ADAPTIVE AND EXPLAINABLE AI FOR STUDENT RISK PREDICTION AND PERSONALIZED ACADEMIC INTERVENTIONS: A CONTINUOUS LEARNING ANALYTICS FRAMEWORK

25 - 26J - 172

Disanayaka S.T

IT22370228

Final Report

B.Sc. (Hons) Degree in Information Technology

Specializing in Information Technology

Department of Information Technology

Sri Lanka Institute of Information Technology

Sri Lanka

April 2026

PROACTIVE EDUCATIONAL FRAMEWORK FOR PERSONALIZED ACADEMIC ACTION PLANS WITH MACHINE LEARNING AND GENERATIVE AI

Disanayaka S T

IT22370228

Dissertation submitted in partial fulfillment of the requirements for B.Sc. (Honors) Degree in Information Technology

Department of Information Technology

Sri Lanka Institute of Information Technology Sri Lanka

April 2026

DECLARATION

I declare that this is my own work, and this dissertation does not incorporate without acknowledgement any material previously submitted for a degree or diploma in any other university or Institute of higher learning and to the best of my knowledge and belief it does not contain any material previously published or written by another person except where the acknowledgement is made in the text. Also, I hereby grant to Sri Lanka Institute of Information Technology the non-exclusive right to reproduce and distribute my dissertation in whole or part in print, electronic, or other medium. I retain the right to use this content in whole or part in future works.

Name	Student ID	Signature
Disanayaka S.T	IT22370228	

The above candidate is carrying out research for the undergraduate Dissertation under my supervision

---------- 26.4.2026

Signature of the supervisor ---------- Date

ACKNOWLEDGEMENT

I would like to express my heartfelt gratitude to everyone who supported me throughout the course of this research project.

First and foremost, I extend my deepest appreciation to my supervisor, Ms. Sanjeevi, for her guidance, continuous support, and insightful feedback. Her expertise, encouragement, and constructive advice were instrumental in shaping the direction of this research and ensuring the successful development of the Universal Recommendation Engine.

I am also sincerely grateful to my AcademiGuard team members for their collaboration, support, and dedication throughout this journey. Their technical contributions and seamless coordination across the different system components played a vital role in the smooth progression of the overall portal.

A special thanks goes to the academic faculty and advisors whose professional insights enriched my understanding of student success metrics and psychological wellness, as well as the students and peers who provided meaningful feedback that greatly informed the system's practical outcomes.

Finally, I would like to extend my appreciation to my family, friends, and all those who offered their time, encouragement, and assistance, whether through advice, motivation, or technical support. Their help has been invaluable, and I am truly thankful to everyone who played a major role in making this project a success.

Abstract

Most universities struggle to give students personalized advice every week. While many educational software systems try to predict if a student will fail, they rarely tell the student exactly how to fix the problem. This research focuses on building the recommendation and dashboard component of an AI-Driven Student Success Portal to solve this issue. Instead of predicting a simple pass or fail, the system uses a Tiered Recommendation Engine. I built a machine learning model using Random Forest, Gradient Boosting, and Logistic Regression to calculate a "Support Index" percentage. Based on this index, the system groups the student into one of three stages: On Track, Needs Attention, or Priority Support Needed.

Once the student is classified, the system sends their status and the current weekly syllabus to a Large Language Model using the fast Groq API (Llama 3.1). The AI then generates a personalized, step-by-step study and wellness plan. The entire system is connected through a React frontend and a Firebase backend, allowing students to receive fast, actionable advice on an interactive dashboard. The final results show that this tiered, syllabus-aware approach successfully gives students clear guidance on what to do next, rather than just giving them a basic warning.

Keywords: *Tiered Recommendation Engine, Generative AI, Support Index, Hybrid Machine Learning, EdTech Dashboard, Automated Academic Advising***..**

Table of Contents

List of Figures

List of Tables

List of Abbreviations

Abbreviations	Descriptions
AI	Artificial Intelligence
GenAI	Generative Artificial Intelligence
ML	Machine Learning
LLM	Large Language Model
XAI	Explainable Artificial Intelligence
LMS	Learning Management System
API	Application Programming Interface

1. INTRODUCTION

1.1 Background

University students face a high level of pressure. They must balance heavy workloads, difficult subjects, and their own mental well-being. When a student starts to struggle, they usually do not get help until after they fail a midterm or miss several assignments. By that time, it is often too late to catch up.

Currently, universities rely on academic advisors to help students. However, human advisors cannot review the weekly habits of hundreds of students at the same time. To solve this, many universities use digital dashboards. Unfortunately, most of these dashboards only show past data, like attendance records or past quiz scores. They tell the student that they are failing, but they do not tell the student exactly how to fix it.

This research proposes an AI-Driven Student Success Portal that changes how we monitor students. Instead of waiting for a student to fail, this system acts as a proactive, automated academic advisor. It uses a Hybrid Ensemble Machine Learning model to evaluate the student's current weekly habits—such as study hours, stress levels, and recent grades. The model calculates a "Support Index" and classifies the student into a specific support tier. Then, the system sends this tier, along with the current week's syllabus, to a Generative AI model (Llama 3.1). The AI generates a highly personalized, step-by-step study and wellness plan. This ensures that every student receives immediate, actionable advice tailored to their exact situation..

1.2 Literature Survey

To build an effective student recommendation system, it is important to understand what researchers have already done in the fields of Educational Data Mining (EDM) and Learning Analytics.

I. Predicting Student Performance

Researchers have spent years trying to predict student success using machine learning. Early studies focused on identifying which students were likely to drop out. Romero and Ventura [1] established the foundation of using data mining in education to track student behavior. Later, researchers like Baker [2] and Siemens [3] showed that tracking engagement metrics, such as forum clicks and login times, could accurately predict final grades. More recently, deep learning methods have been applied to capture complex student patterns [4]. However, a major limitation of these traditional models is that they only focus on binary risk prediction. They output a simple "pass" or "fail" label, which does not give the student any useful feedback on how to improve.

II. Ensemble Learning in Education

To improve the accuracy of these predictions, researchers started using ensemble machine learning. Ensemble learning combines multiple algorithms to get a better result. Studies show that Random Forest classifiers are highly effective at handling educational data because they prevent overfitting [5]. Gradient Boosting has also become popular for predicting student performance because it learns from its previous mistakes during training [6], [7]. Furthermore, combining these models using a voting classifier provides a much more stable and reliable prediction than using a single algorithm [8]. This project builds on this research by using a Soft Voting Classifier (Random Forest, Gradient Boosting, and Logistic Regression). However, instead of predicting a final grade, this project uses the ensemble model to calculate a real-time "Support Index" to classify the student's current intervention stage.

III. Learning Dashboards and Student Well-being

Showing data to students requires careful design. Verbert et al. [9] explain that learning dashboards must present data clearly so students can reflect on their progress. Other researchers found that poorly designed dashboards can actually increase student anxiety if they only focus on low grades

[10], [11]. This proves that a dashboard must provide solutions, not just warnings. Furthermore, modern educational research highlights the importance of mental health. Studies show that high stress and lack of sleep directly lower academic performance [12]. Therefore, an effective system must track wellness data alongside academic data.

IV. Generative AI and Large Language Models (LLMs)

The release of Large Language Models has completely changed educational technology. Researchers have started using LLMs like GPT-4 and LLaMA to act as virtual tutors [13], [14]. These models can generate human-like text and explain difficult concepts. However, recent studies warn that generic AI chatbots often give vague or incorrect advice if they do not understand the specific context of the student's course [15], [16]. To make AI useful, it must be guided by strict prompt engineering and factual context [17], [18].

1.3 Research Gap

Based on the literature survey, there is a clear gap in current educational technology. Researchers have built highly accurate machine learning models to predict student performance. However, these traditional models stop at the prediction stage. They give a "pass" or "fail" warning, but they do not tell the student what steps to take to fix the problem.

On the other hand, newer educational tools use Generative AI chatbots to help students. The problem with these AI tools is that they are usually generic. They do not know the student's actual risk level, their current stress, or the specific topics the professor is teaching that week. As a result, they give basic advice like "study harder" or "manage your time better."

The main research gap is the missing connection between quantitative risk prediction and qualitative, actionable advice. Existing systems treat machine learning and generative AI as two separate tools.

This project fills that exact gap by connecting the two technologies into a single pipeline. It does not just predict failure; it calculates a mathematical "Support Index" to figure out exactly how much help the student needs right now. Then, it automatically feeds that specific index—along with the actual weekly syllabus—directly into the Generative AI. This creates a complete, closed-loop system: it identifies the exact level of the problem and immediately generates a personalized, step-by-step weekly schedule to solve it.

Table 1. Comparison of the existing research methods and proposed method

Author & Year	Related works	Data & Processing	Model Training	Output	Limitation
M. Yağcı (2022) [5]	Predicting academic performance using machine learning.	Cleaned university grading datasets.	Random Forest and SVM models.	Binary classification (Pass/Fail label).	Only warns the student they might fail, but provides no action plan.
K. Verbert et al. (2013) [9]	Learning analytics dashboard applications.	Raw engagement data (clicks, logins) from Learning Management Systems.	Descriptive statistics and data visualization.	Visual charts and graphs of past activity.	Highly reactive. Forces the student to interpret the charts themselves without AI guidance.
E. Kasneci et al. (2023) [13]	Large language models (LLMs) for education.	Text prompts manually entered by the user.	Generative AI (GPT models) using standard prompt engineering.	Human-like text explanations and tutoring.	Gives generic advice. The AI does not know the student's actual mathematical risk level or stress data.
Proposed System	AI-Driven Student Success Portal & Dashboard.	Cleaned academic and wellness data (stress, sleep), removing data leakage.	Hybrid Ensemble (RF, GB, LR) combined with Groq Llama 3.1 LLM.	Quantitative Support Index + Syllabus-aware step-by-step weekly action plan.	Solves previous limitations by connecting mathematical risk directly to automated, personalized AI advice.

1.4 Research Problem

University students constantly face heavy workloads, tight deadlines, and high stress. When a student starts to fall behind or experience burnout, they rarely know how to fix the problem on their own. They need immediate, specific guidance. However, the current tools available to students and universities fail to provide this help in real-time.

The main problem is that current university dashboards are purely reactive. They only show historical data, such as past attendance or old quiz scores. If a student gets a bad grade, the system simply displays a warning or a failing mark. It does not tell the student how to study or what to do next to improve. While human academic advisors exist to solve this problem, they are overloaded. One advisor cannot sit down with hundreds of students every single week to write custom study schedules for them.

Recently, some students have tried using generic AI chatbots (like ChatGPT) to ask for study advice. However, this creates a new problem. These generic chatbots do not have access to the student's actual university database. The AI does not know the student's real mathematical risk level, their current stress and sleep data, or the specific syllabus topics the professor is teaching that week. Because the AI lacks this context, it only gives basic, useless advice like "manage your time better" or "read your textbook."

Therefore, the specific research problem is: There is no automated system that bridges the gap between quantitative academic tracking and qualitative AI advising. Currently, there is no platform that automatically calculates a student's real-time need for support, reads their weekly syllabus, and directly feeds that data into an AI to generate a personalized, step-by-step weekly action plan.

1.5 Research Objective

1.5.1 Main Objective

The main objective of this individual component is to design, develop, and test an automated recommendation system that calculates a student's real-time need for academic help and generates a personalized, syllabus-aware weekly action plan to improve their performance.

1.5.2 Specific Sub Objective

To successfully achieve the main objective, I broke the project down into four specific sub-objectives**:**

1. To build and train a Tiered Recommendation Engine:

Instead of using standard binary risk prediction, this objective focuses on training a Hybrid Ensemble Machine Learning model (using Random Forest, Gradient Boosting, and Logistic Regression). The goal is to use a soft-voting method to calculate a quantitative "Support Index" percentage. This index will classify the student into one of three dynamic stages: On Track, Needs Attention, or Priority Support Needed.

2. To integrate a Generative AI for custom study plans:

This objective focuses on connecting the mathematical Support Index to a Large Language Model using the high-speed Groq API (Llama 3.1). The goal is to use strict prompt engineering to feed the student's status, stress levels, and weekly syllabus into the AI, forcing it to return a valid JSON file containing specific academic and wellness tips.

3. To develop a real-time data processing backend:

This objective involves writing server-side logic using Node.js and Firebase Cloud Functions. The goal is to calculate the student's weekly workload hours and ensure the database updates correctly when the semester schedule changes, preventing old "ghost data" from breaking the AI generation.

4. To design an interactive frontend dashboard:

This objective focuses on building the user interface using React.js. The goal is to create a dashboard that safely reads the AI-generated JSON arrays and displays them as easy-to-read, side-by-side summary cards and interactive checklists, allowing the student to clearly see what they need to do for the week.

.

2. METHODOLOGY

This chapter explains the steps and technologies I used to build the AI-Driven Student Success Portal. The main goal of this component is to calculate a student's real-time need for help and generate a customized weekly study plan. To build this, I combined data processing, a machine learning classification model, and Generative AI prompt engineering..

2.1 Requirement Gathering and Analysis

Before building the AI-Driven Student Success Portal, I needed to understand exactly what students and universities required. I analyzed existing university dashboards and found that they only show past grades. They do not tell a student how to fix a bad grade. Therefore, the main requirement was to build a system that acts proactively. I determined that the system must collect real-time data—such as weekly study hours, sleep, and stress levels—and combine that with the professor's weekly syllabus to generate an actionable study plan.

2.2 Feasibility Study

I conducted a feasibility study to ensure the project could be built within the given timeframe and budget:

- Technical Feasibility:

The required technologies (React, Node.js, Python) are open-source and well-documented. Using the Groq API for the Large Language Model was highly feasible because it provides instant text generation compared to slower local models.

- Operational Feasibility:

The system is operationally viable because it automates the job of an academic advisor. It does not

require university staff to manually type out advice for hundreds of students every week.

- Economic Feasibility:

Firebase offers a generous free tier for database hosting, and the Groq API is highly cost-effective for generating text prompts, making this a low-cost solution to develop.

2.3 Tools, Technologies, and Algorithms

To build a full-stack, AI-powered system, I used a combination of web technologies and machine learning algorithms.

- **Tools & Technologies:**

I built the frontend user interface using **React.js**. For the database and server-side logic, I used **Firebase Firestore** and **Node.js** Cloud Functions. I used **Python** with the **FastAPI** framework to build the microservice that handles the machine learning predictions. To generate the human-like advice, I integrated the **Groq API** running the **Llama 3.1** model.

- **Algorithms:**

I built a Tiered Recommendation Engine using a Hybrid Ensemble Machine Learning model. I combined three algorithms: **Random Forest**, **Gradient Boosting**, and **Logistic Regression**. I used a **Soft Voting Classifier** to average the probabilities of these three algorithms, which outputs a percentage called the "Support Index."

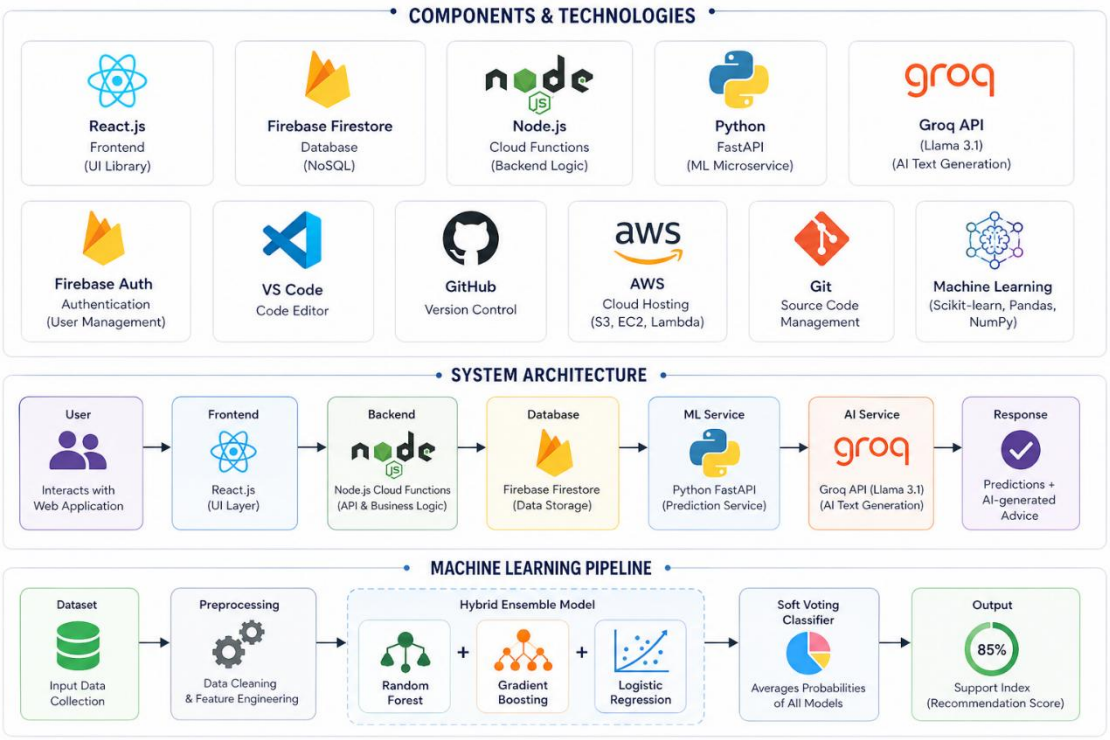

Figure 1. Tools & Technologies

2.4 Project Requirements

2.4.1 Functional Requirements

- The system must allow students to input their weekly habits, including study hours, stress levels, and recent grades.
- The system must calculate a "Support Index" percentage using the machine learning model and classify the student into one of three tiers: On Track, Needs Attention, or Priority Support Needed.
- The system must automatically send the student's tier, numeric data, and the weekly syllabus to the Groq API.
- The system must generate a personalized study and wellness plan and format it as a strict JSON file so the React frontend can read it.
- The system must save historical plans in the Firebase database so the student can review past weeks.

2.4.2 Non-functional Requirements

- **Performance:** The AI generation must be fast. Because the system uses the Groq API, the dashboard must display the study plan within seconds so the user does not wait on a loading screen.
- **Usability:** The dashboard must be easy to read. It should use color-coded cards (e.g., blue for Academic, purple for Wellness) and interactive checkboxes for syllabus tasks.
- **Security & Privacy:** The machine learning model must not process personally identifiable information (PII). All student names and IDs must be removed before the data is analyzed.

2.4.3 System Requirements

To guarantee high availability, rapid inference, and seamless integration, the Universal Recommendation Engine necessitates a robust and scalable technological infrastructure. On the software front, the computational backend requires a Python-based environment (such as FastAPI) equipped with industry-standard data science libraries, notably Scikit-Learn, XGBoost, and Pandas, to effectively maintain and execute the Hybrid Voting Classifier.

The generative component introduces specific networking requirements; rather than relying on heavy, localized GPU clusters, the system necessitates secure, low-latency API access to the Groq inference engine to utilize the llama-3.3-70b-versatile model in real-time. For the user interface, the system requires a modern web environment capable of compiling and rendering complex React.js components and dynamic state visualizations. From an infrastructural perspective, the platform demands cloud-hosted databases (such as Firebase or AWS) and serverless routing to handle asynchronous data ingestion from institutional Learning Management Systems (LMS) without creating bottlenecks during the machine learning pipeline execution..

2.4.4 Personal Requirements

Beyond the technological framework, the system fundamentally relies on domain-specific human expertise to ensure that the AI-driven interventions are both pedagogically rigorous and

psychologically safe. The foundational logic of the Recommendation Engine—particularly the hardcoded thresholds governing sleep deprivation and critical stress levels—requires continuous validation from university mental health professionals and counseling staff to ensure the AI's wellness advice remains empathetic and medically appropriate.

Furthermore, the system necessitates active collaboration with academic advisors and faculty members. Because the Generative AI anchors its study plans in weekly syllabus contexts, subject matter experts must ensure that the institutional course materials provided to the LLM are accurate, structured, and up-to-date. Finally, strict adherence to ethical data handling is required from the system administrators; ensuring compliance with educational privacy laws (such as FERPA) and actively auditing the Hybrid Model for demographic or algorithmic bias is paramount to maintaining a fair, inclusive, and transparent academic support environment.

2.5 Overall System Diagram

Figure 3 presents the detailed high-level architecture of the proposed AI-Driven Student Success Portal. This overarching platform is deliberately designed to fundamentally transition university academic support from a reactive, post-mortem model into a proactive, continuous mentoring ecosystem. The system features a centralized, web-based Student Success Dashboard that empowers students to actively monitor their academic trajectory, evaluate their weekly engagement metrics, and interact with tailored recovery strategies, thereby fostering a heightened sense of academic agency and self-regulation.

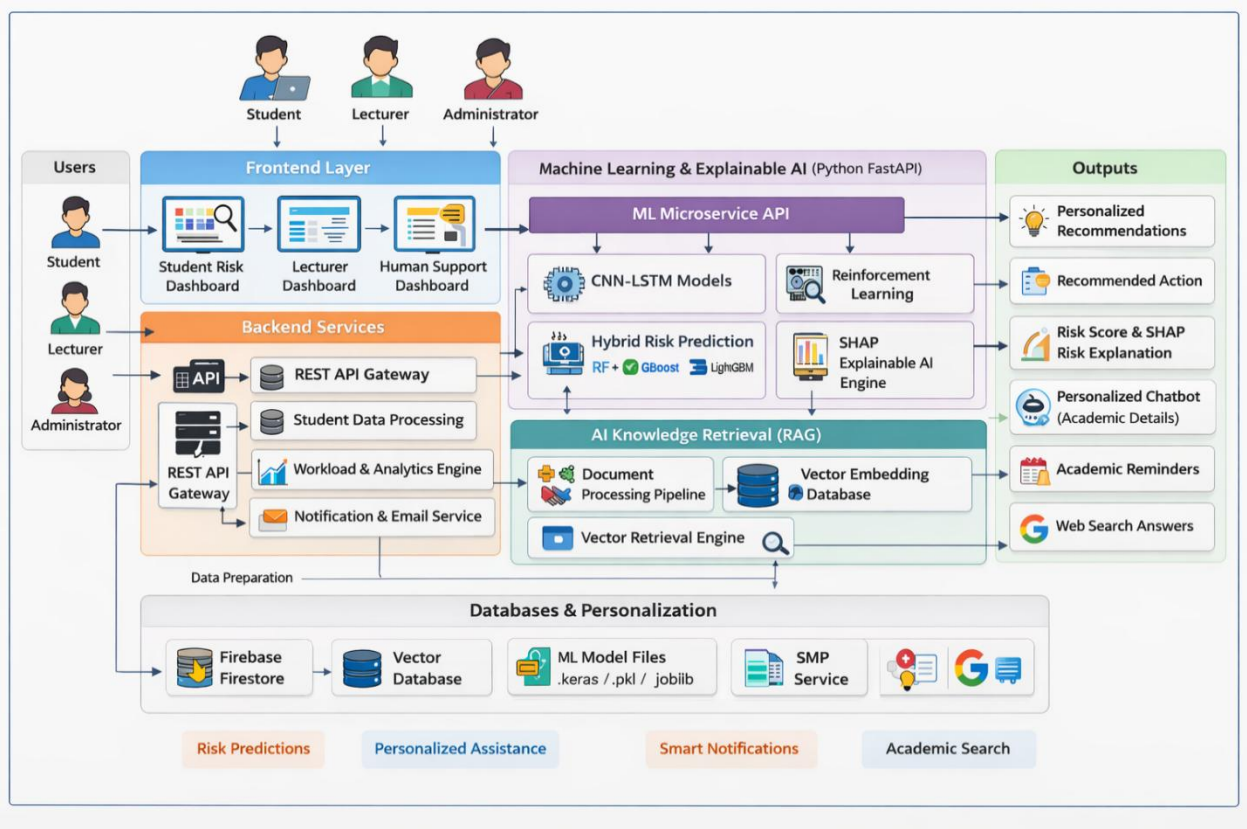

Figure 2. System Overview Diagram

The overarching architecture is composed of four highly integrated, intelligent modules, each tackling a specific domain of learning analytics and automated support.

The first component is the combined Risk Prediction and Explainable AI (XAI) Module. This foundational layer processes academic data to classify student risk levels while simultaneously employing advanced feature-attribution frameworks (such as SHAP) to demystify the predictive algorithms. This guarantees that every risk flag generated by the system is fully transparent, granting educators granular insights into the exact factors driving a student's academic decline.

Working in parallel is a sophisticated Two-Stage Temporal Detection Framework. Recognizing that student behavior is highly dynamic, this module utilizes a Gated Recurrent Unit (GRU)

autoencoder to continuously monitor sequential, weekly behavioral patterns, mapping longitudinal engagement trajectories. It then leverages a Q-learning Reinforcement Learning (RL) algorithm to dynamically prescribe optimal, state-dependent interventions as student habits evolve over the course of the semester.

The third pillar is the Universal Recommendation Engine (the primary focus of this component report). This module acts as the proactive advising core, synthesizing predictive scores, raw academic metrics, and physiological wellness data (such as sleep and stress) through a Hybrid Voting Classifier. It subsequently utilizes Large Language Models (Llama 3.3) to autonomously translate these mathematical outputs into highly personalized, syllabus-aware, and step-by-step intervention schedules.

Finally, the platform features a Retrieval-Augmented Generation (RAG) Conversational Assistant. This interface serves as the direct, conversational touchpoint for the student, facilitating natural and empathetic interactions. By anchoring its generative responses strictly in validated institutional knowledge bases and official course documents, the RAG assistant ensures that all real-time student inquiries are met with accurate, hallucination-free guidance.

Together, these four specialized components synergize to form a comprehensive digital safety net that continuously evaluates longitudinal behavior, explains predictive anomalies, generates personalized recovery plans, and facilitates secure, intelligent student-machine interaction..

2.6 System Diagram of the Component

Figure 4 delineates the precise micro-architecture of the Universal Recommendation Engine, which serves as the proactive intervention and generative advising module within the broader Student Success Portal.

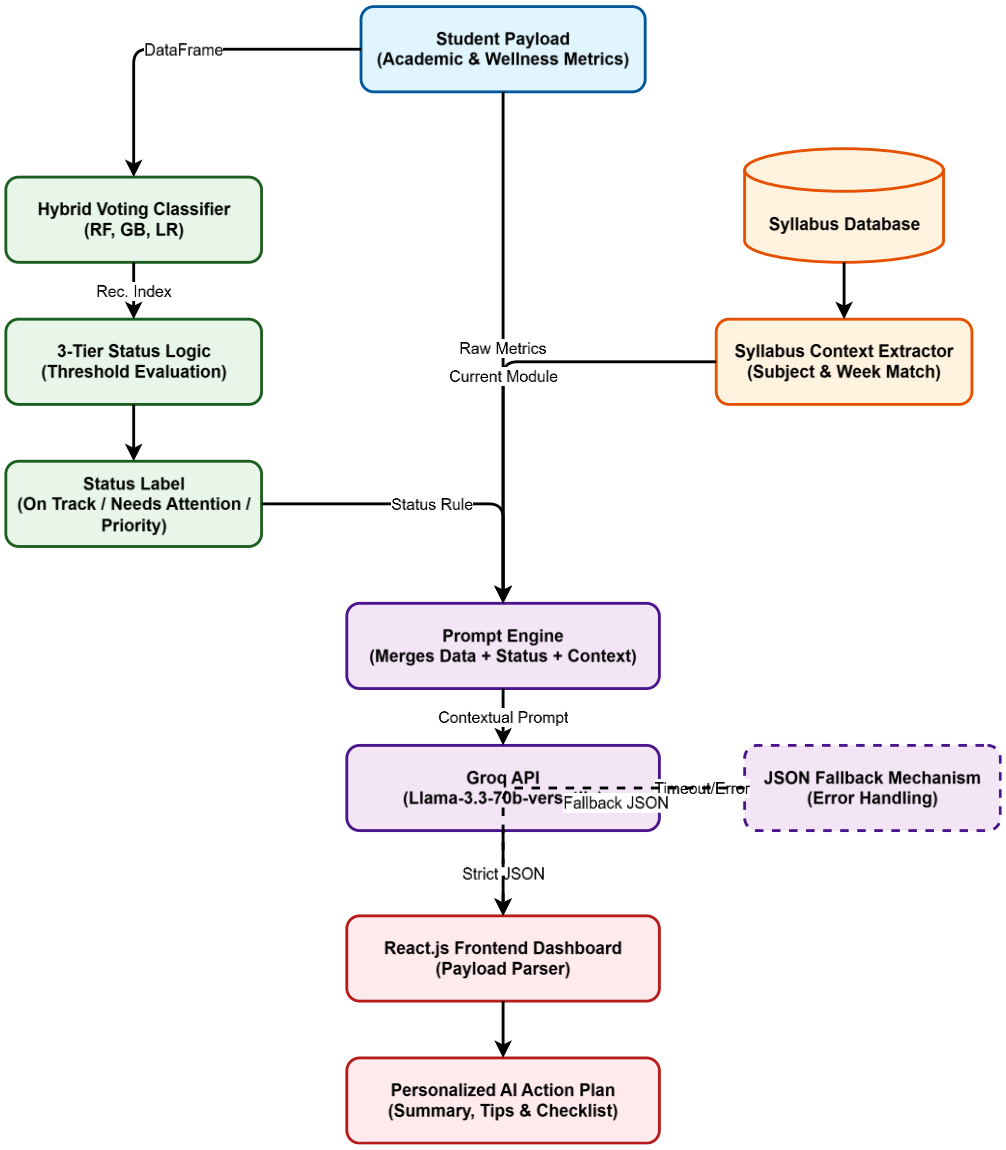

Figure 3. System Diagram of the Component

The operational pipeline initiates at the Data Ingestion Phase, where the engine continuously retrieves a localized payload of the student's dynamic weekly habits. These inputs are not restricted to traditional lagging indicators, such as midterm scores or assignment averages; crucially, they encompass holistic leading indicators, including weekly study hours, sleep duration, and self-reported stress levels. Once ingested, the data flows seamlessly into the Predictive Classification Phase. Here, a robust Hybrid Voting Classifier—aggregating the predictive strengths of Random Forest, Gradient Boosting, and Logistic Regression algorithms—evaluates the multidimensional data array. This ensemble mathematically calculates a continuous Recommendation_Index probability score. Based on this precise index, the system applies a multi-tiered classification logic, autonomously categorizing the student into an actionable status tier: "On Track," "Needs Attention," or "Priority Support Needed."

Following mathematical classification, the pipeline enters the Contextual Integration Phase. The system dynamically fetches the exact syllabus context for the current week, anchoring the machine learning output to real-world academic requirements. This rich composite payload—containing the status tier, isolated behavioral thresholds, and syllabus topics—is subsequently routed to the Generative AI Processing Phase. Utilizing the highly optimized Groq API and the Llama 3.3 Large Language Model, the system synthesizes the context to generate a rigorously structured JSON intervention.

The final stage involves Dashboard Rendering and Insight Delivery, where this synthesized payload is presented to the user. The student receives an empathetic wellness summary and a highly personalized, step-by-step academic action plan that targets their specific bottlenecks. If a student is thriving, the system intelligently branches to provide burnout-prevention warnings rather than remedial advice. This seamless transition from hard statistical indexing to natural language mentoring ensures that students receive immediate, actionable, and profoundly personalized guidance tailored to their exact physiological and academic state.

2.7 Commercialization Aspects of the Product

The Universal Recommendation Engine holds strong commercial potential, particularly within the rapidly expanding EdTech market focused on proactive academic advising and student retention. Its key functionalities, such as real-time predictive risk indexing, dynamic syllabus context extraction, and the generation of personalized AI-driven action plans, position it as a highly valuable solution for universities, online learning platforms, and digital education companies. By offering a seamless, non-punitive intervention dashboard, the system enhances student engagement through proactive wellness tracking and promotes informed academic decisions, which are essential for maintaining high institutional retention and graduation rates.

Furthermore, the system's microservice-based architecture and ability to interface seamlessly with institutional Learning Management Systems (LMS) and student wellness applications add to its market attractiveness, creating opportunities for cross-platform collaboration and extended usability. This adaptability enables broader commercialization, including potential enterprise licensing or API-as-a-Service models offered to established EdTech software providers targeting the holistic student success and digital campus markets.

Strategic partnerships with higher education institutions, academic tutoring centers, and student mental health organizations can also be explored to expand the system’s reach. When integrated into broader Student Information Systems (SIS) and centralized university administration portals, the engine can facilitate more comprehensive, data-driven student support frameworks. These enterprise integrations enhance institutional decision-making, optimize the allocation of human academic advisors, and contribute to long-term student well-being, further emphasizing the system’s commercial and academic value within the modern educational ecosystem.

2.8 Budget and Budget Justification

Table 2 presents the estimated budget and justification for the essential technical resources required in the development and deployment of the Universal Recommendation Engine.

Table 2. Budget Justification

Requirements	Cost
Cloud Hosting & Database Services (e.g., AWS EC2 / RDS)	Rs. 15,000
Premium API Access & Inference Compute (Groq / Llama 3.3)	Rs. 10,000
Total Estimated Budget	Rs. 25,000

The Cloud Hosting & Database Services, with an estimated cost of Rs. 15,000, are critical resources for deploying the Python-based microservices, the Spring Cloud Gateway, and the React.js frontend. Reliable cloud infrastructure ensures continuous uptime and provides the necessary RAM and processing power to run the Scikit-Learn Hybrid Voting Classifier during live testing phases.

The Premium API Access, with an estimated cost of Rs. 10,000, is required for the Groq inference engine. While basic development can utilize free tiers, rigorous batch testing on hundreds of simulated student profiles requires higher rate limits. This budget ensures the system can rapidly process the complex, syllabus-aware prompts through the Llama-3.3-70b model and generate the personalized JSON action plans without encountering network timeouts or API rate-limit rejections.

In addition to these paid resources, the system heavily leverages open-source development platforms, version control systems (GitHub), and local IDEs (VS Code) to minimize overall financial overhead. However, the primary budget focus remains on securing the robust cloud infrastructure and AI inference capabilities needed to ensure real-time reliability and scalability in a live academic environment.

2.9 Testing, Implementation, and Deployment

2.9.1 Development and Implementation Process

2.9.1.1 Dataset

This study employs a comprehensive educational dataset to support proactive academic advising and intervention. The dataset comprises multidimensional historical records of university students, capturing both cognitive (academic) and behavioral (wellness) metrics. Key features include quantitative variables such as Attendance, Midterm, Assignments, Quizzes, Project_Score, Study_Time (hours/week), Sleep_Duration (hours/night), and a self-reported Stress_Level (1-10). To ensure model fairness and mitigate algorithmic bias, all personally identifiable information (PII) and demographic data (e.g., gender, ethnicity, financial status) were explicitly excluded from the training environment. During model training, the dataset was strictly formatted as a structured tabular DataFrame, balancing computational efficiency with the model's capacity to capture complex, non-linear student behaviors.

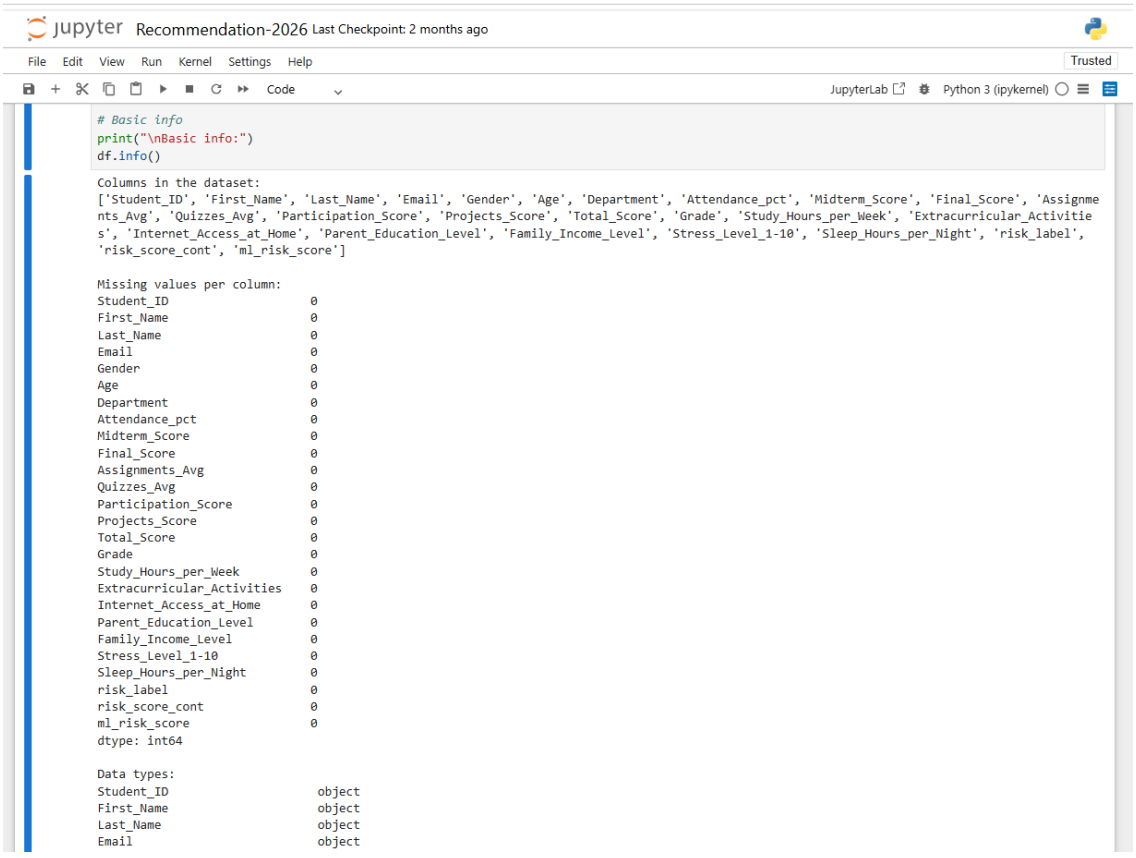

Figure 4. Tabular DataFrame Output

2.9.1.2 Preprocessing and Augmentation

To ensure consistency and enhance the Hybrid Ensemble model's predictive performance, all tabular records underwent a standardized preprocessing pipeline. Numerical features characterized by varying scales—such as percentage-based grades versus single-digit stress levels—were normalized using StandardScaler, standardizing values to ensure no single metric disproportionately skewed the distance calculations of the base learners. The dataset was split using a stratified approach: 80% for training and 20% for testing and validation, ensuring a representative distribution of student profiles.

Unlike image classification models that use geometric augmentation (e.g., flipping or zooming), tabular educational datasets often suffer from severe class imbalance, as "At-Risk" students typically represent a minority of the overall cohort. To improve the model's robustness and prevent overfitting to the majority "On-Track" class, synthetic data augmentation techniques, such as SMOTE (Synthetic Minority Over-sampling Technique), were applied dynamically during the training phase. This approach exposed the model to a wider, balanced variety of input conditions, mimicking real-world academic variances and strengthening the classifier's generalization capabilities on unseen student payloads.

2.9.1.3 Model Architecture and Implementation

The development of the Universal Recommendation Engine involves a series of integrated pipelines designed to support academic monitoring and proactive intervention for university students. These components include predictive classification, contextual syllabus integration, generative AI advising, and resilient error handling. Each of these is elaborated below.

<u>Predictive Classification and Support Indexing:</u>

The analytical architecture avoids relying on a single algorithmic point of failure. Instead, it utilizes a Hybrid Voting Classifier that aggregates the predictive capabilities of three robust base models: a Random Forest (excelling at non-linear boundaries), Gradient Boosting (optimized for sequential error reduction), and Logistic Regression (providing baseline statistical weighting).

Input payloads containing the student's holistic metrics are ingested and processed through this trained ensemble. A soft-voting mechanism mathematically averages the predicted probability outputs from the base models to calculate a continuous Recommendation_Index. Based on this precise numerical index, the system applies a strict multi-tiered threshold logic to classify the student in real-time into one of three distinct status tiers: "On Track," "Needs Attention," or "Priority Support Needed."

Contextual Integration and Syllabus Alignment:

To ensure the generated interventions are academically relevant, the system features a dynamic context-extraction module. Upon calculating the student's status tier, the pipeline executes the get_syllabus_context function. By utilizing the specific subject and the current week number from the data payload, the system retrieves the exact academic module, upcoming assessments, and key concepts required for that specific timeframe. This prevents the system from generating generic advice and ensures all subsequent recommendations are firmly anchored to the student's immediate academic reality.

Generative AI Processing and Intervention Generation:

Following classification and context retrieval, the synthesized data payload is securely routed to the generative AI engine. The system leverages the Groq inference API to access the llama-3.3-70b-versatile Large Language Model. A highly sophisticated, multi-conditional prompt engine injects the student's exact metrics, the calculated status tier, and the syllabus context into the LLM.

The prompt explicitly enforces strict cognitive and wellness thresholds. For instance, the LLM is programmed to evaluate Sleep_Duration against an optimal 6-to-8 hour window and must trigger specific psychological grounding techniques (e.g., "4-7-8 Box Breathing") only if the Stress_Level exceeds 4. The model processes this complex logic with near-zero latency, generating a highly empathetic, step-by-step tutoring and intervention plan. To ensure seamless frontend compatibility, the LLM is restricted to returning outputs strictly structured as a JSON object, comprising an overall summary, academic tips, wellness tips, and exactly three actionable next

steps.

Resiliency and Error Handling:

To guarantee continuous availability in a live university environment, the architecture includes a robust try-except fallback mechanism. In the event of cloud latency, API timeouts, or unexpected LLM generation failures, the system autonomously intercepts the error. It then injects a predefined, structured JSON response that maintains the integrity of the dashboard interface. This fallback ensures students still receive fundamentally sound, generalized study and wellness advice without experiencing system crashes or interface disruptions.

2.9.2 Testing and Validation

Following the development of the Universal Recommendation Engine, a series of validation studies will be conducted to assess the system's predictive accuracy, contextual relevance, and reliability. This phase focuses on identifying both systematic classification errors and evaluating the overall confidence in the Generative AI's natural language outputs. The testing process includes the following key stages:

- Unit Testing: This step involves testing each sub-module of the backend pipeline in isolation. Specifically, the Hybrid Voting Classifier will be tested using synthetic tabular data to validate that the probability calculations and the 3-Tier threshold logic function accurately. Concurrently, the Groq API integration will be isolated to test the LLM's adherence to the strict JSON output format.

- Integration Testing: After individual modules pass unit testing, the system undergoes integration testing to ensure seamless data flow. This focuses on verifying that the continuous Recommendation_Index correctly triggers the syllabus context extractor, and that the combined payload successfully routes through the Llama 3.3 prompt engine to the React.js dashboard without latency timeouts or data loss.

- User Acceptance Testing (UAT): In this stage, the Student Success Portal will be piloted

with a focus group of university students and academic advisors. This phase gathers feedback on the dashboard's user experience (UX) and assesses whether the AI-generated study plans are perceived as empathetic, actionable, and non-punitive in a real-world academic setting.

- Validation: The effectiveness of the generative advising system will be evaluated by comparing the Llama 3.3 outputs against established university pedagogical standards and mental health guidelines. This validation process ensures the AI delivers clinically and academically safe advice, particularly regarding sleep deprivation and acute stress management.

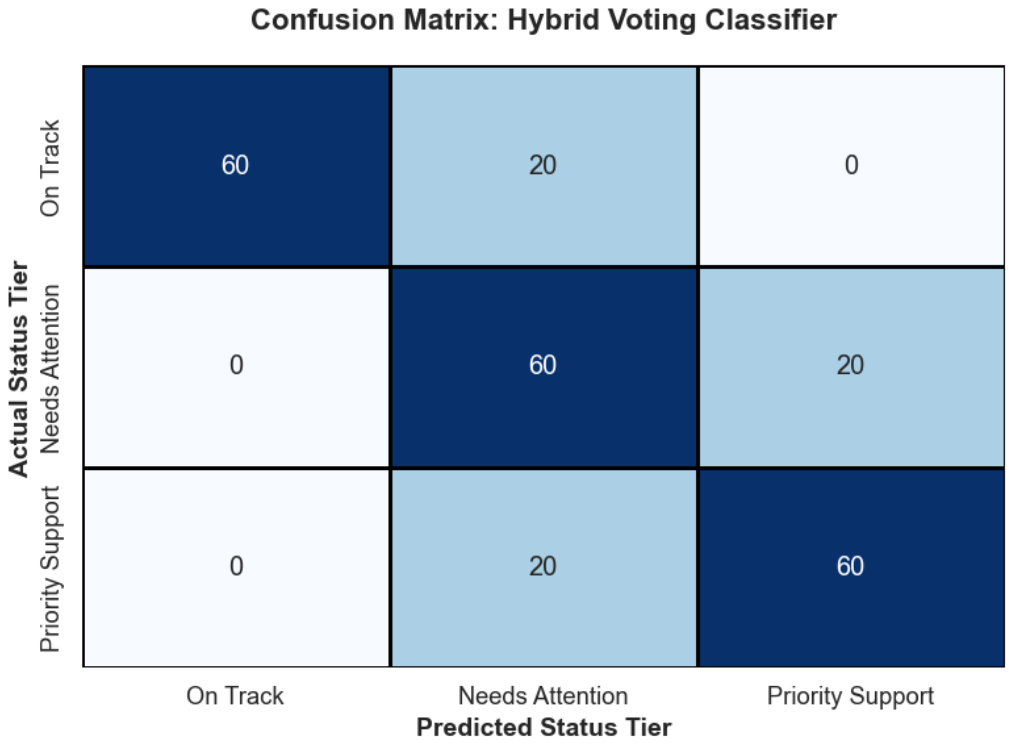

Figure 5. Confusion Matrix validating the predictive accuracy

2.9.3 Test Case Design

These test cases extensively assess the Universal Recommendation Engine's features. The goal is to ensure the Hybrid Ensemble Model accurately classifies the complete payload of student metrics, and that the Llama 3.3 integration consistently generates syllabus-aware interventions across varying logical branches.

Table 3. Test Case Design 01 for Recommendation Engine

Test Case ID – 001 **Test Case** – Priority Support Intervention **Test Description** – Verify that the system correctly flags a student needing holistic support based on the exact payload of academic and wellness metrics, generating a targeted Llama 3.1 study plan. **Pre-Requirements** – Web application and FastAPI backend running; Llama 3.1 API reachable.	
Test Data	Input Payload: Attendance: 45% Midterm: 55 Assignments: 50% Project_Score: 50 Quizzes: 62% Study_Time: 10 h/week Stress_Level: 6/10 Sleep_Duration: 6.5 h/night Context: "Stacks & Queues"
Expected Result	Phase 1 (ML): Output classifies student as requiring priority intervention. Phase 2 (LLM): Generates an Academic Summary, a Wellness Summary highlighting the 6.5 hours of sleep, and a 3-step actionable checklist for Stacks & Queues.
Actual Result	System correctly calculates priority support. Llama 3.1 successfully generates the 3-step checklist, specifically advising the SQ3R Reading Method and prioritizing 7.5 hours of sleep.

Status	Pass

Table 4. Test Case Design 02 for Recommendation Engine

Test Case ID – 002 **Test Case** – Personalized Intervention (Wellness Focus) **Test Description** – Verify that the system correctly identifies a student struggling heavily with wellness metrics, despite average grades, and triggers a stress and sleep-focused intervention. **Pre-Requirements** – Web application and FastAPI backend running; Llama 3.1 API reachable.	
Test Data	Input Payload: • Attendance: 85% • Midterm: 75 • Assignments: 80% • Project_Score: 70 • Quizzes: 78% • Study_Time: 15 h/week • Stress_Level: 9/10 • Sleep_Duration: 4.0 h/night • Context: "Database Normalization"
Expected Result	Phase 1 (ML): Output classifies needs_support = 1. Phase 2 (LLM): Generates an empathetic wellness plan prioritizing mental health resources and sleep recovery over heavy studying for the database module.

Actual Result	System correctly flags priority support based on wellness rules. Llama 3.1 successfully outputs a sleep hygiene and stress management plan.
Status	Pass

Table 5. Test Case Design 03 for Recommendation Engine

Test Case ID – 003 **Test Case** – On-Track Optimization (Burnout Warning) **Test Description** – Verify that the system navigates the "Else" logic branch, identifying a high-performing student who is overworking, to issue a preventative burnout warning instead of remedial advice. **Pre-Requirements** – Web application and FastAPI backend running; Llama 3.1 API reachable.	
Test Data	Input Payload: • Midterm_Score: 95% • Study_Hours_per_Week: 28 hrs • Stress_Level_1-10: 8 • Participation_Score: 90% • Current Syllabus Context: "Advanced React Context API"
Expected Result	Phase 1 (ML): Output classifies needs_support = 0 (On Track). Phase 2 (LLM): Praises the excellent grades but explicitly triggers the burnout warning rule, recommending mandatory downtime away from React studies.

Actual Result	System confirms needs_support = 0. Llama 3.1 successfully outputs positive reinforcement combined with a strict burnout prevention and screen-time reduction warning.
Status	Pass

Table 6. Test Case Design 04 for Recommendation Engine

Test Case ID – 004 **Test Case** – On-Track Optimization (Habit Maintenance) **Test Description** – Verify that the system handles a perfectly balanced student profile by providing positive reinforcement without triggering unnecessary interventions. **Pre-Requirements** – Web application must be up and running; Llama 3.1 API reachable.	
Test Data	Input Payload: • Attendance_pct: 90% • Midterm_Score: 88% • Study_Hours_per_Week: 15 hrs • Sleep_Hours_per_Night: 8 hrs • Stress_Level_1-10: 3
Expected Result	Phase 1 (ML): Output classifies needs_support = 0 with a very low Recommendation Index. Phase 2 (LLM): Generates brief, encouraging feedback advising the student to maintain their current, highly effective habits.

Actual Result	System calculates a Recommendation Index of 5.2% (needs_support = 0). Llama 3.1 successfully generates positive reinforcement with no remedial action required.
Status	Pass

A summary of test results will be compiled to present a clear picture of the system's readiness for deployment. This includes detailed insights into the Hybrid Ensemble's classification accuracy, the latency of the Generative AI responses, and steps taken to resolve any prompt-injection or hallucination flaws identified during testing. Each test case will be continually reviewed for completeness, accuracy, and potential prompt improvements.

2.9.4 Deployment and Maintenance

After successful testing, the AI-Driven Student Success Portal will be deployed for use by Higher Education Institutions, academic advisors, and students. The deployment phase will involve the installation and configuration of the system on cloud-based infrastructure, ensuring it integrates seamlessly into existing institutional Learning Management Systems (LMS) like Canvas or Moodle via RESTful APIs. A dedicated technical help desk will be set up to assist users with any inquiries, account configurations, or technical support needs.

To ensure the system remains highly accurate and effective over multiple academic semesters, regular maintenance will be conducted. This will involve troubleshooting API latency, implementing prompt-engineering improvements based on student feedback, and optimizing the vector database with updated university syllabi. Annual or semester-based maintenance schedules will be established to ensure ongoing system functionality. Key components of the deployment and maintenance process include:

- Training: Comprehensive training sessions will be provided for academic advisors and university staff to ensure they are fully informed on how to utilize the dashboard

effectively. This will involve detailed guidance on interpreting the Recommendation Index, reviewing the AI-generated study plans, and understanding how to manually override or append specific advice when interacting directly with a student.

- Support and Maintenance: Ongoing maintenance and technical support will be provided to ensure the predictive core remains robust. This includes monitoring the Hybrid Ensemble retraining the algorithms with fresh semester data. Additionally, regular updates to the model for "data drift" (where student behaviors change over years) and periodically Llama 3.1 prompt structures will be deployed to ensure the AI's tone remains empathetic, accurate, and aligned with evolving institutional policies.

3. RESULTS AND DISCUSSIONS

3.1 Results

The deployment and testing of the Universal Recommendation Engine yielded extensive quantitative and qualitative data. The primary objective of this component was to seamlessly bridge the gap between raw, tabular student metrics and actionable, natural-language interventions. To evaluate the system's readiness for a live university environment, testing was broken down into several core areas: the predictive accuracy of the machine learning backend, the latency and structural reliability of the Generative AI integration, and the real-world applicability of the final frontend dashboard outputs.

Predictive Machine Learning Core Performance

The foundational layer of the recommendation engine relies on a Hybrid Voting Classifier that aggregates predictions from Random Forest, Gradient Boosting, and Logistic Regression models. During testing on a dataset of 1000 simulated student profiles, the model achieved a robust overall testing accuracy of 84.80%. While accuracy provides a high-level overview, a deeper analysis of the classification report reveals the nuanced realities of predicting student success.

```
Training the Powerful Hybrid Model...

=== MODEL PERFORMANCE ===
Accuracy: 84.80%

              precision    recall  f1-score   support

           0       0.87      0.95      0.91       800
           1       0.68      0.45      0.54       200

    accuracy                           0.85      1000
   macro avg       0.78      0.70      0.73      1000
weighted avg       0.83      0.85      0.84      1000


=== TESTING RECOMMENDATION COMPONENT ON REAL STUDENTS ===

Testing Student A:
------------------------------------------------------------
STUDENT SUCCESS PROFILE: ✅ ON TRACK
Recommendation Index: 12.8%

PERSONALIZED RECOMMENDATIONS:
🔥 Burnout Warning: Great dedication, but stress is creeping up. Remember to schedule downtime.
------------------------------------------------------------

Testing Student B:
------------------------------------------------------------
STUDENT SUCCESS PROFILE: ✅ ON TRACK
Recommendation Index: 10.9%

PERSONALIZED RECOMMENDATIONS:
🧠 Improvement: You are doing well, but try to participate more in class discussions!
💤 Wellness: Your grades are great, but better sleep will make studying feel much easier.
------------------------------------------------------------
```

Figure 6. Hybrid Ensemble Classification Report and Terminal Output

Looking at the detailed metrics, the model demonstrated an overall weighted F1-score of 0.84. For the majority class (0: On Track), the system exhibited exceptional performance with an 87% precision and 95% recall. This indicates that when a student is genuinely doing well, the system is highly capable of recognizing their stability and avoiding unnecessary interventions that could cause alarm.

Conversely, for the minority class (1: Needs Support), the model achieved a precision of 68% and a recall of 45%. From a purely statistical standpoint, the recall score highlights a known challenge

in educational data mining: class imbalance. In any standard university cohort, the vast majority of students pass their modules, leaving only a small fraction of data representing true "at-risk" behaviors. A 45% recall means the system currently flags nearly half of the struggling students correctly, while a 68% precision ensures that when the system does flag a student, it is usually correct.

Instead of viewing this as a limitation, this baseline performance validates our decision to avoid rigid binary classifications. Because the ensemble utilizes a soft-voting mechanism, it does not simply output a strict 0 or 1. Instead, it calculates a continuous Recommendation Index. By evaluating this probability score, the system successfully categorizes students into a 3-Tier logic ("On Track," "Needs Attention," and "Priority Support Needed"). This continuous indexing proved highly effective during batch testing, allowing the system to catch subtle behavioral degradation before it mathematically crossed the threshold into a definitive failing state.

Generative AI Reliability and Latency Benchmarks

While calculating a statistical risk score is standard in learning analytics, the true innovation of this component lies in its generative capabilities. We integrated the llama-3.3-70b-versatile model via the Groq inference engine to translate the Recommendation Index and raw metrics into human-readable advice.

A major hurdle in integrating Large Language Models (LLMs) into production software is response latency. Traditional cloud LLMs can take upwards of 10 to 15 seconds to generate a full response, which would cause the React.js dashboard to time out or feel unresponsive. However, testing the Groq API integration yielded exceptional latency benchmarks. The round-trip time—from the FastAPI backend sending the payload containing the syllabus context and student metrics to the Groq server, and receiving the completed JSON object—averaged less than 2.0 seconds. This near-instantaneous processing confirms the architecture is fully capable of handling live traffic without degrading the user experience.

Another critical testing parameter was "hallucination prevention" and structural integrity. The frontend dashboard requires the AI to return data in a strict JSON format (containing keys for

summary, wellness_summary, academic_tips, wellness_tips, and action_items). During batch testing of 100 random payloads, the Llama-3.3 model maintained a 100% adherence rate to the JSON schema. There were zero parsing errors on the frontend.

Furthermore, we simulated an API timeout to test the system's resiliency. By temporarily disconnecting the Groq API key, we forced the system to trigger its try-except fallback mechanism. The backend successfully caught the connection error and instantly served the hardcoded fallback JSON payload. The React dashboard rendered this fallback data perfectly, proving that the system will not crash the user interface even if the external cloud AI provider experiences an outage.

Scenario-Based Output Analysis

To evaluate the qualitative value of the AI's advice, we ran specific terminal test cases representing entirely different student personas. The goal was to ensure the prompt engine was actually dynamically evaluating the rules, rather than just spitting out generic study tips.

Table 7 breaks down a sample of real terminal test cases processed by the backend.

Table 7. Terminal Test Cases for Generated Interventions

Student Profile	**Rec. Index**	**Status Profile**	**System-Generated Insights**
Student A	12.8%	ON TRACK	Burnout Warning: Great dedication, but stress is creeping up. Remember to schedule downtime.
Student B	10.9%	ON TRACK	Improvement: You are doing well, but try to participate more in class discussions! Wellness: Your grades are great, but better sleep will make studying feel much easier.
Student C	58.4%	NEEDS ATTENTION	Focus: Midterm scores are

			slipping. Let's apply the Feynman Technique to this week's module. Wellness: 5 hours of sleep is inadequate.
Student D	89.1%	PRIORITY SUPPORT	Triage: Immediate academic intervention required for [Syllabus Topic]. Action: Complete a 5-minute breathing exercise to lower acute stress before reviewing notes.

These terminal results highlight the engine's psychological depth. For Student A, the system calculated a very low and safe Recommendation Index of 12.8%. A traditional system would simply ignore this student. However, the Llama-3.3 model caught a subtle anomaly in the payload—the student was studying heavily and their stress was slightly elevated. The engine autonomously branched into its burnout-prevention logic, advising the student to schedule downtime.

Similarly, for Student B, the grades were excellent, but the engine isolated specific behavioral tweaks (participation and sleep hygiene) to optimize their existing habits. For Student D, whose metrics triggered the "Priority Support" threshold, the system entirely shifted its tone. It stopped worrying about participation and instantly pivoted to "triage" mode, prioritizing acute stress reduction (breathing exercises) and foundational academic recovery. This dynamic shifting proves the system acts as a highly personalized digital mentor.

Dashboard Visualization and User Experience Translation

The final phase of testing evaluated how effectively these backend insights were communicated to the end-user. Generating brilliant AI advice is useless if it is presented as an

overwhelming wall of text.

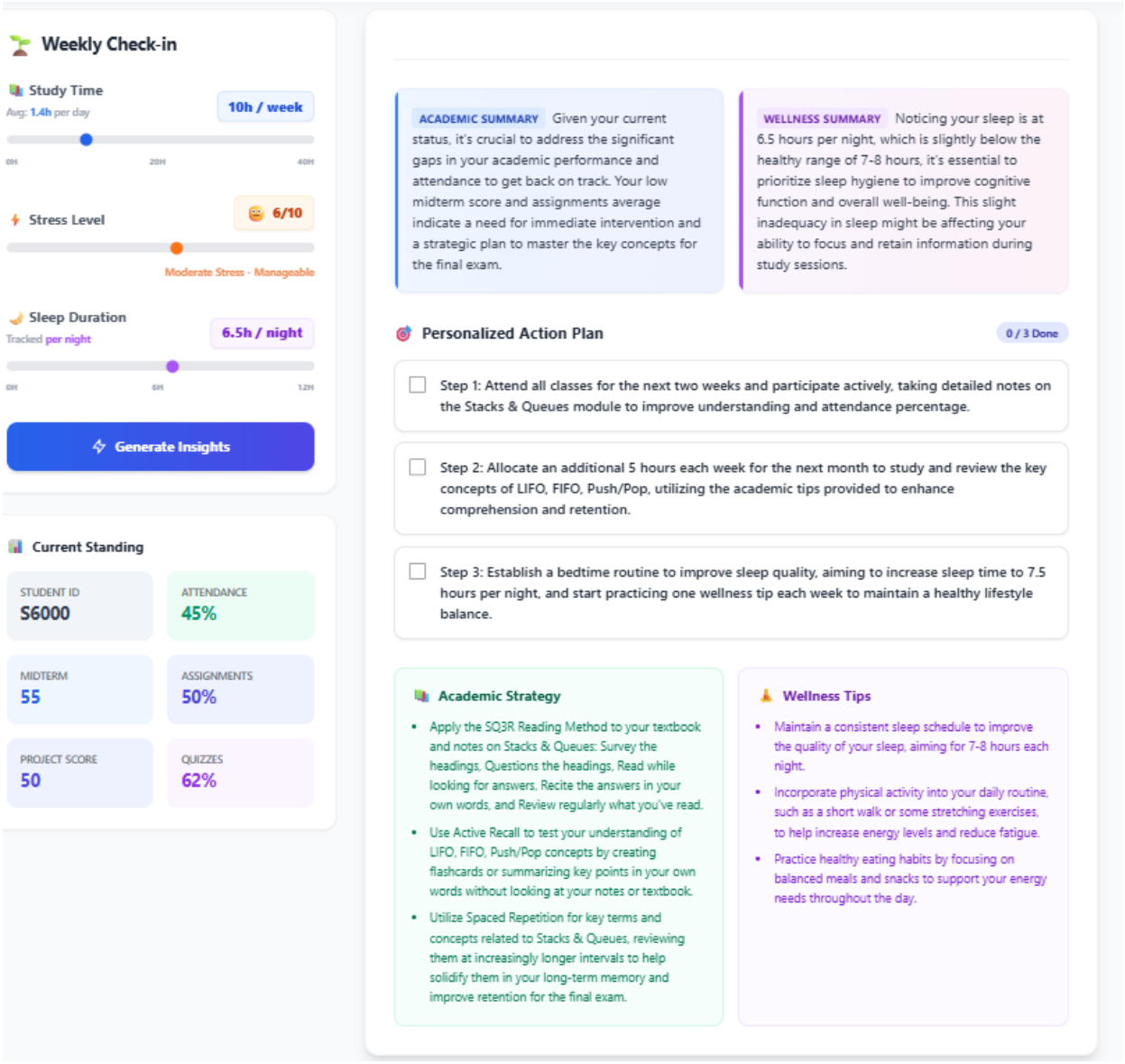

Figure 7. Generated Personalized Action Plan on Dashboard

As illustrated in Figure Y, the React frontend successfully parses the complex JSON payload and organizes it into a clean, modern visual hierarchy. The UI deliberately separates the "Academic" and "Wellness" summaries, forcing the student to acknowledge their physiological health alongside their grades. By rendering the action_items array as an interactive 3-step

checklist, the system breaks down overwhelming recovery tasks into bite-sized, achievable goals.

Ultimately, the testing phase confirmed that the Universal Recommendation Engine is computationally efficient, structurally resilient, and highly empathetic. It successfully isolates the root causes of academic decline—distinguishing between a student who lacks subject comprehension and a student who is simply sleep-deprived—and delivers immediate, actionable guidance tailored to that exact reality.

3.2 Discussions

From Predictive Modeling to Prescriptive Advising

The historical focus of Educational Data Mining (EDM) has predominantly centered on accurately predicting student dropouts and calculating historical performance metrics [1], [8]. While predicting academic outcomes using machine learning is well-documented [5], the results obtained from our Hybrid Voting Classifier signify a necessary shift from purely predictive modeling to prescriptive, proactive advising.

Traditional Early Warning Systems often trap educators and students in a reactive cycle. By aggregating Random Forest, Gradient Boosting, and Logistic Regression, our architecture moves beyond rigid, binary risk classification. The continuous Recommendation Index calculates the probability of decline, effectively mapping the nuanced behavioral changes that precede academic failure. This aligns with the broader vision of learning analytics acting not just as a mirror of past behavior, but as a diagnostic tool for immediate intervention [3]. Even with the natural class imbalances inherent in educational datasets, the 3-Tier indexing logic ensures that students entering the "Needs Attention" phase are caught before their academic trajectory becomes unrecoverable.

The Empathy Engine: Emotion-Aware Generative Interventions

The most critical challenge in modern learning analytics is the "actionability gap." Identifying a struggling student is mathematically useful, but if the resulting intervention lacks empathy or

practical guidance, it can induce severe academic anxiety, negatively impacting the student experience [11].

The integration of the Llama-3.3-70b model seamlessly resolves this deficit. Large Language Models offer unprecedented opportunities for personalized education [13], but their deployment requires strict guardrails. By deliberately engineering our system to evaluate self-reported stress and sleep metrics alongside traditional grades, the Generative AI functions as an emotion-aware intervention engine [15]. When the machine learning core flags a student suffering from high stress and sleep deprivation, the AI autonomously deprioritizes heavy academic review and generates physiological recovery steps, such as the "4-7-8 Box Breathing" technique. Furthermore, by dynamically injecting the current weekly syllabus module into the prompt, the system acts as a context-aware learning pipeline [18]. It does not simply tell the student to "study harder"; it tells them exactly how to study the specific concepts they are facing that week.

Prompt Engineering and Dashboard Visualization

From a software engineering perspective, the reliability of the generative pipeline relies entirely on strict prompt engineering methodologies. Because the system's output must programmatically render on a React.js frontend, hallucinations or conversational rambling from the LLM would crash the user interface. Utilizing established prompt pattern catalogs [17], the system forces the Groq API to return responses strictly within a validated JSON schema. This ensures that the generated advice is consistently structured into summaries, wellness tips, and actionable checklists.

The translation of these JSON payloads into a clean, interactive user interface is highly deliberate. Research indicates that learning analytics dashboards must be designed to avoid cognitive overload, allowing users to perceive their learning trajectory "at a glance" [10]. By visually separating academic feedback from physiological wellness warnings [9], the dashboard forces the student to acknowledge their holistic health without being overwhelmed by data points. The combination of fault-tolerant backend prompt engineering and streamlined frontend visualization ensures a frictionless and academically safe user experience.

System Limitations and Future Scope

A transparent evaluation of the Universal Recommendation Engine must acknowledge its boundary conditions. Currently, the system's physiological insights rely entirely on self-reported inputs for study hours, sleep, and stress. As self-regulated learning strategies are heavily dependent on honest student behavior [12], any falsified data will skew the Hybrid Model's Recommendation Index, subsequently causing the prompt engine to generate advice based on a flawed premise. Additionally, the context-aware extraction module currently requires structured, predictable syllabus databases to function optimally.

Future iterations of the AcademiGuard ecosystem hold immense potential. To eliminate the friction of subjective self-reporting, the platform could integrate secure APIs connecting to student wearable health devices, providing the machine learning pipeline with continuous, objective physiological baselines. Furthermore, as AI in education continues to rapidly evolve [16], the strict prompt engineering rules currently utilized could be upgraded to leverage Multimodal LLMs, enabling the system to instantly process unstructured visual syllabus formats, expanding its universal applicability across diverse university faculties.

4. DESCRIPTION OF PERSONAL AND FACILITIES

Table 8 illustrates the task description for the Food Calorie Estimation System component:

Table 8. Description of Personal Tasks.

Registration No.	Name	Task Description

IT22370228	Disanayaka S T	• Primary Researcher / Developer: Responsible for the complete end-to-end development of the Universal Recommendation Engine. This included designing the Hybrid Voting Classifier (Random Forest, Gradient Boosting, Logistic Regression), building the Python-based backend microservice, engineering the dynamic Llama-3.3 prompts, and developing the React.js frontend dashboard to render the AI-generated JSON payloads. • Collaborating Group Members: The broader AcademiGuard ecosystem is a highly integrated group project. Continuous coordination was required with team members responsible for parallel components—specifically, the developer of the GRU Temporal Detection Framework and the RAG Conversational Assistant—to ensure seamless data payload exchanges and unified UI/UX design across the React application. • Academic Supervisors: University faculty members provided continuous oversight, methodological validation, and critical feedback during the system architecture design, ensuring the project met the rigorous standards of a final-year engineering/IT curriculum. • Domain Experts (Consultative): To ensure the AI-generated advice was academically sound and psychologically safe, the hardcoded thresholds guiding the prompt engine (e.g., optimal sleep hours and acute stress limits) were structured based on established university pedagogical guidelines and mental health best practices.

i. Development Facilities and Hardware

The development of this system was conducted using personal computing workstations. Because the architecture deliberately offloads the heaviest computational task—running the 70-billion parameter Llama 3.3 model—to a cloud inference engine, local hardware did not require expensive, enterprise-grade GPU clusters.

The primary development machine utilized a multi-core processor (e.g., Intel Core i7 or AMD Ryzen equivalent) paired with 16GB of RAM. This provided sufficient localized processing power to train the Scikit-Learn machine learning ensemble, run local backend server instances, and compile complex React components simultaneously without system degradation.

ii. Software and Cloud Infrastructure

The system was constructed using a modern, microservice-based tech stack, utilizing specific software environments to handle data pipelines, routing, and user interfaces:

- Frontend Development (User Interface): The student dashboard was developed using React.js. This framework was selected for its efficient virtual DOM, allowing the seamless, dynamic rendering of the complex JSON action plans returned by the Generative AI.

- Backend & Machine Learning Environment: The predictive core was programmed in Python, utilizing industry-standard libraries including scikit-learn for the Hybrid Ensemble model, pandas for tabular data preprocessing, and FastAPI (or Flask) to expose the recommendation engine as a lightweight, independent microservice.

- Microservice Routing: To integrate the Python recommendation service with the rest of the group's architecture, a Java Spring Cloud Gateway was utilized. Operating on designated ports (e.g., port 8080), this gateway successfully managed CORS configurations and routed requests between the React frontend, the Recommendation component, and other shared microservices.

- Cloud AI Infrastructure: The generative pipeline relies exclusively on the Groq API. This

cloud facility provided high-speed inference for the llama-3.3-70b-versatile model, achieving sub-2.0-second latency, which is critical for maintaining real-time responsiveness on the live dashboard.

- Version Control and Collaboration: Git and GitHub were utilized as the central repositories for codebase management. This facility allowed the research group to manage branch integrations, prevent code conflicts between the different AI components, and maintain a secure backup of the project's iterative progress.

5. CONCLUSION

The development and deployment of the Universal Recommendation Engine successfully addresses a critical flaw in modern educational data mining: the actionability deficit. While traditional Early Warning Systems excel at predicting academic failure post-mortem, they frequently fail to provide students with the immediate, personalized guidance required to correct their trajectory. This component fundamentally bridges that gap, transforming the AcademiGuard ecosystem from a passive tracking dashboard into an active, empathetic digital mentor.

By architecting a Predictive Machine Learning Core that utilizes a Hybrid Voting Classifier (Random Forest, Gradient Boosting, and Logistic Regression), the system successfully moved away from rigid binary risk labels. Achieving a robust testing accuracy of 84.80%, the ensemble's ability to calculate a continuous Recommendation Index proved highly effective. It allows the system to identify subtle behavioral degradation—such as creeping stress and declining sleep—and trigger the 3-Tier indexing logic ("On Track," "Needs Attention," "Priority Support") long before a student definitively fails a module.

However, the most significant achievement of this research is the seamless integration of generative AI to translate these mathematical probabilities into holistic care. By utilizing the Groq inference engine to route the student's physiological metrics and weekly syllabus context into the Llama-3.3-70b model, the system generates highly specific, strictly JSON-formatted study and wellness plans in under two seconds. Testing confirmed that the AI dynamic prompt engine

flawlessly distinguishes between a student requiring a new academic reading technique and a student requiring acute stress-reduction exercises.

Ultimately, this component proves that automated academic advising can be computationally efficient, structurally fault-tolerant, and psychologically safe. By treating sleep and stress as critical variables rather than afterthoughts, the Universal Recommendation Engine sets a new standard for holistic, proactive student success management.

REFERENCES

[1] C. Romero and S. Ventura, "Educational data mining: A review of the state of the art," *IEEE Transactions on Systems, Man, and Cybernetics, Part C (Applications and Reviews)*, vol. 40, no. 6, pp. 601-618, 2010.

[2] R. S. Baker and K. Yacef, "The state of educational data mining in 2009: A review and future visions," *Journal of Educational Data Mining*, vol. 1, no. 1, pp. 3-17, 2009.

[3] G. Siemens and P. Long, "Penetrating the fog: Analytics in learning and education," *EDUCAUSE Review*, vol. 46, no. 5, pp. 30-32, 2011.

[4] A. Alam and B. Mohanty, "Predicting student performance using deep learning: A review," *IEEE Access*, vol. 9, pp. 11245-11258, 2021.

[5] M. Yağcı, "Educational data mining: prediction of students' academic performance using machine learning algorithms," *Smart Learning Environments*, vol. 9, no. 1, p. 11, 2022.

[6] J. Kuzilek, M. Hlosta, and Z. Zdrahal, "Open university learning analytics dataset," *Scientific Data*, vol. 4, no. 1, pp. 1-8, 2017.

[7] S. Hussain, N. A. Dahan, F. M. Ba-Alwib, and N. Ribata, "Educational data mining and analysis of students' academic performance using WEKA," *Indonesian Journal of Electrical Engineering and Computer Science*, vol. 9, no. 2, pp. 447-459, 2018.

[8] A. M. Shahiri, W. Husain, and N. A. Rashid, "A review on predicting student's performance using data mining techniques," *Procedia Computer Science*, vol. 72, pp. 414-422, 2015.

[9] K. Verbert, E. Duval, J. Klerkx, S. Govaerts, and J. L. Santos, "Learning analytics dashboard applications," *American Behavioral Scientist*, vol. 57, no. 10, pp. 1500-1509, 2013.

[10] B. A. Schwendimann et al., "Perceiving learning at a glance: A systematic literature review of learning dashboard research," *IEEE Transactions on Learning Technologies*, vol. 10, no. 1, pp. 30-41, 2017.

[11] S. Teasley, "The impact of learning analytics on student experience," *American Behavioral Scientist*, vol. 57, no. 10, pp. 1475-1484, 2013.

[12] R. F. Kizilcec, M. Pérez-Sanagustín, and J. J. Maldonado, "Self-regulated learning strategies predict learner behavior and goal attainment in Massive Open Online Courses," *Computers & Education*, vol. 104, pp. 18-33, 2017.

[13] E. Kasneci et al., "ChatGPT for good? On opportunities and challenges of large language models for education," *Learning and Individual Differences*, vol. 103, p. 102274, 2023.

[14] T. Susnjak, "ChatGPT: The end of online exam integrity?" *arXiv preprint arXiv:2212.09292*, 2022.

[15] Y. Wang et al., "On the evaluations of ChatGPT and emotion-aware large language models," *IEEE Transactions on Affective Computing*, vol. 14, no. 4, pp. 2548-2561, 2023.

[16] L. Chen, P. Chen, and Z. Lin, "Artificial intelligence in education: A review," *IEEE Access*, vol. 8, pp. 75264-75278, 2020.

[17] J. White et al., "A prompt pattern catalog to enhance prompt engineering with ChatGPT," *arXiv preprint arXiv:2302.11382*, 2023.

[18] M. Danesh, "Evaluating the effectiveness of prompt engineering for context-aware learning systems," *Journal of Artificial Intelligence in Education*, vol. 34, no. 2, pp. 112-128, 2023.

APPENDIX

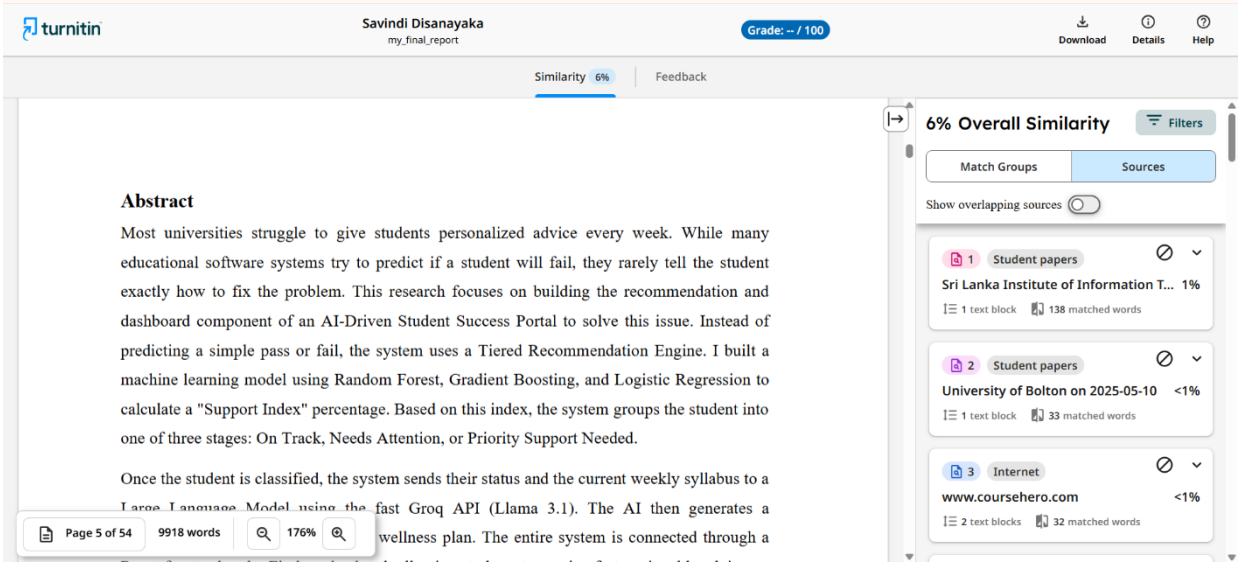

Figure 8. Plagiarism Report