“STUDENT PREDICTION PERFORMANCE USING ID3 AND RANDOM FOREST ALGORITHM”

A PROJECT WORK SUBMITTED TO THE

DEPARTMENT OF BACHELOR OF SCIENCE IN COMPUTER SCIENCE AND INFORMATION TECHNOLOGY (BSC.CSIT)

NATIONAL COLLEGE OF COMPUTER STUDIES

FOR THE AWARD OF

BACHELOR OF SCIENCE IN COMPUTER SCIENCE AND INFORMATION TECHNOLOGY (BSC.CSIT)

BY

RITIKA MAHARJAN

28885/078

5-2-551-89-2021

SUPERVISOR:

SUMIT GHISING

SEPTEMBER,2025

SUPERVISOR RECOMMENDATION

This is to recommend that **Ritika Maharjan** symbol no. 28885/078 registration no. 5-2-551-89-2021 has carried out project work entitled **"STUDENT PREDICTION PERFORMANCE USING ID3 AND RANDOM FOREST ALGORITHM"** for the requirement to the project work in Bachelor in Science (B.Sc.) under my supervision in the Department of Computer Science and Information Technology (CSIT) National College of Computer Studies, Institute of Science and Technology (IoST), Tribhuvan University Nepal.

To my knowledge this work has not been submitted for any other degrees.

She has fulfilled all the requirements laid down by the Institute of Science and Technology (IoST), Tribhuvan University (T.U.), Nepal for the submission of the project work for the partial fulfillment of Bachelor of Science (B.Sc.) degree.

Sumit Ghising

Supervisor

Faculty Member

Department of Computer Science and Information Technology

National College of Computer studies

LETTER OF APPROVAL

This is to certify that the report is evaluated and recommended to the Department of Computer Science and Information Technology for acceptance of the report entitled "**STUDENT PREDICTION PERFORMANCE USING ID3 AND RANDOM FOREST ALGORITHM"** submitted by "RITIKA MAHARJAN" in partial fulfillment for the degree of Bachelor of Science in Computer Science and Information Technology (B.Sc.CSIT), Institute of Science and Technology (IOST), Tribhuvan University.

SIGNATURE of Supervisor Sumit Ghising Faculty Member Department of Computer Science and Information Technology National College of Computer Studies Paknajol, Kathmandu	**SIGNATURE of Coordinator** Rajan Paudel Coordinator of CSIT Department of Computer Science and Information Technology National College of Computer Studies Paknajol, Kathmandu
SIGNATURE of Internal Examiner **Internal Examiner**	**SIGNATURE of External Examiner** **External Examiner**

ACKNOWLEDGEMENTS

I would like to express my sincere gratitude to all those who supported and guided me throughout the development of the project titled "Student Performance Prediction using ID3 and Random Forest Algorithm." This project would not have been possible without the encouragement, assistance, and valuable feedback of several individuals.

First and foremost, I extend my heartfelt thanks to my project supervisor Mr. Sumit Ghising, for their continuous support, insightful suggestions, and expert guidance at every stage of the project. Their encouragement and constructive feedback greatly enhanced the quality of this work.

I am also thankful to the faculty members of the Department of Computer Science and Information Technology at National College of Computer Studies providing a strong academic foundation and for their guidance throughout my academic journey. Their teachings and support have played a crucial role in the successful completion of Student Performance Prediction.

My appreciation also goes to my friends and classmates, who provided encouragement and shared their knowledge and experiences, which contributed to the development of the project. Their collaboration and motivation kept me focused and inspired.

Finally, I am deeply grateful to my family for their constant support, patience, and belief in me. Their encouragement and moral support have been a source of strength throughout this journey.

Thank you all for being part of this achievement.

ABSTRACT

Student performance prediction is a significant application of machine learning aimed at identifying students at academic risk and improving educational outcomes. Student Performance Prediction focuses on predicting student performance using historical academic data and behavioral attributes. The system leverages the ID3 decision tree and Random Forest algorithm to analyze input features such as attendance, assignment scores, exam results, and participation. By training the model on labeled datasets, the system can accurately classify students into performance categories such as high, C(Average), or low. This prediction assists educators in making informed decisions and providing early interventions to underperforming students. Developed using MERN Stack, the project also includes CRUD operations for data management and a user-friendly interface for administrators. The proposed system enhances the decision-making process in educational institutions and aims to promote a data-driven academic environment, thereby helping improve overall student success rates.

KEYWORDS: *ID3(Iterative Dichotomiser 3), Random Forest, Student Performance Prediction, CRUD operation, MERN Stack*

TABLE OF CONTENTS

LIST OF ABBREVIATION

CRUD	Create, Read, Update, Delete
EDM	Educational Data Mining
i.e.	In Essence
ID3	Iterative Dechotomiser 3
IEEE	Institute of Electrical and Electronics Engineers.
JSON	JavaScript Object Notation
JWT	JSON Web Token
MERN	MongoDB, Express.js, React and Node.js
ML	Machine Learning
OOP	Object Oriented Programming
UI	User Interface

LIST OF FIGURES

LIST OF TABLES

CHAPTER 1:

INTRODUCTION

1.1. Introduction

Student Performance Prediction using ID3 Algorithm aims to develop an intelligent system capable of predicting the academic outcomes of students based on various academic and behavioral attributes. In the current educational environment, timely prediction of student performance is essential to identifying learners at risk and supporting them through targeted academic interventions. With the increasing availability of student data, there is a significant opportunity to utilize machine learning techniques to forecast student success, enabling institutions to make data-informed decisions.

The system is designed to take multiple factors into account, including internal assessments, projects, class participation, attendance, and other academic indicators. Based on these parameters, the system classifies students into performance categories such as "A(Excellent)", “B(Good)”, "C(Average)", or "D(Poor)". The primary goal is to assist teachers and administrators in understanding the performance trends of students, identifying potential issues early, and enabling personalized educational support.

The backend of the system is developed using Node.js and Express, providing a scalable server-side framework to handle application logic, user roles, and database interactions. MongoDB is used as the database to store student data, performance records, and user credentials, ensuring a flexible and efficient document-oriented data model. Role-based access is implemented to differentiate functionalities for three main users: Admin, Teacher, and Student**.** The admin manages user accounts and overall data entry; the Teacher can view performance reports of all students, while the student can only view their own performance results.

The frontend is developed using React.js and styled with Tailwind CSS, offering a modern, responsive, and user-friendly interface. This allows seamless interaction with the backend API and ensures that users can access prediction results and student records efficiently. React’s component-based architecture enables dynamic rendering of dashboards and performance insights based on the user role.

The core logic for classification is handled using the ID3 (Iterative Dichotomiser 3) algorithm and the Random Forest algorithm. ID3 is a decision tree algorithm that uses entropy and information gain to determine the most significant attributes for classification, while Random Forest is an ensemble learning method that combines multiple decision trees to enhance accuracy and reduce overfitting. Together, these algorithms analyze the input features provided by the user, construct predictive models, and generate performance labels accordingly. The use of ID3 allows the system to produce understandable rules from the data, making predictions interpretable and justifiable, whereas Random Forest provides robustness and improved generalization.

1.2. Problem Statement

Many educational institutions face challenges in identifying students who may underperform due to various academic and personal factors. Traditional methods of monitoring student performance are often reactive rather than proactive. There is a lack of intelligent systems that can analyze past data and provide early warnings regarding students at risk of failure. Thus, there is a need for a predictive model that can classify students based on their likelihood of success or failure using their academic and demographic attributes.

1.3. Objective

- To develop a web-based application for predicting the performance of students using the ID3 decision tree and Random Forest algorithm based on academic and behavioral data.

1.4. Scope and Limitation

1.4.1. Scope

The system allows users to input student information such as previous grades, class attendance, study time, extracurricular involvement, and assignments, which are used as input features for the ID3 and Random Forest algorithms to predict the student's performance level as "A(Excellent)", “B(Good)”, "C(Average)", or "D(Poor)". It supports three user roles: Admin, Teacher, and Student. Admins have full access to manage user accounts, input or update student data, and oversee the entire system operation. Teachers can view performance reports of all students, analyze patterns, and identify students who may need academic support. Students can log in to view only their own predicted performance along with a breakdown of the influencing factors, encouraging them to take proactive steps toward improvement.

1.4.2. Limitations:

The accuracy of the system is largely dependent on the quality and size of the dataset used for training and prediction. A limited or unbalanced dataset can lead to biased or inaccurate results. Additionally, the system does not consider real-time emotional, psychological, or social factors that may significantly influence a student's academic performance, which limits the depth of its predictions. Since the ID3 algorithm is designed to work with categorical input features, the system is constrained to using data that can be discretized or categorized, potentially overlooking the nuances found in continuous data. Moreover, the model is built using historical academic data, which means it may not quickly adapt to changes in curriculum structure, teaching methods, or assessment patterns, thereby affecting the relevance of its predictions over time.

1.5. Development Methodology

The development of the Student Performance Prediction using ID3 and Random Forest Algorithms followed the Incremental Model, where the system was built and delivered in successive phases. The process began with the initial increment that focused on core features such as role-based login, student data input, and ID3-based prediction. In the following increments, the system was expanded by designing the MongoDB schema, implementing backend logic using Node.js and Express, and developing the React.js frontend styled with Tailwind CSS. Additional increments introduced the Random Forest algorithm to improve prediction accuracy, along with responsive dashboards, performance summaries, and categorized student lists for enhanced usability. Each increment was tested and validated before moving to the next, ensuring smooth integration and reliable performance. This iterative approach enabled continuous feedback, early issue detection, and flexible improvements, resulting in a robust and scalable system that met the evolving requirements effectively.

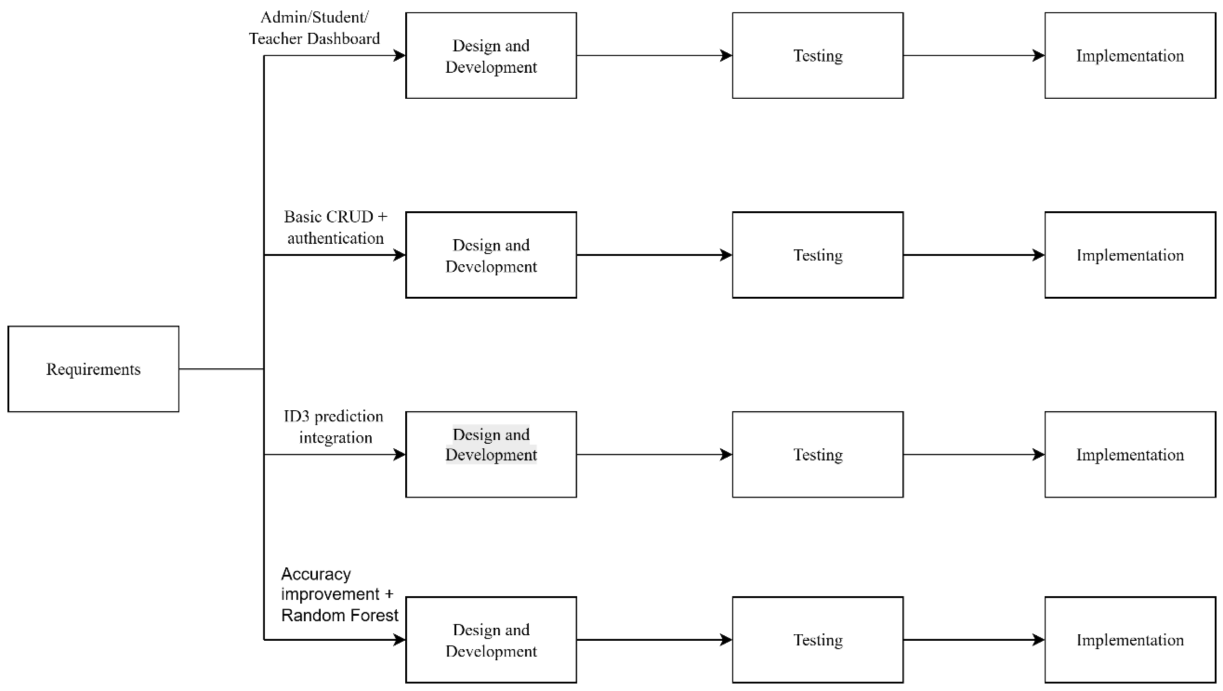

Requirements
Admin/Student/ Teacher Dashboard
Basic CRUD + authentication
ID3 prediction integration
Accuracy improvement + Random Forest
Design and Development
Testing
Implementation

Figure 1.1: Incremental Methodology

1.6. Report Organization

Chapter 1: Introduction

Introduces the project, its goals, methodology, and scope.

Chapter 2: Background Study and Literature Review

Presents background theories, concepts, and previous works related to student performance prediction.

Chapter 3: System Design and Architecture

Describes the overall structure of the system, including architectural components, data flow, and user role interactions.

Chapter 4: System Implementation

Provides implementation details such as the tools, programming languages, and frameworks used (MongoDB, Express, React, Tailwind), integration of the ID3 algorithm using JavaScript in the backend, and includes relevant system interface screenshots.

Chapter 5: Implementation and Testing

Discusses the output of the model, evaluates its performance using accuracy and other metrics, and analyzes how well the system meets its objectives.

Chapter 6: Conclusion and Future Enhancements

Concludes the project by summarizing key achievements and outlines potential improvements and additional features for future development.

CHAPTER 2:
BACKGROUND STUDY AND LITERATURE REVIEW

2.1. Background Study

Educational Data Mining (EDM) is a specialized branch of data mining that focuses on the application of data analysis techniques to educational environments. The primary goal of EDM is to extract useful patterns and knowledge from educational datasets to better understand student behaviors, learning outcomes, and institutional practices. With the rise of digital learning platforms and data-rich academic environments, institutions now collect vast amounts of data related to students' grades, attendance, learning habits, engagement levels, and demographic backgrounds. One of the core tasks in EDM is predictive modeling, which involves the use of classification algorithms to forecast future academic performance based on past records.

Among the various machine learning techniques used in EDM, decision tree algorithms like ID3 (Iterative Dichotomiser 3) are especially popular due to their simplicity, interpretability, and efficiency with categorical data. The ID3 algorithm, developed by Ross Quinlan, builds a decision tree using a top-down, greedy approach. This makes the resulting decision tree capable of clearly explaining how certain attributes influence the outcome, which is a valuable feature for educational stakeholders who need transparent and justifiable predictions.

The ID3 algorithm relies on several key concepts. Entropy is one such concept, which measures the randomness or impurity in a dataset. Each path from the root to a leaf in this tree represents a classification rule based on the input attributes. Studies have also highlighted that ensemble approaches such as Random Forest, which combine multiple decision trees, can enhance accuracy and stability in predicting student outcomes, especially in diverse educational datasets.

In the context of student performance prediction, the ID3 algorithm is particularly well-suited because it can process a range of categorical variables, such as attendance status, grade levels, projects, and more, to determine patterns and classify students into different performance categories. Moreover, since the output is in the form of human-readable rules, it is easier for educators and academic advisors to interpret and act upon the results. This makes ID3 not only a technically sound choice but also a practical one for building systems that support data-informed educational decisions. At the same time, recent works demonstrate that combining decision tree methods with ensemble models like Random Forest can provide even more robust predictions, further strengthening EDM's role in modern education.

2.2. Literature Review

Widaningsih et al. [1] developed an ID3-based prediction model at Telkom University to forecast student performance using holistic selection data. Their study showed that the ID3 algorithm effectively handled categorical academic data and revealed patterns useful for early intervention and resource allocation.

Lakshmi et al. [2] applied the ID3 algorithm to predict student performance based on students' academic background. They highlighted the importance of preprocessing and feature selection in improving accuracy and demonstrated how decision trees help identify struggling students for timely support.

Albreiki, Zaki, and Alashwal [3] conducted a review of educational data mining literature from 2009 to 2021. They found that decision tree algorithms are highly favored for their explainability and effectiveness, stressing their value in identifying at-risk students and enabling intervention programs.

An IEEE conference paper [4] examined student performance classification using supervised learning, focusing on decision tree algorithms. The research generated rules from training data to classify test data, confirming the model's usefulness in helping educators detect students needing extra attention

A comparative study [5] evaluated several machine learning algorithms, including Decision Tree, K-Nearest Neighbor, Naive Bayes, Discriminant Analysis, and Support Vector Machine. Results showed that decision trees achieve high accuracy while remaining interpretable, making them practical in educational contexts where clear explanations and early interventions are essential.

Ghosh and Janan [6] developed a Random Forest model to predict first-year student performance in a Bangladeshi university using eleven factors derived from both student and expert surveys processed via fuzzy ANFIS analysis. Their approach achieved around 96.88% accuracy, demonstrating the effectiveness of combining fuzzy logic with ensemble methods for early identification of student risk

CHAPTER 3:
SYSTEM ANALYSIS

3.1. System Analysis

System analysis is the process of studying and understanding the system's needs and requirements. For the Student Performance Prediction system, it helps identify what features are essential, what constraints exist, and how the system will work logically and practically.

3.1.1. Requirement Analysis

i. Functional Requirements

Functional requirements describe what the system should do. For this project, the core functionalities are:

- **User Authentication and Role Management**: The system allows users to register and log in as Admin, Teacher, or Student.
- **Admin Role**: Admin can add, update, and delete student and teacher records. Admin manages the input dataset required for predictions.
- **Teacher Role**: Teachers can view the performance records of all students, including their prediction results.
- **Student Role**: Students can log in and view their own performance prediction.
- **Prediction Engine**: The system uses the ID3 and random forest algorithm to predict performance labels such as “A(Excellent),” “B(Good),” “C(Average),” or “D(Poor)” based on input features like internal marks, attendance, projects, etc.

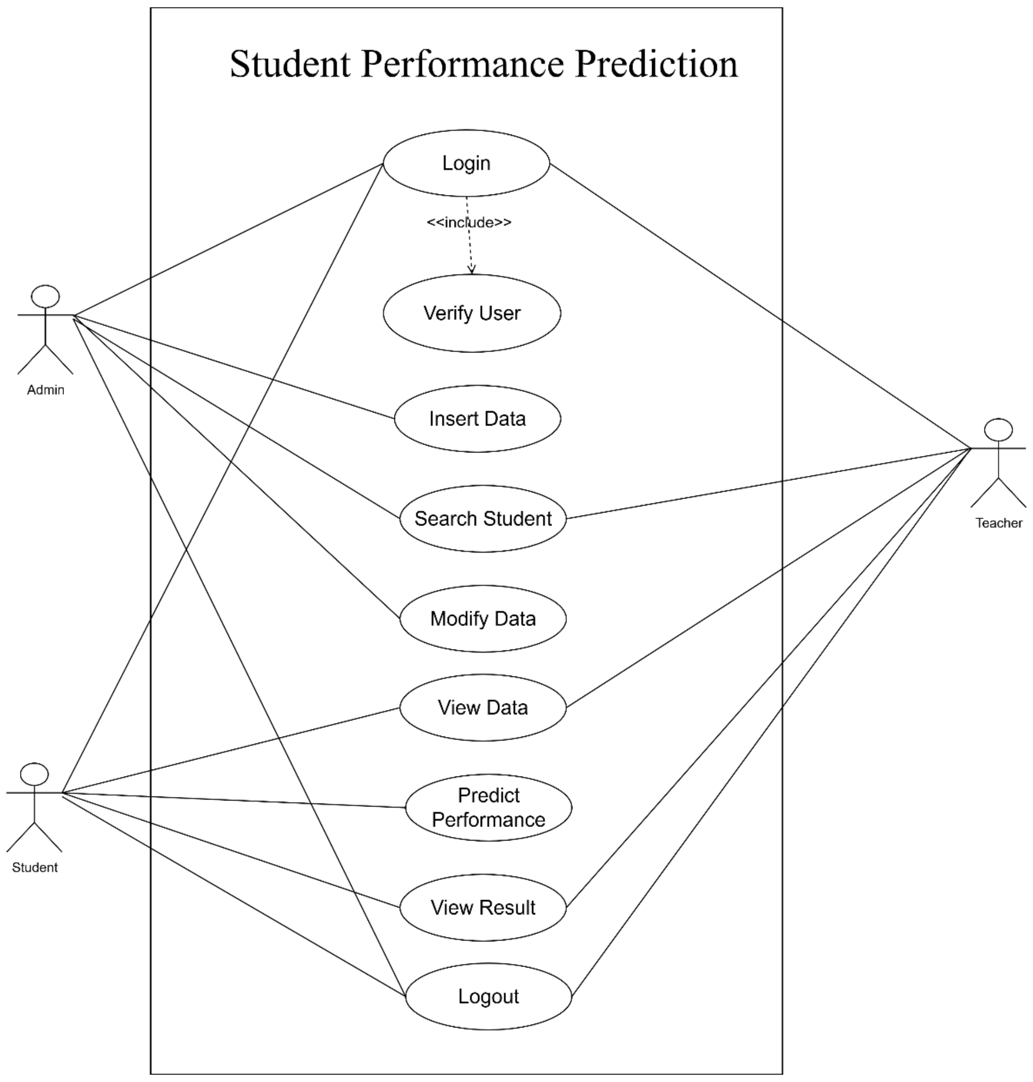


Figure 3.1: Use Case Diagram

ii. Non-Functional Requirements

These are the quality attributes of the system:

- **Performance:** The system processes predictions quickly after data input, ensuring users get results without noticeable delay.
- **Scalability:** It can handle a growing number of student records and multiple users accessing the system simultaneously without slowing down.
- **Usability:** The user interface is easy to use and responsive on all devices, thanks to React and Tailwind CSS, making it simple for Admins, Teachers, and Students to navigate.

- **Security:** Role-based access controls restrict users to only their allowed functions, and secure authentication with JWT protects user data and privacy.
- **Maintainability:** The system’s modular design and clean code structure make it easy to update, fix bugs, and add new features in the future.

3.1.2. Feasibility Analysis

i. Technical Feasibility

The Student Performance Prediction system is technically feasible with the chosen MERN stack technology. MongoDB provides a flexible and scalable document-based database ideal for storing diverse student records and performance data. The backend is developed using Express.js and Node.js, which offer a powerful and efficient environment to implement the core logic, including the ID3 and random forest algorithm written in JavaScript. This integration ensures smooth data processing and prediction generation. On the frontend, React.js combined with Tailwind CSS creates a dynamic, responsive, and user-friendly interface, enabling seamless interaction between users and the system.

ii. Operational Feasibility

From an operational perspective, the system is designed to be straightforward and easy to use. Admins and Teachers have access to intuitive dashboards for managing student data and viewing performance predictions, which simplifies their workflow. Students can log in and access their individual results without any complications. The system requires no specialized training, making it accessible to all users with basic computer skills, thus ensuring smooth day-to-day operation within educational institutions.

iii. Economic Feasibility

The project is economically feasible because it relies entirely on open-source technologies, eliminating the need for expensive software licenses or subscriptions. This cost-effectiveness makes the system suitable for academic institutions or individual educators who may have limited budgets. Additionally, by automating performance prediction, it can reduce the time and resources spent on manual student assessments, providing further economic benefits.

iv. Schedule Feasibility

The development followed a well-structured Incremental Model, where the system was built in successive phases, each adding new functionality to the existing framework. Initial increments focused on core features such as requirements gathering, system design, and basic implementation, while later increments introduced advanced algorithms, dashboards, and role-based access. Each phase included testing and validation before moving forward, ensuring reliability and smooth integration. This iterative approach aligned well with the academic timeline, allowing continuous improvements and timely completion of all deliverables within the set deadlines.

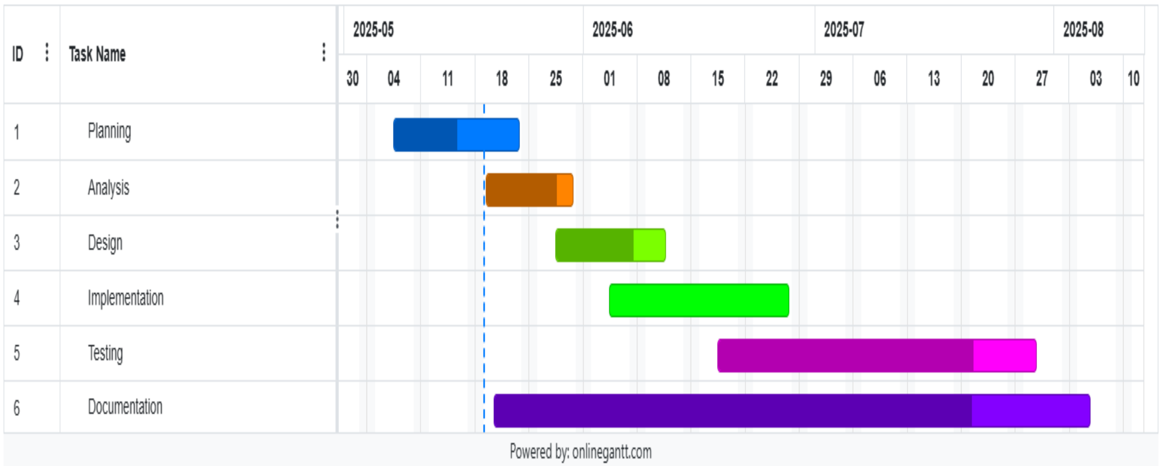


Figure 2.2: Gantt Chart

3.1.3. Analysis

The Student Performance Prediction System applies an object-oriented programming (OOP) approach to ensure modularity, reusability, and scalability. Core components such as user roles, data management, and prediction algorithms are encapsulated into separate classes and modules, promoting clean architecture and easier maintenance. The use of ID3 and Random Forest algorithms enhances prediction accuracy while the MERN stack provides a robust and responsive platform. Overall, the system demonstrates effective integration of OOP principles with machine learning to deliver a secure, flexible, and practical academic support tool.

i. Object Modeling using Class and Object Diagram

Class Diagram

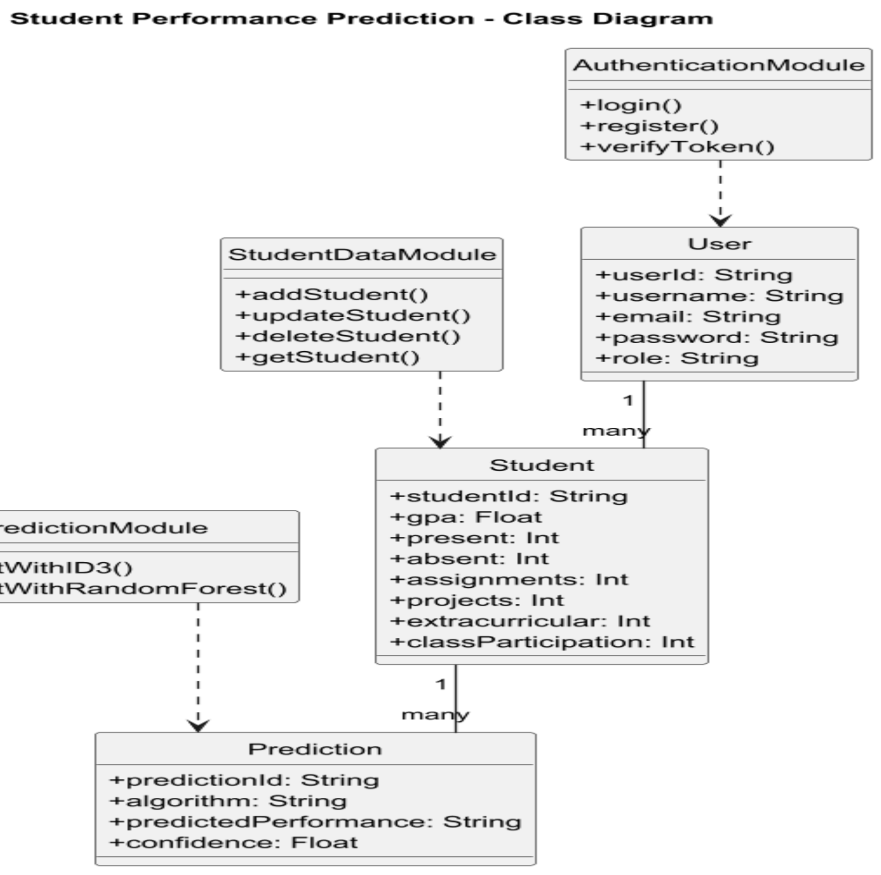


Figure 3.3: Class Diagram

The User class represents the main actors of the system, including Admin, Teacher, and Student, with attributes like userId, name, email, password, and role. It defines basic operations such as login and logout. Inheritance is used so that Teacher and Student extend from User, reusing common features while adding role-specific behaviors.

The Prediction class handles the logic for performance classification. It uses algorithms like ID3 and Random Forest to generate predictions based on student attributes. It maintains methods for training, predicting, and retrieving results, providing the intelligence layer of the system.

The Database class represents the storage unit of the system, handling persistence of user details, student data, and prediction results. It provides methods for saving, updating, and retrieving records from MongoDB.

The Frontend class (React + Tailwind) acts as the interface between the user and the system. It defines methods for input handling, data visualization, and displaying prediction results in a user-friendly way.

Together, these classes interact to form a complete flow where data is collected, processed, and analyzed, and the results are securely presented to the end users.

Object Diagram

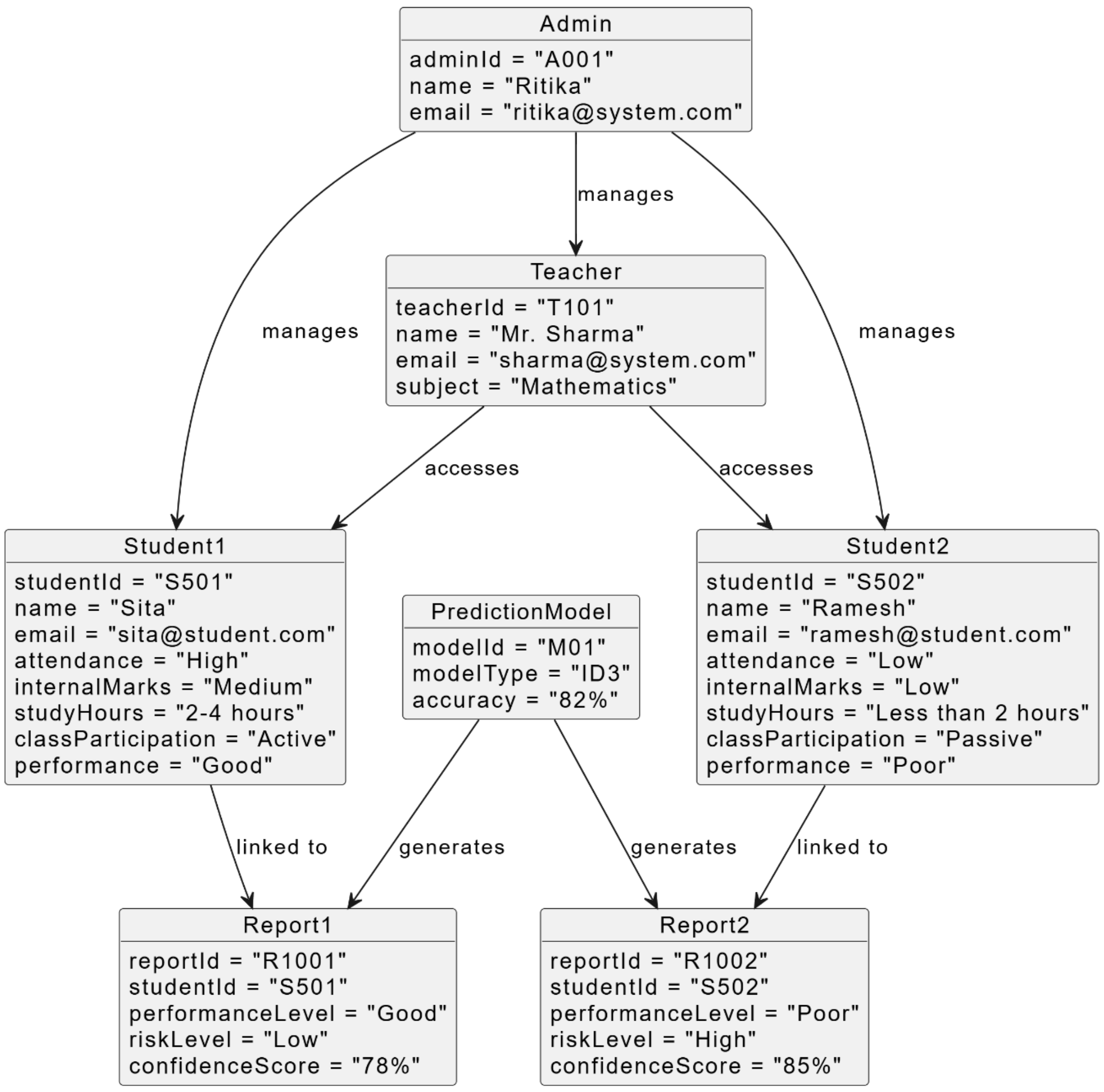


Figure 3.4: Object Diagram

The object diagram represents specific instances of the system's classes and their relationships at a particular moment. For example, admin1 manages teacher1 and student objects like student1 and student2, each containing real data such as attendance, internal marks, study hours, and class participation. The PredictionModel object holds trained ID3 and Random Forest models used to generate predictions and risk levels for each student. This diagram shows how actual objects interact, illustrating the flow of data from user input through the prediction module to the resulting performance output.

ii. Dynamic modelling using State and Sequence Diagram

State Diagram

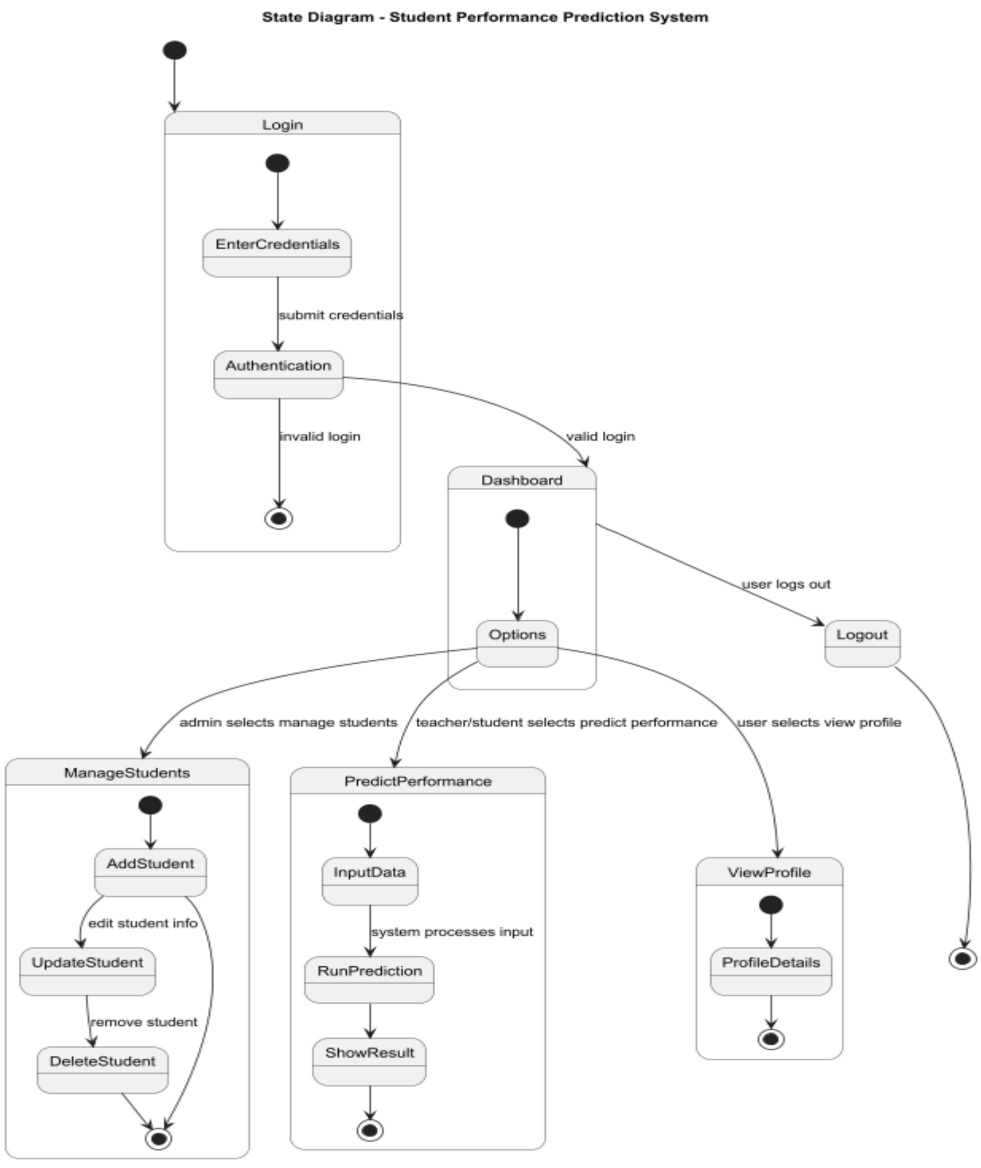


Figure 3.5: State Diagram

The state diagram begins with the Login state, where users such as Admin, Teacher, or Student enter their credentials. Once authenticated, the system transitions to the Dashboard state, which differs depending on the user role. Admins can manage student records, teachers can view and analyze predictions, while students can access their personal performance details. This ensures that role-specific access is maintained throughout the system.

From the dashboard, the system may move to the Data Input/Management state, where Admins can add or update student details such as attendance, marks, study hours, and class participation. This state ensures that the dataset remains up to date and accurate. Teachers and students transition from the dashboard to the Prediction state, where the ID3 or Random Forest algorithm is applied to classify student performance into categories like Excellent, Good, Average, or Poor.

Finally, after prediction, the system transitions to the Result Display state, where the output is shown on the interface. Teachers can view all students' predictions, while students can only see their own. The system can then return to the dashboard for further actions or move to the logout state, completing the user session. This flow ensures a smooth transition across different states while handling secure and role-based interactions.

Sequence Diagram

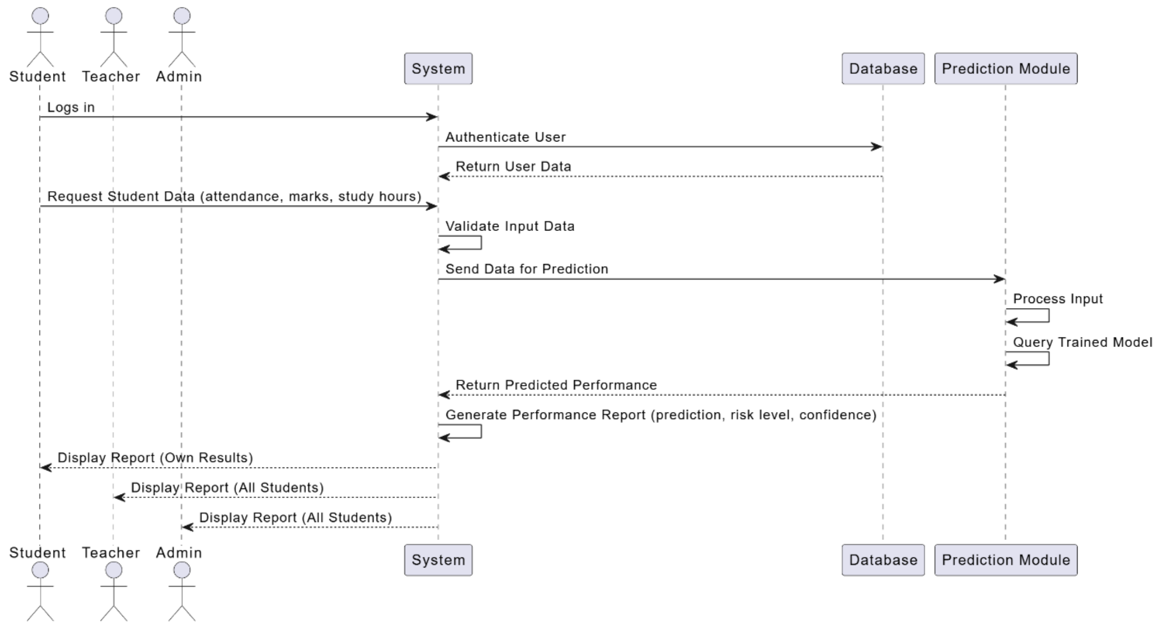

Student
Teacher
Admin
System
Database
Prediction Module
Logs in
Authenticate User
Return User Data
Request Student Data (attendance, marks, study hours)
Validate Input Data
Send Data for Prediction
Process Input
Query Trained Model
Return Predicted Performance
Generate Performance Report (prediction, risk level, confidence)
Display Report (Own Results)
Display Report (All Students)
Display Report (All Students)

Figure 3.6: Sequence Diagram

The sequence diagram starts with the User (Admin, Teacher, or Student) interacting through the Client Browser, which sends a request to the Web Server. The Web Server forwards the request to the Application Server, where authentication is handled using the Authentication

Module. Once validated, the user gains access to the system dashboard appropriate to their role.

Next, the Application Server manages data operations. For Admins, requests may involve adding or updating student records, which are stored in the Database Server. Teachers and Students, on the other hand, request predictions. These requests trigger the Prediction Module, which communicates with the Machine Learning Service containing trained models such as ID3 and Random Forest. The service processes the input data and returns a predicted performance outcome.

Finally, the Application Server sends the prediction results or updated data back to the Client Browser through the Web Server. Teachers may view all students' performance results, while students can only see their personal predictions. This interaction loop ensures smooth communication between components while maintaining role-based access and accurate predictions.

Process Modeling Using Activity Diagram

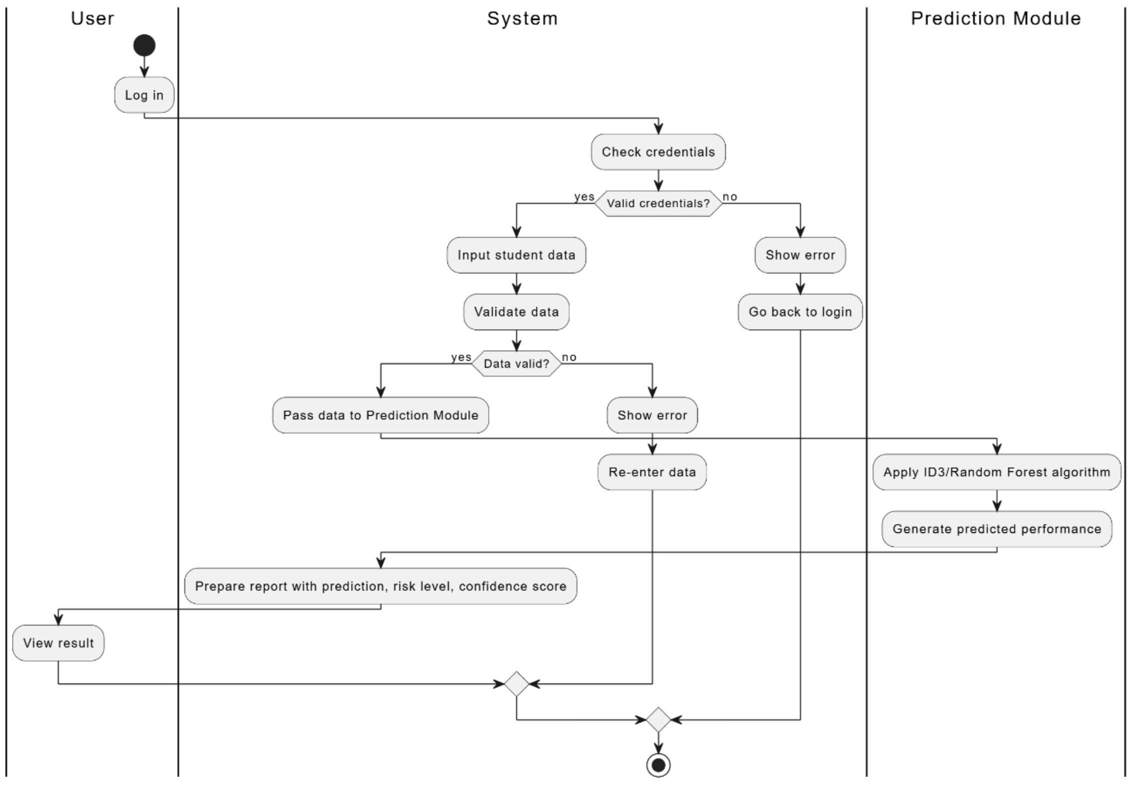


Figure 3.7: Activity Diagram

The activity diagram begins with the user login process, where Admins, Teachers, or Students authenticate themselves through the system. Once authenticated, the flow diverges based on the user role. Admins can proceed to manage student records by adding, updating, or deleting student data, while Teachers and Students move directly to performance-related activities.

For Teachers and Students, the next activity involves data submission and prediction requests. Students may input their details such as attendance, marks, study hours, and class participation, while Teachers can access existing student records. The system then forwards the request to the Prediction Module, which uses the ID3 or Random Forest models from the Machine Learning Service to generate results.

Finally, the activity diagram concludes with displaying the results. Teachers can view overall class or individual student predictions, whereas students receive only their personal performance outcome. This structured flow ensures clarity of operations, role-specific actions, and smooth execution from login to result generation.

CHAPTER 4:
SYSTEM DESIGN

4.1. Design

The system design of Student Performance Prediction is based on object-oriented approach, refining the analysis model into a structured design to guide implementation. The following aspects are considered.

i. Refinement of Class, Object, State, Sequence and Activity Diagram

Class Diagram

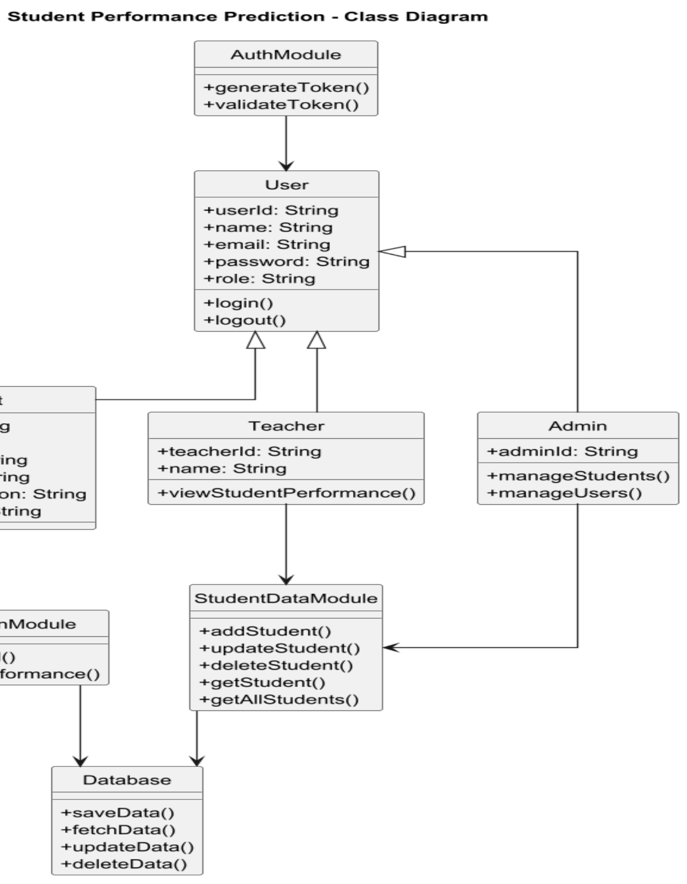


Figure 4.1: Refined Class Diagram

The refined class diagram shows the main classes of the system, including User, Student, Teacher, Admin, Prediction, Authentication, and Database. User is the parent class for Teacher and Student, sharing common attributes like userId, name, email, and password. The Student class contains academic attributes used for predictions, while the Prediction class handles performance classification using ID3 and Random Forest models.

Object Diagram

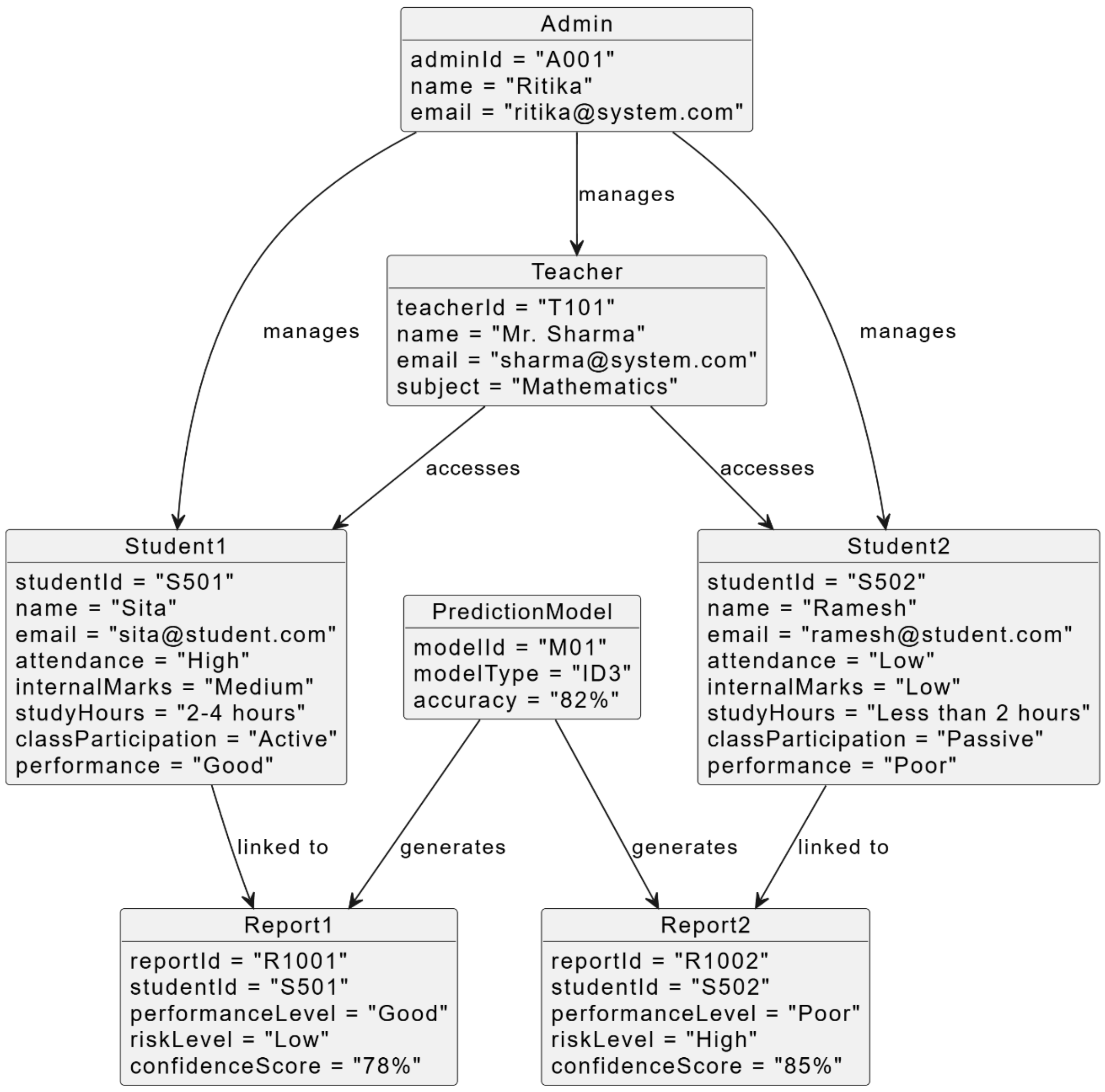


Figure 4.2: Refined Object Diagram

The refined object diagram shows specific instances of the system's classes and how they interact. For example, admin1 manages teacher1 and student objects such as student1 and student2, which contain real data like attendance, marks, and study hours. The PredictionModel object represents trained ID3 and Random Forest models used for generating predictions. This diagram illustrates how actual objects work together in the system.

Sequence Diagram

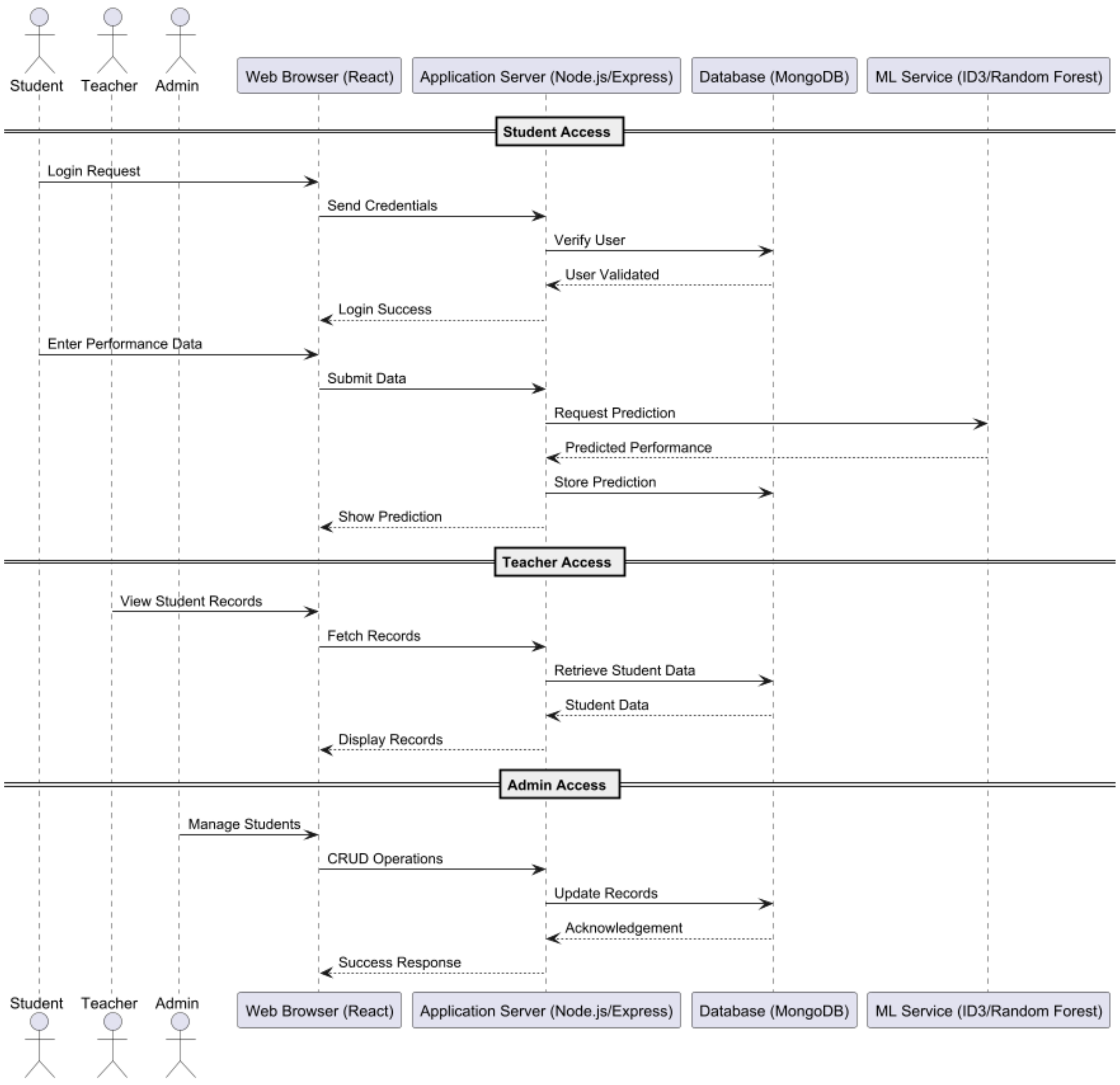


Figure 4.3: Refined Sequence Diagram

The refined sequence diagram shows the step-by-step interactions between users, the system, and its components. A user logs in through the client browser, which sends the request to the web server and then to the application server for authentication. Once authenticated, the user requests student data or predictions, which the application server processes through the Prediction Module using trained ID3 or Random Forest models. The prediction results are then returned to the user via the application and web servers, illustrating the flow of messages and actions in the system

Activity Diagram

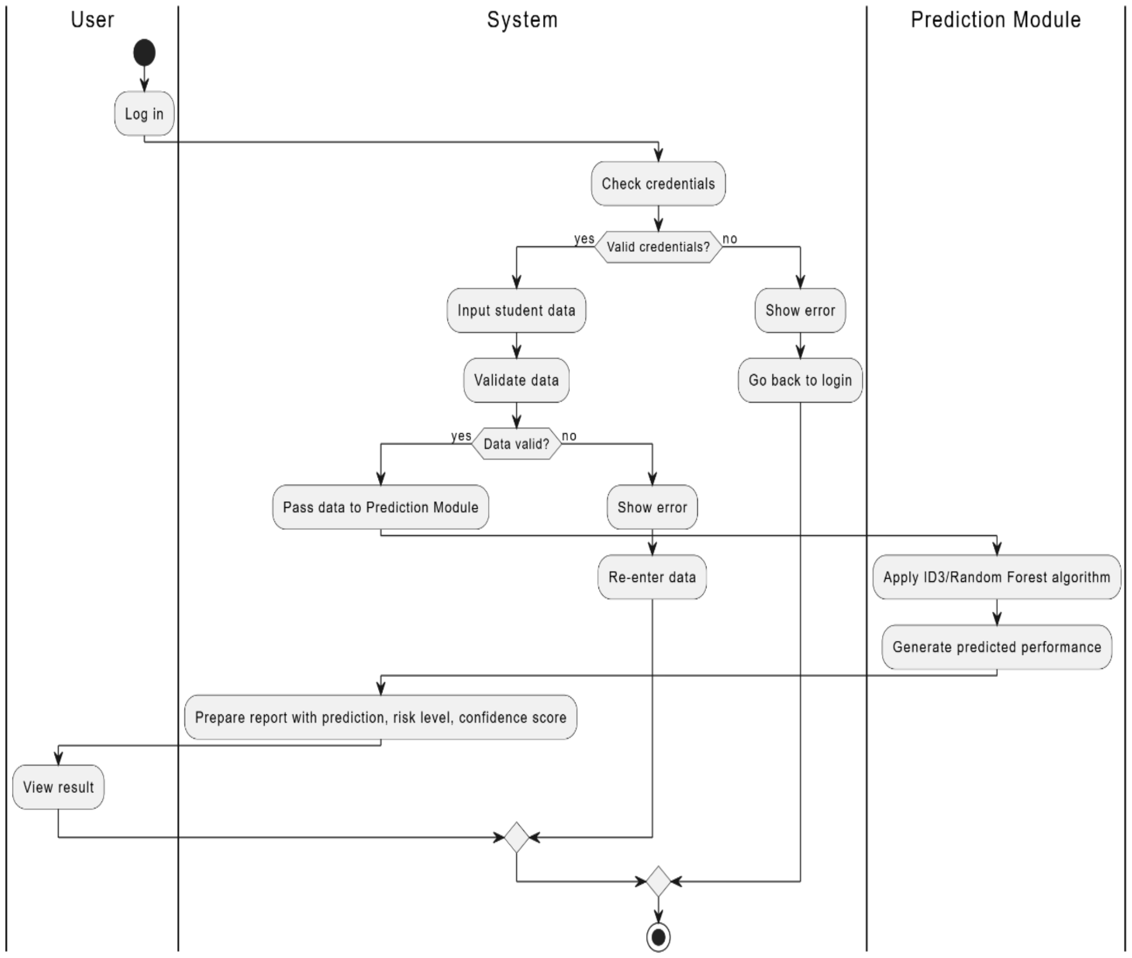


Figure 4.4: Refined Activity Diagram

The refined activity diagram shows the workflow of the Student Performance Prediction system. Users first log in, and the system verifies their credentials. Depending on the role, users can enter or access student data, which is then processed by the Prediction Module using ID3 or Random Forest models. The predicted performance is displayed to the user, and the workflow can return to the dashboard for further actions or end the session.

ii. Component Diagram

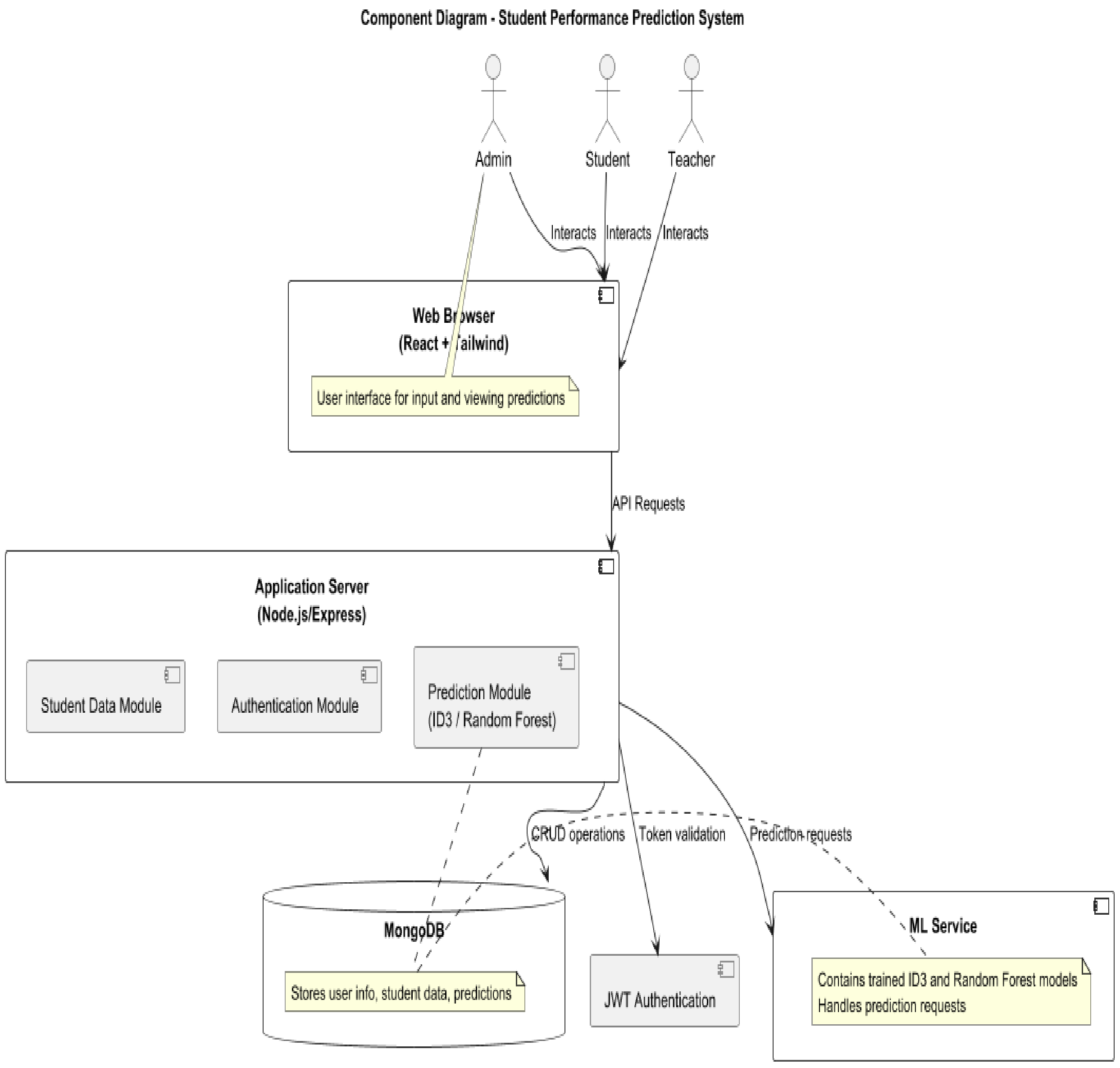


Figure 4.5: Component Diagram

The component diagram illustrates the modular structure of the Student Performance Prediction System by breaking it into independent yet interconnected components. The Frontend Component, developed using React and Tailwind, serves as the user interface that allows Admins, Teachers, and Students to interact with the system. It communicates with the backend through API requests and handles role-based navigation and data input.

At the backend, the Application Server Component is built with Node.js and Express. It contains subcomponents such as the Authentication Module for user login and role management, the Student Data Management Module for handling student records, and the Prediction Module that integrates ID3 and Random Forest models. These backend components are responsible for processing requests, managing logic, and ensuring secure communication between users and services.

The system also includes the Database Component, powered by MongoDB, which stores student data, prediction results, and user information. Additionally, the Machine Learning Service Component is responsible for training and prediction tasks, enabling accurate classification of student performance. All these components interact seamlessly, ensuring modularity, scalability, and easier maintenance of the system.

iii. Deployment Diagram

The deployment diagram represents the physical architecture of the system and shows how software components are distributed across hardware nodes. The Client Node represents the user's device, such as a PC or mobile browser, running the frontend developed with React and Tailwind. Users interact with the system through this node, sending requests to the backend via the web server.

The Web Server Node hosts and serves the frontend application, typically using Nginx or Apache. Requests from the client are forwarded to the Application Server Node, which runs Node.js/Express and contains the core modules including authentication, student data management, prediction (ID3/Random Forest), and report generation. This node handles all business logic and orchestrates interactions between other system nodes.

The Database Server Node stores persistent data such as user information, student records, and prediction results using MongoDB, while the Machine Learning Service Node contains trained models and performs prediction tasks. The system also integrates an External Authentication Service Node for secure login using JWT. The diagram shows how all nodes communicate, ensuring that client requests flow to the appropriate backend services and that predictions are generated and returned efficiently.

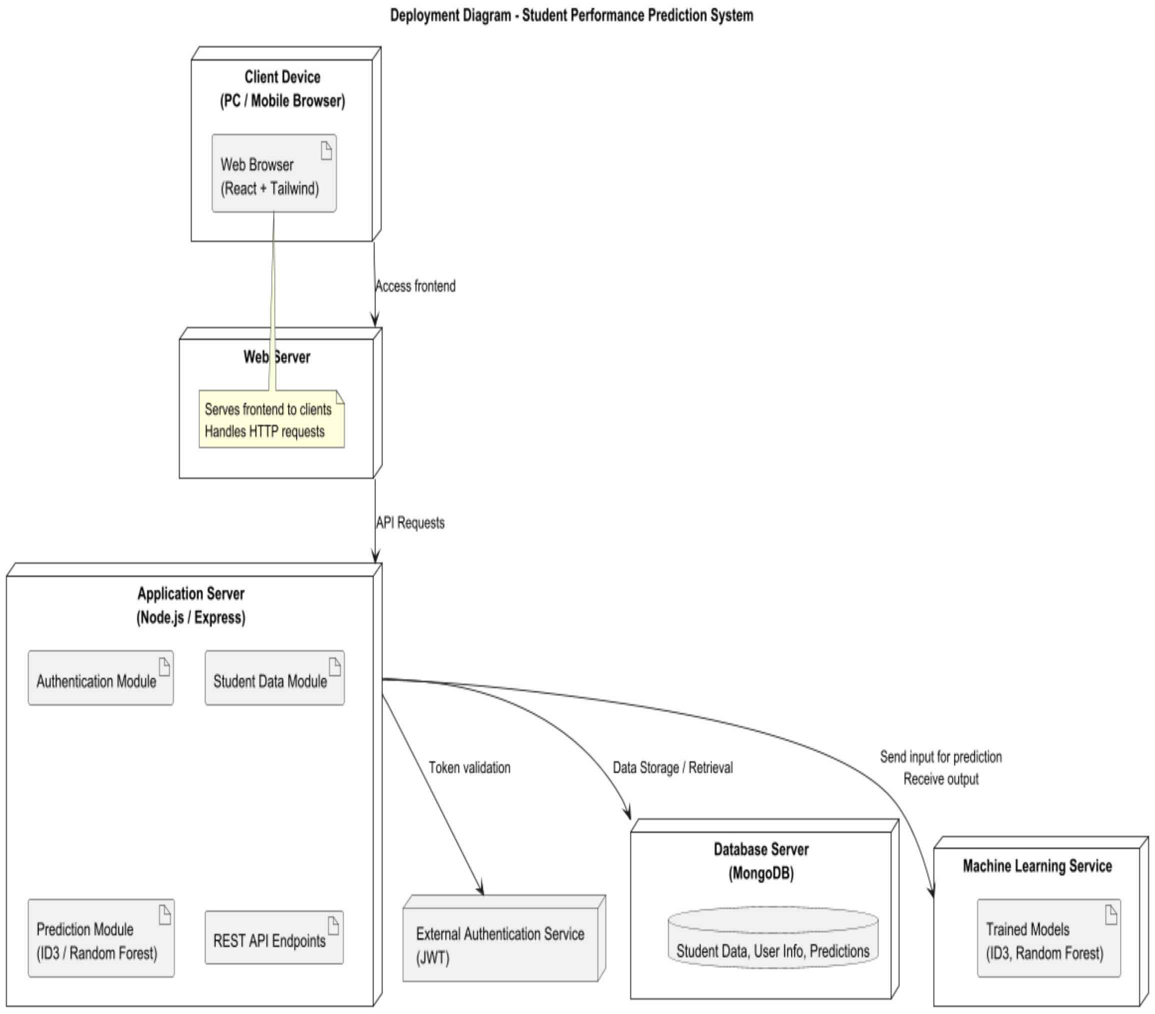


Figure 3.6: Deployment Diagram

4.2. Algorithm Details

The ID3 (Iterative Dichotomiser 3) algorithm is a decision tree algorithm developed by Ross Quinlan. It is commonly used for classification tasks and works by recursively partitioning the dataset based on the attribute that yields the highest information gain**,** which helps in building a decision tree.

ID3 algorithm is used to classify students into performance categories such as A(Excellent)**,** B(Good)**,** C(Average), or D(Poor) based on features like attendance, internal marks, projects, and class participation.

Working Principle of ID3 Algorithm

In the Student Performance Prediction system, the ID3 algorithm is used to classify a student's academic performance based on a set of input attributes. These attributes include Attendance (categorized as High, Medium, or Low), Internal Marks (High, Medium, or Low), Projects and Class Participation (Active or Passive). These features are selected because they represent key indicators of a student's learning behavior and engagement. The target attribute, which the system aims to predict, is the overall Performance of the student, categorized as A(Excellent), B(Good), C(Average), or D(Poor). The ID3 algorithm analyzes these inputs and constructs a decision tree by selecting the most informative attributes at each step using entropy and information gain. Once the tree is built, the system uses it to classify new student data by traversing the tree from the root to a leaf node based on the student's attribute values. The final leaf node reached provides the predicted performance label. This process allows the system to make accurate and interpretable predictions that can assist teachers and administrators in understanding student performance trends and identifying those who may need additional support.

Mathematical Formulas Used in ID3

1. Entropy (S)

Entropy measures the amount of uncertainty or impurity in a dataset.
For a dataset S with classes C1, C2, ..., Cn:

$$Entropy(S) = -\sum_{i=1}^{n} \text{pi(log2)pi}$$

Where:

- p_i is the proportion of examples in class C_i in dataset S.

2. Information Gain (IG)

Information Gain tells us how much entropy is reduced after splitting the data on an attribute A:

$$InformationGain(S, A) = Entropy(S) - \sum_{v \in Values(A)} \frac{|Sv|}{|S|} Entropy(Sv)$$

Where:

- S is the current dataset,
- S_v is the subset of S where attribute A=v,
- $|S_v|/|S|$ is the weight of the subset.

The attribute with the highest Information Gain is selected for the split.

Working Principle of Random Forest Algorithm

Random Forest creates an ensemble of decision trees and combines their outputs to improve prediction accuracy and reduce overfitting. It works based on the principle of bagging (Bootstrap Aggregating), where multiple trees are trained on different random samples of the dataset and their results are aggregated to produce the final output. Each tree might be trained on different student subsets (e.g., attendance, projects, class participation), and predictions are aggregated. For example, if three trees predict "B(Good)," "A(Excellent)," and "B(Good)," then the final prediction will be "B(Good)" based on majority voting. This ensures more robust and reliable predictions than using a single decision tree.

Working Steps of Random Forest:

1. **Data Sampling (Bootstrap Sampling):**
    - From the original dataset, the algorithm creates multiple subsets by randomly sampling with replacement.
    - Each subset is used to train a separate decision tree.
2. Feature Selection (Random Subset of Features):

- At each node of a tree, instead of considering all features, a random subset of features is chosen.
- This introduces randomness and reduces correlation among the trees.

3. **Tree Construction:**
    - Each tree is grown using the selected dataset and features, usually to the maximum depth without pruning.
    - The splitting at each node is based on measures like Information Gain or Gini Index (depending on the tree algorithm used).
4. **Prediction (Ensemble Voting):**
    - For classification problems, each tree outputs a class prediction, and the Random Forest chooses the class with the majority vote.
    - For regression problems, each tree outputs a numerical value, and the Random Forest computes the C(Average) of all predictions.

Mathematical Formulation:

1. **Bootstrap Sampling:**
   Create k bootstrap datasets $D_1, D_2, \ldots, D_k$ from the original dataset D.
2. **Decision Trees:**
   Train decision trees $T_1, T_2, \ldots, T_k$ on each dataset with random feature subsets.
3. **Final Prediction:**
    - **For classification:**

$$\boldsymbol{\hat{y} = mode(T1(x), T2(x), \ldots, Tk(x))}$$

   where mode means the majority vote.

   - **For regression:**

$$\boldsymbol{\hat{y} = \frac{1}{k}\sum_{i=1}^{k} Ti(x)}$$

CHAPTER 5:

IMPLEMENTATION AND TESTING

5.1. Implementation

5.1.1. Tool Used

The following tools and technologies were used for developing the system:

- **Programming Languages:**
    - JavaScript: used for both frontend and backend development
    - HTML/CSS: for basic markup and styling in the frontend
- **Frameworks and Libraries:**
    - React.js: for building dynamic and responsive user interfaces
    - Tailwind CSS: for styling components with utility-first classes
    - Node.js and Express.js: for creating RESTful APIs and implementing backend logic
- **Database Platform:**
    - MongoDB: for document-oriented data storage of users, student records, and predictions
- **ID3 Algorithm Implementation:**
    - Custom implementation of ID3 and Random Forest in JavaScript for seamless backend integration
- **Development Tools:**
    - VS Code: main code editor
    - Thunder Client: for API testing

5.1.2. Implementation Details of Modules

The system is divided into several key modules:

- **Authentication Module:**

Handles login and registration for Admin, Teacher, and Student roles. JWT (JSON Web Token) is used for secure authentication and session management.

Code:

```
const User = require("../models/User");
const Student = require("../models/StudentData");
const bcrypt = require("bcryptjs");
const jwt = require("jsonwebtoken");
const JWT_SECRET = process.env.JWT_SECRET || '1234567890';
exports.register = async (req, res) => {
  const { name, email, password, role } = req.body;
  try {
    const existing = await User.findOne({ email });
    if (existing) return res.status(400).json({ msg: "User already exists" });
    const hashedPassword = await bcrypt.hash(password, 10);
    const newUser = await User.create({
      name, email, password: hashedPassword, role});
    res.status(201).json({ msg: "User registered successfully" });
  } catch (err) {
    res.status(500).json({ error: err.message });}};
exports.login = async (req, res) => {
  const { email, password } = req.body;
  try {
    console.log("Login attempt for:", email);
    const user = await User.findOne({ email });
    const std = await Student.findOne({ email });
    console.log("Found user:", user);
    console.log("Comparing password for user:", user.password);
    console.log("Password", password)// Log before comparing password
    const isMatch = await bcrypt.compare(password, user.password);
    console.log("Password match:", isMatch);
    console.log("Student found:", std);
    if (user.role === 'student') {
      if (!isMatch) {
        const student = await bcrypt.compare(password, std.password);
        if (!student) {
          return res.status(400).json({ msg: "Invalid credentials" }); }}}
    const token = jwt.sign({ id: user._id, role: user.role }, JWT_SECRET, {
      expiresIn: "1d", });
    if (user.role === 'student') {
      return res.json({
        token,
        user: {
            id: std._id, name: std.name, role: std.role, email: std.email, present:
std.present, absent: std.absent, assignments: std.assignments, quizzes: std.quizzes,
midterm: std.midterm, final: std.final, gpa: std.gpa, projects: std.projects,
extracurricular: std.extracurricular, classParticipation: std.classParticipation}});}
    res.json({
      token,
      user: {
        id: user._id, name: user.name, role: user.role, email: user.email
      }});} catch (err) {
```

```
    console.log("Login error:", err);
    res.status(500).json({ error: "Login failed" });}
  };
```

- **User Role Management:**

  Based on the user role, different components and pages are rendered. Role-based access ensures restricted and secure feature availability.

- **Student Module:**

  Includes student data input (attendance, marks, projects, etc.) and allows students to view their own predicted performance.

- **Teacher Module:**

  Teachers can view the list of all students along with their predicted performance results.

- **Admin Module:**

  Admins can manage (add/update/delete) student and teacher data and upload performance data for prediction.

- **ID3 Algorithm Module:**

  A custom JavaScript function calculates entropy and information gain from the input features, constructs a decision tree, and outputs the predicted performance label.

  Code:

```
function entropy(data, label) {
  const total = data.length;
  const counts = {};
  data.forEach(row => {
    const labelVal = row[label];
    counts[labelVal] = (counts[labelVal] || 0) + 1;});
  return Object.values(counts).reduce((sum, count) => {
    const p = count / total;
    return sum - p * Math.log2(p);//entropy calculation
  }, 0);}
function infoGain(data, attr, label) {
  const total = data.length;
  const groups = {};
  data.forEach(row => {
    const val = row[attr];
    groups[val] = groups[val] || [];
    groups[val].push(row);});
  const weightedEntropy = Object.values(groups).reduce((sum, group) => {
    return sum + (group.length / total) * entropy(group, label);
  }, 0);
  return entropy(data, label) - weightedEntropy;
}
```

- **Random Forest Module:**

  A custom JavaScript function builds multiple decision trees using subsets of the data and features, combines their outputs through majority voting, and generates the final predicted performance label for each student.

  Code:

```
const { buildTree, predict } = require('./id3');
function bootstrapSample(data, sampleSize) {
  const sample = [];
  for (let i = 0; i < sampleSize; i++) {
    const randIndex = Math.floor(Math.random() * data.length);
    sample.push(data[randIndex]);}
  return sample;}
function getRandomFeatures(features, maxFeatures) {
  const shuffled = [...features].sort(() => 0.5 - Math.random());
  return shuffled.slice(0, maxFeatures);}
function trainRandomForest(data, features, label, nTrees = 10, maxFeatures = null)
{
  const forest = [];
  const sampleSize = data.length;
  const featureCount = maxFeatures || Math.floor(Math.sqrt(features.length));
  for (let i = 0; i < nTrees; i++) {
    const sample = bootstrapSample(data, sampleSize);
    const selectedFeatures = getRandomFeatures(features, featureCount);
    const tree = buildTree(sample, selectedFeatures, label);
    forest.push({ tree, selectedFeatures });}
  return forest;}
function predictRandomForest(forest, input) {
  const votes = {};
  forest.forEach(({ tree }) => {
    const result = predict(tree, input);
    votes[result] = (votes[result] || 0) + 1;});
  return Object.entries(votes).reduce((a, b) => (a[1] > b[1] ? a : b))[0];}
function forestAccuracy(forest, data, label) {
  let correct = 0;
  data.forEach(sample => {
    const prediction = predictRandomForest(forest, sample);
    if (prediction === sample[label]) correct++;});
  return correct / data.length;}
module.exports = {
  trainRandomForest, predictRandomForest, forestAccuracy
};
```

5.2. Testing

Testing ensures that the system functions correctly and meets all specified requirements. The test cases are organized into two categories: Unit Testing and System Testing, and are presented in a table format for clarity.

5.2.1. Unit Testing

Table 1: Unit Testing

Test Case ID	**Module**	**Test Description**	**Input**	**Expected Output**	**Actual Result**	**Status**
UT01	Authentication	Test login with valid credentials	Email: admin@gmail.com, Password: admin123	Redirect to Admin Dashboard	As Expected	Pass
UT02	Authentication	Test login with invalid credentials	Email: wrong@mail.com, Password: wrongpass	Error message: "Invalid credentials"	As Expected	Pass
UT03	Student Data Entry	Validate required student fields	Missing email field	Error message: "Email is required"	As Expected	Pass

UT04	Prediction Module	Predict performance from valid student input	Attendance: High, Marks: High, Projects: 4+	Prediction: A(Excellent)	As Expected	Pass
UT05	ID3 Algorithm	Check information gain calculation	Sample categorical data set	Correct attribute selected based on highest information gain	As Expected	Pass
UT06	Form Validation	Submit form with invalid data types	Projects: "abc"	Error message: "Invalid input for projects"	As Expected	Pass

5.2.2. System Testing

Table 2: System Testing

Test Case ID	Scenario	Test Description	Input/Action	Expected Output	Actual Result	Status
ST01	Admin Role	Admin adds a new student	Fill student form and submit	Student successfully added to the database	As Expected	Pass

ST02	Teacher Role	Teacher views all student predictions	Teacher logs in and navigates to prediction page	Display of all students with performance categories	As Expected	Pass
ST03	Student Role	Student views own prediction result	Student logs in	Display of only logged-in student's prediction result	As Expected	Pass
ST04	Data Flow	End-to-end prediction process	Admin enters data → Prediction triggered	Accurate prediction generated and visible to users	As Expected	Pass
ST05	Role-based Access Control	Unauthorized access to restricted pages	Student tries to access Admin dashboard URL directly	Redirected to Unauthorized page or access denied message	As Expected	Pass
ST06	UI Responsiveness	View on different screen sizes	Open app on mobile, tablet, and desktop	Proper layout and responsiveness maintained	As Expected	Pass
ST07	System Error Handling	Submit invalid or incomplete form data	Empty required fields	User receives appropriate validation and error messages	As Expected	Pass

5.3. Result Analysis

The system was tested with sample student data. The ID3 and Random Forest algorithm successfully generated accurate predictions for various input cases. The UI was responsive, and all user roles functioned as expected. The result analysis showed that the system was reliable, user-friendly, and effective in classifying student performance. The project met its primary objectives of predicting academic outcomes based on behavioral and academic attributes.

CHAPTER 6:
CONCLUSION AND FUTURE RECOMMENDATION

6.1. Conclusion

The Student Performance Prediction using ID3 and Random Forest Algorithm project successfully demonstrates how machine learning can be applied in education to predict academic performance. By integrating the ID3 and Random Forest algorithm with a full-stack MERN web application, the system offers a practical solution for teachers and administrators to monitor student progress and take early actions when necessary. The use of role-based access, a clean UI, and interpretable decision tree results ensures that the system is not only functional but also user-centric. Overall, the project bridges the gap between data and actionable insight in academic settings.

6.2. Future Recommendation

To enhance the effectiveness and usability of the Student Performance Prediction system, several improvements can be considered in future development. One key enhancement is to expand the feature set by incorporating additional attributes such as parental involvement, participation in extracurricular activities, and individual learning styles, which could lead to more accurate and holistic predictions. Another important recommendation is to add an analytics dashboard for teachers and administrators, offering visual insights such as charts and graphs to track performance trends over time. Implementing real-time prediction capabilities would allow the system to automatically update prediction results as new student data is entered or modified, making the system more dynamic and responsive. Additionally, the system could include a feature to export reports in formats like PDF or Excel, enabling easy sharing and record-keeping. Lastly, incorporating an AI model comparison module that supports other machine learning algorithm such as, Naive Bayes would allow users to compare prediction results and choose the most effective model for their data, thereby improving the overall accuracy and reliability of the system.

References

[1 Albreiki, B., Zaki, N., & Alashwal, H. (2021). A review of machine learning algorithms for predicting student performance in educational settings. Education and Information Technologies, 26*, 4007–4034.]* https://www.mdpi.com/2227-7102/11/9/552

[2 Lakshmi, G. J., Manjula, D., & Begum, S. (2014). Student performance prediction using decision tree algorithm. International Journal of Computer Applications, 108*(15), 12–18.]* https://d1wqtxts1xzle7.cloudfront.net/81118733/Data-Mining-A-prediction-for-Students-Performance-Using-Decision-Tree-libre.pdf

[3 Rahman, M. M., Yeasin, M. A., & Siddique, M. R. (2019). Student performance classification with decision tree algorithm. *In* 2019 International Conference on Robotics, Electrical and Signal Processing Techniques (ICREST) *(pp. 643–647).]* IEEE. https://www.sciencedirect.com/science/article/abs/pii/S221478532105241X

[4 Widaningsih, L., Sari, R. F., Fitriyani, N., & Khaeruddin, M. (2023). ID3-Based prediction model for student performance using holistic selection data. Education Sciences,11*(9),552.]*
https://www.proquest.com/openview/23e626f9e6a536f5088e9ae93881aefb/1?cbl=2069459&pq-origsite=gscholar

[5 Yadav, S. K., Bhardwaj, B., & Pal, S. (2012). Data mining: A prediction for student's performance using classification method. International Journal of Computer Science and Information Security (IJCSIS)*,* *10*(7), 292–297.*]* Retrieved from https://ieeexplore.ieee.org/abstract/document/8697770

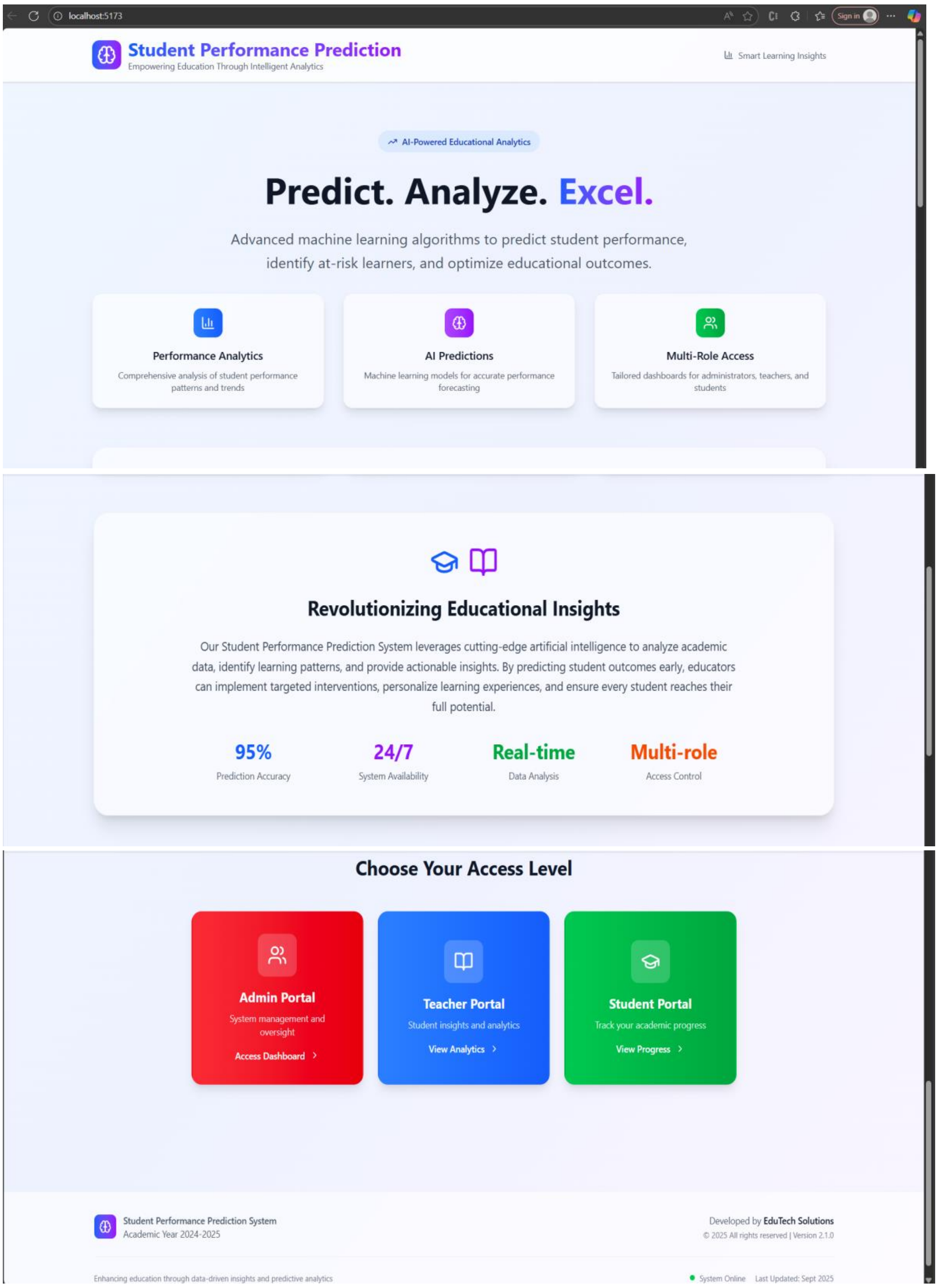
[6 Thakar, P. (2021). Performance analysis and prediction in educational data mining: A research travelogue. In Proceedings of the 11th Annual International Conference on Industrial Engineering and Operations Management (pp. 7263–7272). IEOM Society International.]
https://www.ieomsociety.org/singapore2021/papers/1238.pdf

[7 Soumya, G., Vatsavayi, V. K., & Sreedevi, M. (2021). Prediction of student academic performance using hybrid model. In Proceedings of the 2021 International Conference on Computer Communication and Informatics (ICCCI) (pp. 1–5). ACM.] https://dl.acm.org/doi/abs/10.1145/3446590.3446607

[8 Ab Rashid, R., Awang, H., & Rosli, D. I. (2020). Predicting students' academic performance using data mining techniques. International Journal of Informatics and Computing (IJIC), 10(1), 46–55.]
https://ijic.utm.my/index.php/ijic/article/view/143
[9 Hussain, S., Dahan, N. A., Ba-Alwib, F. M., & Ribata, N. (2020). Educational data mining and analysis of students' academic performance using WEKA. IEEE Access, 8, 147530–147541.]
https://ieeexplore.ieee.org/abstract/document/9167547

Appendices

- Home Page



localhost:5173

Student Performance Prediction
Empowering Education Through Intelligent Analytics

Smart Learning Insights

AI-Powered Educational Analytics

Predict. Analyze. Excel.

Advanced machine learning algorithms to predict student performance, identify at-risk learners, and optimize educational outcomes.

Performance Analytics
Comprehensive analysis of student performance patterns and trends

AI Predictions
Machine learning models for accurate performance forecasting

Multi-Role Access
Tailored dashboards for administrators, teachers, and students

Revolutionizing Educational Insights

Our Student Performance Prediction System leverages cutting-edge artificial intelligence to analyze academic data, identify learning patterns, and provide actionable insights. By predicting student outcomes early, educators can implement targeted interventions, personalize learning experiences, and ensure every student reaches their full potential.

95%	24/7	Real-time	Multi-role
Prediction Accuracy	System Availability	Data Analysis	Access Control

Choose Your Access Level

Admin Portal
System management and oversight
Access Dashboard

Teacher Portal
Student insights and analytics
View Analytics

Student Portal
Track your academic progress
View Progress

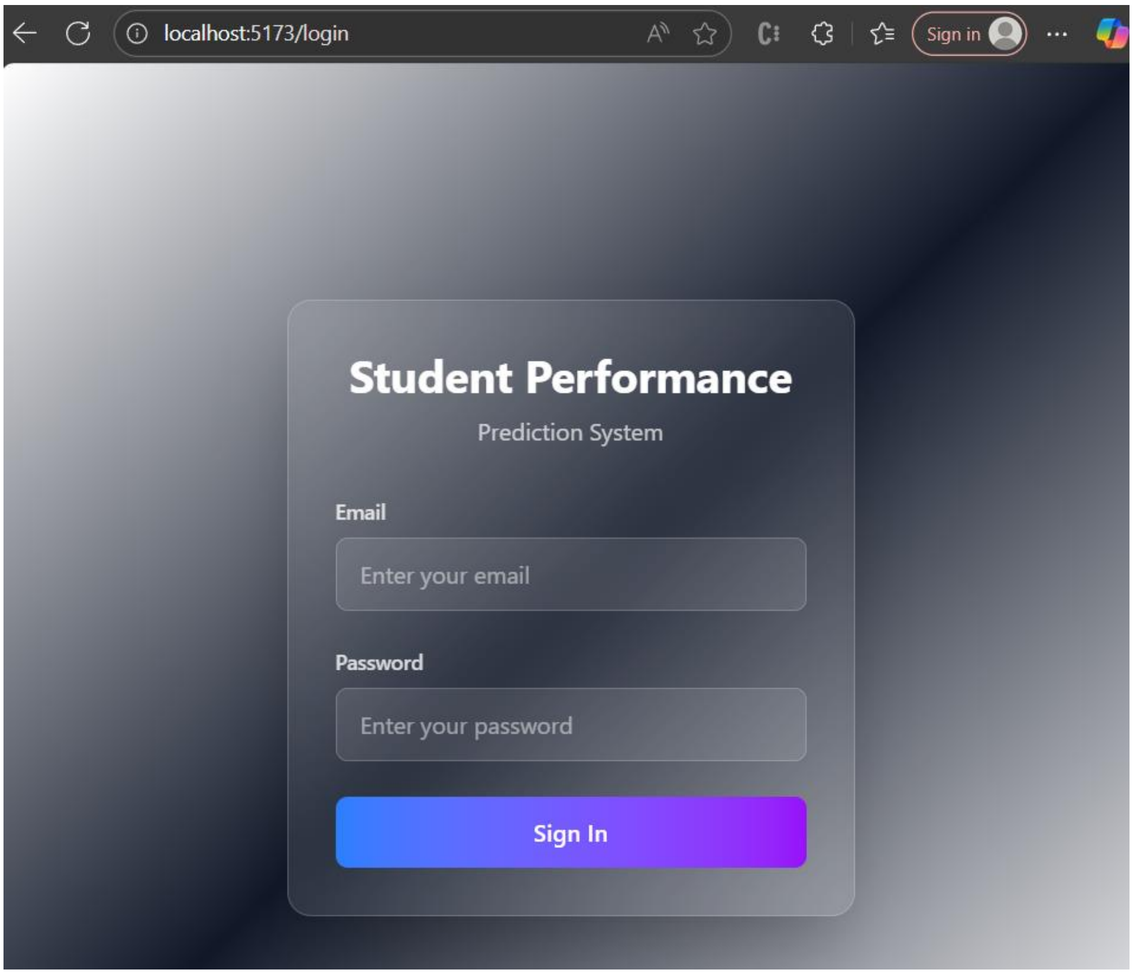
Student Performance Prediction System
Academic Year 2024-2025

Developed by **EduTech Solutions**
© 2025 All rights reserved | Version 2.1.0

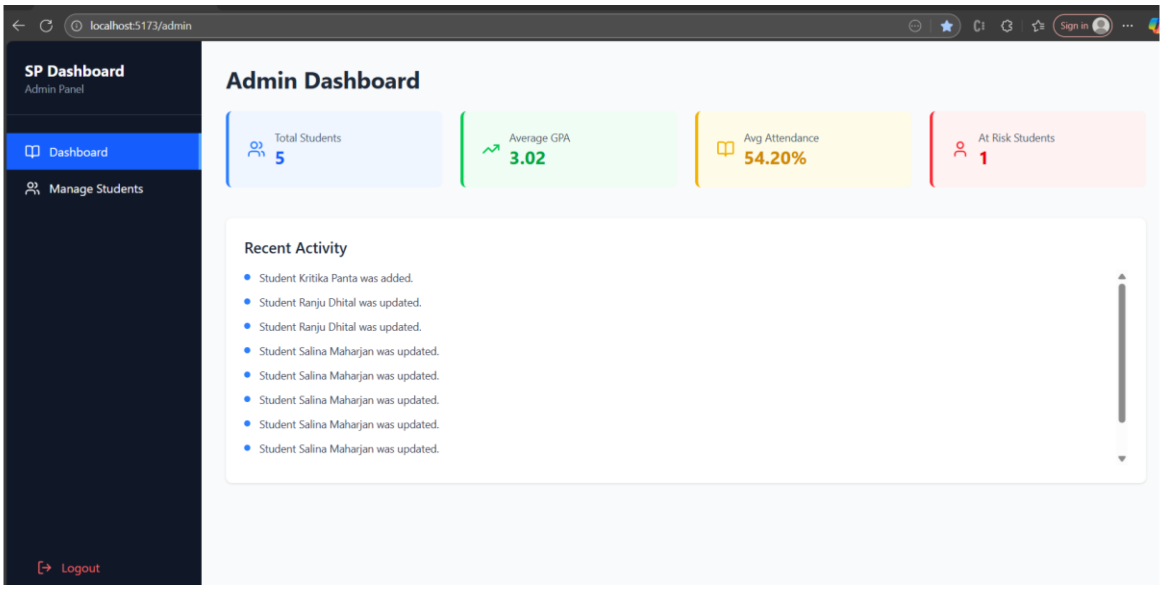
Enhancing education through data-driven insights and predictive analytics

System Online   Last Updated: Sept 2025

- Login Page

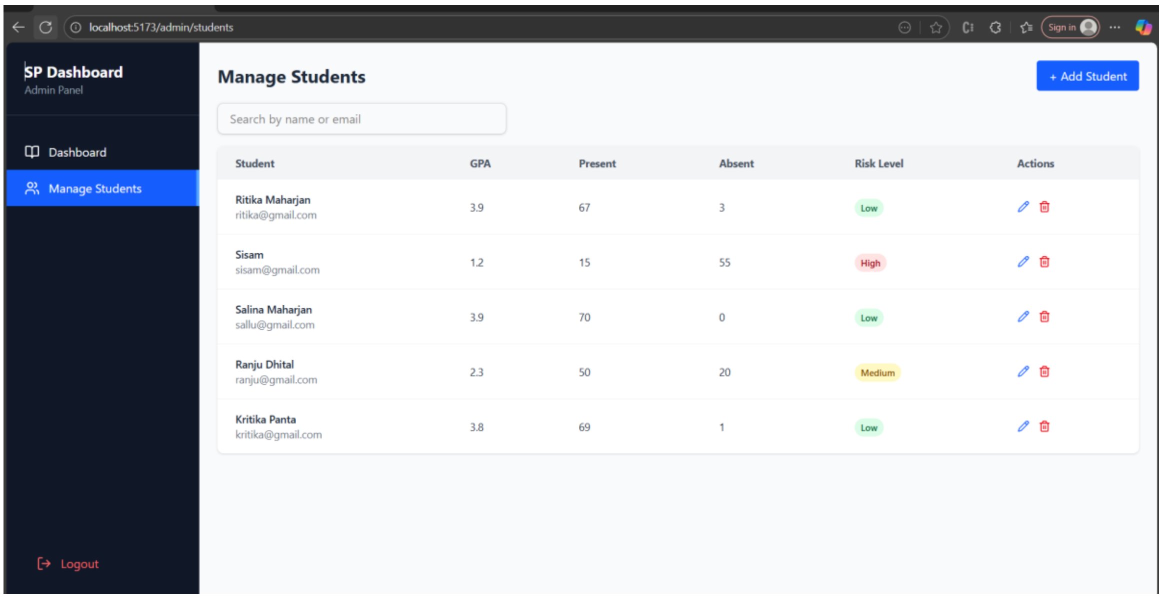


- Admin Dashboard



- Student Kritika Panta was added.
- Student Ranju Dhital was updated.
- Student Ranju Dhital was updated.
- Student Salina Maharjan was updated.
- Student Salina Maharjan was updated.
- Student Salina Maharjan was updated.
- Student Salina Maharjan was updated.
- Student Salina Maharjan was updated.

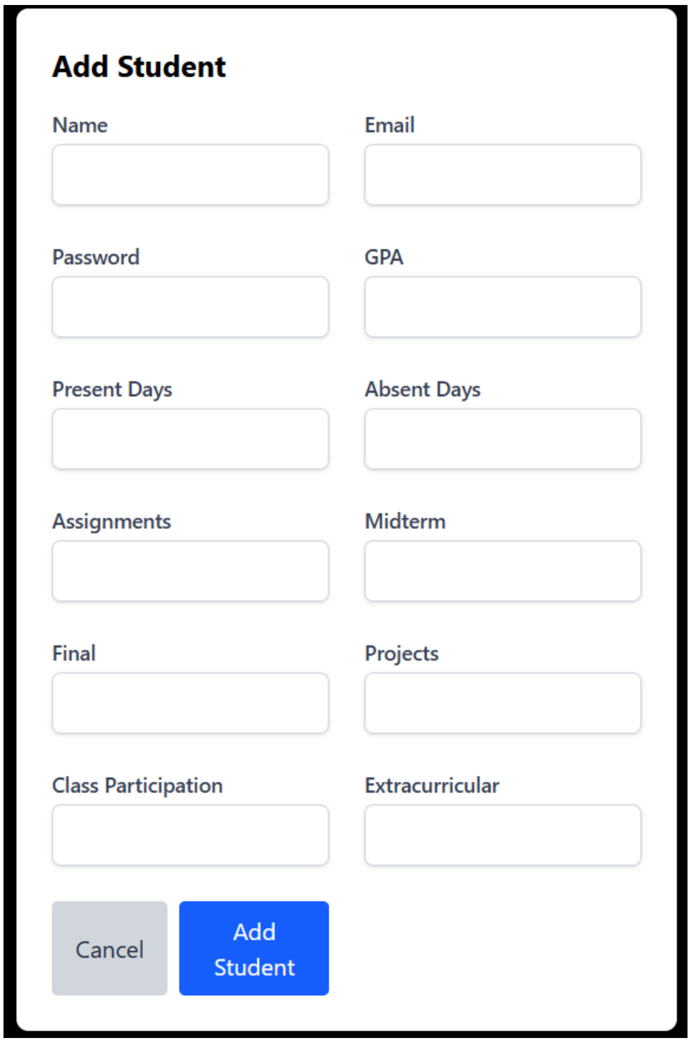
- Manage Students



Manage Students

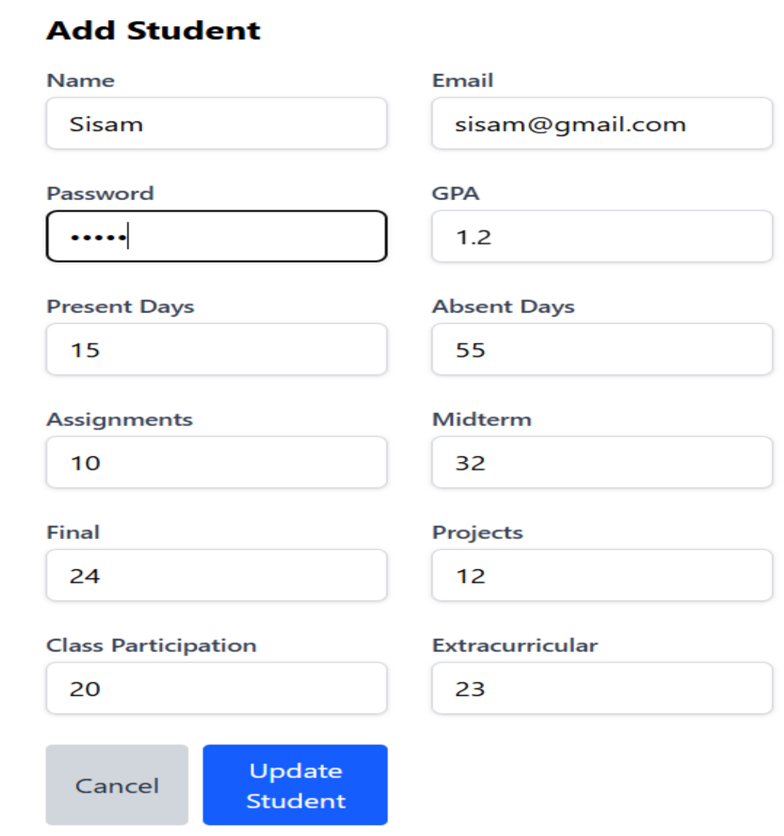
Student	GPA	Present	Absent	Risk Level	Actions
Ritika Maharjan ritika@gmail.com	3.9	67	3	Low	
Sisam sisam@gmail.com	1.2	15	55	High	
Salina Maharjan sallu@gmail.com	3.9	70	0	Low	
Ranju Dhital ranju@gmail.com	2.3	50	20	Medium	
Kritika Panta kritika@gmail.com	3.8	69	1	Low	

- Add Student



c

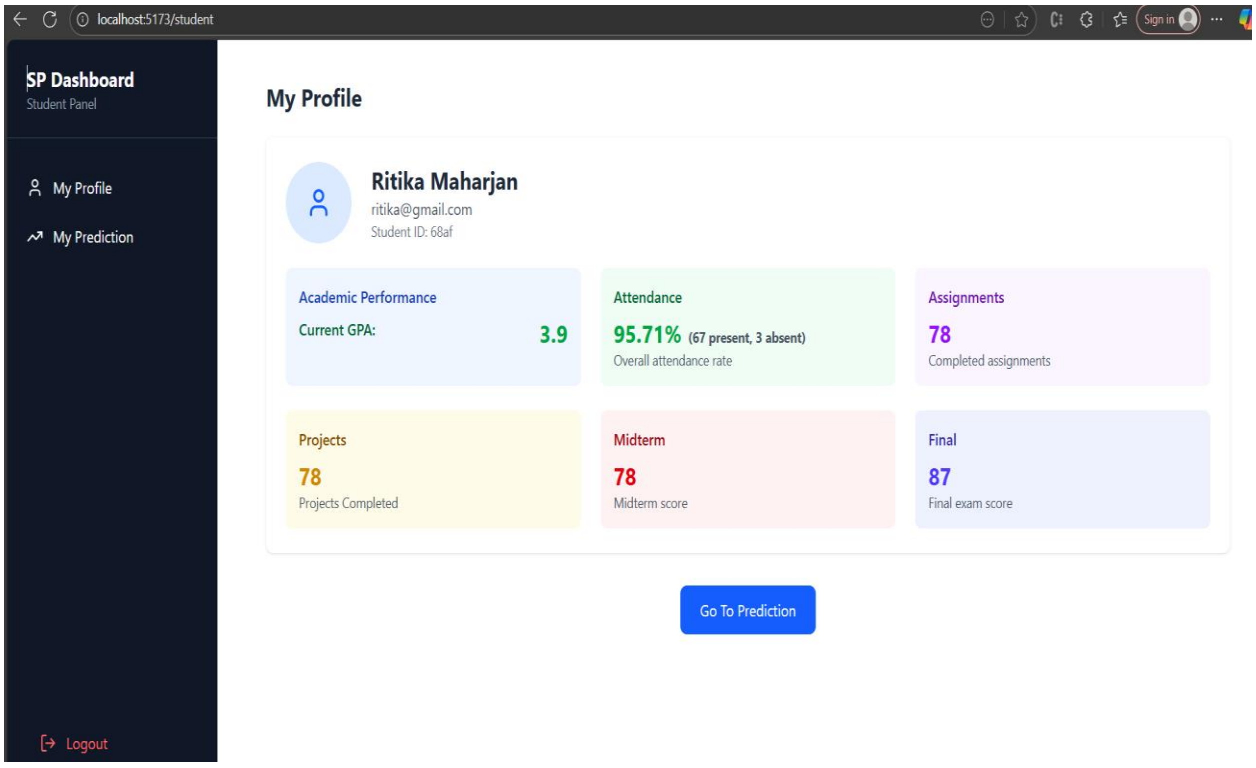
- Update Student



Add Student

Name	Email
Sisam	sisam@gmail.com
Password	GPA
•••••	1.2
Present Days	Absent Days
15	55
Assignments	Midterm
10	32
Final	Projects
24	12
Class Participation	Extracurricular
20	23

- Student Dashboard



My Profile

Ritika Maharjan
ritika@gmail.com
Student ID: 68af

Academic Performance
Current GPA: 3.9

Attendance
95.71% (67 present, 3 absent)
Overall attendance rate

Assignments
78
Completed assignments

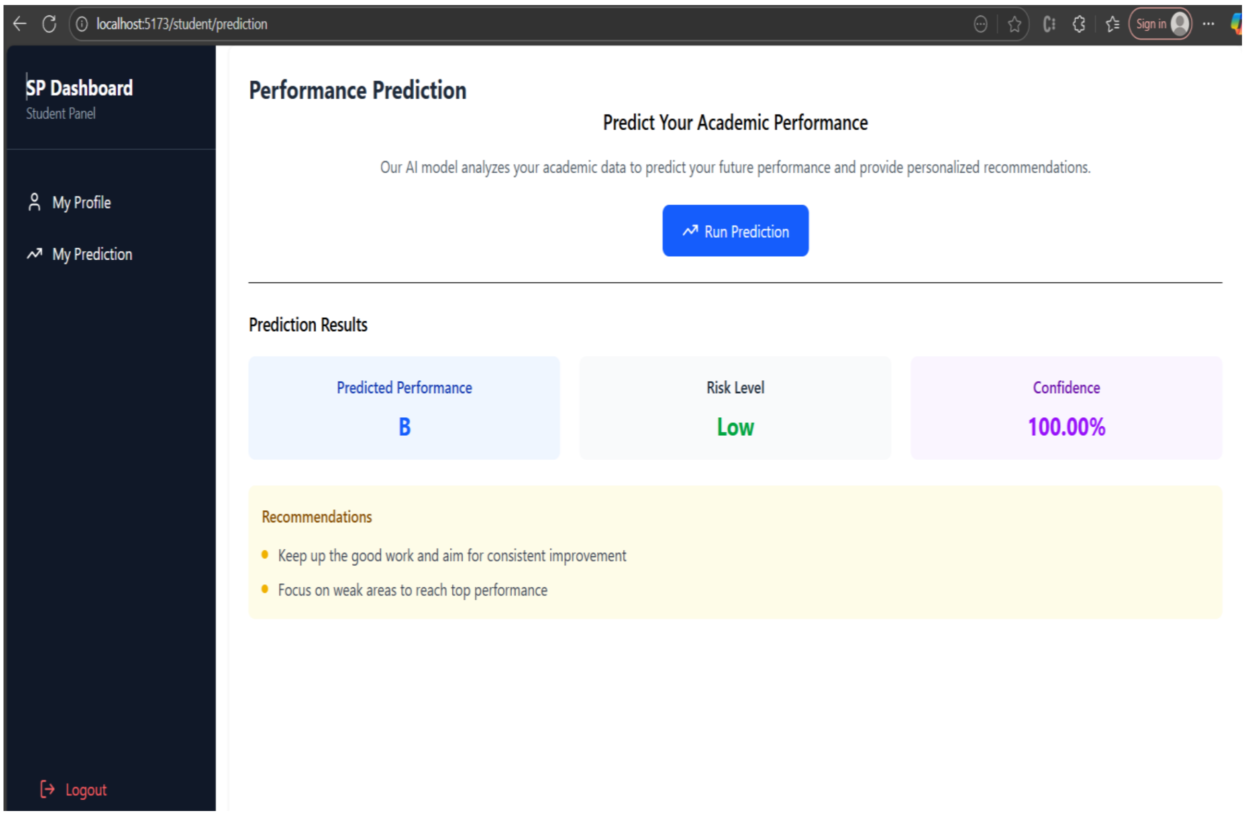
Projects
78
Projects Completed

Midterm
78
Midterm score

Final
87
Final exam score

d

- Performance Prediction



Performance Prediction

Predict Your Academic Performance

Our AI model analyzes your academic data to predict your future performance and provide personalized recommendations.

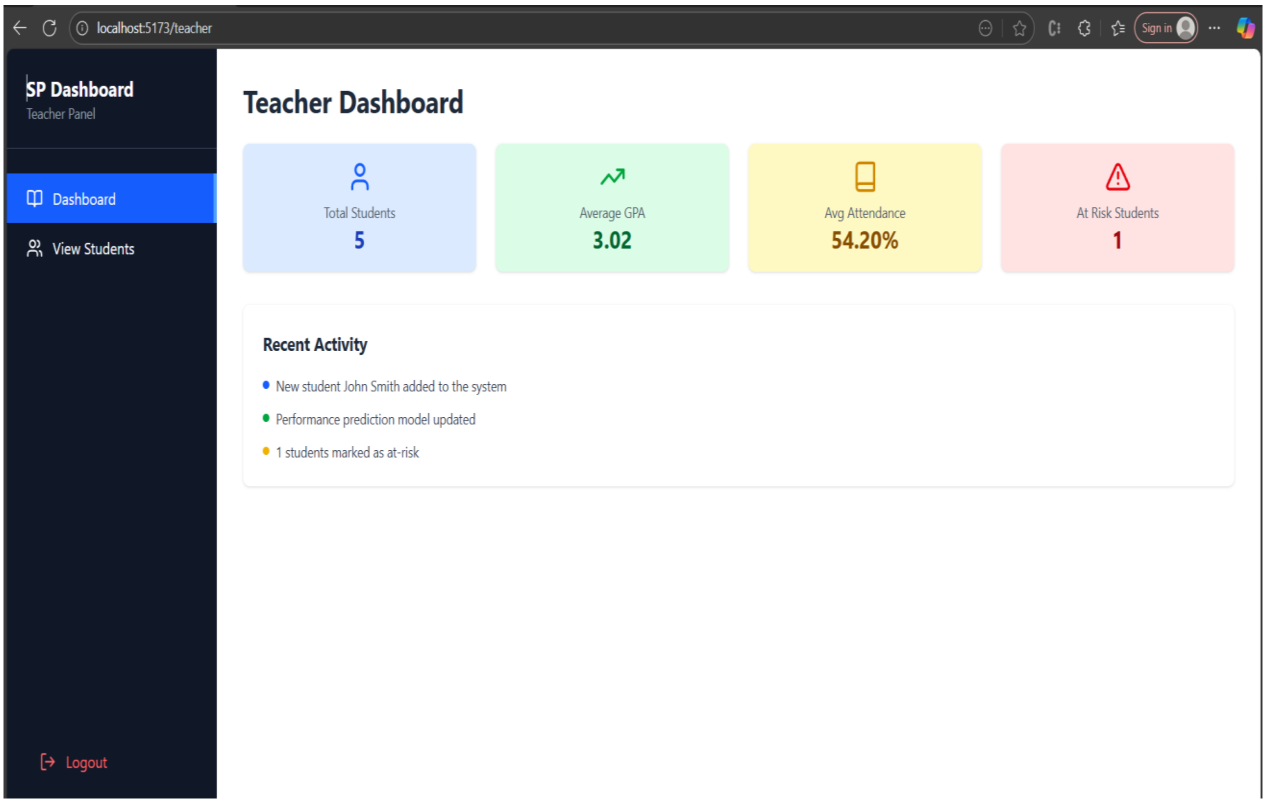
Prediction Results

Predicted Performance	Risk Level	Confidence
B	Low	100.00%

Recommendations

- Keep up the good work and aim for consistent improvement
- Focus on weak areas to reach top performance

- Teacher Dashboard



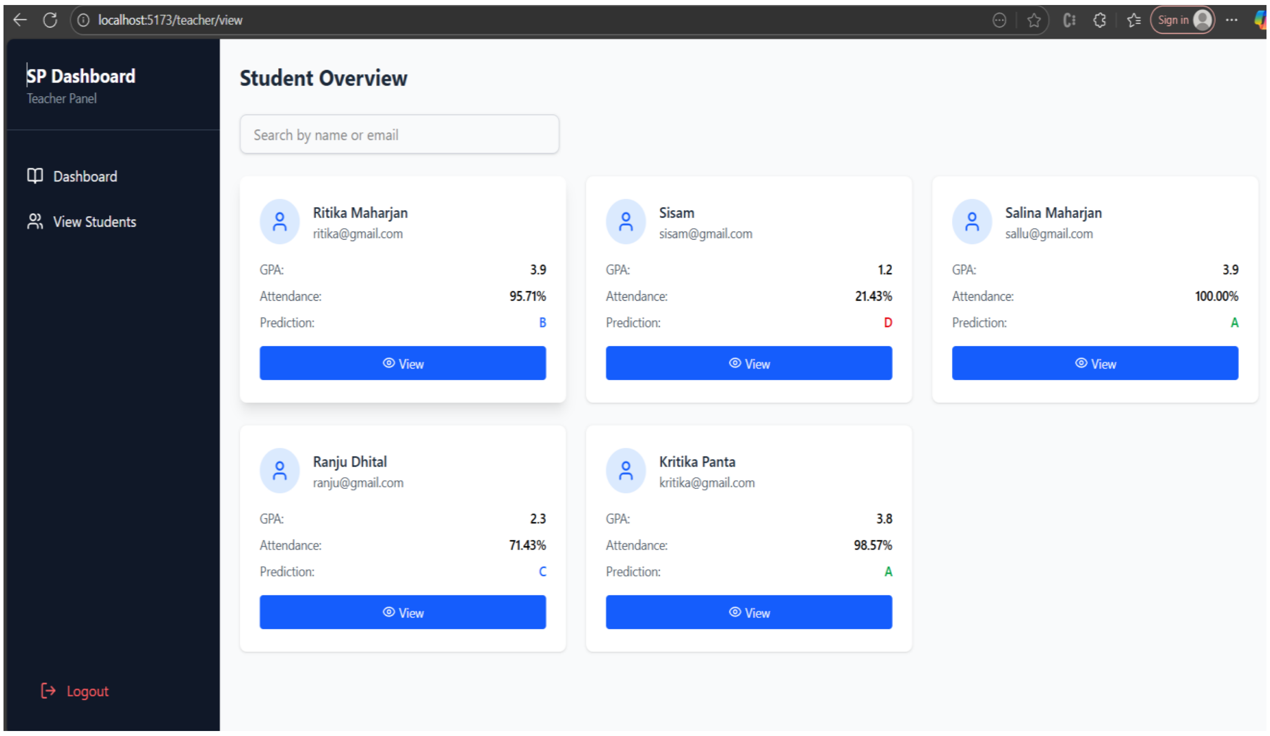
Teacher Dashboard

Total Students	Average GPA	Avg Attendance	At Risk Students
5	3.02	54.20%	1

Recent Activity

- New student John Smith added to the system
- Performance prediction model updated
- 1 students marked as at-risk

- View Students


localhost:5173/teacher/view

SP Dashboard
Teacher Panel

Dashboard
View Students
Logout

Student Overview

Search by name or email

Ritika Maharjan
ritika@gmail.com
GPA: 3.9
Attendance: 95.71%
Prediction: B
View

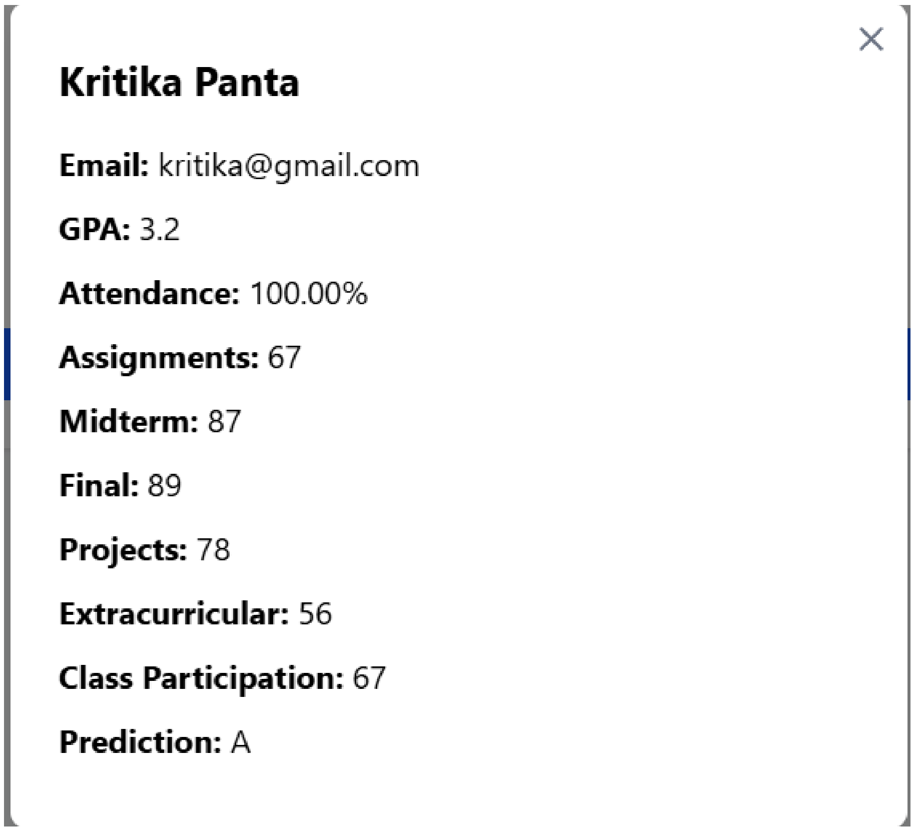
Sisam
sisam@gmail.com
GPA: 1.2
Attendance: 21.43%
Prediction: D
View

Salina Maharjan
sallu@gmail.com
GPA: 3.9
Attendance: 100.00%
Prediction: A
View

Ranju Dhital
ranju@gmail.com
GPA: 2.3
Attendance: 71.43%
Prediction: C
View

Kritika Panta
kritika@gmail.com
GPA: 3.8
Attendance: 98.57%
Prediction: A
View

- View Student Details


Kritika Panta

Email: kritika@gmail.com

GPA: 3.2

Attendance: 100.00%

Assignments: 67

Midterm: 87

Final: 89

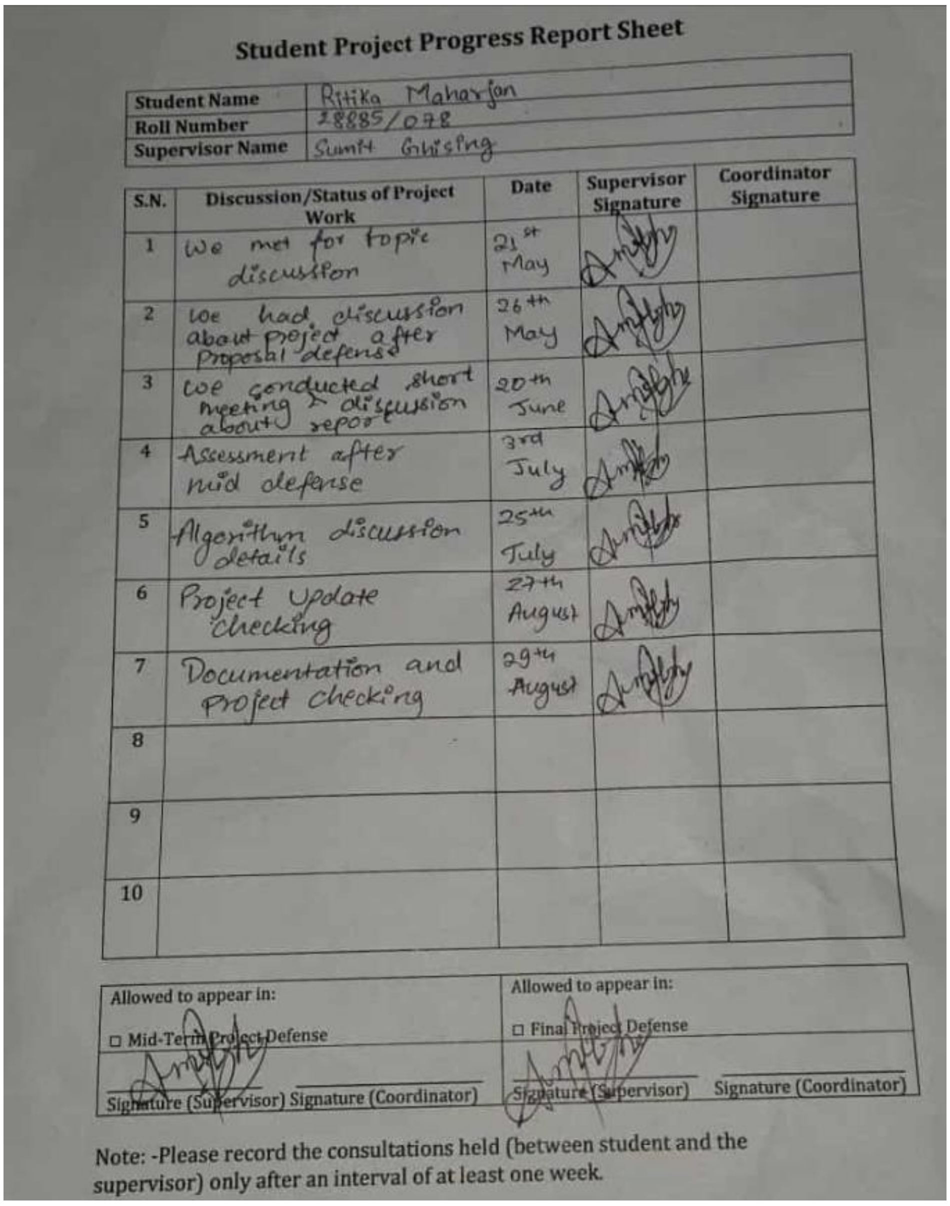
Projects: 78

Extracurricular: 56

Class Participation: 67

Prediction: A

- Log of visits to supervisor



Student Project Progress Report Sheet

Student Name	Ritika Maharjan
Roll Number	28885/078
Supervisor Name	Sumit Ghising

S.N.	Discussion/Status of Project Work	Date	Supervisor Signature	Coordinator Signature
1	We met for topic discussion	21st May	[illegible]	
2	We had discussion about project after proposal defense	26th May	[illegible]	
3	We conducted short meeting & discussion about report	20th June	[illegible]	
4	Assessment after mid defense	3rd July	[illegible]	
5	Algorithm discussion details	25th July	[illegible]	
6	Project update checking	27th August	[illegible]	
7	Documentation and project checking	29th August	[illegible]	
8				
9				
10				

Allowed to appear in: ☐ Mid-Term Project Defense	Allowed to appear in: ☐ Final Project Defense
[illegible] Signature (Supervisor) Signature (Coordinator)	[illegible] Signature (Supervisor) Signature (Coordinator)

Note: -Please record the consultations held (between student and the supervisor) only after an interval of at least one week.