# E-Proctor: Redefining Secure Programming Examinations with an Intelligent Online Platform

**Ugwunna Charles Okechukwu** [1*]
**Akawuku Ifeanyi Godspower** [2]
**Chukwuogo Okechukwu Ejike**[2]
**Joshua John** [3]
**Adedapo Taiwo Obrien** [4]
**Orji Everistus Eze**[5]

Received: 05/١٢/2025
Revised: ١١/٠٢/2026
Accepted: 12/02/2026



[1] Department of Computer Science, College of Science and Computing, Wigwe University, Isiokpo, Rivers State, Nigeria.
[2] Department of Computer Science, Faculty of Physical Sciences, Nnamdi Azikiwe University Awka, Anambra State, Nigeria.
[3] Department of Computer Science, Faculty of Computing, Federal University of Applied Sciences Kachia, Kaduna State, Nigeria.
[4] Department of Computer Science, Faculty of Physical Sciences, Federal University of Agriculture Abeokuta, Ogun State, Nigeria.
[5] Department of Computer Science, School of Science and Technology, Federal Polytechnic Ohodo, Enugu State, Nigeria.
*Corresponding Author's Email: charles.ugwunna@wigweuniversity.edu.ng

# E-Proctor: Redefining Secure Programming Examinations with an Intelligent Online Platform

**Ugwunna Charles Okechukwu**
*Department of Computer Science, College of Science and Computing, Wigwe University, Isiokpo, Rivers State, Nigeria.*
charles.ugwunna@wigweuniversity.edu.ng

**Akawuku Ifeanyi Godspower**
*Department of Computer Science, Faculty of Physical Sciences, Nnamdi Azikiwe University Awka, Anambra State, Nigeria.*
gi.akawuku@unizik.edu.ng

**Chukwuogo Okechukwu Ejike**
*Department of Computer Science, Faculty of Physical Sciences, Nnamdi Azikiwe University Awka, Anambra State, Nigeria.*
oe.chukwuogo@unizik.edu.ng

**Joshua John**
*Department of Computer Science, Faculty of Computing, Federal University of Applied Sciences, Kachia, Ahmadu Bello University, Zaria, Kaduna State, Nigeria.*
jjoshua@abu.edu.ng

**Adedapo Taiwo Obrien**
*Department of Computer Science, Faculty of Physical Sciences, Federal University of Agriculture Abeokuta Ogun State, Nigeria.*
obrienadedapo15@gmail.com

**Orji Everistus Eze**
*Department of Computer Science, School of Science and Technology, Federal Polytechnic Ohodo, Enugu State, Nigeria.*
everistus.orji@fedpod.edu.ng

***Abstract***— Traditional paper-based testing does not normally examine coding capabilities well without replicating real-world experience. That called for the creation of E-Proctor, which is an intelligent web platform that transforms programming tests. The system follows the MVC pattern for scalability and reliability. Developed in VS Code, it uses React.js for the frontend, Express.js on Node.js as the backend, and PostgreSQL as the database. The platform's development environment enables students to code, test, and submit under exam-like conditions for a realistic assessment of skills. It features live testing with anti-cheating tools like window monitoring and browser locking. The system was also tested through the Cypress end-to-end testing suite. The system effectively generated, managed, and graded examinations securely, meeting the project goal. Initial tests show it reduces academic dishonesty and improves programming evaluation quality. Testing confirms it provides a more realistic testing experience and lowers the administrative burden for teachers.

***Keywords***— *Examination Integrity, Examination Management, Integrated Development Environment, Online Proctoring, Online Examinations, and Programming Examinations.*

## I. INTRODUCTION

Given that traditional instruction has been in person, exams have also been given in person [1]. The most prevalent technique in the conventional environment is human proctoring, where one either requests students to take an exam in the testing labs or to turn on their camera [2]. However, these techniques require a lot of hardware, infrastructure, labor, and effort [2]. Also, cheating comes in the form of whispering or mouthing the answer, exchanging scripts or small pieces of paper, bringing in answers written on body parts or clothing, sneaking in phones and other digital devices such as smart glasses and watches [3], or even "giraffing," which is a colloquial term describing the attempt to peek into another student's script [1]. These are managed through physical frisking and checks at the entrance to the exam room for unauthorized materials, the use of CCTV as a deterrent where the learners know they are being monitored, random seating of the learners with adequate spacing, and the use of sufficient and trained invigilators [1].

The field of computer science education has undergone significant transformations, particularly in the way programming skills are taught and assessed [4]. Coding, compilation, linking, testing, and debugging are some of the steps in programming [5]. Installation and configuration of the involved tools and environments are necessary for each step. Although integrated development environments (IDEs) attempt to incorporate and expose all programming tools in a single interface, the user must still install and configure each individual tool [6]. For instance, users will still need to install Python interpreters and set up the IDE to utilize the correct installation (for instance, Python 2.7 or 3.6) even though an IDE for the Python programming language will syntax highlight for Python projects [6].

For the student studying programming, setting and installing programming environments and tools is an often lengthy, labor-intensive, and error-prone task. It will further distract the attention of the student from learning [6]. That's where an integrated development environment delivered online can make a difference.

Traditionally, programming examinations have often been conducted on paper, which is very unrealistic: real developers work on a keyboard with extensive resources at hand, rather than on paper with few or no notes [7].

Transitioning from paper-based programming examinations to web IDEs can significantly enhance the evaluation process in several ways. As [7] highlights, a more practical evaluation would involve asking students to write real working code in more realistic situations, giving them access to test cases,

coding environments, and other amenities as they would have if they were working in the lab or doing homework, but depriving them of the ability to ask their professors or friends if they get stuck.

Online exams must be administered using online proctoring software since they are conducted to test the student's knowledge remotely without the need for proctors [8]. However, cheating is the primary issue with online exams, so the first thing to remember when creating any form of online test system is how to proctor exams in a way that is reliable, effective, and convenient [9].

Such practice of remotely monitoring the actions of an examinee while they are being examined to determine any move that would amount to fraud is known as an online proctoring system, or e-proctoring [10]. It is done with the aid of a camera, microphone, and preferably some control over the computer upon which the examinee is taking the exam. Detection may be carried out by a human agent (the proctor or invigilator), an elementary algorithm, an AI-powered algorithm, or any combination of the above options [10].

However, because online testing and e-proctoring are done online [11], they are susceptible to cyber-attacks and threats [12]. The effectiveness of any proctoring platform is thought to be in outsmarting such attacks and threats [11].

This project aims to improve the security, integrity, and efficacy of online programming assessments by integrating features like window tracking and browser locking to prevent cheating, as well as an intuitive interface for both teachers and students. It also ensures that system functionality, performance, security, and usability testing with actual users are validated to ensure effectiveness and intuitiveness.

Existing research, including the works of [13-20], often requires a stable and high-speed internet connection and modern hardware capabilities. There is a gap in providing robust online proctoring solutions that can adapt to lower bandwidth and less capable devices, which would be crucial for wider applicability, especially in less developed regions. This study proposes a platform whose system architecture is designed to be lightweight and could support lower bandwidth and less capable devices.

While general proctoring solutions are prevalent as seen in the works of [13], [14], [21], and so on, they often fall short of addressing the specific needs of programming assessments, such as IDE compatibility and code analysis. This study is designed to seamlessly integrate online coding environments, offering an environment where programming examinations can take place and that can be accessed.

This study proposes a platform that combines traditional methods of proctoring and other anti-cheating measures, unlike the works of [2], [15], [19], and [22], which focused on using artificial intelligence and other advanced algorithms for facial recognition. This study proposes a method of proctoring that could involve immediate lockdowns of the examination environment, providing a more efficient monitoring solution.

The relevant research work, predominantly between the years 2021 and 2024, is summarized in Table 1, and Table 2 determines the difference between the characteristics of the current systems and the proposed system.

**Table 1:** Summary of existing works

| S/N | Author | Title | Methodology | Strengths | Limitations |
|---|---|---|---|---|---|
| 1 | Labayen et al. [16] | A Real Time Student Authentication and examination supervision System. | Created a system that can perform continuous multimodal biometric authentication and typing pattern recognition. | Real-time processing with minimal hardware requirements.<br>Enhances user convenience. | Reliant on high quality internet connection; |
| 2 | Jia et al., [24] | The Design, Implementation, and Pilot Application of an Intelligent Online Proctoring System for Online Exams | Developed and intelligent online proctoring system that utilizes artificial intelligence to ensure the online examinations integrity. | Responds to the global need for reliable online examination systems. | Effectiveness may be compromised by poor hardware or unstable internet connections. |
| 3 | Motwani et al. [18] | AI-Based Proctoring System for Online Tests | Developed several stage AI-based proctoring systems that integrate various monitoring technologies to oversee online tests. | Comprehensive monitoring capabilities.<br>reduces the need for physical proctoring. | It relies on continuously high-quality internet connection. |
| 4 | Thombare et al. [17] | Smart Exam Proctoring System | Employed a client-side approach using Media pipe for head pose estimation and browser-based event listeners to monitor student behavior during exams. | Reduces server load.<br>Effectively distributes computational needs. | It is dependent on the assessment taker's hardware quality. |
| 5 | Kasinathan et al. [21] | A Virtual Proctoring System with Automated Monitoring | An advanced proctoring system that integrates artificial intelligence to monitor students' behavior such as eye movement and browser activity during exams. | Enhances the integrity of online examinations.<br>Promises a scalable and efficient solution through automation. | Requires high-quality internet and computer hardware potentially limiting accessibility. |
| 6 | Potluri et al. [13] | An Automated Online Proctoring System using Attentive-Net to Assess Student Mischievous Behavior | Developed "Attentive-Net" incorporating face detection, multiple people detection, face spoofing, and head pose detection. | Effective real-time processing; minimizes false alarms; significantly reduces labor and infrastructure needs. | Relies on high quality internet connection; raises privacy concern due to continuous monitoring. |
| 7 | Nurpeisova et al. [14] | Research on the Development of a Proctoring System for Conducting Online Exam in Kazakhstan | Created "Proctor SU" using AI technologies like CNN, R-CNN YOLOv3 for real-time image and activity processing during exams. | Enhances the integrity of online assessments.<br>reduces the need for physical proctoring resources. | Requires reliable internet and advanced computer hardware; |
| 8 | Satre et al. [15] | Online Exam Proctoring System Based on Artificial Intelligence | Utilized AI technologies like YOLO for object detection and CNN for face recognition. | Real-time alerts to examiner.<br>Autonomous monitoring to enhance the integrity of examinations | Dependent on high quality internet connection and sophisticated hardware; |
| 9 | Tajane et al. [19] | Online Exam Proctoring System | Used AI-powered systems capable of detecting cheating through multimodal inputs and YOLO for object detection. | Enhance monitoring capabilities of educators.<br>reduces the need for physical test centers. | It relies on robust hardware.<br>raises ethical issues with continuous surveillance. |
| 10 | Wakchaure et al. [23] | Smart Exam Proctoring System | Explored the use of HAAR Cascades for face detection and Local Binary Pattern Histogram for face recognition. | Enhances online examination security.<br>Maintains the integrity of online examination without physical monitoring. | Reliability issues in diverse lighting and network conditions. |
| 11 | Nethravathi et al. [24] | Smart Artificial Intelligence Based Online Proctoring System | Developed an AI-driven system designed to enhance the integrity of online exams by monitoring test-takers through multiple modalities, including video, audio, and active window capture. | Increased scalability.<br>Cost-effective.<br>Multiple modalities. | Dependence on sophisticated technology can lead to accessibility issues and testing inequities.<br>Controversial accuracy in detecting emotions and intent |
| 12 | Ganar et al. [25] | Online Exam Proctoring System | Developed a multi-dimensional system that integrates user authentication, text detection, audio detection and tab-switching detection to monitor the test environment continuously | Cost-effective.<br>Scalability.<br>Enhanced Security.<br>Real-time Cheating Detection | Privacy Concerns; Reliance on technological accuracy |
| 13 | Bedmutha et al. [20] | Online Proctoring System: A Client-Side Using Deep Learning | Explored an cutting edge client-side approach using AI technologies to manage face detection and suspicious activity detection systems. | Innovative client-side processing reduces server load and minimizes dependence on network stability.<br>Enhances efficiency and scalability. | Heavy reliance on examinees' hardware, potentially affecting fairness and system efficacy.<br>Potential for inconsistent monitoring standards due to variation in device performance. |

**Table 2:** Comparison with the existing systems

| Features/ System | Potluri et al. [13] | Nurpeisova et al. [14] | Kasinathan et al. [21] | Satre et al. [15] | Bedmutha et al. [20] | Proposed System |
|---|---|---|---|---|---|---|
| Real-time Cheating Detection | × | × | ✓ | ✓ | × | × |
| Browser Lockdown Features | × | × | × | × | × | ✓ |
| Scalable Solution | ✓ | ✓ | ✓ | ✓ | ✓ | ✓ |
| Login & Sign-up for Users | × | × | × | × | × | ✓ |
| Examination Management | × | × | × | × | × | ✓ |
| Programming Environment | × | × | × | × | × | ✓ |
| Code Execution within the environment | × | × | × | × | × | ✓ |

While Table 2 compared E-Proctor to other research projects, it is also useful to look at the platforms people use in the real world. Popular sites like HackerRank and LeetCode are excellent for testing coding skills, but they are built for hiring developers, not for running university exams with strict proctoring. On the other hand, Moodle has plugins like CodeRunner that let students write code, but they rely on general security settings rather than tools designed specifically for programming tests. There are also online coding tools like Replit, which are great for writing code but aren't built to stop cheating during an exam. E-Proctor attempts to bridge this gap by combining a proper coding environment with the security features needed for academic exams. However, it is worth noting that E-Proctor is still a prototype; it would need significant development to match the polish and advanced features of these major commercial platforms.

## II. MATERIALS AND METHOD

The proposed system was developed using a waterfall development methodology, which structures the development process into sequential phases: requirements gathering, system design, implementation, testing, deployment, and maintenance. The Waterfall approach was selected for this project due to the rigid nature of academic examination regulations.

However, this approach had limitations. The nature of this methodology prevented iterative user feedback and design refinement during development. Modern web development typically favors Agile methodologies, enabling continuous integration and incremental feature deployment. Future iterations would benefit from Agile practices, including sprint-based development and continuous user testing.

The development phases included defining functionalities for examiners and students (requirements), designing the MVC architecture with React.js, Express.js, and PostgreSQL (design), developing frontend and backend components (implementation), and performing end-to-end Cypress testing (testing).

After analysis has been carried out, that is, user requirements, tool requirements, and software requirements have been gathered. The way a system must function is specified by its functional requirements. It describes the actions that the system needs to execute to satisfy user requirements or expectations [26]. Non-functional requirements, or NFRs, are regarded as a set of specifications that outline the system's operational capabilities and limitations to enhance its functionality.

The functional and nonfunctional criteria required for this project are displayed in Tables 3 and 4. Non-functional requirements describe criteria that can be used to assess how well a system operates, whereas functional requirements define the functions of a system or a component inside a system [27]. A function is defined as a definition of behavior between inputs and outputs.

**Table 3:** Functional Requirements

| No. | Module | Functional Requirement |
|---|---|---|
| 1. | User management | This module allows examiners and students to create accounts by providing the necessary information. allow users access their accounts using their credentials. It also provides a functionality for password reset in case a user forgets their password. |
| 2. | Dashboard management | This module provides examiners with a dashboard where they can manage exam, access submitted work and grade assignments |
| 3. | Examination creation and management | This allows examiners to create examinations, including setting exam title, duration and other relevant details. It also allows for editing and deleting examinations. |
| 4. | Student examination participation | This module allows students to select the examinations they want to take and provide a secure environment where students can take examinations within the stipulated time. |

**Table 4:** Non-functional Requirements

| No. | Non-Functional Requirements | Explanation |
|---|---|---|
| 1. | Performance | This covers the scalability and response time of the system. It ensures that the system can handle multiple users concurrently especially during exam periods. |
| 2. | Reliability | This ensures that the system is always available particularly during exam periods. |
| 3. | Usability | This makes sure that the system has a friendly user interface and is easy to navigate. |

## A. Architecture of the System

The design that is used by the E-Proctor adheres to the Model-View-Controller (MVC) pattern. This architectural pattern is a foundational aspect for handling interactions in most web-based system. It decouples the presentation and interaction details from the underlying data of the system. The system consists of three distinct yet interconnected components: the model, the view, and the controller.

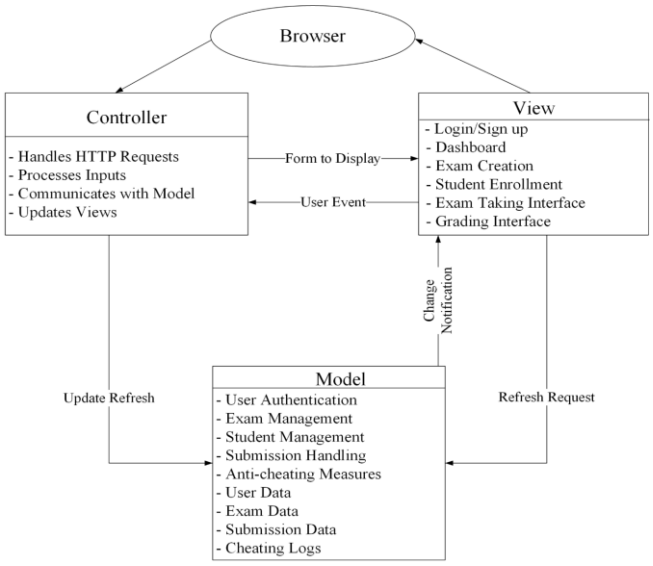


**Figure 1**: System Architecture

## B. Algorithm Design and Implementation

Start

1. System loads.
2. User Authentication:
   a. Users visit the platform, display the homepage with options to "Login" either as an examiner or a student.
   b. If they have no account, they create one
   c. For examiners, who do not have an account, they sign up using their name, email, and password.
   d. Validate input and check if email already exists if not,
   e. Create an examiner account and store details in the database.
   f. For students, they sign up using their name, department, level, matriculation number and password.
   g. Validate input and check if the matriculation number already exists if not,
   h. Create a student account and store details in the database.
   i. After signing up, prompt student to select which examination they want to sit for
   j. Redirect to the login page for both users after successful account creation.
   k. Log examiners in using their email and password and log students in using their matriculation number and password.
   l. Validate credentials against the database. If correct,.
   m. Direct users to their respective dashboards based on their role (examiner or student).
3. Session Management:
   a. Secure user sessions up on login.
   b. Automatic logout after a period of inactivity.
4. Examiner Dashboard:
   a. create a new exam, set title, description, duration, and questions.
   b. View, edit, or delete existing exams.
   c. View student submitted work.
   d. Grade submissions and update grades in the database.
5. Student Dashboard:
   a. Display selected examinations based on the student's enrollment.
   b. Select the examination that they want to take.
   c. Allow students to select an exam they wish to sit for.
   d. Monitor completion time based on the exam duration.
   e. Submit answers to the database for storage and later grading.

End.


https://journals.ust.edu/index.php/JST https://doi.org/10.20428/jst.v31i2.3506

### C. Data flow diagram of the system

Figure 2 shows the data flow diagram that models the flow of data within the system. The data flow diagram illustrates the sequence of events and decision points involved in creating examinations, taking them, grading them, and saving grades.

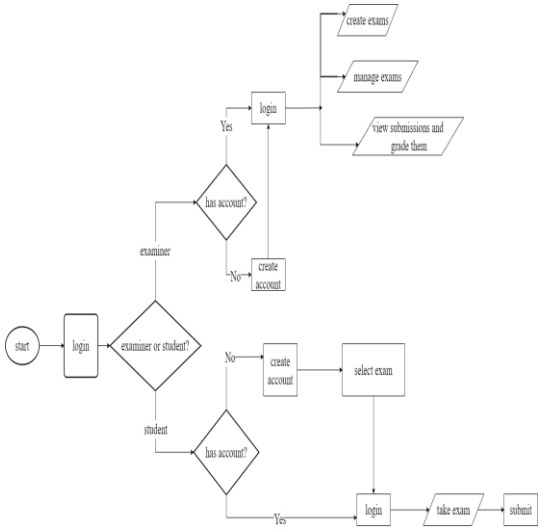


**Figure 2:** Data Flow diagram of the system.

### D. System design

Figure 3 shows the use case diagram of the new system, depicting all the interaction between system functionalities and two primary actors: the examiner and students. The system comprises three main modules: account management, examination management, and grading. The account management module extends to login, create account, and logout functionalities. The examination management module, accessible to the examiner, encompasses four extended use cases: creating an examination, editing an examination, deleting an examination, and viewing an examination. The grading module extends to two functionalities: access submitted work and grade examination. The enrolled student interacts with two basic functionalities: logging in and taking the examination. Additionally, the examiner has access to a specialized functionality for configuring anti-cheating measures.

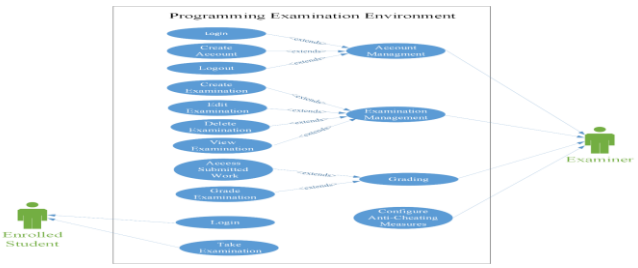


**Figure 3**: Use Case Diagram of the System

The sequence diagram in figure 4 illustrates the interaction flow between six components: examiner, student, dashboard, exam management, anti-cheating, and system. The workflow begins with authentication, followed by conditional paths for account creation or direct dashboard access. The examiner's sequence involves exam creation and anti-cheating implementation, while the student's sequence progresses from login through exam registration to exam completion. The final sequence shows the grading workflow, where examiners assess submitted work and grades are displayed to all stakeholders. All interactions are depicted through horizontal arrows, with dashed lines showing return messages and solid lines representing actions.

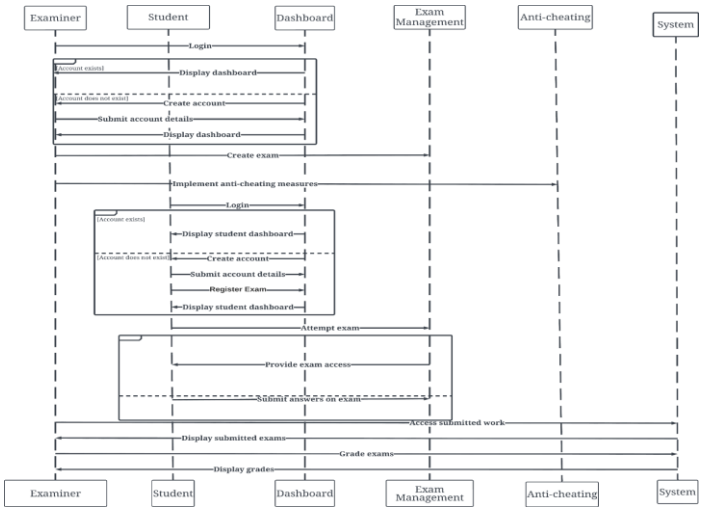


**Figure 4:** Activity diagram of the system

Figure 5 below depicts a class diagram that represents the proposed system with several interconnected components. The system consists of student and examiner users who interact with exams through a dashboard managed by an authentication system. Students, identified by their Matric Number, can register for, take, and submit exams. Examiners, identified by email, can create exams, set anti-cheating measures, and grade student submissions. Each exam has a name, start and end dates, and tracks student enrollment. When a student takes an exam, they create a submission containing their answers and submission time. The Dashboard provides access to both students and examiners, while the Authentication System handles user login and account creation. All interactions are secured through password authentication, ensuring that users can only access their authorized functions within the system.

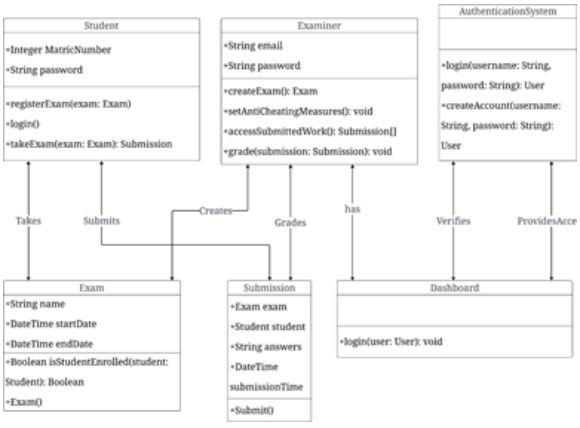


**Figure 5:** Class Diagram of the System


https://journals.ust.edu/index.php/JST https://doi.org/10.20428/jst.v31i2.3506

Examiner, Dashboard, Student, and Exam are the four main classes in the proposed system, along with a Submitted Work class, as shown in Figure 6. The Examiner class manages user authentication and has access to a Dashboard where they can create and manage exams, as well as implement anti-cheating measures. Students can create accounts, log in, enroll in exams, and attempt them. The Exam class contains exam details like name, questions, time limit, and grading criteria, with methods to start and end exams. When students take exams, their work is recorded in the Submitted Work class, which stores their answers and grades. The relationships show that an examiner has a dashboard, the dashboard manages students and exams, and each exam can have multiple submitted work entries from different Students. This structure enables a complete workflow from exam creation to submission and grading.

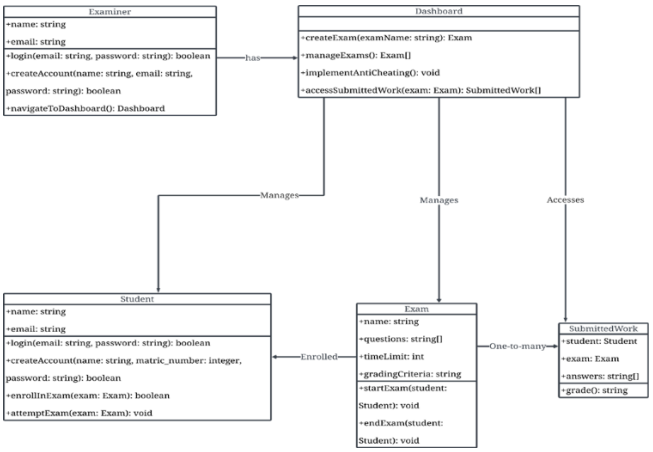


**Figure 6:** Object Diagram of the System

### E. System implementation

The implementation of the system was done in Visual Studio Code, an integrated development environment, because it supports different programming languages instead of having to use different IDEs for different programming languages, and it supports offline use too, which allows for the program to run locally during development [29]. React.js is the framework used to develop the front end of the new system. It is a JavaScript framework used in web application interface development. In the backend of the system, Express based on Node.js, which is a light and versatile web application framework that provides an extensive set of features for both web and mobile applications [28], is used. As the system's database, PostgreSQL, an object-relational database management system (ORDBMS), is used.

## III. RESULTS AND DISCUSSION

### A. User interface of the new system

Figure 7 depicts the landing page of the new system. This is the first page after loading the application, it is the page where each user finds their way to their respective dashboards. After clicking “Get Started”, users are prompted to either login as an examiner or a student.


**Figure 7:** Landing page

Figure 8 shows the sign-up page for examiners. The purpose of this page is to enable the examiners to create an account that will be used with the application. During implementation, the design philosophy was to collect as little information as possible, and thus we collected 3 things:

1. Email address - for sending relevant information to the user

2. First Name & Last Name - To have something to identify the user by

3. Password - To verify the user’s identity during authentication

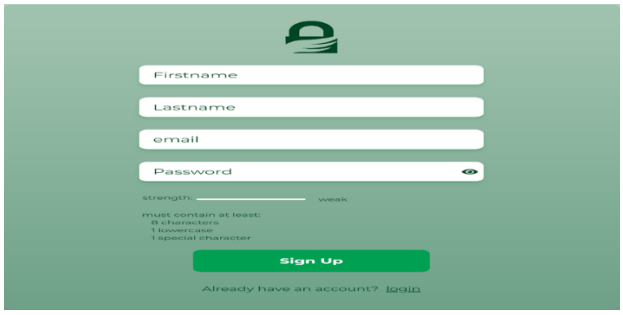


**Figure 8**: Sign Up Page for Examiners


**Figure 9:** Registration Page

The registration page where students select the examination they want to sit for after signing up is depicted in figure 9.


https://journals.ust.edu/index.php/JST https://doi.org/10.20428/jst.v31i2.3506

### B. System testing

System testing is an important stage in an application's development life cycle. It ensures that the application adheres to the specified requirements and behaves as expected [30]. This section outlines the system testing approach, including the test plan, test data, performance evaluation, and limitations of the system. End-to-end testing is the type of test that is going to be carried out on the software built. This includes automating the complete flow of the application from signing up and taking an exam to grading using the Cypress test suite. All the tests carried out are depicted in Table 5.

**Table 5:** Test Summary Report

| Test Case ID | Test Cases | Description | Expected Result | Result | Pass/Fail |
|---|---|---|---|---|---|
| **TC001** | Account Creation/ Signing up | This involves testing whether users (examiners and students) can successfully sign up. | Users should be able to sign up. | Users can sign up. | Pass |
| **TC002** | Logging In | This involves testing whether users (examiners and students) can successfully log in to their dashboards after creating accounts. | Users should be able to log in to their accounts. | Users can log in to their accounts. | Pass |
| **TC003** | Examination Creation | This involves testing whether examiners can create exams. | Examiners should be able to create examinations | Examiners can create examinations | Pass |
| **TC004** | Examination Editing | This involves testing whether examiners can edit exams. | Examiners should be able to edit examinations | Examiners can edit examinations | Pass |
| **TC005** | Examination Deletion | This involves testing whether examiners can delete exams. | Examiners should be able to delete examinations | Examiners can delete examinations | Pass |
| **TC006** | Accessing Submitted Examinations | This involves testing whether examiners can access examinations submitted by students. | Examiners should be able to access submitted examinations | Examiners can access submitted examinations | Pass |
| **TC007** | Student Selecting Examination to Take | This test involves checking whether students select the examinations they want to sit for after signing up. | Students should be able to select examinations after signing up. | Students can select examinations after signing up. | Pass |
| **TC008** | Taking Examination | This involves testing whether students can take the examinations they have selected. | Students should be able to take examinations they have selected. | Students can take examinations they have selected. | Pass |
| **TC009** | Submitting Examinations | This test makes sure students and submit their examinations. | Students should be able to submit their examination after completion. | Students can submit their examination after completion. | Pass |

The system underwent comprehensive end-to-end testing using the Cypress suite to ensure functional reliability. Over the development period, 15 full test cycles were executed across all 9 test cases, resulting in a 100% pass rate. Performance was also monitored during development to gauge responsiveness. In a simulated environment, authentication typically completed between 0.8 and 1.2 seconds, while exam loading required approximately 1.5 to 2.0 seconds. Code submission tasks took slightly longer, averaging 2.0 to 3.0 seconds. The system also demonstrated stability during pilot testing with 10 concurrent users, showing no significant lag.

However, it is important to acknowledge the limitations of this evaluation. Testing was conducted in a controlled environment, meaning that large-scale load testing (with 50+ users) and analysis across unstable network connections were not performed due to resource constraints. Furthermore, while the technical performance appears stable, a statistical comparison of student learning outcomes between this platform and traditional methods has been identified as an area for future work.

### C. Privacy and Ethical Considerations

The proposed monitoring features raise important privacy concerns that must be addressed. The prototype implements basic browser controls (window tracking, browser lockdown, full-screen enforcement) but does not implement video/audio recording or facial recognition.

The data protection methods include password hashing and access control with session-based authentication. However, if video/audio recording or facial recognition is to be integrated, the following must be implemented:

1. Student consent mechanisms with transparent data collection disclosure.

2. Clear data retention and deletion policies.

3. compliance with data protection regulations

## D. CONCLUSIONS

In this paper, a programming examination environment that caters to both examiners and students. The system allows both users, i.e., examiners and students, to sign up and log in to their respective dashboards. It allows examiners to be able to create examinations, enroll students for the examination, and manage the students (i.e., enroll students who have not been originally enrolled or change records of wrong students). Anti-cheating measures have also been put in place to ensure the integrity of the examinations. Students can log into their accounts, access their enrolled examinations, and submit their work for evaluation. The system ensures that only authorized students can take examinations, providing a secure and fair testing environment. The developed system successfully addresses the issues associated with conducting programming examinations online. It offers a friendly user interface for both users. The system represents a significant step forward in modernizing programming examinations, providing a platform that can be adopted by institutions to conduct programming examinations effectively.

Despite these contributions, several limitations must be acknowledged regarding the current iteration of the system. First, stress testing was not performed with large concurrent user loads, and the platform currently relies on a stable internet connection without an offline fallback mechanism. The evaluation is also limited by the lack of comprehensive benchmarking under varied network conditions and the absence of long-term reliability data. Furthermore, advanced monitoring features, such as video/audio recording or facial recognition, were not implemented in this version, and full compliance with data protection regulations like GDPR has not yet been certified. These limitations highlight critical areas for future development before the system can be considered ready for widespread institutional deployment.

**Acknowledgment**

The authors would like to acknowledge the Department of Computer Science, Federal University of Agriculture, Abeokuta, for making use of the school software laboratory for the system testing.

**Declaration:**

Availability of data and materials

The original contributions presented in the study are included in the article; further inquiries can be directed to the corresponding authors.

**Conflict of interest**

The author declared that there is no conflict of interest.

**Funding statement:** Not applicable

**Authors' Contributions**

Dr. Charles Okechukwu Ugwunna: Conceptualization, system modeling, system methodology, algorithms, system testing, system implementation, and coding. Dr. Akawuku Ifeanyi Godspower: Managed data interpretation, prepared system modeling, designed the E-proctor architecture, and conducted a literature review. Dr. Chukwuogo Okechukwu Ejike: Data acquisition, writing part of the original draft, system training, draft reviewing/editing, system implementation, and coding. Dr. Joshua John: Wrote part of the original draft, provided critical feedback during proofreading, and ensured adherence to publication standards, system evaluation, and system implementation. Mr. Adedapo Taiwo Obrien: Conceptualization, system implementation and coding, draft reviewing/editing, and system testing. Mr. Orji Everistus Eze: Methodology, writing part of original draft, correction of original draft, and system analysis and design.

**References**

[01] A. M. Njuguna, "User experience of online examinations and proctoring: A case-based study," Int. J. Curr. Sci. Res. Rev., vol. 5, no. 07, pp. 2326-2335, 2022. DOI: 10.47191/ijcsrr/V5-i7-12

[02] T. Tejaswi, C. Motwani, M. Nagpal, N. Nagdev, and A. Yeole, "AI-based proctoring system for online tests," in Proc. 4th Int. Conf. Adv. Sci. Technol. (ICAST), May 2021. DOI:10.2139/ssrn.3866446

[03] L. W. Daffin Jr and A. A. Jones, "Comparing Student Performance on Proctored and Non-Proctored Exams in Online Psychology Courses," Online Learn., vol. 22, no. 1, pp. 131–145, 2018. DOI: https://doi.org/10.24059/olj.v22i1.1079

[04] K. Andersen, S. E. Thorsteinsson, H. Thorbergsson, and K. S. Gudmundsson, "Adapting engineering examinations from paper to online," in Proc. IEEE Global Eng. Educ. Conf. (EDUCON), Apr. 2020, pp. 1891–1895. DOI:10.1109/EDUCON52537.2022.9766469

[05] C. Kelleher and R. Pausch, "Lowering the barriers to programming: A taxonomy of programming environments and languages for novice programmers," ACM Comput. Surv. (CSUR), vol. 37, no. 2, pp. 83–137, 2005. https://doi.org/10.1145/1089733.1089734

[06] R. Kadar, N. Abdul Wahab, J. Othman, M. Shamsuddin, and S. B. Mahlan, "A study of difficulties in teaching and learning programming: A systematic literature review," Int. J. Acad. Res. Prog. Educ. Dev., vol. 10, no. 3, pp. 591–604, 2021, DOI:10.6007/IJARPED/v10-i3/11100.

[07] A. Ju, B. Mehne, A. Halle, and A. Fox, "In-class coding-based summative assessments: tools, challenges, and experience," in Proc. 23rd Annu. ACM Conf. Innov. Technol.

Comput. Sci. Educ., Jul. 2018, pp. 75–80. https://doi.org/10.1145/3197091.3197094

[08] A. Alghamdi, M. Alanezi, and F. Khan, "Design and implementation of a computer-aided intelligent examination system," Int. J. Emerg. Technol. Learn., vol. 15, no. 1, pp. 30–44, 2020. https://doi.org/10.3991/ijet.v15i01.11102

[09] H. Li, M. Xu, Y. Wang, H. Wei, and H. Qu, "A visual analytics approach to facilitate the proctoring of online exams," in Proc. CHI Conf. Human Factors Comput. Syst., May 2021, pp. 1–17. https://doi.org/10.1145/3411764.3445294

[10] L. Bergmans, N. Bouali, M. Luttikhuis, and A. Rensink, "On the efficacy of online proctoring using Proctorio," in Proc. 13th Int. Conf. Comput. Supported Educ. (CSEDU). https://doi.org/10.5220/0010399602790290

[11] H. M. Mohammed and Q. I. Ali, "Cheating prevention in e-proctoring systems using secure exam browsers: A case study," J. Ilm. Tek. Elektro Komput. Informatika (JITEKI), vol. 8, no. 4, pp. 634–648, Dec. 2022, doi: 10.26555/jiteki.v8i4.25094.

[12] L. Slusky, "Cybersecurity of online proctoring systems," J. Int. Technol. Inf. Manage., vol. 29, no. 1, pp. 56–83, 2020. https://doi.org/10.58729/1941-6679.1445

[13] T. Potluri and V. P. K. Sistla, "A comprehensive survey on the AI-based fully automated online proctoring systems to detect anomalous behavior of the examinee," in Proc. Int. Conf. Recent Trends Microelectron. Autom. Comput. Commun. Syst. (ICMACC), Dec. 2022, pp. 407–411. DOI: 10.1109/ICMACC54824.2022.10093571

[14] A. Nurpeisova, A. Shaushenova, Z. Mutalova, M. Ongarbayeva, S. Niyazbekova, A. Bekenova, and S. Zhumasseitova, "Research on the development of a proctoring system for conducting online exams in Kazakhstan," Computation, vol. 11, no. 6, p. 120, 2023. https://doi.org/10.3390/computation11060120

[15] S. Satre, S. Patil, T. Mane, V. Molawade, T. Gawand, and A. Mishra, "Online exam proctoring system based on artificial intelligence," in Proc. 2023 Int. Conf. Signal Process., Comput., Electron., Power Telecommun. (IConSCEPT), May 2023, pp. 1–6. https://doi.org/10.1109/IConSCEPT57958.2023.10170577

[16] M. Labayen, R. Vea, J. Flórez, N. Aginako, and B. Sierra, "Online Student Authentication and Proctoring System Based on Multimodal Biometrics Technology," IEEE Access, vol. 9, pp. 72398–72411, 2021. DOI: 10.1109/ACCESS.2021.3079375

[17] C. Thombare, K. Sapate, A. Rane, and A. Hutke, "Exam proctoring system," Int. J. Res. Appl.Sci. Eng. Technol., vol. 10, pp. 466–470, 2022. https://doi.org/10.22214/ijraset.2022.42229

[18] S. Motwani, C. Nagpal, M. Motwani, N. Nagdev, and A. Yeole, "AI-based proctoring system for online tests," in Proc. 4th Int. Conf. Adv. Sci. Technol. (ICAST2021), May 2021. https://dx.doi.org/10.2139/ssrn.3866446

[19] K. Tajane, V. Bambale, A. Gomsale, A. Yadav, and S. Walzade, "Online Exam Proctoring System," Int. J. Adv. Res. Sci. Commun. Technol., vol. 3, no. 1, pp. 202–207, 2023. DOI:10.48175/IJARSCT-9027

[20] D. Bedmutha, P. Bapecha, D. Chaure, P. Bora, and M. R. Karnavat, "Online Proctoring System: A Client-Side Approach Using Deep Learning," Authorea Preprints, 2024.

DOI:10.22541/au.170534551.10330347/v1

[21] V. Kasinathan, C. E. Yan, A. Mustapha, V. A. Hameed, T. H. Ching, and V. Thiruchelvam, "Proctorex: An automated online exam proctoring system," Math. Statist. Eng. Appl., vol. 71, no. 3s2, pp. 876–889, 2022.

[22] G. Moukhliss, R. F. Hilali, and H. Belhadaoui, "Intelligent solution for automatic online exam monitoring," Int. J. Electr. Comput. Eng., vol. 13, no. 5, 2023.

DOI:10.11591/ijece. v13i5.pp5333-5341

[23] S. Wakchaure, A. Tambe, P. Gadhave, S. Sandanshiv, and A. Kadam, "Smart Exam Proctoring System," Int. J. Res. Appl. Sci. Eng. Technol. (IJRASET), vol. 11, no. 4, 2023. DOI: 10.22214/ijraset.2023.51358.

[24] J. Jia and Y. He, "The design, implementation and pilot application of an intelligent online proctoring system for online exams," Interact. Technol. Smart Educ., vol. 19, no. 1, pp.112–120, 2021. https://doi.org/10.1108/ITSE-12-2020-0246

[25] E. S. Ganar, S. Mohammad, K. Zohair, and R. Shaikh, "Online Exam Proctoring System," Int. J. Adv. Res. Sci. Commun. Technol., 2023.

DOI:10.15587/1729-4061.2024.306968

[26] K. Kaur and P. Kaur, "The application of AI techniques in requirements classification: a systematic mapping," Artif. Intell. Rev., 2024, doi: 10.1007/s10462-024-10667-1.

[27] M. Glinz, "On non-functional requirements," in Proc. 15th IEEE Int. Requirements Engineering Conf., 2007, pp. 21–26, doi: 10.1109/RE.2007.45.

[28] N. Jain, S. Pise, and P. Patil, "Comparative study of modern web development frameworks," Int. J. Web Eng. Technol., vol. 16, no. 3, pp. 213–229, 2021, doi: 10.1504/IJWET.2021.116968.

[29] J. Tan, Y. Chen, and S. Jiao, "Visual Studio Code in Introductory Computer Science Course: An Experience Report," arXiv preprint arXiv:2303.10174, Mar. 2023. arXiv

[30] P. Ammann and J. Offutt, Introduction to Software Testing, 2nd ed. Cambridge University Press, 2016, doi: 10.1017/CBO9781316771273.