Software requirement specification document for MSA Graduation Management System

Abdelrahman Ezz

Mostafa Mohamed Saeed

Mena Hany

Abdul-Aziz Ashraf

Rana Reda

Supervised by: Dr. Ayman Ezzat

November 1 , 2020

# **1 Introduction**

**1.1 Purpose of this document**

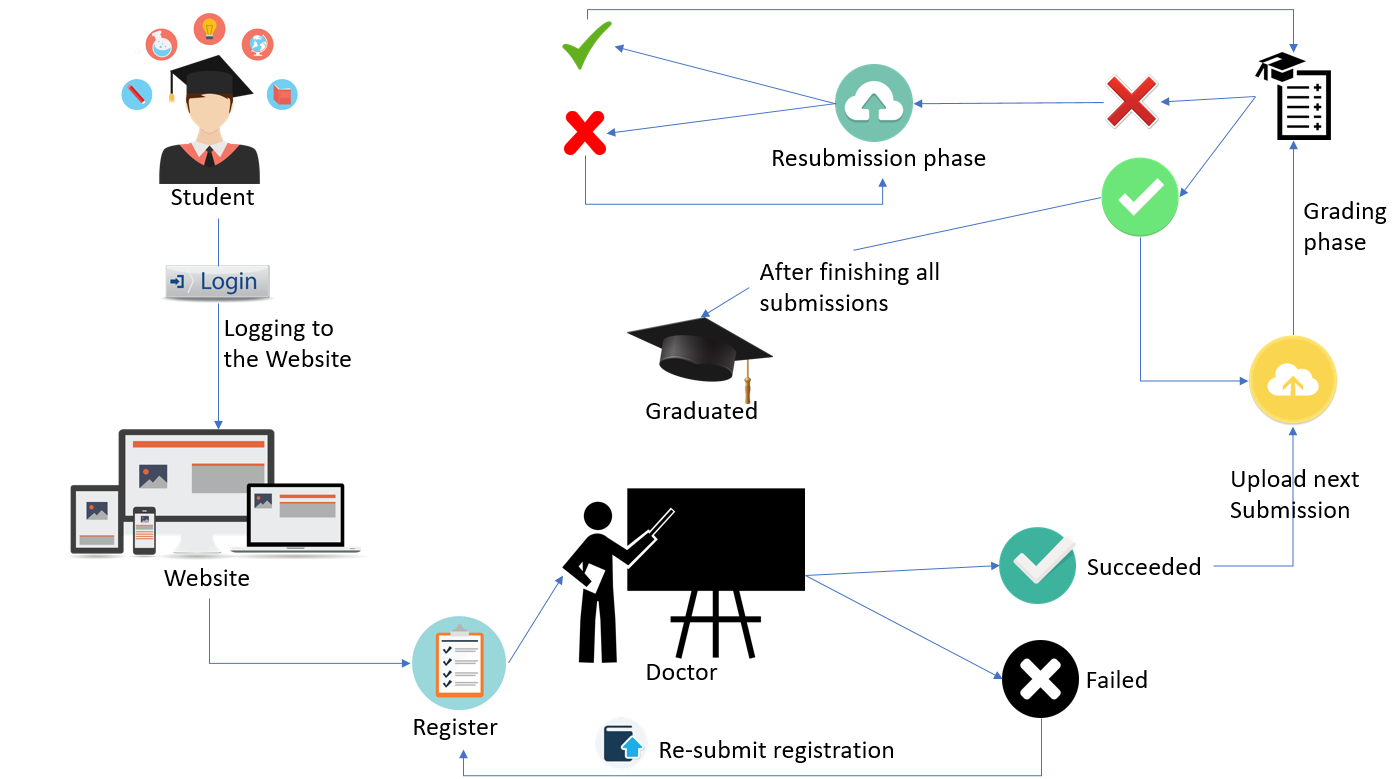
The purpose of this document is to be of an agreement with the stakeholders of the system of how our web application should work, the constraints on the system, deliverables and their deadlines.

## **1.2 Scope of this document**

The team members involved in the defination, analysis, design and implementation of this system are Abdelrahman Ezzeldin, Mostafa Mohammed Saeed, Mena Hany , AbdelAziz Ashraf and Rana Reda; the clients of the System is Dr.Ayman Ezzat ;the users of the system will be system admins and the doctors of faculty of computer science in MSA university, Computer Science Students in MSA university that are in 4th level can login into this web application to register their graduation project and then if it is accepted they will be able to go for the next submition phases that will be reviewd and graded by Computer Science doctors and if we got the average of all these grades and it passed a specific limit then the student has succeeded and will graduate.

## **1.3 Overview**

The system shall alow the student to organize his steps for the graduation project, Students will start with regestiring his graduation project then the supervised doctor on this project wla accept or refuse this project, if the registration is refused then the student has to re-submit his graduation project registration, on the other hand if it is accepted then the student will go for the next submiting phases to complete them all so he can graduate finally.



## **1.4 Business Context**

MSA University is a high education unviersity aiming to provide students with the best education in egypt and thus one of the most important phases for any student in any university is the graduation phase thus a management system is required to manage the process of the grauation processes of any student

# **2 General Description**

## **2.1 Product Functions**

**1)student registrate for graduation:-**

-Registring for entering the graduation.

**2)Thesis submitting:-**

-Submitting the whole phases of the gradation product.

**3)Thesis grading:-**

-Doctor grading the student registration and thesis submition.

## **2.2 Similar System Information**

NO SIMILAR SYSTEM.(For view)

## **2.3 User Characteristics**

###-It's expected that these users will have the basic known background about how to use a web based system so they can deal with our system, As they will be using a web application system

-The user is expected to how to use a mobile or a computer.

## **2.4 User Problem Statement**

-Graduation project management became a complex problem due to the change of failure and success of the student due to the reptitive tasks with different names and phases constraints thus a management system is required for the management of the graduation process from the registetraton for the graduation project to the thesises uploadings to the final graduation submission

## **2.5 User Objectives**

-The user wants a report that has all students and what courses they are enrolled into.

-The user wants to see the academy monthly and annually profit , losses in a graphical way.

-The user wants the parent to be able to see the degrees and instructor reviews to his child

-The parent got the ability to contact the instructor to know how his child is doing there in the academy.

-He student could see the courses he/she is taking, download the material of the course,submit the assignment, viewing assignment,quiz,test degree, also see the instructor comments, and the student could contact the instructor.

## **2.6 General Constraints**

We are putting some constraints for our web based system as we are trying to make our system responsive with maximum 6 seconds , this in condition we need minimum internet speed 4 MB/sec, acceptable hardware with good memory storage, SSD card to get our system run fast without any delay and good hardware components.

# **3 Functional Requirements**

|  |  |
| --- | --- |
| Code | E\_1.1 |
| Name | logIn |
| Type | Functional Requirement |
| Criticality | medium |
| Priority | 7/10 |
| Inputs | User ID(int ID); Password(string password); |
| Source | User’s inputs from the login form |
| Output | Start a session with the user’s data and direct him to the Home Page |
| Description | This will sign the user into the system |
| Destination | - |
| Action | The system will check that all the user’s inputs are filled and that the ID and password are correct and The password will be displayed as in the input of the login from ‘\*’  The system will check if the ID and password are correct from the database table “Users” by comparing the hashed password with the input password by checking if the hashed password is true when compared to the input password using the function (E\_1.2 checkPasswordTrueOrFalse)  The hashed password will be retrieved by the function (E\_1.3 query) |
| Requirements | All the input fields so the user’s data can be checked in the database |
| Expected Risks | The user inputs an incorrect ID and or incorrect password |
| Pre-conditions | - |
| Post-condition | Direct the user to the home page or the log in page again incase of an incorrect data input |

|  |  |
| --- | --- |
| Code | E\_1.3 |
| Name | query |
| Type | Functional Requirement |
| Criticality | medium |
| Priority | 7/10 |
| Inputs | databaseConnection connection; SQL Select Statement (string s); |
| Source | (E\_1.1) |
| Output | Returns an array containing the select statement results or return false if the statement is incorrect or doesn’t exist |
| Description | This function returns the results of the query sent |
| Destination | - |
| Action | The system will use the database connection that connects to the database and will perform the sql statement string that is sent in the parameters |
| Requirements | The database connection, the sql statement a string |
| Expected Risks | SQL statement is incorrect and or the database connection is not established |
| Pre-conditions | The database connection is correct |
| Post-condition | Returns and array of the select statement selection or returns false if the statement |

|  |  |
| --- | --- |
| Code | E\_1.4 |
| Name | logout |
| Type | Functional Requirement |
| Criticality | medium |
| Priority | 7/10 |
| Inputs | signOutButton |
| Source | User input |
| Output | The user will be signed out of the system and his session will come to an end and be closed |
| Description | The user can click a button to log out of the system |
| Destination | - |
| Action | When the clicks the sign out button he will be directed to the login page |
| Requirements | - |
| Expected Risks | - |
| Pre-conditions | The user must be already logged in |
| Post-condition | The user will be logged out and directed to the log in page |

|  |  |
| --- | --- |
| Code | E\_1.6 |
| Name | checkAllIsNotNumbers |
| Type | Functional Requirement |
| Criticality | Medium |
| Priority | 5/10 |
| Inputs | A string |
| Source | - |
| Output | Returns True or False |
| Description | Returns a Boolean of True if the string doesn’t contain an integer or it’s |
| Destination | - |
| Action | This function receives a string (x for example) and removes all the single quotes (‘) through the function (EZ\_1.7) and then it creates an array that contains all the alphabet in small case and upper case and the space character as well and then it splits the passed string (x) function into array of characters and then if any cell in the array is not in the alphabet or is a space this function will return **False** but if all is in the alphabet or is a space it will return **True**. |
| Requirements | - |
| Expected Risks | The passed string contains any special character or a number |
| Pre-conditions | The string is all only characters from the alphabet or a space |
| Post-condition | Return **True** |

|  |  |
| --- | --- |
| Code | E\_1.7 |
| Name | removeSingleQuotes |
| Type | Functional Requirement |
| Criticality | Medium |
| Priority | 5/10 |
| Inputs | A string |
| Source | - |
| Output | Returns the passed String but without any single quotes. |
| Description | It will remove the entire single quotes in the string |
| Destination | - |
| Action | This function is given a string (x) and it will remove all the single quotes in the string and replace them with \’ in order to avoid any problems in future database queries |
| Requirements | - |
| Expected Risks | - |
| Pre-conditions | - |
| Post-condition | Returns the string without any single quotes that cause a problem in the database |

|  |  |
| --- | --- |
| Code | E\_1.8 |
| Name | minMaxLengthChecks |
| Type | Functional Requirement |
| Criticality | High |
| Priority | 8/10 |
| Inputs | A string or an integer |
| Source | - |
| Output | Returns **True** or **False**. |
| Description | It will check the length of the given parameter from the database |
| Destination | - |
| Action | This function is given an ID and a string or integer and it will use (EZ\_1.7 removeSingleQuotes) to remove any single quotes that might harm the database in the given string while making a query. It will retrieve the minLength and maxLength for the given ID attribute from the database table constraints\_table and will do so by using the function (EZ\_1.3 query) to retrieve those values and if the given string or integer length is more than or equal the minimum length and smaller than or equal the maximum length returned from the database for that specific attribute it will return **True** but if the string is less than the minimum length for it in the database or exceeds the maximum length for it in the database it will return **False.**  if the query contains any mistakes or contains a non-existing ID it will return **False** |
| Requirements | An existing ID in the database |
| Expected Risks | -the ID passed doesn’t exist in the constraint\_table in the database or the variable (string or integer) passed violates the minimum or the maximum length constraints |
| Pre-conditions | The ID exists in the constraint\_table in the database and the variable (string or integer) passed follows the constraints rules |
| Post-condition | Return **True** |

|  |  |
| --- | --- |
| Code | E\_1.10 |
| Name | checkNameConstraints |
| Type | Functional Requirement |
| Criticality | Medium |
| Priority | 5/10 |
| Inputs | A string |
| Source | - |
| Output | Returns **True** or **False**. |
| Description | It will check the length constraints for the Name from the database is good or bad |
| Destination | - |
| Action | It will check the length constraints for the Name from the database is greater than or equal to the minimum length and is less than or equal to the maximum length in the database for the Name. if the conditions hold it will return **True** if any of the conditions doesn’t hold it will return **False** |
| Requirements | An existing ID passed is 4 and an integer |
| Expected Risks | the ID passed is not 4 |
| Pre-conditions | Length of the given string follows the database minimum and maximum constraint set for it in the database |
| Post-condition | Return **True** |

|  |  |
| --- | --- |
| Code | E\_1.11 |
| Name | checkEmailConstraints |
| Type | Functional Requirement |
| Criticality | Medium |
| Priority | 5/10 |
| Inputs | A string |
| Source | - |
| Output | Returns **True** or **False**. |
| Description | It will check the length constraints for the Email from the database is good or bad |
| Destination | - |
| Action | It will check the length constraints for the Email from the database is greater than or equal to the minimum length and is less than or equal to the maximum length in the database for the Email. It will also check that the Email string contains @ and any letter before it and any letter after it if the conditions hold it will return **True** if any of the conditions doesn’t hold it will return **False** |
| Requirements | An existing ID passed is 7 and an integer |
| Expected Risks | the ID passed is not 7 |
| Pre-conditions | Length of the given string follows the database minimum and maximum constraint set for it in the database |
| Post-condition | Return **True** |

|  |  |
| --- | --- |
| Code | E\_1.12 |
| Name | checkPhoneNumberConstraints |
| Type | Functional Requirement |
| Criticality | Medium |
| Priority | 5/10 |
| Inputs | A string |
| Source | - |
| Output | Returns **True** or **False**. |
| Description | It will check the length constraints for the Phone Number from the database is good or bad |
| Destination | - |
| Action | It will check the length constraints for the phone number from the database is greater than or equal to the minimum length and is less than or equal to the maximum length in the database for the phone number. And it will check that it’s all numbers only and by using (E\_1.15 checkAllIsNumbers) if the conditions hold it will return **True** if any of the conditions doesn’t hold it will return **False** |
| Requirements | An existing ID passed is 8 and an integer |
| Expected Risks | the ID passed is not 8 |
| Pre-conditions | Length of the given string follows the database minimum and maximum constraint set for it in the database |
| Post-condition | Return **True** |

|  |  |
| --- | --- |
| Code | E\_1.13 |
| Name | CheckPasswordConstraints |
| Type | Functional Requirement |
| Criticality | Medium |
| Priority | 5/10 |
| Inputs | A string |
| Source | - |
| Output | Returns **True** or **False**. |
| Description | It will check the length constraints for the Password from the database is good or bad |
| Destination | - |
| Action | It will check the length constraints for the password from the database is greater than or equal to the minimum length and is less than or equal to the maximum length in the database for the password. if the conditions hold it will return **True** if any of the conditions doesn’t hold it will return **False** |
| Requirements | An existing ID passed is 11 and an integer |
| Expected Risks | the ID passed is not 11 |
| Pre-conditions | Length of the given string follows the database minimum and maximum constraint set for it in the database |
| Post-condition | Return **True** |

|  |  |
| --- | --- |
| Code | E\_1.14 |
| Name | CheckPhaseNameConstraints |
| Type | Functional Requirement |
| Criticality | Medium |
| Priority | 5/10 |
| Inputs | A string |
| Source | - |
| Output | Returns **True** or **False**. |
| Description | It will check the length constraints for the Password from the database is good or bad |
| Destination | - |
| Action | It will check the length constraints for the phase name from the database is greater than or equal to the minimum length and is less than or equal to the maximum length in the database for the phase name. if the conditions hold it will return **True** if any of the conditions doesn’t hold it will return **False** |
| Requirements | An existing ID passed is 13 and an integer |
| Expected Risks | the ID passed is not 13 |
| Pre-conditions | Length of the given string follows the database minimum and maximum constraint set for it in the database |
| Post-condition | Return **True** |

|  |  |
| --- | --- |
| Code | E\_1.15 |
| Name | checkAllIsNumbers |
| Type | Functional Requirement |
| Criticality | Medium |
| Priority | 6/10 |
| Inputs | A string |
| Source | - |
| Output | Returns **True** or **False**. |
| Description | Check if the given string contains only numbers |
| Destination | - |
| Action | It will be given a string or an integer as a parameter. If the parameter sent is of datatype integer it will return **True** if it’s of type string it will be split into an array of characters and it will be compared with numbers from 0 to 9 or contains the negative (-) sign and if a cell in the passed string is not in 0 to 9 as negative or positive it will return **False.** |
| Requirements | An integer or string |
| Expected Risks | The string contains any value that is not a number besides - and the – needs to be at the beginning of the value only |
| Pre-conditions | The variable given is an integer and or string but contains only positive or negative number and no letters or special characters |
| Post-condition | Return **True** |

|  |  |
| --- | --- |
| Code | E\_1.16 |
| Name | Security |
| Type | Non-Functional Requirement |
| Criticality | High |
| Priority | 9/10 |
| Inputs | A string |
| Source | - |
| Output | - |
| Description | - |
| Destination | - |
| Action | The security of the users for logging into the system is important and the security of some sensitive data is quite important in any database system this lead to the creation of the 2 following functional requirements (E\_1.17 hash) , (E\_1.18 dehash), (E\_1.21 removeSingleQuotes)**.** |
| Requirements |  |
| Expected Risks | - |
| Pre-conditions | - |
| Post-condition | - |

|  |  |
| --- | --- |
| Code | E\_1.16 |
| Name | hash |
| Type | Functional Requirement |
| Criticality | High |
| Priority | 9/10 |
| Inputs | A string |
| Source | - |
| Output | Hashed string |
| Description | This function receives a string and hashes it to some hashed value |
| Destination | - |
| Action | This function returns a string of the hashed parameter sent to it by password Default**.** |
| Requirements | A variable to be passed that is not null |
| Expected Risks | The variable sent is null |
| Pre-conditions | The variable sent is not null |
| Post-condition | Returns a string of the hashed string |

|  |  |
| --- | --- |
| Code | E\_1.17 |
| Name | dehash |
| Type | Functional Requirement |
| Criticality | High |
| Priority | 9/10 |
| Inputs | Hashed string and string |
| Source | - |
| Output | **True or False** |
| Description | This function will receive a parameter that is hashed and one that is normal and compare whether they match or not |
| Destination | - |
| Action | This function will receive 2 parameters 1 is a normal string and another is a hashed string and will compare the normal string to the hashed string after dehashing the hashed string and will return **True** if it is valid and **False** if it’s not valid |
| Requirements | A variable to be passed that is not null |
| Expected Risks | The variable sent is null |
| Pre-conditions | The variable sent is not null |
| Post-condition | Returns **True** |

|  |  |
| --- | --- |
| Code | E\_1.18 |
| Name | submitThesisDetails |
| Type | Functional Requirement |
| Criticality | Very High |
| Priority | 10/10 |
| Inputs | Object of type registeration and the student ID |
| Source | - |
| Output | **Insert the student’s thesis to the database** |
| Description | This function is responsible for inserting the students current thesis into the database |
| Destination | - |
| Action | This function will receive an object of type registeration that contains the Registeration ID of the student, thesis ID, and a file uploaded by the student and all of this will be uploaded in the regestierationDetails Table in the database |
| Requirements | - |
| Expected Risks | - |
| Pre-conditions | - |
| Post-condition | Returns the student to the home page |

|  |  |
| --- | --- |
| Code | E\_1.19 |
| Name | updateGrade |
| Type | Functional Requirement |
| Criticality | Very High |
| Priority | 10/10 |
| Inputs | The registeration detail ID, the statue power value, a file text, feedback and thesisupload time stamp |
| Source | - |
| Output | **Change the student’s grade in the database** |
| Description | This function will allow the doctor (professor) to give a grade to the student based on the phase that was submitted |
| Destination | - |
| Action | This will function will allow the doctor to download update the student’s grade in the registerationDetail Table in the database by choosing a statue power from a drop down table that will have it’s values retrieved from the look up table “status” and it will affect the student’s grade and affect the student user type if he passes he will be returned to the userType in the table userType in the database that instates he can upload any thesis submission but if he was graded to fail his last recent registeration detail record will be deleted from the table registerationDetials so when it’s time to upload again the student will have to upload the phase he failed in again |
| Requirements | - |
| Expected Risks | - |
| Pre-conditions | - |
| Post-condition | Updates the student grade and usertype in the database |

|  |  |
| --- | --- |
| Code | E\_1.20 |
| Name | queryStudentRegisterations |
| Type | Functional Requirement |
| Criticality | Very High |
| Priority | 9/10 |
| Inputs | It will query all the student who have submitted a thesis or at least were accepted from the graduation registeration phase and will return an array of Class CRegisteration |
| Source | - |
| Output | **-** |
| Description | This function will return an object of all students who passed the graduation request stage |
| Destination | - |
| Action | This function will return an array of objects of CRegesteration that contains all the students who passed the graduation registeration phase |
| Requirements | - |
| Expected Risks | - |
| Pre-conditions | - |
| Post-condition | Return array of class CRegisteration |

|  |  |
| --- | --- |
| Code | E\_1.21 |
| Name | removeSingleQuotes |
| Type | Functional Requirement |
| Criticality | High |
| Priority | 8/10 |
| Inputs | Any String or integer |
| Source | - |
| Output | **The same string or integer without the single quote (‘)** |
| Description | This function will return the parameter sent to it without any single qutoes |
| Destination | - |
| Action | This function will receive a parameter and it will replace the single quotes (‘) in that parameter with (\’) as the single quotes is harmful to the database and to abide to the security constraints on the system, any single quote that’s an input from the user must be turned into \’ to avoid the dangers of SQL Injections from the database |
| Requirements | - |
| Expected Risks | The parameter sent is not of type string or integer or varchar, I.E. it’s of type object for example |
| Pre-conditions | - |
| Post-condition | Return array of the parameter sent with replaced single quotes |

|  |  |
| --- | --- |
| Code | E\_1.22 |
| Name | Portability |
| Type | Non-Functional Requirement |
| Criticality | High |
| Priority | 8/10 |
| Inputs |  |
| Source | - |
| Output |  |
| Description |  |
| Destination | - |
| Action | The Portability Non-Functional Requirement is in need because of the system requirements so in order to achieve the portability Constraints on the systems, this function will be place constraints on the system such as making the system a web application system, making the system mobile responsive, these constraints and with the organization constraints requires developing the web application system use native php without any frameworks or APIs, this also resulted in the function requirement (E\_1.23 makeSystemMobileResponsive), the php development will be done in the MVC (Model View Controller) Design Pattern Style |
| Requirements | - |
| Expected Risks | - |
| Pre-conditions | - |
| Post-condition | - |

|  |  |
| --- | --- |
| Code | E\_1.22 |
| Name | makeSystemMobileResponsive |
| Type | Functional Requirement |
| Criticality | Medium |
| Priority | 5/10 |
| Inputs | - |
| Source | - |
| Output | **-** |
| Description | This function is to describe how to make the system mobile responsive |
| Destination | - |
| Action | This function is a result of the non-functional requirements of the portability and thus the system needs to be mobile responsive and this function will be done using raw html, css and javascript without any frameworks or APIs to make the application mobile responsive |
| Requirements | - |
| Expected Risks | - |
| Pre-conditions | - |
| Post-condition | - |

|  |  |
| --- | --- |
| Code | E\_1.23 |
| Name | queryNextThesisUpload |
| Type | Functional Requirement |
| Criticality | High |
| Priority | 9/10 |
| Inputs | CRegisteration Object |
| Source | - |
| Output | **The next thesis ID and Name that is required from the student to upload** |
| Description | This function will query to get student’s next thesis to upload |
| Destination | - |
| Action | Will query on the student to check his next thesis upload in the order values of the thesis table according to his past thesis uploads, if he hadn’t uploaded any thesis before then he will start with the thesis order with the minimal order value, if he had uploaded all his thesis then nothing will appear |
| Requirements | - |
| Expected Risks | - |
| Pre-conditions | - |
| Post-condition | - |

|  |  |
| --- | --- |
| Code | E\_1.24 |
| Name | viewForm |
| Type | Functional Requirement |
| Criticality | Medium |
| Priority | 7/10 |
| Inputs | CRegisteration object |
| Source | - |
| Output | **Display the Form of Doctor Accepting or Rejecting Students Graduation Requests** |
| Description | Display the Form of Doctor Accepting or Rejecting Students Graduation Requests |
| Destination | - |
| Action | This function will take a parameter of type CRegisteration and it will display the html of the doctor Accepting or Rejecting Students Graduation Requests according to the object sent |
| Requirements | - |
| Expected Risks | - |
| Pre-conditions | - |
| Post-condition | - |

|  |  |
| --- | --- |
| Code | E\_1.25 |
| Name | viewGradeStudentForm |
| Type | Functional Requirement |
| Criticality | Medium |
| Priority | 7/10 |
| Inputs | CRegisterationDetails Array of objects, status array |
| Source | - |
| Output | **Display the Form of Doctor Grading Students Thesis Uploads** |
| Description | Display the Form of Doctor Grading Students Thesis Uploads |
| Destination | - |
| Action | This function will take a parameter of type CRegisterationDetails and status array and it will display the html of the doctor Grading Students Thesis uploads according to the object sent the student’s details will be displayed |
| Requirements | - |
| Expected Risks | - |
| Pre-conditions | - |
| Post-condition | - |

|  |  |
| --- | --- |
| Code | A\_1.8 |
| Name | viewAllStudentsGraduationRequests |
| Type | Functional Requirement |
| Criticality | Medium |
| Priority | 7/10 |
| Inputs | Registeration Array |
| Source | - |
| Output | **Display the Form of Doctor viewing all students graduation request** |
| Description | Display the Form of Doctor viewing all students graduation request |
| Destination | - |
| Action | The function will take parameter of type registreation array and echo the html all students gradations request by Students Id’s and the requests will be displayed |
| Requirements | - |
| Expected Risks | - |
| Pre-conditions | - |
| Post-condition | - |

|  |  |
| --- | --- |
| Code | A\_1.9 |
| Name | viewSpecificStudentThesisRegisterationDetails |
| Type | Functional Requirement |
| Criticality | Medium |
| Priority | 7/10 |
| Inputs | Registerationdetails Array |
| Source | - |
| Output | **Display the Form of Doctor viewing specific studnet**  **graduation details** |
| Description | Display the Form of Doctor viewing specific studnet graduation details |
| Destination | - |
| Action | The function will take parameter of type registreationdetails array and echo the html specifc students gradation details by Students Id’s and the specific student details will be displayed |
| Requirements | - |
| Expected Risks | - |
| Pre-conditions | - |
| Post-condition | - |

|  |  |
| --- | --- |
| Code | A\_1.10 |
| Name | ViewTableStudentes |
| Type | Functional Requirement |
| Criticality | Medium |
| Priority | 7/10 |
| Inputs | Registeration Array |
| Source | - |
| Output | **Display the Form of Doctor viewing all students** |
| Description | Display the Form of Doctor viewing all students |
| Destination | - |
| Action | The function will take parameter of type registreation array and echo the html of all students gradation by Students Id’s and the all students will be displayed |
| Requirements | - |
| Expected Risks | - |
| Pre-conditions | - |
| Post-condition | - |

|  |  |
| --- | --- |
| Code | A\_1.11 |
| Name | addLoginHTML |
| Type | Functional Requirement |
| Criticality | Medium |
| Priority | 7/10 |
| Inputs | Html objects |
| Source | - |
| Output | **Add the html login form** |
| Description | Add the html login form |
| Destination | - |
| Action | The function will take parameter of type html object and echo the html of the login form |
| Requirements | - |
| Expected Risks | - |
| Pre-conditions | - |
| Post-condition | - |

|  |  |
| --- | --- |
| Code | A\_1.12 |
| Name | addLoginCSS |
| Type | Functional Requirement |
| Criticality | Medium |
| Priority | 7/10 |
| Inputs | CSS objects |
| Source | - |
| Output | **Add the css login form** |
| Description | Add the css login form |
| Destination | - |
| Action | The function will take parameter of type css object and echo the css of the login form |
| Requirements | - |
| Expected Risks | - |
| Pre-conditions | - |
| Post-condition | - |

|  |  |
| --- | --- |
| Code | A\_1.13 |
| Name | viewNavBar |
| Type | Functional Requirement |
| Criticality | Medium |
| Priority | 7/10 |
| Inputs | Userprivalgiesfiles |
| Source | - |
| Output | **Add the navbar of the specific user** |
| Description | Add the navbar of the specific user |
| Destination | - |
| Action | The function will take parameter of type object and echo the navbar of the specific user and its css  And include the user privilege |
| Requirements | - |
| Expected Risks | - |
| Pre-conditions | - |
| Post-condition | - |

|  |  |
| --- | --- |
| Code | A\_1.14 |
| Name | viewThesisForm |
| Type | Functional Requirement |
| Criticality | Medium |
| Priority | 7/10 |
| Inputs | Students object |
| Source | - |
| Output | **view the thesis form to the student** |
| Description | **view the thesis form to the student** |
| Destination | - |
| Action | The function will take parameter of type student and echo the thesis form to the student and its css  And html and displayed student thesis |
| Requirements | - |
| Expected Risks | - |
| Pre-conditions | - |
| Post-condition | - |

|  |  |
| --- | --- |
| Code | A\_2.1 |
| Name | addNavBarCSS |
| Type | Functional Requirement |
| Criticality | medium |
| Priority | 6/10 |
| Inputs | - |
| Source | - |
| Output | Will include the CSS navbar links |
| Description | This will include the CSS style navbar |
| Destination | - |
| Action | The system will include the necessary CSS style links to apply the styles on the navbar and CSS Links |
| Requirements | All the CSS links |
| Expected Risks | - |
| Pre-conditions | - |
| Post-condition | Add the styles to the navbar |

|  |  |
| --- | --- |
| Code | A\_2.2 |
| Name | addNavBar |
| Type | Functional Requirement |
| Criticality | medium |
| Priority | 7/10 |
| Inputs | user Type, user id |
| Source | - |
| Output | Will include the navbar depends on the user type |
| Description | This will include the navbar |
| Destination | - |
| Action | The system will display the navber then check if the user type equal 1 then it’s a doctor and will add doctors navbar view  Then if the user type equal 2 then it’s a student and it will check students nav bar status (A\_2.3) if it false will view the student register grad nav bar  If the check students nav bar status (A\_2.3) equal 1 then  It will show the up- home navbar else it will view the Student View Registration navbar |
| Requirements | use check students nav bar status (A\_2.3) |
| Expected Risks | Unknown user type |
| Pre-conditions | - |
| Post-condition | Add the correct navbar |

|  |  |
| --- | --- |
| Code | A\_2.4 |
| Name | submitRegisteration |
| Type | Functional Requirement |
| Criticality | Very High |
| Priority | 9/10 |
| Inputs | Registeration |
| Source | - |
| Output | It will insert the students id, credit hours, semerterid |
| Description | It will save the students id, credit hours, semerterid into database |
| Destination | - |
| Action | It will make query to insert the students id, credit hours, semerterid into the registration table in the database so he can register for graduation project |
| Requirements | - |
| Expected Risks | - |
| Pre-conditions | - |
| Post-condition | It will insert the student’s graduation registeration in the database and will change the student user type |

|  |  |
| --- | --- |
| Code | A\_2.5 |
| Name | acceptStudent |
| Type | Functional Requirement |
| Criticality | medium |
| Priority | 6/10 |
| Inputs | Student id, database |
| Source | - |
| Output | It will update the acceptance status |
| Description | It will update the student’s acceptance status into database |
| Destination | - |
| Action | It will make query to update the selected student id , acceptance status into the registration table in the database and use function getConnection (M\_3.2)  To connect to the database |
| Requirements | (M\_3.2) |
| Expected Risks | - |
| Pre-conditions | - |
| Post-condition | It will update the student into the database table |

|  |  |
| --- | --- |
| Code | A\_2.6 |
| Name | rejectStudent |
| Type | Functional Requirement |
| Criticality | medium |
| Priority | 6/10 |
| Inputs | Student id, database |
| Source | - |
| Output | It will reject and delete the acceptance status |
| Description | It will reject and delete the student’s acceptance status into database |
| Destination | - |
| Action | It will make query to reject and delete the selected student id , acceptance status into the registration table in the database use function getConnection (M\_3.2)  To connect to the database |
| Requirements | (M\_3.2) |
| Expected Risks | - |
| Pre-conditions | - |
| Post-condition | It will reject and delete the student into the database table |

|  |  |
| --- | --- |
| Code | A\_2.7 |
| Name | queryStudentsRegisterations |
| Type | Functional Requirement |
| Criticality | medium |
| Priority | 7/10 |
| Inputs | database |
| Source | - |
| Output | It will show the Registered students |
| Description | It will show Registered students in a table from the database |
| Destination | - |
| Action | It will make query to show Registered students from the registration table in the database use function getConnection (M\_3.2)  To connect to the database |
| Requirements | (M\_3.2) |
| Expected Risks | - |
| Pre-conditions | - |
| Post-condition | It will show Registered students the student from the database table |

|  |  |
| --- | --- |
| Code | M\_3.2 |
| Name | getConnection |
| Type | Functional Requirement |
| Criticality | medium |
| Priority | 7/10 |
| Inputs |  |
| Source | Used by other functions to establish connection with database |
| Output | Establish connection with database. To make the system able to execute SQL statements |
| Description | Establish connection with database |
| Destination | - |
| Action | Establish connection with the database. So the system can retrieve, add ,delete and update data in the database with the SQL statements |
| Requirements | The database Connection is set to the Correct hostname and database name |
| Expected Risks | The developer inputs wrong hostname or wrong username or wrong password or wrong name of the database |
| Pre-conditions | - |
| Post-condition | Can retrieve or add or delete or update data in the database from the established connection |

|  |  |
| --- | --- |
| Code | M\_3.3 |
| Name | queryGetTimeStamp |
| Type | Functional Requirement |
| Criticality | medium |
| Priority | 7/10 |
| Inputs | Registration ID of the student (int regID); the database class (CDatabase db); |
| Source | E\_1.3 |
| Output | It returns the date of the student’s registration |
| Description | It returns the date of the student’s registration |
| Destination | - |
| Action | The function will do a SQL statement to retrieve the date of the student’s registration from the database and the function (E\_1.3 query) will execute the SQL statement and another function called (M\_3.2 getConnection) will establish the connection with the database |
| Requirements | The database connection and the registration id of the student |
| Expected Risks | The database connection is not established correctly |
| Pre-conditions | A doctor must be logged in to the system so the system can retrieve the date of the student’s registration |
| Post-condition | The doctor will see the date of each student registration in a table |

|  |  |
| --- | --- |
| Code | M\_3.4 |
| Name | querySemesterNameByStudentID |
| Type | Functional Requirement |
| Criticality | medium |
| Priority | 7/10 |
| Inputs | The student ID (int studentID); |
| Source | E\_1.3 |
| Output | It returns the Semester name of the student’s registration. |
| Description | It returns the Semester name of the student’s registration. |
| Destination | - |
| Action | The function will do a SQL statement to retrieve the SemesterID of the student from the database and the function (E\_1.3 query) will execute the SQL statement and another function called (M\_3.2 getConnection) will establish the connection with the database and the function (M\_3.5 querySemesterName) will use the SemesterId to retrieve the SemesterName |
| Requirements | The database connection and the id of the student |
| Expected Risks | The database connection is not established correctly |
| Pre-conditions | A doctor must be logged in to the system so the system can retrieve the semester name of the student |
| Post-condition | The doctor will see the semester name of each student in a table |

|  |  |
| --- | --- |
| Code | M\_3.5 |
| Name | querySemesterName |
| Type | Functional Requirement |
| Criticality | medium |
| Priority | 7/10 |
| Inputs | The semester ID (int semesterID); the database class (CDatabase db); |
| Source | E\_1.3 |
| Output | It returns the Semester name of the student’s registration from the semester ID |
| Description | It returns the Semester name of the student’s registration from the semester ID |
| Destination | - |
| Action | The function will do a SQL statement to retrieve the SemesterName of the student from the database and the function (E\_1.3 query) will execute the SQL statement and another function called (M\_3.2 getConnection) will establish the connection with the database |
| Requirements | The database connection and the semester id of the student |
| Expected Risks | The database connection is not established correctly |
| Pre-conditions | A doctor must be logged in to the system so the system can retrieve the semester name of the student |
| Post-condition | The doctor will see the semester name of each student in a table |

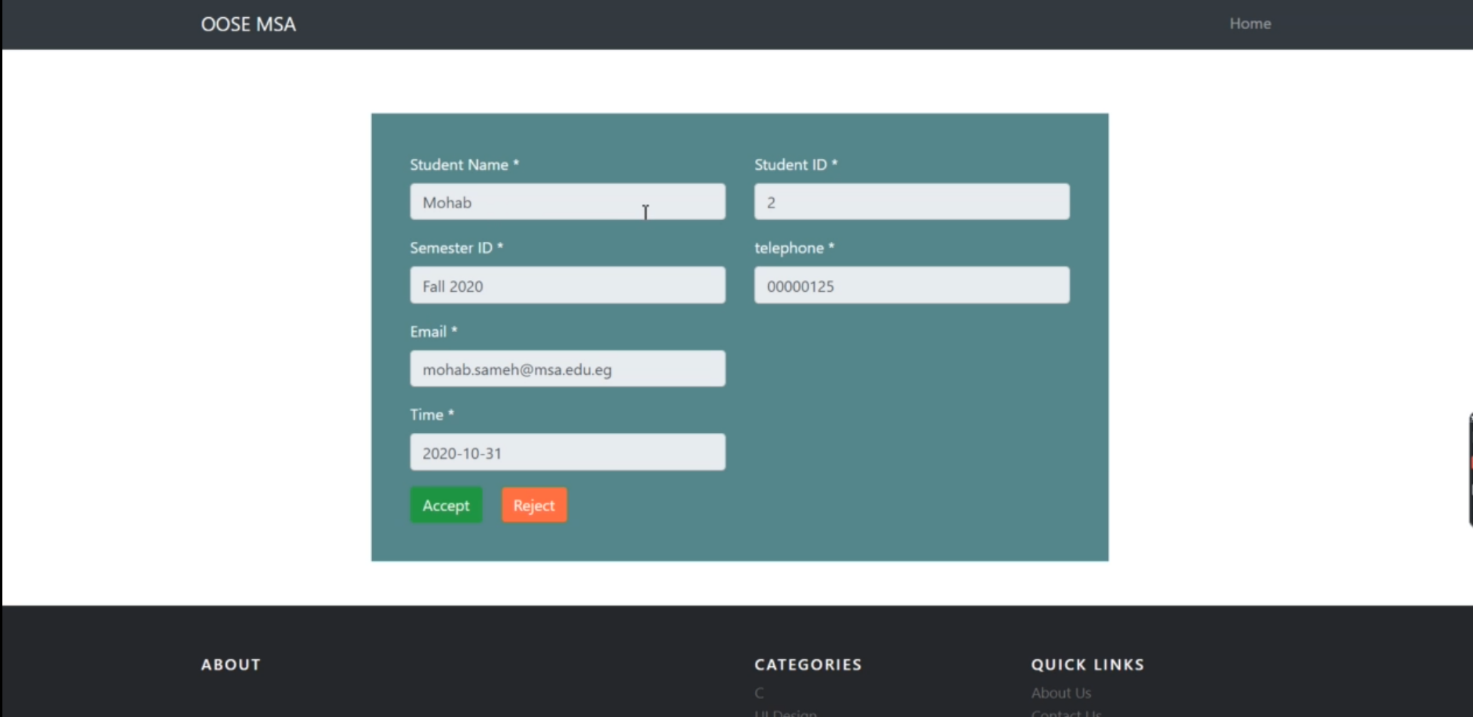
|  |  |
| --- | --- |
| Code | M\_3.6 |
| Name | queryStudentRegisterationID |
| Type | Functional Requirement |
| Criticality | medium |
| Priority | 7/10 |
| Inputs | The student ID (int semesterID); the database class (CDatabase db); |
| Source | E\_1.3 |
| Output | It returns the registration ID from the student’s ID |
| Description | It returns the registration ID from the student’s ID |
| Destination | - |
| Action | The function will do a SQL statement to retrieve the registration ID of the student from the database and the function (E\_1.3 query) will execute the SQL statement and another function called (M\_3.2 getConnection) will establish the connection with the database |
| Requirements | The database connection and the id of the student |
| Expected Risks | The database connection is not established correctly and the student is not registered |
| Pre-conditions |  |
| Post-condition | It will return the registration id of the student if it is available and will the registration id will be used in other function to retrieve data from the data base like the function (M\_3.1 queryAcceptanceStatus, M\_3.3 queryGetTimeStamp) |

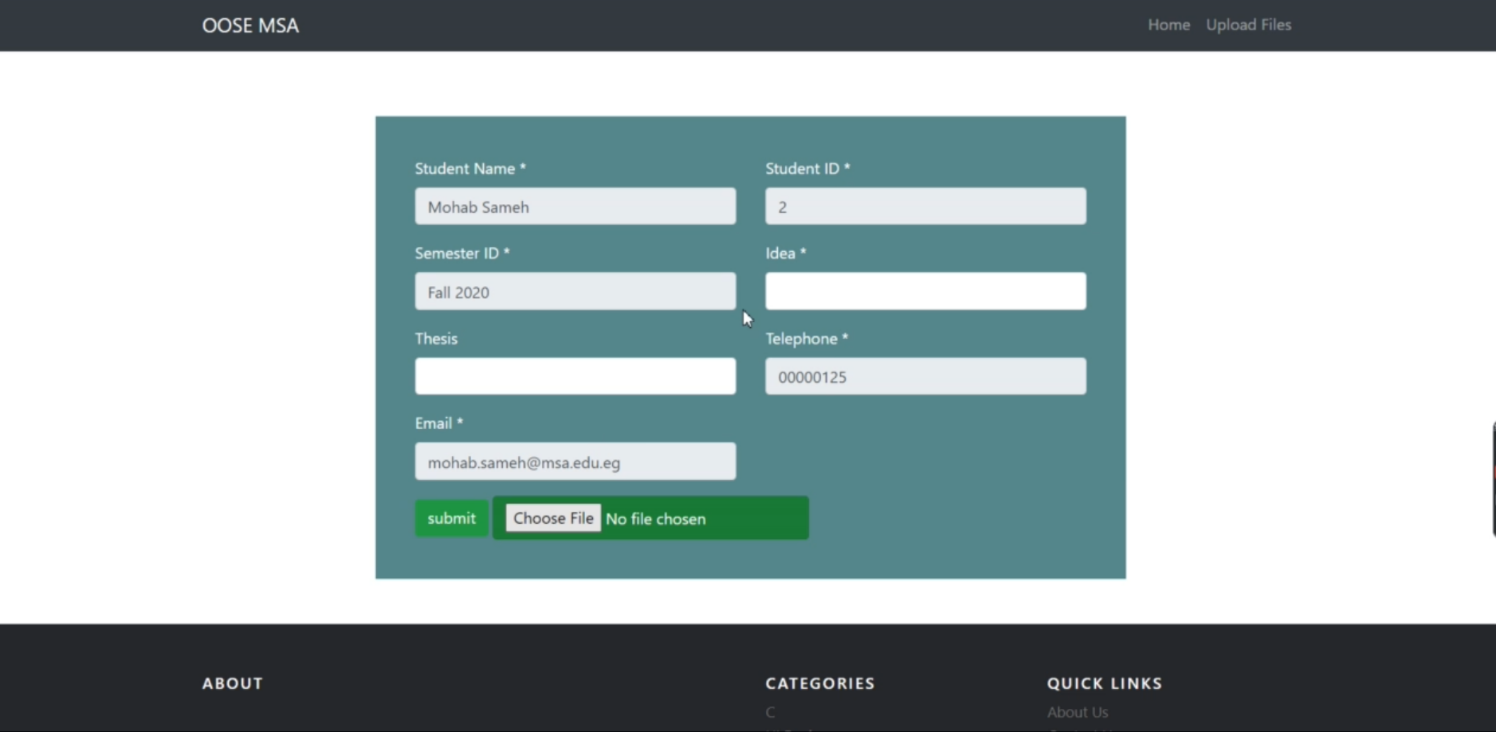
# **4 Interface Requirements**

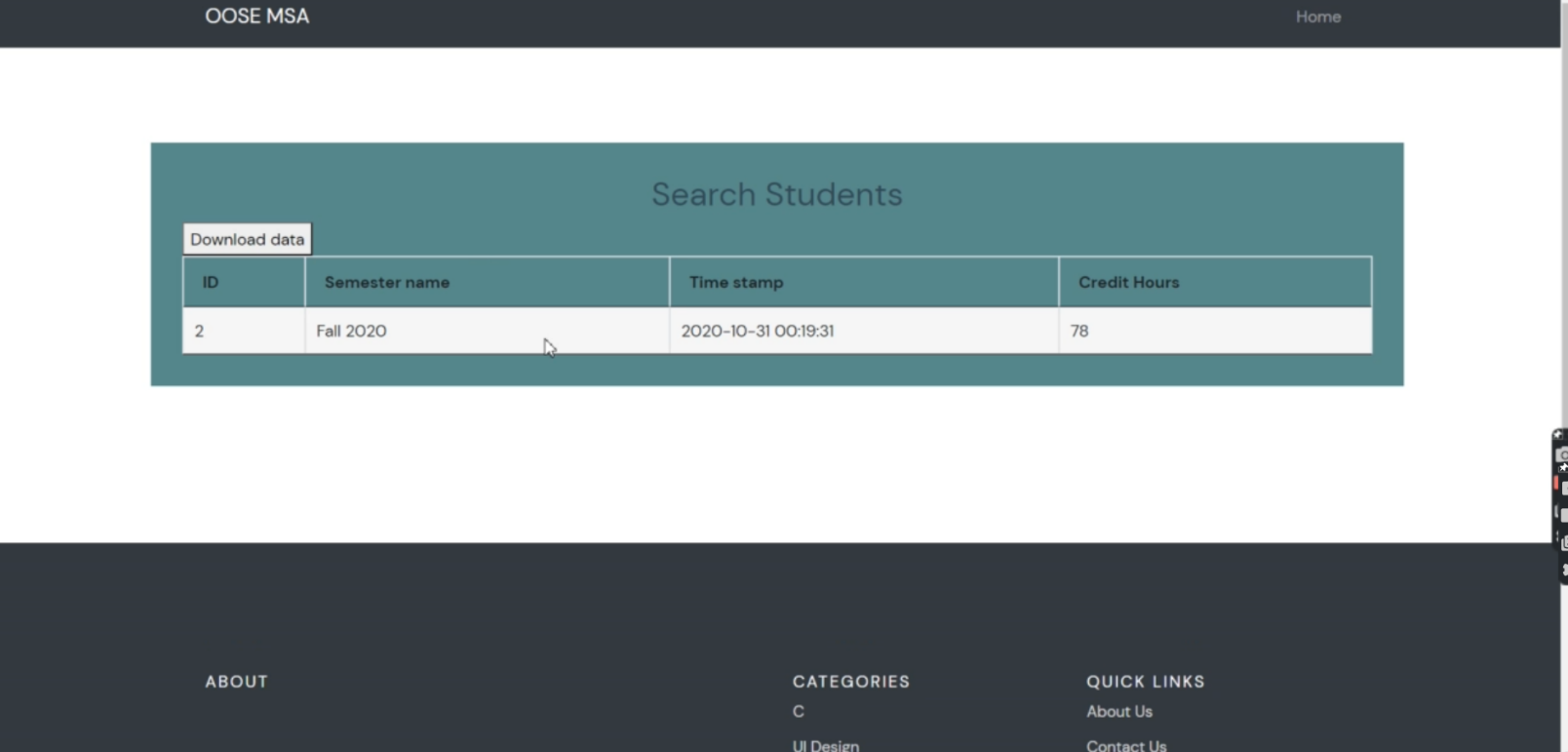
This section describes how the software interfaces with other software products or users for input or output. Examples of such interfaces include library routines, token streams, shared memory, data streams, and so forth.

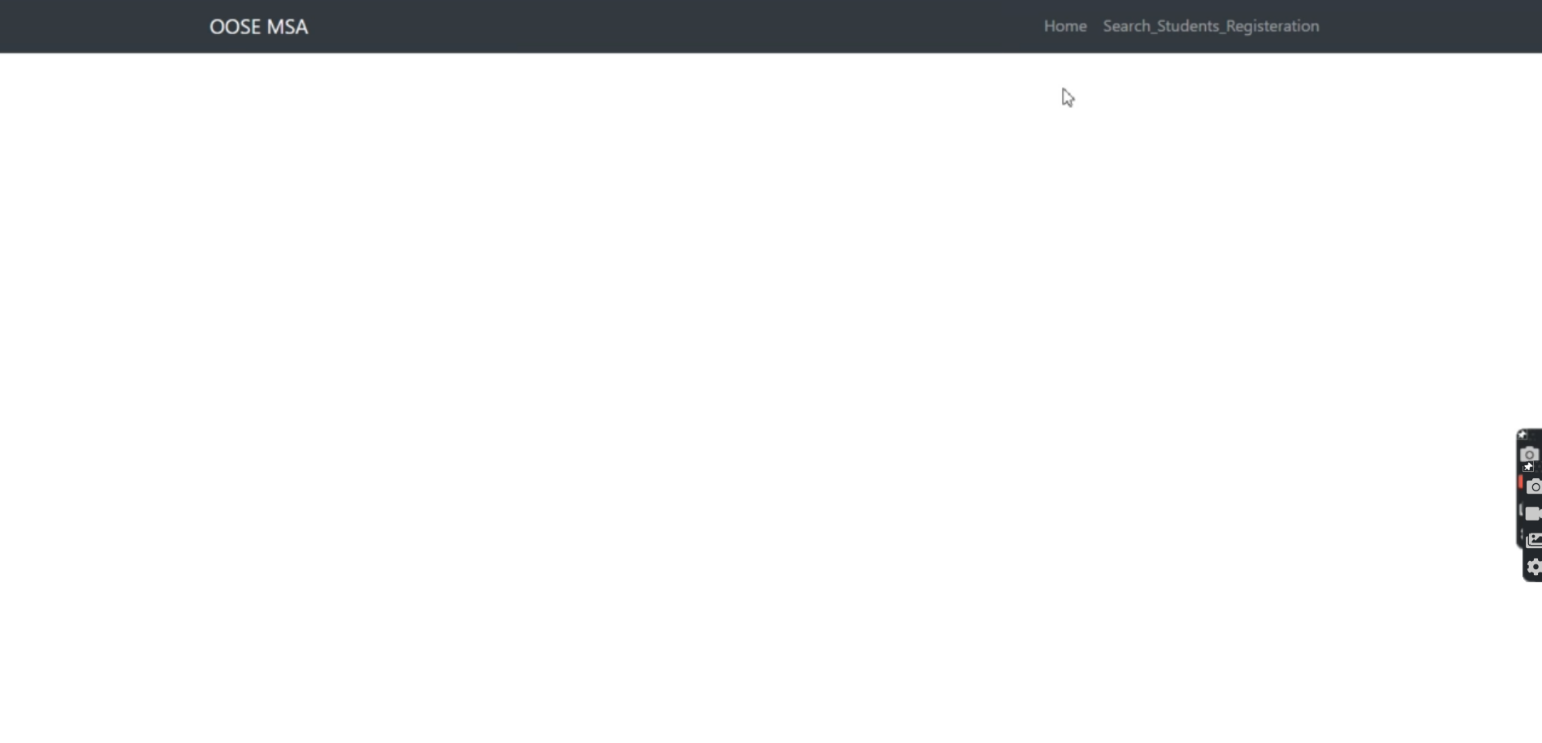
## **4.1 User Interfaces**

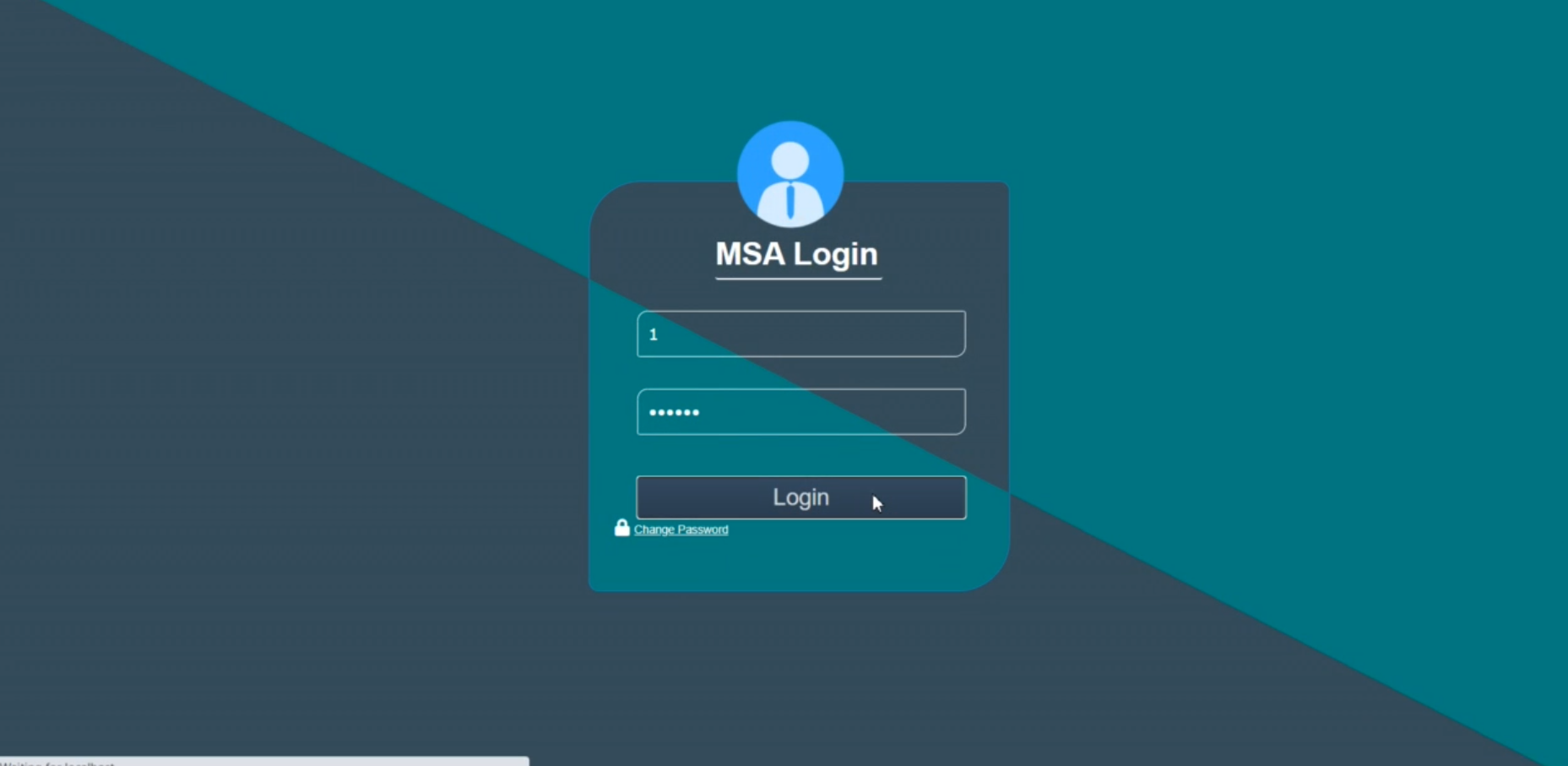
**4.1.1 GUI**

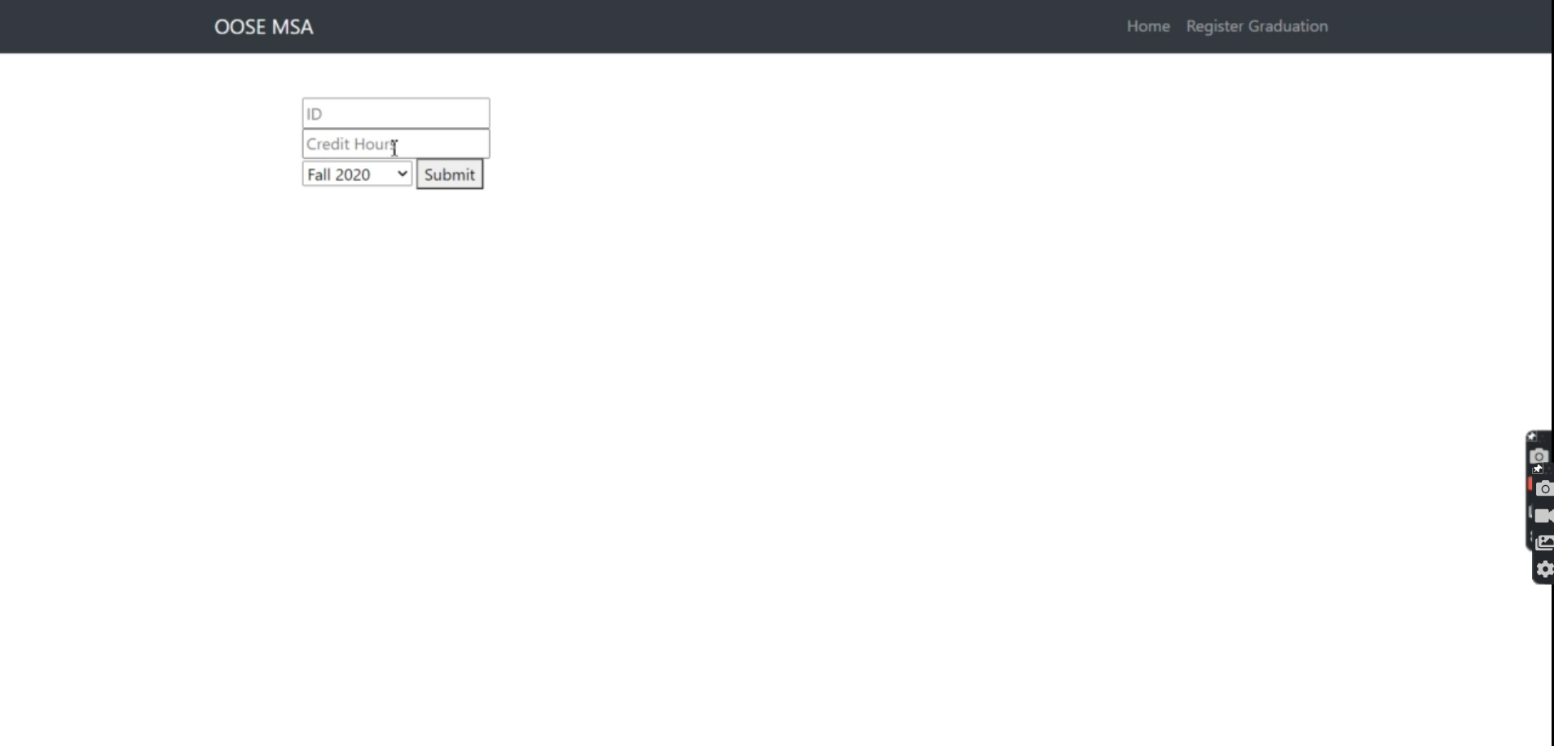












**4.1.2 CLI**

NO COMAND LINE.

**4.1.3 API**

NO API.

**4.1.4 Diagnostics or ROM**

-Using a Web application.

**4.2 Hardware Interfaces**

Printers will interface with the system in order to scan and produce documents.

**4.3 Communications Interfaces**

The FTP, HTTPS.

**4.4 Software Interfaces**

None

**5 Performance Requirements**

The system should take at the very basic tasks no more than 5 seconds and required minimum of 8 GB of RAM.

**6 Design Constraints**

The system must be mobile responsive.

**6.1 Standards Compliance**

NONE.

**6.2 Hardware Limitations**

The system will be able to run on any device

**6.3 others as appropriate**

None

# **7 Other non-functional attributes**

Specifies any other particular non-functional attributes required by the system. Examples are provided below.

**7.1 Security**

-Any user will have their password hashed .(Hashing technique)

-Any information about the students or doctor will be encrypted (openssl encryption)

**7.2 Reliability**

The system needs to be online all day with maximum system failure time of 5 to 10 seconds in total per day, Thus leading to upgrading our server to be more powerful and avoid system failures as much as we can.

**7.3 Portability**

The system is highly portable since it’s web application thus leading to make the web application mobile responsive so it can be portable on PCs , mobile devices and tablets this will be measured by the number of devices that the website works on without any errors and without being down.

**7.4 Application Affinity/Compatibility**

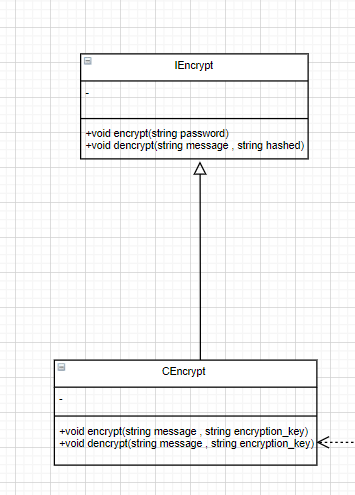
The system shall be compatible with any device browser thus leading to make our web application system mobile responsive to be able to be compatible with any device whatever it was and this will be measured through the user interface from whole devices type that a user can use.

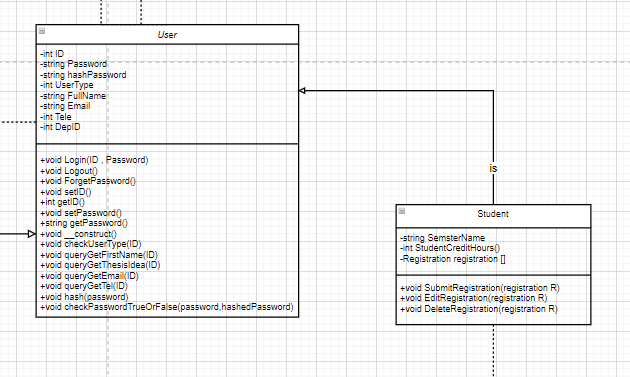
# **8 Preliminary Object-Oriented Domain Analysis**

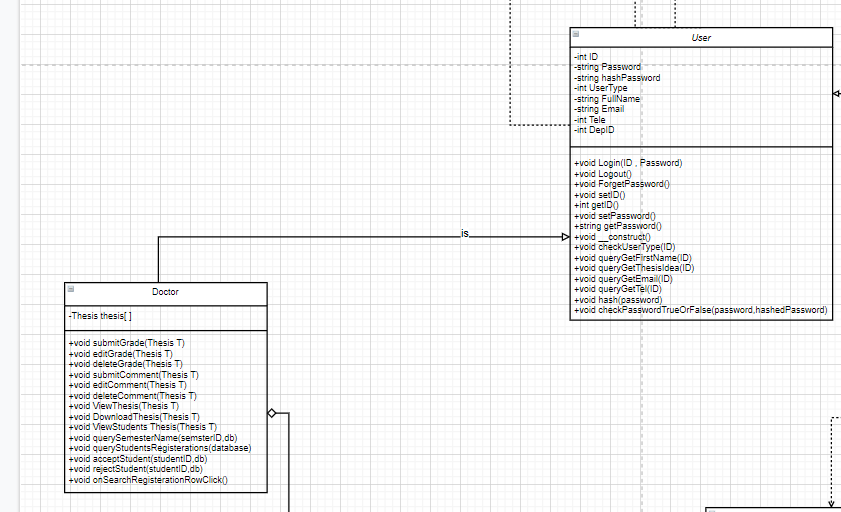
This section presents a list of the fundamental objects that must be modeled within the system to satisfy its requirements. The purpose is to provide an alternative, ”structural” view on the requirements stated above and how they might be satisfied in the system. A primitive class diagram to be delivered.

## **8.1 Inheritance Relationships**

## 







## **8.2 Class descriptions**

This section presents a more detailed description of each class identified during the OO Domain Analysis. For more details on the process giving rise to these descriptions, see Lecture 5.3: OO Domain Analysis and/or texts on object-oriented software development. Each class description should conform to the following structure:

**8.2.1 Class name**

Abstract or Concrete: Indicates whether this class is abstract or concrete.

All in CRC CARDS

**8.2.2 List of Superclasses:**

Iencrypt , Ihash , IconstraintsCheck , User

**8.2.3 List of Subclasses:**

Doctor, Student, Thesis, Registration, RegistrationDetails, RegistrationDetialsSatus, Cdatabase, Cconstraint, Chash, Cencrypt, CSS, HTML.

**8.2.4 Purpose:**

The main purpose of the class is to deliver a simple design to give the developer a clear way to dip in with his code.

ALL IN CRC CARD

**/////////////////////////////////// States the basic purpose of the class. ///////////////////////////////////////////////////**

**8.2.5 Collaborations:**

|  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- |
| Front | |  |  |  |  |  |  |
|  | Class Name: IEncrypt | | Id: 1 | | Type: interface | |  |
|  | Description: it is an interface class that encrypt passwords | | | | Aggregation Use cases:- 0  Associated Use cases:- 0  Inheritance Use cases:- 0  Composition Use cases:- 0 | |  |
|  | Responsibilities  Encrypt  Decrypt | | | Collaborators | | |  |
|  |  |  |  |  |  |  |  |
| Back | |  |  |  |  |  |  |
|  | Attributes | | | | | |  |
|  |  | | |  | | |  |
|  |  | | |  | | |  |
|  |  | | |  | | |  |
|  |  | | |  | | |  |
|  |  | | |  | | |  |
|  |  | | |  | | |  |
|  | Relationships | | |  | | |  |
|  | Generalization (A Kind Of): | | |  | | |  |
|  | Aggregation (Part-of): | | | -- | | |  |
|  | Composition: | | | -- | | |  |
|  | Other Association: | | |  | | |  |
|  |  |  |  |  |  |  |  |

|  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- |
| Front | |  |  |  |  |  |  |
|  | Class Name: CEncrypt | | Id: 2 | | Type: Concrete | |  |
|  | Description: the class of encryption | | | | Aggregation Use cases:- 0  Associated Use cases:- 1  Inheritance Use cases:- 1  Composition Use cases:- 0 | |  |
|  | Responsibilities  Encrypt  Decrypt | | | Collaborators | | |  |
|  |  |  |  |  |  |  |  |
| Back | |  |  |  |  |  |  |
|  | Attributes | | | | | |  |
|  |  | | |  | | |  |
|  |  | | |  | | |  |
|  |  | | |  | | |  |
|  |  | | |  | | |  |
|  |  | | |  | | |  |
|  |  | | |  | | |  |
|  | Relationships | | |  | | |  |
|  | Generalization (A Kind Of): | | | IEncrypt | | |  |
|  | Aggregation (Part-of): | | | -- | | |  |
|  | Composition: | | | -- | | |  |
|  | Other Association: | | | User | | |  |
|  |  |  |  |  |  |  |  |

|  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- |
| Front | |  |  |  |  |  |  |
|  | Class Name: Ihash | | Id: 3 | | Type: Interface | |  |
|  | Description: it is interface class that hashes password | | | | Aggregation Use cases:- 0  Associated Use cases:- 0  Inheritance Use cases:- 0  Composition Use cases:- 0 | |  |
|  | Responsibilities  Hash  Dehash | | | Collaborators | | |  |
|  |  |  |  |  |  |  |  |
| Back | |  |  |  |  |  |  |
|  | Attributes | | | | | |  |
|  |  | | |  | | |  |
|  |  | | |  | | |  |
|  |  | | |  | | |  |
|  |  | | |  | | |  |
|  |  | | |  | | |  |
|  |  | | |  | | |  |
|  | Relationships | | |  | | |  |
|  | Generalization (A Kind Of): | | |  | | |  |
|  | Aggregation (Part-of): | | | -- | | |  |
|  | Composition: | | | -- | | |  |
|  | Other Association: | | |  | | |  |
|  |  |  |  |  |  |  |  |

|  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- |
| Front | |  |  |  |  |  |  |
|  | Class Name: Chash | | Id: 4 | | Type: Concrete | |  |
|  | Description: the class of hashing | | | | Aggregation Use cases:- 0  Associated Use cases:- 1  Inheritance Use cases:- 1  Composition Use cases:- 0 | |  |
|  | Responsibilities  Hash  dehash | | | Collaborators | | |  |
|  |  |  |  |  |  |  |  |
| Back | |  |  |  |  |  |  |
|  | Attributes | | | | | |  |
|  |  | | |  | | |  |
|  |  | | |  | | |  |
|  |  | | |  | | |  |
|  |  | | |  | | |  |
|  |  | | |  | | |  |
|  |  | | |  | | |  |
|  | Relationships | | |  | | |  |
|  | Generalization (A Kind Of): | | | Ihash | | |  |
|  | Aggregation (Part-of): | | | -- | | |  |
|  | Composition: | | | -- | | |  |
|  | Other Association: | | | User | | |  |
|  |  |  |  |  |  |  |  |

|  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- |
| Front | |  |  |  |  |  |  |
|  | Class Name: HTML | | Id: 5 | | Type: Concrete | |  |
|  | Description: the class that includes the html of the web page | | | | Aggregation Use cases:- 0  Associated Use cases:- 1  Inheritance Use cases:- 0  Composition Use cases:- 0 | |  |
|  | Responsibilities  \_\_construct  checkStudentNavBarStatus  addNavBar | | | Collaborators  database | | |  |
|  |  |  |  |  |  |  |  |
| Back | |  |  |  |  |  |  |
|  | Attributes | | | | | |  |
|  | Database database | | |  | | |  |
|  |  | | |  | | |  |
|  |  | | |  | | |  |
|  |  | | |  | | |  |
|  |  | | |  | | |  |
|  |  | | |  | | |  |
|  | Relationships | | |  | | |  |
|  | Generalization (A Kind Of): | | |  | | |  |
|  | Aggregation (Part-of): | | | -- | | |  |
|  | Composition: | | | -- | | |  |
|  | Other Association: | | | CDatabase | | |  |
|  |  |  |  |  |  |  |  |

|  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- |
| Front | |  |  |  |  |  |  |
|  | Class Name: css | | Id: 6 | | Type: Concrete | |  |
|  | Description: the class that includes the css of the web page | | | | Aggregation Use cases:- 0  Associated Use cases:- 1  Inheritance Use cases:- 0  Composition Use cases:- 0 | |  |
|  | Responsibilities  addNavBarStyleLinks | | | Collaborators  database | | |  |
|  |  |  |  |  |  |  |  |
| Back | |  |  |  |  |  |  |
|  | Attributes | | | | | |  |
|  | Database database | | |  | | |  |
|  |  | | |  | | |  |
|  |  | | |  | | |  |
|  |  | | |  | | |  |
|  |  | | |  | | |  |
|  |  | | |  | | |  |
|  | Relationships | | |  | | |  |
|  | Generalization (A Kind Of): | | |  | | |  |
|  | Aggregation (Part-of): | | | -- | | |  |
|  | Composition: | | | -- | | |  |
|  | Other Association: | | | CDatabase | | |  |
|  |  |  |  |  |  |  |  |

|  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- |
| Front | |  |  |  |  |  |  |
|  | Class Name: IConstraintChecks | | Id: 7 | | Type: Interface | |  |
|  | Description: the interface class that has check functions | | | | Aggregation Use cases:- 0  Associated Use cases:- 0  Inheritance Use cases:- 0  Composition Use cases:- 0 | |  |
|  | Responsibilities  checkGradeConstraints  checkAllIsNumbers  checkAllIsNotNumbers  MinMaxLengthChecks  checkFirstNameConstraints  checkMiddleNameConstraints  checkLastNameConstraints  checkPhoneNumberConstraints  checkEmailConstraints  checkPahseNameconstraints  checkPasswordConstraints  checkNameConstraint | | | Collaborators | | |  |
|  |  |  |  |  |  |  |  |
| Back | |  |  |  |  |  |  |
|  | Attributes | | | | | |  |
|  |  | | |  | | |  |
|  |  | | |  | | |  |
|  |  | | |  | | |  |
|  |  | | |  | | |  |
|  |  | | |  | | |  |
|  |  | | |  | | |  |
|  | Relationships | | |  | | |  |
|  | Generalization (A Kind Of): | | |  | | |  |
|  | Aggregation (Part-of): | | | -- | | |  |
|  | Composition: | | | -- | | |  |
|  | Other Association: | | |  | | |  |
|  |  |  |  |  |  |  |  |

|  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- |
| Front | |  |  |  |  |  |  |
|  | Class Name: CConstraintChecks | | Id: 8 | | Type: concrete | |  |
|  | Description: the constraint class that has check functions | | | | Aggregation Use cases:- 0  Associated Use cases:- 1  Inheritance Use cases:- 1  Composition Use cases:- 0 | |  |
|  | Responsibilities  \_\_construct  removeSingleQuotes  checkGradeConstraints  checkAllIsNumbers  checkAllIsNotNumbers  MinMaxLengthChecks  checkFirstNameConstraints  checkMiddleNameConstraints  checkLastNameConstraints  checkPhoneNumberConstraints  checkEmailConstraints  checkPahseNameconstraints  checkPasswordConstraints  checkNameConstraint | | | Collaborators  database | | |  |
|  |  |  |  |  |  |  |  |
| Back | |  |  |  |  |  |  |
|  | Attributes | | | | | |  |
|  | Database database | | |  | | |  |
|  |  | | |  | | |  |
|  |  | | |  | | |  |
|  |  | | |  | | |  |
|  |  | | |  | | |  |
|  |  | | |  | | |  |
|  | Relationships | | |  | | |  |
|  | Generalization (A Kind Of): | | |  | | |  |
|  | Aggregation (Part-of): | | | -- | | |  |
|  | Composition: | | | -- | | |  |
|  | Other Association: | | | CDatabase | | |  |
|  |  |  |  |  |  |  |  |

|  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- |
| Front | |  |  |  |  |  |  |
|  | Class Name: CDatabase | | Id: 9 | | Type: Concrete | |  |
|  | Description: the database operations class | | | | Aggregation Use cases:- 0  Associated Use cases:- 4  Inheritance Use cases:- 0  Composition Use cases:- 0 | |  |
|  | Responsibilities  changeDatabaseConnection  getConnection  \_\_construct  Query  Insert  selectDatabase  selectDefaultDatabase | | | Collaborators  hostName  databasePassword  databaseName  databaseConnection | | |  |
|  |  |  |  |  |  |  |  |
| Back | |  |  |  |  |  |  |
|  | Attributes | | | | | |  |
|  | String hostName | | |  | | |  |
|  | String databasePassword | | |  | | |  |
|  | String databaseName | | |  | | |  |
|  | databaseConnection | | |  | | |  |
|  |  | | |  | | |  |
|  |  | | |  | | |  |
|  | Relationships | | |  | | |  |
|  | Generalization (A Kind Of): | | |  | | |  |
|  | Aggregation (Part-of): | | | -- | | |  |
|  | Composition: | | | -- | | |  |
|  | Other Association: | | | Html,css,user,CConstraintCheck | | |  |
|  |  |  |  |  |  |  |  |

|  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- |
| Front | |  |  |  |  |  |  |
|  | **Class Name**: User | | **Id:** 1 | | **Type:** Superclass | |  |
|  | **Description:** Class includes all system’s users’ profile and data needed regardless to their types | | | | Aggregation Use cases:- 0  Associated Use cases:- 3  Inheritance Use cases:- 0  Composition Use cases:- 0 | |  |
|  | **Responsibilities**  Login(ID, Password)  Logout()  ForgetPassword()  \_construct()  CheckUserType(ID)  queryGetFirstName(ID)  queryGetThesisIdea(ID)  queryGetEmail(ID)  hash(password)  checkPasswordTrueOrFalse(Password,hashedPassword) | | | **Collaborators**  Chash  Cencrypt  CDatabase | | |  |
|  |  |  |  |  |  |  |  |
| **Back** | |  |  |  |  |  |  |
|  | **Attributes** | | | | | |  |
|  | ID[int] | | | Password[string] | | |  |
|  | hashPassword[string] | | | UserType[int] | | |  |
|  | FullName[string] | | | Email[string] | | |  |
|  | Tele[int] | | | DepID[int] | | |  |
|  |  | | |  | | |  |
|  |  | | |  | | |  |
|  | **Relationships** | | |  | | |  |
|  | **Generalization (A Kind Of):** | | | -- | | |  |
|  | **Aggregation (Part-of):** | | | -- | | |  |
|  | **Composition:** | | | -- | | |  |
|  | **Other Association:** | | | Chash  Cencrypt  CDatabase | | |  |
|  |  |  |  |  |  |  |  |

|  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- |
| Front | |  |  |  |  |  |  |
|  | **Class Name**: Student | | **Id:** 2 | | **Type:** Subclass | |  |
|  | **Description:** A classThat specifies a type of user who are categorized as student in the university who will interact with the system. | | | | Aggregation Use cases:- 0  Associated Use cases:-1  Inheritance Use cases:- 1  Composition Use cases:- 0 | |  |
|  | **Responsibilities**  SubmitRegistration(registration R)  EditRegistration(registration R)  DeleteRegistration(registration R) | | | **Collaborators**  User  Registration | | |  |
|  |  |  |  |  |  |  |  |
| **Back** | |  |  |  |  |  |  |
|  | **Attributes** | | | | | |  |
|  | SemesterName[string] | | | StudentCreditHours[int] | | |  |
|  | Registration registration[] | | |  | | |  |
|  |  | | |  | | |  |
|  |  | | |  | | |  |
|  |  | | |  | | |  |
|  |  | | |  | | |  |
|  | **Relationships** | | |  | | |  |
|  | **Generalization (A Kind Of):** | | | User | | |  |
|  | **Aggregation (Part-of):** | | | -- | | |  |
|  | **Composition:** | | | -- | | |  |
|  | **Other Association:** | | | Registration | | |  |
|  |  |  |  |  |  |  |  |

|  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- |
| Front | |  |  |  |  |  |  |
|  | **Class Name**: Doctor | | **Id:** 3 | | **Type:** Subclass | |  |
|  | **Description:** A class That specifies a type of user who are categorized as doctors in the university who will interact with the system. | | | | Aggregation Use cases:- 1  Associated Use cases:-0  Inheritance Use cases:- 1  Composition Use cases:- 0 | |  |
|  | **Responsibilities**  SubmitGrade(Thesis T)  editGrade(Thesis T)  deleteGrade(Thesis T)  SubmitComment(Thesis T)  editComment(Thesis T)  deleteComment(Thesis T)  ViewThesis(Thesis T)  DownloadThesis(Thesis T)  ViewStudents Thesis(Thesis T)  querySemesterName(semesterID,db)  queryStudentsRegistrations(databsae)  acceptStudent(studentId,db)  rejectStudent(studentId,db)  onSearchRegistrationRowClick() | | | **Collaborators**  User  Thesis | | |  |
|  |  |  |  |  |  |  |  |
| **Back** | |  |  |  |  |  |  |
|  | **Attributes** | | | | | |  |
|  | Thesis thesis[] | | |  | | |  |
|  |  | | |  | | |  |
|  |  | | |  | | |  |
|  |  | | |  | | |  |
|  |  | | |  | | |  |
|  |  | | |  | | |  |
|  | **Relationships** | | |  | | |  |
|  | **Generalization (A Kind Of):** | | | User | | |  |
|  | **Aggregation (Part-of):** | | | Thesis | | |  |
|  | **Composition:** | | | -- | | |  |
|  | **Other Association:** | | | -- | | |  |
|  |  |  |  |  |  |  |  |

|  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- |
| Front | |  |  |  |  |  |  |
|  | **Class Name**: Thesis | | **Id:** 4 | | **Type:** Concrete | |  |
|  | **Description:** A class That includes the thesis of each student so that it can be viewed and checked by the dodtors. | | | | Aggregation Use cases:- 0  Associated Use cases:- 0  Inheritance Use cases:- 0  Composition Use cases:- 0 | |  |
|  | **Responsibilities**  SubmitThesis()  DeleteThesis()  EditThesis() | | | **Collaborators**  -- | | |  |
|  |  |  |  |  |  |  |  |
| **Back** | |  |  |  |  |  |  |
|  | **Attributes** | | | | | |  |
|  | ThesisID[int] | | | ThesisDescripition[string] | | |  |
|  | OrderVlaue[int] | | | Grade[int] | | |  |
|  | LinkToTemplate[string] | | |  | | |  |
|  |  | | |  | | |  |
|  |  | | |  | | |  |
|  |  | | |  | | |  |
|  | **Relationships** | | |  | | |  |
|  | **Generalization (A Kind Of):** | | | -- | | |  |
|  | **Aggregation (Part-of):** | | | -- | | |  |
|  | **Composition:** | | | -- | | |  |
|  | **Other Association:** | | | -- | | |  |
|  |  |  |  |  |  |  |  |

|  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| Front | |  |  |  | |  |  | |  | |
|  | **Class Name**: Registration | | **Id:** 5 | | **Type:** Concrete | | |  | |
|  | **Description:** A class that includes all the information about the registered student. | | | | Aggregation Use cases:- 1  Associated Use cases:- 1  Inheritance Use cases:- 0  Composition Use cases:- 0 | | |  | |
|  | **Responsibilities**  CheckRegistrationResults()  queryAcceptanceStatus(ID,database DB)  queryGetTimeStamp(ID,database DB)  querySemesterNameByStudentId(studentID)  queryStudentID(database DB, ID)  queryStudentRegistrationID(studentID, database DB)  queryRegistrationSemesterID(ID, database DB)  querySemesterName(semesterID, database DB)  querySemesterID(database DB) | | | **Collaborators**  RegistrationDetails  CDatabase | | | |  | |
|  |  |  |  |  | |  |  | |  | |
| **Back** | |  |  |  | |  |  | |  | |
|  | **Attributes** | | | | | | |  | |
|  | semesterID[int] | | | ID[int] | | | |  | |
|  | studentID[int] | | | creditHours[float] | | | |  | |
|  | email[string] | | | acceptanceStatus[string] | | | |  | |
|  | Database database[] | | |  | | | |  | |
|  |  | | |  | | | |  | |
|  |  | | |  | | | |  | |
|  | **Relationships** | | |  | | | |  | |
|  | **Generalization (A Kind Of):** | | | -- | | | |  | |
|  | **Aggregation (Part-of):** | | | CDatabase | | | |  | |
|  | **Composition:** | | | -- | | | |  | |
|  | **Other Association:** | | | RegistrationDetails | | | |  | |
|  |  |  |  |  | |  |  | |  | |

|  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| Front | |  |  |  | |  |  | |  | |
|  | **Class Name**: RegistrationDetails | | **Id:** 6 | | **Type:** Concrete | | |  | |
|  | **Description:** A class that includes all the details about the registration itself. | | | | Aggregation Use cases:- 0  Associated Use cases:- 1  Inheritance Use cases:- 0  Composition Use cases:- 0 | | |  | |
|  | **Responsibilities**  DownloadRegistrationDetails() | | | **Collaborators**  Thesis | | | |  | |
|  |  |  |  |  | |  |  | |  | |
| **Back** | |  |  |  | |  |  | |  | |
|  | **Attributes** | | | | | | |  | |
|  | Grade[int] | | | Thesis thesis | | | |  | |
|  |  | | |  | | | |  | |
|  |  | | |  | | | |  | |
|  |  | | |  | | | |  | |
|  |  | | |  | | | |  | |
|  |  | | |  | | | |  | |
|  | **Relationships** | | |  | | | |  | |
|  | **Generalization (A Kind Of):** | | | -- | | | |  | |
|  | **Aggregation (Part-of):** | | | -- | | | |  | |
|  | **Composition:** | | | -- | | | |  | |
|  | **Other Association:** | | | Thesis | | | |  | |
|  |  |  |  |  | |  |  | |  | |

|  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| Front | |  |  |  | |  |  | |  | |
|  | **Class Name**: RegistrationDetailsStatus | | **Id:** 7 | | **Type:** Concrete | | |  | |
|  | **Description:** A class that shows the status of the Registartion details. | | | | Aggregation Use cases:- 0  Associated Use cases:- 1  Inheritance Use cases:- 0  Composition Use cases:- 0 | | |  | |
|  | **Responsibilities**  SubmitStatus(ID, thesis T)  EditStatus(ID, thesis T)  DeleteStatus(ID, thesis T) | | | **Collaborators**  RegistrationDetails | | | |  | |
|  |  |  |  |  | |  |  | |  | |
| **Back** | |  |  |  | |  |  | |  | |
|  | **Attributes** | | | | | | |  | |
|  | ID[int] | | | RegDetailID[int] | | | |  | |
|  | StatusID[int] | | | date DateTime | | | |  | |
|  | Feedback[string] | | | Name[string] | | | |  | |
|  | PowerValue[int] | | | Thesis thesis | | | |  | |
|  |  | | |  | | | |  | |
|  |  | | |  | | | |  | |
|  | **Relationships** | | |  | | | |  | |
|  | **Generalization (A Kind Of):** | | | -- | | | |  | |
|  | **Aggregation (Part-of):** | | |  | | | |  | |
|  | **Composition:** | | | -- | | | |  | |
|  | **Other Association:** | | | RegistrationDetails | | | |  | |
|  |  |  |  |  | |  |  | |  | |

**8.2.6 Attributes:**

Lists each attribute (state variable) associated with each instance of this class, a nd indicates examples of possible values (or a range).

**1-User:-**  
-int ID  
-string Password  
-int UserType  
-string FullName  
-string Email  
-int Tele  
-int DepID  
-message messageobj[ ]

**2-Student:-**  
-int registrationID  
-int nextThesisID  
-string nextThesisName  
-CRegistrationdetails cregistrationdetails []  
-CRegistration cregistration []

**3-RegistrationDetails:-**  
-int registrationDetailsID  
-int registrationID  
-Date registrationDetailTimeStamp  
-int grade  
-int thesisID  
-string thesisName  
-File fileTextMaterial  
-string registrationDetailsStatus

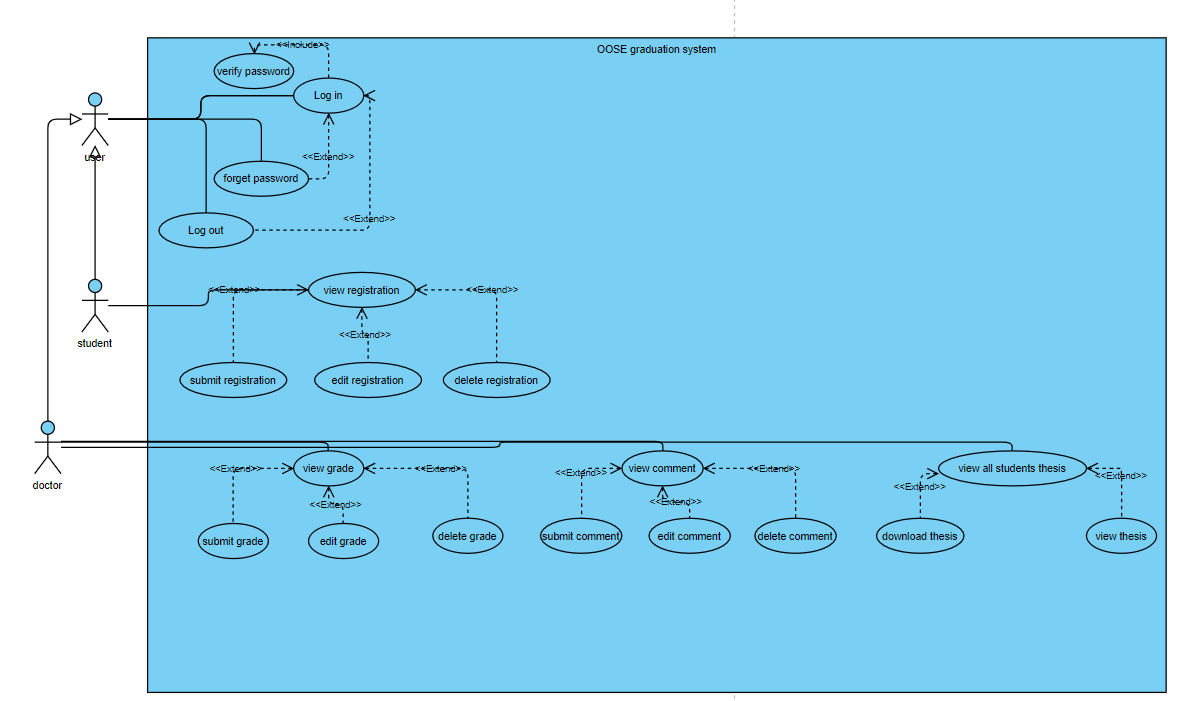
**4-RegestrationDetailsStatus:-**  
-int statusID  
-int registrationDetailsID  
-int doctorID  
-DatestatusGradingTimeStamp  
-string feedback  
-string statusDetails

**5-Registration:-**  
-int semsterID  
-int ID  
-int studentID  
-float creditHours  
-string email  
-string acceptanceStatus  
-Database database[ ]

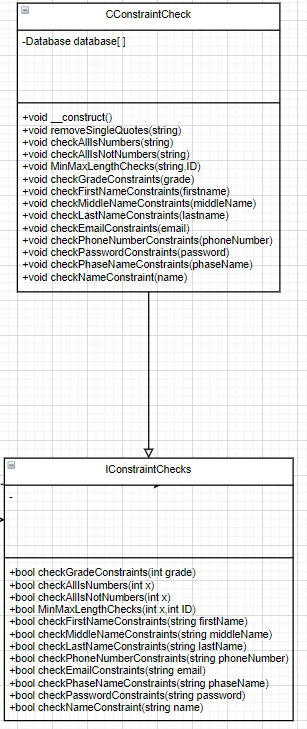
**6-CDatabase:-**  
-string hostName  
-string databasePassword  
-string databaseUserName  
-string databaseName  
-DatabaseConnection

**7-CConstraintCheck:-**  
-Database database[ ]

**8.2.7 Operations**



**8.2.8 Constraints:**



# **9 Operational Scenarios**

**1-Scenario For Adding a Course or an Activity to the System**

**Initial Assumption:** The Student is logged into the system

**Normal:** the student will upload a thesis and wait for it to be graded to move to the next phase

**What Can Go Wrong:** the student uploads wrong thesis file and want to adjust it

**Other Activities:** -

**System State On Completion**: the studen’t submission is uploaded adnd will be graded by a doctor or a group of doctors according to the grading strategy set by the admin and will decide whether he passes or fails this phase.

# **10 Preliminary Schedule Adjusted**

NONE

**11 Preliminary Budget Adjusted**

# **None**

# **12 Appendices**

Specifies other useful information for understanding the requirements. All SRS documents should include at least the following two appendices:

**12.1 Definitions, Acronyms, Abbreviations**

1- Registration (The student registering for his graduation project)

2-Thesis (The student registering files of the whole graduation phases).

**12.2 Collected material**

NONE.

# **13 References**

**References**

NONE.