

# **AUTOMATED GRADING SYSTEM**

**A**

## **MAJOR PROJECT-II REPORT**

Submitted in partial fulfillment of the requirements

for the degree of

**BACHELOR OF TECHNOLOGY**

in

**COMPUTER SCIENCE & ENGINEERING**

By

**GROUP NO. 47**

<b>Bhanu P. Singh</b>	<b>0187CS211049</b>
<b>Amit Ahirwar</b>	<b>0187CS211024</b>
<b>Bhumika Shivhare</b>	<b>0187CS211051</b>
<b>Ashish Kumar Soni</b>	<b>0187CS211040</b>

Under the guidance of

**Prof. Nargish Gupta**

(Assistant Professor)



**Department of Computer Science & Engineering**  
**Sagar Institute of Science & Technology (SISTec), Bhopal (M.P)**

**Approved by AICTE, New Delhi & Govt. of M.P.**  
**Affiliated to Rajiv Gandhi Proudyogiki Vishwavidyalaya, Bhopal (M.P.)**

**June -2025**

**Sagar Institute of Science & Technology (SISTec), Bhopal (M.P)**

**Department of Computer Science & Engineering**



**CERTIFICATE**

We hereby certify that the work which is being presented in the B.Tech. Major Project-II Report entitled **AUTOMATED GRADING SYSTEM**, in partial fulfillment of the requirements for the award of the degree of **Bachelor of Technology**, submitted to the Department of **Computer Science & Engineering**, Sagar Institute of Science & Technology (SISTec), Bhopal (M.P.) is an authentic record of our own work carried out during the period from Jan-2025 to Jun-2025 under the supervision of **Prof. Nargish Gupta**.

The content presented in this project has not been submitted by me for the award of any other degree elsewhere.

**Bhanu P. Singh**

**0187CS211049**

**Amit Ahirwar**

**0187CS211024**

**Bhumika Shivhare**

**0187CS211051**

**Ashish kumar  
Soni**

**0187CS211040**

This is to certify that the above statement made by the candidate is correct to the best of my knowledge.

***Date:***

**Prof. Nargish Gupta  
Project Guide**

**Dr. Amit Kumar Mishra  
HOD, CSE**

**Dr. D.K. Rajoriya  
Principal**

## **ACKNOWLEDGEMENT**

We would like to express our sincere thanks to **Dr. D. K. Rajoriya, Principal, SISTec** and **Dr. Swati Saxena, Vice Principal SISTec** Gandhi Nagar, Bhopal for giving us an opportunity to undertake this project.

We also take this opportunity to express a deep sense of gratitude to **Dr. Amit Kumar Mishra, HOD, Department of Computer Science & Engineering** for his kindhearted support.

We extend our sincere and heartfelt thanks to our guide, **Prof. Nargish Gupta**, for providing us with the right guidance and advice at crucial junctures and for showing us the right way.

I am thankful to the **Project Coordinator, Prof. Deepti Jain**, who devoted her precious time in giving us the information about various aspects and gave support and guidance at every point of time.

I would like to thank all those people who helped me directly or indirectly to complete my project whenever I found myself in any issue.

## **TABLE OF CONTENTS**

<b>TITLE</b>	<b>PAGE NO.</b>
Abstract	i
List of Abbreviation	ii
List of Figures	iii
List of Tables	iv
Chapter 1    Introduction	1
1.1    Current Scenario and Challenges	2
1.2    Need For Automation	3
1.3    Objectives	4
Chapter 2    Software & Hardware Requirements	5
Chapter 3    Problem Description	9
Chapter 4    Literature Survey	12
Chapter 5    Software Requirements Specification	15
5.1 Functional Requirements	16
5.2 Non-Functional Requirements	17
Chapter 6    Software Design	19
6.1 Use Case Diagram	20
6.2 ER Diagram	20
6.3 Table Structure	21
Chapter 7    Output Screen	23
Chapter 8    Deployment	30
References	
Project Summary	
Appendix-1: Glossary of Terms	

## **ABSTRACT**

The Automated Grading System is an intelligent software solution designed to streamline and enhance the process of evaluating student assessments. By leveraging natural language processing (NLP), machine learning (ML), and rule-based algorithms, the system can accurately and efficiently grade multiple types of questions, including multiple-choice, short answers, and descriptive responses. This reduces the manual workload for educators, ensures consistency and fairness in grading, and provides students with timely feedback. Additionally, the system supports customizable grading rubrics and analytics dashboards to monitor student performance. This innovation not only optimizes educational workflows but also contributes to data-driven teaching and learning experiences.

**LIST OF ABBREVIATIONS**

<b>ACRONYM</b>	<b>FULL FORM</b>
SDLC	Software Development Life Cycle
SQL	Structured Query Language
HTML	Hyper Text Markup Language
UML	Unified Modeling Language

**LIST OF FIGURES**

<b>FIG. NO.</b>	<b>TITLE</b>	<b>PAGE NO.</b>
6.1	Use Case Diagram	23
7.1	Grade Submission Portal	26
7.2	Grading Results	27
7.3	Overall Score	28
7.4	Answer Comparison	29

**LIST OF TABLES**

<b>TABLE NO.</b>	<b>TITLE OF TABLE</b>	<b>PAGE NO.</b>
6.1	Student Table	21
6.2	Teacher Table	21
6.3	Course Table	22
6.4	Grading Table	22



# **CHAPTER 1**

# **INTRODUCTION**

# CHAPTER – 1

## INTRODUCTION

---

### 1.1 Current Scenario and Challenges

Education is one of the most critical pillars of society, and the evaluation of students plays a vital role in measuring learning outcomes, identifying gaps, and improving the overall quality of education. Traditionally, the process of grading student assessments has been manual, time-consuming, and often inconsistent due to subjective biases and human error. With the increasing number of students and the widespread adoption of online learning platforms, it has become essential to adopt more efficient and scalable evaluation methods.

In recent years, technology has revolutionized many aspects of education, but assessment methods still lag behind in terms of automation and efficiency. An *\*Automated Grading System\** is a technological solution developed to address this gap. It utilizes machine learning, natural language processing (NLP), and rule-based algorithms to evaluate student responses automatically. Such a system can handle different types of questions including multiple-choice questions (MCQs), fill-in-the-blanks, short answers, and even long descriptive or essay-type questions.

The main advantage of an automated grading system lies in its ability to reduce the workload of educators, minimize grading inconsistencies, and provide students with quicker and more objective feedback. Teachers can then focus more on teaching and student engagement, rather than spending hours on evaluating assignments and tests. Additionally, students benefit from receiving instant feedback which helps them identify their strengths and areas of improvement in real-time.

Moreover, the system can be customized based on different grading rubrics and criteria set by educational institutions. It also generates performance analytics and reports that can assist teachers in tracking student progress over time. These features make the system not just a grading tool, but a smart assessment companion that contributes towards improving teaching strategies and enhancing the learning experience.

Although some educational institutions have begun using digital tools to record academic data, many fail to leverage these systems for predictive analysis. In most cases, the tools are either limited in functionality or inaccessible to faculty due to lack of training or technical resources.

### 1.2 Need for Automation

With the rapid digitization of education and the increasing reliance on online learning platforms, the educational sector has access to an unprecedented amount of student data. From daily attendance and online participation to quiz scores and assignment submissions, every action can be recorded and stored. However, collecting data is not enough. Without proper tools to process, analyze, and interpret this data, it remains just raw information. This is where automation and predictive analytics play a transformative role.

Automating the performance prediction process allows institutions to efficiently manage and analyze student data, identifying patterns and trends that would be difficult to spot manually. For instance, machine learning algorithms can be trained to detect correlations between attendance, test scores, study habits, and final exam outcomes. These correlations can then be used to build predictive models that forecast future performance with considerable accuracy.

A major benefit of automation is the ability to deliver timely alerts and insights. Educators can be notified when a student's performance deviates from expected norms, allowing for early intervention. Rather than waiting for end-of-term exams to assess learning, teachers can take proactive steps to address academic challenges as they arise.

Automation also enhances objectivity and fairness. Unlike human evaluations that may be influenced by personal biases or fatigue, automated systems provide consistent analysis based on data-driven rules. This ensures fair treatment for all students and fosters a more transparent evaluation process.

Moreover, predictive systems enable personalized learning experiences. Based on performance trends, the system can recommend specific resources, tutorials, or study plans tailored to individual learning needs. This level of personalization can boost student engagement and academic achievement, particularly for those who struggle with conventional teaching methods.

In terms of institutional benefits, automated performance prediction systems contribute to efficient resource management. Educational administrators can allocate tutoring resources, counseling, or remedial programs more effectively by knowing which students require assistance and in what areas.

### **1.3 Objectives**

The primary objective of this project is to develop a robust and reliable system that can accurately predict student performance based on historical and real-time academic data. This predictive model will serve as a decision-support tool for teachers, administrators, and even students themselves.

To achieve this goal, the project aims to fulfill the following key objectives:

- **Data Collection and Preprocessing:** Gather relevant academic data from students, such as marks, attendance records, assignment submissions, participation in class, and extracurricular involvement. The data will be cleaned, structured, and standardized to ensure consistency and accuracy.
- **Model Development Using Machine Learning Algorithms:** Employ machine learning algorithms such as Decision Trees, Random Forest, Logistic Regression, or Neural Networks to build a predictive model.

The choice of algorithm will be based on the nature of the dataset and the desired prediction accuracy.

- **Identification of Key Performance Indicators (KPIs):** Determine the factors that most significantly influence student performance, such as attendance, past academic results, or engagement levels. These insights will help educators focus on what matters most.
- **Development of a User Interface:** Design a simple and interactive dashboard or web interface where educators can upload data and view predictions. The interface should include visual elements like graphs, risk indicators, and suggestions for intervention.
- **Alert and Notification System:** Integrate a system that sends alerts when a student is predicted to perform poorly, enabling timely intervention by teachers or mentors.
- **Performance Evaluation and Accuracy Testing:** Test the model on real or synthetic datasets to assess its accuracy, precision, and recall. Based on the results, improvements will be made to enhance prediction reliability.
- **Scalability and Generalization:** Ensure that the system can be scaled to work across different departments or institutions and generalized to different types of educational environments.

By meeting these objectives, System will contribute toward proactive academic management, better student engagement, and ultimately, improved educational outcomes.

# **CHAPTER 2**

# **SOFTWARE AND**

# **HARDWARE**

# **REQUIREMENTS**

## **CHAPTER – 2**

# **SOFTWARE AND HARDWARE REQUIREMENTS**

### **2.1 SOFTWARE REQUIREMENTS**

To build and maintain a robust, scalable, and responsive public transportation automation system, a well-defined set of software tools, frameworks, and platforms is essential. These components will support development, testing, deployment, and real-time functionality.

#### **2.1.1 DEVELOPMENT TOOL**

##### **Backend:**

- Python 3.8+: Core programming language used for machine learning model development and API creation.
- Flask / Django: Lightweight Python-based web frameworks for handling HTTP requests and serving the ML model predictions.
- Scikit-learn: ML library for implementing algorithms like Logistic Regression, Decision Trees, Random Forest, etc.
- Pandas & NumPy: Libraries used for data manipulation, preprocessing, and matrix operations.
- Jupyter Notebook: Used during prototyping and experimentation with data and algorithms.
- MySQL: Relational databases for storing structured student performance data securely.

##### **Frontend:**

- HTML, CSS, JavaScript: Core web technologies used to structure, style, and add interactivity to the user interface.
- Bootstrap: A responsive front-end framework used to speed up UI development and ensure consistent styling.
- Plotly: JavaScript libraries used to display dynamic graphs and performance analytics to users.

#### **2.1.2 DATABASE MANAGEMENT**

- MySQL 8.0+: A reliable, open-source RDBMS to store structured data such as user profiles, bus details, routes, tracking logs, and complaint records.
- MySQL Workbench: Graphical interface for designing, querying, and managing the MySQL database effectively, also used for ER modeling and schema design.

#### **2.1.3 MACHINE LEARNING & DATA PROCESSING**

- Scikit-learn: For classification, feature selection, and evaluation metrics.
- Pandas: Essential for reading, cleaning, and transforming raw academic data.
- Seaborn: Visualization tools for data exploration and insights.
- Pickle: For model serialization and deployment.

### 2.1.4 TESTING TOOLS

- **PyTest** : For unit testing individual modules and functions.
- **Postman**: API testing tool used to check endpoints for model prediction, user data retrieval, and alerts.
- **Selenium**: Automates end-to-end testing of the user interface and dashboard interactions.

### 2.1.5 VERSION CONTROL & DEPLOYMENT

- **Git & GitHub**: Used for version control, code sharing, and managing collaborative development.
- **AWS EC2**: Platforms for hosting web applications and serving ML models to users.
- **Docker**: For containerizing the app and maintaining consistency across different environments.
- **CI/CD (GitHub Actions)**: Automates deployment, testing, and integration pipelines.

## 2.2 HARDWARE REQUIREMENTS

A reliable hardware setup is essential for the development, testing, deployment, and real-time operations of the Transport Management system. These requirements include devices for both the development team and the hardware that must be installed on buses for tracking.

### 2.2.1 DEVELOPMENT AND DEPLOYMENT ENVIRONMENT

#### 2.2.1.1 Developer Workstations:

- **Processor**: Intel i5 10th Gen / AMD Ryzen 5 or higher to support multitasking, data processing, and model training.
- **RAM**: Minimum 8 GB (Recommended 16 GB or more) for efficient handling of large datasets and simultaneous tools.
- **Storage**: 512 GB SSD or higher improves speed in data retrieval, boot time, and I/O operations during development.
- **Operating System**: Windows 10/11, Ubuntu 20.04+, or macOS compatible with major ML frameworks and development environments.
- **Display**: Full HD 1080p or better for better visualization of dashboards, charts, and code.
- **Internet**: High-speed internet connection (50 Mbps or more) essential for accessing cloud services, repositories, and APIs.
- **Graphics Card**: Optional (NVIDIA GTX 1650 or better) useful if the model incorporates deep learning or high-performance visualizations.
- **Keyboard & Mouse**: Ergonomic setup for long hours of development with minimal fatigue.
- **Cooling Pads / External Cooling Fans**: To prevent overheating during long training or build processes.

#### 2.2.2 CLIENT INTERFACE DEVICES (FOR TESTING & USER INTERACTION)

- **Laptops / Tablets**: Used by faculty and administrative staff to access dashboards, performance reports, and analytics insights. Minimum specifications include a dual-core processor, 4 GB RAM, and browser support for responsive UI. Recommended screen

size is between 13 to 15 inches.

- Smartphones: Used for testing mobile responsiveness and end-user experience. Devices should support Android (API level 26+) or iOS (12+) to ensure compatibility. These may also be tested for optional push notifications and mobile alert features.
- Smartboards / Projectors (For Classroom Integration): Useful for real-time projection of analytics and predictive dashboards during presentations or academic sessions. Integration can be done via browser-based dashboard access.
- IoT-based Biometric Devices (Future Scope): Considered for automating attendance or student engagement tracking. Data collected from these devices can be integrated into the prediction system to enhance model performance.



# **CHAPTER 3**

## **PROBLEM**

### **DESCRIPTION**

## CHAPTER – 3

### PROBLEM DESCRIPTION

---

#### 3.1 ISSUES IN STUDENT PERFORMANCE MONITORING

In traditional educational systems, the process of evaluating students' written responses—especially long and descriptive answers—has always been a time-consuming and labor-intensive task. Teachers are often required to assess large volumes of answer scripts, which not only takes a significant amount of time but is also prone to human errors, fatigue, and inconsistency. The subjectivity involved in grading such answers may lead to unfair assessments, causing dissatisfaction among students and a lack of transparency in the evaluation process.

Moreover, as the number of students increases, manual evaluation becomes more inefficient and difficult to scale. In scenarios like online learning platforms, competitive exams, or massive open online courses (MOOCs), timely evaluation of student responses becomes nearly impossible without some level of automation. Traditional grading methods also lack immediate feedback mechanisms, which delays the learning loop and limits students' opportunities to reflect on and improve their performance.

Another issue lies in the reactive nature of existing systems. Institutions often wait until the end of the semester or academic year to analyze student performance, missing valuable opportunities for early intervention. This is largely due to the absence of predictive analytics tools that can assess ongoing academic trends and forecast likely outcomes. In contrast, a proactive system capable of continuous monitoring could send alerts when a student's attendance drops below a threshold or when assignment scores begin to decline, allowing educators to take timely and informed action.

The increasing shift towards digital learning has highlighted the need for an automated, accurate, and scalable grading system that can reduce the burden on educators while ensuring fair assessment for students. While multiple-choice and short-answer questions are relatively easier to evaluate programmatically, the real challenge lies in grading descriptive answers, which require understanding of context, grammar, content relevance, and keyword matching.

Another area of concern is the lack of communication between stakeholders. Parents, who play a crucial role in a student's academic journey, are often not adequately informed about ongoing performance trends. Without real-time performance dashboards or predictive notifications, parents typically receive feedback only during formal meetings or after exam results are published. As noted in recent educational research, real-time communication platforms have been linked with increased parental engagement and student accountability. Yet such tools remain underutilized in many school and college systems.

The current methods also fail to address non-academic factors such as mental health, attendance consistency, classroom participation, or extracurricular engagement all of which can significantly affect academic outcomes. Behavioral patterns, absenteeism trends, and changes in student involvement often go unnoticed due to the lack of analytical tools that can detect anomalies.

Finally, the absence of meaningful visualizations or analytics tools makes decision-making

inefficient. Teachers and administrators struggle to gain actionable insights from raw data, and as a result, efforts to improve teaching strategies or optimize academic policies are often based on assumptions rather than evidence. In today's data-driven world, where educational institutions are expected to be agile and adaptive, this inability to leverage technology for academic analytics places them at a significant disadvantage.

### **3.2 NEED FOR AUTOMATION**

In light of these challenges, the need for an automated and intelligent system becomes undeniable. A smart system like Automated Grading System (AGS) can provide actionable insights by consolidating diverse datasets such as attendance, academic scores, participation, and engagement levels into a centralized platform. With the help of AI/ML models, it can detect patterns and predict future performance with a high degree of accuracy, allowing for timely and personalized interventions.

Automation will enable real-time analysis of student behavior and performance trends. By continuously monitoring data and triggering alerts when specific risk thresholds are met (e.g., low attendance combined with poor grades), the system can assist faculty and mentors in identifying students who need academic support or counseling. This proactive approach shifts the focus from post-result interventions to early-stage academic support, improving overall student outcomes.

A centralized dashboard can empower all stakeholders students, parents, teachers, and administrators with visualized performance metrics, predictive indicators, and personalized recommendations. Faculty members can access intelligent reports suggesting remediation strategies, while parents can stay informed through mobile apps or SMS alerts. This increased transparency strengthens collaboration between home and school.

Additionally, the system can integrate with existing digital infrastructure like learning management systems, biometric attendance devices, or grading software to collect data seamlessly. It can also help institutions manage records more efficiently, reducing manual work and minimizing errors in reporting and documentation.

From a long-term perspective, such a system facilitates data-driven policymaking. Academic heads can study longitudinal trends, identify curriculum gaps, and allocate resources based on real needs. It supports continuous improvement by offering insights into teaching effectiveness, subject difficulty levels, and learner engagement.

By automating, AGS ensures a more inclusive, responsive, and intelligent educational environment that adapts to every learner's needs while optimizing institutional efficiency.

# **CHAPTER 4**

# **LITERATURE**

# **SURVEY**

## CHAPTER – 4

# LITERATURE SURVEY

---

### 4.1 UNDERSTANDING AUTOMATED GRADING SYSTEM DOMAIN

In the educational ecosystem, accurately grading is a crucial task that assists institutions in identifying learning gaps, allocating resources effectively, and enhancing academic outcomes. Traditional assessment methods like examinations and periodic evaluations offer only a snapshot of a student's capabilities, often failing to account for behavioral patterns, engagement levels, attendance consistency, and socio-economic factors. This limitation has created a pressing need for systems that can continuously monitor and analyze diverse student-related data to generate more holistic and timely predictions.

Recent advancements in educational technologies and artificial intelligence have enabled the development of smart systems that can leverage historical academic records, real-time performance data, and contextual parameters to predict student outcomes. These systems empower educators with actionable insights that help them intervene early, design personalized support strategies, and optimize learning plans. With the help of predictive models such as decision trees, random forests, and neural networks, institutions can not only identify students at risk but also gain insights into the underlying factors affecting their progress.

The integration of such automated systems also improves administrative efficiency by offering centralized dashboards, automating alerts for low performance, and generating academic progress reports. In essence, student performance prediction systems aim to bridge the gap between raw data collection and meaningful academic guidance, creating a responsive and intelligent learning environment. As educational institutions continue to adopt digital transformation practices, such systems are becoming indispensable tools for modern pedagogy.

### 4.2 KEY TERMINOLOGIES IN THE DOMAIN

To understand the core functionality of a Automated Grading System, it is essential to explore key terminologies and concepts frequently used in the domain. Predictive Analytics refers to the use of statistical algorithms and machine learning models to forecast future outcomes based on historical data. In this context, it helps predict whether a student is likely to pass, fail, or require additional support.

Feature Engineering is the process of selecting and transforming relevant input variables such as attendance percentage, test scores, project submissions, and demographic data into a form that improves the accuracy of predictive models. Classification Models are algorithms like Decision Trees, Logistic Regression, Support Vector Machines (SVM), and Random Forests that categorize students into different academic risk levels. These models analyze the relationships between input features and performance outcomes.

Academic Dashboard is a user interface that visually represents student data, performance metrics, prediction results, and alerts. It is used by educators, administrators, and sometimes even students to gain insights and monitor academic trends in real-time. Early Warning System (EWS) is a predictive mechanism integrated into the system to flag students who exhibit signs of academic decline. It enables timely intervention by notifying faculty or guardians through alerts or info.

Understanding these engagement patterns is essential to accurately predict future academic outcomes. Additionally, Model Accuracy and Precision are crucial concepts used to evaluate the reliability of the predictive system. Accuracy refers to the overall correctness of the predictions, while precision focuses on how effectively the model identifies true cases of academic risk. Another significant term is Feedback Loop, which allows for continuous improvement of prediction models by integrating new data and adjusting weights or model parameters accordingly. By grasping these terms, educational stakeholders can better understand how predictive models function and how they can be leveraged to make smarter decisions in academic planning and student support.

### **4.3 REAL-TIME WORKING OF AUTOMATED GRADING SYSTEMS**

A real-time Automated Grading System functions by integrating multiple data pipelines that continuously feed into an intelligent backend model capable of processing, analyzing, and generating academic insights. The first phase involves Data Collection, where various sources such as student attendance records, assignment scores, class participation logs, examination marks, and extracurricular involvement data are aggregated. This process may also include behavioral data like login patterns on digital platforms, submission times, and participation in online quizzes. All these parameters are cleaned, preprocessed, and structured using automated scripts to ensure consistency and accuracy. Once the data is standardized, it enters the Model Training phase, where machine learning algorithms such as Random Forest, Decision Trees, Logistic Regression, or Neural Networks are employed to identify hidden patterns and correlations between inputs and performance indicators. These models are evaluated using past academic records to ensure they can generalize well to new, unseen data.

Following this, the system enters a Continuous Monitoring phase where real-time inputs—such as weekly attendance updates, periodic test results, or feedback from instructors—are fed into the model to predict outcomes like course success probability, risk of dropout, or the likelihood of failing a subject. The predictions are then visualized through an interactive dashboard accessible to teachers, students, and administrators. This dashboard not only displays performance forecasts but also recommends interventions such as additional coaching, time management tips, or personalized assignments based on the student's specific needs. Furthermore, an integrated Complaint and Feedback System enables students or faculty to report issues with grading, model fairness, or academic concerns, ensuring transparency and accountability. In cases of flagged academic risk or unusual performance dips, automated alerts are triggered via email or app notifications, prompting timely action from academic counselors or guardians. The real-time nature of the system ensures that no critical sign of academic decline goes unnoticed, providing a 360-degree view of student health. This dynamic loop of data intake, analysis, prediction, and feedback closes the gap between performance monitoring and meaningful academic intervention, ultimately fostering a more supportive and efficient learning environment.

# **CHAPTER 5**

# **SOFTWARE**

# **REQUIREMENTS**

# **SPECIFICATION**

# CHAPTER – 5

## SOFTWARE REQUIREMENTS SPECIFICATION

### 5.1 FUNCTIONAL REQUIREMENTS

These are the essential operations and features must support in order to fulfill its intended purpose effectively and efficiently.

#### **1. User Registration and Authentication:**

The system must support a secure user registration and login process for students, faculty, and administrators. New users should be able to register using institutional email IDs or mobile numbers, and the system should verify these via OTP or email validation. Each user type will have role-based access: students can view their performance, faculty can upload and manage student data, and administrators will have full control over user and data management. The system must also include secure login sessions with encryption, support for password resets, session timeouts, and user account recovery through registered contact methods.

#### **2. Data Input and Management Module:**

Faculty and admin users must be able to input student-related data through easy-to-use interfaces or upload bulk data using CSV/Excel files. The system should accept data related to marks, attendance, assignment grades, behavioral scores, and extracurricular participation. A data validation mechanism must be in place to check for incomplete or inconsistent entries before processing. Admins should have the ability to view, edit, and delete existing records as needed.

#### **3. Grading Engine:**

The system must include an AI/ML-powered engine to predict individual student performance based on historical academic data. It should consider multiple factors like attendance, past grades, assessment trends, and behavioral data to forecast outcomes such as pass/fail status, potential grade, or risk of dropout. The Grading results should be automatically generated upon data update and displayed through intuitive visuals like charts and risk indicators.

#### **4. Student Dashboard:**

Each student must have access to a personalized dashboard where they can view predicted grades, academic trends, subject-wise performance breakdowns, and suggestions for improvement. The dashboard must be interactive and support visualization tools such as pie charts, line graphs, and progress bars. Students should also be able to receive automated feedback messages based on their predicted performance category.

#### **5. Admin Panel for AGS Management:**

Administrators should be provided with a secure and feature-rich admin dashboard to monitor and manage the entire fleet. This panel must support the addition, update, and deletion of bus information including bus number, driver details, route assignments, and operational timings. It should allow real-time status tracking of each bus, receive and review passenger complaints, monitor traffic patterns, and access analytics related to passenger usage and route efficiency.



**6. Faculty Dashboard:**

Faculty members must be able to access dashboards showing class-level analytics, performance distribution, and at-risk students. The dashboard should allow filtering based on course, semester, or subject. Faculty should also be able to generate downloadable reports in formats like PDF or Excel for academic reviews and parent-teacher meetings.

**7. Notification and Alert System:**

The system should feature a complaint submission module where students and faculty can raise academic concerns, report technical issues, or suggest improvements. Complaints should be categorized and assigned unique tracking IDs. Admins must be able to view, prioritize, and respond to these complaints, and users should be able to track complaint resolution status through their dashboard.

**8. Emission Compliance Check:**

The system should send automated notifications to students and faculty via in-app messages or email. Alerts should include reminders about upcoming assessments, warnings about declining performance, and notifications when new predictions are available. For at-risk students, the system must notify faculty and optionally alert guardians or counselors for early intervention.

**5.2 NON-FUNCTIONAL REQUIREMENTS**

These define the quality attributes of the system and how well it performs under various conditions.

**1. Performance Requirements:**

The system must deliver high-speed data processing and response times under all expected loads. Core operations such as login, data retrieval, predictive analysis, and report generation must be optimized to execute within a few seconds to maintain smooth user experience. The backend should be capable of handling numerous concurrent prediction requests and data queries without any significant delays. Performance tuning mechanisms like caching, asynchronous processing, and optimized database indexing should be employed to ensure that the system performs efficiently, especially during peak hours such as exam periods or report release timelines.

**2. Scalability:**

The system must be scalable to support an increasing number of users, academic institutions, and datasets over time. It should be designed in a way that allows seamless integration of additional modules, users, or data streams without the need to redesign existing architecture. Horizontal and vertical scalability must be considered for both the application and database layers, allowing the system to be distributed across multiple servers or expanded in the cloud. As student enrollment and institutional participation grow, the platform should maintain consistent performance and service quality with minimal overhead.

### **3. Security Requirements:**

Security is of utmost importance, especially given the system's role in handling sensitive student academic data. The application must enforce secure authentication mechanisms such as hashed passwords, multi-factor authentication (MFA), and encrypted session tokens. All data transmitted between the server and client should be encrypted using SSL/TLS protocols. Internally, databases must follow secure storage practices with role-based access control to ensure that only authorized personnel can access sensitive information. The system must also be resilient to common security threats such as SQL injection, cross-site scripting (XSS), brute force attacks, and data breaches, and should include regular vulnerability assessments and patch management.

### **4. Usability & Accessibility:**

The interface of the system must be intuitive and user-friendly, catering to both tech-savvy users and those with limited digital literacy. Clear layouts, simple navigation, and helpful tooltips should be implemented throughout the interface. Accessibility considerations must include support for screen readers, alt text for images, appropriate color contrast, and keyboard navigability to ensure compliance with accessibility standards such as WCAG 2.1. The system should be fully responsive, allowing users to access it comfortably from desktop browsers, tablets, and mobile phones without sacrificing functionality or clarity.

### **5. Availability & Reliability:**

The system must remain operational and accessible to users with an uptime of at least 99.5% throughout the academic year. It should be deployed in a fault-tolerant environment that supports real-time monitoring and automated failure recovery. System components should be designed with redundancy, load balancing, and disaster recovery capabilities to ensure that services are not interrupted due to hardware failure or unexpected demand spikes. In case of unexpected downtime, a failover mechanism must activate to restore services with minimal disruption, and the users should be promptly notified of the system status.

### **6. Maintainability & Modularity:**

The system should follow a modular design structure, allowing components to be maintained, updated, or replaced independently without affecting the entire system. Clean and well-documented code should be maintained following industry standards to ensure that future developers can easily understand and work on the system. Logging, error handling, and automated testing should be integrated throughout the system to facilitate quick debugging and regular system health checks. This approach ensures long-term sustainability and adaptability of the software to changing requirements.

### **7. Compliance & Legal Requirements:**

As the system deals with student data, it must comply with applicable education and data privacy regulations such as India's Digital Personal Data Protection (DPDP) Act or international standards like GDPR, depending on the deployment context. The platform should also be audit-ready, maintaining secure logs of access, data modifications, and system usage to ensure accountability and legal traceability. Additionally, institutions must be able to configure the system in accordance with their internal policies and compliance frameworks.

# **CHAPTER 6**

# **SOFTWARE**

# **DESIGN**

## CHAPTER – 6

# SOFTWARE DESIGN

### 6.1 USE CASE DIAGRAM

A Use Case Diagram represents the interaction between users (actors) and the system. It includes the functionalities of the system and how users interact with them.

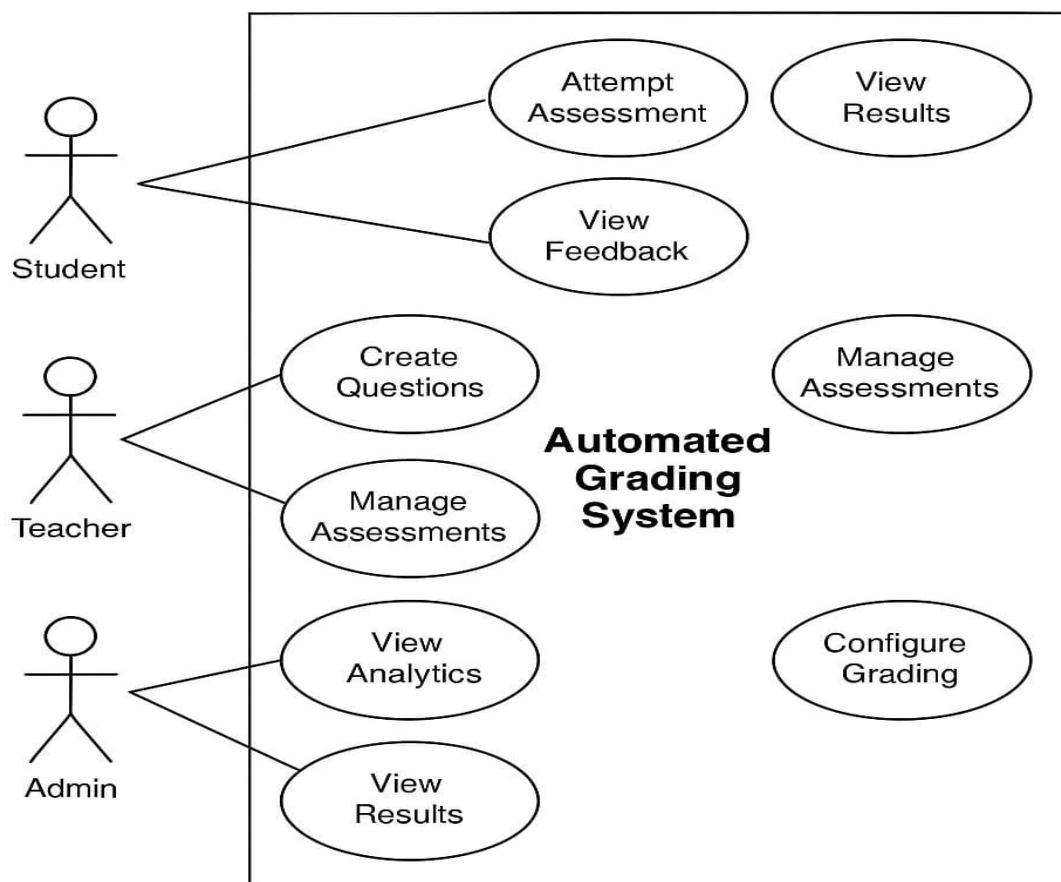
#### 6.1.1 ACTORS IN THE SYSTEM:

**Student (User):** Enters personal and academic details, views performance predictions, and receives feedback or suggestions for improvement.

**Admin:** Manages student records, updates datasets, monitors prediction accuracy, and controls system access and configurations.

**Teacher:** Inputs marks, attendance, behavioral observations, and provides feedback used to generate and refine student performance predictions.

**Use Case Diagram:**



**Figure 6.1: Use Case**

## 6.2 TABLE STRUCTURE

**Table 6.1: Student table**

Field Name	Data Type	Constraints	Purpose
student_id	INT	PRIMARY KEY, AUTO_INCREMENT	Unique ID for each student
name	VARCHAR(100)	NOT NULL	Full name of the student
email	VARCHAR(100)	UNIQUE, NOT NULL	Student's email address
gender	VARCHAR(10)	CHECK (gender IN ('Male','Female','Other'))	Gender of the student
dob	DATE	NOT NULL	Date of birth
department	VARCHAR(50)	NOT NULL	Student's academic department

**Table 6.2: Teacher Table**

Field Name	Data Type	Constraints	Purpose
teacher_id	INT	PRIMARY KEY, AUTO_INCREMENT	Unique ID for each teacher
name	VARCHAR(100)	NOT NULL	Full name of the teacher
email	VARCHAR(100)	UNIQUE, NOT NULL	Teacher's email address
department	VARCHAR(50)	NOT NULL	Academic department

**Table 6.3: Courses Table**

Field Name	Data Type	Constraints	Purpose
course_id	INT	PRIMARY KEY, AUTO_INCREMENT	Unique ID for each course
course_name	VARCHAR(100)	NOT NULL	Name of the course
teacher_id	INT	FOREIGN KEY REFERENCES teacher(teacher_id)	Instructor assigned to the course

**Table 6.4: Grading Table**

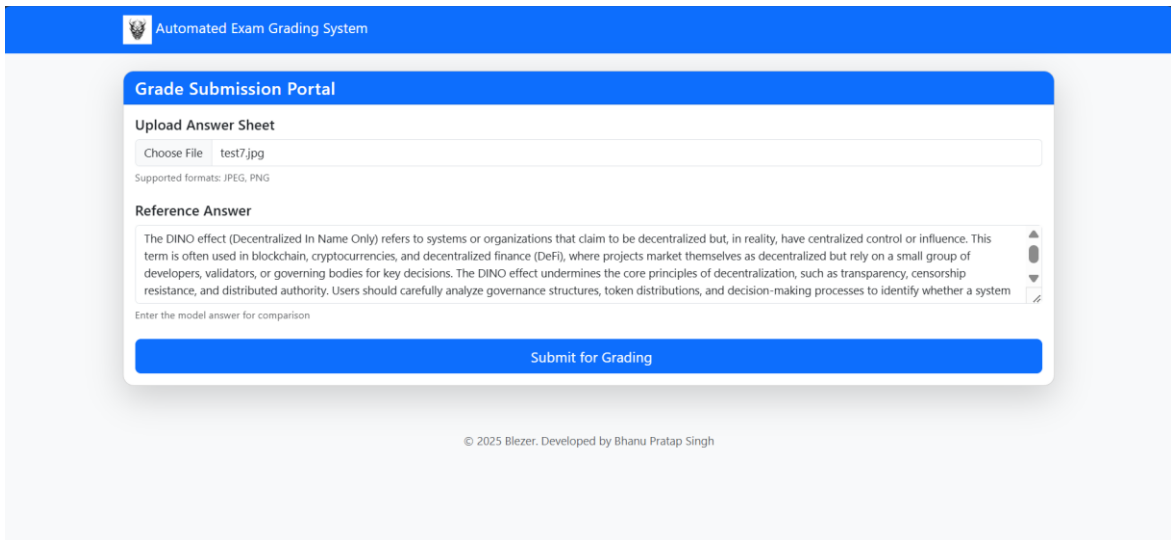
Field Name	Data Type	Constraints	Purpose
Grade_id	INT	PRIMARY KEY, AUTO_INCREMENT	Unique ID for each Grade record
student_id	INT	FOREIGN KEY REFERENCES student(student_id)	Linked student
course_id	INT	FOREIGN KEY REFERENCES course(course_id)	Associated course
marks_obtained	FLOAT	NOT NULL	Marks scored in the course
assignment_score	FLOAT	DEFAULT 0.0	Assignment evaluation score

# **CHAPTER 7**

## **OUTPUT SCREEN**

## CHAPTER – 7

# OUTPUT SCREEN



The screenshot displays the 'Automated Exam Grading System' interface. At the top, a blue header bar contains the system name and a logo. Below this, a central white box with a blue header 'Grade Submission Portal' contains the main functionality. It includes an 'Upload Answer Sheet' section with a file selection button and a text input field showing 'test7.jpg'. Below the upload section is a 'Reference Answer' section with a text area containing a paragraph about the DINO effect. A 'Submit for Grading' button is located at the bottom of the central box. The footer of the page shows the copyright information: '© 2025 Blezer. Developed by Bhanu Pratap Singh'.

Automated Exam Grading System

### Grade Submission Portal

**Upload Answer Sheet**

Choose File test7.jpg

Supported formats: JPEG, PNG

**Reference Answer**

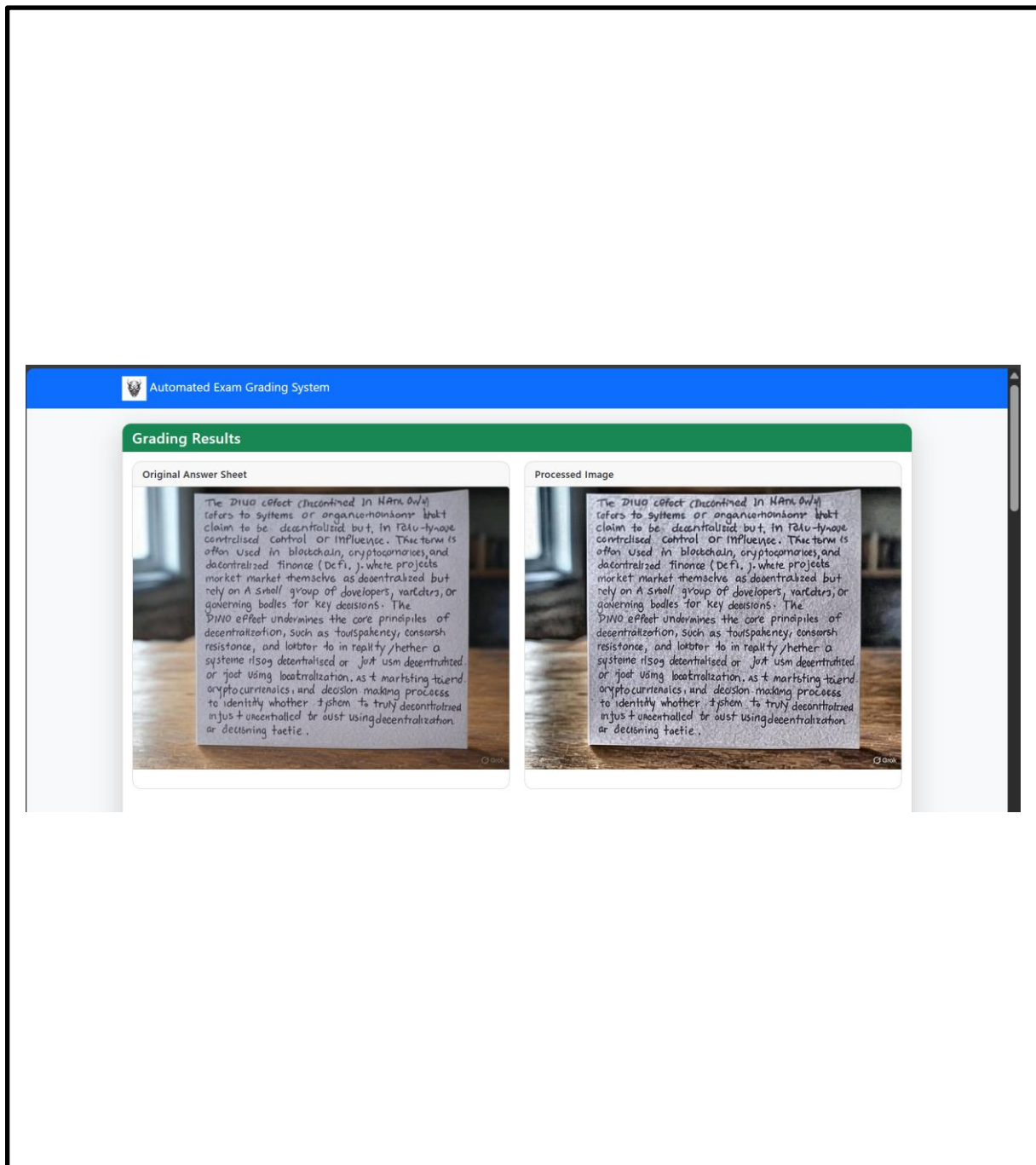
The DINO effect (Decentralized In Name Only) refers to systems or organizations that claim to be decentralized but, in reality, have centralized control or influence. This term is often used in blockchain, cryptocurrencies, and decentralized finance (DeFi), where projects market themselves as decentralized but rely on a small group of developers, validators, or governing bodies for key decisions. The DINO effect undermines the core principles of decentralization, such as transparency, censorship resistance, and distributed authority. Users should carefully analyze governance structures, token distributions, and decision-making processes to identify whether a system

Enter the model answer for comparison

**Submit for Grading**

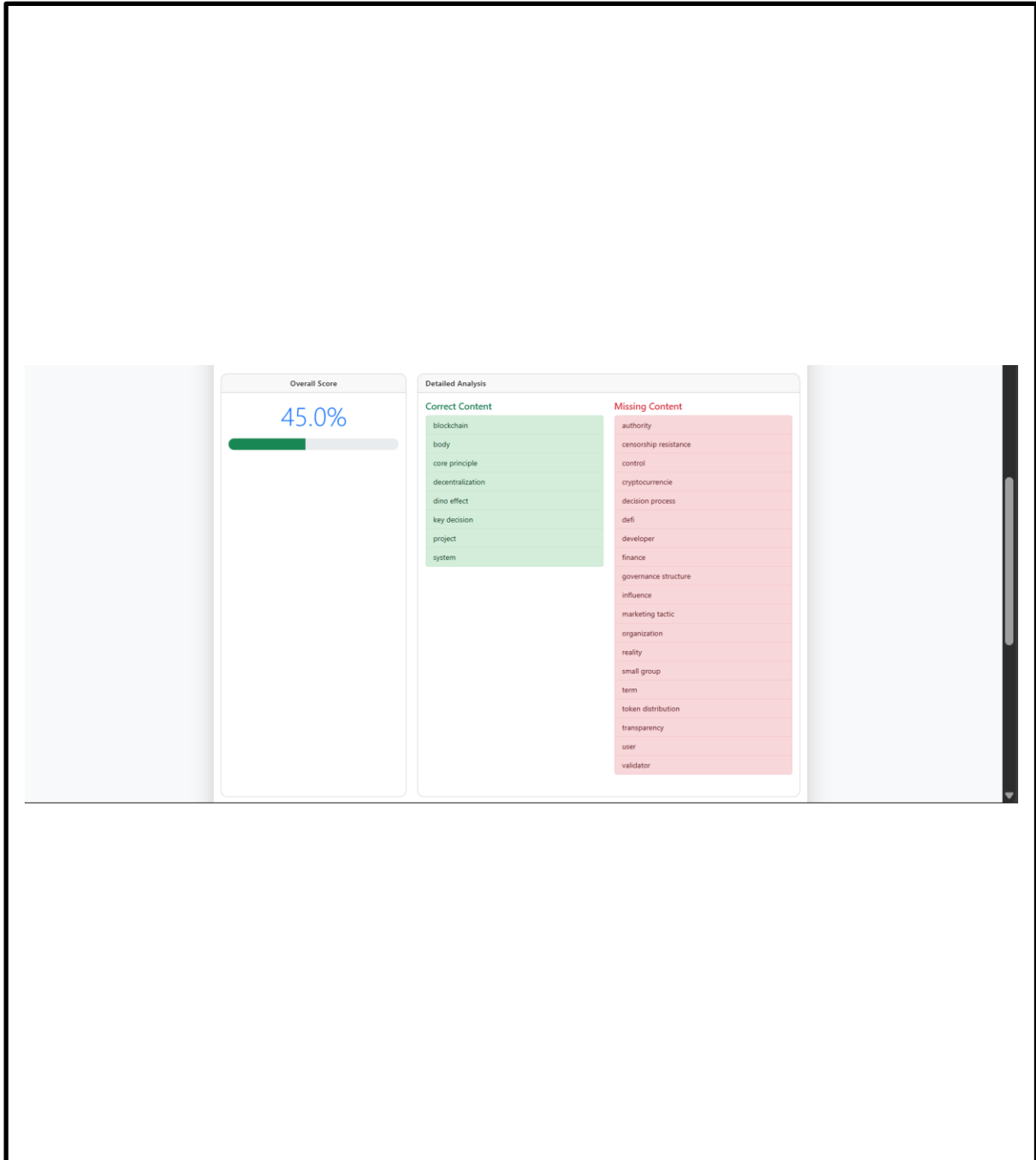
© 2025 Blezer. Developed by Bhanu Pratap Singh

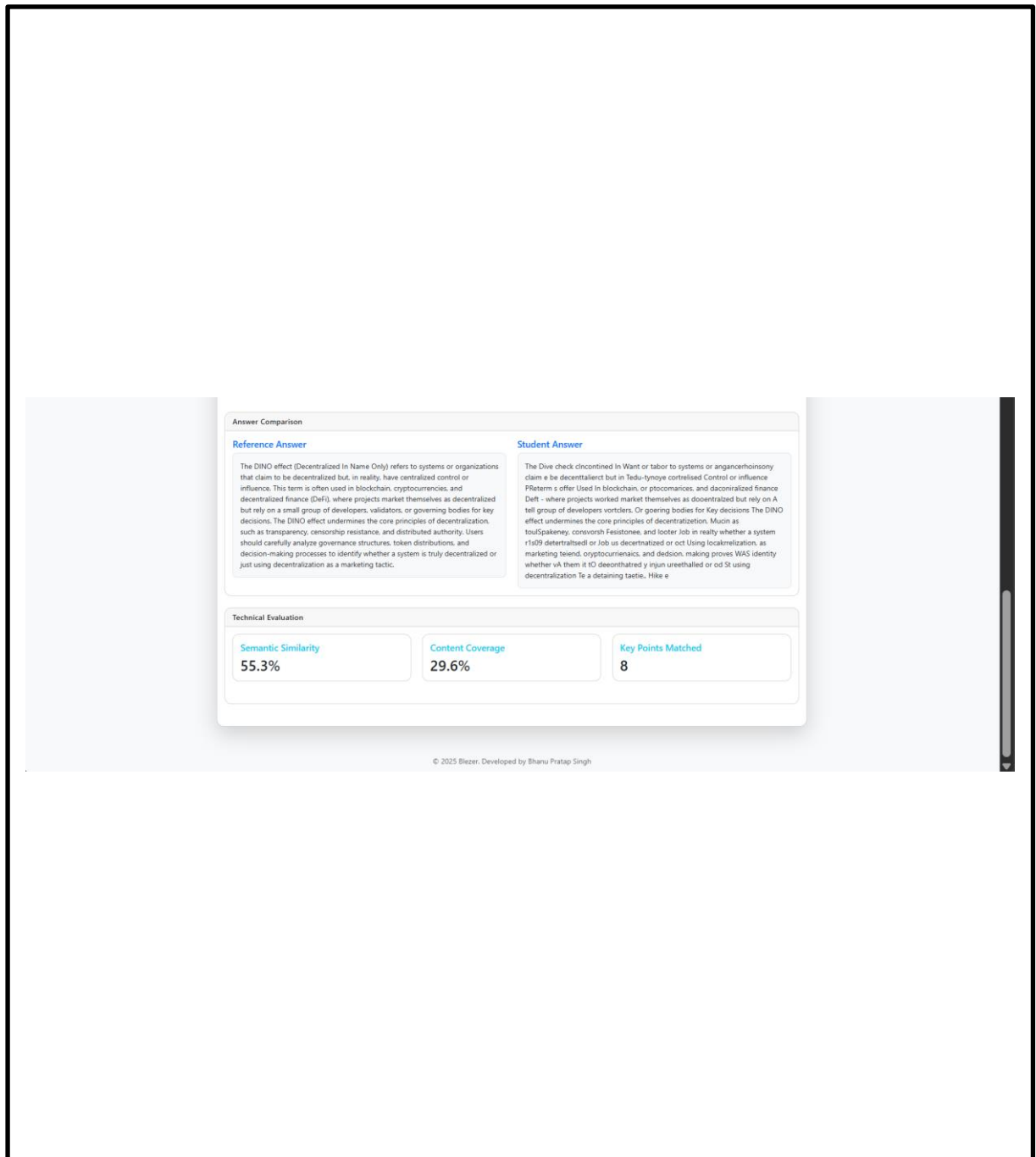
**Figure 7.1: Grade Submission Portal**



**Figure 7.2: Grading Results**



**Figure 7.3: Overall Score**

**Figure 7.4: Answer Comparison**

Student Panel

Dashboard

Upload Data

Export Data

Analytics

Logout

Upload Student Data

Upload CSV File

Choose FileNo file chosen

Upload CSV

Study Hours Per Day

Attendance (%)

Previous Score

Upload

Recent Uploads

Date	Study Hours	Attendance	Previous Score	Predicted Score
2025-04-10	6.0	86.0	77.0	81.62
2025-04-10	8.0	75.0	85.0	80.22
2025-04-10	4.0	80.0	26.0	53.16
2025-04-10	5.0	40.0	60.0	50.22
2025-04-10	8.0	70.0	76.0	73.26


  
jatin@gmail.com

Figure 7.5: Upload Data

# **CHAPTER 8**

# **DEPLOYMENT**

# CHAPTER – 8

## DEPLOYMENT

---

### 8.1 PREREQUISITES

Before proceeding with deployment, the following software and configurations must be available:

#### Software Requirements:

- Python 3.8 or higher
- Flask or Django Web Framework
- MySQL Server version 8.0+ / PostgreSQL
- MySQL Workbench / pgAdmin (for database operations)
- IDE (VS Code / PyCharm) – optional for further development or testing
- Web Browser (Chrome / Firefox)
- Joblib or Pickle for model serialization
- Chart.js or Plotly for frontend data visualization

### 8.2 DATABASE CONFIGURATION

- Install and launch MySQL/PostgreSQL Server and open MySQL Workbench or pgAdmin.
- Create a new schema/database named `student_performance_db`.
- Execute the SQL script (`student_performance_db.sql`) provided with the project to create necessary tables such as `Student`, `Academic_Record`, `Prediction_Result`, `Teacher`, and `Feedback`.
- Update the database connection details in the backend Python code (typically in a config file or within SQLAlchemy setup):
- `SQLALCHEMY_DATABASE_URI=`  
`"mysql+pymysql://root:password@localhost:3306/student_performance_db"`

### 8.3 APPLICATION DEPLOYMENT ON LOCAL SERVER

- Ensure all project dependencies are installed using `pip install -r requirements.txt`.
- Train or load the machine learning model and serialize it using Joblib/Pickle.
- Run the Flask or Django application:
- Once the server starts successfully, open a web browser and enter the URL: `http://localhost:5000` (for Flask) or `http://127.0.0.1:8000` (for Django).

- The application should now be live and accessible. Users can register, log in, enter academic records, view performance predictions, and interact with visual analytics.

## 8.4 DEPLOYMENT TO CLOUD ENVIRONMENT

For wider accessibility and real-time usage, the application can also be hosted on cloud platforms such as:

- AWS EC2 with Ubuntu or Windows, configured with Python and MySQL/PostgreSQL
- Heroku (for Flask/Django with PostgreSQL)
- Google Cloud Platform / Azure App Services
- Upload the source code to the cloud server or platform.
- Install all required dependencies and set environment variables securely.
- Configure and run the backend application server (e.g., using Gunicorn + Nginx for production).
- Host the MySQL/PostgreSQL database either on the same instance or use managed services like AWS RDS.
- Open necessary ports (e.g., 5000/8000 or as per configuration) in the firewall settings.
- Ensure static files and machine learning models are properly loaded and served.

## 8.5 POST-DEPLOYMENT CONSIDERATIONS

- Store all sensitive credentials like database URIs and secret keys in environment variables or .env files.
- Regularly monitor application logs using tools like logging module or third-party services.
- Set up periodic database backups to prevent data loss.
- Implement role-based access control and secure user authentication.
- Update the machine learning model periodically with new data for better accuracy.
- Continuously improve the UI and backend APIs based on user feedback.

## REFERENCES

---

### JOURNALS / RESEARCH PAPERS

1. Kaur, G., & Gupta, V. (2020) 'Predicting Student Academic Performance Using Machine Learning Techniques', *International Journal of Computer Applications*, Vol.176(34), pp.21–26.
2. Mehta, S., & Patel, R. (2021) 'Educational Data Mining for Student Performance Analysis', *Journal of Artificial Intelligence and Soft Computing Research*, Vol.11(1), pp.42–50.
3. Sharma, P., & Das, A. (2022) 'Integration of Data Analytics in Education Sector for Predictive Modeling', *International Journal of Information and Education Technology*, Vol.12(5), pp.382–388.
4. Ramesh, A., & Prasad, S. (2023) 'A Comparative Study on Machine Learning Models for Student Success Prediction', *International Journal of Emerging Technologies in Learning (iJET)*, Vol.18(2), pp.75–82.

### BOOKS

5. Géron, A. (2019) *Hands-On Machine Learning with Scikit-Learn, Keras, and TensorFlow: Concepts, Tools, and Techniques to Build Intelligent Systems*, 2nd Edition, O'Reilly Media.
6. McKinney, W. (2018) *Python for Data Analysis: Data Wrangling with Pandas, NumPy, and IPython*, 2nd Edition, O'Reilly Media.
7. Grinberg, M. (2018) *Flask Web Development: Developing Web Applications with Python*, 2nd Edition, O'Reilly Media.

### WEBSITES (with exact URL up to page)

8. <https://scikit-learn.org>
9. <https://dev.mysql.com/doc>
10. <https://www.chartjs.org>
11. <https://pandas.pydata.org>

## **PROJECT SUMMARY**

### **About Project**

<b>Title of the project</b>	AGSAI
<b>Semester</b>	8
<b>Members</b>	4
<b>Team Leader</b>	Bhanu Pratap Singh
<b>Describe role of every member in the project</b>	Bhanu Pratap Singh– Backend Development, Machine Learning Model Integration. Amit Ahirwar– Frontend Development and User Interface Design. Bhumika Shivhare– Database Design, ER Diagram, and Table Structure Implementation. Manish Kumar Yadav – Documentation, Testing, and Deployment Configuration
<b>What is the motivation for selecting this project?</b>	To provide an intelligent system that predicts student performance using data-driven insights.
<b>Project Type</b> (Desktop Application, Web Application, Mobile App, Web)	Web Application

### **Tools & Technologies**

<b>Programming language used</b>	Python 3.8+
<b>Compiler used (with version)</b>	Python Interpreter (CPython 3.8.10)
<b>IDE used (with version)</b>	Jupyter Notebook (v6.4.12)/ VS Code (v1.85.0)
<b>Front End Technologies (with version, wherever Applicable)</b>	HTML5, CSS3, JavaScript (ES6), Bootstrap 5.3.2, Chart.js v4.4.1 / Plotly v2.27.0
<b>Back End Technologies (with version, wherever applicable)</b>	Flask 2.2.5, Scikit-learn 1.3.2, Pandas 2.2.1, NumPy 1.26.4
<b>Database used (with version)</b>	MySQL 8.0.36



**Software Design& Coding**

<b>Is prototype of the software developed?</b>	Yes
<b>SDLC model followed (Waterfall, Agile, Spiral etc.)</b>	Agile
<b>Why above SDLC model is followed?</b>	Agile provides flexibility, adaptability to changing requirements, and allows early delivery of functional modules.
<b>Justify that the SDLC model mentioned above is followed in the project.</b>	Iterative development through sprints, regular feedback and continuous integration for evolving requirements
<b>Software Design approach followed (Functional or Object Oriented)</b>	Object Oriented
<b>Name the diagrams developed (According to the Design approach followed)</b>	Use Case, ER, Sequence, Class Diagram
<b>In case Object Oriented approach is followed, which of the OOPS principles are covered in design?</b>	Abstraction, Encapsulation, Inheritance, Polymorphism
<b>No. of Tiers (example 3-tier)</b>	3-tier Architecture
<b>Total no. of front-end pages</b>	12
<b>Total no. of tables in database</b>	6
<b>Database in which Normal Form?</b>	3NF
<b>Are the entries in database encrypted?</b>	Yes
<b>Front end validations applied (Yes / No)</b>	Yes
<b>Session management done (in case of web applications)</b>	Yes
<b>Is application browser compatible (in case of web applications)</b>	Yes
<b>Exception handling done (Yes / No)</b>	Yes

<b>Commenting done in code</b> (Yes / No)	Yes
<b>Naming convention followed</b> (Yes / No)	Yes
<b>What difficulties faced during deployment of project?</b>	Ensuring seamless integration between the machine learning model
<b>Total no. of Use-cases</b>	6
<b>Give titles of Use-cases</b>	User Login/Register, Upload Student Academic Data, Predict Student Performance, View Performance Report, Update Student Records, Generate Analytics Report, Admin User Management, Export Prediction Results, Provide Feedback

### **Project Requirements**

<b>MVC architecture followed</b> (Yes / No)	Yes
<b>If yes, write the name of MVC architecture followed</b> (MVC-1, MVC-2)	MVC – 2
<b>Design Pattern used</b> (Yes / No)	Yes
<b>If yes, write the name of Design Pattern used</b>	DAO (Data Access Object)
<b>Interface type</b> (CLI / GUI)	GUI
<b>No. of Actors</b>	2
<b>Name of Actors</b>	Admin, User, Teacher
<b>Total no. of Functional Requirements</b>	12
<b>List few important non-Functional Requirements</b>	Scalability, Usability, Maintainability, Browser Compatibility

### **Testing**

<b>Which testing is performed?</b> (Manual or Automation)	Manual Testing
<b>Is Beta testing done for this project?</b>	Yes

**Write project narrative covering above mentioned points**

“AGSAI” is a web-based intelligent application developed using Python (3.8+) and Flask framework to analyze and predict student academic outcomes based on historical data. It aims to assist educators and institutions in early identification of at-risk students through data-driven insights.

The system follows a modular architecture. Flask handles routing and server-side logic, while the frontend interface is built using HTML5, CSS3, JavaScript (ES6), and Bootstrap 5 for responsive design. Machine learning models, developed using Scikit-learn (1.2), are trained on academic datasets and integrated via RESTful APIs. Data manipulation is performed using Pandas and NumPy libraries.

MySQL 8.0 serves as the backend relational database, storing student profiles, academic scores, predictions, and admin data. SQLAlchemy is used as the ORM for secure and efficient database interaction. The application includes visual analytics with Chart.js and Plotly for intuitive performance monitoring.

Deployment involves setting up a virtual environment, model serialization using Joblib, and hosting via WSGI on Apache or cloud services. Once deployed, the system offers secure login, student data entry, prediction output, and admin functionalities, providing a robust platform to enhance academic decision-making.

Bhanu Pratap Singh	0187CS211049
Amit Ahirwar	0187CS211024
Bhumika Shivhare	0187CS211051
Ashish Kumar Soni	0187CS211040

Guide Signature  
(Prof. Nargish Gupta)

## **APPENDIX-1**

## **GLOSSARY OF TERMS**

---

(In alphabetical order)

### **B**

**BOOTSTRAP** A front-end CSS framework used for building responsive and mobile-first web applications.

### **C**

**CSS** Cascading Style Sheets – A language used for styling HTML content.

### **D**

**DAO** Data Access Object – A design pattern that provides an abstract interface to the database operations.

### **H**

**HTML** HyperText Markup Language – The standard markup language used to create the structure of web pages.

### **J**

**JAVASCRIPT** A scripting language used to create interactive effects within web browsers.

**JDBC** Java Database Connectivity – A Java API that enables communication with databases through SQL.

**JSP** JavaServer Pages – A technology used to create dynamic web content on the server side using Java embedded in HTML.