Software Engineering Project

# Exam Brother

Group 4

| | |
|---|---|
| Anthonius Hendhy W. | 2306161795 |
| Bryan Herdianto | 2306210885 |
| Maxwell Zefanya G. | 2306221200 |

**Program Studi Teknik Komputer**
**Fakultas Teknik**
**Universitas Indonesia**

# Table of Contents

# Chapter 1 - Introduction

## 1.1 Background

This document outlines the Exam Brother project, a Moodle plugin created with the primary purpose of enhancing online exam integrity through AI-assisted proctoring. Named Exam Brother, this tool takes a modern approach by combining lightweight AI monitoring with Moodle's existing exam framework, providing institutions with a more accessible and affordable proctoring solution.

Exam Brother serves as a valuable tool to help educators ensure fairness in online assessments. It achieves this by monitoring student behavior during exams and flagging potential anomalies such as tab switching, unusual head or eye movements, or frozen camera feeds. Unlike traditional proctoring systems, Exam Brother does not automatically disqualify students. Instead, it generates real-time alerts and detailed post-exam reports for proctors, ensuring that final decisions remain human-driven.

Current solutions, such as Safe Exam Browser, often lack AI-driven proctoring features or fail to provide structured reporting and real-time notifications to proctors. Additionally, many existing AI proctoring tools are prohibitively expensive, making them inaccessible to smaller institutions. Exam Brother addresses these challenges by offering a Moodle-native, cost-effective alternative that integrates seamlessly into existing workflows.

The plugin we are developing will run directly in the student's browser with AI analysis. This ensures cross-platform compatibility while keeping sensitive data private, since only event metadata is logged and stored within the institution's Moodle system. By combining affordability, accessibility, and AI-driven insights, Exam Brother aims to make online exams more secure and trustworthy for both students and educators.

## 1.2 Purpose

Students in Indonesia often face challenges in maintaining integrity during online and high-stakes examinations, which can lead to unfair outcomes and undermine trust in the education system. One of the most significant hurdles institutions encounter is the lack of affordable and effective proctoring tools that can reliably monitor online exams without creating unnecessary stress or privacy concerns. The aim of the Exam Brother project is to address this issue by providing a lightweight, Moodle-integrated AI proctoring solution that enhances fairness in examinations while remaining accessible to a wide range of institutions.

In Indonesia, exam fraud has been shown to be a persistent and systemic issue. Before the rollout of computer-based testing (CBT), the Ministry of Education flagged around 33% of junior secondary schools for suspicious answer patterns using an integrity index [1]. When CBT was introduced, average scores declined by 0.4–0.5 standard deviations, strongly suggesting that prior inflated results were linked to widespread cheating [1]. More recently, cases of dishonesty continue to occur even in high-stakes university entrance exams, with reports of hundreds of cheating attempts during the 2025 UTBK, including impersonation and the use of “academic joki” services [2]. These findings reveal the urgent need for more effective and scalable solutions that can complement CBT and protect exam integrity.

Cheating is a problem that can be reduced with consistent monitoring and the use of modern, AI-assisted proctoring tools. Our project seeks to provide this support by flagging suspicious behaviors such as tab switching, camera freezing, or unusual head movements (without automatically penalizing students). By doing so, we aim to foster a fairer and more transparent testing environment, empowering proctors to make informed decisions while ensuring students can participate in exams with confidence.

## 1.3 Literature Review

### 1.3.1 Limitations of Safe Exam Browser (SEB)

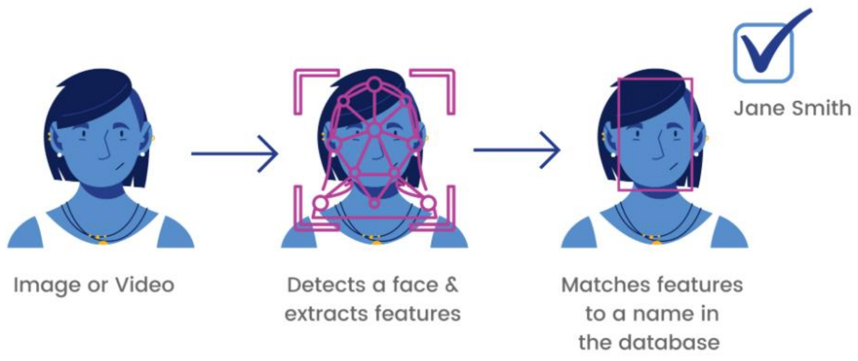
Safe Exam Browser (SEB) is one of the most widely used tools integrated into Moodle for online examinations. It provides a secure environment by restricting functionalities such as opening new browser tabs, switching windows, or accessing unauthorized resources during the test. This ensures that students remain within the exam interface throughout the session.

While SEB plays an important role in minimizing technical cheating, its functionality is primarily rule-based rather than intelligent. For example, it cannot assess whether the student is being distracted, monitored by another person off-camera, or using a second device. Additionally, SEB does not natively include AI-based proctoring features such as gaze detection, face orientation tracking, or anomaly reporting. Institutions relying solely on SEB are limited to a controlled environment but lack deeper behavioral insights into test-takers [3].

Another significant limitation is cost. Though SEB itself is open-source, institutions often need to pair it with expensive third-party proctoring tools to achieve real-time monitoring and reporting. This creates accessibility issues for smaller universities or schools with limited budgets [4].

### 1.3.2 AI-Assisted Proctoring

AI-assisted proctoring systems have emerged to address gaps left by traditional lockdown browsers. These systems use techniques such as facial recognition, gaze tracking, voice detection, and behavioral analysis to flag suspicious activities during exams. For example, gaze estimation can determine whether a student is consistently looking away from the screen, while head pose detection can indicate if they are being assisted by someone off-camera.


Jane Smith
Image or Video
Detects a face & extracts features
Matches features to a name in the database

Recent studies in Indonesia have also tested AI-based proctoring solutions. A web-based tool using deep learning achieved an F1-score of 84.52% in detecting cheating behaviors via webcam during online exams [5]. This indicates that AI can provide reliable additional layers of monitoring. However, challenges remain in reducing false positives, ensuring fairness, and respecting student privacy. Thus, AI should be considered as a supportive tool for human proctors rather than an autonomous judge.

### 1.3.3 Event Logging and Reporting Systems

A critical limitation of many current online proctoring solutions is the lack of structured reporting and notification systems. While certain tools may detect suspicious behavior, they often do not provide proctors with detailed, time-stamped logs or real-time alerts. This makes post-exam evaluation cumbersome, as proctors must manually review long video recordings to verify misconduct.

Event-based logging offers a more scalable approach. Instead of recording and storing entire video feeds, the system can log only flagged incidents, such as:

- "Tab switch detected at 12:35:04"
- "Head orientation: left for 6s at 12:42:19"
- "Camera feed frozen at 12:50:12"

By providing metadata-based reporting, proctors can quickly review key anomalies and make informed decisions. This method also reduces data storage requirements and enhances privacy compared to storing full exam recordings [6].

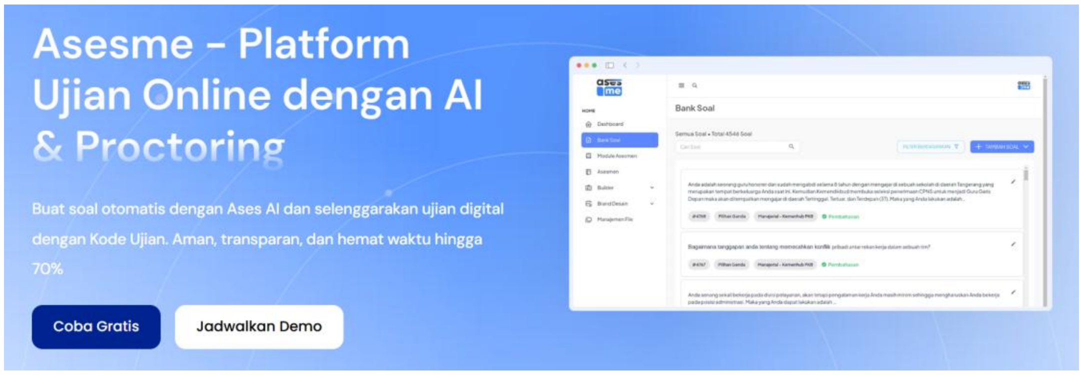
### 1.3.4 Affordability and Accessibility

Cost remains a key barrier for adopting AI proctoring. Commercial platforms often charge IDR 1,000,000 per year or more, with costs increasing based on the number of students participating in the exam. [4]. For smaller schools and universities, especially in developing countries, this creates significant inequities in exam integrity. [7]

These financial barriers highlight the urgent need for affordable, locally adaptable solutions that can be integrated into platforms like Moodle without heavy licensing costs. Exam Brother aims to fill this gap by offering AI-assisted monitoring at lower cost while still maintaining essential features such as anomaly detection, reporting, and real-time alerts.

## 1.4 Similar Competitors

This sort of solution has also been attempted by other platforms in Indonesia. One notable example is Asesme, an online examination platform that provides AI-assisted proctoring, screen monitoring, and live scoring features. It is designed as a full exam system that integrates question banks, randomization, automatic report generation, and even AI-based question creation. Asesme has gained traction among schools and training centers for its complete exam management capabilities.



Asesme - Platform Ujian Online dengan AI & Proctoring

Buat soal otomatis dengan Ases AI dan selenggarakan ujian digital dengan Kode Ujian. Aman, transparan, dan hemat waktu hingga 70%

However, Asesme also comes with its limitations. The platform is primarily a standalone solution rather than an LMS plugin, meaning institutions using Moodle or similar platforms need to adapt their workflow to match Asesme’s system. While it offers camera monitoring and screen tracking, its documentation does not clearly highlight advanced behavioural AI detection such as gaze direction, head movement, or camera freeze detection. Moreover, costs can scale significantly for institutions with large student populations, as pricing often depends on usage tiers. This is where Exam Brother aims to outshine Asesme in several aspects:

- Moodle Plugin Integration
  Exam Brother is built as a Moodle plugin, which means schools and universities already using Moodle don’t need to switch platforms. This ensures seamless adoption with minimal setup.
- Advanced Behavioural AI Monitoring
  Unlike simple screen tracking, our system leverages MediaPipe to detect gaze direction, head orientation, and frozen camera events, giving proctors more accurate and detailed insights.
- Real-time Alerts and Post-Exam Reports
  Exam Brother not only monitors students' lives but also flags suspicious activities with timestamps, delivering structured reports automatically once the exam ends.
- Cost-Effectiveness
  By working directly inside Moodle and storing only metadata (not full video), Exam Brother reduces infrastructure overhead and can be offered at a much lower price point compared to full standalone platforms.

## 1.5 Tools

Exam Brother will be developed as a Moodle plugin, which requires integration with Moodle’s core system through PHP and its official API. Besides that, it also needs real-time monitoring features that will be handled in-browser through JavaScript with MediaPipe, and backend AI processing powered by Python to detect anomalies and generate reports. We include these tools we'll be using in developing the application:

### 1.5.1 PHP & Moodle API

PHP is the primary programming language behind Moodle, and it plays a central role in developing plugins that extend Moodle's functionality. By leveraging PHP together with the Moodle API, Exam Brother can directly integrate with Moodle's existing ecosystem, such as quiz modules, authentication systems, and gradebooks. This ensures that institutions using Moodle do not need to adopt an entirely new system or workflow but can instead enhance their current exam processes with AI proctoring. Furthermore, Moodle's modular architecture allows PHP-based plugins to interact with the LMS seamlessly, providing extensibility while maintaining compatibility with Moodle's frequent updates. [8]

### 1.5.2 JavaScript (with MediaPipe)

JavaScript provides the foundation for in-browser interactivity, and with the integration of MediaPipe, it becomes a powerful tool for real-time visual processing. In Exam Brother, JavaScript combined with MediaPipe is used to analyze the student's webcam feed directly in the browser without requiring additional installations. Features such as eye gaze tracking, head movement detection, and frozen camera identification can be implemented efficiently, ensuring that exams are monitored continuously with minimal latency. Because this processing happens locally in the browser, data transfer is minimized, which not only enhances privacy but also reduces server load and network dependency. [9]

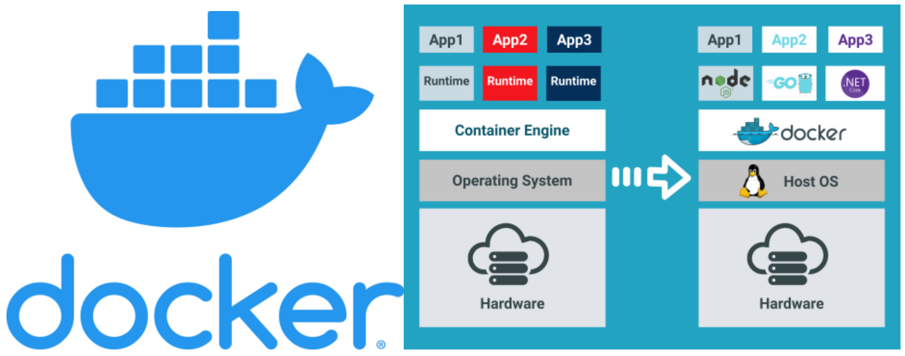
### 1.5.3 Docker

Docker is a container-based deployment utility, commonly used in systems that need scalability and continuous integration, and continuous deployment (CI/CD) environments [10]. A container will contain the application and the necessary dependencies (library, binary, configuration file, etc), completely isolated from the host machine, reducing the need to configure the hardware or virtual machine used to deploy the application software. The core components of Docker include:

a. Docker Container
A lightweight, standalone, and executable software package that includes everything needed to run an application, including the code, runtime, libraries, and dependencies.

b. Docker Image
A read-only template used to create Docker containers. It encapsulates the application and its environment in a consistent and reproducible way.

c. Dockerfile
A text file that defines the instructions and configuration needed to build a Docker image. It specifies the base image, necessary packages, environment variables, and commands to set up the application.

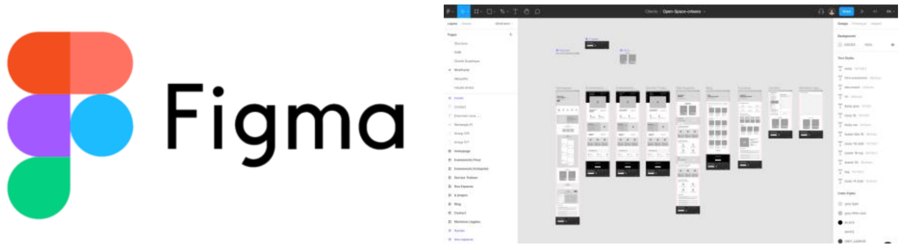


https://www.docker.com/

### 1.5.4 Figma

Figma is a popular cloud-based design tool used for creating user interfaces, user experience designs, prototypes, and collaborative design projects. It allows designers and teams to work on design projects in real time, fostering seamless collaboration and efficient workflows. [11] Some of Figma’s features:

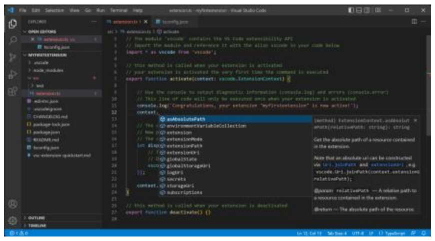
a. Cloud-Based Design
   Figma is a web-based design tool that operates directly within a web browser, allowing designers to access and work on their projects from anywhere with an internet connection. It eliminates the need for installations and updates, ensuring that users always have access to the latest version of the tool.
b. Collaborative Design
   Figma facilitates real-time collaboration, enabling multiple designers to work on a project simultaneously. Team members can view, edit, and comment on designs in real time collaboratively.
c. Design Components and Styles
   Figma offers a robust system for creating design components and styles, promoting design consistency and efficiency. Designers can create reusable components and define styles for text, color, and effects, streamlining the design process and ensuring a cohesive design system.



https://www.figma.com/

### 1.5.5 Visual Studio Code IDE

Visual Studio Code is a powerful Integrated Development Environment (IDE) used by many developers worldwide. Its ability to add custom extensions will help the developer speed up development time. These extensions include code completion, debugging, containers management, integrated terminals, etc. [12]

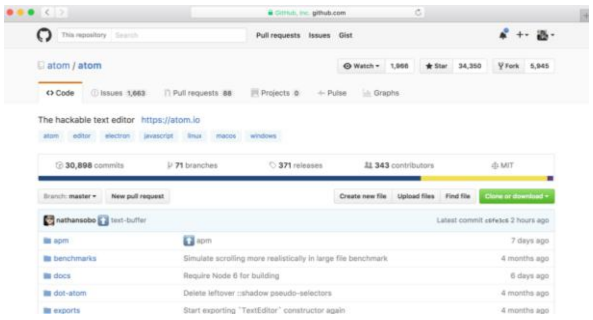


https://code.visualstudio.com/

## 1.5.6 Git & Github

Git is a widely used distributed version control system that enables tracking changes in source code during software development. It allows multiple developers to collaborate on a project simultaneously. Git records changes in a repository, allowing developers to create branches, merge code, and maintain a history of commits.

GitHub is a popular web-based platform that provides hosting services for Git repositories. It enhances collaboration by offering features like pull requests, issue tracking, and project management tools. GitHub acts as a central hub for developers to share, contribute, and manage code efficiently, making it an integral platform in the software development community. [14]



The hackable text editor https://atom.io

atom editor electron javascript linux macos windows

30,898 commits 71 branches 371 releases 343 contributors MIT

nathansobo text-buffer Latest commit c6fe3c6 2 hours ago

| | | |
|---|---|---|
| apm | apm | 7 days ago |
| benchmarks | Simulate scrolling more realistically in large file benchmark | 4 months ago |
| docs | Require Node 6 for building | 6 days ago |
| dot-atom | Delete leftover ::shadow pseudo-selectors | 4 months ago |
| exports | Start exporting `TextEditor` constructor again | 4 months ago |

https://github.com

### 1.5.7 PostgreSQL

PostgreSQL is a powerful open-source relational database management system (RDBMS) known for its robustness, extensibility, and advanced features. It supports a wide range of data types and provides features like transactions, sub-selects, triggers, and views. PostgreSQL follows the SQL standard and extends it with features like JSONB data type for handling JSON data efficiently. It has a strong community and active development, ensuring regular updates and improvements. Additionally, PostgreSQL's scalability and support for complex queries make it a preferred choice for applications ranging from small projects to large, enterprise-level databases. [15]

PostgreSQL

https://www.postgresql.org

### 1.5.8 Trello

Trello is a visual work management tool that helps empower teams to ideate, plan, and manage their work together in a collaborative, productive, and organized way. Trello uses board, list, and card to simplify and standardize the team's word process in an intuitive way. [16]

Trello

https://trello.com

## 1.6 Risk Analysis

Potential risks and concerns associated with the implementation and use of the Exam Brother system:

a. False positive results

A false positive occurs when the system mistakenly identifies a student as cheating. This can lead to unfair treatment and put students at a disadvantage. Such errors may also decrease students' trust in the system and could result in unnecessary stress or disciplinary actions. Continuous monitoring and careful calibration of the system are needed to minimize these occurrences.

b. High latency

The Exam Brother system is intended to operate in real time. High latency could disrupt a student's focus and reduce the system's overall effectiveness. Delays in processing could also interfere with timely feedback or alerts, potentially compromising the fairness and reliability of the exam environment. Optimizing system performance and ensuring stable network conditions are essential to prevent these issues.

c. Privacy concerns

The system relies on video, audio, and behavioral data for monitoring, which may raise privacy and data protection concerns among students and institutions. Mishandling or unauthorized access to this data could violate privacy laws or ethical standards. Implementing strict data retention policies, anonymization techniques, and clear consent mechanisms is necessary to maintain compliance and trust.

d. Unequal access to required hardware

The system's reliance on webcam monitoring assumes all students have access to a functioning camera and adequate lighting. These conditions are not guaranteed for students in low-resource environments, rural areas, or those using shared or older devices. This creates an accessibility gap that may unfairly exclude or disadvantage certain learners, undermining the system's fairness and inclusivity.

# Chapter 2 - Project Management

## 2.1 Roles and Responsibilities

Consciously and explicitly defining each person's role, responsibilities, and success criteria within the team can have an instant positive impact. It ensures that: Everyone knows what they're doing. It sounds simple, but when roles are clear, people know what's expected of them, how to behave, and what they need to accomplish. To ensure the project’s success, a team needs people fulfilling certain roles so one role doesn’t get overloaded with the workload and not produce a positive result.

### 2.1.1 Anthonius Hendhy Wirawan - Back-End Programmer and AI Engineer

Responsibilities:

- Design and implement the back-end ecosystem of the product
- Create pathways for the front-end to interact with the database through APIs
- Implement MediaPipe for AI proctoring features of the product

### 2.1.2 Bryan Herdianto - Front-End Programmer and Application Tester

Responsibilities:

- Design and plan the UI with user experience in mind
- Implement new UI for Moodle-based pages
- Integrate the front-end with the back-end of the product to ensure smooth experience
- Evaluate the product for correctness and smoothness of the program

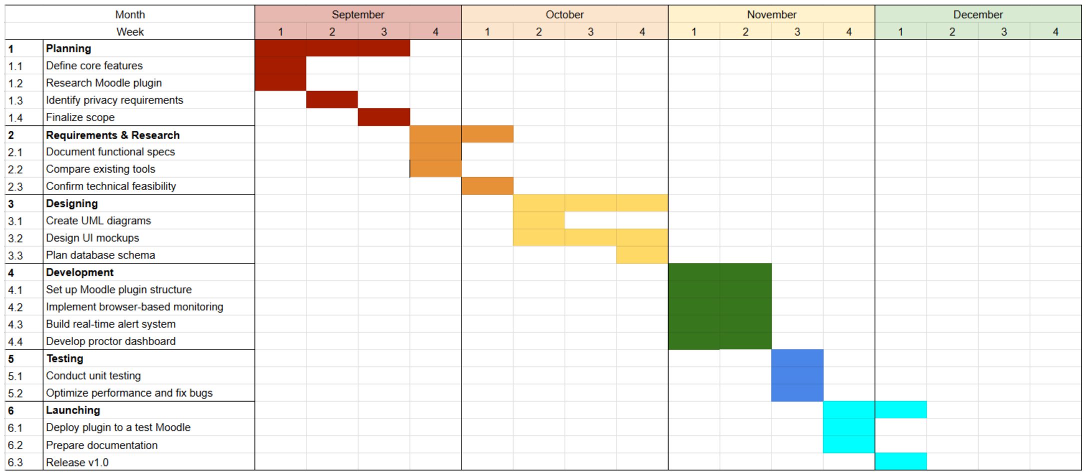
### 2.1.3 Maxwell Zefanya Ginting - Back-End Programmer and AI Engineer

Responsibilities:

- Design and implement the back-end ecosystem of the product
- Create pathways for the front-end to interact with the database through APIs
- Tweak the AI model to ensure correctness and reliability of the algorithm

## 2.2 Schedule and Planning

### 2.2.1 Schedule Overview



| | Month | September | | | | October | | | | November | | | | December | | | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | Week | 1 | 2 | 3 | 4 | 1 | 2 | 3 | 4 | 1 | 2 | 3 | 4 | 1 | 2 | 3 | 4 |
| 1 | **Planning** | 🟥 | 🟥 | 🟥 | | | | | | | | | | | | | |
| 1.1 | Define core features | 🟥 | | | | | | | | | | | | | | | |
| 1.2 | Research Moodle plugin | 🟥 | | | | | | | | | | | | | | | |
| 1.3 | Identify privacy requirements | | 🟥 | | | | | | | | | | | | | | |
| 1.4 | Finalize scope | | | 🟥 | | | | | | | | | | | | | |
| 2 | **Requirements & Research** | | | | 🟧 | 🟧 | | | | | | | | | | | |
| 2.1 | Document functional specs | | | | 🟧 | | | | | | | | | | | | |
| 2.2 | Compare existing tools | | | | 🟧 | | | | | | | | | | | | |
| 2.3 | Confirm technical feasibility | | | | | 🟧 | | | | | | | | | | | |
| 3 | **Designing** | | | | | | 🟨 | 🟨 | 🟨 | | | | | | | | |
| 3.1 | Create UML diagrams | | | | | | 🟨 | | | | | | | | | | |
| 3.2 | Design UI mockups | | | | | | 🟨 | 🟨 | 🟨 | | | | | | | | |
| 3.3 | Plan database schema | | | | | | | | 🟨 | | | | | | | | |
| 4 | **Development** | | | | | | | | | 🟩 | 🟩 | | | | | | |
| 4.1 | Set up Moodle plugin structure | | | | | | | | | 🟩 | 🟩 | | | | | | |
| 4.2 | Implement browser-based monitoring | | | | | | | | | 🟩 | 🟩 | | | | | | |
| 4.3 | Build real-time alert system | | | | | | | | | 🟩 | 🟩 | | | | | | |
| 4.4 | Develop proctor dashboard | | | | | | | | | 🟩 | 🟩 | | | | | | |
| 5 | **Testing** | | | | | | | | | | | 🟦 | | | | | |
| 5.1 | Conduct unit testing | | | | | | | | | | | 🟦 | | | | | |
| 5.2 | Optimize performance and fix bugs | | | | | | | | | | | 🟦 | | | | | |
| 6 | **Launching** | | | | | | | | | | | | 🟦 | 🟦 | | | |
| 6.1 | Deploy plugin to a test Moodle | | | | | | | | | | | | 🟦 | | | | |
| 6.2 | Prepare documentation | | | | | | | | | | | | 🟦 | | | | |
| 6.3 | Release v1.0 | | | | | | | | | | | | | 🟦 | | | |

### 2.2.2 Schedule Details

In order to create a viable environment to develop our product, our team has come to the conclusion that the Waterfall model is best suited to our needs. It is a sequential model, where the developmental life-cycle is divided into distinct phases. As such, for such a model to work properly the current phase must be completed before the team can move on to the next one. The phases that we have planned out and their details are as follows

- Planning (11 September 2025 - 24 September 2025)

  The first action that must be taken before developing a product is to lay a proper groundwork for further phases by creating a plan. Planning consists of carefully defining the core features of the product, drafting some basic requirements in order to develop such a product, and to agree upon the scope of the product. Our team estimates a duration of three weeks needed to complete this phase.

- Requirements & Research (25 September 2025 - 5 October 2025)

  After the plan has been made, a fortnight will be spent on further polishing the requirements of the product. This phase consists of researching the tools needed for development, creating a document of functional specs, and confirming the technical feasibility of the product.

- Designing (6 October 2025 - 2 November 2025)

  The designing phase of the lifecycle spans over the course of three weeks, and exists to ensure that development can go over smoothly. In order to help front-end development, the team creates a mockup of the UI to serve as a benchmark of how things should look. In order to help back-end development, the team creates an overview of how the database should be structured. Lastly, to help both development processes, the team creates various UML diagrams.

- Development (3 November 2025 - 16 November 2025)

  Serving as the crux of the life-cycle, this phase guides the team into various jobs needed for the product to finally take shape. Over the course of two weeks the team will set up the Moodle plugin structure, implement browser-based monitoring, build a real-time alert system, and develop the dashboard that the proctors will use.

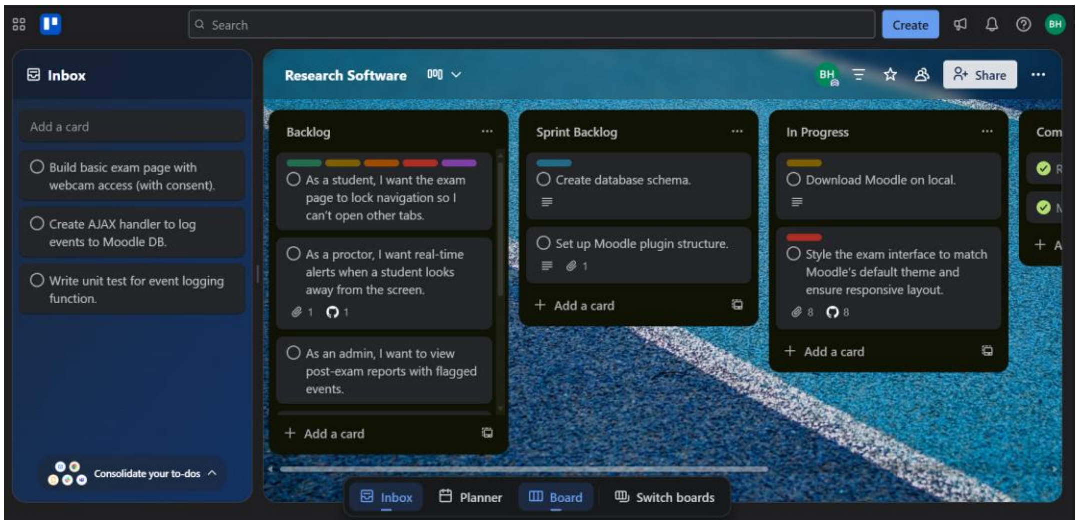
- Testing (17 November 2025 - 23 November 2025)

  Finalizing the product will take a single week, in which the team will go over everything to ensure that the product runs as advertised, and minimize possible unwanted side-effects. To do this, various unit testing will be conducted. If any bugs are found, they will be fixed before the product launches, and if possible optimize performance of the product even further.

- Launching (24 November 2025 - 5 December 2025)

  In order for the client to be able to use the product as advertised, the team will launch the product alongside proper documentation of how the product works. Because this product is a plug-in to Moodle, the team will launch the product embedded within a temporary Moodle environment. If the product proves to be within the client’s expectations, the plug-in then can be used in the client’s Moodle environment.

### 2.2.3 Trello Overview



Inbox
- Build basic exam page with webcam access (with consent).
- Create AJAX handler to log events to Moodle DB.
- Write unit test for event logging function.

Backlog
- As a student, I want the exam page to lock navigation so I can't open other tabs.
- As a proctor, I want real-time alerts when a student looks away from the screen.
- As an admin, I want to view post-exam reports with flagged events.

Sprint Backlog
- Create database schema.
- Set up Moodle plugin structure.

In Progress
- Download Moodle on local.
- Style the exam interface to match Moodle's default theme and ensure responsive layout.

Com

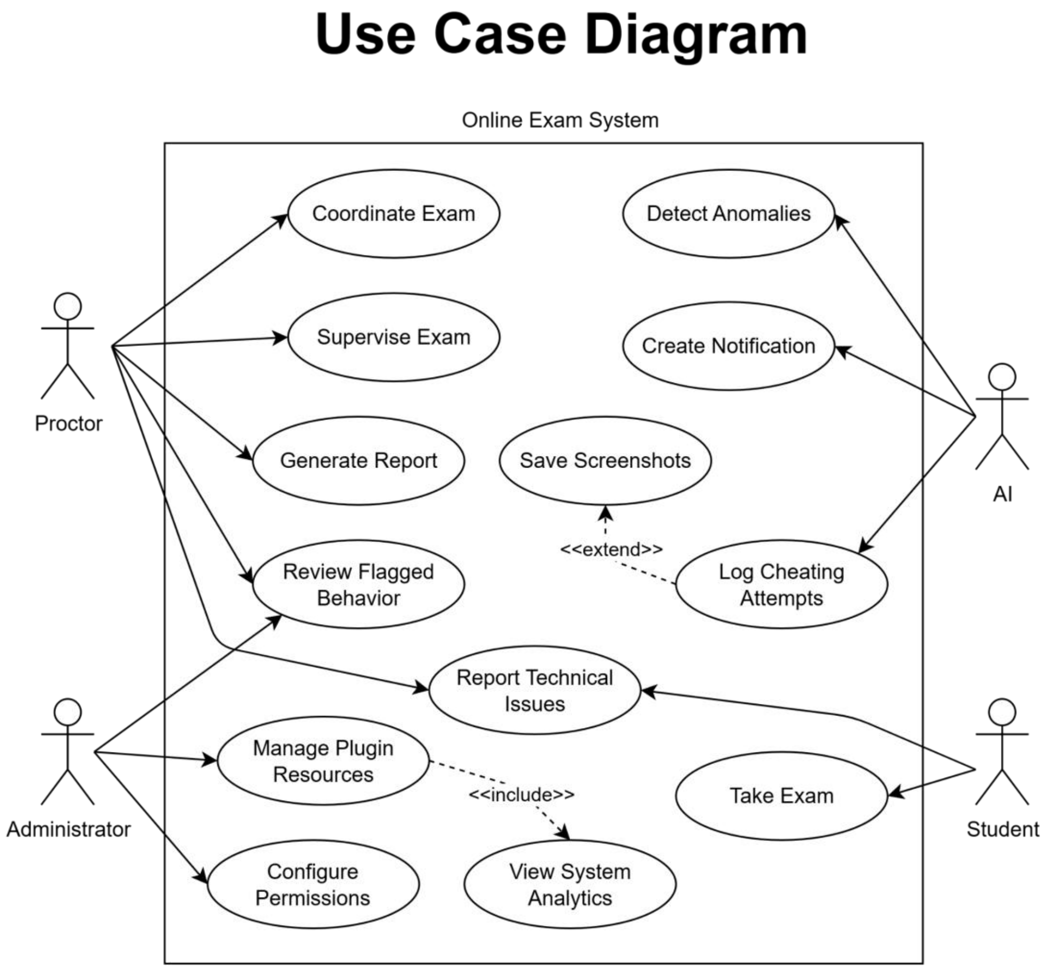
### 2.2.4 Trello Details

Trello is being used as a visual project management tool to organize and track the development of Exam Brother. With its simple board-and-card structure, Trello helps in breaking down complex tasks into manageable pieces, prioritize work, and clearly see what's planned, in progress, or completed.

- Sprint: A short, time-boxed development cycle (2 weeks) during which a set of prioritized features or tasks are built and tested. For Exam Brother, each sprint focuses on delivering a small piece of the plugin, such as implementing tab-switch detection or setting up the database for Moodle.
- Sprint Backlog: The list of specific tasks selected for the current sprint. These are broken-down, actionable items pulled from the main Backlog that you commit to completing within the sprint timeframe, such as creating a handler to log events and styling the exam interface.
- In Progress: Tasks actively being worked on right now. Cards are moved here when starting coding, designing, or testing them to avoid overload and ensure steady progress.
- Completed: Finished tasks that have been coded, tested, and verified. Once a card is moved here, it means the feature or fix is working as intended and ready for review or integration.

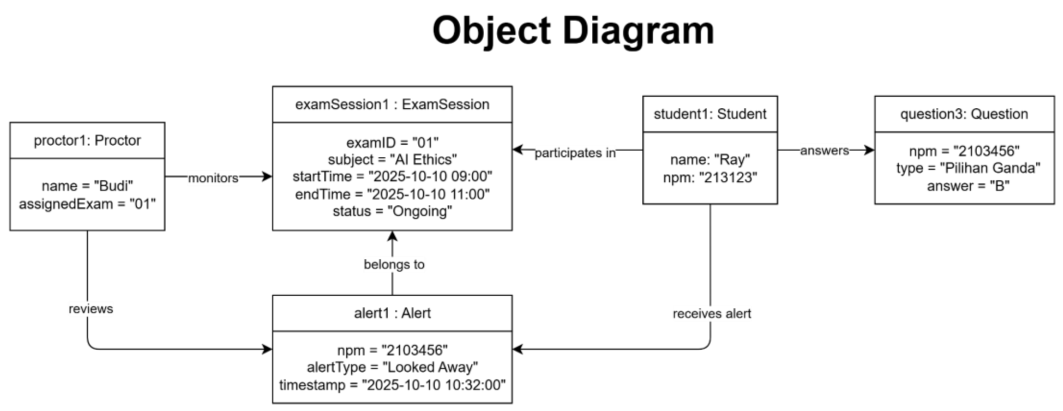
# Chapter 3 - UML Diagrams and Design

## 3.1 Use Case Diagram



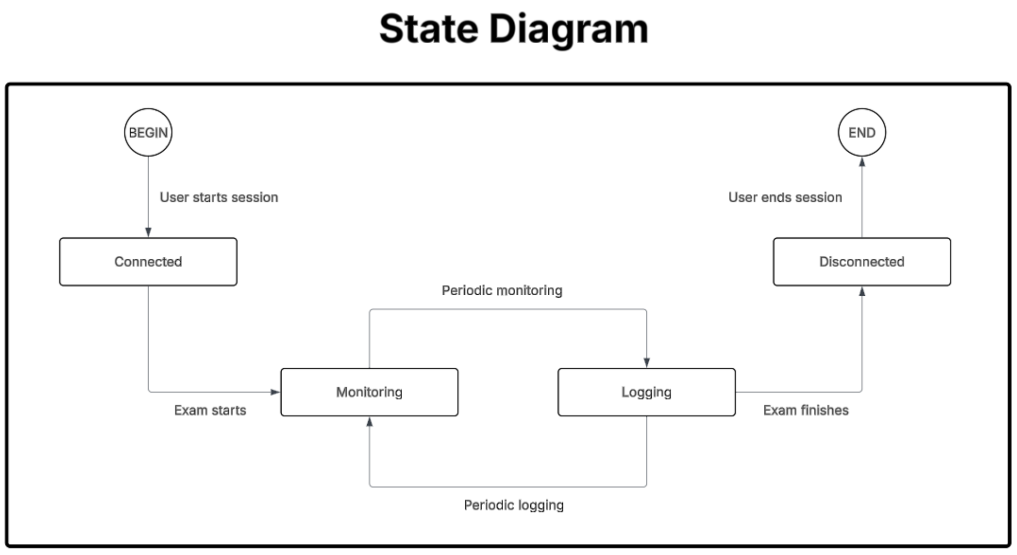
This diagram illustrates the interactions among four main actors: Proctor, Administrator, AI, and Student. The Proctor is responsible for coordinating, supervising, and reviewing suspicious behavior, while AI automatically detects anomalies, generates notifications, and logs cheating attempts. The Student interacts with the system solely to take exams, whereas the Administrator manages plugin resources, configures permissions, and monitors system analytics. Additionally, special relationships such as extend (AI saves screenshots when logging cheating attempts) and include (managing resources includes viewing system analytics) indicate functional dependencies between use cases.

## 3.2 Object Diagram



This object diagram shows a snapshot of real-world instances during an online exam, including a Proctor, Student, ExamSession, Alert, and Question. The Proctor monitors the ongoing exam session, while the student participates in it and answers a multiple-choice question with an answer. When the student looks away from the screen, an alert is triggered with a certain timestamp, which is linked to the student's NPM, while also being reviewed by the Proctor. This diagram illustrates how specific objects interact at runtime, demonstrating the system's behavior in detecting and handling suspicious activity during an exam.

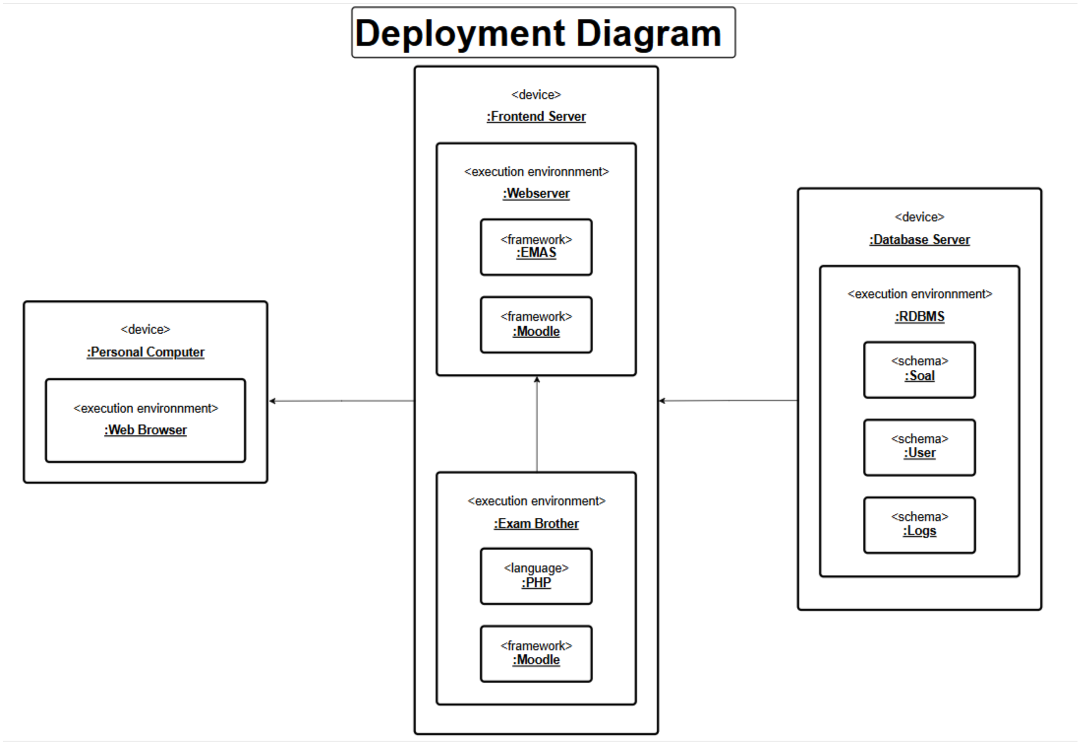
## 3.3 State Diagram



This state diagram illustrates the lifecycle of an exam session in the Exam Brother system, beginning when a user starts the session and entering the Connected state.

Once the exam officially begins, the system transitions to the Monitoring state, where it actively observes student behavior, periodically moving to Logging to record events before returning to monitoring. When the exam concludes, the system moves to the Disconnected state, signaling the end of the session, which then leads to the final END state. The diagram highlights a continuous loop between monitoring and logging during the exam, ensuring real-time tracking and data capture until completion.

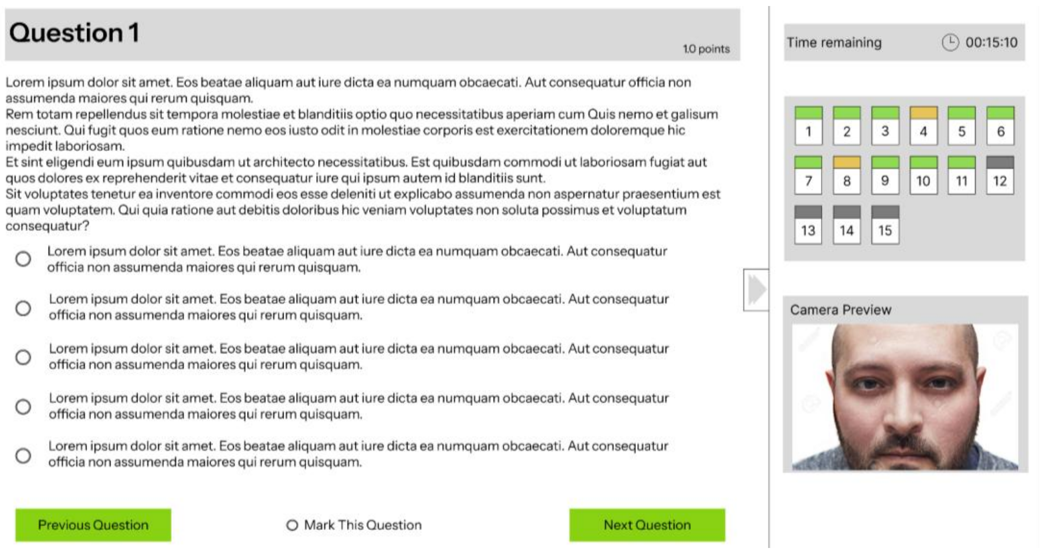
## 3.4 Deployment Diagram



This deployment diagram illustrates the physical architecture of the Exam Brother system. The user interacts via a Personal Computer or Laptop running a Web Browser, which connects to the Frontend Server hosting both the EMAS and Moodle frameworks. The Exam Brother plugin, built with PHP and integrated with Moodle, runs on the same server and communicates with the Database Server, which stores data through an RDBMS using schemas for questions, users, and logs.

## 3.5 Design

### 3.5.1 Student Exam Page



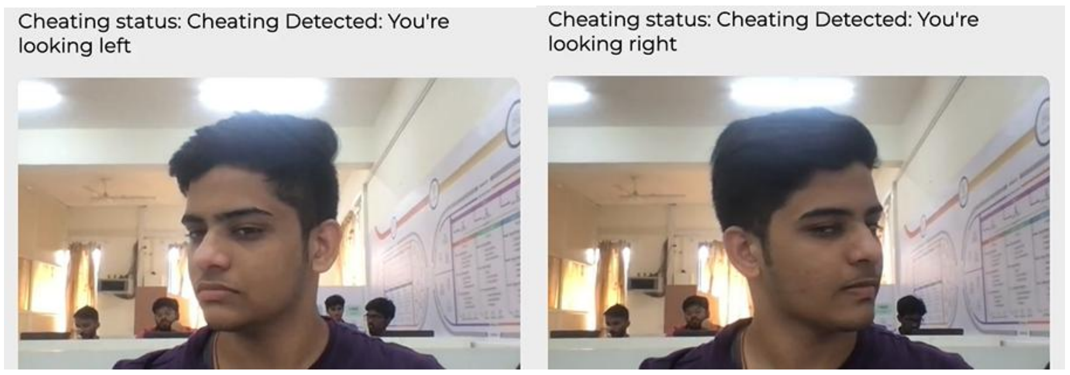
**Question 1**

1.0 points

Lorem ipsum dolor sit amet. Eos beatae aliquam aut iure dicta ea numquam obcaecati. Aut consequatur officia non assumenda maiores qui rerum quisquam.
Rem totam repellendus sit tempora molestiae et blanditiis optio quo necessitatibus aperiam cum Quis nemo et galisum nesciunt. Qui fugit quos eum ratione nemo eos iusto odit in molestiae corporis est exercitationem doloremque hic impedit laboriosam.
Et sint eligendi eum ipsum quibusdam ut architecto necessitatibus. Est quibusdam commodi ut laboriosam fugiat aut quos dolores ex reprehenderit vitae et consequatur iure qui ipsum autem id blanditiis sunt.
Sit voluptates tenetur ea inventore commodi eos esse deleniti ut explicabo assumenda non aspernatur praesentium est quam voluptatem. Qui quia ratione aut debitis doloribus hic veniam voluptates non soluta possimus et voluptatum consequatur?

- ○ Lorem ipsum dolor sit amet. Eos beatae aliquam aut iure dicta ea numquam obcaecati. Aut consequatur officia non assumenda maiores qui rerum quisquam.
- ○ Lorem ipsum dolor sit amet. Eos beatae aliquam aut iure dicta ea numquam obcaecati. Aut consequatur officia non assumenda maiores qui rerum quisquam.
- ○ Lorem ipsum dolor sit amet. Eos beatae aliquam aut iure dicta ea numquam obcaecati. Aut consequatur officia non assumenda maiores qui rerum quisquam.
- ○ Lorem ipsum dolor sit amet. Eos beatae aliquam aut iure dicta ea numquam obcaecati. Aut consequatur officia non assumenda maiores qui rerum quisquam.
- ○ Lorem ipsum dolor sit amet. Eos beatae aliquam aut iure dicta ea numquam obcaecati. Aut consequatur officia non assumenda maiores qui rerum quisquam.

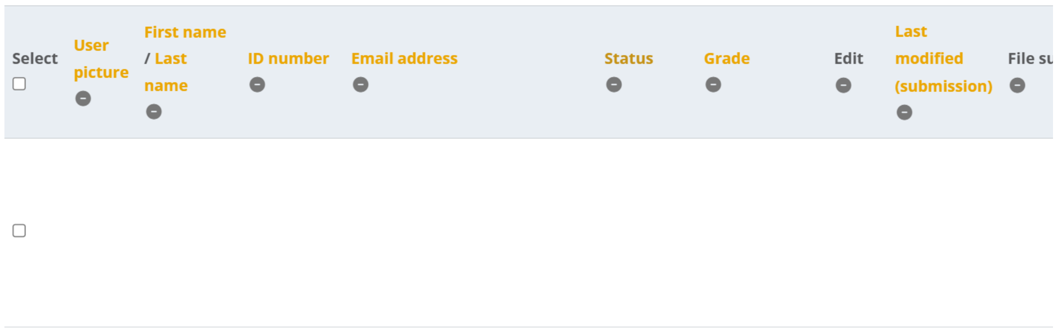
### 3.5.2 Real-time Alert



Cheating status: Cheating Detected: You're looking left

Cheating status: Cheating Detected: You're looking right

### 3.5.3 Proctor Dashboard



| Select ☐ | User picture | First name / Last name | ID number | Email address | Status | Grade | Edit | Last modified (submission) | File su |
|---|---|---|---|---|---|---|---|---|---|
| ☐ | | | | | | | | | |

# Chapter 4 - Test Plan and Results

## 4.1 Testing Introduction

Exam Brother implements standard software testing and test plan documentation procedures to validate our product. This documentation aims to provide the reader with helpful explanations of the quirks and bugs that currently exist. Additionally, this document serves as proof that the final product has been thoroughly tested and upholds professional standards.

## 4.2 Testing Scope

### 4.2.1 In Scope

Exam Brother being a plugin to a Moodle environment means that all of the tests that will be implemented will test the functionalities and usability of the plugin, and not the Moodle environment as a whole. As such, the tests will cover the following components and features:

- Moodle integration
  Evaluates how well the plugin adapts and functions across different Moodle environments.
- Cheating attempt detection
  Assesses the accuracy, responsiveness, and resource usage of the system when detecting cheating attempts.
- Cheating attempt logging
  Measures how clear, complete, and useful the generated cheating attempt logs are for review.
- Live dashboard monitoring
  Evaluates the usability and overall user experience of the real-time monitoring dashboard.
- Post exam reports
  Assesses the readability, accessibility, and overall usefulness of the post-exam reports.

In addition to the features and components described above, the tests will also identify any bugs that may arise during the use of the plugin.

#### 4.2.2 Out of Scope

The tests that will be performed will not take into account or report bugs and other aspects related to the Moodle environment used to test this plugin. This is because the environment may differ from one course to another, which can lead to bugs or quirks that are unique to a particular course. Additionally, the tests will not cover factors that may vary on the back end of the environment, such as the capabilities of the application server or the database server. Therefore, no stress testing of the environment will be conducted.

### 4.3 Quality Objective

The main aim of this testing project is to ensure that Exam Brother, also referred to as the Application Under Test (AUT), achieves a high level of quality. The goal is to deliver a reliable, trustworthy, effective, and easy-to-implement AI surveillance solution for online exams.

- Meeting Functional and Non-Functional Requirements

  We are committed to verifying that Exam Brother (AUT) fulfills its intended functions. This includes ensuring that all features operate correctly, compatibility is maintained across environments, and overall performance meets the required standards.
- Ensure the AUT meets the Client Standards

  Our testing ensures that Exam Brother (AUT) adheres to the standards set by the client. We carefully evaluate the AUT to confirm that it can perform its tasks reliably and consistently.
- Identification and Resolution of Bugs/Issues

  A key priority of our testing process is to identify and address any issues or bugs in Exam Brother (AUT) before release. This ensures that when users begin using the system, it operates smoothly and without disruptive problems.

## 4.4 Roles and Responsibilities

Here are the roles and responsibilities involved in developing Exam Brother:

- Developers

  Developers serve as the architects responsible for the design, construction, and ongoing maintenance of Exam Brother. The following individuals are the developers responsible for building, maintaining, and testing Exam Brother.

  **Anthonius Hendhy Wirawan, Back-End Programmer and AI Engineer**

  Responsibilities:

  - Design and implement the back-end ecosystem of the product
  - Create pathways for the front-end to interact with the database through APIs
  - Implement MediaPipe for AI proctoring features of the product

  **Bryan Herdianto, Front-End Programmer and Application Tester**

  Responsibilities:

  - Design and plan the UI with user experience in mind
  - Implement new UI for Moodle-based pages
  - Integrate the front-end with the back-end of the product to ensure smooth experience

  **Maxwell Zefanya Ginting, Back-End Programmer and AI Engineer**

  Responsibilities:

  - Design and implement the back-end ecosystem of the product
  - Create pathways for the front-end to interact with the database through APIs
  - Tweak the AI model to ensure correctness and reliability of the algorithm

- QA Analyst (Maxwell Zefanya Ginting)

  Responsibility: QA Analysts are pivotal in upholding the quality standards exam brother. They play a central role in meticulously testing the application to unearth bugs and areas for improvement

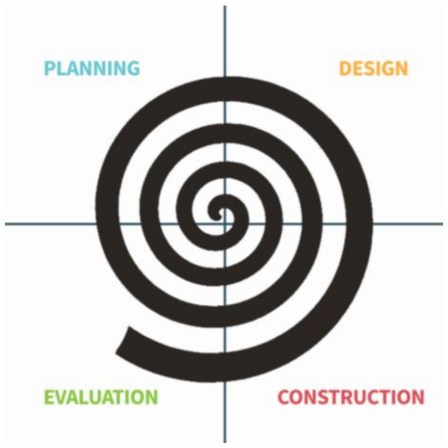
- Test Manager (Anthonius Hendhy Wirawan)

Responsibility: Test Managers serve as overseers of the testing process, crafting and executing the overarching strategy to meet testing goals. Tasked with the allocation of responsibilities among the testing team, they closely monitor and report on the progress of testing activities.

## 4.5 Test Methodology

For the purpose of testing the developed product, we will be using the Spiral Model methodology. Development is carried out incrementally through multiple short cycles, with testing conducted at the end of each cycle. This methodology allows us to adapt to changing requirements and reduce risk throughout the process.

After each testing phase, all bugs and errors identified must be validated to ensure they are legitimate, reproducible, and not duplicates of previously reported issues. They are then classified according to their severity (the impact on the application) and their priority (the urgency of addressing them). Once classified, the bugs are assigned to appropriate members of the development team for investigation and resolution.



## 4.6 Test Levels

In the development phase, the testing will consist of:

- Unit Testing

  In this test level, individual functionality such as the AI face detection, alerting system, will be tested. The goal of this test level is to ensure that every individual unit will behave properly. The test suite will consist of simple inputs to its component and its expected output. The test result will be the report of which component passed the unit test, which component didn't, and which test is the issue.

- Integration Testing

  In this test level, a group of individual functionality will be tested as a whole. The goal of this level is to ensure that the whole functionality of the system will behave and interact with each other properly. The test suite will consist of multiple inputs to different functionality, and each functionality's expected output.

- Regression Testing

  In this test level, previously tested components of the Exam Brother system will be re-evaluated to ensure that recent code changes, bug fixes, or feature updates have not introduced new defects. The objective is to confirm that the system continues to function correctly after modifications and that existing functionalities remain stable and reliable.

- Acceptance Testing

  This is the final test level, performed to validate that the Application Under Test (AUT) meets the client's expectations and requirements. Acceptance testing determines whether the system is ready for deployment. The goal is to ensure that Exam Brother is reliable, user-friendly, and capable of supporting online exam scenarios without critical issues. Once the AUT passes acceptance testing, it is considered suitable for release.

## 4.7 Bug Triage

Bug triage is the **systematic process** of reviewing and classifying reported bugs according to their severity and impact on the software application. This process is crucial in ensuring that discovered bugs are properly prioritized, assigned to the appropriate developers, and efficiently resolved. Our team has devised a **three-step process** that follows the bug triage principle. That steps are as follows:

1. Bug Detection & Confirmation

2. Priority Scaling
3. Bug Fixing

These three steps will be fitted into our current spiral development cycle, and will provide us with a more standardized method for dealing with current/future bugs.

## 4.8 Suspension Criteria and Resumption Requirements

### 4.8.1 Suspension Criteria

Testing activities will be suspended when conditions arise that prevent meaningful or reliable testing from continuing. Testing for the Exam Brother (AUT) will be suspended if any of the following occur:

- Critical defects that block core functionalities (e.g., face detection failing to initialize, dashboard not loading, plugin failing to start).
- System instability, such as repeated crashes, frozen interfaces, or unresponsive components.
- Broken or incomplete builds that prevent testers from executing test cases as intended.
- Test environment failures, including issues with Moodle integration, server outages, or database connection problems.
- Missing or invalid test data/configuration required to proceed with a specific test level.

When any of these conditions occur, testing for the affected components will be halted to avoid inaccurate results and wasted resources.

### 4.8.2 Resumption Criteria

Testing activities will resume once the conditions that triggered the suspension have been resolved. Testing for Exam Brother (AUT) may resume when:

- All critical and blocking defects have been fixed and verified through re-testing.

- A stable and functional build is made available, with confirmation from the development team.
- The test environment is restored, including Moodle configuration, server functionality, and database connectivity.
- All required test data and configurations are corrected and validated.
- The testing team confirms that test cases can be executed without obstruction.

Once these criteria are met, testing will proceed from the point at which it was suspended, ensuring continuity and completeness.

## 4.9 Test Completeness

Testing for the Exam Brother (AUT) will be considered complete when the following criteria have been met:

- All planned test cases have been executed, including both manual and automated tests across all defined test levels (Unit, Integration, System, and Acceptance Testing).
- Test coverage reaches 100% for the features included in the testing scope, ensuring that every functional and non-functional requirement has been fully validated.
- All critical and major bugs have been fixed, verified, and closed.
- All minor or low-priority bugs are documented, and a plan for their resolution in a future release has been agreed upon with the development team.
- The system demonstrates stable performance in the test environment during the final test cycle.
- All test documentation is completed, including test reports, bug reports, and updated test cases.

When all these conditions are satisfied, the testing process will be formally concluded, and the Exam Brother (AUT) will be considered ready for delivery or deployment.

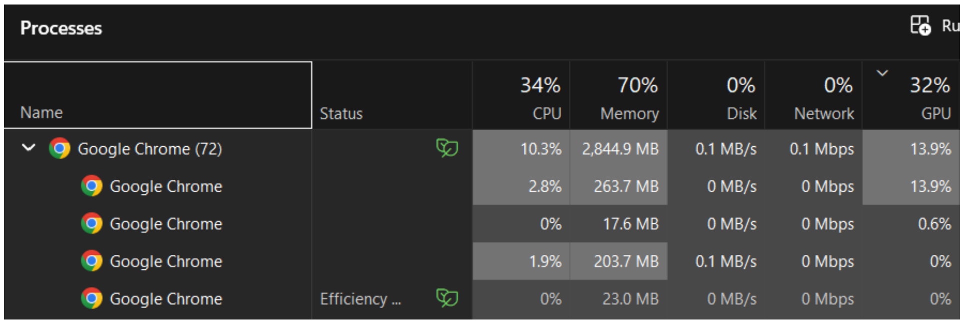
## 4.10 Bug Reporting Process

### 4.10.1 Target Areas

This section outlines the key functional areas of the plugin that will be the primary focus during bug detection activities. These areas were selected based on their complexity, user interaction frequency, and potential impact on system reliability. The target areas include components such as resource usage monitoring, cheating-attempt detection, and other critical operational features that are most likely to exhibit defects or unexpected behavior.

- Resource Usage

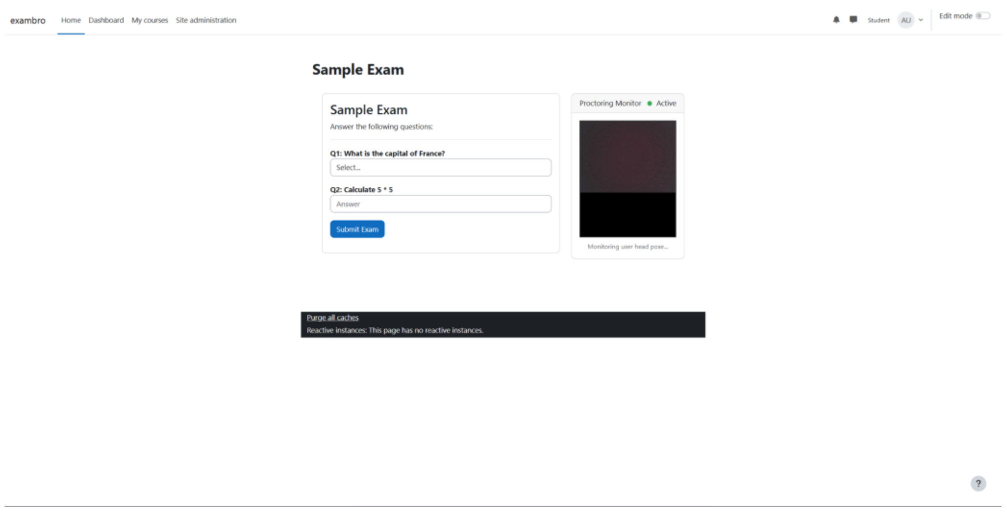
  The plugin consistently utilizes no more than approximately 10% CPU and 15% GPU while running, whether in proctor mode or student mode. Based on the tester's system configuration, we believe these usage levels are low enough to support operation in more complex Moodle environments, as well as on student devices with reasonably weaker hardware.



| Name | Status | 34% CPU | 70% Memory | 0% Disk | 0% Network | 32% GPU |
|---|---|---|---|---|---|---|
| Google Chrome (72) | | 10.3% | 2,844.9 MB | 0.1 MB/s | 0.1 Mbps | 13.9% |
| Google Chrome | | 2.8% | 263.7 MB | 0 MB/s | 0 Mbps | 13.9% |
| Google Chrome | | 0% | 17.6 MB | 0 MB/s | 0 Mbps | 0.6% |
| Google Chrome | | 1.9% | 203.7 MB | 0.1 MB/s | 0 Mbps | 0% |
| Google Chrome | Efficiency ... | 0% | 23.0 MB | 0 MB/s | 0 Mbps | 0% |

- Moodle Integration

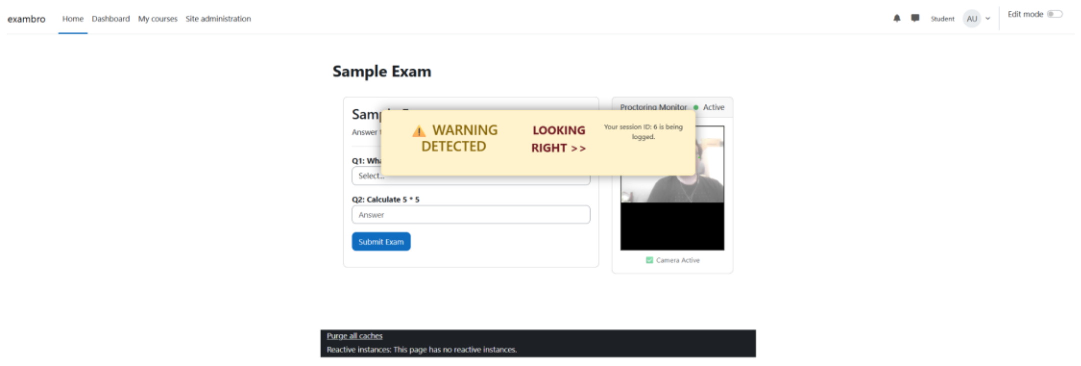
  At the moment, comprehensive Moodle integration testing is not possible because the plugin has only been evaluated in a single Moodle environment. However, within the current testing environment set up by our team, the plugin functions as intended. Students are able to take exams, and proctors can successfully monitor for any suspicious activity.



Reactive instances: This page has no reactive instances.

- Cheating Attempt Detection

  The plugin detects cheating attempts once every 100 milliseconds, or 10 times per second. We consider this frequency high enough to remain responsive, but not so frequent that it overloads the user's system resources. The detection accuracy is also satisfactory, as the plugin was able to identify the tester's attempts at suspicious behavior, such as looking to either side of the webcam.

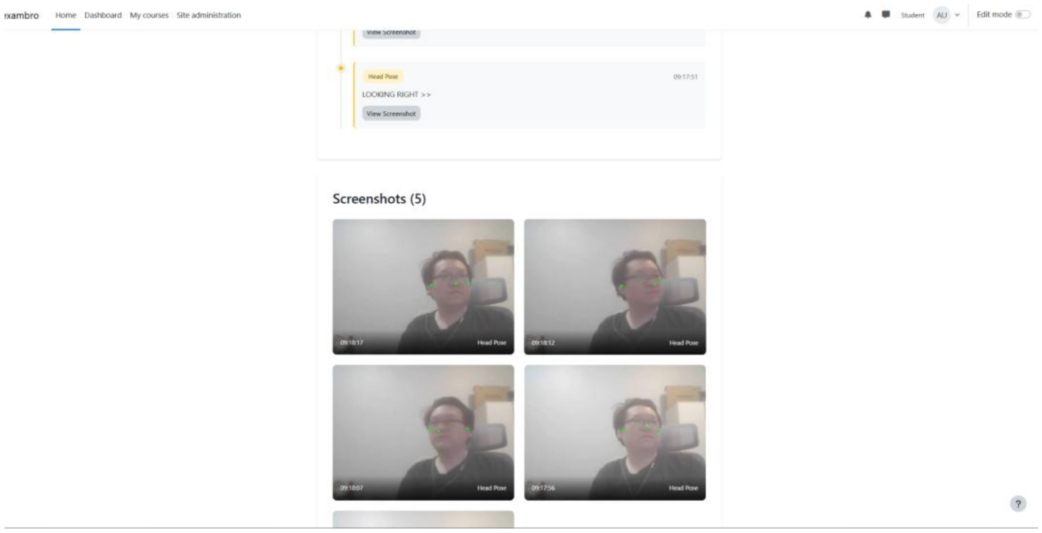


Your session ID: 6 is being logged.

Reactive instances: This page has no reactive instances.

- Cheating Attempt Logging

  Proctors can review any cheating attempts from students through the dashboard page. On this page, the proctor can see webcam captures that the plugin has flagged as suspicious, along with a timeline showing when each event occurred. From this information, the proctor can determine whether the student was actually engaging in cheating behavior.

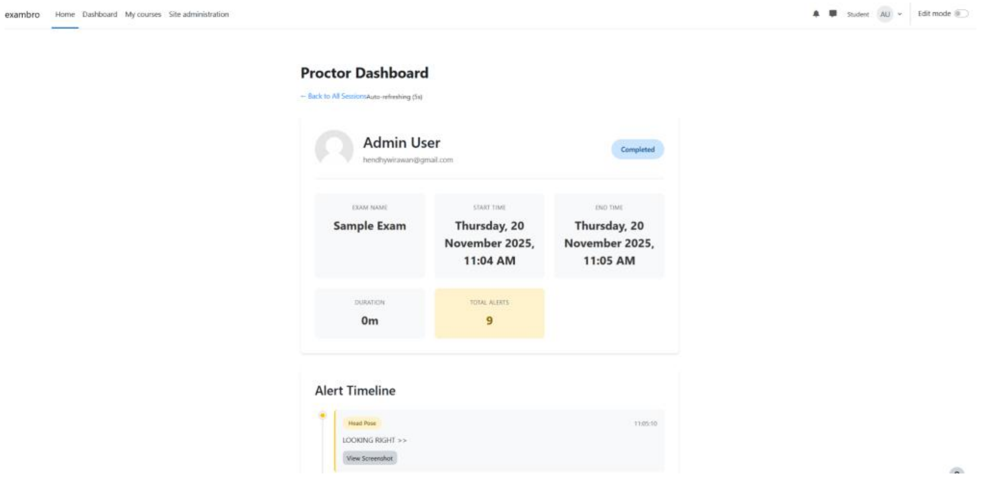


View Screenshot

Head Pose 09:17:51
LOOKING RIGHT >>
View Screenshot

Screenshots (5)

- Live Dashboard Monitoring
  While the exam is in progress, the proctor can access the dashboard to view live alerts for the current session. Any cheating attempts are immediately sent to the proctor, who can then review and analyze them as they occur.



Proctor Dashboard

← Back to All SessionsAuto-refreshing (5s)

Admin User
hendhywirawan@gmail.com
Completed

| EXAM NAME | START TIME | END TIME |
|---|---|---|
| Sample Exam | Thursday, 20 November 2025, 11:04 AM | Thursday, 20 November 2025, 11:05 AM |

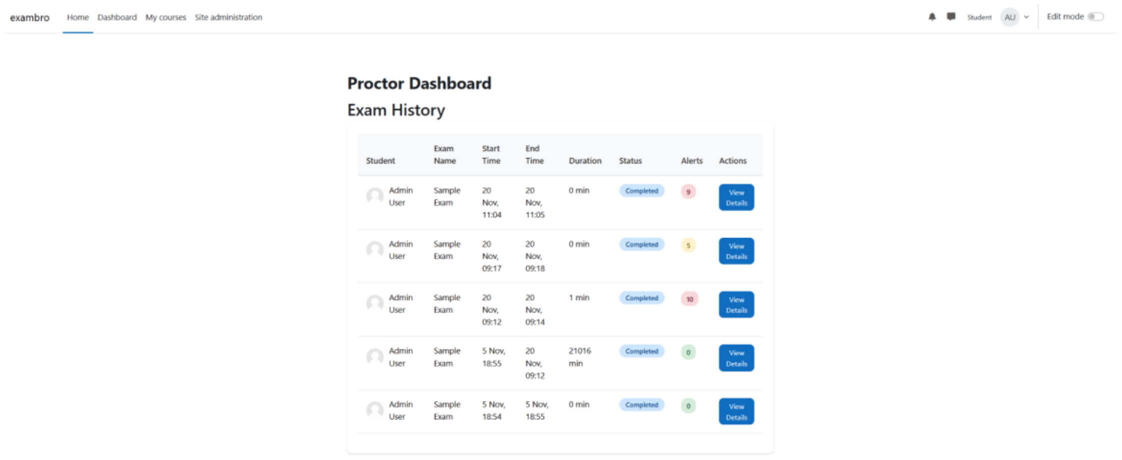
| DURATION | TOTAL ALERTS |
|---|---|
| 0m | 9 |

Alert Timeline

Head Pose 11:05:10
LOOKING RIGHT >>
View Screenshot

- Post Exam Reports

After the session has concluded, the proctor can review any cheating attempts recorded by the plugin. By accessing the exam history page, they can see how many alerts each student received.



Proctor Dashboard

Exam History

| Student | Exam Name | Start Time | End Time | Duration | Status | Alerts | Actions |
|---|---|---|---|---|---|---|---|
| Admin User | Sample Exam | 20 Nov, 11:04 | 20 Nov, 11:05 | 0 min | Completed | 9 | View Details |
| Admin User | Sample Exam | 20 Nov, 09:17 | 20 Nov, 09:18 | 0 min | Completed | 5 | View Details |
| Admin User | Sample Exam | 20 Nov, 09:12 | 20 Nov, 09:14 | 1 min | Completed | 10 | View Details |
| Admin User | Sample Exam | 5 Nov, 18:55 | 20 Nov, 09:12 | 21016 min | Completed | 0 | View Details |
| Admin User | Sample Exam | 5 Nov, 18:54 | 5 Nov, 18:55 | 0 min | Completed | 0 | View Details |

### 4.10.2 Severity Level

Following the bug triage process, we conclude that the next step in developing this plugin is to fix the bugs that have been identified. Before proceeding, these bugs will be categorized based on their importance. For now, they are represented by numerical priority values, as shown in the following table.

| Priority | Description |
|---|---|
| 0 | Will never be fixed. The bug does not affect the core functionality of the plugin and does not cause any inconvenience. |
| 1 | Will be fixed later. The bug is inconvenient for users but does not interfere with the core functionality of the plugin. |
| 2 | Will be fixed immediately. The bug adversely affects the core functionality of the plugin and will be prioritized during bug fixing. |
| 3 | Requires urgent fixing. The development team will halt all other work and address this bug as soon as possible. |

## 4.11 Resource & Environment Needs

### 4.11.1 Testing Tools

In order to run the tests, we use:

- Google Chrome, version 142.0.7444.176 64-bit
- Microsoft Edge, version 142.0.3595.94 64-bit
- Task Manager
- PHPUnit

### 4.11.2 Test Environment

The tests that have been outlined in the previous section uses the following hardware specifications to run the plugin:

| | |
|---|---|
| ● CPU | : 11th Gen Intel(R) Core(TM) i7-11600H |
| ● GPU | : NVIDIA GeForce RTX 3050 Laptop GPU |
| ● RAM | : 24 GB |
| ● Peripherals | : ASUS TUF F15 built-in webcam |

Meanwhile the software specifications are as follows:

| | |
|---|---|
| ● Operating System | : Windows 11 Home Single Language, 25H2 |
| ● Web Browser | : Google Chrome ver 142.0.7444.176 64-bit |

During the testing, the system acts as both the back-end server, front-end server, and the client. The system never utilized more than 10% of CPU resource and 15% of GPU resource. As such, we conclude that the minimum hardware requirements are as follows:

| | |
|---|---|
| ● CPU | : 1 GHz Dual-Core 64-bit processor |
| ● GPU | : DirectX 12 compatible GPU with WDDM 2.0 driver |
| ● RAM | : 8 GB |
| ● Peripherals | : Web camera |

With the same set of software as the minimum software requirements.

## 4.12 Terms/Acronyms

| Term/Acronym | Definition |
|---|---|
| API | Application Program Interface |
| AUT | Application Under Test |
| Bug Triage | The process of evaluating, prioritizing, and categorizing bugs before assigning them for fixing |
| Moodle | A Learning Management System (LMS) used for delivering courses and managing online exams |
| QA | Quality Assurance |
| Test Coverage | The measure of how much of the system's functionality has been tested |
| UI/UX | User Interface/User Experience |

## 4.13 Testing Closure

Exam Brother is an online AI-based proctoring tool designed to help proctors catch cheating attempts and/or suspicious behaviours exhibited by test takers in an exam session. As of the time of writing this document, the plugin is in version 1.0.0. Our team has devised a testing regime to be integrated into our current development cycle using the bug triage as its guidelines. The intent is to make sure that the plugin follows its requirements and stays as bug-free as possible throughout its lifecycle.

The testing regime will test 5 different aspects of the plugin, which are its Moodle integration, cheating attempt detection, cheating attempt logging, live dashboard monitoring, and post exam reports. Results show that the current version satisfies all five aspects to a satisfying degree. Besides that, testing has also exposed two different bugs, of which only one requires immediate attention. Finally, testing the plugin also reveals its hardware and software requirements, which is the minimum specification needed to run this plugin.

Further development cycles will refer to this document and other testing documents made in the future in order to set a methodological way of fixing bugs found in this document. For now, the immediate next step the team will take is to fix the bug needing urgent care, which is the permanent alert bug for exam-takers.

# Chapter 5 - Log Book

## 5.1 Overview

Coding log books serve to document the process of writing and analyzing code, which helps with debugging, project management, and creating a history of work. It provides a record of daily activities, technical challenges, and solutions, which is useful for personal reference, future troubleshooting, and for justifying programming decisions.

## 5.2 Log Book

| Date | Author | Goals | Result |
|---|---|---|---|
| October 31st, 2025 | Maxwell | Create an initial model to detect facial features | Created the model successfully using Mediapipe in Python |
| November 1st, 2025 | Maxwell | Improve initial Python model by adding the ability to detect face looking left or right | Model partly implemented with some bugs existing impacting the accuracy of the model |
|  | Bryan | Create UI and front end feature, and implementing moodle | Created basic UI frontend with existing bugs |
|  | Hendhy | Connect frontend to database table | Connect frontend to database |
| November 2nd, 2025 | Maxwell | Python model bugfix | Removed remaining bugs and cleaned up the model |
| November 16th, 2025 | Hendhy | Porting python's model to javascript | Implemented AI to moodle plugin |
| November 20th, 2025 | Bryan | Added alerting feature to front-end when back-end model detects suspicious activity | Created basic alerting feature requiring further improvements |
| November 22nd, 2025 | Hendhy | Adding and polishing | Reducing alert stick time, |

| | | | |
|---|---|---|---|
| | | AI features | and adding alert if there is no face detected |
| November 30th, 2025 | Bryan | Fixing role-agnostic dashboard bug | Dashboards can now differentiate between proctors and students. |
| December 2nd, 2025 | Bryan | Adding web tab switch detection system. | Created a system to detect when a student switches browser tabs during an exam, improving cheating prevention and monitoring accuracy. |
| | | Integration of Exam Brother with the Moodle main page. | Integrated Exam Brother to appear directly on Moodle's main interface, making it easier for users to access the tool without navigating to separate pages. |

# Chapter 6 - User Manual

## 6.1 Introduction

Within this section the reader will be given an in-depth manual with which they can use to operate the plugin to its maximum capacity. The manual will start simple, outlining the basic requirements to get the plug-in to work in a Moodle environment. After that, this manual will cover how to set-up and use this plug-in in an exam environment, be it for the administrators, proctors, or students.

## 6.2 Requirements

In order to run and use this plug-in, there are a couple of minimum requirements that must be met. These requirements differ between the administrator, proctors, and students.

### 6.2.1 Administrators

- Server hardware with internet connection
- XAMPP Control Panel
- Moodle with its respective course(s)
- PostgreSQL database
- ExamBrother Moodle plug-in

### 6.2.2 Proctors

- Client hardware that meets minimum requirements
- Internet Connection
- Browser with Moodle support

### 6.2.3 Students

- Client hardware that meets minimum requirements
- Internet Connection
- Browser with Moodle support
- Functional web-camera

## 6.3 Administrators

### 6.3.1 Getting Started

- The plugin is already in place at `...\xampp\htdocs\moodle\local\myplugin`.
- Log-in to Moodle using an administrator account.
- If the user logs in with the correct credentials, they will be redirected to their course's main page shortly as an administrator.
- Go to **Site Administration**, then to **Notifications**.
- You should see the plugin ready to install.
- Click **Upgrade Moodle database now**.
- The plugin will create the necessary database tables.

### 6.3.2 Adding roles to proctors and students

- Further reading can be accessed here.
- In the moodle main page, go to **Site Administration**.
- Go to **Users**, and then **Permissions**.
- Click **Assign system roles**.
- Upload the .csv file that contains the list of proctors and students with their respective roles.
- If successful, proctors and students should now be able to access Exam Brother from their respective dashboards.

## 6.4 Proctors

### 6.4.1 Getting Started

- Make sure you have a stable internet connection..
- Log-in to Moodle using a proctor/teacher account using a web browser that meets the requirements..

- If the user logs in with the correct credentials, they will be redirected to their course's main page shortly as a proctor/teacher.
- Within the dashboard, within the same section as **Course**, **Settings**, etc., click the **More** drop-down option.
- Search for and click **Exam Brother**. You should be taken to the plug-in main page.
- You can now access proctor functionalities as a proctor.

### 6.4.2 Live Monitoring

- Within the Exam Brother main-page, click the **View Live Monitor** button. It should take you to the live monitor page.
- Within the live monitor page, you should see a list of people currently connected to the exam session.
- You may view details of each participant by clicking **View Details**.
- If you want to end the exam session, click **End Session**.

### 6.4.3 Exam History

- Within the Exam Brother main-page, click the **View Dashboard** button. It should take you to the proctor dashboard page.
- Within the proctor dashboard monitor page, you should see a list of people alongside their session status, alert count, and further actions.
- If you want to see the details of alerts generated by a particular user, click on **View Details** on that particular user. It will show you the details of each warning, including a timestamp, type of conduct, and a screenshot of your web-camera during that timestamp.

## 6.5 Students

### 6.5.1 Getting Started

- Log-in to Moodle using a student account using a web browser that meets the requirements.
- If the user logs in with the correct credentials, they will be redirected to their course's main page shortly as a student.

### 6.5.2 Taking an Exam

- Make sure you have a stable internet connection.
- Prepare a functional web-camera or built-in camera in your laptop/PC. If there is none, the plug-in will warn you and you won't be able to proceed.
- You may take the exam as you would usually, usually by clicking **Start**.
- Within the exam page, alongside the normal Moodle exam window, you should also see your webcam within that page.
- The camera alongside the plug-in will try to detect any attempts at dishonest conduct, which is where the stable internet connection plays a part.
- You can drag the webcam display window around to where it's the least intrusive for you. This does not hamper the performance of the camera.
- If during the exam the plug-in notices any dishonest conduct, it will notify you in the form of an alert. Take care to not get too many alerts.
- If you have finished the exam, you may end the session as usual.

# Chapter 7 - Test Results

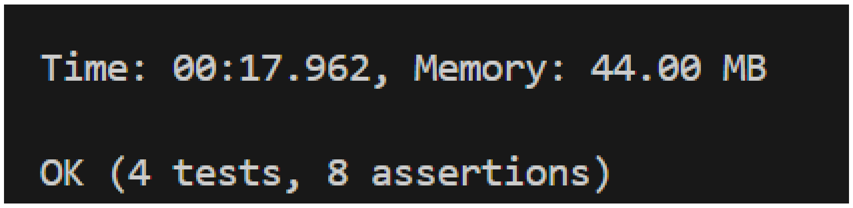
## 7.1 Unit Testing

For unit testing, we use PHPUnit, the standard testing framework for PHP applications and Moodle plugins. PHPUnit allows us to verify that individual components of the system function correctly in isolation.

| Test Case | Description |
|---|---|
| test_create_session | Verifies that a new exam monitoring session can be created and stored in the database |
| test_log_alert | Confirms that alerts (head pose violations) can be logged against a session |
| test_complete_session | Ensures that a session can be properly marked as completed with an end time |
| test_tab_switch_alert | Validates that tab switch violations are correctly recorded with the proper alert type |

```php
<?php

namespace local_myplugin;

defined('MOODLE_INTERNAL') || die();

class lib_test extends \advanced_testcase
{

    /**
     * Test that a session can be created
     */
    public function test_create_session()
    {
        global $DB;

        $this->resetAfterTest(true);

        // Create a test user
        $user = $this->getDataGenerator()->create_user();

        // Create a session record
        $session = new \stdClass();
        $session->userid = $user->id;
        $session->examname = 'Test Quiz';
        $session->starttime = time();
        $session->status = 'active';
        $session->timecreated = time();
        $session->timemodified = time();

        $sessionid = $DB->insert_record('local_myplugin_sessions', $session);

        // Assert the session was created
        $this->assertNotEmpty($sessionid);

        // Verify we can retrieve it
        $retrieved = $DB->get_record('local_myplugin_sessions', ['id' => $sessionid]);
        $this->assertEquals('Test Quiz', $retrieved->examname);
        $this->assertEquals('active', $retrieved->status);
    }

    /**
     * Test that an alert can be logged
     */
    public function test_log_alert()
    {
        global $DB;

        $this->resetAfterTest(true);

        // Create a test user and session
        $user = $this->getDataGenerator()->create_user();

        $session = new \stdClass();
        $session->userid = $user->id;
        $session->examname = 'Test Quiz';
        $session->starttime = time();
        $session->status = 'active';
        $session->timecreated = time();
        $session->timemodified = time();
        $sessionid = $DB->insert_record('local_myplugin_sessions', $session);

        // Create an alert
        $alert = new \stdClass();
        $alert->sessionid = $sessionid;
        $alert->userid = $user->id;
        $alert->alerttype = 'head_pose';
        $alert->description = 'LOOKING LEFT';
        $alert->severity = 1;
        $alert->timecreated = time();

        $alertid = $DB->insert_record('local_myplugin_alerts', $alert);

        // Assert the alert was created
        $this->assertNotEmpty($alertid);

        // Verify the alert count
        $alertcount  =  $DB->count_records('local_myplugin_alerts',  ['sessionid'  =>
$sessionid]);
        $this->assertEquals(1, $alertcount);
    }

    /**
     * Test session completion
     */
    public function test_complete_session()
    {
        global $DB;

        $this->resetAfterTest(true);

        // Create a test user and session
        $user = $this->getDataGenerator()->create_user();

        $session = new \stdClass();
        $session->userid = $user->id;
        $session->examname = 'Test Quiz';
        $session->starttime = time();
        $session->status = 'active';
        $session->timecreated = time();
        $session->timemodified = time();
        $sessionid = $DB->insert_record('local_myplugin_sessions', $session);

        // Complete the session
        $session->id = $sessionid;
        $session->status = 'completed';
        $session->endtime = time();
        $session->timemodified = time();
        $DB->update_record('local_myplugin_sessions', $session);

        // Verify the session is completed
        $retrieved = $DB->get_record('local_myplugin_sessions', ['id' => $sessionid]);
        $this->assertEquals('completed', $retrieved->status);
        $this->assertNotEmpty($retrieved->endtime);
    }

    /**
     * Test tab switch alert type
     */
    public function test_tab_switch_alert()
    {
        global $DB;

        $this->resetAfterTest(true);

        $user = $this->getDataGenerator()->create_user();

        $session = new \stdClass();
        $session->userid = $user->id;
        $session->examname = 'Test Quiz';
        $session->starttime = time();
        $session->status = 'active';
        $session->timecreated = time();
        $session->timemodified = time();
        $sessionid = $DB->insert_record('local_myplugin_sessions', $session);

        // Create a tab_switch alert
        $alert = new \stdClass();
        $alert->sessionid = $sessionid;
        $alert->userid = $user->id;
        $alert->alerttype = 'tab_switch';
        $alert->description = 'TAB SWITCH DETECTED (1/3)';
        $alert->severity = 1;
        $alert->timecreated = time();

        $alertid = $DB->insert_record('local_myplugin_alerts', $alert);

        // Verify alert type
        $retrieved = $DB->get_record('local_myplugin_alerts', ['id' => $alertid]);
        $this->assertEquals('tab_switch', $retrieved->alerttype);
    }
}
```

This is the result from the php unit test:



```
Time: 00:17.962, Memory: 44.00 MB

OK (4 tests, 8 assertions)
```

From the above image, we can see that 4 tests were executed, 8 assertions were validated, and that all tests passed successfully. These tests ensure that the plugin's database operations for session management and alert logging are functioning as expected.

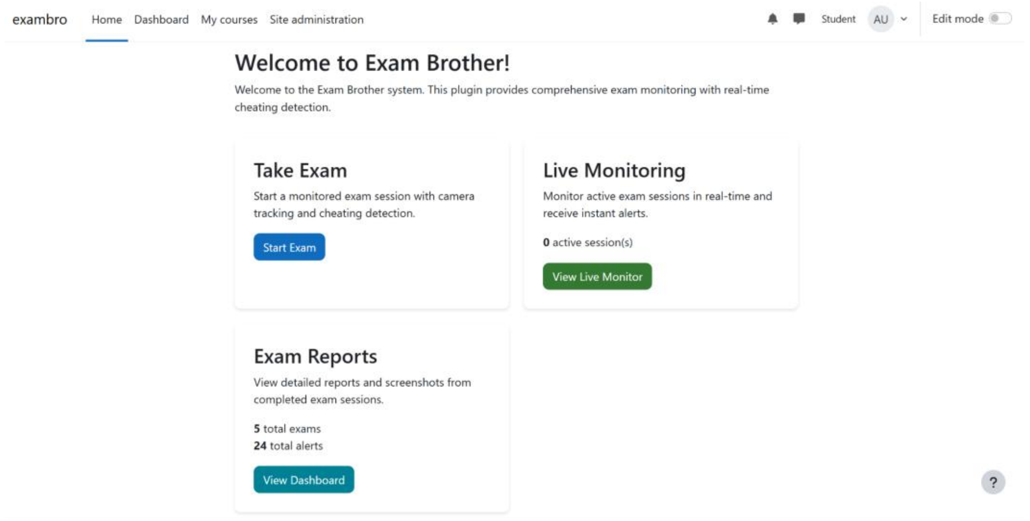
## 7.2 Integration Testing

- Role-agnostic Dashboard

  The current dashboard is unable to differentiate between proctors and students. As a result, students are able to access live monitoring features and exam reports. Additionally, they can access the proctor dashboard, which should be restricted to proctors only.

  Status : FAILED

  Severity : 1



Welcome to Exam Brother!

Welcome to the Exam Brother system. This plugin provides comprehensive exam monitoring with real-time cheating detection.

Take Exam

Start a monitored exam session with camera tracking and cheating detection.

Start Exam

Live Monitoring

Monitor active exam sessions in real-time and receive instant alerts.

**0** active session(s)

View Live Monitor

Exam Reports

View detailed reports and screenshots from completed exam sessions.

**5** total exams

**24** total alerts

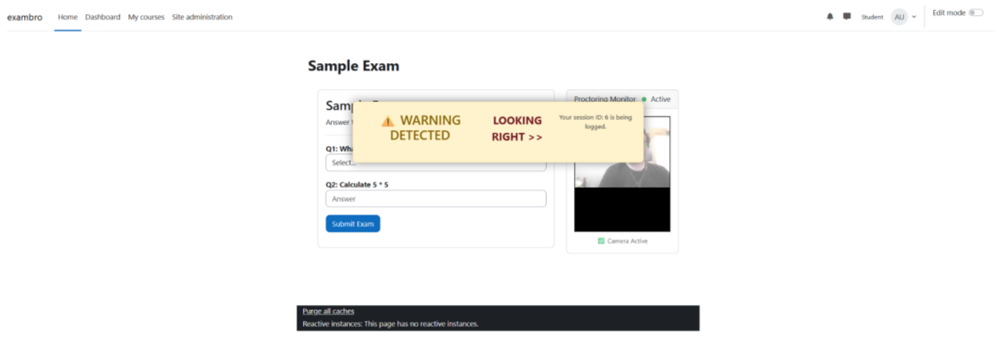
View Dashboard

- Permanent Alert

  When a student is flagged as suspicious, an alert appears on their screen to notify them that they have been suspected of cheating. Normally, this alert should disappear automatically, but in the current implementation it remains visible until the session ends.

  Status : FAILED

Severity : 2

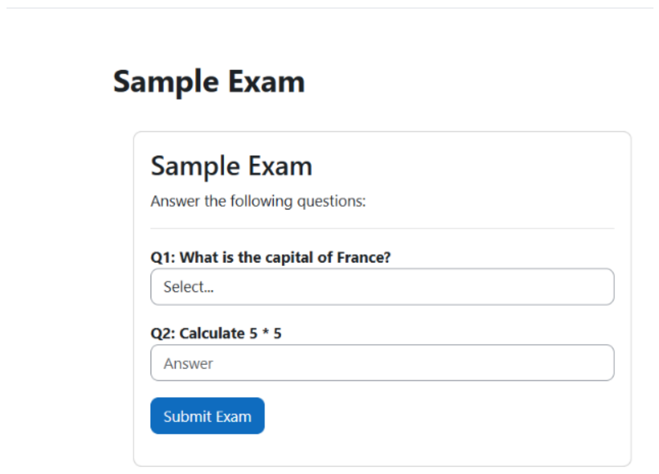


- Quiz Integration

  The questions inside the plugin's quiz are not yet integrated with Moodle's native quiz system. As a result, teachers cannot freely add, edit, or manage questions using Moodle's built-in quiz features. All question creation and management must be done manually inside the plugin, limiting flexibility and preventing seamless integration with Moodle's existing assessment tools.

  Status : FAILED

  Severity : 3



Sample Exam

Sample Exam

Answer the following questions:

Q1: What is the capital of France?

Select...

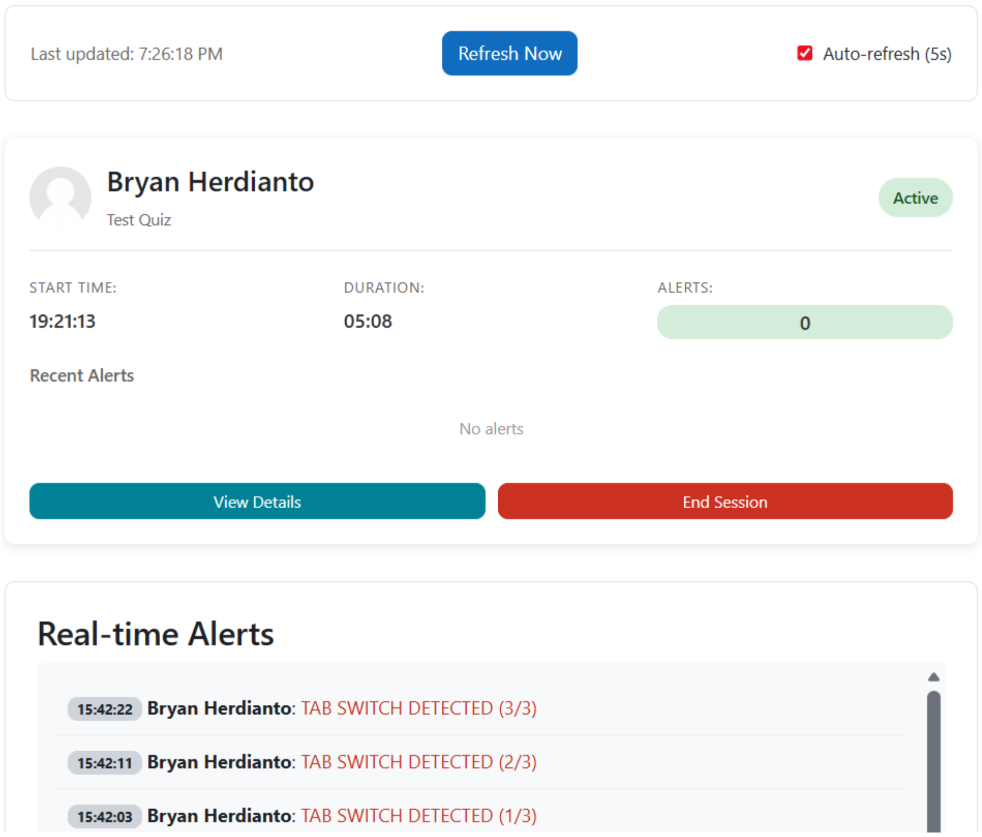
Q2: Calculate 5 * 5

Answer

- Real-time Alerts

The system is designed to provide real-time notifications when students exhibit suspicious behavior, such as looking left or right, or switching browser tabs during an exam session. During testing, these alert triggers were monitored across different scenarios to ensure consistent and accurate detection. The system successfully generated immediate alerts for each simulated behavior, and the proctor dashboard updated in real time without noticeable delay. Additionally, the "End Session" control functioned as expected, allowing the proctor to terminate a student's monitoring session instantly and preventing further data capture.

Status : SUCCESS



Last updated: 7:26:18 PM

Bryan Herdianto

Test Quiz

START TIME: 19:21:13

DURATION: 05:08

ALERTS: 0

Recent Alerts

No alerts

Real-time Alerts

15:42:22 **Bryan Herdianto**: TAB SWITCH DETECTED (3/3)

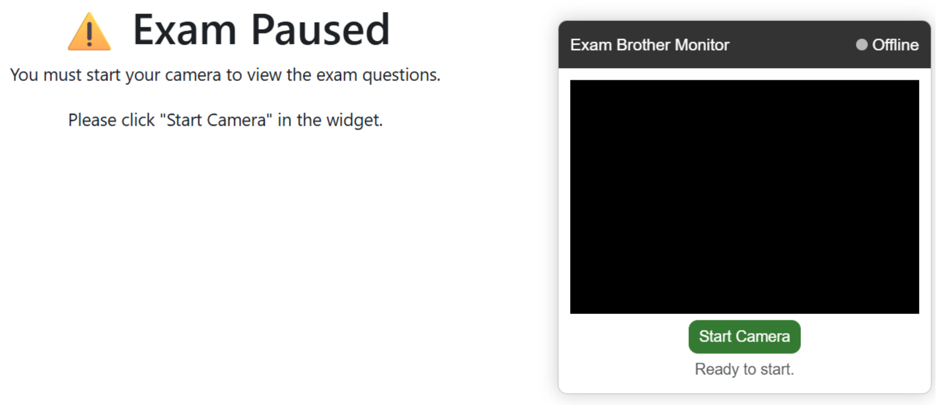
15:42:11 **Bryan Herdianto**: TAB SWITCH DETECTED (2/3)

15:42:03 **Bryan Herdianto**: TAB SWITCH DETECTED (1/3)

- Blocked Exam If No Camera

  The system is intended to prevent students from proceeding with an exam if no functional camera is detected. When the camera is unavailable, disabled, or deliberately obstructed, the system displays a white overlay that blocks access to the exam interface. During testing, multiple scenarios were simulated, like unplugged cameras, denied permissions, and software-level camera blocking. In all cases, the system correctly prevented the student from viewing or answering questions, ensuring that the exam could not be taken without the required monitoring setup. The overlay appeared instantly and remained active until proper camera access was restored.

Status : SUCCESS



Exam Paused

You must start your camera to view the exam questions.

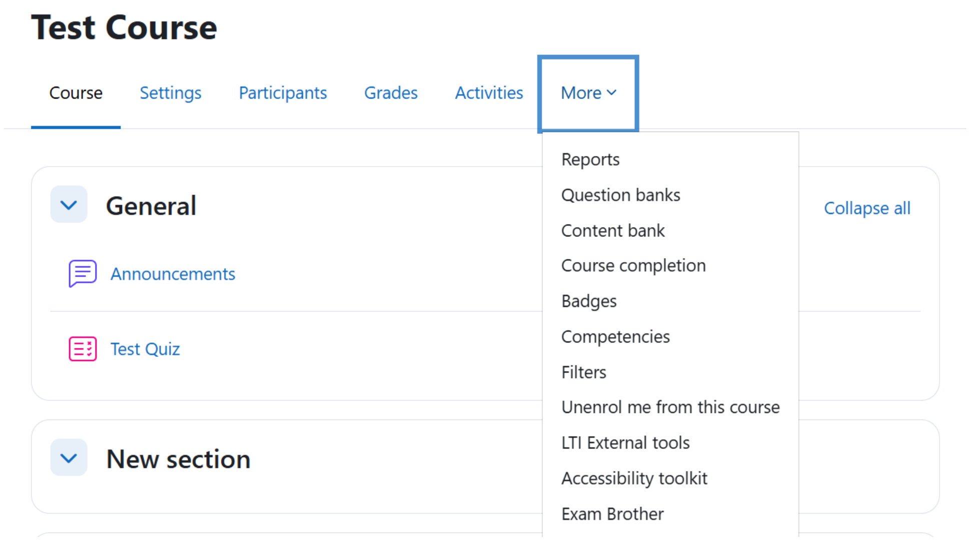
Please click "Start Camera" in the widget.

## 7.3 Regression Testing

- Plugin Shows For Teachers

  Exam Brother now appears directly in the Course navigation tabs within Moodle. This improvement makes it much easier for teachers and proctors to find and access the plugin's features without searching through multiple menus. With this integration, proctors can quickly open the monitoring dashboard or prepare proctored quizzes, resulting in a smoother workflow during exam sessions.

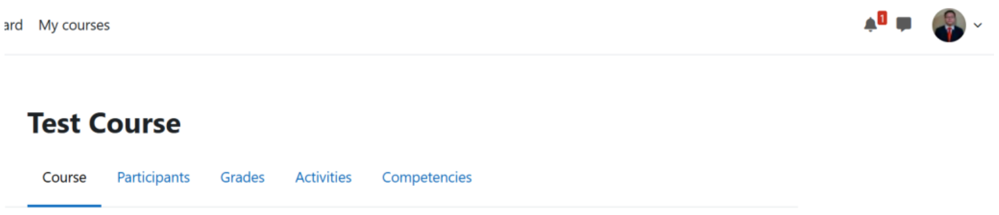
  Status : SUCCESS



- Proctor Dashboard Hidden From Students

  The system correctly restricts access so that only users with proctor or teacher roles can view live monitoring and student reports. From the student's point of view, these pages do not exist, ensuring privacy and preventing unauthorized access to sensitive exam information. Students are only presented with the quiz interface that is directly linked to Exam Brother.

  Status : SUCCESS

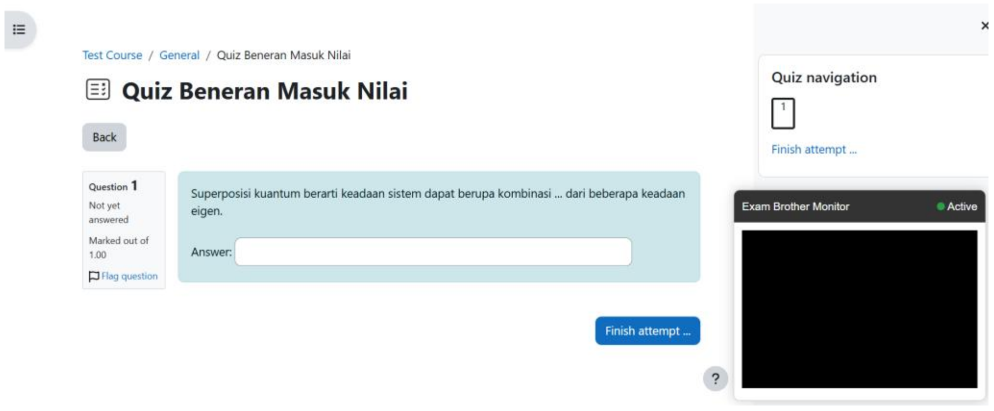


- Moodle Quiz Integration

  Exam Brother is now fully connected with the Moodle quiz environment, allowing students to take their exams while keeping their camera active for monitoring. During the exam, any suspicious actions are detected in real time and sent as alerts both to the student and to the proctor watching the session.

This integration creates a more secure examination flow where misconduct can be identified and acted on immediately.

Status : SUCCESS



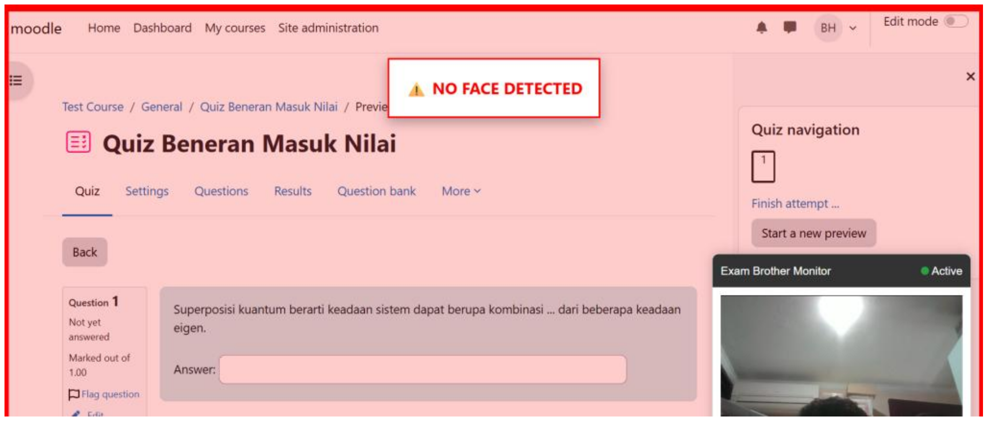
Superposisi kuantum berarti keadaan sistem dapat berupa kombinasi ... dari beberapa keadaan eigen.

Answer:

- Red Short Alert

  Alerts shown to students have been redesigned for better visibility and impact. The previous version displayed small, static alerts that were easy to miss. The new version expands the alert into a large red notification covering the entire screen for a short period of time, ensuring that students clearly understand the warning and cannot ignore it. This improves the effectiveness of the alert system during exams.

  Status : SUCCESS



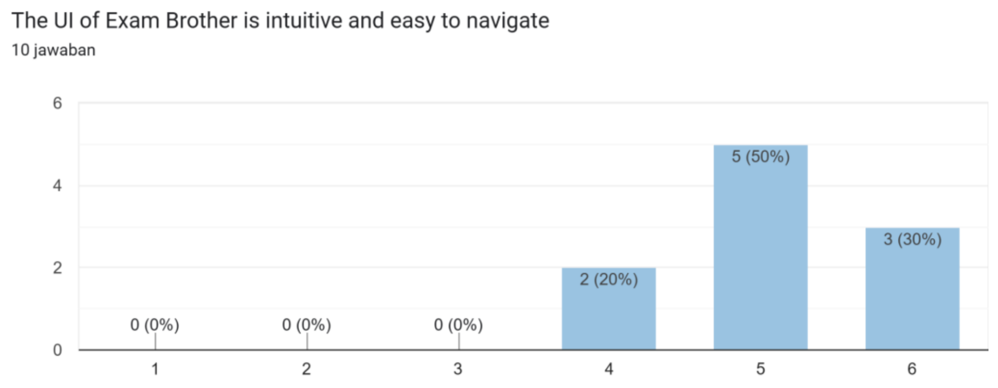
Superposisi kuantum berarti keadaan sistem dapat berupa kombinasi ... dari beberapa keadaan eigen.

Answer:

## 7.4 User Acceptance Testing

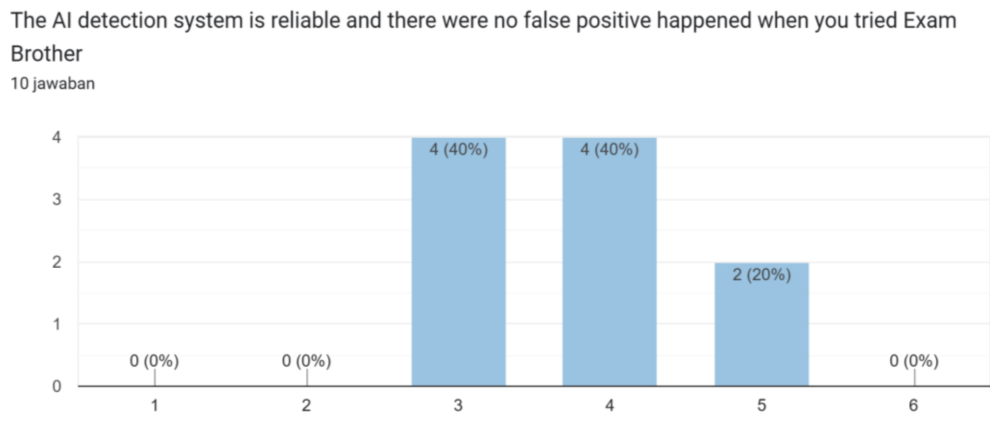
For this user acceptance testing, we asked several students to try a Moodle quiz that had been integrated with the Exam Brother plugin. After completing the quiz, they were asked to fill out a short survey. The survey contained two types of questions: six rating questions on a scale from 1 to 6 (1 means strongly disagree and 6 means strongly agree), and one short-answer question.

- Intuitiveness of UI (User Interface)



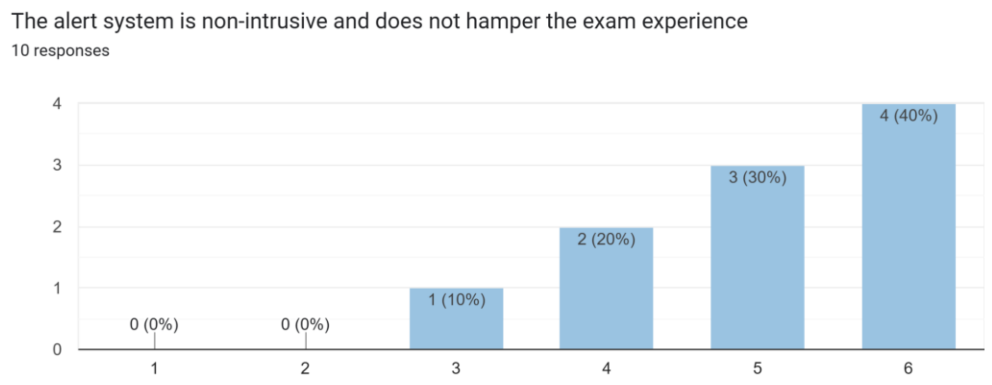
Average: 5.10/6

- Reliability of AI surveillance



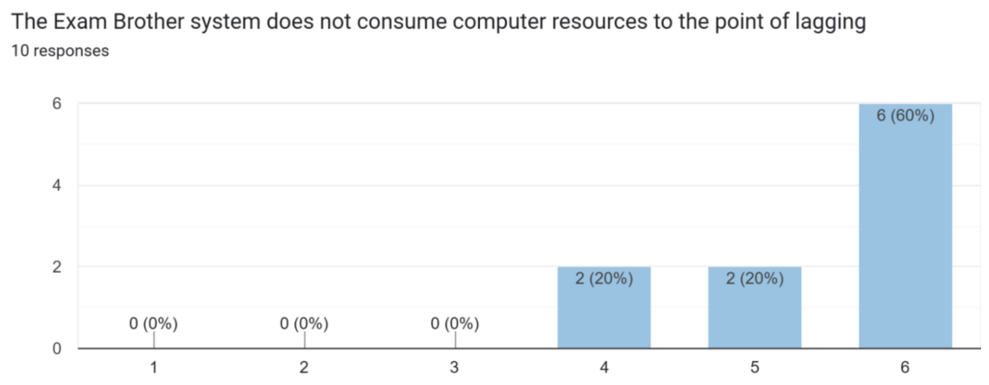
Average: 3.80/6

- Intrusiveness of the alert system



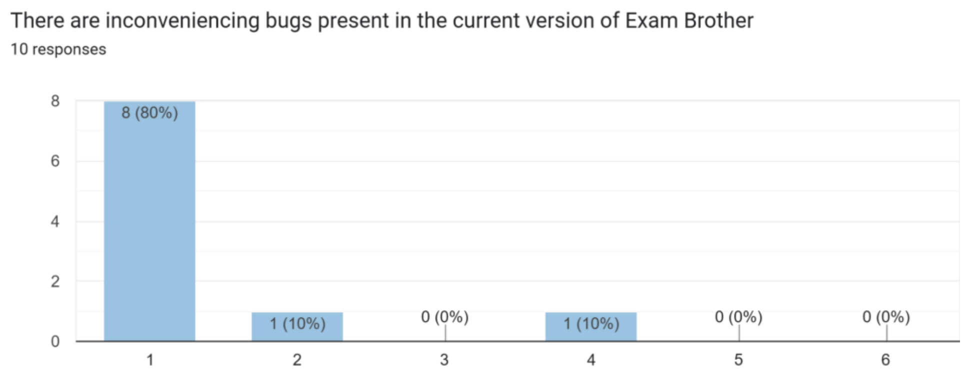
Average: 5.00/6

- Resource Usage



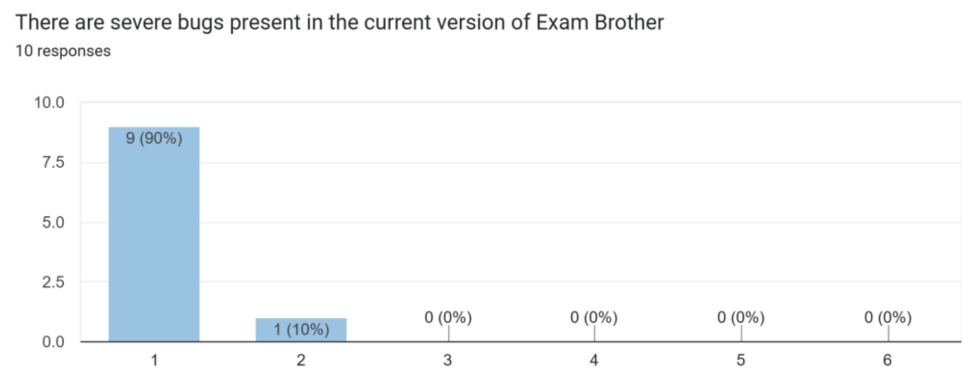
Average: 5.40/6

- Amount of Inconveniencing Bugs



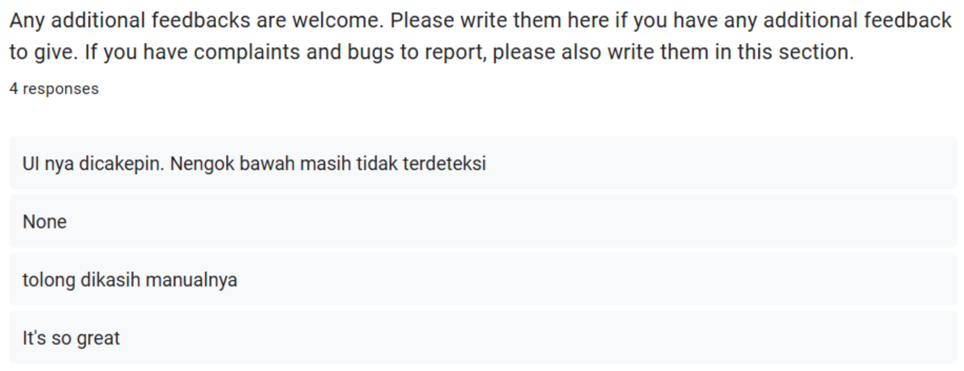
Average: 1.40/6

- Amount of Severe Bugs



Average: 1.10/6

- Additional Feedbacks


Any additional feedbacks are welcome. Please write them here if you have any additional feedback to give. If you have complaints and bugs to report, please also write them in this section.

4 responses

UI nya dicakepin. Nengok bawah masih tidak terdeteksi

None

tolong dikasih manualnya

It's so great

## 7.5 Conclusion

Based on all the testing activities conducted, we can conclude that the Exam Brother plugin functions reliably as both a standalone module and as an integrated component within Moodle. Each feature was verified to operate correctly, and improvements were made wherever enhancements were identified. Bugs discovered during earlier stages were resolved, and regression testing confirmed that these fixes did not introduce new issues.

Feedback from user acceptance testing shows that the user interface is smooth, intuitive, and easy to navigate. Test participants also highlighted that the AI-powered monitoring runs efficiently with minimal performance impact. However, occasional false positives were still observed in the AI’s detection system. While these do not hinder the overall functionality, they represent an important area for future refinement.

Overall, the testing results indicate that the system is stable, functional, and ready for real-world use, with clear opportunities for ongoing improvement in AI accuracy.

# References

[1] Berkhout, E. et al. 2020. From Cheating to Learning: An Evaluation of Fraud Prevention on National Exams in Indonesia. RISE Working Paper Series. 20/046. https://doi.org/10.35489/BSG-RISE-WP_2020/046

[2] J. Post, "State university entrance exam organizers vow to stamp out mass cheating," *The Jakarta Post*, May 28, 2025. [Online]. Available: https://www.thejakartapost.com/indonesia/2025/05/28/state-university-entrance-exam-organizers-vow-to-stamp-out-mass-cheating.html

[3] "Safe exam browser - news," *SEB*. https://safeexambrowser.org/news_en.html

[4] "Testportal: Proctored testing. A necessity or an overused hype? | Guides," *Testportal*. https://www.testportal.net/en/guides/online-test-cheating/online-proctoring/

[5] I. N. Yulita, F. A. Hariz, I. Suryana, and A. S. Prabuwono, "Educational Innovation Faced with COVID-19: Deep Learning for Online Exam Cheating Detection," *Education Sciences*, vol. 13, no. 2, p. 194, Feb. 2023, doi: 10.3390/educsci13020194.

[6] P. M. Newton and K. Essex, "How Common is Cheating in Online Exams and did it Increase During the COVID-19 Pandemic? A Systematic Review," *Journal of Academic Ethics*, vol. 22, no. 2, pp. 323–343, Aug. 2023, doi: 10.1007/s10805-023-09485-5.

[7] "Online Exam Proctoring Market Outlook 2025-2032," *IntelMarketResearch*. https://www.intelmarketresearch.com/online-exam-proctoring-2025-2032-270-4059

[8] "MoodleDocs." https://docs.moodle.org/500/en/Main_page

[9] "MediaPipe Solutions guide," Google AI for Developers. https://ai.google.dev/edge/mediapipe/solutions/guide

[10] Docker Documentation. Docker Overview. https://docs.docker.com/get-started/overview/

[11] Figma About. About Figma. https://www.figma.com/about/

[12] Microsoft. Get Started with Visual Studio Code. https://code.visualstudio.com/learn

[13] "Welcome to Python.org," Python.org. https://www.python.org/doc/

[14] Github, Inc. Overview. https://docs.github.com/en/desktop/overview

[15] The PostgreSQL Global Development Group. What is PostgreSQL? https://www.postgresql.org/about/

[16] Trello. Trello 101: How to Use Trello Boards & Cards. https://trello.com/guide/trello-101