



**COMSATS University Islamabad**  
**Abbottabad, Pakistan**

**SmartGuardPro - Guarding Academic Integrity, One  
Assignment at a Time**

*By*

**Muneeb-Ur-Rehman CIIT/FA20-BCS-016/ATD**  
**Ibrahim Umer CIIT/FA20-BCS-019/ATD**  
**Muhammad Saad CIIT/FA20-BCS-043/ATD**

**Supervisor**  
**Ahsan Khan**

***Bachelor of Science in Computer Science (2020-2024)***

**The candidate confirms that the work submitted is their own and appropriate credit has been given where reference has been made to the work of others.**



**COMSATS University, Islamabad Pakistan**

**A project presented to  
COMSATS University, Islamabad, Abbottabad Campus**

**In partial fulfillment  
of the requirement for the degree of**

***Bachelor of Science in Computer Science (2020-2024)***

***By***

<b>Muneeb-Ur-Rehman</b>	<b>CIIT/FA20-BCS-016/ATD</b>
<b>Ibrahim Umer</b>	<b>CIIT/FA20-BCS-019/ATD</b>
<b>Muhammad Saad</b>	<b>CIIT/FA20-BCS-043/ATD</b>

## **DECLARATION**

We hereby declare that this software, neither whole nor as a part has been copied out from any source. It is further stated that we have developed this software and accompanied a report entirely based on our personal efforts. If any part of this project is proved to be copied out from any source or found to be reproduction of some other. We will stand by the consequences. No Portion of the work presented has been submitted of any application for any other degree or qualification of this or any other university or institute of learning.

-----  
Muneeb-Ur-Rehman

-----  
Ibrahim Umer

-----  
Muhammad Saad

## **CERTIFICATE OF APPROVAL**

It is to certify that the final year project of BS (CS) “SmartGuardPro - Guarding Academic Integrity, One Assignment at a Time” was developed by **Muneeb-Ur-Rehman (CIIT/FA20-BCS-016/ATD)**, **Ibrahim Umer (CIIT/FA20-BCS/019/ATD)** and **Muhammad Saad (CIIT/FA20-BCS-043/ATD)** under the supervision of “Ahsan Khan” and that in his opinion, it is fully adequate, in scope and quality for the degree of Bachelors of Science in Computer Sciences.

-----  
Supervisor

-----  
External Examiner

-----  
Head of Department  
(Department of Computer Science)

## **EXECUTIVE SUMMARY**

Smart Guard Pro is an innovative solution designed to address the challenges faced by educational institutions in ensuring the integrity and fairness of online examinations. With the rapid transition to digital learning platforms, the prevalence of cheating during online tests has emerged as a significant concern for educators. Smart Guard Pro aims to mitigate this problem by leveraging advanced technologies, including facial recognition, real-time monitoring, and plagiarism detection, to create a secure and trustworthy assessment environment.

The core features of Smart Guard Pro include automated attendance management, behavior monitoring, and cheating prevention mechanisms, all integrated into a user-friendly mobile application. By harnessing the power of artificial intelligence and machine learning, Smart Guard Pro provides educators with the tools they need to verify student identities, monitor their activities during exams, and detect instances of academic dishonesty. Additionally, comprehensive reporting and analytics capabilities empower educators to gain insights into students' performance and behavior during assessments.

Through its modular architecture and agile development approach, Smart Guard Pro offers scalability, adaptability, and ease of maintenance. The solution can be seamlessly integrated into existing online learning platforms, enabling educational institutions to enhance the integrity and credibility of their examination processes. Smart Guard Pro represents a significant advancement in online examination security, promising to uphold academic standards and foster a culture of honesty and accountability in education.

## **ACKNOWLEDGEMENT**

All praise is to Almighty Allah who bestowed upon us a minute portion of boundless knowledge by virtue of which we were able to accomplish this challenging task.

We are greatly indebted to our project supervisor “Ahsan Khan”. Without their personal supervision, advice and valuable guidance, completion of this project would have been doubtful. We are deeply indebted to them for their encouragement and continual help during this work.

And we are also thankful to our parents and family who have been a constant source of encouragement for us and brought us the values of honesty & hard work.

Muneeb Ur Rehman

Ibrahim Umer

Muhammad Saad

-----

-----

-----

## **ABBREVIATIONS**

<b>SRS</b>	Software Require Specification
<b>PC</b>	Personal Computer
<b>FR</b>	Functional Requirement
<b>UC</b>	Use Case
<b>DFD</b>	Data Flow Diagram
<b>FT</b>	Functional Testing
<b>ERD</b>	Entity Relationship Diagram

# **TABLE OF CONTENTS**

<b><u>1</u></b>	<b><u>Introduction</u></b> .....	<b>1</b>
1.1	<u>Brief</u> .....	1
1.1.1	<u>Outcome:</u> .....	1
1.1.2	<u>Tools:</u> .....	1
1.1.3	<u>Methodology:</u> .....	2
1.2	<u>Relevance to Course Modules</u> .....	2
1.2.1	<u>Mobile Application Development:</u> .....	2
1.2.2	<u>Human Computer Interaction:</u> .....	3
1.2.3	<u>Report Writing Skills:</u> .....	3
1.2.4	<u>Machine Learning:</u> .....	3
1.2.5	<u>Artificial Intelligence:</u> .....	3
1.2.6	<u>Database:</u> .....	3
1.2.7	<u>Software Engineering:</u> .....	4
1.3	<u>Project Background</u> .....	4
1.4	<u>Literature Review</u> .....	4
1.5	<u>Analysis from Literature Review</u> .....	5
1.5.1	<u>Integration of Advanced Security Measures:</u> .....	5
1.5.2	<u>Focus on User Experience and Accessibility:</u> .....	5
1.5.3	<u>Addressing Cost and Scalability Challenges</u> .....	5
1.5.4	<u>Real-Time Feedback and Analytics</u> .....	5
1.5.5	<u>Enhanced Data Privacy and Compliance</u> .....	6
1.6	<u>Methodology and Software Lifecycle for This Project</u> .....	6
1.6.1	<u>Rationale behind Selected Methodology</u> .....	6
1.6.2	<u>Modular Development:</u> .....	6
1.6.3	<u>Flexibility and Adaptability:</u> .....	6
1.6.4	<u>Clearer Communication:</u> .....	7
1.6.5	<u>Risk Mitigation:</u> .....	7
1.6.6	<u>Customer Involvement:</u> .....	7
1.6.7	<u>Continuous Improvement:</u> .....	7
<b><u>2</u></b>	<b><u>Problem Definition</u></b> .....	<b>8</b>
2.1	<u>Problem Statement</u> .....	8
2.2	<u>Deliverables and Development Requirements</u> .....	8
2.2.1	<u>Identity Verification System:</u> .....	9
2.2.2	<u>Real-time Monitoring:</u> .....	9
2.2.3	<u>Automated Attendance:</u> .....	9
2.2.4	<u>User-friendly Interface:</u> .....	9
2.2.5	<u>Reports and Analytics:</u> .....	9
2.3	<u>Current System (if applicable to your project)</u> .....	10
<b><u>3</u></b>	<b><u>Requirement Analysis</u></b> .....	<b>11</b>



3.1	<u>Use Cases Diagram</u> .....	11
3.2	<u>Detailed Use Case</u> .....	12
3.2.1	<u>Login</u> .....	12
3.2.2	<u>Add Quiz</u> .....	12
3.2.3	<u>View Cheating Reports</u> .....	13
3.2.4	<u>Upload Assignment</u> .....	13
3.2.5	<u>Upload Marks</u> .....	14
3.2.6	<u>Login (Student)</u> .....	14
3.2.7	<u>Attempt Quiz</u> .....	15
3.2.8	<u>Pending Assignments</u> .....	15
3.2.9	<u>Upload Assignment Solutions</u> .....	16
3.2.10	<u>Check Result</u> .....	16
3.2.11	<u>Detect Cheating</u> .....	17
3.2.12	<u>Generate Reports</u> .....	17
3.2.13	<u>Grade Quiz</u> .....	17
3.3	<u>Functional Requirements</u> .....	19
3.3.1	<u>Signup (Instructor)</u> .....	19
3.3.2	<u>Login (Instructor)</u> .....	20
3.3.3	<u>Add Quiz</u> .....	20
3.3.4	<u>Add Assignment</u> .....	21
3.3.5	<u>Grade Assignments</u> .....	22
3.3.6	<u>Signup (Student)</u> .....	22
3.3.7	<u>Login (student)</u> .....	23
3.3.8	<u>Attempt Quiz</u> .....	23
3.3.9	<u>View Result</u> .....	24
3.3.10	<u>Facial Verification</u> .....	24
3.3.11	<u>Store Quiz and Assignments data</u> .....	24
3.3.12	<u>Detection of cheating behaviors</u> .....	25
3.3.13	<u>Generate cheating incidents reports</u> .....	25
3.4	<u>Non-Functional Requirements</u> .....	26
3.4.1	<u>Usability</u> .....	26
3.4.2	<u>Performance</u> .....	27
3.4.3	<u>Security</u> .....	27
<b>4</b>	<b><u>Design and Architecture</u></b> .....	<b>28</b>
4.1	<u>System Architecture</u> .....	28
4.1.1	<u>Class Diagram</u> .....	28
4.1.2	<u>Sequence Diagrams</u> .....	29
4.2	<u>Data Representation [Diagram + Description]</u> .....	31
4.2.1	<u>ER Diagram</u> .....	31
4.2.2	<u>Process Flow/Representation</u> .....	32
<b>5</b>	<b><u>Implementation</u></b> .....	<b>33</b>
5.1	<u>Algorithm</u> .....	33
5.1.1	<u>Teacher Registration</u> .....	33

5.1.2	<u>Student registration:</u>	33
5.1.3	<u>Login Algorithm</u>	33
5.1.4	<u>Facial Recognition Algorithm</u>	34
5.1.5	<u>Activity Monitoring Algorithm</u>	34
5.1.6	<u>Cheating Detection Algorithm</u>	34
5.1.7	<u>Quiz Marking Algorithm</u>	34
5.2	<u>External APIs</u>	35
5.3	<u>User Interface</u>	36
5.3.1	<u>Screen Images</u>	36
<b>6</b>	<b><u>Testing and Evaluation</u></b>	<b>40</b>
6.1	<u>Manual Testing</u>	40
6.2	<u>System testing</u>	40
6.3	<u>Unit Testing</u>	41
6.3.1	<u>Signup</u>	41
6.3.2	<u>Login</u>	42
6.3.3	<u>Dashboard</u>	42
6.4	<u>Functional Testing</u>	43
6.4.1	<u>SignUp/ Registration</u>	43
6.4.2	<u>Login</u>	44
6.4.3	<u>Reset Password</u>	44
6.4.4	<u>Quiz Generation</u>	45
6.4.5	<u>Student facial verification</u>	45
6.4.6	<u>Cheating detection and notifications:</u>	45
6.5	<u>Integration Testing</u>	46
<b>7</b>	<b><u>Conclusion and Future Work</u></b>	<b>47</b>
7.1	<u>Conclusion</u>	47
7.2	<u>Future Work</u>	48
<b>8</b>	<b><u>References</u></b>	<b>50</b>

## **LIST OF FIGURES**

Fig 3.1 Use Case Diagram .....	11
Fig 4.1 Class Diagram.....	28
Fig 4.2 Login SD.....	29
Fig 4.3 Student's SD.....	29
Fig 4.4 Instructor's SD .....	30
Fig 4.5 ER Diagram .....	31
Fig 4.6 Activity Diagram .....	32
Fig 5.1 Home Screen.....	36
Fig 5.2 log in .....	37
Fig 5.3 Signup .....	38
Fig 5.4 Dashboard .....	39

## **LIST OF TABLES**

Table 2.1 Development Requirements .....	9
Table 3.1 View Reports .....	18
Table 3.2 Attempting Quiz.....	19
Table 3.3 Signup FR .....	19
Table 3.4 Login FR .....	20
Table 3.5 Creating Quiz FR .....	20
Table 3.6 Uploading Assignment FR.....	21
Table 3.7 Viewing Cheating Reports.....	21
Table 3.8 Marking Assignment FR.....	22
Table 3.9 Students SignUp FR.....	22
Table 3.10 Student's Login FR.....	23
Table 3.11 Attempting Quiz FR.....	23
Table 3.12 Results FR.....	24
Table 3.13 Facial Verification FR.....	24
Table 3.14 Storing Data .....	24
Table 3.15 Detecting Cheating.....	25
Table 3.16 Generate Reports.....	25
Table 5.1 External APIs .....	35
Table 6.1 System Testing.....	40
Table 6.2 SignUp Testing .....	41
Table 6.3 LogIn Testing.....	42
Table 6.4 Dashboard Testing .....	42
Table 6.5 Registration FT .....	43
Table 6.6 LogIn Testing.....	44
Table 6.7 Reset Password Testing .....	44
Table 6.8 Quiz Generation Testing .....	45
Table 6.9 Facial Verification Testing .....	45
Table 6.10 Notifications Testing.....	45
Table 6.11 Integration Testing .....	46

# **1 Introduction**

Smart Guard is a special app that helps schools and colleges keep online exams fair and secure. It uses clever technology to make sure students can't cheat during their exams. For example, it can recognize each student's face to make sure they're the ones taking the test. It also keeps an eye on what students are doing on their computers to make sure they're not looking up answers. Smart Guard makes exams safer and more honest for everyone.

## **1.1 Brief**

In the era of intelligent applications, our project, "Smart Guard Pro," is a cutting-edge AI- driven mobile application designed to address two critical challenges encountered by educational institutes in an online environment: automated attendance management and cheating prevention in a controlled environment. Harnessing the power of artificial intelligence and modern mobile technologies, SmartGuard Pro seeks to revolutionize examination processes, elevate security standards, and ensure fairness in assessments.

With the help of Artificial Intelligence and machine learning models, our application will provide a platform for instructors to conduct quizzes, assignments, and examinations in a fair environment where the activity of the students will be monitored.

### **1.1.1 Outcome**

The development of Smart Guard Pro is a testament to the successful integration of cutting-edge tools and an adaptive methodology that collectively enhance the project's effectiveness. Utilizing Flutter as the primary development framework enables the creation of high-performance applications that function seamlessly across multiple platforms, ensuring that users have a consistent and engaging experience, whether they are using a smartphone, tablet, or desktop. Coupled with Firebase for backend support, the application benefits from robust features like real-time data synchronization, secure user authentication, and scalable cloud storage. This combination of tools allows for rapid development cycles and the implementation of complex functionalities without compromising performance.

### **1.1.2 Tools**

The development of Smart Guard Pro utilizes a robust set of tools that enhance functionality, collaboration, and efficiency throughout the project lifecycle. Flutter serves as the primary development framework, enabling the creation of natively compiled applications for mobile, web, and desktop from a single codebase. This not only accelerates development but also ensures a consistent user experience across multiple platforms. For backend services, Firebase is employed, providing essential features such as real-

time databases, user authentication, and cloud storage. These capabilities are crucial for managing user data securely and facilitating seamless interactions within the app. Additionally, Git is utilized for version control, allowing the development team to maintain code integrity, track changes, and collaborate efficiently. Git ensures that all contributions are integrated smoothly, fostering a streamlined development process. Supporting tools, such as design software like Figma for UI/UX design, further enhance the development experience by enabling designers to create and share prototypes effectively. Collectively, these tools provide a solid foundation for developing a feature-rich and user-friendly application.

### **1.1.3 Methodology**

The project follows an Agile methodology, characterized by its iterative approach and adaptability to changing project requirements. This methodology promotes regular collaboration and feedback among stakeholders, allowing the development team to quickly respond to user needs and make necessary adjustments throughout the development process. Agile practices, such as sprint planning, daily stand-ups, and retrospective meetings, foster a culture of continuous improvement and open communication. By breaking the project into manageable increments or "sprints," the team can focus on delivering specific features and functionalities within set timeframes, enhancing productivity and maintaining high quality. This methodology not only helps in identifying potential issues early in the development cycle but also ensures that the final product is closely aligned with user expectations and industry standards. Overall, the Agile methodology empowers the Smart Guard Pro team to innovate and deliver a reliable solution that effectively addresses the challenges of online examination integrity.

## **1.2 Relevance to Course Modules**

SmartGuard Pro is deeply intertwined with the principles and concepts covered in our course. It represents a practical application of theoretical knowledge, showcasing how technology can be harnessed to solve real-world problems.

### **1.2.1 Mobile Application Development**

In the Mobile Application Development module, we gained insights into the various technologies and frameworks available for creating mobile applications. This course covered key aspects such as user interface design, backend integration, and performance optimization. The knowledge acquired here is directly applicable to Smart Guard Pro, as we employed Flutter to develop a cross-platform mobile application that offers a seamless user experience across different devices. This hands-on experience has equipped us with the skills necessary to create robust applications that meet user needs and adhere to industry standards.

### **1.2.2 Human Computer Interaction**

The Human-Computer Interaction (HCI) course provided us with valuable principles regarding the design and evaluation of user interfaces to enhance usability. We learned how to create intuitive and accessible interfaces that cater to diverse user populations. For Smart Guard Pro, this understanding was instrumental in designing an engaging and user-friendly interface that facilitates smooth interactions for both instructors and students. By focusing on user experience, we ensured that the application is not only functional but also easy to navigate, ultimately improving user satisfaction and efficiency.

### **1.2.3 Report Writing Skills**

In the Report Writing Skills course, we developed essential competencies in structuring and articulating information effectively. This module taught us the importance of clear communication, proper formatting, and critical analysis in report writing. These skills are vital for documenting the development process of Smart Guard Pro, as we need to convey complex technical concepts and project outcomes comprehensively to various stakeholders. The ability to produce well-organized reports ensures that our project is presented professionally, making it easier for others to understand and assess our work.

### **1.2.4 Machine Learning**

We utilized knowledge from machine learning courses to develop the core algorithm for mushroom classification. This involved applying various machine learning models, training methodologies, and assessment techniques to enhance the accuracy of mushroom identification.

### **1.2.5 Artificial Intelligence**

Our project extensively utilized principles from artificial intelligence, mainly focusing on concepts such as feature selection, model interpretation, and avoiding biases. By leveraging insights gained from AI studies, we aimed to ensure transparency and reliability in the app's decision-making process.

### **1.2.6 Database**

As part of our project, we have created a strong data management system for storing mushroom data and user interactions. Our expertise in database systems has helped us design an effective and safe data storage solution that ensures the security and confidentiality of user information while maintaining the overall integrity of data.

### **1.2.7 Software Engineering**

Throughout the project lifecycle, we adhered to software engineering principles to ensure the development of a high-quality product. This included requirements analysis, system architecture design, iterative development, rigorous testing, and seamless deployment, resulting in a user-friendly and maintainable application.

## **1.3 Project Background**

Traditional examination methods, reliant on in-person invigilation and paper-based assessments, are ill-suited to the demands of online education. In this new paradigm, students are afforded unprecedented access to digital resources and communication channels, presenting opportunities for academic dishonesty and compromising the credibility of examinations.

Recognizing these challenges, the need for innovative solutions to safeguard the integrity of online examinations has become paramount. SmartGuard Pro emerges as a response to this pressing need, offering a comprehensive suite of features designed to mitigate the risks. By harnessing the power of artificial intelligence and modern mobile technologies, SmartGuard Pro seeks to revolutionize the examination process. The project's inception was driven by a deep-seated commitment to uphold academic integrity and ensure equitable assessment practices in the digital age.

## **1.4 Literature Review**

The Literature Review serves as a comprehensive overview of current trends, research findings, and existing products related to online examination systems and academic integrity solutions. With the rise of remote learning, there has been an increased focus on developing systems that can effectively monitor and assess student performance in a virtual environment. Various studies have highlighted the challenges educators face in maintaining the integrity of assessments, particularly with the ease of accessing information online. Research has explored advanced technologies such as biometric authentication, including facial recognition and voice detection, as viable solutions for preventing cheating during online exams.

Several established platforms, such as ProctorU, ExamSoft, and Moodle, have emerged in the educational landscape, offering features that address the need for secure online assessments. ProctorU provides live proctoring services, employing real human proctors to monitor students in real-time. ExamSoft offers a secure exam delivery system with offline capabilities, while Moodle has incorporated various plugins to enhance its assessment functionalities. However, despite the advancements, many existing systems still face limitations, including high costs, user accessibility issues, and challenges related to scalability. This literature review serves as a foundation for understanding how Smart Guard Pro can fill existing gaps in the market and contribute to more effective online examination practices.



## **1.5 Analysis from Literature Review**

The analysis of the literature reveals several critical insights that inform the development of Smart Guard Pro and its positioning within the existing landscape of online examination solutions.

### **1.5.1 Integration of Advanced Security Measures**

Current trends in the literature emphasize the necessity of integrating advanced security measures into online examination systems. This includes real-time monitoring, biometric verification, and the use of artificial intelligence to detect suspicious behavior. Smart Guard Pro aligns with this trend by incorporating cutting-edge features like facial recognition and voice detection to authenticate student identities and monitor their actions during assessments. By leveraging these technologies, Smart Guard Pro enhances the authenticity and integrity of the examination process, addressing the common vulnerabilities identified in existing platforms.

### **1.5.2 Focus on User Experience and Accessibility**

An emerging theme in the literature is the importance of user experience and accessibility in online examination systems. SmartGuard Pro acknowledges the significance of these factors by prioritizing a user-friendly interface and cross-platform compatibility, ensuring seamless access and navigation for both instructors and students.

### **1.5.3 Addressing Cost and Scalability Challenges**

Finally, the literature reveals that many current solutions are often costly and may not scale effectively to accommodate large user bases. Smart Guard Pro aims to provide a more cost-effective solution by utilizing cloud-based services and a scalable architecture that can grow with educational institutions' needs. This approach not only reduces implementation costs but also ensures that the application can adapt to varying user demands, a factor that is crucial for maintaining competitiveness in the evolving educational landscape.

### **1.5.4 Real-Time Feedback and Analytics**

Recent studies have indicated that real-time feedback during exams can significantly enhance the learning experience. Existing platforms often provide post-exam results without immediate feedback, which may hinder students from understanding their mistakes. Smart Guard Pro plans to incorporate features that allow instructors to provide instant feedback during quizzes and exams. This capability can help students learn more effectively and enable instructors to adjust their teaching methods based on real-time performance data.

### **1.5.5 Enhanced Data Privacy and Compliance**

Data privacy has become a crucial concern in the digital age, especially in educational settings where sensitive student information is involved. Many current systems do not adequately address privacy issues, leading to potential legal and ethical challenges. Smart Guard Pro prioritizes data privacy by implementing strict security measures and ensuring compliance with regulations like GDPR. By focusing on protecting user data, Smart Guard Pro not only builds trust among its users but also sets a standard for ethical practices in online examination systems.

## **1.6 Methodology and Software Lifecycle for This Project**

In our project, we will utilize the incremental model for software development. This model is characterized by the development of modules in increments or chunks, which allows us to deliver our project in the form of working functionalities. Each iteration in the incremental model goes through a series of steps, including defining conditions, generating ideas, coding, and performing comparison and analysis. After achieving the initial functionality, each subsequent release builds upon the previous version, adding new capabilities. The incremental model is particularly suited to our project, as it enables us to develop and refine features iteratively. If additional requirements arise after the first version, we can create new iterations of the application to incorporate them seamlessly.

### **1.6.1 Rationale behind Selected Methodology**

In our project, the choice of Object-Oriented Programming (OOP) and the Agile Incremental Model is driven by several factors:

### **1.6.2 Modular Development**

Object-Oriented Programming emphasizes modular development, allowing us to organize code into reusable objects. This modularity facilitates easier maintenance, testing, and scalability of the application. By breaking the project into distinct modules, we can deliver working components incrementally, aligning perfectly with the principles of the incremental model. Each module can be developed, tested, and refined independently, leading to a more manageable development process.

### **1.6.3 Flexibility and Adaptability**

The Agile Incremental Model emphasizes iterative development and responsiveness to changes in requirements. OOP supports this flexibility through its principles of encapsulation and abstraction, enabling us to adapt quickly to evolving project needs. For instance, if a specific feature needs adjustment, we can modify just that module without having to overhaul the entire system. This adaptability is crucial in a dynamic environment where requirements may change based on stakeholder feedback or shifting.

#### **1.6.4 Clearer Communication**

Both OOP and Agile methodologies prioritize clear communication and collaboration among team members. OOP's well-defined objects and classes provide a common language for developers, making it easier to understand the code's structure and functionality. Meanwhile, Agile's emphasis on regular meetings and feedback loops ensures that stakeholders remain engaged and informed about the project's progress. This clear communication helps align expectations and fosters a collaborative working environment.

#### **1.6.5 Risk Mitigation**

The Agile Incremental Model enables us to identify and address risks early in the development process through frequent iterations and feedback loops. By delivering smaller, functional parts of the application, we can uncover potential issues before they escalate into larger problems. OOP's modular structure further aids in risk mitigation by allowing us to isolate and manage risks within specific components. If a particular module faces challenges, it minimizes the impact on the overall project, ensuring a more resilient development process.

#### **1.6.6 Customer Involvement**

Agile methodologies encourage continuous customer involvement throughout the development lifecycle, which ensures that the final product aligns with user needs and expectations. OOP enhances this process by enabling the quick creation of prototypes and mockups, allowing stakeholders to visualize and interact with features early in development. This direct feedback loop is invaluable for validating requirements and making necessary adjustments to better meet user needs.

#### **1.6.7 Continuous Improvement**

Both OOP and Agile methodologies promote a culture of continuous improvement through regular reflection and adaptation. OOP emphasizes code reusability and refactoring, which contributes to maintaining high-quality code. Similarly, Agile focuses on delivering value iteratively, allowing us to refine and enhance the application's functionality over time. By fostering a mindset of ongoing improvement, our project can evolve to better serve users and adapt to new challenges that arise throughout the development process.

## **2 Problem Definition**

In schools and colleges, when students take tests on computers, it's sometimes easy for them to cheat. They might look up answers on the internet or get help from friends. This makes it hard for teachers to know if students are really doing well. Smart Guard is trying to solve this problem. It's a special app that watches students while they take tests online. It makes sure they don't cheat by using smart technology like face recognition to check if it's really the student taking the test. Smart Guard helps teachers trust that students are doing their tests honestly and fairly. The outcome we're aiming for is to make online exams more secure and trustworthy, so that students are evaluated fairly based on their own knowledge and abilities, and teachers can have confidence in the integrity of the assessment process.

### **2.1 Problem Statement**

In educational institutions, the prevalence of online examinations poses a significant challenge of ensuring academic integrity and fairness. With the ease of accessing external resources and collaborating with peers during online tests, the risk of cheating has become a concern for educators. This undermines the credibility of assessment outcomes and creates uncertainty about students' true understanding and proficiency. The problem at hand is to develop a solution that effectively prevents cheating during online examinations, thereby restoring trust in the evaluation process and providing educators with reliable insights into students' learning outcomes. The solution should leverage innovative technologies, such as Smart Guard, to monitor and deter cheating behaviors while maintaining user privacy and accessibility.

The widespread adoption of online examinations has brought forth challenges in maintaining academic integrity and accurate attendance tracking. Students taking online assessments have easy access to external resources, making it difficult for instructors to ensure fair, independent completion of exams. This undermines the credibility of evaluation results and creates ambiguity regarding students' actual knowledge.

Smart Guard Pro addresses this problem by creating an AI-driven solution that prevents cheating and ensures accurate, automated attendance tracking in online examinations. This system aims to uphold academic integrity, provide educators with reliable insights into students' understanding, and maintain the fairness of assessments.

### **2.2 Deliverables and Development Requirements**

Details about different modules of the projects and the technologies used for their development.

### **2.2.1 Identity Verification System**

A facial recognition-based system to authenticate each student's identity before and during the exam, ensuring the student registered for the exam is the one taking it.

### **2.2.2 Real-time Monitoring**

Continuous monitoring of students' activity, using techniques such as eye tracking, screen capture, and voice detection, to detect any unauthorized behavior.

### **2.2.3 Automated Attendance**

Built-in functionality for attendance tracking, logging each student's participation in exams.

### **2.2.4 User-friendly Interface**

An intuitive interface accessible on multiple devices, providing a seamless experience for both instructors and students.

### **2.2.5 Reports and Analytics**

Tools for generating reports and insights post-exam, allowing instructors to review flagged incidents or suspicious behaviors.

*Table 2.1 Development Requirements*

S. No	Technology	Version	Rationale
1	VS Code	Latest	Versatile Code Editor
2	Flutter	3.13	Framework for developing cross platform mobile applications.
3	Node.js	latest	JavaScript runtime environment for server-side scripting
4	Express.js	latest	Framework for node.js
5	Firebase	latest	No SQL database for efficient storage.
6	python	3.11	Programming language
7	Tensorflow lite	latest	ML library for mobile applications

### **2.3 Current System (if applicable to your project)**

In the context of online proctoring, several existing platforms, such as ProctorU, ExamSoft, and Moodle, offer remote monitoring solutions, but they have limitations in terms of real-time attendance tracking and comprehensive user-friendly monitoring. These systems often rely on static measures like webcam monitoring, which may not be as effective at detecting all forms of cheating, and their interfaces may lack the accessibility and ease needed for widespread adoption across devices.

Smart Guard Pro improves upon these solutions by introducing advanced AI-driven monitoring features, such as facial recognition and behavior tracking, alongside a simple, cross-platform interface. This approach prioritizes both user experience and exam security, making it easier for educational institutions to manage online assessments effectively. Through its automated attendance feature and enhanced monitoring, Smart Guard Pro seeks to fill the gaps left by existing solutions, offering a more comprehensive solution tailored for educational integrity in online exams

### 3 Requirement Analysis

Requirement analysis is a crucial phase in software development where project needs and objectives are meticulously gathered, documented, and analyzed.

#### 3.1 Use Cases Diagram

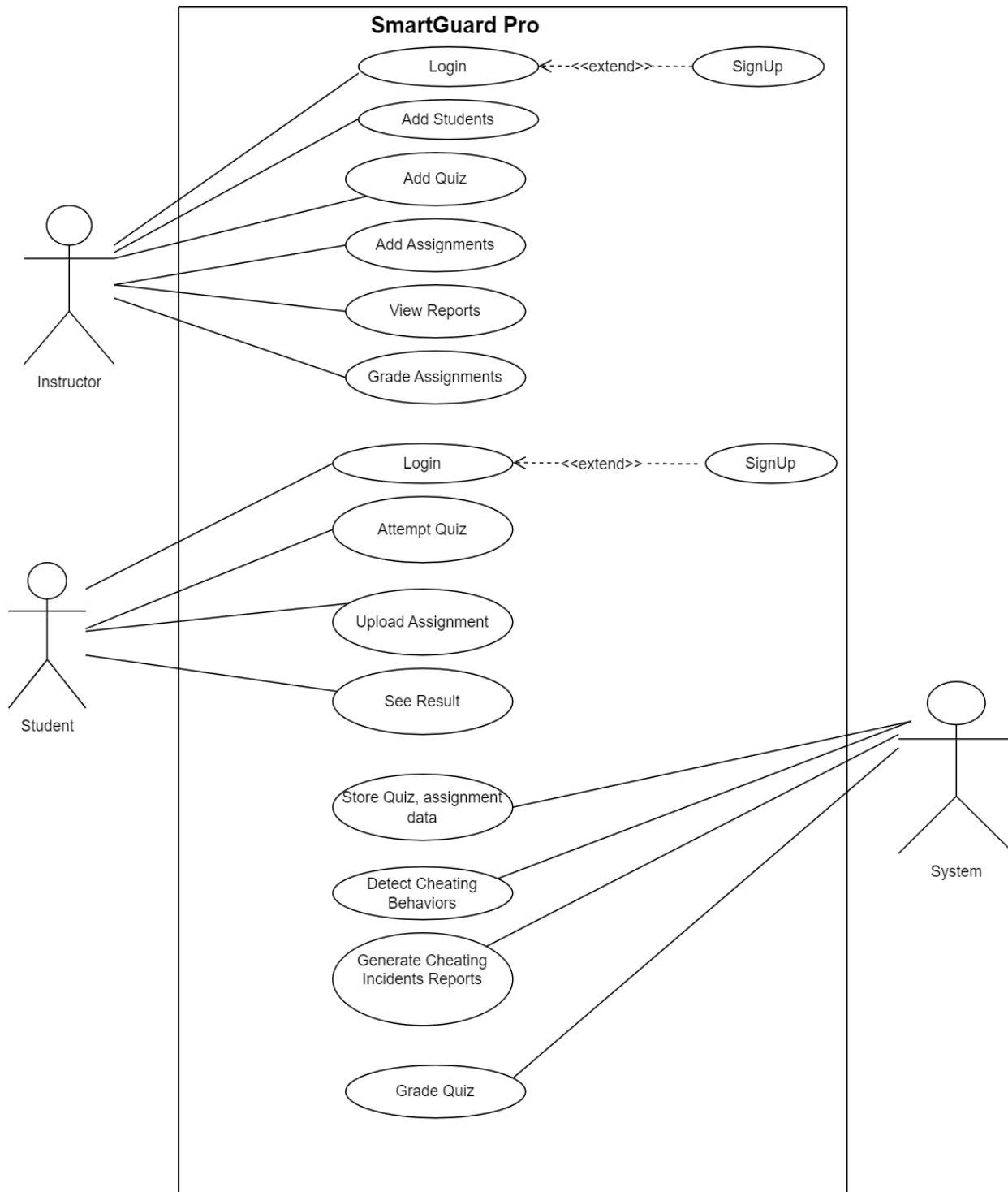


Fig 3.1 Use Case Diagram

## **3.2 Detailed Use Case**

This section gives the explanation of each of the use with their precondition and post conditions.

### **3.2.1 Login**

Allows the instructor to securely log in and access the Smart Guard Pro system.

- **Actor:** Instructor
- **Description**  
The instructor logs into Smart Guard Pro to access their dashboard, which includes functionalities like quiz creation, assignment uploads, and monitoring tools. After entering their unique username and password, the system verifies their credentials. Upon successful authentication, the instructor is granted access to the platform's features and tools, designed to facilitate online assessments and monitor academic integrity.
- **Preconditions**  
The instructor has a registered account with a unique username and password.
- **Postconditions**  
The instructor gains access to the system's features, such as adding quizzes, viewing reports, and managing assignments.

### **3.2.2 Add Quiz**

Enables the instructor to create and schedule quizzes for students.

- **Actor:** Instructor
- **Description**  
The instructor can create a new quiz by specifying the quiz title, description, start date, duration, and any relevant settings (e.g., time limit, question randomization). This quiz setup allows instructors to manage test parameters and configure options that enhance exam security. Once details are entered, the instructor saves the quiz, making it available for students to access on the scheduled date. The system stores all quiz details in the database and assigns it to the correct student group.
- **Preconditions**  
The instructor is logged in, and the necessary quiz information (title, date, duration) is prepared.



- **Postconditions**

The quiz is saved in the system, visible to students when they log in on the scheduled date.

### **3.2.3 View Cheating Reports**

Provides the instructor with detailed reports on flagged cheating incidents during quizzes.

- **Actor:** Instructor

- **Description**

Instructors can access a detailed report that provides insights into potential cheating incidents during quizzes. The report includes information such as flagged behavior, timestamps, and student details, allowing instructors to review and analyze the integrity of each exam session. This report enables instructors to identify any suspicious activity, ensuring fair assessment by addressing cases of academic dishonesty as necessary.

- **Preconditions**

The instructor is logged in and has conducted quizzes with real-time monitoring enabled.

- **Postconditions**

The instructor has viewed detailed reports on flagged incidents, aiding in maintaining academic integrity.

### **3.2.4 Upload Assignment**

Allows the instructor to upload assignments with instructions and deadlines for students.

- **Actor:** Instructor

- **Description**

The instructor can upload new assignments for students to complete, including assignment details, instructions, and deadlines. By providing this material digitally, students are notified about their assignments and can easily download them through the system. This feature simplifies assignment management, ensuring all relevant details are available in one centralized location.

- **Preconditions**

The instructor is logged in and has prepared the assignment files and instructions.

- **Postcondition**

The assignment is available for students to view, download, and complete by the specified due date.

### **3.2.5 Upload Marks**

Enables the instructor to input and save student grades for quizzes and assignments.

- **Actor:** Instructor

- **Description**

This feature allows instructors to upload the scores or grades for students following the completion of quizzes or assignments. Instructors can select the respective exam or assignment and input each student's marks, which are then securely saved in the system. This enables students to access their grades and receive feedback on their performance through the platform.

- **Preconditions**

The instructor is logged in and has completed grading quizzes or assignments.

- **Postconditions**

Students can view their marks under the respective assignment or quiz in their accounts.

### **3.2.6 Login (Student)**

Allows the student to securely log in to their Smart Guard Pro account.

- **Actor:** Student

- **Description**

The student accesses their Smart Guard Pro account by entering their unique username and password. Upon successful login, they are directed to their dashboard, where they can view quizzes, assignments, and results. This login step ensures that only verified students can access the platform's assessment and assignment features, maintaining secure access for each user.

- **Preconditions**

The student has a registered account with a unique username and password.

- **Postconditions**

The student gains access to features like attempting quizzes, viewing pending assignments, and checking results.

### **3.2.7 Attempt Quiz**

Enables the student to participate in scheduled quizzes under monitoring.

- **Actor:** Student

- **Description**

Students can participate in a scheduled quiz by accessing it through their dashboard at the specified time. The system verifies their identity using facial recognition before starting and may monitor their activity during the quiz to prevent cheating. This setup promotes a secure and fair testing environment, ensuring that each student is assessed based on their independent efforts.

- **Preconditions**

The student is logged in, and the quiz is available and scheduled for access.

- **Postconditions**

The student completes the quiz, and their answers are recorded in the system for grading.

### **3.2.8 Pending Assignments**

Displays a list of current assignments for the student to complete.

- **Actor:** Student

- **Description**

Students can view a list of all current assignments that have been assigned but are not yet completed. Each assignment displays its title, description, due date, and any additional instructions provided by the instructor. This feature helps students track deadlines and ensure they meet all course requirements within the stipulated timeframe.

- **Preconditions**

The student is logged in, and assignments have been uploaded by the instructor.

- **Postconditions**

The student can view and download assignment details, enabling them to plan their work accordingly.

### **3.2.9 Upload Assignment Solutions**

Allows the student to submit their completed assignments online.

- **Actor:** Student
- **Description**  
Students can submit completed assignments by uploading their solutions directly through the system. This feature allows students to attach files or text responses, which are then saved in the system for instructors to review. By submitting assignments digitally, students can confirm that their work is submitted on time and receive feedback from their instructors.
- **Preconditions**  
The student is logged in, has completed the assignment, and the submission window is open.
- **Postconditions**  
The submitted assignment is saved in the system and marked as completed in the student's dashboard.

### **3.2.10 Check Result**

Enables the student to view their grades and feedback for quizzes and assignments.

- **Actor:** Student
- **Description**  
Students can view their results or grades for completed quizzes and assignments. By accessing the "Results" section, students can see their performance, along with any comments or feedback from instructors, if applicable. This feature provides students with insights into their academic progress and allows them to review their achievements and areas needing improvement.
- **Preconditions**  
The student is logged in, and the instructor has uploaded the results.
- **Postconditions**  
The student can view their grades and any feedback provided, promoting transparency in the assessment process.

### **3.2.11 Detect Cheating**

Monitors student behavior during quizzes to detect potential cheating.

- **Actor:** System
- **Description**

The system actively monitors students during quizzes, using technologies like facial recognition, eye-tracking, and behavior analysis to detect potential cheating behaviors. If any suspicious activity is identified, the system flags the incident for later review by instructors, ensuring that exams are conducted fairly and transparently.
- **Preconditions**

A quiz is in progress, with monitoring enabled.
- **Postconditions**

Any flagged incidents are recorded in the system, allowing instructors to review and verify academic integrity.

### **3.2.12 Generate Reports**

Creates detailed reports summarizing student activities and flagged incidents.

- **Actor:** System
- **Description**

After each quiz session, the system generates a comprehensive report summarizing student activities, including any flagged behaviors that may indicate cheating. These reports contain timestamps, flagged actions, and detailed data to support instructors in evaluating each student's conduct. By providing this analysis, the system supports instructors in maintaining fair assessment practices.
- **Preconditions**

The quiz has been completed, and monitoring data is available.
- **Postconditions**

The report is generated and stored, accessible to instructors for review and analysis.

### **3.2.13 Grade Quiz**

Automatically grades student quizzes based on predefined answer keys.

- **Actor:** System
- **Description**

The system automatically grades quizzes based on predefined answer keys provided by instructors. This automation speeds up the evaluation process, allowing students to receive timely feedback on their performance. The system then securely saves each student's score in their profile for them to view.
- **Preconditions**

The quiz has been completed, and answer keys are available in the system.
- **Postconditions**

The quiz is graded, scores are saved in the student profiles, and instructors can review or adjust grades if necessary.

*Table 3.1 View Reports*

Use Case ID:	UC5
Use Case Name:	View Reports
Actors:	Instructor
Description:	Instructor views generated reports.
Trigger:	Instructor decides to view reports.
Preconditions:	Instructor is logged in
Postconditions:	Reports are displayed.
Normal Flow:	Instructor clicks on the view reports section. Selects the type of report to view.
Alternative Flows:	None
Exceptions:	None
Business Rules	Data Privacy and Security: Description: Access to student data and performance reports must adhere to privacy and security regulations.
Assumptions:	None

Table 3.2 Attempting Quiz

Use Case ID:	UC9
Use Case Name:	Attempt Quiz
Actors:	Student
Description:	Student attempts a quiz.
Trigger:	Student decides to attempt a quiz
Preconditions:	Student is logged in.
Postconditions:	Quiz attempt is recorded.
Normal Flow:	Student navigates to available quizzes. Selects a quiz to attempt. Answers quiz questions. Submits the quiz
Alternative Flows:	None
Exceptions:	None
Business Rules	Quiz Availability, Description: Quizzes may have specific availability windows, and students can only attempt them during those periods.
Assumptions:	None

### 3.3 Functional Requirements

The following are the functional requirements for all the use cases.

#### 3.3.1 Signup (Instructor)

Table 3.3 Signup FR

Identifier	FR-001
Title	User Authentication
Requirement	The instructor shall be able to create an account with a unique username and password. Usernames created during signup must be unique system wide.
Source	User
Rationale	To ensure each instructor has distinct identity, access the system securely and protect their account from unauthorized access.
Dependencies	None
Priority	High

### **3.3.2 Login (Instructor)**

*Table 3.4 Login FR*

Identifier	FR-002
Title	Login(instructor)
Requirement	Instructors shall be able to log in securely using their credentials. After a specified number of failed login attempts, the system shall temporarily lock the instructor account.
Source	User
Rationale	To provide secure access to Instructor accounts and to prevent unauthorized access through brute force attacks.
Business Rule	None
Dependencies	FR-001
Priority	Medium

### **3.3.3 Add Quiz**

*Table 3.5 Creating Quiz FR*

Identifier	FR-003
Title	Add Quiz
Requirement	Instructors shall be able to create and add a new quiz to the system. Instructors should specify details for the quiz, including title, questions, and time limits. Instructors shall be able to set the availability of quizzes for students (e.g., start and end time).
Source	User
Rationale	To customize quizzes according to the requirements and to control when students can access and take quizzes.
Dependencies	FR-002
Priority	High



### **3.3.4 Add Assignment**

*Table 3.6 Uploading Assignment FR*

Identifier	FR-004
Title	Add Assignment
Requirement	Instructors shall be able to create and add a new assignment to the system. Instructors should specify details for the assignment, including title, description, and submission dates. The system shall allow students to upload their assignment submissions.
Source	User
Rationale	To enable the submission and collection of student assignments
Dependencies	FR-008
Priority	High

*Table 3.7 Viewing Cheating Reports*

Identifier	FR-005
Title	View Reports
Requirement	Instructors shall be able to access and view reports related to student performance and system analytics. The system shall generate reports on student performance, assessment results, and system analytics.
Source	User
Rationale	To provide data-driven insights to student's performance for Instructors.
Dependencies	FR-002
Priority	Medium

### **3.3.5 Grade Assignments**

*Table 3.8 Marking Assignment FR*

Identifier	FR-006
Title	Grading Assignments
Requirement	Instructors shall be able to upload student's assignments marks. The system shall allow students to access their assignments marks.
Source	User
Rationale	To ensure fairness and consistency in grading.
Dependencies	FR-004, FR-008
Priority	Medium

### **3.3.6 Signup (Student)**

*Table 3.9 Students SignUp FR*

Identifier	FR-007
Title	Signup (student)
Requirement	Students shall be able to create an account with a unique username and password. Usernames created during signup must be unique system wide. Student's picture should be captured to use for facial recognition later.
Source	User
Rationale	To ensure each student has a distinct identity in the system and to prevent their accounts from unauthorized access.
Dependencies	None
Priority	High

### **3.3.7 Login (student)**

*Table 3.10 Student's Login FR*

Identifier	FR-008
Title	Login (student)
Requirement	Students shall be able to log in securely using their credentials. Facial recognition shall be performed during login to verify Student identity. After a specified number of failed login attempts, the system shall temporarily lock the student account.
Source	User
Rationale	To enhance login security, to ensure the students themselves are using their accounts and to prevent their accounts from unauthorized access.
Dependencies	FR-007
Priority	High

### **3.3.8 Attempt Quiz**

*Table 3.11 Attempting Quiz FR*

Identifier	FR-009
Title	Attempt Quiz
Requirement	Students shall be able to start and attempt a quiz after logging in. Real-time monitoring of students shall be initiated when a quiz is started.
Source	User
Rationale	To allow students to participate in quizzes and to detect and prevent cheating behaviors during quiz attempts.
Dependencies	FR-008, FR-003, FR-009, FR-014
Priority	High

### **3.3.9 View Result**

*Table 3.12 Results FR*

Identifier	FR-011
Title	View Result
Requirement	Students shall be able to access and view their quiz and assignment results.
Source	User
Rationale	To provide students with feedback on their performance.
Dependencies	FR-008, FR-009, FR-010
Priority	Medium

### **3.3.10 Facial Verification**

*Table 3.13 Facial Verification FR*

Identifier	FR-012
Title	Facial Verification
Requirement	The system shall perform face verification to ensure the identity of users during login and while attempting quiz.
Source	System
Rationale	To enhance system security using facial verification and to maintain fairness in assessments.
Dependencies	FR-008
Priority	High

### **3.3.11 Store Quiz and Assignments data.**

*Table 3.14 Storing Data*

Identifier	FR-013
Title	Store Quiz and Assignments data
Requirement	The system shall store quiz-related data, including questions, answers, and configurations. The system shall store assignment-related data, including titles, descriptions, and submissions.
Source	User

Rationale	To maintain record of quizzes and assignments
Dependencies	FR-003, FR-004, FR-010
Priority	High

### **3.3.12 Detection of cheating behaviors**

*Table 3.15 Detecting Cheating*

Identifier	FR-014
Title	Detection of cheating behaviors
Requirement	The system shall monitor student activities during quizzes to detect cheating behaviors.
Source	User
Rationale	To maintain the integrity of assessments.
Dependencies	FR-010, FR-009
Priority	High

### **3.3.13 Generate cheating incidents reports.**

*Table 3.16 Generate Reports*

Identifier	FR-015
Title	Generate cheating incidents reports.
Requirement	The system shall generate reports on detected cheating incidents.
Source	User
Rationale	To provide evidence of cheating behaviors.
Dependencies	FR-014
Priority	High

### **3.4 Non-Functional Requirements**

Non-functional requirements (NFRs) specify the quality attributes, standards, and constraints of a system rather than its specific functionality. While functional requirements describe **what** a system does, non-functional requirements outline **how well** the system performs, providing a measure of quality and usability. They help ensure the system is reliable, efficient, and user-friendly.

#### **3.4.1 Usability**

Usability defines how user will interact with the system and how the system will respond to user's requests.

- **USE-1: User Authentication**

The system shall provide a user-friendly interface for the authentication process, ensuring ease of use for all types of users.

The authentication process should take no more than 15 seconds on average for a user to log in successfully.

- **USE-2: Quiz Interaction**

The system interface for quizzes shall be intuitive, allowing users (both instructors and students) to navigate, answer questions, and submit with minimal effort.

The average time taken by a student to complete a quiz should not exceed 3 minutes per question.

- **USE-3: Consistent UI and Navigation**

The application should maintain a consistent layout, color scheme, and navigation flow across all pages. This consistency will help users quickly learn how to use the system and reduce cognitive load.

- **USE-4: Responsive Design**

The application must adapt to various screen sizes and orientations (e.g., mobile phones, tablets, desktops) to provide an optimal experience on all devices.

- **USE-5: Onboarding and Help Documentation**

The application should include an onboarding guide and accessible help sections to assist first-time users in understanding key features and workflows.

### **3.4.2 Performance**

Performance refers to the system's ability to operate efficiently and respond quickly under varying conditions, ensuring a smooth and reliable user experience.

- **PERF-1: System Response Time**

The system shall respond to user interactions (login, quiz attempt, assignment submission) within 3 seconds under normal operating conditions.

95% of user interactions should receive a response within the specified time frame.

- **PERF-2: Concurrent User Handling**

The system shall be able to handle concurrent logins and quiz attempts from a minimum of 100 users without a significant decrease in performance.

The system should maintain response times within the defined threshold with 100 concurrent users.

- **PERF-3: Data Retrieval Time**

The system shall retrieve user-specific data (quiz results and assignments results) within 5 seconds of the user's request.

- **PERF-4: Data Processing Speed**

Real-time monitoring functions, such as cheating detection, should process and analyze video streams with less than a 500ms delay ensuring timely notifications to instructors.

Data retrieval (e.g., loading previous quiz attempts or reports) should take no longer than 3 seconds.

- **PERF-5: Resource Usage Optimization**

The application should use resources (e.g., CPU, memory) efficiently, ensuring it does not exceed 80% CPU or memory utilization under typical load to allow for peak-time flexibility.

### **3.4.3 Security**

Any sensitive or personal information provided by the user should be handled securely and in compliance with relevant privacy regulations.

## 4 Design and Architecture

This section outlines the overall structure and framework of Smart Guard Pro, including key components, interactions, and technologies. It provides a high-level view of how the system is organized to meet functional and non-functional requirements.

### 4.1 System Architecture

System Architecture details the core components, data flow, and communication pathways within Smart Guard Pro. It describes how different modules such as monitoring, reporting, and user management interact to deliver seamless and secure exam management.

#### 4.1.1 Class Diagram

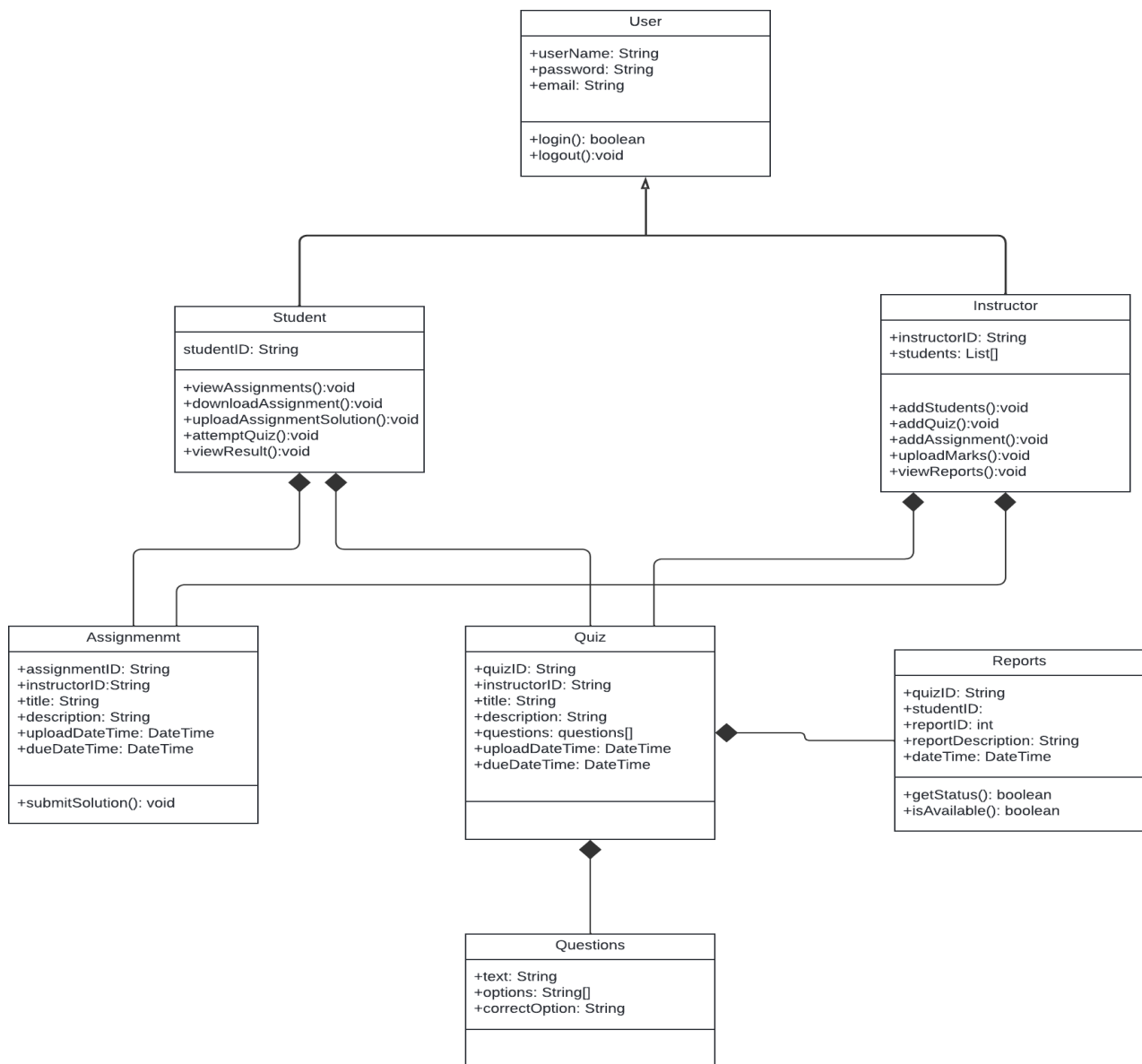


Fig 4.1 Class Diagram



### 4.1.2 Sequence Diagrams

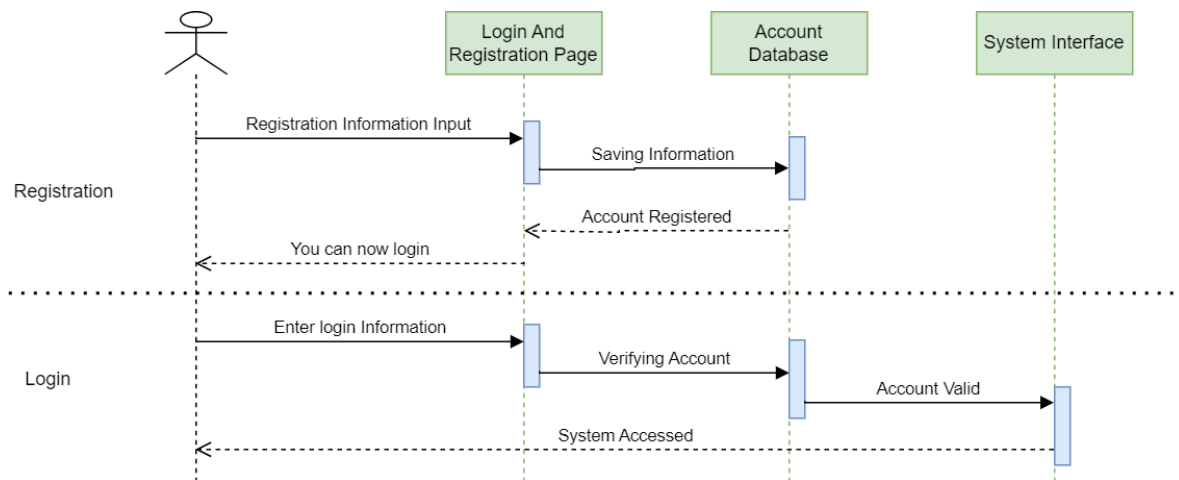


Fig 4.2 Login SD

#### • Student Sequence Diagram

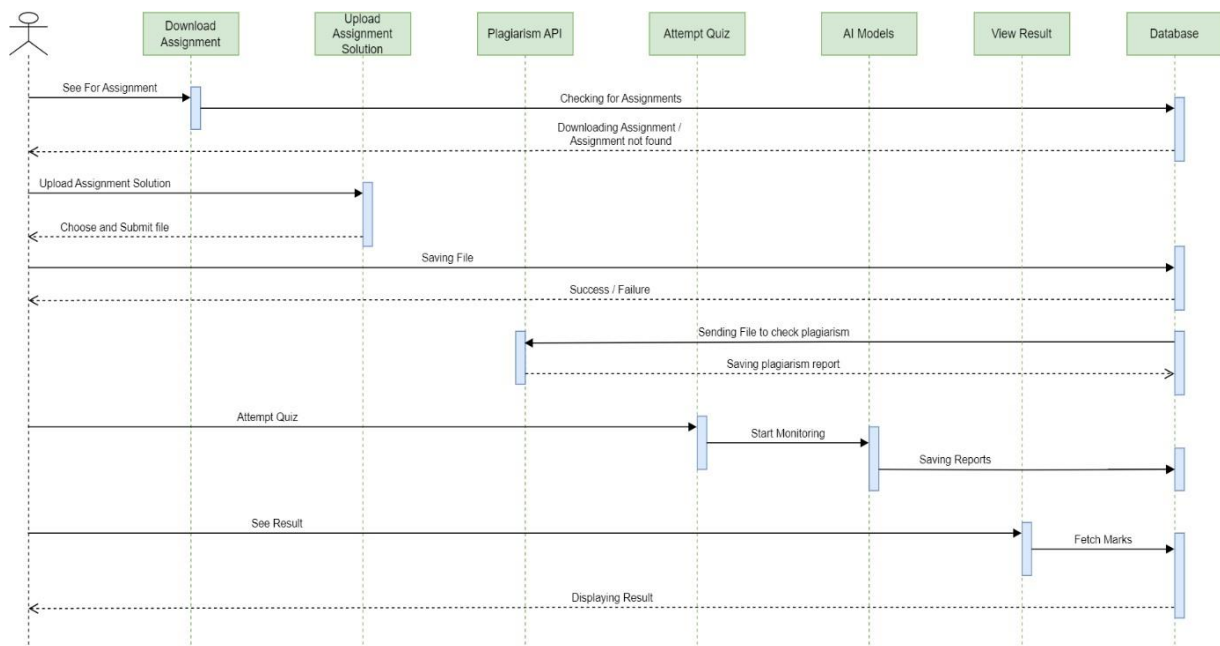


Fig 4.3 Student's SD

- **Teacher Sequence Diagram**

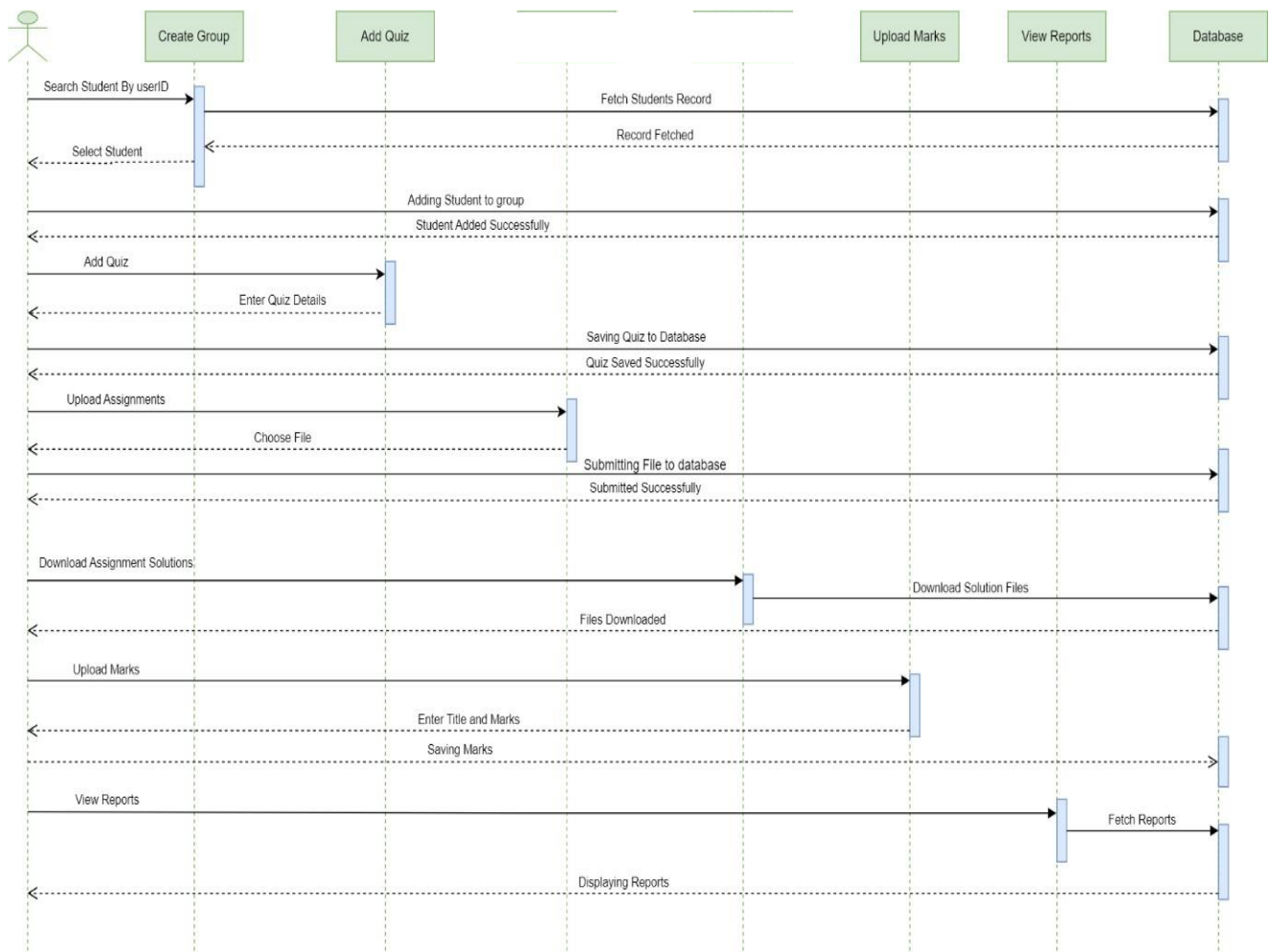


Fig 4.4 Instructor's SD

## 4.2 Data Representation [Diagram + Description]

This section provides a visual and descriptive overview of how data is structured, stored, and accessed within Smart Guard Pro. It includes a diagram illustrating key data entities, their relationships, and how data flows between components, ensuring efficient data handling and accessibility.

### 4.2.1 ER Diagram

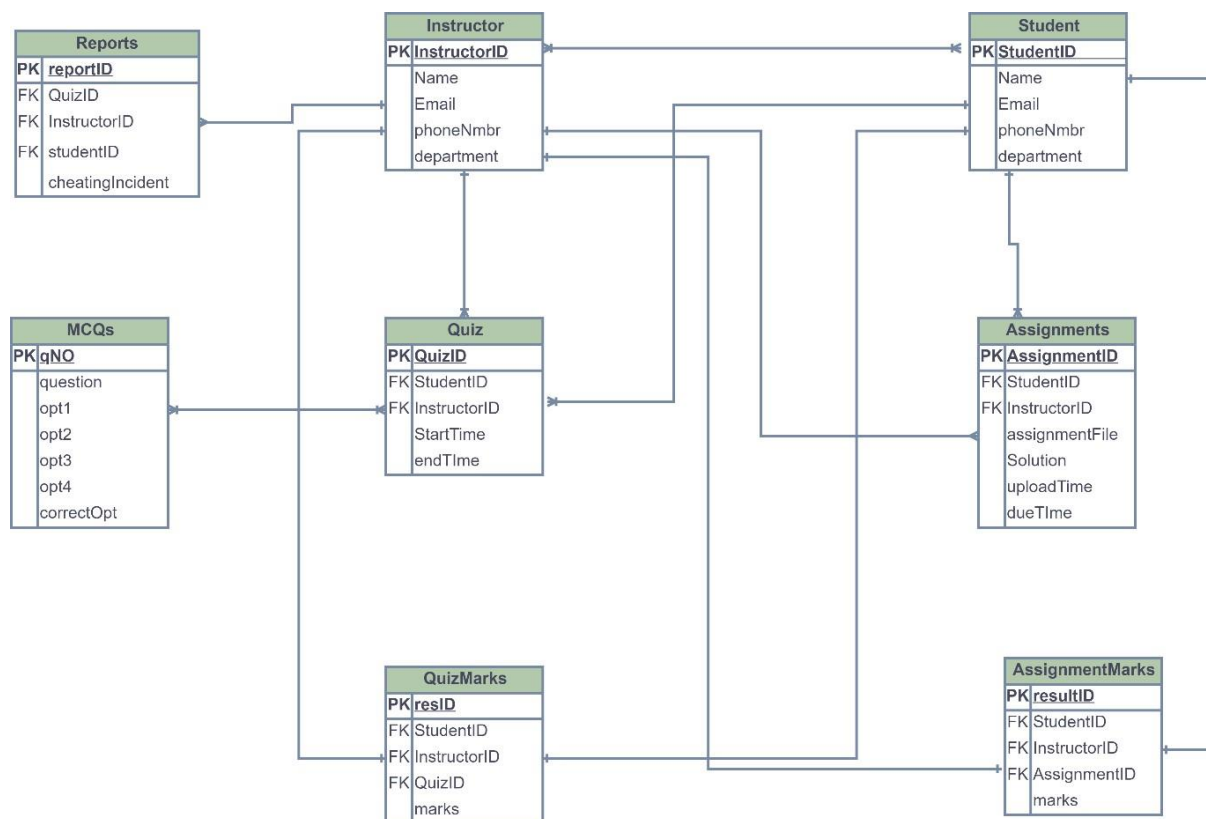


Fig 4.5 ER Diagram

## 4.2.2 Process Flow/Representation

This section illustrates the sequential flow of processes within Smart Guard Pro, detailing how tasks are executed and managed across different system modules. It provides a diagram and explanation to show how user actions, data processing, and system responses are coordinated to ensure smooth operation.

- **Activity Diagram**

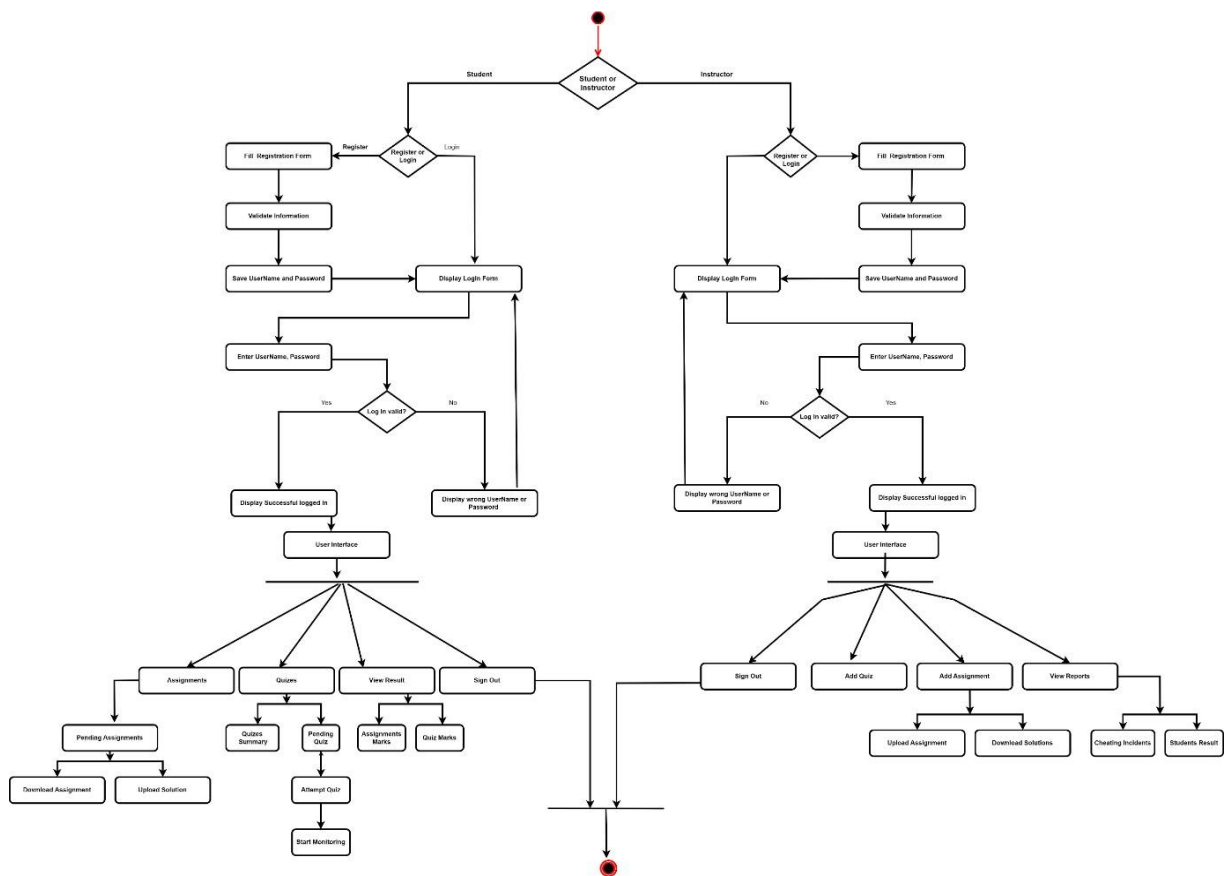


Fig 4.6 Activity Diagram

## 5 Implementation

This chapter provides a detailed discussion of the implementation of Smart Guard Pro, focusing on the algorithms and methodologies used in its major modules, including registration, login, facial recognition, activity monitoring, cheating detection, and quiz marking.

### 5.1 Algorithm

An algorithm is a structured set of steps or instructions used by Smart Guard Pro to perform specific tasks, ensuring that processes are executed efficiently and accurately

#### 5.1.1 Teacher Registration

```
procedure teacherRegistration(name, email, password):  
  if validEmailFormat(email) and strongPassword(password):  
    createTeacherAccount(name, email, password)  
    storeTeacherProfile(name, email)  
  return "Teacher registration successful." else:  
    return "Invalid email format or weak password. Registration  
    failed."
```

#### 5.1.2 Student registration

```
procedure studentRegistration(name, email, password,  
  profilePicture):  
  if validEmailFormat(email) and  
    strongPassword(password):  
    createStudentAccount(name, email, password, profilePicture)  
    storeStudentProfile(name, email, profilePicture)  
  return "Student registration successful." else:  
    return "Invalid email format or weak password. Registration  
    failed."
```

#### 5.1.3 Login Algorithm

Defines Algorithm for both teacher and student's login process.

- **Teacher Login**

```
procedure teacherLogin(email, password):  
  if  
    verifyCredentials(email, password):  
    return "Teacher login successful." else:  
    return "Invalid email or password. Login failed."
```

- **Student login**

```
procedure studentLogin(email, studentImage):  
  storedImage =  
    getStoredImageByEmail(email)
```

```
if compareFacialFeatures(studentImage, storedImage): return
"Student login successful."
else:
    return "Face verification failed. Login denied."
```

#### **5.1.4 Facial Recognition Algorithm**

```
procedure verifyIdentity(studentImage): storedImage =
getStoredImage(studentID)
if compareFacialFeatures(studentImage, storedImage): return
"Identity verified."
else:
    return "Identity verification failed."
```

#### **5.1.5 Activity Monitoring Algorithm**

```
procedure monitorActivity(studentID): while quizInProgress:
currentActivity = captureScreenActivity(studentID) if
containsCheatingBehavior(currentActivity):
    alertInstructor("Cheating behavior detected for Student ID: " +
studentID) logCheatingIncident(studentID, currentActivity)
else:
    continueMonitoring()
```

#### **5.1.6 Cheating Detection Algorithm**

```
procedure detectCheating(studentActivity):
if studentActivity showsAbnormalEyeMovements:
    alertInstructor("Suspicious eye movements detected.")
else if studentActivity indicatesCollaboration:
    alertInstructor("Possible collaboration detected.")
else:
    continueMonitoring()
```

#### **5.1.7 Quiz Marking Algorithm**

```
procedure markQuiz(studentID, quizAnswers): quizScore =
calculateQuizScore(quizAnswers) storeQuizScore(studentID,
quizScore)
return "Quiz marked and score recorded."
```

## 5.2 External APIs

Table 5.1 External APIs

Name of API	Description of API	Purpose of Usage	Function/Classes in which used
Firestore	Firestore APIs are a set of tools and services provided by Google's Firestore platform, designed to help developers build scalable and feature-rich applications more easily. Firestore offers a wide range of APIs that cover various aspects of app development, including authentication, database management, cloud messaging, analytics.	Flutter has excellent integration with Firestore services, providing APIs for authentication (firebase_auth), real-time databases (firebase_database), and much more. These APIs enable you to leverage the powerful features of Firestore in your Flutter application.	Used in Signup.dart, SignIn.dart, Home.dart, Login.dart, quiz.dart,
Gaze Cloud API	The GazeCloud API is a powerful tool that tracks where a user is looking on their screen. It uses the webcam to detect eye movements and can tell which part of the screen the user is focusing on. This technology is useful for a variety of applications, such as monitoring user attention, conducting usability studies, and creating interactive experiences based on where the user is looking.	The GazeCloud API precisely tracks and analyzes user gaze patterns via camera, enabling developers to monitor where users are looking on the screen in real-time. This functionality is essential for us to detect cheating.	Quiz.dart

## **5.3 User Interface**

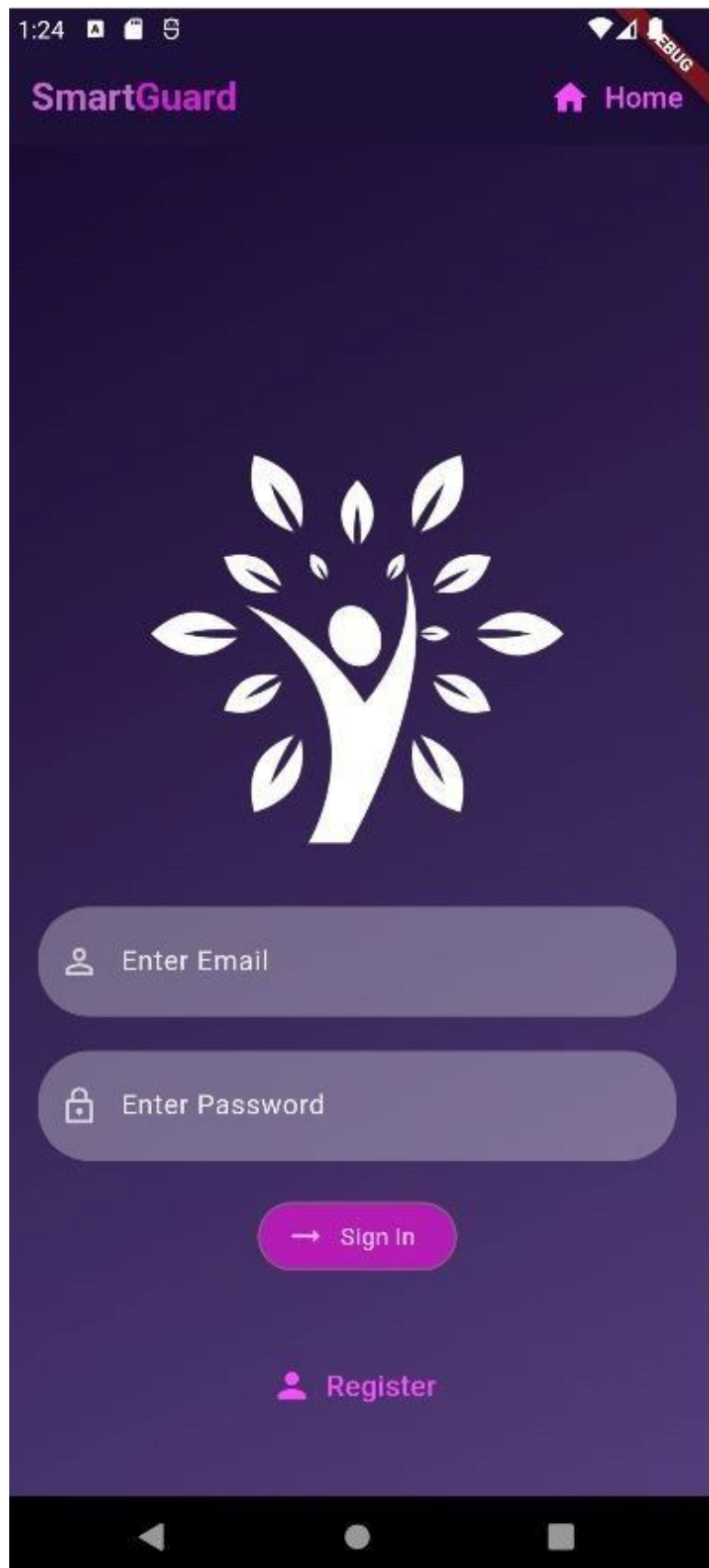
The user interface (UI) is the point of interaction between users and Smart Guard Pro, encompassing all visual elements, controls, and layouts that facilitate user engagement. It is designed to be intuitive and user-friendly, ensuring that both instructors and students can navigate the application efficiently and effectively.

### **5.3.1 Screen Images**

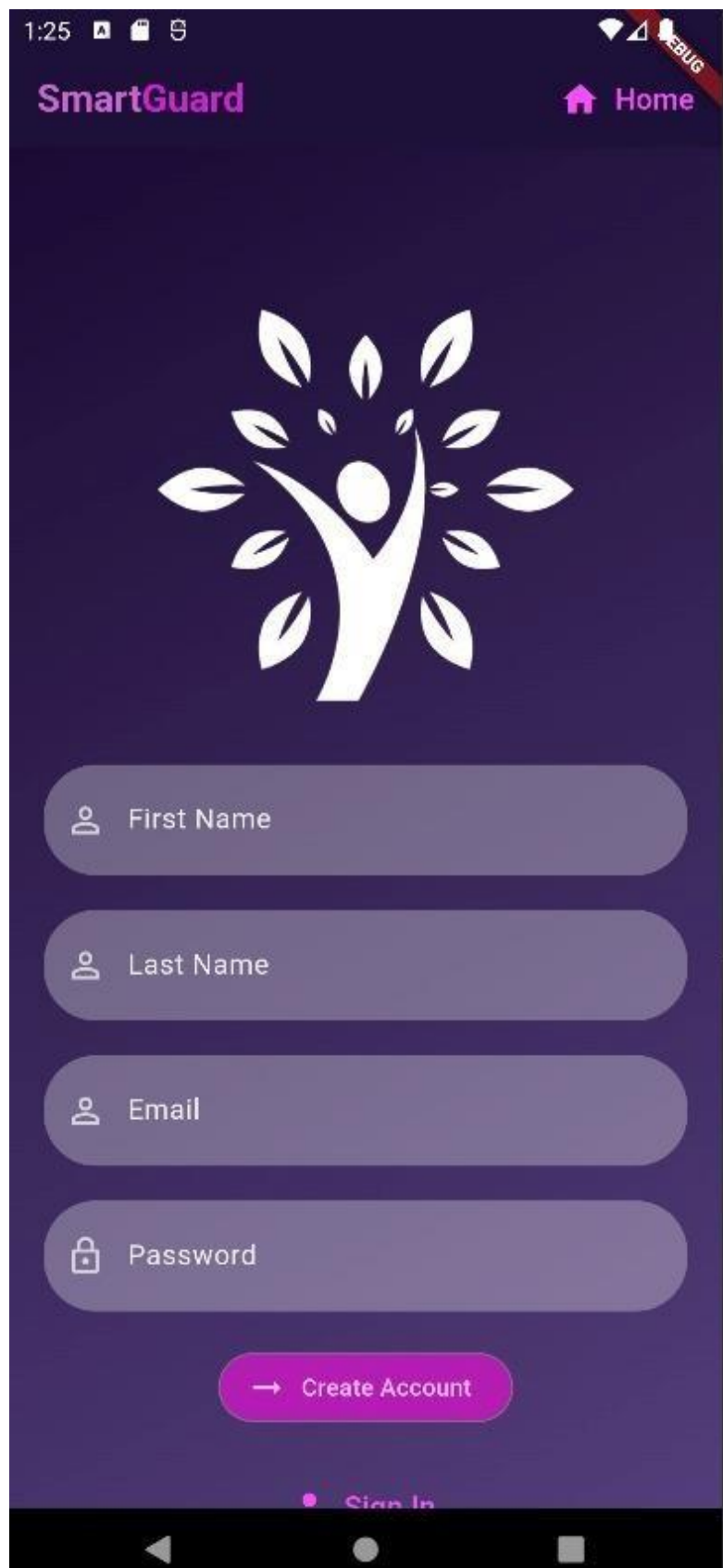


*Fig 5.1 Home Screen*





*Fig 5.2 log in*



The image shows a mobile application interface for SmartGuardPro. At the top, the status bar displays the time 1:25 and various icons. The app's header features the 'SmartGuard' logo in pink and a 'Home' button with a house icon. A red 'DEBUG' banner is visible in the top right corner. The main content area has a dark purple background with a white logo of a stylized tree with a person's arms raised as its trunk. Below the logo are four rounded rectangular input fields: 'First Name', 'Last Name', 'Email', and 'Password'. Each field has a small icon (person, person, envelope, and lock respectively) to its left. At the bottom of the form is a pink button with a right-pointing arrow and the text 'Create Account'. Below this button, the text 'Sign In' is partially visible. The bottom of the screen shows the standard Android navigation bar.

*Fig 5.3 Signup*



*Fig 5.4 Dashboard*

## 6 Testing and Evaluation

During software development, testing ensures that each component functions correctly, while evaluation assesses the software's overall performance and user satisfaction. Various types of testing, such as unit and integration testing, verify functionality, while usability testing and performance metrics gauge user experience and system efficiency.

### 6.1 Manual Testing

Manual testing involves human testers executing test cases without the use of automation tools. It ensures that software functions as expected from a user's perspective, identifying user interface issues, usability concerns, and functional bugs that might be missed by automated tests.

### 6.2 System testing

System testing is a comprehensive evaluation process that verifies the complete and integrated software application, Smart Guard Pro, against its specified requirements. This phase involves executing a series of test cases to assess the system's functionality, performance, security, and usability in real-world scenarios, ensuring that all components work together seamlessly and meet the needs of end-users before deployment. The goal is to identify and resolve any defects or issues to guarantee a reliable and effective application for managing online exams.

Table 6.1 System Testing

Test Case	Description	Expected Result	Actual Result	Status
User Registration	Verify successful user registration	User registration completes without errors, and user data is stored securely	Registration Process Successful. User redirected to the dashboard	Pass
Teacher Login	Verify successful login for teachers	Teacher logs in successfully and is redirected to the teacher dashboard	Login Successful. Teacher redirected to dashboard	Pass
Student Login	Verify successful login for students	Student logs in successfully and is redirected to the student dashboard	Login Successful. Student redirected to dashboard	Pass
Facial Recognition Login	Verify facial recognition during student login	Student's face is recognized, and login is successful	Face recognized. Login Successful. Student redirected to dashboard	Pass

Upload Quiz	Test the functionality of uploading a new quiz	Teacher should be able to upload a quiz, and students should see it	Quiz successfully uploaded and visible to students	Pass
Attempt Quiz	Verify students can attempt quizzes	Students can start and complete quizzes	Quiz attempt successful. Answers submitted	Pass
Cheating Notification	Verify instructor receives notifications of cheating	Instructor is notified immediately with details of the cheating incident	Notification received with incident details	Pass
View Quiz Results	Verify students can view their quiz results	Students can see their scores and feedback	Results displayed correctly	Pass
Update Profile	Ensure users can update their profile information	Users can edit and update their profile details	Profile Successfully Updated	Pass

### 6.3 Unit Testing

Unit testing focuses on validating individual components or modules of Smart Guard Pro in isolation to ensure that each part functions correctly as designed. By testing small units of code—such as functions, classes, or methods—developers can identify and fix bugs early in the development process, enhancing code quality and reliability before integration with other components.

#### 6.3.1 Signup

*Table 6.2 SignUp Testing*

No.	Test case/Test script	Attribute and value	Expected result	Result
1.	Signup as a new user.	Username: <u>idev.saadirshad99@gmail.com</u> Password: qwerty123	User account is successfully created.	Pass
2.	Signup with existing email.	Username: <u>idev.saadirshad99@gmail.com</u> Password: qwerty123	Display message "Email already in use. Please use a different email."	Pass

3.	Signup with invalid email format.	Username: idev.saadirshad99@gmail.com Password: qwerty123	Display message "Invalid email format. Please enter a valid email address."	Pass
----	-----------------------------------	---	---	------

### 6.3.2 Login

Table 6.3 LogIn Testing

No.	Test case/Test script	Attribute and value	Expected result	Result
1.	Login with valid credentials.	Username: <u>idev.saadirshad99@gmail.com</u> Password: qwerty123	User successfully logs in, and the dashboard is loaded.	Pass
2.	Login with incorrect password.	Username: <u>idev.saadirshad99@gmail.com</u> Password: wrongpassword	Display message "Incorrect password. Please try again."	Pass
3.	Login with non-existent email.	Username: <u>nonexistemail@gmail.com</u> Password: anypassword	Display message "Email not found. Please check your email or sign up."	Pass
4.	Login with blank email.	Username: "" Password: anypassword	Display message "Please enter your email."	Pass
5.	Login with blank password.	Username: <u>idev.saadirshad99@gmail.com</u> Password: ""	Display message "Please enter your password."	Pass

### 6.3.3 Dashboard

Table 6.4 Dashboard Testing

No.	Test case/Test script	Attribute and value	Expected result	Result
1.	Verify initial dashboard loading.	Onclick on Login Button, user logged in successfully. Dashboard is displayed.	Dashboard loads with relevant user information and functionalities.	Pass

2.	Add Quiz	From main dashboard user can add quiz	Quiz added	Pass
3.	Attempt Quiz	If there are scheduled quizzes that should be seen	Quiz attempted	Pass
4.	Logout Functionality.	Click on "Logout."	User is logged out, and the login page is displayed.	Pass

## 6.4 Functional Testing

Functional testing evaluates the system's functionality against specified requirements by testing various user interactions and scenarios. This testing ensures that all features of Smart Guard Pro work as intended, including quiz creation, attendance tracking, and cheating detection, verifying that the application delivers the expected outcomes for both instructors and students.

### 6.4.1 SignUp/ Registration

Table 6.5 Registration FT

No.	Test Case/Test Script	Attribute and Value	Expected Result	Result
1	Register with valid details	Username: Saad Irshad Password: qwerty1@ Email: <a href="mailto:idev.saadirshad99@gmail.com">idev.saadirshad99@gmail.com</a>	Account is created successfully	Pass
2	Register with an already existing email	Username: Saad Irshad Password: qwerty1@ Email: <a href="mailto:idev.saadirshad99@gmail.com">idev.saadirshad99@gmail.com</a>	Display message "Email already in use."	Pass
3	Register with invalid email format	Username: Saad Irshad Password: qwerty1@ Email: muneebch309gmail	Display message "Please enter a valid email address."	Pass
4	Register with mismatched passwords	Username: Saad Irshad Password: Muneeb001 Email: <a href="mailto:idev.saadirshad99@gmail.com">idev.saadirshad99@gmail.com</a>	Display message "Input valid passwords."	Pass

## 6.4.2 Login

Table 6.6 LogIn Testing

No.	Test Case	Attribute and Value	Expected Result	Result
1	Login with valid credentials	Email: <u>muneebch309@gmail.com</u> Password: Munee001@	Dashboard is loaded with user-specific information.	Pass
2	Login with incorrect password	Email: <u>muneebch309@gmail.com</u> Password: Munee001	Display message "Incorrect password."	Pass
3	Login with non-existent email	Email: <u>Shakir@gmail.com</u> Password: Munee001@	Display message "Email not found. Please check your email"	Pass

## 6.4.3 Reset Password

Table 6.7 Reset Password Testing

No.	Test Case/Test Script	Attribute and Value	Expected Result	Result
1	Reset password with valid email	Email: <u>idev.saadirshad99@gmail.com</u>	Password reset link is sent to the user's email.	Pass
2	Reset password with non-existent email	Email: <u>nonexistemail@gmail.com</u>	Display message "Email not found."	Pass
3	Reset password with invalid email format	Email: idev.saadirshad99gmail.com	Display message "Please enter a valid email address."	Pass



#### 6.4.4 Quiz Generation

Table 6.8 Quiz Generation Testing

No.	Test Case	Attribute and Value	Expected Result	Result
1	Generate a quiz with valid parameters	Title: "Mathematics", Number of Questions: 10,	A quiz is generated with 10 mcqs questions on Mathematics.	Pass
2	Generate a quiz with no subject selected	Title: (empty), Number of Questions: 10.	Display message "Please select a subject."	Pass
3	Generate a quiz with invalid number of questions	Title: "Mathematics", Number of 10 Questions: "inv"	Display message "Please enter a question."	Pass
4	Generate a quiz with number of questions exceeding limit	Title: "Mathematics", Number of Questions: 1000,	Display message "The number of questions exceeds the limit. Please enter a lower number."	Pass
5	Take the generated quiz	Title: "Mathematics", Number of Questions: 10,	The quiz is taken, answers are submitted, and results are displayed.	Pass
6	View quiz results after completion	Title: "Mathematics", Number of Questions: 10.	The results of the quiz are displayed, showing correct and incorrect answers.	Pass

#### 6.4.5 Student facial verification

Table 6.9 Facial Verification Testing

Function	Test Case/Test Script	Attribute and Value	Expected Result	Result
Student Facial Login Flow	Verify the flow of student facial login	Student's face is verified and login process completes	Facial login process completes successfully	Pass

#### 6.4.6 Cheating detection and notifications:

Table 6.10 Notifications Testing

Function	Test Case	Attribute and Value	Expected Result	Result
Real-time Cheating Detection	Test real-time detection of cheating behaviors	System detects cheating behaviors during quizzes	Cheating behaviors detected in real-time	Pass

## 6.5 Integration Testing

Integration testing assesses the interaction between different modules or components of Smart Guard Pro to ensure they work together seamlessly. This phase identifies any issues arising from the integration of individual units, such as data flow, communication, and compatibility, ensuring that the system functions as a cohesive whole before proceeding to system testing.

*Table 6.11 Integration Testing*

Test Case	Description	Expected Result	Actual Result	Status
Teacher Registers and Logs In	Verify the integration of registration and login functionalities for teachers	Teacher registers and logs in successfully	Registration and login successful	Pass
Student Registers and Logs In	Verify the integration of registration and login functionalities for students	Student registers and logs in successfully	Registration and login successful	Pass
Upload and Attempt Quiz	Verify the integration of quiz upload and attempt functionalities	Teacher uploads quiz and students attempt it	Quiz uploaded and attempted successfully	Pass
Monitor and Detect Cheating	Verify the integration of monitoring and cheating detection functionalities	System monitors quiz and detects cheating behaviors	Cheating detected and flagged	Pass
Detect Cheating and Notify Instructor	Verify the integration of cheating detection.	Instructor receives notification of detected cheating behavior	Notification received with correct details	Pass

## **7 Conclusion and Future Work**

In conclusion, Smart Guard Pro represents a significant advancement in ensuring academic integrity and fairness in online examinations. By leveraging cutting-edge technologies such as artificial intelligence and machine learning, the application effectively addresses the challenges of cheating and automated attendance management in educational settings. Through a comprehensive suite of features including facial recognition, real-time monitoring, and streamlined grading, Smart Guard Pro not only enhances the security of the examination process but also instills confidence among educators and students alike.

Looking ahead, there are several opportunities for future work to further enhance the capabilities of Smart Guard Pro. Future developments could focus on expanding the range of monitoring features, such as incorporating advanced behavioral analytics to detect more subtle cheating behaviors. Additionally, improving user experience through continuous feedback loops and interface enhancements will be essential for maximizing usability. Exploring integrations with learning management systems and expanding support for various devices will also contribute to a more robust and versatile solution. Overall, the ongoing evolution of Smart Guard Pro aims to adapt to the ever-changing landscape of online education, ensuring that it remains a reliable and effective tool for maintaining academic integrity.

### **7.1 Conclusion**

Smart Guard Pro represents a significant advancement in the field of online exam security and student monitoring. By leveraging cutting-edge technologies such as artificial intelligence and facial recognition, the application has successfully addressed the following key objectives:

Implemented robust measures to prevent and detect cheating behaviors during exams, ensuring fairness and integrity in assessments. These measures include advanced algorithms that monitor multiple exam parameters, enabling comprehensive oversight of student actions. As a result, both educators and students benefit from a secure and fair assessment environment that upholds academic standards.

Developed a reliable system for authenticating teachers and students using facial recognition, enhancing user security and accountability. The system leverages cutting-edge facial recognition technology to verify identities quickly and accurately. This feature not only ensures that only verified users participate in exams but also bolsters the credibility of the examination process.

Enabled real-time monitoring of student activities during exams, promptly detecting suspicious behaviors and alerting instructors. The system's monitoring capabilities encompass various exam protocols, including eye movement and posture analysis. This proactive approach to monitoring provides instructors with valuable insights into student engagement and adherence to exam rules.

Implemented a proactive notification system to alert instructors immediately upon detecting cheating behaviors, allowing timely intervention. Notifications are customizable, allowing instructors to set parameters for various types of alerts based on their preferences. This tailored approach ensures that instructors can address issues in real time, maintaining the flow and integrity of the exam.

Through rigorous testing and evaluation, Smart Guard Pro has demonstrated robust performance and reliability across various scenarios. This evaluation included simulations of real exam conditions to validate the system's efficacy in diverse settings. The feedback received from initial deployment and testing phases has been positive, highlighting the system's effectiveness in maintaining exam integrity and enhancing trust in online assessment processes.

## **7.2 Future Work**

As Smart Guard Pro continues to evolve, there are several promising avenues for future development and enhancement:

Explore integrating more advanced machine learning models and artificial intelligence techniques to further enhance the accuracy and capabilities of cheating detection. This includes:

Implement AI algorithms to analyze patterns in student behavior during exams to detect subtle signs of cheating. These algorithms are designed to recognize irregularities that might otherwise go unnoticed by human observers. By identifying patterns in student actions, the system adds a sophisticated layer of monitoring that enhances the overall security of the exam process.

Utilize real-time data processing to improve the speed and accuracy of cheating detection, ensuring prompt intervention. The system is capable of analyzing large data streams from multiple students simultaneously, allowing for immediate identification of suspicious behaviors. This real-time capability helps prevent cheating incidents before they impact exam integrity.

Focus on improving the user interface and overall user experience to make Smart Guard Pro more intuitive and user-friendly for both instructors and students. Key enhancements include responsive design elements to ensure seamless navigation across devices. A refined user experience can significantly reduce the learning curve and improve user satisfaction.

Allow users to customize their dashboards based on their preferences and frequently accessed features. Customization options include personalized widgets, color schemes, and layout adjustments. This flexibility enables users to optimize their workflow, making it easier to access the most relevant information quickly.

Develop a mobile-friendly version of Smart Guard Pro to facilitate easy access and

usage on smartphones and tablets. The mobile version is designed to retain all essential functionality, with an adaptive layout for smaller screens. This mobile accessibility supports remote monitoring and ensures convenience for both students and instructors.

Explore seamless integration with popular Learning Management Systems (LMS) to streamline data sharing and enhance interoperability. This involves establishing standard protocols for secure data exchange, allowing Smart Guard Pro to communicate effortlessly with various LMS platforms. The goal is to enhance user efficiency and reduce data redundancy.

Ensure smooth synchronization of student and course data between Smart Guard Pro and LMS to maintain consistency and accuracy. Automatic synchronization allows for real-time updates, ensuring that both systems reflect the latest student information. This alignment helps prevent discrepancies in records, making data management more reliable.

Enhance the security framework of Smart Guard Pro to fortify data protection and prevent unauthorized access. This includes a comprehensive review of security protocols and adherence to industry standards. By proactively addressing security, the system can better protect sensitive information from potential breaches.

Implement 2FA to add an extra layer of security during user login and access. The two-factor authentication process includes verification via SMS or email, adding a robust layer to the login procedure. This extra security measure helps ensure that only authorized users gain access to the platform.

Upgrade encryption protocols to ensure end-to-end data security and privacy for all user interactions. This involves adopting advanced encryption standards like AES-256, which is trusted for protecting sensitive data. Strengthened encryption safeguards all transmitted information, reinforcing the platform's commitment to privacy.

## **8 References**

1. Al-Shahmeh, M. M. (2023). Flutter Apps Development: Build Cross-Platform Flutter Apps with Trust. Damascus, Syria: Independently Published.
2. Boté, J.-J. (Mar 2019). Dataset Management as a Special Collection. *Collection Management*, 259- 276.
3. Galen, D. (2018). *Software Quality, Concepts and Practice*. Hoboken, New Jersey: Wiley, IEEE Press.
4. Karl Wieggers, J. B. (2013). *Software Requirements*, Third edition. Germany: Microsoft Press.
5. Marco Ortu, G. D. (May 2015). Measuring and Understanding the Effectiveness of JIRA Developers Communities. *Research Gate*, 8-9.
6. Pavel Hamet, J. T. (April 2017). Artificial intelligence in behaviors , S36-S40.
7. Payne, R. (December 2019). *Beginning App Development with Flutter: Create Cross-Platform Mobile Apps*. New York: A press.
8. Phillip A. Laplante, M. K. (2022). *Requirements Engineering for Software and Systems*. New York: Auerbach Publications.
9. Raschka, S. (September 2015). *Python Machine Learning*. Birmingham, UK: Packt Publising Ltd.

# SmartGuardPro - Guarding Academic Integrity,

## ORIGINALITY REPORT

10%  
SIMILARITY INDEX

4%  
INTERNET SOURCES

0%  
PUBLICATIONS

9%  
STUDENT PAPERS

## PRIMARY SOURCES

1 Submitted to Higher Education Commission Pakistan 3%  
Student Paper

2 Submitted to Central Queensland University 2%  
Student Paper

3 fastercapital.com 1%  
Internet Source

4 Submitted to University of Technology, Mauritius <1%  
Student Paper

5 Submitted to Multimedia University <1%  
Student Paper

6 Submitted to Manipal University <1%  
Student Paper

7 Submitted to UOW Malaysia KDU University College Sdn. Bhd <1%  
Student Paper

8 Submitted to Kingston University <1%  
Student Paper

9	Submitted to University of Maryland, University College Student Paper	<1 %
10	Submitted to Swiss School of Business and Management - SSBM Student Paper	<1 %
11	Submitted to Taibah University Student Paper	<1 %
12	Submitted to Flinders University Student Paper	<1 %
13	Submitted to NCC Education Student Paper	<1 %
14	Submitted to University of Malaya Student Paper	<1 %
15	people.ucalgary.ca Internet Source	<1 %
16	Submitted to Informatics Education Limited Student Paper	<1 %
17	Submitted to Islamic University Student Paper	<1 %
18	eprints.utar.edu.my Internet Source	<1 %
19	vle.learning.moe.edu.sg Internet Source	<1 %



20	<a href="http://www.iihglobal.com">www.iihglobal.com</a> Internet Source	<1 %
21	<a href="http://www.sanmin.com.tw">www.sanmin.com.tw</a> Internet Source	<1 %
22	Submitted to University of Greenwich Student Paper	<1 %
23	de Sá, Carlos Manuel Ramires. "Collaborative Conceptual Modelling Through a Semantic Wiki", Universidade do Porto (Portugal), 2024 Publication	<1 %
24	<a href="http://www.scribd.com">www.scribd.com</a> Internet Source	<1 %
25	<a href="http://www.slideshare.net">www.slideshare.net</a> Internet Source	<1 %

Exclude quotes Off

Exclude matches Off

Exclude bibliography Off