2024 - 2025

|  |
| --- |
| WORKSHOP PROJECT ASSIGNMENT SYSTEM Team: LST2 |
| Analysis and design report |
| Alex van Poppel,  Esmanur Akbaș,  German Novykov,  Paschal Chukwubuikem Ifewulu,  Oleksii Chekun |

Table of contents

[1. Introduction 4](#_Toc184738482)

[2. Requirements analysis 5](#_Toc184738483)

[2.1. Functional requirements 6](#_Toc184738484)

[2.1.1. Use Case Diagram 6](#_Toc184738485)

[2.1.2. Use Case Descriptions 7](#_Toc184738486)

[2.1.2.1. Create account 7](#_Toc184738487)

[2.1.2.2. Login account 7](#_Toc184738488)

[2.1.2.3. Submit training session information 7](#_Toc184738489)

[2.1.2.4. Review finished training session proposal 7](#_Toc184738490)

[2.1.2.5. Fill-in student evaluation 8](#_Toc184738491)

[2.1.2.6. Submit teams’ information and client preferences 8](#_Toc184738492)

[2.1.2.7. Submit client submission information 8](#_Toc184738493)

[2.1.2.8. Submit client meeting date 8](#_Toc184738494)

[2.1.2.9. Upload concept training session plan 9](#_Toc184738495)

[2.1.2.10. Submit self-reflection 9](#_Toc184738496)

[2.1.2.11. Release client info 9](#_Toc184738497)

[2.1.2.12. Task opening and deadlines 9](#_Toc184738498)

[2.1.2.13. Manage students 10](#_Toc184738499)

[2.1.2.14. Manage teams 10](#_Toc184738500)

[2.1.2.15. Manage clients 10](#_Toc184738501)

[2.1.2.16. Manually add users 10](#_Toc184738502)

[2.1.2.17. Approve users 11](#_Toc184738503)

[2.1.2.18. Review teams training session submission 11](#_Toc184738504)

[2.1.2.19. Review concept 11](#_Toc184738505)

[2.1.2.20. Edit checklist 11](#_Toc184738506)

[2.1.2.21. Edit Task 12](#_Toc184738507)

[2.1.2.22. Edit Questionnaire 12](#_Toc184738508)

[2.1.2.23. Manage evaluation forms 12](#_Toc184738509)

[2.1.1.22. Manage users 12](#_Toc184738510)

[2.1.2.24. Fill-in student evaluation 13](#_Toc184738511)

[2.2. Non-functional requirements 14](#_Toc184738512)

[2.2.1.1. 1. Implementation 14](#_Toc184738513)

[2.2.1.2. 2. External Interface 14](#_Toc184738514)

[2.2.1.3. 2. Performance 14](#_Toc184738515)

[2.2.1.4. 4. Quality Requirements 14](#_Toc184738516)

[2.2.1.5. Other Relevant Non-functional Requirements 15](#_Toc184738517)

[3. Priority by functionality 16](#_Toc184738518)

[3.1. Priorities Table 16](#_Toc184738519)

[4. Conclusions 17](#_Toc184738520)

[5. Glossary 18](#_Toc184738521)

[6. Bibliography 19](#_Toc184738522)

Table of figures

[Figure 1. UseCase Diagram Universal-user, Client, Student, Time 6](#_Toc184758612)

[Figure 2. UseCase Diagram Audience,Teacher, Admin 6](#_Toc184758613)

Table of tables

[Table 1. MoSCoW priorities table 16](#_Toc184736196)

# Introduction

This document serves as a comprehensive guide for the development of a **Training Session Management System**, a platform designed to streamline the collaboration between students, clients, and educational staff while ensuring efficiency, reliability, and compliance with industry standards. By addressing both functional and non-functional requirements, this document defines the scope, priorities, and technical foundations of the project.

The workshop project system is tailored to accommodate multiple stakeholders, including:

* **Students**: To form teams, manage their projects, and submit their training session plans and evaluations.
* **Clients**: To provide session details, review proposals, and assess student performance.
* **Teachers**: To oversee student progress, manage client relationships, and evaluate submissions.
* **Administrators**: To maintain user roles, ensure system integrity, and monitor overall operations.

The primary goals of this system are to:

* **Simplify Workflows**: Facilitate seamless account management, team collaboration, and task tracking for all users.
* **Ensure Data Integrity and Security**: Protect sensitive information with robust encryption and GDPR-compliant practices. GDPR is a is a European union regulation on information privacy.
* **Promote Scalability**: Support peak usage scenarios with scalable infrastructure.
* **Enhance Usability**: Deliver a user-friendly interface with accessibility compliance and multilingual support.

This document further establishes:

* Detailed **functional requirements** describing the system's core capabilities, such as account creation, training session submissions, and evaluation management.
* Essential **non-functional requirements**, including performance benchmarks, security protocols, and system quality standards.
* Implementation priorities using the **MoSCoW method**, categorizing functionalities into Must Have, Should Have, Could Have, and Will not Have for effective project planning. (GeeksForGeeks, 2024)

By aligning stakeholder needs with a robust technical architecture, this system is designed to foster collaboration, ensure efficiency, and deliver a professional platform for training session management.

# Requirements analysis

The **Requirements Analysis** section serves as the cornerstone for designing and developing the Training Session Management System. It provides a comprehensive and structured overview of the system's expectations, ensuring alignment between stakeholder goals and the technical implementation. This section is essential for identifying and addressing user needs, minimizing ambiguities, and creating a roadmap that guides the development process effectively.

The analysis is divided into two key categories, ensuring both the functional capabilities and the system's overall quality are clearly defined:

**Functional Requirements:**

Functional requirements outline the system's core capabilities, focusing on the interactions and tasks that users can perform. These are detailed through:

* **Use Case Diagrams:** Visual representations of system functionalities, segmented for clarity and scalability. (GeeksForGeeks, 2024)
* **Use Case Descriptions:** Detailed narratives that explain the normal flows, alternative scenarios, preconditions for each functionality, and links to the use cases in our Figma prototype.

These requirements encompass a wide range of activities, from account management and training session submissions to evaluations and administrative oversight. Each functionality is designed to cater to the unique needs of different user roles, including students, teachers, clients, administrators, and audiences.

**Non-Functional Requirements:**

Non-functional requirements establish the standards and constraints that ensure the system's reliability, usability, and maintainability. They focus on quality attributes such as:

* **Implementation Details:** Defining the technical stack, including frameworks, databases, and hosting infrastructure.
* **Performance Metrics:** Ensuring fast response times, scalability, and robust error handling.
* **Quality Assurance:** Setting benchmarks for uptime, responsiveness, and data protection, while emphasizing internationalization and user support.

By addressing these requirements, the system is designed to deliver not only functional excellence but also a seamless user experience that meets the highest standards of quality and reliability. Through this meticulous analysis, the project lays a solid foundation for building a system that is efficient, scalable, and aligned with the diverse needs of its stakeholders.

## Functional requirements

### Use Case Diagram

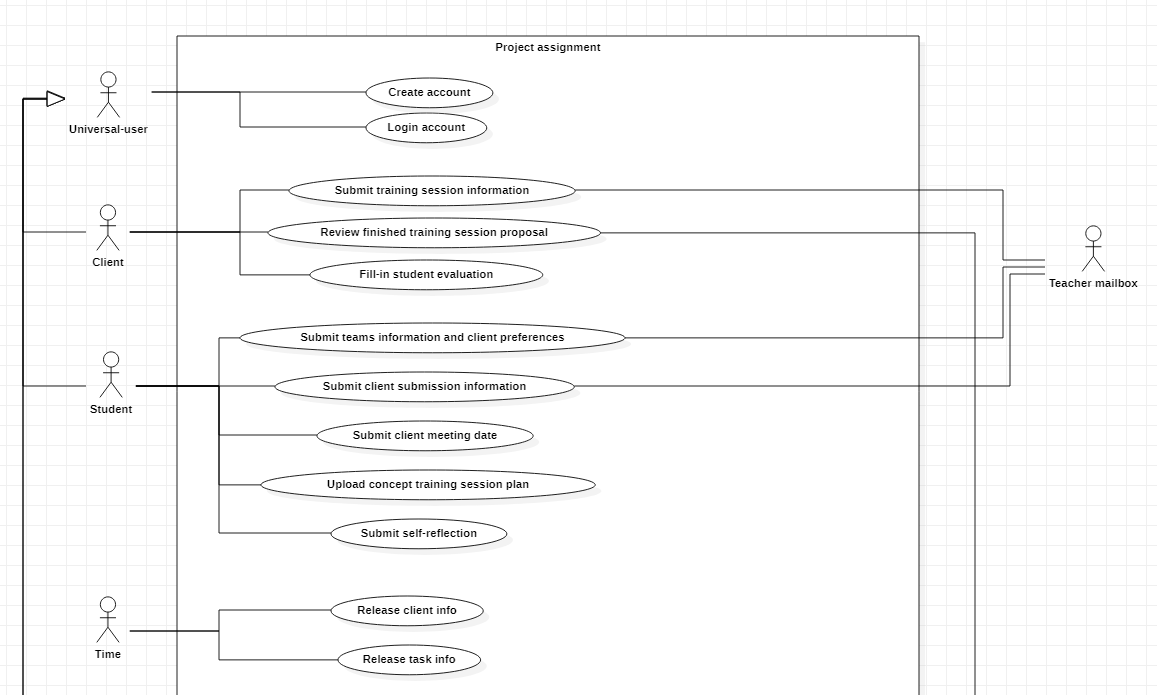


Figure 1. UseCase Diagram Universal-user, Client, Student, Time

A diagram with text on it

Description automatically generated with medium confidence

Figure 2. UseCase Diagram Audience, Teacher, Admin

(GeeksForGeeks, 2024)

### Use Case Descriptions

#### Create account

Functionality: As a universal user, I can create an account.

Normal flow: The system displays a form with information account information (name, role, email, password, etc) necessary to create an account. The actor fills in this information and creates the account.

Alternatives:

* **Not a Thomas more email address:**
  + - The system sends a request for account approval to the teachers.

[**Link**](https://www.figma.com/proto/6MK75jd9kkfBlR47gV4451/LST2-Prototype?node-id=8-8&node-type=canvas&t=t1QVF7QA30jfSTu9-0&scaling=min-zoom&content-scaling=fixed&page-id=0%3A1)

#### Login account

Functionality: As a universal user, I can log into my account.

Precondition: The actor has an account.

Normal flow: The system displays the login page and request for the actors' credentials. The actor fills in the credentials necessary to log in. The system redirects the actor to their dashboard.

[**Link**](https://www.figma.com/proto/6MK75jd9kkfBlR47gV4451/LST2-Prototype?node-id=8-9&node-type=canvas&t=t1QVF7QA30jfSTu9-0&scaling=min-zoom&content-scaling=fixed&page-id=7%3A2)

#### Submit training session information

Functionality: As a client, I can submit training session information.

Normal flow: The system displays a form for the contact person(s) information (name, email, phone number, etc). The actor fills in the information requested. The system displays a form with training session(s) information (Target audience, timeframe, subject, etc). The actor fills in the information. The system displays a form for other information that would be helpful to know. The actor fills in the information and submits the form.

[**Link**](https://www.figma.com/proto/6MK75jd9kkfBlR47gV4451/LST2-Prototype?node-id=8-10&node-type=canvas&t=t1QVF7QA30jfSTu9-0&scaling=min-zoom&content-scaling=fixed&page-id=7%3A3&starting-point-node-id=8%3A10)

#### Review finished training session proposal

Functionality: As a client, I can view the finished training session proposal.

Normal flow: The system displays the proposal the students have provided and prompts the actor to approve or reject the proposal. The actor fills in a comment for the students and approves the proposal. The system sends an email to the student notifying them.

Alternatives:

* **Proposal gets rejected:**
  + - The system sends an email to the students notifying them that their proposal has been rejected with the comments of the actor.

[**Link**](https://www.figma.com/proto/6MK75jd9kkfBlR47gV4451/LST2-Prototype?node-id=8-11&node-type=canvas&t=t1QVF7QA30jfSTu9-0&scaling=min-zoom&content-scaling=fixed&page-id=7%3A4)

#### Fill-in student evaluation

Functionality: As a client, I can fill in the student evaluation.

Normal flow: The system displays the student evaluation form for general info (time, date, etc) and student evaluation (student involvement, strengths, etc.). The actor fills in the information. The system displays a list of qualities with checkboxes the students are rated on. The actor checks the boxes and submits the form.

[**Link**](https://www.figma.com/proto/6MK75jd9kkfBlR47gV4451/LST2-Prototype?node-id=8-12&node-type=canvas&t=t1QVF7QA30jfSTu9-0&scaling=min-zoom&content-scaling=fixed&page-id=7%3A5&starting-point-node-id=8%3A12)

#### Submit teams’ information and client preferences

Functionality: As a student, I can submit details about their team.

Normal flow: The system shows a form with the information needed to create a team (Coach, team members, etc.) The actor fills in the information and creates the team. The system displays a list of options for training sessions to participate in. The actor ranks their top three training sessions. The system sends in their team request for approval.

[**Link**](https://www.figma.com/proto/6MK75jd9kkfBlR47gV4451/LST2-Prototype?node-id=43-2&node-type=canvas&t=t1QVF7QA30jfSTu9-0&scaling=min-zoom&content-scaling=fixed&page-id=31%3A16&starting-point-node-id=43%3A2)

#### Submit client submission information

Functionality: As a student, I can submit a client to be a part of the program

Precondition: The deadline for the preference submission has elapsed.

Normal flow: The system displays a form with information about the client/contact person (name, email, company, etc). The actor fills in this information and submits the application.

[**Link**](https://www.figma.com/proto/6MK75jd9kkfBlR47gV4451/LST2-Prototype?node-id=47-13&node-type=canvas&t=t1QVF7QA30jfSTu9-0&scaling=min-zoom&content-scaling=fixed&page-id=43%3A474&starting-point-node-id=47%3A13)

#### Submit client meeting date

Functionality: As a student, I can submit the date for the client meeting.

Precondition:

* The actor must have planned a meeting date with their client.

Normal flow: The system displays a form for the client meeting date and any other info of note. The actor fills in the information and submits the form.

[**Link**](https://www.figma.com/proto/6MK75jd9kkfBlR47gV4451/LST2-Prototype?node-id=69-39&node-type=canvas&t=t1QVF7QA30jfSTu9-0&scaling=min-zoom&content-scaling=fixed&page-id=64%3A34&starting-point-node-id=69%3A39)

#### Upload concept training session plan

Functionality: As a student, I can upload concepts related to their training session.

Normal flow: The system prompts the user to upload the plan alongside filling in a checklist. The actor goes through the checklist and submits their session plan.

* **Not filled in checklist:**
  + - The system does not allow the actor to upload the concept until they qualify for all the checkbox criteria.

[**Link**](https://www.figma.com/proto/6MK75jd9kkfBlR47gV4451/LST2-Prototype?node-id=126-705&node-type=canvas&t=t1QVF7QA30jfSTu9-0&scaling=min-zoom&content-scaling=fixed&page-id=64%3A35)

#### Submit self-reflection

Functionality: As a student, I can submit a self-reflection document detailing their learning and experiences.

Normal flow: The system shows the user a form for training session information (“How did it go,” “Were you prepared,” etc). The actor fills in the information. The system displays a form with information about the students’ conclusions after the session (“What are your strengths,” “What would you differently next time,” etc). The actor fills in this information. The system shows a form for additional information and the training session pictures. The actor fills this in and submits the form.

* **Paper form uploaded:**
  + - The system closes the reflection task.

[**Link**](https://www.figma.com/proto/6MK75jd9kkfBlR47gV4451/LST2-Prototype?node-id=164-149&node-type=canvas&t=t1QVF7QA30jfSTu9-0&scaling=min-zoom&content-scaling=fixed&page-id=64%3A37&starting-point-node-id=164%3A149)

#### Release client info

Functionality: As time goes on, I will open and close tasks for the students when specific deadlines pass.

Normal flow: The system displays the client information for the students.

[**Link**](https://www.figma.com/proto/6MK75jd9kkfBlR47gV4451/LST2-Prototype?node-id=164-730&node-type=canvas&t=t1QVF7QA30jfSTu9-0&scaling=min-zoom&content-scaling=fixed&page-id=164%3A510&starting-point-node-id=164%3A730)

#### Task opening and deadlines

Functionality: As time goes on, I will open and close tasks for the students when specific deadlines pass.

Precondition: An opening deadline has passed.

Normal flow: The system opens the task for the students.

Alternatives:

* **The closing deadline has passed:**
  + - * The system closes the task for the students.

[**Link**](https://www.figma.com/proto/6MK75jd9kkfBlR47gV4451/LST2-Prototype?node-id=164-730&node-type=canvas&t=t1QVF7QA30jfSTu9-0&scaling=min-zoom&content-scaling=fixed&page-id=164%3A510&starting-point-node-id=164%3A730)

#### Manage students

Functionality: As a teacher, I can manage student information.

Normal flow: The system displays a list of all students. The actor selects a student. The system displays information about them and their group (team, email, coach, training session info, etc.) The actor can edit/look at this information.

[**Link**](https://www.figma.com/proto/6MK75jd9kkfBlR47gV4451/LST2-Prototype?node-id=123-332&node-type=frame&t=t1QVF7QA30jfSTu9-0&scaling=min-zoom&content-scaling=fixed&page-id=27%3A11&starting-point-node-id=123%3A332&show-proto-sidebar=1)

#### Manage teams

Functionality: As a teacher, I can manage team information.

Normal flow: The system displays a list of all teams. The actor selects a team. The system displays team-specific information about the group and their progress (client, language, team members, proposal, etc.) The actor can edit/look at this information.

[**Link**](https://www.figma.com/design/6MK75jd9kkfBlR47gV4451/LST2-Prototype?node-id=27-16&node-type=canvas&t=t1QVF7QA30jfSTu9-0)

#### Manage clients

Functionality: As a teacher, I can manage client-related information.

Normal flow: The system displays a list of clients associated. The actor selects a client. The system displays client specific information about them and their workshops (Teams, meeting session, workshop(s) info, etc.). The actor can edit/look at this information

[**Link**](https://www.figma.com/proto/6MK75jd9kkfBlR47gV4451/LST2-Prototype?node-id=449-55&node-type=canvas&t=t1QVF7QA30jfSTu9-0&scaling=min-zoom&content-scaling=fixed&page-id=27%3A17&starting-point-node-id=449%3A55)

#### Manually add users

Functionality: As a teacher, I can manually add users into the system.

Normal flow: The system displays a form with basic information needed to create an account (Name, email, role, etc.) The client fills in this information and selects the role of client. The system displays a form with more client-specific information (company, workshop subject, workshop audience, etc.) The actor fills in this information. The system displays a form for information about the training session (Goals, time period, etc.) The actor fills in this information and creates the account.

* **Student role is chosen:**
  + - The system prompts to user to pick a coach for the student. The actor picks a coach and creates the account.

[**Link**](https://www.figma.com/proto/6MK75jd9kkfBlR47gV4451/LST2-Prototype?node-id=445-192&node-type=canvas&t=t1QVF7QA30jfSTu9-0&scaling=min-zoom&content-scaling=fixed&page-id=445%3A169&starting-point-node-id=445%3A192)

#### Approve users

Functionality: As a teacher, I can approve or reject applications to become users.

Normal flow: The system displays a list of requests that have yet to be processed. The actor approves the user's account. The system finishes registering the account.

* **The request is rejected:**
  + - The system does not register the account, and the request gets deleted.

[**Link**](https://www.figma.com/proto/6MK75jd9kkfBlR47gV4451/LST2-Prototype?node-id=445-192&node-type=canvas&t=t1QVF7QA30jfSTu9-0&scaling=min-zoom&content-scaling=fixed&page-id=445%3A169&starting-point-node-id=445%3A192)

#### Review teams training session submission

Functionality: As a teacher, I can approve or deny a team’s request to collaborate with a client.

Normal flow: The system displays a list of all the current requests. The actor approves the team’s request to work with a client. The system registers the team.

* **The request is rejected:**
  + - The system does not register the team and sends the students an email.

[**Link**](https://www.figma.com/proto/6MK75jd9kkfBlR47gV4451/LST2-Prototype?node-id=164-383&node-type=canvas&t=t1QVF7QA30jfSTu9-0&scaling=min-zoom&content-scaling=fixed&page-id=27%3A18)

#### Review concept

Functionality: As a teacher, I can approve the first concept submitted by the students.

Precondition: Students must have submitted a concept for review.

Normal flow: The system displays all the concepts that the actor must review. The actor chooses which concept they want to review. The system displays the concept and prompts the actor to give any comments. The actor comments on and approves the concept. The system sends an email to the students about the approval.

Alternatives:

* **Submission gets rejected:**
  + - The system sends an email to the students notifying them that their submission has been rejected.

[**Link**](https://www.figma.com/proto/6MK75jd9kkfBlR47gV4451/LST2-Prototype?node-id=126-150&node-type=canvas&t=t1QVF7QA30jfSTu9-0&scaling=min-zoom&content-scaling=fixed&page-id=45%3A18&starting-point-node-id=126%3A150)

#### Edit checklist

Functionality: As a teacher, I edit the checklist shown to students before submitting a concept.

Normal flow: The system displays a list of all the current checklist questions. The actor adds another question. The system adds the question to the list.

[**Link**](https://www.figma.com/proto/6MK75jd9kkfBlR47gV4451/LST2-Prototype?node-id=942-304&node-type=canvas&t=t1QVF7QA30jfSTu9-0&scaling=min-zoom&content-scaling=fixed&page-id=942%3A303)

#### Edit Task

Functionality: As a teacher, I can edit the description and deadlines for the tasks given to the students.

Normal flow: The system has the current details for the tasks. The actor edits the details of the task and saves the changes. The system saves the changes.

[**Link**](https://www.figma.com/proto/6MK75jd9kkfBlR47gV4451/LST2-Prototype?node-id=942-935&node-type=canvas&t=t1QVF7QA30jfSTu9-0&scaling=min-zoom&content-scaling=fixed&page-id=942%3A919)

#### Edit Questionnaire

Functionality: As a teacher, I can edit the different forms that other users of the application must fill in.

Normal flow: The system displays a list of all the current types of evaluation forms used in the application. The actor selects which form they want to edit. The system displays a list of questions currently in this form and if they are required to submit the form. The actor makes the changes they want and saves. The system saves the changes.

[**Link**](https://www.figma.com/proto/6MK75jd9kkfBlR47gV4451/LST2-Prototype?node-id=942-490&node-type=canvas&t=t1QVF7QA30jfSTu9-0&scaling=min-zoom&content-scaling=fixed&page-id=942%3A489)

#### Manage evaluation forms

Functionality: As a teacher, I can view and manage the evaluation forms.

Normal flow: The system displays a list of the current active teams. The actor selects which team they want to edit the forms for. The system displays the evaluation forms for this team. The actor selects whichever form they would like to see. The system displays the form. The actor can see the answers that the user gave to the evaluation.

[**Link**](https://www.figma.com/proto/6MK75jd9kkfBlR47gV4451/LST2-Prototype?node-id=942-490&node-type=canvas&t=t1QVF7QA30jfSTu9-0&scaling=min-zoom&content-scaling=fixed&page-id=942%3A489)

#### 2.1.1.22. Manage users

Functionality: As an admin, I can manage the roles of teacher accounts.

Normal flow: The system displays a list of all the current users, including teachers. The actor selects which user they want to manage. The system displays the details of this user. The actor edits the details of the user and saves them. The system saves the changes.

[**Link**](https://www.figma.com/proto/6MK75jd9kkfBlR47gV4451/LST2-Prototype?node-id=126-368&node-type=canvas&t=t1QVF7QA30jfSTu9-0&scaling=min-zoom&content-scaling=fixed&page-id=89%3A2&starting-point-node-id=126%3A368)

#### Fill-in student evaluation

Functionality: As an audience member, I can fill in the student evaluation form.

Preconditions: The actor has a link to the page.

Normal flow: The system displays the student-evaluation form checklist for general questions. The actor fills in the checklist. The system displays a list of open-ended questions about the session. The actor fills in the questions and submits the form.

[**Link**](https://www.figma.com/proto/6MK75jd9kkfBlR47gV4451/LST2-Prototype?node-id=123-124&node-type=canvas&t=t1QVF7QA30jfSTu9-0&scaling=min-zoom&content-scaling=fixed&page-id=89%3A3&starting-point-node-id=123%3A124)

## Non-functional requirements

#### 1. Implementation

* **Hardware/System Software:**
  + - The system will run on standard cloud web servers using Microsoft Azure.
* **Programming Language:**
  + - Development will primarily use the PHP Laravel framework and the Livewire framework. And Tailwind CSS for styling (LiveWire, n.d.) (Tailwind, n.d.) (Laravel, n.d.)
* **Database:**
  + - A relational database, specifically MySQL, will be used due to its scalability and compatibility with the chosen frameworks. (ORACLE, n.d.)
* **Development Platform:**
  + - The system will be developed on a local-based platform, Laravel Herd. (Laravel, BeyondCode, n.d.)

#### 2. External Interface

* **Software systems:**
  + **Mailserver:**
    - The system must send an output to the mail server to send actors emails according to the use case diagram descriptions.
    - The sending of email should be done via the SMTP protocol. And in the MIME (Multipurpose Internet Mail Extensions) format. (Amazon, n.d.) (Lenovo, n.d.)

#### 2. Performance

* **Response Times:**
  + - Interactive functionalities (e.g., login, data submission) must respond within 1 second. Background processes like report generation should be completed within 10 seconds.
* **Scalability:**
  + - The system must handle a minimum of twenty simultaneous users under normal operations, with scaling to accommodate up to fifty during peak usage.
* **Error Handling:**
  + - The system must gracefully handle performance degradation by queuing excess requests during peak load instead of dropping them. Errors will be logged with contextual information (e.g., timestamps, severity levels, affected endpoints) to aid in debugging. User-friendly error messages will be displayed without exposing sensitive information.

#### 4. Quality Requirements

* **Security:**
  + Aimed at guaranteeing:
    - CIA (Confidentiality, Integrity, Availability) of information (Washington University in St. Louis, n.d.)
    - The sensitive information of all users should not be public to those using the application. Unless specifically stated otherwise. (e.g., revealing client contact info)
    - Depending on the degree of criticality of the system:
    - Many users' data would be exposed. None of this info is highly sensitive, but this should still be avoided at all costs no matter what.
* **Uptime:** 
  + - The system must maintain a minimum uptime of 99.5% as per cloud hosting SLA (Service Level Agreement). (atlassian., n.d.)
* **Responsiveness:** 
  + - The UI should be fully responsive across all device screen sizes, leveraging Tailwind CSS for styling. (Tailwind, n.d.)
* **Data Protection:** 
  + - All data transfers must use HTTPS to encrypt communications. (CloudFlare, n.d.)
* **Robustness:**
  + - The system must handle unexpected inputs gracefully without crashing or exposing sensitive data.
* **Documentation**:
  + - Provide clear and concise documentation for developers and end users

#### Other Relevant Non-functional Requirements

* **Internationalization:**
  + - The system will support the English language for user interfaces and notifications.
* **User Support:**
  + - If the student encounters a bug/issue. They will be able to contact their coach and/or get in touch with the system administrator.

# Priority by functionality

## Priorities Table

Table . MoSCoW priorities table

|  |  |
| --- | --- |
| Use case | Priority |
| Create account | Must Have |
| Login account | Must Have |
| Submit training session information | Must Have |
| Review finished training session proposal | Must Have |
| Fill-in student evaluation | Must Have |
| Submit teams’ information and client preferences | Must Have |
| Submit client submission information | Should Have |
| Submit client meeting date | Must Have |
| Upload concept training session plan | Must Have |
| Submit self-reflection | Must Have |
| Release client info | Must Have |
| Task opening and deadlines | Must have |
| Manage students | Must Have |
| Manage teams | Must Have |
| Manage clients | Must Have |
| Manually add users | Must Have |
| Approve users | Should Have |
| Review teams training session submission | Must Have |
| Review concept | Should Have |
| Edit checklist | Must have |
| Edit task | Must have |
| Edit questionnaire | Must have |
| Manage evaluation forms | Must Have |

# Conclusions

The Training Session Management System successfully addresses the need for a streamlined platform that enhances collaboration between students, clients, and educators. By clearly defining both functional and non-functional requirements, the system ensures seamless account management, training session planning, and user evaluations while maintaining high standards of security and accessibility.

The chosen technologies, including PHP Laravel and MySQL, provide a reliable foundation for scalability and performance, meeting the system's technical demands. Prioritizing tasks using the MoSCoW method has ensured that the most critical functionalities are delivered effectively, aligning with stakeholder needs.

This project has provided valuable insights into balancing technical and user-centered design to create a practical and efficient solution. The system not only meets its objectives but also sets the stage for continuous improvement, fostering innovation and collaboration.

# Glossary

|  |  |
| --- | --- |
| Term | Definition |
| Accessibility Compliance | Adherence to guidelines and standards that ensure software is usable by individuals with disabilities. |
| CIA (Confidentiality, Integrity, Availability) | A core principle of information security ensuring that information is kept secret, unaltered, and accessible to authorized users when needed.  (Washington University in St. Louis, n.d.) |
| Figma Prototype | A design prototype created using the Figma tool, which allows teams to collaborate on user interface design. |
| GDPR (General Data Protection Regulation) | European Union legislation that governs the processing and protection of personal data.  (European Union, 2016) |
| HTTPS (Hypertext Transfer Protocol Secure) | An extension of HTTP that uses encryption (SSL/TLS) to secure data transmitted between a web browser and a server, ensuring privacy and data integrity.  (CloudFlare, n.d.) |
| Implementation Details | Specifics of how a software system is built, including hardware, software, and programming choices. |
| Internationalization | The process of designing software so it can be adapted to different languages and regions without engineering changes. |
| Laravel Herd | A local development environment specifically tailored for Laravel projects. It simplifies the setup of PHP, databases, and other dependencies needed for Laravel development on a local machine.  (Laravel, BeyondCode, n.d.) |
| Livewire Framework | A Laravel-specific framework that enables dynamic, reactive user interfaces without writing JavaScript. It integrates directly into Laravel views, allowing real-time interactions and data binding.  (LiveWire, n.d.) |
| MIME (Multipurpose Internet Mail Extensions) | A standard for formatting files and messages so they can be transmitted over the Internet. Commonly used for specifying the type of content in emails and web pages.  (Lenovo, n.d.) |
| MoSCoW Method | A prioritization technique used in project management that categorizes requirements into Must Have, Should Have, Could Have, and Won't Have.  (GeeksForGeeks, 2024) |
| MySQL | A popular open-source relational database management system (RDBMS) used to store, organize, and retrieve data for applications. It uses structured query language (SQL) for database operations.  (ORACLE, n.d.) |
| Non-Functional Requirements (NFRs) | Criteria that define the quality attributes of a system, such as performance, security, usability, and scalability.  (GeeksForGeeks, 2024) |
| PHP Laravel Framework | A web application framework written in PHP that provides an elegant syntax, a range of built-in tools, and modular packages to streamline web development tasks such as routing, authentication, and database interaction.  (Laravel, n.d.) |
| Relational Database | A type of database that uses structured tables to store and manage data, with relationships between the tables. |
| SLA (Service Level Agreement) | A formal agreement between a service provider and a client that defines the level of service expected, such as uptime guarantees and response times.  (atlassian., n.d.) |
| SMTP (Simple Mail Transfer Protocol) | A protocol used for sending emails across the Internet.  (Amazon, n.d.) |
| Tailwind CSS | A utility-first CSS framework used to build responsive and modern user interfaces.  (Tailwind, n.d.) |
| Use Case Diagram | A visual representation of the interactions between users (actors) and the system, showing the system’s functionalities.  (GeeksForGeeks, 2024) |
| User-Friendly Interface | An intuitive and accessible interface designed to provide a positive user experience. |

# Bibliography

Amazon. (n.d.). *What is SMTP*. Retrieved from aws.amazon: https://aws.amazon.com/what-is/smtp/

atlassian. (n.d.). *Service request managenent*. Retrieved from atlassian.: https://www.atlassian.com/itsm/service-request-management/slas

CloudFlare. (n.d.). *What is https?* Retrieved from CloudFlare: https://www.cloudflare.com/learning/ssl/what-is-https/

European Union. (2016, 05 04). *Euro Lex*. Retrieved from Europa.eu: https://eur-lex.europa.eu/eli/reg/2016/679/oj

GeeksForGeeks. (2024, 5 28). *Moscow Prioritization in product management*. Retrieved from GeeksForGeeks: https://www.geeksforgeeks.org/moscow-prioritization-in-product-management/

GeeksForGeeks. (2024, 10 04). *Non-functional requirement sin software engineering*. Retrieved from GeeksForGeeks: https://www.geeksforgeeks.org/non-functional-requirements-in-software-engineering/

GeeksForGeeks. (2024, 10 14). *UseCase Diagram*. Retrieved from GeeksForGeeks: https://www.geeksforgeeks.org/use-case-diagram/

Laravel. (n.d.). *Homepage*. Retrieved from Laravel: https://laravel.com/

Laravel, BeyondCode. (n.d.). *Documentation*. Retrieved from LaravelHerd: https://herd.laravel.com/docs/windows/1/getting-started/about-herd

Lenovo. (n.d.). *What is MIME?* Retrieved from Lenovo: https://www.lenovo.com/us/en/glossary/what-is-mime/

LiveWire. (n.d.). *HomePage*. Retrieved from LiveWire: https://laravel-livewire.com/

ORACLE. (n.d.). *homepage*. Retrieved from MYSQL: https://www.mysql.com/

Tailwind. (n.d.). *HomePage*. Retrieved from TailwindCSS: https://tailwindcss.com/

Washington University in St. Louis. (n.d.). *Confidentiality, Integrity, and Availability: The CIA Triad*. Retrieved from informationsecurity: https://informationsecurity.wustl.edu/items/confidentiality-integrity-and-availability-the-cia-triad/