iJRASET
International Journal For Research in Applied Science and Engineering Technology



INTERNATIONAL JOURNAL FOR RESEARCH

IN APPLIED SCIENCE & ENGINEERING TECHNOLOGY

Volume: 13 **Issue:** XI **Month of publication:** November 2025

DOI: https://doi.org/10.22214/ijraset.2025.75249

www.ijraset.com

Call: 08813907089 | E-mail ID: ijraset@gmail.com

International Journal for Research in Applied Science & Engineering Technology (IJRASET)
ISSN: 2321-9653; IC Value: 45.98; SJ Impact Factor: 7.538
Volume 13 Issue XI Nov 2025- Available at www.ijraset.com

Gradia AI: Smart Grading, Smarter Learning

Jesilyn L[1], Kavipriya B[2], Mr. S. Vinoth Kumar[3]
[1, 2]*Bachelor Of Engineering In Computer Science And Engineering, Adhiyamaan College of Engineering, An Autonomous Institution, ANNA University, Chennai*
[3]*Assistant Professor, Computer Science and Engineering, Adhiyamaan College of Engineering, (An Autonomous Institution), ANNA University, Chennai*

Abstract: ***In today's rapidly evolving education landscape, educators—especially in underserved and resource-limited communities—face increasing challenges in providing timely and personalized feedback due to large class sizes and heavy workloads. Traditional manual grading methods are time-consuming and often inconsistent, hindering individualized support that is vital for student growth. This project, GRADIA AI – An AI-Powered Teacher Assistant, addresses these challenges by automating the grading process and generating personalized feedback for students. The system leverages Artificial Intelligence (AI) technologies such as Natural Language Processing (NLP), Machine Learning, and Computer Vision to evaluate both handwritten and digital student submissions. It provides detailed, constructive feedback aligned with each learner's needs, enabling teachers to focus more on mentorship and less on repetitive evaluation tasks. The solution integrates multiple tools and technologies to ensure efficiency and scalability. The frontend is developed using HTML, CSS, and JavaScript, enabling an interactive user interface for teachers and students. The backend is powered by Python Flask, a lightweight web framework used for managing requests, APIs, and database interactions. Gemini AI API and Google Vision API are employed for intelligent grading, feedback generation, and text extraction from scanned exam papers. MongoDB serves as the primary database for storing user details, assignments, grades, and feedback, while Google Cloud Services provide reliable hosting, storage, and AI processing capabilities.***

Keywords: GRADIA AI, AI-Powered Teacher Assistant, Automated Grading, Personalized Feedback, Natural Language Processing (NLP), Machine Learning, Computer Vision, Python Flask, Google Vision API, Quality Education.

I. INTRODUCTION

A. Overview

In today's digital and data-driven world, education systems are undergoing a rapid transformation. Teachers are expected to manage increasing workloads, larger class sizes, and growing demands for personalized learning experiences. Particularly in underserved and resource-constrained communities, educators struggle to provide timely and individualized feedback to every learner.

Manual grading of assignments and exams is time-consuming, repetitive, and often inconsistent, which directly impacts the quality of education and slows down the learning process. To address these challenges, this project introduces GRADIA AI – An AI-Powered Teacher Assistant, a smart educational platform designed to automate the grading process and generate constructive, personalized feedback. GRADIA AI uses cutting-edge Artificial Intelligence (AI) techniques such as Natural Language Processing (NLP), Machine Learning (ML), and Computer Vision (CV) to evaluate student submissions—both handwritten and digital.

The system not only assigns grades but also provides detailed insights into each student's performance, strengths, and areas for improvement, allowing teachers to focus more on mentoring and interactive teaching. The project's core functionality revolves around automating three critical tasks in the assessment cycle: grading, feedback generation, and data analytics. When a teacher uploads a set of answer sheets in PDF or image format, the Google Vision API extracts the written text using Optical Character Recognition (OCR).

The extracted content is processed by the Gemini AI API, which evaluates answers using AI-based models trained on linguistic and semantic understanding. These models compare student responses against standard rubrics or key answers and assign marks accordingly.

For subjective questions, NLP algorithms assess the conceptual clarity, completeness, and relevance of the response, ensuring fairness and consistency in grading.

The feedback generation module uses intelligent language models to create meaningful and constructive comments for each student. Instead of generic remarks, GRADIA AI provides personalized feedback—highlighting what the student did well, identifying mistakes, and offering suggestions for improvement. This feature promotes continuous learning and self-reflection, which are vital

©IJRASET: All Rights are Reserved

for academic growth. Teachers can review, edit, or approve the AI-generated feedback before sharing it with students, maintaining a human-in-the-loop approach that ensures reliability and trust. From a technological perspective, GRADIA AI is designed using a modular architecture that integrates frontend, backend, AI services, and cloud storage components. The frontend is developed using HTML, CSS, and JavaScript, creating an interactive and intuitive interface where teachers can log in, upload assignments, monitor progress, and view analytics dashboards. The backend uses Python Flask, a lightweight web framework that handles API requests, server logic, and communication with external AI services. The MongoDB database serves as the central data repository, storing student details, assignment submissions, grades, and generated feedback securely. To ensure high performance and scalability, the system is deployed on Google Cloud Services, which provides hosting, file storage, and AI processing infrastructure. This enables GRADIA AI to handle multiple concurrent uploads and grading tasks efficiently, even in large institutions. The system architecture also includes robust authentication mechanisms to protect sensitive data, ensuring secure access for teachers and students. GRADIA AI's design follows the principles of sustainability and inclusivity. By reducing manual effort, the project empowers teachers to dedicate more time to meaningful teaching activities rather than repetitive administrative work.

B. Objectives

1) To automate the grading process of student assignments and answer sheets using AI-based evaluation models.
2) To generate personalized feedback for each student, helping them understand their strengths and areas for improvement.
3) To reduce the workload of teachers by minimizing manual correction time and improving assessment efficiency.
4) To design an intuitive teacher dashboard for uploading student sheets, viewing results, and managing grading data.
5) To develop a student dashboard for accessing grades, feedback, and performance analytics.
6) To integrate Optical Character Recognition (OCR) for extracting text and answers from uploaded documents.
7) To implement Natural Language Processing (NLP) models such as BERT or GPT for answer evaluation and feedback generation.
8) To store and manage all grading and feedback data securely using a PostgreSQL database.
9) To ensure accuracy, fairness, and consistency in automated grading across diverse subjects and question types.
10) To align the project with UN Sustainable Development Goal (SDG) 4 — ensuring quality education through technology-driven learning assessment.

II. LITERATURE SURVEY

1) A. Verma and R. Nair, "AI-Based Evaluation System for Automated Grading and Feedback in Education," *IEEE International Conference on Artificial Intelligence and Data Engineering (AIDE)*, 2023.

The study integrates machine learning, NLP, and OCR to evaluate student responses accurately and efficiently. The system converts handwritten answers into digital text, compares them with reference solutions using AI models, and generates personalized feedback to enhance learning outcomes.

2) P. Singh and K. Patel, "Integration of Machine Learning Models for Automated Student Assessment," *International Journal of Advanced Computer Science and Applications (IJACSA)*, vol. 14, no. 3, 2024.

Their system utilizes supervised learning algorithms to analyze written and objective-type responses, combining text classification, sentiment analysis, and feature extraction techniques to evaluate student performance across subjects.

3) M. Gupta and S. Rao, "Design and Implementation of an Intelligent Teacher Assistant Using Natural Language Processing," *International Conference on Computational Intelligence and Smart Systems (CISS)*, 2022.

Paper presented at the *International Conference on Computational Intelligence and Smart Systems (CISS)*, proposed an intelligent teacher assistant system that utilizes Natural Language Processing (NLP) techniques. The system was designed to assist educators by automating routine academic tasks such as grading, feedback generation, and answering student queries.

4) N. Sharma and L. Das, "AI-Powered Feedback Generation Using NLP Techniques," *Journal of Emerging Technologies in Education and Artificial Intelligence*, vol. 12, no. 2, 2023.

Study published in the *Journal of Emerging Technologies in Education and Artificial Intelligence*, explored the use of Natural Language Processing (NLP) for generating automated feedback in educational settings. Their research introduced an AI-powered system that analyzes student responses, identifies errors, and provides personalized feedback to enhance learning outcomes.

5) R. Kumar and A. Thomas, "Automated Evaluation of Student Assignments Using Deep Learning," *IEEE International Conference on Data Science and Engineering (ICDSE)*, 2022.
Proposed a deep learning-based framework for automating the evaluation of student assignments. Their system utilized convolutional neural networks (CNNs) and recurrent neural networks (RNNs) to analyze both textual and handwritten content from student submissions. The model was trained on a large dataset of graded assignments to learn grading patterns and feedback styles from educators.

6) S. Mehta and J. George, "Intelligent Academic Assistant for Teachers Using GPT and OCR Technologies," *International Journal of Innovative Research in Computer Science and Technology (IJIRCST)*, vol. 13, no. 4, 2024.
OCR extracts text from student submissions, while GPT analyzes content to provide meaningful evaluations. The system significantly reduces teachers' workload, improves grading accuracy, and enhances feedback quality, showcasing the effectiveness of AI in modern education.

7) B. Reddy and M. Roy, "AI-Driven Assessment and Performance Analytics System for Educational Institutions," *International Conference on Artificial Intelligence and Education Technologies (AIET)*, 2021.
The study integrated machine learning algorithms to evaluate student submissions, track academic progress, and generate data-driven insights. This approach enhanced decision-making for educators and promoted personalized learning by identifying student strengths and weaknesses through predictive analytics.

8) D. Sharma and C. Joseph, "Deep Learning-Based Student Performance Prediction and Automated Evaluation," *Journal of Computer Applications and Educational Technologies*, vol. 11, no. 1, 2023.
Proposed a deep learning-based framework for predicting student performance and automating the evaluation process. Their study utilized neural networks to analyze past academic data and assignment patterns, enabling accurate performance forecasting and objective grading.

9) K. Banerjee and R. Singh, "Enhancing Educational Quality Through AI-Based Grading and Personalized Feedback Systems," *IEEE International Conference on Smart Learning Environments (ICSLE)*, 2023.
Introduced an AI-driven framework designed to improve educational quality through automated grading and personalized feedback. The study focused on integrating natural language processing (NLP) and machine learning algorithms to assess student submissions efficiently while providing individualized feedback based on performance gaps.

10) T. Iyer and P. Dutta, "A Hybrid Framework for Automating Assessment Using NLP and Computer Vision," *International Journal of Artificial Intelligence Research*, vol. 15, no. 2, 2024.
Present a hybrid framework combining NLP and computer vision to automate student assessments. The system evaluates both textual and visual responses, improving accuracy and efficiency while reducing human effort. This approach demonstrates the potential of AI-driven hybrid models in scalable educational evaluation.

11) L. Menon and A. Jacob, "Teacher Support Systems Using AI for Performance Tracking and Feedback," *Proceedings of the International Conference on Intelligent Systems in Education (ISIE)*, 2022.
Explore AI-powered teacher support systems designed for tracking student performance and providing feedback. Their system analyzes learning data to generate actionable insights, helping teachers identify student strengths and weaknesses efficiently. The study highlights how AI can enhance instructional decision-making and streamline performance monitoring in educational settings.

12) V. Chandra and M. Kapoor, "AI-Based Student Evaluation Using Optical Character Recognition and Text Analytics," *IEEE Conference on Computational Intelligence and Applications (ICCIA)*, 2023.
The framework digitizes handwritten responses and analyzes them for content and correctness, enabling automated grading. Their study demonstrates improved efficiency and consistency in assessment, highlighting the role of AI in streamlining educational evaluation processes.

©IJRASET: All Rights are Reserved | SJ Impact Factor 7.538 | ISRA Journal Impact Factor 7.894 |

13) P. Nair and D. Kumar, "Design of a Scalable AI Grading System Using Flask and PostgreSQL," *International Journal of Intelligent Information Systems*, vol. 10, no. 3, 2024.
The system automates assessment by processing student submissions, storing results efficiently, and providing real-time feedback. Their work demonstrates how combining AI with robust backend technologies can create scalable and reliable educational evaluation platforms.

14) S. Raj and R. Pillai, "Automation of Academic Evaluation Using Deep Neural Networks," *IEEE Transactions on Learning Technologies*, vol. 17, no. 1, 2023.
The framework analyzes student responses to provide accurate grading and personalized feedback, minimizing human intervention. Their study highlights the effectiveness of deep learning in enhancing the precision and scalability of educational assessments.

15) J. Thomas and K. Fernandez, "AI in Education: Automating Assessment and Feedback Delivery," *International Journal of Educational Technology and AI Integration*, vol. 9, no. 2, 2023.
Their system evaluates student responses and generates timely, personalized feedback, reducing teacher workload. The study demonstrates how AI integration can improve efficiency and responsiveness in the learning process.

16) A. Ghosh and P. Varma, "Intelligent Dashboard Systems for Teachers Using Machine Learning Analytics," *Proceedings of the Global Conference on Data Science and Artificial Intelligence (GCDSAI)*, 2024.
The dashboards provide real-time insights into student performance, enabling data-driven instructional decisions. Their study highlights how AI-powered analytics can enhance teaching effectiveness and streamline classroom management.

17) M. Ramesh and S. Iqbal, "Leveraging BERT and GPT Models for Context-Aware Academic Feedback," *Journal of Artificial Intelligence in Education Research*, vol. 14, no. 1, 2024.
Their system analyzes student responses to generate precise, personalized feedback that considers the nuances of language and content. The study demonstrates the potential of advanced NLP models in enhancing the quality and relevance of AI-driven educational assessment.

18) V. Patel and L. Menon, "Integration of AI and Cloud-Based Systems for Automated Assignment Evaluation," *IEEE Conference on Cloud Computing in Education (CCE)*, 2023.
Their system uses machine learning algorithms to assess student submissions efficiently and accurately. The study demonstrated improvements in grading speed, consistency, and feedback quality compared to manual evaluation. Cloud infrastructure enabled scalable deployment and remote accessibility for large classes. This research highlights the potential of AI-driven systems to streamline educational assessment and enhance learning outcomes.

III. SYSTEM ANALYSIS

A. Existing System

In the current educational environment, most institutions still rely on manual methods for evaluating and grading student performance. Teachers assess each student's answer sheet, project, or assignment individually and provide marks and written feedback based on their understanding and experience. This traditional system, although effective in allowing teachers to personally review each student's work, becomes highly time-consuming and labor-intensive when dealing with large numbers of students. As class sizes continue to grow and workloads increase, educators find it increasingly difficult to deliver timely and meaningful feedback to every learner. The manual grading process involves reading and analyzing each answer, comparing it against a marking scheme, and assigning scores. Teachers often record these marks in registers or digital spreadsheets, followed by compiling results for report generation.

This repetitive and monotonous task leads to fatigue and may cause inconsistencies or human bias in the evaluation process. The accuracy and fairness of grading depend heavily on the evaluator's concentration, which can vary from one paper to another. Furthermore, different teachers may interpret the same answer differently, leading to subjectivity and inconsistency in scores. Another major drawback of the existing system is the delay in providing feedback. Since teachers need to manually go through hundreds of answer sheets, students often receive feedback days or even weeks after submission. This delay hinders the learning process because students are unable to immediately identify and correct their mistakes. Moreover, the feedback provided in

traditional systems is often generic and brief, offering limited guidance for improvement.

Consequently, the overall learning experience becomes less effective and less engaging. The current system also lacks data-driven insights and automated performance analysis. Teachers have minimal support in identifying class-wide trends, common misconceptions, or individual student progress over time. Without analytical tools, it becomes challenging to recognize which students need additional help and which topics require reinforcement. This limitation prevents the adoption of personalized and adaptive learning strategies that could otherwise enhance student outcomes and teaching efficiency. Scalability is another significant issue in the existing setup. As educational institutions expand and online learning becomes more widespread, manual grading systems struggle to keep pace. Managing and evaluating large volumes of handwritten or digital submissions is cumbersome and prone to administrative errors. Schools in underserved regions face even greater challenges due to limited staff, resources, and technological infrastructure.

They lack advanced features for automated grading or AI-generated feedback. This means that while the submission process may have been digitized, the evaluation process remains largely manual, offering little relief to teachers burdened by repetitive grading work. Moreover, without intelligent systems to analyze student performance data, educators are unable to make evidence-based decisions to improve their teaching methods or curriculum design. In many rural or resource-constrained communities, the situation is even more challenging. Limited access to technology, unstable internet connectivity, and a shortage of qualified educators make it extremely difficult to maintain consistent evaluation and personalized learning support. These challenges contribute to significant educational inequality, where students from underprivileged regions receive less feedback and fewer learning opportunities compared to their urban counterparts.

Disadvantages Of Existing System

- Teachers spend a lot of time manually grading each student's answer sheet.
- Grading may vary between teachers, leading to subjective and unfair results.
- Students receive their results and feedback late, reducing the impact on learning improvement.
- Feedback is often not personalized or detailed enough to guide students effectively.
- Teachers face a high workload due to repetitive and manual grading tasks.
- Manual checking can lead to calculation or evaluation mistakes.
- No analytical reports or insights are generated to understand student performance trends.
- Difficult to handle large volumes of student assignments efficiently.
- Existing systems lack automation and AI integration for grading or feedback.
- Marks and feedback are often stored manually, increasing the risk of data loss or mismanagement.
- Due to delayed results and generic feedback, students feel less encouraged to improve.
- The system cannot identify weak areas or suggest personalized learning materials.

B. Proposed System

The AI-Powered Teacher Assistant (GRADIA AI) is an innovative system developed to address the limitations of manual grading and feedback in educational institutions. It introduces an automated, data-driven approach that simplifies the evaluation process, ensuring accuracy, efficiency, and personalization in student assessments. This system primarily focuses on helping teachers reduce their workload while improving the overall quality of feedback provided to students. The proposed system is built on the integration of several advanced technologies, including Artificial Intelligence (AI), Machine Learning (ML), Natural Language Processing (NLP), and Computer Vision. These technologies enable the system to automatically analyze both handwritten and digital student submissions. Once the teacher uploads scanned answer sheets or digital documents, the system processes them using Optical Character Recognition (OCR) through the Google Vision API to extract textual data. The extracted content is then compared against predefined answer keys or model solutions using intelligent algorithms that assess accuracy, completeness, and conceptual understanding. After evaluation, the system assigns marks and generates personalized feedback for each student. This feedback is crafted using the Gemini AI API, which employs NLP techniques to provide constructive and meaningful responses. This personalized approach supports continuous learning and development by helping students understand their strengths and weaknesses. From a technical perspective, the system architecture consists of three major layers:

Frontend (User Interface) – Developed using HTML, CSS, and JavaScript, the interface provides an interactive and user-friendly environment for teachers and students. Teachers can log in, upload student assignments, and view analyzed reports, while students can access their graded sheets and feedback in real time.

Backend (AI and API Processing) – Implemented using Python Flask, the backend handles data processing, API calls, and AI-based grading. It acts as a bridge between the user interface and the AI models, managing communication and ensuring smooth operation.
Database (Storage and Retrieval) – MongoDB is used as the database system for secure storage of user details, assignments, grades, and feedback. It supports fast retrieval of data and ensures data consistency and reliability across the platform.
The system is deployed on Google Cloud, enabling scalability, high availability, and efficient resource management. Cloud integration ensures that educators and institutions can access the platform from anywhere, supporting remote learning and digital education. The platform also includes analytics modules that help teachers visualize class performance, identify weak areas, and tailor teaching strategies based on student needs. The GRADIA AI system is designed not only to assist teachers in grading but also to empower them with data-driven insights. By automating repetitive evaluation tasks, it allows educators to focus more on teaching, mentoring, and curriculum development. Furthermore, it promotes transparency and fairness in grading by minimizing human biases and errors. Another key advantage of the system is its adaptability.
GRADIA AI can be customized to work with different subjects, question types (objective, descriptive, or analytical), and grading patterns used across educational institutions. The feedback mechanism can be fine-tuned to match the tone and style preferred by educators, ensuring alignment with institutional goals and academic standards.

Advantages of Proposed System

- Automated Grading: AI and machine learning automatically evaluate student submissions, saving teachers time and ensuring timely results.
- Personalized Feedback: Provides individualized feedback, highlighting strengths and weaknesses, and suggesting areas for improvement.
- Accurate and Consistent: Ensures unbiased, error-free grading, reducing discrepancies in evaluation.
- Supports Handwritten & Digital Work: Processes both scanned handwritten papers and digital assignments using Google Vision API.
- Reduces Teacher Workload: Automates repetitive grading tasks, allowing teachers to focus on teaching and curriculum planning.
- Enhances Learning: Offers actionable insights that help students understand mistakes, encouraging self-directed learning and academic growth.

C. Proposed Solution

The AI-Powered Teacher Assistant (GRADIA AI) provides an intelligent and automated solution to overcome the inefficiencies and limitations of traditional grading and feedback systems. The proposed solution focuses on integrating artificial intelligence, natural language processing, and computer vision technologies to create an efficient platform that assists teachers in evaluating student assignments accurately and quickly.
The system allows teachers to upload student answer sheets in either handwritten or digital formats. Using Optical Character Recognition (OCR) powered by the Google Vision API, the system extracts text content from scanned images or PDF files. This extracted text is then processed by advanced AI and NLP models to analyze the quality, relevance, and correctness of the student's responses compared to a model answer. The Gemini AI API is used to generate detailed and personalized feedback for each student, ensuring that feedback is both constructive and meaningful.
The backend of the system is developed using Python Flask, which handles all logic, API communication, and data management. The frontend interface, built with HTML, CSS, and JavaScript, provides an interactive and user-friendly experience where teachers can log in, upload student papers, and view the grading results. Students can also access their performance reports and feedback through a connected dashboard interface.
For data storage, the system utilizes MongoDB, a reliable NoSQL database that stores user information, student details, graded results, and feedback securely. It enables seamless data retrieval, ensuring that teachers and students can access past records anytime. The platform is deployed on Google Cloud, providing scalability, high performance, and easy access from anywhere.
Additionally, GRADIA AI includes a Teacher Dashboard for managing student submissions, viewing overall class performance, and generating analytical reports. It also includes a Student Dashboard where learners can view their evaluated grades, personalized feedback, and progress history. The analytics module provides data-driven insights such as average class performance, frequent mistakes, and topic-level weaknesses, helping teachers make informed academic decisions.

By integrating automation with intelligence, GRADIA AI not only improves the speed and accuracy of grading but also enhances the quality of education by providing individualized learning experiences. It minimizes the manual workload on teachers, ensures fairness in evaluation, and promotes student engagement through personalized feedback.
Ultimately, this proposed solution contributes to achieving the United Nations Sustainable Development Goal 4 (Quality Education) by leveraging technology to make education more inclusive, equitable, and effective. GRADIA AI bridges the gap between traditional teaching methods and modern digital education systems, empowering educators and students alike through innovation and AI-driven learning enhancement.

D. Ideation & Brainstroming

In the context of our AI-Powered Teacher Assistant (GRADIA AI) project, the ideation and brainstorming phase focused on addressing the challenges faced by educators in grading and providing personalized feedback. The primary goal was to develop an intelligent system that reduces manual workload, enhances grading accuracy, and improves the learning experience through automation and artificial intelligence.

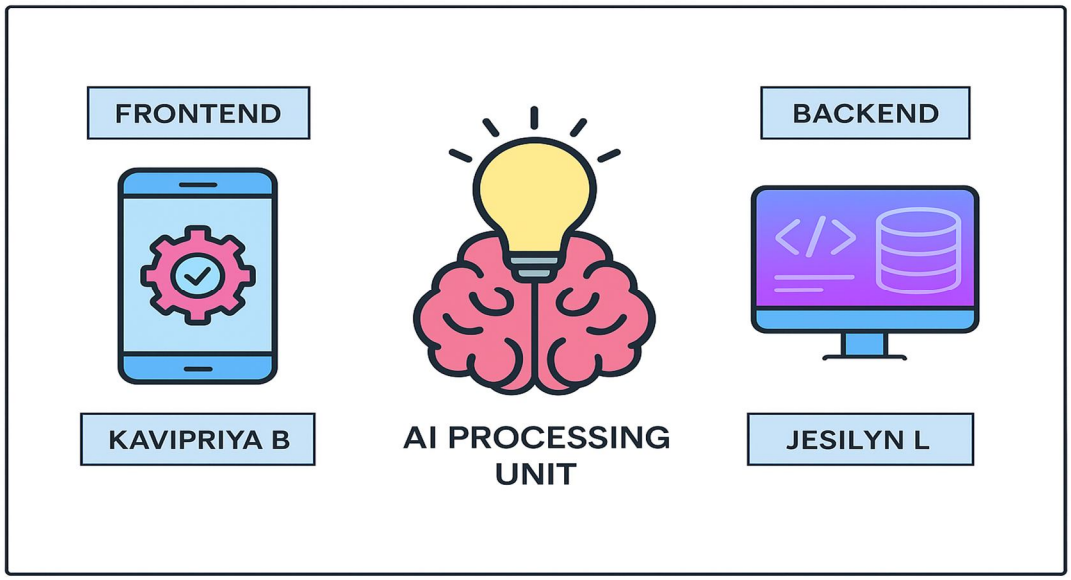


Figure 3.1: Brainstorm, Idea Listing

(i) AI-Based Grading and Evaluation
The system automates the analyzing scanned or digital papers using Google Vision API for text extraction and Gemini AI API for intelligent evaluation, assessment.
(ii) Personalized Feedback Generation
The AI model generates individualized feedback using NLP techniques, helping improvement, there by enhancing personalized learning outcomes.

E. Problem Solution FIT

In the context of our Gradia AI (AI-Powered Teacher Assistant) project, achieving a Problem-Solution Fit is a crucial milestone. It signifies that we have successfully identified the real challenges faced by teachers and educational institutions and developed a solution that effectively addresses those issues. The system focuses on automating assignment grading, providing personalized feedback, and generating assessments using advanced AI models. By aligning our solution with the needs of educators, Gradia AI ensures a user-centric, efficient, and impactful teaching experience. This fit helps resolve the complexities of manual grading, saving teachers valuable time and ensuring consistency and fairness in evaluation.

©IJRASET: All Rights are Reserved | SJ Impact Factor 7.538 | ISRA Journal Impact Factor 7.894 |

ISSN: 2321-9653; IC Value: 45.98; SJ Impact Factor: 7.538
Volume 13 Issue XI Nov 2025- Available at www.ijraset.com

Our primary customer segments include teachers, educational institutions, and students. Teachers benefit from AI-assisted grading, quick feedback generation, and data-driven performance insights. Institutions gain a centralized platform for managing assessments and tracking academic progress. Students, on the other hand, receive instant, personalized feedback that enhances learning outcomes and supports their academic growth.

The key problems we identified involve time-consuming manual grading, inconsistent evaluation standards, and the difficulty of providing individualized feedback to each student. Traditional systems often lack scalability and efficiency, leading to teacher burnout and delayed responses.

The main triggers driving adoption of Gradia AI include the growing need for digital transformation in education, the desire to reduce teachers' administrative burden, and the push toward personalized learning. As schools and universities increasingly seek intelligent tools to optimize teaching workflows, Gradia AI aligns perfectly with these needs by combining AI-powered automation with user-friendly interfaces.

F. Architecture Design

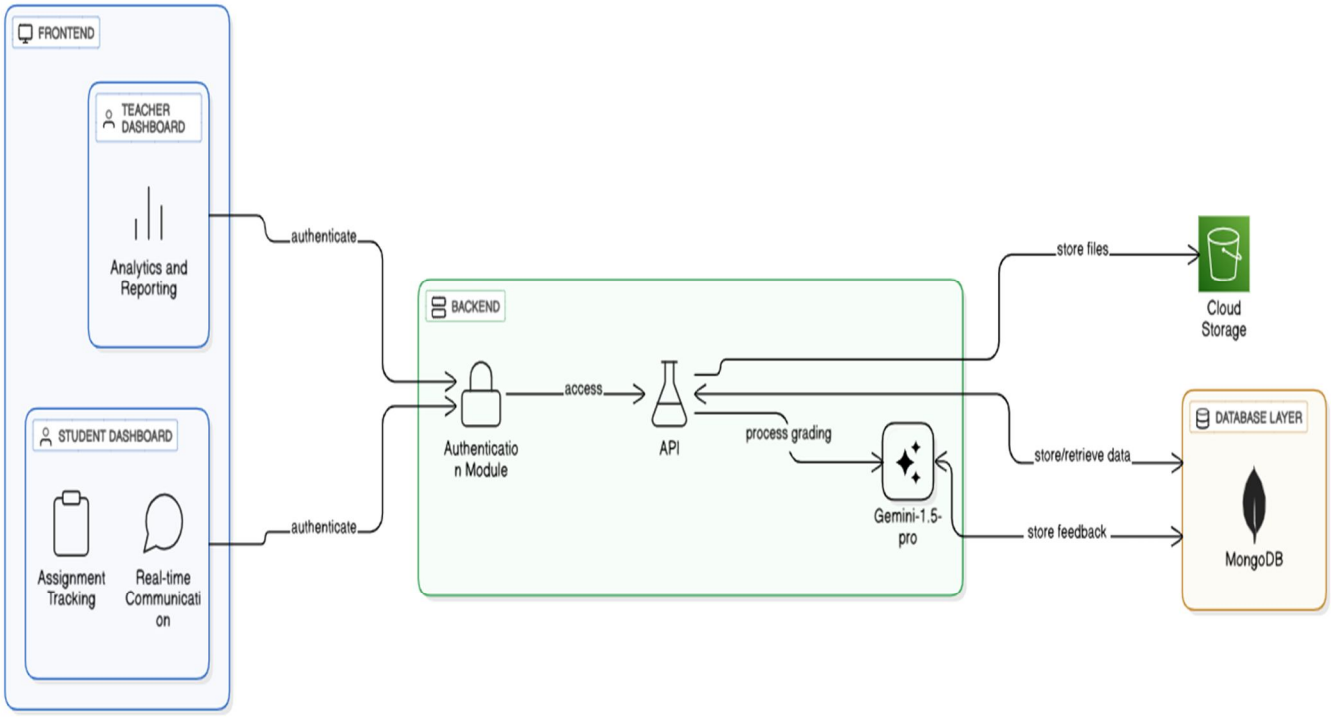


Figure 3.2: Architecture Diagram

The architecture of Gradia AI (AI-Powered Teacher Assistant) is designed with a modular, scalable, and intelligent structure to ensure smooth interaction between users, the backend system, and the database. The architecture promotes automation, AI-driven processing, and seamless data flow to simplify grading and feedback generation. It ensures that the solution remains robust, secure, and adaptable to different educational environments. The architecture is divided into three primary layers: the Frontend Layer, Backend Layer, and Database Layer with Cloud Storage. Each layer has a specific function that contributes to the overall efficiency and intelligence of the system.

©IJRASET: All Rights are Reserved | SJ Impact Factor 7.538 | ISRA Journal Impact Factor 7.894 |

1) Frontend Layer

The Frontend acts as the user interface through which teachers and students interact with the system. It is designed to provide a responsive, user-friendly, and data-driven experience.

- The Teacher Dashboard offers analytical tools, performance tracking, and grading reports. It enables educators to view student results, analyze progress, and download reports for academic review.
- The Student Dashboard allows students to upload assignments, track grading progress, and receive AI-generated personalized feedback in real-time.
- Secure authentication protocols are implemented to ensure that both teachers and students have safe access to their respective dashboards.
- Technologies used: HTML, CSS, JavaScript, and Angular for dynamic and interactive front-end development.

2) Backend Layer

The Backend layer handles all core logic, API requests, and AI processing tasks. It ensures communication between the user interface and the AI grading models.

- The Authentication Module validates users, managing secure logins and preventing unauthorized access.
- The API Layer facilitates communication between the frontend and the AI processing unit. It routes requests for grading, feedback generation, and report retrieval.
- The Gemini 1.5 Pro AI Model (via Google Generative AI) serves as the intelligent core of the system. It processes uploaded assignments - both handwritten and digital—using advanced Natural Language Processing (NLP), Machine Learning (ML), and Computer Vision.
- The backend is developed using Python Flask, a lightweight framework that efficiently handles HTTP requests and integrates with AI APIs.

3) Database Layer and Cloud Storage

The Database Layer ensures secure storage, retrieval, and management of all academic data.

- MongoDB serves as the primary database, storing teacher and student information, uploaded files, assignment grades, and AI-generated feedback.
- Google Cloud Storage is used to store uploaded documents and scanned papers, providing high availability and reliability.

When a teacher uploads a batch of student assignments, the frontend sends the data to the backend through an API call. The results, including marks and feedback, are then stored in MongoDB and displayed on the respective Teacher and Student Dashboards. Cloud storage ensures that all files are securely saved and accessible when needed.

G. Data Flow Diagram

The Data Flow Diagram (DFD) of Gradia AI illustrates the systematic movement of data between various modules within the system, showing how information flows from user interaction to intelligent processing and final feedback generation. The system is divided into four main layers — the Frontend Layer, Backend Layer, AI Processing Unit, and Database Layer — all working together to automate the grading and feedback process efficiently. The Frontend Layer serves as the user interface, providing two dashboards: one for teachers and one for students. Through the Teacher Dashboard, teachers can log in securely, upload student assignments, and view AI-generated grades, feedback, and performance insights. The Student Dashboard allows students to log in, view their graded results, download feedback, and track academic progress. These dashboards communicate with the backend through secure APIs. The Backend Layer acts as the control center, managing all requests, authentication, and communication between the frontend and AI modules. When a teacher uploads an assignment, it is processed through the Grading API, which sends the data to the AI Processing Unit for evaluation.

The AI Processing Unit forms the intelligence core of Gradia AI, combining technologies such as Google Vision API, OpenCV, and Gemini 1.5 Pro. The Vision API extracts handwritten or printed text from scanned answer sheets using Optical Character Recognition (OCR), while OpenCV handles preprocessing tasks like image enhancement and text alignment to ensure clarity and accuracy. Once the content is extracted, Gemini 1.5 Pro analyzes the responses, compares them with reference materials, and generates accurate grading along with personalized feedback highlighting the student's strengths and areas for improvement. The processed results and feedback are then stored in the Database Layer, which uses MongoDB for secure and scalable data storage.

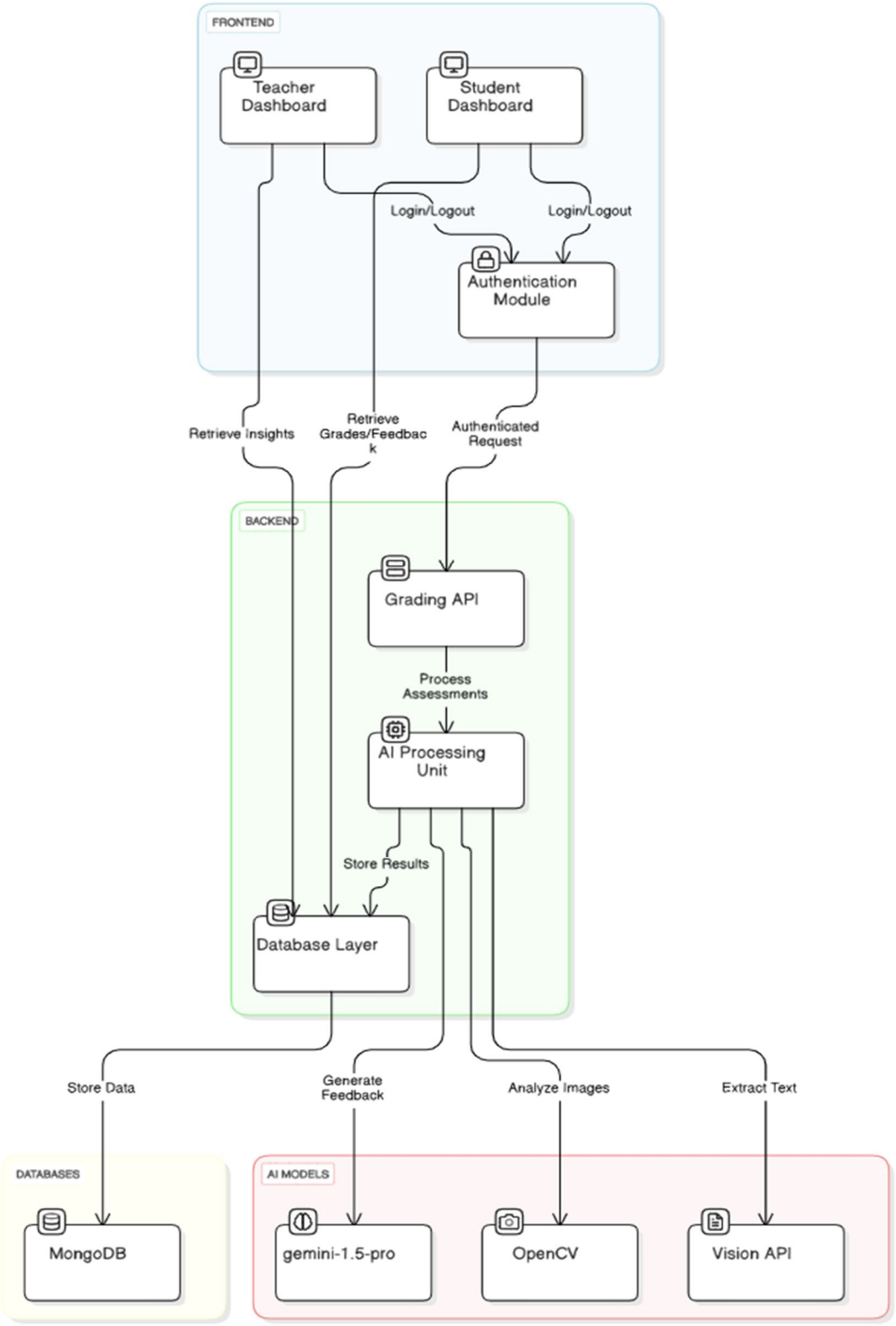


Figure 3.3: Data Flow Diagram

1) Frontend Layer

The Frontend Layer of Gradia AI serves as the primary interaction hub between users and the system, offering an intuitive and user-friendly interface for both teachers and students. It enables smooth navigation and efficient task management, ensuring that users can easily perform their respective academic functions. The Teacher Dashboard allows educators to upload student assignments or answer sheets for evaluation. In addition, it displays detailed analytics such as grade summaries, performance graphs, and AI-generated feedback reports.

Teachers can review, modify, and send personalized feedback directly to students, thereby ensuring accuracy and individualized support. The Student Dashboard, on the other hand, allows learners to log in securely and access their evaluated assignments. Students can view grades, download marked answer sheets, and read AI-generated feedback that highlights their strengths and areas for improvement. They can also track their progress over time, promoting self-assessment and continuous learning. The user interface includes responsive design, color-coded indicators, and modern data visualizations to make interaction engaging and interpretation of results seamless. All communication between the frontend and backend occurs securely through encrypted API calls, ensuring the protection of user data and maintaining a smooth workflow.

2) Backend Layer

The Backend Layer functions as the central processing system of Gradia AI, managing requests, orchestrating workflows, and connecting the frontend with the AI Processing and Database layers. It ensures that data exchange across the system is efficient, reliable, and secure.

The Authentication and Authorization component verifies login credentials for teachers, students, and administrators, providing access based on predefined permissions. This maintains data confidentiality and system integrity. The Grading API acts as the communication bridge between user input and the AI models. It receives uploaded assignments, sends them to the AI Processing Unit for analysis, and retrieves the evaluation results and feedback to be displayed on the frontend. Furthermore, the backend includes a Feedback Management module that organizes AI-generated feedback, maps it to corresponding students, and stores it in the database for future retrieval. In addition, the backend performs Error Handling and Logging, monitoring the status of AI operations and managing exceptions such as incomplete data or unreadable files. This ensures smooth system operation and transparency throughout the grading process.

3) AI Processing Unit

At the core of Gradia AI lies the AI Processing Unit, which integrates multiple intelligent technologies to simulate the human-like evaluation process of teachers. This unit comprises three major AI components — Google Vision API, OpenCV, and Gemini 1.5 Pro — along with Natural Language Processing (NLP) capabilities for deep contextual understanding. The Google Vision API performs Optical Character Recognition (OCR) to extract handwritten or printed text from scanned answer sheets, allowing the system to read and interpret student responses automatically.

OpenCV is responsible for preprocessing tasks such as image resizing, noise removal, contrast enhancement, and layout correction, improving the clarity of the extracted content. The Gemini 1.5 Pro model functions as the intelligent grader; it analyzes the extracted text, compares student responses with reference materials, and generates both objective and descriptive scoring NLP techniques ensure contextual understanding of essay-type answers, enabling accurate and unbiased grading. Collectively, the AI Processing Unit ensures that Gradia AI delivers fast, consistent, and fair evaluations, minimizing the manual workload for teachers while maintaining academic accuracy.

4) Database Layer

The Database Layer, powered by MongoDB, serves as the secure foundation for data storage and management in Gradia AI. It efficiently handles both structured and unstructured data to support seamless retrieval and high-performance analytics. The database stores essential information such as student profiles, login credentials, uploaded assignments, evaluation results, and AI-generated feedback. Additionally, it maintains performance analytics, allowing teachers to access summarized reports and trend data that highlight student progress over time. MongoDB's flexible document-based architecture enables scalability, while built-in encryption and access control mechanisms ensure data integrity and protection from unauthorized access. This layer plays a crucial role in ensuring that every piece of information — from assignment uploads to performance insights — remains well-organized, secure, and easily accessible when needed by teachers or students.

©IJRASET: All Rights are Reserved | SJ Impact Factor 7.538 | ISRA Journal Impact Factor 7.894 |

5) System Advantages

The Gradia AI system offers multiple advantages that enhance both teaching and learning experiences. It automates the grading process, drastically reducing the manual workload for educators. The AI-generated feedback is personalized and constructive, helping students understand their mistakes and improve accordingly. The system promotes transparency and accessibility, ensuring that both teachers and students can view results anytime through their respective dashboards. By digitizing the evaluation process, Gradia AI also reduces paper usage, contributing to a more sustainable and eco-friendly academic environment. Furthermore, the system supports data-driven decision-making by providing performance analytics that help teachers identify learning gaps and adopt better teaching strategies. Overall, Gradia AI transforms the traditional grading process into an intelligent, efficient, and eco-conscious solution aligned with the goals of Quality Education (UN SDG 4).

IV. SYSTEM REQUIREMENTS

A. Hardware Requirements

1) Processor: Quad-core processor or higher (Intel i5 / AMD Ryzen 5 or above)
2) RAM: Minimum 8 GB (16 GB recommended for handling AI processing and data operations)

B. Software Requirements

1) Operating System: Windows 10 or later / Linux (Ubuntu 20.04+)
2) Frontend Technologies: HTML5, CSS3, JavaScript
3) Backend Framework: Python (Flask / Django)
4) AI Libraries: OpenCV, Google Vision API, Gemini 1.5 Pro (Generative AI), and NLP modules
5) Database: MongoDB
6) Cloud Storage: Google Cloud
7) Version Control: GitHub

V. IMPLEMENTATION

A. Data Collection

In the context of the Gradia AI System, data collection plays a vital role in automating the grading process and generating personalized feedback efficiently. The system collects data from teachers, students, and AI-generated outputs, enabling a seamless workflow from assignment submission to performance analysis. All collected data supports grading accuracy, user management, and academic insight generation.

User-Generated Data

Teachers upload student answer sheets or assessment files through the Teacher Dashboard. Each upload includes metadata such as student name, ID, subject, class, and exam details. Teachers may also enter grading rubrics or reference materials to guide AI evaluation. Students, on the other hand, provide their login credentials and access their evaluated assignments and feedback. This user-contributed data forms the foundation for personalized grading and learning analytics.

System-Generated Data

Once the teacher uploads an answer sheet, the system automatically extracts, analyzes, and evaluates the content through the AI Processing Unit. It generates structured grading data, personalized feedback, and performance summaries. The system also records timestamps, evaluation logs, and AI confidence scores for transparency and auditing purposes.

Data Integration

The Gradia AI architecture integrates multiple modules — including the Teacher Dashboard, Student Dashboard, AI Processing Unit, and MongoDB Database — into a unified system. Data flows seamlessly between the frontend and backend through secure APIs. The system can also be integrated with institutional learning management systems (LMS) or external databases, allowing schools and universities to manage assessments and results within a centralized environment.

Data Validation and Quality Control

To ensure the reliability of results, Gradia AI employs strict data validation techniques. Uploaded files are verified for format compatibility, completeness, and legibility. The backend performs server-side validation to detect errors such as corrupted files, missing student information, or duplicate submissions. The AI model also undergoes continuous accuracy testing to maintain consistent and unbiased grading standards.

Privacy and Security

Data privacy is a top priority in the Gradia AI system. All user credentials, assignment files, and grading information are securely encrypted during transmission and storage. Role-based authentication ensures that only authorized users (teachers, students, and admins) can access specific data. The system follows institutional and international data protection standards to maintain full confidentiality and prevent unauthorized access or data leakage.

B. Components Design

Designing the components for the Gradia AI System is a crucial step in building a robust, intelligent, and user-friendly platform. Each component in the system is carefully structured to manage grading workflows, process student assessments, and display AI-generated insights efficiently.

Understand the Project Requirements

The first step involves clearly understanding the project's objectives and desired functionalities. Gradia AI is designed to automate the grading process, generate personalized feedback, and provide data-driven insights for teachers and students. Key requirements include secure login and authentication, assignment uploads, AI-based grading and feedback, performance tracking, and role-based dashboards for teachers and students..

Break Down the User Interface

The user interface is divided into multiple modules catering to different roles—teachers, students, and administrators. Each user has a dedicated dashboard with relevant functionalities. Teachers can upload student assignments, view AI-evaluated grades, and send feedback, while students can view their performance and improvement suggestions.

Identify Reusable Elements

Reusable elements such as forms, buttons, tables, charts, and alert messages are designed uniformly across all dashboards. These UI components are implemented consistently using Angular for seamless integration and maintainability. Reusability reduces redundancy in development and ensures design consistency throughout the application.

Handling Routing and Navigation

Routing and navigation play a critical role in connecting different parts of the system. The application uses Angular's routing mechanism to direct users to their specific dashboards based on their login credentials. Secure routing ensures that teachers, students, and administrators access only their respective modules. Smooth transitions between pages like login, grading dashboard, feedback section, and performance insights enhance the user experience and prevent unauthorized access.

Implement Notifications and Alerts

The system includes automated notifications and alert components to keep users informed about important activities. Teachers receive notifications once the grading process is completed, while students get alerts when new feedback or performance reports are available. These real-time notifications ensure timely communication and enhance engagement. The notification system is integrated into both dashboards to improve transparency and streamline communication between teachers and students.

Data Flow

In the AI-Powered Teacher Assistant System, data flows securely from the teacher's dashboard to the application server, where it undergoes validation, AI analysis, and processing before being stored in the database. When a teacher uploads student assignments, the backend system performs grading using AI models and stores the results and feedback in the database. The student and teacher dashboards access this stored data dynamically for displaying performance insights and reports.

UI/UX Design

The user interface of the AI-Powered Teacher Assistant System is designed with a focus on simplicity, clarity, and usability. Built using HTML, CSS, and Angular, the frontend ensures a responsive and visually appealing design. Teachers and students can easily navigate through dashboards, upload assignments, and view feedback effortlessly. The UI incorporates meaningful visual indicators, charts, and progress bars to present academic insights clearly.

Accessibility

Accessibility is an integral part of the design, ensuring that all users can interact with the system comfortably. The system supports keyboard navigation, clear labeling for all form elements, and easily distinguishable buttons.

Security and Authentication

The system includes role-based authentication to maintain strict access control between teachers, students, and administrators. Teachers and admins have permissions to upload, grade, and review assignments, while students can only view their respective

results and feedback. Strong input validation, session management, and encryption are implemented to prevent unauthorized access and data tampering.

Documentation

Comprehensive documentation is maintained throughout the development process. Each module—such as grading, feedback generation, and dashboards—is described in detail, including its purpose, parameters, and functionality. updates or maintenance tasks, ensuring the system remains well-documented and easy to manage.

Third-Party Libraries and Tools

The system integrates several third-party libraries and APIs to enhance its performance and functionality. Google Generative AI (for feedback generation), OpenAI API, and OCR tools (for assignment text extraction) are used for AI-driven operations. The backend is built with Python Flask, and PostgreSQL is used for secure data management. The frontend utilizes Angular libraries for charts and UI components.

Scalability

The component design of the AI-Powered Teacher Assistant System supports high scalability and modularity. Each feature—such as grading, feedback, or report generation—is developed as an independent module that can be extended easily.

Code Style and Conventions

Consistent coding standards are followed across the entire project to maintain readability and efficiency. Proper indentation, variable naming conventions, and code commenting are enforced to promote clean development practices

Version Control

Version control is managed through Git and GitHub, allowing seamless collaboration among developers. Each feature, bug fix, or enhancement is handled through dedicated branches and pull requests.

Code Review

Regular code reviews are conducted to maintain quality, performance, and adherence to standards. Peer review sessions help detect potential issues early, improve logic implementation, and ensure that all modules integrate smoothly with the overall system architecture.

Iterate and Refine

As the project evolves, continuous refinement of components ensures that the system meets emerging requirements and user feedback. The structure and features are regularly optimized based on testing outcomes and usability reviews. The iterative process enhances overall performance, usability, and system stability.

Testing and Quality Assurance

Comprehensive testing is carried out at every stage of the project. This includes unit testing, integration testing, and user acceptance testing. Each module—such as login, grading, and feedback—is tested independently to ensure functionality and data consistency. Rigorous QA ensures the system is secure, efficient, and free from critical bugs.

Deployment and Maintenance

The AI-Powered Teacher Assistant System is deployed on a secure web server for high availability and performance. Maintenance activities, including bug fixes and feature updates, are scheduled periodically. Any reported issues are resolved promptly, and system logs are reviewed for performance optimization. The maintenance process ensures the system remains stable, efficient, and aligned with evolving educational needs.

C. Software Description

1) HTML: HTML (HyperText Markup Language) provides the foundational structure for all web pages in Gradia AI. It defines the layout of elements such as assignment upload forms, AI feedback sections, grading results, and dashboards.
2) CSS: CSS (Cascading Style Sheets) is used to style and design the visual aspects of Gradia AI. It enhances user experience by ensuring a clean, modern, and consistent interface across all modules.
3) Python (Flask Framework): Python Flask serves as the backend framework for Gradia AI. It handles core logic such as user authentication, assignment uploads, and communication with the AI processing unit. Flask APIs connect the frontend with the AI model, ensuring smooth data exchange between the user interface and the backend services.
4) MONGODB: MongoDB Compass are database management tools used to visualize, query, and manage stored data. They provide an intuitive interface to monitor grading results, feedback logs, and user details in real-time.

5) Google Generative AI (Gemini 1.5 Pro): The Gemini 1.5 Pro AI model powers the core grading and feedback generation system. It uses Natural Language Processing (NLP) and machine learning to analyze student responses, generate personalized feedback, and provide teachers with detailed grading insights.

6) VISUAL STUDIO CODE: Visual Studio Code (VS Code) is a versatile and lightweight source code editor developed by Microsoft, widely adopted by developers for building modern web applications. In the AI-Powered Teacher Assistant System, VS Code serves as the main development environment for creating and managing the frontend (Angular, HTML, CSS, and JavaScript) and backend (Python Flask/Django) components. It provides an efficient platform for writing, debugging, and organizing project files with an intuitive interface that allows seamless switching between editors, terminals, and version control tools. Features such as syntax highlighting, IntelliSense, debugging, and code formatting enhance accuracy and productivity.

   VS Code also integrates Git version control, enabling the development team to collaborate efficiently, track changes, and maintain a clean, well-structured codebase. With extensive extension support and cross-platform compatibility (Windows, Linux, macOS), it ensures a smooth and consistent development experience. Overall, VS Code provides a flexible and productive workspace for developing, testing, and maintaining the AI-Powered Teacher Assistant System.

7) PYTHON (FLASK / DJANGO FRAMEWORK): Python serves as the primary backend language for the AI-Powered Teacher Assistant System, using the Flask or Django framework for handling server-side logic and API integration. The backend is responsible for connecting the AI models, managing user authentication, and processing grading and feedback requests. Flask/Django provides a scalable and secure environment that allows teachers to upload student sheets, automatically grade them using integrated AI models, and store the results in a MONGODB database. These frameworks simplify API creation and communication between the frontend and the AI engine, ensuring fast and reliable data processing.

8) GOOGLE CLOUD VISION API: Google Cloud Vision API is used for optical character recognition (OCR) and intelligent data extraction in the AI-Powered Teacher Assistant System. It processes uploaded student answer sheets or handwritten content and converts them into structured digital text that can be analyzed by the AI model. The Vision API provides high accuracy in text recognition, enabling the system to handle diverse document types, including scanned sheets and PDF submissions. By integrating this service, the project eliminates manual data entry and speeds up the grading. Its cloud-based architecture ensures scalability, reliability, and continuous improvement in recognition accuracy, making it an essential component of the system's AI-based automation workflow.

9) OPENAI / GEMINI API: The AI-Powered Teacher Assistant System utilizes OpenAI or Google Gemini APIs to generate automated grading, personalized feedback, and assessment insights. These generative AI models analyze the extracted text from student sheets and produce human-like evaluation comments and suggestions for improvement. The integration enhances teaching efficiency by providing detailed, consistent, and unbiased feedback. The API's natural language understanding capabilities allow it to assess descriptive answers, summarize key points, and strategies. This feature represents the core intelligence of the system, aligning with the project's goal of supporting UN SDG 4: Quality Education through technology-driven, personalized learning support.

10) GIT AND GITHUB: Git is a distributed version control system that helps developers manage and track changes in the project codebase, while GitHub provides a cloud-based platform for hosting and collaborating in repository. In the AI-Powered Teacher Assistant System, Git and GitHub are used to maintain version control across frontend and backend modules, enabling team collaboration, bug tracking, and continuous integration.

11) MONGODB DATABASE: MongoDB is an open-source, NoSQL database designed to store large volumes of structured and unstructured data efficiently. In the AI-Powered Teacher Assistant System, MongoDB is used to manage and store diverse types of data such as student details, uploaded assignments, AI-generated grades, and personalized feedback. Unlike traditional relational databases, MongoDB uses a document-oriented model, storing data in flexible JSON-like collections. This allows the system to handle variable data formats generated from AI models and OCR outputs without the need for predefined schemas.

D. Result

The AI-Powered Teacher Assistant System demonstrates significant improvements in automating the grading and feedback process, offering teachers a robust, intelligent tool to evaluate student performance efficiently. The system employs advanced AI algorithms to accurately analyze both textual and visual submissions, ensuring precise assessment across diverse question formats. Constructive feedback is generated automatically for each student, highlighting strengths, identifying areas for improvement, and providing actionable guidance for learning enhancement.

All assessment data is securely stored in a centralized MongoDB database, allowing seamless access and management. Teachers can monitor class-wide performance through interactive, real-time dashboards that display performance trends, learning gaps, and comparative analytics, enabling data-driven instructional decisions. For students, the platform ensures timely, personalized feedback, fostering continuous improvement and self-directed learning.

By integrating AI into the educational workflow, the system not only reduces teacher workload but also promotes efficiency, transparency, and consistent evaluation standards. Furthermore, the platform aligns with UN Sustainable Development Goal 4 (Quality Education) by supporting equitable, technology-enhanced learning experiences. Overall, the project delivers a scalable, reliable, and intelligent solution that strengthens both teaching effectiveness and student learning outcomes, demonstrating the transformative potential of AI in modern education.

VI. CONCLUSION AND FUTURE ENHANCEMENT

A. Conclusion

The AI-Powered Teacher Assistant System marks a significant step toward the digital transformation of education by leveraging Artificial Intelligence (AI) and automation to simplify and enhance the teaching and learning process. The system efficiently automates the evaluation of student assignments, generates personalized feedback, and provides insightful performance analytics for teachers. This not only minimizes manual effort but also ensures fairness, accuracy, and consistency in grading. Through the integration of Flask (Python) for backend processing, Angular for an interactive frontend interface, and MongoDB for secure data storage and management, the project delivers a smooth and responsive user experience. Teachers can upload student answer sheets, receive AI-generated grades and feedback, and monitor student progress through dynamic dashboards. Students, in turn, can access their performance data and receive constructive feedback that helps them improve their learning outcomes. The incorporation of Natural Language Processing (NLP) and Optical Character Recognition (OCR) technologies enhances the system's ability to process written responses, analyze student work, and provide meaningful evaluation insights. The system's intuitive design ensures accessibility and usability for educators with minimal technical expertise. It demonstrates how technology can empower educators, streamline administrative tasks, and improve learning efficiency in both academic and institutional settings.

B. Future Scope

The AI-Powered Teacher Assistant System has vast potential for future development and enhancement. As technology continues to advance, this system can evolve into a comprehensive platform that fully supports digital and personalized education. In the future, it can be integrated with popular Learning Management Systems (LMS) such as Google Classroom or Moodle to automatically grade assignments, record marks, and synchronize feedback. Advanced AI techniques like plagiarism detection and predictive analytics can be incorporated to ensure academic integrity and provide teachers with insights into student performance trends. Additionally, features such as voice and speech analysis can enable the evaluation of oral presentations and communication skills. The system can also be improved with adaptive learning insights, where AI identifies each student's learning patterns and recommends customized learning materials to address weaknesses. Cloud-based deployment can further enhance scalability, ensuring that large datasets and multiple users can be handled efficiently. A mobile application version of the platform would allow teachers and students to access the system from anywhere, increasing convenience and flexibility. Furthermore, adding multilingual and accessibility features would make the system inclusive for students from diverse backgrounds and with different learning needs. Overall, the future scope of the AI-Powered Teacher Assistant System lies in expanding its intelligence, accessibility, and integration capabilities to create a more efficient, data-driven, and inclusive educational ecosystem.

APPENDICES

SOURCE CODE

Teacher Dashboard.html

```
<!DOCTYPE html>
<html lang="en">
<head>
  <meta charset="UTF-8" />
  <meta name="viewport" content="width=device-width, initial-scale=1.0"/>
  <title>Teacher Dashboard - Gradia AI</title>
 <scriptsrc="https://kit.fontawesome.com/a076d05399.js"crossorigin="anonymous"></script>
```

ISSN: 2321-9653; IC Value: 45.98; SJ Impact Factor: 7.538
Volume 13 Issue XI Nov 2025- Available at www.ijraset.com

```
<style>
 * {
  margin: 0;
  padding: 0;
  box-sizing: border-box;
  font-family: Arial, sans-serif;
 }
 body {
  background: #EEF2F5;
  color: #333;
 }
 .header {
  display: flex;
  align-items: center;
  justify-content: space-between;
  background: white;
  padding: 15px 20px;
  border-bottom: 3px solid #007BFF;
  box-shadow: 0 2px 4px rgba(0, 0, 0, 0.1);
 }
 .logo {
  display: flex;
  align-items: center;
  font-size: 18px;
  font-weight: bold;
  color: #007BFF;
 }
 .logo img {
  width: 30px;
  height: 30px;
  margin-right: 8px;
 }
 .dashboard-title {
  font-size: 24px;
  font-weight: bold;
  color: #333;
 }
 .main-container {
  display: flex;
  flex-direction: column;
  align-items: center;
  padding: 40px 20px;
 }
 .main-container h2 {
  font-size: 28px;
  color: #007BFF;
  margin-bottom: 30px;
  display: flex;
  align-items: center;
 }
```

```
.main-container h2 i {
  margin-right: 10px;
}
.input-field {
  width: 637px;
  padding: 12px 15px;
  margin: 10px 0;
  border: 1.5px solid #007BFF;
  border-radius: 5px;
  font-size: 16px;
}

.button-group {
  display: flex;
  gap: 20px;
  margin: 20px 0;
  flex-wrap: wrap;
  justify-content: center;
}

.upload-button, .scan-button {
  flex: 1;
  min-width: 200px;
  padding: 12px;
  background: white;
  border: 1.5px solid #007BFF;
  border-radius: 8px;
  font-size: 16px;
  color: #333;
  cursor: pointer;
  text-align: center;
  transition: 0.3s;
}

.upload-button:hover, .scan-button:hover {
  background: #f0f8ff;
}

.action-button {
  width: 637px;
  padding: 12px;
  background: #007BFF;
  color: white;
  border: none;
  border-radius: 5px;
  font-size: 16px;
  font-weight: bold;
  cursor: pointer;
  margin-top: 10px;
  transition: background 0.3s;
```

```
    }

    .action-button:hover {
      background: #0056b3;
    }

    .disabled-button {
      width: 637px;
      padding: 12px;
      background: #007BFF;
      color: white;
      border: none;
      border-radius: 5px;
      font-size: 16px;
      font-weight: bold;
      margin-top: 10px;

    }
    .feedback-textarea {
  width: 637px;
  height: 50px;
  padding: 12px 15px;
  margin-top: 20px;
  border: 1.5px solid #007BFF;
  border-radius: 5px;
  font-size: 16px;
  resize: vertical; /* Optional: allows resizing vertically */
  background: #fff;
  color: #333;
}
.profile-pic {
      width: 40px;
      height: 40px;
      border-radius: 50%;
      object-fit: cover;
      border: 2px solid #007BFF;
    }
    .response-box {
  margin-top: 20px;
  width: 637px;
  min-height: 100px; /* Optional: ensures box has some height even when empty */
  padding: 20px;
  border: 1.5px solid #007BFF;
  border-radius: 8px;
  background-color: #fff;
  color: #000; /* Black font color */
  box-shadow: 0 2px 8px rgba(0, 0, 0, 0.05);
  font-size: 16px;
}
.response-box h3,
```

```
.response-box h4,
.response-box p,
.response-box li,
.response-box ul {
  color: #000; /* Force black text inside any tag */
}
  </style>
</head>
<body>
  <!-- Header -->
  <header class="header">
    <div class="logo">
      <img src="logo.png" alt="Gradia AI Logo" />
      <span>Teacher Dashboard</span>
    </div>
    <div class="profile-container">
      <img src="meena.jpg" alt="Profile" class="profile-pic">
    </div>
  </header>
  <!-- Main Content -->
  <div class="main-container">
    <h2><i class="fas fa-file-alt"></i> Assessment & Grading</h2>
    <!-- Input Fields -->
    <input id="name" type="text" placeholder="Student Name" class="input-field" required />
    <input id="registerNo" type="text" placeholder="Register No" class="input-field" required />
    <input id="class" type="text" placeholder="Class" class="input-field" required />

    <!-- Upload Inputs (hidden) -->
    <input type="file" id="questionPaperFile" accept="application/pdf" style="display: none;" />
    <input type="file" id="answerPaperFile" accept="application/pdf" style="display: none;" />
    <!-- Upload Buttons and Scan Button -->
    <div class="button-group">
 <divclass="upload-button" onclick="document.getElementById('questionPaperFile').click()">
       <i class="fas fa-upload"></i> Upload Question Paper
     </div>
<divclass="upload-button" onclick="document.getElementById('answerPaperFile').click()">
       <i class="fas fa-upload"></i> Upload Answer Paper
     </div>
     <div class="scan-button" onclick="scanPapers()">
       <i class="fas fa-camera"></i> Scan Paper
     </div>
    </div>

    <!-- Action Buttons -->
    <button class="action-button" id="analyzeAndGrade">Analyze & Grade</button>
    <!-- Feedback & Grade Response Textarea -->
    <div id="responseMessage" class="response-box"></div>
    <button class="disabled-button" id="storeFeedback" disabled>Store Results</button>
  </div>
  <script>
```

```
// Store the latest feedback and grade
let latestFeedback = "";
let latestGrade = "";
let latestFilename = "";
const questionFileInput = document.getElementById('questionPaperFile');
const answerFileInput = document.getElementById('answerPaperFile');
const responseBox = document.getElementById('responseMessage');
const storeFeedbackBtn = document.getElementById('storeFeedback');
const feedbackTextarea = document.getElementById('responseMessage'); // NEW: grab textarea element
// File input handlers (optional: alert users what they selected)
questionFileInput.addEventListener('change', (event) => {
  const file = event.target.files[0];
  if (file) {
    alert(`Selected Question Paper: ${file.name}`);
  }
});
answerFileInput.addEventListener('change', (event) => {
  const file = event.target.files[0];
  if (file) {
    alert(` Selected Answer Paper: ${file.name}`);
  }
});
// Format the feedback to display as HTML
function formatFeedback(feedback) {
  let formatted = feedback
    .replace(/1\.\s*\*\*(.*?)\*\*/g, '<h4>1. $1</h4><ul>')
    .replace(/2\.\s*\*\*(.*?)\*\*/g, '</ul><h4>2. $1</h4><ul>')
    .replace(/3\.\s*\*\*(.*?)\*\*/g, '</ul><h4>3. $1</h4><ul>')
    .replace(/\*\*(.*?)\*\*/g, '<strong>$1</strong>')
    .replace(/\*\s(.*?)(\n|$)/g, '<li>$1</li>')
    .replace(/\n/g, '<br>');
  if (!formatted.endsWith('</ul>')) {
    formatted += '</ul>';
  }
  return formatted;
}
// Analyze and Grade Handler
document.getElementById('analyzeAndGrade').addEventListener('click', async () => {
  if (questionFileInput.files.length === 0 || answerFileInput.files.length === 0) {
    alert("Please upload both Question Paper and Answer Paper PDFs.");
    return;
  }
  const formData = new FormData();
  formData.append('questionPaper', questionFileInput.files[0]);
  formData.append('answerPaper', answerFileInput.files[0]);
  latestFilename = answerFileInput.files[0].name;

  responseBox.innerHTML = " Analyzing and grading... Please wait.";
  responseBox.style.color = "blue";
  try {
```

```
    const response = await fetch("http://127.0.0.1:5001/teacher/compare_and_grade",{
      method: 'POST',
      body: formData
    });
    const result = await response.json();
    if (response.ok) {
      latestFeedback = result.feedback || result.message;
      latestGrade = result.grade || "N/A";
      responseBox.innerHTML = `
        <h3>Feedback & Grade</h3>
        ${formatFeedback(latestFeedback)}
        <h4>Final Grade: <span style="color: #28a745;">${latestGrade}</span></h4>
      `;
      // Show raw feedback + grade inside the textarea
      feedbackTextarea.value = `${latestFeedback}\n\nFinal Grade: ${latestGrade}`;
      storeFeedbackBtn.disabled = false;
    } else {
      responseBox.innerHTML = `<p style="color:red;"> ${result.message || "Something went wrong!"}</p>`;
      feedbackTextarea.value = ""; // Clear textarea on error
      storeFeedbackBtn.disabled = true;
    }
  } catch (error) {
    console.error(" Error:", error);
    responseBox.innerHTML = `<p style="color:red;"> Error analyzing files!</p>`;
    feedbackTextarea.value = ""; // Clear textarea on error
    storeFeedbackBtn.disabled = true;
  }
});
// Store feedback/grade handler
storeFeedbackBtn.addEventListener('click', async () => {
  const name = document.getElementById('name').value.trim();
  const registerNo = document.getElementById('registerNo').value.trim();
  const studentClass = document.getElementById('class').value.trim();
  if (!latestFeedback || !latestGrade) {
    alert(" No feedback/grade available. Please analyze first.");
    return;
  }
  if (!name || !registerNo || !studentClass) {
    alert(" Please fill in Student Name, Register No, and Class!");
    return;
  }
  const payload = {
    filename: latestFilename,
    grade: latestGrade,
    feedback: latestFeedback,
    name: name,
    register_no: registerNo,
    student_class: studentClass
  };
  try {
```

```
  const response = await fetch("http://127.0.0.1:5001/teacher/store_feedback", {
   method: 'POST',
   headers: { 'Content-Type': 'application/json' },
   body: JSON.stringify(payload)
  });
  const result = await response.json();
  if (response.ok) {
   alert(" Feedback and grade stored successfully!");
   storeFeedbackBtn.disabled = true;
  } else {
   alert(result.message || " Failed to store data.");
  }
 } catch (error) {
  console.error(" Error storing feedback:", error);
  alert(" Error storing feedback!");
 }
});
function scanPapers() {
 alert(" Scan paper feature coming soon!");
}</script>
```

Python Code

```
from flask import Flask, request, jsonify, render_template
from flask_cors import CORS
import os
import re
from werkzeug.utils import secure_filename
from services.pdf_service
import process_pdf_with_structured_output
from services.ai_service import generate_feedback
from services.db_service import store_result, collection  # Import the MongoDB collection for queries
from services.ai_service import generate_feedback_comparison
app = Flask(__name__)
CORS(app)
UPLOAD_FOLDER = "uploads"
ALLOWED_EXTENSIONS = {"pdf"}
app.config["UPLOAD_FOLDER"] = UPLOAD_FOLDER
app.config["MAX_CONTENT_LENGTH"] = 16 * 1024 * 1024  # 16 MB limit
os.makedirs(UPLOAD_FOLDER, exist_ok=True)
def allowed_file(filename):
   return "." in filename and filename.rsplit(".", 1)[1].lower() in ALLOWED_EXTENSIONS
def extract_grade(feedback):
   match = re.search(r'Grade[:]*\s*(\d{1,2}\/10)', feedback, re.IGNORECASE)
   if match:
      grade = match.group(1)
      print(f" Extracted grade: {grade}")
      return grade
   else:
      print(" No grade found in feedback. Defaulting to 'N/A'.")
      return "N/A"
```

```
@app.route('/')
def index():
    return render_template('teacher1.html')
# Analyze and Grade Route (Only analyzes, doesn't store)
@app.route('/teacher/analyze_and_grade', methods=['POST'])
def analyze_and_grade():
    if "file" not in request.files:
        return jsonify({"message": " No file uploaded"}), 400
    file = request.files["file"]
    if file.filename == "":
        return jsonify({"message": " No file selected"}), 400
    if not allowed_file(file.filename):
        return jsonify({"message": " Only PDF files are allowed"}), 400
    filename = secure_filename(file.filename)
    filepath = os.path.join(app.config["UPLOAD_FOLDER"], filename)
    file.save(filepath)
    try:
        print("Starting text extraction...")
        extracted_text = process_pdf_with_structured_output(filepath)
    except Exception as e:
        return jsonify({"message": f" Error extracting text: {e}"}), 500
    if not extracted_text:
        return jsonify({"message": "No text found in the PDF"}), 400
    try:
        print(" Sending text to AI for feedback...")
        feedback = generate_feedback(extracted_text)
    except Exception as e:
        return jsonify({"message": f" Error generating feedback: {e}"}), 500
    try:
        os.remove(filepath)
    except Exception as e:
        print(f" Warning: Could not delete file {filepath}: {e}")
    grade = extract_grade(feedback)
    # Return feedback & grade but do not store yet!
    return jsonify({
        "message": " Grading complete!",
        "grade": grade,
        "feedback": feedback
    }), 200
# Store Feedback Route (Teacher decides when to store)
@app.route('/teacher/store_feedback', methods=['POST'])
def store_feedback():
    data = request.get_json()
    filename = data.get("filename")
    grade = data.get("grade")
    feedback = data.get("feedback")
    name = data.get("name")
    register_no = data.get("register_no")
    student_class = data.get("student_class")
    if not filename or not grade or not feedback or not name or not register_no or not student_class:
```

```
        return jsonify({"message": " Missing required fields!"}), 400
    try:
        store_result(filename, grade, feedback, name, register_no, student_class)
        return jsonify({"message": " Feedback and grade stored successfully!"}), 200
    except Exception as e:
        print(f" Error storing feedback: {e}")
        return jsonify({"message": f" Error storing feedback: {e}"}), 500
@app.route('/student/view_result', methods=['POST'])
def student_view_result():
    data = request.get_json()
    name = data.get("name")
    register_no = data.get("register_no")
    student_class = data.get("student_class")
    if not name or not register_no or not student_class:
        return jsonify({"message": " Missing required fields!"}), 400
    try:
        # Query MongoDB for the student result
        query = {
            "name": name,
            "registerNo": register_no,
            "class": student_class
        }
        print(f" Searching for student record: {query}")
        result = collection.find_one(query)
        if result:
            return jsonify({
                "grade": result.get("grade", "N/A"),
                "feedback": result.get("feedback", "No feedback found")
            }), 200
        else:
            return jsonify({"message": " No result found for the given credentials."}), 404
    except Exception as e:
        print(f" Error retrieving student result: {e}")
        return jsonify({"message": f" Error retrieving student result: {e}"}), 500
@app.route('/teacher/compare_and_grade', methods=['POST'])
def compare_and_grade():
    if 'questionPaper' not in request.files or 'answerPaper' not in request.files:
        return jsonify({"message": " Both files are required!"}), 400
    question_file = request.files['questionPaper']
    answer_file = request.files['answerPaper']
    if not allowed_file(question_file.filename) or not allowed_file(answer_file.filename):
        return jsonify({"message": " Invalid file format! Only PDF files allowed."}), 400
    q_filename = secure_filename(question_file.filename)
    a_filename = secure_filename(answer_file.filename)
    q_filepath = os.path.join(app.config["UPLOAD_FOLDER"], q_filename)
    a_filepath = os.path.join(app.config["UPLOAD_FOLDER"], a_filename)
    question_file.save(q_filepath)
    answer_file.save(a_filepath)
    try:
        print(" Extracting question paper text...")
```

```
        question_text = process_pdf_with_structured_output(q_filepath)
        print(f" Question text extracted ({len(question_text)} chars)")
        print(" Extracting answer paper text...")
        answer_text = process_pdf_with_structured_output(a_filepath)
        print(f" Answer text extracted ({len(answer_text)} chars)")
        if not question_text or not answer_text:
            return jsonify({"message": " Failed to extract text from one or both PDFs!"}), 400
    except Exception as e:
        return jsonify({"message": f" Error extracting PDF text: {e}"}), 500
    try:
        feedback = generate_feedback_comparison(question_text, answer_text)
    except Exception as e:
        return jsonify({"message": f"Error generating feedback: {e}"}), 500
    grade = extract_grade(feedback)
    try:
        os.remove(q_filepath)
        os.remove(a_filepath)
    except Exception as e:
        print(f" Could not delete temp files: {e}")
    return jsonify({
        "message": " Grading complete!",
        "grade": grade,
        "feedback": feedback
    }), 200
if __name__ == "__main__":
    print(" Server running at http://127.0.0.1:5001/")
```

Screen Shots



Figure 5.1: Home Page

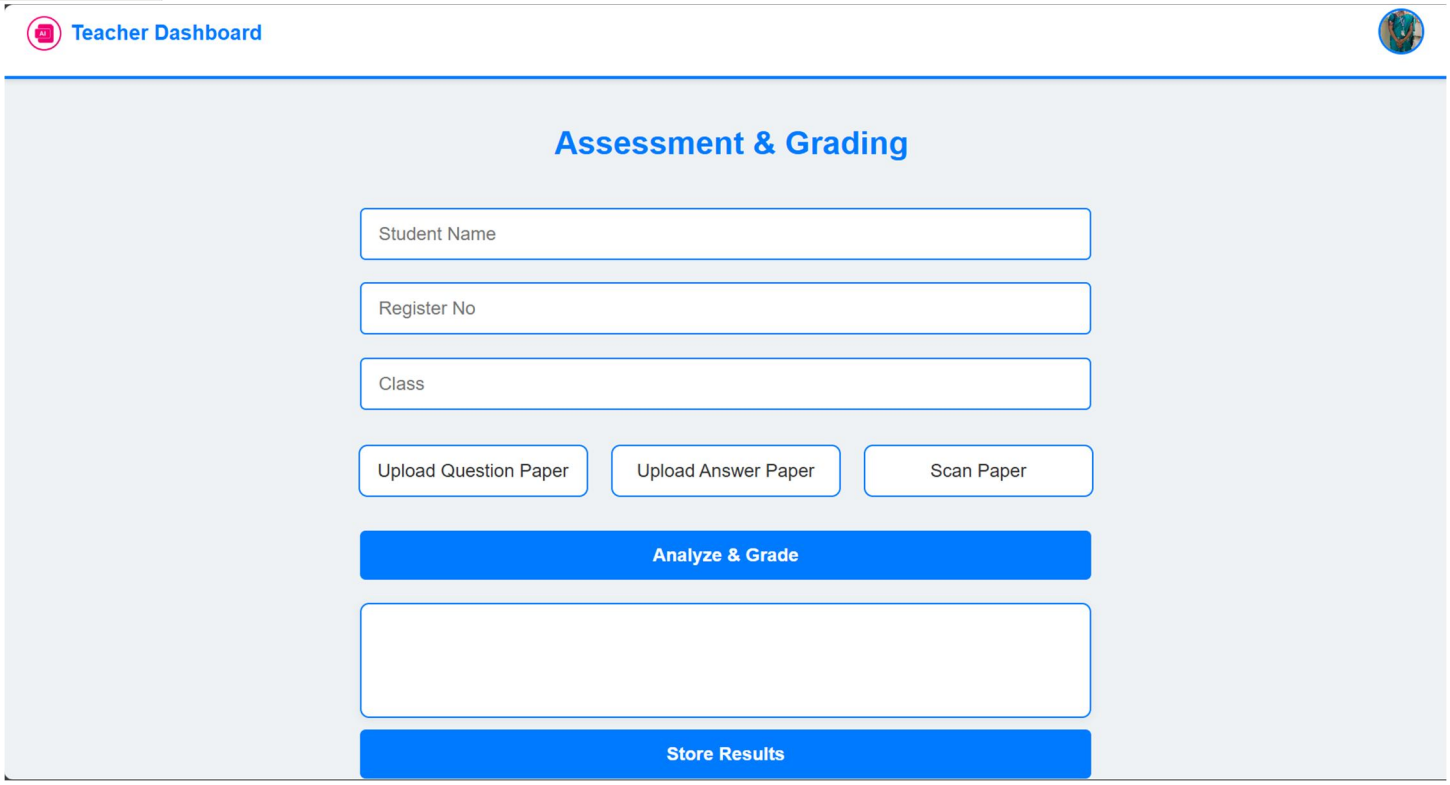


Figure 5.2: Teacher Dashboard

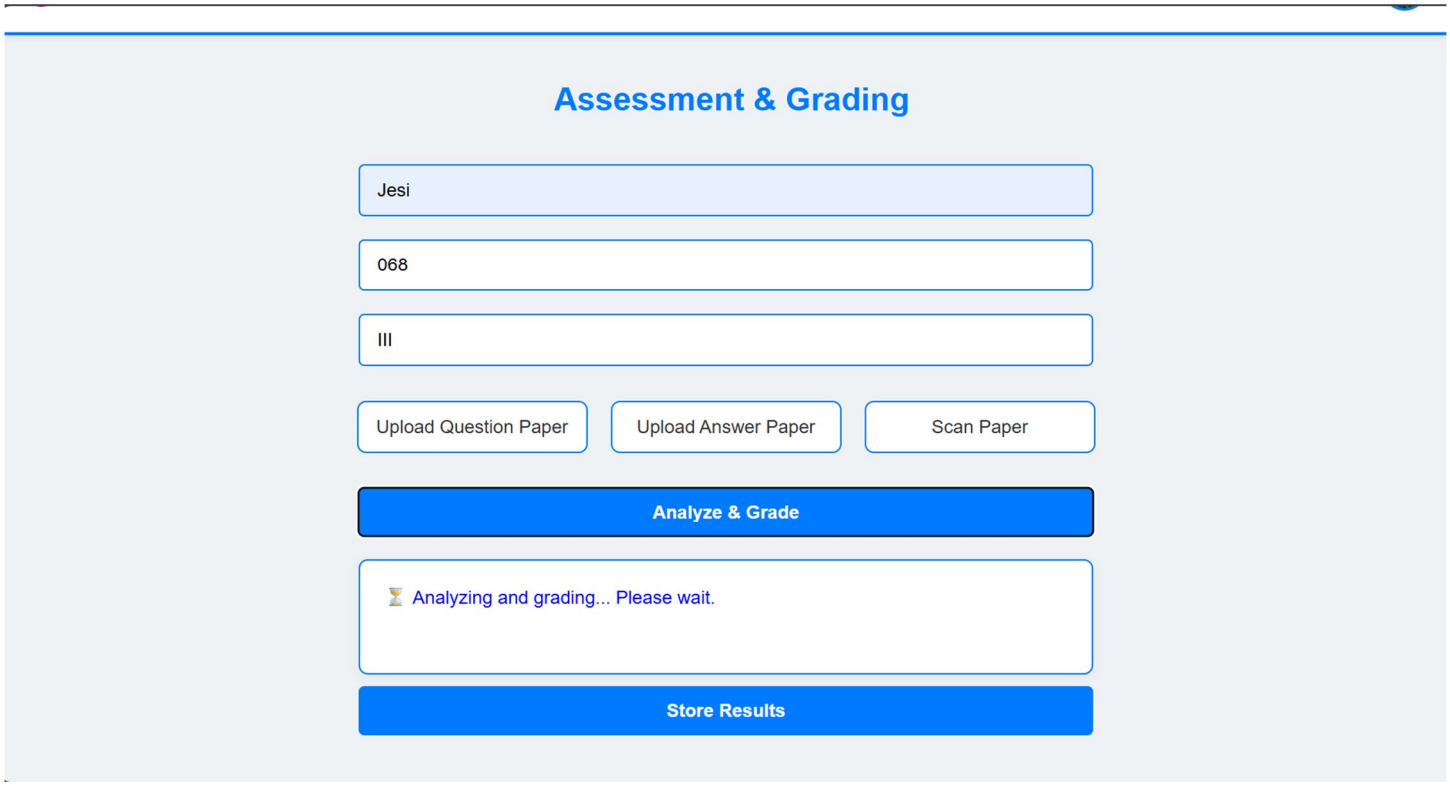


Figure 5.3: Analyze and Grade

ISSN: 2321-9653; IC Value: 45.98; SJ Impact Factor: 7.538
Volume 13 Issue XI Nov 2025- Available at www.ijraset.com

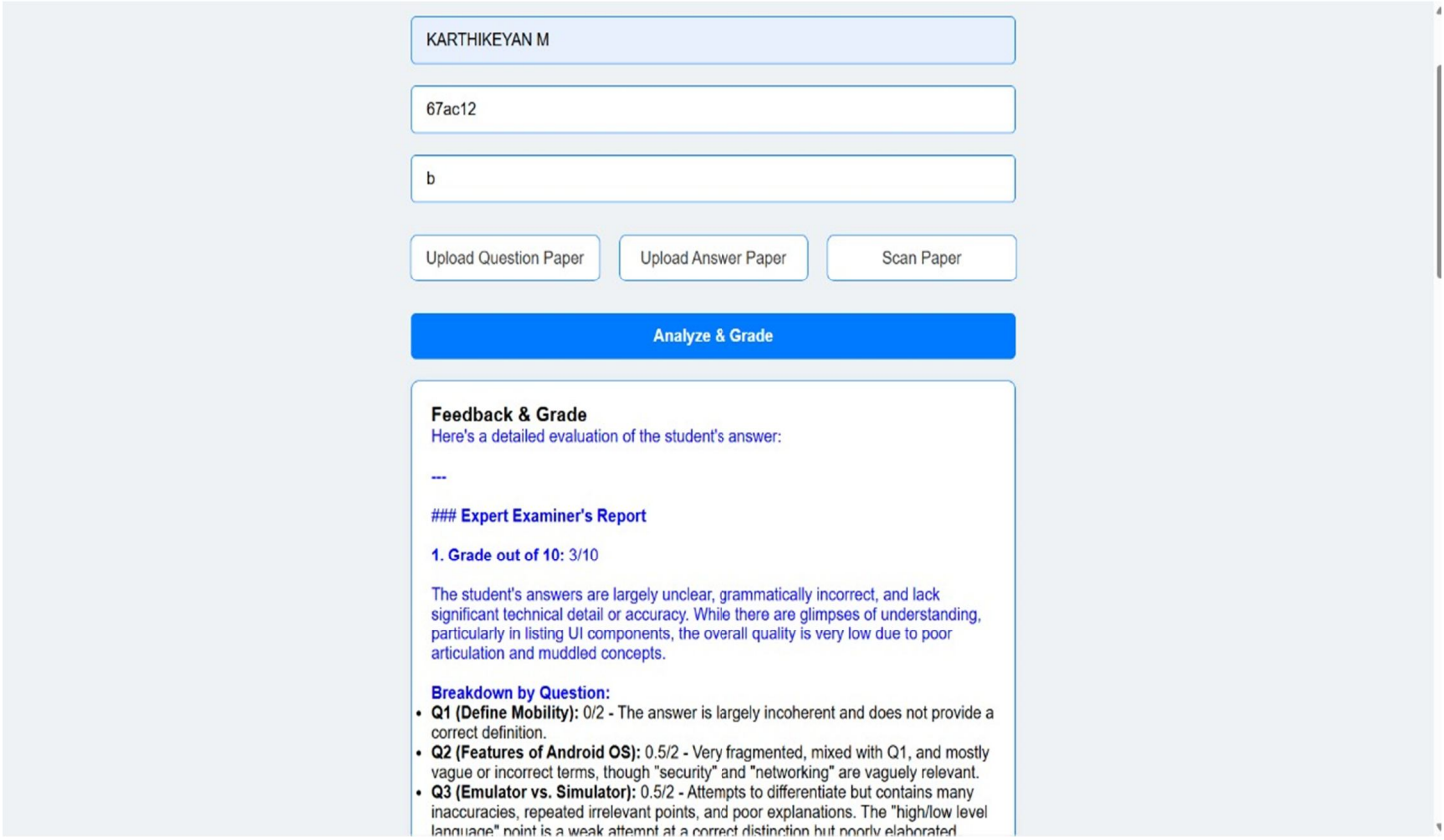


Figure 5.4: Feedback Generation

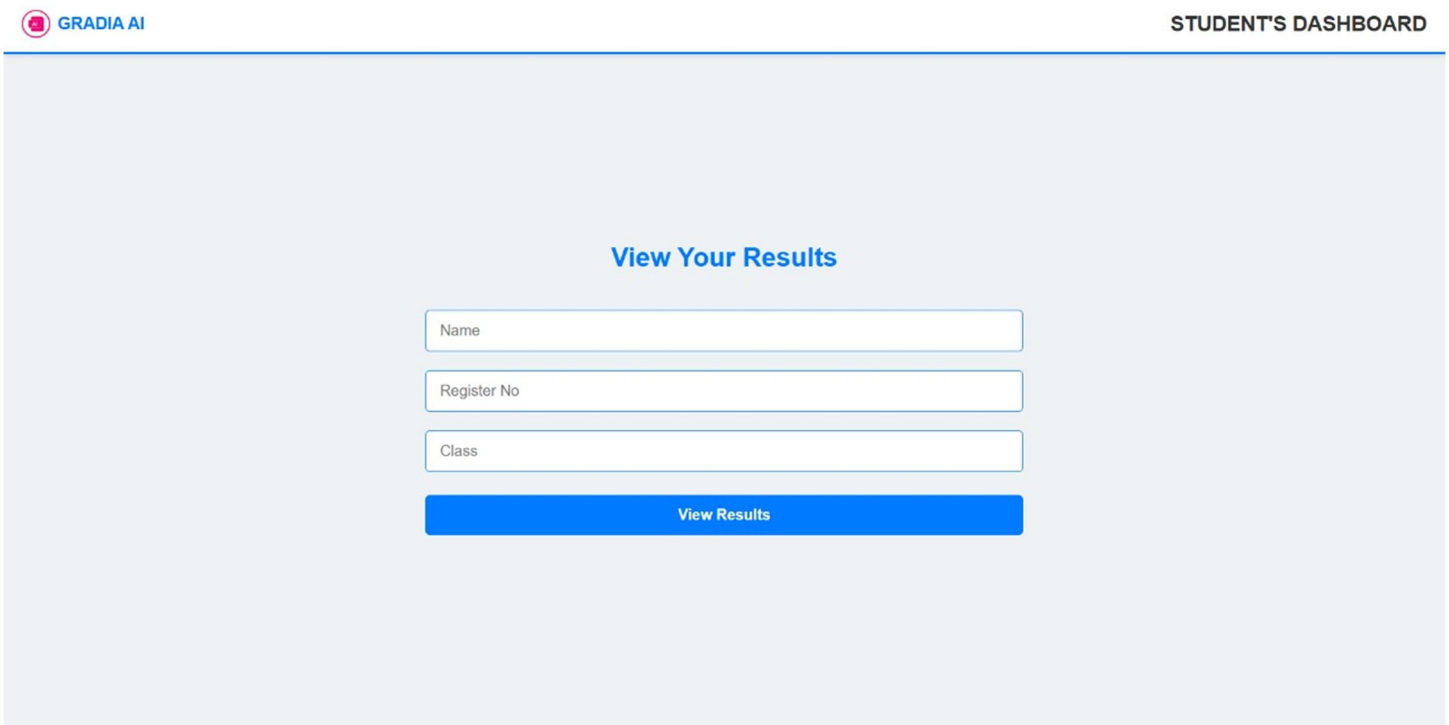


Figure 5.5: Student Dashboard

©IJRASET: All Rights are Reserved | SJ Impact Factor 7.538 | ISRA Journal Impact Factor 7.894 |

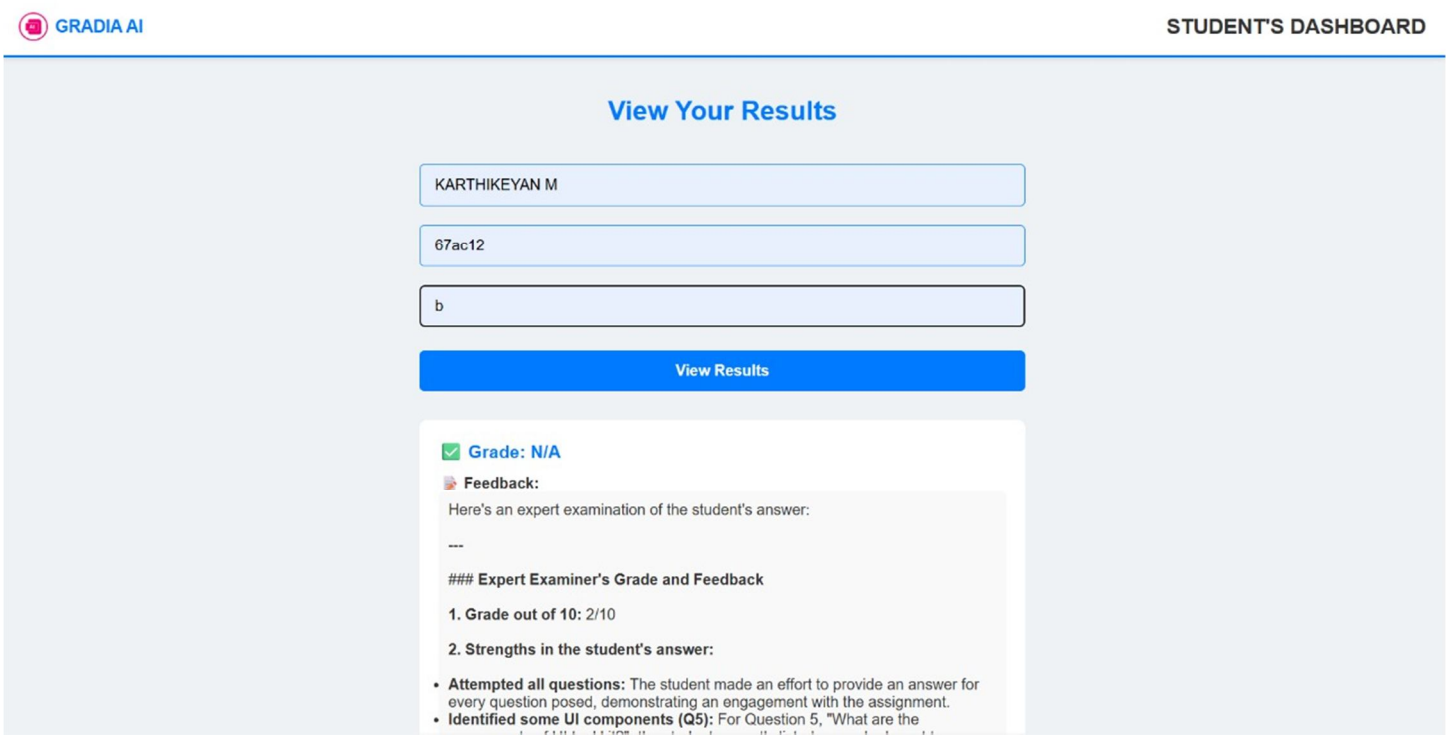


Figure 5.6: Review Result and Feedback

VII. ACKNOWLEDGEMENT

It is one of the most efficient tasks in life to choose the appropriate words to express one's gratitude to the beneficiaries. We are very much grateful to God who helped us all the way through the project and how molded us into what we are today.

We are grateful to our beloved Principal for his kind support and encouragement Dr. R. RADHAKRISHNAN, M.E., Ph.D., Adhiyamaan College of Engineering (An Autonomous Institution), Hosur for providing the opportunity to do this work in premises.

We acknowledge our heartful gratitude to Dr. G. FATHIMA, M.E., Ph.D., Professor and Head of the Department, Department of Computer Science and Engineering, Adhiyamaan College of Engineering (An Autonomous Institution), Hosur, for her guidance and valuable suggestions and encouragement throughout this project and made us to complete this project successfully.

We are highly indebted to Mr. S. VINOTH KUMAR, M.E., Supervisor, Assistant Professor, Department of Computer Science and Engineering, Adhiyamaan College of Engineering (An Autonomous Institution), Hosur, whose immense support encouragement and valuable guidance were responsible to complete the project successfully.

We also extent our thanks to Project Coordinator and all Staff Members for their support in complete this project successfully.

Finally, we would like to thank to our parents, without their motivational and support would not have been possible for us to complete this project successfully.

REFERENCES

[1] A. Verma and R. Nair, "AI-Based Evaluation System for Automated Grading and Feedback in Education," *IEEE International Conference on Artificial Intelligence and Data Engineering (AIDE)*, 2023.

[2] P. Singh and K. Patel, "Integration of Machine Learning Models for Automated Student Assessment," *International Journal of Advanced Computer Science and Applications (IJACSA)*, vol. 14, no. 3, 2024.

[3] M. Gupta and S. Rao, "Design and Implementation of an Intelligent Teacher Assistant Using Natural Language Processing," *International Conference on Computational Intelligence and Smart Systems (CISS)*, 2022.

[4] N. Sharma and L. Das, "AI-Powered Feedback Generation Using NLP Techniques," *Journal of Emerging Technologies in Education and Artificial Intelligence*, vol. 12, no. 2, 2023.

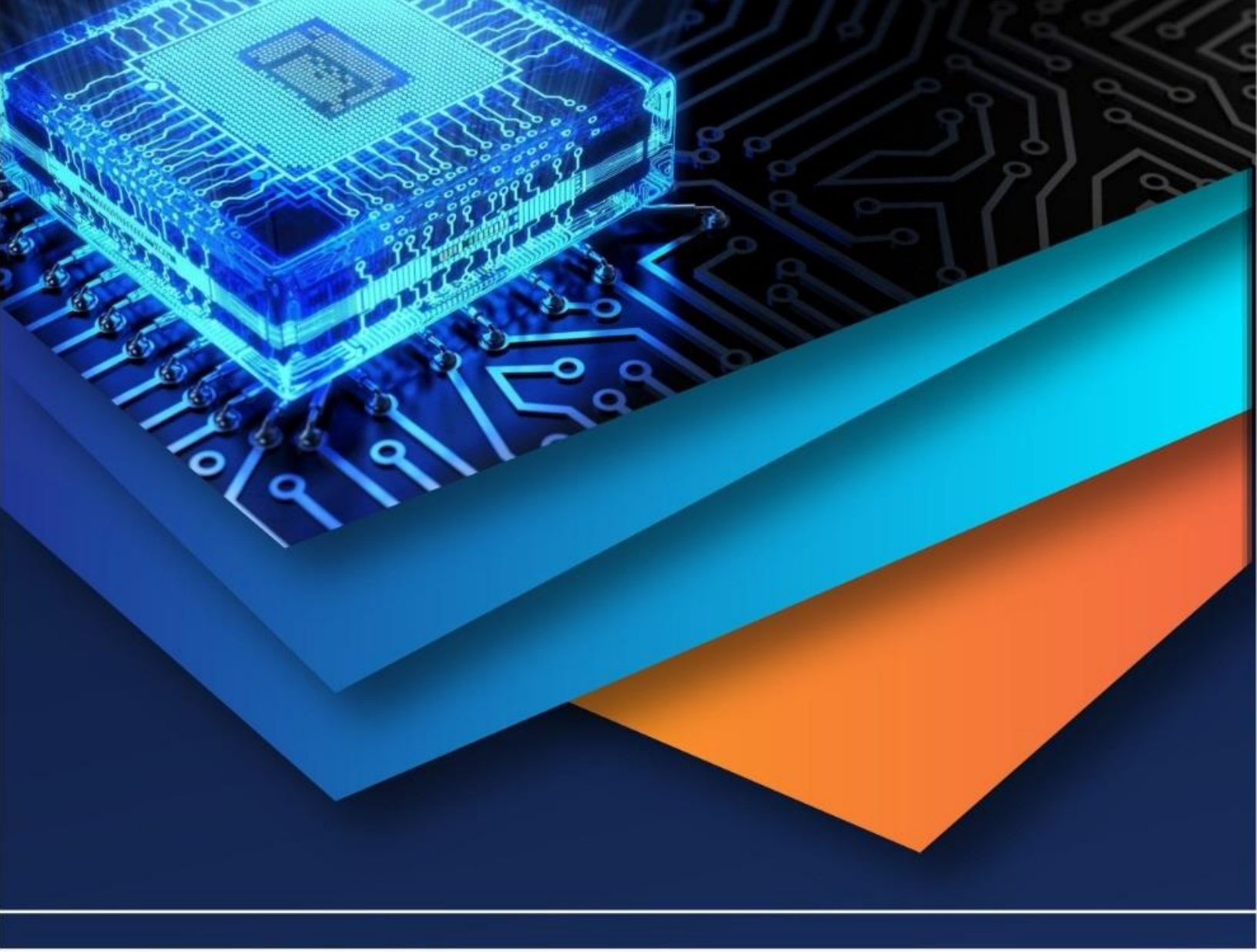
[5] R. Kumar and A. Thomas, "Automated Evaluation of Student Assignments Using Deep Learning," *IEEE International Conference on Data Science and Engineering (ICDSE)*, 2022.

[6] S. Mehta and J. George, "Intelligent Academic Assistant for Teachers Using GPT and OCR Technologies," *International Journal of Innovative Research in Computer Science and Technology (IJIRCST)*, vol. 13, no. 4, 2024.

[7] B. Reddy and M. Roy, "AI-Driven Assessment and Performance Analytics System for Educational Institutions," *International Conference on Artificial Intelligence and Education Technologies (AIET)*, 2021.

[8] D. Sharma and C. Joseph, "Deep Learning-Based Student Performance Prediction and Automated Evaluation," *Journal of Computer Applications and Educational Technologies*, vol. 11, no. 1, 2023.
[9] K. Banerjee and R. Singh, "Enhancing Educational Quality Through AI-Based Grading and Personalized Feedback Systems," IEEE International Conference on Smart Learning Environments (ICSLE), 2023.
[10] T. Iyer and P. Dutta, "A Hybrid Framework for Automating Assessment Using NLP and Computer Vision," International Journal of Artificial Intelligence Research, vol. 15, no. 2, 2024.
[11] L. Menon and A. Jacob, "Teacher Support Systems Using AI for Performance Tracking and Feedback," Proceedings of the International Conference on Intelligent Systems in Education (ISIE), 2022.
[12] V. Chandra and M. Kapoor, "AI-Based Student Evaluation Using Optical Character Recognition and Text Analytics," IEEE Conference on Computational Intelligence and Applications (ICCIA), 2023.
[13] P. Nair and D. Kumar, "Design of a Scalable AI Grading System Using Flask and PostgreSQL," International Journal of Intelligent Information Systems, vol. 10, no. 3, 2024.
[14] S. Raj and R. Pillai, "Automation of Academic Evaluation Using Deep Neural Networks," IEEE Transactions on Learning Technologies, vol. 17, no. 1, 2023.
[15] J. Thomas and K. Fernandez, "AI in Education: Automating Assessment and Feedback Delivery," International Journal of Educational Technology and AI Integration, vol. 9, no. 2, 2023.
[16] A. Ghosh and P. Varma, "Intelligent Dashboard Systems for Teachers Using Machine Learning Analytics," Proceedings of the Global Conference on Data Science and Artificial Intelligence (GCDSAI), 2024.
[17] M. Ramesh and S. Iqbal, "Leveraging BERT and GPT Models for Context-Aware Academic Feedback," Journal of Artificial Intelligence in Education Research, vol. 14, no. 1, 2024.
[18] V. Patel and L. Menon, "Integration of AI and Cloud-Based Systems for Automated Assignment Evaluation," IEEE Conference on Cloud Computing in Education (CCE), 2023.

©IJRASET: All Rights are Reserved | SJ Impact Factor 7.538 | ISRA Journal Impact Factor 7.894 |



10.22214/IJRASET

45.98

IMPACT FACTOR: 7.129

IMPACT FACTOR: 7.429

INTERNATIONAL JOURNAL FOR RESEARCH

IN APPLIED SCIENCE & ENGINEERING TECHNOLOGY

Call : 08813907089 (24*7 Support on Whatsapp)