

**Software Requirements Specification**

**Sistem informatic distribuit**

**pentru administrarea proiectelor studenților**

**Prepared by:**

**Birdau Mihaela**

**Tuiu Andrei**

Table of Contents

[1.Introduction 4](#_Toc510081163)

[1.1 Purpose 4](#_Toc510081164)

[1.2 Document Conventions 4](#_Toc510081165)

[1.3 Intended Audience and Reading Suggestions 4](#_Toc510081166)

[1.4 Product Scope 4](#_Toc510081167)

[1.5 References 4](#_Toc510081168)

[2. Overall description 5](#_Toc510081169)

[2.1 Product perspective 5](#_Toc510081170)

[2.2 Product functions 5](#_Toc510081171)

[2.3 User Classes and Characteristics 6](#_Toc510081172)

[2.4 Operating Environment 6](#_Toc510081173)

[2.5 Design and Implementation Constraints 6](#_Toc510081174)

[2.6 User Documentation 6](#_Toc510081175)

[2.7 Assumptions and dependencies 6](#_Toc510081176)

[3 External interface Requirements 6](#_Toc510081177)

[3.1 User interfaces 6](#_Toc510081178)

[3.2 Hardware Interfaces 12](#_Toc510081179)

[3.3 Software interfaces 12](#_Toc510081180)

[3.4 Communications Interfaces 12](#_Toc510081181)

[4. System Features 12](#_Toc510081182)

[4.1 System feature 1 12](#_Toc510081183)

[4.2 System feature 2 12](#_Toc510081184)

[4.3 System feature 3 12](#_Toc510081185)

[4.4 System feature 4 13](#_Toc510081186)

[4.5 System feature 5 13](#_Toc510081187)

[4.6 System feature 6 13](#_Toc510081188)

[4.7 System feature 7 13](#_Toc510081189)

[4.8 System feature 8 13](#_Toc510081190)

[4.9 System feature 9 14](#_Toc510081191)

[4.10 System feature 10 14](#_Toc510081192)

[4.11 System feature 11 14](#_Toc510081193)

[4.12 System feature 12 14](#_Toc510081194)

[4.13 System feature 13 14](#_Toc510081195)

[4.14 System feature 14 15](#_Toc510081196)

[4.15 System feature 15 15](#_Toc510081197)

[4.16 System feature 16 15](#_Toc510081198)

[4.17 System feature 17 15](#_Toc510081199)

[4.18 System feature 18 15](#_Toc510081200)

[4.19 System feature 19 16](#_Toc510081201)

[4.20 System feature 20 16](#_Toc510081202)

[4.21 System feature 21 16](#_Toc510081203)

[4.22 System feature 22 16](#_Toc510081204)

[4.23 System feature 23 17](#_Toc510081205)

[4.24 System feature 24 17](#_Toc510081206)

[4.25 System feature 25 17](#_Toc510081207)

[4.26 System feature 26 17](#_Toc510081208)

[4.27 System feature 27 17](#_Toc510081209)

[4.28 System feature 28 18](#_Toc510081210)

[4.29 System feature 29 18](#_Toc510081211)

[4.30 System feature 30 18](#_Toc510081212)

[4.31 System feature 31 18](#_Toc510081213)

[4.32 System feature 32 18](#_Toc510081214)

[4.33 System feature 33 19](#_Toc510081215)

[4.34 System feature 34 19](#_Toc510081216)

[4.35 System feature 35 19](#_Toc510081217)

[4.36 System feature 36 19](#_Toc510081218)

[4.37 System feature 37 20](#_Toc510081219)

[4.38 System feature 38 20](#_Toc510081220)

[5. Other Nonfunctional Requirements 20](#_Toc510081221)

[5.1 Performance requirements 20](#_Toc510081222)

[5.1.1 Prominent search feature 20](#_Toc510081223)

[5.1.2 Usage of the search feature 20](#_Toc510081224)

[5.2 Safety Requirements 20](#_Toc510081225)

[5.3 Security Requirements 20](#_Toc510081226)

[5.4 Software Quality Attributes 21](#_Toc510081227)

[5.5 Business Rules 22](#_Toc510081228)

[6. Other Requirements 22](#_Toc510081229)

[Appendix A: Glossary 23](#_Toc510081230)

[Appendix B: Analysis Models 24](#_Toc510081231)

[Appendix C: To Be Determined List 25](#_Toc510081232)

# 1.Introduction

This section gives a scope description and overview of everything included in this SRS document. Also, the purpose for this document is described and a list of abbreviations and definitions is provided.

## 1.1 Purpose

The purpose of this document is to give a detailed description of the requirements for an informatics and distributed system used for student projects management. Also, the purpose and complete declaration for the development of system will be illustrated. It will also explain system constraints, interface and target's audience of the application.

## 1.2 Document Conventions

Oracle Code Conventions for the Java Programming Language

Every requirement statement is to have its own priority.

## 1.3 Intended Audience and Reading Suggestions

This Software Requirements document is intended for:

− Developers who can review project’s capabilities and more easily understand where their efforts should be targeted to improve or add more features to it (design and code the application – it sets the guidelines for future

development).

− Project testers can use this document as a base for their testing strategy as some bugs are easier to find using a requirements document. This way testing becomes more methodically organized.

− End users of this application who wish to read about what this project can do.

## 1.4 Product Scope

The platform is a desktop application which is dedicated to teachers and students. This is useful for teachers because they are able to keep the tracking of the students assignments, to evaluate their work and add comments in order to grow up the student self development. Also, receive submissions of work from students in one area that is easy to manage everything.

The students can submit their assignment in a fast way and view their marks using only an internet connection. They can communicate with teachers, in order to clarify the misunderstandings regarding the project . So, the application enables the communication by e-mail between users.

## 1.5 References

[1] IEEE Software Engineering Standards Committee, “IEEE Std 830-1998, IEEE Recommended Practice for Software Requirements Specifications”, October 20, 1998

[2] Feldt R,”re\_lecture5b\_100914”, unpublished.

[3] Code Conventions for the Java Programming Language -http://www.oracle.com/technetwork/java/codeconvtoc-136057.html

[4] Pencil - https://pencil.evolus.vn/

[5]Visual paradigm - https://www.visual-paradigm.com/

# 2. Overall description

## 2.1 Product perspective

This section will give an overview of the whole system. The system will be explained in its context to show how the system interacts with other systems and introduce the basic functionality of it. At last, the constraints and assumptions for the system will be presented.

This system will consist of an desktop application with two interfaces: user and admin.

## 2.2 Product functions

The application will implement the following features:

- User log-in

- Retrive password - password will be sent to external email.

- Change password

- Check if the password is equals with confirmation password

- Check if the password have the specific pattern

- View profile

- View all courses

- Search a course by name, teacher's name, specialization

- Send enroll request for a course

- View enrolled courses

- Search an enrolled course by name, teacher's name, specialization

- View assignments for an enrolled course

- View details about an assignment

- Upload files for an assignment

- View mail inbox

- View details for a selected recieved mail.

- View sent mails.

- View details for a selected sent mail.

- Compose a mail

- View courses where he teacher

- Accept/Decline a student enroll request

- View all students enrolled for a course

- Remove students from a course

- Create a new assignment

- Modify an assignment

- Remove an assignment

- Download files that a student upload for an assignment

- Evaluate the students

- Create new users

- Check if all fields for a new user are fill correctly

- Modify the informations about a users

- Delete a user

- Generate a password for a new user and have a specific pattern

- Create new courses

- Modify a course

- Delete a course

- Create new specializations

- Modify a specialization

- Delete a specialization

- View all users

- Search a user by a field

- Internationalization

With the desktop application, the users will be able to search for courses. The result will be based on the criteria the user inputs. There are several search criteria. The result of the search will be viewed either in a list view. The list view will have one list item for each course matching the search criteria. In both views the users will be able to either select a course and get information about it. The application will provide functionality to manage the system.

## 2.3 User Classes and Characteristics

There are three types of users that interact with the system: students, teachers and administrators. Each of these three types of users has different use of the system, so each of them has their own requirements. The students can search a course by its name or teacher's name. This means that the user have is able to view the assignments for that course. For each assignment is specified the deadline, the assigned mark and last upload. The teachers can view all assignments for a course. Also, they can add new assignments, modify the old ones or remove them. The administrators are managing the overall system.

## 2.4 Operating Environment

Minimal operating system Windows 8.1.

Minimal java version 1.8

Minimal CPU 32 bits

## 2.5 Design and Implementation Constraints

The Internet connection and the capacity of the database are constraints for application.

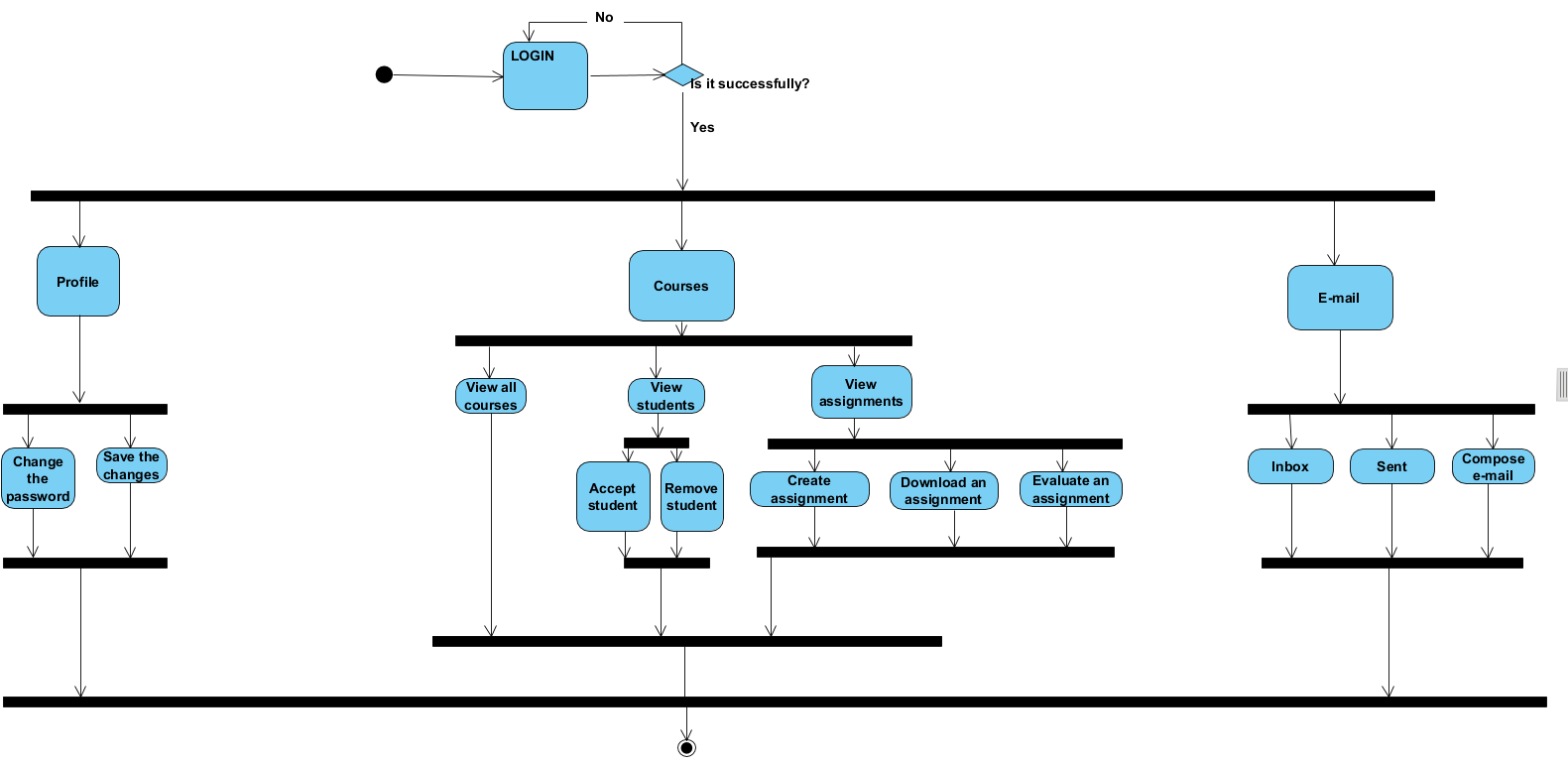
## 2.6 User Documentation

Not applicable.

## 2.7 Assumptions and dependencies

One assumption about the product is that it will always be used on computers that have enough performance.

## 2.8 Admin activity diagram



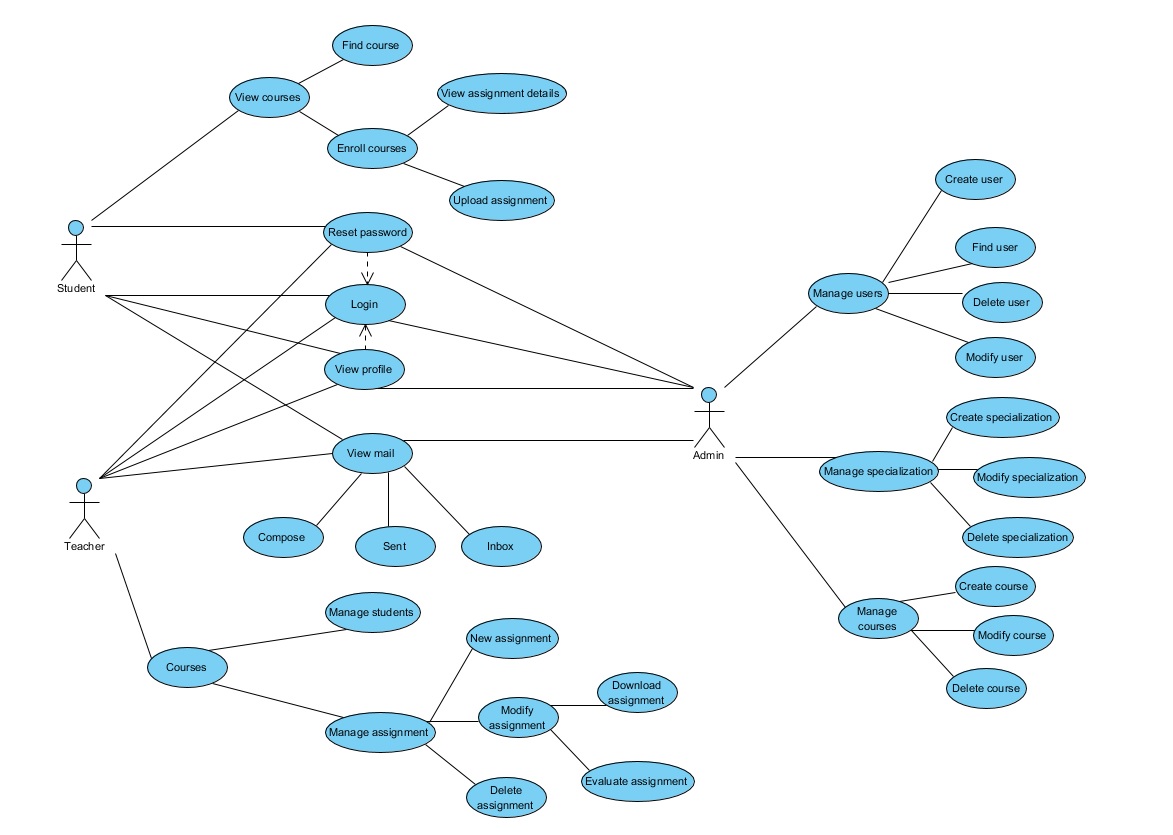
## 2.9 Teacher activity diagramC:\Users\itsix\AppData\Local\Microsoft\Windows\INetCache\Content.Word\46507733_1952379301722425_8633225139334414336_n.png

## 2.10 Student activity diagram46503384_2258882397720735_5026283132815409152_n

## 2.11 Class diagram

## 

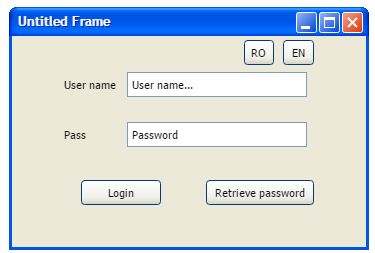
## 2.12 Entity-relation diagram



# 3 External interface Requirements

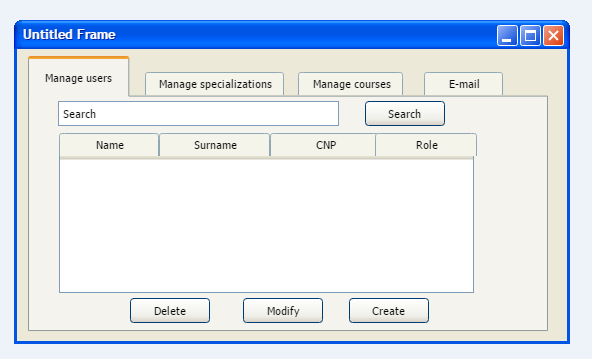
## 3.1 User interfaces

When any user opens the application, he will see the log-in page.

****

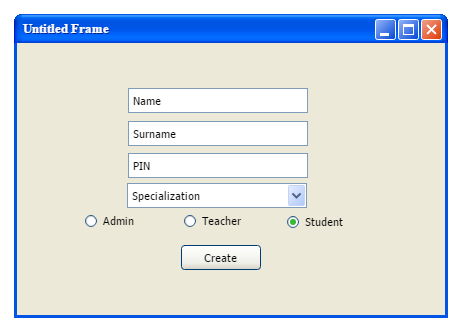
*Figure1. Login*

If the user is administrator, after log-in, the page will look like this.



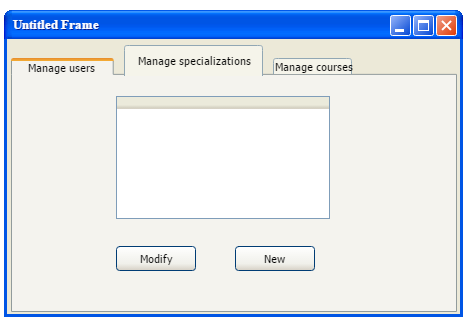
*Figure2. Admin - main frame*

From what it is observed(figure 2 ), he has the possibility to manage the accounts for each type of user: teacher, student or admin. A list with all users will be displayed and he can search a specific user by name, surname, personal identification number, role, withal he can select one of them from the list in order to delete, modify it. If you want to create a new account, click on Create button and a new window will be opened in order to complete the fields.



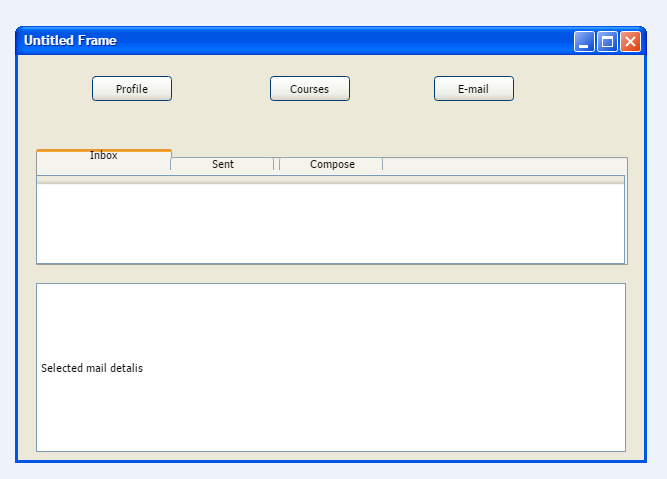
*Figure3. Admin -create a new account for student*

The specializations, courses are managed also by the administrator. He can delete, modify them or create new ones, these operations make pop-up to new windows.



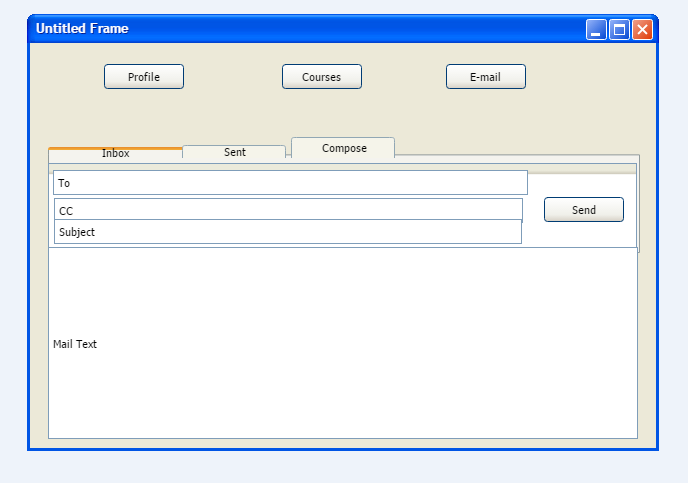
*Figure4. Admin - manage specializations tab( same frame for courses)*

Every user send/read e-mails and answer to them. Note: the administrator can only read his e-mails.

**

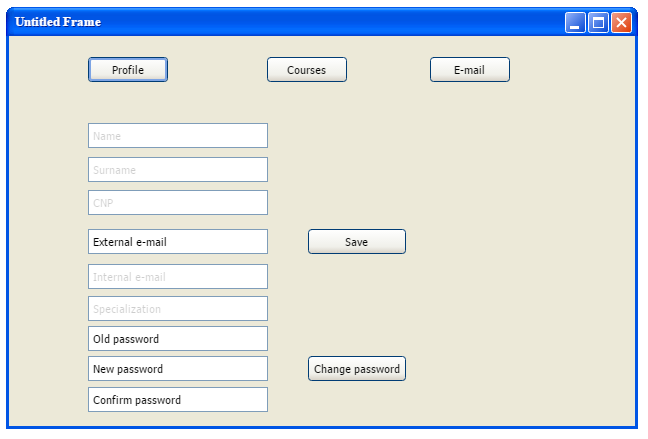
*Figure6 . Inbox e-mail*

In the Inbox tab, the user will view all received e-mails and will have the chance to manage them. The remove operation will be allowed.

**

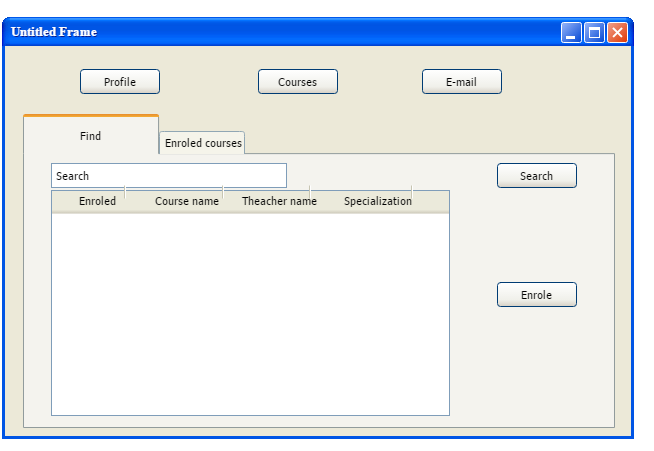
*Figure7. Student- Compose an e-mail*

If the user is student than will be able to view his profile.



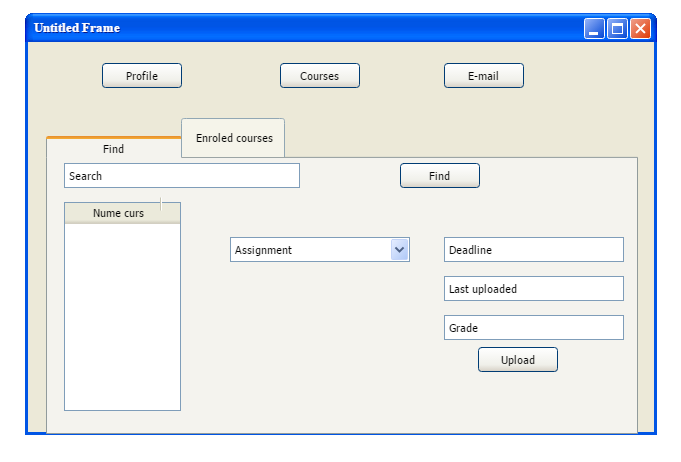
*Figure5. Profile information frame(for student as example)*

When Profile tab is pressed, the user has the possibility to change the password and add an external e-mail. In order to keep the changes, Save button should be pressed.

The student user can view a list with all courses, which are assigned for student's specialization, and search a specific one. He has the possibility to choose some of them and enrole on them, but this operation is not made in the same time, it should be performed selecting one by one.

*Figure8. View all courses*

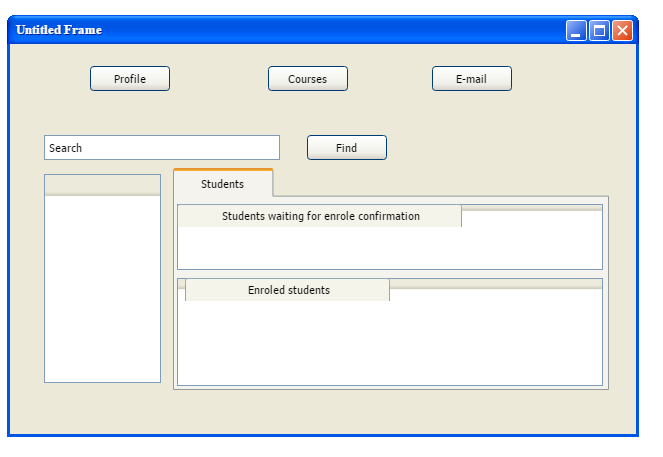
The user has also to view all courses where is enrolled with a click on Enrolled Courses, and they will be displayed on the left side of page. On the right side, the user can choose an assignment by selecting a course from the list. Each assignment has a specific deadline, the date of the last upload, a mark only if the project was uploaded and teacher evaluated it.



*Figure 9. Student- Enrolled courses tab*

Using the Upload button, the user is able to load the files from computer with a maximum capacity of 25mb.

The teacher user has the same fields as student user, but how is built one of them is completely different. The Courses field is the changed one. Press on it and the user can view on the left side all his courses and on the right all students which are enrolled on course. Also, he can see the students which send the request for enrolling but they don't receive an answer.



*Figure 9. Teacher - Courses tab*

## 3.2 Hardware Interfaces

Not applicable.

## 3.3 Software interfaces

The communication between the database and the application consists of operation concerning both reading and modifying the data.

ResultSet executeQuery(String sql)throws SQLException –read from databse

int executeUpdate(String sql) throws SQLException – write in database

## 3.4 Communications Interfaces

Not applicable.

# 4. System Features

This section includes the requirements that specify all the fundamental actions of the software system.

## 4.1 System feature 1

**ID: SF1**

TITLE : User log-in

DESC: Given that an user has registered, then the user should be able to log in to the application, using CNP and password.

RAT: In order for a user to log-in in the desktop application

DEP: Open the application

REQ1 : Create the user interface for login.

REQ2 : Check if the user is in database.

## 4.2 System feature 2

**ID: SF2**

TITLE : Retrieve password

DESC : Given that an user has registered, then the user should be able to retrieve his password by e-mail.

RAT: In order for a user to retrieve his/her password.

DEP: SF1

REQ1 : Create the user interface for retrieve password.

REQ2 : Check if the user is in database.

REQ3 : Send an e-mail with the password.

## 4.3 System feature 3

**ID: SF3**

TITLE : View profile

DESC : Given a logged in user, the user should be able to view his profile information.

RAT: In order for a user to view his personal information.

DEP: SF1

REQ1 : Create the user interface for view profile.

REQ2 : Populate the interface with the data from database.

## 4.4 System feature 4

**ID: SF4**

TITLE : Change password

DESC : Given a logged in user , the user should be able to change his password from profile frame.

RAT: In order for a user to change his password.

DEP: SF1, SF3

REQ1 : Create the user interface for changing password.

REQ2 : Update user password in database.

## 4.5 System feature 5

**ID: SF5**

TITLE : Check password

DESC: Check if the new password is equals with confirmed password and the old password is correct.

RAT: In order for application to keep the integrity of data, safety goal.

DEP: SF1, SF4

REQ1 : Compare new password with confirmation password.

REQ2 : Compare old password with the password in the database.

## 4.6 System feature 6

**ID: SF6**

TITLE : Password pattern

DESC : Check if the password has the specific pattern.

RAT: In order for application to keep the integrity of data, safety goal.

DEP: SF1, SF4

REQ1 : Check if the password has the specific pattern.

## 4.7 System feature 7

**ID: SF7**

TITLE : Mail inbox

DESC : Given a logged in user, he should be able to see the received e-mails and manage them.

RAT: In order for a user to see all e-mails and delete some of them if he wants.

DEP: SF1

REQ1 : Create the user interface for inbox.

REQ2 : Populate the interface with data from database.

## 4.8 System feature 8

**ID: SF8**

TITLE : Mails sent

DESC : Given a logged in user, the user should be able to see the sent e-mails and manage them.

RAT: In order for a user to see all sent e-mails by him and delete some of them if he wants.

DEP: SF1

REQ1 : Create the user interface for send e-mails.

REQ2 : Populate the interface with data from database.

## 4.9 System feature 9

**ID: SF9**

TITLE : Compose mail

DESC : Given a logged in user, the user should be able to send e-mails.

RAT: In order for a user to respond at received e-mails or ask other user about a specific topic.

DEP: SF1

REQ1 : Create the user interface for send e-mail.

REQ2 : Update the database with the new e-mail.

## 4.10 System feature 10

**ID: SF10**

TITLE : Input validation

DESC: Check if all input fields for a user are filled correctly.

RAT: In order for application to keep the integrity of data, safety goal.

DEP: SF1

REQ1 : Check if all input fields are filled correctly.

**The admin user**

## 4.11 System feature 11

**ID: SF21**

TITLE : View all users

DESC : Given a logged in admin, the admin should be able to view all users and their information.

RAT: In order for admin to view with all users, to manage them very easy.

DEP: SF1

REQ1 : Create the user interface for viewing all users.

REQ2 : Populate the view with users from database.

## 4.12 System feature 12

**ID: SF22**

TITLE : Create new account

DESC : Given a logged in admin, the admin should be able to create new accounts for different type of users.

RAT: In order for admin to populate the application with new users.

DEP: SF1, SF21

REQ1 : Create the user interface for creating a new account.

REQ2 : Update the database with the new account.

## 4.13 System feature 13

**ID: SF23**

TITLE : Delete an account

DESC : Given a logged in admin, the admin should be able to delete accounts.

RAT: In order for admin to clean up the database if the account is no longer usable or is invalid. Also, this operation can be performed if an user ask to remove his account.

DEP: SF1, SF21

REQ1 : Create the user interface for deleting an account.

REQ2 : Update the database with the new e-mail.

## 4.14 System feature 14

**ID: SF24**

TITLE : Modify an account

DESC : Given a logged in admin, the admin should be able to modify the information for an account.

RAT: In order for user to be able to modify the information by sending an e-mail to admin.

DEP: SF1, SF21, SF9

REQ1 : Create the user interface for modifying an account.

REQ2 : Update the database with the modifications.

## 4.15 System feature 15

**ID: SF25**

TITLE : Generate password

DESC: Generate a password for a new user with the specific pattern.

RAT: In order for user to have the data protected.

DEP: SF1, SF22

REQ1 : Generate a password for a new user with the specific pattern.

## 4.16 System feature 16

**ID: SF26**

TITLE : View all courses

DESC : Given a logged in admin, he should be able to view a list with all courses.

RAT: In order for admin to manage all courses.

DEP: SF1

REQ1 : Create the user interface for viewing all courses.

REQ2 : Populate the view with courses from database.

## 4.17 System feature 17

**ID: SF27**

TITLE : Create new courses

DESC : Given a logged in admin, the admin should be able to create new course. A course can have many teachers.

RAT: In order for admin to add new courses if it is necessary.

DEP: SF1, SF26

REQ1 : Create the user interface for creating a new course.

REQ2 : Update the database with the new course.

## 4.18 System feature 18

**ID: SF28**

TITLE : Delete a course

DESC : Given a logged in admin, the admin should be able to delete course.

RAT: In order for admin to delete some courses which are no longer useful and clean up the database.

DEP: SF1, SF26

REQ1 : Create the user interface for deleting a course.

REQ2 : Delete the course from database.

## 4.19 System feature 19

**ID: SF29**

TITLE : Modify a course

DESC : Given a logged in admin, the admin should be able to modify the information for a course.

RAT: In order for admin to modify courses in case of a new teacher or the specialization is no longer available.

DEP: SF1, SF26

REQ1 : Create the user interface for modifying a course.

REQ2 : Update the database with the modifications.

## 4.20 System feature 20

**ID: SF210**

TITLE : View all specializations

DESC : Given a logged in admin, he should be able to view a list with all specializations.

RAT: In order for admin to manage all specializations.

DEP: SF1

REQ1 : Create the user interface for viewing all specializations.

REQ2 : Populate the view with data from database.

## 4.21 System feature 21

**ID: SF211**

TITLE : Create new specialization

DESC : Given a logged in admin, the admin should be able to create new specialization.

RAT: In order for admin to add new specializations if it is necessary.

DEP: SF1, SF210

REQ1 : Create the user interface for creating a specialization.

REQ2 : Update the database with the new specialization.

## 4.22 System feature 22

**ID: SF212**

TITLE : Delete a specialization

DESC : Given a logged in admin, the admin should be able to delete specialization.

RAT: In order for admin to delete some specializations which are no longer available and clean up the database.

DEP: SF1, SF210

REQ1 : Create the user interface for deleting a specialization.

REQ2 : Delete the specialization from database.

## 4.23 System feature 23

**ID: SF213**

TITLE : Modify the specialization

DESC : Given a logged in admin, the admin should be able to modify the information for a specialization.

RAT: In order for admin to modify the name of specialization.

DEP: SF1, SF210

REQ1 : Create the user interface for modifying a specialization.

REQ2 : Update the database with the modifications.

## 4.24 System feature 24

**ID: SF214**

TITLE : Search an user

DESC : Given a logged in admin, the admin should be able to search an user by using any field

RAT: In order for admin to find an user very quickly, not scrolling on the list.

DEP: SF1,SF21

REQ1 : Create the user interface for searching an user.

REQ2 : Get data from database.

**The student user**

## 4.25 System feature 25

**ID: SF31**

TITLE : View all courses

DESC : Given a logged in student, he should be able to view all courses even if they are not from his specialization.

RAT: In order for student to see all possible courses.

DEP: SF1

REQ1 : Create the user interface for viewing all courses.

REQ2 : Populate the view with the data from database.

## 4.26 System feature 26

**ID: SF32**

TITLE : Search a course

DESC : Given a logged in student, he should be able to search a course by its name, teacher’s name, specialization.

RAT: In order for student to find quickly a specific course.

DEP: SF1, SF31

REQ1 : Create the user interface for searching a course.

REQ2 : Get data from database.

## 4.27 System feature 27

**ID: SF33**

TITLE : Enrole on course

DESC : Given a logged in student, he should be able to send a request to be enrolled to a course. When his request is accepted or declined he will receive an automatic e-mail with the answer from the teacher.

RAT: In order for student to access the all available information for that course.

DEP: SF1, SF31

REQ1 : Create the user interface for enrolling on a course.

REQ2 : Update the database with enrolling request.

## 4.28 System feature 28

**ID: SF34**

TITLE : View enrolled courses

DESC : Given a logged in student, he should be able to view all the courses where he is enrolled at.

RAT: In order for student to be able to see only the enrolled courses.

DEP: SF1, SF33

REQ1 : Create the user interface for viewing enrolled courses.

REQ2 : Populate the view with data from database.

## 4.29 System feature 29

**ID: SF35**

TITLE : Search - an enrolled course

DESC : Given a logged in student, he should be able to search an enrolled course by its name, teacher’s name, specialization, request confirmation date.

RAT: In order for student to find quickly a course.

DEP: SF1, SF33, SF34

REQ1 : Create the user interface for searching an enrolled course.

REQ2 : Get data from database.

## 4.30 System feature 30

**ID: SF36**

TITLE : View assignments for an enrolled course

DESC : Given a logged in student, he should be able to see all the assignments for an enrolled course.

RAT: In order for student to view all assignments for an enrolled course.

DEP: SF1, SF33, SF34

REQ1 : Create the user interface for viewing assignments for an enrolled course.

REQ2 : Populate the view with data from database.

## 4.31 System feature 31

**ID: SF37**

TITLE: View details about an assignment

DESC : Given a logged in student, he should be able to see details about an assignment, like assignment name, deadline, last upload.

RAT: In order for student to manage all things for delivering the assignment according to its requirements.

DEP: SF1, SF36

REQ1 : Create the user interface for viewing details about an assignment.

REQ2 : Populate the view with data from database.

## 4.32 System feature 32

**ID: SF38**

TITLE : Upload files

DESC : Given a logged in student, he should be able to upload files for an assignment.

RAT: In order for student to upload his work by using internet connection.

DEP: SF1, SF36

REQ1 : Create the user interface for uploading a file.

REQ2 : Upload the file in the database.

**The teacher user**

## 4.33 System feature 33

**ID: SF41**

TITLE : View all courses

DESC : Given a logged in teacher, he should be able to view all courses where he teaches.

RAT: In order for teacher to see all his courses.

DEP: SF1

REQ1 : Create the user interface for viewing all courses.

REQ2 : Populate the view with data from database.

## 4.34 System feature 34

**ID: SF42**

TITLE : View all assignments for a course

DESC : Given a logged in teacher, he should be able to view all assignments which are assigned to a course.

RAT: In order for teacher to see all assignments for a course.

DEP: SF1, SF41

REQ1 : Create the user interface for viewing all assignments for course.

REQ2 : Populate the view with data from database.

## 4.35 System feature 35

**ID: SF43**

TITLE : Create a new assignment

DESC : Given a logged in teacher, he should be able to create a new assignment

RAT: In order for teacher to create a new assignment which will be added to database.

DEP: SF1, SF42

REQ1 : Create the user interface for creating a new assignment.

REQ2 : Insert the assignment in the database.

## 4.36 System feature 36

**ID: SF44**

TITLE : Delete an assignment

DESC : Given a logged in teacher, he should be able to delete an assignment

RAT: In order for teacher to delete an assignment which is no longer available.

DEP: SF1, SF42

REQ1 : Create the user interface for deleting an assignment.

REQ2 : Populate the view with data from database.

## 4.37 System feature 37

**ID: SF45**

TITLE : Evaluate an assignment

DESC : Given a logged in teacher, he should be able to evaluate an assignment

RAT: In order for teacher to give a grade, he should evaluate it.

DEP: SF1, SF42

REQ1 : Create the user interface for evaluating an assignment.

REQ2 : Update the database.

## 4.38 System feature 38

**ID: SF46**

TITLE : Modify assignment

DESC : Given a logged in teacher, he should be able to modify an assignment

RAT: In order for teacher to modify an assignment.

DEP: SF1, SF42

REQ1 : Create the user interface for modifying an assignment.

REQ2 : Update the database with the modifications.

# 5. Other Nonfunctional Requirements

## 5.1 Performance requirements

### 5.1.1 Prominent search feature

**ID: QR1**

TITLE: Prominent search feature

DESC: The search feature should be prominent and easy to find for the user.

RAT: In order to for a user to find the search feature easily.

DEP: none

### 5.1.2 Usage of the search feature

**ID: QR2**

TITLE: Usage of the search feature

DESC: The different search options should be evident, simple and easy to understand.

RAT: In order to for a user to perform a search easily.

DEP: none

## 5.2 Safety Requirements

Not applicable.

## 5.3 Security Requirements

**ID: QR6**

TAG: Communication security

GIST: Security of the communication between the system and server.

SCALE: The messages should be encrypted for log-in communications, so others cannot get user-name and password from those messages.

METER: Attempts to get user-name and password through obtained messages on 1000 log-in session during testing.

MUST: 100% of the Communication Messages in the communication of a log-in session should be encrypted.

Communication Messages: Defined: Every exchanged of information between client and server.

**ID: QR7**

TAG: User login account security

GIST: Security of accounts. 25

SCALE: If a registered user tries to log in to the application with a non-existing account then the user should not be logged in. The user should be notified about log-in failure.

METER: 1000 attempts to log-in with a non-existing user account during testing.

MUST: 100% of the time.

**ID: QR8**

TAG: Admin create account security

GIST: The security of creating account for admins of the system.

SCALE: If an admin wants to create an account and the desired e-mail is occupied, the admin should be asked to choose a different e-mail.

METER: Measurements obtained on 1000 hours of usage during testing.

MUST: 100% of the time

**ID: QR9**

TAG: User create account security

GIST: The security of creating account for users of the system.

SCALE: If an user wants to create an account and the desired password doesn’t meet the pattern, the unregistered user should be asked to choose a different password.

METER: Measurements obtained on 1000 hours of usage during testing.

MUST: 100% of the time

**ID: QR11**

TAG: Visible password

SCALE: A user should be able to see typed password.

METER: Measurements obtained on 1000 hours of usage during testing.

MUST: 100% of the time

## 5.4 Software Quality Attributes

**ID: QR10**

TAG: Input Validation

SCALE Check if all input fields are filled correctly.

METER: Measurements obtained on 1000 hours of usage during testing.

MUST: 100% of the time

**ID: QR12**

TITLE: Application extendibility

DESC: The application should be easy to extend. The code should be written in a way that it favors implementation of new functions.

RAT: In order for future functions to be implemented easily to the application.

DEP: none

**ID: QR13**

TITLE: Application testability

DESC: Test environments should be built for the application to allow testing of the applications different functions.

RAT: In order to test the application.

DEP: none

## 5.5 Business Rules

Not applicable.

# 6. Other Requirements

**ID: QR3**

TAG: Hard drive space

GIST: Hard drive space.

SCALE: The application’s need of hard drive space.

METER: MB.

MUST: No more than 40 MB.

PLAN: No more than 30 MB.

WISH: No more than 20 MB.

MB: DEFINED: Megabyte

**ID: QR4**

TAG: Application memory usage

GIST: The amount of Operate System memory occupied by the application.

SCALE: MB.

METER: Observations done from the performance log during testing

MUST: No more than 40 MB.

PLAN: No more than 30 MB

WISH: No more than 20 MB

MB: DEFINED: Megabyte.

**ID: QR5**

TITLE: Internet Connection

DESC: The application should be connected to the Internet.

RAT: In order for the application to communicate with the database.

DEP: none

**ID: QR14**

TITLE: Internationalization

DESC: The user should have the possibility to choose the preferred language. No log-in is required.

RAT: In order for user to be easy to adapt with the application.

DEP: open the desktop application

# Appendix A: Glossary

|  |  |
| --- | --- |
| Term | Definition |
| DEP | Dependency |
| FR | Functional Requirement |
| PIN | Personal Identification Number |
| DESC | Description |
| User | Someone who interacts with the application |
| SCALE | The scale of measure used by the requirement contained in a PLanguage statement [2] |
| TAG | A unique, persistent identifier contained in a PLanguage statement [2] |
| METER | The process or device used to establish location on a SCALE contained in a PLanguage statement [2] |
| MUST | The minimum level required to avoid failure contained in a PLanguage statement [2] |
| GIST | A short, simple description of the concept contained in a PLanguage statement [2] |
| PLAN | The level at which good success can be claimed contained in a PLanguage statement [2] |
| WISH | A desirable level of achievement that may not be attainable through available means contained in a PLanguage statement [2] |
| Admin/Administrator | System administrator who is given specific permission for managing and controlling the system |
| RAT | Rational |

# 

# Appendix B: Frameworks

**Spring Boot**

Enables you to develop enterprise-ready applications that you can “just run”

**JPA**

Is a collection of classes and methods to persistently store the vast amounts of data into a database

**Spring Injection**

Dependency injection is a pattern through which to implement IoC, where the control being inverted is the setting of object’s dependencies.

**Maven**

Making the build process easy. Providing a uniform build system. Providing quality project information. Providing guidelines for best practices development. Allowing transparent migration to new features

**Flyway**

For migrating the database on application startup