**Agile Records**

**Report**

**Team Members:**

**Mike Schultz**

**Justin Sexton**

**Daniel Torres**

**German Villalobos**

**Carlos Moreno**

**Table of Contents**

1. [Introduction 2](#_Toc246755789)

1.1 [Product Description 2](#_Toc246755792)

2. [Domain Knowledge 3](#_Toc246755790)

2.1 [Glossary 3](#_Toc246755792)

3. [Class Diagram 4](#_Toc246755793)

4. [Functional Requirements 5](#_Toc246755793)

[4.1 Application Requirements 5](#_Toc246755794)

4.2 [Use Cases 6](#_Toc246755794)-7

[Use Case 1 6](#_Toc246755795)

[Use Case 2 6](#_Toc246755796)

[Use Case 3 7](#_Toc246755797)

...

[Use Case Sequence Diagram 7](#_Toc263067935)

[Use case 1 7](#_Toc263067936)

[General Description 7](#_Toc263067937)

[Sequence Diagram 7](#_Toc263067938)

4.3 [Non-Functional Requirements 8](#_Toc246755801)

[Cost Constraints 8](#_Toc246755802)

[Reliability 8](#_Toc246755803)

[Time Constraints 8](#_Toc246755805)

5. [Process Model & Team Organization 9](#_Toc246755801)

6. Test Cases [10](#_Toc246755801)

[References 8](#_Toc246755801)

# Project Description

As part of our software engineering course we will design a Student Information Management System, which will keep track of student details. The system will have two modes of usage; administrator and user. Users will have to log in using different passwords. The system will Student information will be kept track of through a semester, from day one to the end of their course completion. A user, be they a student or the system administrator will have to first login on a login page. They will input their username and password and the system will compare the information inserted with the information in the database of all users.

If user is a student they will be sent to the main student page will list all relevant student information from the database system. The top of the page will include: student first name, last name, studentID, and their overall current GPA. The center will have a list view containing information of every course the student is currently enrolled in. This will include information such as course name, course CRN, course time, and a clickable box of grades that will allow students to view their course grades. If student selects on grades a new page will open that will allow students to view their scores for the course. There will be an option to exit this page. In addition, there will be a logout/exit option on the student home page.

If user is an administrator their main page will contain a list view of all student information in the center. Information displayed will include X. In addition, there will be option to add a student, as well as remove a student. When removing students, administrator will manually check the boxes to the left of the student and remove all students at once. This action will remove all information of the student in the database tables’ student and grades. If administrator selects the option to add a new student, a new page will open which will have boxes for them to fill in all information. New records will then be added to the database tables student, grades, course, X..

# Domain Knowledge

## Glossary

|  |  |
| --- | --- |
| **Term** | **Definition** |
| Student |  |
| Course |  |
| Register |  |
| GPA |  |
| Grades |  |

|  |  |
| --- | --- |
| **Acronym** | **Meaning** |
| GUI | Graphical User Interphase |
| DB | Database |
| DBMS | Database Management System |
|  |  |
|  |  |

# Class Diagram

# 

# Functional Requirements

## Application requirements

# User login system

# Storage and update of student information;

## Student’s name (first and last)

## Student’s email

## Student’s password

## Student’s Id

## Registered courses in current semester

## Exam’s score in each course

## Current semester GPA

# Storage and update of student course information;

## Course name

## CRN

## Course time

## Course grades

# Project to be programmed in Java

# Other

# Use Cases

### 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. |
| Tasks | Actor will have a grid view of all students in the system, sorted alphabetically. |
| 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. |

### Use Case 3

|  |  |
| --- | --- |
| Goals of actor | Add student into system |
| 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 b |
| 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 information |
| Tasks | Actor must check all students 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. |
| 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. |
| 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. |

# Non-Functional Requirements

## Cost Constraints

* No cost constraints. There is no budget.

## Reliability

## Time Constraints

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

# Process Model & Team Organization

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 Sexxton
* Database Manager: Mike Schultz
* Secretary: German Villalobos
* Programmer: Daniel Torres, Carlos Moreno, German Villalobos