James Diamond 500657801
Marcel-Pierre Samuels 500617694

CPS 731 - Software Engineering I
Project - Designing an Online Registration System

Phase 1 - Requirements gathering

Gathering Process:

Given the scope of this project, the following was done to aid in the requirements gathering stage.

Firstly, the current system at Ryerson was observed. We went through RAMSS and made a note of what we liked and didn't like, as well as what basic operations we liked and would use for our own system. What we found from observing RAMSS was that overall it is a fine system but it can be very slow. We have made a requirement for our system to have a maximum of 10 seconds per transaction. As the system scales we want to avoid a slow user experience, which can lead to frustration and avoidance of the system.

Some of the features in RAMSS we did include in our own system, which can be seen from the use cases but have been organized differently. An example of this is the student user "View/Edit Account Details" where in here the student is able to easily see anything that pertains to specifically them and their account in our system and have the options, where applicable, to make changes. We felt that the student experience should be a very simple and painless one and this was made a high priority.

RAMSS does have a simple user interface, which we liked, and gave a decent idea of the minimal functional requirements we would need. To aid us farther in the requirements gathering we spoke with some students and an office member. They were asked a few simple questions about the current system, such as what they liked and didn't like, which we then took and wrote what we thought would make for a simple and efficient system of our own. A few TA's were also consulted to get more of a faculty perspective. Not specifically with RAMSS but with D2L as well. It was important to get information from a back-end user to make sure we did not miss any of the functional requirements for the three main account types.

The rest of the requirements gathering were very straight forward. Once we had an idea of what functional requirements each user type needed to get a functional system

operating, the non-functional requirements seemed rather obvious. From the people we spoke to, as well as our own experience in RAMSS, we ranked these non-functional requirements to make an effective system. Some features such as security and cost are very important, but something like server up-time we wanted to improve upon RAMSS and aim for the gold standard at 99.999% uptime. With a system such as this, with the many users it has across many different schedules we did not want to limit people from accessing the system if we did not have to.

Lastly, we needed a system that was scalable and modifiable. Starting from a few users to up to 4000 simultaneous users and possibly supporting tens of thousands of accounts, the system needs to be able to grow as its user database does. This is very important and can have an impact of the cost, user experience, up-time, and transaction time. A scalable system is key here; this system should only be designed once and not have to re-do anything as the system needs to grow.

Functional:

General

- List available courses
- Web accessibility
- Store system content on server
- public/private listing of courses

3 different users with separate requirements

Registrar

- Check if class can be enrolled in
- Enroll students
- Unenroll students
- Add classes
- Remove classes
- Edit class visibility
- Release Grades
- Create student accounts
    - Login
    - Student ids

Faculty

- Create course
- Remove course
- Set class size limit
- Edit class information

        - I.e. course description, cost of course, number of students
    - Post grades
    - Login/Logout

Students

- Enroll
- Drop
- Swap
- Shopping cart
- View enrolled courses (current schedule)
- View account details
    - See student number, change password, see fees due, see enrolled courses list view, see course grades
- login/out

Non-Functional:

- Support 4000 users simultaneously
- GUI login screen with password protection
- Scaleable for many students
- Reasonable Costs for the project
- Ease of use (Easy to navigate and enroll)
    - Mobile devices
- Security
    - Each user group has distinct permissions
    - Ensuring course content is kept private to each user
    - User ID is kept personal
    - Cost cost
- Performance
    - Able to handle the load of multiple process
    - 10s should be a max time for transactions
        - I.e. enrolling, dropping, logging in, etc
- Availability (Access should be 24/7)
    - Aiming for 99.999% uptime
    - Maintenance time, middle of the night during not peak hours
- Scalable for many more than 4000 students in the system
    - Just 4000 simultaneous
- Modifiability

- Increasing the number of users able to access

Priority

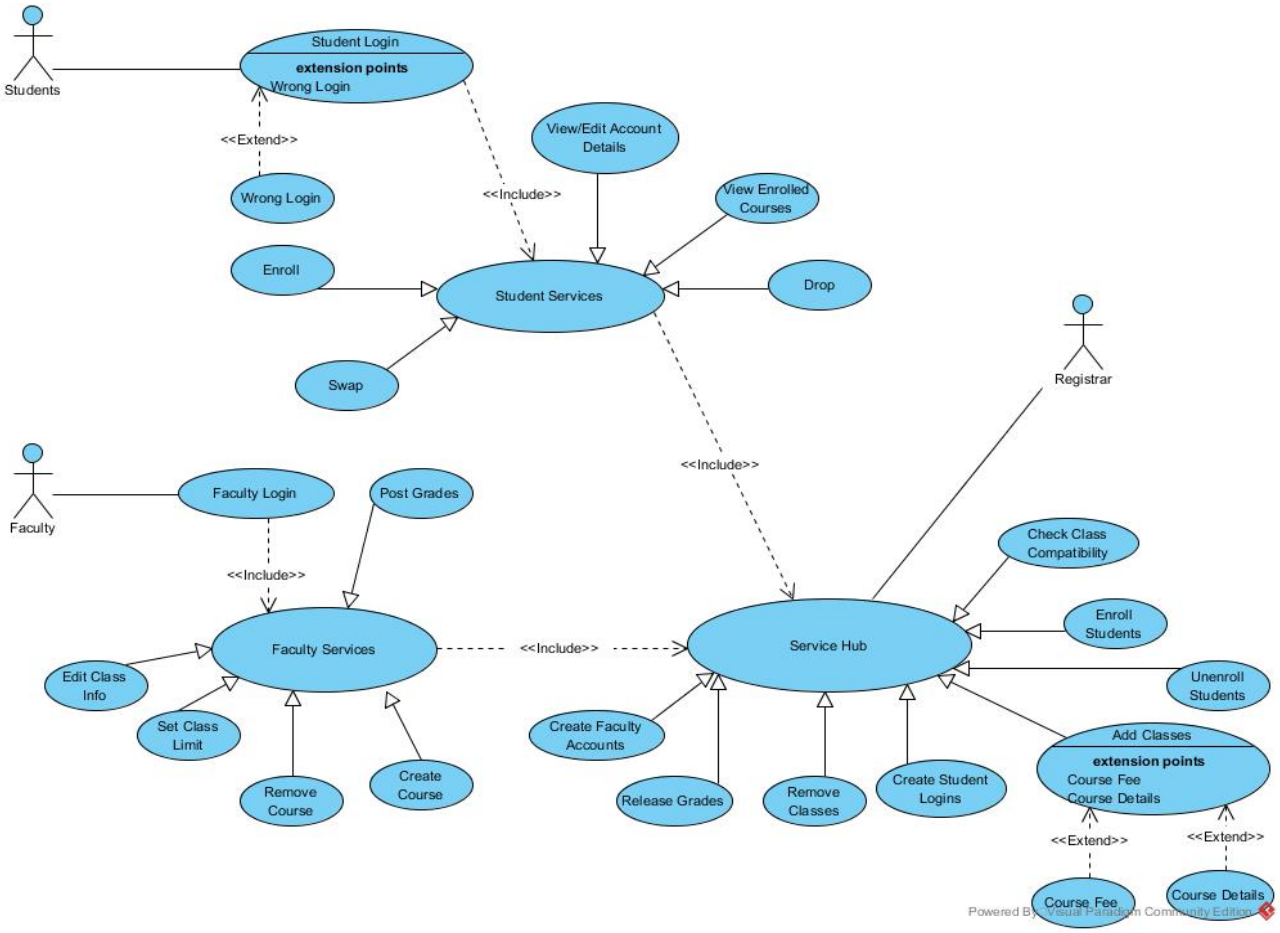
Functional

High Priority	Medium Priority	Low Priority
List available courses	public/private listing of courses	Edit class visibility
Web accessibility	View enrolled courses	Edit class specifics
Store System content on server	View account details	Post grades
Enroll, Drop, Swap	Set class size limit	Release Grades
Shopping Cart	Create Course	
Login/Logout	Remove Course	
Enroll students		
Unenroll students		
Add classes		
Remove classes		
Check if class can be enrolled in		
Create Student accounts		

Non-Functional

High Priority	Medium Priority	Low Priority
GUI login screen with password protection	10s should be a max time for transactions	
Support 4000 users simultaneously	Increasing the number of users able to access	
Scaleable for many students		
Ease of use		
Ensuring course content is kept private to each user		
User ID is kept personal		
Each user group has distinct permissions		
Able to handle the load of multiple process		
Enrolment Costs		
Availability (Access should be 24/7)		
Project Cost		

UC Diagram



Use Cases

Use Case Name: Student Login

Brief Description:

- To initiate the access to one's student services account.

Primary Actors

- Students

Secondary Actors

- None

Preconditions:

1. Student must have a Ryerson Student Login

Main flow:

1. Prompt to enter student username and password
2. If Student enters the wrong login
   2.1. Extend (Wrong Login)
3. Student successfully enters login to Online Registration System
   3.1. include(Student Services)

Postconditions:

1. Login succesfully entered

Alternative flows:

1. None

Extension Use Case: Wrong Login

Brief Description:

- Student provides an invalid Login when attempting to verify their identity.

Primary Actors:

- Student

Secondary Actors:

- None

Segment 1 Preconditions:

1. The user tried their student login.
2. The user provided incorrect login information.

Segment 1 Main flow:

1. The system requests the user to attempt to correctly verify their identity once again.
2. The system checks the new login information provided by the student.
3. If the login information provided is correct
   3.1 The Online Registration System proceeds to the Student Services Use Case
4. Else
   4.1 The Registration System restarts this use case, requesting the user to enter a valid login once again.

Segment 1 Postconditions:

1. The new login information provided by the student is correct.

Alternative flows:

- None

Include Use Case Name: Student Services

Brief Description:

- Student Service hub for all Student services accessible by use case Student

Primary Actors

- Student

Secondary Actors

- None

Preconditions:

1. Student Login successfully entered.

Main flow:

1. Prompt screen so the student can view all the Student Service options
2. When student services are accessed all information is updated and acquired through Service Hub.
   2.1. Include (Service Hub)

Postconditions:

1. Update Student information
2. Update Service Hub

Alternative flows:

- None

Use Case Name: Enroll

Brief Description:

- All the student to enroll into courses of their choice

Primary Actors

- Students

Secondary Actors

- None

Preconditions:

1. Student successfully logged in and Student Services promoted all options.

Main flow:

1. Student Clicks the enroll option
2. The system prompt a search where the student can input and find the courses of their liking and add it to their schedule

Postconditions:

1. Course is found
2. Course is successfully added to student schedule.

Alternative flows:

- None

Use Case Name: View/Edit Account Details

Brief Description:

- Student can view and edit all details pertaining to their account

Primary Actors

- Student

Secondary Actors

- None

Preconditions:

1. Student successfully logged in and Student Services promoted all options.

Main flow:

1. Student selects View/Edit Account Details options
2. The Registration System shows details such as Student username, Password change, Student email, Student number, Student Fees

Postconditions:

1. The requested details are showed on the screen or edited.

Alternative flows:

- **None**

Use Case Name: Swap

Brief Description:

- Allows the Student to swap one currently enrolled course with an non enrolled course

Primary Actors

- Student

Secondary Actors

- None

Preconditions:

1. Student successfully logged in and Student Services promoted all options.
2. Has already enrolled into a course

Main flow:

1. Student selects the swap option
2. Student then enters the new course of their choice and the currently course they wish to change
3. IF time constraints are fine
   3.1. Swap the course
4. ELSE prompt to choose a different course

Postconditions:

1. Changes are successfully made and schedule is updated.

Alternative flows:

- None

Use Case Name: View Enrolled Courses

Brief Description:

- View courses that the Student has enrolled in

Primary Actors

- Student

Secondary Actors

- None

Preconditions:

1. Student successfully logged in and Student Services promoted all options.
2. Has already enrolled into a course

Main flow:

1. Student chooses the View Enrolled Courses option
2. All courses and their details are shown
3. IF there isn't any courses added
   3.1. Prompt student to enroll into courses

Postconditions:

1. All course details are shown

Alternative flows:

- None

Use Case Name: Drop

Brief Description:

- Student can unenroll from courses

Primary Actors

- Students

Secondary Actors

- None

Preconditions:

1. Student successfully logged in and Student Services promoted all options.
2. Has already enrolled into a course.

Main flow:

1. Student choose the Drop option
2. All courses that are currently enrolled are shown and you can drop any choice
3. IF no courses have been enrolled enrolled
   3.1. Prompt student to enroll into courses before dropping a course

Postconditions:

1. Students courses are successfully updated

Alternative flows:

- None

Use Case Name: Faculty Login

Brief Description:

- To initiate the access to one's faculty services account.

Primary Actors

- Faculty

Secondary Actors

- None

Preconditions:

1. Faculty must have a Ryerson Faculty Login

Main flow:

1. Prompt to enter faculty username and password
3. If faculty enters the wrong login
   2.1. Extend (Wrong Login)
3. Faculty successfully enters login to Online Registration System
   3.1. include(Faculty Services)

Postconditions:

2. Login succesfully entered

Alternative flows:

2. None

Include Use Case Name: Faculty Services

Brief Description:

- Faculty Service hub for all Faculty services accessible by use case Faculty

Primary Actors

- Faculty

Secondary Actors

- None

Preconditions:

1. Faculty Login successfully entered.

Main flow:

1. Prompt screen so the faculty can view all the Faculty Service options

When student services are accessed all information is updated and acquired through Service Hub.

2.1. Include (Service Hub)

Postconditions:

1. Update Faculty information
2. Update Service Hub

Alternative flows:

- None

Use Case Name: Post Grades

Brief Description:

- Post grades for the courses in which the Faculty teaches

Primary Actors

- Faculty

Secondary Actors

- None

Preconditions:

1. Successfully logged in Faculty account
2. Must have a course created

Main flow:

1. Faculty selects the Post Grades option
2. Faculty can post grades of each student for the class that they teach
3. IF class is not added
   3.1. Prompt faculty to Create a course

Postconditions:

1. Grade successfully posted

Alternative flows:

- None

Use Case Name: Edit Course Info

Brief Description:

- Edit course information that the Faculty teaches

Primary Actors

- Faculty

Secondary Actors

- None

Preconditions:

1. Successfully logged in Faculty Account
2. Must have a Course created

Main flow:

1. Select the Edit Course Info option

2. Edit the description of the course and its outline
3. IF no course exists
    3.1. Prompt Faculty to Create Course

Postconditions:

1. Course information successfully updated

Alternative flows:

- None

Use Case Name: Insert Class Limit

Brief Description:

-

Primary Actors

- Faculty

Secondary Actors

- None

Preconditions:

1. Successfully logged in Faculty Account
2. Must have a Course created

Main flow:

1. Select the Insert Class Limit option
2. Edit the Class size limit of each Course
3. IF no course exists
    3.1. Prompt Faculty to Create Course

Postconditions:

1. Class size successfully updated

Alternative flows:

- None

Use Case Name: Remove Course

Brief Description:

- Remove a course that the Faculty is teaching

Primary Actors

- Faculty

Secondary Actors

- None

Preconditions:

1. Successfully logged in Faculty Account
2. Must have a Course created

Main flow:

1. Select the Remove course option
2. Remove any course that is listed in which the Faculty teaches
3. IF no course exists
   3.1. Prompt Faculty to Create Course

Postconditions:

1. Courses updated

Alternative flows:

- None

Use Case Name: Create Course

Brief Description:

- Create a course that the Faculty will teach

Primary Actors

- Faculty

Secondary Actors

- None

Preconditions:

1. Successfully logged in Faculty Account

Main flow:

1. Faculty creates a course they would like to teach

Postconditions:

1. Courses updated

Alternative flows:

- None

Include Use Case Name: Service Hub

Brief Description:

- Service hub for all information that Student Services and Faculty Services rely on which is accessible by the Registrar

Primary Actors

- Registrar

Secondary Actors

- None

Preconditions:

1. Registrar accesses Service Hub

Main flow:

1. Prompt screen so the registrar can view all the Service Hub options
2. When Service Hub is accessed all the Student Services and Faculty Services are regulated through hre

Postconditions:

1. Update Service information

Alternative flows:

- None

Use Case Name: Add Classes

Brief Description:

- Add courses into the database so Faculty and Students can find classes to enroll and teach

Primary Actors

- Registrar

Secondary Actors

- None

Preconditions:

1. Registrar accesses service hub

Main flow:

1. Select the Add Classes option
2. Add Classes as well as the Course Fee and Details
   2.1. Extends(Course Fee)
   2.2. Extends(Course Details)

Postconditions:

1. Classes Successfully added

Alternative flows:

- None

Extension Use Case: Course Fee

Brief Description:

- Provides price of the Course

Primary Actors:

- Registrar

Secondary Actors:

- None

Segment 1 Preconditions:

1. Registrar accesses service hub
2. A Class is Created

Segment 1 Main flow:

1. Registrar adds the cost of the Course

Segment 1 Postconditions:

Course Fee successfully updated

Alternative flows:

- None

Extension Use Case: Course Details

Brief Description:

- Update details about a course

Primary Actors:

- Registrar

Secondary Actors:

- None

Segment 1 Preconditions:

1. Registrar accesses service hub
2. A Class is Created

Segment 1 Main flow:

1. Registrar adds details to the courses description

Segment 1 Postconditions:

1. Course Details successfully updated

Alternative flows:

- None

Use Case Name: Create Faculty Accounts

Brief Description:
- Create accounts for Faculty users

Primary Actors
- Registrar

Secondary Actors
- None

Preconditions:
1. Registrar accesses service hub

Main flow:
1. Registrar creates Faculty accounts so the Faculty can login and access information

Postconditions:
1. Account successfully created

Alternative flows:
- None

Use Case Name: Release Grades

Brief Description:
- Release grades so students can view their marks

Primary Actors
- Registrar

Secondary Actors
- None

Preconditions:
1. Registrar accesses service hub
2. Grades must already be posted

Main flow:
1. Registrar releases the grades that have been posted from the Faculty

Postconditions:
1. Grades are posted for Student viewing

Alternative flows:
- None

Use Case Name: Remove Courses

Brief Description:

- Remove courses that were once created

Primary Actors

- Registrar

Secondary Actors

- None

Preconditions:

1. Registrar accesses service hub
2. A Class is Created

Main flow:

1. Remove courses that were once created
2. IF no course is created
   2.1. Prompt Registrar to create courses

Postconditions:

1. Courses have been updated

Alternative flows:

- None

Use Case Name: Create Student Logins

Brief Description:

- Create logins so Student users can access information

Primary Actors

- Registrar

Secondary Actors

- None

Preconditions:

1. Registrar accesses service hub

Main flow:

1. Create an account for student users to enable them to login and access information

Postconditions:

1. Student Login has been created

Alternative flows:

- None

Use Case Name: Enroll Students

Brief Description:

- Enroll Students into the course that they added

Primary Actors

- Registrar

Secondary Actors

- None

Preconditions:

1. Registrar accesses service hub
2. A Course is Created

Main flow:

1. Enroll student into the course of their choosing

Postconditions:

1. Successfully update the course of the student's choosing

Alternative flows:

- None

Use Case Name: Unenroll Students

Brief Description:

- Remove Student from a course that they are enrolled in

Primary Actors

- Registrar

Secondary Actors

- None

Preconditions:

1. Registrar accesses service hub
2. A Class is Created
3. Students must be previously enrolled

Main flow:

1. Remove student from the course that they are currently enrolled in

Postconditions:

1. Update the course in which the student has been removed from

Alternative flows:

- None

Use Case Name: Check Class Compatibility

Brief Description:

- Check if the class the student request is compatible with their time slot

Primary Actors

- Registrar

Secondary Actors

- None

Preconditions:

1. Registrar accesses service hub
2. A Class is Created

Main flow:

1. Check if the time slot of the students requested course fits
2. IF the class fits
   2.1. Enroll student
   2.2 Else
      2.2.1. Revoke request

Postconditions:

1. Update the compatibility of course

Alternative flows:

- None