Engagement in extracurricular activities correlates with academic involvement (75% when GPA is 3.0 or higher). The weights of academic deficits (attendance: 30percent, GPA: 30percent), assessment gaps (20percent), activity latency (10percent), financial instability 7% and mental health 3% are scored with a composite risk score with Gaussian noise (σ=4) to make it realistic (Dega et al., 2023). The quantile ranges provide a 70/25/5% -Low/Medium/High risk (Education at a Glance, 2023). This feature engineering approach develops derived variables, which are then handled using a scikit-learn pipeline that includes median imputation and standardization, to produce a powerful preprocessing method in accordance with evidence on tabular classification performance (de Amorim et al., 2023). This is a pedagogically informed but synthetic dataset that can be reproduced and has

a multifactorial set of dropout determinants as has been found in recent educational studies (Da Conceicão Silva et al., 2025; Cheng et al., 2025).

3.3 Technology Stack

Table 3. 1 Technology Stack

Layer / Component	Purpose	Key Technologies & Libraries
Application Framework	Serves web pages, handles API requests, user sessions, and routing.	Flask, Flask-CORS
Data Persistence	Stores all application data, including user accounts, student records, and prediction history.	SQLite, SQLAlchemy (ORM)
Security & Authentication	Manages user login sessions, password hashing, and role-based access control.	Flask Sessions, Werkzeug Security
Data Simulation	Generates a synthetic dataset of student profiles with realistic academic and behavioral attributes for model development.	NumPy, pandas, Faker
Machine Learning Core	A leakage-proof pipeline for model training and prediction, handling imputation, scaling, and classification.	scikit-learn (RandomForestClassifier, Pipeline, SimpleImputer, StandardScaler)
Class Imbalance Handling	Addresses the uneven distribution of risk labels (Low/Medium/High) during model training.	imbalanced-learn (RandomOverSampler)
Model Calibration	Adjusts the model's raw probabilities to make them more accurate and interpretable.	scikit-learn (CalibratedClassifierCV)
Model Validation	Provides a robust, unbiased estimate of model performance and generalizability.	Nested Cross-Validation, Randomized SearchCV
Explainability (XAI)	Provides global and per-instance explanations for model predictions to ensure transparency.	SHAP (SHapley Additive exPlanations)
Model Management	Versions, stores, and loads the trained model and all associated performance metrics and metadata.	Pickle, JSON

3.4 Procedures

3.4.1 System Initialization and Model Training Pipeline

On the first run, the application will execute a complete system configuration where the SQLite database schema with foreign key constraints is created, and model artifact file paths are configured. The simulation of the data is a fixed random seed of 42 (Megalooikonomou et al., 2023) in order to be reproducible, whereby the student attributes are sampled according to statistically calibrated distributions. The machine learning pipeline has stringent validation measures that ensure data leakage. Median imputation and standardization are also included in feature preprocessing. RandomOverSampler integration is used to overcome the emergence of class imbalance. A class-weighted Random Forest classifier is optimized by randomized hyperparameter search used in stratified inner folds, which are in turn embedded in a large 5-fold outer validation loop (Kapoor & Narayanan, 2023; Wainer & Cawley, 2021). This framework does not ignore the fact that Random Forests are sensitive to parameter settings, or that scaling features should be done correctly (Contreras et al., 2021; de Amorim et al., 2023). The calibration of the models is done by either isotonic or sigmoid, which is chosen by minimization of Expected Calibration Error(ECE) and is accompanied by the macro Average Precision as a second consideration (Pavlovic, 2025). Persistence diagrams record calibration reliability to allow audited purposes (Silva Filho et al., 2023). The decision thresholds are determined based on the study of pooled validation probabilities and use F1-optimized cutoffs on the High and Low risk classes, and another High-risk cutoff aimed at achieving about 80% recall. Such locked thresholds and associated analysis tables are represented by model artifacts and executed when making a prediction, and are taken to guarantee robustness of the system in cases where model artifacts do not exist (Ferri et al., 2019; Esposito et al., 2021). The complete calibrated pipeline is saved to disk as dropout model.pkl, which contains nested cross-validation summaries, locked thresholds, test measures, feature importance analyses, SHAP explanations, and reliability visualizations, enabling reproducibility and audit needs (Bosch, 2020). The prediction endpoint provides probability distributions and threshold-based risk classifications. The quality of probabilities is constantly checked with the help of calibration curves and ECE metrics (Silva Filho et al., 2023). PredictionHistory table logs enable all prediction events and administrative interfaces offer an oversight with performance metrics, confusion matrices and risk distribution analytics and calibration visualizations.

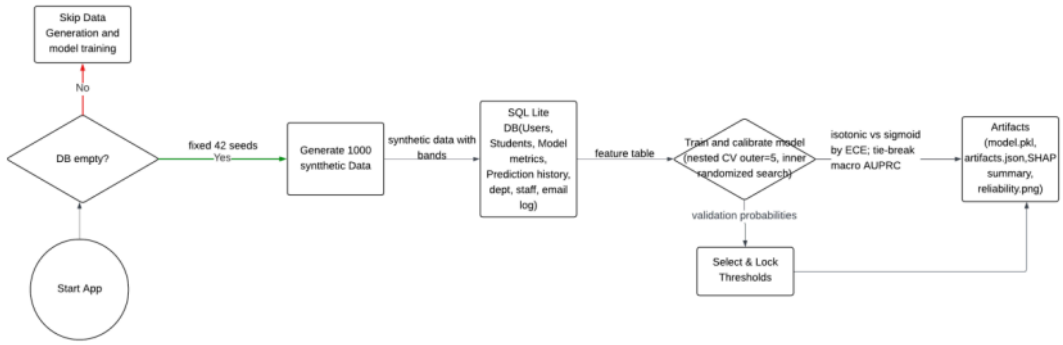


Figure 3. 1 Model Training Procedure

3.4.2 Messaging and Department Integration

The platform also includes a specific Messaging and department integration system to allow proactive interventions on the at-risk students, where the stakeholders can take the required actions (Bosch, 2020). It is designed based on role-based architecture with the staff of academic, financial, and mental health departments, as well as the admin being automatically redirected to dedicated dashboards. This communication system will enable employees to reach students who have been marked as high-risk by the predictive model in real-time. This is a direct communication channel that is considered vital in bridging a feedback connection between the data-driven understanding and human action, which is a foundational principle to responsible AI in education (Fu and Weng, 2024). Moreover, the system also has its internal communication tools that facilitate the cross-departmental coordination of the staff. That can be used to support a student with a complex set of issues in a holistic approach (Fu & Weng, 2024).

3.5 Analysis Plan

Operating points are derived from calibrated probabilities on the pooled validation data from the outer cross-validation loop. For each outer split, the inner search tunes a Random Forest pipeline. The probability calibration method is then chosen on the corresponding validation fold by minimizing ECE, with macro-AUPRC as the tiebreaker (isotonic vs sigmoid). The calibrated probabilities from all outer folds are concatenated to form a single validation set for threshold selection. For each candidate threshold, it computes precision, recall, and F1 on the positive class and builds per-class threshold tables. From these tables, it extracts four candidates:

(i) the High band threshold that maximizes F1,

(ii) the High band threshold whose recall is closest to 0.80,

(iii) the low-band threshold that maximizes F1, and

(iv) a precision-tilted Low alternative that maximizes $F_{0.5}$.

For deployment, the system locks two operating points:

- t_{high} = the High threshold targeting recall ≈ 0.80
- t_{low} = the Low threshold at F1-max

The following formal definitions were used for the reported metrics for model analysis. Let *TP*, *FP*, *TN*, and *FN* denote True Positives, False Positives, True Negatives, and False Negatives, respectively.

- **Precision:** Precision=$\frac{TP}{TP+FP}$
- **Recall (Sensitivity):** Recall=$\frac{TP}{TP+FN}$
- **Macro F1-Score:** $\frac{2\times Precision_{macro}\times Recall_{macro}}{Precision_{macro}\times Recall_{macro}}$; Where $Precision_{macro}$ and $Recall_{macro}$ are the arithmetic means of the per-class precision and recall values.
- **Macro AUROC:** The area under the Receiver Operating Characteristic curve, computed for each class in a one-vs-rest manner and then averaged.

$$AUPRC_{macro} = \frac{1}{K}\sum_{c=1}^{K} AUC_c$$

- **Macro AUPRC:** Average precision computed for each one versus rest view, then averaged across classes:

$$AUPRC_{macro} = \frac{1}{K}\sum_{c=1}^{k} AP_c$$

- **Brier Score (multiclass):** Mean of class wise Brier losses on binarized targets:

$$Brier = \frac{1}{N}\sum_{i=1}^{N}\sum_{k=1}^{K}(p_ik - y_ik)^2$$

- **Expected Calibration Error (ECE):** For each class, partition the predicted probabilities into M=10 equal-width bins. Within b, let conf(b) be the mean predicted probability and acc(b) the mean observed outcome. Per-class ECE is

$$\sum_{b=1}^{M}\frac{n_b}{N}|acc(b) - conf(b)|$$

3.6 Ethical and Privacy Considerations

The ethical basis of the project is informed by its intent as a decision-support tool to launch timely student outreach as opposed to automatic disciplinary or eligibility decision making. This calls on the need to be proportional, transparent, and have human-in-the-loop governance, which are consistent with the principles

of responsible AI that prioritize the welfare of students and explainability (Fu & Weng, 2024; Zhu et al., 2025). Ethics awareness must be pre-check by the IRB before using institutional data with minimal-risk attitude in form of data minimization and aggregation. A notice aware of information should state the purpose, information deployed, risk, and the opportunity of opting out. Since the predictors such as attendance are sensitive, advisor training is important to avoid the stigmatization of the results by describing the outputs as risk indicators rather than labels (Aina et al., 2022). Technically, the prototype has a Flask/SQLite architecture under hashed passwords and role-based access control which offers auditable footprint. Data protection requires privacy conscious logging that captures activities devoid of sensitive information and balances retention with institutional policy, which promotes accountability (Bosch, 2020; Cândido et al., 2019).

3.7 Limitation of the Method

Internal cross-validation is an accurate but cautious estimate that cannot be used to predict new intakes or institutions. Even minor changes in data collection or engagement patterns can drastically undermine performance (Wainer and Cawley, 2021). Higher education studies indicate that there are site-specific effects, which should not be assumed to be portable without any outside testing (Vaarma & Li, 2024). We can only observe a piece of the dropout pathway with administrative features but may not capture mental-health strain or family context, which may be insufficient to spot risk in certain subgroups (Aina et al., 2022). This lack of adolescent mental-health profile data in routinely gathered information is emphasized by evidence that connects mental-health profile to later dropout (Andersen et al., 2021).

3.8 Conclusion

The approach creates a scalable early-warning system due to the combination of a Flask web application and a powerful machine learning pipeline (Anand et al., 2022). The system is a Monte Carlo simulation that creates a realistic training of 1,000 student profiles. The algorithm is a class-aware random forest classifier that is trained in a leakage-resistant nested cross-validation setup, with full preprocessing and hyperparameter optimization (Kapoor & Narayanan, 2023; Wainer & Cawley, 2021). Reliability of probabilities is secured through model calibration, whereas locked decision thresholds represent operational trade-offs (Ferri et al., 2019; Esposito et al., 2021). Discrimination, calibration, and decision-focused metrics are strictly used to measure performance. All artifacts, thresholds and prediction events are also versioned to facilitate auditability and model governance (Bosch, 2020; Cândido et al., 2019). The framework recognizes limited generalizability, and the continued monitoring of performance is required to make models recalibrate the predictive accuracy (Wainer & Cawley, 2021).

Chapter 4: System Design

4.1 System Architecture

The application follows client-server-centric web architecture. A browser-based UI serves static HTML, CSS, and JS directly from the Flask app's project root and exposes authenticated endpoints for training, prediction, analytics, and administration. Model artifacts persisted locally (in the form of a pickled pipeline, JSON metadata, and a calibration plot), while operational data resides in a SQLite database managed via SQL Alchemy.

4.1.1 System components:

(i) Presentation layer (browser UI): static pages rendered or served at routes such as index.html, about.html, auth.html, admin-dashboard.html, admin-departments.html, dpet-academic.html, dept-finance.html, dept-mental-health.html, user-profile.html.

(ii) Application layer (Flask): HTTP routing, authentication via session and checks (_current_user_or_401, _require_admin), request validation, and JSON responses.

(iii) Modeling layer (scikit-learn pipeline): a leakage-safe pipeline with SimpleImputer, StandardScaler, RandomOverSampler, and RandomForestClassifier with class_weight='balanced', tuned in an inner randomized search and calibrated (isotonic or sigmoid) on held-out folds. Artifacts include thresholds for risk banding, nested CV summaries, feature importance, and SHAP summaries when available.

(iv) Persistence layer (SQLite and files): SQLAlchemy models persist users, students, metrics, and prediction history; files persist dropout_model.pkl, model_artifacts.json, and calibration_reliability.png.

(v) Training data readiness: if no data are present, the app seeds students (1000 synthetic rows) with realistic variability and performs initial training, ensuring a usable baseline.

4.2 System Workflows: Authorization, Training, Prediction, and Management

The Flux App is a web application designed to manage user interactions, administrative functions, and a core machine learning pipeline for predicting student dropout risk. The system integrates several key components, including a Flask Application core, a Database (SQLite), a File System for storing model artifacts, and distinct UI interfaces for User, Staff, and Admin roles, all managed within a Session framework. Upon application startup, the system initializes the database environment. This bootstrapping process guarantees a consistent and ready state for the application (Anand et al., 2022).

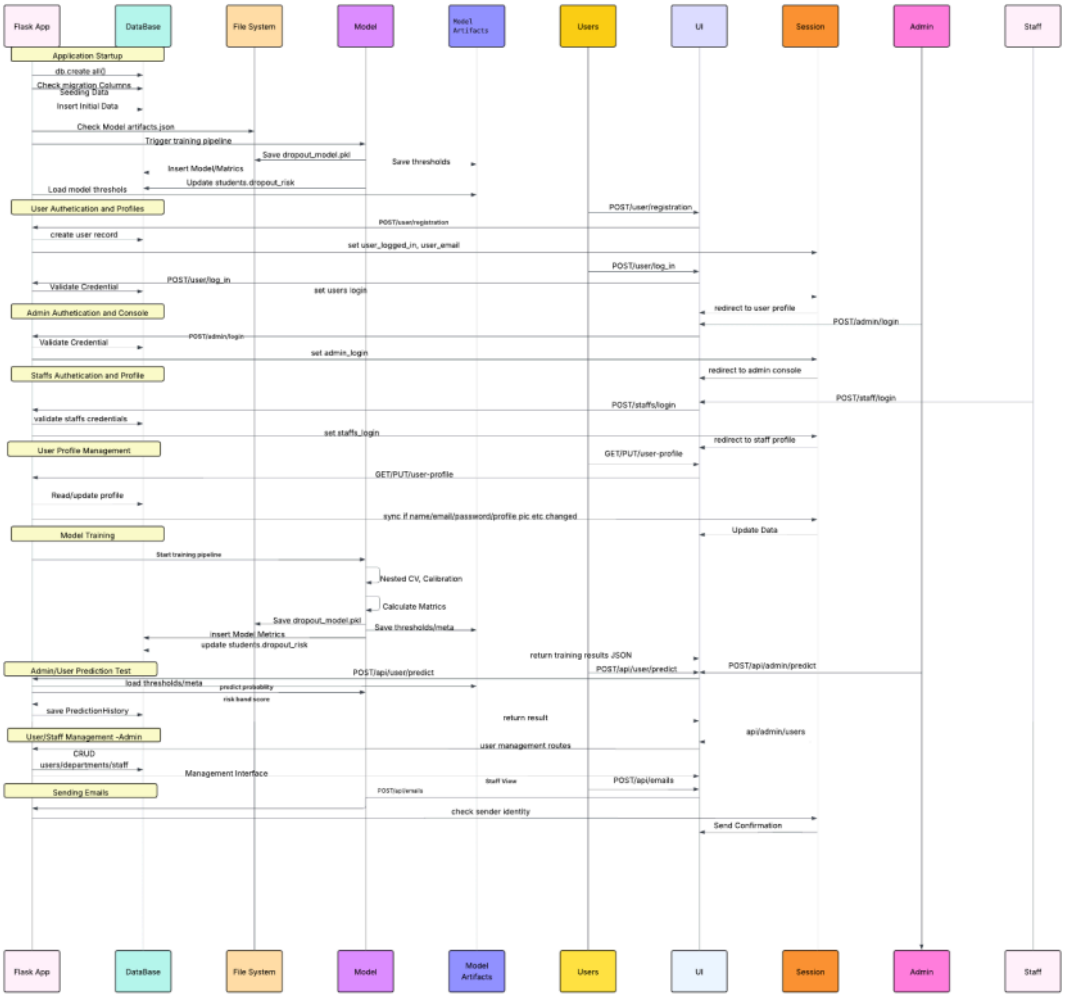


Figure 4. 1 System work flow UML sequence diagram

The system supports three roles via sessions. Users register at POST /user/register and log in at POST /user/login or POST /login, which sets user_logged_in. Staff authenticate at POST /staff/login, which sets staff_logged_in. Admins authenticate at POST /admin/login or POST /admin-login, which sets admin_logged_in and unlocks the console. These flows follow common Flask session patterns for small data systems, aiding simplicity and traceability, while keeping routes explicit for auditing purposes (Anand et al., 2022). At startup, train_dropout_model builds features from Student records, applies stratified folds and probability calibration, then saves dropout_model.pkl and model_artifacts.json and refreshes each student's dropout_risk. Calibration and the careful choice of metrics improve the quality and robustness of probability estimates under class imbalance, supporting fairer decisions (Silva Filho et al., 2023; Richardson et al., 2024). POST /api/user/predict loads the latest model and locked thresholds from ARTIFACTS_PATH,

computes class probabilities, assigns a risk band with _apply_locked_bands, and stores an auditable PredictionHistory entry for every call, reinforcing responsible machine learning practices and operational logging (Bosch and Bosch, 2020; Foalem et al., 2025). Users manage profiles at GET or POST /user-profile. Administrators manage users, departments, and staff through /api/admin/me, /api/admin/users, /api/admin/departments, and /api/admin/staff. Internal communication is provided by /api/emails for sending and listing messages (Anand et al., 2022).

4.3 System requirements

4.3.1 Functional requirements

The platform provides authentication and role-based access for users, staff, and administrators, with guarded administrative workflows and session management (Anand et al., 2022; Suraya & Sholeh, 2021). Administrative services expose JSON endpoints for creating, reading, updating, and deleting users and departments. Student-facing APIs collect academic, engagement, financial, and well-being inputs for dashboards. The model lifecycle supports on-demand training, nested cross-validation, probability calibration, threshold selection, and artefact persistence, enabling auditable risk estimation (Ferri et al., 2019; Patel et al., 2021). Prediction services return class probabilities, risk bands, and optional explainability values, along with a history log for review. Analytics endpoints deliver counts, metrics, confusion matrices, and feature importance for oversight, with charts rendered in a lightweight library and sortable tables for exploration (Chart.js Contributors, 2024; DataTables, n.d.).

4.3.2 Non-functional requirements

Security relies on password hashing, session checks, strict administrative gating, and safe file handling. Reliability and auditability are achieved through persisted metrics, thresholds, and calibration diagrams that support traceable decision making (Silva Filho et al., 2023). Performance is maintained by separating training from inference and returning compact JSON for a single page interface. Maintainability and interoperability follow a Flask and SQLite architecture with REST style services. Operational visibility is supported through structured logging and monitoring, consistent with recommendations for observability and responsible machine learning practice (Bosch, 2020; Foalem et al., 2025).

4.4 Database design

4.4.1 Schema overview

Eight tables are used: admin, user, student, model_metrics, prediction_history, department, staff, and email_log. Key columns and data types are defined directly in the models.

Entities and attributes:

Table 4. 1 Entities and attributes

Entity	Columns / Details
Admin	id (PK), email unique, password_hash, name, department, profile_photo.
User	id (PK), name, email unique, password_hash, department, profile_photo.
Student	id (PK), name, email unique, attendance float, gpa float, assignment_avg float, quiz_avg float, dropout_risk, last_activity datetime, financial_status, mental_health_score int, extracurricular_activity boolean.
ModelMetrics	id (PK), created_at (UTC default), accuracy, precision, recall, test_accuracy, test_precision, test_recall, plus JSON-text summaries confusion_matrix, risk_distribution, feature_importance, and a features_used list.
PredictionHistory	id (PK), user_id (FK, ON DELETE CASCADE), created_at (UTC default), prediction (band label), risk_score float, features (JSON text), probabilities (JSON text). It also has an ORM backref User.predictions with cascade='all, delete-orphan' and passive_deletes=True.
Department	id (PK), slug unique and indexed, name, created_at.
Staff	id (PK), name, email unique, password_hash, department_id (FK, ON DELETE CASCADE), profile_photo. Relationship backref on Department.staff uses cascade='all, delete-orphan' and passive_deletes=True.
EmailLog	id (PK), from_role, from_email, to_email, subject, body, created_at, and soft-delete flags read_by_recipient, deleted_by_sender, deleted_by_recipient. No foreign keys are declared by design.

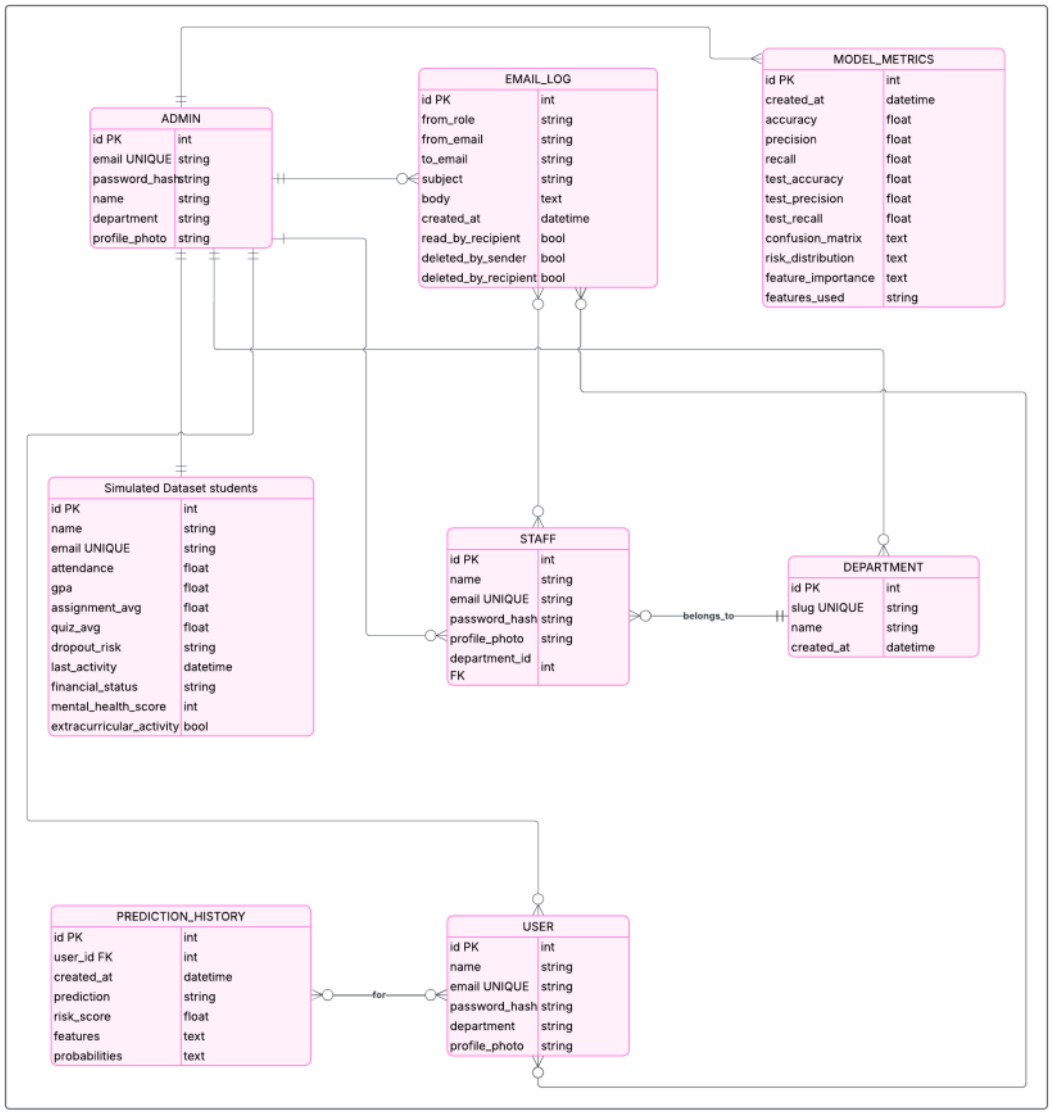


Figure 4. 2 System ERD Diagram

4.4.2 Database Integrity and Performance

Uniqueness is enforced on email fields in admin, user, and staff accounts to prevent duplicate accounts and simplify authentication and messaging lookups. The department slug is unique and efficient for routing and administrative search. These choices reflect routine relational design patterns for small web applications using Flask and SQLite (Anand et al., 2022; Suraya & Sholeh, 2021). PredictionHistory references User with delete cascade, so that deleting a user also removes associated prediction events. Staff references the Department with a delete cascade so that deleting a department removes its staff members. Enabling and

relying on database-level referential integrity avoids application-side drift and is consistent with recommended practice in Flask-backed SQLite projects (Suraya & Sholeh, 2021; Anand et al., 2022). Messaging integrity is implemented without foreign keys by design. EmailLog stores from_role, from_email, and to_email as strings. Each message can be hidden from a given party by setting deleted_by_sender or deleted_by_recipient. A message is permanently removed only when both parties have deleted it. This pattern provides durable auditability while allowing role-agnostic communication, which is valuable for the responsible operation and continuous auditing of machine learning-enabled applications (Bosch, 2020; Cândido et al., 2019; Foalem et al., 2025).

4.4.3 Database Security considerations

Account relations store password_hash rather than raw credentials. Hashing is performed in the application layer using standard library support in the Flask ecosystem during creation and update, which is consistent with secure handling described in common Flask patterns for SQLite backed systems (Anand et al., 2022; Suraya & Sholeh, 2021). Emails and profile photos are stored as strings and file paths rather than binary objects, which simplifies storage and reduces database size for routine operations.

4.4.4 Database cycle

The scheme separates identities and roles from messaging so that admin, staff, and user accounts can be managed independently of the communication log. EmailLog remains durable across account lifecycle events to support monitoring and compliance reviews as recommended in the logging literature for machine learning systems and responsible AI operations (Bosch, 2020; Cândido et al., 2019; Foalem et al., 2025). Delete cascades on PredictionHistory and Staff to ensure that predictive audit trails and departmental rosters remain consistent with administrative actions. ModelMetrics persists development and holds out summaries, including structures required for calibration review and decision analysis, which aligns with contemporary emphasis on calibration and decision-relevant reporting in supervised learning for educational risk and similar tabular prediction contexts (Silva Filho et al., 2023; Richardson et al., 2024).

4.4 User Interface Design

The interface prioritizes clarity and consistency through a shared frame with a fixed header, left navigation, and generous content area, a pattern shown to reduce cognitive load in operational dashboards (Dolatabadi et al., 2024). Predictable interactions and clear feedback support sustained use over time in real world settings, which aligns with evidence on dashboard adoption and lifecycle sustainment (Rossi et al., 2025). Consent language, transparent data use, and user control elements reflect human-centred guidance for education technology (Fu & Weng, 2024) and broader responsible practice in learning contexts (Zhu et al., 2025). All consoles follow a two-pane structure. Persistent links keep orientation stable while content switches without a full reload for a single page application feel. Unread counters and active states support early warning workflows where timely outreach influences attendance and engagement, as observed in

education interventions (Wu & Weiland, 2024). Emphasis on risk and status cues is appropriate given the retention stakes documented across systems and countries (Aina et al., 2022; Education at a Glance, 2023). A compact token set governs color, spacing, and layout constants, ensuring branding remains coherent across landing pages and consoles. Cards present metrics and tools. Risk bands are represented as compact chips for low, medium, and high risk, with mapping from probability to action informed by research on learned and robust thresholds (Patel et al., 2021; Ferri et al., 2019). Tables in admin and historical views provide sort and export via a proven grid component (DataTables, n.d.). Charts render in responsive containers for trend and model diagnostics using a lightweight library suitable for interactive dashboards (Chart.js Contributors, 2024). Discrimination and operating point summaries utilize ROC and precision-recall views for imbalanced outcomes (Richardson et al., 2024). Reliability diagrams convey probability quality and connect to expected calibration error, facilitating reader intuition (Silva Filho et al., 2023; Pavlovic, 2025; ICLR Blogposts, 2025). Explainability summaries for dropout indicators can share the same canvas to support transparent action in education settings (Silva et al., 2025). Semantic markup, visible focus styles, and consistent button semantics improve keyboard flow and comprehension. Delivery utilizes Flask with SQLite to maintain a lightweight deployment for education teams (Anand et al., 2022; Suraya & Sholeh, 2021). Toasts and status banners leverage logging and observability guidance to enhance auditability in responsible machine learning operations (Bosch, 2020; Cândido et al., 2019; Foalem et al., 2025).

4.5 User Interface Components

The table below is a mapping of key pages and sections of the system to their functions, the template files, and the appropriate figures in the appendix.

Table 4. 2 User Interface Components

Page / Section	Core Function	Template	Ref Appendix- A
Home Page	a landing page of the system	index.html	4.3,4.4,4.5,4.6,
About	Describes the system and the team members	about.html	4.7,4.8
Authentication	Central login/registration portal for all	auth.html	4.9,4.10,4.11,4.12
User Profile Overview	View and edit basic user information	user-profile.html	4.23
User Profile - Prediction	Displays the risk prediction, probability, and SHAP.	user-profile.html	4.24
User Profile - Progress	Shows a trend chart of the user's progress.	user-profile.html	4.25
User Profile - History	Searchable, sortable table of past user history	user-profile.html	4.26
User Profile - Mailbox	Integrated email interface for managing messages.	user-profile.html	4.27,4.28,4.29
Admin Dashboard - Overview	View and edit basic admin information	admin-dashboard.html	4.13,

Admin Dashboard - Analytics	Displays key performance indicators (KPIs) and model performance charts.	admin-dashboard.html	4.14
Admin Dashboard- Governance	monitors model performance and ensures the reliability of predictions.	admin-dashboard.html	4.15
Admin Dashboard – Manage students	A comprehensive, filterable table of all users and their status.	admin-dashboard.html	4.16
Monte Carlo simulation	Shows all the synthetic students data used in training	admin-dashboard.html	

Admin Dashboard - Predict Risk	Presents a calculated risk score with a visual progress bar and probabilities	admin-dashboard.html	4.17
Admin Departments – Create	Form to simultaneously create a new department and its first staff account.	admin-departments.html	4.18
Admin view Departments , staff and mail management	Table displaying all departments with basic info and management options.	admin-departments.html	4.19,4.20,4.21,4.22, 4.23
Finance, Mental health and Academic dept	Provides staff profile, email service and students managements	dept-finance.html, dept-mental-health.html, dept-academic.html,	4.30 -4.39

4.6 Conclusion

The system design presents a comprehensive, web-based decision-support architecture for predicting student dropout. It implements a client-server model using Flask for the application layer (Anand et al., 2022), a browser-based presentation layer, and SQLite for persistence. The core is a rigorously engineered machine learning pipeline that employs a Random Forest classifier, trained with nested cross-validation and probability calibration to ensure reliable, trustworthy risk scores (Silva Filho et al., 2023). The system enforces role-based access control, with all predictions logged for auditability to support responsible AI

practice (Bosch, 2020). By integrating a leakage-proof model, a secure database schema, and an intuitive UI, the design fulfills both functional needs for risk triage and non-functional requirements for transparency and maintainability, creating a responsible and operational early-warning tool.

Chapter 5 Implementation

5.1 Backend Implementation

5.1.1 System configuration

The app is a Python/Flask microservice with SQLite persistence, NumPy/Pandas for data wrangling, scikit-learn for modeling, and Matplotlib (headless) for reliability plots. SQLAlchemy provides ORM, CORS enables browser clients, and warnings are muted for clean logs. Model artifacts (pickle and JSON) and calibration plots are path-configured at startup. Optional integrations, including imbalanced-learn (oversampling) and SHAP (explanations), are automatically detected and toggle features without breaking core flows (Richardson et al., 2024; Pavlovic, 2025).

```python
# dependencies
try:
    from imblearn.pipeline import Pipeline as ImbPipeline
    from imblearn.over_sampling import RandomOverSampler
    IMB_READY = True
except Exception:
    IMB_READY = False

try:
    import shap
    SHAP_READY = True
except Exception:
    SHAP_READY = False
```

```python
# App configuration
app = Flask(
    __name__,
    static_folder='.',
    template_folder='.'
)
app.secret_key = os.getenv("SECRET_KEY", "dev-secret")
CORS(app)
basedir = os.path.abspath(os.path.dirname(__file__))
UPLOADS_DIR = os.path.join(basedir, 'uploads')
os.makedirs(UPLOADS_DIR, exist_ok=True)
app.config['ADMIN_PICS'] = UPLOADS_DIR
app.config['SQLALCHEMY_DATABASE_URI'] = f"sqlite:///{os.path.join(basedir, 'app.db')}"
app.config['SQLALCHEMY_TRACK_MODIFICATIONS'] = False
app.config['MODEL_PATH'] = os.path.join(basedir, 'dropout_model.pkl')
app.config['ARTIFACTS_PATH'] = os.path.join(basedir, 'model_artifacts.json')
app.config['CALIB_PNG'] = os.path.join(basedir, 'calibration_reliability.png')
db = SQLAlchemy(app)
warnings.filterwarnings("ignore", category=UserWarning)
```

Figure 5. 1 System configuration

5.1.2 Database models

Authentication for Admin, Staff, and User principals is secured via credential hashing Aina et al., 2021. The central Student entity models key feature vectors, incorporating predictors empirically validated in

educational data mining literature, such as academic performance (GPA), engagement (attendance), and psychosocial factors (Aina et al., 2021). Model lifecycle management is facilitated through the ModelMetrics entity, which persists performance indicators (e.g., accuracy, precision, recall) and artifact metadata for model versioning. All inference requests are immutably logged in the PredictionHistory table, ensuring full auditability. The EmailLog entity provides a verifiable ledger of all system communications.

```
106  class ModelMetrics(db.Model):
108      accuracy = db.Column(db.Float)
109      precision = db.Column(db.Float)
110      recall = db.Column(db.Float)
111      features_used = db.Column(db.String(500))
112      created_at = db.Column(db.DateTime, default=lambda: datetime.now(timezone.utc))
113      test_accuracy = db.Column(db.Float)
114      test_precision = db.Column(db.Float)
115      test_recall = db.Column(db.Float)
116      confusion_matrix = db.Column(db.Text)
117      risk_distribution = db.Column(db.Text)
118      feature_importance = db.Column(db.Text)
119
120  class PredictionHistory(db.Model):
121      id = db.Column(db.Integer, primary_key=True)
122      user_id = db.Column(
123          db.Integer,
124          db.ForeignKey('user.id', ondelete='CASCADE'),
125          nullable=False
126      )
127      created_at = db.Column(db.DateTime, default=lambda: datetime.now(timezone.utc))
128      prediction = db.Column(db.String(16), nullable=False)
129      risk_score = db.Column(db.Float, nullable=False)
130      features = db.Column(db.Text, nullable=False)
131      probabilities = db.Column(db.Text, nullable=False)
132      user = db.relationship(
133          'User',
134          backref=db.backref(
135              'predictions',
136              lazy=True,
137              cascade='all, delete-orphan',
138              passive_deletes=True
```

```python
class Department(db.Model):
    id = db.Column(db.Integer, primary_key=True)
    slug = db.Column(db.String(64), unique=True, nullable=False, index=True)
    name = db.Column(db.String(120), nullable=True)
    created_at = db.Column(db.DateTime, default=lambda: datetime.now(timezone.utc))

class Staff(db.Model):
    id = db.Column(db.Integer, primary_key=True)
    name = db.Column(db.String(100), nullable=False)
    email = db.Column(db.String(120), unique=True, nullable=False)
    password_hash = db.Column(db.String(128), nullable=False)
    department_id = db.Column(db.Integer,
                              db.ForeignKey('department.id', ondelete='CASCADE'),
                              nullable=False)
    department = db.relationship(
        'Department',
        backref=db.backref('staff', lazy=True, cascade='all, delete-orphan', passive_deletes=True)
    )
    profile_photo = db.Column(db.String(200), nullable=True)

class EmailLog(db.Model):
    id = db.Column(db.Integer, primary_key=True)
    from_role = db.Column(db.String(16), nullable=False)
    from_email = db.Column(db.String(120), nullable=False)
    to_email   = db.Column(db.String(120), nullable=False)
    subject    = db.Column(db.String(200))
    body       = db.Column(db.Text, nullable=False)
    created_at = db.Column(db.DateTime, default=lambda: datetime.now(timezone.utc))
    read_by_recipient  = db.Column(db.Boolean, default=False)
    deleted_by_sender  = db.Column(db.Boolean, default=False)
    deleted_by_recipient = db.Column(db.Boolean, default=False)
```

Figure 5.2.Database Implementation

5.1.3 Prediction Factors labelling and time handling

The get_factor_status function converts raw student metrics into human-readable risk bands (Low, Medium, High) for dashboard legends. Strong values in attendance, GPA, and mental health indicate lower risk, while recent activity is inverted, as lengthy inactivity signals disengagement. This categorical summarization enables rapid triage and clear staff communication (Rossi et al., 2025). Attendance cut-points reflect established monitoring tiers, such as the 90% rule for initial concern (Liza, 2020). Additionally, the ensure_utc function normalizes timestamps to UTC, ensuring consistent comparison of logs and events across environments and bolstering auditability for ML operations (Bosch & Bosch, 2020).

```python
def get_factor_status(name: str, value: float):
    if name == 'attendance':
        return 'Low' if value >= 80 else 'Medium' if value >= 70 else 'High'
    if name == 'gpa':
        return 'Low' if value >= 3.0 else 'Medium' if value >= 2.5 else 'High'
    if name == 'assignment_avg':
        return 'Low' if value >= 70 else 'Medium' if value >= 60 else 'High'
    if name == 'quiz_avg':
        return 'Low' if value >= 70 else 'Medium' if value >= 60 else 'High'
    if name == 'mental_health_score':
        return 'Low' if value >= 7 else 'Medium' if value >= 5 else 'High'
    if name == 'days_since_last_activity':
        return 'Low' if value <= 7 else 'Medium' if value <= 30 else 'High'
    if name == 'extracurricular_activity':
        return 'Low' if bool(value) else 'Medium'
    return 'Medium'
```

```python
def ensure_utc(dt):
    if dt is None:
        return datetime.min.replace(tzinfo=timezone.utc)
    if dt.tzinfo is None:
        return dt.replace(tzinfo=timezone.utc)
    return dt
```

Figure 5.3.Prediction Factors labelling and time handling

5.1.5 Monte Carlo Simulation, risk scoring, and calibration

The system builds privacy-friendly student data using the Faker library. It draws attendance, GPA, and assessment values from clipped normal distributions and writes records to the database. This mirrors common practice for prototyping educational prediction services where realistic cohorts are needed before institutional integration is approved (Vaarma and Li, 2024).

```python
# Synthetic data generation
fake = None
def _fake():
    global fake
    if fake is None:
        try:
            from faker import Faker
            fake = Faker()
        except Exception:
            class _Mini:
                def name(self): return f"Student {random.randint(1,99999)}"
                def email(self): return f"s{random.randint(1,99999)}@example.com"
            fake = _Mini()
    return fake
```

Figure 5.4.Monte Carlo Simulation

Mental health scores are sampled on a 10 scale. Financial stability is tied to GPA bands. Extracurricular participation has a higher probability of a stronger GPA. These choices encode literature-motivated correlates without exposing real records (Aina et al., 2022).

```python
def generate_mental_health():
    return int(np.clip(np.round(np.random.normal(6.8, 1.8)), 1, 10))

def financial_status_from_gpa(gpa):
    return 'Stable' if gpa >= 2.7 else ('Scholarship' if gpa >= 3.5 else 'Unstable')

def generate_extracurricular(gpa):
    return bool(np.random.rand() < (0.75 if gpa >= 3.0 else 0.45))
```

```python
def generate_random_students(n=1000, seed=42):
    rng = np.random.default_rng(seed)
    for _ in range(n):
        name = _fake().name()
        email = _fake().email()
        attendance = float(np.clip(rng.normal(84, 10), 40, 100))
        gpa = float(np.clip(rng.normal(3.0, 0.5), 0.5, 4.0))
        assignment_avg = float(np.clip(rng.normal(70, 12), 0, 100))
        quiz_avg = float(np.clip(rng.normal(72, 15), 0, 100))
        last_activity = datetime.now() - timedelta(days=int(np.clip(abs(rng.normal(8, 10)), 0, 90)))
        fin = financial_status_from_gpa(gpa)
        mh = generate_mental_health()
        ec = generate_extracurricular(gpa)
        try:
            db.session.add(Student(
                name=name, email=email,
                attendance=attendance, gpa=gpa,
                assignment_avg=assignment_avg, quiz_avg=quiz_avg,
                last_activity=last_activity, financial_status=fin,
                mental_health_score=mh, extracurricular_activity=ec
            ))
            db.session.commit()
        except Exception:
            db.session.rollback()
```

```python
def _compute_risk_scores(df: pd.DataFrame):
    days = df['days_since_last_activity'].values
    scores = (
        0.30*(100 - df['attendance'].values) +
        0.30*(4.0 - df['gpa'].values)*25 +
        0.10*(100 - df['assignment_avg'].values) +
        0.10*(100 - df['quiz_avg'].values) +
        0.10*np.clip((days - 7)/7, 0, 1) * 100 +
        0.07*df['financial_unstable'].values*100 +
        0.03*(10 - df['mental_health_score'].values)*10
    )
    return scores + np.random.default_rng(42).normal(0, 4, size=len(df))
```

Figure 5.5.Monte Carlo Simulation and risk scoring

Training uses a consistent feature frame and constructs a continuous risk score that increases with disengagement and weaker attainment. The score is converted to a three-level label using quantiles to maintain realistic prevalence, a practical approach for early warning contexts where withdrawals are rare but preceded by measurable changes (Wu and Weiland, 2024).

```python
def _label_from_scores(scores: np.ndarray, target_dist=(0.70, 0.25, 0.05)):
    low_p, med_p, high_p = target_dist
    q_high = 1.0 - high_p
    q_med = 1.0 - (high_p + med_p)
    th_high = np.quantile(scores, q_high)
    th_med = np.quantile(scores, q_med)
    return np.where(scores >= th_high, 'High', np.where(scores >= th_med, 'Medium', 'Low'))
```

Figure 5.6.Monte Carlo Simulation, risk scoring, and calibration

Calibration and evaluation utilities report multiclass ECE and macro areas under precision recall and under receiver operating characteristics. Reliability diagrams visualize how predicted probabilities align with observed frequencies for each class. These diagnostics are recommended when models support triage decisions and when class balance varies across cohorts (Silva Filho et al., 2023). Macro precision recall area is highlighted because it is sensitive to positive class prevalence and is informative for screening pipelines in education data (Richardson et al., 2024).

```python
def _ece_multiclass(y_true, proba, n_bins=10):
    y_true = np.asarray(y_true)
    cls = np.unique(y_true)
    y_bin = _binarize(y_true, classes=cls)
    eces = []
    for j in range(len(cls)):
        p = proba[:, j]
        t = y_bin[:, j]
        bins = np.linspace(0, 1, n_bins+1)
        ece = 0.0
        for b in range(n_bins):
            lo, hi = bins[b], bins[b+1]
            m = (p >= lo) & (p < hi) if b < n_bins-1 else (p >= lo) & (p <= hi)
            if not np.any(m):
                continue
            conf = p[m].mean()
            acc = t[m].mean()
            ece += np.abs(acc - conf) * (m.mean())
        eces.append(ece)
    return float(np.mean(eces)) if eces else 0.0
```

```python
def _auprc_macro(y_true, proba, classes):
    Y = label_binarize(y_true, classes=classes)
    return float(average_precision_score(Y, proba, average='macro'))

def _auroc_macro(y_true, proba, classes):
    Y = label_binarize(y_true, classes=classes)
    try:
        return float(roc_auc_score(Y, proba, average='macro', multi_class='ovr'))
    except Exception:
        return float('nan')
```

```python
def _reliability_plot(y_true, proba, path_png):
    try:
        y_true = np.asarray(y_true)
        classes = np.unique(y_true)
        Y = label_binarize(y_true, classes=classes)
        bins = np.linspace(0,1,11)
        plt.figure(figsize=(6,5))
        for j in range(len(classes)):
            p = proba[:, j]; t = Y[:, j]
            xs, ys = [], []
            for b in range(10):
                lo, hi = bins[b], bins[b+1]
                m = (p >= lo) & (p < hi) if b < 9 else (p >= lo) & (p <= hi)
                if np.any(m):
                    xs.append(p[m].mean()); ys.append(t[m].mean())
            plt.plot(xs, ys, marker='o', label=f'class={classes[j]}')
        plt.plot([0,1],[0,1],'--', linewidth=1)
        plt.xlabel('Predicted probability'); plt.ylabel('Observed frequency')
        plt.title('Reliability (calibration) diagram')
        plt.legend(); plt.tight_layout()
        plt.savefig(path_png, dpi=160); plt.close()
    except Exception:
        pass
```

Figure 5.7.Monte Carlo Simulation, risk scoring, and calibration

5.1.6 Thresholding and operating points

The threshold table scans a grid from 0.05 to 0.95 and records precision, recall, and F1 for the chosen positive label. From these curves, the system locks the F1 optimal thresholds for High and for Low. It also computes an F 0.5 option for Low to emphasize precision when false alarms are costly, and a High threshold that achieves about 80% recall for screening. This follows decision threshold selection under uncertain operating conditions and class imbalance (Ferri et al., 2019). Adjusting the operating point rather than the model parameters is a standard way to trade sensitivity and specificity in risk stratification (Esposito et al., 2021). The banding rule then maps probabilities to High, Low, or otherwise Medium using the locked values. For reporting, feature importances are read directly when available, supporting interpretable dashboards in imbalanced contexts (Richardson et al., 2024).

```python
def _threshold_table(y_true, proba, classes, positive_label, grid=None):
        grid = np.linspace(0.05, 0.95, 19)
    idx = list(classes).index(positive_label)
    p = proba[:, idx]
    y = (np.asarray(y_true) == positive_label).astype(int)
    rows = []
    for thr in grid:
        pred = (p >= thr).astype(int)
        prec = precision_score(y, pred, zero_division=0)
        rec  = recall_score(y, pred, zero_division=0)
        f1   = f1_score(y, pred, zero_division=0)
        rows.append([float(thr), float(prec), float(rec), float(f1)])
    return pd.DataFrame(rows, columns=['threshold','precision','recall','f1'])

def _pick_thresholds_from_val(y_val, proba_val, classes):
    tbl_high = _threshold_table(y_val, proba_val, classes, 'High')
    tbl_low  = _threshold_table(y_val, proba_val, classes, 'Low')

    th_high = float(tbl_high.loc[tbl_high['f1'].idxmax(), 'threshold'])
    th_low  = float(tbl_low.loc[tbl_low['f1'].idxmax(),  'threshold'])

    beta = 0.5
    f05 = (1 + beta**2) * (tbl_low['precision'] * tbl_low['recall']) / (beta**2 * tbl_low['precision'] + tbl_low['recall'] + 1e-12
    th_low_f05 = float(tbl_low.loc[f05.idxmax(), 'threshold']) if len(tbl_low) > 0 else th_low

    th_high_recall80 = float(tbl_high.loc[(tbl_high['recall'] - 0.80).abs().idxmin(), 'threshold'])

    return {
        "high_f1": th_high, "low_f1": th_low,
        "alt": {"high_target_recall_0.80": th_high_recall80, "low_f0.5": th_low_f05},
        "tables": {"high": tbl_high.to_dict(orient='records'),
                   "low":  tbl_low.to_dict(orient='records')}
```

```python
def _apply_locked_bands(proba_row: Dict[str,float], thr_high: float, thr_low: float):
    if proba_row.get('High',0.0) >= thr_high:
        return 'High'
    if proba_row.get('Low',0.0) >= thr_low:
        return 'Low'
    return 'Medium'

def _extract_feature_importances_from_classifier(clf):
    imp = getattr(clf, 'feature_importances_', None)
    if imp is None:
        return None
    return np.asarray(imp, dtype=float)
```

Figure 5.8.Thresholding and operating points

5.1.7 Leakage Control and Probability Calibration

The training stack imputes missing values with medians, standardized features, then fits a class-weighted Random Forest. When imbalanced learn is available, oversampling is placed inside the pipeline, so resampling occurs within each fold, limiting information leakage during cross-validation as recommended by Wainer and Cawley 2021. Hyperparameters are selected with randomized search over tree count, depth, and split criteria. After choosing the best estimator, probabilities are calibrated by testing isotonic and sigmoid functions on held-out folds, then selecting the option with the lower expected calibration error and stronger macro area under the precision-recall curve, following the guidance in Silva Filho et al. (2023). This yields reliable risk scores for threshold-based screening.

```python
# Core training with leakage-free nested cross-validation
def _build_base_pipeline():
    steps = [
        ('imputer', SimpleImputer(strategy='median')),
        ('scaler', StandardScaler()),
    ]
    if IMB_READY:
        steps.append(('sampler', RandomOverSampler(random_state=42)))
        pipe_cls = ImbPipeline
    else:
        pipe_cls = Pipeline
    steps.append(('classifier', RandomForestClassifier(random_state=42, class_weight='balanced')))
    return pipe_cls(steps)

def _param_space():
    return {
        'classifier__n_estimators': [300, 500, 800, 1200],
        'classifier__max_depth': [8, 12, 16, None],
        'classifier__min_samples_split': [2, 5, 10],
        'classifier__min_samples_leaf': [1, 2, 4],
        'classifier__max_features': ['sqrt', 'log2', None],
    }
```

```python
def _tune_inner(X, y, n_splits=3, random_state=42):
    cv = StratifiedKFold(n_splits=n_splits, shuffle=True, random_state=random_state)
    search = RandomizedSearchCV(
        _build_base_pipeline(),
        param_distributions=_param_space(),
        n_iter=25, scoring='f1_weighted', cv=cv, n_jobs=-1, refit=True, random_state=random_state, verbose=0
    )
    search.fit(X, y)
    return search.best_estimator_, search.best_params_

def _choose_calibrator(estimator, X_train, y_train, X_val, y_val):
    methods = ['isotonic', 'sigmoid']
    results = []
    for m in methods:
        pipe_cls = ImbPipeline if IMB_READY else Pipeline
        cal = pipe_cls(list(estimator.steps[:-1]) + [
            ('classifier', CalibratedClassifierCV(
                estimator=estimator.named_steps['classifier'], method=m, cv=3
            ))
        ])
        cal.fit(X_train, y_train)
        proba = cal.predict_proba(X_val)
        classes = cal.named_steps['classifier'].classes_
        ece = _ece_multiclass(y_val, proba, n_bins=10)
        auprc = _auprc_macro(y_val, proba, classes)
        results.append((m, ece, auprc, cal))
    results.sort(key=lambda r: (r[1], -r[2]))
    return results[0]
```

Figure 5.9.Leakage Control and Probability Calibration

5.1.8 Probability calibration and reliability assessment using isotonic and sigmoid with ECE and macro AUPRC

train_dropout_model materializes a Pandas DataFrame from Student rows, synthesizes supervisory labels by calling _compute_risk_scores then _label_from_scores, and partitions the data with train_test_split to keep a held-out test set. Model selection employs a 5-fold StratifiedKFold outer loop with an inner RandomizedSearchCV over a pipeline that may include RandomOverSampler, ensuring that tuning and any resampling occur strictly within the fold to minimize selection bias and leakage (Wainer and Cawley, 2021;

Kapoor and Narayanan, 2023). Calibration is chosen per outer fold by wrapping the tuned estimator in CalibratedClassifierCV with both isotonic and sigmoid options and selecting the variant with lower _ece_multiclass and, in ties, higher _auprc_macro. This aligns the system with best practice that emphasises calibrated probabilities and explicit expected calibration error tracking for multi-class classifiers (Silva Filho et al., 2023; Pavlovic, 2025). Evaluation records macro_f1, balanced_accuracy, accuracy, weighted precision and recall, auroc_macro, auprc_macro, brier, and ece, then aggregates the outer-fold means and standard deviations. The use of macro AUPRC complements AUROC when class supports differ, giving sensitivity to minority categories in imbalanced educational risk settings (Richardson et al., 2024). Feature influence is derived from feature_importances_ on the tuned Random Forest, with a fallback to permutation_importance; if available, a global SHAP summary is computed with shap.TreeExplainer to support transparent dashboards. Decision thresholds are data-driven. _threshold_table sweeps candidate cut-offs on validation probabilities, and _pick_thresholds_from_val locks two operating points: a high-risk cut intended to meet an eight-tenths recall target and a low-risk cut that maximises F1 for the Low class. _apply_locked_bands then maps each probability row to High, Medium, or Low, a practical approach when costs are uncertain and class imbalance is present (Ferri, Hernández-Orallo, and Flach, 2019). Finally, the calibrated pipeline is fit on X_trainval, evaluated on X_test, and persisted with pickle to MODEL_PATH. A JSON bundle at ARTIFACTS_PATH stores metrics, chosen calibration method, locked thresholds, feature importance, and SHAP summaries. The code propagates predictions to every Student by calling predict_proba and applying the locked thresholds, commits a ModelMetrics row for auditability, and powers the runtime /api/user/predict path, which rebuilds a one-row DataFrame, returns probabilities, a band, a scalar risk_score, and appends a PredictionHistory entry for longitudinal tracking (Suraya and Sholeh, 2021).

```
507  def train_dropout_model():
510      if len(students) < 60:
511          raise RuntimeError("Not enough students to train a model.")
512
513      data = {
514          'attendance': [s.attendance for s in students],
515          'gpa': [s.gpa for s in students],
516          'assignment_avg': [s.assignment_avg for s in students],
517          'quiz_avg': [s.quiz_avg for s in students],
518          'days_since_last_activity': [(datetime.now() - (s.last_activity or datetime.min)).days for s in st
519          'financial_stable': [1 if s.financial_status == 'Stable' else 0 for s in students],
520          'financial_unstable': [1 if s.financial_status == 'Unstable' else 0 for s in students],
521          'mental_health_score': [s.mental_health_score or 5 for s in students],
522          'extracurricular_activity': [1 if s.extracurricular_activity else 0 for s in students]
523      }
524      df = pd.DataFrame(data)
525
526      risk_scores = _compute_risk_scores(df)
527      labels = _label_from_scores(risk_scores, target_dist=(0.70, 0.25, 0.05))
528      df['dropout_risk'] = labels
529
530      X = df.drop(columns=['dropout_risk']).astype(float)
531      y = df['dropout_risk'].values
532      classes = np.unique(y)
533
534      X_trainval, X_test, y_trainval, y_test = train_test_split(
535          X, y, test_size=0.20, stratify=y, random_state=42
536      )
537
538      outer = StratifiedKFold(n_splits=5, shuffle=True, random_state=42)
539      outer_metrics = []
540      all_val_probs, all_val_true = [], []
541
542      for outer_train_idx, outer_val_idx in outer.split(X_trainval, y_trainval):
543          X_tr, X_val = X_trainval.iloc[outer_train_idx], X_trainval.iloc[outer_val_idx]
544          y_tr, y_val = y_trainval[outer_train_idx], y_trainval[outer_val_idx]
545
```

Figure 5.10.Probability calibration and reliability assessment using isotonic and sigmoid with ECE and macro AUPRC

```
    for outer_train_idx, outer_val_idx in outer.split(X_trainval, y_trainval):
        X_tr, X_val = X_trainval.iloc[outer_train_idx], X_trainval.iloc[outer_val_idx]
        y_tr, y_val = y_trainval[outer_train_idx], y_trainval[outer_val_idx]

        best_estimator, _ = _tune_inner(X_tr, y_tr, n_splits=3, random_state=42)

        method, ece_val, auprc_val, cal_pipeline = _choose_calibrator(best_estimator, X_tr, y_tr, X_val, y_val)

        y_pred = cal_pipeline.predict(X_val)
        proba = cal_pipeline.predict_proba(X_val)

        macro_f1 = f1_score(y_val, y_pred, average='macro', zero_division=0)
        bal_acc = balanced_accuracy_score(y_val, y_pred)
        acc = accuracy_score(y_val, y_pred)
        prec_w = precision_score(y_val, y_pred, average='weighted', zero_division=0)
        rec_w = recall_score(y_val, y_pred, average='weighted', zero_division=0)
        auroc = _auroc_macro(y_val, proba, classes)
        auprc = _auprc_macro(y_val, proba, classes)

        Ybin = label_binarize(y_val, classes=classes)
        briers = []
        for j in range(len(classes)):
            briers.append(brier_score_loss(Ybin[:, j], proba[:, j]))
        brier = float(np.mean(briers))
        ece = _ece_multiclass(y_val, proba)

        outer_metrics.append({
            "accuracy": acc, "precision_w": prec_w, "recall_w": rec_w,
            "macro_f1": macro_f1, "balanced_accuracy": bal_acc,
            "auroc_macro": auroc, "auprc_macro": auprc,
            "brier": brier, "ece": ece, "cal_method": method
        })
        all_val_probs.append(proba); all_val_true.append(y_val)
```

```
def train_dropout_model():
    def agg(key):
        arr = np.array([m[key] for m in outer_metrics if not (isinstance(m[key], float) and np.isnan(m[key]))])
        if arr.size:
            return (float(arr.mean()), float(arr.std()))
        else:
            return (float('nan'), float('nan'))

    nested_summary = {k: {"mean": agg(k)[0], "std": agg(k)[1]} for k in outer_metrics[0].keys() if k != "cal_method"}
    nested_summary["cal_method_majority"] = max(
        set([m["cal_method"] for m in outer_metrics]),
        key=[m["cal_method"] for m in outer_metrics].count
    )

    all_val_probs = np.vstack(all_val_probs)
    all_val_true = np.concatenate(all_val_true)
    thresholds = _pick_thresholds_from_val(all_val_true, all_val_probs, classes)

    X_tr, X_val, y_tr, y_val = train_test_split(X_trainval, y_trainval, test_size=0.20, stratify=y_trainval, random_state=7)
    final_estimator, final_params = _tune_inner(X_tr, y_tr, n_splits=3, random_state=123)
    method, ece_val, auprc_val, final_pipeline = _choose_calibrator(final_estimator, X_tr, y_tr, X_val, y_val)

    pipe_cls = ImbPipeline if IMB_READY else Pipeline
    final_pipeline = pipe_cls(list(final_estimator.steps[:-1]) + [
        ('classifier', CalibratedClassifierCV(
            estimator=final_estimator.named_steps['classifier'],
            method=method, cv=5
        ))
    ])
    final_pipeline.fit(X_trainval, y_trainval)
```

Figure 5.11.Probability calibration and reliability assessment using isotonic and sigmoid with ECE and macro AUPRC (continue)

```
605
606         y_pred_test = final_pipeline.predict(X_test)
607         proba_test = final_pipeline.predict_proba(X_test)
608
609         test_acc = accuracy_score(y_test, y_pred_test)
610         test_prec = precision_score(y_test, y_pred_test, average='weighted', zero_division=0)
611         test_rec  = recall_score(y_test, y_pred_test, average='weighted', zero_division=0)
612         test_macro_f1 = f1_score(y_test, y_pred_test, average='macro', zero_division=0)
613         test_bal_acc  = balanced_accuracy_score(y_test, y_pred_test)
614         test_auprc    = _auprc_macro(y_test, proba_test, classes)
615         test_auroc    = _auroc_macro(y_test, proba_test, classes)
616
617         Ybin_t = label_binarize(y_test, classes=classes)
618         briers = [brier_score_loss(Ybin_t[:, j], proba_test[:, j]) for j in range(len(classes))]
619         test_brier = float(np.mean(briers))
620         test_ece   = _ece_multiclass(y_test, proba_test)
621
622         _reliability_plot(y_test, proba_test, app.config['CALIB_PNG'])
623
624         tuned_rf = final_estimator.named_steps['classifier']
625         importances = _extract_feature_importances_from_classifier(tuned_rf)
626         feat_names = list(X.columns)
627         if importances is None:
628             try:
629                 r = permutation_importance(final_pipeline, X_test, y_test, n_repeats=10, random_state=0, n_jobs=-1)
630                 importances = r.importances_mean
631             except Exception:
632                 importances = np.zeros(len(feat_names))
633         feature_imp = dict(zip(feat_names, [float(x) for x in importances]))
634
635         shap_summary = {}
636         if SHAP_READY:
637             try:
```

```
635         shap_summary = {}
636         if SHAP_READY:
637             try:
638                 explainer = shap.TreeExplainer(tuned_rf)
639                 sv = explainer.shap_values(X_trainval)
640                 if isinstance(sv, list):
641                     mean_abs = np.mean(np.abs(np.array(sv)), axis=(0,1))
642                 else:
643                     mean_abs = np.mean(np.abs(sv), axis=0)
644                 shap_summary = dict(zip(feat_names, [float(x) for x in mean_abs]))
645             except Exception:
646                 pass
647
648         with open(app.config['MODEL_PATH'], 'wb') as f:
649             pickle.dump(final_pipeline, f)
650
651         preds_all = final_pipeline.predict_proba(X)
652         classes_list = list(final_pipeline.named_steps['classifier'].classes_)
653         idxH = classes_list.index('High'); idxL = classes_list.index('Low'); idxM = classes_list.index('Medium')
654         thrH = thresholds["alt"]["high_target_recall_0.80"]
655         thrL = thresholds["low_f1"]
656         for stu, prows in zip(students, preds_all):
657             prob_row = {'High': float(prows[idxH]), 'Medium': float(prows[idxM]), 'Low': float(prows[idxL])}
658             stu.dropout_risk = _apply_locked_bands(prob_row, thrH, thrL)
659         db.session.commit()
660
661         mm = ModelMetrics(
662             accuracy=test_acc, precision=test_prec, recall=test_rec,
663             features_used=", ".join(feat_names),
664             test_accuracy=test_acc, test_precision=test_prec, test_recall=test_rec,
665             confusion_matrix=json.dumps(confusion_matrix(y_test, y_pred_test).tolist()),
666             risk_distribution=json.dumps({
667                 "Low": int(np.sum(y_test=='Low')),
668                 "Medium": int(np.sum(y_test=='Medium')),
669                 "High": int(np.sum(y_test=='High'))
```

Figure 5.12.Probability calibration and reliability assessment using isotonic and sigmoid with ECE and macro AUPRC(continue)

```
    artifacts = {
        "timestamp": datetime.now(timezone.utc).isoformat(),
        "model": "RandomForestClassifier (calibrated)",
        "oversampling": "RandomOverSampler in CV" if IMB_READY else "Not available (install imblearn)",
        "calibration": {"chosen_method": method, "val_ece": ece_val, "val_auprc": auprc_val},
        "nested_cv": nested_summary,
        "locked_thresholds": thresholds,
        "test_metrics": {
            "accuracy": test_acc, "precision_w": test_prec, "recall_w": test_rec,
            "macro_f1": test_macro_f1, "balanced_accuracy": test_bal_acc,
            "auroc_macro": test_auroc, "auprc_macro": test_auprc,
            "brier": test_brier, "ece": test_ece
        },
        "feature_importance": feature_imp,
        "shap_summary": shap_summary,
    }
    with open(app.config['ARTIFACTS_PATH'], 'w') as f:
        json.dump(artifacts, f, indent=2)

    print(f"Model trained. Test macro-F1={test_macro_f1:.3f}, AUROC={test_auroc:.3f}, AUPRC={test_auprc:.3f} (method={method})")
    return artifacts
```

```
# User prediction and save history
@app.route('/api/user/predict', methods=['POST'])
def api_user_predict():
    user = _current_user_or_401()
    if not user:
        return jsonify({"error":"Not authenticated"}), 401

    latest = ModelMetrics.query.order_by(ModelMetrics.created_at.desc()).first()
    if not latest or not os.path.exists(app.config['MODEL_PATH']):
        return jsonify({"error":"Model not trained yet"}), 503

    with open(app.config['MODEL_PATH'], 'rb') as f:
        model = pickle.load(f)

    payload = request.get_json(force=True)

    features = {
        'attendance': float(payload['attendance']),
        'gpa': float(payload['gpa']),
        'assignment_avg': float(payload['assignment_avg']),
        'quiz_avg': float(payload['quiz_avg']),
        'days_since_last_activity': float(payload['days_since_last_activity']),
        'financial_status': payload.get('financial_status', 'Stable'),
        'financial_stable': 1 if payload.get('financial_status') == 'Stable' else 0,
        'financial_unstable': 1 if payload.get('financial_status') == 'Unstable' else 0,
        'mental_health_score': int(payload.get('mental_health_score', 6)),
        'extracurricular_activity': 1 if payload.get('extracurricular_activity', False) else 0
    }

    X = pd.DataFrame([{
        'attendance': features['attendance'],
        'gpa': features['gpa'],
        'assignment_avg': features['assignment_avg'],
        'quiz_avg': features['quiz_avg'],
        'days_since_last_activity': features['days_since_last_activity'],
        'financial_stable': features['financial_stable']
```

Figure 5.13.Probability calibration and reliability assessment using isotonic and sigmoid with ECE and macro AUPRC

5.1.9 Endpoints for user prediction, history, authentication, administration

The history reader at /api/user/predictions validates the active session via _current_user_or_401, accepts a limit parameter, orders by creation time, normalizes legacy risk scores from 0 to 1 into a 0 to 100 scale, and returns each item's band, probabilities, features, and a summary count. It also loads persisted artifacts so the client can render calibration context without reloading the model, a pragmatic pattern in Flask with SQLite for compact student services (Anand, Shivaram, and Karthikeyan, 2022). Ownership is enforced by /api/user/predictions/<pred_id> for single deletion and /api/user/predictions/clear for deletion, preventing cross-user access while keeping the payloads stable for dashboards that must remain interpretable over time (Rossi, Adams, Aarons, and McGovern, 2025).

```
@app.route('/api/user/predictions')
def api_user_predictions():
    user = _current_user_or_401()
    if not user:
        return jsonify({"error":"Not authenticated"}), 401

    limit = int(request.args.get('limit', 100))
    q = (PredictionHistory.query
         .filter_by(user_id=user.id)
         .order_by(PredictionHistory.created_at.desc())
         .limit(limit).all())

    items = []
    for r in q:
        # Normalize legacy 0-1 scores to 0-100
        raw = r.risk_score if r.risk_score is not None else 0.0
        score = raw * 100.0 if raw <= 1.0 else raw
        items.append({
            "id": r.id,
            "created_at": r.created_at.isoformat(),
            "prediction": r.prediction,
            "risk_score": round(score, 2),
            "features": json.loads(r.features),
            "probabilities": json.loads(r.probabilities)
        })

    artifacts = {}
    if os.path.exists(app.config['ARTIFACTS_PATH']):
        with open(app.config['ARTIFACTS_PATH'],'r') as f:
            artifacts = json.load(f)

    return jsonify({
        "items": items,
        "summary": {"count": PredictionHistory.query.filter_by(user_id=user.id).count()},
        "artifacts": artifacts
    })
```

```python
# Delete prediction the user owns
@app.route('/api/user/predictions/<int:pred_id>', methods=['DELETE'])
def api_user_prediction_delete(pred_id):
    user = _current_user_or_401()
    if not user:
        return jsonify({"error": "Not authenticated"}), 401
    row = PredictionHistory.query.filter_by(id=pred_id, user_id=user.id).first()
    if not row:
        return jsonify({"error": "Not found"}), 404
    db.session.delete(row)
    db.session.commit()
    return jsonify({"ok": True})

# clear ALL predictions for current user
@app.route('/api/user/predictions/clear', methods=['DELETE'])
def api_user_predictions_clear():
    user = _current_user_or_401()
    if not user:
        return jsonify({"error": "Not authenticated"}), 401
    PredictionHistory.query.filter_by(user_id=user.id).delete()
    db.session.commit()
    return jsonify({"ok": True})

# Authentication
def _current_user_or_401():
    if not session.get('user_logged_in'):
        return None
    email = (session.get('user_email') or '').strip().lower()
    if not email:
        return None
    return User.query.filter(func.lower(User.email) == email).first()

# Views of public pages
@app.route('/')
def root_index(): return render_template('index.html')
```

Figure 5.14.Endpoints for user predictions and history

Public views are served at /, /index.html, and /about.html. The authentication surface comprises /auth.html, /user/register, /user/login and its alias /login, plus /user/logout, with lower-cased emails, password hashing through generate_password_hash, and verification via check_password_hash. The protected profile is available at /user-profile, while /user-profile/update updates name and department and safely persists profile photos using secure_filename, a design consistent with lightweight academic information systems built on Flask and SQLAlchemy (Suraya and Sholeh, 2021). Administrative access is provided through /admin/login and /admin-dashboard, with profile management at /api/admin/me, cohort oversight at /api/admin/users, and record management at /api/admin/users/<user_id>. Finally, /api/students exposes student attributes and the current dropout risk label, supporting early warning workflows in education technology (Wu and Weiland, 2024).

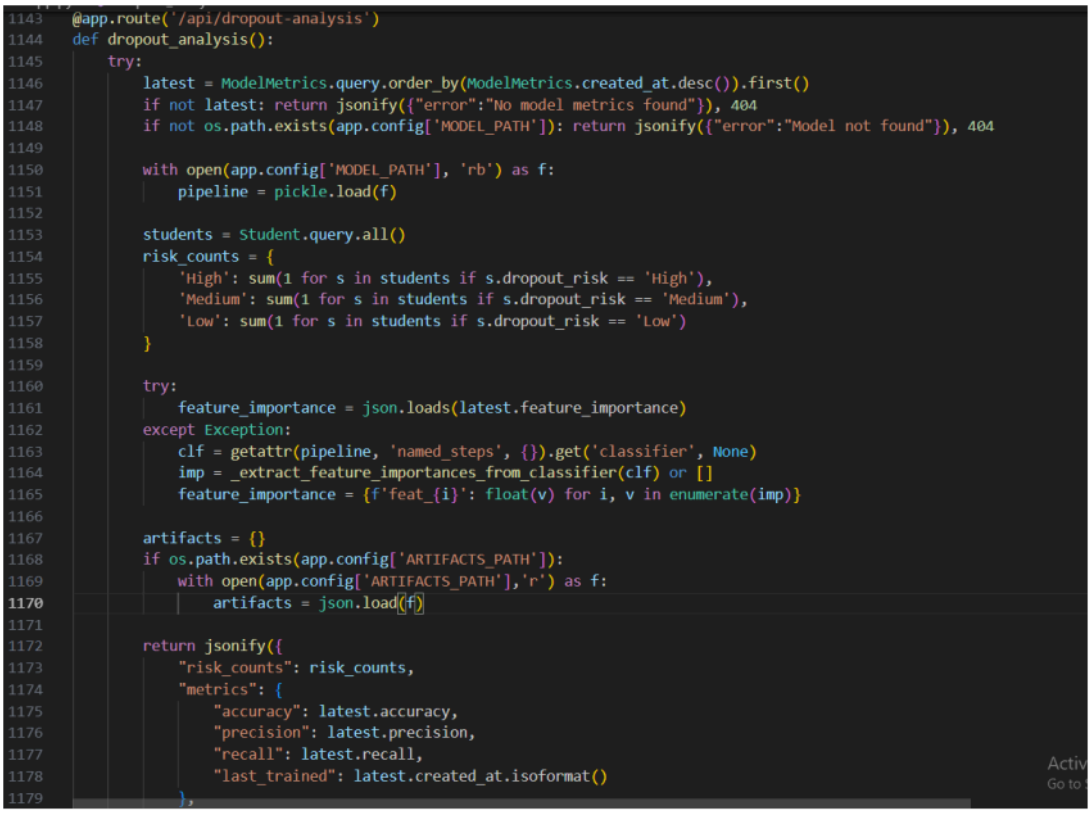
```python
@app.route('/user/register', methods=['POST'], strict_slashes=False)
def user_register():
    if request.is_json:
        data = request.get_json(silent=True) or {}
        name = (data.get('name') or '').strip()
        email = (data.get('email') or '').strip().lower()
        password = data.get('password') or ''
    else:
        name = (request.form.get('name') or '').strip()
        email = (request.form.get('email') or '').strip().lower()
        password = request.form.get('password') or ''
    if not all([name, email, password]):
        if request.is_json:
            return jsonify({"ok": False, "error": "All fields required"}), 400
        flash('All fields required', 'danger'); return redirect(url_for('auth_page'))
    if User.query.filter(func.lower(User.email) == email).first():
        if request.is_json:
            return jsonify({"ok": False, "error": "Email already registered"}), 409
        flash('Email already registered', 'danger'); return redirect(url_for('auth_page'))
    u = User(name=name, email=email, password_hash=generate_password_hash(password))
    db.session.add(u); db.session.commit()
    if request.is_json:
        return jsonify({"ok": True, "user_id": u.id}), 201
    flash('Registration successful!', 'success'); return redirect(url_for('auth_page'))

# login page accepts both /user/login and /login (POST)
@app.route('/user/login', methods=['POST'])
@app.route('/login', methods=['POST'])
def user_login():
    email = (request.form.get('email') or '').strip().lower()
    password = request.form.get('password') or ''
    u = User.query.filter(func.lower(User.email) == email).first()
    if u and check_password_hash(u.password_hash, password):
        session['user_logged_in'] = True; session['user_email'] = u.email
        return redirect(url_for('user_profile'))
    flash('Invalid email or password', 'danger'); return redirect(url_for('auth_page'))
```

```python
# login page accepts both /user/login and /login (POST)
@app.route('/user/login', methods=['POST'])
@app.route('/login', methods=['POST'])
def user_login():
    email = (request.form.get('email') or '').strip().lower()
    password = request.form.get('password') or ''
    u = User.query.filter(func.lower(User.email) == email).first()
    if u and check_password_hash(u.password_hash, password):
        session['user_logged_in'] = True; session['user_email'] = u.email
        return redirect(url_for('user_profile'))
    flash('Invalid email or password', 'danger'); return redirect(url_for('auth_page'))

@app.route('/user/logout')
def user_logout():
    session.pop('user_logged_in', None); session.pop('user_email', None)
    return redirect(url_for('auth_page'))

# User profile (protected)
@app.route('/user-profile')
def user_profile():
    if not session.get('user_logged_in'):
        return redirect(url_for('auth_page'))
    user = User.query.filter_by(email=session['user_email']).first()
    return render_template('user-profile.html', current_user=user)

@app.route('/user-profile.html')
def user_profile_html():
    return redirect(url_for('user_profile'))

@app.route('/user-profile/update', methods=['POST'])
def update_user_profile():
    if not session.get('user_logged_in'):
        return redirect(url_for('auth_page'))
    user = User.query.filter_by(email=session['user_email']).first()
    user.name = request.form.get('name', user.name)
```

Figure 5.15.Endpoints for user authentication and administration

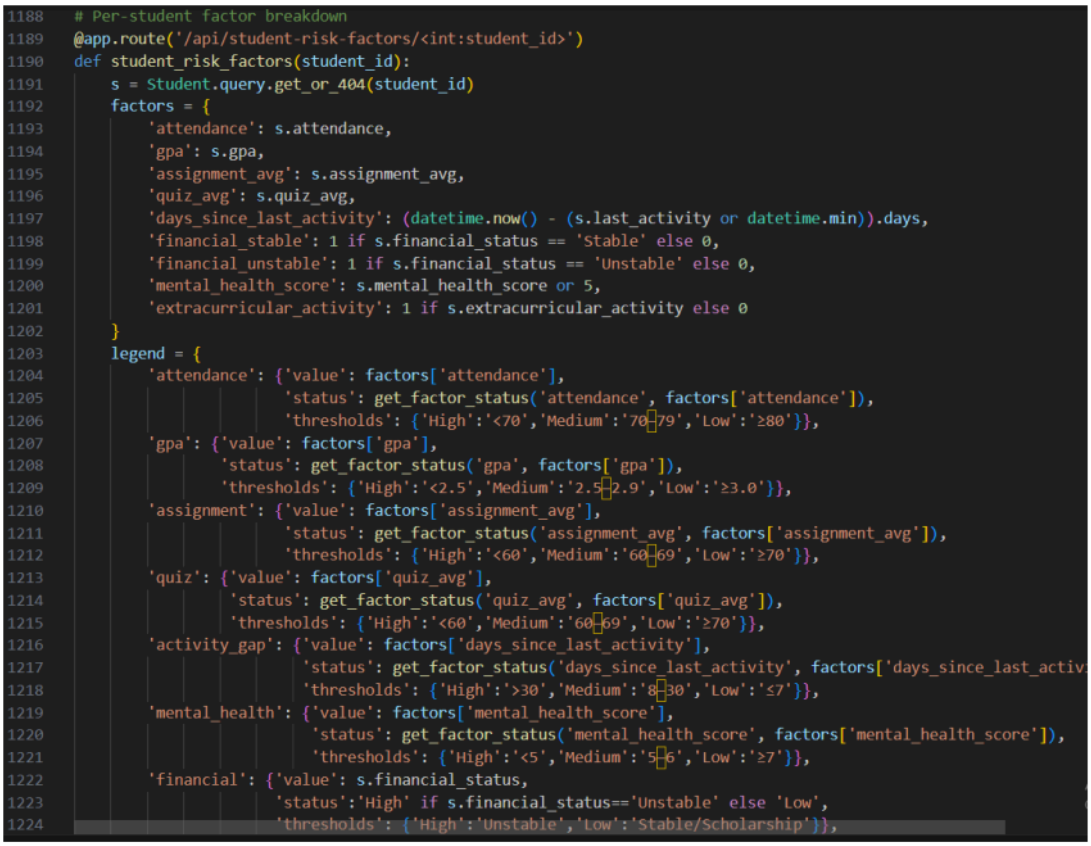
5.1.10 Analytic Dashboard and Real-Time Prediction APIs

/api/dropout-analysis loads the latest ModelMetrics and the persisted calibrated pipeline, then returns population risk counts, summary metrics, confusion matrix, feature importance, and the saved reliability diagram path. Calibration artefacts expose probability quality. /api/student-risk-factors/<id> surfaces a student's raw features and maps each to a traffic light status via get_factor_status. Attendance bands reflect practice where 90% is a salient boundary for monitoring risk in schools (Liza, 2020). Including a mental health signal is motivated by its association with later dropout in longitudinal studies (Lindhardt et al., 2022). /api/predict-risk builds a one row DataFrame, calls predict_proba, and converts probabilities to High, Medium, or Low using locked thresholds that target recall for High and F1 for Low. This balances false negatives and false positives under uncertain costs and class imbalance (Ferri et al., 2019). AUPRC reporting supports rare class evaluation (Richardson et al., 2024).



```python
@app.route('/api/dropout-analysis')
def dropout_analysis():
    try:
        latest = ModelMetrics.query.order_by(ModelMetrics.created_at.desc()).first()
        if not latest: return jsonify({"error":"No model metrics found"}), 404
        if not os.path.exists(app.config['MODEL_PATH']): return jsonify({"error":"Model not found"}), 404

        with open(app.config['MODEL_PATH'], 'rb') as f:
            pipeline = pickle.load(f)

        students = Student.query.all()
        risk_counts = {
            'High': sum(1 for s in students if s.dropout_risk == 'High'),
            'Medium': sum(1 for s in students if s.dropout_risk == 'Medium'),
            'Low': sum(1 for s in students if s.dropout_risk == 'Low')
        }

        try:
            feature_importance = json.loads(latest.feature_importance)
        except Exception:
            clf = getattr(pipeline, 'named_steps', {}).get('classifier', None)
            imp = _extract_feature_importances_from_classifier(clf) or []
            feature_importance = {f'feat_{i}': float(v) for i, v in enumerate(imp)}

        artifacts = {}
        if os.path.exists(app.config['ARTIFACTS_PATH']):
            with open(app.config['ARTIFACTS_PATH'],'r') as f:
                artifacts = json.load(f)

        return jsonify({
            "risk_counts": risk_counts,
            "metrics": {
                "accuracy": latest.accuracy,
                "precision": latest.precision,
                "recall": latest.recall,
                "last_trained": latest.created_at.isoformat()
            },
```

Figure 5.16.Dashboard and Real-Time Prediction APIs



```python
# Per-student factor breakdown
@app.route('/api/student-risk-factors/<int:student_id>')
def student_risk_factors(student_id):
    s = Student.query.get_or_404(student_id)
    factors = {
        'attendance': s.attendance,
        'gpa': s.gpa,
        'assignment_avg': s.assignment_avg,
        'quiz_avg': s.quiz_avg,
        'days_since_last_activity': (datetime.now() - (s.last_activity or datetime.min)).days,
        'financial_stable': 1 if s.financial_status == 'Stable' else 0,
        'financial_unstable': 1 if s.financial_status == 'Unstable' else 0,
        'mental_health_score': s.mental_health_score or 5,
        'extracurricular_activity': 1 if s.extracurricular_activity else 0
    }
    legend = {
        'attendance': {'value': factors['attendance'],
                       'status': get_factor_status('attendance', factors['attendance']),
                       'thresholds': {'High':'<70','Medium':'70–79','Low':'≥80'}},
        'gpa': {'value': factors['gpa'],
                'status': get_factor_status('gpa', factors['gpa']),
                'thresholds': {'High':'<2.5','Medium':'2.5–2.9','Low':'≥3.0'}},
        'assignment': {'value': factors['assignment_avg'],
                       'status': get_factor_status('assignment_avg', factors['assignment_avg']),
                       'thresholds': {'High':'<60','Medium':'60–69','Low':'≥70'}},
        'quiz': {'value': factors['quiz_avg'],
                 'status': get_factor_status('quiz_avg', factors['quiz_avg']),
                 'thresholds': {'High':'<60','Medium':'60–69','Low':'≥70'}},
        'activity_gap': {'value': factors['days_since_last_activity'],
                         'status': get_factor_status('days_since_last_activity', factors['days_since_last_activ
                         'thresholds': {'High':'>30','Medium':'8–30','Low':'≤7'}},
        'mental_health': {'value': factors['mental_health_score'],
                          'status': get_factor_status('mental_health_score', factors['mental_health_score']),
                          'thresholds': {'High':'<5','Medium':'5–6','Low':'≥7'}},
        'financial': {'value': s.financial_status,
                      'status':'High' if s.financial_status=='Unstable' else 'Low',
                      'thresholds': {'High':'Unstable','Low':'Stable/Scholarship'}},
```

Figure 5.17. Dashboard and Real-Time Prediction APIs(per factor break down)

```python
# Individual prediction with locked thresholds
@app.route('/api/predict-risk', methods=['POST'])
def predict_risk():
    try:
        latest = ModelMetrics.query.order_by(ModelMetrics.created_at.desc()).first()
        if not latest or not os.path.exists(app.config['MODEL_PATH']):
            return jsonify({"error": "Model not trained yet"}), 503
        with open(app.config['MODEL_PATH'], 'rb') as f:
            model = pickle.load(f)

        payload = request.get_json(force=True)
        features = {
            'attendance': float(payload['attendance']),
            'gpa': float(payload['gpa']),
            'assignment_avg': float(payload['assignment_avg']),
            'quiz_avg': float(payload['quiz_avg']),
            'days_since_last_activity': float(payload['days_since_last_activity']),
            'financial_stable': 1 if payload.get('financial_status') == 'Stable' else 0,
            'financial_unstable': 1 if payload.get('financial_status') == 'Unstable' else 0,
            'mental_health_score': int(payload.get('mental_health_score', 6)),
            'extracurricular_activity': 1 if payload.get('extracurricular_activity', False) else 0
        }
        X = pd.DataFrame([features])
        proba = model.predict_proba(X)[0]
        labels = list(model.named_steps['classifier'].classes_)
        prob_dict = {labels[i]: float(proba[i]) for i in range(len(labels))}

        if os.path.exists(app.config['ARTIFACTS_PATH']):
            with open(app.config['ARTIFACTS_PATH'],'r') as f:
                art = json.load(f)
            locked = art.get('locked_thresholds', {})
            thr_high = locked.get('alt', {}).get('high_target_recall_0.80',
                                                 locked.get('high_f1', 0.6))
```

Figure 5.18. Individual Prediction Using a locked threshold

5.1.11 Department and Staff Administration APIs

The department endpoint at /api/admin/departments reads all departments with staff counts when invoked via GET. It creates a new department on POST after normalizing a unique slug and optionally seeding an initial staff member. Such RESTful handlers follow common Flask and SQLAlchemy patterns for form or JSON payloads and immediate persistence, which simplifies transactional data management in lightweight academic systems (Suraya and Sholeh, 2021). The update path /api/admin/departments/<int:dept_id> updates the slug and name on PUT with collision checks and removes a department on DELETE, returning proper status codes. Staff management is mirrored in this design at /api/admin/staff for listing by department filter or creating staff bound to a department, and at /api/admin/staff/<int:staff_id> for updates or deletion. This symmetry reduces maintenance overhead and aligns with pragmatic Flask SQLite back ends used in small-scale information systems (Anand et al., 2022). Role-gated helpers _require_admin and _current_staff enforce session checks before any mutation, while _dept_profile_page provides a clean redirect target so authenticated staff land on their department dashboard, supporting navigability and user-centered flows in operations interfaces (Dolatabadi et al., 2024).

```python
# ---- Departments ----
@app.route('/api/admin/departments', methods=['GET', 'POST'])
def api_admin_departments():
    _require_admin()

    if request.method == 'GET':
        rows = (Department.query
                .order_by(Department.slug.asc())
                .all())
        out = []
        for d in rows:
            out.append({
                "id": d.id,
                "slug": d.slug,
                "name": d.name,
                "staff_count": len(d.staff)
            })
        return jsonify(out)

    if request.is_json:
        data = request.get_json(silent=True) or {}
    else:
        data = request.form or {}

    slug_in = data.get('slug') or data.get('department') or ''
    slug = _slugify(slug_in)
    staff_name = (data.get('staff_name') or '').strip()
    staff_email = (data.get('staff_email') or data.get('email') or '').strip().lower()
    staff_password = (data.get('staff_password') or data.get('password') or '')

    if not slug:
        return jsonify({"ok": False, "error": "Department slug is required"}), 400
```

```python
@app.route('/api/admin/departments/<int:dept_id>', methods=['PUT', 'DELETE'])
def api_admin_department_modify(dept_id):
    _require_admin()
    d = Department.query.get_or_404(dept_id)

    if request.method == 'DELETE':
        db.session.delete(d)
        db.session.commit()
        return jsonify({"ok": True})

    if request.is_json:
        data = request.get_json(silent=True) or {}
    else:
        data = request.form or {}
    new_slug = _slugify(data.get('slug') or data.get('new_slug') or d.slug)
    new_name = (data.get('name') or d.name)
    if new_slug != d.slug and Department.query.filter_by(slug=new_slug).first():
        return jsonify({"ok": False, "error": "Slug already in use"}), 409
    d.slug = new_slug
    d.name = new_name
    db.session.commit()
    return jsonify({"ok": True, "department": {"id": d.id, "slug": d.slug, "name": d.name}})
```

Figure 5.19.Admin Department Creation

```python
# ---- Staff ----
@app.route('/api/admin/staff', methods=['GET', 'POST'])
def api_admin_staff():
    _require_admin()

    if request.method == 'GET':
        dept_slug = request.args.get('dept')
        q = Staff.query
        if dept_slug:
            dep = Department.query.filter_by(slug=_slugify(dept_slug)).first()
            if not dep:
                return jsonify([])
            q = q.filter_by(department_id=dep.id)
        rows = q.order_by(Staff.id.desc()).all()
        return jsonify([{
            "id": s.id,
            "name": s.name,
            "email": s.email,
            "department": {"id": s.department.id, "slug": s.department.slug, "name": s.department.name}
        } for s in rows])

    # POST create a staff under a department
    if request.is_json:
        data = request.get_json(silent=True) or {}
    else:
        data = request.form or {}

    dept_slug = _slugify(data.get('dept') or data.get('department') or '')
    dept_id = data.get('department_id')

    dep = None
    if dept_id:
        dep = Department.query.get(dept_id)
    elif dept_slug:
        dep = Department.query.filter_by(slug=dept_slug).first()
```

Figure 5.20.Staff account creation under a dept

```python
@app.route('/api/admin/staff/<int:staff_id>', methods=['PUT', 'DELETE'])
def api_admin_staff_modify(staff_id):
    _require_admin()
    s = Staff.query.get_or_404(staff_id)

    if request.method == 'DELETE':
        db.session.delete(s); db.session.commit()
        return jsonify({"ok": True})

    if request.is_json:
        data = request.get_json(silent=True) or {}
    else:
        data = request.form or {}

    new_name = data.get('name', s.name)
    new_email = (data.get('email') or s.email).strip().lower()
    new_password = data.get('password')
    new_dept_slug = data.get('dept') or data.get('department')
    new_dept_id = data.get('department_id')
    if new_dept_id or new_dept_slug:
        dep = Department.query.get(new_dept_id) if new_dept_id else Department.query.filter_by(slug=_slugify(new_dept_slug)).firs
        if not dep:
            return jsonify({"ok": False, "error": "Target department not found"}), 400
        s.department_id = dep.id

    if new_email != s.email and Staff.query.filter_by(email=new_email).first():
        return jsonify({"ok": False, "error": "Email already in use"}), 409
```

Figure 5.21.Staff account manage & password retrieval

```python
# Staff authentication
def _current_staff():
    """Returns the logged-in Staff row or None."""
    if not session.get('staff_logged_in'):
        return None
    email = (session.get('staff_email') or '').strip().lower()
    if not email:
        return None
    return Staff.query.filter(func.lower(Staff.email) == email).first()

def _dept_profile_page(slug: str) -> str:
    slug = (slug or '').strip().lower()
    mapping = {
        'academic': '/dept-academic.html',
        'finance': '/dept-finance.html',
        'mental_health': '/dept-mental-health.html',
        'mental-health': '/dept-mental-health.html',
    }
    return mapping.get(slug, '/admin-dashboard')
```

Figure 5.22. Staff authentication

5.1.12 Department directory and internal messaging services

The staff listing endpoint at /api/department/staff lets authenticated staff discover colleagues by scope. With dept=same it returns only peers from the caller department, dept=all returns everyone, and a specific slug returns that unit. Results are ordered by name and serialized through _json_staff_public, which exposes only safe identity fields. The companion endpoint /api/department/admins lists administrative contacts and requires an active staff session. Session checks use _current_staff, and all handlers reply with compact JSON for straightforward client rendering, a typical pattern in Flask data apps that rely on SQLite persistence (Suraya and Sholeh, 2021). The unified mail endpoint /api/emails implements send and list. On POST it validates recipients and body, inserts EmailLog rows, and returns the count sent. On GET it supports mailboxes named inbox, sent, and trash with search over subject and body, plus soft delete flags that decide visibility. This log-centric approach enables later auditing of communications and aligns with recommendations for traceable application logging in responsible machine learning systems (Foalem et al., 2025).

```python
# --- LIST STAFF  ---
@app.get("/api/department/staff")
def api_department_staff():
    s = _current_staff()
    scope = (request.args.get("dept") or "").strip().lower()

    q = Staff.query
    if scope in ("", "same"):
        q = q.filter(Staff.department_id == s.department_id)
    elif scope != "all":
        dep = Department.query.filter(func.lower(Department.slug) == scope).first()
        if not dep: return jsonify([])
        q = q.filter(Staff.department_id == dep.id)

    rows = q.order_by(func.lower(Staff.name)).all()
    return jsonify([_json_staff_public(x) for x in rows])
```

```python
@app.get("/api/department/admins")
def api_department_admins():
    s = _current_staff()  # require a staff session
    if not s:
        return jsonify({"error": "Not authenticated"}), 401
    rows = Admin.query.order_by(func.lower(Admin.name)).all()
    out = []
    for a in rows:
        out.append({
            "id": a.id,
            "name": a.name or "Admin",
            "email": a.email,
            "department": {"slug": "administration", "name": a.department or "Administration"}
        })
    return jsonify(out)
```

Figure 5.23.Staff list

```python
@app.route("/api/emails", methods=["GET", "POST"])
def api_emails():
    # use override-aware resolver
    role, me = _whoami_from_request()

    # SEND
    if request.method == "POST":
        data = request.get_json(force=True) or {}
        subject = (data.get("subject") or data.get("topic") or "Message").strip()
        body    = (data.get("body") or "").strip()
        to_raw  = data.get("to") or data.get("to_email") or data.get("to_emails")
        recipients = [to_raw] if isinstance(to_raw, str) else [str(x) for x in (to_raw or [])]
        recipients = [x.strip().lower() for x in recipients if x and "@" in x]

        if not recipients:
            return jsonify({"error": "Recipient required"}), 400
        if not body:
            return jsonify({"error": "Message body is required"}), 400

        for addr in recipients:
            db.session.add(EmailLog(
                from_role=role, from_email=me, to_email=addr,
                subject=subject, body=body
            ))
        db.session.commit()

        print(f"[mail] {role} {me} to {', '.join(recipients)} : {subject}\n{body}\n", flush=True)
        return jsonify({"ok": True, "sent": len(recipients)})

    # LIST
    box = (request.args.get("box") or "inbox").strip().lower()  # inbox|sent|trash
    q   = (request.args.get("q") or "").strip().lower()
    limit = max(1, min(int(request.args.get("limit", 200)), 500))

    query = EmailLog.query
```

Figure 5.24.Emails set up

5.1.13 Application Bootstrapping and Runtime

At startup the program opens a Flask application context, creates tables, and performs light schema checks on SQLite using PRAGMA table info, a practical pattern for small education systems built with Flask and SQLite (Anand et al., 2022; Suraya and Sholeh, 2021). Column gaps such as profile photo in staff and read flags in email log are added with ALTER TABLE ADD COLUMN, which is safe for additive evolution in SQLite backed prototypes (Anand et al., 2022). When the database is empty the system seeds realistic student records and inserts a default administrator, then warms the model by training once so that dashboards and prediction routes respond immediately, while persisting model files and artifacts for reproducibility and traceability (Bosch, 2020; Cândido et al., 2019). Finally, the server runs in debug mode with the reloader disabled to prevent duplicate initialization work, such as reseeding or retraining, during development, a simple but effective guard for consistent startup behavior in data-centric Flask services (Anand et al., 2022).

```python
# Entrypoint
if __name__ == '__main__':
    with app.app_context():
        db.create_all()
        basedir = os.path.abspath(os.path.dirname(__file__))
        app.config.setdefault('SQLALCHEMY_DATABASE_URI', f"sqlite:///{os.path.join(basedir, 'app.db')}")
        app.config['MODEL_PATH'] = app.config.get('MODEL_PATH') or os.path.join(basedir, 'dropout_model.pkl')
        app.config['ARTIFACTS_PATH'] = app.config.get('ARTIFACTS_PATH') or os.path.join(basedir, 'model_artifacts.json')
        db_path = os.path.join(basedir, 'app.db')
        model_path = app.config['MODEL_PATH']
        artifacts_path = app.config['ARTIFACTS_PATH']
        try:
            from sqlalchemy import text, inspect
            insp = inspect(db.engine)
            if insp.has_table('staff'):
                res = db.session.execute(text("PRAGMA table_info(staff)"))
                cols = []
                for row in res:
                    # SQLAlchemy 2.x Row: use _mapping; fallback to positional index for older versions
                    mapping = getattr(row, "_mapping", None)
                    cols.append(mapping['name'] if mapping and 'name' in mapping else row[1])
                if 'profile_photo' not in cols:
                    db.session.execute(text("ALTER TABLE staff ADD COLUMN profile_photo VARCHAR(200)"))
                    db.session.commit()
        except Exception as mig_err:
            print(f"[warn] Skipped staff.profile_photo migration check: {mig_err}", flush=True)

        print(
            "\nUsing files:\n"
            f" • DB:       {db_path}\n"
            f" • Model:    {model_path}\n"
            f" • Artifacts:{artifacts_path}\n",
            flush=True
        )
```

```python
        )
        try:
            from sqlalchemy import text, inspect
            insp = inspect(db.engine)
            if insp.has_table('email_log'):
                res = db.session.execute(text("PRAGMA table_info(email_log)"))
                cols = []
                for row in res:
                    mapping = getattr(row, "_mapping", None)
                    cols.append(mapping['name'] if mapping and 'name' in mapping else row[1])
                if 'read_by_recipient' not in cols:
                    db.session.execute(text("ALTER TABLE email_log ADD COLUMN read_by_recipient BOOLEAN DEFAULT 0"))
                if 'deleted_by_sender' not in cols:
                    db.session.execute(text("ALTER TABLE email_log ADD COLUMN deleted_by_sender BOOLEAN DEFAULT 0"))
                if 'deleted_by_recipient' not in cols:
                    db.session.execute(text("ALTER TABLE email_log ADD COLUMN deleted_by_recipient BOOLEAN DEFAULT 0"))
                db.session.commit()
        except Exception as mig_err:
            print(f"[warn] Skipped email_log migration check: {mig_err}", flush=True)

        try:
            need_seed = (not os.path.exists(db_path)) or (Student.query.count() == 0)
        except Exception:
            need_seed = True

        if need_seed:
            print("Generating enhanced student data...", flush=True)
            generate_random_students(1000)
            print(f"Generated {Student.query.count()} students.", flush=True)
```

Figure 5.25. Application Bootstrapping and Runtime

5.2 UI Implementation

5.2.1 auth.html

Tab switching uses switchTab with new bootstrap.Tab and the show method to activate panes referenced by #login-tab, #register-tab, #staff-tab, and #admin-tab via data-bs-toggle and data-bs-target. This in-place

navigation maintains context and lowers interaction cost in dashboard workflows (Dolatabadi et al., 2024; Rossi et al., 2025).

```
<!-- Auth Section -->
<div class="auth-container">
  <div class="auth-wrapper">
    <div class="auth-header">
      <h1 class="auth-title">Account Access</h1>
      <p class="auth-subtitle">Choose your authentication method below</p>
    </div>

    <ul class="nav form-tabs nav-tabs" id="authTabs" role="tablist">
      <li class="nav-item" role="presentation">
        <button class="nav-link active" id="login-tab" data-bs-toggle="tab" data-bs-target="#login" type="button" role="tab">
          Login
        </button>
      </li>
      <li class="nav-item" role="presentation">
        <button class="nav-link" id="register-tab" data-bs-toggle="tab" data-bs-target="#register" type="button" role="tab">
          Register
        </button>
      </li>
      <li class="nav-item" role="presentation">
        <button class="nav-link" id="staff-tab" data-bs-toggle="tab" data-bs-target="#staff" type="button" role="tab">
          Staff
        </button>
      </li>
      <!-- Admin tab -->
      <li class="nav-item" role="presentation">
        <button class="nav-link" id="admin-tab" data-bs-toggle="tab" data-bs-target="#admin" type="button" role="tab">
          Admin
        </button>
      </li>
    </ul>
```

Figure 5.26.auth.html aside

The Staff flow binds submit on #staffForm, calls preventDefault, disables #staffLoginBtn, and posts with fetch to the staff endpoint using a JSON body created with JSON.stringify. The response path first honors r.redirected, otherwise awaits r.json, checks r.ok, extracts data?.staff?.department?.slug, resolves a target with pageFor for academic, finance, or mental health, and sets window.location.href.

```
// Staff login
(function(){
  const form = document.getElementById('staffForm');
  const btn  = document.getElementById('staffLoginBtn');
  const alertBox = document.getElementById('staffAlert');
  if (!form) return;

  function pageFor(slug) {
    const key = String(slug || '').toLowerCase();
    const map = {
      'academic': '/dept-academic.html',
      'finance': '/dept-finance.html',
      'mental_health': '/dept-mental-health.html',
      'mental-health': '/dept-mental-health.html'
    };
    return map[key] || '/admin-dashboard';
  }

  function showError(msg){
    if(!alertBox) return;
    alertBox.textContent = msg || 'Invalid email or password';
    alertBox.classList.remove('d-none');
  }
  function hideError(){
    if(!alertBox) return;
    alertBox.classList.add('d-none');
    alertBox.textContent = '';
  }
```

```
    const r = await fetch('/staff/login', {
      method: 'POST',
      headers: {'Content-Type':'application/json'},
      body: JSON.stringify({ email, password })
    });

    if (r.redirected) {
      window.location.href = r.url;
      return;
    }

    let data = {};
    try { data = await r.json(); } catch {}

    if (!r.ok) {
      showError(data.error || 'Invalid email or password');
      btn.disabled = false;
      return;
    }

    const slug = data?.staff?.department?.slug || data?.department?.slug;
    const dest = data?.redirect || pageFor(slug);
    window.location.href = dest;

  } catch (err) {
    form.submit();
  } finally {
    btn.disabled = false;
```

Figure 5.27.Auth -staff login

Errors toggle #staffAlert by adding or removing d-none, network failure falls back to form.submit, and finally restores the button. These choices reflect human-centred transparency and recoverability (Fu & Weng, 2024; Zhu et al., 2025) and observable, auditable behaviour (Bosch, 2020; Cândido et al., 2019).

```
// Admin login
(function(){
  const form = document.getElementById('adminForm');
  const btn  = document.getElementById('adminLoginBtn');
  const alertBox = document.getElementById('adminAlert');
  if (!form) return;

  function showError(msg){
    if(!alertBox) return;
    alertBox.textContent = msg || 'Invalid admin email or password';
    alertBox.classList.remove('d-none');
  }
  function hideError(){
    if(!alertBox) return;
    alertBox.classList.add('d-none');
    alertBox.textContent = '';
  }

  form.addEventListener('submit', async (e)=>{
    hideError();
    e.preventDefault();
    btn.disabled = true;

    try {
      const email = (form.email.value || '').trim();
      const password = (form.password.value || '').trim();
      if (!email || !password) {
        showError('Enter admin email and password.');
        btn.disabled = false;
        return;
      }

      // Use form-encoded to match the backend route
```

Figure 5.28.auth-admin login

The Admin flow mirrors this on #adminForm using URLSearchParams and application form URL-encoded, surfacing failures in #adminAlert, consistent with responsible ML operations (Foalem et al., 2025).

5.2.2 Admin-Dashbaord.html

A fixed header, navbar, a collapsible aside#sidenav, and main establish the shell. Cards, progress bars, and tables display KPIs and governance elements that are consistent with dashboard HCI guidance (Rossi et al., 2025; Dolatabadi et al., 2024).

```
</header>

<div class="sidenav-backdrop" id="backdrop"></div>

<div class="app">
  <!-- Aside -->
  <aside id="sidenav">
    <div class="brand"><div class="logo"><span>Admin Console</span></div></div>
    <nav class="sidenav py-2">

      <a href="#overview" class="active">Overview</a>
      <a href="#analytics">Analytics</a>
      <a href="#governance">Governance & Validation</a>
      <a href="#users">Manage Students</a>
      <a href="#mc-students">Monte Carlo Simulation</a>
      <a href="#predict">Predict Risk</a>
      <a href="admin-departments.html">Manage Departments & Staffs</a>


    </nav>
  </aside>

```

Figure 5.29.admin dashboard - aside

JavaScript organizes behavior into consistent functions. Hash-based routing in showSectionFromHash() toggles sections and coordinates the mobile drawer via toggleSidenav(). Utility helpers (pct, setBar, fmtDate, fmtMeanStd, riskBadge, badgeClass) standardize formatting. loadAnalysis() calls /api/dropout-analysis, updates KPI spans (#kpi-acc, #kpi-prec, #kpi-recall), progress bars, and renders three Chart.js visuals (fiChart, riskDistChart, metricsChart).

```
<script>
  /* Global state */
  let dtUsers=null, dtStudents=null, fiChart=null, metricsChart=null, riskDistChart=null, probChart=null;

  /* Sidebar drawer */
  const sidenav=document.getElementById('sidenav'), backdrop=document.getElementById('backdrop');
  const toggleSidenav=(open)=>{ if(open){sidenav.classList.add('open'); backdrop.classList.add('show');} else {sidenav.classList
  document.getElementById('toggleSidenav').addEventListener('click', ()=>toggleSidenav(true));
  backdrop.addEventListener('click', ()=>toggleSidenav(false));

  /* Utilities */
  const pct=(v)=> (v==null? '—' : (v*100).toFixed(1)+'%');
  const setBar=(el,v)=>{ if(el) el.style.width=Math.max(0,Math.min(100,v*100))+'%'; };
  const fmtDate=(s)=> s ? dayjs.utc(s).local().format('YYYY-MM-DD HH:mm') : '—';
  const fmtMeanStd=(obj,key)=>{ const m=obj?.[key]?.mean, sd=obj?.[key]?.std; return (m==null||sd==null)?'—':`${m.toFixed(3)} ±
  const riskBadge=(r)=>{ const m={High:'text-bg-danger',Medium:'text-bg-warning',Low:'text-bg-success'}; return `<span class="ba
  const badgeClass=(s)=> s==='High'?'text-bg-danger':(s==='Medium'?'text-bg-warning':(s==='Low'?'text-bg-success':'text-bg-secon
```

Figure 5.30.admin dashboard - mobile view & utility helpers

It also populates nested-CV summaries and displays the chosen calibrator and locked thresholds, aligning with probability calibration and operating-point selection practices (Silva Filho et al., 2023; Ferri et al., 2019).

```
693     /* Load analysis */
694     async function loadAnalysis(){
695       try{
696         const res=await fetch('/api/dropout-analysis', { credentials: 'same-origin' });
697         if(!res.ok) throw new Error('Failed to load analysis');
698         const data=await res.json();
699
700         const acc=data.metrics?.accuracy, prec=data.metrics?.precision, rec=data.metrics?.recall;
701         document.getElementById('kpi-acc').textContent=pct(acc);
702         document.getElementById('kpi-prec').textContent=pct(prec);
703         document.getElementById('kpi-recall').textContent=pct(rec);
704         document.getElementById('kpi-last').textContent=fmtDate(data.metrics?.last_trained);
705         setBar(document.getElementById('bar-acc'), acc||0);
706         setBar(document.getElementById('bar-prec'), prec||0);
707         setBar(document.getElementById('bar-recall'), rec||0);
708
709         const r=data.risk_counts||{}; const total=(r.High||0)+(r.Medium||0)+(r.Low||0);
710         document.getElementById('risk-high').textContent=`${r.High||0}`;
711         document.getElementById('risk-medium').textContent=`${r.Medium||0}`;
712         document.getElementById('risk-low').textContent=`${r.Low||0}`;
713         setBar(document.getElementById('bar-high'), total? (r.High/total):0);
714         setBar(document.getElementById('bar-medium'), total? (r.Medium/total):0);
715         setBar(document.getElementById('bar-low'), total? (r.Low/total):0);
716
717         const fi=data.feature_importance||{};
718         const pairs=Object.entries(fi).sort((a,b)=>b[1]-a[1]).slice(0,12);
719         const labels=pairs.map(([k])=>k), vals=pairs.map(([,v])=>v);
720         if(fiChart) fiChart.destroy();
721         fiChart=new Chart(document.getElementById('fiChart'),{
722           type:'bar',
723           data:{labels, datasets:[{label:'Importance', data:vals}]},
724           options:{responsive:true, maintainAspectRatio:true, indexAxis:'y', plugins:{legend:{display:fals
725         });
726
```

Figure 5.31.admin dashboard - analysis

User management is loaded via loadUsers() from /api/admin/users, rendered into #usersTable, and enhanced with DataTables (including paging and CSV export). Inline actions open the "Edit User" modal and issue PUT/DELETE requests to /api/admin/users/{id}. Student monitoring uses loadStudents() against /api/students, with risk-chip filters wired by setupRiskFilters() to a column search. The "Explain" flow opens #explainModal by fetching /api/student-risk-factors/{id} and drawing a human-readable legend in openExplain().

```
/* Users table */
async function loadUsers(){
  try{
    const res=await fetch('/api/admin/users', { credentials: 'same-origin' });
    if(!res.ok) return;
    const users=await res.json();
    const tbody=document.querySelector('#usersTable tbody'); tbody.innerHTML='';
    users.forEach(u=>{
      const tr=document.createElement('tr');
      const photo = u.profile_photo || '/uploads/default.png';
      const bandBadge = (u.last_band ? riskBadge(u.last_band) : '<span class="text-muted">—</span>');
      const riskTxt = (Number.isFinite(u.last_risk) ? `${Number(u.last_risk).toFixed(1)}/100` : '—');
      tr.innerHTML=`<td>${u.id}</td>
        <td><img src="${photo}" alt="photo" class="rounded-circle border" style="width:44px;height:44px;object-fit:cover"></td
        <td>${u.name||''}</td><td>${u.email||''}</td><td>${u.role||'user'}</td><td>${u.status||'active'}</td>
        <td>${bandBadge}</td>
        <td>${riskTxt}</td>
        <td>
          <button class="btn btn-sm btn-outline-primary me-1" data-action="edit"
            data-id="${u.id}" data-name="${u.name||''}" data-email="${u.email||''}" data-photo="${photo}">
            <i class="fa-solid fa-pen"></i></button>
          <button class="btn btn-sm btn-outline-danger" data-action="del" data-id="${u.id}">
            <i class="fa-solid fa-trash"></i></button>
        </td>`;
      tbody.appendChild(tr);
    });
    if(dtUsers) dtUsers.destroy();
    dtUsers = $('#usersTable').DataTable({
      pageLength:10, lengthChange:false, order:[[0,'desc']],
      dom:'Bfrtip', buttons:[{extend:'csvHtml5', className:'btn btn-sm btn-outline-secondary'}],
      autoWidth:false, responsive:true
```

```
/* Students table */
async function loadStudents(){
  const res = await fetch('/api/students', { credentials: 'same-origin' });
  if(!res.ok) return;
  const students = await res.json();

  const tbody = document.querySelector('#studentsTable tbody');
  tbody.innerHTML = '';

  students.forEach(s=>{
    const tr = document.createElement('tr');
    tr.innerHTML = `
      <td>${s.id}</td><td>${s.name}</td><td>${s.email}</td>
      <td>${Number(s.attendance).toFixed(0)}%</td><td>${Number(s.gpa).toFixed(2)}</td>
      <td>${Number(s.assignment_avg).toFixed(0)}%</td><td>${Number(s.quiz_avg).toFixed(0)}%</td>
      <td>${riskBadge(s.dropout_risk)}</td>
      <td>${(s.last_activity_days??'—')} days</td>
      <td>${s.financial_status||'—'}</td><td>${s.mental_health_score??'—'}</td>
      <td>${s.extracurricular_activity?'Yes':'No'}</td>
      <td>
        <button class="btn btn-sm btn-outline-info"
                data-explain="${s.id}"
                data-name="${s.name}">
          <i class="fa-regular fa-eye"></i>
        </button>
      </td>`;
    tbody.appendChild(tr);
  });
```

Figure 5.32.admin dashboard -load users and synthetic students' data

The real-time inference form #predictForm composes a JSON payload, posts it to /api/predict-risk, and renderPrediction() binds the returned prediction, risk score, and probabilities into a doughnut chart, along with factor statuses, surfacing locked thresholds for transparency (Patel et al., 2021; Fu & Weng, 2024).

```
/* Predict form */
document.getElementById('predictForm').addEventListener('submit', async (e)=>{
  e.preventDefault();
  const f=e.target;
  const payload={
    attendance:Number(f.attendance.value), gpa:Number(f.gpa.value),
    assignment_avg:Number(f.assignment_avg.value), quiz_avg:Number(f.quiz_avg.value),
    days_since_last_activity:Number(f.days_since_last_activity.value),
    financial_status:f.financial_status.value, mental_health_score:Number(f.mental_health_score.value),
    extracurricular_activity:(f.extracurricular_activity.value==='true')
  };
  try{
    const res=await fetch('/api/predict-risk',{method:'POST', headers:{'Content-Type':'application/json'}, body:JSON.stringify
    if(!res.ok){ const err=await res.json().catch(()=>({error:'Prediction failed'})); throw new Error(err.error||'Prediction f
    const data=await res.json(); renderPrediction(data);
    document.getElementById('predResult').scrollIntoView({behavior:'smooth', block:'nearest'});
  }catch(err){ alert(err.message); }
});

function renderPrediction(data){
  document.getElementById('predEmpty').classList.add('d-none');
  document.getElementById('predResult').classList.remove('d-none');
  document.getElementById('pred-model').textContent='Calibrated RF';

  const band=data.prediction||'–';
  const badge=document.getElementById('predBadge'); badge.className='badge rounded-pill';
  if(band==='High') badge.classList.add('text-bg-danger'); else if(band==='Medium') badge.classList.add('text-bg-warning'); el
  badge.textContent=band;
```

Figure 5.33.admin dashboard - prediction form

Profile self-service loads /api/admin/me in loadProfile() and persists via PUT. Finally, adjustTables() ensures that DataTables remain responsive on resize, and init() arranges the page for a smooth startup. The design complements the Flask back end and supports auditable, responsible administrative workflows (Suraya & Sholeh, 2021).

```
/* Profile modal */
const profileModalEl=document.getElementById('profileModal');
const profileModal= new bootstrap.Modal(profileModalEl);
document.getElementById('btnProfile').addEventListener('click', async ()=>{ await loadProfile(); profileModal.show(); });
async function loadProfile(){
  try{
    const r=await fetch('/api/admin/me', {credentials:'same-origin'});
    const me= r.ok ? (await r.json()) : {};
    profileForm.name.value = me.name || '{{ current_user.name if current_user else "" }}';
    profileForm.email.value = me.email || '{{ current_user.email if current_user else "" }}';
    profileForm.department.value = me.department || '{{ current_user.department if current_user else "" }}';
    document.getElementById('profilePhotoPreview').src = me.profile_photo || '{{ current_user.profile_photo if current_user el
    profileForm.profile_photo.value = "";
  }catch(_){
    document.getElementById('profilePhotoPreview').src = '{{ current_user.profile_photo if current_user else "/uploads/default
  }
}
document.getElementById('profileForm').addEventListener('submit', async (e)=>{
  e.preventDefault();
  const fd=new FormData(e.target);
  try{
    const r=await fetch('/api/admin/me', {method:'PUT', body:fd, credentials:'same-origin'});
    if(r.ok){ profileModal.hide(); location.reload(); }
    else { alert('Failed to update profile'); }
  }catch(_){ alert('Failed to update profile'); }
});
```

Figure 5.34. admin dashboard - edit admin profile

5.2.3 User-profie.html

The interface implements a dynamic dashboard with five main sections: Overview, Predict, Progress, History, and Mailbox. Section navigation is managed through a showSection(id) function that toggles visibility between panels.

```
<div class="app">
  <!-- SIDEBAR -->
  <aside id="sidenav">
    <div class="brand">User Console</div>
    <div class="p-3 border-bottom">
      <div class="d-flex align-items-center gap-3">
        {% if current_user.profile_photo %}
          <img src="{{ current_user.profile_photo }}" class="avatar" alt="User photo">
        {% else %}
          <img src="/uploads/default.png" class="avatar" alt="User photo" onerror="this.style.visibility='hidden'">
        {% endif %}
        <div>
          <div class="fw-semibold">{{ current_user.name }}</div>
          <div class="mini">{{ current_user.email }}</div>
        </div>
      </div>
    </div>

    <nav class="sidenav py-2">
      <a href="#overview" class="nav-link active" data-link="overview">Overview</a>
      <a href="#predict"  class="nav-link" data-link="predict">Predict</a>
      <a href="#progress" class="nav-link" data-link="progress">Progress</a>
      <a href="#history"  class="nav-link" data-link="history">History</a>
      <a href="#mailbox" class="nav-link" data-link="mailbox">Mailbox</a>
    </nav>

    <div class="p-3">
      <div class="mini">Model</div>
      <div><span id="modelName" class="badge text-bg-light">—</span></div>
      <div class="mini mt-3">Calibration</div>
      <div><span id="calibInfo" class="badge text-bg-light">—</span></div>
      <div class="mini mt-3">Last trained</div>
      <div><span id="lastTrained" class="badge text-bg-light">—</span></div>
```



```
/*  Model meta  */
function setModelMeta(meta){
  const artifacts = meta?.artifacts || meta || {};
  const modelName = artifacts?.model || meta?.model || '—';
  const calib = (artifacts?.calibration?.chosen_method) || (meta?.calibration?.chosen_method) || '—';
  const last = (meta?.metrics?.last_trained) || (meta?.model_meta?.last_trained) || null;
  const lastStr = last ? new Date(last).toLocaleString() : (document.getElementById('lastTrained').textContent || '—');

  ['modelName','ovModel','predModel'].forEach(id => { const el=document.getElementById(id); if(el) el.textContent=modelName; });
  ['calibInfo','ovCalib'].forEach(id => { const el=document.getElementById(id); if(el) el.textContent=calib; });
  ['lastTrained','ovLast'].forEach(id => { const el=document.getElementById(id); if(el) el.textContent=lastStr; });

  const locked = artifacts?.locked_thresholds || meta?.locked_thresholds || {};
  const altH = locked?.alt?.['high_target_recall_0.80'];
  const thrH = (locked?.high ?? altH);
  const thrL = (locked?.low ?? locked?.low_f1);
  if(thrH!=null) document.getElementById('ovThrH').textContent = nfmt(thrH,2);
  if(thrL!=null) document.getElementById('ovThrL').textContent = nfmt(thrL,2);
}
function setProbBars({High=0,Medium=0,Low=0},{txtH,txtM,txtL,barH,barM,barL}){
  txtH.textContent=pct(High,1);  barH.style.width=(High*100||0)+'%';
  txtM.textContent=pct(Medium,1);barM.style.width=(Medium*100||0)+'%';
  txtL.textContent=pct(Low,1);   barL.style.width=(Low*100||0)+'%';
}
```

Figure 5.35.user profile - aside and load model, calibration, last trained

Client-server data synchronization employs fetch(..., { credentials: 'same-origin' }) to maintain session integrity during API communications. The backend utilizes Jinja templating to hydrate initial state through context variables, ensuring server-side data is immediately available at render time.

The analytics panel surfaces critical model metadata including:

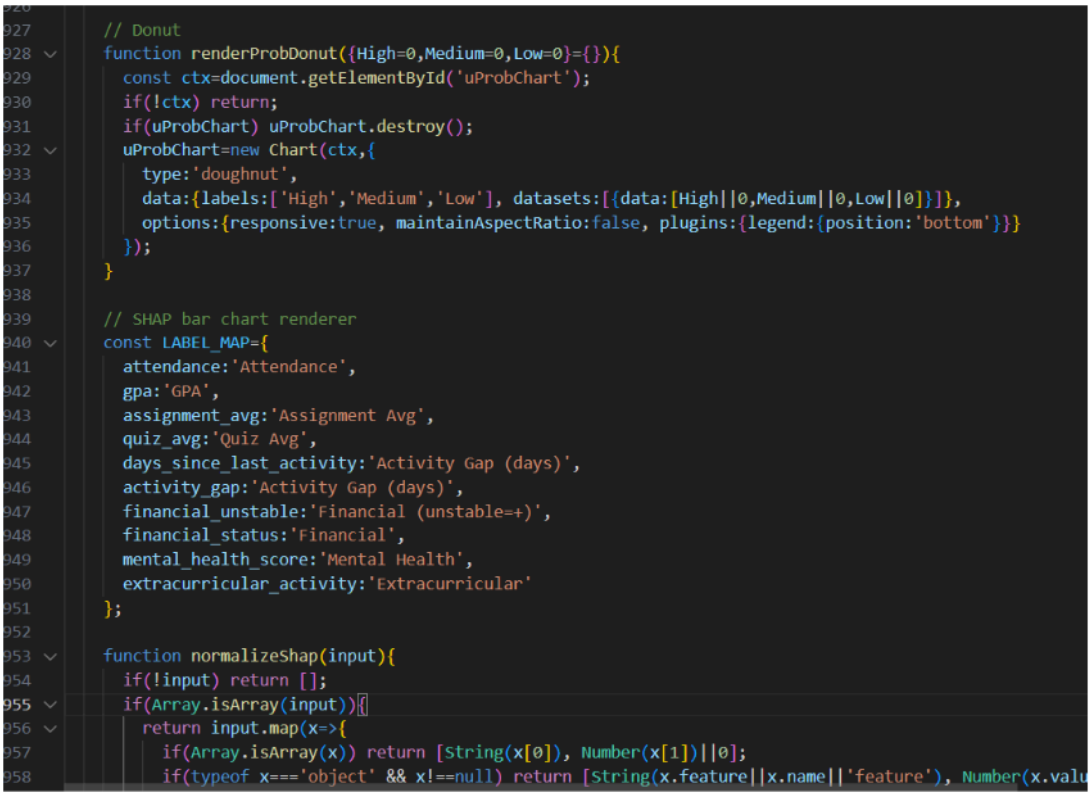
- Active model identification (model_name)
- Calibration configuration (calibration.chosen_method)
- Training recency metrics (metrics.last_trained)
- System-configured risk thresholds in locked state

The prediction form posts features attendance, GPA, assignment, and quiz averages, etc to /api/user/predict. The response updates band, 0–100 risk score, and posterior probabilities (High, Medium, Low) with threshold text for interpretability.

```
</section>

<!-- PREDICT -->
<section id="predict" data-section class="mb-tight">
  <div class="row g-tight">
    <div class="col-12 col-xl-6 d-flex">
      <div class="card h-100 flex-fill">
        <div class="card-header fw-semibold">Input Factors</div>
        <div class="card-body">
          <form id="predictForm" class="row g-tight">
            <div class="col-6">
              <label class="form-label mini">Attendance (%)</label>
              <input class="form-control" type="number" name="attendance" min="0" max="100" step="1" value="82" required
            </div>
            <div class="col-6">
              <label class="form-label mini">GPA (0-4)</label>
              <input class="form-control" type="number" name="gpa" min="0" max="4" step="0.01" value="3.1" required>
            </div>
            <div class="col-6">
              <label class="form-label mini">Assignment Avg</label>
              <input class="form-control" type="number" name="assignment_avg" min="0" max="100" step="1" value="74" requ
            </div>
            <div class="col-6">
              <label class="form-label mini">Quiz Avg</label>
              <input class="form-control" type="number" name="quiz_avg" min="0" max="100" step="1" value="70" required>
            </div>
            <div class="col-6">
              <label class="form-label mini">Days Since Last Activity</label>
```

Figure 5.36.User profile - prediction form

Visualizations are implemented with Chart.js v4, including (a) a time-series trend line, (b) a doughnut chart to display class posterior probabilities, and (c) a horizontal SHAP contribution bar chart (indexAxis = "y") with symmetric x-axis limits to represent signed feature attributions (Chart.js Contributors, 2024).



```
// Donut
function renderProbDonut({High=0,Medium=0,Low=0}={}){
  const ctx=document.getElementById('uProbChart');
  if(!ctx) return;
  if(uProbChart) uProbChart.destroy();
  uProbChart=new Chart(ctx,{
    type:'doughnut',
    data:{labels:['High','Medium','Low'], datasets:[{data:[High||0,Medium||0,Low||0]}]},
    options:{responsive:true, maintainAspectRatio:false, plugins:{legend:{position:'bottom'}}}
  });
}

// SHAP bar chart renderer
const LABEL_MAP={
  attendance:'Attendance',
  gpa:'GPA',
  assignment_avg:'Assignment Avg',
  quiz_avg:'Quiz Avg',
  days_since_last_activity:'Activity Gap (days)',
  activity_gap:'Activity Gap (days)',
  financial_unstable:'Financial (unstable=+)',
  financial_status:'Financial',
  mental_health_score:'Mental Health',
  extracurricular_activity:'Extracurricular'
};

function normalizeShap(input){
  if(!input) return [];
  if(Array.isArray(input)){
    return input.map(x=>{
      if(Array.isArray(x)) return [String(x[0]), Number(x[1])||0];
      if(typeof x==='object' && x!==null) return [String(x.feature||x.name||'feature'), Number(x.valu
```

```
<script>
  function renderShap(input){
    const ctx=document.getElementById('shapChart');
    if(!ctx) return;

    if(!rows.length){
      emptyEl.classList.remove('d-none'); // show message
      if(shapChart){ shapChart.destroy(); shapChart=null; }
      return;
    }
    emptyEl.classList.add('d-none');

    // sort by absolute contribution and take top 8
    rows.sort((a,b)=>Math.abs(b[1]) - Math.abs(a[1]));
    const top = rows.slice(0,8);
    const labels = top.map(([k])=> LABEL_MAP[k] || k);
    const vals = top.map(([,v])=> v);

    // symmetric axis
    const maxAbs = Math.max(0.1, ...vals.map(v=>Math.abs(v)));
    const xMin = -maxAbs*1.15, xMax=maxAbs*1.15;

    if(shapChart) shapChart.destroy();
    shapChart=new Chart(ctx,{
      type:'bar',
      data:{ labels, datasets:[{ label:'Contribution to risk (±)', data: vals }]},
      options:{
        responsive:true, maintainAspectRatio:false,
        indexAxis:'y',
        plugins:{ legend:{display:false} },
        scales:{ x:{ beginAtZero:true, min:xMin, max:xMax } }
      }
    });
  }
```

Figure 5.37.user profile- result visualization (chart.js)

Instance-level shap_values are normalized by the normalizeShap function. When absent, the UI falls back to using artifacts.shap_summary.

```
function normalizeShap(input){
  if(!input) return [];
  if(Array.isArray(input)){
    return input.map(x=>{
      if(Array.isArray(x)) return [String(x[0]), Number(x[1])||0];
      if(typeof x==='object' && x!==null) return [String(x.feature||x.name||'feature'), Number(x.value||x.contribution||0)];
      return null;
    }).filter(Boolean);
  }
  if(typeof input==='object'){
    return Object.entries(input).map(([k,v])=>[String(k), Number(v)||0]);
  }
  return [];
}
```

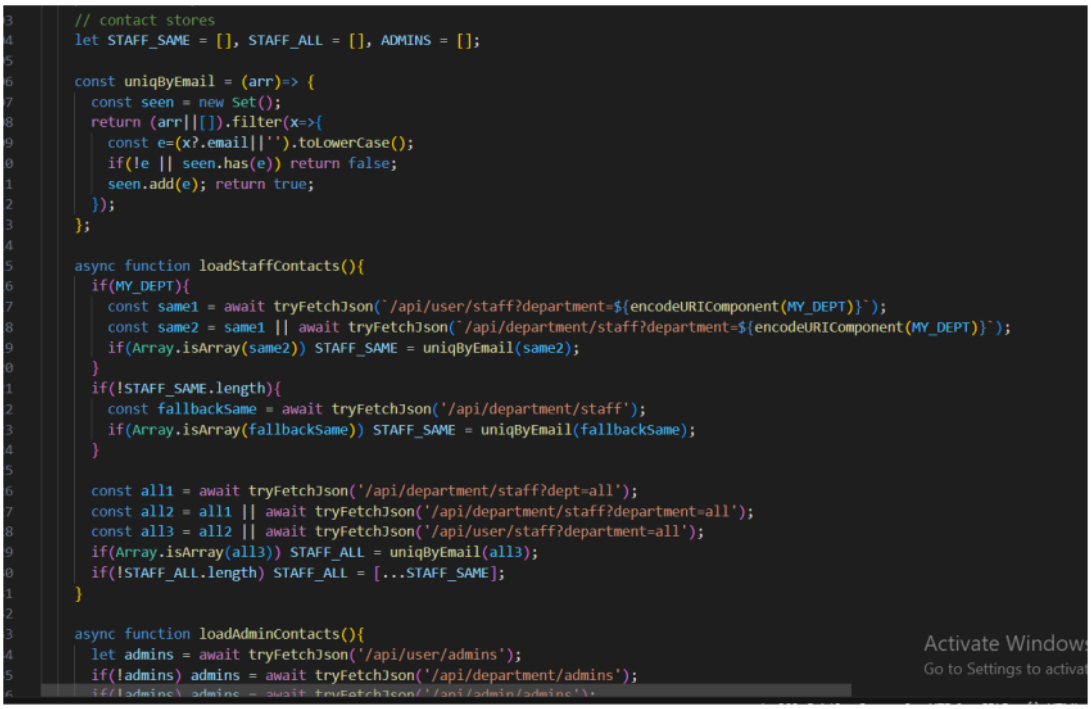
Figure 5.3. user profile -shap normalization

The buildProcess heuristic contrasts consecutive feature vectors, assigns direction weights to each dimension, ranks absolute impacts, and identifies the top changes. History uses DataTables for paging, ordering, and searching.

```
/*  History and progress  */
let dt=null;

async function loadHistory(){
  try{
    const r=await fetch('/api/user/predictions?limit=200',{credentials:'same-origin'});
    if(r.status===401){ location.href='/auth.html'; return; }
    if(!r.ok){ console.error('History load failed', r.status); return; }
    const payload = await r.json();
    const items = payload.items || [];
    const artifacts = payload.artifacts || {};
    setModelMeta(artifacts);

    // If no instance SHAP was drawn, render global summary
    if(!hasInstanceShap && artifacts?.shap_summary){
      renderShap(artifacts.shap_summary);
    }

    document.getElementById('emptyHistory').classList.toggle('d-none', items.length>0);

    const rows = items.map(x=>[
      new Date(x.created_at).toLocaleString(),
      `<span class="band ${(x.prediction||'').toLowerCase()}">${x.prediction||'—'}</span>`,
      nfmt(x.risk_score,1)+'/100',
      nfmt(x.features?.attendance,0),
      nfmt(x.features?.gpa,2),
      nfmt(x.features?.assignment_avg,0),
      nfmt(x.features?.quiz_avg,0),
      nfmt(x.features?.days_since_last_activity,0),
      x.features?.financial_status ?? ((x.features?.financial_unstable)?'Unstable':'Stable'),
```

Figure 5.38.user profile- load history

The mailbox subsystem enforces privacy with filterForPrivacy (Inbox to=MY_EMAIL, Sent from=MY_EMAIL), supports reply/restore/delete, and bulk-sends with sendEmailBulk over Promise.allSettled.



```
// contact stores
let STAFF_SAME = [], STAFF_ALL = [], ADMINS = [];

const uniqByEmail = (arr)=> {
  const seen = new Set();
  return (arr||[]).filter(x=>{
    const e=(x?.email||'').toLowerCase();
    if(!e || seen.has(e)) return false;
    seen.add(e); return true;
  });
};

async function loadStaffContacts(){
  if(MY_DEPT){
    const same1 = await tryFetchJson(`/api/user/staff?department=${encodeURIComponent(MY_DEPT)}`);
    const same2 = same1 || await tryFetchJson(`/api/department/staff?department=${encodeURIComponent(MY_DEPT)}`);
    if(Array.isArray(same2)) STAFF_SAME = uniqByEmail(same2);
  }
  if(!STAFF_SAME.length){
    const fallbackSame = await tryFetchJson('/api/department/staff');
    if(Array.isArray(fallbackSame)) STAFF_SAME = uniqByEmail(fallbackSame);
  }

  const all1 = await tryFetchJson('/api/department/staff?dept=all');
  const all2 = all1 || await tryFetchJson('/api/department/staff?department=all');
  const all3 = all2 || await tryFetchJson('/api/user/staff?department=all');
  if(Array.isArray(all3)) STAFF_ALL = uniqByEmail(all3);
  if(!STAFF_ALL.length) STAFF_ALL = [...STAFF_SAME];
}

async function loadAdminContacts(){
  let admins = await tryFetchJson('/api/user/admins');
  if(!admins) admins = await tryFetchJson('/api/department/admins');
  if(!admins) admins = await tryFetchJson('/api/admin/admins');
```


```
  if(scope==='staff_same'){
    const list = (STAFF_SAME||[]).map(s=>({
      email:s.email, name:s.name||'(no name)', dept:(s.department?.name || s.department?.slug || '')
    })).filter(x=>x.email).sort((a,b)=>(a.name||'').localeCompare(b.name||''));
    list.forEach(x=>{ const o=document.createElement('option'); o.value=x.email; o.textContent=`${x.name} <${x.email}> — ${x.d
    to.size = 1;
    hint.textContent = list.length? 'Select a staff contact (your department).' : 'No staff contacts available.';
  }else if(scope==='admin'){
    const list = (ADMINS||[]).map(a=>({email:a.email, name:a.name||'Admin'})).filter(x=>x.email)
      .sort((a,b)=>(a.name||'').localeCompare(b.name||''));
    list.forEach(x=>{ const o=document.createElement('option'); o.value=x.email; o.textContent=`${x.name} <${x.email}>`; to.ap
    to.size = 1;
    hint.textContent = list.length? 'Select an admin.' : 'No admin contacts available.';
  }else if(scope==='staff_all'){
    const list = (STAFF_ALL||[]).map(s=>({
      email:s.email, name:s.name||'(no name)', dept:(s.department?.name || s.department?.slug || '')
    })).filter(x=>x.email).sort((a,b)=>(a.name||'').localeCompare(b.name||''));
    renderRecipientPanel(list, true);
    hint.textContent = `All staff loaded (${list.length}). Deselect any you don't need.`;
  }else if(scope==='admin_all'){
    const list = (ADMINS||[]).map(a=>({email:a.email, name:a.name||'Admin', isAdmin:true, dept:'Admin'})).filter(x=>x.email)
      .sort((a,b)=>(a.name||'').localeCompare(b.name||''));
    renderRecipientPanel(list, true);
    hint.textContent = `All admins loaded (${list.length}). Deselect any you don't need.`;
  }else{ // custom
    hint.textContent = 'Enter any email address.';
  }
```

```
async function actRestore(id){ const r=await fetch(userURL(`/api/emails/${id}/restore`), {method:'PUT', credentials:'same-orig
async function actRead(id){ const r=await fetch(userURL(`/api/emails/${id}/read`), {method:'PUT', credentials:'same-origin'});

const STATE={folder:'inbox', list:[], selected:null};
const SENT={list:[], selected:null};

function listRow(m, folder){
  const who = (folder==='sent' ? getAddr(m,'to') : getAddr(m,'from'));
  const created = m.created_at ? new Date(m.created_at).toLocaleTimeString([], {hour:'2-digit',minute:'2-digit'}) : '';
  return `<div class="mail-item" data-id="${m.id}">
    <div>
      <div class="who">${esc(who||'(unknown)')}</div>
      <div class="sub">${esc(m.subject||'(no subject)')}</div>
    </div>
    <div class="meta"><span>${created}</span><span class="dot ${m.read?'read':''}"></span></div>
  </div>`;
}

function filterForPrivacy(items, box){
  return (items||[]).filter(m=>{
    const to = getAddr(m,'to').toLowerCase();
    const from = getAddr(m,'from').toLowerCase();
    if(box==='sent') return from===MY_EMAIL;
    if(box==='inbox') return to===MY_EMAIL;
    return (to===MY_EMAIL || from===MY_EMAIL);
  });
}
```

Figure 5.39.user profile - mailbox setup

5.2.4 admin-departments.html

This page is the admin workspace for creating departments, onboarding staff, and running internal mail. A fixed header, left navigation, and wide content area maintain orientation and reduce cognitive load in operational dashboards (Dolatabadi et al., 2024; Rossi et al., 2025). Section switching utilizes showSection to toggle hidden sections, persist the last view in localStorage, and close the mobile aside via an accessible toggle. Department setup is handled by createDept, which POSTs JSON to /api/admin/departments to create the unit and the first staff account. listDepts renders inline edits and binds Save and Delete through PUT and DELETE. listStaff supports password reset and removal. Compose hydrates recipients with buildRecipientOptions, while sendEmail POSTs JSON to /api/emails. The Inbox, Sent, and Trash views each call loadMail, loadSent, and loadTrash, respectively. hydrateListIfNeeded fetches full items when summaries are incomplete. Polling every twenty seconds updates unread counters via updateInboxBadgeUI; keyboard shortcuts and enableResizers improve efficiency. Inline alerts, resilient fallbacks, and esc sanitization support trustworthy, recoverable interactions for education technology (Fu & Weng, 2024; Zhu et al., 2025). Including credentials with fetch, clear error surfacing, and audit-friendly behaviors reflect observability guidance for dependable systems (Bosch, 2020; Cândido et al., 2019; Foalem et al., 2025). Deployment remains lightweight on Flask and SQLite (Anand et al., 2022; Suraya & Sholeh, 2021).

```
</header>

<!-- Aside -->
<aside id="sidebar" aria-label="Sidebar">

  <div class="aside-title">Dept Management</div>
  <div class="nav" id="leftNav">
    <a href="#" data-target="sec-create">Create Departments</a>
    <a href="#" data-target="sec-depts">View Departments</a>
    <a href="#" data-target="sec-staff">View Staffs</a>
    <a href="#" data-target="sec-compose">Create New Mail</a>
    <a href="#" data-target="sec-inbox" id="navInbox">Inbox <span id="inboxBadge" class="notif-badge">0</span></a>
    <a href="#" data-target="sec-sent">Sent</a>
    <a href="#" data-target="sec-trash">Trash</a>
  </div>

  <div class="aside-foot">
    Signed in as <span id="whoami">loading…</span>
  </div>
</aside>
```

Figure 5.40.admin department -aside

```
/* ---------- Section switching ---------- */
const nav = byId('leftNav');
const sections = {
  'sec-create': byId('sec-create'),
  'sec-depts': byId('sec-depts'),
  'sec-staff': byId('sec-staff'),
  'sec-compose': byId('sec-compose'),
  'sec-inbox': byId('sec-inbox'),
  'sec-sent': byId('sec-sent'),
  'sec-trash': byId('sec-trash'),
};
function showSection(id){
  Object.entries(sections).forEach(([k, el]) => el.classList.toggle('hidden', k !== id));
  nav.querySelectorAll('a').forEach(a => a.classList.toggle('active', a.dataset.target === id));
  window.scrollTo({ top: 0, behavior: 'smooth' });
  if (document.body.classList.contains('aside-open')) { document.body.classList.remove('aside-open'); asideToggle?.setAttribute('a
  if (id === 'sec-inbox') { markInboxSeen(INBOX_LAST_MAX_ID); updateInboxBadgeUI(LAST_UNREAD_COUNT); }
  localStorage.setItem('admin_last_section', id);
}
nav.addEventListener('click', (e)=>{
  const a = e.target.closest('a[data-target]'); if (!a) return;
  e.preventDefault(); showSection(a.dataset.target);
});
byId('composeBtn')?.addEventListener('click', ()=>showSection('sec-compose'));
(function restoreLastSection(){ const last = localStorage.getItem('admin_last_section'); if(last && sections[last]) showSection(la
```

Figure 5.41.admin department - switching section

```
/* ---------- Departments and Staff ---------- */
const slugSel   = byId('slug');
const deptName  = byId('deptName');
const staffName = byId('staffName');
const emailInp  = byId('email');
const passInp   = byId('password');
const createMsg = byId('createMsg');

const tbody = byId('tbody'); const empty = byId('empty');
const staffBody = byId('staffBody'); const staffEmpty = byId('staffEmpty');

async function createDept() {
  const slug = (slugSel?.value || '').trim();
  const dName = (deptName?.value || '').trim();
  const sName = (staffName?.value || '').trim();
  const email = (emailInp?.value || '').trim();
  const password = (passInp?.value || '').trim();

  if (!slug) return (createMsg.textContent = 'Pick a slug');
  if (!email || !password) return (createMsg.textContent = 'Email & password required');

  createMsg.textContent = 'Creating…';

  const depRes = await fetch('/api/admin/departments', {
    method:'POST',
    headers:{'Content-Type':'application/json'},
    body: JSON.stringify({ slug, name: dName || slug, staff_name: sName || slug, staff_email: email, staff_password: password }),
    ...FETCH_OPTS
  });
```

```
async function listDepts() {
  tbody.innerHTML = `<tr><td colspan="5" class="muted">Loading…</td></tr>`;
  try {
    const res = await fetch('/api/admin/departments', { ...FETCH_OPTS });
    if (!res.ok) {
      tbody.innerHTML = `<tr><td colspan="5" class="muted">Failed to load (${res.status})</td></tr>`;
      empty.style.display = 'none'; return;
    }
    const rows = await res.json();
    if (!rows.length) { tbody.innerHTML = ''; empty.style.display = 'block'; return; }
    empty.style.display = 'none'; tbody.innerHTML = '';
    for (const d of rows) {
      const tr = document.createElement('tr'); tr.className = 'row';
      const safeName = esc(d.name || ''); const safeSlug = esc(d.slug || '');
      tr.innerHTML = `
        <td>#${d.id}</td>
        <td>
          <input class="mono slugEdit" data-id="${d.id}" value="${safeSlug}" />
          <div class="muted">Slug must be one of academic, finance, mental_health</div>
        </td>
        <td>
          <input class="nameEdit" data-id="${d.id}" value="${safeName}" />
          <div class="muted">Shown on staff UIs (e.g., “Academic”)</div>
        </td>
        <td><span class="pill">${d.staff_count ?? 0} staff</span></td>
        <td class="tools">
          <button class="btn btn-ok btn-sm" data-action="save" data-id="${d.id}">Save</button>
          <button class="btn btn-danger btn-sm" data-action="del" data-id="${d.id}">Delete</button>
        </td>`;
```

Figure 5.42.admin department -create and list dept - staff

```
    tbody.querySelectorAll('button[data-action="del"]').forEach(btn=>{
      btn.addEventListener('click', async ()=>{
        const id = btn.dataset.id;
        if (!confirm('Delete this department? This will cascade linked staff.')) return;
        btn.disabled = true;
        const res = await fetch(`/api/admin/departments/${id}`, { method:'DELETE', ...FETCH_OPTS });
        btn.disabled = false;
        if (!res.ok) return alert('Delete failed');
        await listDepts(); await listStaff();
      });
    });
  }
```

```
async function listStaff() {
  staffBody.innerHTML = `<tr><td colspan="5" class="muted">Loading…</td></tr>`;
  try {
    const res = await fetch('/api/admin/staff', { ...FETCH_OPTS });
    if (!res.ok) {
      staffBody.innerHTML = `<tr><td colspan="5" class="muted">Failed to load (${res.status})</td></tr>`;
      staffEmpty.style.display = 'none'; return;
    }
    const rows = await res.json();
    if (!rows.length) { staffBody.innerHTML = ''; staffEmpty.style.display = 'block'; return; }
    staffEmpty.style.display = 'none'; staffBody.innerHTML = '';
    for (const s of rows) {
      const depLabel = s.department && (s.department.name || s.department.slug) || '';
      const tr = document.createElement('tr'); tr.className = 'row';
      tr.innerHTML = `
        <td>#${s.id}</td>
        <td>${esc(s.name || '')}</td>
        <td class="mono">${esc(s.email || '')}</td>
        <td><span class="pill">${esc(depLabel)}</span></td>
        <td class="tools">
          <input class="mono" style="width:200px" placeholder="new password" data-id="${s.id}" />
          <button class="btn btn-ghost btn-sm" data-action="reset" data-id="${s.id}">Reset Password</button>
          <button class="btn btn-danger btn-sm" data-action="remove" data-id="${s.id}">Remove</button>
        </td>`;
      staffBody.appendChild(tr);
    }
    wireStaffButtons();
  } catch {
    staffBody.innerHTML = `<tr><td colspan="5" class="muted">Error loading</td></tr>`;
  }
}
```


Figure 5.43.admin department -create and list dept - staff

```
762
763  /* ---------- Email (compose/list) ---------- */
764  let USERS=[], STAFF=[], ADMINS=[];
765  function prettifyDept(d){ return d ? (d.name || d.slug || '') : ''; }
766
767  async function loadUsersForCompose(){
768    try{
769      const r=await fetch('/api/admin/users', { ...FETCH_OPTS });
770      USERS = r.ok ? await r.json() : [];
771      if($('#m_scope')?.value==='users') buildRecipientOptions();
772    }catch{ USERS=[]; }
773  }
774  async function loadStaffForCompose(){
775    try{
776      const r=await fetch('/api/admin/staff', { ...FETCH_OPTS });
777      STAFF = r.ok ? await r.json() : [];
778      if($('#m_scope')?.value==='staff') buildRecipientOptions();
779    }catch{ STAFF=[]; }
780  }
781  async function loadAdminsForCompose(){
782    try{
783      const r=await fetch('/api/admin/me', { ...FETCH_OPTS });
784      if(!r.ok){ ADMINS=[]; return; }
785      const me=await r.json();
786      ADMINS = [{name: me.name||'Admin', email: me.email}];
787      if($('#m_scope')?.value==='admin') buildRecipientOptions();
788    }catch{ ADMINS=[]; }
789  }
790
791  function buildRecipientOptions(){
792    const scope=$('#m_scope')?.value, to=$('#m_to'), hint=$('#m_hint');
```

```
847  /* ---------- Inbox / Sent / Trash ---------- */
848  async function actDelete(id){ const r=await fetch(`/api/emails/${id}`, {method:'DELETE', ...FETCH_OPTS}); let d=null; try{ d=await
849  async function actRestore(id){ const r=await fetch(`/api/emails/${id}/restore`, {method:'PUT', ...FETCH_OPTS}); let d=null; try{ d
850  async function actRead(id){ const r=await fetch(`/api/emails/${id}/read`, {method:'PUT', ...FETCH_OPTS}); if(!r.ok){ try{ const d=
851
852  const STATE={folder:'inbox', list:[], selected:null};
853  let INBOX_LAST_MAX_ID = 0;
854  let LAST_UNREAD_COUNT = 0;
855  let POLL_TIMER = null;
856  function badgeEl(){ return byId('inboxBadge'); }
857  function inboxLink(){ return byId('navInbox'); }
858  function lastSeenKey(){ return `inbox_last_seen_${(ME?.email||'').toLowerCase()}`; }
859  function getLastSeenId(){ return Number(localStorage.getItem(lastSeenKey())||0); }
860  function setLastSeenId(id){ localStorage.setItem(lastSeenKey(), String(id||0)); }
861  function computeUnreadAndMax(list){
862    let unread=0, maxId=0; for(const m of list){ if(!m.read) unread++; if(Number(m.id)>maxId) maxId = Number(m.id); } return { unrea
863  }
864  function updateInboxBadgeUI(unread){
865    LAST_UNREAD_COUNT = unread;
866    const b = badgeEl(); const link = inboxLink();
867    if (unread>0){ b.style.display='inline-block'; b.textContent = String(unread); }
868    else { b.style.display='none'; b.textContent = '0'; }
869    if (INBOX_LAST_MAX_ID > getLastSeenId() && unread>0) link.classList.add('notify-pulse');
870    else link.classList.remove('notify-pulse');
871  }
872  function markInboxSeen(latestId){ if(latestId) setLastSeenId(latestId); inboxLink().classList.remove('notify-pulse'); }
873  async function pollInbox(){
874    try{
875      const r = await fetch('/api/emails?box=inbox&limit=200', { ...FETCH_OPTS });
876      if(!r.ok) return;
```

Figure 5.44.admin dept email handling

Chapter 6 Results and Discussion

6.1.1 Model Timestamp

Table 6. 1 Results - System model training timestamp

Field	Value
Timestamp	2025-09-30T21:18:19.217290+00:00
Model	RandomForestClassifier (calibrated)
Oversampling	Not available (install imblearn)

This figures indicates a build completed on September 30, 2025, with a Random Forest Classifier that uses isotonic calibration and no oversampling.

6.1.2 Calibration summary- model_artifacts.json

Table 6. 2 Result -Calibration summary

Item	Value
Chosen calibrator	isotonic
Validation ECE	0.0408
Validation AUPRC	0.8523
Nested ECE (mean±std)	0.051 ± 0.005

These values indicate that predicted risks align well with observed frequencies, which is essential for banded triage and risk communication in student support settings (Silva Filho et al., 2023).

6.1.3 Nested cross-validation (mean ± std)- model_artifacts.json

Table 6. 3 Results -Nested Cross-validation

Metric	Mean ± Std
Accuracy	0.843 ± 0.028
Precision (weighted)	0.843 ± 0.033
Recall (weighted)	0.843 ± 0.028
Macro-F1	0.699 ± 0.042
Balanced accuracy	0.678 ± 0.032
AUROC (macro)	0.930 ± 0.019
AUPRC (macro)	0.781 ± 0.067
Brier score	0.074 ± 0.011
Expected Calibration Error (ECE)	0.051 ± 0.005

Accuracy, weighted precision, and weighted recall each average about 0.843, macro F1 averages about 0.699, balanced accuracy about 0.678, macro AUROC about 0.930, macro AUPRC about 0.781, and Brier about 0.074. Reading ROC together with thresholder metrics is appropriate for imbalanced evaluation and is consistent with recent recommendations for assessing models under class skew (Richardson et al., 2024). The nested design limits optimistic selection effects and supports reproducibility, while careful control of feature leakage remains essential during splitting and engineering (Wainer & Cawley, 2021; Kapoor & Narayanan, 2023).

6.1.4 Locked operating points - model_artifacts.json

Table 6. 4 Results threshold

Rule	Threshold
High band P(High) ≥	**0.35**
Low band P(Low) ≥	**0.45**

Governance fixed the High Band at a probability of at least 0.35 and the Low band at a probability of at least 0.45. Choosing cut points that reflect the relative costs of misses and false alarms follows decision-analytic recommendations and can be aligned with aggregate behavior when governance wishes to mirror operational practice (Ferri et al., 2019; Patel et al., 2021).

6.1.5 Threshold curve maxima (per band)- model_artifacts. Json

Table 6. 5 Results - Threshold curves per band

Band	Threshold (argmax F1)	Precision	Recall	F1
High	0.35	0.5778	0.6500	**0.6118**
Low	0.45	0.9024	0.9428	**0.9221**

For the High band the F1 peak is near 0.612 at threshold 0.35 with precision near 0.578 and recall near 0.650. For the Low band the F1 peak is near 0.922 at threshold 0.45 with precision near 0.902 and recall near 0.943. Reading these curves helps administrators balance the cost of missing students at risk against the workload from additional alerts, which should be specified in deployment playbooks for education settings (Zhu et al., 2025).

6.1.6 Held-out test performance- model_artifacts. Json

Table 6. 6 Results - model performance

Metric	Value
Accuracy	0.830
Precision (weighted)	0.834
Recall (weighted)	0.830
Macro-F1	0.758
Balanced accuracy	0.731
AUROC (macro)	0.934
AUPRC (macro)	0.796
Brier score	0.082
ECE	0.048

The hold-out results closely track cross-validation, with macro AUROC at approximately 0.934, macro AUPRC at approximately 0.796, macro F1 at approximately 0.758, balanced accuracy at approximately 0.731, Brier score at approximately 0.082, and expected calibration error at approximately 0.048. This combination indicates a strong ranking and dependable probability quality, both of which are necessary for banding and downstream interventions in early warning systems that aim to reduce absenteeism and disengagement (Wu & Weiland, 2024).

6.1.7 Feature importance - model_artifacts. Json

Table 6. 7 Results - feature importance

Feature	Importance	Share (%)
gpa	0.2348	23.48
days_since_last_activity	0.2029	20.29
attendance	0.1425	14.25
financial_stable	0.1065	10.65
quiz_avg	0.1003	10.03
financial_unstable	0.0820	8.20
assignment_avg	0.0776	7.76
mental_health_score	0.0449	4.49
extracurricular_activity	0.0086	0.86

The highest contributions come from grade point average, days since last activity, and attendance, followed by financial stability indicators, quiz and assignment averages, then mental health score and extracurricular activity. Engagement gaps and attendance are established precursors of withdrawal in early warning research and practice, with sector guidance highlighting a 90% attendance benchmark for sustained engagement (Wu & Weiland, 2024; Liza, 2020). Socio-economic context is a recognized determinant of dropout risk, supporting the role of financial indicators in the feature set (Aina et al., 2022). Mental health has been associated with later withdrawal in adolescent cohorts, which justifies the inclusion of well-being indicators for screening (Andersen et al., 2021; Lindhardt et al., 2022). Population summaries link extracurricular participation with engagement, explaining the small but non-zero contribution observed here (Extracurricular participation and student engagement, n.d.).

6.1.8 Reliability Diagram

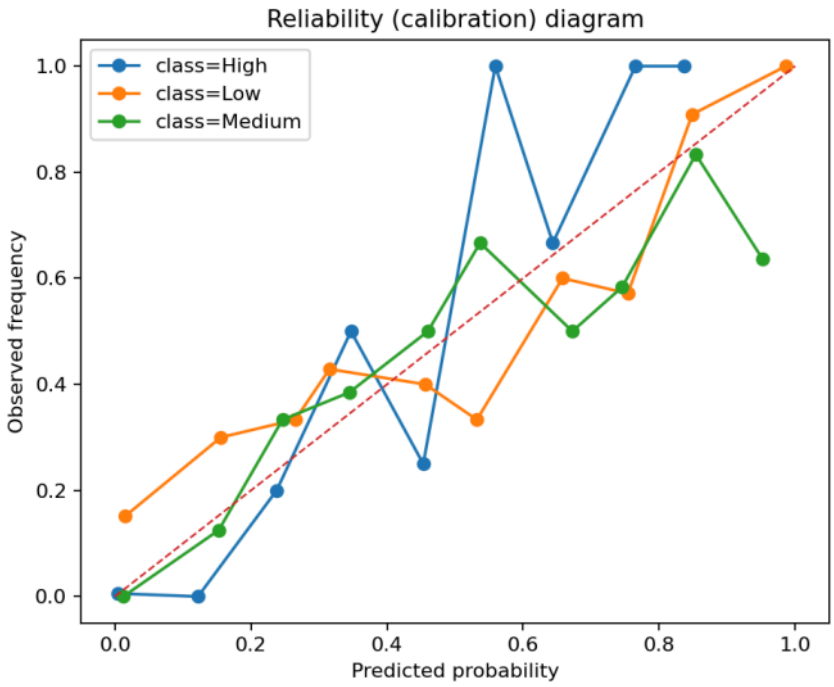


Figure 6. 1 Results - Reliability diagram

The class-conditional reliability diagram has shown a good level of model calibration with the High, Medium, and Low risk curves falling near the identity line, which indicates that the predicted frequencies are in parallel to the observed frequencies (Silva Filho et al., 2023). Small deviations of the Medium band are tolerated when using low bin samples and do not reflect systematic bias, which is in line with the low summary metrics. The high risk curve action justifies the operational level of 0.35 and suggests a low, high-recall level of 0.20 to provide sensitivity-based interventions. Conversely, the Low-risk curve follows a linear pattern and coincides with its best level at 0.45 (Ferri et al., 2019).

6.1.9 Admin Console -Dashboard

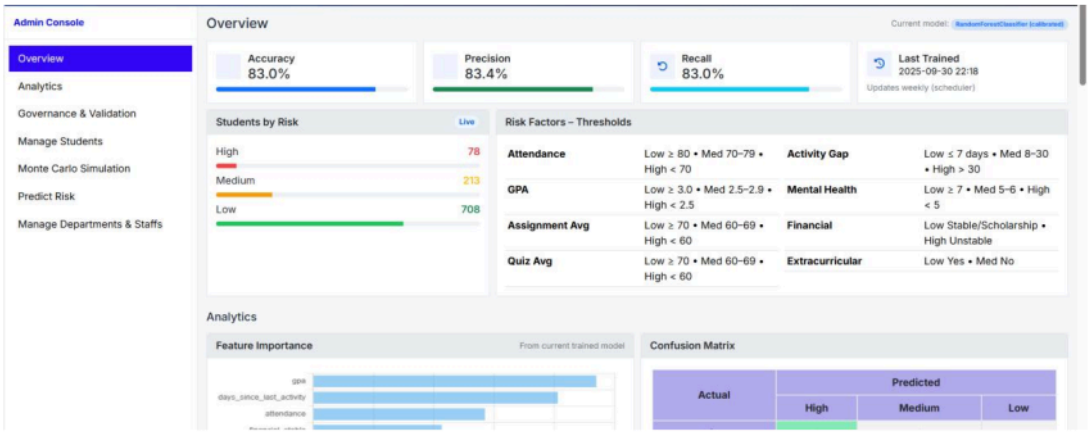


Figure 6. 2 Results - Admin Dashboard

The overview page summarizes live model status and system health. Headline metrics show accuracy, precision, and recall with the current calibrated RandomForest and last-trained time. Students by risk displays counts for High, Medium, and Low. Risk-factor thresholds are listed for quick reference.

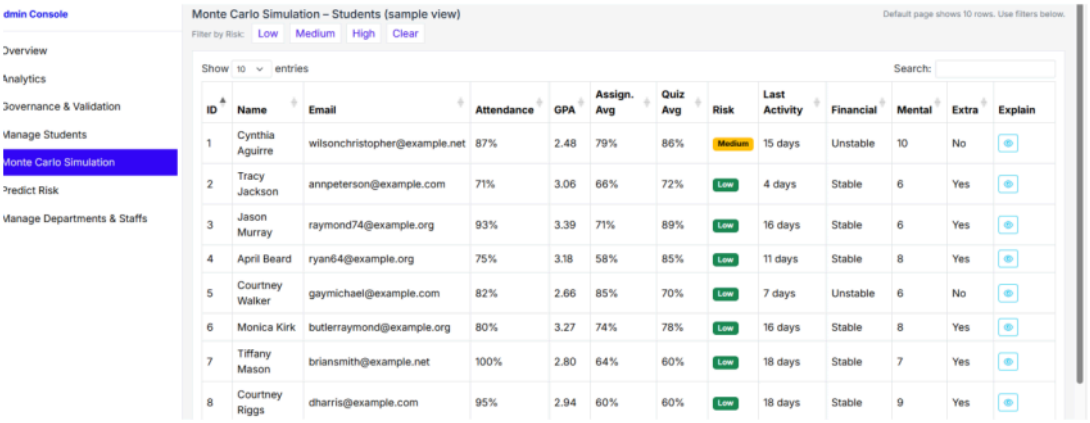


Figure 6. 3 Results - Admin dashboard (Monte Carlo simulated students table)

The monte carlo simulation view lists a sample cohort with interactive filters for Low, Medium, and High risk and an Explain button to open the per-student SHAP view.

6.1.10 Risk Prediction - Real-time

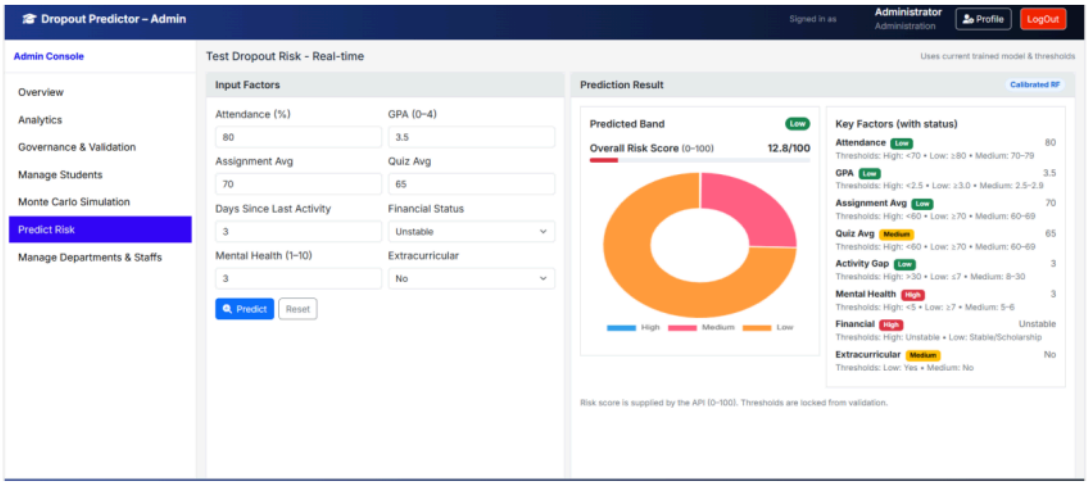


Figure 6. 4 Results- Admin Dashboard (real time prediction)

The panel shows a single-student prediction with a calibrated Random Forest. The result side reports the band and risk and a donut showing class probabilities. Key Factors compares each feature to governance-locked thresholds. The result also opens a SHAP view, revealing how each feature contributes to increasing or decreasing risk for this student, thereby supporting transparent casework (Da Conceicão Silva et al., 2025).

Overall, the model demonstrates outstanding discriminative ability on the held-out test set (AUROC = 0.934, AUPRC = 0.796), indicating a powerful capacity to rank students by risk (Marcolino et al., 2025; Vaarma & Li, 2024). More critically, its strong performance on metrics for imbalanced classifications macro F1 (0.758) and balanced accuracy (0.731) suggests usefulness beyond ranking to effective real-world identification of the at-risk minority (Richardson et al., 2024). The methodological rigour of nested cross-validation and the strong agreement between CV and test set metrics provide confidence the model is not overfitted and its performance is reproducible (Wainer & Cawley, 2021). Operationally, the model's high calibration, evidenced by a low Brier score (0.082) and ECE (0.048), is a standout result (Nikolić et al., 2025; Silva Filho et al., 2023). Feature importance analysis reveals learned patterns consistent with established educational research, with academic performance (GPA) and behavioral engagement (e.g., attendance) as dominant, actionable predictors (Liza, 2020; Wu & Weiland, 2024).The implementation of a web-based dashboard integrating real-time predictions, SHAP explanations, and reliability metrics is a significant step forward. Features like Monte Carlo simulations enhance this accessible, transparent tool for practitioners for stress-testing and structured audit logs, which are essential for ensuring long-term reliability, ethical accountability, and compliance (Anand et al., 2022).

6.2 Synthesis of Findings in Relation to Research Objectives

Random Forest model showed excellent results with a macro AUROC of 0.934 and a balanced accuracy of 0.731 on the held-out test set. Its strength was validated through nested cross-validation, and its effectively calibrated probabilities (ECE: 0.048) allow predicting the risks with high accuracy (Wainer and Cawley, 2021; Silva Filho et al., 2023), which fulfill Obj-1. The importance of the model elements in the world feature is that the most significant drivers were GPA (23.48%), and attendance (14.25%), which is consistent with the educational literature (Wu and Weiland, 2024). In the case of individual cases, a SHAP-based interface visually describes every prediction, which enables the staff to view the exact factors that increase the risk of a student, which supports Obj-2. The system deals with reliability of the models in a probabilistic manner. An analysis inspired by Monte Carlo sets the best decision levels (e.g., High-Risk at 0.35) and aids scheduled retraining, which guarantees long-term performance (Dega et al., 2023), which directly addresses Obj-3. Lastly, a web-based dashboard was adopted, operating on real-time grounds, which consolidated the prediction, SHAP explanations, and reliability measures on one screen to facilitate timely interventions. Developed using a lightweight Flask and SQLite stack, the system comprises structured logging and audit trails to maintain data privacy and ethical operation standards (Anand et al., 2022; Rossi et al., 2025), which was successfully met with Obj-4.

Chapter 7 Conclusion and Future Work

7.1 Conclusion

The project's significant component is a highly tuned Random Forest classifier, which achieves good performance on a Monte Carlo simulated educational dataset. The last model reached a test accuracy of 83 having a macro F1-score of 0.758. More importantly, to the unbalanced task of determining the at-risk students, it received an AUROC of 0.934 and an AUPRC of 0.796. These findings are in line with the literature on the effectiveness of tree-based models to work with tabular learning data (Villar and de Andrade, 2024; Richardson et al., 2024). The post-processing was performed in the form of isotonic regression minimizing the Expected Calibration Error (ECE) of the model to 0.048, with the risk scores of the model being reliably interpreted as confidence measures. This constitutes the key to facilitating interventions based on data as recent studies on the significance of calibration in educational metrics point to (Silva Filho et al., 2023; Pavlovic, 2025). Multi-level explainability is used to bridge the gap between the algorithmic output and human understanding in the system. The interventions are directly tied to predictions through a role conscious email console. This module enables specific communication processes, including the receipt of read messages and soft-deletion, while supporting responsible and privacy-conscious communication (Rossi et al., 2025; Anand et al., 2022). The system uses various strategic thresholds (e.g., a high-precision cut at 0.45 and a high-recall cut at 0.20) to give institutes workload-impact flexibility depending on the operational costs they incur, which is a crucial factor in the applied context (Ferri et al., 2019; Patel et al., 2021).

7.2 Future Work

The model was trained on a simulated dataset. Its external validity must be assessed across diverse educational contexts and its performance monitored longitudinally to account for concept drift and varying baseline attrition rates (Education at a Glance, 2023; Aina et al., 2022). The current implementation computes SHAP values per instance but does not persist a global summary for longitudinal trend analysis. Populating this global explanation is a key next step. Furthermore, moving from static to adaptive thresholding, where decision boundaries are periodically re-estimated from accumulated institutional feedback, would enhance the system's responsiveness and fairness (Patel et al., 2021). A significant improvement would be the formal implementation of Monte Carlo techniques for uncertainty quantification. Providing predictive intervals, as previously discussed, would represent a major step forward in communicating model confidence and supporting more nuanced intervention strategies. In conclusion, this thesis has demonstrated the viability and value of an end-to-end early-warning system that seamlessly integrates machine learning with human-centered design. By addressing the outlined future work, this platform can evolve into a more adaptive, transparent, and trustworthy tool for fostering student success.

References

Aina, C., Baici, E., Casalone, G., & Pastore, F. (2022). The determinants of university dropout: A review of the socio-economic literature. *Socio-Economic Planning Sciences, 79,* Article 101102. https://doi.org/10.1016/j.seps.2021.101102

Anand, V., Shivaram, A. M., & Karthikeyan, T. (2022). Volunteer data management using Flask and SQLite database. *International Advanced Research Journal in Science, Engineering and Technology.* https://iarjset.com/papers/volunteer-data-management-using-flask-and-sqlite-database/

Andersen, S., Davidsen, M., Nielsen, L., & Tolstrup, J. S. (2021). Mental health groups in high school students and later school dropout: a latent class and register-based follow-up analysis of the Danish National Youth Study. *BMC Psychology, 9*(1). https://doi.org/10.1186/s40359-021-00621-7

Blog, I. (n.d.). *Understanding model calibration - A gentle introduction and visual exploration of calibration and the expected calibration error (ECE) | ICLR Blogposts 2025.* https://iclr-blogposts.github.io/2025/blog/calibration/

Bosch, J., & Bosch, J. (2020). *Software logging for machine learning*. arXiv. https://doi.org/10.48550/arXiv.2001.10794

Bunkhumpornpat, C., Boonchieng, E., Chouvatut, V., & Lipsky, D. (2024). FLEX-SMOTE: Synthetic over-sampling technique that flexibly adjusts to different minority class distributions. *Patterns (New York, N.Y.), 5*(11), Article 101073. https://doi.org/10.1016/j.patter.2024.101073

Cândido, J., Aniche, M., & van Deursen, A. (2019). *Log-based software monitoring: A systematic mapping study*. arXiv. https://doi.org/10.48550/arXiv.1912.05878

Chart.js Contributors. (2024). *Chart.js* (Version 4.4.1) [JavaScript library]. https://www.chartjs.org/

Cheng, J., Yang, Z., Cao, J., Yang, Y., & Zheng, X. (2025, May 16). Predicting student dropout risk with a Dual-Modal Abrupt Behavioral Changes approach. *arXiv.org.* https://arxiv.org/abs/2505.11119

Contreras, P., Orellana-Alvear, J., Muñoz, P., Bendix, J., & Célleri, R. (2021). Influence of Random Forest Hyperparameterization on short-term runoff forecasting in an Andean mountain catchment. *Atmosphere, 12*(2), 238. https://doi.org/10.3390/atmos12020238

Da Conceicão Silva, F., Santana, A. M., & Feitosa, R. M. (2025). An investigation into dropout indicators in secondary technical education using explainable artificial intelligence. *IEEE Revista Iberoamericana De Tecnologias Del Aprendizaje, 1.* https://doi.org/10.1109/rita.2025.3566095

de Amorim, L. B. V., Cavalcanti, G. D. C., & Cruz, R. M. O. (2023). The choice of scaling technique matters for classification performance. *Applied Soft Computing, 133,* Article 109924. https://doi.org/10.1016/j.asoc.2022.109924

Dega, S., Dietrich, P., Schrön, M., & Paasche, H. (2023). Probabilistic prediction by means of the propagation of response variable uncertainty through a Monte Carlo approach in regression random forest:

Application to soil moisture regionalization. *Frontiers in Environmental Science, 11*. https://doi.org/10.3389/fenvs.2023.1009191

DataTables. (n.d.). *DataTables*. https://datatables.net

Dolatabadi, S. H., Gatial, E., Budinska, I., & Balogh, Z. (2024). Integrating Human-Computer Interaction Principles in User-Centered Dashboard Design: Insights from Maintenance Management. *2024 IEEE 28th International Conference on Intelligent Engineering Systems (INES)*, 000219–000224. https://doi.org/10.1109/INES63318.2024.10629098

Education at a Glance 2023. (2023). In *Education at a glance. OECD indicators/Education at a glance.* https://doi.org/10.1787/e13bef63-en

Esposito, C., Landrum, G. A., Schneider, N., Stiefl, N., & Riniker, S. (2021). GHOST: Adjusting the decision threshold to handle imbalanced data in machine learning. *Journal of Chemical Information and Modeling, 61*(6), 2623–2640. https://doi.org/10.1021/acs.jcim.1c00160

Extracurricular participation and student engagement. (n.d.). https://nces.ed.gov/pubs95/web/95741.asp

Ferri, C., Hernández-Orallo, J., & Flach, P. (2019). Setting decision thresholds when operating conditions are uncertain. *Data Mining and Knowledge Discovery, 33*(4), 805–847. https://doi.org/10.1007/s10618-019-00613-7

Foalem, L., Spahn, S., Kaminski, J., Hausen, D., Ngonga Ngomo, A.-C., & Arndt, N. (2025). *Logging requirement for continuous auditing of responsible machine learning-based applications*. arXiv. https://doi.org/10.48550/arXiv.2508.17851

Jain, A., Dubey, A. K., Khan, S., Panwar, A., Alkhatib, M., & Alshahrani, A. M. (2025). A PSO weighted ensemble framework with SMOTE balancing for student dropout prediction in smart education systems. *Scientific Reports, 15*(1), Article 17463. https://doi.org/10.1038/s41598-025-97506-1

Kapoor, S., & Narayanan, A. (2023). Leakage and the reproducibility crisis in machine-learning-based science. *Patterns (New York, N.Y.), 4*(9), Article 100804. https://doi.org/10.1016/j.patter.2023.100804

Kummaraka, U., & Srisuradetchai, P. (2025). Monte Carlo dropout neural networks for forecasting sinusoidal time series: performance evaluation and uncertainty quantification. *Applied Sciences, 15*(8), 4363. https://doi.org/10.3390/app15084363

LeCoz, A., Herbin, S., & Adjed, F. (2024). Confidence calibration of classifiers with many classes. *arXiv.org*.

Lindhardt, L., Lindholdt, L., Lund, T., & Mortensen, O. S. (2022). Self-reported mental health in adolescents attending school and its association with later school dropout: A prospective 2.5-year follow-up study. *Scandinavian Journal of Public Health, 50*(8), 1164–1171. https://doi.org/10.1177/14034948221089112

Liza. (2020, May 15). Ask Steph: Why is 90% attendance significant? | CT RISE Network. *CT RISE Network*. https://www.ctrise.org/blog/ask-steph-why-is-90-attendance-significant/

Marcolino, M. R., Porto, T. R., Primo, T. T., Targino, R., Ramos, V., Queiroga, E. M., Munoz, R., & Cechinel, C. (2025). Student dropout prediction through machine learning optimization: insights from Moodle log data. *Scientific Reports, 15*(1). https://doi.org/10.1038/s41598-025-93918-1

Megalooikonomou, K. G., & Beligiannis, G. N. (2023). Random Forests Machine Learning Applied to PEER Structural Performance Experimental Columns Database. *Applied Sciences*, *13*(23), 12821. https://doi.org/10.3390/app132312821

Nikolić, M., Nikolić, D., Stefanović, M., Koprivica, S., & Stefanović, D. (2025). Mitigating algorithmic bias through probability calibration: A case study on lead generation data. *Mathematics (Basel), 13*(13), 2183. https://doi.org/10.3390/math13132183

Patel, B. S., Steinberg, E., Pfohl, S. R., & Shah, N. H. (2021). Learning decision thresholds for risk stratification models from aggregate clinician behavior. *Journal of the American Medical Informatics Association, 28*(10), 2258–2264. https://doi.org/10.1093/jamia/ocab159

Pavlovic, M. (2025, January 31). Understanding model calibration -- A gentle introduction and visual exploration of calibration and the expected calibration error (ECE). *arXiv.org.* https://arxiv.org/abs/2501.19047

Putra, L. G. R., Prasetya, D. D., & Mayadi, M. (2025). Student dropout prediction using Random Forest and XGBoost method. *Intensif (Online), 9*(1), 147–157. https://doi.org/10.29407/intensif.v9i1.21191

Richardson, E., Trevizani, R., Greenbaum, J. A., Carter, H., Nielsen, M., & Peters, B. (2024). The receiver operating characteristic curve accurately assesses imbalanced datasets. *Patterns (New York, N.Y.), 5*(6), Article 100994. https://doi.org/10.1016/j.patter.2024.100994

Silva Filho, T., Song, H., Perello-Nieto, M., Santos-Rodriguez, R., Kull, M., & Flach, P. (2023). Classifier calibration: A survey on how to assess and improve predicted class probabilities. *Machine Learning, 112*(9), 3211–3260. https://doi.org/10.1007/s10994-023-06336-7

Silva, F. da C., Santana, A. M., & Feitosa, R. M. (2025). An Investigation Into Dropout Indicators in Secondary Technical Education Using Explainable Artificial Intelligence. *IEEE-RITA*, *20*, 105–114. https://doi.org/10.1109/RITA.2025.3566095

Rossi, F. S., Adams, M. C. B., Aarons, G., & McGovern, M. P. (2025). From glitter to gold: recommendations for effective dashboards from design through sustainment. *Implementation Science : IS*, *20*(1), Article 16. https://doi.org/10.1186/s13012-025-01430-x

Suraya, S., & Sholeh, M. (2021). Designing and implementing a database for thesis data management by using the Python Flask framework. *International Journal of Engineering, Science and Information Technology, 2*(1), 9–14. https://doi.org/10.52088/ijesty.v2i1.197

UCI Machine Learning Repository. (n.d.). https://archive.ics.uci.edu/dataset/697/predict%2Bstudents%2Bdropout%2Band%2Bacademic%2Bsuccess

Vaarma, M., & Li, H. (2024). Predicting student dropouts with machine learning: An empirical study in Finnish higher education. *Technology in Society, 76,* Article 102474. https://doi.org/10.1016/j.techsoc.2024.102474

Villar, A., & de Andrade, C. R. V. (2024). Supervised machine learning algorithms for predicting student dropout and academic success: A comparative study. *Discover Artificial Intelligence, 4*(1), Article 2. https://doi.org/10.1007/s44163-023-00079-z

Wainer, J., & Cawley, G. (2021). Nested cross-validation when selecting classifiers is overzealous for most practical applications. *Expert Systems with Applications, 182,* Article 115222. https://doi.org/10.1016/j.eswa.2021.115222

Zhu, H., Sun, Y., & Yang, J. (2025). Towards responsible artificial intelligence in education: A systematic review on identifying and mitigating ethical risks. *Humanities & Social Sciences Communications, 12*(1), Article 1111. https://doi.org/10.1057/s41599-025-05252-6

Wu, T., & Weiland, C. (2024). Leveraging modern machine learning to improve early warning systems and reduce chronic absenteeism in early childhood. In Annenberg Institute at Brown University, *EdWorkingPaper* [Report]. https://edworkingpapers.com/sites/default/files/ai24-1081.pdf

Appendix A (User Interface)

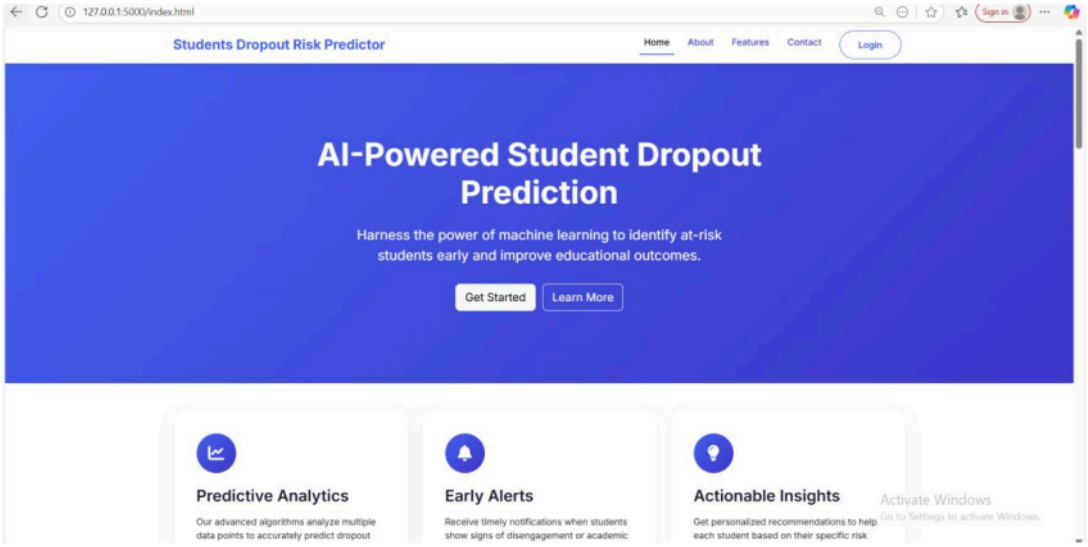


Figure 4. 3 Home page

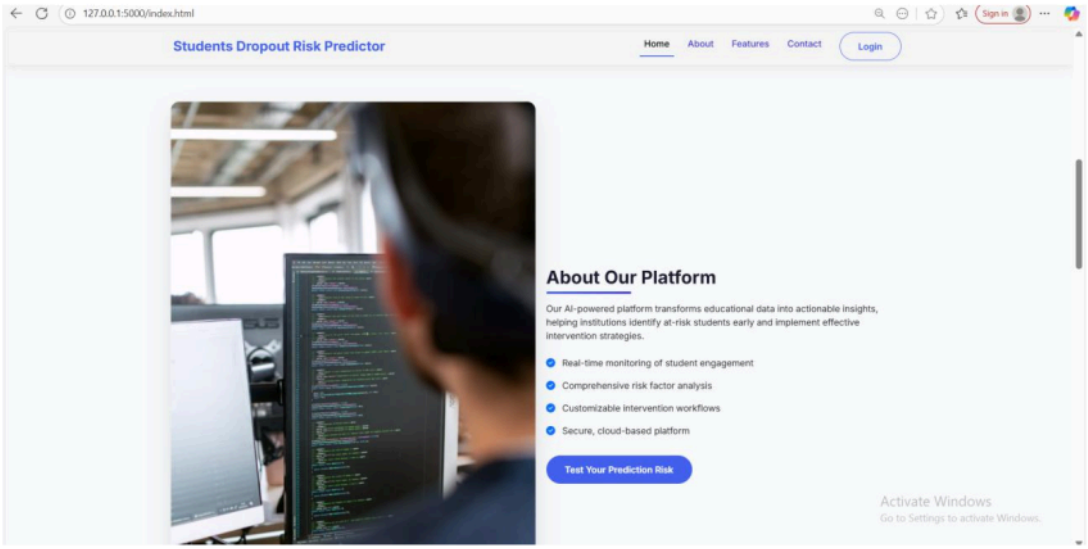


Figure 4. 4 Home page about

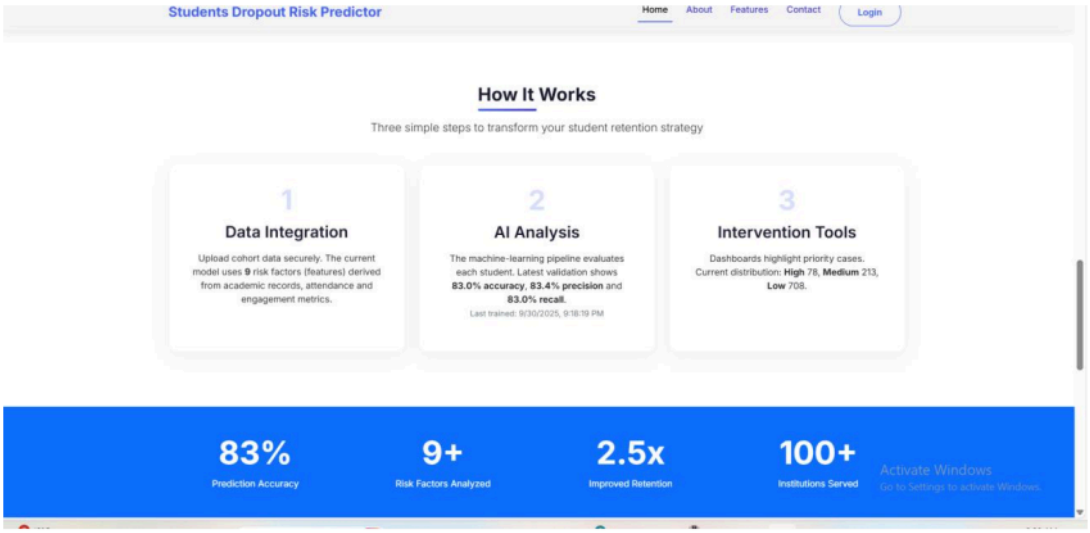


Figure 4. 5 Home Page how it works

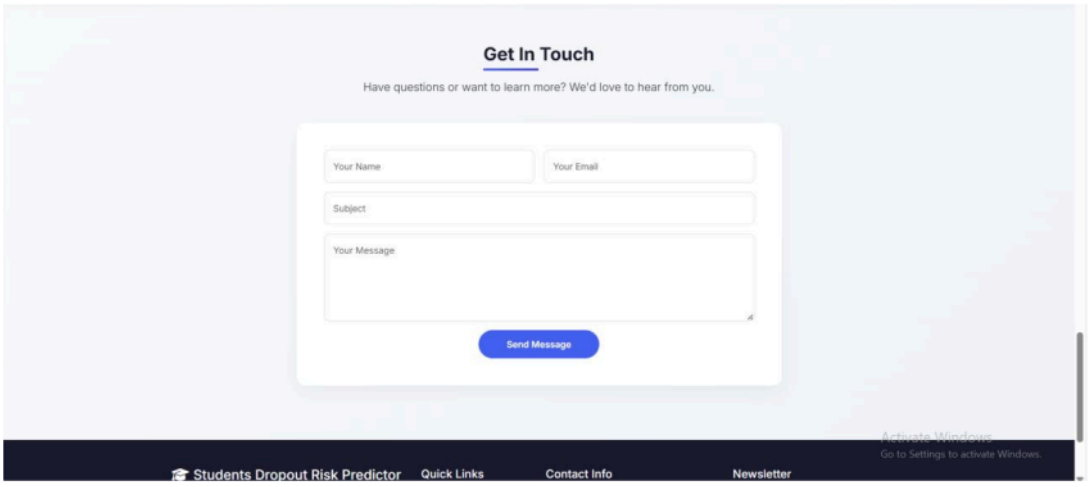


Figure 4. 6 Home Page get it touch

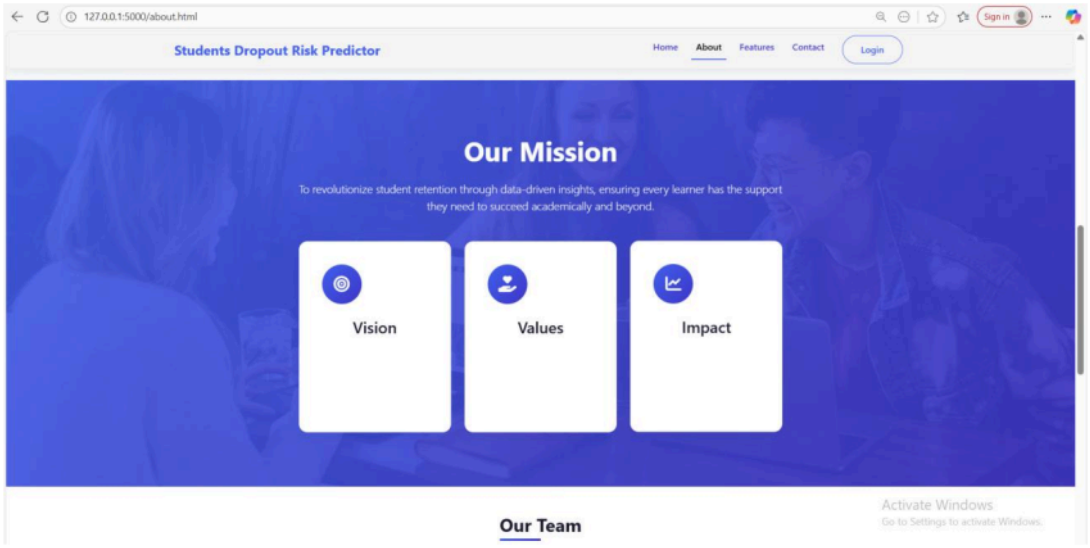


Figure 4. 7 About Page

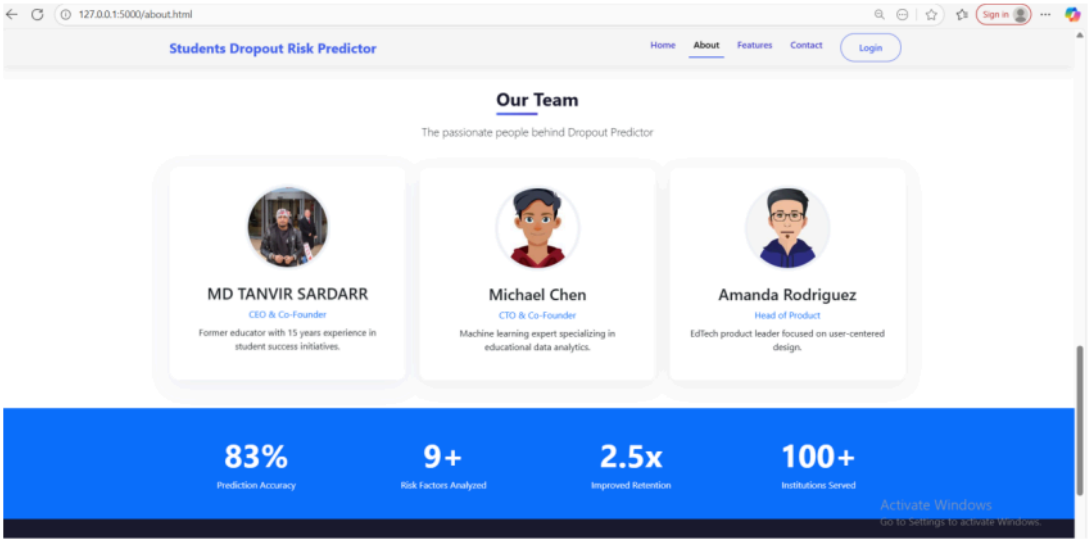


Figure 4. 8 About page team members

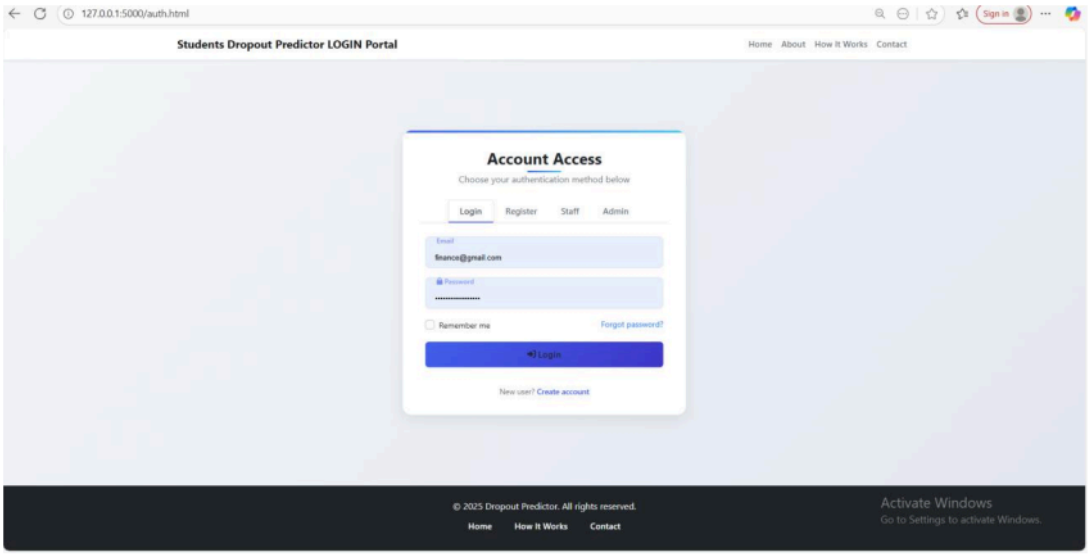


Figure 4. 9 Login Page

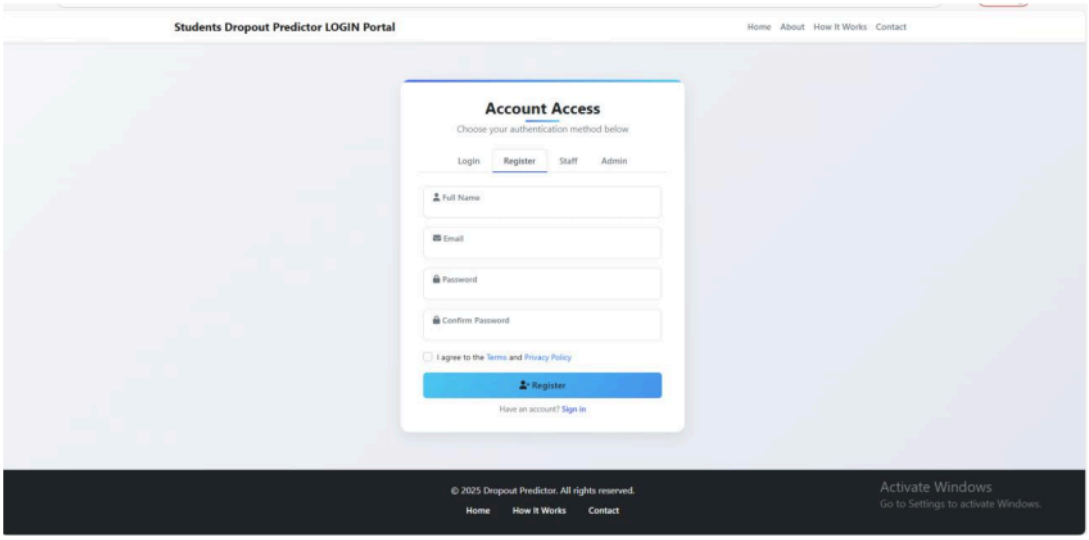


Figure 4. 10 Login page (register account)

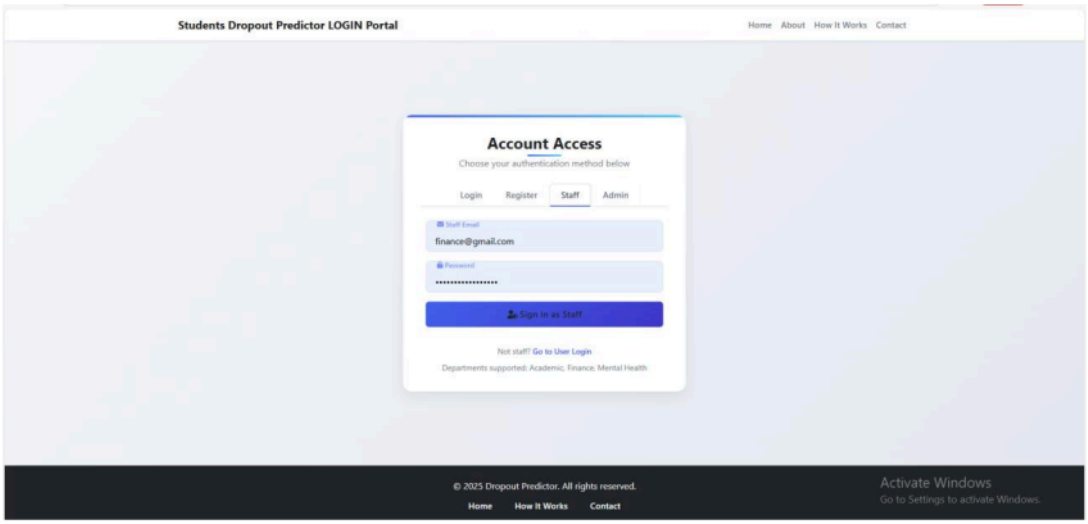


Figure 4. 11 Login Page (Staff Login)

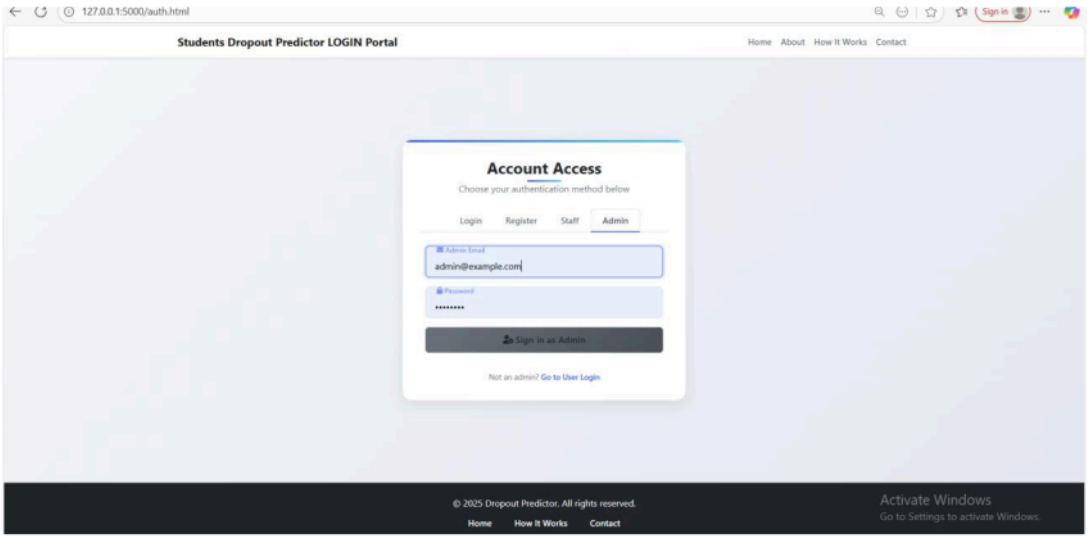


Figure 4. 12 Login Page (Admin Login)

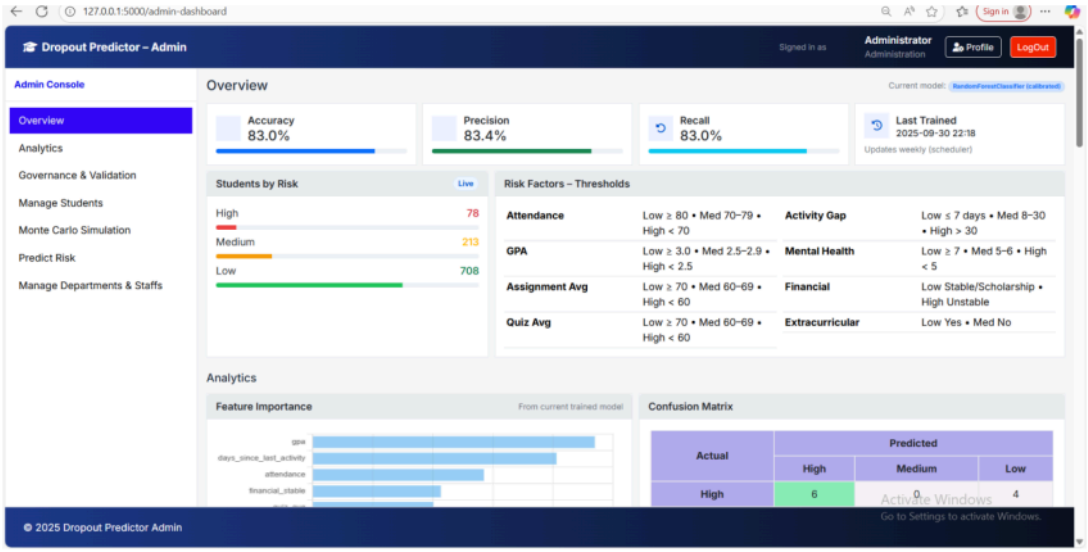


Figure 4. 13 Admin Dashboard (Overview)

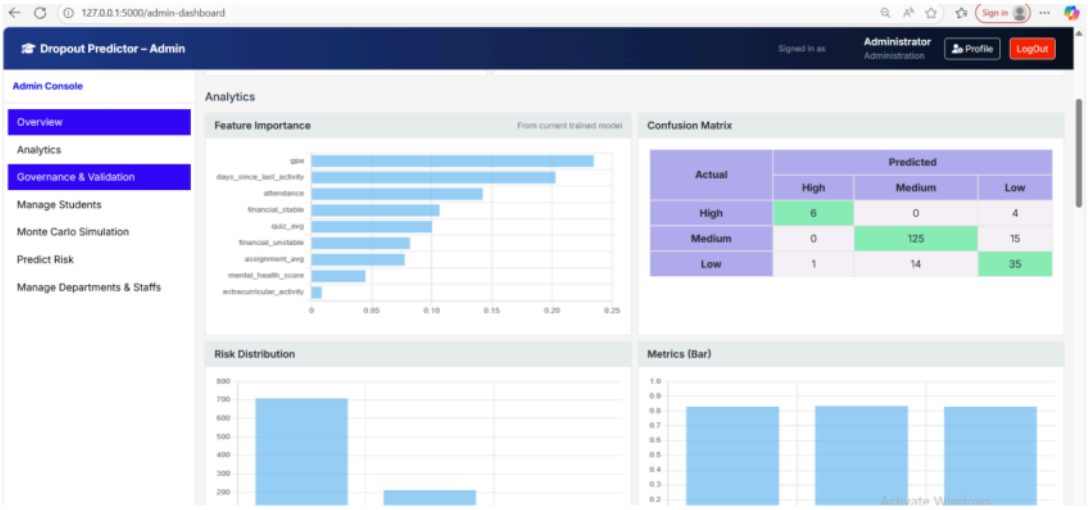


Figure 4. 14 Admin Dashboard (Analytics)

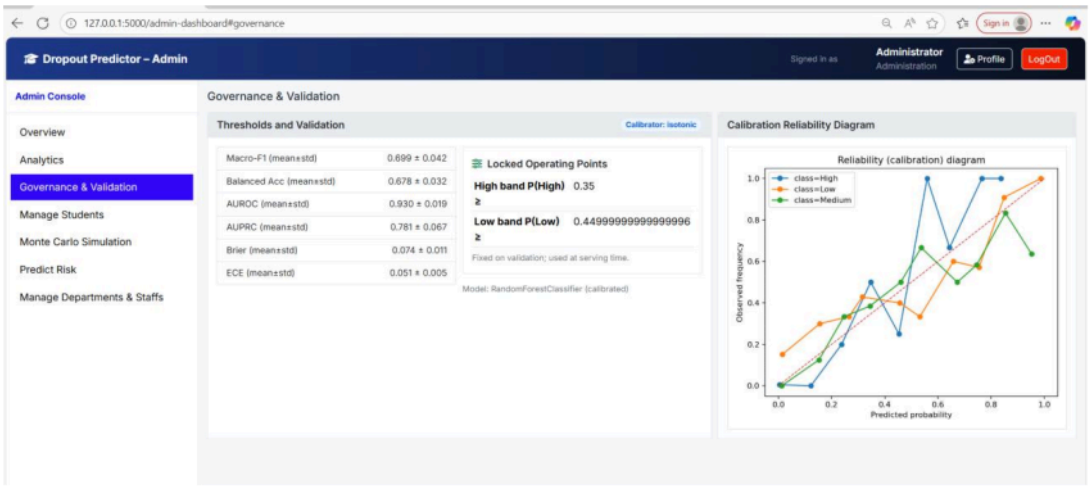


Figure 4. 15 Admin Dashboard (Governance and Validation)

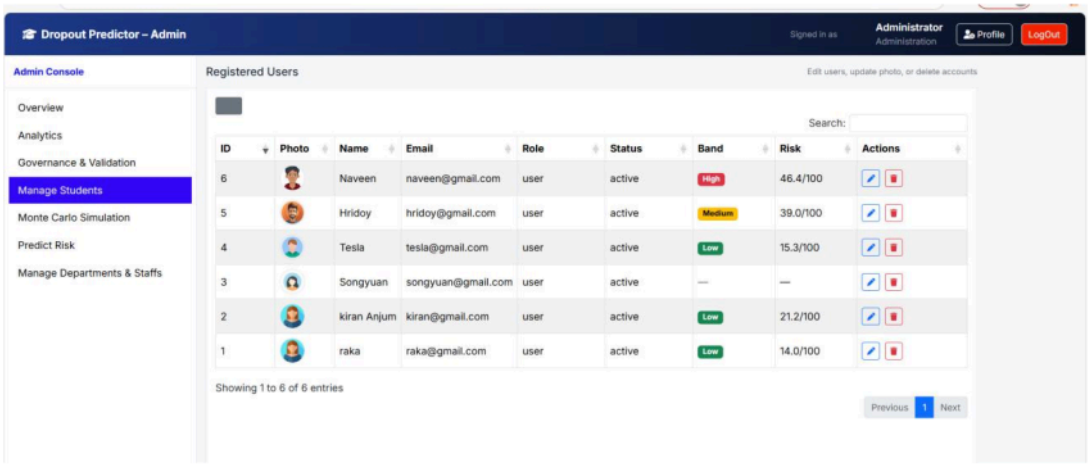


Figure 4. 16 Admin Dashboard (Manage Students/View users)

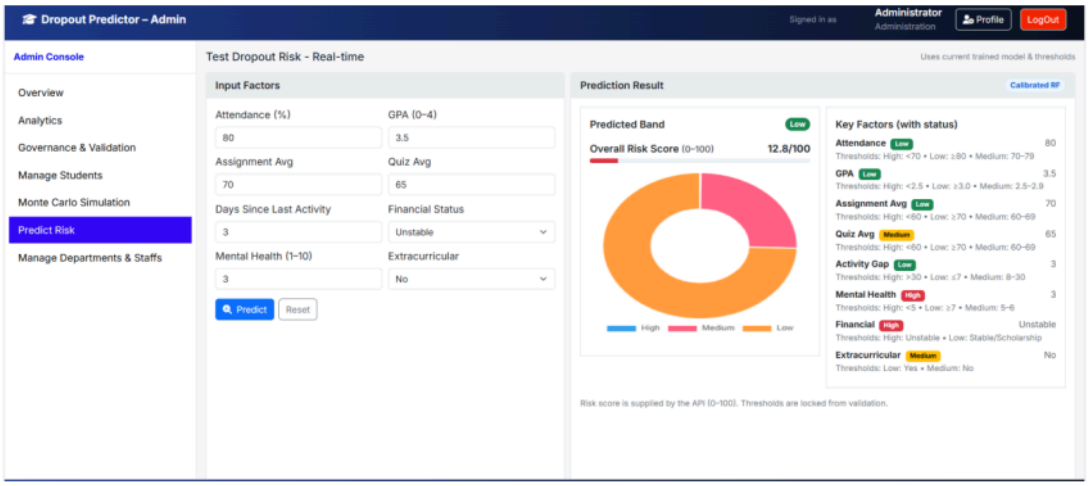


Figure 4. 17 Admin Dashboard (Predict Risk)

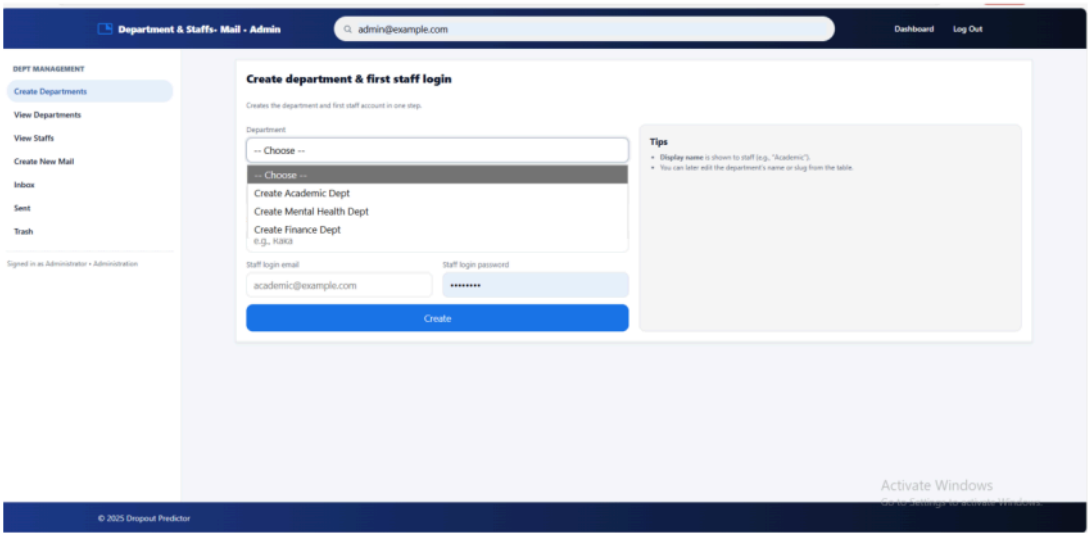


Figure 4. 18 Admin (Create Dept)

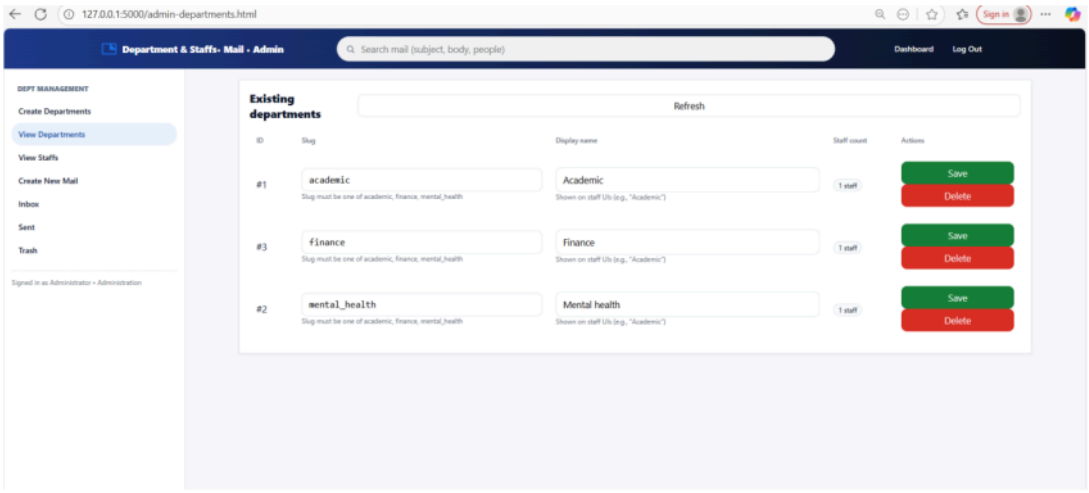


Figure 4. 19 Admin View Departments

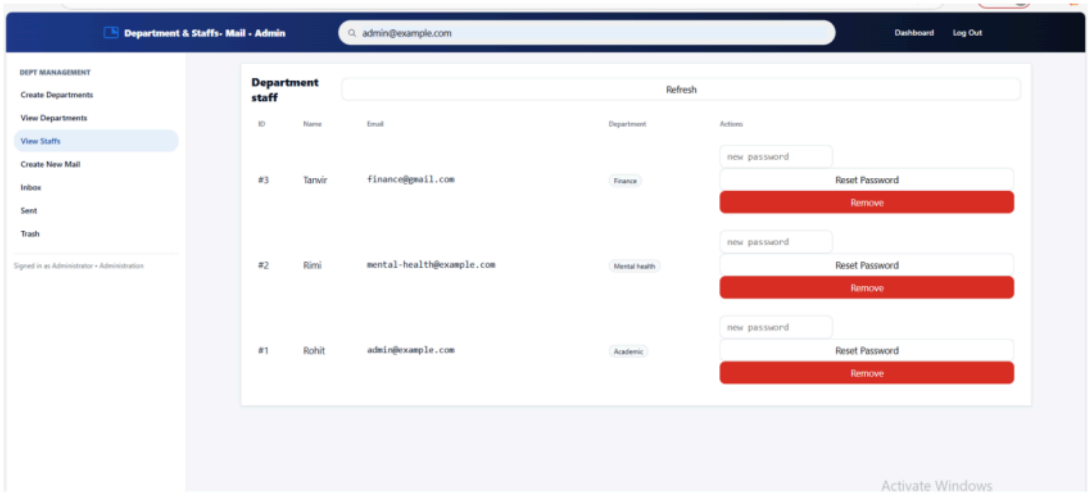


Figure 4. 20 Admin View Staff

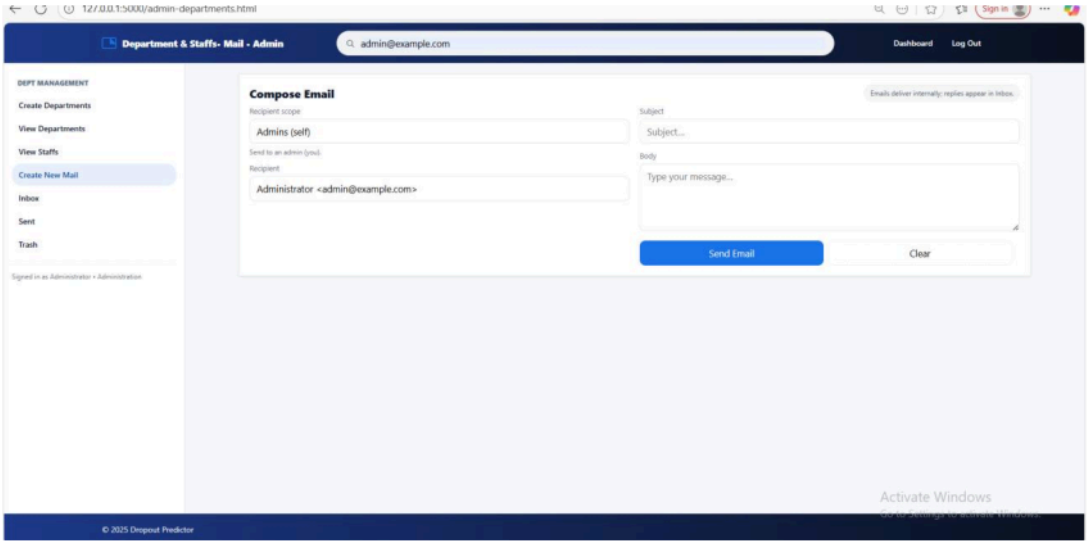


Figure 4. 21 Admin Create New mail

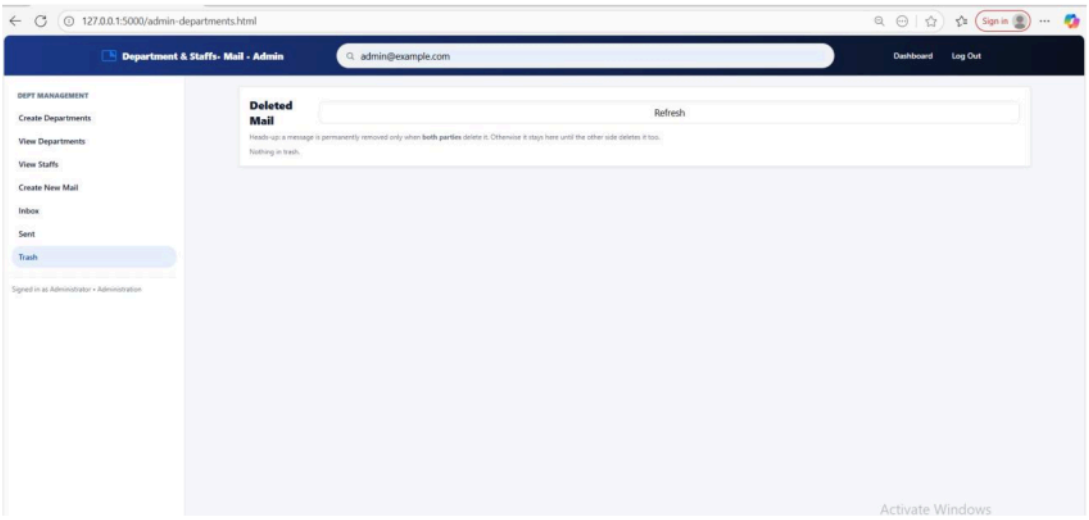


Figure 4. 22 Admin inbox, Sent and Trash mail

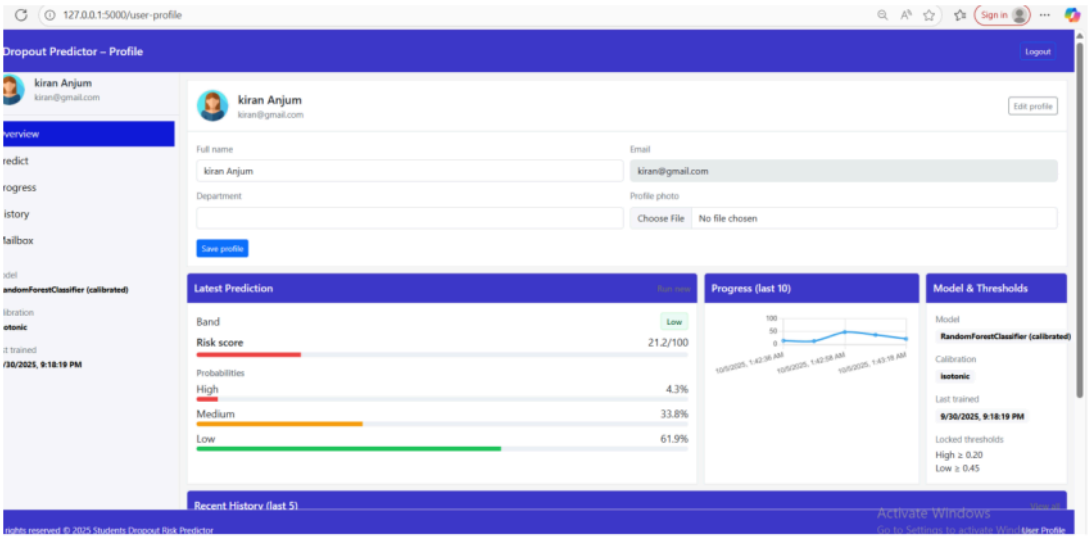


Figure 4. 23 User Profile Overview

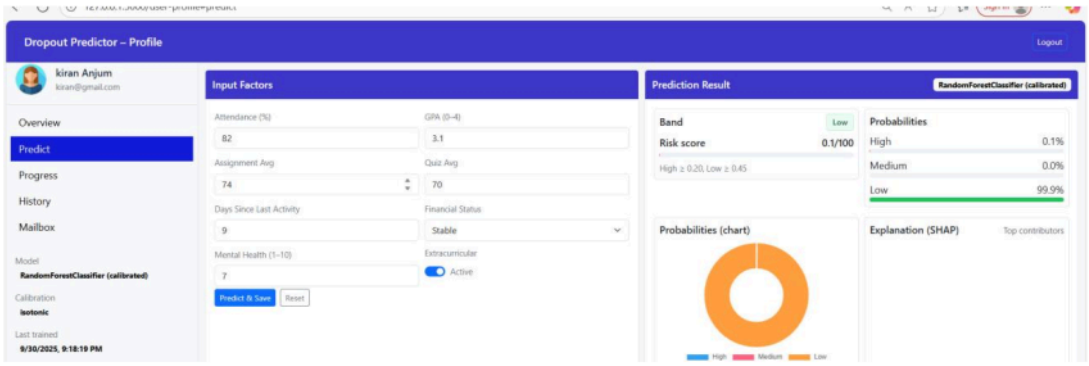


Figure 4. 24 User Profile predict risk

Figure 4. 25 User Profile Risk Progress chart

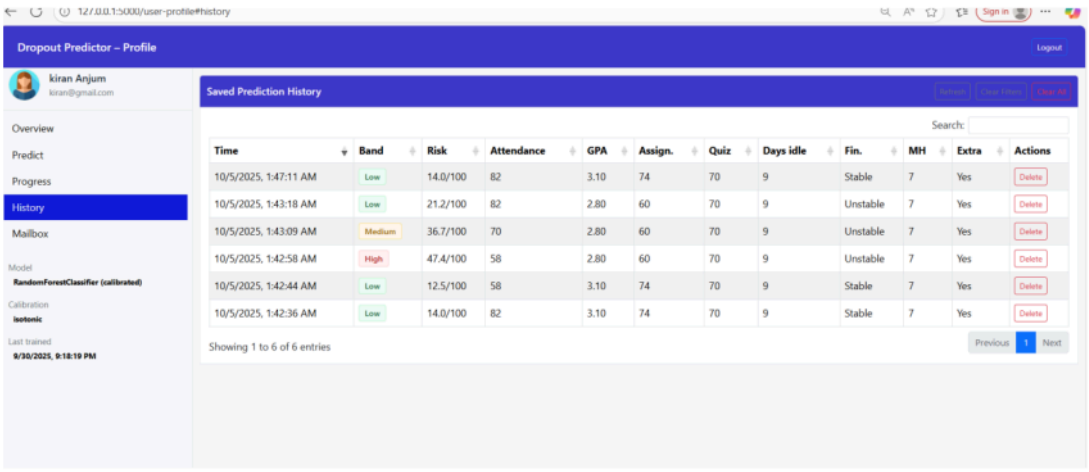


Figure 4. 26 User Profile Prediction history

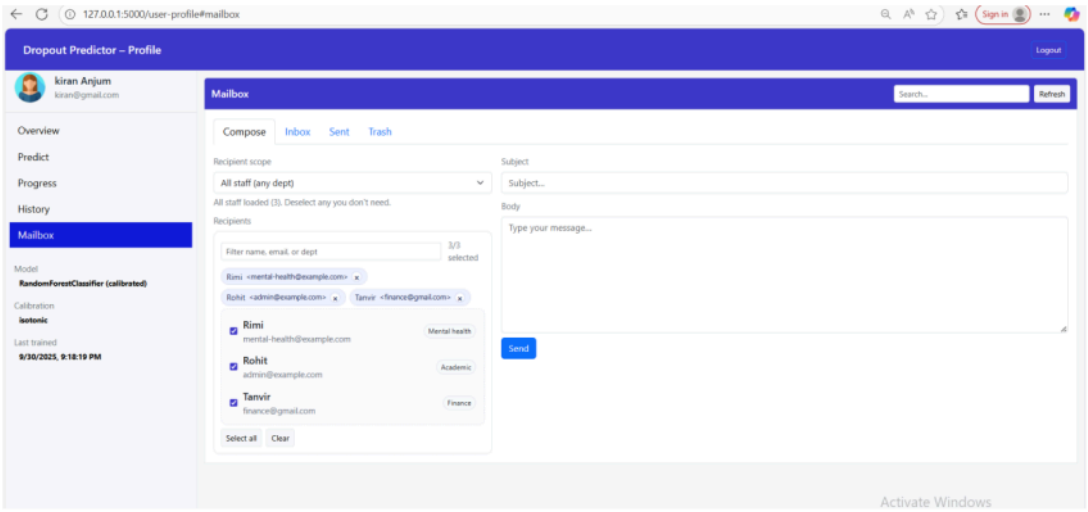


Figure 4. 27 user Profile Mailbox (Create New mail)

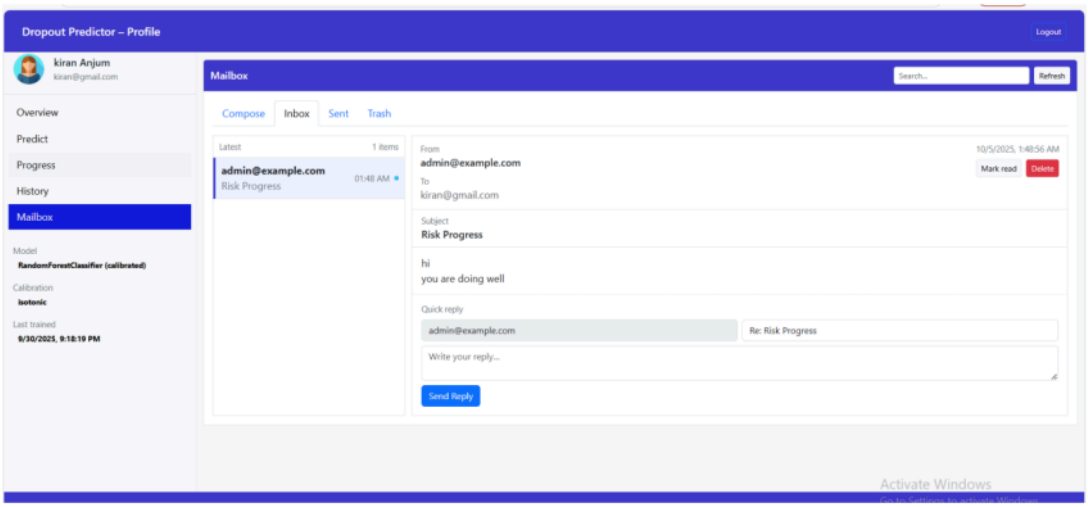


Figure 4. 28 User Profile mailbox (Inbox)

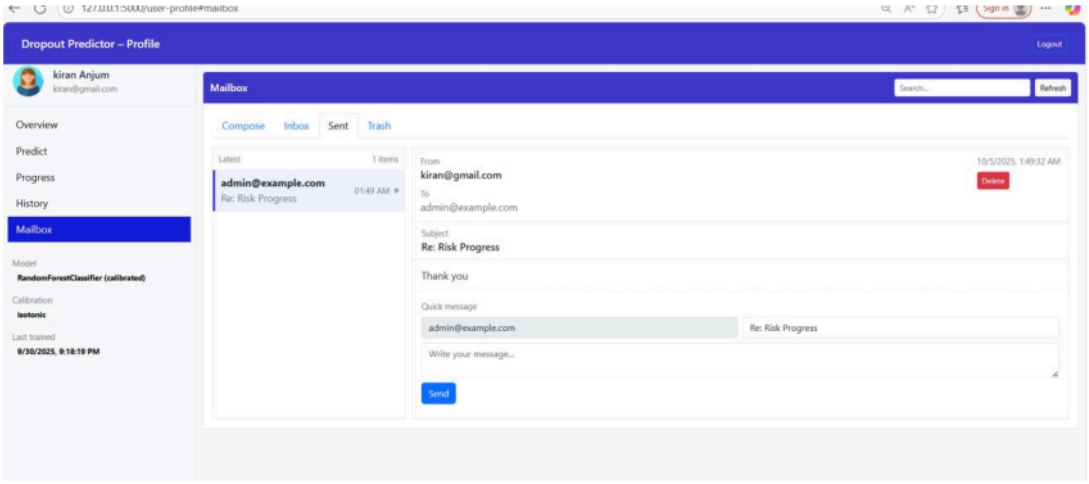


Figure 4. 29 User Profile Mailbox (Sent)

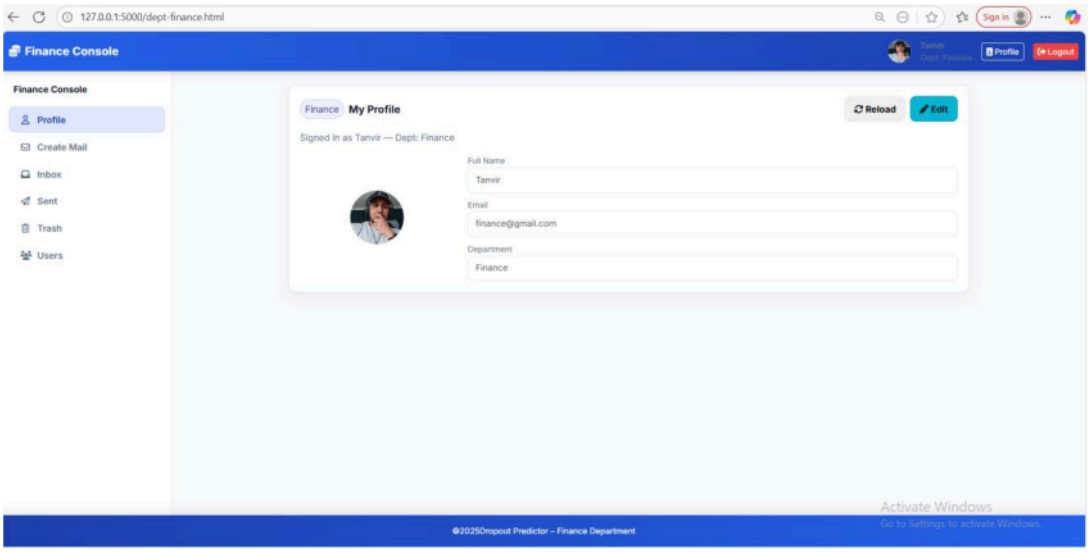


Figure 4. 30 Department Profile (Finance)

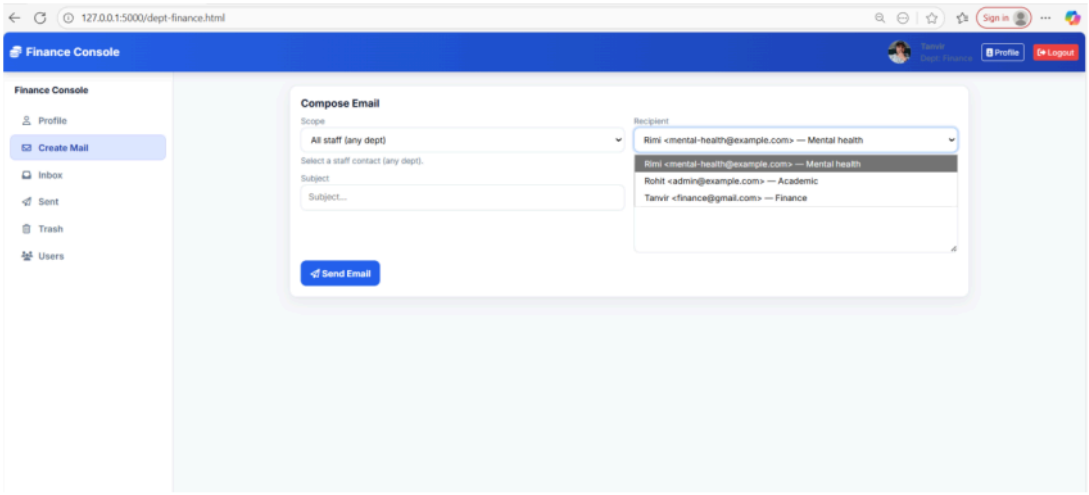


Figure 4. 31 Department Profile Create Mail (Finance)

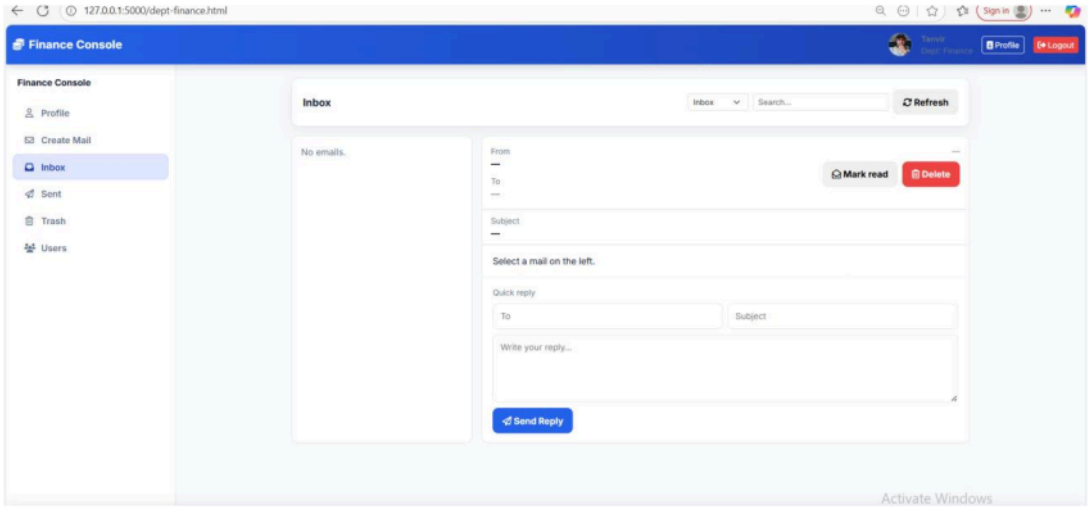


Figure 4. 32 Department Profile inbox (Finance)

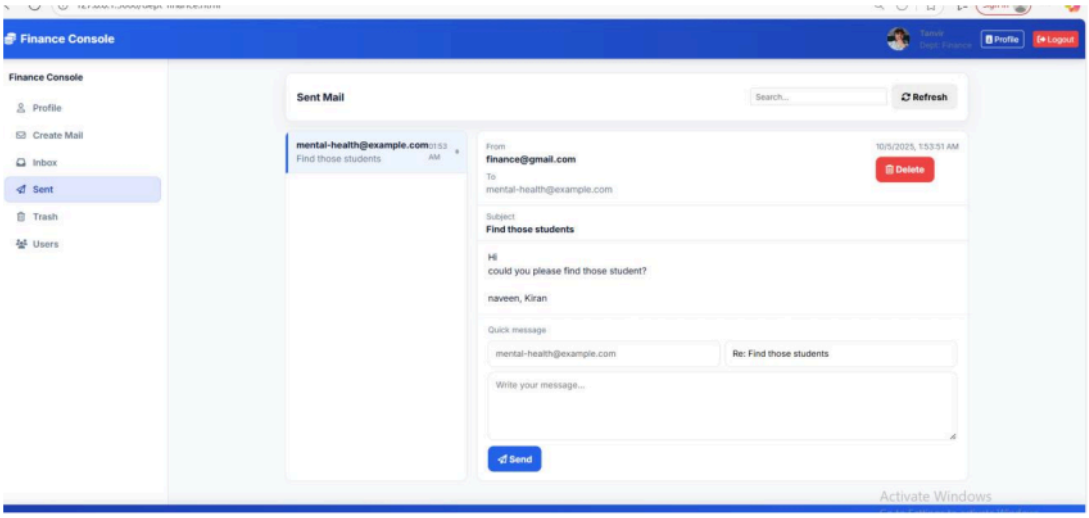


Figure 4. 33 Finance Dept- mailbox

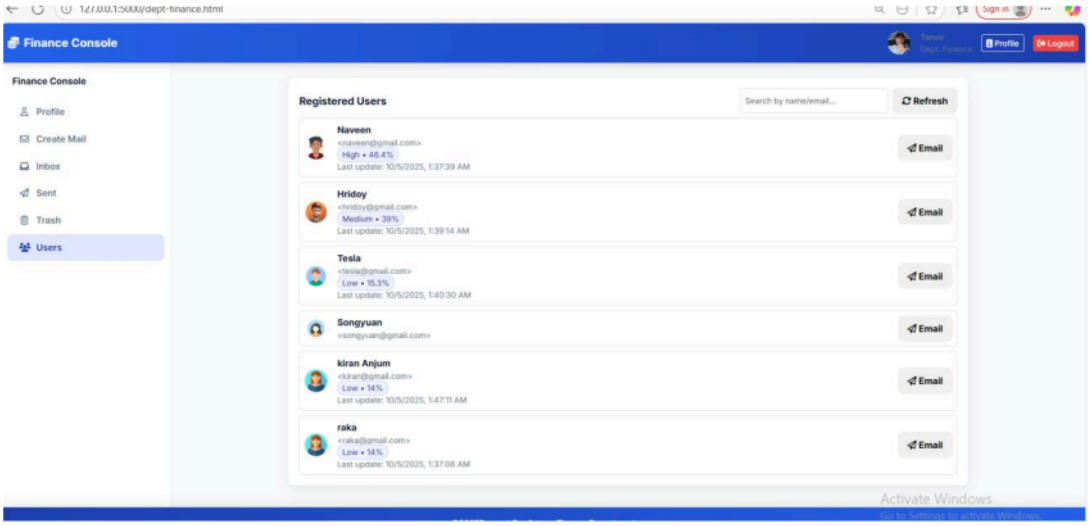


Figure 4. 34 Finance Dept (View Users)

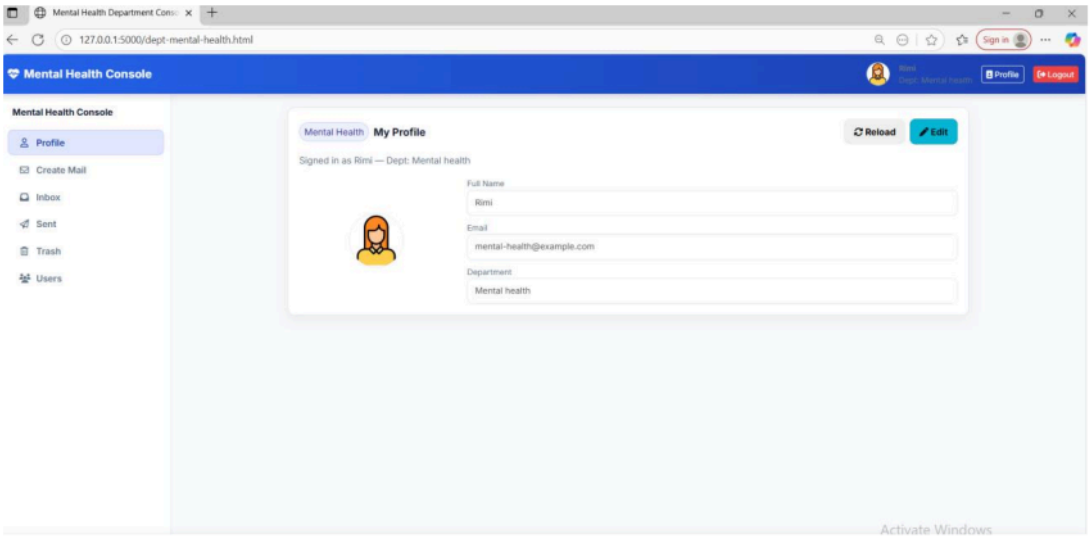


Figure 4. 35 Mental health dept - staff profile

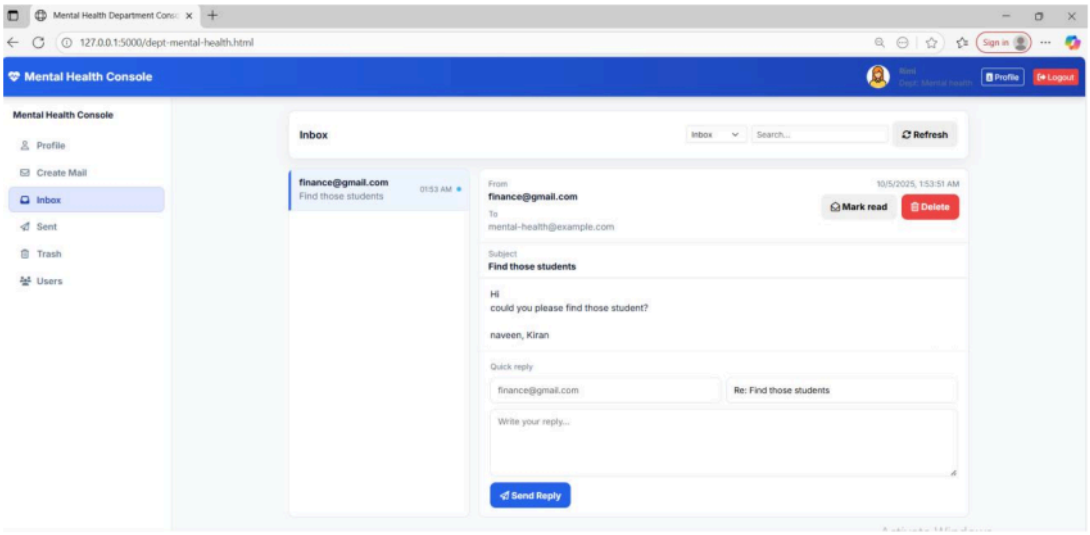


Figure 4. 36 Mental health dept -mailbox

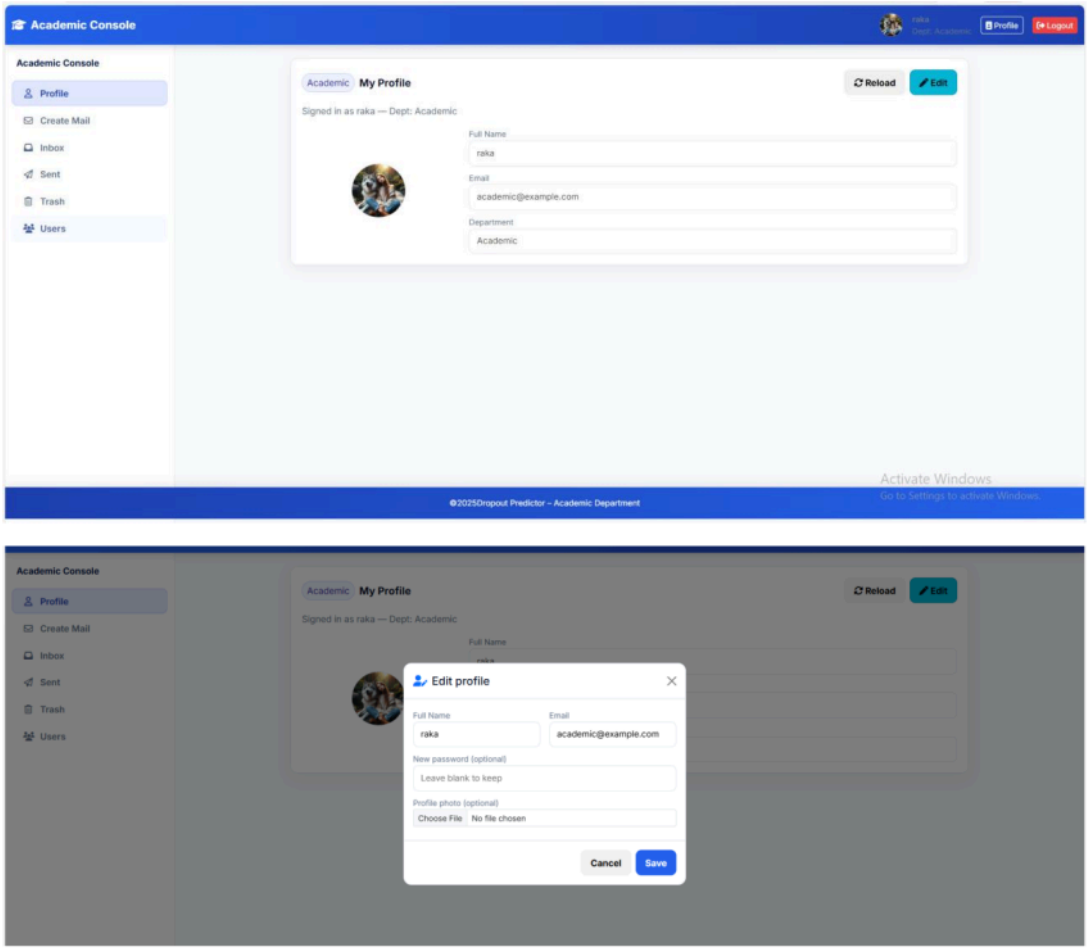


Figure 4. 37 Academic Dept -staff profile

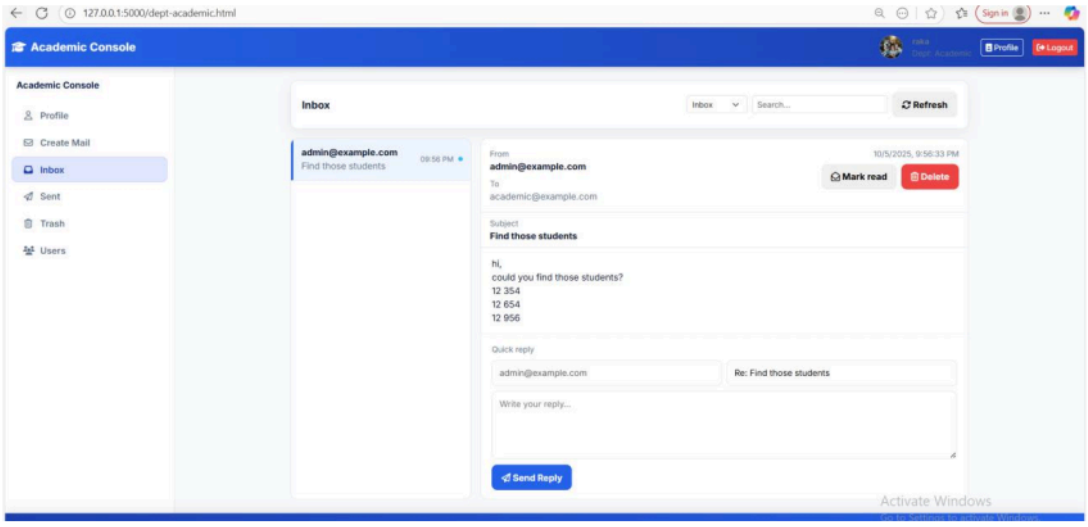


Figure 4. 38 Academic Dept – Mailbox

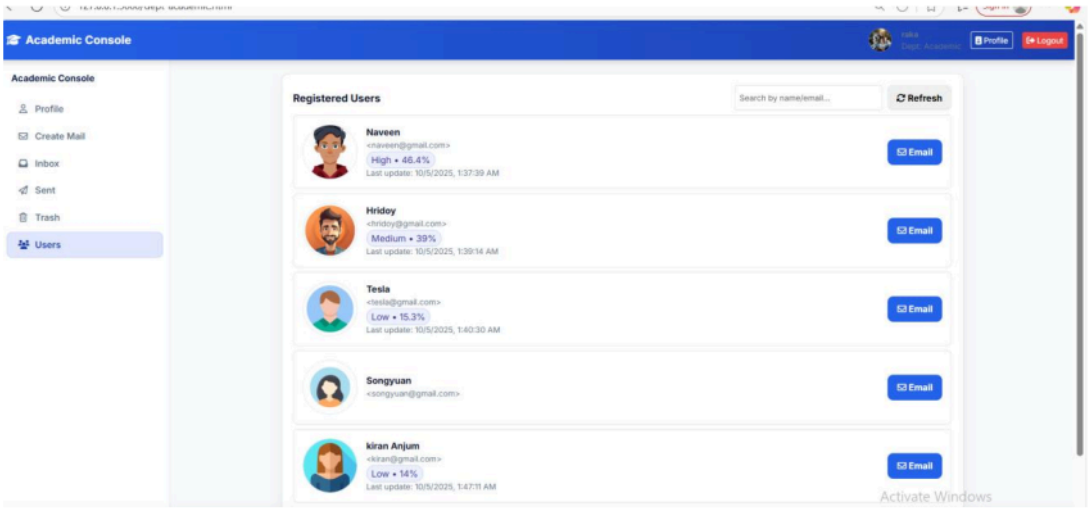


Figure 4. 39 Academic Dept- Users

Appendix B – Ethical Approval

Faculty of Science, Business & Enterprise
Science & Engineering Research Project Form - Student

Please type your responses below and email to SciEng-REC@chester.ac.uk
Handwritten copies will not be accepted.

Your Name:	MD TANVIR SARDARR
Your student number:	2425307
Email Address:	2425307@CHESTER.AC.UK
Programme of Study:	MASTER OF SCIENCE IN DATA SCIENCE
Name of Principal Supervisor:	**Nkosi Mpofu**
Title of Research Project:	**A random forest-based system for predicting student dropout risk from academic and attendance records**
Start date of project: 18 April 2025	Anticipated end date of project: 7 Aug 2025

Please explain why you want to carry out this research and what your proposed method will be (no more than 100 words):
My objective is to develop a web-based system that analyzes students' academic records and attendance data to identify their academic withdrawal risks. Detecting students early in their academic risk profile enables the creation of specific support systems that promote academic achievement and reduce student withdrawal rates. This system will be simulated, with Monte-carlo used to generate synthetic data, capturing data such as GPA and attendance. These records are stored in an SQLite database and used to train a Random Forest model that predicts the likelihood of student dropout. The validated model is implemented in a Flask-based platform interface application, which provides a handling system interface and interactive dashboards accessible to students and administrative staff for customized analytics and actionable information.

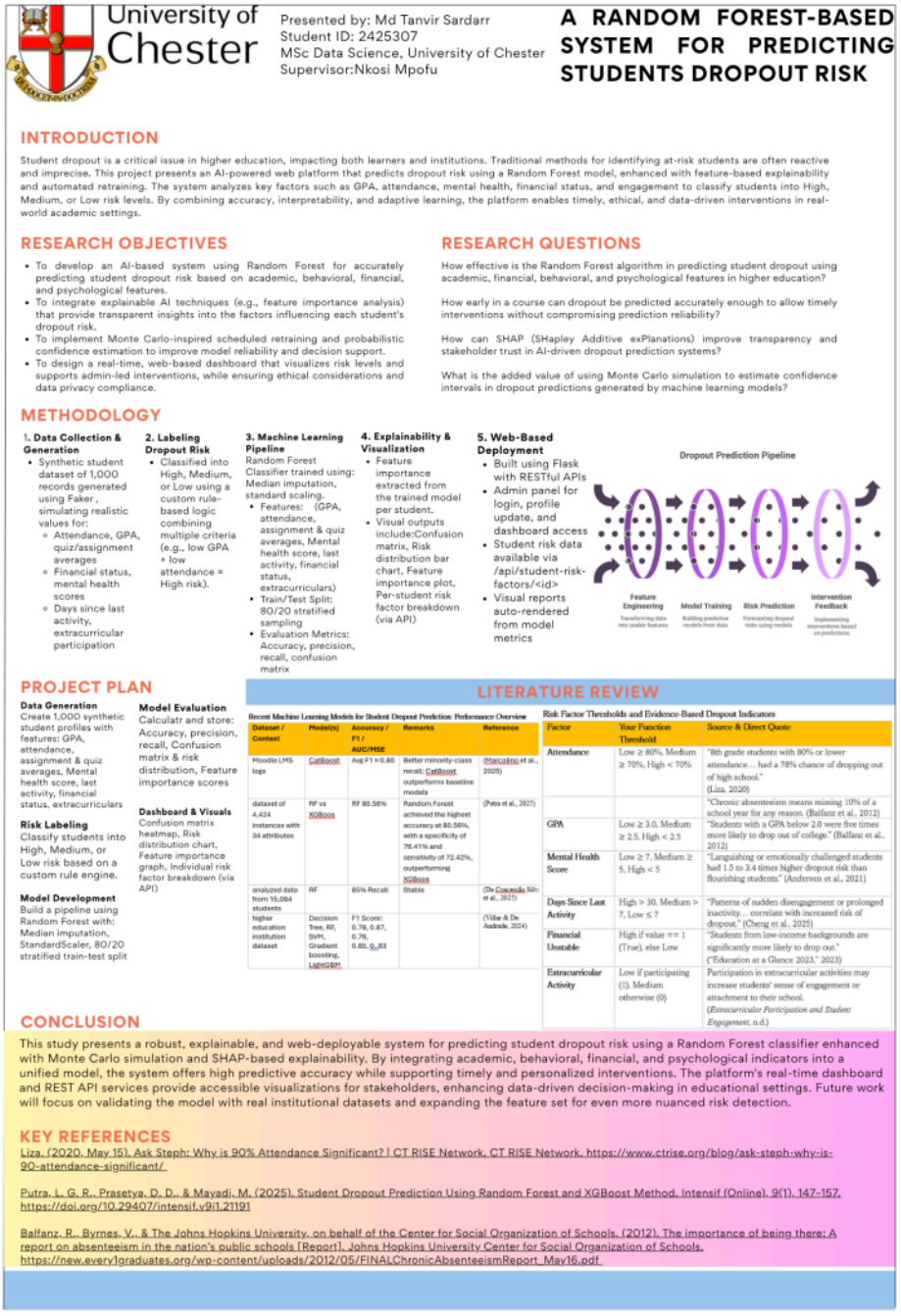
Please respond to the following questions:

Question	**Response**
Will your research be based on reviewing existing literature?	Entirely ☐ Partly ☐ Not at all ☐ Not sure at this stage √
Will your research involve mathematical work?	Yes ☐ No √ Not sure at this stage ☐
Are you planning to develop or test any computer system or piece of hardware or software?	Yes ☐ No ☐ Not sure at this stage √
Will your research involve you carrying out testing using an isolated network or virtual machine?	Yes ☐ No ☐ Not sure at this stage √
Are you intending to use research data available from an online source?	Yes☐ No√ Not sure at this stage ☐ (Monti Carlo Simulation will be used to generate Random data)

If yes, have you checked that there are no copyright or data protection issues involved in you working with and reproducing this data?	Yes, I've checked and there are no issues √ Yes, I've checked and there are issues ☐ No, I haven't checked ☐
Will your research involve work carried out in a science laboratory?	Yes ☐ No√ Not sure at this stage ☐
Will your research involve fieldwork?	Yes ☐ No√ Not sure at this stage ☐
If you have answered 'yes' to either of these, have you carried out a Risk Assessment?	Yes ☐ No√ Not sure at this stage ☐
Risk assessment reference number:	
Will you need to liaise with the Laboratory Manager regarding any special requirements to be observed in addition to standard lab procedures and PPE?	Yes ☐ No√
Will your project involve you having contact with human participants, e.g. through interviews, focus groups, data gathering via questionnaires, surveys on social media, etc.?	Yes ☐ No√ Not sure at this stage ☐
Will your project involve you having contact with animals or animal tissues?	Yes ☐ No√ Not sure at this stage ☐
If you are not working directly with animals, or animal tissues, are you using research data about these which has collected by another person or organisation?	Yes ☐ No √ Not sure at this stage ☐
Is permission needed to use this data?	Yes, permission is needed and I have got permission ☐ No permission is required√ I haven't checked ☐
Does your project involve the NHS in any way?	Yes ☐ No√ Not sure at this stage ☐
Is your project likely to engage with the natural environment, e.g. by utilising samples collected from nature, producing hazardous chemical by-products, creating noise pollution, etc.?	Yes ☐ No√ Not sure at this stage ☐
Will your project, including data-gathering or collaborative activities, involve research outside of England?	Yes ☐ No√ Not sure at this stage ☐
If you are likely to travel outside of England to conduct research, where are you intending to go?	Please provide details: NO
Are you aware of any risks to you in travelling to the destinations named above?	Yes ☐ No √
Do you think this project might need ethical approval?	Yes√ No ☐
Have you discussed this with your supervisor?	Yes√ No ☐

Your signature:	MD TANVIR SARDARR
Supervisor's Signature:	
Comments from the Science & Engineering Research Ethics Committee	No ethical concerns identified
Signature of the Chair of the Science & Engineering Research Ethics Committee:	
Date:	27/05/2025

Appendix C -Poster Presentation



University of Chester

Presented by: Md Tanvir Sardarr
Student ID: 2425307
MSc Data Science, University of Chester
Supervisor:Nkosi Mpofu

A RANDOM FOREST-BASED SYSTEM FOR PREDICTING STUDENTS DROPOUT RISK

INTRODUCTION

Student dropout is a critical issue in higher education, impacting both learners and institutions. Traditional methods for identifying at-risk students are often reactive and imprecise. This project presents an AI-powered web platform that predicts dropout risk using a Random Forest model, enhanced with feature-based explainability and automated retraining. The system analyzes key factors such as GPA, attendance, mental health, financial status, and engagement to classify students into High, Medium, or Low risk levels. By combining accuracy, interpretability, and adaptive learning, the platform enables timely, ethical, and data-driven interventions in real-world academic settings.

RESEARCH OBJECTIVES

- To develop an AI-based system using Random Forest for accurately predicting student dropout risk based on academic, behavioral, financial, and psychological features.
- To integrate explainable AI techniques (e.g., feature importance analysis) that provide transparent insights into the factors influencing each student's dropout risk.
- To implement Monte Carlo-inspired scheduled retraining and probabilistic confidence estimation to improve model reliability and decision support.
- To design a real-time, web-based dashboard that visualizes risk levels and supports admin-led interventions, while ensuring ethical considerations and data privacy compliance.

RESEARCH QUESTIONS

How effective is the Random Forest algorithm in predicting student dropout using academic, financial, behavioral, and psychological features in higher education?

How early in a course can dropout be predicted accurately enough to allow timely interventions without compromising prediction reliability?

How can SHAP (SHapley Additive exPlanations) improve transparency and stakeholder trust in AI-driven dropout prediction systems?

What is the added value of using Monte Carlo simulation to estimate confidence intervals in dropout predictions generated by machine learning models?

METHODOLOGY

1. Data Collection & Generation
- Synthetic student dataset of 1,000 records generated using Faker , simulating realistic values for:
  - Attendance, GPA, quiz/assignment averages
  - Financial status, mental health scores
  - Days since last activity, extracurricular participation

2. Labeling Dropout Risk
- Classified into High, Medium, or Low using a custom rule-based logic combining multiple criteria (e.g., low GPA + low attendance = High risk).

3. Machine Learning Pipeline
Random Forest Classifier trained using: Median imputation, standard scaling.
- Features: (GPA, attendance, assignment & quiz averages, Mental health score, last activity, financial status, extracurriculars)
- Train/Test Split: 80/20 stratified sampling
- Evaluation Metrics: Accuracy, precision, recall, confusion matrix

4. Explainability & Visualization
- Feature importance extracted from the trained model per student.
- Visual outputs include:Confusion matrix, Risk distribution bar chart, Feature importance plot, Per-student risk factor breakdown (via API)

5. Web-Based Deployment
- Built using Flask with RESTful APIs
- Admin panel for login, profile update, and dashboard access
- Student risk data available via /api/student-risk-factors/<id>
- Visual reports auto-rendered from model metrics

PROJECT PLAN

Data Generation
Create 1,000 synthetic student profiles with features: GPA, attendance, assignment & quiz averages, Mental health score, last activity, financial status, extracurriculars

Risk Labeling
Classify students into High, Medium, or Low risk based on a custom rule engine.

Model Development
Build a pipeline using Random Forest with: Median imputation, StandardScaler, 80/20 stratified train-test split

Model Evaluation
Calculatr and store: Accuracy, precision, recall, Confusion matrix & risk distribution, Feature importance scores

Dashboard & Visuals
Confusion matrix heatmap, Risk distribution chart, Feature importance graph, Individual risk factor breakdown (via API)

LITERATURE REVIEW

Recent Machine Learning Models for Student Dropout Prediction: Performance Overview

Dataset / Context	Model(s)	Accuracy / F1 / AUC/MSE	Remarks	Reference
Moodle LMS logs	CatBoost	Avg F1 = 0.80	Better minority-class recall; CatBoost outperforms baseline models	(Marcolino et al., 2025)
dataset of 4,424 instances with 34 attributes	RF vs XGBoos	RF 80.56%	Random Forest achieved the highest accuracy at 80.56%, with a specificity of 76.41% and sensitivity of 72.42%, outperforming XGBoos	(Pato et al., 2025)
analyzed data from 15,064 students	RF	85% Recall	Stable	(Da Cruz-nda Silv et al., 2023)
higher education institution dataset	Decision Tree, RF, SVM, Gradient boosting, LightGBM	F1 Score: 0.78, 0.87, 0.78, 0.85, 0.83		(Villar & De Andrade, 2024)

Risk Factor Thresholds and Evidence-Based Dropout Indicators

Factor	Your Function Threshold	Source & Direct Quote
Attendance	Low ≥ 80%, Medium ≥ 70%, High < 70%	"8th grade students with 80% or lower attendance… had a 78% chance of dropping out of high school." (Liza, 2020)
		"Chronic absenteeism means missing 10% of a school year for any reason. (Balfanz et al., 2012)
GPA	Low ≥ 3.0, Medium ≥ 2.5, High < 2.5	"Students with a GPA below 2.0 were five times more likely to drop out of college." (Balfanz et al., 2012)
Mental Health Score	Low ≥ 7, Medium ≥ 5, High < 5	"Languishing or emotionally challenged students had 1.5 to 3.4 times higher dropout risk than flourishing students." (Andersen et al., 2021)
Days Since Last Activity	High > 30, Medium > 7, Low ≤ 7	"Patterns of sudden disengagement or prolonged inactivity… correlate with increased risk of dropout." (Cheng et al., 2025)
Financial Unstable	High if value == 1 (True), else Low	"Students from low-income backgrounds are significantly more likely to drop out." ("Education at a Glance 2023," 2023)
Extracurricular Activity	Low if participating (1), Medium otherwise (0)	Participation in extracurricular activities may increase students' sense of engagement or attachment to their school. (*Extracurricular Participation and Student Engagement*, n.d.)

CONCLUSION

This study presents a robust, explainable, and web-deployable system for predicting student dropout risk using a Random Forest classifier enhanced with Monte Carlo simulation and SHAP-based explainability. By integrating academic, behavioral, financial, and psychological indicators into a unified model, the system offers high predictive accuracy while supporting timely and personalized interventions. The platform's real-time dashboard and REST API services provide accessible visualizations for stakeholders, enhancing data-driven decision-making in educational settings. Future work will focus on validating the model with real institutional datasets and expanding the feature set for even more nuanced risk detection.

KEY REFERENCES

Liza. (2020, May 15). Ask Steph: Why is 90% Attendance Significant? | CT RISE Network. CT RISE Network. https://www.ctrise.org/blog/ask-steph-why-is-90-attendance-significant/

Putra, I. G. R., Prasetya, D. D., & Mayadi, M. (2025). Student Dropout Prediction Using Random Forest and XGBoost Method. Intensif (Online), 9(1), 147–157. https://doi.org/10.29407/intensif.v9i1.21191

Balfanz, R., Byrnes, V., & The Johns Hopkins University, on behalf of the Center for Social Organization of Schools. (2012). The importance of being there: A report on absenteeism in the nation's public schools [Report]. Johns Hopkins University Center for Social Organization of Schools. https://new.every1graduates.org/wp-content/uploads/2012/05/FINALChronicAbsenteeismReport_May16.pdf