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.2025.75711

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

AI-Driven Automated Programming Test Evaluator

Manjusha Indrajit Patil[1], Pranali Ramesh Kendale[2], Snehal Nandalal Zodage[3], Vidyarani Madhukar Hajare[4], Kalyani Umesh Sutar[5]
First-Fifth AIML, First-Fifth AITRC, Vita

Abstract: ***Exams including practical programming are a crucial component of evaluating students' coding and problem-solving capabilities. However, it might be challenging to objectively evaluate a candidate's skills because standard programming exam approaches are prone to cheating. It takes a lot of time to review programming assignments. These drawbacks emphasize the need for a new coding evaluation method that may offer a more impartial and precise evaluation of a candidate's coding abilities. By using an automated approach, the initiative seeks to revolutionize the way practical tests are administered using cutting-edge technologies. Exam scheduling, assigning distinct problem statements, automatically evaluating submissions, providing a fair evaluation, and preventing malpractice with secure exam controls are all made easier by the platform. Additionally, the system is adaptable to all pupils because it supports a variety of programming languages.***

Keywords: AI-Based Test Assessment, Secure Exam Controls, Intelligent Code Evaluation System, Outcome-Based Assessment, Student Performance Analytics.

I. INTRODUCTION

Students can concentrate on their work without worrying about technical difficulties, and teachers can upload problem statements with ease. The system improves the exam experience overall, saves time, and minimizes administrative effort. Exams are administered securely and with ease thanks to our user-friendly software. Additionally, the approach lessens the traditional assessment process's subjectivity, biases, and time-consuming nature. Practical programming exams are an essential part of assessing students' coding and problem-solving skills. However, because traditional programming exam methodologies are prone to cheating, it may be difficult to assess a candidate's skills fairly. Reviewing programming assignments takes a lot of time.

Practical programming exams are an essential part of assessing students' coding and problem-solving skills. However, because traditional programming exam methodologies are prone to cheating, it may be difficult to assess a candidate's skills fairly. Reviewing programming assignments takes a lot of time. These shortcomings highlight the need for a new coding assessment technique that would provide a more objective and accurate assessment of a candidate's coding skills.

II. OBJECTIVES

1) To make exam-related procedures including scheduling, problem statement assignment, and student submission evaluation more efficient.
2) Use tools like browser lock APIs and AI-powered proctoring to keep exams fair and stop dishonest behavior like cheating and unapproved answer sharing.
3) To ensure objective and consistent results, use AI and machine learning to automatically grade programs based on functionality, efficiency, and adherence to best practices.
4) Incorporate multi-language compilers to enable students to code in the programming languages of their choice, accommodating a range of technical skills and learning requirements.

III.PROBLEM STATEMENT

The project aims to automating job allocation and evaluation while maintaining fairness, efficiency, and integrity, the project seeks to modernize and simplify practical programming exams.

A. Problem Description

The exams including practical programming are an essential component of assessing students' coding and problem-solving skills. However, there are a number of issues with the conventional ways of administering these tests that make them less effective and equitable.

Among these is the manual distribution of problem statements, which frequently results in biases and raises the possibility that students would share or anticipate exam material. Furthermore, manual grading procedures are laborious, unreliable, and prone to human mistake, which leads to unjust assessments.

The students at educational institutions frequently employ out-of-date or inadequate tools, such as incompatible software environments or missing compilers, which forces them to use outside resources like online compilers. This raises the possibility of malpractice, such as improper use of the internet or unapproved solution sharing. Teachers are also distracted from teaching and mentoring by the administrative workload, which includes organizing tests, keeping records, and reviewing submissions.

In order to overcome these inefficiencies, guarantee consistent evaluation, and preserve the integrity of the examination process, a strong, automated, and secure system is required. By combining cutting-edge technologies and creative ideas, the proposed project aims to address these issues and establish an efficient and equitable assessment environment for teachers and students.

B. Proposed Solutions

The objective is to use AI and machine learning to automate the testing procedure and guarantee impartial evaluation. to improve practical programming tests' effectiveness, security, and fairness. By giving each student a distinct question at random, the approach automates problem distribution and lowers the likelihood of cheating. Through APIs like JDoodle and HackerRank, it also supports a variety of programming languages, enabling students to write code in the language of their choice. The system uses watermarking to stop unwanted access or sharing, restricts copy-paste functionality, and imposes full-screen mode in order to preserve exam integrity.

Using machine learning methods like Decision Trees and Support Vector Machines (SVMs), the system evaluates student code automatically and impartially based on accuracy, efficiency, and compliance with coding standards. The MERN stack—MongoDB, Express.js, React, and Node.js—is used throughout the platform's development to guarantee a scalable, effective, and user-friendly experience for both teachers and students.

IV. SYSTEM ARCHITECTURE

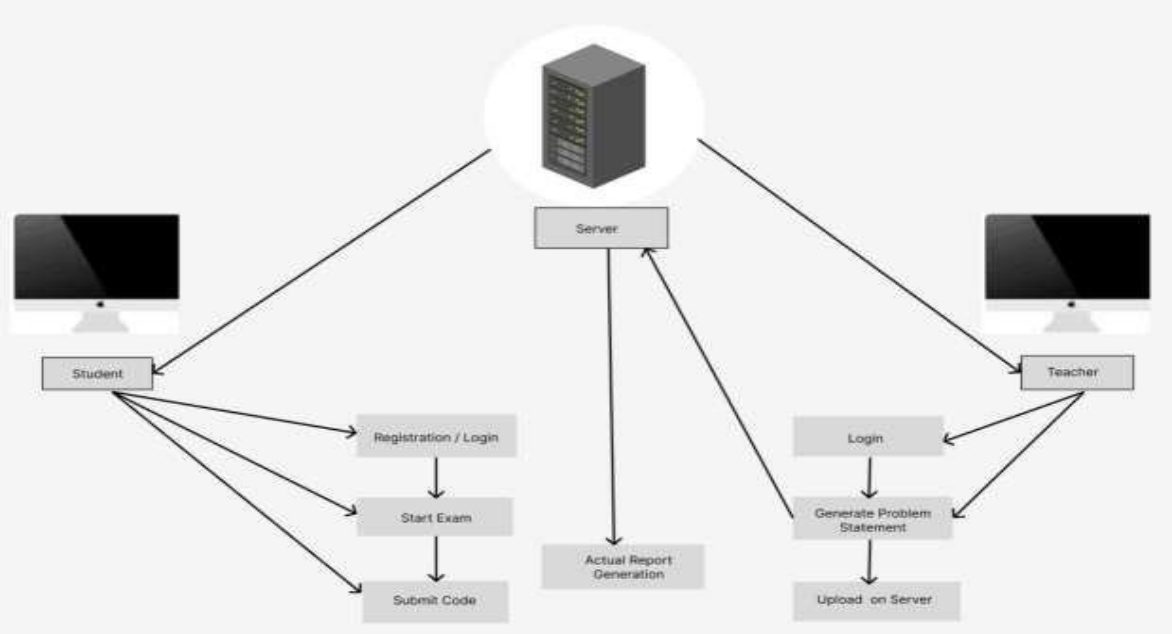


Fig. 1 System Architecture

V. METHODOLOGY

A. Defining Core Modules and Features

The first step in the project is to determine the fundamental elements of the practical examination system. Exam creation, scheduling, coding interface, issue statement display, automated evaluation, teacher and student modules, and result production are some of these.

B. Software Architecture Design

Future improvements are made possible by the system's scalability and flexibility. To preserve modularity, each module—such as the student, teacher, and problem repository—is created as a stand-alone unit.

C. Implementation and Technology Selection

Due to its scalability and efficiency, the MERN stack (MongoDB, Express.js, React.js, and Node.js) was selected. Database administration, API creation, and authentication are handled by the backend. APIs such as JDoodle and HackerRank are used to implement multi-language compiler integration .OpenCV, TensorFlow, and Scikit-Learn are used to create AI-powered evaluation models.

VI. MODULES

A. Signing up and Logging in Module

1) By entering the required information, users (teachers and students) can register on the platform through the registration module.
2) It guarantees safe login access, distinguishing between instructors and pupils and sending them to the appropriate dashboards.

B. Dashboard for Faculty

1) The faculty dashboard allows teachers to have their own profiles.
2) By inputting problem statements, a time restriction, and additional requirements, teachers can build tests.
3) Teachers are able to keep track of students' performance and finished tests.

C. Dashboard for Students

1) This module allows students to access examinations and modify their profiles.
2) Students can view the given problem statements and begin the exam.

D. Module for Exam Management

1) Students in this module will be able to view problem statements that teachers have uploaded.
2) To prevent repetition and preserve equity, each student is given a unique issue statement.
3) Within the allotted time, students are able to write and turn in their code solutions.

E. Module for Submitting and Compiling code

1) Enables students to use the platform to write, test, and submit their code.
2) Integrates with compilers to run the code and verify its functionality.

F. AI-Powered Assessment Module

1) Based on criteria like accuracy, efficiency, and compliance with best practices, this module automatically assesses submitted code.
2) It guarantees impartial and consistent grading for every student.

G. Report Generation Module

1) This module creates student performance reports based on code that students provide.
2) After that, this report will be kept on the faculty dashboard.

VII. EXPECTED OUTCOMES

1) Faster and Automated Practical tests: The system will reduce manual labor and expedite the procedure by automating it, which will enable tests to be administered more swiftly. Assigning problem statements at random ensures that no student has an edge, promoting fairness and lowering the likelihood of collaboration or cheating.
2) Improved Exam Security and Prevention of Cheating: Exam environments can be locked down with full-screen mode and copy-paste restrictions, making it difficult for students to access outside resources or plagiarize. Additionally, this security approach establishes a regulated environment, which is essential for preserving assessment integrity.

3) Automated and Fair Code Evaluation: A fair and consistent evaluation procedure can be maintained by using AI-assisted grading based on code quality (accuracy, efficiency). The efficacy and accuracy of the code can be objectively evaluated using machine learning models such as Decision Trees and Support Vector Machines (SVMs), reducing any biases or inconsistencies that may arise from human grading.

4) Support for Multiple Programming Languages: Students with varying skill sets or preferences can be accommodated by the system thanks to the flexibility in language selection (e.g., Python, Java, C++). The user experience is further improved by integrating compilers like JDoodle and HackerRank, which give students user-friendly coding platforms.

VIII. CHALLENGES

1) AI Proctoring Accuracy: It can be challenging to make sure AI-powered proctoring successfully identifies cheating without reporting false positives.
2) Scalability: It can be difficult to manage many exam takers at once without experiencing system slowdowns or breakdowns.
3) Security Issues: Preventing data breaches, hacking, and illegal access to test materials.
4) Bias in AI Evaluation: Certain coding methods may be favored by AI-based grading algorithms, or they may not accurately assess innovative solutions.
5) Internet Dependency: All students may not have access to reliable internet connections, which are necessary for online exams.

IX. CONCLUSION AND FUTURE SCOPE

A. Conclusion

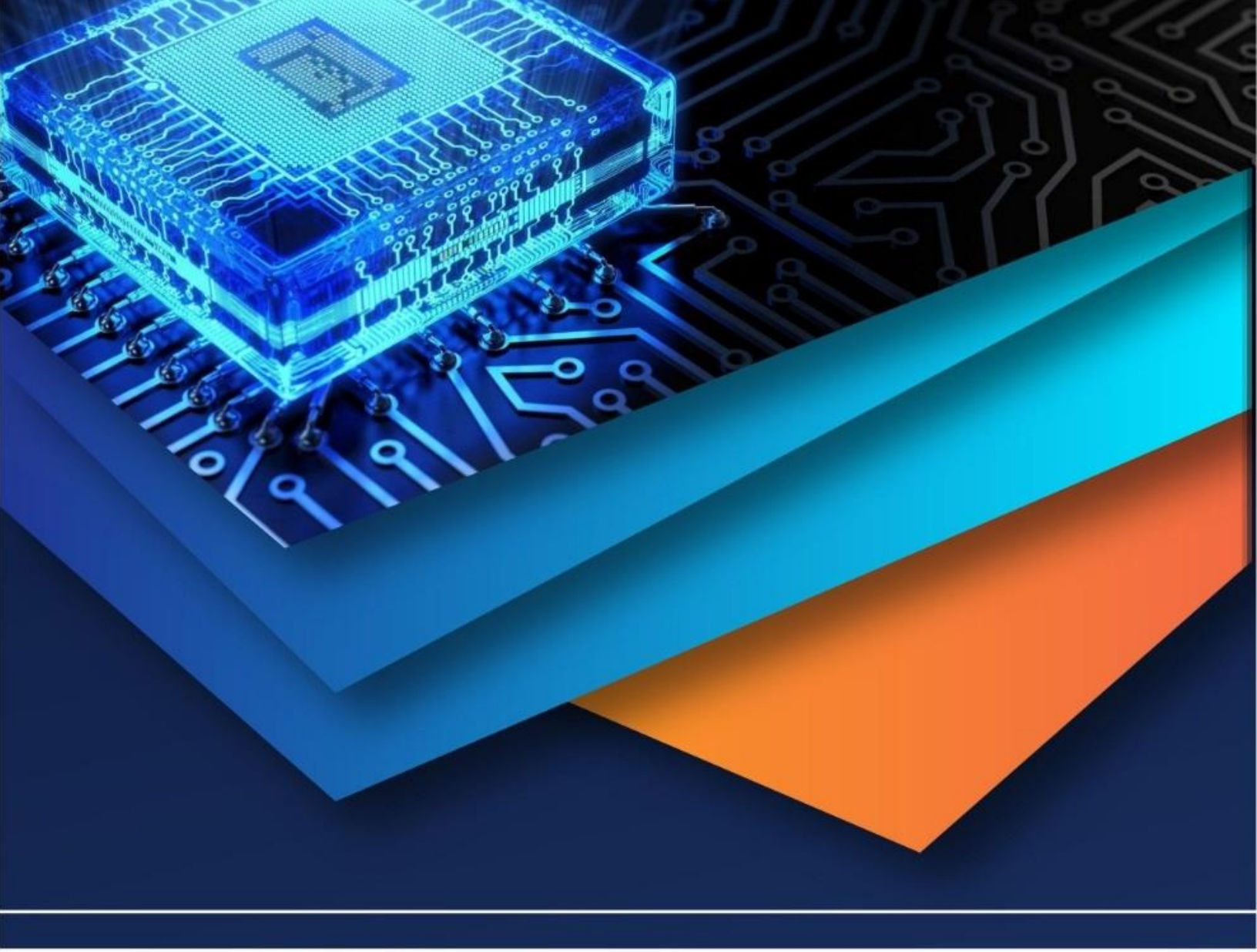
Coding tests are now fair, effective, and secure thanks to the automated programming testing system. Using AI and machine learning, it reduces cheating, does away with manual grading, and offers objective assessments. The system is adaptable for students because it supports a variety of programming languages. It lessens the strain for professors and guarantees a seamless examination procedure by automating functions like scheduling, issue assignment, and evaluation.

B. Future Scope

1) More Advanced AI Grading: Enhance the AI to more accurately assess innovative and intricate solutions.
2) Voice & Gesture Proctoring: By examining students' voices and body language, you can improve the detection of cheating.
3) Offline Mode: Create a version that students in faraway locations can use without the internet.
4) Personalized Feedback: AI can give students thorough feedback to help them get better at coding.
5) Integration with Learning Platforms: For a smooth learning process, link the system to sites like Coursera and Udemy.

REFERENCES

[1] Douce, C., Livingstone, D., & Orwell, J. (2005). Automatic Test-Based Assessment of Programming: A Review. Journal on Educational Resources in Computing, 5(3), Article,4
[2] Cotroneo, D., Foggia, A., Improta, C., Liguori, P., & Natella, R. (2024). Automating the Correctness Assessment of AI-Generated Code for Security Contexts. Preprint submitted to Journal of Systems and Software, November 6, 2024.
[3] Mekterović, I., Brkić, L., Milašinović, B., & Baranović, M. (2020). Building a comprehensive automated programming assessment system. IEEE Access, 8.
[4] Saikkonen, R., Malmi, L., & Korhonen, A. (2001). Fully Automatic Assessment of Programming Exercises. Proceedings of the 6th Annual Conference on Innovation and Technology in Computer Science Education (ITiCSE), Canterbury, UK.
[5] Liu, D., Qu, X., Dong, J., Zhou, P., Xu, Z., Wang, H., Di, X., Lu, W., & Cheng, Y. (2023). Transform-Equivariant Consistency Learning for Temporal Sentence Grounding. Journal of the ACM, 37(4), Article 111.



10.22214/IJRASET

45.98

IMPACT FACTOR: 7.129

IMPACT FACTOR: 7.429

INTERNATIONAL JOURNAL FOR RESEARCH

IN APPLIED SCIENCE & ENGINEERING TECHNOLOGY