GROUP 1:

EduTrack

COSC 612 / AIT 624

SPRINT 2

EduTrack
Empowering Academic Success Through Insight

Group Members:

Naga Dhanushya Ram Munnanuru
Stephen Aboagye-Ntow
Muhammad Adam
Ayandayo Adeleke
Ravinder Maini

Towson University Fall 2025

TABLE OF CONTENTS

Naga Dhanushya Ram Munnanuru

Towson, MD | nmunnanuru@gmail.com | (704) 910-7707 | Linkedln | GitHub

EDUCATION

Towson University, Master of Science in Computer Science | Towson, MD Jan. 2025 - Dec. 2026

- **GPA:** 3.77/4.0
- **Relevant Coursework:** Database Management Systems, Operating System Principles, Computer Networks.

Geethanjali College of Engineering and Technology, Bachelor of Technology in Computer Science and Engineering – Data Science | Hyderabad, India Aug. 2020 – Aug. 2024

- **GPA:** 3.0/4.0
- **Relevant Coursework:** Data Structures, Artificial Intelligence, Machine Learning, Object Oriented Programming.

SKILLS

Programming Languages: Java, SQL, Python, HTML.
Databases, Technologies & Tools: SQL Server, Anaconda, Jupyter Notebook, Eclipse IDE.
Operating Systems & Software: Linux, Microsoft Office Suite (Excel, PowerPoint, Word), Google Suite.

WORK EXPERIENCE

Trizula Digital Services, Data Analyst Intern | Hyderabad, India Sept. 2024 – Jan. 2025

- Collaborated with a cross-functional team to analyze and interpret large datasets, demonstrating strong communication and teamwork.
- Developed and delivered actionable insights through data visualizations, improving stakeholder decision-making.

IICL, Data Analyst Intern | Hyderabad, India Feb. 2024 – May 2024

- Led data collection and cleansing initiatives, ensuring high-quality inputs for analysis.
- Engaged with clients to gather requirements and tailored deliverables to meet business objectives.

Cantilever Labs, Intern (Brain Tumor Detection) | Hyderabad, India July 2022 – Sept. 2022

- Implemented a CNN-based pipeline for MRI/CT brain tumor detection using Python (Anaconda) with data normalization and augmentation.
- Documented UML diagrams and testing procedures to ensure reproducibility and audit ability.

PROJECTS

ORBITEL - Job Portal System | Towson University Jan. 2025 – May 2025

- Built a full-stack web application using Spring Boot, Spring Security, JPA (Hibernate), Thymeleaf, MySQL, and H2 for academic evaluation.
- Enabled role-based access control for Admin, Employer, and Job Seeker with CRUD operations and secure authentication.

Satellite Image Time Series Analysis for Crop Mapping | Geethanjali College of Engineering and Technology Dec. 2023 – May 2024

- Conducted crop mapping using U-Net and Sentinel-2 data, achieving $\tilde{9}0\%$ accuracy through rigorous data preprocessing.
- Applied problem solving skills to address data imbalances and collaborated with peers to enhance model performance.

Real-Time Image Animation System | Geethanjali College of Engineering and Technology July 2023 – Dec. 2023

- Developed a deep learning-based real-time image animation pipeline targeting video-conferencing, gaming, and AR use-cases.
- Adapted to challenges during implementation, demonstrating problem solving and adaptability in ensuring system efficiency.

Brain Tumor Detection Using ML | Geethanjali College of Engineering and Technology July 2022 – Sept. 2022

- Designed a machine learning model for automated brain tumor detection, reducing reliance on manual diagnostics.
- Communicated findings with clarity to nontechnical stakeholders, showcasing adaptability and collaboration.
- MRI and CT imaging data sets were used, ensuring robust and reliable results.

PUBLICATIONS

- **Co-Author:** *Satellite Image Time Series Analysis for Crop Mapping Using U-Net, Sentinel Dataset*, International Journal of Novel Research and Development (IJNRD), Vol. 9, Issue 4, April 2024.

STEPHEN ABOAGYE-NTOW

Towson, MD | Saboagy1@students.towson.edu

PROJECTS: Understanding and Addressing the Impacts of Wetland Mowing to Facilitate Meeting the Chesapeake Bay Wetland Enhancement Goals - 2024
PUBLICATION: Location, biophysical and agronomic parameters for croplands in northern Ghana

EXPERIENCE

TEACHING ASSISTANT

Computer and Information Science Department | Towson University

2024 – PRESENT

- Evaluate and Grade assignments, quizzes and Labs with timely and constructive feedback
- Proctor Quizzes and Exams

JUNIOR DATA SCIENTIST

Consortium for Digitalization of Climate Data | Accra, Ghana

2023

- Profiled the entire Climate Data Store of Ghana
- Field Data Collection

RESEARCH TECHNOLOGIST

Ghana Space Science and Technology Inst. | Accra, Ghana

2019 - 2023

- Assisted Research Scientists with data analytics and Map production.

EDUCATION

PH. D STUDENT IN INFORMATION TECHNOLOGY

Towson University

JUNE 2024 - PRESENT

- Relevant coursework: Software Engineering and Data Mining

SKILLS AND TOOLS

- **Programming Languages:** R, Python, C++, Bash Scripting
- **Earth Observation and Map production**: QGIS and ArcGIS
- **Statistical and Machine Learning Modeling**

Ayandayo Adeleke

Owings Mills, MD | (703) 459-5202 | ayandayoadeleke@gmail.com | linkedin.com/in/ayandayo-adeleke

SUMMARY

PhD student in Information Technology with research interests in human-computer interaction (HCI). Experienced AWS Solutions Architect with a proven track record in designing and implementing scalable cloud solutions that leverage cloud technologies to solve complex challenges.

EXPERIENCE

Amazon Web Services Arlington, VA

Solution Architect ***Jan 2022 – Jan 2025***

- Collaborated with sales teams to deliver proof-of-concept architectures and technical workshops, contributing to successful deal closures and expanded customer usage.
- Led 100+ technical sessions, architecture reviews, and modernization strategies, improving customer satisfaction across diverse clients.
- Architected a cost-optimized cloud solution that cut storage costs by 50% and improved reliability by 10%, driving AWS service adoption.
- Designed a semantic search solution using AWS services, reducing support queries by 20% and boosting user engagement
- Implemented disaster recovery solutions that met RTO/RPO requirements, ensuring business continuity.
- Delivered technical workshops and led sessions for underserved communities through AWS Impact Accelerator programs.

Ministry of Finance Ibadan, Nigeria

Accounting Assistant ***Jan 2017 – October 2017***

- Reviewed financial records to extract and verify key data for reconciliation reports.
- Ensured accuracy in ledger by identifying discrepancies in bank and financial statements.
- Maintained documentation and data logs to support audits and internal reviews.

LEADERSHIP EXPERIENCE

Vice President, Jerusalem Youth Fellowship 2014 - 2016

- Assisted in the planning and execution of youth events, including social gatherings and outreach programs
- Oversaw program activities, mentoring and volunteer efforts for youth engagement and impact

EDUCATION

Towson University Towson, MD

PhD Information Technology (In Progress)

Bowie State University Bowie, MD

M.S. Management Information Systems, GPA: 3.9/4.0

Ekiti State University Ekiti, Nigeria

B.S Economics

CERTIFICATIONS

AWS Certified Solutions Architect - Professional	Oct 2023
AWS Certified Data Analytics - Specialty Certification	Dec 2023
AWS Developer Associate	Dec 2022
AWS Certified Solutions Architect - Associate	April 2022

SKILLS

Technical Skills: Python, R, SQL, AWS Solutions Architecture, Excel
Soft Skills: Strong Communication, Team Collaboration, Detail Oriented

RAVINDER MAINI

Laurel, MD | (301) 828-0960 | ravindermaini001@gmail.com | linkedin.com/in/raviindermaini

PROFESSIONAL SUMMARY

Experienced Software Test Engineer with proven expertise in designing, testing, and maintaining software systems. Skilled in automation, debugging, and QA process optimization using Selenium and Agile methodologies. Recognized for improving testing efficiency, minimizing downtime, and delivering high-quality, reliable software solutions that enhance user experience and operational performance.

TECHNICAL SKILLS

Languages: Java, Python, HTML, CSS, JavaScript, SQL
Frameworks & Tools: OOP, Flask, MySQL, Selenium, GitHub, JIRA, AWS
Software: Microsoft Office Suite, Visio, Outlook, Excel
Certifications: DHS Trusted Tester, Section 508 Standards
Methodologies: SDLC, STLC, Agile, Scrum, Waterfall
Accessibility Tools: ANDI, Screen Reader

PROFESSIONAL EXPERIENCE

Software Test Engineer | USCIS | Jun 2024 – Present
• Automated manual testing scenarios using Selenium, boosting test coverage and reducing manual effort.
• Partnered with developers and leads to resolve functionality issues and enhance workflow efficiency.
• Designed and maintained Requirement Traceability Matrix (RTM) and detailed test plans ensuring full coverage.
• Conducted regression and integration testing improving stability and overall quality.
• Enhanced UI/UX for improved accessibility and user satisfaction.
• Executed regression test suites validating post-release integrity.

Software Test Engineer Intern | SpecsWorks | May 2023 – Feb 2024
• Applied QA best practices under Agile, Scrum, and Waterfall environments.
• Developed and executed test plans, identifying and documenting defects efficiently.
• Collaborated with developers for debugging and process improvement.
• Enhanced existing application interfaces, improving performance and usability.
• Performed regression and integration testing, ensuring product quality before release.

EDUCATION

Master of Science in Computer Science – Towson University | Expected Dec 2026 | GPA: 4.00/4.00
Bachelor of Science in Computer Science – University of Maryland Global Campus | Aug 2022 | GPA: 3.78/4.00

Muhammad Adam

Place of birth: Nigeria | **Nationality:** Nigerian (Nigeria) | **Phone number:** (+1) 4104991894 (Mobile) | **Email address:** madam2@students.towson.edu |

Address: 42 Solar Cir, 21234, Baltimore, United States (Home)

WORK EXPERIENCE

MATRIX INTERNATIONAL ACADEMY, GOMBE – GOMBE, NIGERIA

MANAGEMENT INFORMATION OFFICER (MIO) – 01/09/2019 – 25/12/2024

CLARITAE VIDERE LIMITED – LAGOS, NIGERIA

MACHINE LEARNING INTERN - REMOTE – 16/12/2021 – 30/03/2023

EDUCATION AND TRAINING

CURRENT Baltimore, United States

PH.D IN INFORMATION TECHNOLOGY (IN VIEW) Towson University

Website https://www.towson.edu/

16/11/2011 – 05/10/2015 Gombe

B.SC (HONS) COMPUTER SCIENCE Gombe State University

Website https://gsu.edu.ng/home/

SKILLS

Computer vision | Deep Learning | Machine Learning | Python | Java

RECOMMENDATIONS

Dr. Qingqing Li Assistant Professor

Academic advisor.

Email qingqingli@towson.edu | **Phone** (+1) 4107044532

PROJECTS

Computerized Hospital Billing System.

A research work submitted to the Department of Mathematics, Faculty of Sciences, Gombe State University in partial fulfillment for the award of the Degree in Computer Science.

PUBLICATIONS

2025

Deep Learning Approaches for Automatic Livestock Detection in UAV Imagery: State-of-the-Art and Future Directions

Adam, M., Song, J., Yu, W., & Li, Q. (2025). Deep Learning Approaches for Automatic Livestock Detection in UAV Imagery: State-of-the-Art and Future Directions. Future Internet, 17(9), 431. https://doi.org/10.3390/fi17090431

1. Planning and Scheduling

Table 1

Assignee	**Email**	**Task**	**Duration**	**Dependency**	**Due Date**
Naga Dhanushya Ram Munnanuru	nmunnan1@students.towson.edu	Revise and Refine System (Improved Problem Statement)	3hrs	Feedback from Sprint A1	10/14/25
Muhammad Adam (coordinator)	madam2@students.towson.edu	Use cases, requirements, diagrams	6hrs	Improved Problem Statement	10/16/25
Ayandayo Adeleke	aadelek7@students.towson.edu	Formatting and Compilation of Report	4hrs	All tasks	10/19/25
Ravinder Maini	rmaini1@students.towson.edu	Planning and Scheduling, Collaboration	3hrs	None	10/13/25
Stephen Aboagye-Ntow	sabaogy1@students.towson.edu	Class diagrams, database specification/analysis	6hrs	Use cases, requirements, diagrams	10/18/25

2. Problem Statement

a) What is your product, on a high level?

Our product is a Student Grade Prediction & Recommendation System, a machine learning–based web application that predicts whether a student will pass or fail a course. The system provides students with their chances of passing as well as tailored advice (e.g., attend classes more often, study longer, decrease stress levels) to improve their academic standing.

b) Whom is it for?

The system exists for:

- **Students:** Perform predictions and get tailored recommendations to improve their academic standing.
- **Teachers/Advisors:** Monitor student performance, identify at-risk students and if possible, provide personalized guidance to help them succeed.
- **Administrators:** Manage user accounts, control system access, and oversee platform operations.

c) What problem does it solve?

The current gradebook systems, which include Blackboard and Canvas show students their actual scores but they lack any system to alert them about their performance or offer help. Students discover their risk status only after completing their final exams which creates limited time for improvement. This system solves the problem by:

- The system enables users to predict student achievement results before the official grades are released.
- Generating recommendations based on the student situation and helping them recognize their potential low points to improve

d) What alternatives are available?

- Learning Management Systems (LMS) like Blackboard and Canvas show performance data but lack predictive analytics and personalized recommendations.
- Blackboard Analytics is a commercial, costly tool that targets administrators, not students.
- No widely available low-cost system provides student-level explainable predictions with recommendations.

e) Why is this project compelling and worth developing?

The system offers:

- **Early Intervention:** helping at-risk students succeed before it's too late.
- **Accessible design:** lightweight web interface (Streamlit GUI).
- **Affordable Deployment:** built entirely with free/open-source tools and deployable on free cloud platforms.
- **Dual Academic & Operational Value:** demonstrates ML research and provides universities with a working tool to improve retention and success.

f) Top-level objectives, differentiators, target customers, and scope.

- **Objectives:**
  - Assess the potential success of a student.
  - Show the chances of success together with essential risk elements.
  - The program should create individualized plans which help students develop their study methods and create effective daily routines.
- **Differentiators:**
  - The system merges machine learning prediction models with rule-based recommendation systems.
  - The system bases its operation on explaining its decision-making steps instead of achieving the highest possible accuracy rates.
  - Users can train models with new data through the system while performing batch predictions from imported CSV files.
- **Target Customers:** University students, advisors, and instructors.
- **Scope:**
  - Cloud-hosted web application with a web interface.
  - The system requires student data entry functions, CRUD operations, and an ML-based prediction system.
  - Dashboards for students and advisors.
  - Messaging feature between students and their teachers.

g) What are the competitors, and what is novel in your approach?

- The competition includes both Learning Management System dashboards (Canvas and Blackboard) and commercial analytics platforms.
- **Novelty:**
  - The system integrates explainable AI technology, which reveals the specific reasons behind its failure prediction for students.
  - The system implements low-cost deployment through its use of open-source frameworks, including scikit-learn, Streamlit, and pandas.
  - Provides personalized recommendations that can be customized by advisors.

h) Can the system be built with available resources and technology?

Yes. The system operates by:

- **Front-end:** Streamlit
- **Back-end / ML Engine:** Python 3.10+, scikit-learn, pandas, NumPy, joblib.
- **Database:** SQLite
- **Hosting & Deployment:** Utilize Hugging Face Spaces or Streamlit Community Cloud for early versions, and Docker + Render/Railway for scalable deployment.
- **Baseline Models:** Decision Trees, expandable to Deep Learning architectures.

i) What is interesting about this project from a technical point of view?

- The high-performance ML models achieve the following results: Accuracy ~92.8%, F1 ~94.1%, ROC-AUC ~98.6%.
- Streamlit GUI provides users with an interactive two-column interface that enables real-time prediction functionality.

- The Recommendation Engine follows two rules which state that students who attend less than 75% of classes need to improve their attendance and students who experience stress levels above 7 should receive stress management strategies.
- The Explainable AI system shows which specific features (attendance, assignments, study hours) affect the prediction results.
- Cloud-ready architecture exists as a deployable system that uses Docker and open-source hosting platforms.

3. System Requirement

3.1. Use Cases

Use Case 1: Manage User Accounts

Actors: Admin

Description: The Admin can create new user accounts for Teachers and Students, update existing user details, or remove user accounts. This ensures only authorized users have access and keeps the system's user directory up-to-date.

Alternate Path: If the Admin attempts to add a user that already exists (e.g., duplicate email or ID), the system shows an error and prompts to enter a unique identifier. For updates, if invalid or incomplete data is provided, the system rejects the changes and requests correct input. In the case of deletions, if the user account is linked to other records (e.g. a teacher with classes assigned), the system will warn the Admin and may restrict deletion until those links are handled.

Pre-condition: The Admin is logged into the system with administrator privileges.

Use Case 2: Record Student Performance

Actors: Teacher

Description: The Teacher enters or updates students' performance data into the system. This use case allows teachers to maintain an up-to-date record of each student's academic performance metrics.

Alternate Path: If the Teacher inputs data in an incorrect format or leaves required fields blank, the system will display validation errors and not save the record. The Teacher can then correct the input and resubmit. In cases where a student's record for a particular exam or term already exists, the system will update that record instead of creating a duplicate.

Pre-condition: The Teacher is authenticated and has the appropriate permissions to manage performance data (typically for students in their class or subject).

Use Case 3: Predict Student Performance

Actors: Teacher

Description: The Teacher triggers the system's predictive model (a trained machine learning model) to forecast a student's future performance or risk level. For example, the Teacher can select a student and run the model to predict whether the student is likely to excel, pass, or be at risk of failing based on the data in their Student Record.

Alternate Path: If the student's performance data is incomplete or the model cannot generate a reliable prediction (e.g., due to missing values or the model not being trained on certain new data), the system will inform the Teacher that the prediction is unavailable or not confident. The Teacher may need to ensure all necessary data (grades, attendance, etc.) are recorded before retrying.

Pre-condition: The system has a trained machine learning prediction model available, and the Teacher is logged in with access to the prediction feature. Relevant student performance data must already be recorded in the system.

Use Case 4: Provide Feedback

Actors: Teacher

Description: The Teacher provides feedback or improvement suggestions to a student based on their performance. For instance, if a student's predicted performance is poor or their recorded grades are low, the Teacher can submit a feedback note or recommendation (such as advising extra tutoring or noting specific areas to improve) which the student can later view.

Alternate Path: If the Teacher submits feedback without selecting a valid target student or leaves the feedback message blank, the system will prompt for the missing information. In case the system supports sending notifications, an alternate flow could include notifying the student once feedback is submitted.

Pre-condition: The Teacher is logged in and has access to students' records for providing feedback. A performance record for the student should exist to contextualize the feedback.

Use case 5: View Personal Performance

Actors: Student

Description: The Student views their own performance data and related information through the system. This includes seeing their grades, attendance, overall progress, as well as any predictions of performance and feedback given by Teachers. The purpose is to keep students informed about their academic standing.

Alternate Path: If the student has no performance data recorded yet (for example, a new student or no grades entered for the term), the system will display a message indicating that no performance records are available. If the Student tries to access another student's data (an unauthorized action), the system will deny access.

Pre-condition: The Student is authenticated in the system. Their performance records (grades, etc.) and any teacher feedback must exist in the system for those to be viewable.

Use Case 6: Generate Performance Report

Actors: Admin

Description: The Admin generates an overall performance report for a class, grade level, or the entire school. This report aggregates student performance data (average grades, pass/fail rates, etc.) and can highlight trends or identify students at risk. It helps administrators and educators assess the effectiveness of instructional programs and allocate resources.

Alternate Path: If there is insufficient data in the system (e.g., no grades have been entered for the term), the system will produce an empty or partial report and inform the Admin that data is missing. In some cases, the Admin can select different parameters (such as date range, specific classes) for the report; an alternate path would handle invalid or out-of-range parameters by prompting the Admin to adjust the report criteria.

Pre-condition: The Admin is logged in. A significant amount of student performance data from teachers should be available in the system for the report generation (otherwise the report will have minimal content).

3.2. Requirements

Requirement 1: Manage User Accounts

Use Case Name: Manage User Accounts

Introduction: This requirement specifies the system's capability to allow an administrator to manage user accounts. The Admin should be able to create new accounts for teachers and students, edit existing account information, or remove accounts as needed. This ensures proper access control and user management for the application.

Inputs: The Admin provides user details such as name, email, role (Teacher or Student), and initial login credentials when creating a new account. For updating an account, the Admin inputs the modified details (e.g., updated email or name). To delete an account, the Admin specifies the target user (e.g., by selecting from a list of users or entering an ID).

Requirement Description: The system **shall** present an account management interface to Admin users. Upon *creating* a new account, the system shall validate that all required fields are provided and that the username/email is unique. If valid, the system stores the new user in the database with the specified role. For *updates*, the system shall allow the Admin to change permissible fields (e.g., update a typo in the name or reset a password) and then save the changes to the database. For *deletions*, the system shall remove the user's record from the database (after confirming the action) and also remove or reassign any records that are dependent on that user (for example, student performance records or feedback entries linked to a deleted user should be handled appropriately). The account management functions are restricted to Admin users only; the system shall enforce authorization checks so that non-admins cannot perform these actions.

Outputs: Confirmation messages are displayed for each successful operation (e.g., "User created successfully", "Account updated", "User deleted"). In case of errors (such as duplicate email, validation failure, or unauthorized attempt), the system shows an error message explaining the issue (e.g., "Email already in use, please choose another"). After changes, the updated list of users is visible to the Admin, reflecting any additions or removals.

Requirement 2: Record Student Performance

Use Case Name: Record Student Performance

Introduction: This requirement covers the input and update of student academic data by a Teacher. The system must allow teachers to record various performance metrics for students, such as exam scores, assignment grades, and attendance, which form the student's performance record.

Inputs: The Teacher selects a target student and enters performance details. Inputs can include course/subject identifiers, exam or assignment names, and the scores or grades achieved. The teacher may also input attendance information (e.g., percentage of classes attended) or other performance indicators.

Requirement Description: The system **shall** provide a form or interface where Teachers can enter student performance data. When a Teacher submits this data, the system will validate it (for example, checking that scores are within a valid range, required fields like student ID and score are not blank, etc.). Valid entries are then saved to the student's performance record in the database. If a record for the specified exam/assignment already exists for that student, the system shall update the existing record instead of creating a duplicate. The design should associate the logged-in Teacher's identity with the data entry for accountability (e.g., log which teacher entered or updated the record). The system shall only allow teachers to record data for students that they are assigned to (ensuring one Teacher cannot accidentally overwrite another's records). All transactions should maintain data integrity, for example, if the database update fails for some reason, the system will report a failure and not partially save incorrect data.

Outputs: After submission, the system provides feedback on the operation. On success, a message like "Performance record saved successfully" is shown, and the new data becomes visible in the student's profile/report. The updated performance can trigger recalculation of any aggregates (e.g., average score) which the system may display. On validation error or failure, an error message is shown (e.g., "Score must be a number between 0 and 100" or "Unauthorized: you cannot record data for this student"). The outputs also include the updated performance listings in the UI for the Teacher to review.

Requirement 3: Predict Student Performance

Use Case Name: Predict Student Performance

Introduction: This requirement defines the system's functionality to use a machine learning model to forecast a student's future performance or risk level. A Teacher can request a prediction for a particular student, and the system will return a prediction based on the data available. The goal is to help identify students who might need intervention or to forecast outcomes like final grades.

Inputs: The Teacher selects which student(s) to run the prediction for. This could be done by choosing a student from a list or viewing a student's profile and clicking a "Predict Performance" button. No direct numerical input is needed from the Teacher, as the prediction uses existing data; however, the Teacher's selection (student identifier or class) is an input that tells the system what data to analyze.

Requirement Description: When the Teacher initiates a prediction, the system **shall** gather the relevant data for the selected student from their Student Record (e.g., past grades, attendance, participation, etc.). This data is fed into the pre-trained machine learning model. The system then generates a prediction of performance, for instance; it might predict the student's final grade for the term or the probability of the student passing a course. The requirement is that the prediction model is accessible through the system's interface and can be executed quickly. The system shall present the prediction result to the Teacher in a readable format. The system must ensure that only authorized users (Teachers/Admins) can run predictions, as it uses sensitive student data.

Outputs: The primary output is the prediction result for the student's performance. This could be a category (e.g., "Pass" or "Fail"), a score (like a predicted grade or numeric score), or a probability. The system will display this result on the UI, possibly alongside the student's current data. For example, it might show a colored indicator or a textual message such as "Predicted outcome: Pass (85% confidence)". If the prediction cannot be produced, the output would be an error or notification (e.g., "Prediction unavailable: insufficient data"). All prediction outputs should be clearly labeled as predictions (not actual results) so that users understand it's a forecast. The system may also log that a prediction was generated (for auditing or future improvements).

Requirement 4: Provide Feedback

Use Case Name: Provide Feedback

Introduction: This requirement describes how Teachers can input qualitative feedback into the system for a student. The feedback feature complements numeric performance records and predictions by enabling teachers to communicate suggestions, comments, or recommendations to students through the system.

Inputs: The Teacher selects a student to provide feedback for (usually in the context of that student's performance profile) and enters a textual message. Inputs include the student's identifier (or selecting from a list) and the content of the feedback message. Optionally, the Teacher might categorize the feedback (e.g., positive reinforcement, improvement suggestion, and warning) by selecting a type or priority, although this is an extra feature not explicitly required.

Requirement Description: The system **shall** allow a Teacher to submit a feedback comment tied to a specific student. When submitted, the feedback is stored in the system (in a Feedback record linked to the Teacher and student). The system should timestamp the feedback entry. Only teachers (and possibly admins) can create feedback entries; students can only read feedback, not alter it. The requirement includes validation, such as ensuring that the feedback message is not empty and possibly limiting the length to a reasonable amount. The system should also provide an interface for teachers to review and edit their own submitted feedback (in case of mistakes). Privacy controls might be implied: e.g., one Teacher should generally only see feedback they provided or that which is shared among relevant staff. The feedback feature should be integrated such that when a student views their performance, they can also see the feedback messages.

Outputs: After submission, the system confirms that the feedback has been saved (e.g., "Feedback sent to student"). The new feedback entry becomes visible in the student's view (usually along with the author's name and date). In the Teacher's interface, the submitted feedback might appear in the context of the student's record. If an error occurs (like network/database failure or unauthorized access attempt), an error message is shown (e.g., "Failed to submit feedback, please try again"). If notifications are enabled, the student (and possibly the Admin) will receive a notification alert about the new feedback.

Requirement 5: View Personal Performance

Use Case Name: View Personal Performance

Introduction: This requirement specifies that students can access a personalized view of their academic performance data through the system. The system will serve as a student portal where individuals see their own records, including grades, attendance, any teacher feedback, and predictive information about their performance.

Inputs: The Student initiates this by logging into the system and navigating to a "My Performance" or dashboard section. No manual data input is required from the student's side to view the information. The primary input is the student's identity (taken from their login session), which the system uses to fetch the relevant records.

Requirement Description: Once authenticated, the system **shall** retrieve all performance-related data for the logged-in student. This includes their grades for assignments/exams, attendance percentage, cumulative scores or GPA, and any feedback notes from teachers. The system will compile this into a dashboard or report format for the student. The requirement is that students can only see **their own** data access control is crucial here (a student cannot see other students' records). The system should present the data in an easy-to-understand manner.

Outputs: The output is the student's performance report displayed on the screen. This includes lists of grades, attendance records, and sections for teacher feedback and prediction results.

Requirement 6: Generate Performance Report

Use Case Name: Generate Performance Report

Introduction: This requirement covers the ability of an Admin to produce a comprehensive report of student performance at an aggregate level. The system will compile data across many students to help administrators see overall trends, averages, and identify outliers or at-risk students.

Inputs: The Admin may specify parameters for the report generation. Inputs could include the scope of the report. For a general report, the default might be all students for the current term if no specific filter is given.

Requirement Description: When the Admin requests a report, the system **shall** aggregate relevant data from the database. Depending on the scope, it will gather all students' performance records in that scope. The Admin should be able to save or print this report, so the requirement may include an option to export the report (as PDF or CSV, for instance). Data security is important: only Admin (or authorized management personnel) can generate and view such broad reports, because they contain sensitive information about multiple students.

Outputs: The output is a formatted performance report, displayed on the Admin's screen (and optionally downloadable). Upon successful generation, the report view is the output. If no data is available for the chosen parameters, the report output will explicitly state "No data available for the selected range." In case of errors (like a database timeout if the dataset is huge), the system will output an error message to the Admin (e.g., "Report generation failed, please try again later"). A successful report generation might also be logged in the system for auditing (this is not a user-visible output, but part of system outputs in a broader sense).

3.3. Use Case Diagram

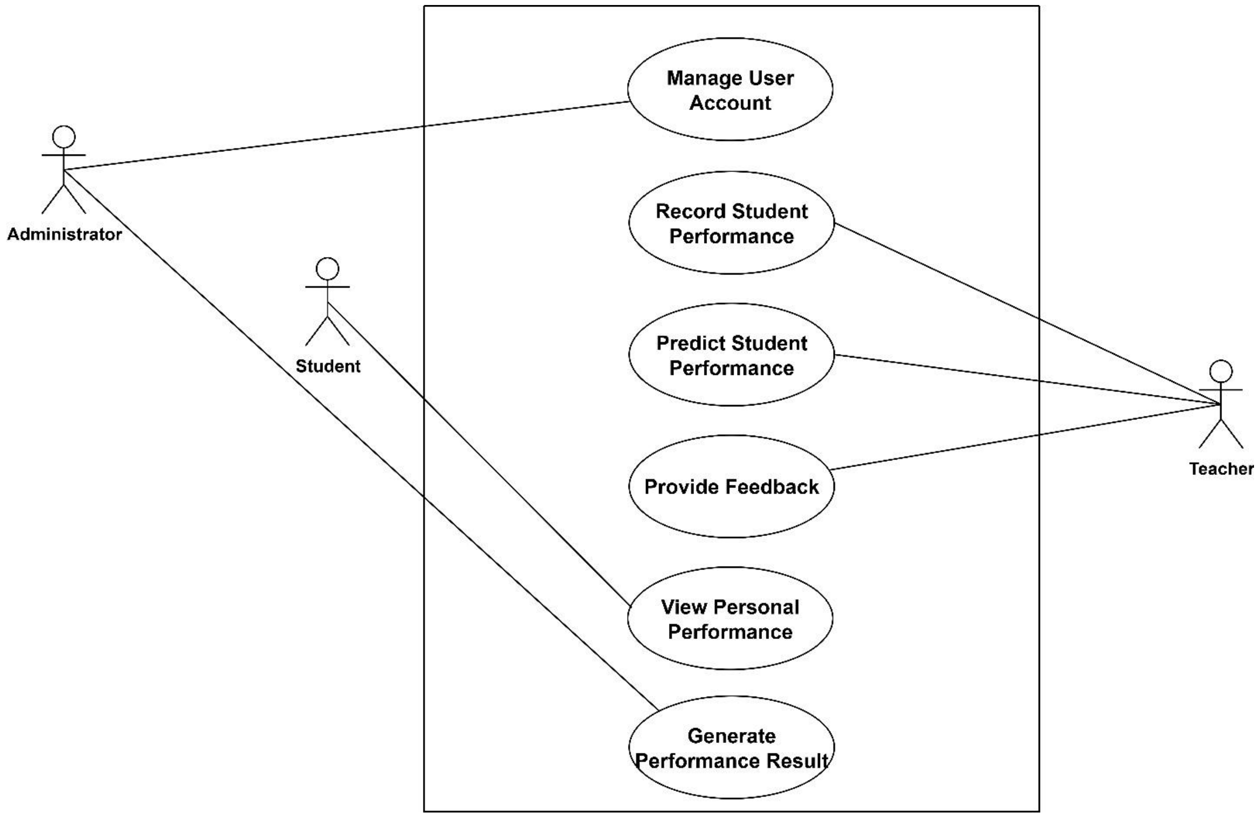

Manage User Account
Record Student Performance
Predict Student Performance
Provide Feedback
View Personal Performance
Generate Performance Result
Administrator
Student
Teacher

Fig. 1 Use Case Diagram for the system

3.4. Context Diagram

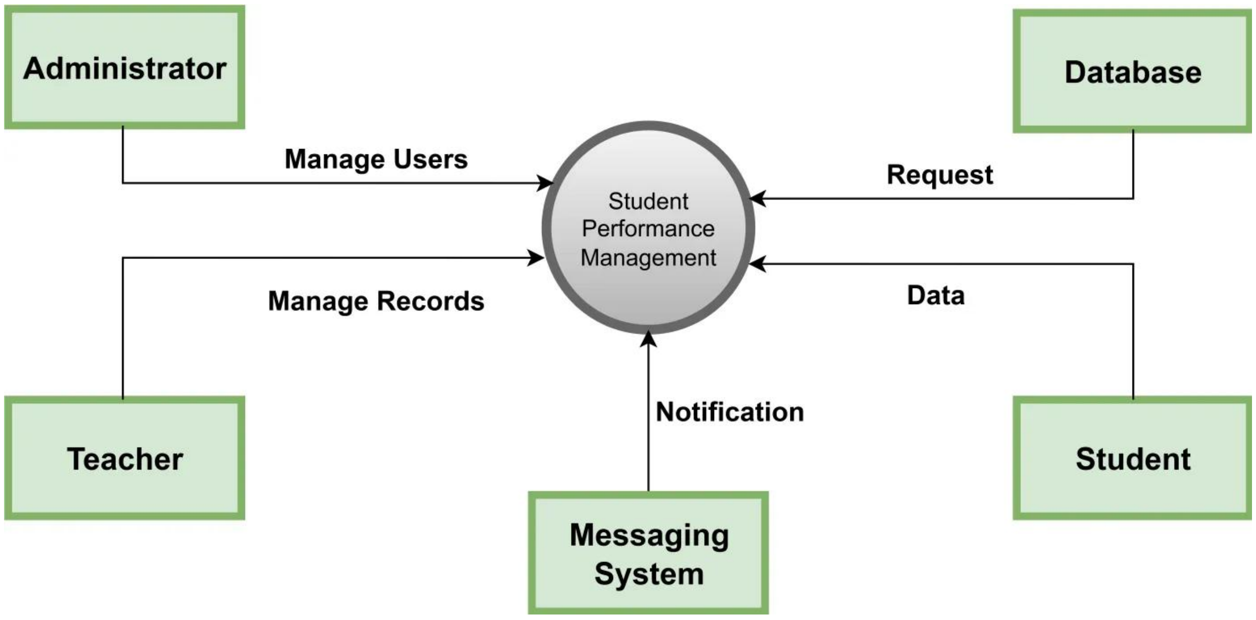


Fig. 2 Context Diagram

4. System Modeling

4.1. Class Diagrams

Class diagrams show the static structure of the system comprised in its classes, attributes, methods and associations amongst objects.

a) Objects

Users (Base for Students, Teachers and Admin)

- Attributes: userID, email, password, name, userRole
- Operations: login(), logout(), updateProfile(), changePassword(), deleteAccout()

Student

- Attributes: studentID, userID, gpa, departmentID, academicStatus
- Operations: submitAcadmeicData(), viewPrediction(), getRecommendation()

Teacher

- Attributes: teacherID, userID, name, email, department
- Operations: viewStudent(), provideFeedback(), customizeRules(), viewPredictionResults(), viewStudentPerformance(), exportStudentReport ()

Administrator

- AdminID, userID, name, email
- Operations: manageUserAccounts(), updateSystemSettings(), createUser()

StudentAcademicData

- Attributes: dataID, studentID, attendance, studyHours, examScores, stressLevel, sleepHours, participation
- Operations: calculateGPA(), compareWithClassAverage(), validateData()

PredictionResults

- Attributes: predictionID, studentID, modelID, predictionStatus, passPercentage, failPercentage
- Operations: PredictPassFail(), getExplanation(), displayProbability(), generateRecommendation(), identifyRiskFactors()

Recommendation

- Attributes: recommendationID, predictionID, studentID, recommendationType, message
- Operations: generateRecommendation(), evaluateThresholds(), snedToStudent(), describepotentialImprovemtn(), custtomizeMessage()

MLModel Class

- Attributes: modelID, modelName, accuracy, modelType

- Operations: trainModel(), predictPassFail(), saveModel(), getFeatureImportance(), retrain()

Message
- Attributes: messageID, senderID, recieverID, subject, content, readStatus, attachmentPath
- Operations: sendMessage(), getContent(), addAttachment(), getAttachment()

Dashboard
- Attributes: DashboardID, userID, dashboardType, customSettings
- Operations: displayMetrics(), generateCharts(), getStudentPresictionSummary(), getPerformanceComaparison()

b) Associations Between Objects

- Users – Students (One to One)
- Users - Teacher (One to One)
- Users - Administrator - (One to One)
- Student - StudentAcademicDAta (One to Many)
- Student – PredictionResults (One to Many)
- PredictionResults – Recommendations (One to Many)
- MLModels – PredictionResults (One to One)
- Users – Messages (Many to Many)
- Users – Dashboards (One to One)

c) Multiplicity

- A user can be a student (1:1)
- A user can be a teacher (1:1)
- A user can be an Administrator (1:1)
- A student can submit multiple academic data (1:N)
- A student can have multiple prediction results over time (1:N)
- An ML model can generate multiple prediction results over time (1:N)
- A prediction can generate or suggest multiple recommendations (1:N)
- A User can have receive multiple messages(1:N)
- A User is associated with a dashboard (1:1)

d) Attributes of objects

Each class comprises attributes representing the data it holds. Example:

- Student: name, studentID, userID, gpa, departmentID, academicStatus
- Administrator: adminID, name, email, role
- StudentAcademicData: dataID, studentID, attendance, studyHours, examScores, stressLevel

- recommendation: recommendationID, predictionID, studentID, recommendationType, message

e) Operations/Methods defined on objects

Each class defines operations that can be performed on its instances. Example:

- Student: enterData(), viewPrediction(), viewRecommendation()
- Administrator: manageUsers(), updateSystemSettings()
- Teacher/Professor/Instructor: viewStudent(), provideFeedback(), customizeRules()
- PredictionResults: PredictPassFail(), getExplanation(), displayProbability(), generateRecommendation(), identifyRiskFactors()

f) System Class Diagram

The Class diagram shows the relationship, constraints between entities involved in the EduTrack grade and recommendation system.

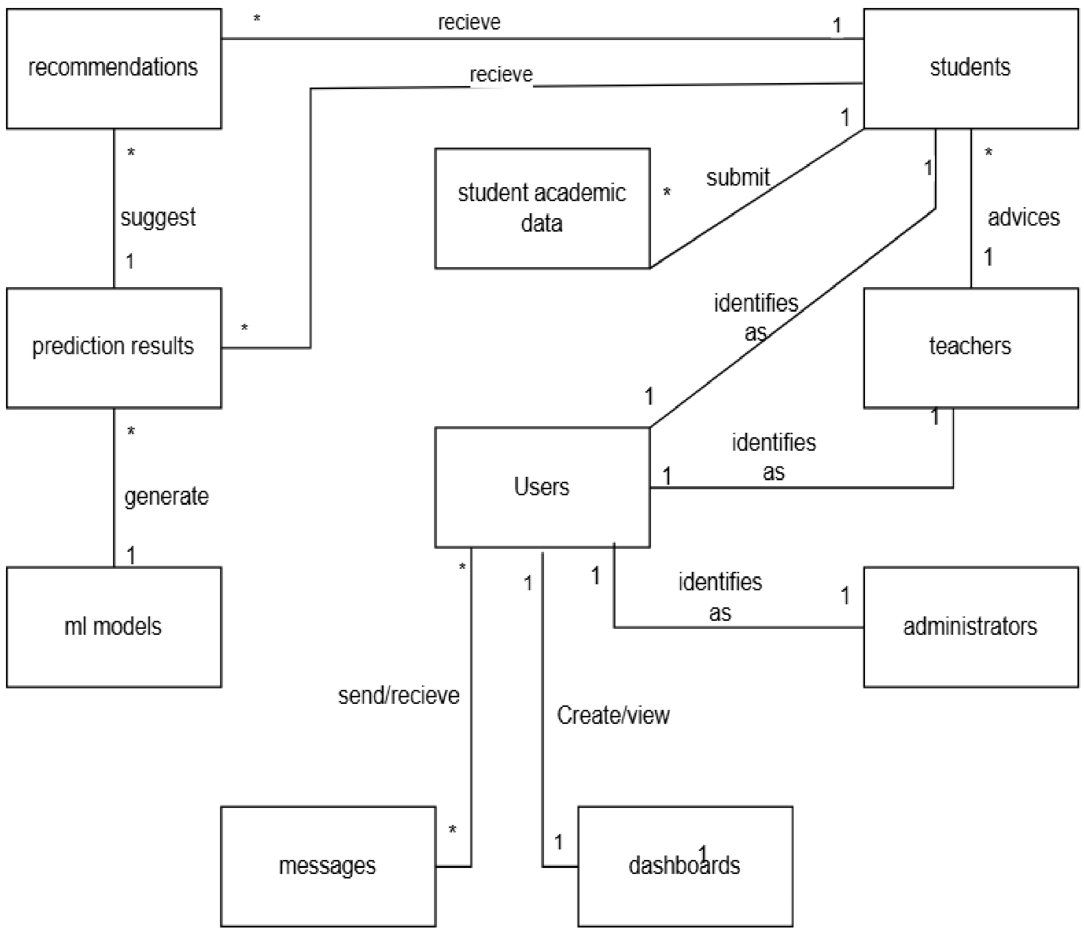


Fig. 3 System Class Diagram

4.2. Database Specification and Analysis

The EduTrack system requires an optimized database design for efficient operations (querying, retrieval and security)

Table: Users

Attribute	Type	Key
UserID	INTEGER	Primary Key
name	TEXT	
email	TEXT	UNIQUE
password	TEXT	
userRole	TEXT	

Table: Students

Attribute	Type	Key
StudentID	INTEGER	Primary Key
userID	INTEGER	Foreign Key
gpa	REAL	
departmentID	INTEGER	
academicStatus	TEXT	

Table: Teachers/Professor/Instructors

Attribute	Type	Key
teacherID	INTEGER	Primary Key
name	TEXT	
email	TEXT	
department	TEXT	

Table: Administrators

Attribute	Type	Key
adminID	INTEGER	Primary Key
userID	INTEGER	Foreign Key
name	TEXT	
email	TEXT	

Table: StudentAcademicData

Attribute	Type	Key
dataID	INTEGER	Primary Key
studentID	INTEGER	Foreign Key
attendance	REAL	
studyHours	REAL	
examScores	INT	
stressLevel	INTEGER	
sleepHours	REAL	
participation	REAL	

Table: PredictionResults

Attribute	Type	Key
predictionID	INTEGER	Primary Key
StudentID	INTEGER	Foreign Key
ModelID	INTEGER	Foreign Key
predictedStatus	TEXT	
passPercentage	REAL	
failPercentage	REAL	

Table: Recommendation

Attribute	Type	Key
recommendationID	INTEGER	Primary Key
predictionID	INTEGER	Foreign Key
studentID	INTEGER	Foreign Key
recommendationType	TEXT	
message	TEXT	

Table: ML Model

Attribute	Type	Key
modelID	INTEGER	Primary Key
modelName	TEXT	
accuracy	REAL	
modelType	TEXT	

Table: Message

Attribute	Type	Key
MessageID	INTEGER	Primary Key
SenderID	INTEGER	
receiverID	INTEGER	
subject	TEXT	
content	TEXT	
readStatus	TEXT	

Table: Dashboard

Attribute	Type	Key
dashboardID	INTEGER	Primary Key
userID	INTEGER	
dashboardType	TEXT	
customSettings	JSON	

Association between Tables

- User to Student: (1:1)
- Users to teacher (1:1)
- Users to Administrator (1:1)
- Users to Messages (M:N)
- Users to Dashboards (1:1)
- Student to academicData (1:N)
- Student to predictionResults (1:N)
- ML model to predictionReslults (1:N)
- predictionResults to recommendations (1:N)

Database Multiplicity

- A User can be a student
- A User can be a teacher
- A User can be an administrator
- A User can receive multiple messages
- A User has access to a dashboard
- A student can have multiple prediction results over time
- An ML model can have multiple predictions over time
- A prediction result can generate multiple recommendations

Security Considerations

- User authentication using hashed passwords
- Data encryption for sensitive fields
- Role-based access control (Least Privilege)
- SQL injection prevention via prepared statements.

Modelling Operation

We are using SQLite which is a Relational Database Management System.

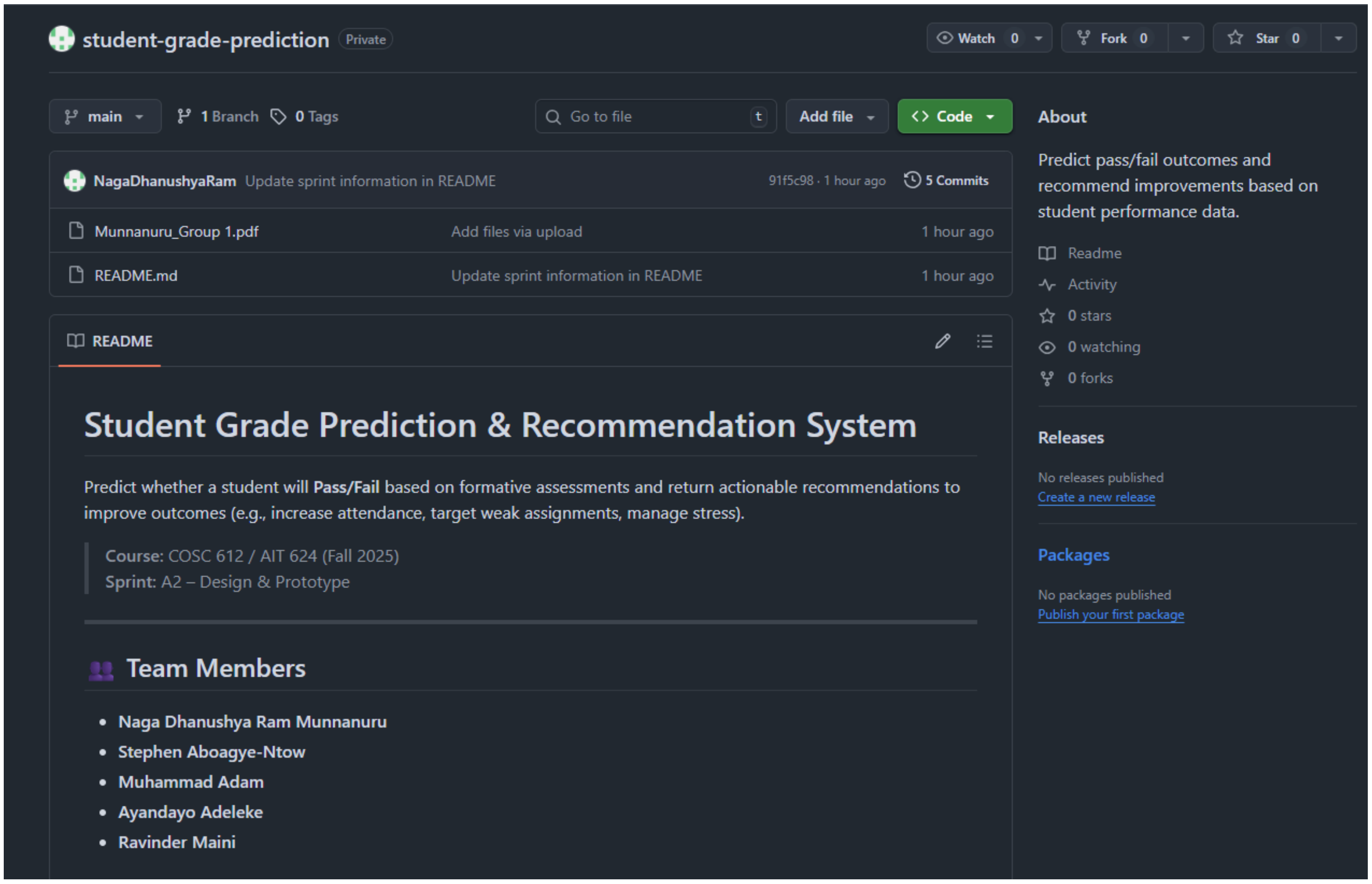
Final System Class Diagram

The system class diagram constitutes all described component/objects above with their relationships and constraints.

Conclusion

This system modelling analysis is the abstraction and graphical representation of the EduTrack system to understand and communicate its different perspectives, ensure scalability, security and simplicity.

5. Appendix



NagaDhanushyaRam Update sprint information in README 91f5c98 · 1 hour ago 5 Commits

Munnanuru_Group 1.pdf	Add files via upload	1 hour ago
README.md	Update sprint information in README	1 hour ago

Predict pass/fail outcomes and recommend improvements based on student performance data.

No releases published
Create a new release

No packages published
Publish your first package

Student Grade Prediction & Recommendation System

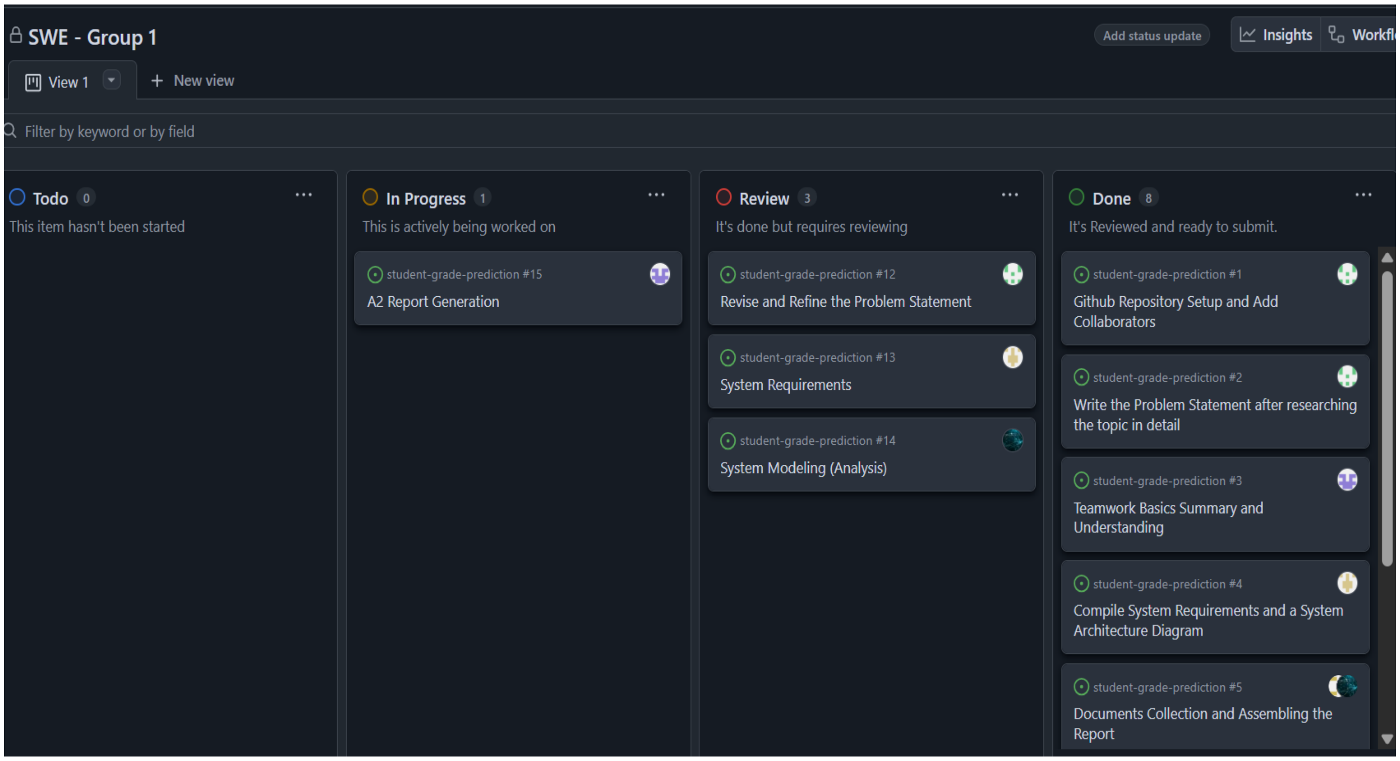
Predict whether a student will **Pass/Fail** based on formative assessments and return actionable recommendations to improve outcomes (e.g., increase attendance, target weak assignments, manage stress).

> **Course:** COSC 612 / AIT 624 (Fall 2025)
> **Sprint:** A2 – Design & Prototype

Team Members

- **Naga Dhanushya Ram Munnanuru**
- **Stephen Aboagye-Ntow**
- **Muhammad Adam**
- **Ayandayo Adeleke**
- **Ravinder Maini**

Fig. 4 GitHub Repository Homepage


SWE - Group 1

Add status update · Insights · Workfl

View 1 · New view

Filter by keyword or by field

Todo 0
This item hasn't been started

In Progress 1
This is actively being worked on

- student-grade-prediction #15 — A2 Report Generation

Review 3
It's done but requires reviewing

- student-grade-prediction #12 — Revise and Refine the Problem Statement
- student-grade-prediction #13 — System Requirements
- student-grade-prediction #14 — System Modeling (Analysis)

Done 8
It's Reviewed and ready to submit.

- student-grade-prediction #1 — Github Repository Setup and Add Collaborators
- student-grade-prediction #2 — Write the Problem Statement after researching the topic in detail
- student-grade-prediction #3 — Teamwork Basics Summary and Understanding
- student-grade-prediction #4 — Compile System Requirements and a System Architecture Diagram
- student-grade-prediction #5 — Documents Collection and Assembling the Report

Fig. 5 Kanban Board on GitHub