Software Requirements Specification

for

SGS (Streamlined Grading System)

Version 1.0 approved

Prepared by Brian Fong, Keith Chan, Dean Fernandes, Darren Yang, Junho Sohn, Sherman Chao Wen Chow, Johny Kuang, Tyler Arseneault, Hans Kim

Simon Fraser University

March 25, 2019

Table of Contents

Revision History

Name	Date	Reason For Changes	Version
Brian Fong	March 25, 2019	Initial import of template and modifications	v1.0
Keith Chan	March 25, 2019	Filling in Section 1 (Introduction)	v1.0
Brian Fong	March 25, 2019	Filling in Section 1 (Introduction)	v1.0
Brian Fong	March 27, 2019	Filling in Section 2 & 3	v1.0
Keith Chan	March 27, 2019	Filling in Section 2 (Overall Description)	v1.0
Junho Sohn	March 28, 2019	Filled Section 4 (System features, use cases)	v1.0
Sherman Chow	March 28, 2019	Filled Appendix B (Data flow, activity diagrams)	v1.0

1. Introduction

1.1 Purpose

This document specifies the software requirements for the Streamlined Grading System (SGS) release 1, version 0. The current scope of the SGS will allow for marking automation, plagiarism checks, and transfer of grades to the appropriate course management system. The SGS will also have a login system for professors and teaching assistants. The professors will be able to designate the privileges that a teaching assistant should have. Furthermore, the SGS will contain a database to store previous exams, assignments, and questions to be used by professors.

1.2 Document Conventions

Each requirement statement is to have its own priority. Section headers and subsection headers must be bolded and font size set to 18 and 14 respectively. Underlining or highlighting may be reserved for specific words or requirements that merit special attention.

1.3 Intended Audience and Reading Suggestions

Different Types of Readers	**Suggested Reading Areas**
Current and Future Developers	Recommended reading: Section 1 to Section 6 following the order given in the table of contents
Designers	Recommended reading: Section 2 (Overall Description)
Customers and Users	Recommended reading: Section 2 (Overall Description)
Stakeholders	Recommended reading: Section 2 (Overall Description) Section 5 (Other Non-functional Requirements)

The following sections of the Software Requirements Specification will detail an overall description and is organized as follows: external interface requirements, system features, non-functional requirements, and other requirements.

1.4 Product Scope

The SGS is a software which helps streamline the grading process for Teaching Assistants and Professors/Course Instructors. This software will reduce the time needed for marking assignments and tests as well as being able to detect plagiarism and cheating. The system will be integrated with external software to allow for importing and exporting of assignments and grades, for quick transfer of grades to the university's course management system. A database will also be used to store previous exams and assignments as well as a question database to use for future courses between different professors.

The software will allow for reduced time of marking assignments for teaching assistants, and result both in more time for the teaching assistants to spend on student interactions and less overall teaching assistants needed per course. This will allow post-secondary institutions using the SGS to reduce costs for hiring teaching assistants while maintaining quality of learning for students. Being able to pool questions and assignments will allow for reduced time in generating assignments and exams for professors, allowing them to redirect more focus on teaching students and having more office hours.

1.5 References

Ieeexplore.ieee.org. (1984). *830-1984 - IEEE Guide for Software Requirements Specifications*. [online] Available at: https://ieeexplore.ieee.org/document/278253 [Accessed 26 Mar. 2019].

TechWhirl. (2019). *Writing Software Requirements Specifications (SRS) | TechWhirl*. [online] Available at: https://techwhirl.com/writing-software-requirements-specifications/ [Accessed 26 Mar. 2019].

2. Overall Description

2.1 Product Perspective

The SGS is a new, self-contained product. The need for the SGS arose from the need to have a streamlined service for professors and teaching assistants. The Dean had decided to allocate funds for the development of the SGS in hopes of saving time for professors and teaching assistants in the hopes of increasing the quality of education of student and reducing teaching assistant cost. In collaboration with stakeholders, a set of requirements has been drafted which will be partially documented in more detail later in the SRS. The SRS of the SGS does not define a component of a larger system. The SGS will interact with external plagiarism detection tools and the post-secondary institution's course management system.

2.2 Product Functions

Product functions that the user must be able to perform include:

1. Logging in and proper authentication for the appropriate account
2. Streamlined marking of essays, problem sets and programming assignments
   - 2.1. Importing assignments, grades and class lists from a course management system
   - 2.2. Uploading marked assignments with comments and feedback, and grades to a course management system
3. Detecting and flagging plagiarism in a given assignment
4. Creating and viewing grading rubrics
5. Accessing and saving test questions into a database

More details will be located in Section 3.
Refer to Appendix B to view a data flow diagram illustrating the functions being used.

2.3 User Classes and Characteristics

User Classes:

1. System Administrator
   - 1.1. Manages installation of the system
   - 1.2. Can create and manage user accounts
2. Admin Assistant
   - 2.1. Creates and manages courses (including assigning the instructor, teaching assistants, and students)
3. Instructors
   - 3.1. Creates and manages rubrics, assignments, and exams for the course
4. Marker
   - 4.1. Includes instructors and teaching assistants
   - 4.2. Grades and re-grades assignments and exams
   - 4.3. Exports results to a course management system

2.4 Operating Environment

The SGS is cross platform and will be able to run on Windows, Mac, and Linux. The SGS will require internet access to be able to access other programs and systems such as the course management system used by the post-secondary institution and additional plagiarism detection tools.

2.5 Design and Implementation Constraints

First issue may be the plagiarism checker and the availability of a pre-existing software that will allow smooth integration with the system to be built as well as not going against any licensing policy. The developers should aim to use a pre-existing system to reduce workload, but may need to expect to develop a bespoke one.

Secondly, developers need to comply with pre-existing data requirements and privacy regulations already in place to ensure that the student's privacy is properly protected and that they are in line with the intended security needed. For example, if a post-secondary institution wishes to use its local servers to store data, developers must adapt the SGS accordingly. Security is important as well as it ensures that students will not easily be able to access and modify their own grades.

Depending on the contract and the given time proposed for the contract, ABC software consultants, may be liable to maintaining the software in the given period. However, afterwards, if ABC chooses not to continue on with a new contract, the system administrator or in house developer of the post-secondary institution must continue on maintenance.

2.6 Assumptions and Dependencies

Assumed factors:

1. The plagiarism checker is available and readily usable for the SGS
   1.1. If not, then appropriate measures should be taken to develop a bespoke system that can be integrated into the SGS
2. The post-secondary institution prefers to have the SGS available on all major operating systems
   2.1. If not, the SGS will only need to be developed for the preferred operating systems
3. The external source for importing assignments will work and that future scalability and adaptations will be available as needed by the professors or teaching assistant
4. The raw scanned results from multiple choice exams is readable by the SGS
5. The interface to access the database must be adaptable to different types of databases in the case where a new database must be used, there should be minimal effort in switching between databases.

3. General Business Requirements

1. The SGS will reduce time spent marking by two hours per week per marker so that the post-secondary institution can reduce TA costs.
2. The SGS will allow markers to upload student work complete with feedback to the course management system so that students can find feedback for their work online.
3. The SGS will allow instructors and teaching assistants to view all material needed to mark assignments and exams on one platform so that markers can be more organized and reduce time spent marking.
4. The SGS will interact with third party plagiarism detection tools to reduce development costs.
5. The SGS will interact with the post-secondary institution's existing course management system so that the institution can have a streamlined process when updating grades to save time for instructors and TAs.

4. System Features

4.1 Creating Rubrics

4.1.1 Description and Priority

Instructors shall be able to create grading rubrics for each assignment or exam. After being created, these rubrics can be viewed by the instructor and teaching assistants for the course. The system will require a rubric to be submitted by the instructor responsible for that course for the assignments to be graded against.

Benefit	7
Penalty	4
Cost	5
Risk	6
Priority	Medium

4.1.2 Stimulus/Response Sequences

Instructor requests to create or edit a rubric/System opens rubric editing component
Marker requests to view a rubric/System opens rubric in a non-editable format

4.1.3 Functional Requirements

REQ-1: The system will provide an editor for the user to create and edit rubrics
REQ-2: The system will save rubric created by the user
REQ-3: The system will allow the user to view the rubric in a non-editable format

4.1.4 Use Cases

1. Pre-conditions
    - 1.1. The instructor must be assigned to the course that they want to make the rubric for.
    - 1.2. There must be an assignment to which a rubric can be assigned to.

2. Post-conditions
    - 2.1. Instructor must be able to copy a rubric and assign it to other multiple assignments.
    - 2.2. Rubric must be able to be modified or deleted only by instructor.
    - 2.3. Both professor and TA must be able to grade student's assignments using the rubric they created.
    - 2.4. Students must be able to view the rubric and their grade in each criteria of rubric.

3. Normal Flow
    - 3.1. The instructor selects the course.
    - 3.2. The instructor selects the assignment which professor wants to make rubric for.
    - 3.3. The instructor creates new empty rubric.
        - 3.3.1. Once the empty rubric has been created, it is saved into the database.
        - 3.3.2. The system displays the input forms that the instructor must fill in
    - 3.4. The instructor describes the grading criteria for each item in the rubric.
    - 3.5. The instructor assigns grading scheme for each item in the rubric.
        - 3.5.1. The rubric is saved again to the database where it can be accessed again later.

4. Alternative Flows
    - 4.1. The professor realizes that he has entered the wrong number of points for an earlier rubric item
        - 4.1.1. The professor selects the rubric item that needs changing
        - 4.1.2. The professor enters the corrected number of points
        - 4.1.3. The professor returns to the end of the list of entered rubric item to continue entering more rubric items
    - 4.2. The professor decides to edit a rubric item description for an earlier rubric item.
        - 4.2.1. The professor selects the rubric item that needs changing
        - 4.2.2. The professor enters the corrected number of points
        - 4.2.3. The professor returns to the end of the list of entered rubric item to continue entering more rubric items
    - 4.3. The professor decides to create a new rubric by copying an existing one as a template
        - 4.3.1. The professor selects the course and assignment to make the rubric for.
        - 4.3.2. The professor creates a new empty rubric
        - 4.3.3. The professor selects "copy template" from the menu options within the rubric creation page
        - 4.3.4. The professor selects an existing rubric to copy the details from
        - 4.3.5. The professor edits the fields as they want changed, then saves the rubric
    - 4.4. The professor decides to remove one of the rubric items he has already entered
        - 4.4.1. The professor selects the rubric item to delete
        - 4.4.2. The professor deletes the rubric item
        - 4.4.3. The professor returns to the end of the list of entered rubric

5. Exceptional Flows
    - 5.1. Rubric is not applicable for a custom assignment (non-graded, surveys, etc.)
        - 5.1.1. SGS will alert the instructor that the rubric option is not available.

5.2. Instructor tries to create and assign rubric to assignment that already has a rubric

5.2.1. SGS displays pop-up window to ask if the instructor wants to replace the old rubric or not

4.2 Grading Essays/Problem Sets

4.2.1 Description and Priority

Markers shall view the student's essay/problem set, rubric, and solution at the same time. The marker shall add feedback and a grade to the student's work and rubric, respectively, then save the file.

Benefit	9
Penalty	5
Cost	3
Risk	6
Priority	High

4.2.2 Stimulus/Response Sequences

1. Marker requests to grade work/System opens the student work, rubric and solution
2. Marker clicks on specific section of work/System allows marker to highlight or add comments to this section
3. Marker clicks the save button/System saves the updated file with feedback into the original file
4. Marker inputs points earned by student for each rubric item/System saves points for each student

4.2.3 Functional Requirements

REQ-1: The system must store work such as assignments and exams

REQ-2: The system will automatically grade certain questions that are preset to be automatically graded

REQ-3: The system will allow an interface for user to grade work

REQ-4: The system will save the graded work for later access

REQ-5: The system will store points inputted by the user for a student corresponding to the rubric of the work

4.2.4 Use Cases

1. Pre-conditions

1.1. The SGS is currently running.

1.2. The SGS is connected to the database

1.3. The TA is logged in and authenticated

1. Post-conditions

1.1. A new test is created and saved to the database.

1.2. The test results of the selected student code submissions are graded and displayed in the user interface.

2. Normal Flow
    - 2.1. The TA selects the desired course and coding assignment he or she wants to create a test for.
    - 2.2. The system shows a list of student code submissions and any existing tests for the coding assignment.
    - 2.3. The TA selects “Create a test”, which will display a form, prompting the TA to select a programming language, and enter in any inputs and expected outputs.
    - 2.4. The TA writes code for a test directly in a source code editor within the user interface.
    - 2.5. The TA selects “Save test” to save the test to the database and associate it with the coding assignment.
    - 2.6. The TA selects the student code submissions to be ran against the tests, and clicks “Run tests”.
    - 2.7. The system automatically runs the selected student code submissions against the tests
    - 2.8. The system saves the outputs of each test run in the database.
    - 2.9. The system calculates the grade of the coding assignment according to the number of tests passed and failed.
    - 2.10. The user interface displays a list of entries for each student code submission along with their grade.
    - 2.11. The TA selects a specific entry to view the logs and output of the test runs on the student code submission.

3. Alternative Flows
    - 3.1. Remove an existing test
        - 3.1.1. The TA selects an existing test to edit
        - 3.1.2. The system displays a form of the existing test details (programming language, inputs, expected outputs, and code) in the user interface
        - 3.1.3. The TA selects “Remove test”
        - 3.1.4. The system removes the test from the database
    - 3.2. Modify an existing test
        - 3.2.1. The TA selects an existing test to edit
        - 3.2.2. The system displays a form of the existing test details (programming language, inputs, expected outputs, and code) in the user interface
        - 3.2.3. The TA selects a different programming language, and enters in new inputs and expected outputs
        - 3.2.4. The TA modifies the existing test code in the source code editor within the user interface
        - 3.2.5. The TA selects “Save test” to save the updated test to the database

4.3 Checking Assignments for Plagiarism

4.3.1 Description and Priority

The system shall be able to access an external plagiarism detection tool and display the results generated by the tool.

Benefit	4
Penalty	2
Cost	4
Risk	3
Priority	Low

4.3.2 Stimulus/Response Sequences

1. Marker requests a plagiarism test performed by the system/System inputs student work into the plagiarism detection tool and flags assignment appropriately
2. Marker manually flags assignment and inputs justification for detected plagiarism/System applies flag and justification to assignment

4.3.3 Functional Requirements

REQ-1: The system has an interface to a plagiarism test
REQ-2: The system's interface will input selected work into the plagiarism detection tool
REQ-3: The system's interface will respond with the result from the plagiarism to be displayed for the user

4.4 Importing Assignments

4.4.1 Description and Priority

Markers shall be able to import the student work that they have been assigned to mark, from the course management system.

Benefit	7
Penalty	4
Cost	5
Risk	4
Priority	Medium

4.4.2 Stimulus/Response Sequences

1. Marker requests student work/System accesses course management system and imports student work

4.4.3 Functional Requirements

REQ-1: The system will be able to import assignments and exams from the course management system

REQ-2: The system will be able to authenticate whether a user has the required permissions to import a file

4.5 Exporting Grades, Graded Assignments, and Plagiarism Summaries

4.5.1 Description and Priority

Markers shall be able to export grades and graded assignments to the appropriate course management system. Likewise, the export function should allow the exportation of the plagiarism summaries to the course management system as well

Benefit	7
Penalty	6
Cost	5
Risk	7
Priority	High

4.5.2 Stimulus/Response Sequences

1. Marker requests to export file/System opens the file which may be the graded assignment, grades, or plagiarism summary
2. Marker clicks on destination to export to/System allows marker to confirm exportation
3. Marker clicks the accept button/System exports the files to the chosen destination
4. Marker inputs points earned by student for each rubric item/System saves points for each student

4.5.3 Functional Requirements

REQ-1: The system will export all student's earned points for all rubric items into a .csv file that can be loaded into the Course Management System

5. Other Nonfunctional Requirements

5.1 Security Requirements

1. Markers shall not be allowed to access resources they do not have the permissions for
2. The system shall not allow the exporting of information to anywhere besides the course management system
3. The system shall authenticate users with a user ID and password

5.2 Business Rules

1. All Users
    - 1.1. May have multiple roles that can be chosen after login
    - 1.2. Can edit their own passwords
2. System Administrators
    - 2.1. Can create and manage accounts
    - 2.2. Can search up accounts by name or employee ID
3. Admin Assistant
    - 3.1. Can create courses and assign an instructor to the course as well as multiple teaching assistant and students
    - 3.2. Can remove instructor/TAs/students assigned to the course
    - 3.3. Can modify or delete a course
4. Instructor
    - 4.1. Can create and modify activities created for the course
    - 4.2. Can create and modify grading
    - 4.3. Can delete activities
5. Markers
    - 5.1. Choose an activity from their chosen course to mark
    - 5.2. Can select student to grade and can grade essays, problem sets and programming assignments
    - 5.3. Can export all student's earned points for all items into a csv file that can be loaded into the course management system

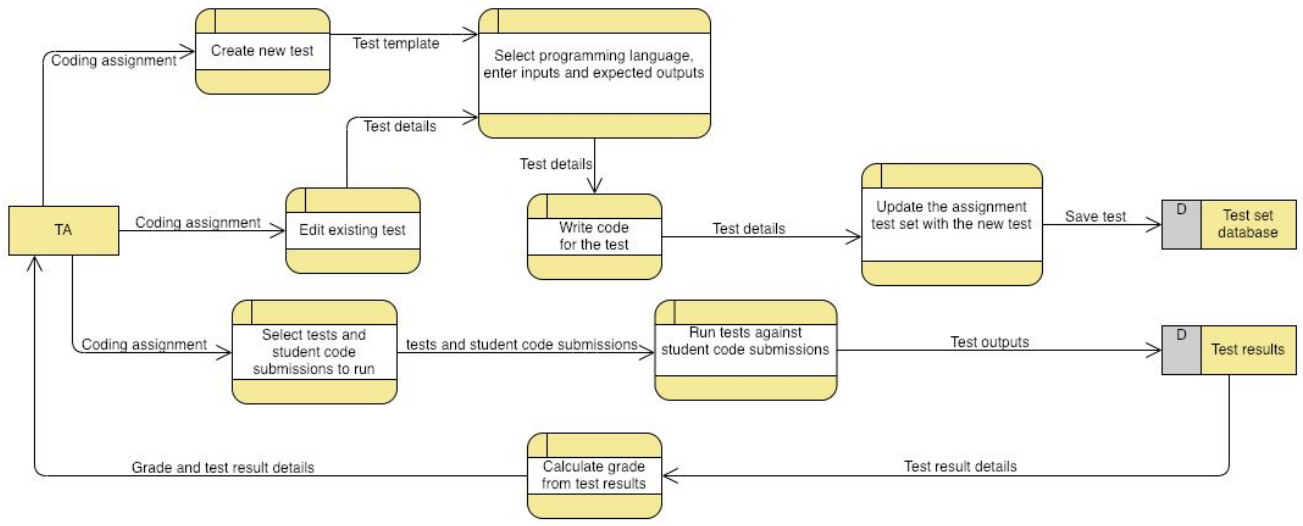
Appendix A: Glossary

Academic dishonesty: cheating and plagiarism by students
Account: an established relationship between a user and the Streamline Grading System where the user will be granted specific access based on his/her account username and password
Account (blocked): an account that has been disabled/denied access
Admin Assistant: a person who manages course, instructor, TA, and students.
Assignment: a self-contained project for assessment purposes
Bar codes: a unique identifier for each student associated with their student ID
Bubble sheet: a special sheet of paper where students can input multiple choice answers by using pencil to fill in "bubble" shapes that have a corresponding answer; they can be fed through a machine for automated marking
Comment: a feedback markers put on the assignments submitted by students in a sticky note format
Commented file: a submitted assignment where sticky notes are added on by markers for feedback purposes.
Console input file: one of the files that might be used for the testing of a programming solution. File(s) inserted on the console with information that are opened, read, and used during the execution of the program; the information in the input file will affect the results of the program.
Console output file: one of the files that might be used for the testing of a programming solution. File that is created by the program and generated by the console.
Course Activity: an assignment(Essay, Problem set, M/C test, or Programming) with a provided name, solution, and language, created by the Instructor for a specific course
Course management system: the default management system used by the school
CSV file: A file containing the list of students (names and Ids) which is used for creating courses directory
Directory: a virtual location(folder) in the computer storing information
Essay: a piece of writing written by students
Grade: the percentage value for the assignment that corresponds to the grade
Group activities: assignments associated with multiple students
Highlight: an action of emphasizing specific parts of the writings by colouring the background of the text
Markers: It refers both a TA and instructor
Multiple choice exam builder: tool to build exams in a multiple choice format
Multiple choice grader: tool used for reading and then marking multiple choice questions by comparing student solutions to solution key
Multiple choice module: a separate and independent set from the SGS which can be brought into SGS, containing the multiple choice exam question library, multiple choice exam builder, and multiple choice grader.
Multiple choice test: a type of test where a student has to choose 2-6 possible given options as an answer
Multiple choice exam question library: a library from multiple choice module containing a list of multiple choice questions. A question can be searched by entering keyword and question can be added
Programming assignment: an assignment which requires students to program

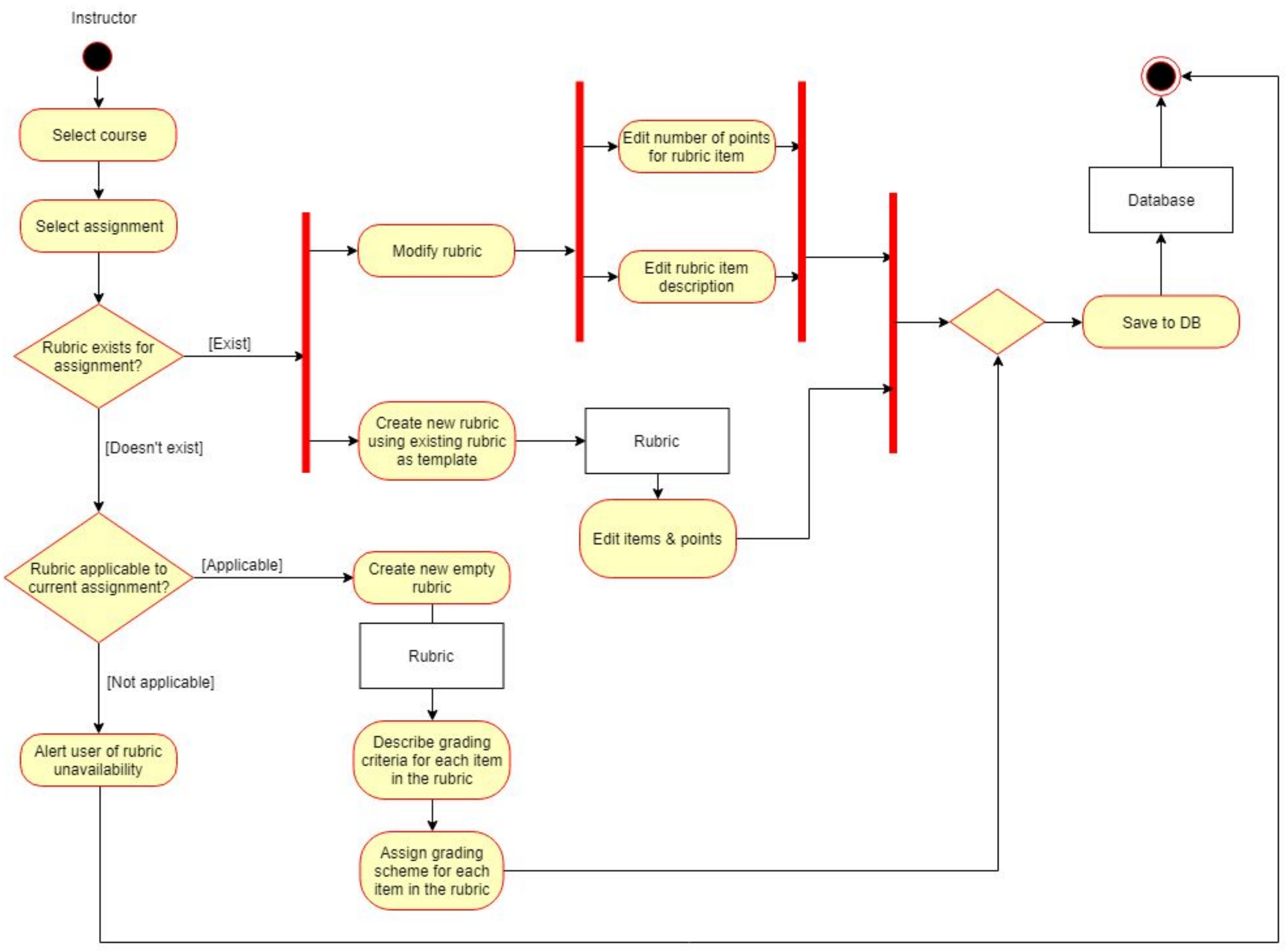
Resubmission: an act of submitting an assignment again
Role: an act of play every user is involving in the SGS. There are 4 roles: Instructor, TA, system admin, admin assistant. Each user may have multiple roles.
Rubric: a form filled to evaluate the assignment against
Rubric item: a rubric is consisted of two items: description and maximum points. Description describes what aspects the markers look at during marking. Maximum points describe a maximum point a student can get for the item.
Solution output file: the output file that the student's programming assignment output should match with
Statistical analysis: a statistical analysis based on the exam result of multiple choice
Sticky note: a virtual small piece of paper which gets written by markers in order to provide feedback to students
Student: a person enrolled in the class identified by a unique number/ID
Student number: a unique number assigned to student in the university
Student subdirectory: a directory for a specific student that submitted an assignment for a particular activity
Subdirectory: a virtual location (folder) storing particular students' submitted assignments
System Administrator: manages installation of the system, search for accounts, and manage (i.e. modify, delete, block, and unblock) any account
TA - Teaching Assistant: marks assignments but does not have full access rights compared to the prof
Test case: set of pre-made tests and criteria used to determine the correctness of coding assignments
Test solution: output files created and written
User: a person who uses the SGS

Appendix B: Analysis Models

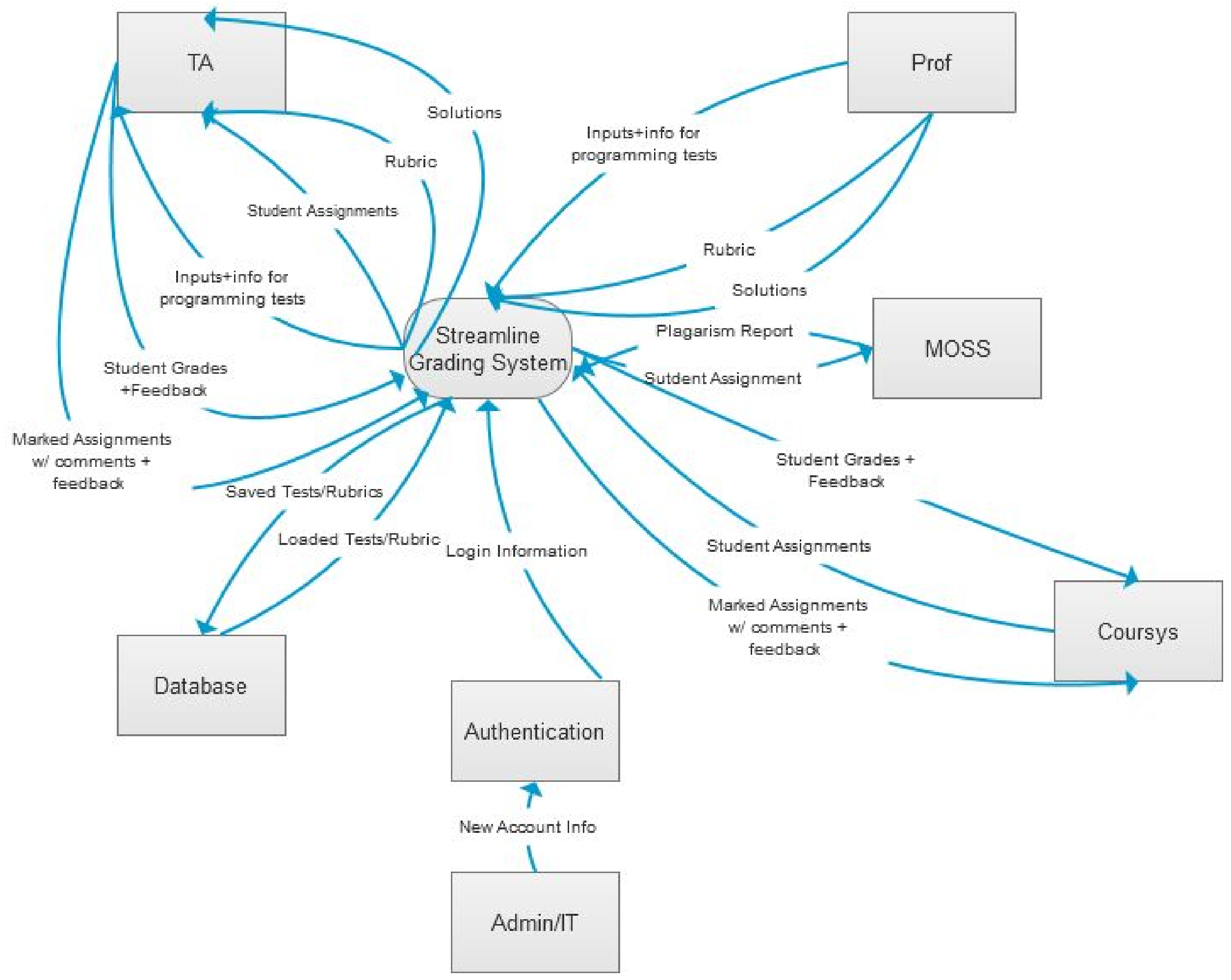
Data flow diagram for use case "Grading coding assignments (including creating tests)"


Create new test
Coding assignment
Test template
Select programming language, enter inputs and expected outputs
Test details
Test details
TA
Coding assignment
Edit existing test
Write code for the test
Test details
Update the assignment test set with the new test
Save test
D
Test set database
Coding assignment
Select tests and student code submissions to run
tests and student code submissions
Run tests against student code submissions
Test outputs
D
Test results
Grade and test result details
Calculate grade from test results
Test result details

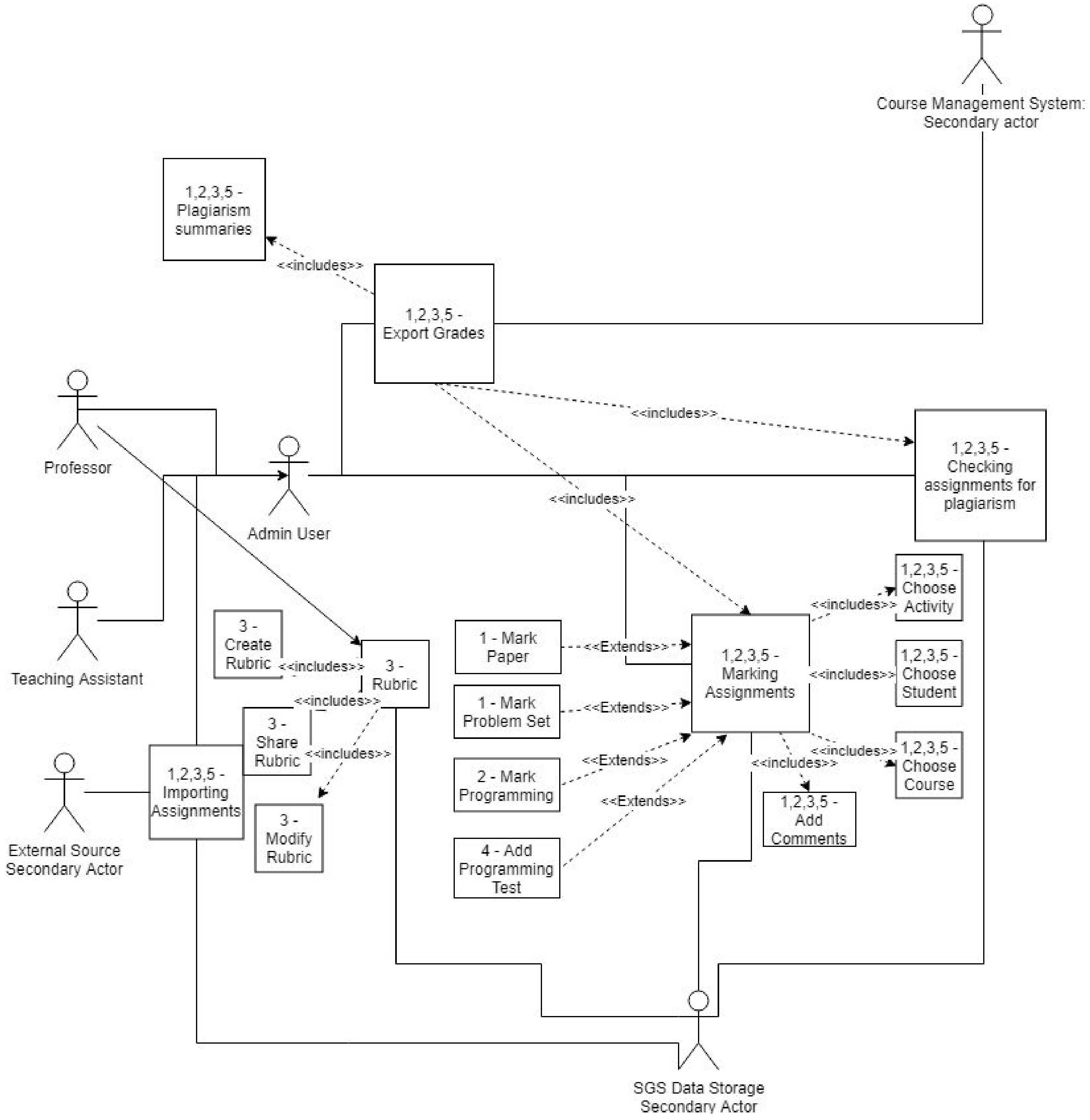
Activity diagram for use case "Creating a rubric"


Instructor
Select course
Select assignment
Rubric exists for assignment?
[Exist]
[Doesn't exist]
Modify rubric
Edit number of points for rubric item
Edit rubric item description
Create new rubric using existing rubric as template
Rubric
Edit items & points
Rubric applicable to current assignment?
[Applicable]
[Not applicable]
Create new empty rubric
Rubric
Describe grading criteria for each item in the rubric
Assign grading scheme for each item in the rubric
Alert user of rubric unavailability
Save to DB
Database

Data Flow diagram for Product Functions



Use Case diagram for the SGS system



Data Models

Account Table

accountID (PK)	username	password	firstName	middleName	lastName	accountType

Course Table

CourseID (FK)	**instructorID (FK)**	CourseName	CourseNumber	startDate	endDate	NumberOfStudents	isExpired

Assignment Table (Which TA is assigned to mark which assignment)

AssignmentID (PK)	**accountID (FK)**	**courseID(FK)**	isExpired	dueDate

Rubric Table

CourseID (FK)	**AssignmentID (FK)**	**criteriaID (FK)**	MaxPoints

Rubric Grade Table

CourseID (FK)	**assignmentID (FK)**	StudentID	CriteriaColumnNumber	studentPoints

Criteria Table

criteriaID (PK)	criteriaTitle	criteriaDescription

Programming Test Table

CourseID (FK)	**AssginmentID (FK)**	programmingLanguage	numberOfTest	Environment

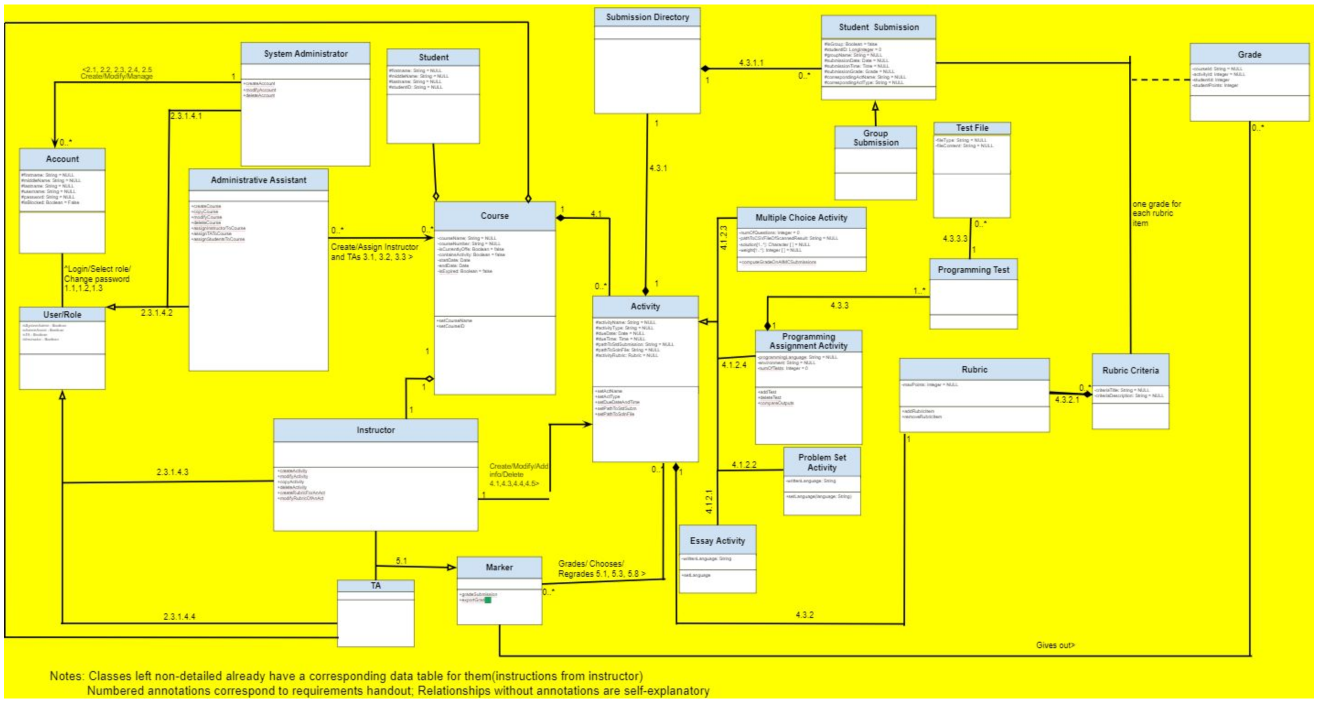
Student Submission Grade Table

CourseID (FK)	**assignmentID (FK)**	StudentID	SubmissionConfirmation Number	SubmissionDate&Time	plagiarismFlag

Question Database (To keep a list of past exam questions)

QuestionID (PK)	CourseID	dateExamined	Type	Question	Answer

Class Diagram (based on Data Models)



Notes: Classes left non-detailed already have a corresponding data table for them(instructions from instructor)
Numbered annotations correspond to requirements handout; Relationships without annotations are self-explanatory