Software Requirements Specification

Student Code Online Review and Evaluation (2.0)

S.C.O.R.E. (2.0) aims to enhance and extend the original S.C.O.R.E. application. Adding functionality in server connections, grading with rubrics, and cheat detections are all intentions to deploy the system into CSE classrooms.

Team Members

Dorothy Ammons - dammons2022@my.fit.edu
Patrick Kelly - pkelly2022@my.fit.edu
Shamik Bera - sbera2022@my.fit.edu
Rak Alsharif - ralsharif2021@my.fit.edu

Faculty Advisor and Client

Raghuveer Mohan

November 23, 2025

Table of Contents

Introduction

1.a Purpose

The main purpose of this document is to show the requirements of the novel features or functionalities of the Student Course Online Review Evaluation (SCORE) platform during the development phase.

1.b Scope

The scope of the Student Course Online Review Evaluation (SCORE) platform is to provide more capabilities to the current application, such as: making it easier to detect cheating through AI and collusion, improving the command line functionalities, and eventually implementing the application into the Florida Tech CSE classrooms in order to gather feedback/suggestions from students and professors.

1.c Definitions, acronyms and abbreviations

- SFTP: Secure File Transfer Protocol
- Generative AI: Generative Artificial Intelligence
- MOSS: Measure of Software Similarity
- Flask: Web framework for python, used to build the API and handle HTTP requests
- HTTPS: Hypertext Transfer Protocol Secure, communicates with clients over the network to Flask
- CSV: Comma Separated Values, file type used to store text in tables
- Canvas: Classrooms portal for Florida Institute of Technology students and professors
- API: Application Programming Interface, allows communication between software applications
- OAuth: Authorization protocol used to authorize users
- CLI client: Command Line Interface client, allows users to communicate with client server through command line interface

1.d References

- MOSS: https://theory.stanford.edu/~aiken/moss/
- Flask: https://flask.palletsprojects.com/en/stable/
- OAuth: https://oauth.net/2/

1.e Overview

The overall objective of the Student Course Online Review Evaluation (SCORE) is to expand its current functionalities by implementing advanced cheating with AI and collusion detection, improving accessibility for command line users, and integrating feedback improvements from the students and professors.

Overall Description

2.a Product Perspective

This system is a follow-up to the previous S.C.O.R.E. project. The program will add new functionalities and improvements to the existing system. These enhancements will interact with the current system's architecture and framework while maintaining the same look and feel of the previous version.

2.b Product Functions

The main features of the application will be:

- Canvas integration
- Custom grading rubrics
- Visuals for cheat detections
- HTTPS connection access for off campus users

2.c User Classes and Characteristics

Admin users will have access to all of the system files and are responsible for the upkeep and maintenance of the program. Professors are users who will be able to upload Canvas rosters to their classes, add custom rubrics to their assignments, export grades from an assignment, and view visuals displaying various methods of integrity violations, including: plagiarism, AI usage, and collusion. Students are users who will be able to access the shell client off of campus.

2.d Operating Environment

The system will operate through a serverless application hosted on Google Cloud Run. Data storage for relational databases will be on Cloud SQL and Cloud Storage will hold all submitted files and CSVs.

2.e Constraints

All additions must remain compatible with the current systems logic and processes. Database space and speed must be managed as it is limited by Cloud SQL. The execution of committed code is limited by lack of concurrency on isolated containers.

2.f Assumptions and Dependencies

Some product functionality relies on cooperation from the MOSS API to return collusion reports. The running of the server relies heavily on the containers provided by Google Cloud Run and is limited by its memory and processing speed. The relational database is reliant on space and speed provided by Cloud SQL.

Requirements

3.a Functional Requirements

3.a.1 Import Rosters

After a professor creates their class on the web page, an import button allowing the import of a CSV file will be prompted. For terminal users, “upload roster <file>”, after selecting the class, will accept the CSV file. This CSV file should be directly exported from a Canvas class roster. Then, all students on that file will be added to the roster associated with that chosen class.

3.a.2 Export Grades

At any point before or after an assignment is due, professors will have the option to export the student grades for that assignment. The exported file will be in a CSV format, in the style so that it may be directly uploaded to Canvas. On the web page, the button to export grades will be present on the assignment page. Within the terminal, “export grades” may be run after selecting an assignment in order to receive the CSV.

3.a.3 MOSS Similarity Detection

At any point before or after an assignment is due, professors will have the option to detect similarities between submissions. On the web page, this option is presented with a button on the selected assignment page that will run the MOSS API across the stored submissions. The professor also has the option to decide the threshold in which similarity scores are considered cheating. For example: a professor may select 80% similarity to be an indication of collusion. This is important for how the visualization is performed.

3.a.4 MOSS Score Visualization

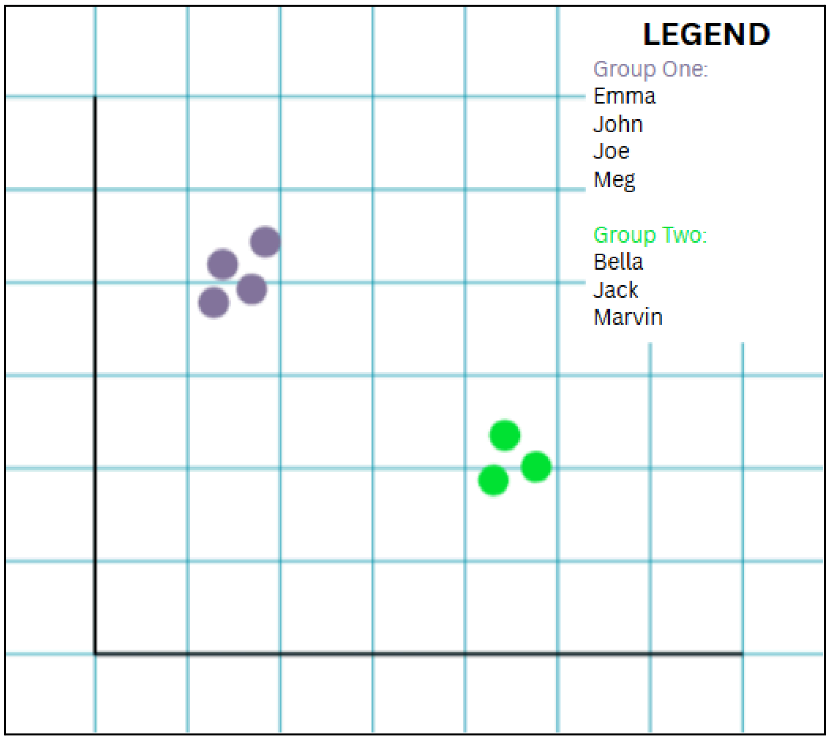
After the “detect similarities” button is clicked, the information MOSS API returns will be transformed and visualized for the professor’s view in two ways. The first visual will be an n by n matrix, n being the number of students who submitted to the assignment, where both the columns and rows contain are headed by the student’s names. The cross sections will represent the similarity scores between those student’s submissions. Scores will be highlighted in accordance with the selected or default threshold, much higher than the threshold should be bright red, much lower than the threshold should be bright green, and close to the threshold should be shades of yellow or orange.

Example matrix:

	Josh	Emma	Brad
Josh	N/A	90%	25%
Emma	90%	N/A	40%
Brad	25%	40%	N/A

The second visual will be clusters, where only the students whose submissions were at or above the threshold are shown. This is to show professors if groups of students have similar code, rather than one on one. A legend will be present to display all the names of the students within each cluster.

Example Cluster Graph:



3.a.5 AI Detection

At any point after an assignment's creation, professors have the option to check the submissions saved in the database for the probabilities that they are AI-generated. The option is presented with a button on the assignment's page. Once clicked, a hard coded LLM will predict the probability that each submission was generated by AI in a separate container. A table is then displayed with each student's name next to their probability percentages. Higher percentages indicate a bigger likelihood to be AI.

3.a.6 Custom Rubrics

Once an assignment has been created, alongside the test cases, boxes for rubric based scaling will be shown. A total points box will allow the input of whole numbers 1 and up. Each test case can be given its own number of points, a whole number 1 and up (this can be higher than the overall points due to extra credit). Points may also be allocated for attempt, compilation and time limits, within the same range. Point reductions may be introduced for time limits and late submissions as well. The rubric will be saved to the database and applied to all submissions for that assignment.

3.a.7 Remote Command Line Server Connection

The system will provide access to an HTTPS connection to authenticated users through the user's terminal. This will allow users to perform terminal line commands that directly use the functionality provided by the application. The CLI Client python file will be provided in order to process commands smoothly.

- Example input
  - python3 scoreportal.py login
- Example output
  - File opens a browser window to complete Google OAuth login
  - Once logged in, access token is saved in CLI file
  - Log in does not need to be repeated

- Example input (student)
  - python3 scoreportal.py help
- Example output (student)
  - Commands include:
  - list classes
  - select <class>
  - list assignments
  - select <assignment>
  - submit <file>
  - status <taskID>

- Example input (professor)
  - python3 scoreportal.py help
- Example output (professor)
  - Commands include:
  - create class
  - list classes
  - select <class>
  - delete <class>
  - upload roster <file>

    - create assignment
    - list assignments
    - select <assignment>
    - delete <assignment>
    - export grades

- Example input
    - python3 scoreportal.py list classes
- Example output
    - CSE4081 - Intro Ana of Algorithms
    - CSE4101 - Computer Science Proj 1

- Example input
    - python3 scoreportal.py select CSE4081
    - python3 scoreportal.py list assignments
- Example output
    - Binary Search Tree

- Example input
    - python3 scoreportal.py select Binary Search Tree
    - python3 scoreportal.py submit <file>
    - python3 scoreportal.py status 1
- Example output
    - Success! Submitted. Your task ID is: 1. Say “status 1” to view results.
    - Test case 1 failed: Check for leaf nodes with equal values.
    - Grade: 9/10

All changes to the server database are processed through the connection with the Flask API via HTTPS. As code submissions are made, they are added to a queue and users are given a task ID. The code execution runs on a separate container so that the executions are unable to clog requests to the main server. Then, that separate container stores the results to the database. Users may request the status of their task using the given ID in which the server queries the database to return if the task is still in progress or if it is finished, in which the grade and test case failures are outprinted.

3.b Interface Requirements

3.b.1 Command Line Connection

All users can interact with the S.C.O.R.E. (2.0) application through command line operations that communicate with the main server and databases through HTTPS.

Student Functionality

- Students are able to log into the system through the terminal, which will open a browser and authenticate their user through OAuth.
- The student's list of classes is retrieved and available to view at any time.
- To reach the assignments of a specific class, students are able to select that class.
- Once a class is selected, students are able to view the class's assignments.
- Assignments are able to be selected at which point information about assignment expectations and due dates are outprinted.
- Within the selected assignment, students are able to submit their code solutions as a file.
- After making a submission, students may check the status of their code in order to receive their current grade and advice towards their code based on failed test cases.
- Any further submissions towards the same assignment will erase previous attempts.

Professor Functionality

- Professors are able to log into the system through the terminal, which will open a browser and authenticate their user through OAuth.
- The professor's list of classes is retrieved and available to view at any time.
- New classes can be created, and existing classes can be removed.
- Within a selected class, professors are able to upload a CSV roster in order to add students.
- Professors can view their full list of assignments.
- Assignments may be added or removed from a class.
- At any time, professors are able to export the grades for an assignment in CSV format for use in Canvas.

3.b.2 Web Application

All changes to the web application brought forth by S.C.O.R.E. (2.0) are relative to the professor's views and functionalities. This includes:

- Rubric options will be available on the assignment page alongside the test cases (see 3.a.6 for more details).
- Visualizations of MOSS similarity scores will be shown after the professor clicks the "detect similarities" button and enters an optional threshold, all on the page of a selected assignment (see 3.a.4 for more details).
- An AI probability score table will be generated if the professor clicks the "detect AI" button, also on the selected assignment's page (see 3.a.5 for more details).
- An option to import rosters will be present on the class-creation page (see 3.a.1 for more details).
- An option to export grades will be present on the selected assignment page (see 3.a.2 for more details).

3.c Security Requirements

3.c.1 User Authentication

All users must be authenticated through Google OAuth and subsequently TRACKS, while using their Florida Institute of Technology email address and password. In order to be added to a class, that authenticated email associated with the user must match the email associated with their Canvas account. This authentication will also aid in establishing professor and student roles.

3.c.2 CLI Connections

All data delivered over the CLI client connection will be processed through HTTPS communications.

3.c.3 Contarization

Code will be run in isolated containers to ensure they do not interfere with main server processes as well as maintaining effective concurrent connections.

3.c.4 Data Deletion

After an assignment or class is removed, all data related to submissions or rosters (in the case of class deletion) will be removed.