WWW.IJRPR.COM

International Journal of Research Publication and Reviews

Journal homepage: www.ijrpr.com ISSN 2582-7421

CodeAssess AI: An Automated Java Assessment Engine for Student Programming Practice

Mrs. Deepthi Nair P, Sri Kartheeka M, SriChand U S, SriiSharan S, Srimathi G

[a][Computer Science and Engineering, Sri Shakthi Institute of Engineering and Technology, Coimbatore and 641062, India]

A B S T R A C T :

CodeAssess AI is a web-based application designed to simplify Java programming practice and automated assessment for students. It integrates a programming question bank, Monaco code editor, Java compilation and execution, and backend-managed test-case evaluation. The frontend uses React and TypeScript, while the backend is developed using Python and FastAPI. The current question bank contains 100 question records. Project verification reported successful frontend validation and one accepted Java submission that passed three out of three test cases. However, comprehensive testing, secure execution isolation, and live AI feedback require further verification. The system provides a foundation for improving programming practice through automated evaluation and future AI-assisted feedback.

Keywords: Java Assessment, Automated Evaluation, React, FastAPI, Monaco Editor, AI-Assisted Feedback

Introduction

Programming education requires regular practice and reliable evaluation. Manual code checking and output verification can be time-consuming, particularly when many students submit programming assignments.

CodeAssess AI provides a unified web environment for accessing Java programming questions, writing code, executing programs, and evaluating outputs against predefined test cases. React and TypeScript support the frontend, while Python and FastAPI manage backend operations. The system aims to simplify programming practice and establish a foundation for automated assessment and future AI-assisted feedback.

Literature survey

Douce et al. reviewed automatic test-based programming assessment and its educational applications [1]. Combéfis examined automated code assessment techniques and tools [2]. Messer et al. reviewed automated grading and feedback systems for programming education [3].

Recent research has also explored generative AI for programming assessment and feedback [4], [5]. These approaches may provide explanatory guidance but require validation to ensure correctness.

The reviewed work highlights the importance of combining reliable test-based evaluation with useful feedback. CodeAssess AI addresses this objective through an integrated coding interface, backend-managed questions, and Java assessment workflow.

Methodology

The system follows a frontend–backend architecture. React and TypeScript provide the user interface, while Monaco Editor enables Java code editing. FastAPI exposes endpoints for retrieving questions and processing supported execution and submission requests (Fig. 1).

Programming questions are stored in a JSON-based question bank containing 100 records. The backend coordinates Java compilation and execution using the JDK and compares program outputs against backend-managed test cases.

The assessment workflow consists of:

1. Retrieving a programming question.
2. Writing Java source code.
3. Sending the request to the backend.
4. Validating the request and executing the program.
5. Comparing actual and expected outputs.
6. Returning the assessment result to the frontend.

The project also contains a Gemini integration intended for AI-assisted feedback, but successful live responses require further verification.

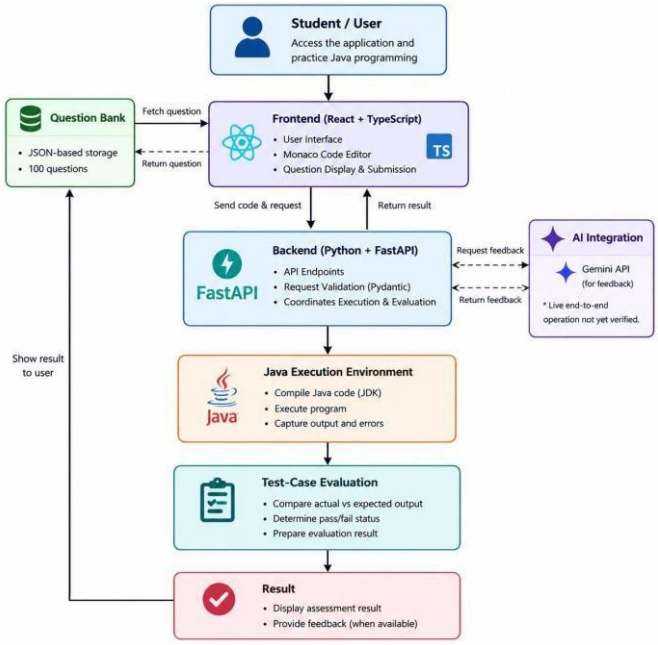


Fig. 1 - System architecture and workflow of CodeAssess AI.

Existing system

Conventional programming assessment uses manual evaluation, local compilers, or online coding platforms. Manual assessment can require considerable time, while separate local tools may require students to manage questions, execution, and output checking independently. The disadvantages of such systems are:

- Manual evaluation can be time-consuming.
- Limited test cases may fail to reveal certain errors.
- Students may need multiple tools for coding and assessment.
- Error explanations may be limited.
- Consistent assessment requires well-defined test cases.
- Untrusted code execution requires appropriate security controls.

Proposed system

CodeAssess AI integrates question retrieval, code editing, Java execution, and test-based assessment within one web application.

5.1. Main modules

- *Question Management:* Retrieves programming questions from the JSON question bank.
- *Code Editor:* Uses Monaco Editor to write and modify Java programs.
- *API Communication:* Connects the frontend with backend services.
- *Java Execution:* Coordinates compilation and execution through the JDK.
- *Assessment:* Compares program outputs against backend-managed test cases.
- *AI Integration:* Provides a foundation for explanatory feedback, subject to successful integration testing.

The modular architecture simplifies maintenance and supports future improvements in security, scoring, and feedback.

System requirements

The software requirements of the system are:

- React and TypeScript
- Vite and React Router
- Monaco Editor

- Python and FastAPI
- Pydantic and Uvicorn
- Java Development Kit
- JSON-based question bank
- Gemini API integration

A computer capable of running the frontend, backend, and Java development tools is required. Figs. 2 and 3 show the home page and the practice interface, and Figs. 4 and 5 show the assessment result screen of the application.

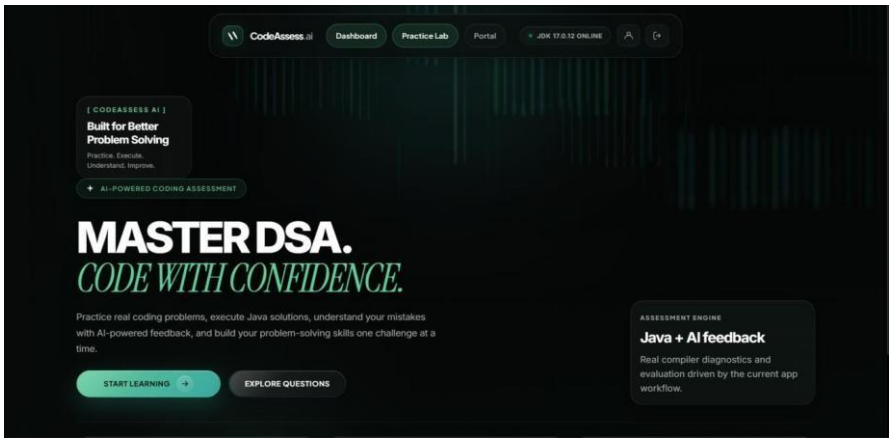


Fig. 2 - Home page of CodeAssess AI.

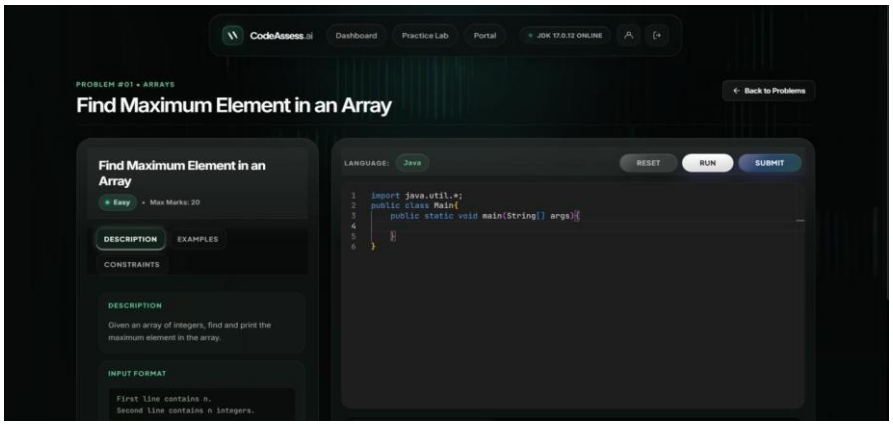


Fig. 3 - Practice interface with problem description and Monaco code editor.

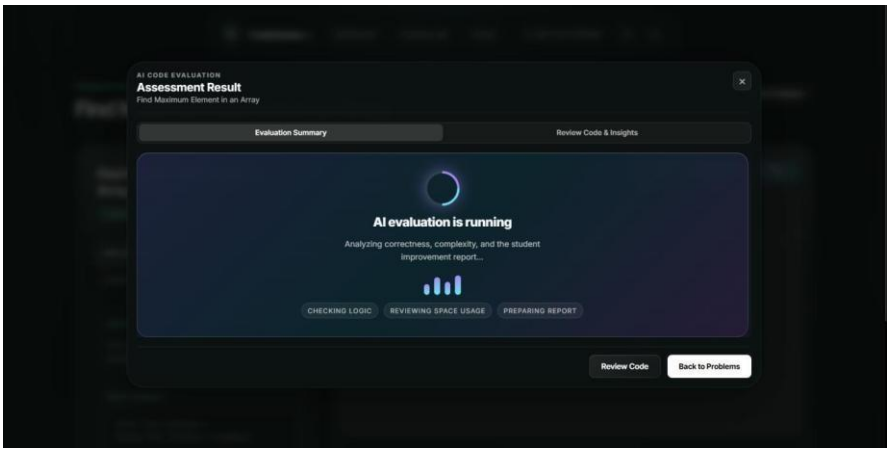


Fig. 4 - AI code evaluation in progress.

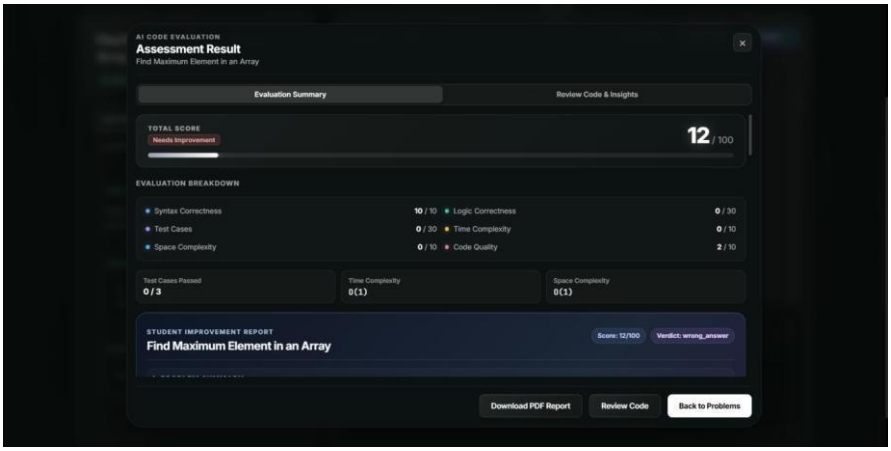


Fig. 5 - Assessment result with evaluation summary and score breakdown.

Results and discussion

The available project audit reported successful frontend build, linting, and TypeScript validation. The application displayed 100 programming question records.

One Java submission integration test was accepted after passing three out of three applicable test cases. This confirms one successful test instance but does not establish overall assessment accuracy.

The latest reported backend test suite had 12 passing tests and one failure caused by an outdated expectation of eight questions instead of the current 100. Further testing is required to resolve this issue and validate error handling, execution security, and assessment reliability.

Conclusion and future enhancements

CodeAssess AI integrates a web-based Java editor, centralized question bank, backend APIs, and test-case-based evaluation to support student programming practice. Initial verification demonstrates that key components have been implemented and exercised.

Future enhancements include secure sandboxed execution, comprehensive testing, partial-marking rules, database-backed submission history, authentication, and validated AI-generated feedback. Additional programming-language support may be introduced after separate execution workflows are implemented and tested.

The project provides a foundation for automated Java assessment, while further validation is necessary before making broader claims about reliability or educational effectiveness.

REFERENCES

[1] C. Douce, D. Livingstone, and J. Orwell, "Automatic Test-Based Assessment of Programming: A Review," *Journal on Educational Resources in Computing*, vol. 5, no. 3, 2005.

[2] S. Combéfis, "Automated Code Assessment for Education: Review, Classification and Perspectives on Techniques and Tools," *Software*, vol. 1, no. 1, pp. 3–30, 2022.

[3] M. Messer, N. C. C. Brown, M. Kölling, and M. Shi, "Automated Grading and Feedback Tools for Programming Education: A Systematic Review," *ACM Transactions on Computing Education*, vol. 24, no. 1, 2024.

[4] "The Future of Grading Programming Assignments in Education: The Role of ChatGPT in Automating the Assessment and Feedback Process," *Thinking Skills and Creativity*, vol. 52, 2024.

[5] Y. Qian, M. Liu, and L. Zhu, "Exploring the Quality and Effectiveness of AI-Generated Feedback in Introductory Programming," *Humanities and Social Sciences Communications*, vol. 13, 2026.

[6] React Documentation. https://react.dev/

[7] FastAPI Documentation. https://fastapi.tiangolo.com/

[8] Microsoft Monaco Editor. https://microsoft.github.io/monaco-editor/

[9] Oracle Java Documentation. https://docs.oracle.com/en/java/

[10] Google AI for Developers, Gemini API Documentation. https://ai.google.dev/gemini-api/docs/