Exam Monitoring Platform

JOSÉ EDUARDO LEITÃO JORGE
(Licenciado)

Dissertação para obtenção do Grau de Mestre em Engenharia Informática e de Computadores

Orientador:
Doutor Fernando Miguel Gamboa de Carvalho

Júri:
Presidente: Doutor Nuno Miguel Soares Datia
Vogais:
Doutor Filipe Bastos de Freitas
Doutor Fernando Miguel Gamboa de Carvalho

Novembro de 2025

Exam Monitoring Platform

JOSÉ EDUARDO LEITÃO JORGE
(Licenciado)

Dissertação para obtenção do Grau de Mestre em Engenharia Informática e de Computadores

Orientador:
Doutor Fernando Miguel Gamboa de Carvalho, IPL/ISEL

Júri:
Presidente: Doutor Nuno Miguel Soares Datia, IPL/ISEL
Vogais:
Doutor Filipe Bastos de Freitas, IPL/ISEL
Doutor Fernando Miguel Gamboa de Carvalho, IPL/ISEL

Novembro de 2025

Acknowledgements

This project would not have been possible without the collaboration and support of several individuals. I would like to express my gratitude to all those who, directly or indirectly, contributed to its completion.

First, I would like to thank my supervisor, Fernando Miguel Gamboa de Carvalho, for his guidance and support throughout this process. His patience, knowledge, and advice were essential to the successful completion of this project. I am grateful for our meetings and for his direction, especially during the most challenging moments.

To my family, especially my mother Luísa, I extend my thanks for their constant support during this journey. Their presence and encouragement were vital for me to complete this work.

I also thank my friends for their support, advice, and for being available to discuss ideas and clarify doubts throughout this project. In particular, I am immensely grateful to Gonçalo Fernandes, Pedro Apolinário, and António Goulão for their help in testing the plugin and assisting in finding its flaws.

Finally, thank you to everyone who contributed to making this master's project a reality.

Statement of integrity

I declare that this project work is the result of my personal and independent research. Its content is original, and all sources listed in the bibliographic references were consulted and are duly mentioned in the text. I further declare that all scientific and technical references relevant to the development of the work are duly cited and included in the bibliographic references.

The author

Assinado por: **José Eduardo Leitão Jorge**
Num. de Identificação: 15747605
Data: 2025.11.27 19:54:38+00'00'

Lisbon, 27 November, 2025

Abstract

The increase in fraud in in-person exams, facilitated by Artificial Intelligence (AI) tools, represents a significant challenge to academic integrity. At the *Instituto Superior de Engenharia de Lisboa* (ISEL), as in other universities and schools, this problem has intensified in recent years.

In response, this master's thesis proposes the development of a secure and efficient web solution, the Exam Management Platform (EMP) and the Student Exam Plugin (SEP). The EMP is a web application for teachers, which allows for the management of exams and real time supervision. The SEP is an extension for Google Chrome, installed on the student's computer, that monitors their actions during the exam.

The solution uses students' laptops and the Google Chrome browser, eliminating the need to install third-party software and reducing infrastructure costs. Its functionalities include user authentication, real-time monitoring of activities such as access to unauthorised Uniform Resource Locators (URLs), changes in window focus, browser resizing, right-clicks, and text selection. It also includes a notification system to alert supervisors to irregularities. All actions are logged for post-exam analysis.

The system's architecture integrates front-end and back-end technologies, communicating via RESTful Application Programming Interfaces (APIs) and Server-Sent Events (SSE) for real-time updates. Security is reinforced by Hypertext Transfer Protocol Secure (HTTPS), Two-Factor Authentication (2FA), authorisation tokens, and hash validation to detect plugin tampering.

Tests, including scenarios in a controlled environment, demonstrated the system's effectiveness in detecting and logging events, ensuring bidirectional communication, and verifying the plugin's integrity. Although limited to Google Chrome and dependent on network connectivity, the system constitutes a practical response to ensure the integrity of in-person assessments.

Keywords

Online Examinations; Academic Integrity; Monitoring Platform; Browser Extension; Real-Time Monitoring.

Resumo

O aumento de fraudes em exames presenciais, facilitado por ferramentas de Inteligência Artificial (AI), representa um desafio significativo para a integridade académica. No Instituto Superior de Engenharia de Lisboa (ISEL), tal como noutras faculdades e escolas, este problema tem-se intensificado nos últimos anos.

Em resposta, esta tese de mestrado propõe o desenvolvimento de uma solução *web* segura e eficiente, a *Exam Management Platform* (EMP) e o *Student Exam Plugin* (SEP). A EMP é uma aplicação *web* para professores, que permite a gestão de exames e a supervisão em tempo real. O SEP é uma extensão para o *Google Chrome*, instalada no computador do aluno, que monitora as suas ações durante o exame.

A solução recorre aos computadores portáteis dos alunos e ao navegador *Google Chrome*, eliminando a necessidade de instalar *software* de terceiros e reduzindo os custos de infraestrutura. Entre as funcionalidades estão a autenticação de utilizadores, a monitorização em tempo real de atividades como acesso a URLs não autorizados, mudanças de foco da janela, redimensionamento do navegador, cliques com o botão direito e seleção de texto. Inclui ainda um sistema de notificações para alertar os supervisores para irregularidades. Todas as ações são registadas para análise posterior ao exame.

A arquitetura do sistema integra tecnologias de *front-end* e *back-end*, comunicando através de *APIs RESTful* e *Server-Sent Events* (SSE) para atualizações em tempo real. A segurança é reforçada por *Hypertext Transfer Protocol Secure* (HTTPS), *Two-Factor Authentication* (2FA), *tokens* de autorização e validação por *hash* para detetar adulterações no *plugin*.

Os testes, incluindo cenários num ambiente controlado, demonstraram a eficácia do sistema na deteção e registo de eventos, garantindo a comunicação bidirecional e verificando a integridade do *plugin*. Embora limitada ao *Google Chrome* e dependente da conectividade de rede, o sistema constitui uma resposta prática para garatir a integridade de avaliações presenciais.

Palavras chave

Exames Online; Integridade Académica; Plataforma de Monitorização; Extensão de Navegador; Monitoramento em Tempo Real.

Contents

List of Figures

List of Tables

Acronyms

2FA Two-Factor Authentication

ACID Atomicity, Consistency, Isolation, Durability

ADL Advanced Distributed Learning

AI Artificial Intelligence

API Application Programming Interface

AU Assignable Unit

AWS Amazon Web Services

BYOD Bring Your Own Device

CAM Content Aggregation Model

cmi5 Computer Managed Instruction 5

CPU Central Processing Unit

CSP Content Security Policy

CSRF Cross-Site Request Forgery

CSS Cascading Style Sheets

DAL Data Access Layer

DoD Department of Defense

DOM Document Object Model

EMP Exam Management Platform

ER Entity-Relationship

GDPR General Data Protection Regulation

HBS Handlebars

HTML Hypertext Markup Language

HTTP HyperText Transfer Protocol

HTTPS Hypertext Transfer Protocol Secure

ISEL Instituto Superior de Engenharia de Lisboa

I/O Input/Output

JSON JavaScript Object Notation

JWT JSON Web Token

LMS Learning Management System

LRS Learning Record Store

NoSQL Not Only SQL

OAuth Open Authorization

ODM Object-Document Mapper

ORM Object-Relational Mapping

OTP One-Time Password

PDF Portable Document Format

RBAC Role-Based Access Control

Redis REmote DIctionary Server

RTE Run-Time Environment

SCORM Sharable Content Object Reference Model

SCO Sharable Content Object

SEB Safe Exam Browser

SEP Student Exam Plugin

SIM Subscriber Identity Module

SMS Short Message Service

SN Sequencing and Navigation

SQL Structured Query Language

SQLi Structured Query Language injection

SSE Server-Sent Events

TCP Transmission Control Protocol

TLS Transport Layer Security

UI User Interface

U.S. United States

URL Uniform Resource Locator

VM Virtual Machine

xAPI Experience API

XML Extensible Markup Language

XSS Cross-Site Scripting

Chapter 1

Introduction

This chapter lays the foundation for the development of a monitoring platform designed to manage and supervise computer-based exams in physical exam environments. It starts by exploring the context and motivation behind this work, outlines the objectives the thesis aims to accomplish, and concludes with a summary of the document's structure. The deployed version of the Exam Management Platform (EMP) is hosted at `https://exam-monitoring-platform.onrender.com/`, and the Student Exam Plugin (SEP) can be found at `https://chromewebstore.google.com/detail/exam-monitoring-plugin/hngoklcnpkekadcackclnlkocibaijil`. The source code for the project is available in the GitHub repository at `https://github.com/isel-sw-projects/exam.git`.

Section 1.1 explores the context and the motivation driving this work. It examines the challenges associated with online examinations, the evolving needs of educational institutions, and the role of technology in addressing these issues. By offering this background, the section establishes the basis for understanding the necessity and significance of the proposed monitoring platform.

Section 1.2 defines the objectives of this work, aligning them with the challenges identified in the previous section.

Section 1.3 describes the approach taken to design, implement, and evaluate the system.

Finally, Section 1.4 concludes by presenting an overview of the document's structure. This section serves as a roadmap, outlining the organisation of the thesis and summarising the content of each chapter to guide the reader.

1.1 Context and motivation

In recent years, the need for robust examination monitoring platforms has grown significantly. Institutions worldwide are increasingly relying on digital tools to administer examinations, particularly after the COVID-19 pandemic highlighted the importance of digital transformation in education and accelerated the widespread adoption of digital examination solutions [24]. While the pandemic primarily drove the development of remote examinations conducted from home or outside institutional premises, online examinations offer flexibility and accessibility but also introduce challenges in maintaining academic integrity and ensuring the authenticity of participants.

In contrast, virtual examinations are conducted on-site within institutional facilities, using either computers provided by the institution or students' own personal laptops. This approach combines the security and supervision of traditional in-person examinations with the benefits of digital assessment, including automated submissions and the potential for exams to be graded automatically.

This project focuses specifically on such virtual examinations held in physical rooms, where students and teachers are present on-site. Rather than addressing remote online solutions, this master thesis develops a monitoring platform for computer-based examinations conducted within controlled examination environments, utilising students' laptops while maintaining the security and supervision advantages of conventional in-person examination settings.

1.1.1 Scope of the Problem

The management of examinations presents several challenges across different educational institutions and sectors. Effective exam administration must address concerns related to security, accessibility, and cost-efficiency.

From an institutional perspective, dedicated examination rooms, typically equipped with computers for each student, have traditionally been used to ensure exam integrity. These rooms provide a controlled environment, reducing the probability of cheating. However, they come with significant costs, including infrastructure maintenance and technological investments.

Authentication remains a critical issue in exam management. Traditional methods such as passwords are susceptible to security vulnerabilities, including sharing and breaches. To mitigate these risks, institutions have explored biometric authentication solutions such as fingerprint and facial recognition [31]. While effective, these solutions introduce concerns related to privacy, implementation costs, and user approval.

With the rise of virtual learning, digital solutions for exam management have become more prevalent. Various approaches have been implemented, including Artificial Intelligence (AI)-based proctoring systems, browser lockdown mechanisms, and remote monitoring solutions. These solutions are particularly used in remote or distance examinations, such as those conducted during the COVID-19 pandemic, and aim to strike a balance between ensuring exam security and maintaining a seamless user experience [3, 48].

Overall, the challenge is to develop a secure and scalable computer-based examination platform specifically designed for supervised, on-site examinations. The platform must ensure authentication and exam integrity while using students' own laptops within a physical examination room at the institution.

1.2 Objectives

This project has been designed to create a secure and efficient examination environment through a web-based solution, focusing on management, monitoring, and enforcement of exam policies. The primary and secondary objectives of the project are outlined below.

1.2.1 Primary Objectives

The primary objectives aim to establish a functional and secure exam management system with the following goals:

- Secure Examination Process: Ensure a secure and controlled examination environment, in physical exam rooms, where student actions are monitored in real-time, and unauthorized behaviour is promptly flagged;
- Efficient Exam Supervision: Provide a platform for supervisors and teachers to monitor student activity and address potential issues in real time;
- User-friendly Exam Management: Create an intuitive interface for teachers to set up and manage exams, configure virtual exam rooms, and handle student and supervisor lists.

1.2.2 Secondary Objectives

The following secondary objectives support the broader functionality of the system, enhancing integration and security:

- Scalability Testing: Conduct load and scalability tests to simulate a large number of students using the plugin simultaneously, ensuring the platform can handle the volume of logs without compromising performance or responsiveness.

These objectives aim to provide a reliable system that enhances the integrity and efficiency of the examination process.

1.3 Approach

To achieve the objectives outlined in this project, a web-based application was developed, using existing technologies to ensure ease of use. The proposed solution aims to minimize the need for installing third-party software by making use of available tools.

The approach involves using students' personal laptops as the primary examination device, reducing infrastructure costs and eliminating the need for dedicated examination rooms. By using web technologies, particularly Google Chrome, the platform can provide seamless monitoring capabilities with minimal system requirements.

The examination management platform is accessible to teachers through a web interface, allowing them to create, monitor, and evaluate exams in real time. The platform includes features such as authentication, real-time activity tracking, and notifications to supervisors in case of suspicious activity.

By focusing on a browser-based solution, the approach ensures cross-platform compatibility and reduces dependency on specialized hardware or software installations. This approach makes the examination management system more accessible to various educational institutions.

1.4 Document structure

The remainder of this document is structured as follows:

- Chapter 2: State-Of-The-Art - This chapter reviews existing technologies and methodologies relevant to virtual exam monitoring. It provides insights into the strengths and limitations of different approaches, helping to position the proposed platform within the current landscape;
- Chapter 3: Related work - This chapter explores research on online (remote) and virtual (classroom-based) examination security and monitoring, compares different solutions, and identifies gaps that this project seeks to address. It also examines topics such as virtualized exams, AI-driven proctoring, and legal and ethical considerations;
- Chapter 4: Developed Work - This chapter presents the architecture, technology stack, and implementation of the Exam Management Platform (EMP) and the Student Exam Plugin (SEP). It details the proposed method and software tools used, as well as the security measures and implementation challenges that were addressed;
- Chapter 5: EMP and SEP Usage and Workflow - This chapter describes the practical use and workflow of the platform, including account creation, exam preparation, student interaction, and real-time monitoring;
- Chapter 6: Testing and Evaluation - This chapter explains the methodology used for testing the system's effectiveness and presents the results of unit testing, as well as tests

for the EMP and SEP. It also includes an analysis of results from controlled environment testing;

- Chapter 7: Conclusion - This final chapter provides an overall view of the problem and the developed solution, summarising the achieved results and outlining directions for future work.

Chapter 2

State-Of-The-Art

This chapter explores the current state of technologies, methodologies and tools relevant to the development of a monitoring platform in a web context. It aims to provide a comprehensive understanding of the existing solutions and their limitations, helping to position the proposed platform within the current landscape.

Section 2.1 covers the core aspects of back-end environments, including their capacity for scale, security, maintainability, and integration with different database types. It presents an overview of widely used frameworks and database management systems relevant to monitoring applications.

Section 2.2 introduces different frameworks for web development, focusing on their efficiency, security features, and integration capabilities.

Section 2.3 describes the role of browser extensions in monitoring activity. It discusses their benefits, limitations, and security considerations.

Section 2.4 examines various authentication methods, including Two-Factor Authentication (2FA), One-Time Password (OTP), and Biometric Identification Methods, assessing their advantages and potential challenges.

Section 2.5 reviews real-time communication technologies, comparing Polling, WebSockets, and Server-Sent Events (SSE) to determine the most suitable approach for this project.

Lastly, Section 2.6 describes standards for e-learning platform integration, including Sharable Content Object Reference Model (SCORM), Experience API (xAPI), and Computer Managed Instruction 5 (cmi5). It considers their capabilities for activity tracking and interoperability in learning management systems, especially in the context of monitoring browser-based activity.

2.1 Back-end Environment

The back-end environment of a web application is responsible for handling data processing, authentication, and communication with the frontend. It serves as the backbone of the application, ensuring efficient data handling and logic execution. Back-end environments typically consist of programming languages, databases, frameworks, and hosting services that work together to support the overall functionality of the application.

One of the primary concerns when choosing a back-end environment is scalability. The system must be capable of handling increasing numbers of users and data without compromising performance. This is particularly important for online (remote) and virtual (classroom-based) examination platforms where multiple students access the system simultaneously. Node.js has emerged as a popular choice due to its non-blocking, event-driven architecture, which ensures optimal performance under heavy loads [57].

Security is another critical aspect of back-end environments. Applications must implement robust authentication and authorization mechanisms to prevent unauthorized access to sensitive data. Common security measures include encryption, multi-factor authentication, and Role-Based Access Control (RBAC), all of which help protect against data breaches [2].

In addition to performance and security, maintainability is a crucial factor. A well-structured back-end environment allows developers to easily update and extend functionalities without causing interruptions. Using modular architectures and following best coding practices enhances maintainability, making it easier to incorporate new features or fix issues when they arise [57].

Finally, hosting solutions are a vital component of back-end environments. Cloud-based services such as Amazon Web Services (AWS), Azure, and Google Cloud offer scalable and reliable hosting options, reducing the need for on-premises infrastructure. These services provide automated backups, security compliance, and high availability, ensuring the application is always accessible [8].

2.1.1 Overview of Popular Back-end Environments

Several back-end environments are used in modern web development, each offering unique features and capabilities. Some of the most popular back-end environments include [57]:

- Node.js (JavaScript): A runtime environment that allows the execution of JavaScript on the server side, known for its non-blocking, event-driven architecture;
- Django (Python): A high-level framework that follows the "batteries-included" philosophy, offering built-in features for rapid development;
- Spring Boot (Java): A framework that simplifies the development of deployable applications with built-in dependencies and configuration management;
- Ruby on Rails (Ruby): Focuses on convention over configuration, enabling developers to build applications quickly with less boilerplate code.

2.1.2 Comparison of Back-end Environments

Each back-end environment has its own advantages and disadvantages depending on the application requirements. Below is a comparison between Node.js and other popular environments [57]:

Table 2.1 Comparison between back-end environments.

Environment	Scalable	Security	Enterprise	Performance	Approach
Node.js	✓	-	-	✓	Configuration
Django	✓	✓	✓	-	Convention
Spring Boot	✓	✓	✓	✓	Annotations
Ruby on Rails	✓	✓	-	-	Convention

The Table 2.1 highlights the key characteristics of scalability, security, enterprise suitability, performance, and configuration approach for four modern back-end environments. The data allows for an analysis of each technology's strengths and limitations, helping to determine the most suitable choice for different development needs.

In terms of scalability, all the analysed environments (Node.js, Django, Spring Boot, and Ruby on Rails) support distributed architectures, load balancing, and modern cloud computing technologies, making them well-suited for scalable applications.

Regarding security, Django, Spring Boot, and Ruby on Rails offer built-in protection against common security threats such as Structured Query Language injection (SQLi), Cross-Site Scripting (XSS), and Cross-Site Request Forgery (CSRF). Django, in particular, is designed with security in mind, enforcing best practices by default. Node.js, however, does not provide built-in security mechanisms and relies on external libraries and developer implementation to secure applications, making it more prone to vulnerabilities if not properly configured.

From an enterprise perspective, Django and Spring Boot stand out as the most robust frameworks, offering best development practices, strong security, and integration with large corporate systems. Ruby on Rails, while secure and developer-friendly, is less commonly used in enterprise environments due to performance limitations at scale. Node.js, while highly scalable and performant, lacks the built-in enterprise-level security and structure required for large-scale regulated applications.

When it comes to performance, Node.js and Spring Boot stand out in handling high concurrency and low-latency applications, making them ideal for real-time and data-intensive environments. Django and Ruby on Rails, while efficient, are traditionally more suited to applications that prioritise security and development speed over extreme performance optimisation.

The configuration approach varies between the frameworks. Django and Ruby on Rails follow the "Convention over Configuration" principle, reducing the need for manual configurations and simplifying development. Spring Boot utilises annotations, balancing flexibility with structured configuration. Node.js relies on manual configuration, providing developers with

maximum control but requiring more setup effort.

The choice of back-end environment depends on the project's requirements. For enterprise-level applications with strong security needs, Django and Spring Boot are the best options. Ruby on Rails is well-suited for startups and projects that prioritise rapid development over enterprise-scale performance. Node.js is ideal for applications requiring high concurrency, efficient Application Programming Interface (API)s, and flexible configuration, though additional security measures must be implemented.

If the priority is developer productivity and predefined conventions, Django and Ruby on Rails provide a streamlined experience. If a high degree of customisation is required, Node.js allows greater flexibility. For projects requiring both enterprise robustness and high performance, Spring Boot stands out as the strongest choice.

Each technology has its strengths and weaknesses, and the best choice depends on balancing system requirements, team expertise, and business needs.

2.1.3 Security Considerations in Back-end Development

Security is a fundamental aspect of back-end development, especially in applications that manage sensitive data. Common security threats include SQLi, XSS, and CSRF [29]. To mitigate these risks, back-end environments provide various security features such as:

- Authentication and Authorization: Using JSON Web Token (JWT) or Open Authorization (OAuth) for secure user verification;
- Data Encryption: Protecting data at rest and in transit using encryption protocols like Transport Layer Security (TLS);
- Input Validation: Preventing injection attacks by cleaning user inputs;
- Security Headers: Adding HyperText Transfer Protocol (HTTP) headers such as Content Security Policy (CSP) to protect against attacks.

Frameworks like Django, Spring Boot and Ruby on Rails offer built-in security mechanisms, while Express.js (JavaScript/Node.js) is minimalistic and does not include built-in security mechanisms. Instead, developers typically use middleware like Helmet.js to set security-related HTTP headers and express-rate-limit for rate limiting.

2.1.4 Database Technologies

Databases play a vital role in storing and managing application data. The two main types are relational databases, which use Structured Query Language (SQL) and non-relational, known as Not Only SQL (NoSQL), each designed to meet different application requirements [9].

Relational databases, such as MySQL and PostgreSQL, store data in structured tables with predefined schemas. They support Atomicity, Consistency, Isolation, Durability (ACID) properties, making them ideal for applications requiring complex queries and relationships. The advantages of relational databases include strong data integrity and consistency, a standardised query language, and suitability for transactional applications. However, they have scalability limitations and require predefined schemas [9, 42].

NoSQL databases, such as MongoDB and Cassandra, store data in flexible formats such as key-value pairs or documents. They are designed for scalability and can handle large volumes of unstructured data. The advantages of NoSQL databases include high scalability and flexibility, a schema-less design for dynamic data structures, and faster read/write operations. However, they often follow an eventual consistency model rather than immediate consistency, and they lack standardization compared to SQL databases [9, 42].

One of the critical aspects of choosing a database is how easily it integrates with backend applications. Many programming environments provide libraries and frameworks that simplify database connectivity and management.

For Node.js, there are various packages that facilitate integration with both SQL and NoSQL databases:

- SQL Databases: Libraries such as Sequelize (for MySQL, PostgreSQL, and SQLite) and Knex.js (a SQL query builder) provide an abstraction layer to simplify queries and database management [44];
- NoSQL Databases: Tools like Mongoose for MongoDB offer an Object-Document Mapper (ODM) approach, allowing developers to define schemas while still leveraging the flexibility of NoSQL [23];
- Other Databases: For REmote DIctionary Server (Redis), there are two main options: ioredis and node-redis. ioredis offers features like clustering support, built-in Sentinel, and efficient Input/Output (I/O) handling, making it suitable for more complex use cases. Redis Sentinel is a system designed to provide high availability for Redis. It monitors Redis instances, detects failures, and performs automatic failover when necessary. Sentinel ensures that if the master node fails, a replica is promoted to master, and clients are notified of the change. On the other hand, node-redis, the official Redis client for Node.js, provides a simpler API but requires additional configuration for certain features [45].

The availability of well-supported libraries and Object-Relational Mapping (ORM) tools significantly influences database selection, as they streamline development, improve maintainability, and reduce the complexity of writing raw queries.

However, choosing a database depends on not just its technical capabilities but also how efficiently it can be integrated into the application's architecture.

2.2 Back-end Web Framework

Back-end web frameworks provide a structured foundation for building server-side applications, offering tools and libraries that streamline development processes. These frameworks help developers manage routing, middleware, and request handling efficiently.

Express.js is one of the most popular back-end web frameworks for Node.js. It provides a set of features for building web applications and APIs. Express simplifies routing, integrates seamlessly with middleware, and supports templating engines for dynamic content rendering. Its lightweight nature and extensive community support make it an ideal choice for developers [50].

One of the primary advantages of using Express.js is its middleware architecture. Middleware functions allow developers to process incoming requests, add logging, handle authentication, and modify responses before they reach the client. This approach enhances code reusability and maintainability, making it easier to extend the application [50].

Express.js also integrates well with various database solutions, allowing developers to use relational or NoSQL databases depending on the project's needs. Popular database integration options include Sequelize for SQL databases and Mongoose for MongoDB. These libraries simplify database interactions, enabling efficient data retrieval and manipulation [23, 44].

Security is a crucial consideration when using backend frameworks like Express.js. Included features such as helmet.js for securing HTTP headers and express-validator for input validation help protect applications from common security vulnerabilities, such as XSS and SQLi attacks [21].

Despite its advantages, Express.js does have some limitations. Its minimalistic nature means developers may need to write more repetitive code compared to more opinionated frameworks like Django or Ruby on Rails. However, this flexibility allows for greater customization and control over the application architecture [50].

2.3 Browser Plugins

A browser plugin, or extension, is a software module that enhances the capabilities of a web browser by adding specific features or functionalities. These plugins integrate with the browser's user interface and can monitor, modify, or block web activity based on predefined rules. Browser plugins are customisable, allowing them to meet specific requirements, such as restricting access to unauthorised websites or monitoring user behaviour.

Browser plugins operate within a sandboxed environment and require explicit permissions to access browser APIs or user data, ensuring security and controlled operation. However, a study found that over 50% of installed extensions were high risk and had the potential to cause extensive damage to organizations using them. These extensions could capture sensitive data from enterprise applications, execute malicious JavaScript, and transmit protected information, including banking details and login credentials, to external parties [27].

To mitigate these risks, it's essential to implement robust security measures when developing and deploying browser plugins. Regular updates and adherence to security best practices can help protect against vulnerabilities.

2.3.1 General Capabilities

The functionality of a browser plugin can be broadly categorised into several areas. Their ability to inject scripts directly into a webpage's Document Object Model (DOM) is fundamental, enabling them to alter content, track user interactions, and modify page behaviour [5] .

Key capabilities include:

- Content Modification: Plugins can add, remove, or modify elements on a webpage. For instance, ad-blockers operate by injecting scripts that identify and hide advertisements, while accessibility plugins can alter fonts or colours to improve readability [43];
- User Activity Monitoring: Plugins can listen for various user-generated events, such as mouse clicks, keyboard inputs, and text selection. This capability is extensively used in web analytics and can be adapted for educational purposes to track student engagement [1, 5];
- Browser Control and Automation: Extensions can manage browser functions, such as tab handling, bookmark synchronisation, and history management. Some plugins can automate repetitive tasks by simulating user actions [18, 43];
- Communication with External Services: Plugins are able to send and receive data from external servers, which is a crucial feature for monitoring applications. This allows them to log activities, authenticate users, and retrieve specific configurations from a backend system [1].

2.3.2 Inherent Limitations

Despite their versatility, browser plugins operate within a constrained environment, leading to certain limitations that must be considered:

- Security Risks: As mentioned previously, plugins represent a potential security vulnerability, and malicious plugins can be used for data collection, phishing, or injecting malware. This necessitates a strict review process by browser developers and careful consideration by users [1, 5, 43];
- Performance Overhead: A large number of active plugins or a single poorly coded plugin can consume significant memory and Central Processing Unit (CPU) resources, leading to reduced browser performance and a degraded user experience [26];
- Cross-Browser Compatibility: Plugins are developed for specific browser APIs. A plugin for Chrome, for example, is not natively compatible with Firefox or Safari. This lack of interoperability requires separate development efforts for each browser, increasing the complexity and cost of deploying a solution across multiple platforms [18];
- API Restrictions: Browser APIs, while powerful, are not limitless. They impose sandboxing and other security measures that prevent plugins from accessing certain parts of the operating system or browser data, such as a user's local files or password manager, without explicit permission. This can restrict the scope of what a plugin is able to monitor or control [5, 43].

2.4 Authentication Technologies in Examination Platforms

In the context of virtual examination platforms, robust authentication mechanisms are essential to ensure that only authorized individuals can access and participate in assessments. This section analyses various authentication technologies, including 2FA, OTP, authentication applications, and biometric identification methods, highlighting their advantages and disadvantages.

2.4.1 Two-Factor Authentication (2FA)

Two-Factor Authentication enhances security by requiring users to provide two distinct forms of identification before granting access. Typically, this involves something the user knows (e.g., a password) and something the user has (e.g., a physical token or a mobile device). By combining these two factors, 2FA adds an extra layer of protection, making it more challenging for unauthorized individuals to gain access, even if they have obtained the user's password [39].

One of the key advantages of 2FA is enhanced security. By requiring two forms of identification, it significantly reduces the likelihood of unauthorised access. Another advantage is the mitigation of password vulnerabilities. Even if a password is compromised, the second factor acts as a barrier against unauthorised entry [39, 46].

However, 2FA also has some drawbacks, with user convenience being one of the main concerns. The additional step in the login process can be seen as cumbersome by some users. Additionally, there is a dependency on secondary devices. If the second factor involves a physical device, losing access to this device can prevent legitimate access [39].

2.4.2 One-Time Password (OTP) and Authentication Applications

One-Time Passwords are unique codes generated for a single login session or transaction. These codes are often delivered via Short Message Service (SMS) or generated by authentication applications installed on a user's device. Authentication apps, such as Google Authenticator or Authy, generate time-based OTPs that users enter alongside their regular passwords [10, 35].

One of the key advantages of OTPs is their resistance to replay attacks [25]. Since OTPs are valid for only a short period, intercepted codes are less likely to be used maliciously. Another advantage is the enhanced security over SMS. Authenticator apps are widely regarded as a secure option for two-factor authentication, as they are not vulnerable to Subscriber Identity Module (SIM) swapping or interception of SMS messages [10].

However, there are some disadvantages to OTPs. One major drawback is device dependency. Access to the authentication app is required, and losing the device can complicate the authentication process. Another disadvantage is the initial setup complexity. Users must install and configure the authentication app, which may be challenging for less technologically proficient individuals [10, 47].

2.4.3 Biometric Identification Methods

Biometric authentication utilises unique physiological or behavioural characteristics, such as fingerprints, facial recognition, or iris scans, to verify an individual's identity. This method is increasingly integrated into devices like smartphones and laptops.

One of the key advantages of biometric authentication is its uniqueness and inability to be shared. Biometric traits are unique to each individual and cannot be easily shared or duplicated, providing a high level of security. Another advantage is user convenience. Biometric authentication offers more convenience than other two-factor authentication methods, as users do not need to remember extra passwords or carry additional devices [38].

However, biometric authentication also has some disadvantages. One major concern is privacy. The collection and storage of biometric data raise significant privacy issues, as unauthorised access to this data can lead to identity theft. Another disadvantage is the potential for false positives and negatives. Biometric systems are not infallible and can sometimes misidentify individuals, leading to access issues. Additionally, there is the issue of permanence. Unlike passwords, biometric data cannot be changed if compromised, posing a long-term security risk [38, 56].

2.4.4 Conclusion

Each authentication method offers distinct advantages and challenges. 2FA and OTP provide enhanced security by adding layers to the authentication process, though they may introduce user convenience issues. Biometric identification offers a balance between security and convenience but raises concerns regarding privacy and data protection. Selecting the appropriate authentication method for a virtual examination platform requires careful consideration of security requirements, user experience, and potential risks associated with each technology.

2.5 Real-Time Activity Monitoring

Real-time activity monitoring is crucial for applications that demand immediate feedback and interaction, such as virtual examinations, collaborative tools, and live data dashboards. Efficient communication between clients (browsers) and servers is essential to ensure seamless interaction and responsiveness. Some technologies enable this real-time communication, each with unique mechanisms, benefits, and limitations. This section explores three primary methods, polling, WebSockets, and Server-Sent Events (SSE), offering a comparative analysis to help selecting the most suitable approach for an effective monitoring platform.

2.5.1 Polling and Long Polling

Polling is one of the earliest techniques employed to achieve real-time communication between a client and a server. In this approach, the client periodically sends HTTP requests to the server to check for new information. If new data is available, the server responds with the updated information, otherwise, it returns an empty response. This method is easy to implement and does not require persistent connections. However, it can lead to increased server load and unnecessary network traffic, especially when the polling frequency is high, as the server must handle numerous requests that may not always result in data transmission [20].

To address the inefficiencies of traditional polling, long polling was introduced. In long polling, the client sends a request to the server, and if the server does not have new information, it holds the request open until new data becomes available or a timeout occurs. Once the server responds, the client immediately sends a new request, ensuring that the server can promptly push new data to the client as it becomes available. While this approach reduces the number of requests compared to traditional polling, it still involves some overhead due to the continuous opening and closing of HTTP connections [20].

2.5.2 WebSockets

WebSockets represent a significant advancement in real-time web communication. They establish a persistent, full-duplex connection between the client and server over a single Transmission Control Protocol (TCP) connection, allowing for bidirectional data transfer without the need for repeated HTTP requests. This persistent connection enables the server to push data to the client instantly, making WebSockets highly efficient for applications requiring real-time updates. However, maintaining numerous open connections can increase server resource consumption, and implementing WebSockets may be more complex compared to simpler methods like polling [20, 32].

2.5.3 Server-Sent Events (SSE)

Server-Sent Events (SSE) provide a mechanism for servers to push updates to clients over a continuous HTTP connection. Unlike WebSockets, which support bidirectional communication, SSE is unidirectional, allowing only the server to send data to the client. This method is suitable for applications where updates flow primarily from the server to the client, such as news feeds, stock price updates, or real-time notifications. SSE is simpler to implement than WebSockets and benefits from automatic reconnection and event ID management. However, it is limited to server-to-client communication and may not be suitable for applications requiring client-to-server messaging [12].

2.5.4 Comparison and Considerations

When selecting the appropriate technology for a real-time monitoring platform, some factors should be taken into account. Complexity is a key consideration, as polling and long polling are relatively easy to implement but can result in inefficient resource utilisation. In contrast, WebSockets and SSE provide more efficient communication methods, but with increased implementation and management complexity.

The communication pattern required by the application also plays a significant role in the decision-making process. If bidirectional communication is needed, WebSockets are the most suitable choice. However, for unidirectional communication from the server to the client, SSE offers a simpler and more efficient solution.

Scalability is another important factor, as maintaining a large number of persistent connections with WebSockets can be resource-intensive for the server. Similarly, polling and long polling can place a significant strain on server resources due to the frequent HTTP requests. SSE provides a balance by enabling efficient server-to-client communication without the overhead associated with full-duplex connections.

Browser support must also be considered, as most modern browsers are compatible with WebSockets and SSE. Nevertheless, it is crucial to verify compatibility with the users' browsers, particularly if older versions need to be supported.

In conclusion, the choice between polling, WebSockets, and SSE depends on the specific

requirements of the application, including the desired communication pattern, scalability needs, and implementation complexity. For applications requiring real-time, bidirectional communication, WebSockets are often the preferred option. On the other hand, if the focus is on simpler, unidirectional updates from server to client, SSE may be the better choice. While polling and long polling can serve basic real-time needs, they may lead to inefficiencies in resource utilisation.

2.6 E-learning Standards for Learning Management System (LMS)

A Learning Management System (LMS), is a software application or web-based technology used to plan, implement and assess a specific learning process. It's used for e-learning practices and, in its most common form, consists of two elements: a server that performs the base functionality and a User Interface (UI) that is operated by instructors, students and administrators [28].

LMS like Moodle serve as central hubs for educational content delivery, administration, and tracking. To ensure interoperability and consistent data exchange between different e-learning content, tools, and platforms, various technical standards have been developed. These standards dictate how learning content communicates with the LMS and how learning activities are recorded. Among the most prominent are SCORM, xAPI, and cmi5, each with distinct capabilities and levels of sophistication, influencing their suitability for different types of educational data tracking.

2.6.1 Sharable Content Object Reference Model (SCORM)

SCORM is a widely adopted e-learning standard, considered a foundational standard for packaging and delivering web-based learning content. Introduced by the Advanced Distributed Learning (ADL) Initiative, SCORM defines content structure and communication with an LMS. Its primary function is to enable content reuse and portability across various SCORM-compliant LMS platforms, including Moodle [37, 55].

SCORM tracks basic learner interactions such as completion status (e.g., "completed" or "incomplete"), success status (e.g., "passed" or "failed"), score, and time spent on content [41]. Content is typically packaged into "Sharable Content Objects (SCOs)" that are self-contained and interact with the LMS server via an API based on JavaScript, communicating data in a predefined set of key-value pairs. While effective for traditional linear courses, SCORM's limitations rest in its rigid data model, which is less adaptable to capturing complex, informal, or experiential learning activities occurring outside formal course structures [37].

SCORM defines a Run-Time Environment (RTE) for content aggregation and learning objects for web-based e-learning. This component standard addresses content sharing and reuse across platforms. SCORM documents primarily cover three aspects: the Content Aggregation Model (CAM), the RTE, and Sequencing and Navigation (SN) [55].

The SCORM RTE provides a mechanism that permits learning resources to be utilised

across different LMS. SCORM establishes a set of data model parameters and employs a unified data element for compiling source code, enabling tracking across platforms. The SCORM RTE standard focuses on three key aspects: Launch, API, and Data Model.

Launch defines a common method for restarting object content and presenting learning resources in the LMS. The API comprises a series of predefined functions that serve as application program interfaces, with the LMS server-side providing an API Adapter for function execution. The SCORM Data Model can be abstract, presenting understanding and mastery challenges for users and developers. However, research into the SCORM RTE model established a Data Model detection procedure using JavaScript. Observation of the Data Model's logical performance on the Moodle platform demonstrated the interaction between the LMS and the SCORM Data Model and detailed the application characteristics of data parameters. This provides assistance for designing a LMS or developing online educational material using the SCORM criterion [55].

2.6.2 Experience API (xAPI)

The Experience API, known as xAPI, represents an evolution in learning technology standards, moving beyond SCORM's constraints. It permits the collection of data on user activities from diverse sources, including mobile applications, virtual reality environments, and gaming platforms. This capability contrasts with SCORM's focus on traditional, linear learning tasks. xAPI facilitates data analysis and evaluation of educational programmes, encompassing methods such as microlearning and adaptive educational programmes [37].

Unlike SCORM's focus on content packages, xAPI is structured around a statement format: "Actor did Verb Object" (e.g., "José completed the quiz"). These statements are stored in a Learning Record Store (LRS), which can operate independently or be integrated within an LMS [37].

The flexibility of xAPI's data model allows it to capture data from diverse sources, including simulations, games, mobile applications, and real-world performance. This makes it suitable for tracking complex behaviours and non-traditional learning paths. Its extensible nature allows the definition of new "verbs" and "objects" as required, providing a data richness that SCORM does not offer [37].

In the context of the intelligent big data era, education has undergone changes. Educational resources are gradually digitising from traditional paper resources, representing a development trend. The effective utilisation of resources has become a challenge, as resource waste can occur during use, resulting in low resource utilisation. xAPI addresses this by providing a standard for integrating resources through analysis of data generated during the learning process [15].

Based on the xAPI standard, learning data can be standardised and stored. This enables a division basis for the cross-media integration of educational digital resources [15]. xAPI implementation requires infrastructure complexity and technical skill [37].

2.6.3 Computer Managed Instruction 5 (cmi5)

The cmi5 serves as a specification that bridges the gap between SCORM and xAPI, providing guidance for "plug and play" interoperability of e-learning content across Learning Management Systems [33].

It operates as an xAPI Profile, incorporating a vocabulary model and specific xAPI Statement patterns. The design intent for cmi5 was to facilitate the transition from SCORM-based content to the xAPI specification, combining the flexibility of xAPI for diverse learning activity tracking with the structural framework of SCORM that learning technologies have historically used. SCORM standards were not extensible enough to support emerging technologies and adequately capture learning and performance data, which motivated the development of cmi5 [33].

There are four essential elements within cmi5 [52]:

- Assignable Unit (AU): AUs are the launchable components of content, encompassing concepts such as completion, pass/fail status, score, and duration. Each learning activity within a course structure represents an AU;
- Course Package: This element comprises all AUs within a course and their launch parameters. It represents the course structure and acts as a package of references to each AU. A cmi5 course package is an Extensible Markup Language (XML) file, typically named "cmi5.xml" when saved in a ZIP file;
- LMS: The LMS is responsible for launching cmi5 content, registering learners, tracking learner progress, and conducting analysis and reporting on performance;
- LRS: The LRS receives, stores, and returns xAPI statements. It functions as the central repository for learning activity data, interacting with all other tools that send or retrieve such data.

The cmi5 process uses xAPI to track and transmit data about learning activities to a conformant LRS. The workflow involves the author creating AUs and a course structure, which the administrator then imports into the LMS. The LMS records learner enrollment as a registration. When a learner initiates the launch of an AU, the LMS prepares launch and session data, recording this information in the LRS before redirecting the learner to the AU presentation. Learners interact with the AU, which then records activities, scores, mastery, and completion data to the LRS [52].

cmi5 offers benefits for organisations seeking to use xAPI's tracking capabilities while maintaining LMS integration. It supports capturing advanced learning data beyond SCORM's capabilities by using xAPI with controlled vocabularies, ensuring interoperability across systems. It is the preferred xAPI Profile for incorporating xAPI-enabled activities launched from an LMS, addressing potential data portability and semantic interoperability issues that could arise with other xAPI profiles. Furthermore, cmi5 enables the incorporation of newer training technologies, such as simulations, virtual reality, augmented reality, and non-browser-based

applications, which SCORM could not effectively track. cmi5 also plays an important role in the Department of Defense (DoD) modernization efforts, facilitating the transition from SCORM-based LMS-centric courseware to a distributed learning environment [33].

2.6.4 Comparison of SCORM, xAPI, and cmi5

This section compares SCORM, xAPI, and cmi5 standards in e-learning. The comparison shows their features for different learning setups and tracking requirements. The Table 2.2 summarizes these features, followed by a discussion of the observations.

Table 2.2 Comparison of SCORM, xAPI, and cmi5 Features [51].

Feature	SCORM	xAPI	cmi5
Defined content launch	✓	-	✓
Track "anything"	-	✓	✓
Normalized reporting	✓	-	✓
Mobile applications	-	✓	✓
Distributed content	-	✓*[1]	✓
Data portability	-	✓	✓
Extensibility	-	✓	✓
Normalized completion criteria	-	-	✓
Multiple lesson support	-	-	✓

[1] While xAPI doesn't define a content packaging standard, it supports the concept of a learning activity residing anywhere.

The Table 2.2 shows how e-learning standards have changed to handle new learning situations:

- Defined Content Launch: Both SCORM and cmi5 offer a process for launching content within an LMS. This means they provide rules for how a course or learning activity should start. xAPI, by itself, does not specify a content launch mechanism, focusing instead on data capture;
- Track "anything": xAPI and cmi5 stand out in this area. They allow for the tracking of a variety of learning experiences, including those outside LMS environments (e.g., mobile apps, simulations, VR). SCORM is more limited, primarily designed for linear, course tracking;
- Normalized Reporting: SCORM and cmi5 both provide normalized reporting. This means they have established standards for how learning data is structured and reported, making it easier to analyze and interpret across systems. While xAPI can track "anything," without a profile like cmi5, the raw data might lack the structure for normalization;

- Mobile Applications: xAPI and cmi5 are designed to support tracking within mobile applications, enabling a flexible learning experience. SCORM does not inherently support mobile application tracking as its content typically needs to reside on the same server as the LMS;
- Distributed Content: cmi5 fully supports distributed content, meaning learning materials do not need to be hosted on the same server as the LMS. xAPI conceptually allows content to reside anywhere, though it lacks a content packaging standard. SCORM requires all content to be on the same server or domain as the LMS;
- Data Portability: xAPI and cmi5 enable data portability, allowing learning data to be easily shared and exchanged across systems. SCORM data is typically confined within the LMS where it was collected;
- Extensibility: Both xAPI and cmi5 are extensible, capable of capturing a range of learning data beyond completion and score. This flexibility allows them to adapt to new learning technologies and methodologies. SCORM is less extensible, with a data model;
- Normalized Completion Criteria: cmi5 stands out by establishing interoperable rules for defining completion and mastery of learning activities. This provides structure for xAPI data. xAPI on its own does not define satisfaction criteria, and SCORM's completion tracking is tied to its structure;
- Multiple Lesson Support: cmi5 packages allow for Assignable Units in a hierarchy with progression criteria, similar to how SCORM handles Sharable Content Objects. This provides a structured approach to course delivery while leveraging xAPI's tracking power.

To summarise, SCORM set the stage for basic e-learning. xAPI changed tracking with flexibility for different learning experiences. cmi5 combines SCORM's structure with xAPI's data capture. It works well for current and future learning environments.

2.6.5 Applicability to the Browser Monitoring Plugin

For a browser monitoring plugin designed for academic examinations, the relevance of SCORM, xAPI, and cmi5 varies significantly based on their inherent functionalities and data models. The primary objective of such a plugin is to detect behaviours indicative of academic misconduct.

SCORM, while a foundational standard for content delivery and basic progress tracking within LMSs, offers limited utility for browser activity monitoring. Its data model is structured for interactions within a Sharable Content Object (SCO), focusing on completion, score, and time spent on predefined content elements [33]. SCORM is not designed to capture broader browser behaviours such as tab switching, window focus changes, copy-pasting, or access to external websites, all of which are critical for assessing exam integrity. Its reliance on content residing within the LMS domain further restricts its applicability for monitoring activities outside this boundary. Therefore, for collecting detailed behavioural data on a student's browser during an examination, SCORM's capabilities are insufficient.

The xAPI presents the most suitable standard for a browser monitoring plugin due to its flexible and extensible data model. xAPI's "Actor Verb Object" statement structure allows for the recording of virtually any learning experience, including specific and custom browser activities [15, 37]. For instance, statements can be formulated to record "Student A opened a new tab," "Student A copied text from a webpage," or "Student A navigated to a forbidden url." This capability enables the plugin to construct a detailed record of student behaviour, extending beyond what a traditional LMS typically records. The flexibility of xAPI is crucial for capturing events directly relevant to detecting academic misconduct.

cmi5, as an xAPI Profile, bridges the structured launch and reporting of SCORM with the flexible tracking of xAPI. While it provides rules for how online courses are imported, launched, and tracked using an LMS and xAPI, its primary benefit lies in standardizing content delivery and formal reporting back to the LMS for course completion and credit [53]. For a standalone browser plugin focused purely on monitoring behaviour during an examination, the additional overhead and specific rules of cmi5 for content packaging and formal launch might be less critical than xAPI's raw data capture capabilities. However, if a formal, structured integration of monitoring data with an LMS's grading or proctoring module is required, where specific completion criteria or session management are essential, cmi5 could provide a robust framework for reliable data exchange beyond raw xAPI statements [33]. It is particularly relevant for integrating newer training technologies launched from an LMS [54].

In conclusion, while SCORM is foundational for content delivery, it offers limited direct value for the detailed behavioural monitoring required by a browser plugin. xAPI, with its highly adaptable data model, is the most appropriate standard for recording detailed and custom monitoring events pertinent to exam integrity. cmi5 could become relevant if a more formal, structured integration of monitoring data with an LMS for official record-keeping is required, providing a bridge between the flexibility of xAPI and the traditional structure of LMS interactions. The selection of the most suitable standard depends on the specific level of detail and integration desired for the monitoring data within the broader e-learning environment.

Chapter 3

Related Work

The digitization of education has led to the emergence of e-exam technologies, particularly during the COVID-19 pandemic, as educational institutions seek robust and secure solutions for online (remote) and virtual (classroom-based) assessments. E-learning platforms have made these forms of exams a viable assessment method.

However, they also introduce challenges, particularly with regard to cheating and academic integrity. Unlike traditional exams supervised by physical proctors, online exams provide students greater access to unauthorized resources, remote assistance, and the potential for impersonation. Motivations for digital cheating include performance pressure, lack of preparation, and a perception of weakness in detecting dishonesty [6].

This chapter will explore two primary approaches to e-assessment technology:

- Virtualized Exams with Physical Exam Rooms (Section 3.1);
- Online Exams with a Virtual Exam Room (Section 3.2).

Each approach addresses critical issues related to maintaining academic integrity, secure authentication, and effective monitoring. Some specific vulnerabilities include:

- Lack of Physical Oversight: Without real-time human supervision, online exams are more susceptible to undetected dishonesty.
- Access to Unauthorized Resources: Students can quickly search online for answers or communicate with others during exams, which is known as "digital cheating" [34];
- Identity Verification Challenges: Ensuring the test-taker's identity remains difficult without strong authentication methods [16].

These challenges underscore the need for technological solutions to uphold the integrity of e-assessments and support fair and secure testing environments for all participants.

3.1 Virtualized Exams with Physical Exam Rooms

This approach to enhancing exam security within a controlled physical environment involves three primary components: Lockdown Browsers and Proctoring Systems, Continuous Authentication, and Hybrid Implementation. Each of these elements contributes to a more secure and supervised digital examination process, balancing the flexibility of digital exams with the oversight benefits found in traditional settings.

- Lockdown Browsers and Proctoring Systems: Lockdown browsers like Safe Exam Browser (SEB) and Respondus LockDown Browser are widely used to restrict access to unauthorized resources by blocking other applications and websites. These tools are valuable in controlled physical environments where monitoring is possible, though limitations arise in Bring Your Own Device (BYOD) settings, where students might circumvent restrictions [30];
- Continuous Authentication: Biometric-based systems (e.g., facial recognition and fingerprint readers) and behavioural analysis (e.g., keyboard and mouse dynamics) enable continuous verification in physical rooms. This technology can complement physical proctoring by alerting supervisors to potential issues [6, 16];
- Hybrid Implementation: Some institutions, such as Michigan State University, blend biometric and behavioural monitoring with physical proctoring, allowing for centralized remote monitoring across various exam rooms. This hybrid model demonstrates how physical oversight can be enhanced with advanced technologies to detect suspicious behaviours more effectively [34].

3.2 Online Exams with a Virtual Exam Room

This approach fully embraces the concept of online assessments, aiming to create a secure, controlled examination environment where students can take exams remotely with enhanced monitoring systems. It consists of five main components: E-Assessment and Cheating Challenges, AI-Driven Proctoring and Behavioural Detection, Continuous Authentication and Multimodal Biometrics, Legal and Ethical Considerations, and Future Directions in Virtual Proctoring Technology. Each of these components contributes to recreating the controlled conditions of physical exams through advanced authentication, proctoring, and monitoring techniques.

3.2.1 E-Assessment and Cheating Challenges

The shift to online learning environments has introduced novel challenges to academic integrity, primarily arising from increased student access to unauthorized resources and the inherent absence of physical oversight. Key concerns in online assessments include identity verification and the effective monitoring of unauthorized activities [6, 34]. Consequently, robust systems for supervising online examinations are crucial.

One approach involves engineering software platforms that facilitate online exams by mandating camera and microphone activation throughout the assessment period. These systems leverage real-time monitoring to observe and analyse student behaviour, thereby detecting potential instances of academic dishonesty. Such software can enforce measures like limiting students to a single application during the exam [13]. Furthermore, fraud detection in online exams often employs techniques such as real-time face recognition via webcam streams, comparing detected faces with preloaded encodings, and generating alerts for unrecognized faces or the user's absence [36].

Real-world applications of these principles are evident in various institutions. Online assessment platforms, such as Exam.net, have been utilized to facilitate structured digital examinations. For instance, the University of Bucharest incorporated Exam.net in its Romanian language courses to establish controlled browser environments and enable real-time supervision. This platform empowered teachers to track submissions and receive alerts regarding suspicious activities like irregular navigation or late completions, thereby promoting fairness while maintaining instructional flexibility.

Similarly, the University of Milan adapted its examination protocols to manage large-scale online written exams. Initially, teachers employed conferencing platforms and mobile phone cameras for student monitoring. For examinations exceeding one hundred candidates, Proctorio was adopted to automate surveillance. This system recorded video and audio, which were subsequently analysed to flag suspicious behaviours such as head movements, sound anomalies, or a student's absence from view [17].

3.2.2 AI-Driven Proctoring and Behavioural Detection

Advanced proctoring systems like Proctorio and TestWe use AI to monitor students through facial recognition, keystroke logging, and environmental sound analysis, aiming to reduce the need for human proctors. While effective to some extent, these methods can raise privacy concerns due to continuous webcam surveillance [7, 24].

Exam proctoring systems utilise artificial intelligence, computer vision, and machine learning to supervise examinations and maintain academic integrity. These systems monitor student behaviour in real-time, incorporating features such as facial recognition, audio detection, and head movement analysis to prevent cheating [49].

An AI-proctored exam portal combined with a mobile companion application offers an approach to academic integrity, leveraging AI for proctoring and enabling instructors to securely prepare and conduct exams.

The platform includes elements like multiple-choice exams, written assessments with a question bank, and integrated course modules with quizzes and videos. AI algorithms track the exam process in real-time and generate reports for each student, with the mobile companion app enhancing the assessment's integrity [11]. Eye-tracking can detect and examine human visual attention, emotional conditions, and cognitive processes such as efforts to recall a concept or fear of running out of time. This is useful for identifying behaviour patterns in learning

and problem-solving during online exams [14].

3.2.3 Continuous Authentication and Integrated Biometrics

Online exams use continuous biometric and behavioural monitoring to verify student identity and detect unauthorized activity throughout the exam. Techniques like facial recognition, eye-tracking, and keystroke analysis allow for secure, ongoing authentication, addressing impersonation and cheating risks [6, 34].

Proctoring systems establish a controlled examination environment through facial recognition, user activity monitoring, and browser behaviour tracking. This approach combines machine learning-based facial recognition for identity verification, logging of user actions, and browser monitoring to detect behaviours.

The results demonstrate a decrease in cheating incidents, increased exam credibility, and improved perception of fairness among students and instructors. By encouraging accountability, the system fosters a culture of honesty in the online education environment [40].

3.2.4 Legal and Ethical Considerations

The privacy implications of AI-driven proctoring have provoked controversy. Lawsuits, such as those involving TestWe in France and Cleveland State University in the United States (U.S.), highlight growing concerns around data collection practices in virtual proctoring. These cases suggest the need for more privacy-conscious solutions in virtual exam settings, with an emphasis on compliance with regulations like the General Data Protection Regulation (GDPR) [19].

3.2.5 Future Directions in Virtual Proctoring Technology

Emerging trends in virtual exam proctoring involve adaptive AI models, hybrid solutions that combine multiple monitoring methods, and engagement tools that promote learning without intrusive surveillance. Platforms like Socrative, which emphasize active learning, represent an alternative to intense proctoring by creating a positive assessment environment [4]. Furthermore, future systems may incorporate context-aware, personalised formative assessment in collaborative online environments to better accommodate learners' styles and contexts. Formative assessment constitutes an effective approach for enhancing student motivation and learning outcomes [22].

At the University of Bucharest, blended learning models have been adopted in the Preparatory Year programme to support Romanian language instruction. These include synchronous and asynchronous tasks, backward course design, and continuous digital assessment. Such models aim to balance pedagogy with accountability while avoiding over-surveillance.

3.3 Conclusion

Technological approaches to e-assessment challenges differ significantly depending on the context. Virtual exams conducted in physical spaces offer tighter control but inherently limit flexibility. In contrast, online solutions necessitate a strong focus on effective monitoring, robust privacy measures, and secure authentication.

However, institutions like the University of Milan and the University of Bucharest are showing how hybrid models can successfully balance both pedagogical requirements and assessment integrity. Platforms such as Exam.net further demonstrate that when paired with appropriate safeguards, digital systems can indeed foster fair and secure online assessment environments.

Chapter 4

Developed Work

This chapter presents the architecture, technology stack, and implementation of the Exam Management Platform (EMP) and the Student Exam Plugin (SEP), along with the mechanisms that ensure secure and reliable examination monitoring. It describes the operation of each component, the communication between them, and the security measures implemented to preserve the integrity of the examination process.

Section 4.1: Proposed Method introduces the system architecture, describing the roles of the EMP and SEP and the steps followed to create a secure and efficient examination environment.

Section 4.2: Technology Stack Overview details the technologies used in both the backend and frontend, including server configuration, database management, and the Chrome extension's development environment.

Section 4.3: Implementation of the Exam Management Platform explains the server configuration, data access layer, business logic, authentication and authorisation mechanisms, and the web interface that supports exam management and monitoring.

Section 4.4: Implementation of the Student Exam Plugin outlines the plugin's architecture, including manifest configuration, background processes, content scripts, and user interface, as well as how it integrates with the platform for real-time activity monitoring.

Section 4.5: EMP and SEP Integration and Communication describes the bidirectional communication between the platform and the plugin, the data exchanged, and the protocols used to ensure secure and authenticated interactions.

Section 4.6: Security Measures discusses the multi-layered approach adopted to safeguard user authentication, data integrity, and monitoring accuracy. This includes the deployment at `https://exam-monitoring-platform.onrender.com`, two-factor authentication for Teachers and Supervisors, short-lived sessions for Students, the use of authorisation tokens in all plugin requests, and a hash-based validation mechanism to detect plugin tampering.

Lastly, Section 4.7: Implementation Challenges and Solutions addresses the technical issues encountered during the development process, such as synchronising multi-tab monitoring, maintaining performance, ensuring RBAC, and adapting to Chrome Extension Manifest V3, along with the solutions implemented to overcome them.

4.1 Proposed Method

To approach this problem we started by defining the architecture of the proposed solution, describing the operation and role of each component in the system, providing a clear understanding of how they interact to create a secure and efficient examination environment.

The proposed solution is presented in Figure 4.1. This solution compromises two major components:

- Exam Management Platform (EMP);
- Student Exam Plugin (SEP).

The EMP is a web application available for teachers to manage exams and control students' activity during exam. The SEP is a browser plugin installed on the student's computer, responsible for monitoring their actions and ensuring compliance with the exam rules. This solution assumes that students use their own laptops and Google Chrome as the browser to access the exam.

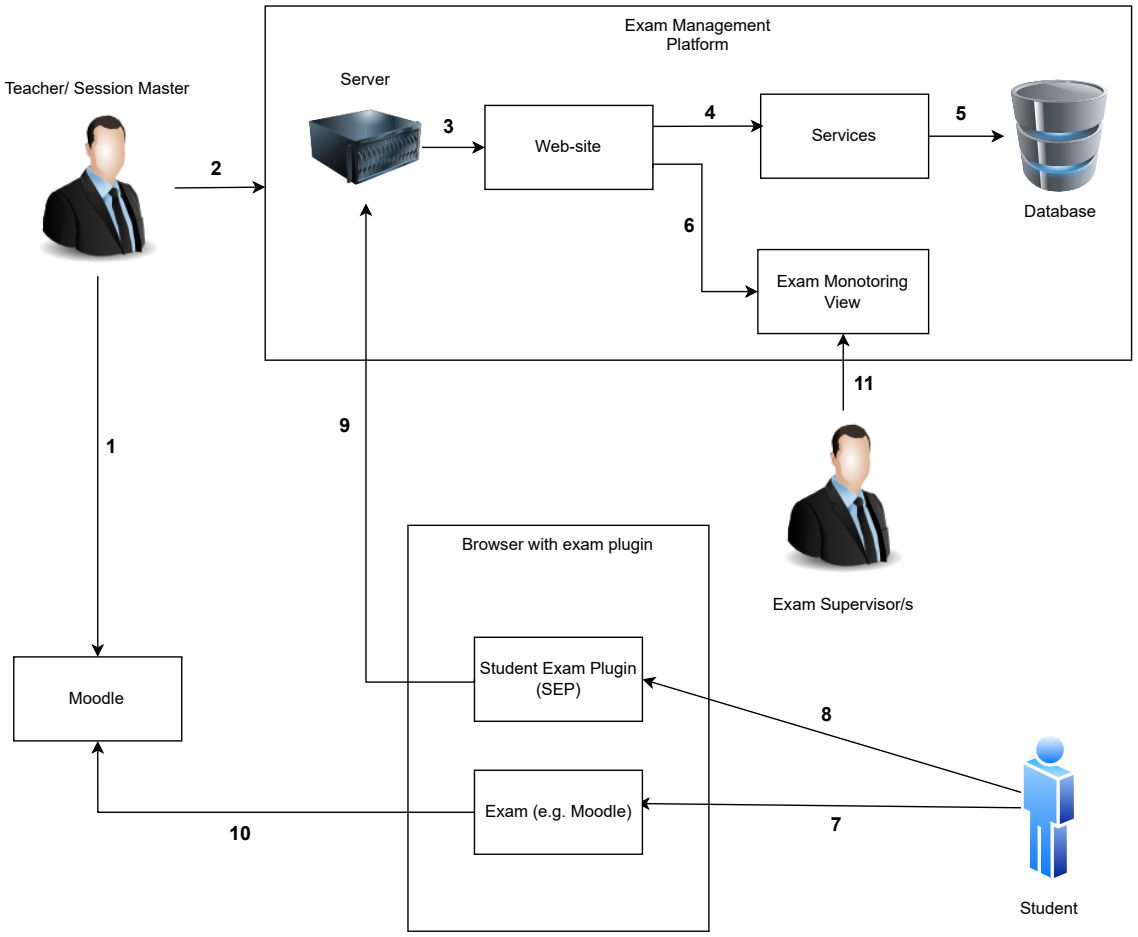


Figure 4.1 Solution architecture with corresponding interaction flow.

The interactions illustrated in Figure 4.1 can be described as follows:

1 **Exam Creation (Moodle):** The Teacher creates the examination within Moodle (LMS);

2 **Exam Creation (EMP):** The Teacher creates the examination on the Exam Management Platform (EMP);

3 **Server/Website Communication:** The Server communicates with the EMP Web-site;

4 **Website/Services Communication:** The Web-site interacts with the EMP Services module;

5 **Data Persistence:** The Services module communicates with the Database to store the examination data;

6 **Teacher Viewing Access:** The Web-site renders the Exam Monitoring View, enabling teachers to monitor student activity in real time;

7 **Student Access:** The Student accesses the examination (e.g., via Moodle);

8 **Plugin Authentication:** The Student must authenticate themselves within the SEP, establishing an association between the student and their seating location within the classroom;

9 **Validation and Monitoring:** The SEP transmits the entered data to the Server for validation. If the data is valid, the student is authenticated, and all subsequent activity performed by the student is sent by the plugin to the server for real-time monitoring;

10 **LMS Data Association:** The LMS (Moodle) is used to store data and associate it with the students (Outside the scope of this project);

11 **Supervisor Monitoring:** The Exam Supervisor(s) access the Exam Monitoring View to track student activity in real time.

The proposed method aims to provide a platform for monitoring examinations by integrating real-time activity tracking with web-based authentication mechanisms. The architecture consists of a Google Chrome extension for monitoring user activity and a web application for managing exams, users, and authentication processes.

The monitoring system follows these steps:

- Exam Management – Teachers can create, update, and manage exams through the web application. The system provides tools for listing enrolled students and assigned supervisors;
- User Authentication – Authentication is managed through a Chrome extension for students and a web application for teachers and supervisors. Students authenticate using their student identifier, the exam identifier, and the desk code assigned to their seat.

Teachers log in with a username and password, followed by a six-digit code sent to their email for 2FA, enhancing security. Supervisors authenticate by providing their email address, to which a six-digit code is sent, granting them secure access to the system;

- Real-Time Activity Monitoring – The Chrome extension continuously checks whether the student navigates away from the examination page and logs any unauthorised activity;
- Notification System – In case of unauthorised behaviour, notifications are triggered to alert supervisors;
- Secure Logging and Data Storage – All actions within the system are logged and securely stored for subsequent analysis. On the platform, teachers have access to a dedicated button that controls logging. When this button is active, all logs occurring during an examination are recorded, whereas if it is deactivated, logging is suspended.

The architecture integrates both front-end and back-end technologies, with communication between components facilitated through RESTful APIs. This ensures scalability and interoperability between different system components.

4.2 Technology Stack Overview

The Exam Management Platform and Student Exam Plugin were developed with a focus on scalability, security, and efficiency. These needs guided the choice of technologies.

The backend of the monitoring platform is implemented using `Node.js`, with the `Express.js` framework employed for handling RESTful API endpoints and routing HTTP requests. PostgreSQL is used as the database management system, with the pg module facilitating communication between the application and the database. The business logic and data access components are structured into five distinct modules:

- `moniplat-server.mjs`: Defines the main `Express.js` server, including middleware configuration (e.g., `cors`, `cookie-parser`, `express-session`, `passport`), database initialisation, and route definition. This module acts as the application's main entry point;
- `moniplat-db.mjs`: This module implements the Data Access Layer (DAL), offering functions to execute SQL queries and manage data. It's responsible for all data operations, including creating, retrieving, updating, and deleting information related to entities such as exams, virtual rooms, and system logs;
- `moniplat-data-mem.mjs`: Provides an in-memory storage alternative for development and testing purposes. It simulates interactions with the database without requiring persistent storage;
- `moniplat-services.mjs`: This module represents the business logic layer. It orchestrates application operations, interacting with `moniplat-db.mjs` or `moniplat-data-mem.mjs` to execute actions, apply business rules, and manage validation and error handling;
- `moniplat-web-site.mjs`: This module is responsible for serving the application's web pages and managing user interface interaction. It defines routes for rendering pages and utilises Handlebars (HBS) to create dynamic views. `SendGrid`, a cloud-based email delivery platform and API service, is integrated to support email functionalities, such as password recovery and the delivery of verification codes to users.

Server-Sent Events (SSE) are also employed for real-time communication, allowing the server to push updates to the client without the client needing to continuously poll for new information.

On the frontend, the user interface is developed using standard web technologies such as Hypertext Markup Language (HTML), Cascading Style Sheets (CSS), and client-side JavaScript. Dynamic page rendering is performed via the HBS templating engine, which allows server-side data injection before pages are sent to the client.

The Student Exam Plugin, on the other hand, is a Google Chrome extension, predominantly developed in JavaScript. Its main components include `background.js`, which handles background operations, `content.js`, which interacts with the content of web pages and `popup.js`, which manages the logic associated with the plugin's popup interface. The visual

structure of the popup is defined in `popup.html`, while the plugin's configuration and permissions are specified in `manifest.json`. To minimise code, Webpack was utilised, configured via `webpack.config.js`. Obfuscation would have been a stronger option for preventing reverse engineering and improving security, but the Chrome Web Store prohibits extensions containing obfuscated code. Minimisation was therefore adopted as an acceptable alternative. Although not equivalent to obfuscation, minimised code is harder to read and thus provides a limited level of protection.

Integration and communication between the platform and the plugin are facilitated through HTTP requests. The plugin transmits monitoring data to the platform's backend, while the platform supplies the plugin with essential information, such as the authorised Uniform Resource Locator (URL) associated with the current exam session. In addition, the plugin sends authentication data provided by the student to the backend and receives a corresponding response indicating success or failure, which determines whether monitoring functionality is activated.

4.3 Implementation of the Exam Management Platform

This section describes the implementation of the server and the core business logic of the EMP. It outlines the main technical components used to turn the initial design into a functional system. The content includes the EMP architecture, server configuration and application startup, the interaction between the data layer and the database, the organisation of business logic through service modules, and the mechanisms for authentication and authorisation. It also covers the development of web views and the user interface. The objective is to explain how each component supports exam management and monitoring functions.

4.3.1 Exam Management Platform architecture

The architecture for the Exam Management Platform is depicted in Figure 4.2. This architecture includes several key modules that work together to manage exams and control student activity. The `moniplat-server.mjs` module serves as the entry point and is responsible for server configuration. It communicates with `moniplat-web-site.mjs`, which handles web page delivery and user interface interactions. Both of these modules interact with the `moniplat-services.mjs` module, which contains the business logic. The `moniplat-services.mjs` module can connect to either `moniplat-data-mem.mjs`, an in-memory storage alternative for testing, or `moniplat-db.mjs`, which is responsible for direct database interactions.

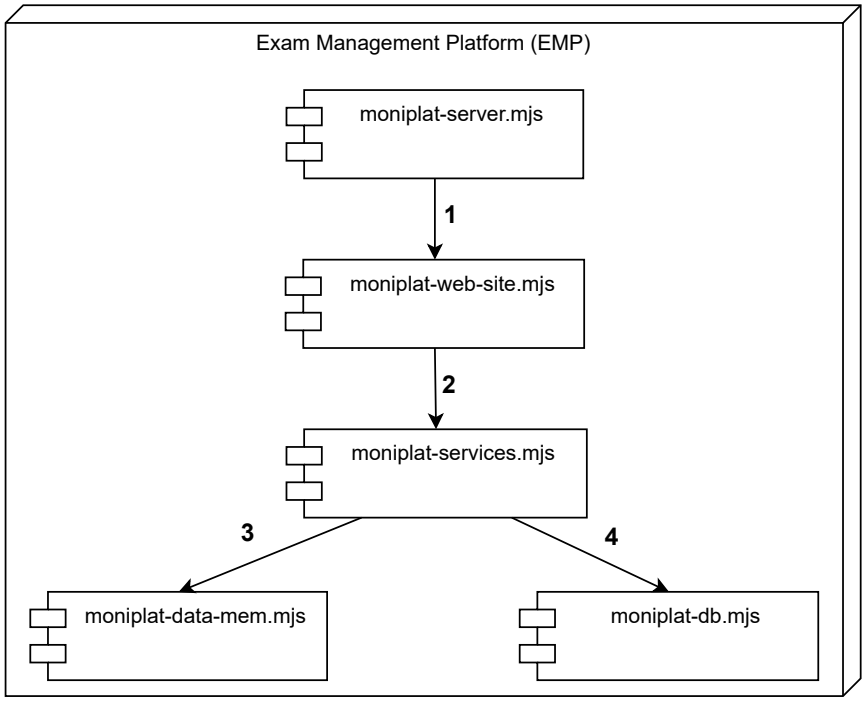


Figure 4.2 EMP Architecture.

The interactions illustrated in Figure 4.2 can be described as follows:

1 **Server/Website Communication:** The `moniplat-server.mjs` component communicates with the `moniplat-web-site.mjs` component;

2 **Website/Services Communication:** The `moniplat-web-site.mjs` component communicates with the `moniplat-services.mjs` component;

3 **Services/In-Memory Communication:** The `moniplat-services.mjs` component communicates with the in-memory data management module (`moniplat-data-mem.mjs`);

4 **Services/Database Communication:** The `moniplat-services.mjs` component communicates with the database persistence module (`moniplat-db.mjs`).

4.3.2 Server Configuration and Entry Point

The `moniplat-server.mjs` serves as the central orchestration point for the `Express.js` application. It's responsible for the server's initial configuration, the integration of essential middleware, and the definition of all application routes. The `Express.js` configuration includes middleware such as `cors` for handling cross-origin requests, `cookie-parser` and `express-session` for session and cookie management, and `passport` for implementing user authentication. Additionally, `morgan` is utilised to log HTTP requests, facilitating debugging and system monitoring.

The connection to the PostgreSQL database is established through a connection pool, which optimises resource management by reusing existing connections rather than creating

new ones for each request. The `moniplat-db.mjs` module, which encapsulates data access operations, and the `moniplat-services.mjs` module, which implements the business logic, are imported and integrated as dependencies. This design separates different responsibilities, making the system easier to manage and update. Application routes are defined to either serve web pages via `moniplat-web-site.mjs` or handle requests through API endpoints.

4.3.3 Data Layer and Database

The `moniplat-db.mjs` module is responsible for direct interaction with the PostgreSQL database. It contains functions that encapsulate the required SQL operations. PostgreSQL connections are managed through a connection pool, which reduces the overhead associated with repeatedly opening and closing connections.

While the `moniplat-data-mem.mjs` module provides an in-memory data storage alternative for testing purposes, the server configuration (`moniplat-server.mjs`) relies on `moniplat-db.mjs` as the primary data source in production. This architecture supports flexibility by allowing the substitution of the persistent database with the in-memory module in development or testing contexts where data persistence isn't required.

The Figure 4.3 presents the Entity-Relationship (ER) model, which illustrates the structure and relationships of the main entities within the database, namely Student, Teacher, Supervisor, Exam, Virtual Room and Log.

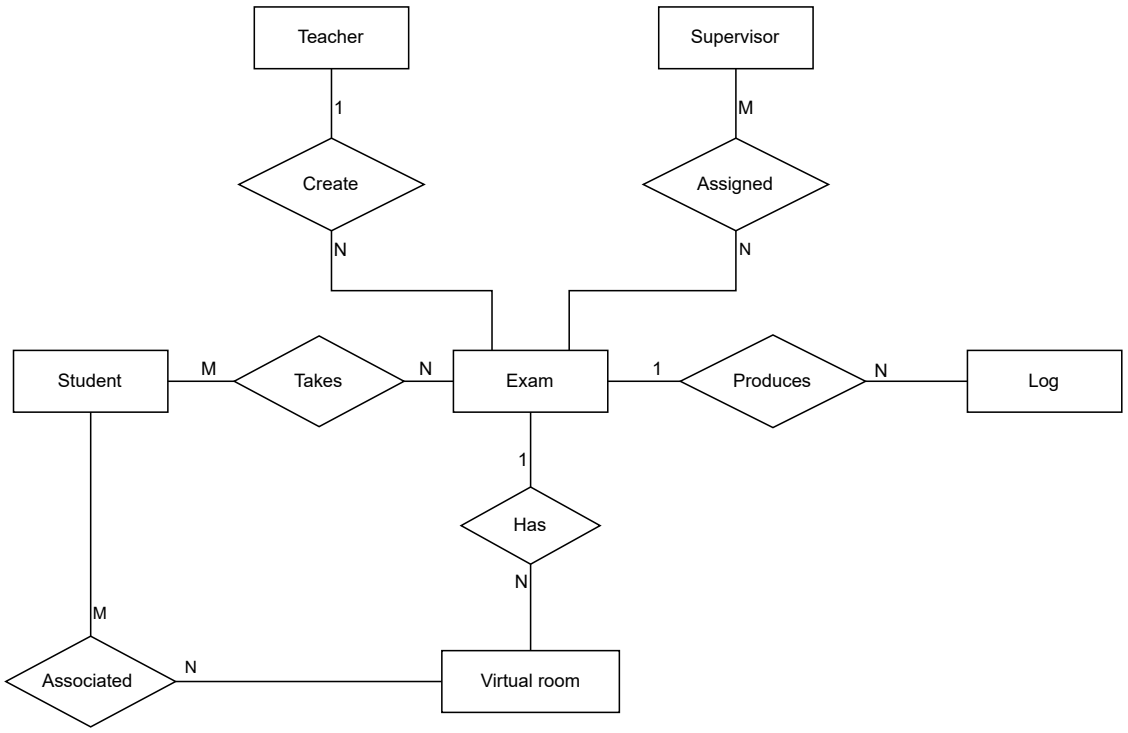


Figure 4.3 ER model of the database.

4.3.4 Business Logic and Services

The `moniplat-services.mjs` module constitutes the business logic layer of the application. It interacts with the data access layer to perform operations while enforcing the relevant business rules. This separation of concerns ensures that the application's logic remains decoupled from both database interactions and user interface rendering, thereby enhancing code clarity and maintainability. Error handling is centralised within this module, where functions reject Promises upon failure, enabling consistent and structured error propagation.

4.3.5 Authentication and Authorisation Management

Authentication within the platform is implemented using `Passport.js`, a middleware for `Node.js` that facilitates user authentication. This system enables the verification of credentials for different user roles (Teachers, Supervisors, and Students) and the creation of secure sessions. For Teachers and Supervisors, session cookies are configured with a maximum age of seven days, as defined in the global Express session options, balancing usability with session persistence and security.

Student authentication, managed through the plugin, follows a distinct mechanism. In this case, the session has a maximum age of ten seconds, which is automatically renewed through periodic communication between the plugin and the server, such as the transmission of activity logs.

This short session duration was deliberately selected to reduce the risk of circumvention attempts. For example, if the student closes the browser and doesn't reopen it within ten seconds, their plugin session is considered lost. This approach helps mitigate fraud scenarios where a student might try to circumvent monitoring by closing and reopening the browser.

Authorisation is implemented through a RBAC model, where different user roles have distinct permissions to access and interact with platform resources:

- Teachers: Have full control over their exams. They can create and configure exams, and access all associated data, including virtual rooms and monitoring logs for students, as demonstrated in Figure 4.4, which represents the use cases a teacher can perform on the EMP;
- Supervisors: Has read-only access to virtual rooms for the exams assigned to them. Cannot create, edit, or delete exams or associated resources, as demonstrated in Figure 4.5, which represents the use cases a supervisor can perform on the EMP;
- Student: Has no access to the platform interface. Interacts exclusively through the plugin, which submits monitoring data and retrieves necessary exam configuration parameters.

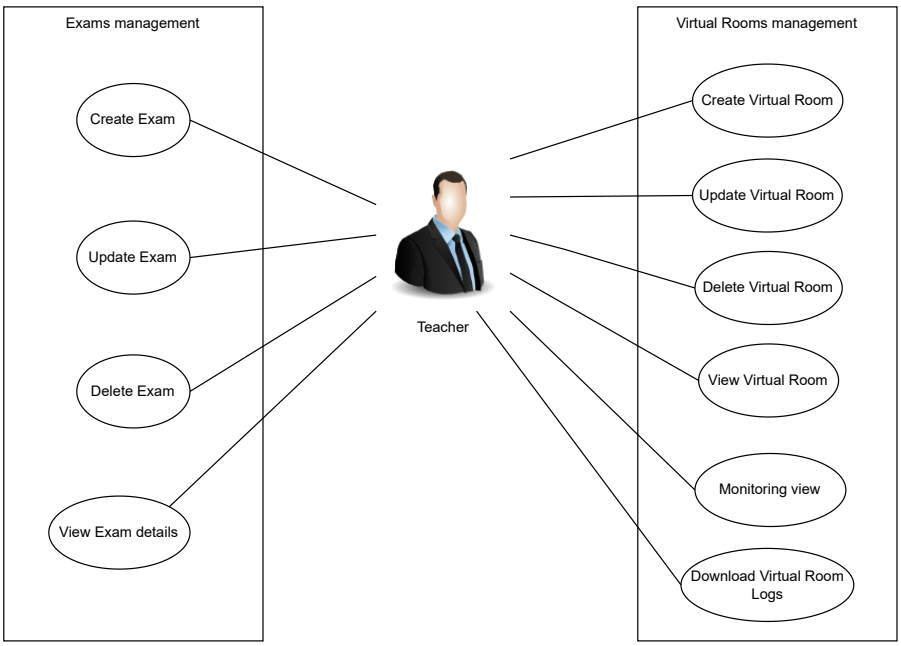


Figure 4.4 Teachers use cases.

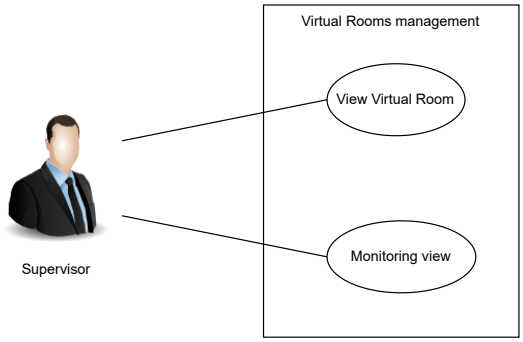


Figure 4.5 Supervisor use cases.

Furthermore, both Teachers and Supervisors are restricted to interacting only with the exams explicitly assigned to them. This ensures that users cannot access or manipulate data from unrelated exams, thereby enforcing data segregation and preserving confidentiality.

4.3.6 Views and Web Interface

The `moniplat-web-site.mjs` module serves as the backend component responsible for generating and delivering the application's web pages, as well as handling user interactions through the interface. It acts as an intermediary between the server-side business logic and the content rendered in the user's browser. In addition, this module also manages user authentications.

The module defines functions corresponding to the main views of the application. For instance, `getHome` renders the homepage, `getLogin` manages the login interface (with role-specific variants such as `getLoginTeacher` and `getLoginSupervisor`), and `getMonitoringView` handles the main monitoring interface. Each function fetches the required data from the service layer (`moniplatServices`) and uses this data to render the corresponding view.

Dynamic page generation is implemented using the HBS templating engine. Handlebars enables the injection of server-side data into HTML templates before they are sent to the client. This means the pages aren't static, the rendered pages reflect real-time information, such as exam identifiers, student lists, activity logs, and virtual room layouts, based on the current state of the system and the permissions of the authenticated user. This approach enables the delivery of contextual and interactive user interfaces.

4.4 Implementation of the Student Exam Plugin

This section describes the development of the Student Exam Plugin, a key component of the Exam Management Platform. It covers the plugin's architecture, manifest configuration, background processes, content script interactions, and user interface. It explains how the plugin integrates with the platform to monitor student activity during exams.

4.4.1 Student Exam Plugin archictecture

The architecture of the Student Exam Plugin includes several key components, as shown in Figure 4.6. The `popup.js` file handles the logic for the user interface, which is defined by `popup.html`. The `popup.js` module communicates with `background.js`, which serves as the central coordinator for the plugin. The `background.js` module manages background operations, browser events, and communication between other components and the server. The `background.js` module also interacts with `content.js`, which is responsible for detecting user interactions within web pages.

The `background.js`, `content.js`, and `popup.js` communicate with Chrome APIs to perform their respective functions, for example, checking whether a student is opening new tabs using `chrome.tabs`, or storing and retrieving information using `chrome.storage`, among others.

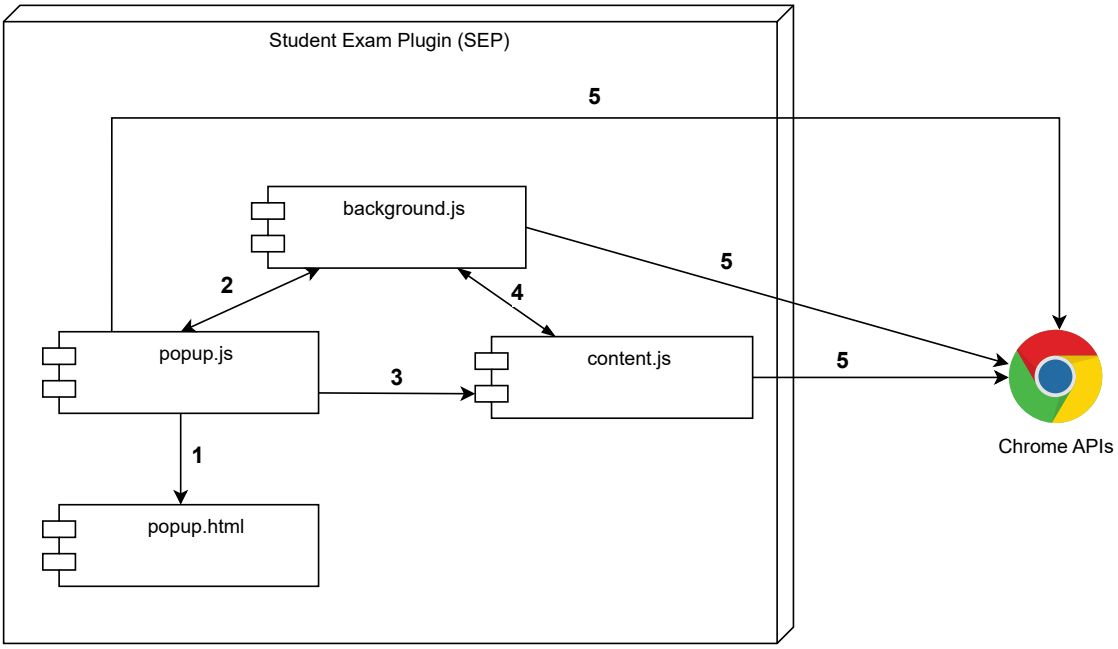


Figure 4.6 SEP Architecture.

The interactions illustrated in Figure 4.6 can be described as follows:

1 **Authentication Data Retrieval:** The `popup.js` module obtains the authentication data entered by the student within the `popup.html` interface;

2 **Monitoring Control:** The `popup.js` module also communicates with `background.js` to initiate or terminate the monitoring process when the student logs into or out of the system;

3 **Content Script Execution:** The `popup.js` module executes the script contained within `content.js`;

4 **Event Detection and Reporting:** The `content.js` module detects events and sends them to `background.js`. Furthermore, `background.js`, similar to `popup.js`, executes the script present in `content.js`;

5 **Chrome API Communication:** These modules (`background.js`, `popup.js`, and `content.js`) communicate with the Chrome APIs.

4.4.2 manifest.json

The `manifest.json` file is the backbone of the plugin, defining its identity and functionalities. It specifies essential metadata such as name, version, and description, and most importantly, the necessary permissions and the components that make up the plugin, as shown in the Figure 4.7.

```
{
  "manifest_version": 3,
  "name": "Exam Monitoring Plugin",
  "version": "2.7",
  "description": "Monitors browser activity during academic exams by tracking tab usage, window focus, and login state to ensure exam integrity.",
  "icons": {
    "16": "images/Isel_logo_16.png",
    "32": "images/Isel_logo_32.png",
    "48": "images/Isel_logo_48.png",
    "128": "images/Isel_logo_128.png"
  },
  "permissions": [
  "tabs",
  "windows",
  "storage",
  "scripting"
],
  "background": {
    "service_worker": "background.js"
  },
  "action": {
    "default_popup": "popup.html",
    "default_icon": {
      "16": "images/Isel_logo_16.png",
      "32": "images/Isel_logo_32.png",
      "48": "images/Isel_logo_48.png",
      "128": "images/Isel_logo_128.png"
    }
  },
  "host_permissions": ["<all_urls>"]
}
```

Figure 4.7 manifest.json file

The field "manifest_version": 3 signifies compliance with the latest and most secure version of Chrome's extension architecture. The declared permissions are particularly important, as they grant access to specific Chrome APIs essential for monitoring functionality. These include tabs (for accessing information about open tabs), windows (for detecting focus changes), storage (for local data storage), and scripting (for injecting scripts into web pages).

The background section points to `background.js`, which works in the background to handle events and communication. The action section configures the behaviour of the extension icon, including linking it to `popup.html`, which shows the user interface when clicked. Additionally, the host_permissions field includes ["<all_urls>"], allowing the extension to access and interact with any website. This permission is necessary so that the plugin can monitor student activity regardless of the site being visited.

4.4.3 Background (background.js)

The `background.js` acts as the central coordinator of the plugin. It runs in the background and handles core logic, browser events, and communication between the plugin components and the server.

One of its key functions is to mediate communication between `content.js` and the backend. It listens for messages from `content.js`, which detects actions on web pages, such as right-clicks or text selection using the `chrome.runtime.onMessage.addListener` API. When a message is received, `background.js` processes the monitoring data and forwards it to the server using the notifyAdministrator function. This function sends HTTP POST requests to the `/student-logs` endpoint via the Fetch API.

In addition to forwarding data, `background.js` also manages the validation of visited URLs. It retrieves the allowed base URL for the exam via the `fetchAllowedUrl` function and uses `isAllowedUrl` to check whether the current tab's URL matches. To monitor activity over time, a `monitoringInterval` is set up. At each interval, the script checks the browser window's state (`chrome.windows.getLastFocused`) and the active tab's URL (`chrome.tabs.query`). If the tab does not match the allowed URL, the window is not focused, or the browser is not in fullscreen or maximised mode, the `notifyAdministrator` function is triggered to report the event.

The `background.js` also listens for Chrome events such as `chrome.tabs.onActivated` (when the user switches tabs) and `chrome.windows.onFocusChanged` (when the browser window gains or loses focus). Additionally, it ensures that `content.js` is injected into all tabs on startup and after the student logs in, using the functions `injectContentScriptIntoAllTabsOnStartup` and `injectContentScriptIntoTab`.

4.4.4 Content Script (content.js)

The `content.js` serves as the plugin's interface with the content of web pages. It is injected by `background.js` and runs in an isolated context. Its role is to detect specific user interactions within the page, such as visibilitychange (to track tab or window focus changes), copy, paste, cut (for text manipulation), contextmenu (for right-click usage), and mouseup (to detect text selection).

When one of these events occurs, `content.js` collects relevant details, including the current page URL, any selected text, or clipboard content. This data is then sent to `background.js` using the `sendMessageToBackground` function, where it is processed and forwarded to the server.

To avoid multiple injections of the script into the same page which could cause errors or duplicated behaviour, a `window.contentScriptLoaded` flag is used to ensure the script is only injected once.

4.4.5 User Interface (popup.html) and popup.js

The `popup.html` defines the plugin's graphical user interface, a small window that appears when clicking the plugin icon in the Chrome toolbar. This interface is simple, focused on the login form (with fields for examId, studentId, deskcode and deskcode confirmation) and a login/logout button. The behaviour and logic behind this interface are handled by `popup.js`.

The `popup.js` is responsible for managing the login state by retrieving it from `chrome.storage.local` and displaying the corresponding interface. Using `chrome.storage.local` ensures that student credentials and login status persist across sessions. It also communicates with `background.js` to initiate or terminate monitoring when the student logs in or out. Additionally, following a successful login, `popup.js` injects the `content.js` script into all open tabs, activating monitoring immediately.

4.5 EMP and SEP Integration and Communication

The communication between the EMP and the SEP is fundamental to the system's functionality. The Figure 4.8 represents the communication flow between the EMP and the SEP.

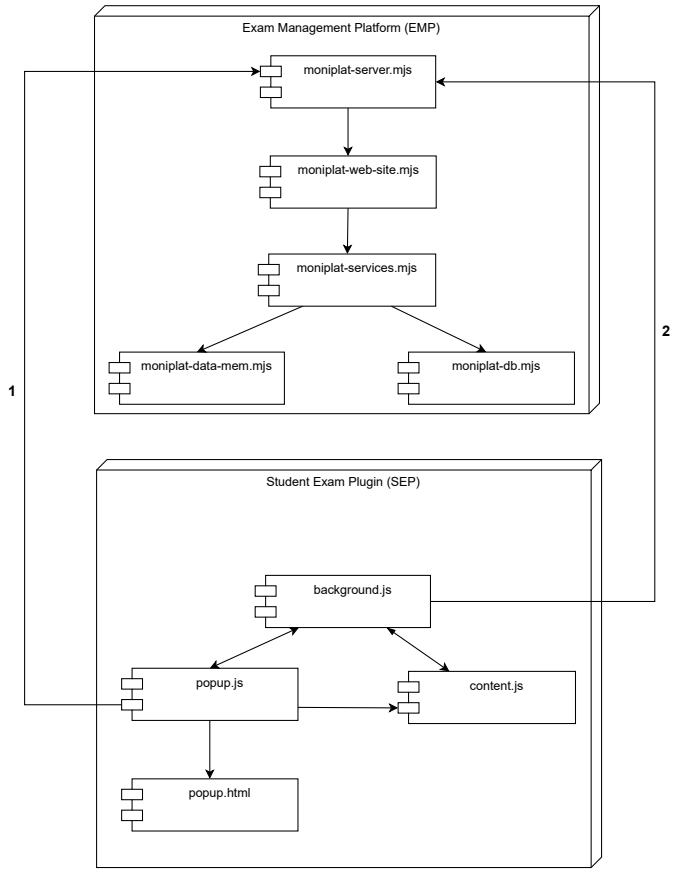


Figure 4.8 UML diagram illustrating communication between the EMP and SEP.

The interactions illustrated in Figure 4.8 can be described as follows:

1 **Login Validation:** The `popup.js` module transmits the student's login credentials to the EMP (`moniplat-server.mjs`) to verify the correctness of the data;

2 **Activity Reporting and Exam URL Retrieval:** The `background.js` module continuously sends requests to the EMP Server (`moniplat-server.mjs`) containing the student's activity data and, initially, to retrieve the URL of the exam for which the student has been authenticated.

From the plugin to the platform, monitoring data is transmitted in two main stages. First, `content.js`, injected into web pages, is responsible not only for placing a green rectangle in the upper-left corner of the student's screen, but also by detecting specific events (such as right-clicks, text selection, and similar interactions) and capturing the corresponding information. This data is then relayed to `background.js` via Chrome's internal messaging sys-

tem (`chrome.runtime.sendMessage`), using the `sendMessageToBackground` function. Once received, `background.js` processes and packages the monitoring data before sending it to the platform's backend via the `notifyAdministrator` function. This function performs HTTP POST requests to the `/student-logs` endpoint using the Fetch API. The transmitted data is structured as a JavaScript Object Notation (JSON) object containing the event type (type), student and exam identifiers (studentId, examId), desk code (deskCode), the hashed authentication token (auth), timestamp, and an event-specific payload (e.g., the visited URL, copied text, or the reason for the alert). Once received, the platform processes this data and, if necessary, can trigger administrator notifications, as handled by the `site.notifyAdministrator` function in `moniplat-server.mjs`.

In the opposite direction, from the platform to the plugin, essential information, such as the authorised URL for a specific exam, is retrieved directly from the server. This is achieved through the `fetchAllowedUrl` function in `background.js`, which performs a GET request to obtain the data.

This mechanism allows the platform to dynamically define the URLs permitted during an exam, with the plugin enforcing compliance on the client side. Authentication tokens (auth) play a critical role in all communications from the plugin to the platform. Stored securely in `chrome.storage.local` upon successful login, the token is included in each request to verify the identity of the sender. Security is further reinforced through the use of SHA-256 hashes for validate the integrity of the plugin.

4.6 Security Measures

Ensuring the security of the platform is crucial, particularly regarding user authentication, data integrity, and exam monitoring. A multi-layered security approach is adopted, combining technological safeguards with procedural controls to prevent unauthorized access and misconduct during examinations.

For user authentication, the system leverages `Passport.js` middleware to verify the identity of users and establish secure sessions. Teachers authenticate using a username and password, while Supervisors authenticate using their email address. In both cases, a Two-Factor Authentication is enforced by sending a six-character verification code to the user's email, which must be entered to complete the login process. Session cookies for Teachers and Supervisors are configured with a maximum age of seven days, balancing usability with security.

Student authentication, managed through the plugin, follows a distinct mechanism where the session has a maximum age of ten seconds. This short duration is designed to mitigate fraud and is automatically renewed through periodic communication between the plugin and the server, such as the transmission of activity logs. This approach enhances security while managing user access. To further protect against unauthorized access, session expiration policies are enforced, requiring users to periodically reauthenticate.

The Exam Management Platform is hosted online at `https://exam-monitoring-platform.onrender.com` and served over HTTPS, ensuring encrypted communication between the client and the server. This prevents unauthorised interception of authentication credentials, monitoring logs, and other sensitive information. Additionally, every request sent from the SEP to the platform includes an authorisation token, which ensures that only legitimate, authenticated requests are processed, mitigating the risk of forged or unauthorised requests.

To further protect against code tampering, the student authentication process includes the generation and transmission of a hash value created from the combination of the SEP source code and the plugin's unique identifier. This mechanism allows the server to verify the authenticity of the plugin in use. If the plugin's source code is altered, the resulting hash will differ from the expected value, causing authentication to be denied and preventing the use of modified or unauthorised plugin versions.

To improve security and prevent reverse engineering, obfuscation would have been a stronger option for the plugin's code. However, as the Chrome Web Store prohibits extensions with obfuscated code, minimization was adopted as an acceptable alternative. Although not as effective as obfuscation, minimized code is harder to read, providing a limited level of protection.

Additionally, RBAC ensures that students, teachers, and supervisors have only the necessary permissions required for their respective tasks, reducing the risk of unauthorized data manipulation.

Another critical security concern is verifying that students are physically present in the exam room. A potential risk is that a student could share their desk code with an external individual, allowing them to take the exam on their behalf. In such cases, the monitoring interface would falsely indicate the student's presence.

To mitigate this issue, an additional visual authentication indicator has been implemented. Upon authentication via the SEP, a green rectangle appears on the student's screen as a visual confirmation of their logged-in status, as shown in Figure 5.6. This rectangle remains visible throughout the exam and disappears only when the student logs out.

For this security measure to be effective, supervisors must physically verify that all students have the green rectangle displayed on their screens before the exam begins. Since the indicator is tied to an authenticated session, any student attempting to bypass the system by logging in remotely or switching users would lose the visual confirmation, alerting exam supervisors. This approach combines automated security controls with human supervision, reinforcing the reliability of the monitoring process.

Despite these safeguards, it is important to acknowledge that students may attempt to find new ways to circumvent security measures. If additional vulnerabilities are identified, further enhancements will be developed to strengthen exam security and maintain the integrity of the monitoring system.

4.7 Implementation Challenges and Solutions

Building a distributed monitoring system comprising web platform components and a browser extension involved addressing several technical challenges.

One of the main challenges was synchronising activity across multiple browser tabs and windows for monitoring purposes. Capturing all relevant events, regardless of which tab or window was active, required a consistent state management strategy. This was achieved by centralising the monitoring logic in `background.js`, which listens for global Chrome events such as `chrome.tabs.onActivated` and `chrome.windows.onFocusChanged` to detect changes in focus and tab activity. Meanwhile, `content.js` operates within each web page to detect local events and communicates with `background.js`. Shared state, including login details, is maintained using `chrome.storage.local`.

Performance was another key consideration. Continuous monitoring, if not managed carefully, can impact both browser responsiveness and backend load. To address this, cooldown intervals were implemented in `content.js` (e.g., `KEYBOARD_COOLDOWN`, `RIGHT_CLICK_COOLDOWN`) to limit the frequency of logs. In `background.js`, periodic checks are handled using a `monitoring Interval` rather than frequent event listeners. On the server side, connection pooling with `pg.Pool` and a layered architecture help manage concurrent requests efficiently.

Data security was also a core requirement. All communication between the plugin and the platform takes place over HTTPS (`https://exam-monitoring-platform.onrender.com`) to ensure encrypted data transmission. Authentication tokens (auth) are issued by the server upon login and included in each subsequent request, verifying the identity of the sender and preventing unauthorised access. In addition to securing communication, measures were taken to preserve the integrity of the plugin itself. A SHA-256 hash is computed using the source code of the plugin's `popup.js` file combined with the plugin ID (via `getPopupJsSourceHash`). This hash

is sent to the server and verified using the `validatePlugin` function in `moniplat-services.mjs`. This mechanism ensures that the plugin's code has not been altered after deployment, helping detect tampering or unauthorised modifications.

Finally, managing access control for different user roles (Students, Teachers, and Supervisors) posed its own challenges. This was handled using `Passport.js` for authentication and session control. Route protection is enforced through role-based checks, such as `checkAuthorizedTeacherVirtualRoom` in `moniplat-db.mjs`, ensuring that only authorised users access sensitive data.

Chapter 5

EMP and SEP Usage and Workflow

This chapter provides a detailed walkthrough of the Exam Management Platform and Student Exam Plugin. It outlines each phase of the exam process, from account creation and exam preparation to student interaction with the plugin and the real-time supervision of students during an examination, highlighting the responsibilities of each actor involved.

Section 5.1 explains how to Create an Account on the platform. It details the information required for registration and the two-factor authentication process for teachers.

Section 5.2 describes the Exam Preparation phase. It walks through the steps a teacher must follow to set up an exam, including creating a virtual room and adding students and supervisors.

Section 5.3 focuses on Student Interaction with the Plugin, detailing the authentication process and how students use the plugin to connect with a physical and virtual seat.

Section 5.4 highlights the roles and responsibilities of Supervisors in monitoring the examination and intervening when necessary.

Section 5.5 explains the concept of Virtual Rooms and how they are used to host students and generate unique desk codes.

Section 5.6 delves into the Real-time Monitoring and Logs feature, describing the monitoring view for supervisors and how event logs are generated and stored.

Lastly, Section 5.7 presents the complete Exam Execution workflow, from the student's authentication to the plugin's real-time monitoring capabilities and event detection.

The plugin is available on the Chrome Web Store at `https://chromewebstore.google.com/detail/exam-monitoring-plugin/hngoklcnpkekadcackclnlkocibaijil`, and the live application can be accessed at `https://exam-monitoring-platform.onrender.com/`.

5.1 Create account

Only teachers are required to create an account on the Exam Management Platform. This account allows them to create and manage exams, as well as supervise exams to which they have been added by other teachers.

To register, the teacher must complete the form shown in Figure 5.1, providing the following information:

- **User Name**: The chosen username for accessing the platform;
- **Password**: A password defined by the teacher;
- **Confirm Password**: A repeated entry of the password to ensure accuracy;
- **Email**: A valid institutional email address ending in `@isel.pt`. Only emails with this domain are accepted.This restriction ensures that only ISEL teachers can create an account. This mitigates the risk of students creating accounts to access the platform, create exams, and attempt to find vulnerabilities in the Student Exam Plugin.

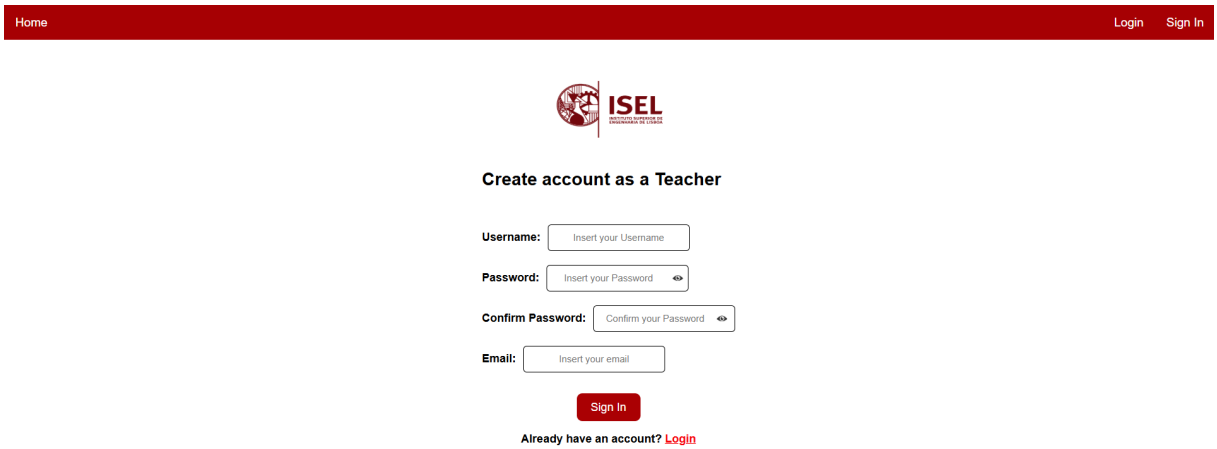
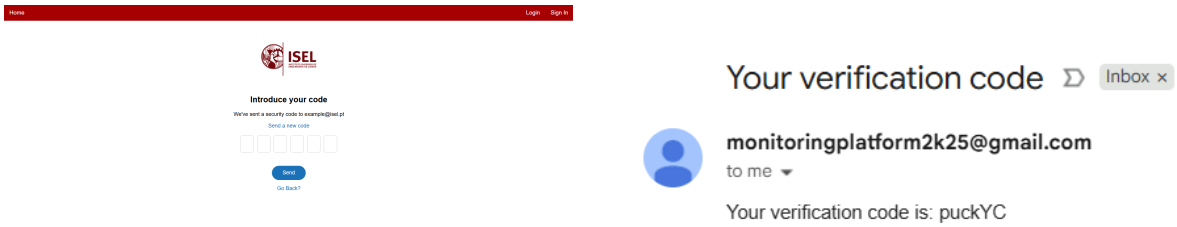


Figure 5.1 Account creation form for teachers.

Upon submitting the form, a six-character verification code is sent to the teacher's institutional email as a form of two-factor authentication.

The teacher must then enter this code into the platform, as illustrated in Figure 5.2. If the code is correct, the account is created, the user is authenticated and he is redirected to the homepage.



(a) Verification code sent to institutional email.

(b) Message received by the user containing the verification code.

Figure 5.2 Two-factor authentication via institutional email during account creation.

After creating an account and logging in, the teacher is authenticated and can access both the exams they have created (initially empty for new accounts) and any exams where they have been assigned as a supervisor by another teacher.

If a teacher registers using an institutional email address that had previously been added as a supervisor (without an associated account), the system automatically transfers the list of supervised exams to the new teacher account. These exams remain associated with the user in a supervisory role. After this transfer, the temporary supervisor account is deleted, ensuring no duplication or inconsistency in access control.

5.2 Exam Preparation

During the exam preparation process, the teacher must first create an account on the platform. This grants them access to the exam creation functionalities. To create an exam, the teacher must complete the form shown in Figure 5.3, providing the following information:

- **Title of the Exam**: Specifies the title of the examination;
- **Date of the Exam**: Indicates when the exam will take place;
- **Exam URL**: The URL where the exam will be available;
- **List of Students**: Identifies all students who are eligible to take the examination;
- **Additional Supervisors**: Lists other teachers/supervisors, apart from the teacher who is creating the exam, responsible for monitoring the exam. Supervisors can be added either by email or by username. Only email addresses with the `@isel.pt` domain are accepted. If the email is valid and not yet associated with an account the system automatically creates a supervisor account associated with that email. If the email matches an existing teacher or supervisor account, the exam is added to that user's list of supervised exams. Alternatively, if the supervisor is already a registered teacher, they can be added directly by entering their username;
- **Virtual Room Layout**: Defines the dimensions of the virtual rooms (e.g., 5x5, 6x5, etc.), ensuring that the physical and virtual seating arrangements correspond.

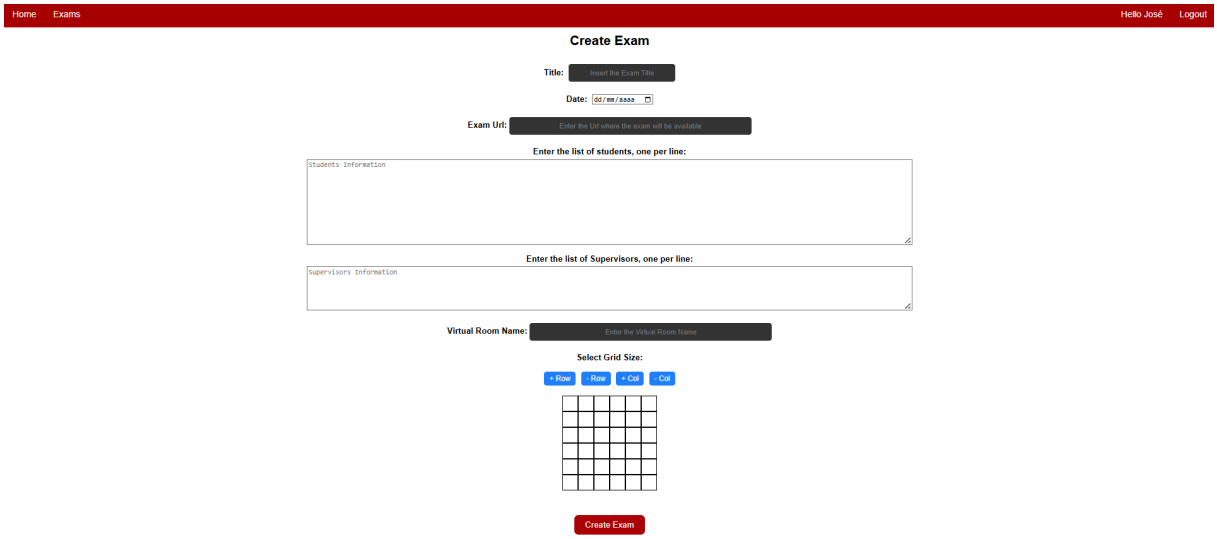


Figure 5.3 Exam creation form.

Once the form is completed, the exam is created and ready for use. The system automatically generates unique codes corresponding to each "position" in the virtual room, as shown in Figure 5.4. These codes must then be printed by the teachers or supervisors and placed on each desk in the physical examination room.

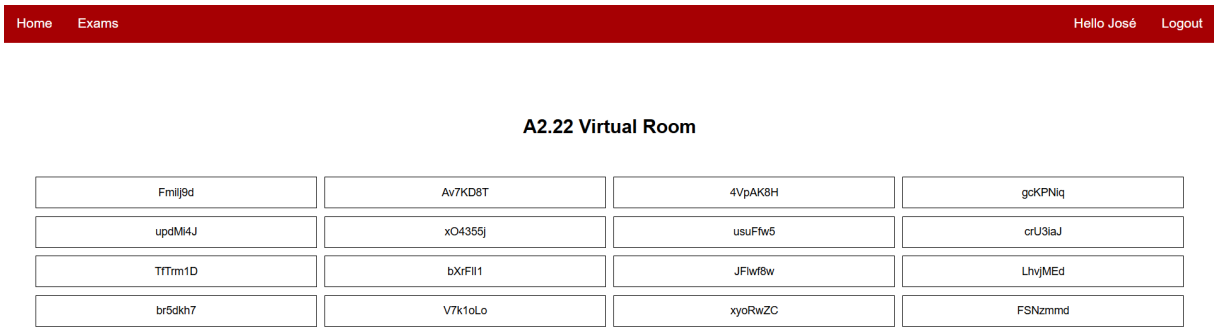


Figure 5.4 A2.22 Virtual Room with a 4x4 layout.

5.3 Student Interaction with the Plugin

In order to take the exam, students are required to install the monitoring plugin on their Google Chrome browser. The plugin can be found at `https://chromewebstore.google.com/detail/exam-monitoring-plugin/hngoklcnpkekadcackclnlkocibaijil`.

At the physical location of the examination, each seat is identified with a unique code, created by the EMP. Students select their desired seats and input the corresponding code into the plugin as part of their authentication process as shown in Figure 5.5. This process establishes a link between the student and the physical seat, which is mirrored in the virtual room.

Figure 5.5 Student authentication form.

To authenticate, students provide their identifier (e.g., student number), the exam ID, provided by the teacher, and the unique seat code. After successful authentication, a green rectangle appears on their screen as shown in Figure 5.6, indicating that they are authenticated. This visual indicator remains visible throughout the exam, disappearing only when the student logs out.

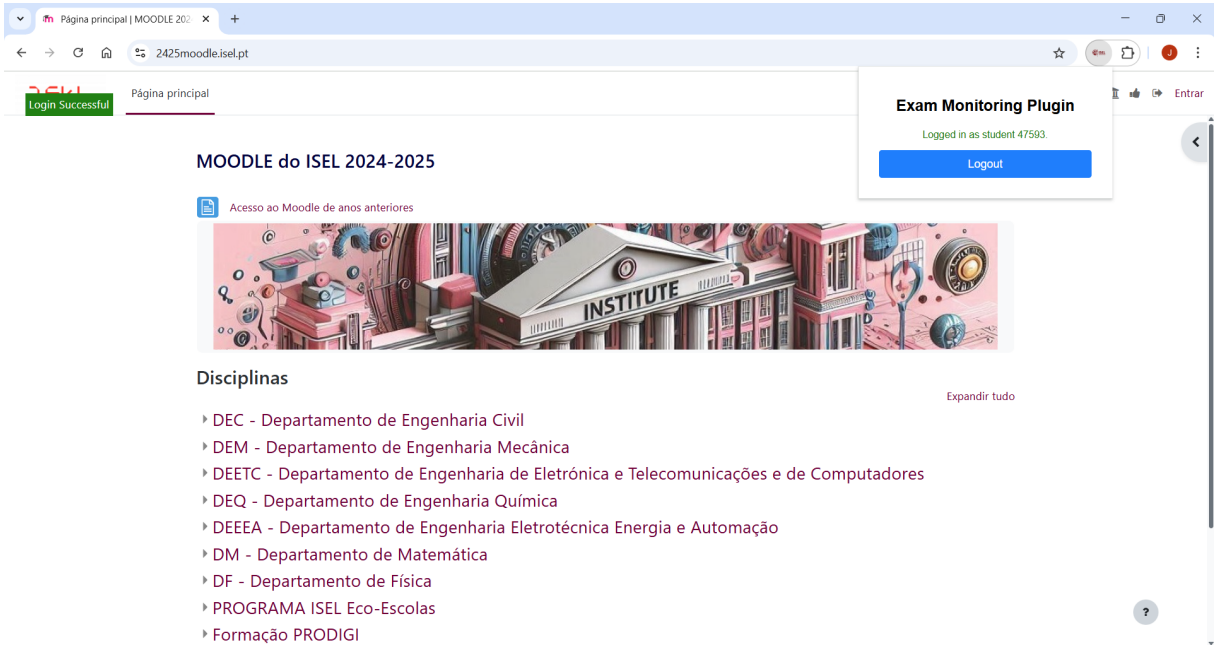


Figure 5.6 Student's view after logging in successfully.

5.4 Supervisors and their Roles

Supervisors play a crucial role in ensuring the integrity of examinations. Their responsibilities include:

- Monitoring student activity: Supervisors are tasked with overseeing real-time student behaviour to identify suspicious activities. This involves tracking browser usage, monitoring screen activity, and ensuring compliance with examination guidelines;
- Intervening when necessary: Whenever irregularities are detected, supervisors should go to the physical seat indicated by the virtual room and intervene appropriately;
- Coordinating with other supervisors: For large-scale examinations, multiple supervisors may be assigned. Effective communication between supervisors ensures that all participants are adequately monitored.

5.5 Virtual Rooms

Virtual rooms serve as the digital equivalent of physical examination rooms. These rooms are designed to:

- Host Students: Each virtual room can accommodate a set number of students. For example, a 5x6 virtual room can hold up to 30 students;
- Generate Desk Codes: The system generates unique codes for each seat, which are used for student authentication, ensuring that the physical and virtual seating arrangements correspond.

An exam can have more than one virtual room. To create a virtual room, you need to provide the room's name and its size (for example, 4x5, 5x6), as shown in Figure 5.7. The flexibility of virtual rooms allows institutions to adapt the platform to various examination scenarios, ranging from small quizzes to large-scale final exams.

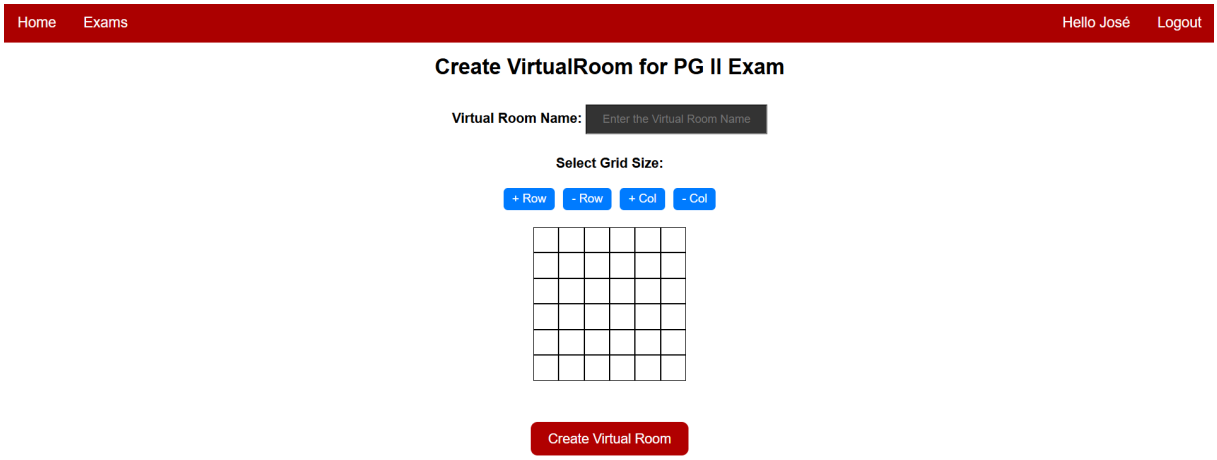


Figure 5.7 Create Virtual Room form.

5.6 Real-time Monitoring and Logs

5.6.1 Monitoring View

The Monitoring View is a crucial component of the EMP, providing exam supervisors with real-time data about each student's activity. This interface allows supervisors to monitor the progress of the exam, viewing which students are currently taking the test and receiving alerts if any irregular behaviour is detected. Examples of monitored behaviours include:

- **Access to unauthorised URLs**: The plugin detects and reports navigation to websites not assigned for the exam, including the specific URLs accessed;
- **Navigation to permitted URLs**: Authorised navigations are logged to confirm correct exam access;
- **Loss of window focus**: If a student switches to another application or minimises the browser, an alert is triggered;
- **Browser not maximised or fullscreen**: The plugin identifies when the browser is resized out of the required viewing state, which may indicate attempts to hide activity;
- **Right-click events**: Any use of the context menu, the menu that appears when a user right-clicks, is logged, helping to prevent actions such as using Google Lens;
- **Text selection**: The plugin records when a student selects text exceeding the defined threshold of 10 characters, which may suggest an attempt to copy content;
- **Plugin integrity violations**: The integrity of the plugin is verified during initialisation. If any core file, such as `popup.js`, is altered, the system either blocks login or generates an alert;
- **Login and logout events**: Student authentication and session termination are recorded, marking the start and end of monitoring.

Alerts generated by the plugin notify teachers and supervisors on the platform, enabling prompt action. Supervisors can physically visit the student's seat to investigate and address any potential issues, ensuring compliance with examination protocols.

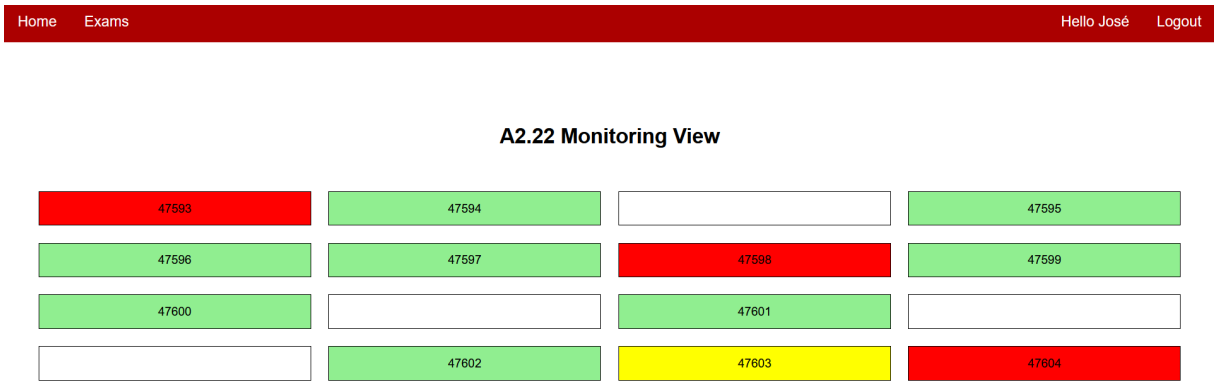


Figure 5.8 Monitoring View of a Virtual Room with a 4x4 layout.

Figure 5.8 illustrates a demonstration of the monitoring view of a 4x4 virtual room. We can observe rectangles in four colours: green, red, white and yellow. The white rectangles represent empty seats with no students assigned. The green, red and yellow rectangles, on the other hand, represent students, with the content of each rectangle being the student's identifier, in this case, the student number. The colour of the rectangle indicates the student's status. Students with green rectangles are not committing any irregularities, while those with red rectangles are engaged in some form of irregularity at that specific moment. A yellow rectangle indicates a student who attempted to log in using a modified plugin. This yellow rectangle is displayed for only ten seconds before disappearing. Supervisors should then proceed to the physical seat indicated by the red and yellow rectangles and take appropriate action.

This real-time monitoring is essential to maintaining the integrity of the exam process, ensuring that all students are adhering to the rules and that any suspicious behaviour is swiftly resolved.

5.6.2 Logs

In addition to visual alerts, the system stores logs generated in the virtual room in its database throughout the exam, recording the timestamp at which each event occurs while the button shown in Figure 5.9 is switched on.

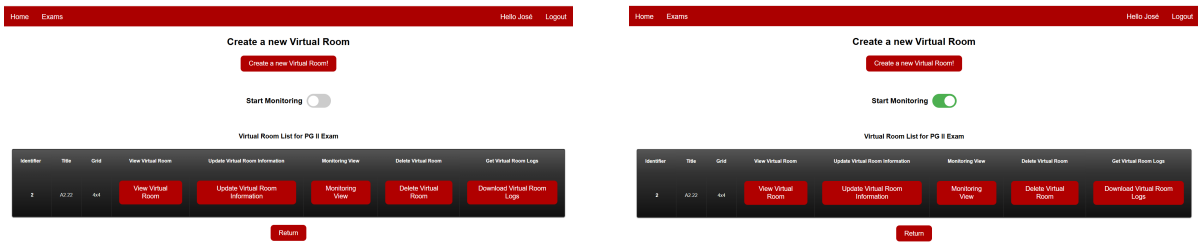


Figure 5.9 Monitoring Button.

Only the teacher responsible for the exam has access to the logs of virtual rooms. To access these logs, the teacher must click the “Download Virtual Room Logs” button, as shown in Figure 5.10.

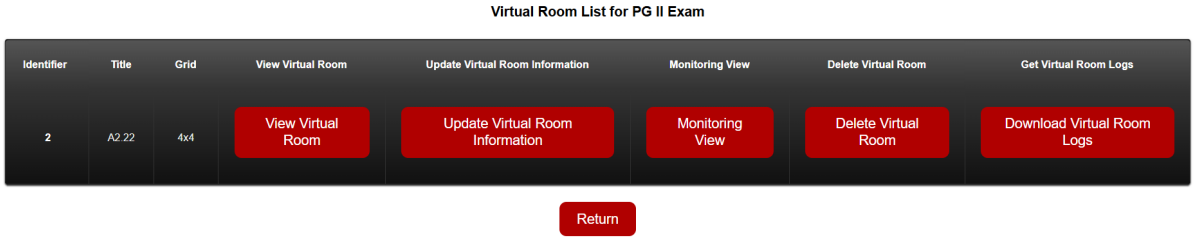


Figure 5.10 PGII Exam Virtual Rooms.

Upon clicking the "Download Virtual Room Logs” button, a .txt file containing the logs associated with that virtual room is downloaded. Each log entry includes the timestamp, the type of event, and the identifier of the student who triggered it. The Figure 5.11 shows the logs of the A2.22 Virtual Room. These logs are essential for post-exam analysis and for documenting potential misconduct.

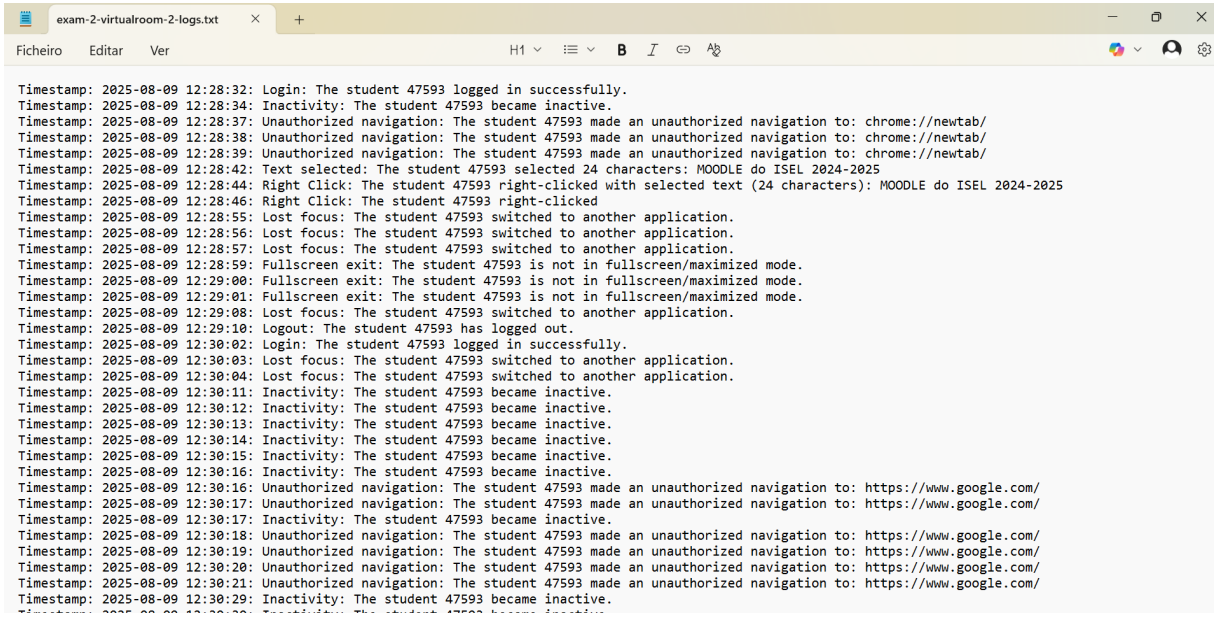


Figure 5.11 A2.22 Virtual Room Logs.

5.7 Exam Execution

On the day of the examination, teachers will print the seat codes in the virtual room, as shown in Figure 5.4, and place them on the corresponding physical seats. Each student will then sit at a desk with an assigned code. Using the SEP, the student must complete four fields in the plugin interface: the exam identifier (provided by the teacher), the student identifier, the desk code, and a confirmation of the desk code. This process creates an association between the student and the seat, as shown in Figure 5.5.

Once these values are submitted, the SEP sends a request to the EMP to validate the association. The system checks whether the desk code is valid and matches the confirmation, whether the student is on the authorised list, and whether the exam identifier is correct. The EMP also checks the integrity of the plugin. If validation is successful, a green rectangle appears on the screen, as shown in Figure 5.6, confirming the student's authentication and initiating the monitoring process.

During the examination, the SEP actively monitors the student's browser activity. It detects events such as navigation to permitted or unauthorised URLs, including the exact address accessed, loss of window focus when the student switches to another application, and cases where the browser is not maximised or in fullscreen mode. The plugin also logs right-click actions, which may indicate attempts to use tools like Google Lens, and monitors any text selection that exceeds ten characters. Additionally, the SEP verifies the integrity of the plugin to ensure that it has not been altered. If the student is using a modified version of the plugin, authentication is rejected, and the teacher is immediately notified that the student attempted to access the system using an unauthorised version.

All detected events are timestamped and sent in real time to the EMP.

Chapter 6

Testing and Evaluation

The testing and evaluation phase is a key part of the development process, as it demonstrates the system's functionality, reliability, and security. It confirms the system's ability to perform as intended and identifies potential vulnerabilities.

This chapter details the testing methodology applied to both the Exam Management Platform and the Student Exam Plugin. It outlines the specific tests conducted, presents their results, and discusses the conclusions and limitations identified.

Section 6.1 presents the testing methodology adopted, describing the unit, system, and controlled environment testing stages.

Section 6.2 introduces the test cases and results for both the EMP and the SEP, highlighting expected and observed behaviours.

Section 6.3 provides an analysis of the results, identifying the system's strengths, limitations, and areas for improvement.

6.1 Testing Methodology

To verify the quality and functionality of the Exam Management Platform and the Student Exam Plugin, several types of tests were performed:

- **Unit Testing**: These tests focused on individual components of the system. All functions in the `moniplat-services.mjs` module were covered by unit tests to check their expected behaviour;
- **System Testing**: This involved simulating a complete exam scenario with one teacher and one student to ensure the entire system functions correctly. The tests covered the full process, from the teacher creating an account and setting up the exam to the student logging into the plugin, the logging of actions and their corresponding display in the exam's monitoring view. These tests aimed to verify that the complete system met the required functional and security specifications;
- **Controlled Environment Testing**: The system was tested in a simulated, controlled environment involving multiple participants. Three classmates were invited to intentionally

attempt actions that the plugin might not detect. This approach allowed for the identification of potential gaps in detection capabilities that were not revealed during previous testing stages. It also helped to assess the system's ability to handle multiple simultaneous requests and log all student actions in a realistic exam setting.

6.2 Test Cases and Results

This section presents the testing procedures and results for the Exam Management Platform and the Student Exam Plugin. Testing was conducted at multiple levels, including unit tests on the `moniplat-services.mjs` module, system tests of the main EMP workflows, SEP tests and controlled environment tests. These tests aimed to verify the functionality, reliability, and security of the platform and its components.

6.2.1 Unit Testing

Unit tests were performed to verify the functionality of individual components within the EMP. The tests focused on the `moniplat-services.mjs` module, covering user authentication, exam management, and virtual room operations. The purpose of these tests was to ensure backend reliability. All tested functions behaved as expected. A total of 107 unit tests were performed, and all passed. Figure 6.1 presents the unit test results.

```
validateVerificationCodeTeacher test
  #Test validateVerificationCodeTeacher with an invalid verification code.
    ✓ should return an error

validateVerificationCodeTeacher test
  ✓ It should return the user information

validateVerificationCodeSupervisor test
  #Test validateVerificationCodeSupervisor with an invalid verification code.
    ✓ should return an error

validateVerificationCodeSupervisor test
  ✓ It should return the user information

validatePlugin test
  ✓ It should return true

getMonitoringStatus test
  ✓ It should return the monitoring status of the specified exam.

getMonitoringStatus test
  #Test getMonitoringStatus when the examId is invalid.
    ✓ should return an error

updateMonitoringStatus test
  ✓ It should return the updated monitoring status of the specified exam.

updateMonitoringStatus test
  #Test updateMonitoringStatus when a teacher tries to access an exam he is not assigned to
    ✓ should return an error


107 passing (116ms)
```

Figure 6.1 Unit Tests results.

6.2.2 EMP Testing

System testing was performed to evaluate the main workflows of the EMP. Test cases addressed teacher and supervisor account management, the exam lifecycle, and virtual room operations. The goal was to verify that the platform correctly handled all expected actions and displayed results accurately in the monitoring view.

The test cases, with expected and observed results, are in Table 6.1. All scenarios were executed successfully. This confirms that the EMP provides a secure authentication process, allows for full lifecycle management of exams and virtual rooms, and offers supervisors real-time monitoring capabilities during exam sessions.

Table 6.1 EMP Functionality Test Cases

Test Case	Scenario Description	Expected Outcome	Observed Outcome
Sign In Teacher	A new user provides valid credentials and institutional email (@isel.pt).	A verification code is sent to the email. After entering the code, the account is created, the user is authenticated and he is redirected to the homepage.	Success: The new teacher account is created, the teacher is authenticated, and he is redirected to the homepage.
Login Teacher	A teacher enters their credentials on the login page.	A verification code is sent to their associated email address. After the teacher enters the correct code, he is authenticated and redirected to the homepage.	Success: The teacher is logged in and redirected to the homepage.
Login Supervisor	A supervisor enters their institutional email to log in.	A verification code is sent to the supervisor's email. After entering the code, the supervisor is authenticated and redirected to the homepage.	Success: The supervisor is successfully authenticated and redirected to the homepage.

Continued on next page

Table 6.1 – Continued from previous page

Test Case	Scenario Description	Expected Outcome	Observed Outcome
Forgot Password	A teacher provides their registered institutional email to reset password.	A password reset email with a unique token is sent to the teacher's email address. To reset their password, the teacher clicks the link in the email, which directs him to a password change page. Here, he can enter and confirm his new password. Once submitted, the old password is replaced with the new one, completing the process.	Success: The teacher receives the password reset email, follows the link to the password change page, and successfully sets a new password, replacing the old one.
Get All Exams	A logged-in teacher navigates to the exams page.	A list of all exams created by the teacher is displayed.	Success: All exams created by the teacher are listed.
View Supervised Exams	A logged-in teacher or supervisor can view all exams they are assigned to supervise on the Supervised Exams page.	A list of all exams where the user is a supervisor is displayed.	Success: The complete list of supervised exams is successfully displayed.
Create Exam	A teacher fills out exam creation form (title, date, URL, students, supervisors, virtual room name, virtual room layout).	The new exam is successfully created and added to the list of exams.	Success: The new exam appears in the teacher's exam list and can be managed by the teacher.
Update Exam	A teacher modifies the details of an existing exam.	The exam details are updated successfully in the system.	Success: The changes to the exam are saved and are reflected in the exam details view.

Continued on next page

Table 6.1 – Continued from previous page

Test Case	Scenario Description	Expected Outcome	Observed Outcome
View Exam Details	A teacher selects an exam from the list.	The detailed information for that exam, excluding assigned virtual rooms, is displayed.	Success: The selected exam's details are shown correctly, without the associated virtual rooms.
Delete Exam	A teacher chooses to delete an exam.	The system updates the exam and its associated virtual rooms by setting their active flag to 0, preventing them from being displayed.	Success: The exam and its associated virtual rooms are flagged as inactive and no longer visible to the teacher.
Get all Virtual Rooms	A teacher or supervisor navigates to the virtual rooms page for a specific exam.	The system displays a list of all active virtual rooms associated with that exam.	Success: All active virtual rooms linked to the exam are listed.
View Virtual Room	A teacher or supervisor selects a virtual room from the virtual room list.	The virtual room's grid layout, which includes all corresponding desk codes, is displayed.	Success: The virtual room is successfully displayed.
Create Virtual Room	A teacher fills out virtual room creation form (virtual room name and grid size).	Virtual room is created with unique desk codes.	Success: The virtual room is created and desk codes generated.
Update Virtual Room	A teacher modifies the virtual room's name or grid size.	The virtual room's information is updated.	Success: The changes are saved, and the virtual room's details are updated.
Delete Virtual Room	A teacher chooses to delete a virtual room.	The system updates the virtual room by setting its active flag to 0, preventing it from appearing in the exam.	Success: The virtual room is marked as inactive and no longer displayed under the exam.
Access Monitoring View	A teacher or supervisor opens the monitoring view of a virtual room.	The system loads the monitoring view with the seat grid and student status (white, green, red or yellow).	Success: The monitoring view displays correctly, showing real-time student status updates.

Continued on next page

Table 6.1 – Continued from previous page

Test Case	Scenario Description	Expected Outcome	Observed Outcome
Download Virtual Room Logs	A teacher downloads logs for a virtual room.	A .txt file containing all student activity logs for that virtual room is downloaded.	Success: The logs are correctly retrieved from the database and downloaded in a .txt file.

6.2.3 SEP Testing

The SEP was tested to verify its detection of unauthorised behaviour and communication of events to the EMP. Test cases included login and logout, navigation to authorised and unauthorised URLs, browser focus changes, window resizing, right-click actions, text selection, and plugin integrity checks.

The outcomes demonstrated that the plugin reliably identified both permitted and unauthorised actions, updated the monitoring view in real time, and ensured that all event logs were correctly transmitted and stored in the EMP. Table 6.2 summarises the test cases, expected results, and observed outcomes.

Table 6.2 SEP Functionality Test Cases

Test Case	Scenario Description	Expected Outcome	Observed Outcome
Student Login and Script Injection	The student enters valid credentials in popup.html and logs in.	The student is successfully authenticated, monitoring is initiated, and a green rectangle is inserted into all open and future tabs. In the monitoring view, the student's corresponding seat must appear green.	Success: The student is successfully authenticated, the green rectangle is inserted into all into all open and future tabs and monitoring logs are now being sent to the server. The student's seat in the monitoring view is marked in green.

Continued on next page

Table 6.2 – Continued from previous page

Test Case	Scenario Description	Expected Outcome	Observed Outcome
Navigation to Permitted URL	The student accesses the URL where the exam is available.	The `background.js` detects the navigation and calls the `notifyAdministrator` function with the action labelled as authorised (action: 'authorised'). In the monitoring view, the student's corresponding seat must appear green.	Success: Navigation is recorded as authorised on the platform. The event log is sent but not saved in the database, and the student's seat in the monitoring view is marked green.
Navigation to Unauthorised URL	The student attempted to access an URL that did not match the one assigned for the exam.	The `background.js` detects the unauthorised navigation and sends a log message with the specific URL by calling the `notifyAdministrator` function with the action labelled as unauthorised (action: 'unauthorised'). In the monitoring view, the student's corresponding seat must appear red.	Success: Navigation is recorded as unauthorised. The event log was also sent and saved to the database, and in the monitoring view the student's seat color was changed to red.

Continued on next page

Table 6.2 – Continued from previous page

Test Case	Scenario Description	Expected Outcome	Observed Outcome
Loss of Window Focus	The student switches to another application.	The `monitoringInterval` in `background.js` detects the loss of focus and sends a log message by calling the `notifyAdministrator` function with the action labelled as unauthorised (action: 'unauthorised'). In the monitoring view, the student's corresponding seat must appear red.	Success: Loss of focus is recorded as unauthorised. The event log is sent and saved to the database, and in the monitoring view the student's seat colour is changed to red.
Browser Not Maximised/ Fullscreen	The student resizes the browser window out of the maximised/fullscreen state.	The `monitoringInterval` in `background.js` detects that the window is not maximised or in fullscreen mode and sends a log message by calling the `notifyAdministrator` function with the action labelled as unauthorised (action: 'unauthorised'). In the monitoring view, the student's corresponding seat must appear red.	Success: The window state is recorded as unauthorised. The event log is sent and saved to the database, and in the monitoring view the student's seat colour is changed to red.

Continued on next page

Table 6.2 – Continued from previous page

Test Case	Scenario Description	Expected Outcome	Observed Outcome
Right-Click Detection (Google Lens Prevention)	The student performs a right-click action, which is detected to prevent the use of tools like Google Lens.	The `content.js` detects the contextmenu event. The log message is sent to `background.js`, which calls `notifyAdministrator` function with the action labelled as unauthorised (action: 'unauthorised'). In the monitoring view, the student's corresponding seat must appear red.	Success: The right-click event is recorded as unauthorised. The event log is sent and saved to the database, and in the monitoring view the student's seat colour is changed to red.
Text Selection	The student selects a portion of text exceeding 10 characters.	The `content.js` detects the mouseup event when a user selects text. A log message with the selected text is sent to `background.js`, which calls the `notifyAdministrator` function with the action labelled as 'unauthorised' (action: 'unauthorised'). In the monitoring view, the student's corresponding seat must appear red.	Success: The mouseup event caused by text selection is recorded as unauthorised. The event log, including the selected text, is sent and saved to the database, and in the monitoring view the student's seat colour is changed to red.

Continued on next page

Table 6.2 – Continued from previous page

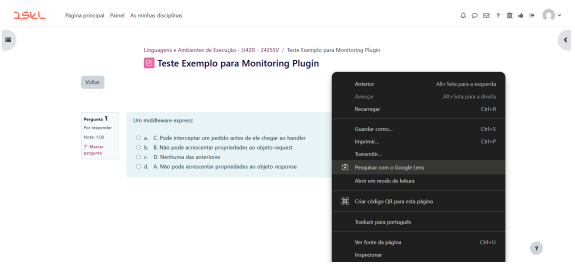
Test Case	Scenario Description	Expected Outcome	Observed Outcome
Student Logout	The student clicks the logout button in the popup.	The student's session is terminated and the monitoring is stopped. The student is removed from the monitoring view, and their seat changes to white.	Success: Upon logout, monitoring for the student is immediately deactivated. The Logout event is recorded, the log is sent and saved to the database, and the student is removed from the monitoring view, with their seat colour changed to white.
Plugin Integrity Validation	An attempt is made to modify the plugin's source code and log in using the plugin with the altered code.	During initialisation, if the hash sent by the plugin does not correspond to the one stored on the server, authentication fails. The server then generates a log message stating that the student attempted to access using an invalid plugin. The system prevents the student from logging in, and in the monitoring view, the seat designated for the desk code he entered is marked in yellow.	Success: The invalid plugin hash is detected during initialisation, preventing the student from logging in. The event log is sent and saved to the database, and in the monitoring view, the seat designated for the desk code entered is marked in yellow.

6.2.4 Controlled Environment Testing

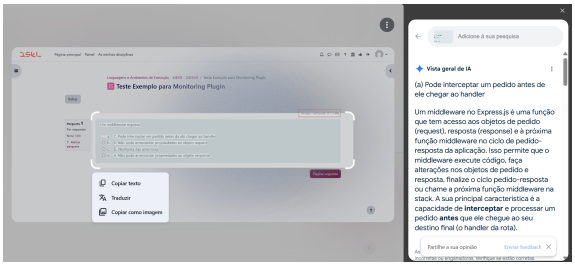
Controlled environment testing was conducted with three classmates to evaluate whether the system could handle multiple simultaneous requests and to identify potential gaps in detection capabilities that had not been revealed during unit and system testing.

During this stage, students were asked to deliberately attempt actions that might not be captured by the SEP.

One such case involved using Google Lens through the right-click context menu. When the student performed a right-click, the browser's context menu appeared, which included the option to Search with Google Lens. If selected, a window opened on the right-hand side of the screen displaying the Google Lens interface, as shown in Figure 6.2. This vulnerability was subsequently mitigated by implementing right-click detection in the plugin, ensuring that such actions are now registered as unauthorised and reflected in the monitoring view.



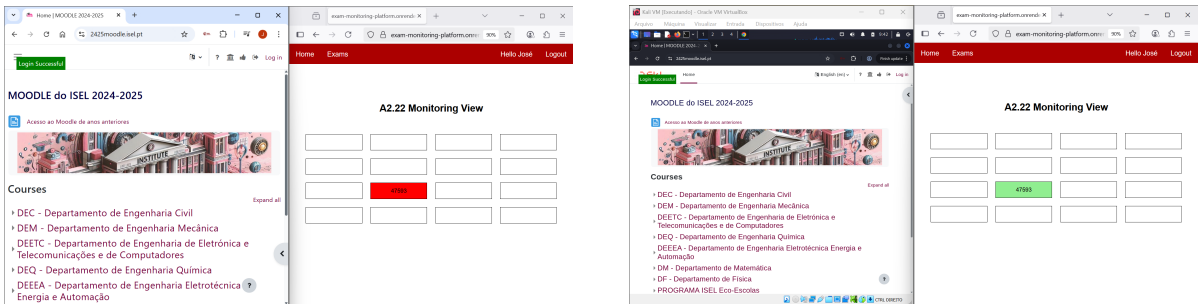
(a) Browser's context menu with the option to "Search with Google Lens".



(b) Google Lens interface.

Figure 6.2 Using Google Lens through the right-click context menu.

Another limitation identified was the use of Virtual Machines (VMs). When the plugin runs inside a VM, the plugin registers the virtual environment as being constantly in the foreground. This occurs because the host operating system views the entire VM as a single, full-screen application. Consequently, the plugin cannot detect if a student minimizes the VM to access local files or browse outside the VM, as illustrated in Figure 6.3. As this limitation cannot be resolved with software, it must be addressed through procedural measures. Teachers and supervisors must conduct in-person inspections. When performing these checks, they must not only confirm that the student is logged into the plugin, as indicated by the green rectangle in the upper-left corner of the screen, but also ensure the student is not running the exam within a virtual machine.



(a) Split screen without using a virtual machine.

(b) Split screen using a virtual machine.

Figure 6.3 Comparison of screen splitting with and without a virtual machine.

Apart from this, the initial plugin implementation was not configured to detect when the browser window was not maximised. As a result, a student could split the screen and place a Portable Document Format (PDF) or any other document alongside the exam window. Provided they only scrolled through the document without clicking on it, the plugin would not register any irregularity, since the window focus was never lost.

All other tested scenarios, including attempts to manipulate the plugin through Chrome's extension settings, such as enabling or disabling options like file URL access or incognito mode, were handled correctly, as shown in Figure 6.4. In each case, the plugin continued to operate as expected, detecting irregularities and transmitting events to the EMP.

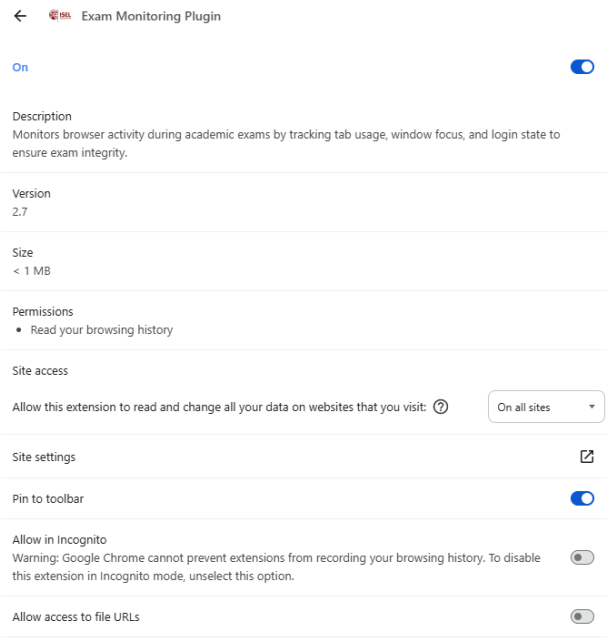


Figure 6.4 Disable extension settings.

These controlled environment tests confirmed the platform's ability to handle concurrent requests and highlighted both strengths and inherent limitations of the monitoring approach.

6.3 Analysis of Results

The results from the different testing stages confirmed that the EMP performs the intended monitoring and management tasks.

Unit and EMP Testing confirmed the stability and reliability of the core components. The 107 unit tests on the `moniplat-services.mjs` module validated the backend's functionality. The system tests of the EMP's workflows, as detailed in Table 6.1, confirmed that the platform correctly handled all expected actions.

The system tests of the SEP, as detailed in Table 6.2, showed that the Student Exam Plugin consistently detected and reported unauthorised behaviour. The plugin responded to actions such as navigation to unauthorised URLs, focus loss, window resizing, right-click use, text selection, and attempts to modify plugin code. Each case was logged in real time and reflected in the monitoring view. The integrity check based on hash validation blocked altered versions of the plugin and prevented access when tampering was detected.

During the controlled environment test, some ways of bypassing the plugin were identified. For example, the system was not initially configured to detect when the user accessed the context menu to use tools such as Google Lens. Likewise, it was not configured to detect when the browser window was not maximised, allowing the student to split the screen and keep a PDF or another file open alongside the exam window. As long as the document was only scrolled through without being clicked, the plugin did not register any irregularity, since the window focus was not lost. These vulnerabilities were subsequently addressed, and the plugin was updated to detect and log such actions. A remaining limitation is the inability to detect when the exam runs inside a virtual machine, as the host operating system treats the VM as a single window. This can only be mitigated through procedural measures, such as in-person inspections.

The strengths observed include:

- **Reliable Detection of Key Activities**: The system The system successfully identified and recorded critical events, including navigation to authorised and unauthorised URLs, unmaximized windows, focus changes, text interactions such as text selection, and right-click actions. These behaviours were confirmed during both real exam conditions and controlled testing;
- **Consistent Bidirectional Communication**: Information exchange from the plugin to the platform (logs), and from the platform to the plugin (authorised URLs) was stable and consistent, even under simulated exam conditions;
- **Plugin Integrity Verification**: Hash validation provided an initial layer of protection against unauthorised modification of the plugin code;
- **Responsiveness**: Logs were transmitted and processed with minimal delay, enabling effective monitoring during the exam.

Areas for improvement:

- **Browser Compatibility**: Currently, this plugin is exclusively designed for Google Chrome. A significant enhancement would be to broaden its availability to other widely used browsers, such as Mozilla Firefox and Opera, among others. This expansion would provide greater flexibility and accessibility for users;
- **Network Dependence**: The system's ability to transmit data depends on the student's internet connection. Connectivity issues may lead to temporary loss of monitoring data and can also cause the student's session to be lost;
- **Virtual Machine Vulnerability**: A limitation is the inability of the plugin to detect when an exam is being run inside a VM. The plugin cannot register actions that occur outside the virtual environment. This must be addressed through procedural measures, such as in-person inspections.

Overall, the testing confirmed that the combined operation of the Exam Management Platform and Student Exam Plugin achieves the primary objective of enabling real-time detection and logging of unauthorised actions during online examinations. The backend services, exam management workflows, and monitoring features functioned correctly, while the plugin provided consistent event detection and communication. Limitations remain in relation to network dependence, browser availability, and the inability to detect certain behaviours when students rely on virtual machines, but the system as tested provides a reliable foundation for secure exam monitoring.

Chapter 7

Conclusion

This chapter summarises the problem addressed in this thesis and the proposed solution, providing an overview of the development, evaluation, and outcomes of the project. It also reviews the main testing results and identifies areas for further improvement. Section 7.1 outlines directions for future work to extend the system's functionality.

This master's thesis addresses the challenge of maintaining academic integrity in in-person examinations through the development of the Exam Management Platform (EMP) and the Student Exam Plugin (SEP). The solution leverages student laptops and the Google Chrome browser, eliminating the need for third-party software installations.

The development and subsequent testing confirmed that the system's architecture is both suitable and operational for its intended objectives. Controlled exam simulations validated its core functionalities, demonstrating accurate and reliable monitoring of student activity. The data collected through the SEP provided valuable insight into student behaviour, supporting both real-time supervision and post-exam analysis.

The system's strengths lie in its consistent event detection, secure bidirectional communication, and integrity mechanisms such as hash validation to prevent tampering. These features collectively deliver a practical response to the challenges of digital exam supervision.

While the system successfully meets its primary monitoring requirements, testing identified some inherent limitations and opportunities for future development. The SEP is coupled with Chrome APIs, meaning future browser updates may necessitate code adjustments. Additionally, the plugin's monitoring is limited to the browser context, it cannot detect activities outside of the browser, such as a student consulting physical materials or using mobile devices.

Furthermore, students may be able to bypass the plugin's monitoring by using virtual machines. When the plugin runs inside a Virtual Machine (VM), it can't accurately monitor student activity because the host operating system views the entire virtual environment as a single, full-screen application. This prevents the plugin from detecting if a student minimizes the VM or accesses local files outside of it.

In summary, despite these constraints, the system achieves its core objectives of providing secure, real-time exam monitoring and valuable behavioural insights. The system demonstrates

its potential as a robust tool for enhancing exam integrity in supervised digital assessments.

7.1 Future Work

Future work on the platform should address improvements in usability, scalability, and integration. A first step is the enhancement of the user interface, with the addition of a real-time log view that allows supervisors to follow events as they occur. Filtering mechanisms and clearer alerts for critical situations would also strengthen the monitoring process.

Another area for improvement concerns the management of user data. Allowing teachers to update their own profile information, such as usernames and email addresses, directly within the platform would improve usability. Functional extensions could also include the introduction of a periodic screenshot capture mechanism. Although such a feature would need to be carefully balanced against privacy and storage considerations, it could provide an additional layer of evidence in cases of suspicious behaviour.

From a performance perspective, the system would benefit from further load and scalability testing, particularly in scenarios involving large student groups, ensuring that the system maintains stability under heavier use. On the integration side, adopting a learning technology standard such as xAPI would enable interoperability with learning management systems like Moodle. This approach would allow monitoring events to be recorded in a structured way and stored in a Learning Record Store (LRS) for later analysis.

Finally, the migration of the frontend from Handlebars to React would improve maintainability and support the introduction of more dynamic, component-based features.

Together, these developments would extend the platform's functionality and consolidate the system as a scalable, secure, and adaptable solution for ensuring academic integrity in computer-based examination settings.

Bibliography

[1] Agarwal, S. (2022). Helping or hindering? how browser extensions undermine security. In *Proceedings of the 2022 ACM SIGSAC Conference on Computer and Communications Security*, CCS '22, page 23–37, New York, NY, USA. Association for Computing Machinery.

[2] Atrash, S. (2024). 5 Backend Security Best Practices Every Developer Should Know. `https://medium.com/@sanaatrash09/5-backend-security-best-practices-every-developer-should-know-90469b9f131f`. Accessed on 27/11/2025.

[3] Aurelia, S., Thanuja, R., Chowdhury, S., and Hu, Y.-C. (2023). Ai-based online proctoring: a review of the state-of-the-art techniques and open challenges. *Multimedia Tools and Applications*, 83.

[4] Balta, N., Perera-Rodríguez, V.-H., and Hervás-Gómez, C. (2017). Using socrative as an online homework platform to increase students' exam scores. *Education and Information Technologies*, 23(2):837–850.

[5] Barth, A., Felt, A. P., Saxena, P., and Boodman, A. (2010). Protecting Browsers from Extension Vulnerabilities. *Network and Distributed System Security Symposium*.

[6] Bawarith, R., Abdullah, and Anas (2017). E-Exam Cheating Detection System. *International Journal of Advanced Computer Science and Applications*, 8(4).

[7] Bergmans., L., Bouali., N., Luttikhuis., M., and Rensink., A. (2021). On the efficacy of online proctoring using proctorio. In *Proceedings of the 13th International Conference on Computer Supported Education - Volume 1: CSEDU*, pages 279–290. INSTICC, SciTePress.

[8] Borra, P. (2024). Comparison and analysis of leading cloud service providers (aws, azure and gcp). *INTERNATIONAL JOURNAL OF ADVANCED RESEARCH IN ENGINEERING & TECHNOLOGY*, 15:266–278.

[9] Capris, T., Melo, P., Garcia, N. M., Pires, I. M., and Zdravevski, E. (2022). Comparison of sql and nosql databases with different workloads: Mongodb vs mysql evaluation. In *2022 International Conference on Data Analytics for Business and Industry (ICDABI)*, pages 214–218.

[10] Cyber, B. G. (2024). Authenticator apps vs. sms for two-factor authentication - blue goat cyber. `https://bluegoatcyber.com/blog/authenticator-apps-vs-sms-for-two-factor-authentication-which-is-safer/`. Accessed on 27/11/2025.

[11] Daniel, R. C. and Caleb Andrew, H. (2024). AI-proctored exam portal with mobile companion application. In *2024 3rd International Conference on Applied Artificial Intelligence and Computing (ICAAIC)*, pages 47–51.

[12] de la Torre, L., Chacón, J., Chaos, D., Dormido, S., and Sanchez, J. (2019). Using server-sent events for event-based control in networked control systems. *IFAC-PapersOnLine*, 52:260–265.

[13] Desai, U., Naik, S., Tari, S., Dessai, S., and Shetgaonkar, P. (2024). Unauthorised activity detection during online exam. In *2024 15th International Conference on Computing Communication and Networking Technologies (ICCCNT)*, pages 1–8.

[14] Dilini, N., Senaratne, A., Yasarathna, T., Warnajith, N., and Seneviratne, L. (2021). Cheating detection in browser-based online exams through eye gaze tracking. In *2021 6th International Conference on Information Technology Research (ICITR)*, pages 1–8.

[15] Fang, H., Man, W., Kong, X., and Zhang, X. (2023). Research on the cross-media integration process of educational digital resources based on xapi data. In *2023 3rd International Conference on Educational Technology (ICET)*, pages 194–198.

[16] Ghizlane, M., Hicham, B., and Reda, F. H. (2019). A new model of automatic and continuous online exam monitoring. In *2019 International Conference on Systems of Collaboration Big Data, Internet of Things & Security (SysCoBIoTS)*, pages 1–5.

[17] Goffredo, H., Yuri, P., , Daniela, S., and Nello, S. (2021). Online written exams at the university of milan during covid-19 crisis. *IADIS INTERNATIONAL JOURNAL ON WWW/INTERNET*, 19(2).

[18] Guha, A., Fredrikson, M., Livshits, B., and Swamy, N. (2011). Verified security for browser extensions. In *2011 IEEE Symposium on Security and Privacy*, pages 115–130.

[19] Gullo, K. (2023). After students challenged proctoring software, french court slaps. *Electronic Frontier Foundation*. `https://www.eff.org/deeplinks/2023/03/after-students-challenged-proctoring-software-french-court-slaps-testwe-app`. Accessed on 27/11/2025.

[20] Gupta, B. and Vani, D. M. (2018). An overview of web sockets: The future of real-time communication. *International Research Journal of Engineering and Technology (IRJET)*, 05(12):434–437.

[21] Gursky, N. (2023). What is a helmet content security policy, and do you need it? `https://cybeready.com/content-security-policy/helmet-content-security-policy`. Accessed on 27/11/2025.

[22] Hadyaoui, A. and Cheniti-Belcadhi, L. (2022). Towards a context-aware personalized formative assessment in a collaborative online environment. In *2022 IEEE/ACS 19th International Conference on Computer Systems and Applications (AICCSA)*, pages 1–6.

[23] Hall, J. (2024). Getting started with mongodb & mongoose | mongodb. https://www.mongodb.com/developer/languages/javascript/getting-started-with-mongodb-and-mongoose/. Accessed on 27/11/2025.

[24] Haus, G., Pasquinelli, Y., Scaccia, D., and Scarabottolo, N. (2020). ONLINE WRITTEN EXAMS DURING COVID-19 CRISIS. *Proceedings of the 14th International Conference on e-Learning*.

[25] Janakiraman, S., Sree, K. S., Manasa, V. L., Rajagopalan, S., Thenmozhi, K., and Amirtharajan, R. (2018). Otp on demand - an embedded system for user authentication. In *2018 International Conference on Computer Communication and Informatics (ICCCI)*, pages 1–5.

[26] Jin, B., Li, H., and Zou, Y. (2024). Impact of extensions on browser performance: An empirical study on google chrome.

[27] Journal, I. (2023). Security risks are present in over 50% of browser extensions. https://www.iaesjournal.com/security-risks-are-present-in-over-50-of-browser-extensions/. Accessed on 27/11/2025.

[28] Kaiwinit, S., Ratchatawetchakul, Y., Satchawatee, N., and Wongsim, M. (2024). Ruamrean: Transformative developments in learning management systems. In *2024 5th Technology Innovation Management and Engineering Science International Conference (TIMES-iCON)*, pages 1–5.

[29] Khan, R. A., Khan, S. U., Khan, H. U., and Ilyas, M. (2022). Systematic literature review on security risks and its practices in secure software development. *IEEE Access*, 10:5456–5481.

[30] Kuppers, B., Kerber, F., Meyer, U., and Schroeder, U. (2017). Beyond lockdown: Towards reliable e-assessment. Bildungsräume 2017.

[31] Lazzouni, M. (2024). Biometric authentication for online exams. https://trainingmag.com/biometric-authentication-for-online-exams/. Accessed on 27/11/2025.

[32] Liu, Q. and Sun, X. (2012). Research of web real-time communication based on web socket. *International Journal of Communications Network and System Sciences*, 5(12):797–801.

[33] Miller, B., Rutherford, T., Pack, A., and Johnson, A. (2021). Bridging the scorm and xapi gap: The role of cmi5. *Interservice/Industry Training, Simulation, and Education Conference (I/ITSEC)*, pages 1–11.

[34] Moukhliss, G., Hilali, R. F., and Belhadaoui, H. (2023). Intelligent solution for automatic online exam monitoring. *International Journal of Power Electronics and Drive Systems/International Journal of Electrical and Computer Engineering*, 13(5):5333.

[35] Nath, A. and Mondal, T. (2016). Issues and challenges in two factor authentication algorithms. *International Journal of Latest Trends in Engineering and Technology(IJLTET)*, 6:318–327.

[36] P, J., Kumar S, V., and Rajasimman R, V. (2025). Online exam fraud detection using different techniques. In *2025 International Conference on Frontier Technologies and Solutions (ICFTS)*, pages 1–8.

[37] Penmetsa, M. K. R. (2025). A review of scorm and xapi integration standards for corporate learning platforms. *International Research Journal of Modernization in Engineering Technology and Science*, 7:2855–2861.

[38] Perez, J. L. (2022). Biometric authentication: Advantages and disadvantages. `https://recordia.net/en/understanding-biometric-authentication-advantages-and-disadvantages/`. Accessed on 27/11/2025.

[39] Procyon (2024). Pros and cons of multi factor authentication (2fa) explained. `https://www.procyon.ai/glossary/pros-and-cons-of-multi-factor-authentication-2fa-explained`. Accessed on 27/11/2025.

[40] Sakhipov, A., Omirzak, I., and Fedenko, A. (2025). Beyond face recognition: A multi-layered approach to academic integrity in online exams. *Electronic Journal of e-Learning*, 23(1):81–95.

[41] SCORM.com (2009). Scorm run-time environment: Explanation and examples. `https://scorm.com/scorm-explained/technical-scorm/run-time/`. Accessed on 27/11/2025.

[42] Shetty, J., Dash, D., and Joish, A. K. (2020). Review paper on web frameworks, databases and web stacks. *International Research Journal of Engineering and Technology (IRJET)*, 07(04).

[43] Singh, S., Varshney, G., Singh, T. K., and Mishra, V. (2025). A study on malicious browser extensions in 2025.

[44] StackShare (2017a). Knex.js vs sequelize | what are the differences? `https://stackshare.io/stackups/knex-js-vs-sequelize`. Accessed on 27/11/2025.

[45] StackShare (2017b). npm ioredis vs npm redis | what are the differences? `https://stackshare.io/stackups/npm-ioredis-vs-npm-redis`. Accessed on 27/11/2025.

[46] Suleski, T., Ahmed, M., Yang, W., and Wang, E. (2023). A review of multi-factor authentication in the internet of healthcare things. *Digital Health*, 9:20552076231177144.

[47] Tools4ever (2023). What is an otp (one-time password)? - tools4ever. `https://www.tools4ever.com/glossary/what-is-an-otp-one-time-password/`. Accessed on 27/11/2025.

[48] Tweissi, A., Etaiwi, W., and Al-Eisawi, D. (2022). The accuracy of ai-based automatic proctoring in online exams. *Electronic Journal of e-Learning*, 20.

[49] V, H., Fernandez, A. F., N, P., and N, T. (2024). Exam proctoring system using machine learning. In *2024 International Conference on Smart Technologies for Sustainable Development Goals (ICSTSDG)*, pages 1–7.

[50] Verma, S. (2024). Express.js and its usage in web development. *International Journal of Research Publication and Reviews*, 5:3247–3249.

[51] xAPI (2021a). Comparison of SCORM, xAPI and cmi5. `https://xapi.com/cmi5/comparison-of-scorm-xapi-and-cmi5/`. Accessed on 27/11/2025.

[52] xAPI (2021b). How does cmi5 work? See the cmi5 process in action and how cmi5 works. `https://xapi.com/cmi5/how-cmi5-works/`. Accessed on 27/11/2025.

[53] xAPI (2021c). What is cmi5: An overview of the cmi5 specification. `https://xapi.com/cmi5/overview/`. Accessed on 27/11/2025.

[54] xAPI (2021d). Why cmi5? how cmi5 helps organizations launch and track content. `https://xapi.com/cmi5/benefits/`. Accessed on 27/11/2025.

[55] Yan, W., Yiping, L., Tingting, Z., and Jianzhong, C. (2010). Research of data model of scorm run-time environment. In *2010 3rd International Conference on Information Management, Innovation Management and Industrial Engineering*, volume 2, pages 240–243.

[56] Yang, W., Wang, S., Sahri, N. M., Karie, N. M., Ahmed, M., and Valli, C. (2021). Biometrics for internet-of-things security: A review. *Sensors (Basel)*, 21(18):6163.

[57] Zanevych, O. (2024). Advancing web development: A comparative analysis of modern frameworks for rest and graphql back-end services. *Grail of Science*, (37):216–228.

Appendix A

Sequence Diagrams

A.1 Authentication Sequence diagrams

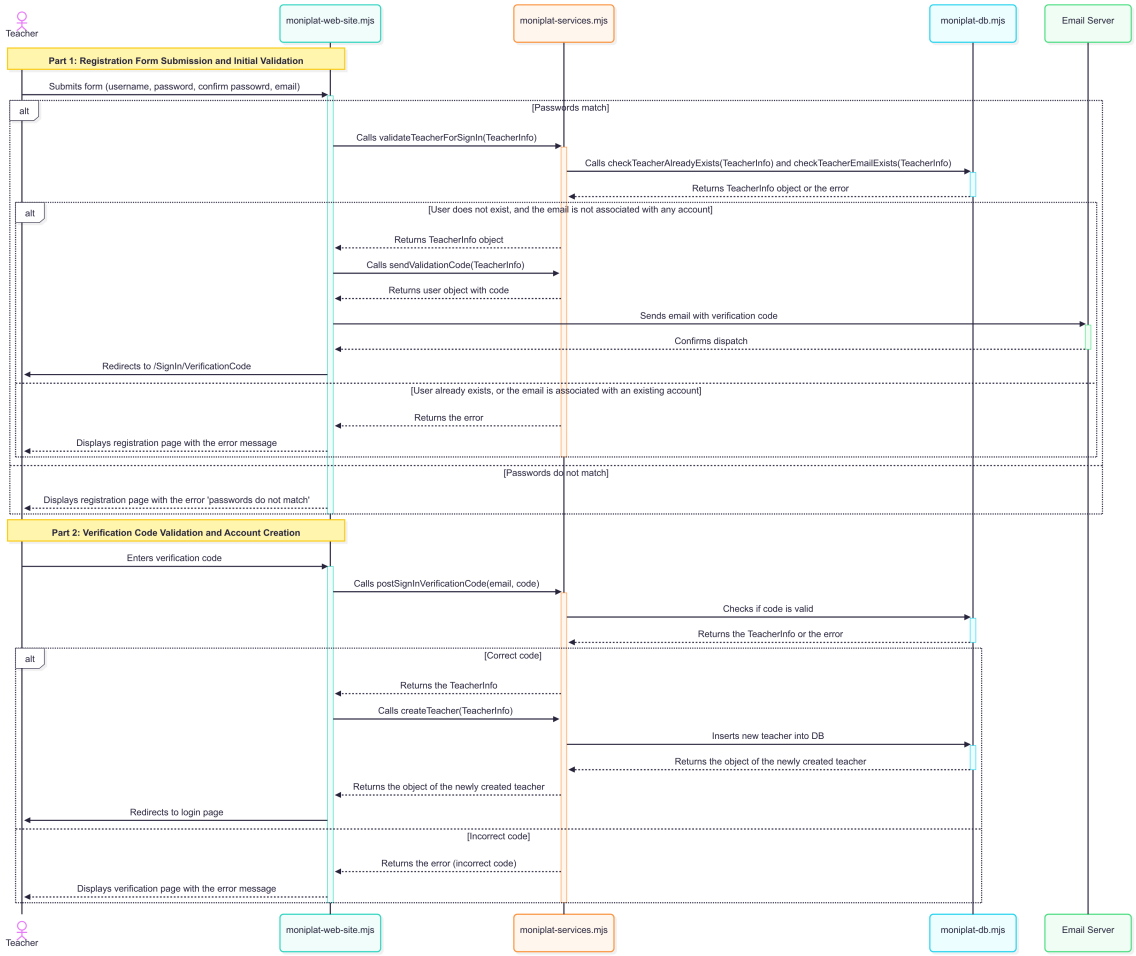


Figure A.1 Teacher Sign In sequence diagram.

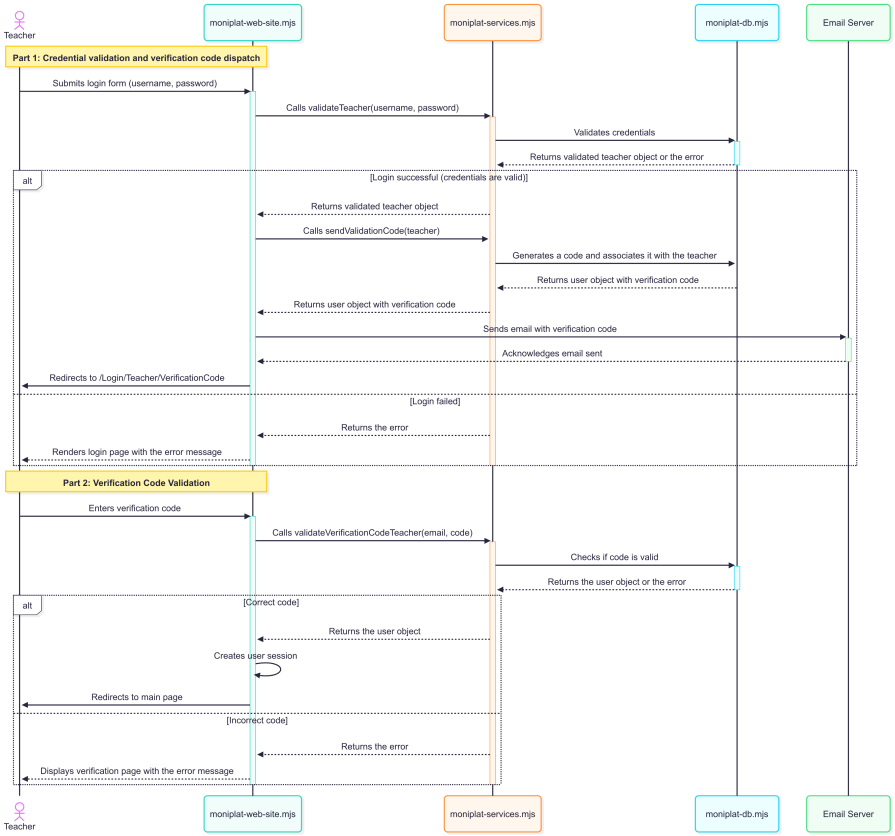


Figure A.2 Teacher login sequence diagram.

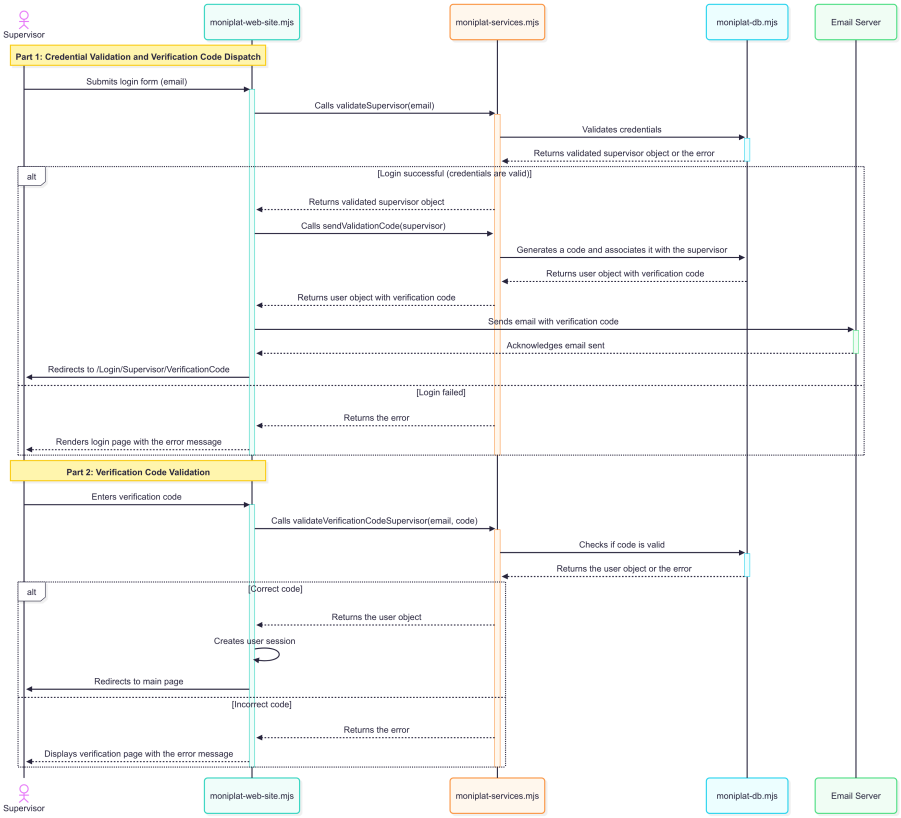


Figure A.3 Supervisor login sequence diagram.

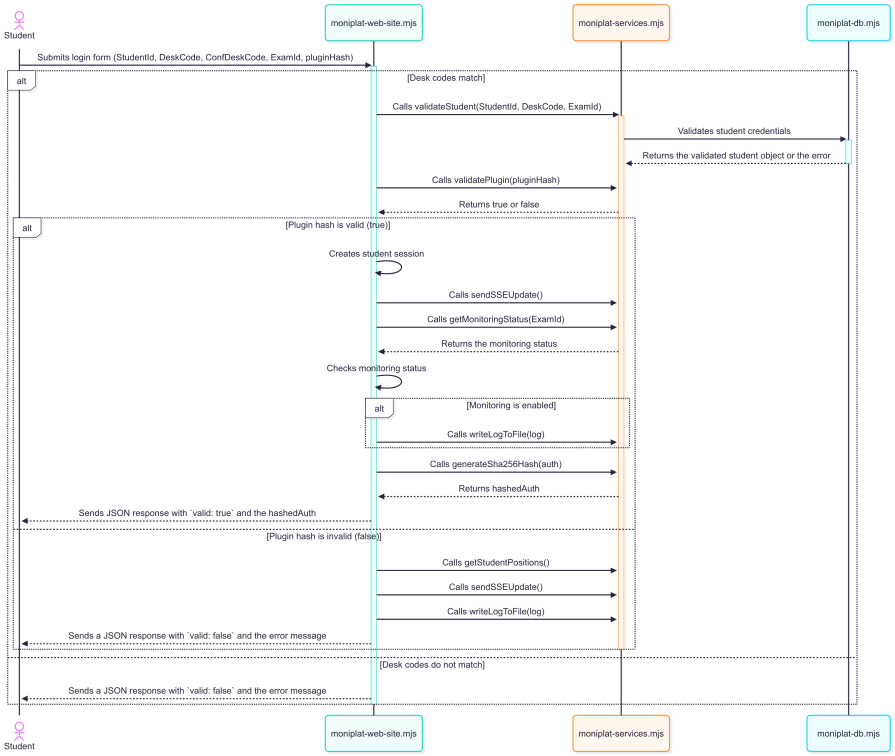


Figure A.4 Login student sequence diagram.

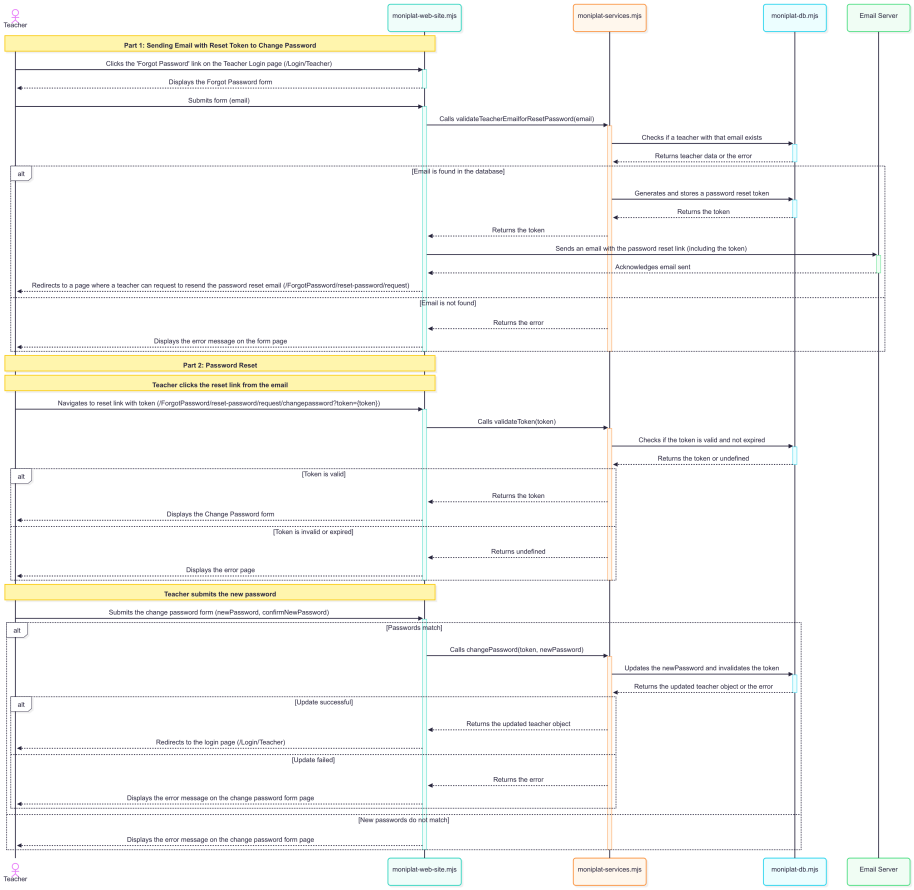


Figure A.5 Change password sequence diagram.

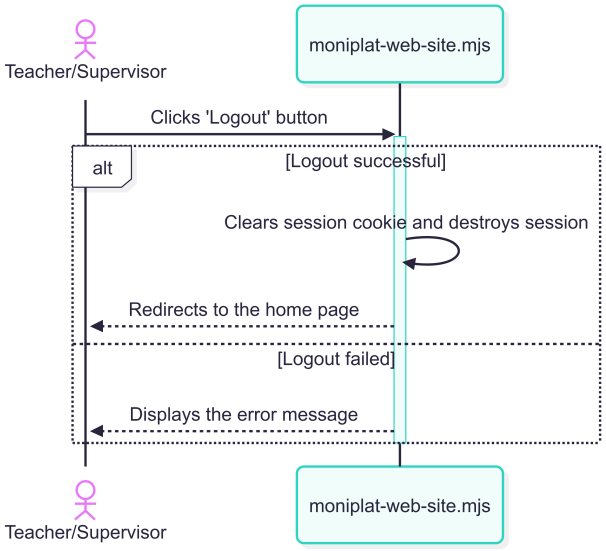


Figure A.6 Logout teachers and supervisors sequence diagram.

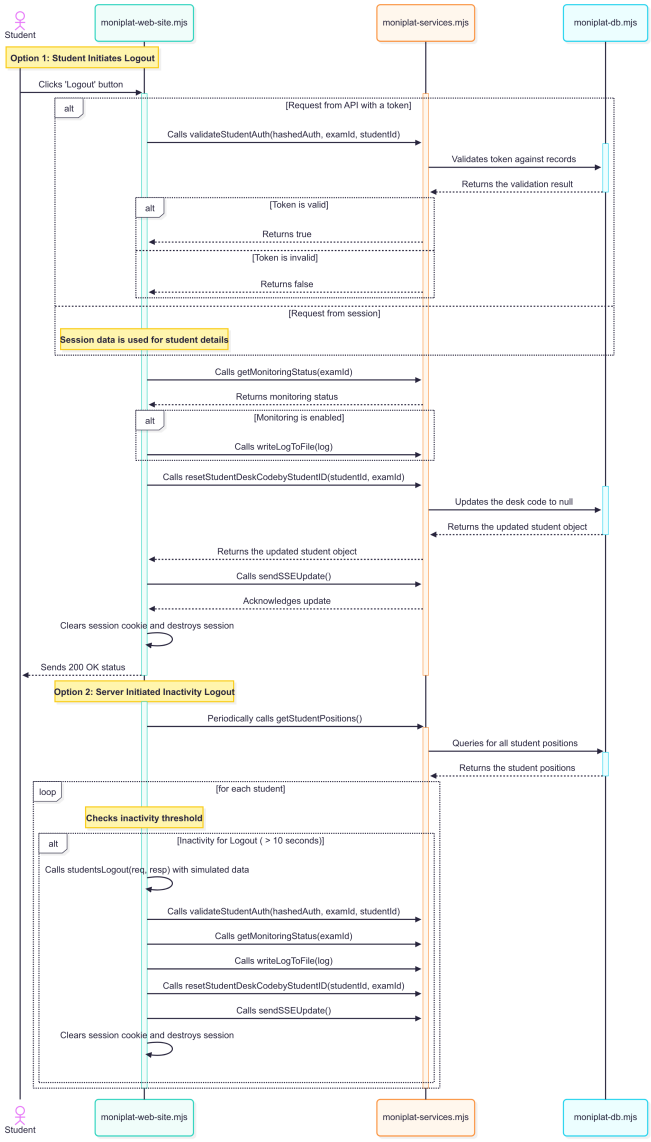


Figure A.7 Logout student sequence diagram.

A.2 Exam Management

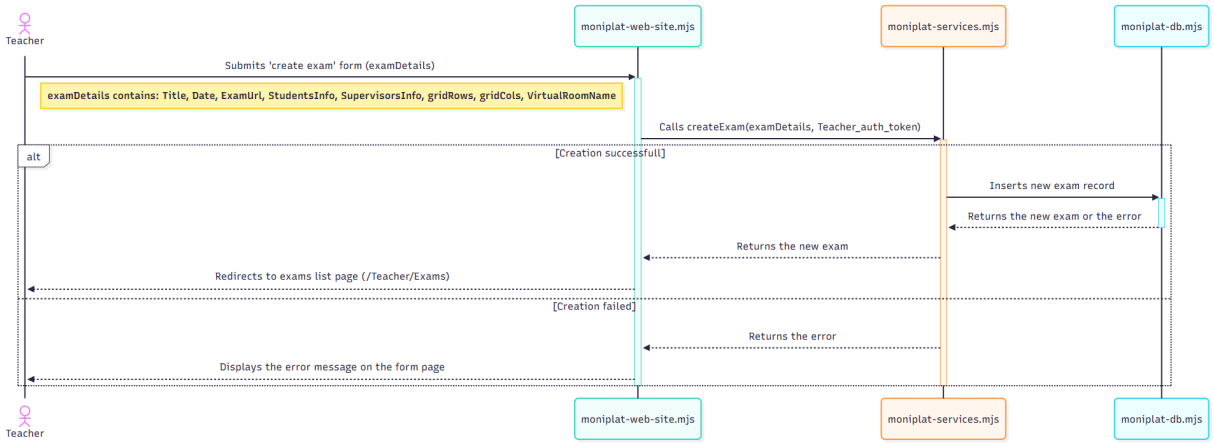


Figure A.8 Create exam sequence diagram.

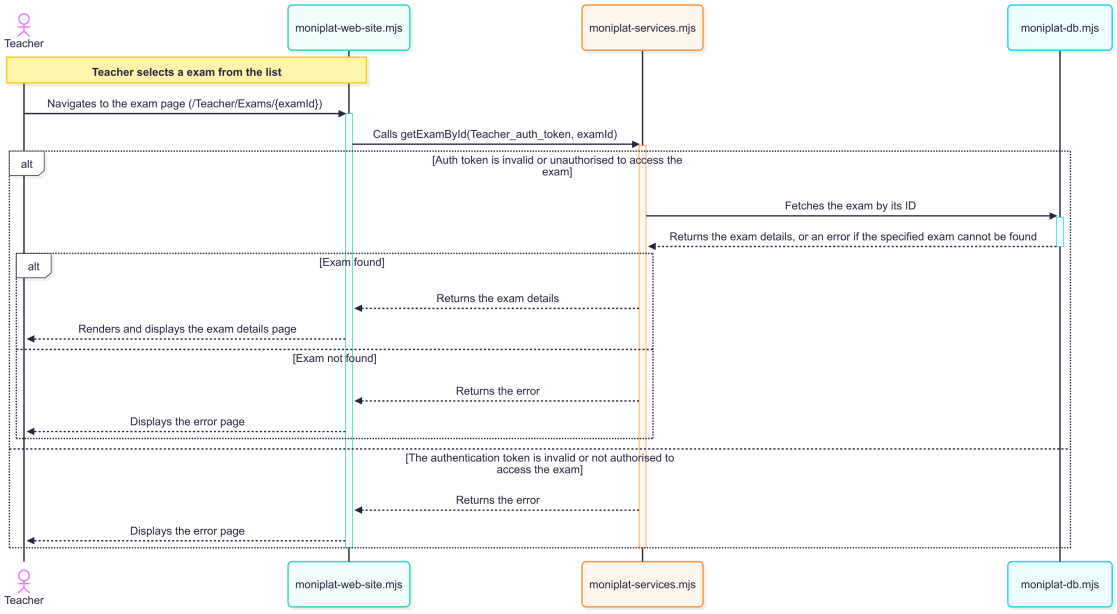


Figure A.9 Get exam teacher sequence diagram.

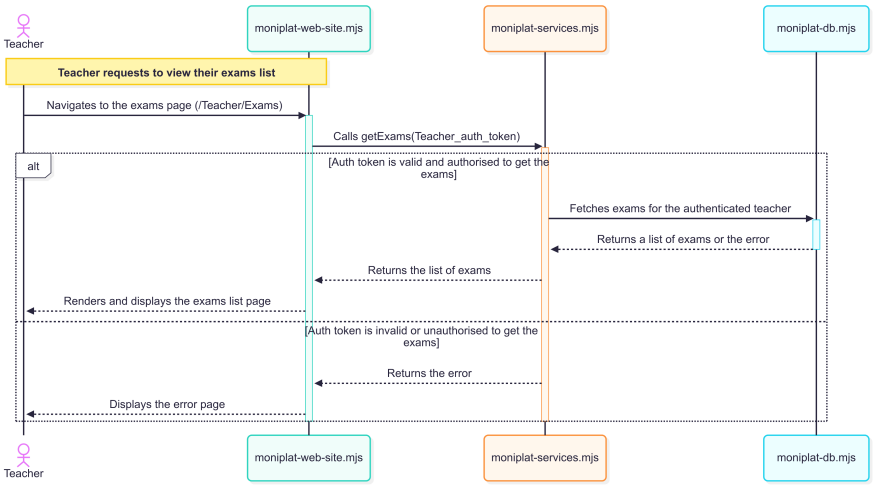


Figure A.10 Get exams Teacher sequence diagram.

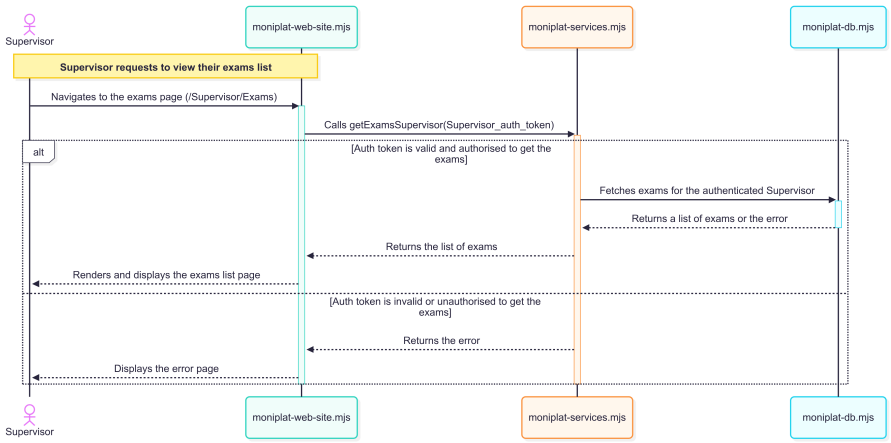


Figure A.11 Get exams supervisor sequence diagram.

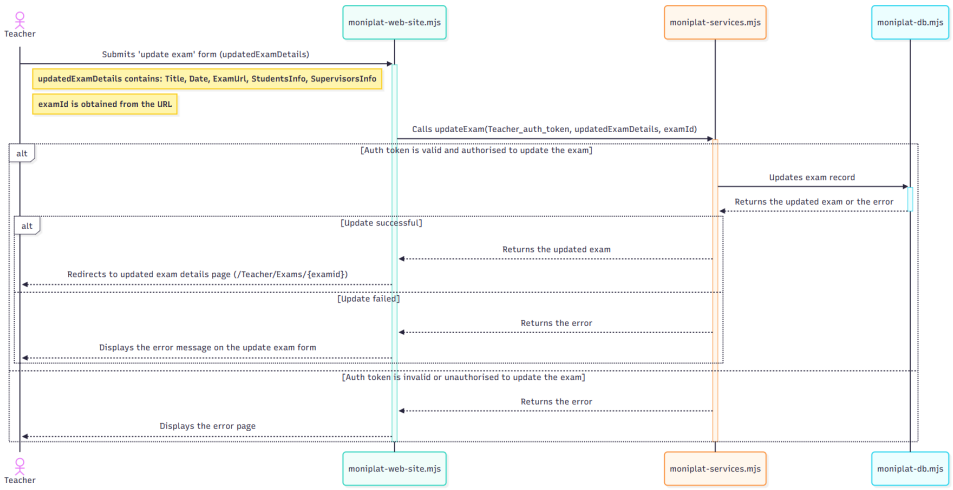


Figure A.12 Update exam sequence diagram.

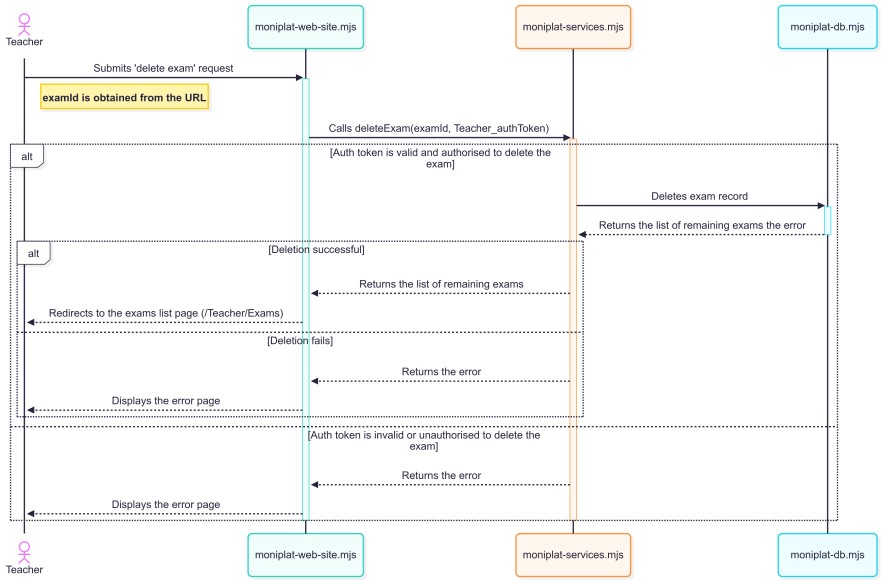


Figure A.13 Delete exam sequence diagram.

A.3 Virtual Room Management

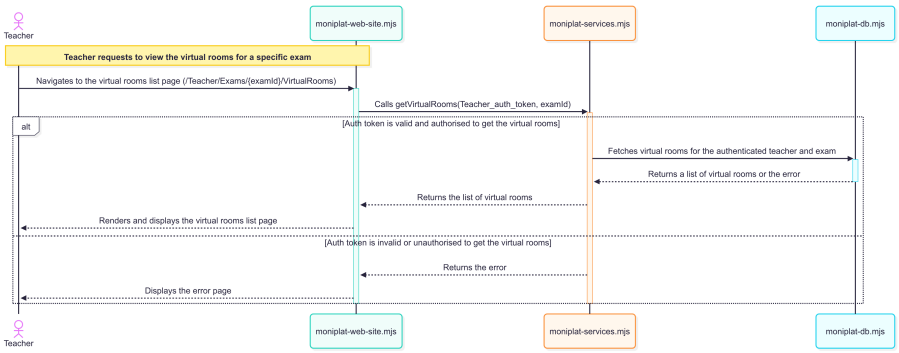


Figure A.14 Get virtual rooms sequence diagram.

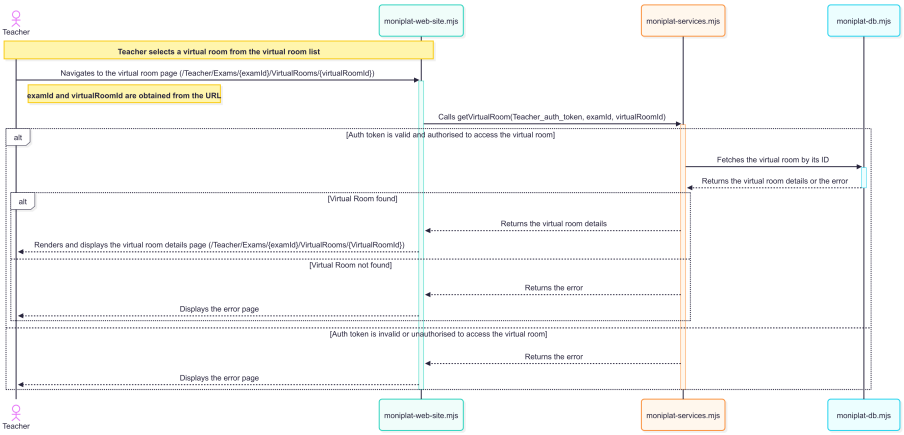


Figure A.15 Get virtual room sequence diagram.

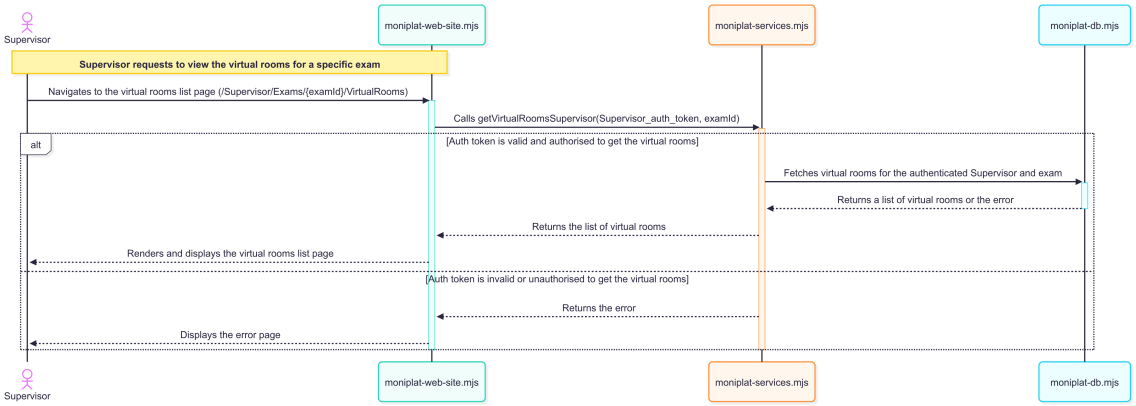


Figure A.16 Get virtual rooms supervisor sequence diagram.

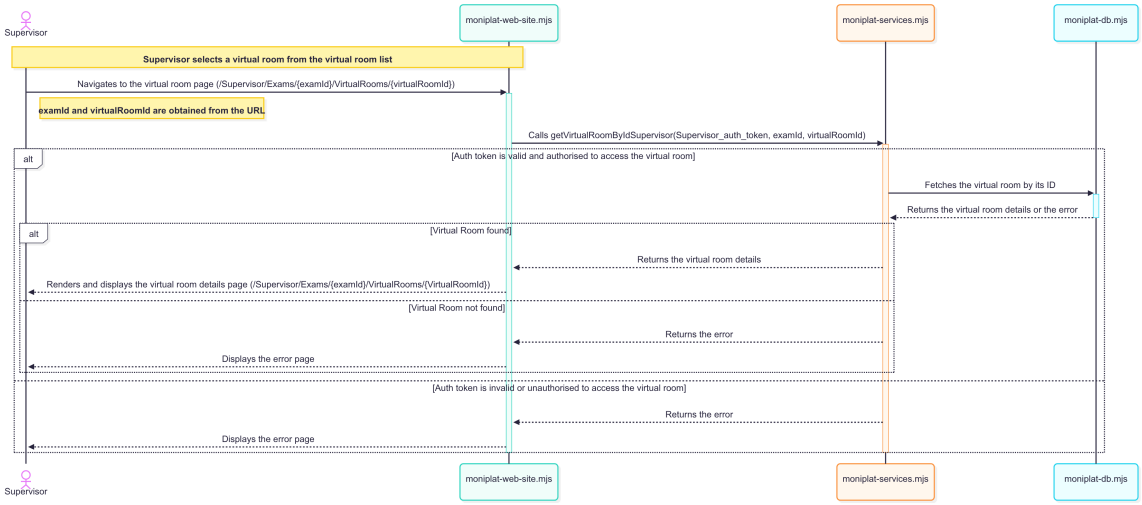


Figure A.17 Get virtual room supervisor sequence diagram.

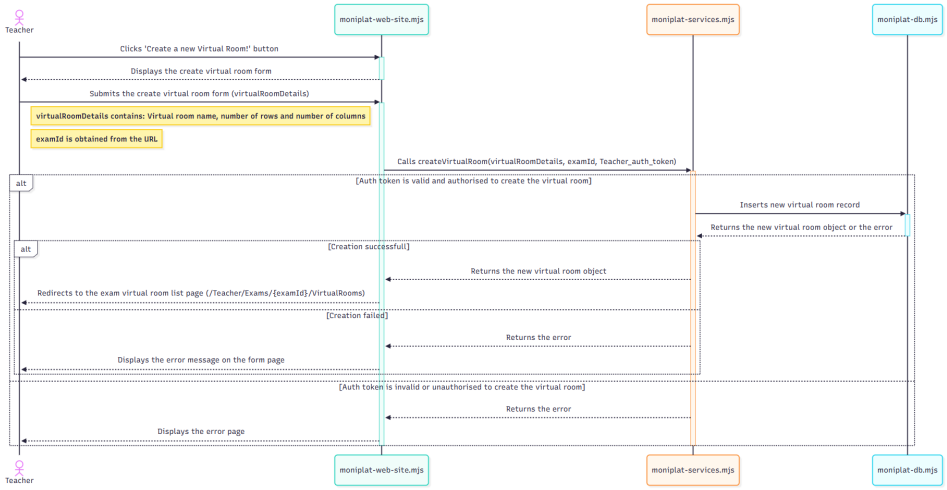


Figure A.18 Create virtual room sequence diagram.

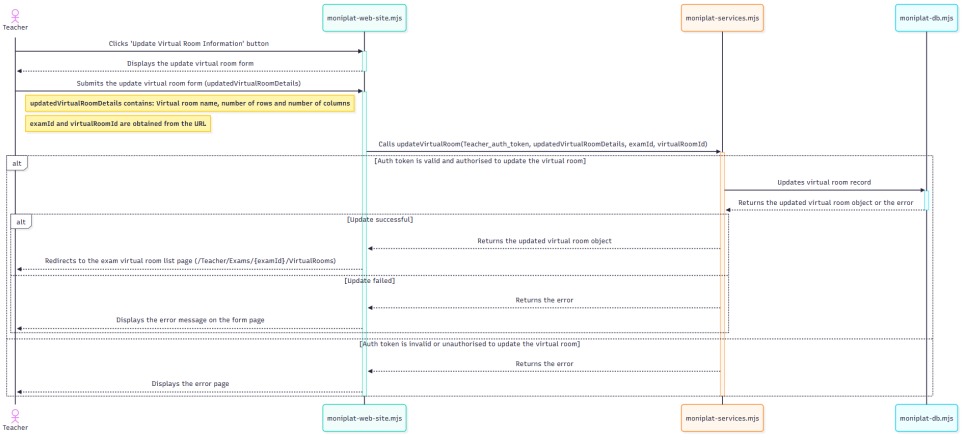


Figure A.19 Update virtual room sequence diagram.

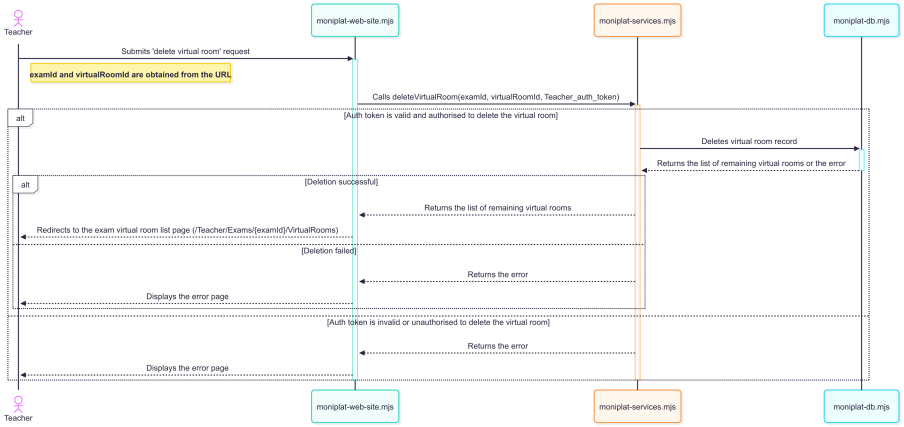


Figure A.20 Delete virtual room sequence diagram.

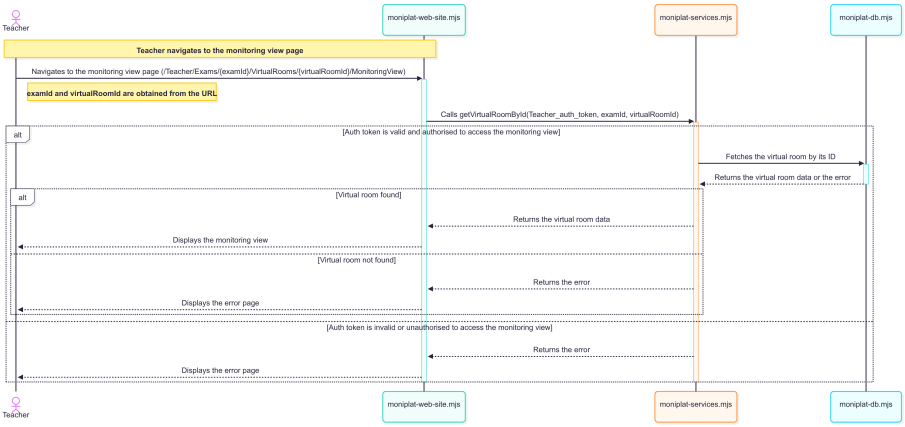


Figure A.21 Get monitoring view sequence diagram.

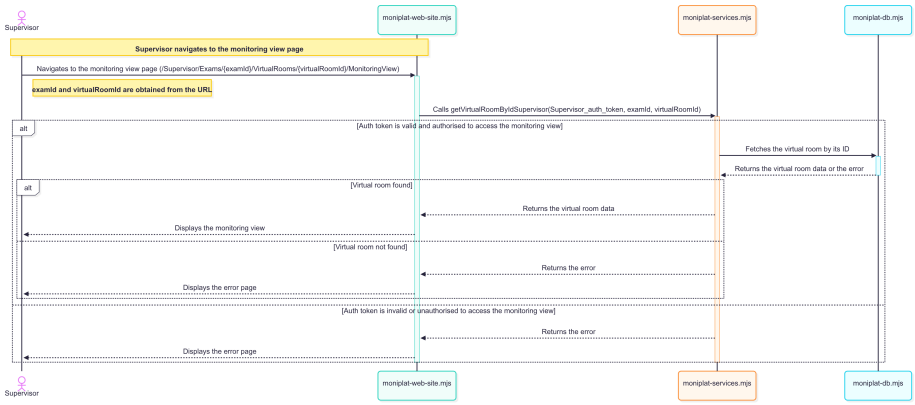


Figure A.22 Get monitoring view supervisor sequence diagram.

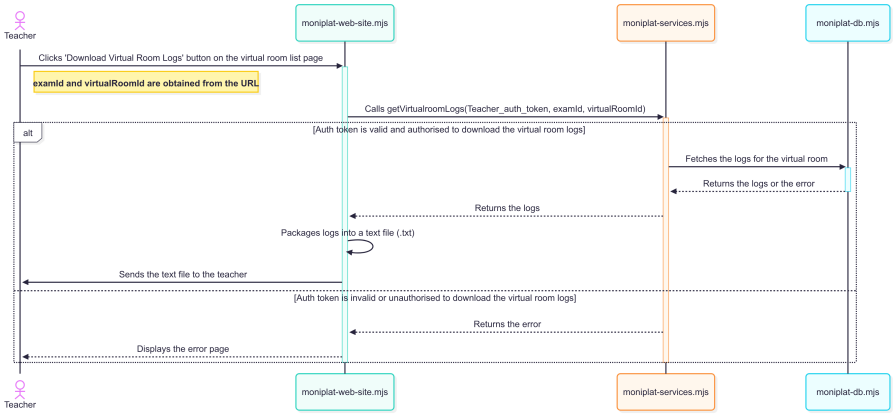


Figure A.23 Download virtual room logs sequence diagram.