Simon Fraser University
School of Computing Science

Cmpt 275 Project

Date: 2015-06-20

Team name: Eleven ®

Project Deliverable #: 4

Project Deliverable Name: Release 2 Requirements Specification

Phase(s) covered by this deliverable: Use Cases, Use Case diagram, First draft class diagram

---

Phase leader(s): Nari Shin

Team members and Student #:

Yinglun Qiao	301191776
Janice Sargent	301160485
Zhi Cheng	301184486
Susan Hamilton	301209641
Seong Jun Kim	301154246
Nari Shin	301178863
Fan Liu	301168674
Te Lun Chen	301200832
Roy Chan	301202770

Grade:

Revision History

Revision	Status	Publication/ Revision Date	By:
2.0	Created a skeleton for the Requirement Analysis section	Sunday June 14, *2015*	Fan Liu Yinglun Qiao
2.1	Modified ‘Manage Account’ by adding 1.1.3.1 and all of 1.1.4 2.3	Tuesday June 16, 2015	Zhi Cheng
2.2	Put initial model for use case	Wednesday June 17, 2015	Colin Qiao
2.3	Added list of use cases for rubic which includes create rubric, modify rubric, and enter grades in rubric	Wednesday June 17, 2015	Andy Chen
2.4	Added 1-1.1.1.1.2.1,1-1.1.2.1,1-1.1.3.1.1,1-1.1.4,1-2.1.1,1-3.1.3.1,1-3.2.2, 1-7.10 Edit 1-3.1,1-3.1.1.1.4,1-3.1.5, 1-4.1.3., 1-5.1,1-5.4.1,1-6.1.1, 1-7.4 Deleted 1-2.1.1.2(3,4), 1-2.1.3.2,1-3.1.1.2.2, 1-4.2.1, 1-7.8	*Thursday* June 18, 2015	Susan Hamilton
2.5	Added list of use cases for Managing Accounts (create, modify, unblock and block)	Thursday June 18, 2015	Susan Hamilton
2.6	Added use cases for modify and copying courses	Thursday June 18, 2015	Nari Shin

2.7	Added use cases for grading essays and problem sets	Tuesday June 18, 2015	Roy Chan
2.8	Added use cases for create course and delete course	Thursday June 18, 2015	Janice Sargent
2.9	Added use cases for bonus and penalty, csv files	Thursday June 18, 2015	Fan Liu
3.0	Added use cases for grading multiple choice test and access to system	Thursday June 18, 2015	Oscar Cheng
3.1	Added use cases for managing activities	Friday June 19, 2015	Josh Kim
3.2	Created use case diagram	Saturday June 20, 2015	Oscar Cheng
3.3	Created class diagram	Saturday June 20, 2015	Andy Chen

Table of Contents

Part I

Requirements Specification

1 - Functional Requirements

1. Manage Account Information

1.1. A System Administrator manages all accounts for this system. This includes any or all of the following actions:

1.1.1. Creating an account

1.1.1.1. For each account, the following attributes need to be provided:

1.1.1.1.1. Role(s)

1.1.1.1.1.1. Administrator

1.1.1.1.1.2. System Administrator

1.1.1.1.1.3. Administrative Assistant

1.1.1.1.1.4. Instructor

1.1.1.1.1.5. Teacher Assistant

1.1.1.1.2. Name

1.1.1.1.3. Employee ID

1.1.1.1.4. User ID

1.1.1.1.5. Temporary password

1.1.2. Modifying an account

1.1.2.1. Can modify all areas listed under 1.1.1.1. which includes generating a new temporary password

1.1.2.1.1. Can not leave any field blank

1.1.2.1.2. If they are the only System Administrator they can not modify their System Administrator role

1.1.3. Deleting an account

1.1.3.1. If there exists only one System Administrator in the system and the System Administrator try to delete his or herself account, then the deletion will be forbidden

1.1.4. Blocking an account

1.1.4.1. If a user incorrectly attempts to enter their password 6 times then the account is blocked

1.1.4.2. If they are the only System Administrator they can not block their own account

2. Manage Courses

2.1. An Administrative Assistant manages all courses for this system. This includes any or all of the following actions:

2.1.1. Creating a course

2.1.1.1. For each course, the following attributes need to be provided:

2.1.1.1.1. Course number

2.1.1.1.2. Course name

2.1.1.1.3. Start date

2.1.1.1.4. End date

2.1.1.1.5. Instructor name

2.1.1.1.6. Instructor Employee ID

2.1.1.1.7. Teaching Assistant name(s)

2.1.1.1.7.1. Can have 0 or more

2.1.1.1.8. Teaching Assistant Employee ID(s)

2.1.1.1.9. List of students' names and IDs

2.1.1.1.9.1. By loading a csv file into the system. See 6.1.

2.1.1.2. Markers and students can be added to the course upon the creation or after the creation

2.1.1.3. Can only add the items listed in 2.1.1.1 when creating a course

2.1.2. Modifying a course

2.1.2.1. All attributes in list 2.1.1.1 can be modified

2.1.2.1.1. List of students will be modified by importing a new list

2.1.2.1.2. Only existing TAs have the right to be added as an TA to a course

2.1.3. Deleting a course

2.1.3.1. Courses can be deleted when there are no activities associated with it

2.1.3.2. Courses cannot be deleted once activities has been added

2.1.4. Copying a course

2.1.4.1. Only the attributes of the courses listed under 2.1.1.1. Except the following:

2.1.4.1.1. List of students' names and IDs

2.1.4.1.2. Start date

2.1.4.1.3. End date

2.1.4.1.4. Any materials and activities added by Instructor

2.2. An Administrative Assistant cannot modify any activities or grades for any courses

2.3. The system will use students' number to identify students

3. Manage Activities

3.1. An Instructor can manage the activities to be graded during each of his/her courses. This includes any of the following actions:

3.1.1. Creating an activity

3.1.1.1. The following attributes must be specified:

3.1.1.1.1. Activity Name

3.1.1.1.2. Activity Language: English, French, C++, Java etc.

3.1.1.1.3. Activity Type:

3.1.1.1.3.1. essay

3.1.1.1.3.2. multiple choice

3.1.1.1.3.3. problem set

3.1.1.1.3.4. programming assignment

3.1.1.1.4. Individual or group activity

3.1.1.1.5. Activity Solution or solutions

3.1.1.1.5.1. May have single solution, multiple solution or no solution (mutiple choice has answer keys)

3.1.1.1.5.2. File path is provided by Instructor

3.1.1.2. Additional specifications that can be added and modified later:

3.1.1.2.1. A rubric (mandatory)

3.1.1.2.2. Description of students' submissions (mandatory)

3.1.1.2.3. A path to the directory which contains every student's submission (mandatory)

3.1.1.2.4. Due date (optional)

3.1.1.2.5. Input files:

3.1.1.2.5.1. For essay activities, the input file is the sample essay the insturctor provides

3.1.1.2.5.2. For mutiple choice activities, the input file is a csv file which contain the students' anwser keys for the mutiple choice questions

3.1.1.2.5.3. For programming activity, the input file is one console input, one console output, input, output files, and possibly solution codes

3.1.1.2.6. Late policy

3.1.1.2.7. Bonus marks

3.1.2. Modifying an activity

3.1.2.1. May modify information specified under 3.1.1

3.1.3. Deleting an activity

3.1.4. Copying an acitivty

3.1.4.1. Some or all activities can be copied from the Instructor's previous courses or taught by different Instructors from the same course

3.1.4.2. The following items are not copied: The path to students' submissions, students' grades for each item in the rubric, solutions to essays and problem sets, and due dates

3.2. The system will update the detailed information of each of the activities entered by the Instructor

3.3. The system will release information to the Instructor only relating to his/her course

3.4. Accessing activities

3.4.1. Instructors can only access actitives related to his or her own courses

4. Manage Rubric

4.1. An Instructor manages all rubrics on this system. This includes all of the following actions:

4.1.1. Creating a rubric

4.1.1.1.1. Rubric is created after activity has been created

4.1.2. Modifying a rubric

4.1.3. Copying a rubric

4.1.3.1. Rubric is copied automatically when the assicated activity is copied

4.2. A rubric must have:

4.2.1. Description of items to be graded

4.2.2. Weights for each item

4.3. At any time the Instructor can set or update the rubric

5. Manage Solution and Grades

5.1. The assgined Markers are Teaching Assistants and Instructors, and can do the following:

5.1.1. Can view the work of each student only one at a time

5.1.2. Can add comments directly to submitted work

5.1.2.1. Except for multiple choice activities

5.1.3. Can enter grades for each item in rubric after viewing student's work related to that portion of the rubric

5.1.3.1. Grading programming activities, essays, problem sets, and multiple choice questions are below

5.1.4. Are able to grade or regrade any student's work

5.1.4.1. There is no time limit on how long a marker is able to regrade an assignment

5.2. Grading programming assignments

5.2.1. The system accepts a series of tests to be executed for testing submissions' correctness

5.2.1.1. Each test includes:

5.2.1.1.1. Paths to Input files and console inputs

5.2.1.1.2. Correct output files and console output

5.2.2. The system helps an Instructor assemble input and output files

5.2.3. The system compares correct output and output of student's code

5.2.4. The system should display related parts of rubric to marker

5.2.4.1. A marker will enter a grade for that portion

5.2.5. The system must be able to compile in C, C++, JAVA, and PYTHON

5.2.5.1. The system must be compatible with the environments such as Visual Studio and Eclipse

5.2.5.2. If a code does not compile, the system must notify the marker

5.3. Grading multiple choice tests

5.3.1. An Instructor provides answer keys and a list of points

5.3.2. An Instructor can modify answer keys and weights if necessary

5.4. Grading essays and problem sets

5.4.1. A marker can manually look for a section of a student's work and the reference section from the solution file, and compare it to the corresponding section of the rubric

5.4.1.1. The marker can then give the corresponding mark for this item in the rubric

5.5. Group submission

5.5.1. Each group is identified by their group name.

5.5.2. Every member of the group receives same grade

5.6. Submitting back graded activities

5.6.1. Through the Course Management System

5.7. Plagiarism

5.7.1. Dealt with in the Course Management System

5.8. The system must be able to handle bonuses and deductions

5.8.1. Early Bonuses

5.8.1.1. There are up to three possible early submission dates. Each date is specified by the number of hours early, and the percentage of the grade to be added

5.8.2. Late Policy

5.8.2.1. There are up to three possible late submission dates. Each date is specified by the number of hours late, and the percentage of the grade to be deducted

6. Manage Transferring of Documents

6.1. The system accepts lists of students information through a csv file

6.1.1. One line per student

6.1.2. Each line includes:

6.1.2.1. The student's name

6.1.2.2. The student's computing ID

6.2. The system receives students' submissions from the Course Management System. This includes the following:

6.2.1. A list of students or groups who submit the assignments

6.2.2. The time of submission generated by the Course Management System

6.3. The csv files generated by the system can be loaded back into the Course Management System

6.3.1. System can generate csv files about the grade of each part in the rubric for an activity, and the answers to a mutiple choice test

6.3.2. These files can be generated whenever an instructor requests them

6.4. Commented work on calculations or derivation problems, and essays can be returned to students through the Course Management System

7. Access the System

7.1. An administrator can only goes through the same process as a Marker and update grade

7.2. The system checks the authentication of each user for the login to the system

7.3. The system requires the following identification items to log in a user:

7.3.1. User ID

7.3.2. Password

7.4. The system provides new temporary password for users who fail to log in 6 times

7.4.1. The processes listed from 2.2.1.2.1 - 2.2.1.2.3 are carried out

7.5. The system requires the user to specify the role of the user currently has upon login, and only make user log in as that selected role.

7.6. The system is able to support multiple users

7.7. Instructors are only able to see the list of courses they are assigned to

7.7.1. When a course is selected, they are able to see a list of existing activities, and they can manage or mark those activities

7.8. Logs of the system

7.8.1. Logs contain a list of any changes to the system

7.9. Installation and Backup

7.9.1. Managed by System Administrator

7.9.2. Including backups of

7.9.2.1. Course information

7.9.2.2. Rubric

7.9.2.3. Activities

2 - Non-Functional Requirements

I - Quality Requirements

1. Usability

1. The system provides an interface to help the Instructors to assemble all necessary input and output files for programming assignments
    1. There is no limit on the number of programming tests

    Allows a marker to enter the path to the solution code, sample
    1. inputs, sample outputs, console input, and console outputs
2. If a user has multiple roles, when they have logged onto the system they will have an option to select which role they wish to log on as. Once the user has selected a role, then the user will only log on to the role that the user has chosen.

2. Dependability

1. Reliability
    1. The system keeps a backup of activity solutions in case of the solution link provided by the instructor changes
2. Security
    1. User Logins
        1. User ID
            1. If the user has multiple roles, then the user needs to choose her role to sign in
        2. Password
            1. A System Administrator sends by courier a temporary password for each new account
            2. The user is prompted upon initial login to change to a new password on their own
            3. If the user forgets the password, or fails to login after 6 unsuccessful attempts, the above steps 2.2.1.2.1 and 2.2.1.2.2 are repeated
        3. The employee IDs that are used to uniquely identify each employee should be a 12-digit number

4. The user IDs that are used to uniquely identify each user in the system cannot exceed 8 characters

3. Performance

1. The system grades only one assignment at a time
2. Switching between a student's submitted work, the Instructor's solution, and the activity's rubric should be no more than one click away

4. Supportability

1. Maintainability and Portability
    1. Database Management
        1. Backups occur regularly throughout the term, and formal backups occur at the end of each term
        2. Persistent data will be stored in a MS SQL database. All data will be kept for 5 years
2. Adaptability
    1. Ability to add any directories as a marker wishes.
        1. Eg, create directories to store solutions.

II - Constraints

1. Implementation Requirements

1. The group have agreed to implement the system by using Java in Eclipse as the main programming language
2. The system should run on a Windows platform

2. Interface Requirements

1. Format of csv file
    1. The csv file generated by the system contains one student in each row, and one item in the rubric in each column
    2. For multiple choice tests, each row in the csv will be results for one student. Each column will provide the choices made by the student for each of the questions on the multiple choice test.

3. Operations Requirements

1. Students' codes must be able to compile when marking

        1. Versions of the programming assignment environments should be the same as those in CSIL
    2. Student cannot be graded by multiple markers simultaneously. If two markers are trying to grade the same student at the same time, one of the them will be given a warning and blocked
    3. If a rubric is modified during marking then all students' activities must be remarked
    4. Solution files to activities within a course is provided by instructors via a path

4. Packaging Requirements

    1. Installation will occur initially by System Administrator to set up the system

5. Legal Requirements

    1. The system development must comply with SFU regulations

3 - Prioritization of Functional Requirements

3.1 **LIST 1** (core or critical)

- **Account Management (1)**
    - Account Creation (1.1.1)
    - Account Deletion (1.1.3)
    - Attributes
        - Roles (Administrator, System Admin etc.) (1.1.1.1.1)
        - Name, Employee ID, User ID, Password (1.1.1.1.2 - 1.1.1.1.5)

- **Course Management (2)**
    - Course Creation (2.1.1)
        - Number, Name, Start/End dates (2.1.1.1.1-2.1.1.1.4)
        - Administrative Assistant can assign Instructors, students and Teaching Assistants (2.3)
- **Activity Management (3)**
    - Activity Creation (3.1.1)
        - After course creation
            - Name, Language, Type and Solution (3.1.1.1.1-3.1.1.1.4)
                - Information and Files provided by Instructor (3.1.1.1.4.1)
- **Rubric (4)**
    - Rubric Creation (4.1.1)
    - Directory (4.2.1)
    - Description and weights for items to be graded (4.2.2)
- **Grading (5)**
    - Markers can view student submissions one student at a time (5.1.1)
    - Markers can enter grades for each item in rubric after viewing student submission (5.1.3)
    - Programming Assignments (5.2)
        - Input files and console inputs (5.2.1.1.1)
        - Correct output files and console output (5.2.1.1.2)
        - Help Instructor assemble I/O files (5.2.2)
        - Compare I/O of student’s code submissions (5.2.3)

    - Multiple Choice (5.3)
        - Answer Key, Weights (5.3.1)
- **System (7)**
    - Installation of the system (7.9)
    - Login/Logout (7.2)
    - Identification (7.3)
        - Computer ID and student number (7.3.1)

3.2 **LIST 2** (most important of the remaining)

- **Account Management (1)**
    - Account Modification(s) (1.1.2)
- **Course Management (2)**
    - Course Modification(s) (2.1.2)
    - Attributes that can be added after course has been created
        - Instructor name (2.1.2.1.1)
        - Employee ID (2.1.2.1.2)
        - Teaching Assistant names (2.1.2.1.3)
        - Teaching Assistant employee ID (2.1.2.1.4)
        - List of students' names (2.1.2.1.5)
        - Computer ID of students (2.1.2.1.6)
        - Student number (2.1.2.1.7)
    - Course Deletion (2.1.3)
- **Activity Management (3)**
    - Activity Modifications (3.1.2)
    - Activity Deletion (3.1.3)
    - Attributes that can be added or modified after activity has been created (3.1.1.2)
        - Rubric(3.1.1.2.1)
        - Description (3.1.1.2.2)
        - Due date (3.1.1.2.3)
        - Input file (3.1.1.2.4)
        - Programming tests (3.1.1.2.5)
- **Rubric (4)**
    - Modify rubric (4.1.2)
- **Grading (5)**

    - Add comments to student's work (5.1.2)
    - Multiple Choice (5.3)
        - Modify Answer keys and Weights (5.3.2)
- **Documents (6)**
    - Can be loaded back into course management system (6.2)
      CSV generated when Instructor needs it (6.2.1)
- **System (7)**
    - Supports multiple users (7.5)

3.3 LIST 3 (needed but less important to be functioning of core features of the system)

- **Activity Management (3)**
    - copying from previous courses or activities from the same course (3.2.1)
- **Rubric (4)**
    - copying rubric from previous course offerings (4.1.3)
- **Grading (5)**
    - Ability to regrade student's work (5.1.4)
    - Late Policy (5.6.2)
        - Up to three late submission dates (5.6.2.1)
    - Bonus Marks (5.6)
        - Early bonuses (5.6.1)
            - Up to three possible early submission dates (5.6.1.1)
- **System (7)**
    - Login Failure (7.4)
        - System Administrator sends temporary password (7.4.1)

3.4 LIST 4 (not necessary to the successful and efficient operation of the core system)

- **Grading (5)**
    - Plagiarism (5.7)
        - Instructor may use outside tools (5.7.1)
- **System (7)**
    - Logs (7.7)
        - containing any changes to system (7.7.1)

- Notifications (7.8)
- Backup (7.9)
    - Managed by System Administrator (7.9.1)
    - Including backups of (7.9.2)
        - Course information (7.9.2.1)
        - Rubric (7.9.2.2)
        - Activities (7.9.2.3)

4 - Prioritization of Non-Functional Requirements

x.x : Quality Requirements x.x : Constraints

4.1 LIST 1 (core or critical)

- **Dependability (2)**
  - Security (2.2)
    - User Logins (2.2.1)
      - User ID and password are required (2.2.1.1)
- **Operations Requirements** (3)
  - Students' code can be compiled (3.1)
    - Same versions of programming environment as CSIL (3.1.1)
  - Student cannot be graded simultaneously (3.2)
- **Packaging Requirement** (4)
  - The System Administrator installs and sets up system (4.1)
- **Legal Requirements** (5)
  - The system is subject to SFU regulations (5.1)

4.2 LIST 2 (most important of the remaining)

- **Performance (3)**
  - One assignment is graded at a time (3.1)
- **Dependability (2)**
  - Security (2.2)
    - User Logins (2.2.1)
      - Employee IDs are uniquely identify by a 12-digit number (2.2.1.2)
- **Supportability (4)**
  - Adaptability (4.2)
    - Marker can add directories to store information (4.2.1)
- **Implementation Requirement (1)**
  - The system runs on Windows platform (1.2)
- **Operations Requirements (3)**
  - Activities are remarked if rubric is changed during marking (3.3)
  - Solution files are provided via a path (3.4)

- **Interface Requirements (2)**
    - Csv file (2.1)
        - Csv file has students in rows and activities in columns (2.1.1)
        - Csv file of multiple choice test has students in rows and choices in columns (2.1.2)

4.3 LIST 3 (needed but less important to be functioning of core features of the system)

- **Usability (1)**
    - Input and output files assemble interface for programming assignments is provided (1.1)
        - No limit on the number of programming assignments (1.1.1)
        - Marker can enter the path to the solution code, sample inputs, sample outputs, console input, and console outputs (1.1.2)
- **Dependability (2)**
    - Reliability (2.1)
        - The system keeps back up of activity solution (2.1.1)
    - Security (2.2)
        - User Logins (2.2.1)
            - User ID is no more than 8 characters (2.2.1.3)
- **Supportability (4)**
    - Maintainability (4.1)
        - Database management (4.1.1)
            - Backup (4.1.1.1)
            - Data are stored in MS SQL database and kept for 5 years (4.1.1.2)

4.4 LIST 4 (not necessary to the successful and efficient operation of the core system aka only the client wants this)

- **Usability (1)**
    - If user has multiple roles, the user needs to select one when log in (1.2)
- **Dependability (2)**
    - Security (2.2)
        - User Logins (2.2.1)

        - Allow maximum 6 unsuccessful login attempts (2.2.1.4)
- **Performance (3)**
    - Switch between tasks in one click away (3.2)
- **Implementation requirement (1)**
    - The system uses Java as programming language (1.1)

Part II

Requirements Analysis / System Models

1 - Use Cases

Accounts

Use Case: Create Account
List of Requirements Satisfied by the use case: 1.1.1
Initiating Actor: System Administrator
Description: The System Administrator selects “Manage Accounts” in the Tasks Available interface and then “Create Account”. The System Administrator will enter the information for the attributes: Role(s) (Can give multiple roles to a user by clicking the plus button that adds another role), Name (First, Middle (opt), Last), Employee ID, User ID, and generates a Temporary Password. Once the System Administrator is finished inputting correctly the attributes of the account, and no field is left empty, the System Administrator can push “Save.” The System will then direct the user to the Modify Account Screen (see the use case below) to allow for the System Administrator to see what the information was that the user has saved to the database and when satisfied they push “Save”. The System Administrator is then brought back to the Manage Accounts Interface.

Use Case: Modify Account
List of Requirements Satisfied by the use case: 1.1.2
Initiating Actor: System Administrator
Description: The System Administrator selects “Manage Accounts” in the Tasks Available interface and then “Modify Account”. At the Search for Account Interface the System Administrator enters a valid Employee ID of the account and pushes “Enter”. The System Administrator is then directed to the Modify Account Interface which displays the attributes information: the account’s Role(s), Name (first, middle (opt), last), Employee ID, and the User ID. The System Administrator will modify any of the information for the attributes and can add a Role and once finished, and every field is not left empty, they push “Save” and they are taken back to the Manage Accounts Interface. If this is the only System Administrator or the only unblocked one then they can’t modify their System Administrator Role.

Use Case: Delete Account
List of Requirements Satisfied by the use case: 1.1.3
Initiating Actor: System Administrator
Description: The System Administrator selects “Manage Accounts” in the Tasks Available interface and then “Delete Account”. At the Search for Account Interface the System Administrator enters a valid Employee ID of the account and pushes “Enter”. The System Administrator is then directed to the Delete Account Interface which displays the attributes information: the account’s Role(s), Name (first, middle (opt), last), Employee ID, and the User ID and then the System Administrator will push “Delete”. If this is the only System Administrator or the only unblocked one then they can’t delete their own account. Also the deleted user, if active, is logged out.

Use Case: Block Account

List of Requirements Satisfied by the use case: 1.1.4
Initiating Actor: System Administrator
Description: The System Administrator selects “Manage Accounts” in the Tasks Available interface and then “Block Account”. At the Search for Account Interface the System Administrator enters a valid Employee ID of the account and pushes “Enter”. The System Administrator is then directed to the Block Account Interface which displays the attributes information: the account’s Role(s), Name (first, middle (opt), last), Employee ID, and the User ID and then the System Administrator will push “Block”. If this is the only System Administrator or the only unblocked one then they can’t block their own account. Also the blocked user, if active, is logged out.

Use Case: Unblock Account
List of Requirements Satisfied by the use case: 1.1.4
Initiating Actor: System Administrator
Description: The System Administrator selects “Manage Accounts” in the Tasks Available interface and then “Unblock Account”. At the Search for Account Interface the System Administrator enters a valid Employee ID of the account and pushes “Enter”. The System Administrator is then directed to the Unblock Account Interface which displays the attributes information: the account’s Role(s), Name (first, middle (opt), last), Employee ID, and the User ID and then the System Administrator will push “Unblock”.

Use Case: Backup
List of Requirements Satisfied by the use case: 7.8
Initiating Actor: System Administrator
Description: The System Administrator selects “Backup” in the Tasks Available interface. In the Backup interface the user can view when the last backup occurred and manage when a Periodical Backup takes place (ie. Hourly, Daily, Monthly, etc.) and push “Save”. The System Administrator will then be taken back to the Tasks Available interface. However in the Backup interface the System Administrator is also able to do a Manual Backup by clicking on the Manual Backup button, which they may want to perform after a course has completed. The system will then lead them to a screen that advises them where the backup was made to and then send them back to the Tasks Available interface.

Use Case: Restore
List of Requirements Satisfied by the use case: 7.9
Initiating Actor: System Administrator
Description: The System Administrator selects “Restore” in the Tasks Available interface. From this interface the System Administrator is taken to the Restore interface where they can choose a time they would like to restore the system to and then the System Administrator pushes “Confirm.” After this a confirmation window comes up advising the System Administrator that everything past the last backup will be lost and that perhaps they may wish to backup before they restore. If the System Administrator is satisfied they may then push “Confirm” and the system will be restored.

Use Case: Installation
List of Requirements Satisfied by the use case: 7.8

Initiating Actor: System Administrator
Brief Description: The System Administrator selects “Installation” in the Tasks Available interface. This will then lead the System Administrator to the Installation interface where the System Administrator can push a button that says “Install the latest updates” after which the system will search for the latest updates and install them.

Courses

Use Case: Create Course
List of Requirements Satisfied by the use case: 2.1.1
Actor: Administrative Assistant
Description: Administrative Assistant selects to ‘create a course’ from ‘Manage Courses’ screen; a ‘creating a course’ screen will be displayed. Administrative Assistant needs to fill in the course number, course name, start date and end date in order to create a course. The instructor’s name, employee ID, TA’s name and employee ID and the csv file of student list can be filled in either at the time of creating the course or at a later time/date.

Use Case: Modifying Courses
List of Requirements Satisfied by the use case: 2.1.2
Actor: Administrative Assistant
Description: The user would first have to search to course in order to modify it by specifying the course number, start date, and end date. After the course has been found, the modifiable attributes will be displayed to the user which includes course name, course number, start date, end date, instructor name and ID, TA(s) name and ID, and the list of students. In order to modify the students, the user would import a new list of students to the grading system. In order to add TAs to the course, only users with the role of a TA can be made a TA of a course and an error message will display otherwise. The other attributes can be filled into their individual fields that are manually filled in by the user.

Use Case: Delete Course
List of Requirements Satisfied by the use case: 2.1.3
Actor: Administrative Assistant
Description: Administrative Assistant selects to ‘delete a course’ from ‘Manage Courses’ screen; a ‘course to delete’ screen will be displayed. Administrative Assistant needs to fill in course name, course number, start date, and end date. Course can only be deleted if there isn’t any activity created for this course

Use Case: Copying Courses
List of Requirements satisfied by the use case: 2.1.4
Actor: Administrative Assistant
Description: The user would first have to search the course they would like to copy by specifying the course number, start date, and end date. After the course has been found, the system will display the following attributes that will be copied over to the new course which is the course name, course number, instructor name and ID, and TA(s) name and ID. After the information to be copied is displayed to a user, the system would prompt the user to confirm. The start and end date, list of students, and any activities added by the instructor would not be copied over to the new course.

Activities

Use Case: Manage Activities
List of Requirements satisfied by the use case: 3.1.1 – 3.1.4
Actor: Instructor
Description: Manage activities include the following use cases: *Create an activity*, *Modify an activity*, *Delete an activity*, *Copy an activity*.

Use Case: Create an activity
List of Requirements satisfied by the use case: 3.1.1
Actor: Instructor
Brief Description:

1. The Instructor chooses the specific course he or she wants to add an activity or activities in.
2. The Instructor chooses the option to create an activity. He or she must fill in the following list of attributes during creation time: activity name, path to solution files, activity type. After all other information can be specified and modified later. However, the Instructor must add the following items to the activity during creation or after: a rubric, a path to the directory containing students' submissions, number of tests to run (if it is a programming activity).
3. Other items that the Instructor may include: Description of an activity, due date and the necessary input and output files depending on the type of the activity.
4. The Instructor finishes creating the activity and the information is registered into the Database. The system acknowledges the creation of the activity by displaying the appropriate message box to the Instructor.

Use Case: Delete An Activity
List of Requirements satisfied by the use case: 3.1.3
Actor: Instructor
Brief Description: The Instructor chooses the specific course he or she wants to delete an activity or activities from. The Instructor selects the option to delete the specific activity he or she wants. The system then deletes the associated information related to the activity from the Database, and acknowledges the Instructor by displaying the appropriate message box. However, the system will block the Instructor from deleting an activity while it is being graded.

Use Case: Modify an activity
List of Requirements satisfied by the use case: 3.1.2
Actor: Instructor
Brief Description: The Instructor chooses the specific course he or she wants to modify an activity or activities from. The Instructor selects the option to modify an activity. He or she is able to change any items that are listed under the *Creation activity* use case. The Instructor finishes modifying the activity and the any updated data is saved into the Database. The system acknowledges the Instructor by displaying the appropriate message box.

Use Case: Copy an activity

List of Requirements satisfied by the use case: 3.2
Actor: Instructor
Brief Description: The Instructor chooses the specific course he or she wants to copy an activity into. The Instructor then chooses a specific or all activities he or she wishes to copy, from a list of courses he or she has taught in the past or is currently teaching. However, the following items are not copied: the path to students' submissions, students' grades for each item in the rubric, solutions to essays and problem sets, and due dates. The Instructor finishes copying the activities. The system then saves the associated data into the Database. The system then acknowledges the Instructor by displaying the appropriate message box

Use Case: Set Bonuses and Penalty Policy
List of Requirements satisfied by the use case: 5.8
Actor: Instructor
Brief Description: The Instructor decides to add an early bonuses/ late penalty policy. The Instructor inputs the information about the early bonuses/ late penalty policy Instructor enters at most three times that in term of hours as criteria to issue bonuses and penalties. For each time, instructor input percentage of grade that will be added or deducted from the grade of students' activities, which is in the range of 0% to 100%. Once the Instructor finishes input the information of bonuses and penalty, the Instructor can push "Validate". The early bonuses/ late penalty policy will be saved under the activity in the database. The System will then direct the user to the Create Activities interface.

Use Case: Modify Bonuses and Penalty Policy
List of Requirements satisfied by the use case: 5.8
Actor: Instructor
Description: The Instructor decides to modify an early bonuses/ late penalty policy. The Instructor re-input the information about the early bonuses/ late penalty policy Instructor modifies the times that in term of hours as criteria to issue new bonuses and penalties. For each time, instructor re-input percentage of grade that will be added or deducted from the grade of students' activities, which is in the range of 0% to 100%. Once the Instructor finishes input the information of bonuses and penalty, the Instructor can push "Validate". The System will then direct the user to the Modify Activities interface.

Rubric

Use Case: Manage Rubric
List of Requirements satisfied by the use case: 4.1
Actor: Instructor
Brief Description: Manage rubric includes *Create a rubric*, *Modify a rubric*, *Delete a rubric*. The use cases are included below.

Use Case: Create Rubric
List of Requirements satisfied by the use case: 4.1.1, 4.1.1.1.1, 4.2, 4.2.1, 4.2.2, 4.2.3, 4.3
Actor: Instructor

Description: The Instructor decides to create a rubric. The Instructor will input the expected outcomes and points for each requirement. If the Instructor decides to add more requirements in the rubric, he or she can click expand to add more elements. Once the Instructor finished inputting the requirements, he or she can press confirmed to save the rubric. The list of requirements and points will be save under the specified activity in the SGS database.

Use Case: Modify Rubric
List of Requirements satisfied by the use case: 4.1.2, 4.2, 4.2.1, 4.2.2, 4.2.3, 4.3
Actor: Instructor
Description:
The Instructor decides to modify a rubric.
The Instructor will change the expected outcomes and points for each requirement
The rubric of the specified activity will be extracted from the SGS database. If the Instructor decides to modify more requirements in the rubric, he or she can click expand to change or add more elements.
Once the Instructor finished editing the requirements, he or she can press confirmed to save the rubric.
The list of requirements and points will be save under the specified activity in the SGS database.

Grading and Solutions

Use Case: Grade an activity
List of Requirements satisfied by the use case: 5.1.3, 5.1.5
Actor: Marker
Description:
The Marker first chooses a course he or she wants to grade an activity or activities from.
The Marker then selects the activity he or she wants to grade.
Depending on the type of activity, it can be *Grade a programming activity*, *Grade an essay*, *Grade multiple choice tests* or *Grade problem sets*. (All use cases covered by group members)

Use Case: Regrade an activity
List of Requirements satisfied by the use case: 5.1.5
Actor: Marker
Brief Description:
The Marker first chooses a course he or she wants to regrade an activity or activities from.
The Marker then selects the activity he or she wants to regrade.
The Marker will then go through the same process as *Grade an activity (use case)*. The Depending on the type of activity, it can be *Grade a programming activity*, *Grade an essay*, *Grade multiple choice tests* or *Grade problem sets*. (All use cases covered by group members)

Use Case: Create programming tests
List of Requirements satisfied by the use case: 5.2.3-5.2.5
Actor: Instructor
Brief Description:

1. The user has logged in successfully as Instructor, chooses the specific programming activity he or she has created under the specific course.
2. The Instructor selects the option to add multiple programming tests to modify the activity.
3. For each programming test, the Instructor is able to specify the path to one console input, one console output, sample solutions (if there are any), multiple input and output files with the help of the system.
4. The Instructor then finishes preparing the programming activity by specifying the environment and language to be run (if he or she has not done so during creation time).
5. The system saves the information to Database and acknowledges the Instructor by showing a message box

Use Case: Grade a programming activity
List of Requirements satisfied by the use case: 5.2.3 – 5.2.5
Actor: Marker
Brief Description:
The Marker prepares the tests and environment settings in order to grade. (*Prepare a Programming Activity* use case).
The Marker selects a student to grade.
The system finishes compiling the student's submissions, and running all tests.
The system compares the correct output and the output of the student's output.
The system then allows the Marker to switch between checking the student's codes outputs, solution codes and opening up rubric to grade no more than one click away.
The Marker then enters the grades for the specific item listed in the rubric.
The system saves the grades into Database for the student.

Use Case: Grade an Essay
List of Requirements satisfied by the use case: 5.4
Actor: Marker
Brief Description:
The Marker selects a student to grade (by student ID.)
System displays student's Essay submission, rubric, and information regarding the submission (Name of Activity, Student number, etc).
System displays boxes to mark each item in the rubric, individually.
The Marker enters the grades for each individual item listed in the rubric.
System saves the grades into the Database for the student.

Use Case: Grade a Problem Set
List of Requirements satisfied by the use case: 5.4
Actor: Marker

Brief Description:
The Marker selects a student to grade (by student ID.)
System displays general information about the submission (Name of Activity, student ID, etc). System gives the option to view the student submission and sample solution in separate windows.
The Marker views the submission and sample solution, which is retrieved from the Database.
The Marker makes direct comments onto the document and saves it. The edited document is saved into the Database.
The Marker enters the grades for the problem set.
System saves the grades into the Database for the student.

Use Case: Provide Multiple Choice Answer Key
List of Requirements satisfied by the use case: 5.3.1
Actor: Instructor
Brief Description: After the instructor creates the multiple choice activity, the instructor provides the answer key with its points to the system.

Use Case: Grading Multiple Choice
List of Requirements satisfied by the use case: 5.3
Actor: Marker
Brief Description:
Instructors need set up the answer keys before grading, and specify the path to the students' or group's answers which is stored in .csv file.
Instructor or TA can select the student ID which he/she want to grade.
For group submission, the group name will be showed instead of student ID.
After the system finishes grading, the grade of the target will be stored into course management system.

System

Use Case: Log into the system
List of Requirements satisfied by the use case: 7.3
Actor: users of the system (Users include System Administrator, Administrator, administrative assistant, instructor, TA)
Brief Description:
Users include System Administrator, Administrator, administrative assistant, instructor, and TA.
The user needs to type in his/her user ID and password.
The system will connect to the database and check whether the user ID exists or not.
If the user ID exists then the system check whether the password match.
If password match with the one stored in database, login successful
If password did not match, login fails. The system keeps record the time of the user login fail.

Use Case: Block Account
List of Requirements satisfied by the use case: 1.1.4
Actor: Grading System
Brief Description:

If the user types his/her password wrong 6 times, the account associated with the user will be blocked. User need to contact with the System Administrator to unblock his/her account.

Use Case: Specify the role of the user
List of Requirements satisfied by the use case: 7.5
Actor: Users of the system (Users include System Administrator, Administrator, administrative assistant, instructor, TA)
Brief Description:
If the user type in correct user ID and password and login into the system successfully, the user need specify the role (System Administrator, Administrator, administrative assistant, instructor, TA) of the user. The user is only allowed choose one role at a time. If the user have multiple roles in the system and want change his/her role, the user needs to log off the system then log in the system and reselect the role the user want.

Formal Use Case

Create a programming activity

For Streamlined Grading System (SGS)

1. Participating actors:

§ Initiated by Instructor

§ Communicates with Database

2. Preconditions:

§ The user is logged in as an Instructor correctly.

§ The Instructor chooses a specific course he or she wishes to create a programming activity from.

3. Flow of events:

1. The Instructor requests the creation of a programming activity.
2. SGS responds by presenting the Instructor with an interface associated with activity creation.
3. The Instructor specifies the activity name, activity type, activity language and activity solution or solutions upon creation (mandatory fields).
4. The Instructor then chooses the option to add programming tests.
5. SGS responds by displaying the Instructor with another interface associated with creating programming tests.
6. The Instructor specifies the number of programming tests to run. For each test, the Instructor adds the path to one console input, one console output, multiple input files, multiple output files, and the solution codes if applicable. (***Create programming tests*** use case by Yinglun)
7. The Instructor creates a rubric for the activity.
8. SGS responds by presenting the Instructor another interface associated with rubrics.
9. The Instructor specifies the list of expected outcomes, and the number of points associated with each outcome for the rubric. (***Create a rubric*** use case by Telun)
10. The Instructor finishes creating the activity by specifying the path to a directory containing students' submissions.

4. Post Conditions:

§ The Instructor notifies SGS when he or she finishes creating the activity.

§ SGS acknowledges the Instructor back by displaying the appropriate message box, and saves all the data made by the Instructor into the Database.

5. Exceptional flow of events #1:

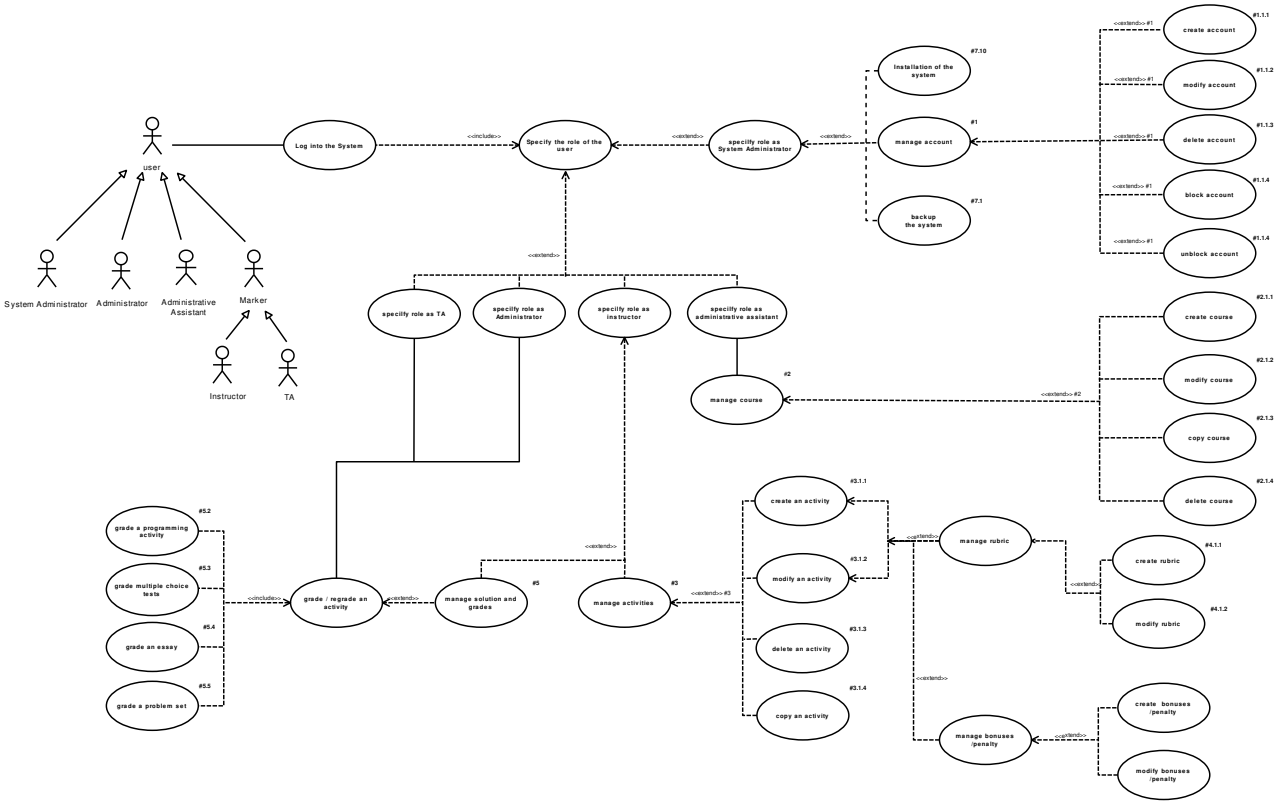
1. The Instructor decides to leave steps 4, 7, or 10 above blank during creation and adds them at a later date.
2. The Instructor logs back into SGS correctly, chooses the course he or she wishes to modify an activity from.
3. The Instructor selects the appropriate modify activity option to start editing his or her activities.
4. Continue from Step 3 under **Flow of events**.

6. Exceptional flow of events #2:

1. The Instructor enters an activity name which is already been created for the course.
2. SGS responds by displaying the appropriate warning message
3. Continue from Step 3 under **Flow of events.**

2 - Use Case Diagram
(Included in the next page)

Use Case diagram together - Use Case Diagram



3 - Class Diagram
(Included in the next page)

Class Diagram - Class Diagram