Journal of Educational Studies in
Mathematics and Computer Science

Interactive Web-Based Coding Environment with Monitoring and Student Progress Tracking Features

Marina Svičević [1, a, *], Nemanja Vučićević [1, b], Branislav Bogdanović [1, c]
[1]University of Kragujevac, Faculty of Science, Kragujevac, Serbia
[a]marina.svicevic@pmf.kg.ac.rs, [b]nemanja.vucicevic@pmf.kg.ac.rs, [c]branislav.bogdanovic95@gmail.com

Received: 12 May 2025; **Accepted:** 22 July 2025

Abstract

In higher education, particularly in programming instruction, students often struggle to follow complex material and maintain consistent practice, while educators face challenges in tracking student progress, providing timely feedback, and ensuring academic integrity during assessments. Existing web-based platforms offer valuable features such as automated code assessment and real-time feedback, but they frequently lack integrated monitoring, localization, and alignment with specific institutional needs.

To address these challenges, IMI-Code, an interactive web-based platform, was developed at the Faculty of Science, University of Kragujevac. The platform enables students to write, execute, and submit code in various programming languages (C, C++, C#, Python) directly through a web browser, without additional software installation. A key innovation is the integration of supervision functionalities, including webcam snapshots, tab-switch detection, and activity logs, which provide enhanced oversight and insight into student engagement and problem-solving.

With automated assessment, immediate feedback, detailed reports for instructors, and bilingual support (Serbian and English), IMI-Code supports formative assessment and data-driven learning. Its flexibility and alignment with departmental structures make it suitable for diverse instructional settings, including remote learning. The paper discusses the platform's pedagogical value, current capabilities, and future development potential, including interoperability with academic systems and applicability in secondary education.

Keywords: Programming Education, Automated Assessment, Student Progress Tracking, Web-based Coding, Academic Integrity

MSC2020: 97P40, 97U50

1. Introduction

In the era of digital transformation, where technology plays an essential role in everyday life, including education, the need for tools that support more effective knowledge transfer and teaching improvement is increasingly recognized. Informatics education, especially programming instruction, requires students to be actively involved, engage in regular self-assessment, and apply knowledge directly, which are needs that traditional classroom formats often struggle to meet. Conventional methods of teaching and assessment demand considerable time and effort from both students and educators and frequently fail to provide timely feedback, which is crucial for mastering programming concepts (Tseng et al., 2024; Patel & Joshi, 2025). The issue of innovation in teaching

* Corresponding author
E-mail address: marina.svicevic@pmf.kg.ac.rs

Original Research Article

methods in computer science is the focus of various studies. In the paper by Sekhar & Goud, (2024) an interactive approach to learning Python programming among computer science students is presented. The teaching methods used included Think Pair Share, laboratory tasks, Moodle, and mock interviews, with the conclusion that applying these methods significantly improved student engagement, understanding, and academic performance.

Within formal education, many students face challenges in following the material, particularly when managing multiple courses that require ongoing practice. They often lack an interactive and engaging platform that would help them better understand content while receiving immediate feedback. At the same time, educators encounter difficulties in objectively tracking student progress, designing tests, and tailoring instruction to individual needs (Gu et al., 2024; Alves & Cipriano, 2025). Existing solutions, while offering valuable features such as automatic code assessment and basic analytics, often fall short in areas such as integrated supervision, localization, and alignment with specific institutional needs.

To address these challenges, the IMI-Code platform (https://imi.pmf.kg.ac.rs/imi-code/) was developed at the Department of Mathematics and Informatics, Faculty of Science, University of Kragujevac. It is designed to support programming education by allowing students to write, run, and submit code directly within a web browser in several programming languages, including C, C++, C#, and Python. Teachers have access to student submissions, test performance, and patterns of work and engagement, enabling more informed instruction and more effective assessment.

A notable feature of the platform is the integration of monitoring options, such as webcam recording, detection of browser tab changes, and logging of user activity throughout the test. These elements allow instructors to gain insight into student behavior during assessments and help ensure fairness in online settings. The platform also provides real-time feedback to students and detailed reports to teachers, supporting continuous learning and data-informed instruction.

With a simple interface, support for both Serbian and English, and customization aligned with institutional course structures, IMI-Code is suitable for diverse teaching scenarios, including distance learning. This paper presents the platform's core features, its contribution to teaching and learning in programming courses, and possibilities for future development and broader application.

2. Related Work and Existing Solutions

Numerous tools and research studies in recent years have focused on improving programming education through interactive web platforms, automated assessment, and systematic tracking of student progress. This section presents selected examples of existing solutions and similar approaches, highlighting certain limitations and challenges that hinder their broader applicability or effective use in specific higher education contexts, such as those in Serbia.

2.1. Existing Research on Web-Based Assessment and Student Progress Tracking

In contemporary higher education, web-based platforms for the automatic assessment of programming assignments have become essential tools in programming instruction. Numerous studies over the past few years emphasize their pedagogical value and their role in facilitating distance learning, providing feedback, and monitoring student performance. For example, Zinovieva et al. (2021) analyzed the use of online programming simulators for training both students and future IT professionals. Their findings highlight the potential of such tools in organizing effective remote instruction, gamifying the learning experience, and enhancing the development of professional competencies. Similarly, Robinson and Carroll (2017) examined the benefits of open-source online platforms in large-group teaching contexts. Their study emphasized the role of web-based systems in reducing administrative workload for instructors and increasing the frequency and immediacy of feedback provided to students - benefits that surpass traditional classroom models.

Several recent studies have focused on the design, implementation, and evaluation of specific systems developed to support programming education. Mekterović et al. (2020) present Edgar, a comprehensive platform created at the University of Zagreb for the automatic evaluation of programming assignments. In their research, the authors report the platform's implementation across more than eight courses, involving over a thousand students per semester. The system supports multiple programming languages, detailed reporting, and visualizations of student progress. Notably, Edgar integrates monitoring features such as webcam snapshots, browser focus detection, and IP logging, providing instructors with tools for maintaining academic integrity during online assessments.

Duong & Chen (2024) describe ProgEdu4Web, designed for teaching web programming. The system was tested in a course with 111 students and demonstrated capabilities not only for automated functionality grading but also for static code analysis and team collaboration support. The evaluation showed positive effects on student motivation and technical skills. However, the study does not address supervision or detection of misconduct.

Auer et al. (2022) introduce MISITcms, developed at the University of Augsburg and tested over multiple semesters in programming courses. The platform allows code submission through a web interface, automatic execution in Docker containers, and grading based on predefined test cases. The evaluation showed that students perceived the system as efficient and transparent, while instructors reported reduced workload. However, the platform does not include monitoring features or tools for plagiarism detection.

Piccolo et al. (2025) present CodeBuddy, used at Brigham Young University and other institutions for short programming exercises. The system supports immediate feedback, multiple test runs, pair programming, and A/B testing of pedagogical strategies. While CodeBuddy includes some access control mechanisms, such as limiting external resources during tests, it does not offer built-in video surveillance or cheating detection via software.

The work of Frankford et al. (2025) deserves particular attention. It showcases the design of an online IDE integrated into the Artemis platform. This system brings modern development environment functionalities to the browser, supporting automatic assessment, learning visualization, and code quality analysis. The evaluation includes stress testing under real conditions. However, academic integrity is not addressed, and the platform lacks monitoring features.

The widely used platforms discussed in the previous section have also been the subject of recent research. CodeRunner has been examined both in terms of its pedagogical impact (Haidar et al., 2023) and its integration with AI-based problem generators for Moodle environments (Serato & Sta Romana, 2024). Replit has been investigated as a cloud-based coding platform that enhances usability, interactivity, and engagement among diploma-level computer science students (Gran et al., 2024). OpenDSA has significantly contributed to the development of interactive e-textbooks and algorithm visualization tools by integrating automated assessment, algorithm visualizations, and community-driven content design (Fouh et al., 2014; Shaffer et al., 2011).

These studies collectively illustrate a wide range of approaches in designing and utilizing web-based platforms for programming education. While some systems, like Edgar, include advanced mechanisms for anomaly detection and monitoring, others focus solely on the quality of feedback and assessment efficiency.

Although numerous web-based platforms for learning programming offer robust features for automatic code assessment, feedback, and partial analytics, they often do not fully meet the specific needs of instructors and students in local academic environments. A review of platforms such as Edgar, CodeBuddy, and MISITcms reveals significant progress in automation, integration with learning management systems, and support for code quality analysis. Most of these systems lack built-in monitoring mechanisms or rely on external tools to maintain academic integrity during assessments.

2.2. Overview of Interactive Coding Platforms as a learning environment

Interactive coding platforms and automated student solution assessment tools play an important role in modern higher education, particularly in computer science courses. Their implementation enables students to acquire practical skills directly, while providing instructors with tools for objective and efficient performance tracking.

One of the most influential platforms in this domain is *HackerRank* for Classrooms, developed as an educational extension of the industry-oriented *HackerRank* platform for technical interviews. It provides an environment where instructors can create tests, monitor student progress, and generate analytical reports. The platform supports multiple programming languages and operates entirely online. There are limitations in terms of customization, such as the lack of open-source availability, the absence of interface localization, and the inability to integrate the tool into existing university systems in ways that would meet the specific needs of diverse institutional contexts (HackerRank).

CodeRunner is a plugin for Moodle, widely adopted by academic institutions already using this learning management system. It enables instructors to create programming questions where students submit source code that is automatically executed and graded in a controlled environment. Teachers can define test cases precisely, receive feedback, and track results in real time. However, configuring the system for complex examples requires significant technical knowledge, and the setup process can be demanding. This makes the platform less suitable for courses without technical support or instructors unfamiliar with Moodle configuration (CodeRunner).

Replit, a browser-based coding platform, formerly supported a feature called Teams for Education that allowed instructors to create virtual classrooms, assign tasks, and collaborate with students. Learners could write and run code without installing any software, while teachers had access to version history and student workflows. However, support for Teams for Education was discontinued in November 2023, significantly limiting the platform's applicability in formal education. Although Replit remains popular for independent learning and smaller projects, its use in structured teaching now requires alternative methods for tracking student progress (Replit).

Google Colab is a cloud-based coding environment developed by Google that allows users to write and execute Python code directly in the browser, with no need for local installation. It is widely used in education for teaching programming, data science, and machine learning, as it provides free access to computational resources, including GPUs. There are limitations in terms of classroom management and progress tracking, as the platform does not natively offer features for monitoring student work or organizing assignments within a centralized system. Instructors must rely on external tools or manual methods to oversee student activities (Colaboratory).

OpenDSA was developed as part of a project funded by the National Science Foundation, with a focus on visualizing algorithms and data structures. The platform includes numerous interactive modules, simulators, and tutorials, and allows instructors to monitor student progress via integrated LMS systems. Unlike the previously mentioned tools, *OpenDSA* is not a coding environment but a system for the theoretical understanding of algorithmic concepts. Its limitations include the technical complexity of creating new modules and the absence of support for real-time coding and testing (OpenDSA).

3. Description of the IMI-Code Platform

The IMI-Code platform was designed and developed as a digital support tool for programming instruction, with the explicit aim of enhancing the learning and assessment process at universities in Serbia. It shares many features with other platforms, such as support for multiple programming languages, real-time feedback, and an intuitive web-based coding environment. The special feature lies in the integration of monitoring functions, including webcam usage, browser window focus loss

detection and activity tracking during task completion – capabilities that are rarely found in other open-source solutions.

Moreover, IMI-Code is fully localized: it supports the Serbian language, organizes students by departments, and is tailored to the needs of instructors at the Faculty of Science, University of Kragujevac. As such, it serves as an example of a platform that combines technological efficiency with pedagogical and institutional relevance, which will be further elaborated in the following sections.

3.1. Context and Target User Groups

The IMI-Code platform was developed to support programming education in an academic context, adapting to the faculty's organizational structure and the diverse roles of its users. Unlike commercial solutions that rely on a universal course model, IMI-Code supports a multi-level hierarchy based on institutes and individual courses. This structure allows instructors to manage content in alignment with the internal organization of their teaching responsibilities, ensuring transparent connections between students, the courses they attend, and the instructors who deliver them.

The platform distinguishes between three levels of access: unregistered users (guests), students, and administrators (teachers). A guest is any user who visits the application's landing page without registration and cannot access any of the platform's features. A student is a registered user who can view the tests for the courses they have selected independently on their profile page and submit solutions to active test assignments. An administrator, usually a teacher, has extended privileges - in addition to access at student level, administrators can create and manage courses, configure test settings and assign roles to other users.

New users register by providing basic information and activating their account via email verification. Forgotten passwords can be reset through the registered email. All new users are assigned the student role and choose courses independently through their profile, ensuring flexibility and a simple onboarding process. Administrators (teachers) retain full control over course integrity and content. They can define course names, create tasks, configure tests, and analyze student results. To grant administrative privileges to a user, an existing administrator must assign the teacher role through the user management interface. Roles are not automatically assigned, which ensures an added layer of security and institutional oversight.

This role-based structure provides controlled access, clear responsibility allocation, and straightforward administrative supervision. With an architecture that reflects institutionally organized teaching, the IMI-Code platform enables seamless integration with existing academic systems without the need for additional adaptation.

3.2. Supporting Learning and Evaluation

The IMI-Code platform is designed to support the teaching process through functionalities that enable the creation, administration, and analysis of programming tests entirely within a web-based environment. The main objective is to provide students with an interactive workspace that offers real-time feedback on tasks, while equipping instructors with tools for systematic progress tracking, response analysis, and academic integrity assurance. Additionally, the platform is bilingual, supporting both Serbian and English, making it suitable for exchange students as well as entire study programs conducted in English.

Test Creation. Instructors with administrator privileges have access to an interface for creating and editing tests within each of their assigned courses. After selecting a course, a list of existing tests is displayed, along with the option to create a new one (Fig. 1). During the creation process, instructors define basic settings such as the test title, programming language to be used, start date and time, test duration, and whether webcam monitoring will be enabled (Fig. 2).

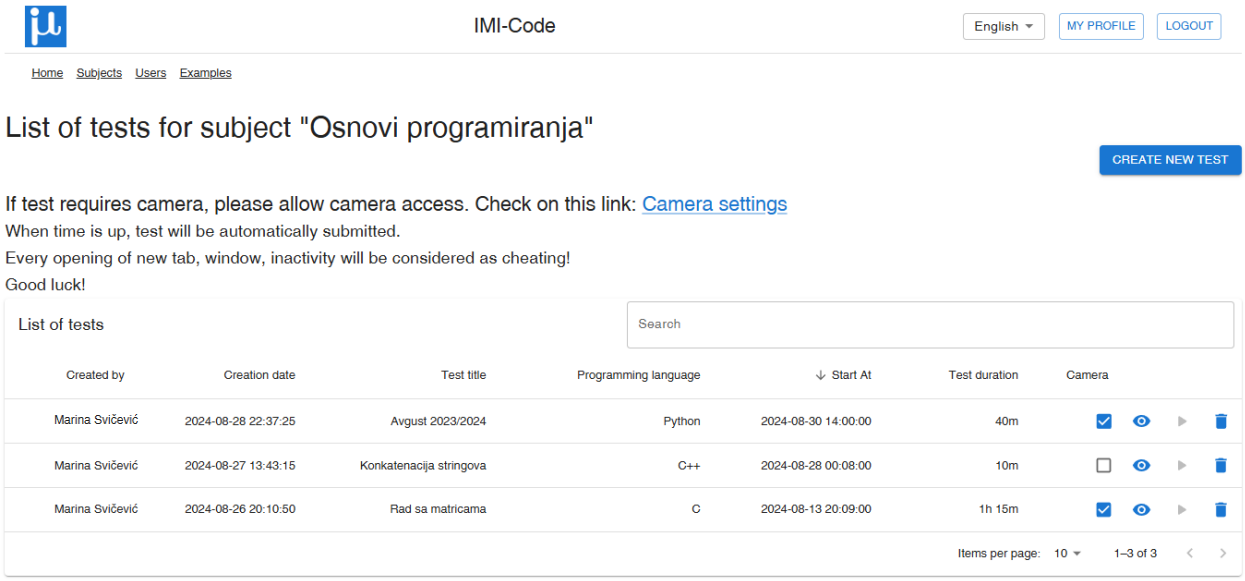

Fig. 1. Test list for the selected course and adding a new test.

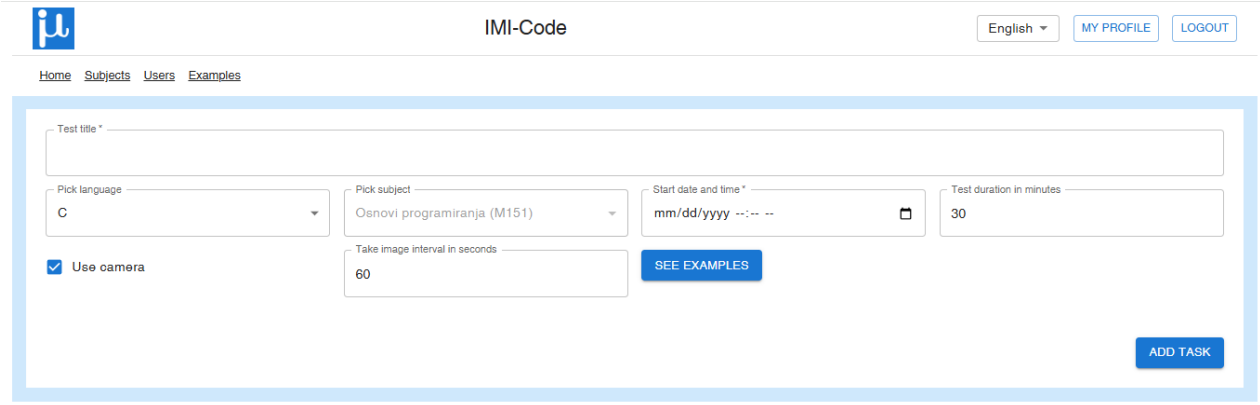

Fig. 2. Basic information about the test being created.

Adding Tasks. For each test, the instructor can add one or more tasks. The task editor supports LaTeX syntax for formatting mathematical and programming content, with real-time preview. Tasks include the problem description, input-output specifications, and test cases, which are saved in plain .txt format to facilitate comparison during automated grading. The platform allows multiple examples and output files to be added for each task (Fig. 3). It is important to note that input files for test cases do not have to be entered directly in the editor; if the file upload option is selected, a file can be uploaded as the input for a given test case. Upon completion, the test can be previewed, saved for later activation, or scheduled for release at a defined time.

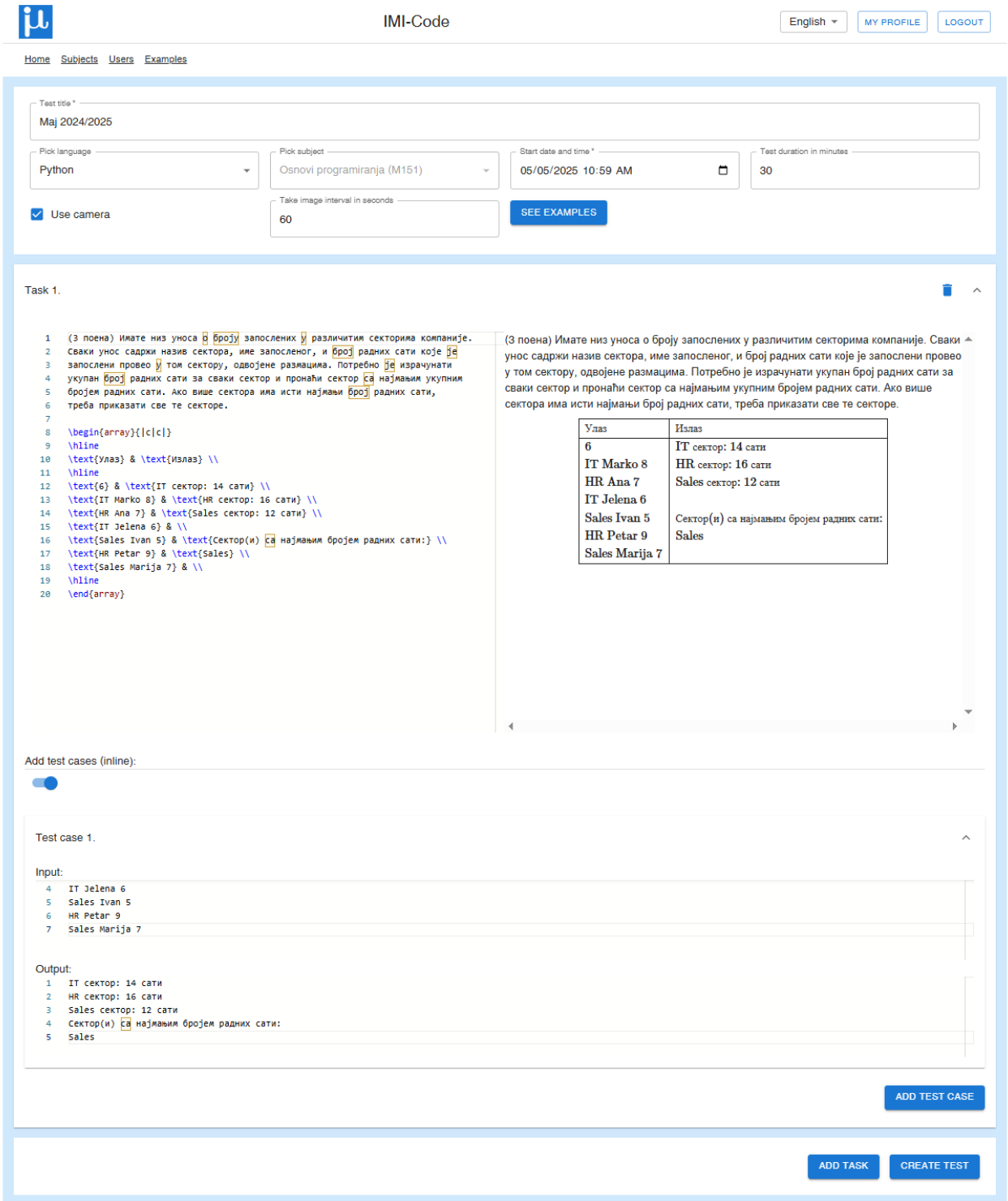

Fig. 3. Adding tasks and test cases.

Execution and Feedback. During the test, students log in to the platform and are initially shown the first task. However, they can expand the view to access the full test, allowing them to choose the order in which they solve tasks, revisit previous ones, and resubmit solutions. The integrated editor enables students to write code, run it, and receive feedback in real time (Fig. 4). Feedback is always provided for the first test case defined by the instructor when creating the task (a single input-output pair), regardless of the total number of test cases. The remaining time for the test is also displayed. Code execution is limited to 15 seconds to prevent system blockage due to infinite loops. Students receive only a success or failure message, without being shown the expected output.

The system automatically records the timestamp of each execution and logs any attempt to leave the active browser tab.

Fig. 4. Example of a running test.

Automatic Analysis and Results. After a student submits their answers or the time expires, the platform generates a detailed report. The instructor can view each individual solution (and download each code submission), the submission time, as well as webcam recordings if that option was enabled. The results can be exported in .csv format for further analysis or archiving (Fig. 5).

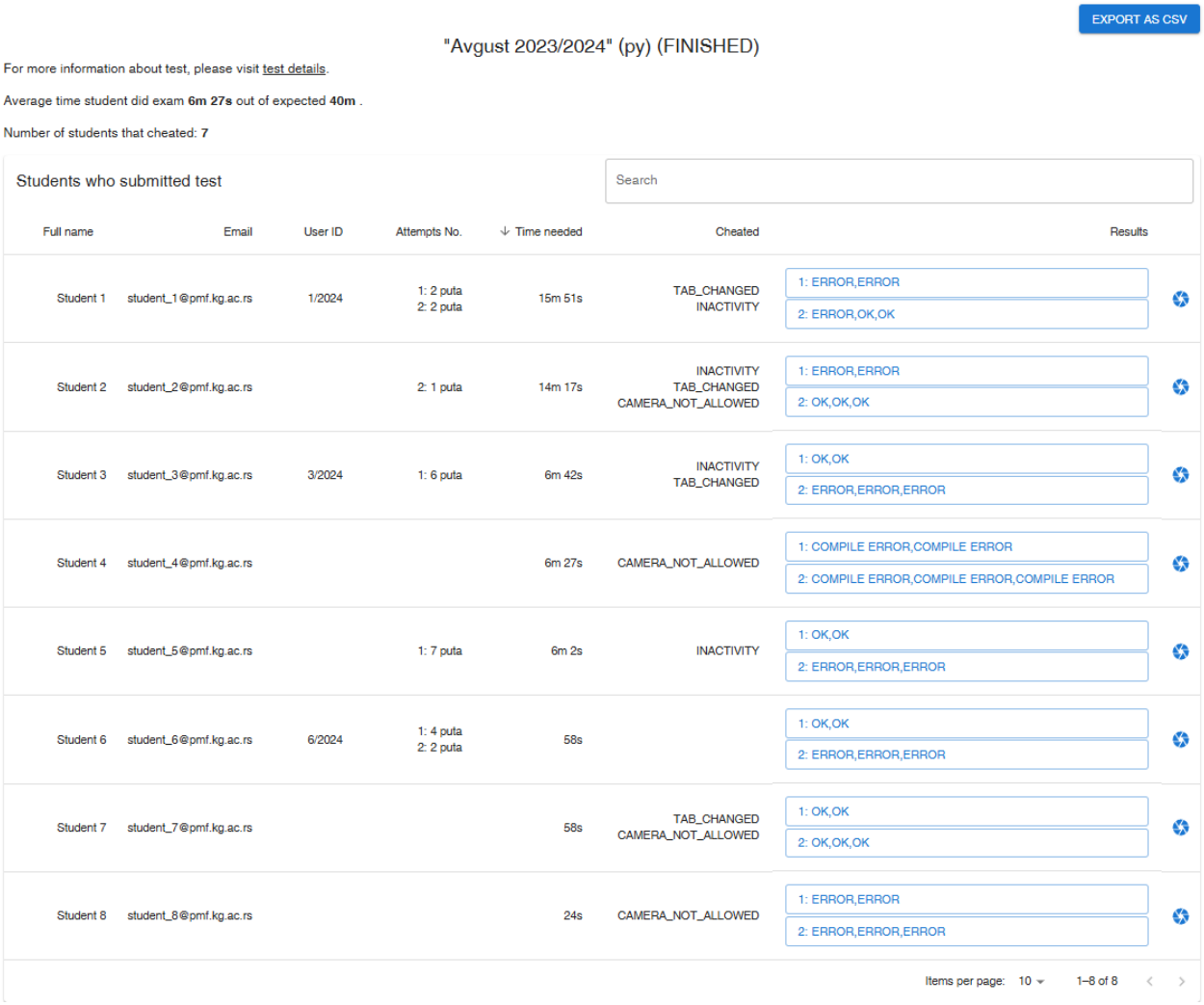

Fig. 5. Example of a completed test with student results.

Such architecture enables instructors to objectively evaluate student solutions with the support of detailed reports, while providing students with continuous feedback throughout their work. All functionalities are tailored for use in an educational context, without the need for additional software or library installations on student computers.

3.3. Monitoring and Integrity Mechanisms

In today's digital learning environments, ensuring reliability and fairness in assessment is a major challenge, especially in the context of online testing. The IMI-Code platform integrates multiple mechanisms for monitoring and maintaining the integrity of assessments, focusing on minimizing disruption to students while maximizing instructor insight.

One of the primary mechanisms is webcam-based video monitoring. When creating a test, instructors can enable this option, after which the system periodically captures images of the student during the test at unpredictable intervals. These images are time-stamped and stored as part of the test report (Fig. 5). If the camera is inactive or the student blocks its function, this is also recorded and flagged in the test summary as a potential irregularity.

Students are clearly informed about the rules regarding the use of their webcam before the test, and their explicit consent is required before accessing a monitored test. The collected data is used solely for monitoring purposes and is stored in accordance with the platform's privacy policy. Recordings are not used for any other purpose.

Additionally, the platform detects when the student leaves the active browser tab (e.g., by opening another tab or application). Each loss of focus is logged with a timestamp, allowing

instructors to identify potential instances of misconduct during the test. Information about the number of tab-switching events, as well as their timing relative to task activity, is also included in tabular result summaries (Fig. 5).

These mechanisms are not intended to replace human supervision but to serve as an additional layer of control, particularly in cases where in-person proctoring is not feasible. At the same time, they enhance transparency and trust in the outcomes of digital assessments.

3.4. Localization, Accessibility, and UX Considerations

One of the advantages of the IMI-Code platform lies in its thoughtfully designed user interface, which adapts to the needs of different user groups, as well as in its localization features. Unlike global tools that offer only English interfaces (or multilingual options that do not include Serbian), IMI-Code supports both Serbian and English and allows users to freely choose the interface language. This is especially helpful for students whose previous education did not place a strong emphasis on English.

In addition to language support, the interface is designed to reduce cognitive load during task solving. All interactions take place on a single page, minimizing the need for navigation and helping students stay focused. The code editor and task description are displayed together on the same screen. Although the first task is shown initially, students have the option to expand their view and see the entire list of tasks, allowing them to choose the order in which they solve them and return to previously attempted ones. This flexibility supports both focused engagement and an individual pace of work, in line with pedagogical principles of gradual and adaptive learning.

Since the platform runs entirely in a web browser, no additional software installation is required. This makes it easy to use both on university and personal computers, without technical barriers. Its accessibility ensures that IMI-Code can be used in various settings, including situations where students do not have access to high-end equipment.

Thanks to these features, including support for Serbian and English, a simple and focused interface, and broad technical accessibility, IMI-Code offers a programming environment that is tailored to the needs of students, instructors, and the practical conditions of university-level teaching.

4. Discussion

The IMI-Code platform was designed with a strong pedagogical focus, aiming to support learning through continuous progress tracking and the encouragement of independent work. Unlike traditional testing systems that primarily serve as a final knowledge check without insight into the process of solving problems, IMI-Code enables a dynamic and interactive approach to assessment and understanding. This is achieved through immediate feedback during problem-solving and through detailed reports that provide instructors with insight into how students approached tasks, the timeline of their actions, the success of their attempts, and their behavior during the session.

The platform supports a data-informed teaching model, allowing instructors to analyze student results in real time or retrospectively - at the level of the entire test, individual tasks, or specific input-output cases. Such analysis makes it possible to identify common mistakes, recognize behavior patterns, and pinpoint areas that require additional attention. In practice, these insights help tailor instruction to individual student needs, supporting the principles of personalized and adaptive learning. Unlike platforms such as CodeBuddy (Piccolo et al., 2025) and MISITcms (Auer et al., 2022), which lack built-in monitoring functionalities, IMI-Code provides integrated supervision through webcam activity and tab-switch detection. These features respond to the growing need for academic integrity mechanisms during online assessments, previously highlighted as missing or insufficient in various studies.

In addition to assessment, IMI-Code plays an important role as a tool for fostering self-directed learning. Students can submit and run their solutions multiple times during the test, receiving

limited but helpful feedback for one test case. Although the system does not display the expected output, it informs students whether their solution is correct. This form of feedback encourages students to analyze and debug their code independently. It promotes self-assessment, logical reasoning, and builds confidence through progressive engagement. Furthermore, by accessing the learning history, students can track their own progress and reflect on their learning pace and strategies. As emphasized by Robinson and Carroll (2017), immediate feedback and interaction are key elements in supporting learning in large-group programming courses. IMI-Code builds upon this by offering iterative task attempts with guided feedback, promoting self-assessment and reflective learning.

Progress tracking on the platform goes beyond correct or incorrect answers. The system automatically logs timestamps for each code execution and task submission, enabling instructors to analyze working pace and focus. It also records technical indicators such as tab-switching behavior and webcam activity. While these features are primarily designed to support test integrity, they also offer useful context for interpreting performance, helping to identify potential difficulties or irregularities during test-taking. According to Mekterović et al. (2020), detailed analytics and reporting functionalities are essential for data-informed instruction in programming courses. IMI-Code extends these ideas by allowing instructors to access time-stamped execution histories, behavior patterns, and task-specific analysis in real time or retrospectively.

IMI-Code provides a comprehensive digital environment for learning support, behavior analysis, and diagnosis of challenges in mastering programming concepts. These features make the platform a pedagogically relevant tool aligned with modern teaching approaches based on student engagement and guided learning.

5. Future Development and Integration Potential

Although the IMI-Code platform already provides a stable and pedagogically oriented environment for practicing programming and conducting assessments, there are several concrete opportunities for further improvement and broader application.

One important area relates to expanding support for additional programming languages. The current version includes C, C++, C#, and Python, which align well with the curriculum of most computer science and informatics courses. However, adding support for other languages such as Java, JavaScript, or R would make the platform applicable to courses that focus on web development, data analysis, or interactive visualizations. This extension would further increase the relevance of IMI-Code in diverse educational contexts where programming is important but not necessarily the core focus.

Another important direction for future development is enabling better connection and data exchange between IMI-Code and learning management systems such as Moodle or other platforms used at universities. Enabling data exchange between IMI-Code and university services would allow automatic synchronization of information on assessments, attendance, grades, and course enrollment, reducing the need for manual input and parallel maintenance. Such integration would lead to a more efficient teaching process and better alignment of digital tools. A particularly promising option is the integration with student portals and academic databases. Currently, students select their courses manually after account activation. If connected to the official academic information system, students could be automatically assigned to both current and transferred courses, eliminating selection errors and providing instructors with an immediate overview of course enrollment.

In terms of expanding its target audience, the platform also has strong potential for use in secondary education. With programming increasingly present in the curricula of high schools and technical schools, a localized and pedagogically adapted environment like IMI-Code could be a valuable resource for both teachers and students. With minimal adjustments, such as a simplified interface, support for tutorials, and formative testing, the platform could be effectively used for entrance exam preparation or programming competitions.

In addition, future development could take into account suggestions received as feedback from students, particularly regarding interface usability, feature expectations and clarity of feedback mechanisms. Consideration of the user experience and learner perspective would support better engagement and further pedagogical alignment.

Finally, future upgrades could include more advanced analytics modules, such as visual reports, graphical progress overviews, or recommendation systems based on student performance data. These features would further enhance the platform's role in data-informed teaching and personalized learning.

Advancing in these directions would significantly broaden the scope and value of IMI-Code, positioning it not just as a testing tool but as a comprehensive support system for programming education and digital transformation in teaching.

6. Conclusion

This paper presents IMI-Code as an interactive web-based programming environment with integrated monitoring and student progress tracking, developed for the needs of higher education. Unlike many commercial solutions, the platform is adapted to the specific context of programming education in Serbia. It supports the Serbian language, aligns with institutional course structures, and includes monitoring features during assessments to help ensure academic integrity.

IMI-Code enables students to build practical programming skills and logical thinking through structured tasks, real-time feedback, and the opportunity to make multiple attempts. For instructors, it offers tools to manage, tailor, and evaluate assignments while providing a comprehensive overview of student activity and progress. Features such as webcam-based supervision and browser activity tracking further strengthen the reliability of assessment, making the platform appropriate for formal examination settings.

In addition to its current technical reliability and pedagogical value, the platform holds strong potential for future development. Integration with university information systems, support for additional programming languages, and adaptation for secondary education use could extend its relevance and usability across different educational environments. A plan for further research in this area includes expanding the range of supported programming languages and conducting evaluations of student performance while using the platform. The aim of such evaluations would be to assess the impact of the platform on learning outcomes and use the findings to guide its continuous improvement.

In conclusion, IMI-Code should be seen not only as a technological solution but as a pedagogical tool that combines interactivity, learning analytics, and assessment integrity in a single environment, contributing to the improvement of programming instruction and the broader digital transformation of education.

Acknowledgment

This work was supported by the Serbian Ministry of Education, Science and Technological Development (Agreement No. 451-03-137/2025-03/ 200122).

References

ABD AZIZ, M., SHAARANI, A. S., BAHARIN, S. S. K., OTHMAN, Z., & HASSAN, N. H. (2024, December). Building Strong Foundations in C++: Scaffolded Exercises with CodeRunner. In *2024 International Conference on TVET Excellence & Development (ICTeD)* (pp. 98-102). IEEE.

ALVES, P., & CIPRIANO, B. P. (2025). Automated Assessment in Mobile Programming Courses: Leveraging GitHub Classroom and Flutter for Enhanced Student Outcomes. *arXiv preprint arXiv:2504.04230*.

ASGARI, M., TSAI, F. C., MANNILA, L., STRÖMBÄCK, F., & SADIQUE, K. M. (2024). Students' perspectives on using digital tools in programming courses: A cross country case study between Sweden and Taiwan. *Discover Education*, *3*(1), 57.

AUER, F., FREI, J., MÜLLER, D., & KRAMER, F. (2022). Perspective on code submission and automated evaluation platforms for university teaching.

CODERUNNER. University of Canterbury. Retrieved July 10, 2025, from https://coderunner.org.nz

COLABORATORY. Google Research. Retrieved July 10, 2025, from https://colab.research.google.com

DUONG, H. T., & CHEN, H. M. (2024). ProgEdu4Web: An automated assessment tool for motivating the learning of web programming course. *Computer Applications in Engineering Education*, *32*(5), e22770.

FOUH, E., KARAVIRTA, V., BREAKIRON, D. A., HAMOUDA, S., HALL, S., NAPS, T. L., & SHAFFER, C. A. (2014). Design and architecture of an interactive eTextbook–The OpenDSA system. *Science of computer programming*, *88*, 22-40.

FRANKFORD, E., CRAZZOLARA, D., VIERHAUSER, M., MEIßNER, N., KRUSCHE, S., & BREU, R. (2025). An Online Integrated Development Environment for Automated Programming Assessment Systems. *arXiv preprint arXiv:2503.13127*.

GRAN, S. S., ENGKAMAT, A., & LEONG, Y. M. (2023). From Syntax to Solutions: Evaluating the Efficacy of a Real-Time Programming Platform. *International Journal of Business and Technology Management*, *5*(4), 297-302.

GU, R., ROJAS, J. M., & SHIN, D. (2024, December). Can test generation and program repair inform automated assessment of programming projects?. In *2025 IEEE Conference on Software Testing, Verification and Validation (ICST)*. Institute of Electrical and Electronics Engineers (IEEE).

HACKERRANK FOR SCHOOLS. HackerRank. Retrieved July 10, 2025, from https://www.hackerrank.com/products/school

HAIDAR, S., YAACOUB, A., & IONASCU, F. (2023, June). Evaluating the effectiveness of a new programming teaching methodology using CodeRunner. In *The Learning Ideas Conference* (pp. 211-223). Cham: Springer Nature Switzerland.

MEKTEROVIĆ, I., BRKIĆ, L., MILAŠINOVIĆ, B., & BARANOVIĆ, M. (2020). Building a comprehensive automated programming assessment system. *IEEE access*, *8*, 81154-81172.

OPENDSA. Virginia Tech. Retrieved July 10, 2025, from https://opendsa-server.cs.vt.edu

PATEL, A., & JOSHI, H. (2025). DebugProGrade: Improving automated assessment of coding assignments with a focus on debugging. *Journal of Informatics and Web Engineering*, *4*(1), 184-199.

PICCOLO, S. R., TUFT, E., TATLOW, P. J., ELIASON, Z., & STEPHENSON, A. (2025). CodeBuddy: A Programming Assignment Management System for Short-Form Exercises. *Journal of Open Research Software*, *13*(1).

REPLIT. Replit Inc. Retrieved July 10, 2025, from https://replit.com

ROBINSON, P. E., & CARROLL, J. (2017, April). An online learning platform for teaching, learning, and assessment of programming. In *2017 IEEE Global Engineering Education Conference (EDUCON)* (pp. 547-556). IEEE.

SEKHAR, P. R., & GOUD, S. (2024). Collaborative Learning Techniques in Python Programming: A Case Study with CSE Students at Anurag University. *Journal of Engineering Education Transformations*, *38*.

SERATO, J. V. D., & STA. ROMANA, C. L. (2024, April). Development and Utilization of AI-Enabled Automatic Programming Problem Generator Using the CodeRunner Plugin of Moodle. In *Proceedings of the 2024 10th International Conference on Education and Training Technologies* (pp. 147-152).

SHAFFER, C. A., KARAVIRTA, V., KORHONEN, A., & NAPS, T. L. (2011, November). OpenDSA: beginning a community active-ebook project. In *Proceedings of the 11th Koli Calling International Conference on computing education research* (pp. 112-117).

TSENG, E. Q., HUANG, P. C., HSU, C., WU, P. Y., KU, C. T., & KANG, Y. (2024, December). CodEv: An Automated Grading Framework Leveraging Large Language Models for Consistent and Constructive Feedback. In *2024 IEEE International Conference on Big Data (BigData)* (pp. 5442-5449). IEEE.

ZINOVIEVA, I. S., ARTEMCHUK, V. O., IATSYSHYN, A. V., POPOV, O. O., KOVACH, V. O., IATSYSHYN, A. V., ... & RADCHENKO, O. V. (2021, March). The use of online coding platforms as additional distance tools in programming education. In *Journal of physics: Conference series* (Vol. 1840, No. 1, p. 012029). IOP Publishing.