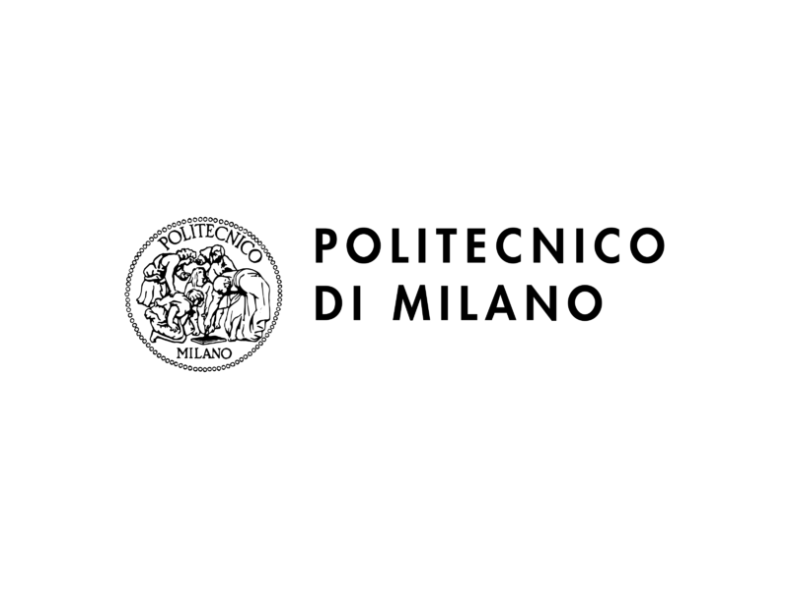
|  |
| --- |
| Petulicchio Lorenzo – Talacci Mattia  [Data] |
| Automation and Control Engineering |
| Requirements Analysis and Specification Document |
| Software Engineering Project |

 Immagine che contiene Carattere, testo, Elementi grafici, design

Descrizione generata automaticamente

Sommario

[1 - INTRODUCTION 2](#_Toc139710988)

[1.1 - Document structure 2](#_Toc139710989)

[1.2 - Project purpose 2](#_Toc139710990)

[1.3 - Current situation and problems 2](#_Toc139710991)

[1.4 - Targets 4](#_Toc139710992)

[1.5 General description 4](#_Toc139710993)

[1.6 Users characteristics 5](#_Toc139710994)

[2 - FEASIBILITY STUDY 6](#_Toc139710995)

[2.1 - Technical and technological Feasibility 6](#_Toc139710996)

[2.3 – Student Survey Responses 6](#_Toc139710997)

[3 - USE CASES AND SCENARIOS 8](#_Toc139710998)

[3.1 - Scenarios 8](#_Toc139710999)

[3.2 – Use cases 9](#_Toc139711000)

[10](#_Toc139711001)

[4 - LIST OF REQUIREMENTS 10](#_Toc139711002)

[4.1 - External Interface Requirements 10](#_Toc139711003)

[4.1.2 Hardware Interfaces 10](#_Toc139711004)

[4.1.3 Software Interfaces 10](#_Toc139711005)

[4.3 Performance Requirements 10](#_Toc139711006)

[5 - DOMAIN MODEL 11](#_Toc139711007)

[5.1 - Class diagram 11](#_Toc139711008)

[6 – OPTIMIZATION ALGORITHM LOGICS 12](#_Toc139711009)

# 1 - INTRODUCTION

This document constitutes the Requirement Analysis and Specification Document (RASD). Its purpose is to analyse the requirements that will lay the foundations of application services, to specify the application domain, the entities involved and their relationship, to clearly explain the objectives, the constraints and the features that are going to be implemented.

## 1.1 - Document structure

The document is divided into 6 chapters. The first one provides an overview of the current situation to deal with, describing the job that the software will facilitate pointing out the target and explaining generally it and which are the entities.

The second chapter is about feasibility study, here the feasibility of the project is discusses in terms of technical and technological aspects. Moreover is present a section regarding a survey for student that helps to evaluate how to define the commitment required by an exam.

Then are presented the use cases and scenarios to understand better the possible situation of usage

The fourth chapter regards the requirements, they are listed divided by type.

The fifth one contains the class diagram and describes the whole system.

While the last one presents the logic of the optimization algorithm.

## 1.2 - Project purpose

In the university context the exam session is the most important period for students, they have a specific range of time in which they can take their exam and in most of the cases a specific date and time slot is assigned to each exam so the student has to decide when to take it. The cases in which the student has a significant number of exams are many and the cases in which a student has to take exams of past semesters are common too.

To decide when an exam is scheduled a complex process is used by the registrar’s office of Politecnico and this brings inefficiency in particular because the student feedback is not taken into account.

The purpose of SchedulEx is to automate the process of scheduling the exam calendar session in Politecnico di Milano. The main idea is to have a web app that helps the university during the exam session scheduling. Using it the work flow is reduced and it prevents several revisions of the calendar that takes time that employees can dedicate to other activities and also the professors and students can plan in advance the exam session.

This project could not only help employees of the registrar's office in their job but it aims to facilitate students with their own session planning too. This is due to the fact that different types of information regarding the exam commitment are processed, they are directly obtained from a batch of students. The project could also have secondary purposes, for example managing in the best way possible the presence of students in Politecnico facilities.

## 1.3 - Current situation and problems

Actually, at Politecnico di Milano the registrar’s office takes care to create the exams schedule for each exam session. This job is manually and not so efficient. Indeed, it takes a long time and it requires different reviews. The exams to schedule are split among personnel of the registrar's office, after this, the exams scheduling are put together and each one makes sure that there is no overlapping.

In this way a first draft is obtained, then it is sent to professors who reporting their requirements by requesting respective changes by a certain date beyond which changes can no longer be accepted

At this step, the modified draft is published to the students that analyse it and notify changes to do in order to meet their requirements.

Finally, the examination session schedule is definitively published.

With this situation the requirements taken into consideration are:

* for students:
  + exams of the same semester and same year must have two night of distance each other;
  + The exams of the same course must have at least 14 days of distance.
* for professors:
  + they can give their unavailability for what concerns days and time slot.
* for facilities:
  + the calendar is only predictive and only during the verifying phase the room are assigned;
  + It tries to do a equidistribution weighting with the number of students for each course in order to not have more than a certain number of students simultaneously.

The logic adopted to schedule the exams is: firstly are scheduled intercourses and courses with multiple staggers, secondly the mandatory exams and lastly are scheduled the elective courses exams.

Each exam is characterised only by a number (between 1 and 6) that identifies the semester, the identification year and semester is not used.

**Semester enumeration**

|  |  |  |
| --- | --- | --- |
|  | January-February | June-July |
| 1st year | 1 | 2 |
| 2nd year | 3 | 4 |
| 3rd year | 5 | 6 |

This method brings several problems:

1. A lot of resources in terms of time is spent due to the flow adopted to obtain the final schedule.
2. The students most of the time receive a calendar that if difficult to fit with their necessity, for example the time to prepare well for an exam is not compliant with the distance between some exams.
3. Students receive the exam calendar not very early, so they will make their plan late without having the possibility to study during the semester in a very effective way.

## 1.4 - Targets

Schedulex aims to optimise student’s experience during exam session. To do so the following target are taken as the benchmark for evaluating performance:

1. Maximize the exam distance
   * Creating an optimal session student can plan better when to take an exam
2. Meet the requirements of
   * professors that could have some days of unavailability of need a specific time between two calls of their exam
   * students that can’t have in the exam session calendar two exams with a distance of less than two nights. Moreover two calls of the same exam must have at least fourteen days of distance
   * Politecnico facilities (classroom in particular), because to have a good situation to take an exam there must be number large enough of sits in the classroom
3. Automating a long and labor-intensive process making it:
   * shorter in order to give the calendar to student and professor in advance
   * more robust to variation, because it could happen that is necessary to change something in the planning and it should be done easily.

## 1.5 General description

The software can be used at the beginning of each semester in order to be able to publish the provisional calendar at times compatible with the personal organisation of individual students with regard to their exam session.

The software is designed precisely from the needs of students who are struggling with preparing multiple exams at the same time resulting in high stress levels, etc.

In addition, our software aims to automate a process that is currently done by hand to avoid human error and optimise processing time. The users involved are:

- Active users: *secretariat*

- Passive users: *students, professors* and *Polimi*

The system will mainly be used by the student secretariat and dedicated committee members.

For experimental purposes only, the improvement and consequent optimization of the forecast calendar for the Leonardo site for the 3I school is considered. This is the largest school, it may be extended to other schools in the future.

Only the Automation Engineering calendar is currently being considered;

During the winter and summer exam sessions there are 2 calls per exam, while in fall just one. Since most engineering schools have this limit on the number of calls of each exam. However, the software can also be used for study programmes with a number of calls greater than 2.

The allocation of classrooms is done later, at the time of exam scheduling, a constraint on the maximum number of students simultaneously present at the Politecnico for exams is considered. It is probable that then it is not feasible to manage the classrooms but this is not something that you can know in advance as the student's registration for the calls closes a few days before the date of the exam (this deadline is temporally after the publication of the calendar so in case of problems changes will be made by hand later as needed as in fact is still done by the student secretariat).

## 1.6 Users characteristics

The people involved in the direct or indirect use of the software belong to three types: professors, students and Politecnico employees.

Professors are the PoliMi’s professors who are teachers of a course as written in the study plan card of the course, in this category we include also assistants, tutors or other collaborators of the main professor that help him during the exam.

Students, they are all students enrolled in PoliMi, from the first year to the fifth, without distinction if they are enrolled in Bachelor Degree, Master Degree or Single Courses. Everyone who has to take at least one exam, doesn't care whether they will take an exam during the exam session or which exams they will take, and it doesn’t care if they are in the course or not. Each student that can potentially take an exam in the session in which the software is utilised is in this category. For example, students who are in Erasmus or similar program and can’t take any exam are not in this category as laureandi that have finished their exams.

PoliMi employees are firstly registrar's office personnel and commission personnel

These three classes are divided in this way

Direct user:

* PoliMi employees (registrar's office personnel and commission personnel)

Indirect user:

* Professors
* Students

Direct involved entities

* **Registrar's office employees:** they are the ones that will directly interact with the software, in particular with the GUI because they insert input data, they make the software perform the scheduling and they receive the output (calendar).
* **Professors of commission:** They communicate independently with their colleagues to gather information about unavailability of each professor of each course. Actually is required that professors need to communicate each own unavailability, otherwise it will not consider in scheduling.

Indirect involved entities

* **Students**: they receive the software output (a calendar with all scheduled exams). They can find the calendar on Polimi website, and their role is to decide which exams take and in which call take it
* **Professors**: They have to communicate their unavailability in terms of days or time slot or request about cable classroom necessity to registar’s office via email or form

- Referenti spazi ed aule

# 

# 2 - FEASIBILITY STUDY

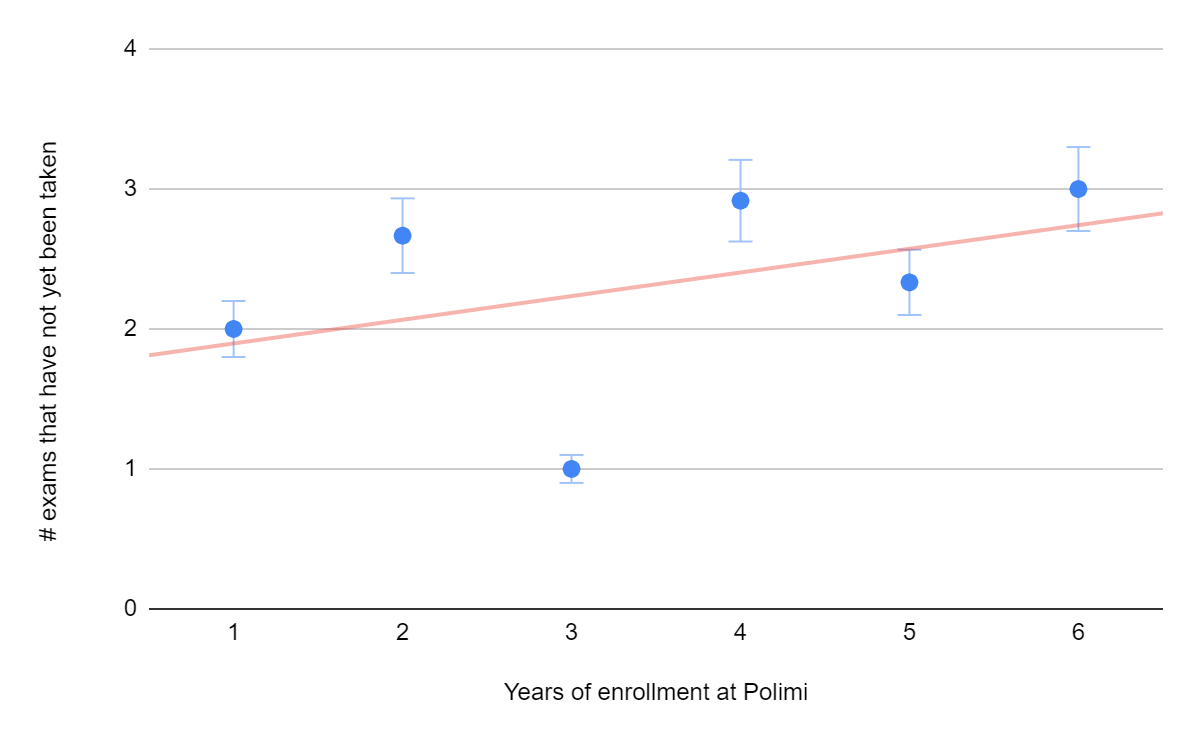
## 2.1 - Technical and technological Feasibility

All the technologies (better explained in the Design Document) are available on the market and largely diffused so the project is feasible in terms of realization and maintenance. The knowledge needed to realize the project are notions from different fields: web app, database, back-end and optimization algorithm. But being very used nowadays it is easy for other people to develop the software.

The critical point lies in the optimization algorithm because depending on the parameters such as number of exams, available days, constraints and weights parameters it is possible that a feasible solution can’t be provided, so having soft constraints and changing the weight parameters can be fundamental to obtain a feasible solution. Moreover, regarding the hardware there aren’t strict requirements, the only things to take care about is the ram of the processor that is necessary to solve the optimization algorithm in a reasonable time.

## 2.3 – Student Survey Responses

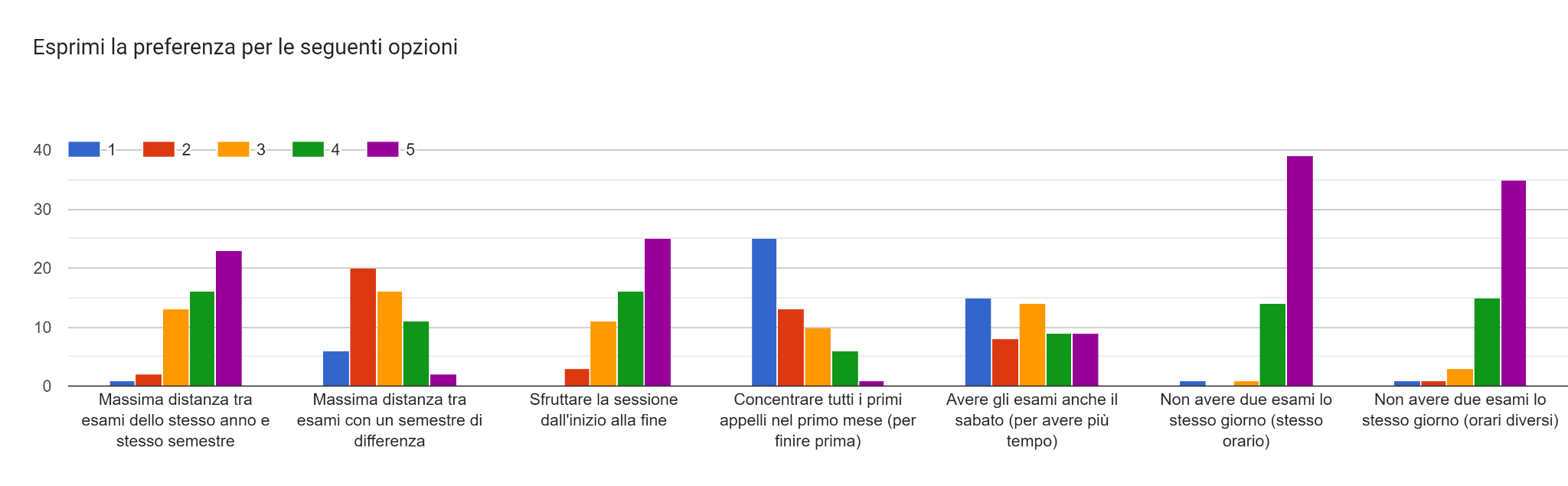
To study the feasibility of the optimization algorithm, in particular the aspects regarding the weights parameters we provided students a survey which is divided into five sections: student profile, current situation, optimal exams session, required commitment of exams, exam planning.

The majority of respondents were students from the Automation and Control Engineering department (56%), with others mainly from the 3I Engineering department. In the **first section** of the survey, our goal was to gather information about the current situation and students' perceptions of the problem. We asked participants to indicate the number of exams they have not yet taken, plotted against their years of enrollment at Polimi. As expected, this number increased linearly with the years of enrollment, reflecting the increasing number of exams to be completed. However, we realised that this question was not well formulated, as there was no distinction made between bachelor and master's degree students.

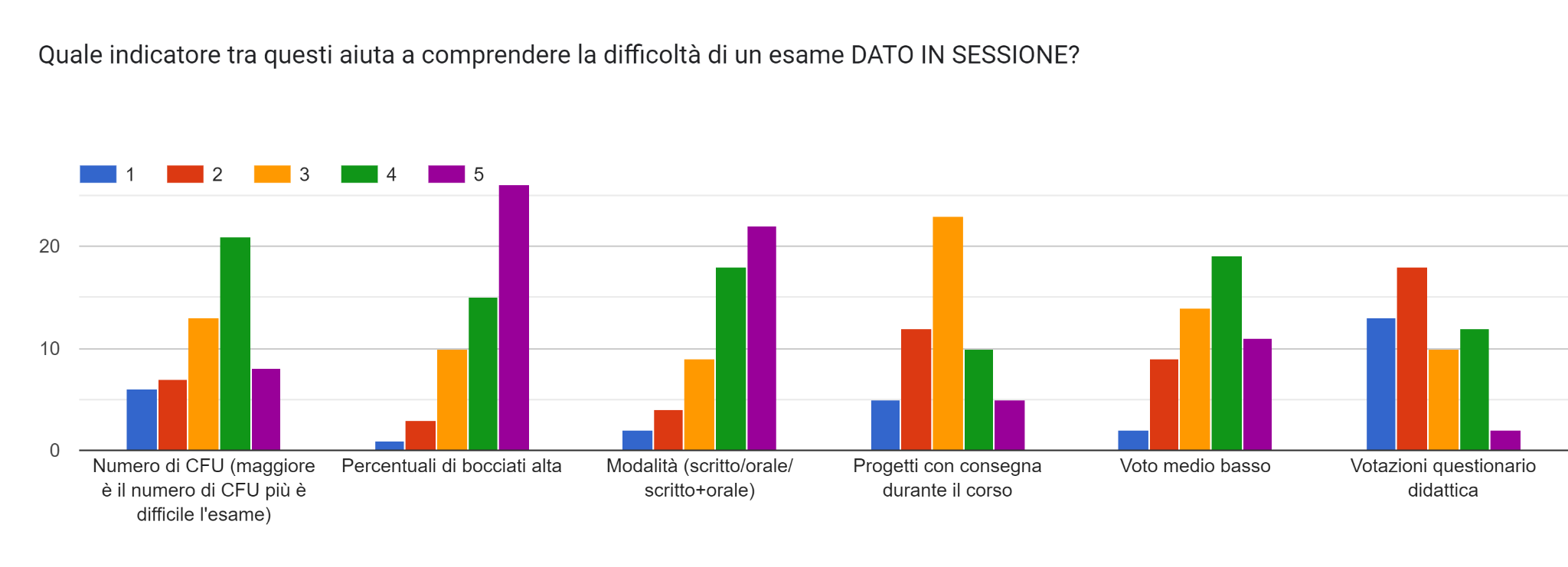
71.2% of the students rated the arrangement of exams during the session as an average of 1 or 2 (on a scale of 1 to 4), associating the following main issues:

|  |  |
| --- | --- |
| *Exams too close together* | 70.9% |
| *Poorly utilised session period* | 47.3% |
| *First call date too close to end of lectures* | 41.8% |
| *Overlapping exams* | 41.8% |
| *Methods of examinations that overlap* | 36.4% |

In the **second section**, we asked students to evaluate potential targets to focus our optimization algorithm on. As predicted, the most requested aspects were exam overlaps and distances between exams. Although the option "Maximizing the use of the entire exam session" was selected as very important, students did not have a clear stance on utilizing Saturdays to make the most of the exam period.



In the **third section** we try to identify possible parameters to evaluate the difficulty of an exam and his relative required effort, in an objective way.

Through the responses in this section, we can perform more targeted parameter tuning based on the available data. Indeed, “CFU” credits are not considered an objective measure indicating the actual amount of study time required for a specific exam, but at least can be a first metric for understanding the size of a course. The "high failure rate" is considered the most relevant factor for our purpose, unlike the "teaching evaluation questionnaire" and "low average grade". It would be interesting to further develop the "Examination mode" category, as it was highly selected by students, but since it lacks differentiation between options (written, oral, written/oral), it will not be taken into consideration in this initial development phase.We tried to ask how many days it took to prepare for exams categorised as "little or very effort demanding," but the results are very uneven and of little use to an implantation in this early development.

# 3 - USE CASES AND SCENARIOS

## - Scenarios

In this section are presented the possible scenarios, being the software used exclusively by the registrar’s office the main use cases regard it but are listed also cases in which students or professors interact with the output of the software or with the directly involved users.

1. It is time to start scheduling the exam session for the 3i school, Professor Svelto creates a session to schedule the associated calendar, together with the members of the reference committee, Alice, Mario and Carlo. Everyone will need to collect the unavailability of professors and be able to initialize problems related to that school.
2. Professor **Svelto**, after communicating with the school dean, initializes the scheduling problem for School 3I with the session start and end dates present on the academic calendar.
3. Professor **Franchi** had a previously unscheduled commitment that does not allow him to be able to be there on the 20th of June. Since it is a possible exam day and the deadline for being able to report unavailabilities has not yet passed, the professor notifies the student secretary of his new unavailability to be changed. **Mario** receives Professor **Franchi**'s request to add unavailabilities and enters the new unavailability in the associated session.
4. Professor **Amaldi** will hold a conference at a foreign university that does not allow him to be present from July 25-30. **Carlo** receives a request to add Professor **Amaldi**'s unavailability and places the new unavailability in the associated session.
5. Professor **Clericetti** also teaches in a science high school, and is unavailable during the session period every week on Tuesdays. **Alice** receives the request to add Professor **Clericetti**'s unavailabilities and enters the new recurring unavailability into the associated session.
6. On Saturday, July 8, **Politecnico** organized an open-day for high school students approaching their postgraduate choice. For this reason, some classrooms in some buildings on the Leonardo campus will not be available. **Mario** adds this unavailability to the problem session associated.
7. Professor **Giannetto** had told the student secretary in early March that he would have a commitment in the second week of July, but this commitment has been cancelled so he tells the student secretary that he is available during the period he had previously stated as unavailable. **Alice** receives the change from Professor **Giannetto** that he is available at a time when he had previously said he was unavailable and changes the professor's unavailability to the associate session.
8. Having collected all the unavailabilities, and compared with colleagues, **Carlo** starts the automatic scheduling process.
9. The chair publishes new rules regarding the distance between two appeals and the number of appeals of the same exam in a single session, the student secretary makes these changes and starts a new session, to get a new schedule.
10. Having obtained a schedule, Professor **Svelto** wants to post the resulting calendar with exam-related dates on the school website for all students to view.

## 3.2 – Use cases

1. **Create ProblemSession**

|  |  |
| --- | --- |
| *Participating actors* | * Secretary Users |
| *Entry Condition* | Usually, an exam session calendar must be created some months before the actual start |
| *Flow of events* | * **Secretary User** activates the “Create ProblemSession” function * **SchedulEx** respond by presenting the GUI to insert all the needed information. * **Secretary Users,** once they logged in, can open the same page created before to add any information to it |
| *Exit Condition* |  |

1. **Set StartEndDate**

|  |  |
| --- | --- |
| *Participating actors* | * Secretary User * School Dean (*indirect user)* |
| *Entry Condition* | A ProblemSession must exist |
| *Flow of events* | * **Secretary User** activates the “Open ProblemSession” and **SchedulEx** return the GUI for the relative ProblemSession. * Here he clicks on the button and activates “set StartEndDate”, is returned a calendar in which he can select easily the start and the end of the exam session * After Prof Svelto confirmation, the selected dates are stored by **SchedulEx** |
| *Exit Condition* | An acknowledgement of the data saving that occurred is visualized |

1. **Add Unavailability – Single day**

|  |  |
| --- | --- |
| *Participating actors* | * Secretary User * Professor (*indirect user)* |
| *Entry Condition* | A ProblemSession must exist |
| *Flow of events* | * ***Professor***communicates to the secretary his unavailability. * **Secretary User** activates the “Open ProblemSession” that involves Prof Franchi and **SchedulEx** return the GUI for the relative ProblemSession. * Here he clicks on button and activates the “Add Unavailability”. * **Schedulex** return a form in which **Secretary User** can defined the type (professor), the name (Franchi), and dates in which professor is unavailable. * Thanks to a GUI, **Secretary User** specifies the involved day, and add to the Unavailability form * After **Secretary User** confirmation, the selected dates are stored by **SchedulEx** |
| *Exit Condition* | An acknowledgement of the data saving that occurred is visualized |

1. **Add Unavailability – Period**

|  |  |
| --- | --- |
| *Participating actors* | * Secretary User * Professor (*indirect user)* |
| *Entry Condition* | A ProblemSession must exist |
| *Flow of events* | * ***Professor***communicates to the secretary his unavailability. * **Secretary User** activates the “Open ProblemSession” that involves Prof Franchi and **SchedulEx** return the GUI for the relative ProblemSession. * Here he clicks on button and activates the “Add Unavailability”. * **Schedulex** return a form in which **Secretary User** can defined the type (professor), the name (Franchi), and dates in which professor is unavailable. * Thanks to a GUI, **Secretary User** specifies the period, and add to the Unavailability form * After **Secretary User** confirmation, the selected dates and are stored by **SchedulEx** |
| *Exit Condition* | An acknowledgement of the data saving that occurred is visualized |

1. **Add Unavailability – Recurrent week day**

|  |  |
| --- | --- |
| *Participating actors* | * Secretary User * Professor (*indirect user)* |
| *Entry Condition* | A ProblemSession must exist |
| *Flow of events* | * ***Professor***communicates to the secretary his unavailability. * **Secretary User** activates the “Open ProblemSession” that involves **Professor** and **SchedulEx** return the GUI for the relative ProblemSession. * Here he clicks on button and activates the “Add Unavailability”. * **Schedulex** return a form in which **Secretary User** can defined the type (professor), the name (Franchi), and dates in which professor is unavailable. * Thanks to a GUI, **Secretary User** specifies the day of recurring unavailability, and add to the Unavailability form. * After **Secretary User** confirmation, the overall dates are computed and stored by **SchedulEx** |
| *Exit Condition* | An acknowledgement of the data saving that occurred is visualized |

1. **Add Unavailability – University Room Unavailability**

|  |  |
| --- | --- |
| *Participating actors* | * Secretary User * Politecnico (*indirect user)* |
| *Entry Condition* | A ProblemSession must exist |
| *Flow of events* | * ***Politecnico***communicates to the secretary that some class will not be able to use during certain dates. * **Secretary User** activates the “Open ProblemSession” that involves the rooms that will be used by the School involved and **SchedulEx** return the GUI for the relative ProblemSession. * Here he clicks on button and activates the “Add Unavailability”. * **Schedulex** return a form in which **Secretary User** can defined the type (University), the identifier of the rooms (e.g. “3.1.3”), and dates in which those rooms are not available. * Thanks to a GUI, **Secretary User** specifies the day of this rooms unavailability, and add to the Unavailability form. * After **Secretary User** confirmation, the overall dates are computed and stored by **SchedulEx** |
| *Exit Condition* | An acknowledgement of the data saving that occurred is visualized |

1. **Modify/Delete Unavailability**

|  |  |
| --- | --- |
| *Participating actors* | * Secretary User * Professor (*indirect user)* |
| *Entry Condition* | A ProblemSession must exist |
| *Flow of events* | * ***Professor***communicates to the secretary his change about his unavailability. * **Secretary User** activates the “Open ProblemSession” that involves **Professor** and **SchedulEx** return the GUI for the relative ProblemSession. * Here he search for the unavailability associated to **Professor** and select it. * **SchedulEx** return a resume of that unavailability in which **Secretary User** can performs some action to modify the input field or delete it. * **Secretary User** activates “Delete unavailability” function and **SchedulEx** perform the action. |
| *Exit Condition* | An acknowledgement of the data deleting that occurred is visualized |

1. **Start Optimization**

|  |  |
| --- | --- |
| *Participating actors* | * Secretary User |
| *Entry Condition* | A ProblemSession must exist. (All information has been entered) |
| *Flow of events* | * **Secretary User** needs to compute the overall calendar considering all information entered before. To do so, he/she activates the “Start Optimization”. * **SchedulEx** start to compute the relative problem session, and provide information on its status. * **Secretary User** cannot modify no more the Problem Session |
| *Exit Condition* | An acknowledgement of the starting process that occurred is visualized. |

1. **Change Settings**

|  |  |
| --- | --- |
| *Participating actors* | * Student Secretary |
| *Entry Condition* | A problem session must exist |
| *Flow of events* | * Student Secretary receives the new settings * **Secretary User** activates the “Open ProblemSession” that involves the rooms that will be used by the School involved and **SchedulEx** return the GUI for the relative ProblemSession. * Here he clicks on button and activates the “Change Settings”. * The secretary user makes the changes * After **Secretary User** confirmation, the overall dates are computed and stored by **SchedulEx.** |
| *Exit Condition* | An acknowledgement of the data saving that occurred is visualized |

1. **Calendar publishing**

|  |  |
| --- | --- |
| *Participating actors* | * Secretary User |
| *Entry Condition* | Problem Session must be scheduled |
| *Flow of events* | * Secretary User takes the scheduled calendar. * Secretary User enters in the Politecnico website. * Secretary User post on the website the calendar. |
| *Exit Condition* | An acknowledgement of the fact that the calendar is online |

# - LIST OF REQUIREMENTS

The goal of SchedulEx, as mentioned above, is to automate the scheduling process of the calendar, leaving users free to collect the necessary information and unavailability in the manner they prefer. The whole part of communication between direct and indirect users is not counted within the software implementation.

## 4.1 – Functional Requirements

1. SchedulEx must let the user to create, modify, delete multiple sessions to compute the Optimal Calendar associated to a specific School.
2. SchedulEx must let the user to modify multiple sessions to compute the Optimal Calendar associated to a specific School.
3. SchedulEx must let the user to delete multiple sessions to compute the Optimal Calendar associated to a specific School.
4. SchedulEx must let the user to login and visualize all the sessions that has been created or create new one.
5. SchedulEx must let the user to insert School name for which the ProblemSession is defined.
6. SchedulEx must let the user to *insert* exam session dates, more precisely the Start and End Date.
7. SchedulEx must let the user to *change* exam session dates, more precisely the Start and End Date.
8. SchedulEx must let the user to add an unavailability.
9. SchedulEx must let the user to delete an unavailability.
10. SchedulEx must let the user to select unavailability type (Professor or Politecnico).
11. SchedulEx must let the user to select professor name for the unavailability from a list or insert a string identifier for the rooms.
12. SchedulEx must not allow to select a professor that doesn’t belong to School previously selected.
13. SchedulEx must let the user to add a single day for a specific unavailability.
14. SchedulEx must let the user to add a multiple day period for a specific unavailability.
15. SchedulEx must let the user to add a recurrent day of the week for a specific unavailability.
16. SchedulEx must let the user to delete a single day for a specific unavailability.
17. SchedulEx must let the user to delete a multiple day period for a specific unavailability.
18. SchedulEx must let the user to delete a recurrent day of the week for a specific unavailability.
19. SchedulEx must let the user to change the distance between exams .
20. SchedulEx must let the user to change the distance between calls of the same exam.
21. SchedulEx must let the user to change the distance between calls for a specific exam to be selected.
22. SchedulEx must let the user to start the optimization even if not all professor have unavailability
23. SchedulEx must let the user to start the optimization only if start and end date of the exam session and the settings are present.
24. SchedulEx must find a solution of the scheduling problem otherwise it must return a message of impossibility to schedule the problem.

## 4.2 - External Interface Requirements

The following section will give a more detailed description, in terms of hardware,

software and communication interfaces.

## 4.2.2 - Hardware Interfaces

The hardware needed to use the software of this project is a pc with monitor (to visualise graphical user interface), keyboard and mouse to insert input data and a printer (not strictly necessary) to print the calendar. Having a workstation where lunch the optimization algorithm is necessary in order to have an acceptable execution time.a

## 4.2.3 - Software Interfaces

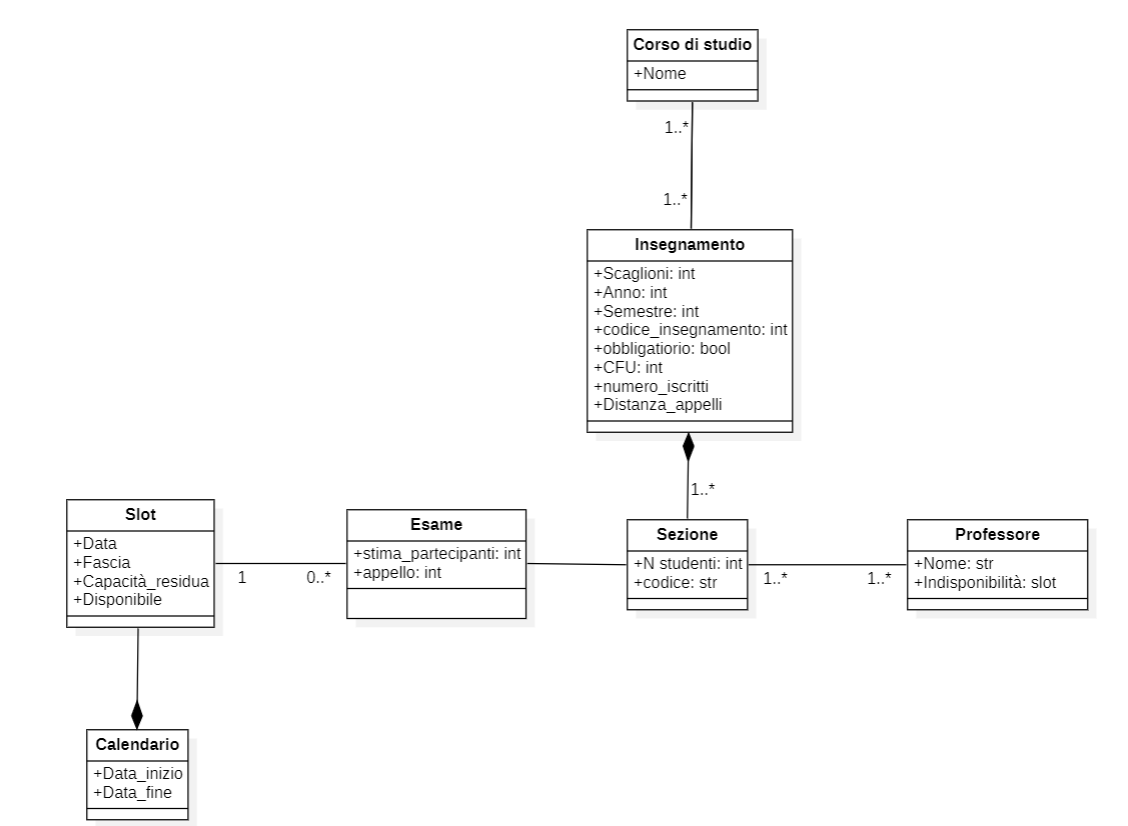
The front-end (GUI) must communicate with the back-end, this last one must communicate with the database and should launch the optimization algorithm. The back end communicates with database both for its proper request but also for front end ones.

## 4.3 Performance Requirements

The system does not have specific requirements on performances such as response time. The optimization algorithm can take a long time to be solved but there aren't any problems.

# 5 - DOMAIN MODEL

## 5.1 - Class diagram



# 6 – OPTIMIZATION ALGORITHM LOGICS

In this section we focus on logic notation of the requirements in terms of rules that must be satisfied for our optimization algorithm. To this aim we introduce the following notation:

For each study programme (CdS) we have a set of exams . Each elements of these sets are characterised by: Semester (), professor (), average number of attending students (). In our problem we will consider only one CdS for simplicity.

It possible to rewrite the rules requirements as following:

* Exams of the same semester and same year must have two night of distance each other;

Removing the last constraint index we can identify a possible soft constrain in order to limit the number of exam that occurs on the same day overall the CdS.

* The exams of the same course must have at least 14 days of distance.

* For each professor who reports his or her unavailability, we need to avoid assigning exam that involve him or her, in those dates.