# Integrative Project of the 4th Semester
# System Specification

Licenciatura em Engenharia Informática, ISEP - 2022/2023

Porto, March 12, 2023

Alexandre Bragança (ATB)

Version 11, 2023-03-01

# Revision History

| Revision | Date | Author(s) | Description |
|---|---|---|---|
| 1 | 2023-01-23 | atb | initital version based on document from Paulo Maio (PSM) with inputs from PAG, LLF and ASC |
| 2 | 2023-01-27 | atb | updates diagram with proposed architecture - ASC |
| 3 | 2023-01-30 | atb | several updates in the text - PAG |
| 4 | 2023-01-31 | atb | several updates in the text - AMD |
| 5 | 2023-01-31 | atb | added reference to protocol specification - ASC |
| 6 | 2023-02-01 | atb | further detail about non-functional requirements |
| 7 | 2023-02-02 | atb | updates on the requirements |
| 8 | 2023-02-03 | atb | adds NFR13 - LLF |
| 9 | 2023-02-03 | atb | updates NFR10 and adds NFR14 - ASC |
| 10 | 2023-02-03 | atb | updates requirements FRE01, NFR09 and FRE04 - AMD |
| 11 | 2023-03-01 | atb | updates supplementary information - PAG |

# Contents

# List of Figures

Chapter 1

# Context

In the 2022-2023 academic year, the fourth semester (i.e. 2nd year, 2nd semester) of the Degree in Informatics Engineering (LEI) of the Instituto Superior de Engenharia do Porto (ISEP) adopts a teaching-learning process based on the development of a single project that enhances the integration and application of knowledge, skills and competencies of all course units taught through the semester: Applications Engineering (EAPLI), Laboratory and Project IV (LAPR4), Languages and Programming (LPROG), Computer Networks (RCOMP) and Computer Systems (SCOMP). The project, common to all course units, consists of developing the system described in this document in accordance with the general procedures described earlier in the document (written in Portuguese) with the same designation, whose subtitle is "Descrição de Funcionamento".

Chapter

# 2 System Description

**Learning Systems (LS)**[1] is an IT company specialized in the development of learning solutions. During the pandemic years, and because of the lockdowns and the movements towards remote working, the company identified new opportunities for supporting remote learning.

Resulting from these opportunities, the company intends to explore a new innovative software solution for remote learning. For the moment the goal is to develop, in an exploratory way, a new remote learning platform called **eCourse**, using a minimum viable product (MVP) approach. The results of this exploratory development will be of high value and the basis for how the company will evaluate the viability of the project as a possible commercial solution.

The company set the duration of this MVP project as 3 months.

The platform should support the activities of the major actors of a course, mainly teachers, students and managers.

Among several functionalities (described in the next chapter), the new platform should have the following distinctive features:

- **Automated Exams** The platform should support a fully automated approach for exams and evaluation. Teachers should be able to design exams using a simple and intuitive language and students should be able to take these exams and automatically have feedback and resulting grades after submitting them. The platform should also present dashboards about exam results both to students and teachers. Figure 2.1 presents an illustration of an online exam.

- **Shared Boards** Boards are one of the main used tools for teaching. From the old chalk boards to the new connected digital boards, boards are one of the most successful teaching tool. The project aims to implement the concept of shared board, as a board that can

[1] Fictional company.

be used to share and organize ideas and information. Figure 2.2 presents an illustration of a shared board used for managing projects.

Figure 2.1: Automated Exam and Evaluations (image from freepik.com)


Figure 2.2: A Shared Board Used for Managing a Project


Taking these objectives into account, the goal of this project it to develop a functional prototype that (i) allows validating the main ideas exposed in this document and (ii) constitutes, by itself, the pillars of what a possible commercial system could be.

Chapter

# 3 Requirements

This chapter presents the requirements for the solution divided in functional requirements and other type of requirements.

> ☞ VERY IMPORTANT: **All requirements (either functional or non-functional) must follow best practices** for the **software engineering process** (e.g., analysis, design, implementation, tests, deployment) as well as the **implementation/-coding (e.g., code quality)**.

## 3.1 Functional Requirements

### 3.1.1 Users

This set of functionalities related to the management of users, It should be only available to managers of the system.

- **FRU01 - Management of Users** Create, disable/enable, and list users of the system (Teachers and Students, as well as Managers)

### 3.1.2 Courses

These are the functionalities related to courses.

- **FRC01 - Create Course** Different editions of a course are to be considered different courses (e.g., Intro-Math-Sem01, Intro-Math-Sem02). Only managers are able to execute this functionality.
- **FRC02 - Open/Close Enrollments in Course** Only managers are able to execute this functionality.

- **FRC03 - Open/Close Course** Only managers are able to execute this functionality.
- **FRC04 - Set Course Teachers** Each course may have several teachers and must have only one Teacher in Charge. Only managers are able to execute this functionality.
- **FRC05 - List Courses** All users are able to execute this functionality.
- **FRC06 - Bulk Enroll Students in Course** This can be made by importing a csv file with students. Only managers are able to execute this functionality.
- **FRC07 - Request Enrollment in Course** The student requests to be enrolled in a course
- **FRC08 - Approve/Reject Enlistment in Course** A manager approves or rejects a students' application to a course.
- **FRC09 - Schedule of Class** A teacher schedule a class (always a recurring class, happens every week). System must check if the Teacher is available for the class period
- **FRC10 - Schedule of Extra Class** A teacher schedule an extraordinary class. An extra class is a one occurrence only and it has a specific group of participants (a subset of the class students)
- **FRC11 - Update Schedule of Class** A teacher changes the time of a specific class (only changes a specific occurrence of a recurring class).

### 3.1.3 Exams

These are functionalities related to exams, its specification and execution.

- **FRE01 - Create Exam** A Teacher creates a new exam. This includes the specification of the exam (i.e., its structure, in accordance with a grammar for exams that is used to validate the specification of the exam)
- **FRE02 - List Exams** The system displays to a student his/her future exams
- **FRE03 - List Course Exams** The system displays to a teacher of a course all the exams of the course
- **FRE04 - Take Exam** A Student takes an exam and answer its questions. At the end of the exam, the system should display the feedback and result (i.e., grade) of the exam. The feedback and grade of the exam should be automatically calculated by a parser based on the grammar defined for exams structure.
- **FRE05 - List Grades** The system displays to a students his/her grades

- **FRE06 - List Course Grades** The system displays the grades of a class (to a teacher of that course)

### 3.1.4 Shared Boards

These are functionalities related to shared boards.

- **FRB01 - Create Board** A user creates a board
- **FRB02 - Share Board** A user shares a board it owns with other users
- **FRB03 - View Board** The system should display, in "real-time", the shared boards available to the user in a HTML page
- **FRB04 - Create Post-it** A user creates a post-it on a board
- **FRB05 - Change Post-it** A user changes a post-it
- **FRB06 - Undo Post-it Change** A user undo the last change in a post-it
- **FRB07 - View Board History** A user views a history of updates in a board
- **FRB08 - Archive Board** The owner of a board can archive the board

### 3.1.5 Meetings

These are functionalities related to the schedule of meetings.

- **FRM01 - Schedule a Meeting** A user schedules a meeting. The system must check if all participants are available and send invitations to participants.
- **FRM02 Cancel Meeting** A user cancels a meeting that he owns
- **FRM03 Accept/Reject Meeting** A user accepts or rejects an invitation to a meeting
- **FRM04 List Participants** The system displays the lists of participants in a meeting and the response status (accept or reject meeting)

## 3.2 Other Requirements

This section presents some specific non-functional requirements. This includes some constraints and concerns that should be taken into account when designing and implementing the solution.

- **NFR01 - Programming language** The solution should be implemented using Java as the main language. Other languages can be used in accordance with more specific requirements.
- **NFR02 - Technical Documentation** Project documentation should be always available on the project repository ("docs" folder, markdown format) and, when applicable, in accordance to the UML notation. The development process of every US (e.g.: analysis, design, testing, etc.) must be reported (as part of the documentation).
- **NFR03 - Test-driven development** The team should develop a relevant set of automated tests for every US / Class / Method. The team should aim to adopt a test-driven development approach.
- **NFR04 - Source Control** The source code of the solution as well as all the documentation and related artifacts should be versioned in a GitHub repository to be provided to the students. Only the main (master) branch will be used (e.g., as a source for releases)
- **NFR05 - Continuous Integration** The Github repository will provide night builds with publishing of results and metrics.
- **NFR06 - Deployment and Scripts** The repository should include the necessary scripts to build and deploy the solution in a variety of systems (at least Linux and Windows). It should also include a `readme.md` file in the root folder explaining how to build, deploy and execute the solution.
- **NFR07 - Database** By configuration, the system must support that data persistence is done either "in memory" or in a relational database (RDB). Although in-memory database solutions can be used during development and testing, the solution must include a final deployment where a persistent relational database is used. The system should have the ability to initialize some default data.
- **NFR08 - Authentication and Authorization** The system must support and apply authentication and authorization for all its users and functionalities.
- **NFR09 - Exam Language** The support for exams (its design, feedback and grading) must follow specific technical requirements, specified in LPROG. The ANTLR tool should be used (`https://www.antlr.org/`).
- **NFR10 - Shared Board Architecture** This functional part of the system has very specific technical requirements. It must follow a client-server architecture, where a client application is used to access the shared boards that should be implemented in a server. Communications between these two components must follow specific protocol

described in a document from RCOMP ("Application Protocol"). Also, the client application can not access the relational database, it can only access the server application. The client app should implement an HTTP server to be able to generate the "views" of the boards. This should be done automatically, without the intervention of the users (i.e., without reload of the web page) by using AJAX.

- **NFR11 - Explore Shared Board Synchronization** This functional part of the system has very specific technical requirements, particularly some concerns about synchronization problems. In fact, several clients will try to concurrently update boards. As such, to explore and study this concurrency scenario a "model" of a solution must be implemented and evaluated exceptionally using the C programming language, and using processes and semaphores. Specific requirements will be provided in SCOMP.
- **NFR12 - Presentation and Communication** This project has some specific requirements regarding communication and presentation of the project and its results. This is a concern of the project and its related to the presentations for the sprint reviews in the context of LAPR4. LAPR4 will provide further specification for this requirement.
- **NFR13 - Design and Implement Shared Board Synchronization** This functional part of the system has very specific technical requirements, particularly some concerns about synchronization problems. In fact, several clients will try to concurrently update boards. As such, the solution design and implementation must be based on threads, condition variables and mutexes. Specific requirements will be provided in SCOMP.
- **NFR14 - Deployment** The solution should be deployed using several network nodes. It is expected that, at least, the relational database server and the shared board server be deployed in nodes different from localhost, preferably in the cloud.

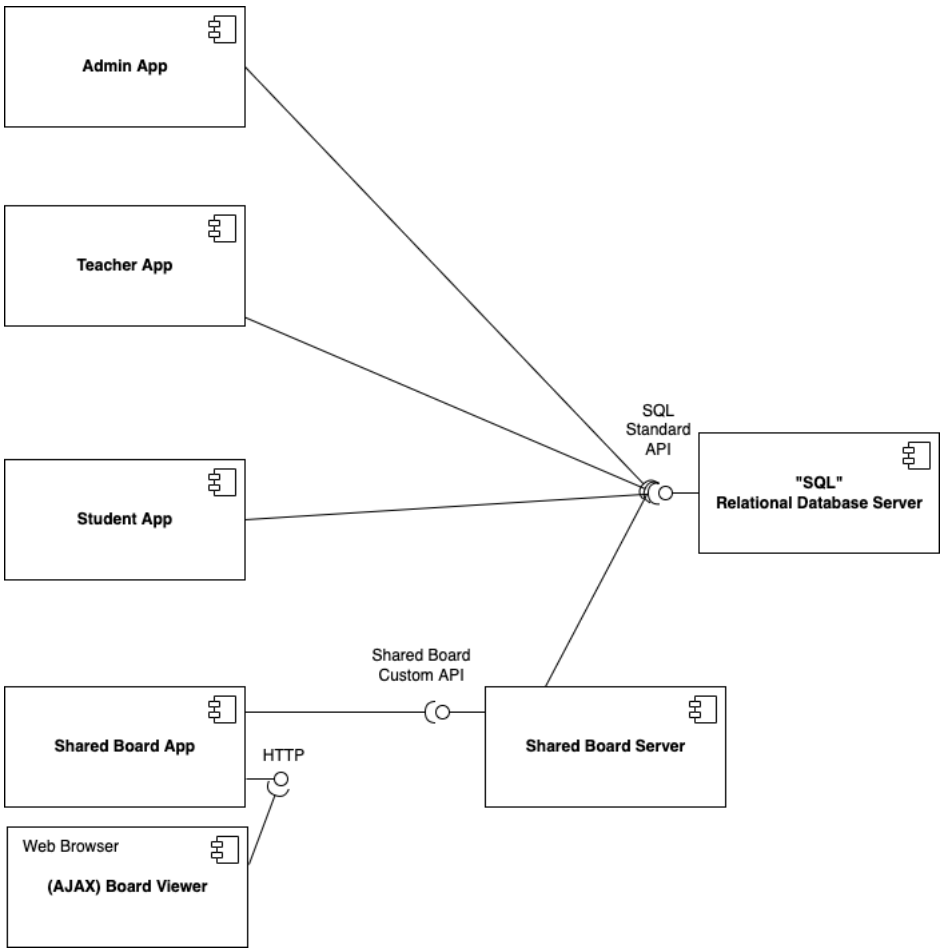
Chapter

# 4 Envisioned Applications

Figure 4.1 presents the envisioned set of applications (i.e., components of the system).

Figure 4.1: Envisioned Applications

.



The Admin app is used by managers to manage courses, users and enrollment of students. The teacher and student apps are used for the specific functionalities of each respective actor These three apps connect directly to the database.

Shared boards follow a specific design (as described in NFR07). There is a separation between a frontend (Shared Board App) and a backend (Shared Board Server). The server implements the shared boards and receives updates from the clients. As updates are executed in the server, the server notifies the clients of these updates. As such, all clients are able to maintain a "real-time" clone of the shared boards. The Shared Board App implements a "small" HTTP server that serves a page that displays a board view.

Chapter

# 5 Supplementary Information

This chapter presents supplementary and more detailed specification regarding what the solution should support.

## 5.1 General Description

This section presents an overview of the system in the perspective of its major human actors.

### 5.1.1 Managers

Managers (they can be also named as Administrators) manage all the users of the system as well as courses and students enrollment. All users should be identified in the system by their email. Each user should also provide its full name and short name. In order to be authenticated by the system the users must also provide a password. Each course should have a unique identifier (automatically generated) and a unique name. A course may be open or closed. A closed course can not have any activity. Each course must have a teacher in charge, but may also have other teachers. Courses are not divided into sections (i.e., "turmas"). Students of the course participate in all classes of the course. A course has also a small textual descriptions of its contents. Courses may have a minimum and a maximum number of enrolled students. This may limit the possibility of opening (i.e. starting) courses that do not satisfy the limits.

All users can create and use boards as well as meetings.

### 5.1.2 Teachers

Teachers are responsible for scheduling classes for their courses. A class is always a recurring weekly event. A class must have a unique title, a date and a duration. It should be impossible to schedule classes that are coincident with other classes of the same course. The system should

also warn if one of the participants in the class (either a teacher or a student) has other classes at the same time. Teachers can schedule extra classes.

Teachers can create exams. An exam is related to a specific course. It has a unique title and a small description. Exams have also a open and a close date. The open date is the time when students can start to take the exam. The close date is the deadline for students to submit the exam. The specific rules for designing, executing and evaluation exams are described in Section 5.2.4.

A Teacher is characterized by his/her name, date of birth, tax payer number and an acronym inputed by the administrator (e.g., "AALB").

All users can create and use boards as well as meetings.

### 5.1.3 Students

Students enroll in courses. They take exams. They can view the results of their exams.

A student is characterized by his/her name, date of birth, tax payer number and a mechanographic number assigned automatically by the system based on the year of registration and a sequential number, e.g., "202300001".

All users can create and use boards as well as meetings.

## 5.2 Specific Concepts

This section presents details about the main concepts of the system.

### 5.2.1 Course

A course is characterized by its title and a unique code inputed by the administrator, e.g., "JAVA-1".

The usual workflow related to the course can be illustrated as follows:

event create (-> state close) -> event open (->state open) -> event open enrollments (-> state enroll) -> event close enrollments (-> state in progress) -> event close (-> state closed)

### 5.2.2 Classes and Meetings

Classes and meetings are events that happen in some time and have a duration. They also have participants. However there is no concept of location related to classes and meetings. They do not take place in a specific location and "nothing" happens at the time of the event.

### 5.2.3 Shared Boards

A shared board is a digital implementation of a post-it board (see Figure 2.2). It has a unique title. It is divided into a certain number of columns and rows. Therefore it has a certain number

of cells. For the moment it makes sense to define the maximum number of rows to 20 and of columns to 10. But this should be a setting in a property file. Columns and rows may have titles. They may also be identified by an integer number from 1 to the maximum number.

The user that creates the board is its owner. The owner can share the board with other users. Users may have read or write access to the board.

Users with write permission may post content to a cell in the board. The content can be a text or an image. When the server commits a post it also should notify all clients with access to the board of the update.

The system should maintain an history/log of all the updates in the board.

The system should generate "real-time" HTML rendering of the board, presenting all the actual contents of the board, the row and column titles, the board title, the authors (short name and email) of the posts and a timestamp of each post. This "real-time" HTML rendering should be available in the server and in each client.

The user who created a post in a cell can change that post. It may change its contents or move the post into a free cell. The user can also undo the last change.

The owner of a post can delete it.

The owner of a board can archive the board when it will no longer be used.

### 5.2.4 Exams

For the moment the goal is to support online exams that can be automatically graded.

The system must provide a language to support the specification and "execution" of exams. The language must support the **design of the exam layout and its questions** as well as **solutions, feedback and grading**. The system must also support the **automatic production of feedback and grading for the answers** given by students when they take the exam. Usually this is done at the end of the exam. Therefore, you must analyze the problem and design a solution that supports the two goals: validate the structure of the exam; evaluate the answers of each exam (giving feedback and calculating the grade).

Figure 5.1 presents an illustration of a running exam (in red the parts of the test specification).

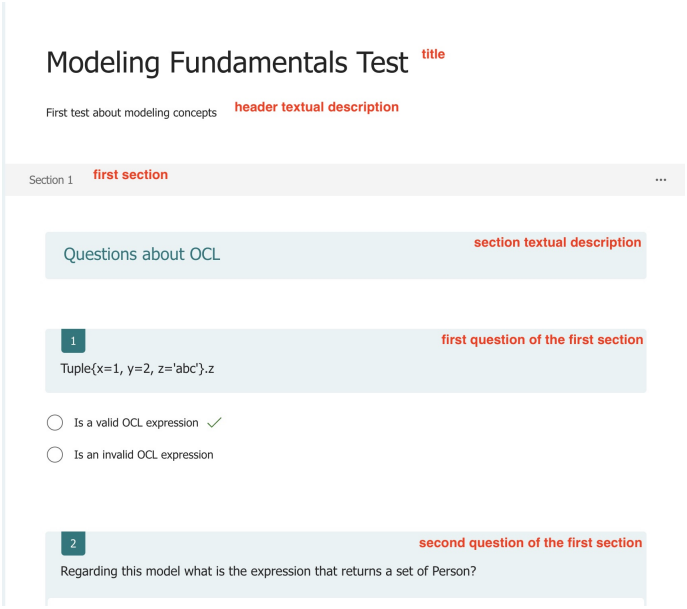
#### Exam Organization

Each exam specification must include a set of components or sections:

- **Title** The title of the exam specification.
- **Header** This component must include global settings, such as the type of feedback (none, on-submission, after-closing) and the type of grade (none, on-submission, after-closing). It may also contain a textual description to appear at the beginning of the exam.

Figure 5.1: Example Exam

.




- **Sequence of Sections** After the header, the exam is composed of sequence of one or more sections. Each section is a group of questions. Each section may contain a textual description to appear at the beginning of the group of questions. The type of questions that can be used in the sections is described next.

### Question Types

The solution should be based on the following type of questions that are inspired on the Moodle platform. You should support, as much as possible, the features that are supported in Moodle, as described in the following items.

- **Matching** - A list of sub-questions is provided, along with a list of answers. The respondent must "match" the correct answers with each question.
  - `https://docs.moodle.org/401/en/Matching_question_type`.
- **Multiple Choice** - With the Multiple Choice question type you can create single-answer and multiple-answer questions and weight individual answers.
  - `https://docs.moodle.org/401/en/Multiple_Choice_question_type`.
- **Short Answer** - In response to a question, the respondent types a word or phrase. There may be several possible correct answers, with different grades. Answers may or may not be sensitive to case.
  - `https://docs.moodle.org/401/en/Short-Answer_question_type`.

- **Numerical** - From the student perspective, a numerical question looks just like a short-answer question. The difference is that numerical answers are allowed to have an accepted error. This allows a continuous range of answers to be set.
  - `https://docs.moodle.org/401/en/Numerical_question_type`.
- **Select Missing Words** - Students select a missing word or phrase from a dropdown menu. Items may be grouped and used more than once.
  - `https://docs.moodle.org/401/en/Select_missing_words_question_type`.
- **True/False** - In response to a question, the respondent selects from two options: True or False.
  - `https://docs.moodle.org/401/en/True/False_question_type`.