**Agile Records**

**Report**

**Team Members:**

**Mike Schultz**

**Justin Sexton**

**Daniel Torres**

**German Villalobos**

**Carlos Moreno**

**Table of Contents**

1. [Product Descriptions 2](#_Toc246755790)-3

1.1 [Intro 2](#_Toc246755792)-3

1.2 Student Functionality [2](#_Toc246755792)

1.3 [Administrator Functionality 2](#_Toc246755792)

2. Functional Requirements 4

2.1 [User Login System 2](#_Toc246755792)-3

2.2 Student Information to be stored/calculated [2](#_Toc246755792)

2.3 [Student Course Information to be stored/calculated 2](#_Toc246755792)-3

2.4 Updating student and student course information [2](#_Toc246755792)

2.3 [Viewing Student information 2](#_Toc246755792)-3

3. [Non-Functional Requirements](#_Toc246755793) 5

3.1 [Cost Constraints 5](#_Toc246755802)

3.2 [Reliability 5](#_Toc246755803)

3.3 [Time Constraints 5](#_Toc246755805)

4. [Use Cases 6-8](#_Toc246755793)

4.1 [Use case descriptions 5](#_Toc246755803)

4.2 [Use Case Sequence Diagrams 10-14](#_Toc246755805)

5. [Technical Processes & Team Organization 9](#_Toc246755801)

5.1 [Process Model 9](#_Toc246755802)

5.2 [Tools, & Techniques 9](#_Toc246755803)

5.3 [Programming Environment 9](#_Toc246755802)

5.4 [Database Environment 9](#_Toc246755803)

5.5 [Version control 9](#_Toc246755802)

6. Class Diagram [8](#_Toc246755801)

7. [Test Cases 9](#_Toc246755801)

# Project Description

* 1. **Intro**

As part of our software engineering course, we will design a Student Information Management System. This system will keep track of student details and allow administrators to edit these details. The system will have two modes of usage; administrator mode and user mode. Users will be able to login using their passwords and view information the system has kept track of via database. Upon giving the system credentials, users will be logged in as an administrator or student depending on which account the given credentials belong too.

**1.2 Student Functionality**

If user logs in on student account, they will be sent to the main student page, which will list all relevant student information from the database system.

The top of the page will include:

* Student First Name
* Student Last Name
* Student ID
* Student Unofficial GPA.

The center of the page will display a list containing information of every course the student is currently enrolled in.

This will include information such as:

* Course Name
* Course CRN Number
* Course Date Interval
* Course Time Interval

Each course element within the list will be click-able. Clicking a course element within the list will display another window containing that grades that student has earned within that course.

The bottom of the page will simply list the user-name of the currently logged in user.

**1.3 Administrator Functionality**

If user logs in on administrator account, the application will display the main administrator page, which will contain a list of all students and their information at the center-left of the window along with a student counter at the top of the window.

Information displayed within list will include:

* Student ID
* Student User-name
* Student Last Login Time
* Student GPA
* Student Major

Clicking on one of the listed student elements will display that student’s view to the center-right of the screen. From this view, administrators will be able to edit the student’s information.

In addition, there will be options to add a student, as well as remove a student. When removing students, administrator will select the student elements from the list they want deleted, then simply press the remove button provided. This action will remove all information of the student or students from the database student table. If administrator select the option to add a new student, a create student window will be displayed. This window will contain information fields the administrator can fill in then add information to the database.

The create student window will contain the following data input:

* Student ID
* Student User-name
* Student Last Login Time
* Student Major
* Add/Remove Course

The bottom of the window will simply display the user-name currently logged in.

# Functional Requirements

# 2.1 User Login System

* Login system checks whether user is an administrator or student. If they are not in the database then they will have to be added by the administrator.
* Student login sessions will be directed to student view window, which will supply read only access to that login instance.
* Admin login sessions will be directed to an admin view window, allowing the admin to view and edit every student within the database.

# 2.2 Student Information to be Stored/Calculated

* Student name (First and Last)
* Student username
* Student email
* Student password
* Student ID
* Student courses
* Unofficial GPA

**2.3 Student Course Information to be Stored/Calculated**

* Course Name
* Course GPA
* Course CRN
* Course date interval
* Course time interval
* Course scores.

**2.4 Updating Student and Student Course Information**

* Allow admins to view and manage all the above properties.

**2.5 Viewing Student Information**

* Allow students to only view their information and no other students.

# Non-Functional Requirements

## 3.1 Cost Constraints

* No cost constraints. There is no budget.

## 3.2 Reliability

## 3.3 Time Constraints

1. Unorthodox meeting schedule.
2. University courses and course work.
3. Work schedules.
4. Individual task ambiguity.

# Use Cases

## 4.1 Use Cases: Descriptions

### Use Case 1

|  |  |
| --- | --- |
| Actor(s) | Student, Administrator |
| Goals of actor | To login to application |
| Tasks | Actor must press select option to log in and enter user information. Input is then checked against information in database user table. If information is correct, then user is successfully logged in. |
| Preconditions | User information must be in the database, be they administrator or student. |
| Exceptions | The actor may quit/exit the program. |

### Use Case 2

|  |  |
| --- | --- |
| Goals of actor | To view student information. (Administrator) |
| Tasks | Actor will have a grid view of all students in the system, sorted alphabetically. Actor may choose to filter students by entering criteria in the search box. |
| Preconditions | The program must have launched successfully, user information must be in the database, and user must have logged in. |
| Exceptions | The actor may quit/exit the program. Actor may choose to remove a student from the database or to add a new student into the system. Actor may also choose to update a certain attribute of student. |
| Alternatives | Actor may select on individual student and view their information.  They may also select on a different individual or enter new search criteria. |

### Use Case 3

|  |  |
| --- | --- |
| Goals of actor | Add student |
| Tasks | Actor must press select option to log in and enter user information. Actor must have administrator access. Once logged in they must select the option to add a new student. A new page will open. Each box must be filled in. Once filled in, actor must go through “add course” and “add grade” use cases. Once completed all information will be added to the database system. Records will be added in Student, course, and grades tables. |
| Preconditions | The program must have launched successfully and user must have administrator access. |
| Exceptions | The actor may quit/exit the program. Actor may also to choose to add course and add grade. |

### Use Case 4

|  |  |
| --- | --- |
| Goals of actor | Add course information for student |
| Tasks | Actor must press select option to add a course from the add student page. A new page will open with fill in boxes, each corresponding to course information necessary. Once filled in actor must go through “add grades” use case. Once completed, information will be added to Course and Student tables. |
| Preconditions | The program must have launched successfully. User must have administrator access. Actor must also have to selected option to add a student and filled in student information boxes. |
| Exceptions | The actor may quit/exit the program. Add grades. |

### Use Case 5

|  |  |
| --- | --- |
| Goals of actor | Add grades for student |
| Tasks | Actor must press select option to add grades from “add course” page. Actor must then fill in required information |
| Preconditions | The program must have launched successfully. Actor must have administrator access and have gone through the “add student” and “add course” use cases |
| Exceptions | The actor may quit/exit the program. Actor may also to choose to add course and add grade. |

### Use Case 6

|  |  |
| --- | --- |
| Goals of actor | Remove student |
| Tasks | Actor must select individual student whom they wish to remove and select the option to remove. Once selected, the database will check all records corresponding to that/those student(s) and remove them. Actor will first filter to find the student which they would like to remove from the system. |
| Preconditions | The program must have launched successfully. Actor must have administrator access and student information must be in the database. |
| Exceptions | The actor may quit/exit the program. Actor may choose to cancel delete and or select option to add a student. |

### Use Case 7

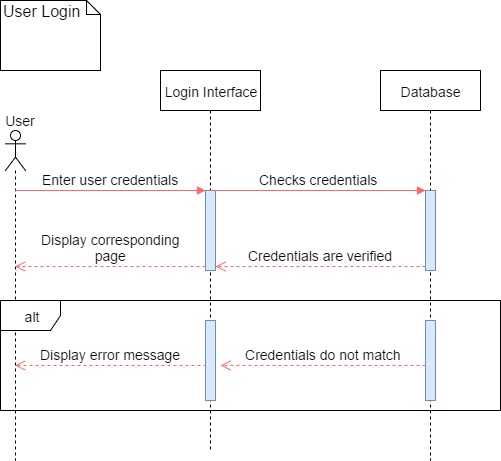
|  |  |
| --- | --- |
| Goals of actor | Update student information |
| Tasks | Actor must press select option to log in and enter user information. Once logged in, if status is administrator then they may choose a student and select option to update student information. A new page will appear where they may change certain information. |
| Preconditions | The program must have launched successfully and user information must be in the database. User must be of administrator status. |
| Exceptions | The actor may quit/exit the program. The actor may choose to go back to previous page by exiting current page. |
| Alternatives | Actor may select on an individual course and choose to change information, add, or even remove a course. In addition they may choose to add, edit, or remove grade information of a course. |

### Use Case 8

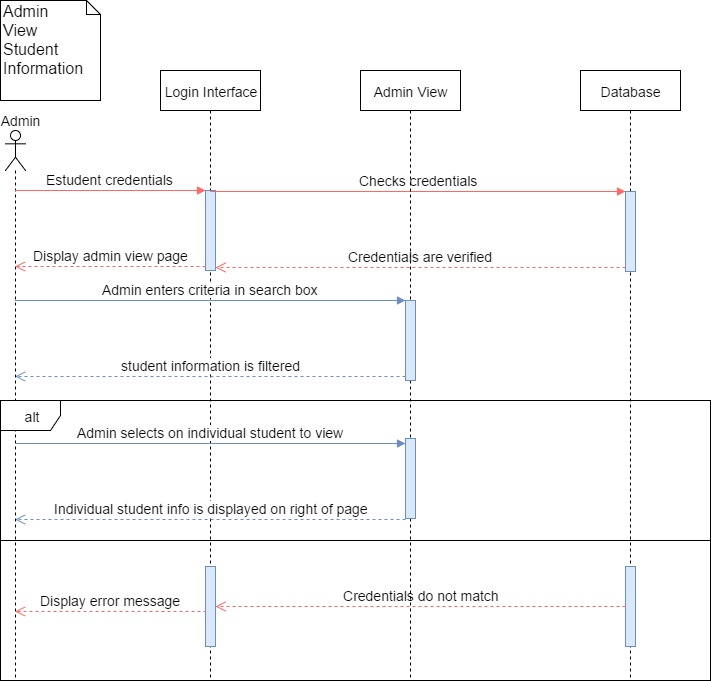
|  |  |
| --- | --- |
| Goals of actor | View Student Information (Student view) |
| Tasks | Actor must press select option to log in and enter user information. Once logged in, if status is student then a window with their course information will appear, containing other information (username, id, gpa, etc.) |
| Preconditions | The program must have launched successfully and user information must be in the database. User must be of student status. |
| Exceptions | The actor may quit/exit the program. The actor may choose to go back to previous page by exiting current page. |
| Alternatives | Student may select individual course to view grade information. |

## 4.2 Use Cases: Descriptions

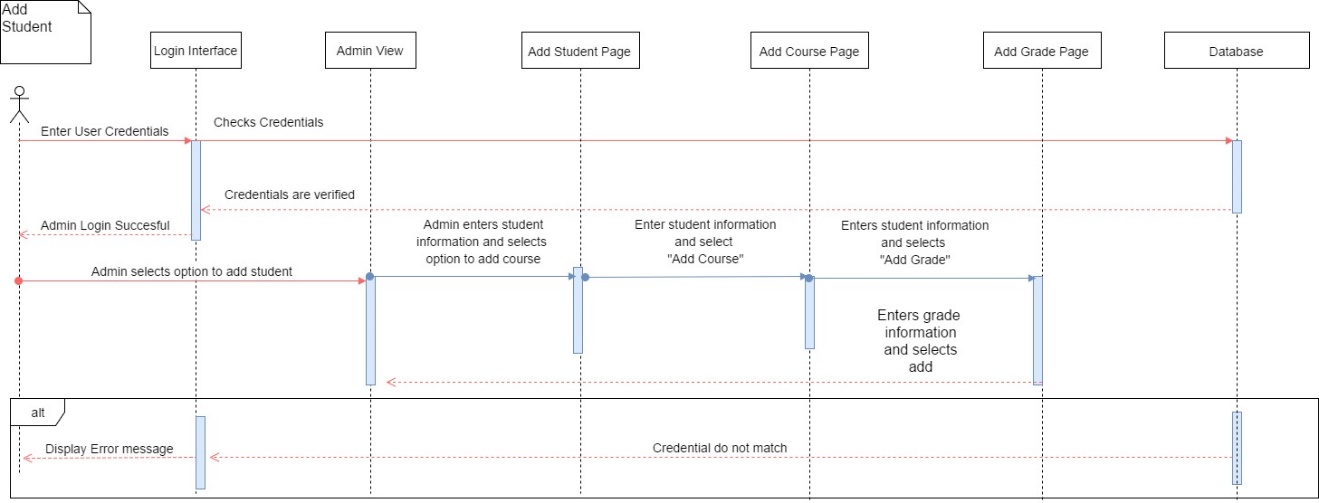
### Actions to be able to login.

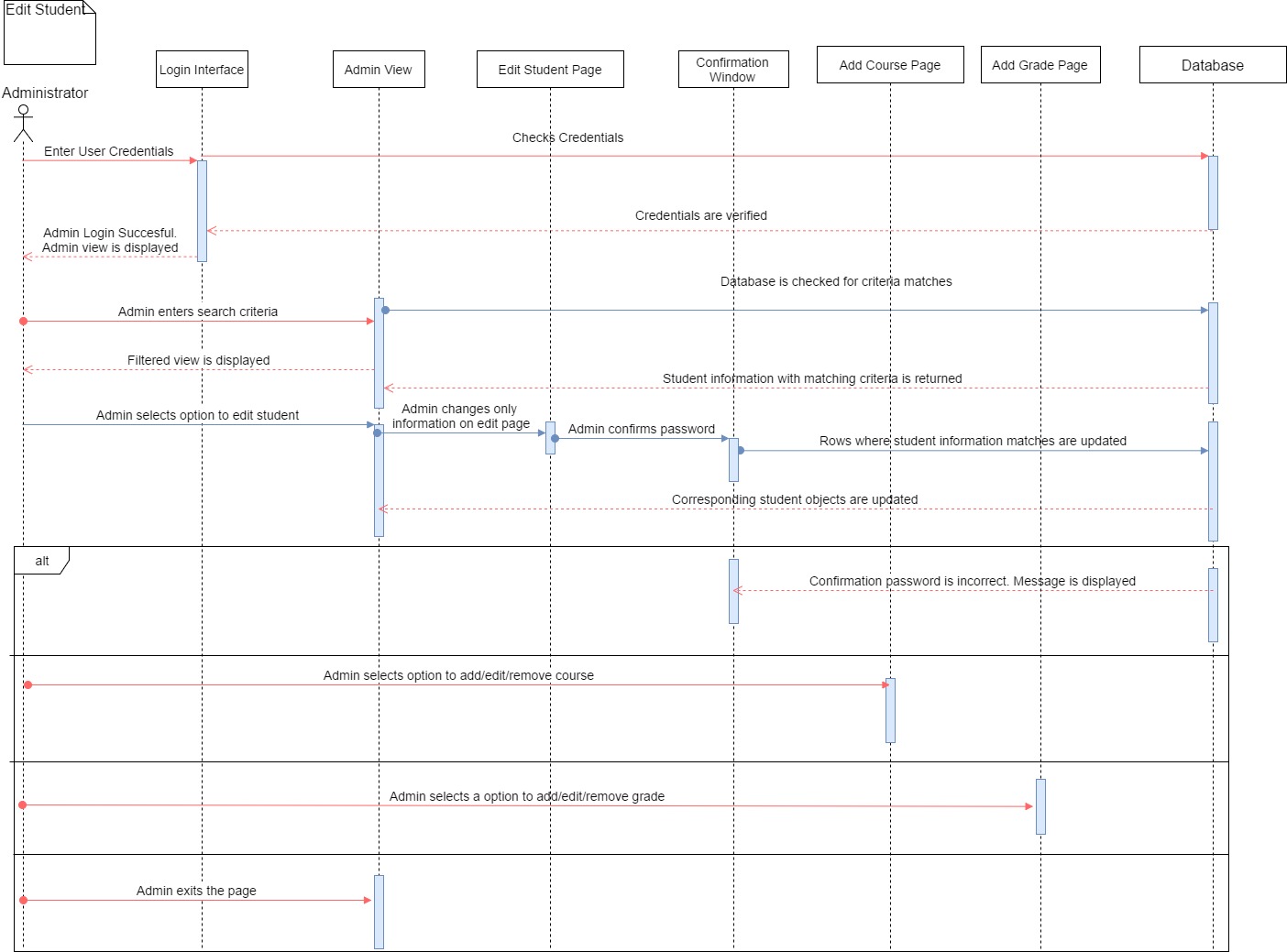


### Actions to be able for Admin to view student information.

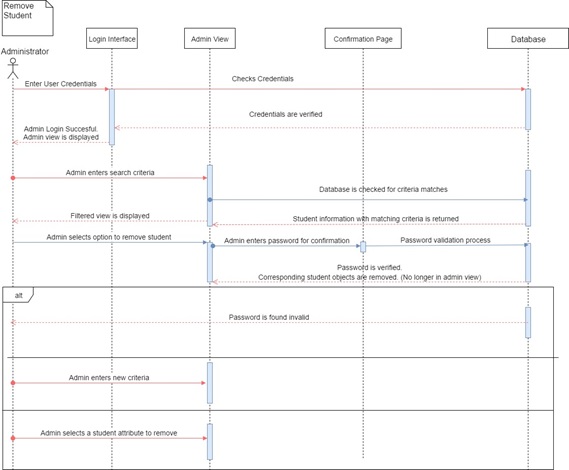


Actions for admin to add a student

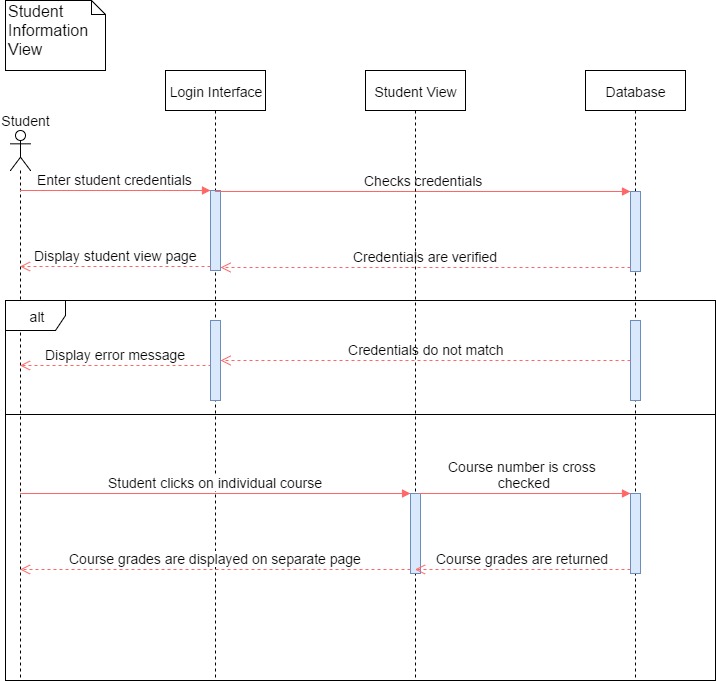


Actions to update/edit student information

Actions to remove student from system



Actions for student to view their information



# Technical Processes & Team Organization

## Process Model

For our project we will be using the prototype and waterfall life-cycle models. In addition we decided to implement a combination of democratic and chief team dynamics. We have assigned the following positions for each team member:

* Project Leader: Justin Sexton
* Database Manager: Mike Schultz
* Secretary: German Villalobos
* Programmer: Daniel Torres, Carlos Moreno, German Villalobos.

## Tools, & Techniques

The workflows will be performed in accordance with the Unified Process. Methods of communication include: Email, schedule meetings, and online app SLACK.

### Programming Environment

### The product will be implemented in Java with the following IDE: IntelliJ.

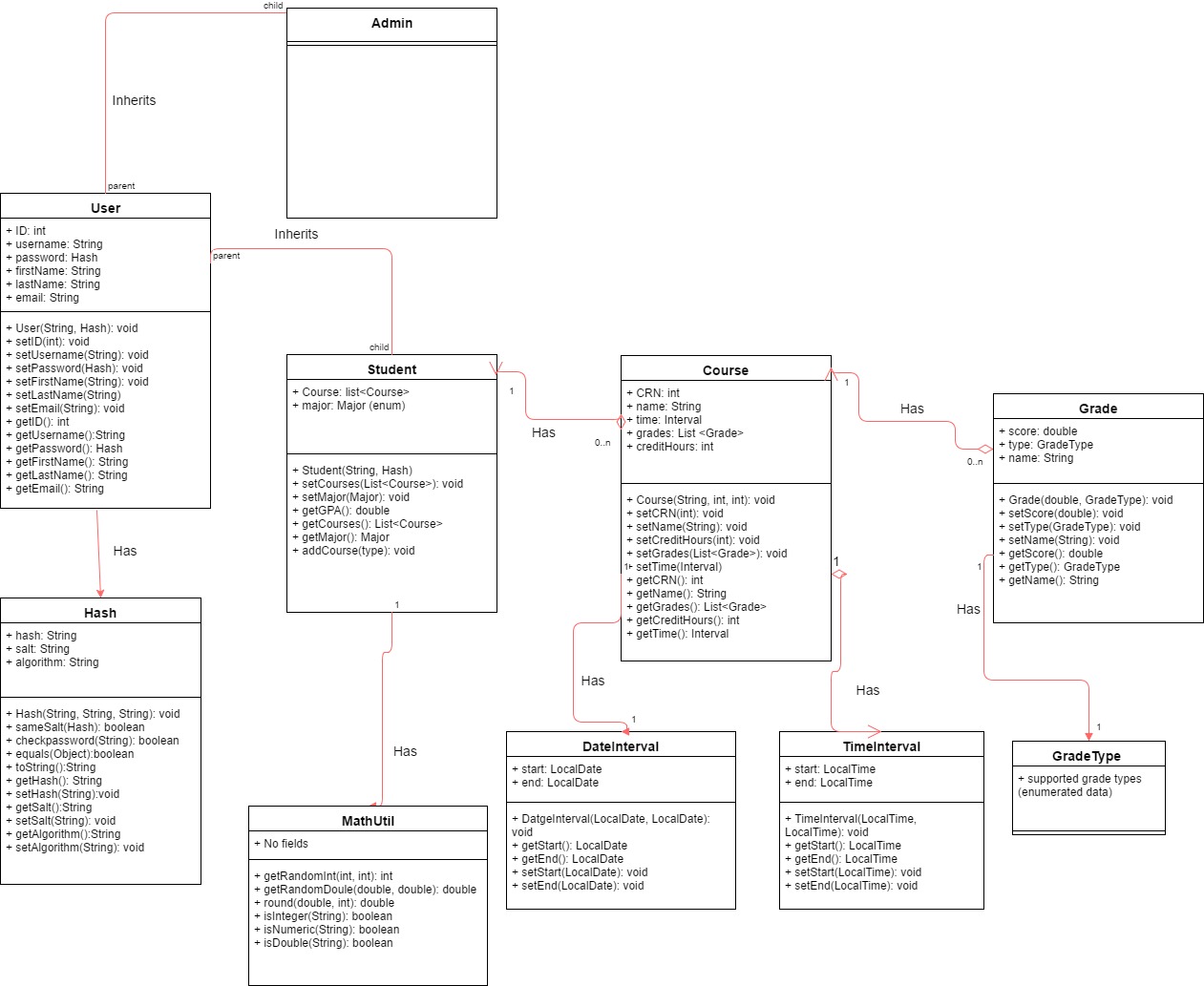
### Database Environment

SQL Workbench – local database.

### Version Control

Windows 7 to Windows 10 and OSX.

# Class Diagram



# Test Cases

**General Tests**

**Login**

Description: Test the application login service.

Pre-Conditions: The system displays the login window.

|  |  |  |
| --- | --- | --- |
| Step | Action | Expected System Response |
| 1 | Press login button after typing credentials | System queries database using given username and password. If credentials are incorrect in anyway, display error message to user, otherwise close login window and display the appropriate window depending on where the credentials came from. |

Post-Conditions: The user will now be able to view their account information.

**Student Tests**

**View Course Grades**

Description: Test whether students can successfully view their course grades.

Pre-Conditions: User must have already logged in. User must have courses to access.

|  |  |  |
| --- | --- | --- |
| Step | Action | Expected System Response |
| 1 | Double click the wanted course in the course table view. | System displays new window with more in-depth course information. |

Post-Condition: Student is now viewing their wanted course’s in-depth information.

**Admin Tests**

**Search for wanted student**

Description: Test whether admin’s can successfully filter their student list by names, major ID.

Pre-Conditions: User must have logged in as admin. There must be students in the database to filter.

|  |  |  |
| --- | --- | --- |
| Step | Action | Expected System Response |
| 1 | Type wanted criteria in search bar above student table. | System filters out all names, majors and ID’s that do not match the criteria out of the student table. |

Post-Conditions: Admin is now able to view only students with the typed in search criteria.

**Add new student**

Description: Test whether admin’s can successfully build and add new student accounts to the database.

Pre-Conditions: User must have logged in as admin.

|  |  |  |
| --- | --- | --- |
| Step | Action | Expected System Response |
| 1 | Click “Add” button below student list. | System displays create student window for admin to fill in required and optional student account parameters. |
| 2 | Click “Create” button after all required fields and any optional fields are filled. | System displays password confirmation window after all given information has been checked for incorrect data types or values. If incorrect values are found, an error message is displayed instead. |
| 3 | Click “Confirm” button after confirmation password has been typed. | If password is correct, system creates student object and stores it within database, otherwise an incorrect password message is displayed giving the user another chance to type their password. |

Post-Condition: New student account is now added to database.

**Remove student**

Description: Test whether admin can successfully remove all selected students from database.

Pre-Conditions: User must have logged in as admin. There must be students in the database to filter.

|  |  |  |
| --- | --- | --- |
| Step | Action | Expected System Response |
| 1 | After selecting all students to be removed from database via the student list, click the “Remove” button. | System displays password confirmation window. |
| 2 | Click “Confirm” button after confirmation password has been typed. | If password is correct, system removes all selected students from database, otherwise an incorrect password message is displayed giving the user another chance to type their password. |

Post-Conditions: Wanted students are now removed from the database.