International Journal of Web of Multidisciplinary Studies
(Peer-Reviewed, Open Access, Fully Refereed International Journal)
website: https://ijwos.com OPEN ACCESS E-ISSN : 3049-2424
Volume 3 Issue 9 , September 2026 DOI: https://doi.org/10.71366/ijwos03092659607

CodeSense AI: An Explainable Behavioral Analytics Framework for Virtual Coding Laboratories Using AI-Based Focus Monitoring and Skill Assessment

Krupa Rani S[1], C Spoorthi[2]
[1] *Assistant Professor, Department of Computer Science, GM University Davangere,India*
[2] *Student, Department of Computer Science, GM University Davangere, India.*

Article Info

Article History:

Published: 02 Sept 2026

Publication Issue:
Volume 3, Issue 9
September-2026

Page Number:
1-13

Corresponding Author:
C Spoorthi

Abstract:

Practical programming laboratories are essential for developing coding skills, but existing virtual coding platforms primarily evaluate the final code submission and provide limited insight into student engagement during lab sessions. This paper presents CodeSense AI, an explainable behavioral analytics framework for virtual coding laboratories that integrates AI-based focus monitoring with programming skill assessment. The proposed system provides a secure coding environment while continuously tracking behavioral metrics such as typing activity, idle time, tab switching, copy-paste attempts, and webcam-based presence detection. These metrics are analyzed to generate Focus Score and Integrity Score, enabling fair and transparent evaluation of student participation. Additionally, an AI-assisted code review module offers personalized feedback on code quality and programming logic. A teacher dashboard presents real-time activity analytics, engagement reports, and performance summaries to support effective monitoring and assessment. The proposed framework improves the authenticity of programming lab evaluations, enhances student accountability, and provides data-driven insights for educators in modern virtual learning environments.

Keywords: Virtual Coding Laboratory, Behavioral Analytics, Artificial Intelligence (AI), Focus Monitoring, Skill Assessment, Programming Education, Code Review, Student Engagement, Integrity Score, Teacher Analytics Dashboard

1. Introduction

Programming laboratories play a vital role in helping students develop practical coding and problem-solving skills. With the increasing adoption of online and hybrid learning, virtual coding platforms have become widely used for conducting programming labs. However, most existing platforms primarily evaluate the final code submission and provide limited information about the student's actual participation and coding behavior during the session.

Traditional virtual coding environments do not effectively monitor activities such as copy-paste attempts, tab switching, idle time, or student presence, making it difficult for instructors to ensure fair assessments and identify genuine learning efforts. As a result, teachers often face challenges in evaluating student engagement and maintaining the integrity of programming lab examinations.

To address these limitations, this paper proposes CodeSense AI, an explainable behavioral analytics framework for virtual coding laboratories. The proposed system combines a secure coding environment with AI-based focus monitoring and skill assessment by tracking behavioral metrics such as typing activity, idle time, tab switching, copy-paste attempts, and webcam-based presence detection. These metrics are used to generate Focus Score and Integrity Score, providing a transparent evaluation of student engagement.

In addition, the framework integrates AI-assisted code review to provide constructive feedback on code quality and programming logic, while a teacher dashboard offers real-time analytics, activity timelines, and performance reports.

The proposed approach aims to improve the authenticity, fairness, and effectiveness of programming lab assessments in modern educational environments.

2. Statement of the Problem

Traditional programming laboratories and existing online coding platforms primarily evaluate students based on the final code submission, without considering their coding behavior or level of engagement during the lab session. This makes it difficult for instructors to determine whether students have completed the work independently or relied on unfair practices such as copy-pasting code or switching to external resources.

Additionally, teachers face challenges in monitoring multiple students simultaneously, especially in remote and hybrid learning environments. The absence of behavioral analytics, focus tracking, and measurable engagement data limits the effectiveness and transparency of programming skill assessment.

Therefore, there is a need for an intelligent virtual coding laboratory that combines secure coding, behavioral monitoring, AI-assisted code evaluation, and engagement analytics to ensure fair, authentic, and data-driven assessment of students' programming skills.

3. Study Methodology

A. Description of Study Area

The study focuses on the development and evaluation of an **AI-enabled Virtual Coding Laboratory** for programming education in higher educational institutions. The system is designed for use by students and instructors in computer science and information technology courses, where practical coding sessions are conducted in online or hybrid learning environments. The proposed platform provides a secure web-based coding environment that allows students to complete assigned programming tasks while continuously monitoring their coding behavior and engagement. It collects behavioral metrics such as typing activity, idle time, tab switching, copy-paste attempts, and webcam-based presence detection to assess student focus and participation during lab sessions.

The study covers the design, implementation, and evaluation of the platform using modern web technologies and AI-assisted analytics. The effectiveness of the system is assessed based on its ability to improve the authenticity of programming assessments, provide meaningful engagement analytics, and support instructors through a comprehensive teacher dashboard. The study does not include advanced biometric monitoring or operating system-level security features, as these are considered outside the scope of the proposed framework.

B. Data Collection and Analysis

The proposed **CodeSense AI** framework collects both programming performance data and student behavioral data during virtual coding lab sessions. Programming data includes assigned tasks, source code submissions, test case results, and AI-generated code review feedback. Behavioral data includes typing activity, idle time, tab switching events, copy-paste attempts, session duration, and webcam-based presence detection.

All collected data is securely stored in a PostgreSQL database and analyzed to evaluate student engagement and programming performance. Behavioral metrics are processed to calculate **Focus Score** and **Integrity Score**, while programming metrics such as code correctness, test case pass rate, and code quality are used to assess technical skills. The combined analysis is presented through a teacher dashboard, providing activity timelines, engagement reports, performance summaries, and comparative analytics to support fair and data-driven assessment of students' programming abilities.

4. Results and Discussion

The proposed **CodeSense AI** framework was successfully designed and implemented as a web-based virtual coding laboratory to provide a secure, intelligent, and transparent environment for programming practical sessions. The

system integrates a virtual code editor, student activity monitoring, AI-assisted code evaluation, and a teacher analytics dashboard into a single platform. Unlike traditional online coding environments that primarily focus on evaluating the final program output, CodeSense AI continuously monitors student engagement and coding behavior throughout the entire lab session. This enables instructors to assess not only the correctness of submitted programs but also the authenticity of students' participation and overall learning behavior.

The system was developed using **Next.js**, **React**, **Tailwind CSS**, and **Monaco Code Editor** for the frontend, while **Next.js API Routes** handled backend operations. Student and session data were securely stored in a **PostgreSQL** database. Browser-based monitoring techniques, including the **Visibility API**, clipboard event listeners, keyboard activity tracking, and webcam-based presence detection, were integrated to collect behavioral metrics without requiring additional hardware or software installation. The AI-assisted code review module utilized a Large Language Model (LLM) to generate constructive feedback regarding program logic, coding practices, and possible improvements.

Virtual Coding Laboratory Environment

The first objective of the proposed framework was to provide students with a dedicated virtual coding laboratory where they could perform assigned programming tasks in a controlled environment. After successful authentication, students are presented with a dashboard containing their assigned programming exercises, lab schedules, and previous submissions. Upon selecting a task, students enter the coding workspace, which includes an integrated Monaco Code Editor supporting multiple programming languages.

The coding environment offers several features designed to improve the overall learning experience. These include syntax highlighting, automatic code saving, timer-based lab sessions, programming language selection, and structured code submission. Unlike conventional online editors, the proposed platform continuously records coding activities throughout the session rather than evaluating only the final submitted program.

The secure coding environment successfully prevented unauthorized clipboard usage by disabling copy-paste operations within the editor. Whenever a student attempted to paste external code, the system recorded the event as a behavioral flag without interrupting the coding session. This approach discouraged unfair practices while maintaining a smooth user experience.

Student Activity Monitoring

One of the major contributions of the proposed system is its continuous behavioral monitoring mechanism. During every lab session, multiple browser events are captured and stored as activity logs. These events include typing frequency, periods of inactivity, browser tab switching, loss of window focus, copy-paste attempts, session duration, and webcam presence status.

The monitoring process operates in real time and does not interfere with the student's programming activities. Each event is timestamped and stored in the database for further analysis. The collected behavioral information provides instructors with detailed insights into how students approached programming tasks rather than simply observing the final output.

Typing activity proved to be one of the most informative indicators of student engagement. Students actively working on programming tasks generated continuous keyboard events throughout the session, while prolonged inactivity indicated possible distractions or interruptions. Similarly, repeated tab switching and browser focus loss suggested that students frequently accessed external resources, prompting instructors to review these sessions more carefully.

Idle time measurement also played an important role in understanding student behavior. Instead of considering short pauses as inactivity, the system applied configurable idle thresholds to distinguish between normal thinking time and prolonged disengagement. This reduced the likelihood of incorrectly identifying genuine problem-solving activities as suspicious behavior.

Behavior Tracking Results

The implemented monitoring module successfully captured all targeted behavioral events during virtual lab sessions. The system generated detailed activity logs containing information such as login time, task start time, typing intervals, copy-paste attempts, tab switching events, idle duration, and session completion time. These logs formed the foundation for subsequent focus and integrity analysis.

An example of the behavioral information recorded during a programming session is presented in **Table 4.1**

Table 4.1 Sample Student Activity Log

Metric	Recorded Value
Session Duration	60 minutes
Active Coding Time	47 minutes
Idle Time	8 minutes
Tab Switching Events	3
Copy-Paste Attempts	1
Window Focus Loss	3
Webcam Presence	Present
Auto Save Events	92

The results demonstrate that the monitoring system can successfully collect detailed behavioral information without affecting the coding experience. Such information is valuable for understanding student engagement and identifying sessions that require additional review by instructors.

Secure Code Submission

The code submission module operated successfully throughout testing. Students were able to submit their solutions before the session timer expired, while automatic saving ensured that no work was lost due to unexpected browser interruptions. Each submission included not only the program source code but also the complete behavioral profile recorded during the session. This integration enables instructors to evaluate both technical performance and student engagement simultaneously.

In addition to storing source code, the system maintained metadata such as submission time, execution status, selected programming language, activity statistics, and AI-generated review feedback. This comprehensive dataset provides a richer assessment compared to conventional online programming systems.

AI-Assisted Code Review

The AI-assisted evaluation module generated constructive feedback immediately after program submission. Instead of assigning only a correctness score, the system analyzed code readability, logical structure, variable naming conventions, modularity, and potential improvements.

Students received personalized suggestions highlighting strengths as well as areas requiring improvement. For example, if a submitted program contained unnecessary nested loops or repetitive code, the AI module recommended simplified approaches and better coding practices. Similarly, if meaningful variable names or comments were missing, the feedback encouraged students to improve code readability.

This immediate feedback mechanism transformed the coding laboratory from a purely evaluation-oriented environment into a learning-centered platform. Students could understand not only whether their programs were correct but also how they could enhance code quality according to industry best practices.

Initial Performance Observations

During implementation and testing, the proposed framework demonstrated stable performance across all major modules. The coding environment remained responsive even while monitoring multiple behavioral events simultaneously. Browser event listeners operated efficiently without introducing noticeable latency in typing or code editing.

The integration of activity monitoring, secure code submission, AI-assisted code review, and database logging confirmed the feasibility of combining behavioral analytics with programming assessment in a single web-based platform. The collected results indicate that the proposed system successfully addresses several limitations of existing virtual coding laboratories by providing continuous engagement monitoring, transparent activity tracking, and richer performance evaluation.

The next section discusses the analysis of **Focus Score**, **Integrity Score**, teacher dashboard analytics, behavioral trends, and overall system effectiveness based on the collected activity data.

Focus Score Analysis

One of the primary objectives of the proposed CodeSense AI framework is to measure student engagement during programming laboratory sessions through a quantitative Focus Score. Unlike conventional virtual coding platforms that assess only the final program output, the proposed system continuously evaluates student behavior throughout the coding session. The Focus Score is calculated using multiple behavioral indicators, including active coding time, typing consistency, idle duration, tab switching frequency, copy-paste attempts, and webcam-based presence detection.

Each behavioral metric contributes to the overall engagement score. Students who remain active within the coding environment, maintain consistent typing activity, and avoid unnecessary browser interruptions receive higher Focus Scores. Conversely, excessive idle periods, repeated tab switching, frequent copy-paste attempts, or prolonged absence from the webcam reduce the calculated score. This multi-factor evaluation provides instructors with a more comprehensive understanding of student participation than traditional submission-based assessment.

During testing, the Focus Score successfully differentiated between students who actively engaged in programming tasks and those who demonstrated irregular or potentially suspicious behavior. Students with consistent coding activity generally achieved higher scores, while sessions characterized by extended inactivity or multiple browser interruptions

resulted in comparatively lower engagement ratings. These observations indicate that behavioral analytics can effectively complement traditional programming assessment methods.

Integrity Score Evaluation

In addition to engagement measurement, the proposed system introduces an Integrity Score to evaluate the authenticity of each coding session. The Integrity Score is derived from behavioral indicators associated with potential unfair practices, including copy-paste attempts, browser tab switching, repeated loss of window focus, and prolonged inactivity.

The system does not automatically classify any student as engaging in malpractice based on a single event. Instead, multiple behavioral signals are considered together before generating the final Integrity Score. For example, occasional tab switching for documentation or brief thinking pauses are treated as normal programming behavior. However, repeated switching between browser tabs combined with several copy-paste attempts and extended idle periods may indicate reduced coding integrity and are reflected in the final assessment.

This balanced evaluation minimizes false positives while providing instructors with meaningful insights into student behavior. Rather than replacing instructor judgment, the Integrity Score serves as an additional decision-support metric during laboratory evaluations.

Behavioral Analytics Dashboard

The teacher dashboard developed as part of the proposed framework provides a centralized interface for monitoring student activities and reviewing programming performance. Instead of manually supervising every student during laboratory sessions, instructors can access detailed analytics generated automatically by the system.

The dashboard presents key information including student attendance, assigned programming tasks, submission status, Focus Score, Integrity Score, session duration, activity timeline, typing frequency, idle time, copy-paste attempts, and browser tab switching events. Individual student reports and class-wide performance summaries are displayed through interactive charts and tables, enabling instructors to quickly identify both highly engaged students and those requiring additional guidance.

A timeline view further enhances transparency by displaying significant events throughout the coding session. This chronological representation allows instructors to observe periods of active programming, inactivity, browser focus changes, and code submission events, providing a complete overview of each student's learning behavior.

AI-Assisted Code Review Results

The integration of Large Language Models (LLMs) into the proposed framework enables automated qualitative assessment of submitted programs. After code submission, the AI module analyzes the program and generates constructive feedback related to programming logic, readability, coding standards, and possible optimizations.

Testing demonstrated that AI-generated feedback extends beyond traditional automated grading systems. While conventional online judges typically evaluate only execution correctness and test case results, the proposed framework also highlights programming best practices. Students receive suggestions regarding meaningful variable names, modular code organization, unnecessary code duplication, algorithm efficiency, and logical improvements.

The immediate availability of personalized feedback encourages self-learning and iterative improvement. Students can understand not only whether their solution is correct but also how it can be enhanced according to accepted software development practices. This contributes to better programming skills and supports continuous learning.

Student Performance Evaluation

The proposed framework combines technical programming performance with behavioral analytics to produce a more balanced evaluation of student learning. Instead of relying solely on code correctness, multiple performance indicators are considered simultaneously.

Programming performance includes:

- Code correctness
- Test case pass rate
- Code quality
- Programming logic

Behavioral performance includes:

- Active coding time
- Idle duration
- Tab switching frequency
- Copy-paste attempts
- Webcam presence
- Focus Score
- Integrity Score

This integrated assessment provides instructors with a broader understanding of student performance and helps distinguish genuine programming effort from superficial completion of programming tasks.

An example of a student performance summary generated by the system is shown in **Table 4.2**

Table 4.2 Sample Student Performance Report

Evaluation Metric	**Result**
Programming Task	Completed
Code Correctness	92%
Test Cases Passed	18 / 20
AI Code Quality Rating	Good
Active Coding Time	48 Minutes

Idle Time	6 Minutes
Tab Switching	2
Copy-Paste Attempts	0
Focus Score	91%
Integrity Score	95%

The generated report provides a comprehensive overview of both programming competency and coding behavior, enabling more transparent laboratory evaluation.

Teacher Monitoring Efficiency

One of the significant outcomes observed during implementation is the improvement in instructor monitoring efficiency. In conventional programming laboratories, instructors often face difficulty supervising multiple students simultaneously, particularly in online or hybrid learning environments. Monitoring individual coding activities manually is both time-consuming and prone to oversight.

The proposed dashboard significantly reduces this challenge by automatically collecting, organizing, and visualizing behavioral metrics. Instead of continuously observing every student, instructors can identify unusual activity through generated analytics and review only those sessions requiring further attention. This improves resource utilization while ensuring fair evaluation across the entire class.

The availability of class-wide engagement summaries also enables instructors to identify common learning difficulties. For example, if multiple students demonstrate prolonged idle periods on a particular programming task, it may indicate that the assignment requires additional explanation or instructional support.

Behavioral Trends Observed

Analysis of collected behavioral logs revealed several consistent patterns during coding sessions. Students who actively participated in programming tasks generally exhibited continuous typing activity, limited browser switching, minimal idle time, and consistent webcam presence. These sessions resulted in high Focus Scores and Integrity Scores.

Conversely, sessions involving repeated browser focus loss, multiple copy-paste attempts, and prolonged inactivity produced comparatively lower behavioral scores. Such patterns do not necessarily confirm academic misconduct but provide instructors with valuable information for reviewing the corresponding submissions more carefully.

The behavioral analytics generated by CodeSense AI therefore function as a decision-support mechanism rather than a replacement for instructor evaluation. By combining technical programming assessment with engagement analytics, the proposed framework provides a more transparent and holistic understanding of student learning behavior.

The following section presents a comparative discussion of the proposed framework with existing virtual coding platforms, highlights the overall effectiveness of the system, discusses its limitations, and outlines future research directions.

Comparative Analysis with Existing Systems

A major objective of this research was to overcome the limitations of existing virtual coding platforms by integrating behavioral analytics with programming assessment. Traditional online coding environments mainly evaluate program correctness and provide limited information regarding student engagement during laboratory sessions. As a result, instructors often have difficulty determining whether a submitted solution genuinely reflects a student's programming ability.

The proposed CodeSense AI framework extends the capabilities of conventional coding platforms by combining secure code submission, behavioral monitoring, AI-assisted code review, and teacher analytics into a unified system. This integrated approach provides a more transparent and comprehensive evaluation of programming laboratory performance.

A comparison between existing virtual coding platforms and the proposed framework is presented in **Table 4.3**

Table 4.3 Comparison of Existing Systems and Proposed CodeSense AI Framework

Feature	**Existing Systems**	**Proposed CodeSense AI**
Code Submission	✓	✓
Code Correctness Evaluation	✓	✓
AI Code Review	Limited	✓
Copy-Paste Detection	✗	✓
Tab Switching Detection	✗	✓
Idle Time Monitoring	✗	✓
Typing Activity Tracking	✗	✓
Webcam Presence Detection	✗	✓

Focus Score Generation	✗	✓
Integrity Score Calculation	✗	✓
Teacher Analytics Dashboard	Basic	Comprehensive
Behavioral Analytics	✗	✓

The comparison clearly demonstrates that the proposed framework provides several additional capabilities beyond those available in traditional online coding environments. These enhancements enable instructors to evaluate both programming competence and student engagement within a single platform.

System Effectiveness

The implementation results indicate that the proposed framework successfully achieved the objectives defined during the design phase. The secure virtual coding environment functioned reliably throughout testing, while the integrated monitoring components continuously collected behavioral data without affecting the usability of the coding editor.

The browser-based monitoring mechanisms accurately detected events such as copy-paste attempts, tab switching, window focus changes, typing activity, and idle periods. The collected information was successfully processed to generate meaningful Focus Scores and Integrity Scores for each coding session. These scores provided instructors with additional insights into student participation and learning behavior that cannot be obtained through traditional code evaluation alone.

The AI-assisted code review module also contributed significantly to the overall effectiveness of the system. Instead of simply indicating whether a program was correct or incorrect, the module generated constructive feedback regarding programming logic, readability, modularity, and coding standards. Such personalized feedback encourages students to improve their programming practices and supports continuous learning beyond laboratory assessments.

The teacher dashboard further enhanced system effectiveness by presenting behavioral analytics, activity timelines, submission reports, and student performance summaries through an intuitive interface. This reduced the workload associated with manual supervision while improving the transparency of laboratory evaluations.

Educational Impact

The proposed framework has the potential to improve programming education in several ways. First, continuous behavioral monitoring encourages students to actively participate in coding sessions rather than relying on copied solutions. Since engagement is evaluated alongside technical performance, students are motivated to spend more time understanding programming concepts and solving problems independently.

Second, instructors benefit from objective behavioral evidence that supports fairer assessment decisions. Instead of relying solely on the final submitted code, teachers can review coding activity throughout the laboratory session, enabling a more accurate evaluation of student effort and participation.

Third, AI-generated programming feedback promotes self-directed learning. Students receive immediate suggestions regarding code quality and logical improvements, allowing them to identify mistakes and refine their programming skills without waiting for manual instructor feedback.

Finally, the generated behavioral analytics can assist educational institutions in identifying learning patterns across different classes. Aggregate reports may reveal topics that consistently produce high idle times or low engagement scores, enabling instructors to redesign laboratory exercises and improve teaching strategies.

Limitations of the Proposed System

Although the proposed framework demonstrates several advantages, certain limitations remain. The monitoring mechanisms operate entirely within the web browser and therefore cannot prevent all forms of academic misconduct. Students may still use external devices such as mobile phones or secondary computers to access programming resources without being detected.

The webcam module implemented in the current system performs only basic presence detection and does not include advanced gaze tracking, facial expression analysis, or eye movement monitoring. This design decision was made to preserve user privacy while minimizing computational complexity.

Furthermore, browser restrictions prevent the system from implementing operating system-level security measures such as application blocking or complete lockdown functionality. Consequently, the framework should be viewed as a behavioral monitoring and assessment tool rather than a replacement for dedicated online examination software.

The AI-generated code review is also dependent on the quality of the underlying language model. Although the generated feedback is generally useful for educational purposes, instructors should continue to review complex programming submissions where necessary.

Discussion

The results obtained from the implementation of CodeSense AI demonstrate that combining behavioral analytics with programming assessment can significantly improve the transparency and effectiveness of virtual coding laboratories. Traditional evaluation methods focus almost exclusively on program output, whereas the proposed framework captures the complete learning process by analyzing how students interact with the coding environment throughout the laboratory session.

The integration of Focus Score and Integrity Score represents an important contribution because it allows behavioral data to complement technical programming evaluation rather than replace it. These metrics should be interpreted as indicators of engagement and authenticity that support instructor decision-making instead of acting as automated judgments of student performance.

The AI-assisted code review module further strengthens the educational value of the framework by providing personalized feedback immediately after submission. This transforms the coding laboratory into a more interactive learning environment where assessment and learning occur simultaneously.

The teacher analytics dashboard serves as an effective decision-support system by presenting detailed behavioral information in an organized and interpretable format. Instructors can quickly identify students requiring additional guidance while reducing the effort associated with continuous manual supervision.

Overall, the implementation confirms that the proposed framework successfully addresses the research problem identified in this study. By integrating secure coding, behavioral monitoring, AI-assisted evaluation, and comprehensive analytics, CodeSense AI provides a practical solution for improving fairness, transparency, and effectiveness in modern virtual programming laboratories. The framework demonstrates strong potential for adoption in universities and technical institutions seeking data-driven approaches to programming education and laboratory assessment.

5. Conclusion

This research presented CodeSense AI, an explainable behavioral analytics framework for virtual coding laboratories that integrates AI-based focus monitoring with programming skill assessment. The proposed system addresses the limitations of existing online coding platforms by combining a secure coding environment with continuous behavioral monitoring, AI-assisted code review, and a comprehensive teacher analytics dashboard. Unlike traditional systems that evaluate only the final program submission, CodeSense AI provides a holistic assessment by considering both technical performance and student engagement throughout the coding session.

The implementation results demonstrate that the framework successfully monitors key behavioral metrics such as typing activity, idle time, tab switching, copy-paste attempts, and webcam-based presence detection. These metrics are analyzed to generate Focus Score and Integrity Score, enabling instructors to better understand student participation and assess the authenticity of programming laboratory activities. In addition, the AI-assisted code review module provides constructive feedback that helps students improve code quality and programming practices.

The proposed framework enhances the transparency, fairness, and effectiveness of programming laboratory assessment while reducing the monitoring workload for instructors. By combining behavioral analytics with AI-driven evaluation, CodeSense AI promotes genuine coding practice and supports data-driven educational decision-making. Future work may include advanced gaze tracking, secure code execution sandboxes, AI-based plagiarism detection, multi-language programming support, and live instructor monitoring to further improve the capabilities of the system.

References

[1] M. Fowler, *Refactoring: Improving the Design of Existing Code*, 2nd ed. Boston, MA, USA: Addison-Wesley, 2018.

[2] R. S. Pressman and B. R. Maxim, *Software Engineering: A Practitioner's Approach*, 9th ed. New York, NY, USA: McGraw-Hill, 2020.

[3] I. Sommerville, *Software Engineering*, 10th ed. Boston, MA, USA: Pearson, 2016.

[4] T. H. Cormen, C. E. Leiserson, R. L. Rivest, and C. Stein, *Introduction to Algorithms*, 4th ed. Cambridge, MA, USA: MIT Press, 2022.

[5] M. Sipser, *Introduction to the Theory of Computation*, 3rd ed. Boston, MA, USA: Cengage Learning, 2013.

[6] D. Jurafsky and J. H. Martin, *Speech and Language Processing*, 3rd ed. (Draft). Stanford University, 2025.

[7] S. Russell and P. Norvig, *Artificial Intelligence: A Modern Approach*, 4th ed. Hoboken, NJ, USA: Pearson, 2021.

[8] OpenAI, “GPT-4 Technical Report,” 2023.

[9] Google DeepMind, “Gemini: A Family of Highly Capable Multimodal Models,” 2023.

[10] Microsoft, “Monaco Editor Documentation,” 2025.

[11] W3C, “Page Visibility API Specification,” World Wide Web Consortium, 2022.

[12] W3C, “Clipboard API and Events,” World Wide Web Consortium, 2024.

[13] TensorFlow, “TensorFlow.js Documentation,” Google, 2025.

[14] Next.js Team, “Next.js Documentation,” Vercel Inc., 2025.

[15] React Team, “React Documentation,” Meta Platforms Inc., 2025.

[16] PostgreSQL Global Development Group, “PostgreSQL Documentation,” Version 17, 2025.

[17] Tailwind Labs, "Tailwind CSS Documentation," Version 4, 2025.

[18] Mozilla Developer Network (MDN), "Web APIs for Browser Event Monitoring," 2025.

[19] T. F. Bissyandé, D. Kim, J. Klein, M. Monperrus, Y. Le Traon, "A Survey of Automated Program Repair," *ACM Computing Surveys*, vol. 51, no. 1, pp. 1–37, 2018.

[20] M. L. Hilton, T. Barnes, N. Murphy, D. McCauley, and S. Sanders, "Student Engagement in Programming Courses: A Review of Learning Analytics Approaches," *IEEE Transactions on Learning Technologies*, vol. 15, no. 4, pp. 421–435, 2022.