Journal of Information Systems Engineering and Management
2026, 11(1)e-ISSN: 2468-4376
https://www.jisem-journal.com/ Research Article

New Approaches to Computer Science Assessment in the AI Age

Eric Howard[1,2], Hardique Dasore[1], Shah Haque[1], Raddhika Kuttala[1], Mohamad Mahmoud Al Zein[1]

[1]Southern Cross Institute
[2]Macquarie University

ARTICLE INFO: Received: 01 December 2025 Revised: 04 January 2026 Accepted: 12 January 2026

Abstract

The rapid advancement of Artificial Intelligence (AI) has introduced both opportunities and challenges in higher education, particularly in computer science assessment. AI-generated content from tools like ChatGPT has diminished the effectiveness of traditional plagiarism detection systems, necessitating a shift in evaluation strategies. This paper proposes a comprehensive rethinking of assessment methodologies by leveraging free and accessible online platforms that require students to share unique, verifiable links to their work. By incorporating platforms such as Replit, SQLFiddle, Google Cloud Shell, TryHackMe, and Cisco Packet Tracer, among others, educators can promote hands-on learning, improve academic integrity, and foster deeper conceptual understanding. These innovative approaches ensure that assessments measure genuine student effort and proficiency rather than their ability to replicate AI-generated responses.

1 Introduction

Artificial Intelligence (AI) is reshaping education, influencing both teaching methodologies and assessment practices. The rise of AI-powered tools like ChatGPT and code generators has significantly altered how students approach learning and completing assignments. While these tools provide immense benefits in accessibility and automation, they pose substantial challenges concerning academic integrity and traditional assessment models.

The ability of AI to generate human-like text and solve complex programming problems raises concerns about plagiarism, authenticity, and the effectiveness of conventional evaluation systems. Many current plagiarism detection tools struggle to distinguish between human-written and AI-generated content, making traditional essay and coding assignments increasingly unreliable for assessing students' true capabilities. It is often hard to tell whether students are simply copying the answers or actually understanding the material, especially when they use AI-generated materials. In addition, reliance on AI-assisted answers diminishes critical thinking, problem solving, and hands-on application, key competencies in computer science education. This leads to a rising gap between theoretical understanding and real-world experience that promotes a reconsideration of assessment design.

The challenges with the traditional assessments are :

- Traditional exams lose their effectiveness when AI tools produce nearly flawless answers
- AI-generated responses can bypass plagiarism detection technologies
- AI-generated materials are used by the students without showing that they truly understand the ideas
- Although AI systems can generate accurate responses, they do not support students in developing their problem-solving abilities.
- When AI technologies are used excessively, student submissions lose their individuality, which makes it more difficult to evaluate each student's progress in learning.

To address these issues, educators must shift towards assessment methods that emphasize practical engagement, unique student input, and direct interaction with tools and platforms that require hands-on execution. One promising solution is leveraging online platforms that allow students to generate unique

workspaces, perform live coding, and share verifiable links to their completed work. Such methods ensure transparency, originality, and demonstrable understanding of computer science concepts.

Additionally, an effective assessment model should promote collaboration, real-world problem-solving, and experiential learning. By utilizing interactive platforms such as Replit, SQLFiddle, Webminal, TryHackMe, and Cisco Packet Tracer, students are encouraged to actively engage with the subject matter rather than passively reproducing AI-generated responses. These platforms enable instructors to track student progress, validate the authenticity of submissions, and facilitate a more comprehensive understanding of each learner's capabilities. Peer-reviewed projects, open-ended problem-solving tasks, and oral exams can all be added to assessments to increase their efficiency and make sure that students understand and apply theoretical ideas in relevant ways.

This paper proposes a framework for integrating free, accessible, and verifiable online tools into computer science assessments, ensuring that students develop essential computational thinking skills and maintain academic integrity. The following sections discuss various domains within computer science and propose effective strategies to redesign assessments using these digital tools.

With the rise of AI-driven text generation, students increasingly rely on automated tools to complete assignments. This trend undermines learning objectives, particularly in computer science courses where problem-solving and application are crucial. Instead of AI detection software, which has proven unreliable, educators should reconsider assessment methods to prioritize genuine student engagement and deeper understanding.Educators can design tests that are more resilient to AI-driven shortcuts while reiterating essential learning objectives by incorporating dynamic problem solving activities and adaptive learning technology.

2 Proposed Solutions

With rapid advancements in technology and the increasing role of Artificial Intelligence (AI) in education, traditional assessment methods are no longer sufficient in evaluating students' genuine understanding and problem-solving skills. Assessments need to be redesigned to focus on hands-on experiences, critical thinking, and verifiable work submissions to maintain academic integrity and encourage deeper engagement with learning material.

Current assessment models, which rely on written reports and static code submissions, fail to reflect the dynamic nature of real-world computing environments. Students often resort to AI-generated content or reusing solutions from past submissions, reducing the authenticity of their work. To address this, the implementation of interactive, cloud-based platforms where students can execute and share their work in a verifiable manner is crucial. These approaches allow educators to track individual contributions, ensure originality, and assess students based on their actual engagement and understanding.

One of the key aspects of innovative assessment methods is utilizing live coding and real-time problem-solving platforms. Tools such as Replit, JSFiddle, and Google Cloud Shell allow students to write, execute, and share their code with unique links that verify authenticity. These platforms maintain version histories, preventing students from submitting AI-generated code without modifications. Similarly, structured environments for databases like SQLFiddle and DB-Fiddle help students demonstrate their ability to construct, query, and manipulate data in real-time rather than merely submitting static SQL scripts.

Another essential component of modern assessment is leveraging cybersecurity and networking simulations. Platforms such as TryHackMe and Hack The Box enable students to engage in hands-on security exercises, where they can complete penetration testing challenges and document their findings through shared progress links. Additionally, networking tools like Cisco Packet Tracer and Eve-NG allow students to design, simulate, and troubleshoot complex network infrastructures while generating unique session identifiers to validate their work.

In Linux system administration courses, students often need to demonstrate their proficiency in configuring and managing servers. By integrating cloud-based Linux terminals such as Webminal and JupyterHub, students can execute shell commands in an isolated environment and provide instructors with traceable links to their completed tasks. This ensures that their work is original and reflective of their individual understanding.

Beyond hands-on coding exercises, effective assessment strategies should also emphasize peer collaboration and project-based learning. Encouraging students to engage in team-based coding projects, group troubleshooting exercises, and real-world problem-solving simulations fosters critical thinking and mimics industry-standard workflows. Collaborative learning environments like GitHub Classroom and CoCalc provide mechanisms for students to submit work, track progress, and receive real-time feedback from peers and instructors.

Moreover, to ensure the security and authenticity of academic submissions, assessments should integrate secure documentation practices such as PDF flattening. This method prevents students from making post-submission modifications to their work. Open-source tools like Ghostscript enable students to convert reports, configurations, and screenshots into non-editable formats, ensuring transparency and preventing academic dishonesty.

By adopting these innovative approaches, computer science educators can create a more robust, engaging, and integrity-driven assessment model. These solutions promote genuine learning, real-world applicability, and verifiable assessments, ultimately better preparing students for careers in technology and computer science. Traditional assessment methods in computer science education often rely on written assignments, static code submissions, and theoretical problem-solving. However, these approaches fail to capture the hands-on, applied nature of the field. To foster genuine learning, new strategies must emphasize active engagement, originality, and the use of interactive tools that require students to generate and share unique solutions. The following subsections explore specific methods for key computer science disciplines, leveraging free online platforms to enhance assessment validity and integrity.

One of the fundamental challenges in assessing programming and technical skills is ensuring students demonstrate their own work rather than copying from external sources, including AI-generated solutions. This necessitates a shift toward assessments that involve real-time coding exercises, problem-solving tasks in collaborative environments, and the use of cloud-based platforms that track individual progress. By requiring students to submit shareable links from platforms like Replit, SQLFiddle, and Webminal, instructors can ensure that students engage in meaningful work and demonstrate their problem-solving processes.

For cybersecurity and networking courses, hands-on practice is essential. The use of virtual labs, emulated networks, and cybersecurity challenges ensures that students gain practical experience while providing verifiable evidence of their work. Platforms such as TryHackMe, Hack The Box, and Cisco Packet Tracer allow students to document their processes and submit unique task links, making it easier to assess their individual contributions and problem-solving capabilities.

A broader pedagogical shift is also necessary in database management, Linux administration, and system security assessments. Instead of static documentation, assessments should focus on interactive database queries, cloud-based Linux command execution, and real-world problem-solving scenarios that require dynamic engagement with learning materials. Encouraging students to use free platforms that generate session-based, shareable workspaces provides a scalable and effective way to ensure the authenticity and originality of their submissions.

This paper advocates a multi-faceted approach to assessment, integrating real-world simulations, interactive problem-solving tasks, and collaborative learning environments. These approaches not only mitigate the risks of AI-generated plagiarism but also align assessment methods with industry practices, preparing students for professional careers in computer science. The following subsections present detailed recommendations for redesigning assessments in various domains of computer science education.

This paper suggests innovative methods for conducting assessments in key computer science domains:

2.1 Programming Assignments

One of the major challenges in programming education today is ensuring that students produce original work and do not rely on AI-generated solutions from tools such as ChatGPT. Traditional code submission formats, such as uploading scripts or submitting code snippets in written documents, make it difficult for educators to verify whether students genuinely engaged in problem-solving or simply copied pre-written solutions. To address this, a new assessment paradigm is required—one that ensures students actively participate in coding exercises and produce verifiable work.

To counteract the issue of AI-generated code plagiarism, educators should require students to use cloud-based coding platforms that generate unique shareable links for their work. These platforms provide transparency by allowing instructors to track student progress, monitor code development over time, and verify that students have personally written their solutions. Version control features in platforms such as Replit, JSFiddle, and CodePen make it possible to analyze students' iterative problem-solving approaches, providing insights into their thought processes and debugging skills.

Another advantage of using these platforms is their ability to prevent direct copy-pasting from external sources. Many online coding environments include auto-tracking functionalities that log every keystroke and prevent blind submission of AI-generated responses. By requiring students to write and test their code directly within a controlled environment, educators can ensure that learning outcomes are met and that students truly understand the concepts being taught.

Beyond basic programming exercises, instructors can integrate interactive, real-time coding assignments that mimic real-world development workflows. Pair programming assessments, live coding challenges, and debugging tasks within cloud-based environments promote engagement and discourage passive reliance on AI-generated answers. By incorporating collaborative elements such as peer code reviews and group projects, students gain valuable industry-relevant experience while reinforcing their understanding of core programming principles.

In addition to enforcing originality, these platforms provide scalability for instructors managing large classes. Automated testing frameworks built into these environments allow for instant feedback and grading, reducing the administrative burden of manual code evaluation. Moreover, integrating oral code explanations or written reflections on coding approaches can further solidify students' comprehension and prevent reliance on pre-generated AI responses.

Ultimately, shifting programming assessments from static submissions to cloud-based, interactive environments ensures academic integrity, fosters active learning, and aligns with industry best practices. The following platforms offer suitable environments for implementing such assessment methods:

Traditional programming assignments often rely on static code submissions, making it difficult to assess a student's actual engagement and problem-solving process. Instead of submitting individual code files, students should be required to use interactive coding platforms that generate unique shareable links, ensuring that their work is both original and verifiable. By implementing this approach, educators can enhance transparency in assessments and encourage students to develop coding skills in a real-world setting.

One of the key advantages of using online coding platforms is the ability to track a student's progress in real time. Many of these platforms maintain version histories, allowing instructors to see how a student approached the problem, debugged errors, and refined their solutions over time. This provides deeper insight into their learning process beyond just evaluating the final output. Additionally, features such as collaborative coding, peer reviews, and instructor feedback can be integrated seamlessly into the workflow, making assessments more interactive and engaging.

A critical component of programming education is the ability to write and test code across different environments. Online coding platforms like Replit, JSFiddle, and Google Cloud Shell enable students to execute code in the cloud without the need for complex local configurations. These platforms support multiple programming languages and frameworks, allowing students to experiment with various coding paradigms while working on their assignments. Furthermore, the ability to share live coding sessions enables educators to conduct in-class coding exercises and live debugging sessions, fostering a more interactive learning experience.

To ensure fairness and prevent plagiarism, platforms that generate unique session links and track user activity should be prioritized. This mitigates the risks of students copying AI-generated solutions without fully understanding the logic behind the code. Additionally, requiring students to explain their code through comments, video walkthroughs, or written explanations further strengthens the assessment model by verifying their comprehension.

The following platforms provide an effective environment for programming assignments: Instead of traditional code submissions, students should be required to use platforms that generate unique shareable links for their work. These platforms ensure originality and transparency in coding assessments.

- **Replit:** Enables students to share unique coding project links, ensuring personal accountability.
- **JSFiddle / CodePen:** For web development assignments, students submit live, working code rather than static submissions.
- **Trinket:** A browser-based environment for interactive coding that allows sharing unique project links.
- **Sololearn:** Provides interactive coding lessons with shareable code snippets.
- **Google Gemini Code Assist:** An AI-powered coding tool supporting multiple programming languages, with shareable code snippets.
- **JDoodle:** An online compiler supporting over 88 programming languages, enabling execution and sharing through unique URLs.

3 Database Queries and Designs

Ensuring academic integrity in database-related assessments is crucial for evaluating students' actual proficiency in Structured Query Language (SQL) and database design concepts. Traditional submission methods, such as static query results or screenshots, are easily manipulated or generated by AI tools, making it difficult to assess genuine student effort. To counteract this, educators should leverage online database platforms that generate unique, shareable links to students' query executions. These platforms enable students to work on live database environments, fostering hands-on learning, collaboration, and verifiable assessments.

By requiring students to use interactive database platforms, instructors can ensure that assessments emphasize problem-solving and query optimization rather than memorization. These tools allow students to experiment with different SQL commands, visualize results dynamically, and troubleshoot errors in real time. Furthermore, students can submit their assignments by sharing unique session URLs, allowing instructors to verify their work efficiently.

The following free online platforms provide effective solutions for conducting database-related assessments:

- **SQLFiddle:** A free online tool that enables students to write, test, and execute SQL queries across multiple database engines, including MySQL, PostgreSQL, and SQLite. The platform generates unique URLs for each query session, ensuring transparent and accountable assessments.
- **DB-Fiddle:** Similar to SQLFiddle, DB-Fiddle supports MySQL and PostgreSQL databases and provides an interactive interface where students can create, modify, and execute queries. The platform allows students to share unique query links with instructors for real-time feedback.
- **Mode Studio:** An advanced SQL-based platform that integrates data visualization tools with SQL querying. Students can execute cloud-based queries and generate shareable project links for submission.
- **Google Cloud BigQuery:** A powerful cloud-based SQL execution platform that allows students to work with large datasets. Assignments can be structured around real-world data analysis, with students submitting shareable query execution links.
- **DBVisualizer:** A versatile database tool that supports SQL execution and query optimization. It allows students to connect to various database servers and submit shareable outputs of their executed queries.
- **SQLite Online:** A lightweight, browser-based SQL execution environment where students can create databases, write queries, and generate unique session links for verification.
- **HackerRank for SQL:** A structured SQL assessment platform where students can solve challenges, receive instant feedback, and share progress links. The system tracks students' performance over time, making it a valuable tool for formative assessments.

By integrating these online platforms into database-related assessments, educators can promote hands-on learning, ensure originality, and encourage students to develop problem-solving skills in SQL and database design. This shift away from traditional assessments enhances student engagement and prepares them for real-world database management challenges.

4 Linux System Administration

Linux system administration is a fundamental aspect of computer science education, requiring students to develop hands-on skills in system management, command-line operations, networking, and security configurations. Traditional assessments often involve written explanations or static screenshots of command outputs, which can be easily manipulated. To ensure academic integrity and accurately evaluate students' practical skills, assessments should utilize cloud-based Linux environments that generate unique session links, allowing instructors to verify and assess student work dynamically.

Interactive Linux environments provide students with an opportunity to execute real commands, troubleshoot issues, and document their processes. These platforms also support collaborative learning by enabling students to share their sessions, seek peer feedback, and receive instructor evaluations in real-time.

By leveraging free cloud-based tools, students can gain real-world experience in system administration without requiring complex local setups.

The following platforms offer effective solutions for conducting Linux system administration assessments:

- **Google Cloud Shell:** A cloud-based Linux shell that provides a persistent command-line interface for managing cloud resources, executing scripts, and configuring system settings. Students can share their session logs for assessment.
- **JupyterHub:** A multi-user Jupyter notebook environment that allows students to run shell commands, execute Python scripts, and document their work interactively. Notebooks can be shared with unique URLs, ensuring accountability.
- **Webminal:** An online Linux terminal that enables students to practice shell commands, write scripts, and perform system configurations in a secure, browser-based environment. Each session generates a unique link for submission.
- **CoCalc:** A cloud-based computational environment that includes a Linux terminal, scripting tools, and collaborative document editing. Students can work on assignments in real-time and share their progress via unique session links.
- **Katacoda:** An interactive learning platform that provides pre-configured Linux environments for practicing system administration tasks. Students can complete guided exercises and submit session links as proof of their work.
- **JS Linux:** A lightweight, web-based Linux emulator that allows students to run command-line operations in a simulated Linux environment. Each session can be shared, allowing for remote verification of work.
- **Terminus Web Terminal:** A browser-based terminal that supports SSH connections, allowing students to remotely access and manage Linux servers while generating shareable session logs.
- **LinuxZoo:** A virtual Linux environment that enables students to practice system administration tasks, including package management, user account creation, and network configurations. Each user receives a unique instance, ensuring individualized assessments.

By incorporating these cloud-based Linux environments into assessments, educators can ensure students develop practical system administration skills while maintaining academic integrity. These platforms provide instructors with the ability to track command history, verify session authenticity, and assess students based on their hands-on experience rather than static documentation.

5 Cybersecurity and Information Security

Practical, hands-on experience is critical in cybersecurity education, as theoretical knowledge alone is insufficient for understanding real-world security threats and mitigation strategies. Traditional written assessments fail to capture the complexity of penetration testing, vulnerability assessment, digital forensics, and incident response. Instead, students should engage with live security labs and simulations that provide unique, verifiable session links for their work.

By utilizing interactive cybersecurity environments, students can develop the skills necessary to protect networks, analyze security threats, and apply ethical hacking techniques under controlled conditions. The following platforms provide effective solutions for hands-on cybersecurity and information security assessments:

- **TryHackMe:** A guided cybersecurity training platform that offers hands-on exercises in ethical hacking, penetration testing, and digital forensics. Students can complete challenges in sandboxed environments and submit unique progress links for evaluation.
- **Hack The Box:** A penetration testing platform where students can explore real-world security challenges, exploit vulnerabilities, and conduct network assessments. Each user receives unique access credentials, ensuring individual work is verifiable.

- **Parrot Security OS:** A security-focused Linux distribution equipped with penetration testing tools, cryptography utilities, and forensic analysis features. Students can conduct security assessments in isolated virtualized environments and generate detailed reports.
- **MISP (Malware Information Sharing Platform):** A collaborative threat intelligence-sharing framework that allows students to analyze malware data, investigate cybersecurity incidents, and generate unique session logs for submission.
- **CyberSecLab:** A virtual cybersecurity lab that provides simulated attack scenarios where students can practice ethical hacking techniques, malware analysis, and security hardening.
- **Blue Team Labs Online (BTLO):** A defensive cybersecurity training platform where students can engage in digital forensics, log analysis, and threat hunting challenges. Unique lab access links allow instructors to verify individual progress.
- **RangeForce:** A cloud-based cybersecurity training environment that simulates real-world security incidents, including malware detection, network defense, and intrusion response.
- **Cyborg Security Training Platform:** A comprehensive security training environment that provides hands-on experience with threat hunting, SIEM (Security Information and Event Management) analysis, and cyber defense strategies.
- **OpenSOC.io:** A live security operations center (SOC) simulation where students can analyze cybersecurity threats and respond to real-world attack scenarios using industry-standard security tools.
- **LetsDefend.io:** A cloud-based security operations training environment where students practice security monitoring, threat detection, and incident response. Instructors can track student progress via unique session URLs.

By integrating these cybersecurity platforms into assessments, educators can ensure students develop real-world security skills while maintaining academic integrity. These tools allow instructors to verify student engagement, track activity logs, and assess practical problem-solving abilities in a secure, controlled environment.

5.1 Networking Assignments

Networking courses should shift from theoretical assessments to interactive simulations where students share unique links to their configurations.

- **Cisco Packet Tracer:** Students save and share unique simulation files.
- **Eve-NG:** Provides shareable network emulation sessions.
- **GNS3:** Allows students to export and submit live network configurations.

6 General Computer Science Education Tools

As computer science education advances, the need for dynamic and interactive learning tools has never been greater. Traditional methods, such as textbooks and recorded lectures, often fall short in engaging students or equipping them with the hands-on experience needed for real-world applications. By integrating interactive online platforms, educators can transform learning into an immersive experience that promotes critical thinking, problem-solving, and a deeper grasp of fundamental computer science concepts.

One of the key benefits of online learning tools is the flexibility they offer. Students can progress at their own pace, mastering concepts through structured lessons and coding exercises that increase in complexity. Unlike passive learning methods, these platforms provide real-time feedback, allowing learners to experiment, troubleshoot errors, and refine their computational thinking skills—an essential foundation for programming and software development.

Collaboration is another powerful advantage of digital learning environments. Many coding platforms feature peer reviews, group projects, and instructor feedback, fostering a community-driven approach that mirrors real-world software development. Engaging in collaborative coding helps students develop teamwork skills, gain diverse perspectives, and prepare for the highly interactive nature of the tech industry.

Educators, in turn, can track student progress through interactive dashboards, offering personalized guidance to enhance learning outcomes.

Beyond programming, online platforms extend their reach to various computer science fields, including cybersecurity, networking, databases, and system administration. Virtual labs allow students to engage in hands-on exercises—configuring networks, managing databases, and securing systems against cyber threats. These realistic simulations provide practical experience that directly translates into professional competency.

Perhaps most importantly, these digital tools make computer science education more accessible and inclusive. With many platforms offering free or low-cost access, financial barriers are reduced, opening doors for a broader and more diverse group of learners. Cloud-based solutions further enhance accessibility by eliminating the need for expensive hardware, enabling students to learn from anywhere with an internet connection.

By embracing interactive and adaptive learning tools, educators can revolutionize computer science education—making it more engaging, practical, and widely available to learners of all backgrounds. The future of tech education lies in fostering an environment where students don't just consume information but actively create, collaborate, and problem-solve in meaningful ways.

To ensure that assessments reflect genuine student efforts and mastery of concepts, these tools incorporate shareable project links and unique session IDs. This prevents academic dishonesty and makes it easier for instructors to verify the authenticity of submissions. By leveraging these technologies, educators can create more effective and fair evaluation methods that encourage students to actively engage with their coursework.

The following platforms exemplify some of the best free resources available for general computer science education:

Technology in computer science education is evolving rapidly, necessitating the adoption of interactive platforms that facilitate hands-on learning. Many free tools provide comprehensive learning experiences, allowing students to practice coding, explore algorithms, and engage in problem-solving exercises while ensuring verifiability of their work through shareable links. These tools are instrumental in reinforcing foundational and advanced computer science concepts in an engaging and interactive manner.

To maintain the integrity of digital submissions, PDF flattening plays a crucial role. Many assignments require screenshots, reports, and documentation of system configurations, which can be altered if submitted in an editable format. By converting documents into non-editable static images using open-source tools like Ghostscript, instructors can ensure that submissions remain unchanged. This approach enhances security and ensures academic honesty by preserving the authenticity of student work.

One key advantage of these tools is their accessibility; students can log in with their Google accounts and store progress, share work, and collaborate with peers. This enables continuous learning outside the classroom and ensures that assignments reflect students' individual efforts. The following platforms offer structured lessons, practice environments, and real-time coding capabilities suitable for general computer science education:

- **Codecademy:** Provides interactive coding lessons across various programming languages and topics, with shareable progress links.
- **CS First by Google:** A computer science curriculum that allows students to create and share interactive projects.

7 Use of Virtualization

Virtualization has become an essential tool in modern computer science education, providing students with a controlled, flexible, and cost-effective environment to practice various computing tasks. Virtualization technology enables students to create and manage isolated environments that replicate real-world computing infrastructures, allowing them to experiment with system configurations, networking, software installations, and cybersecurity practices. Unlike traditional assessments that rely on theoretical knowledge or static screenshots, virtualization ensures that students demonstrate their hands-on skills through live, interactive environments.

A major advantage of virtualization is its ability to support diverse learning scenarios, including system administration, software development, cloud computing, and cybersecurity. Virtual machines (VMs) provide a sandboxed environment where students can install and configure operating systems, set up

networks, and experiment with security tools without affecting their local machines. This flexibility enables students to gain practical experience that aligns with industry standards and workplace expectations.

For networking and cybersecurity courses, virtualization platforms such as Eve-NG, GNS3, and Cisco Packet Tracer allow students to create complex network topologies and simulate real-world scenarios. Students can configure routers, switches, and firewalls, troubleshoot connectivity issues, and analyze network traffic. Instructors can assess students based on their ability to design and implement functional network infrastructures using these tools.

In system administration courses, virtualization solutions like VirtualBox and KVM/QEMU enable students to deploy and manage various operating systems, configure server environments, and troubleshoot system issues. These platforms allow for the replication of enterprise-level IT environments where students can install software, manage user accounts, configure security policies, and monitor system performance. Instructors can validate student work through VM snapshots, logs, and unique session links.

For cloud computing and DevOps-related courses, virtualization plays a crucial role in familiarizing students with containerization and cloud platforms. Tools such as Docker, Kubernetes, and Google Cloud Shell provide students with practical exposure to cloud-native technologies, infrastructure automation, and containerized application deployment. By integrating these tools into assessments, students can gain hands-on experience with modern cloud infrastructure and continuous integration/continuous deployment (CI/CD) workflows.

Additionally, virtualization fosters collaboration and remote learning by allowing students to share their virtual environments with instructors and peers. Cloud-based solutions such as CoCalc and JupyterHub enable students to work on shared instances, execute Linux commands, and write scripts in a collaborative environment. These platforms generate unique session links, making it easier for instructors to track individual contributions and provide feedback.

By incorporating virtualization into computer science assessments, educators can ensure that students acquire practical, verifiable skills that extend beyond theoretical knowledge. The ability to create, configure, and manage virtual environments provides students with a deeper understanding of computing concepts and prepares them for real-world challenges in IT and software development. The following platforms offer robust virtualization environments suitable for academic assessments:

In modern computer science education, practical assessments require hands-on experience with virtualization and documentation tools. Free virtualization solutions allow students to create, manage, and experiment with different computing environments without the need for expensive hardware. Furthermore, securing and verifying submissions through PDF flattening ensures document authenticity and prevents unauthorized modifications.

Virtualization platforms enable students to set up and configure systems for testing and development. These tools support operating system installation, networking experiments, and system administration tasks. Free and open-source solutions such as VirtualBox and KVM/QEMU provide robust environments for running virtual machines, allowing students to demonstrate their understanding of system configurations and security practices.

Additionally, in cybersecurity courses, students need to document their investigative processes, configurations, and results from penetration testing exercises. Virtualization platforms enable them to create isolated environments for testing vulnerabilities, practicing ethical hacking techniques, and configuring network security policies. These tasks can be verified by instructors through shareable virtual machine snapshots or detailed reports.

By integrating free virtualization tools and secure documentation practices, computer science programs can offer more effective and verifiable assessments. These strategies reinforce hands-on learning, provide students with real-world system administration experience, and uphold academic integrity.

For system administration and cybersecurity assessments, students should provide evidence of their work using free virtualization tools.

- **VirtualBox:** A free alternative to VMware for setting up virtual environments.
- **KVM/QEMU:** A Linux-based virtualization tool for advanced system administration tasks.

8 Web Development and Web Design

Web development and web design are essential components of computer science education, particularly in front-end programming, responsive layout, and user interface (UI) design. Traditional submission formats—such as zipped HTML/CSS/JS files or screenshots—lack interactivity, offer limited transparency, and are prone to academic dishonesty through AI-generated or copied code.

To address these challenges, assessments should incorporate cloud-based development platforms that allow students to create and submit live, interactive websites using shareable URLs. These platforms enable instructors to review version histories, inspect real-time outputs, and verify student work directly in the browser, thereby promoting authenticity and deep engagement.

Recommended platforms for live, interactive web development include:

- **CodePen:** A live HTML/CSS/JavaScript environment where students can build and share fully interactive "pens" using unique URLs.
- **JSFiddle:** Ideal for short, testable HTML/CSS/JS snippets. Every fiddle generates a public link for submission and feedback.
- **W3Schools Tryit Editor:** Allows students to experiment with front-end code using built-in tutorials and share their work through generated links.
- **Glitch:** Supports full-stack web apps with collaborative editing. Students can remix starter templates and submit hosted URLs for assessment.
- **Trinket:** An easy-to-use coding platform for front-end and Python web apps. Each project has a shareable link for embedding or submission.
- **GitHub Pages:** Enables students to host static websites directly from their GitHub repositories, complete with version control.

Free Hosting Platforms with Custom Subdomains

For final submissions or portfolio projects, students should be encouraged to host their websites using free static hosting platforms that provide custom subdomains. These tools allow students to publish their work publicly and give instructors permanent, verifiable links for assessment.

- **Tiiny.host:** A simple drag-and-drop static web host that gives each upload a unique subdomain (e.g., `studentproject.tiiny.site`). Ideal for quick hosting of HTML/CSS/JS projects.
- **Netlify:** Provides continuous deployment from Git repositories and generates a custom subdomain (e.g., `student-project.netlify.app`) for each site.
- **Vercel:** Supports front-end frameworks like React, Next.js, and static sites. Students receive unique subdomains (e.g., `username.vercel.app`) that are publicly accessible.
- **GitHub Pages:** Offers free hosting for static sites directly from a GitHub repository. Students can use URLs such as `username.github.io/project-name`, with full version tracking.
- **Render (static hosting):** Allows free deployment of static websites via GitHub integration with auto-generated subdomains.

These platforms not only verify student submissions through public access but also teach deployment workflows commonly used in the tech industry. Encouraging students to publish their work fosters accountability, provides portfolio-ready deliverables, and reinforces the real-world relevance of classroom assignments.

Assessment tasks can be further enhanced by incorporating:

- Written reflections on design decisions and layout responsiveness.
- Responsive design testing (desktop vs mobile).
- Peer reviews and accessibility audits using tools like Lighthouse.

By leveraging these free, real-time development and deployment platforms, web design assessments can become interactive, verifiable, and aligned with industry practices, while also reinforcing academic integrity.

Journal of Information Systems Engineering and Management
2026, 11(1)e-ISSN: 2468-4376
https://www.jisem-journal.com/ Research Article

9 AI-Resilient Assessment Design: Principles, Evidence, Equity, and Future Directions

The emergence of generative Artificial Intelligence has fundamentally altered the relationship between assessment, authorship, and evidence in computer science education. Traditional assessment models that rely on static artefacts, written explanations, or final code submissions no longer provide sufficient assurance of genuine student engagement or conceptual understanding. In response, assessment design must move beyond output based evaluation and instead emphasize process visibility, execution level evidence, and demonstrable interaction with computing systems.

At the core of AI resilient assessment design is a shift toward process oriented evaluation. Rather than assessing correctness alone, instructors must evaluate how students arrive at solutions, how they respond to errors, and how they adapt their approaches in response to technical constraints. Cloud based development environments, virtual laboratories, online database platforms, and system administration sandboxes inherently support this shift by recording execution histories, configuration changes, version iterations, and session level activity. These artefacts form a verifiable trail of student engagement that cannot be replicated by generic AI generated responses.

Traceability plays a central role in maintaining academic integrity without resorting to unreliable AI detection mechanisms. When students submit shareable links to live environments, instructors can directly observe query executions, command histories, network configurations, or deployed applications within the context of a unique session. This form of evidence allows assessment decisions to be grounded in observable technical activity rather than textual similarity metrics or probabilistic detection scores. Importantly, traceability supports transparency and fairness, as all students are evaluated using the same evidence standards regardless of their writing style or prior exposure to AI tools.

Assessment rubrics within this framework should explicitly value iteration, debugging, and refinement. Marks can be allocated for documented failed attempts, alternative strategies explored, and reflective commentary tied directly to executed actions. This approach reframes error as an integral component of learning rather than a deficiency to be concealed. In contrast to AI generated outputs, which often present artificially polished solutions, authentic student work naturally reflects trial, adjustment, and progressive understanding.

Oral verification and reflective justification further strengthen assessment robustness when applied selectively. Short explanations of specific code segments, database queries, network configurations, or security findings can confirm authorship and conceptual understanding without imposing the burden of full oral examinations. Reflective components that require students to justify decisions, explain encountered challenges, and reference concrete execution outcomes are particularly resistant to outsourcing, as they must align precisely with artefacts visible in submitted environments.

Equity and accessibility are essential considerations in the adoption of platform based assessment models. Free, browser based tools reduce dependence on specialized hardware and ensure that students from diverse socioeconomic backgrounds can participate on equal terms. Cloud based environments also support asynchronous engagement, enabling students to work flexibly while still producing verifiable outputs. This flexibility is especially important for students balancing study with employment, caregiving responsibilities, or health related constraints.

From an ethical perspective, the proposed assessment model avoids invasive monitoring or surveillance practices. Integrity is maintained through artefact verification rather than behavioral tracking, preserving student privacy while ensuring accountability. This distinction is critical for maintaining trust between institutions and learners and aligns with broader ethical principles in educational technology adoption.

The implications of this assessment approach extend beyond individual units. At the curriculum level, consistent use of verifiable evidence types supports skill development across programs and simplifies moderation, accreditation, and quality assurance processes. External reviewers can directly inspect live artefacts, reducing ambiguity and increasing confidence in assessment outcomes. Institutional academic integrity policies should be updated to explicitly recognize execution based evidence and process documentation as valid assessment mechanisms, reducing reliance on contested AI detection technologies.

Professional development for academic staff is a necessary enabler of this transition. Educators require guidance in designing tasks that leverage interactive platforms effectively, interpreting process evidence, and constructing rubrics that reward authentic engagement. Shared exemplars, standardized evidence expectations, and coordinated assessment strategies can facilitate consistent implementation across disciplines.

Looking forward, assessment design must remain adaptive as AI tools continue to evolve. Rather than positioning AI solely as a threat, future assessment models may incorporate controlled AI usage as an

explicit component of learning. Tasks that require students to critique, modify, or extend AI generated outputs within live environments can transform AI into an object of analysis rather than a hidden shortcut. Longitudinal research is needed to evaluate whether students assessed through execution based, verifiable methods demonstrate stronger retention, deeper conceptual transfer, and improved workplace readiness compared to those evaluated using traditional submission models.

Ultimately, resilient assessment in computer science education is achieved not by attempting to eliminate AI influence, but by aligning evaluation methods with authentic learning processes. By emphasizing execution, traceability, reflection, and equity, institutions can uphold academic rigor while preparing students for professional practice in an AI mediated technological landscape.

10 Concluding remarks

By leveraging platforms such as Replit, SQLFiddle, Google Cloud Shell, TryHackMe, and Cisco Packet Tracer, institutions can transition to assessments that require active student participation. These tools provide unique shareable links that allow instructors to verify originality, track student progress over time, and promote deeper learning through hands-on experimentation. Furthermore, this shift moves beyond simple correctness-based grading to more comprehensive evaluations that consider problem-solving approaches, debugging skills, and iterative development processes.

The use of online interpreters, real-time coding platforms, and cloud-based system administration tools ensures that students interact with programming, cybersecurity, and networking concepts in a meaningful way. In addition, incorporating virtualization and secure documentation practices such as PDF flattening further strengthens the integrity of academic assessments by providing verifiable evidence of students' hands-on engagement.

Moreover, the integration of collaboration-focused environments, such as GitHub Classroom and CoCalc, prepares students for industry practices by encouraging teamwork and real-time feedback. The inclusion of real-world scenarios, interactive challenges, and shared learning experiences helps bridge the gap between academia and professional environments, equipping students with the skills needed for success in the workforce.

A key takeaway from this research is that academic integrity can no longer rely solely on plagiarism detection software. Instead, computer science educators must embrace innovative assessment models that demand active student involvement and verifiable submission methods. By focusing on problem-solving, collaboration, and hands-on experimentation, educators can foster an environment that encourages genuine learning while maintaining the integrity of academic work.

In conclusion, adapting computer science assessments to the modern technological landscape requires a fundamental shift in pedagogical approaches. The use of interactive and verifiable online tools presents an effective strategy to counteract AI-generated plagiarism while promoting critical thinking and authentic skill development. Institutions must actively work toward adopting these strategies to ensure that students graduate with practical experience, deep understanding, and the ability to apply their knowledge in real-world settings. Future research should explore the long-term impact of these methodologies and identify further enhancements to maintain academic rigor in an era of rapid technological advancement. A comprehensive rethinking of assessment in computer science education is necessary to adapt to modern technological advancements. Instead of relying on traditional written submissions, universities should encourage students to submit assessments through shareable links generated from online platforms, ensuring hands-on problem-solving, originality, and academic integrity.

References

[1] Ateeq, A., Alzoraiki, M., Milhem, M., & Ateeq, R. A. (2024). Artificial intelligence in education: implications for academic integrity and the shift toward holistic assessment. *Frontiers in Education*.

[2] Farag, W., Nadeem, M., & Helal, M. (2024). Assessment Transformation in the Age of AI: Moving Beyond the Influence of Generative Tools. *IEEE*.

[3] Lancaster, T. (2018). Academic Integrity for Computer Science Instructors. *Springer Book Chapter*.

[4] Steponenaite, A. (2023). Plagiarism in AI Empowered World. *Lecture Notes in Computer Science*

[5] Strik, B. H., Menolli, A., & Brancher, J. D. (2024). GPT AI in Computer Science Education: A Systematic Mapping Study. *SBIE Conference Proceedings*

[6] Hazzan, O., & Erez, Y. (2024). Generative AI in Computer Science Education.

[7] Luckin, R., Holmes, W., Griffiths, M., & Forcier, L. B. (2016). Intelligence Unleashed: An Argument for AI in Education. Pearson Education.

[8] Replit. (n.d.). The collaborative browser-based IDE. Retrieved April 15, 2025, from https://replit.com

[9] SQL Fiddle. (n.d.). Online SQL playground for testing and sharing SQL queries. Retrieved April 15, 2025, from http://sqlfiddle.com/

[10] TryHackMe. (n.d.). Learn cybersecurity practically. Retrieved April 15, 2025, from https://tryhackme.com/

[11] Cisco Networking Academy. (n.d.). Cisco Packet Tracer. Retrieved April 15, 2025, from https://www.netacad.com/courses/packet-tracer

[12] Webminal. (n.d.). Online Linux Terminal for learning and practicing commands. Retrieved April 15, 2025, from https://www.webminal.org/

[13] CoCalc. (n.d.). Collaborative calculation and data science environment. Retrieved April 15, 2025, from https://cocalc.com

[14] Tiiny.host. (n.d.). Drag and drop website hosting with custom subdomains. Retrieved April 15, 2025, from https://tiiny.host

[15] Netlify. (n.d.). Deploy modern websites and web apps. Retrieved April 15, 2025, from https://www.netlify.com

[16] Vercel. (n.d.). Frontend cloud platform for static and dynamic apps. Retrieved April 15, 2025, from https://vercel.com

[17] GitHub Pages. (n.d.). Host your website from a GitHub repository. Retrieved April 15, 2025, from https://pages.github.com