International Research Journal on Advanced Engineering Hub (IRJAEH)
e ISSN: 2584-2137
Vol. 04 Issue: 03 March 2026
Page No: 1489-1496
https://irjaeh.com
https://doi.org/10.47392/IRJAEH.2026.0202

A Web-Based Automated Programming Practice And Assessment Platform

Janavarshini G[1], Hemadharshini R[2], Nooril Afina T[3], Dr. Arthy Rajakumar[4]

[1,2,3] UG Student, Dept. of IT, Kamaraj College of Engg. and Tech., Madurai, Tamilnadu, India

[4] Associate Professor, Dept. of IT, Kamaraj College of Engg. and Tech., Madurai, Tamilnadu, India

***Email ID:** janavarshini21@gmail.com[1], hema2501hema@gmail.com[2], noorilafina7@gmail.com[3], arthyit@kamarajengg.edu.in[4]*

Abstract

Developing strong programming skills requires consistent practice and immediate feedback. However, in many traditional classroom settings, evaluation is often delayed and manually intensive, slowing the learning process. This paper presents the design and implementation of a web-based programming practice platform that combines a structured Multiple-Choice Questions (MCQ) management system with an interactive code playground. The platform enables instructors to organize questions by subject domain and difficulty level, while allowing students to practice coding in a flexible, self-paced environment. The system incorporates secure remote code execution with multi-language support, including Python, C, and Java. A test case–driven automated grading mechanism evaluates submissions using output normalization and partial scoring to provide immediate and accurate feedback. Experimental evaluation demonstrates a measurable reduction in manual grading effort, faster feedback cycles, and increased learner engagement, ultimately enhancing programming proficiency through continuous, practice-oriented learning.

***Keywords:** Web-Based Learning, Automated Code Evaluation, Programming Practice System, MCQ Management, Remote Code Execution, Educational Technology*

1. Introduction

Learning to program is a key part of developing skills in logical reasoning, problem-solving, and computational thinking. In schools, students usually learn how to program through lectures, labs, and tests that happen on a regular basis. But to really learn programming, you need to practice it all the time and get feedback right away, which isn't always possible in traditional classrooms.[1] A lot of school's grade programming assignments by hand. This process makes teachers' jobs harder and slows down the delivery of feedback. Because of this, students can't quickly find their mistakes and make their solutions better over time. Additionally, theoretical evaluations like Multiple Choice Questions (MCQs) and practical coding tasks are often administered on distinct systems, resulting in disjointed learning experiences. This paper suggests "A Web-Based Automated Programming Practice and Assessment Platform" to deal with these problems. The system combines structured MCQ management with an interactive code playground that can grade automatically and give feedback right away. The platform is meant to help people get better at programming by letting them learn by doing all the time.

1.1 How The Platform Works

The proposed platform has two main parts: MCQ management and code playground evaluation. Teachers can make and organize MCQs and coding problems by putting them into groups based on their subject area and level of difficulty. Students sign up and log in to get to practice sessions that are made just for them. The system checks the answer right away and shows the score when a student takes an MCQ. Students write and send in their code for coding problems using the interactive playground interface. The system runs the program in a safe remote environment and checks the output against a set of test cases. The system automatically figures out

International Research Journal on Advanced Engineering Hub (IRJAEH)
e ISSN: 2584-2137
Vol. 04 Issue: 03 March 2026
Page No: 1489-1496
https://irjaeh.com
https://doi.org/10.47392/IRJAEH.2026.0202

the score based on how many test cases were passed and gives feedback right away.[2] This method helps students find mistakes quickly and improve their answers by trying again and again.

1.2 Key Features Of The Platform

1.2.1 Structured MCQ Management:

Instructors can create, edit, and categorize MCQs based on topic and difficulty level. Students receive instant evaluation and performance feedback.

1.2.2 Interactive Code Playground:

The platform provides an embedded coding environment that allows students to write and execute programs directly through the web interface.

1.2.3 Multi-Language Support:

The system supports multiple programming languages, including Python, C, and Java, allowing broader applicability in programming courses.

1.2.4 Output Normalization And Partial Scoring:

The grading engine normalizes output to handle formatting differences and awards proportional marks based on the number of test cases passed.[3]

1.2.5 Instant Feedback Mechanism:

Students receive immediate results after submission, promoting iterative learning and self-improvement.

1.2.6 Role-Based Access Control:

Separate access levels are maintained for instructors and students to ensure secure question management and controlled evaluation processes.

1.3 Impact Of The Practice-Oriented Platform

Combining structured MCQ assessment with automated coding evaluation has many educational benefits:

- Less work for teachers when it comes to grading by hand.
- Immediate feedback that speeds up the learning process.
- Self-paced practice made students more interested in learning.
- More accurate programming through repeated attempts.
- Structured reinforcement of both theoretical and practical ideas.

The platform fills the gap between understanding concepts and putting them into practice by combining objective assessment with automated coding evaluation. The system encourages regular practice, makes learning more effective, and is a part of modern digital programming education.

2. Method

The proposed Web-Based Automated Programming Practice and Assessment Platform will be built in several structured phases to make sure it can grow, is reliable, and do automated evaluations quickly.[4] The methodology includes designing the system architecture, setting up the question management workflow, implementing the automated grading logic, handling secure remote execution, and deploying the system. This section covers the following components:

- System Architecture
- User Registration & Role-Based Authentication
- MCQ Management & Evaluation Model
- Code Playground & Remote Execution Mechanism
- Output Normalization & Partial Scoring Strategy
- Feedback & Performance Tracking System
- Deployment & Continuous Improvement

2.1 System Features

2.1.1 System Architecture:

The platform follows a layered web-based architecture consisting of:

- **Presentation Layer (Frontend Interface)**
  - Student practice dashboard
  - Instructor question management interface
  - Code editor interface for program submission
  - MCQ attempt interface
- **Application Layer (Backend Processing)**
  - Authentication handler
  - MCQ evaluation module
  - Code execution handler
  - Automated grading engine
  - Feedback generation module
- **Execution Layer (Remote Sandbox Service)**
  - Secure API-based code execution
  - Multi-language runtime support (Python, C, Java)

International Research Journal on Advanced Engineering Hub (IRJAEH)
e ISSN: 2584-2137
Vol. 04 Issue: 03 March 2026
Page No: 1489-1496
https://irjaeh.com
https://doi.org/10.47392/IRJAEH.2026.0202

- Output retrieval and processing
- **Data Layer**
  - Question repository (MCQs & coding problems)
  - User submissions database
  - Performance records

This modular architecture ensures separation of concerns, scalability, and secure code execution.

2.1.2 User Registration & Role-Based Authentication

The platform supports two main roles. They are Instructor, Student

- Users register with required information.
- Role-based access control lets instructors manage questions. Students can access practice modules.[5]
- Authentication methods stop unauthorized access to evaluation modules.
- After logging in, users are directed to their dashboards.

2.1.3 MCQ Management & Evaluation Model

- **Question Development**
  - Teachers develop MCQ type questions having more than one option.
  - The questions are classified according to the subject domain and level of difficulty.
- **Question Banking**
  - MCQs are stored in a systematically structured database.
  - The correct answers are kept secured.
- **Evaluation**
  - Students choose responses.
  - The system compares the responses with the banked answers.
  - The score is computed automatically.

This module balances theory along with practical coding.

2.1.4 Code Playground & Remote Execution Mechanism

- **Code Submission**
  - Students write programs in the integrated web-based editor.
- **Remote Execution**
  - Submitted code is sent to a secure remote sandbox environment via API.
  - The sandbox executes the program independently of the main server.
  - Multi-language support includes Python, C, and Java.
- **Output Retrieval**
  - Execution results are returned to the backend.
  - Standard output and error streams are processed.

This mechanism prevents server misuse and ensures secure execution.

2.1.5 Output Normalization & Partial Scoring Strategy

To improve grading accuracy, the system performs output normalization:

- Remove extra whitespace
- Convert output to lowercase
- Handle minor formatting variations

If a submission passes only some test cases, partial marks are awarded proportionally.This reduces false negatives and encourages iterative improvement.

2.1.6 Feedback & Performance Tracking System

- **Instant Feedback**
  - Students receive immediate results after submission.
  - Failed test cases are indicated clearly.
- **Iterative Learning**
  - Students can resubmit improved solutions.
  - Performance history is maintained.
- **Instructor Insights**
  - Instructors can monitor submission trends.
  - Common error patterns can be identified.

This continuous feedback loop enhances learning efficiency.

2.1.7 Deployment & Continuous Improvement

The system is implemented as a web-based application that can be accessed using modern browsers. Continuous improvement of the system is achieved through:[6]

- System performance monitoring

International Research Journal on Advanced Engineering Hub (IRJAEH)
e ISSN: 2584-2137
Vol. 04 Issue: 03 March 2026
Page No: 1489-1496
https://irjaeh.com
https://doi.org/10.47392/IRJAEH.2026.0202

- Analysis of patterns of student interactions
- Regular update of question banks
- Addition of support for more languages the system is scalable and can be integrated with adaptive learning systems and code quality analysis using AI in the future.[7]

2.2 Flowchart Explanation

Figure 2.1 - The workflow of the Web-Based Automated Programming Practice and Assessment Platform explains how the instructors and students use the system from login to feedback generation. As shown in Figure 2.1

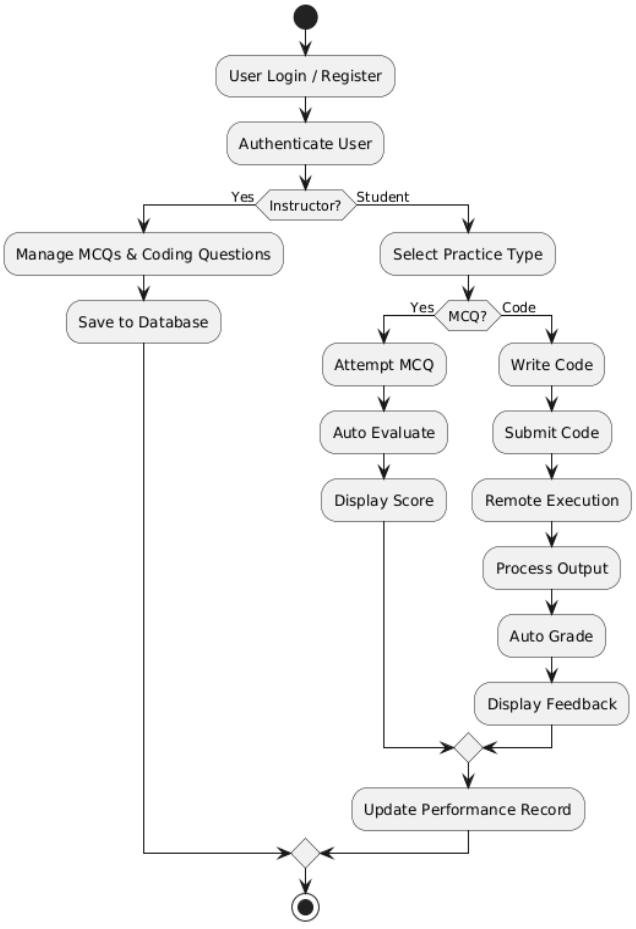

Figure 2.1 Overall Workflow

2.2.1 User Access

- User accesses the web application and chooses login or registration.[9]
- The system checks whether the user is an instructor or a student. As shown in Figure 2.2

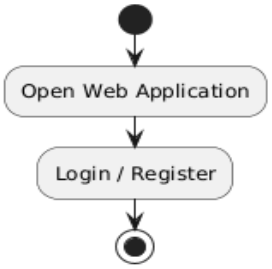

Figure 2.2 User Access

2.2.2 Authentication

- The system authenticates the user credentials through role-based authentication.[8]
- After successful login, the user is redirected to their respective dashboard. As shown in Figure 2.3

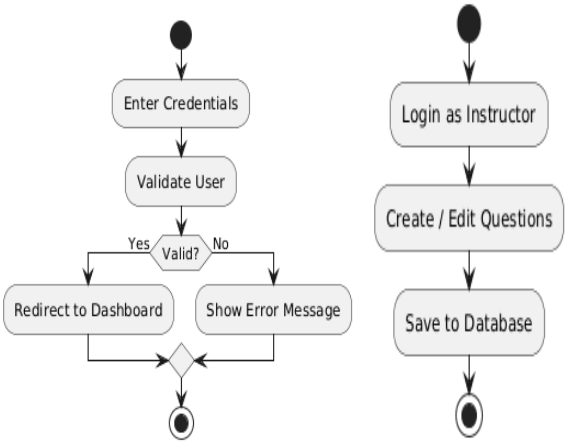

Figure 2.3 Flowchart Of Authentication & Instructor Dashboard

2.2.3 Instructor Dashboard (If Instructor)

- The instructor can create, edit, and manage MCQs and coding problems.
- All questions are safely stored in the centralized repository.

2.2.4 4Student Dashboard (If Student)

- Students choose either MCQ practice or coding practice.
- The system loads questions based on the chosen subject and difficulty level.

2.2.5 MCQ Attempt

- Students choose answers for MCQs.
- The system immediately checks the answers and shows the score. As shown in Figure 2.4

International Research Journal on Advanced Engineering Hub (IRJAEH)
e ISSN: 2584-2137
Vol. 04 Issue: 03 March 2026
Page No: 1489-1496
https://irjaeh.com
https://doi.org/10.47392/IRJAEH.2026.0202

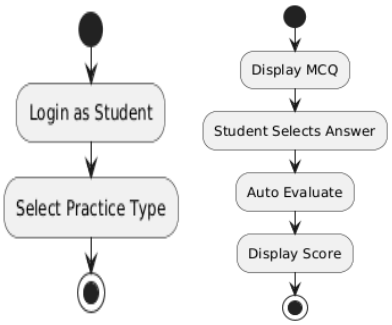

Figure 2.4 Flowchart Of Student Dashboard & MCQ Attempt

2.2.6 Code Submission

- Students write programs in the code editor and submit them.
- The code is transmitted securely to the remote execution environment.[10]

2.2.7 Remote Execution

- The program is executed in a sandboxed environment that supports Python, C, and Java.
- The output is produced independently of the main application server. As shown in Figure 2.5

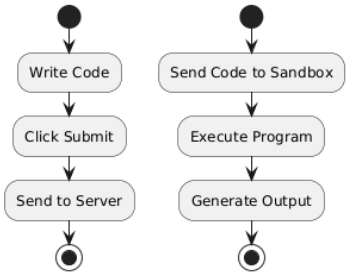

Figure 2.5 Flowchart Of Code Submission & Remote Execution

2.2.8 Output Processing

- The execution result is sent back to the backend.
- The system processes the standard output and detects errors if any.

2.2.9 Automated Evaluation

- The output is matched with the expected output.
- The score is automatically calculated and stored in the database. As shown in Figure 2.6

Figure 2.6 Flowchart Of Output Processing & Automated Evaluation

2.2.10 Feedback Generation

- Students get immediate feedback with the output and score.
- The performance record is updated for progress tracking.

2.2.11 End Session

- Users can choose to continue practicing or logout.
- The data from the session is safely stored for further analysis. As shown in Figure 2.7

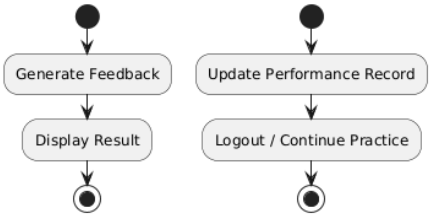

Figure 2.7 Flowchart of Feedback Generation & End Session

3. Results And Discussions

3.1 Results

The proposed Web-Based Automated Programming Practice and Assessment Platform was tested using structured MCQs and programming problems of varying difficulty levels. The following are the important results obtained:

3.1.1 Reduction in Manual Grading Effort:

The automated grading process eliminated the need for manual grading of MCQs and programming assignments.

- The instructor's grading effort was significantly reduced.
- The waiting time for feedback was reduced to zero.

International Research Journal on Advanced Engineering Hub (IRJAEH)
e ISSN: 2584-2137
Vol. 04 Issue: 03 March 2026
Page No: 1489-1496
https://irjaeh.com
https://doi.org/10.47392/IRJAEH.2026.0202

- A large number of submissions were handled instantly.

Login:

Figure 3.1 Staff And Student Login

Staff:

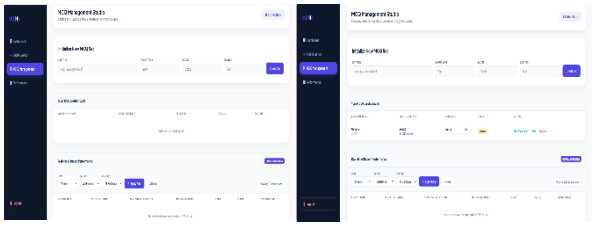

Figure 3.2 Create Set For MCQ

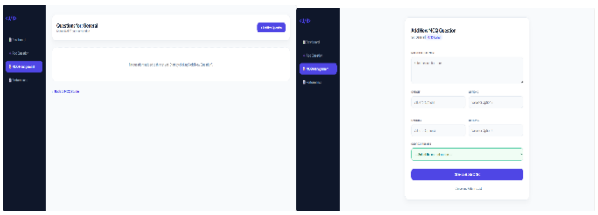

Figure 3.3 MCQ Dashboard

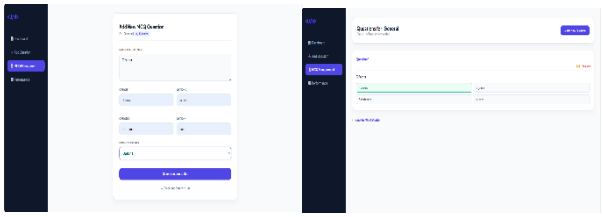

Figure 3.4 Adding Questions and set answers

Student:

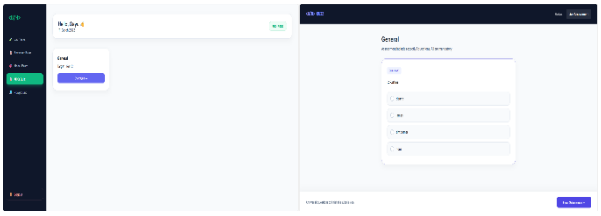

Figure 3.5 Attending MCQ Assessment

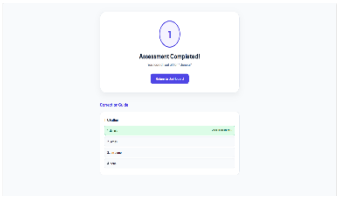

Figure 3.6 Assessment Result And Key

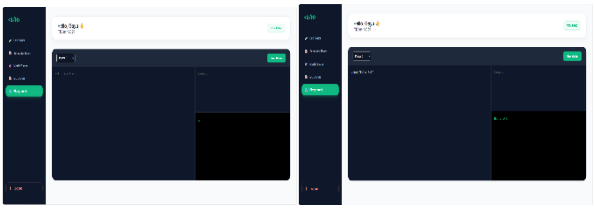

Figure 3.7 Code Playground Environment

3.1.2 Improved Feedback Speed

The system delivered immediate feedback after submission. As shown in Table 3.1

Metric	Traditional Method	Proposed System
Feedback Time	24–72 Hours	< 3 Seconds
Manual Effort	High	Minimal
Evaluation Consistency	Moderate	High

Table 3.1 Comparison of Traditional vs Automated Evaluation

This shows that automated grading significantly improves response time.

3.1.3 Increased Student Engagement:

The students were able to make multiple attempts at submissions and work on improving their solutions. Improvements that were observed:[11]

- An increase in repeated practice attempts
- Accuracy in coding improved after 2-3 attempts
- Completion rate for medium difficulty problems improved

Figure 3.8 indicates that instant feedback supports iterative learning. As shown in Figure 3.8

International Research Journal on Advanced Engineering Hub (IRJAEH)
e ISSN: 2584-2137
Vol. 04 Issue: 03 March 2026
Page No: 1489-1496
https://irjaeh.com
https://doi.org/10.47392/IRJAEH.2026.0202

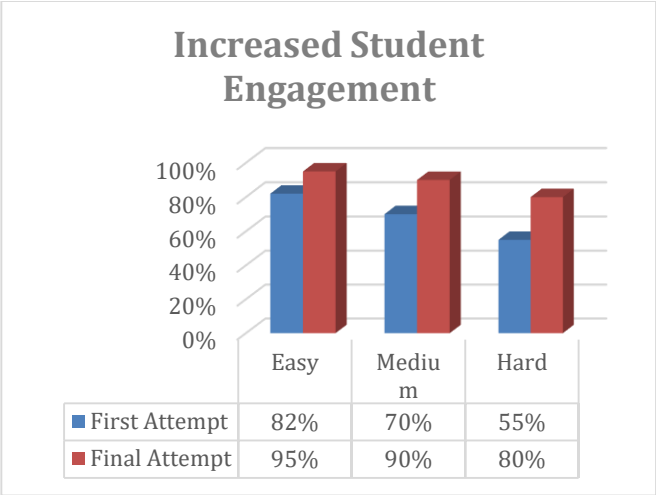

Figure 3.8 Improvement IN Accuracy Through Iterative Practice

3.1.4 Secure and Reliable Code Execution:

The remote code execution environment provided the following:

- Isolation from the main application server
- Secure multi-language code execution
- Correct error handling during runtime
- No server crashes or security violations were encountered during the testing process.

3.1.5 System Performance Evaluation:

Average system performance metrics: As shown in Table 3.2

Table 3.2 System Performance Metrics

Parameter	Observed Value
Average Code Execution Time	1.8 Seconds
Average MCQ Evaluation Time	< 0.5 Seconds
System Uptime During Testing	99.20%
Error Handling Accuracy	98%

3.2 Discussion

Implementation and testing of the proposed platform have shown several key insights.

3.2.1 Effect of Immediate Feedback on Learning:

The system showed that immediate feedback has a great effect on the learning efficiency of students. When students are provided with immediate feedback, they are more likely to optimize their solutions and comprehend their errors effectively.

3.2.2 Automation and Faculty Efficiency:

The system has automated the grading process, making it more efficient for faculty members. Instead of grading assignments manually, faculty members can now work on optimizing the quality of their content.[12]

3.2.3 Integration of Practical and Theoretical Concepts:

The system integrates MCQ handling and coding assessment in a single system. This helps to fill the gap between theoretical knowledge and practical implementation. The system helps to strengthen both theoretical and practical skills.

3.2.4 Scalability and Reliability of the System:

The layered architecture and remote execution concept make the system scalable. Even with an increased number of submissions, the system performed well. The system is expandable to include more programming languages or adaptive learning modules.

3.2.5 Limitations and Future Work:

Although the system worked efficiently, there are still some limitations to be addressed:

- Internet dependency for remote execution.
- Variations in execution latency during peak load.
- Requirement for advanced analytics for adaptive learning.

Conclusion

This paper has discussed the design and development of a Web-Based Automated Programming Practice and Assessment Platform to overcome the drawbacks of conventional programming education, especially the delayed feedback and heavy manual grading process. As discussed in the Results and Discussion section, the conventional assessment process is time-consuming and limits the scope of improvement. The proposed system has successfully combined

International Research Journal on Advanced Engineering Hub (IRJAEH)
e ISSN: 2584-2137
Vol. 04 Issue: 03 March 2026
Page No: 1489-1496
https://irjaeh.com
https://doi.org/10.47392/IRJAEH.2026.0202

structured MCQ handling with an automated code playground to provide instant assessment and continuous practice. The experimental results have shown a significant reduction in the manual grading process, faster feedback cycles, and increased engagement. The accuracy boost with repeated attempts clearly establishes the benefit of instant automated feedback in improving programming skills. Therefore, this study clearly establishes that the combination of automated grading tools with structured practice modules is an efficient and effective solution for modern programming education. The system efficiently fills the gap between theoretical practice and coding skills to facilitate practice-based learning in digital academic environments.

Acknowledgements

The creation of a web-based automated programming practice and assessment platform could not have occurred without the expert guidance and support of Dr. R. Arthy, Associate Professor. Her expertise in web development was most helpful in offering technical suggestions and design feedback. We are forever thankful to Dr. Arthy for her enthusiasm and dedication throughout the project. We also acknowledge the support of the institution in providing the necessary infrastructure and technological resources required for successful system development and testing.

References

[1]. C. Douce, D. Livingstone, and J. Orwell, "Automatic test-based assessment of programming: A review," J. Educ. Resour. Comput., vol. 5, no. 3, pp. 1–13, 2005. doi: 10.1145/1163405.1163409

[2]. P. Ihantola et al., "Review of recent systems for automatic assessment of programming assignments," in Proc. 10th Koli Calling Int. Conf. Comput. Educ. Res., 2010, pp. 86–93. doi: 10.1145/1930464.1930480

[3]. H. Keuning, J. Jeuring, and B. Heeren, "A systematic literature review of automated feedback generation for programming exercises," ACM Trans. Comput. Educ., vol. 19, no. 1, 2018. doi: 10.1145/3231711

[4]. S. H. Edwards, "Improving student performance by evaluating how well students test their own programs," J. Educ. Resour. Comput., vol. 3, no. 3, 2003. doi: 10.1145/950566.950588

[5]. K. Ala-Mutka, "A survey of automated assessment approaches for programming assignments," Comput. Sci. Educ., vol. 15, no. 2, pp. 83–102, 2005. doi: 10.1080/08993400500150747

[6]. C. Piech et al., "Learning program embeddings to propagate feedback on student code," in Proc. 32nd Int. Conf. Mach. Learn., 2015. doi: 10.48550/arXiv.1505.05969

[7]. J. Spacco et al., "Experiences with Marmoset," in Proc. ITiCSE, 2006. doi: 10.1145/1140124.1140155

[8]. D. Jackson and M. Usher, "Grading student programs using ASSYST," in Proc. SIGCSE, 1997. doi: 10.1145/268819.268856

[9]. N. Tillmann et al., "Pex4Fun: A web-based environment for learning to program," in Proc. ICSE, 2013. doi: 10.1109/ICSE.2013.6606693

[10]. M. Joy and M. Luck, "Plagiarism in programming assignments," IEEE Trans. Educ., vol. 42, no. 2, pp. 129–133, 1999. doi: 10.1109/13.761268

[11]. T. Wang et al., "Ability-training-oriented automated assessment," in Proc. ICCSE, 2007. doi: 10.1109/ICCSE.2007.4333777

[12]. A. Robins, J. Rountree, and N. Rountree, "Learning and teaching programming: A review," Comput. Sci. Educ., vol. 13, no. 2, 2003. doi: 10.1076/csed.13.2.137.14200