Project Title: Student Code Online Review and Evaluation 2.0
Names and email addresses of 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 from CSE: Raghuveer Mohan, rmohan@fit.edu
Client name and affiliation: Raghuveer Mohan, CSE Professor
Date(s) of Meeting(s) with the Client for developing this Plan: January 21st, 2026

Goals

- Increase the capabilities of the current S.C.O.R.E. application
  - Add detections to catch, deter, and visualize cheating
    - Collusion detection
    - AI detection
  - Add roster importing and grade exporting
  - Implement customizable rubrics for automatic grading

- Implement S.C.O.R.E. into classrooms
  - Allow current Florida Tech CSE classes to use S.C.O.R.E.
  - Collect feedback from professors and students
  - Make improvements based on user suggestions

Motivations

With great progress being made on our project, we are hopeful we can reach our goals. Our aim is to give professors and students an easier method to create/submit assignments.

Approach

Canvas Imports and Exports

Professor

- Professors will be able to upload Canvas rosters on S.C.O.R.E..
- Professors will be able to export the S.C.O.R.E. grades in a format accepted by canvas.

Collusion Detection

Professor

- Professors will be able to view every submission’s plagiarism score.
- Professors will be able to view similarities between S.C.O.R.E. submissions.

Generative AI Detection

Professor

- Professors will be able to view similarity scores for every submission compared to generative AI output.

Rubrics

Professors

- Professors will be able to create rubrics that have custom point systems for completion, test cases, late scores, and alike.
- Assignments will be automatically graded based on rubric criteria.

Students

- Students will be able to view rubrics for their assignments.

Novel features/functionalities

Automated rubric based grading

While the previous edition of the application did allow for the creation of assignments, there were no capabilities for adding rubrics or grading scales to account for point reductions, such as late penalties, or point allocation, such as for run time. Other web apps, Kattis for example, also do not allow this feature nor the ability to create custom assignments and test cases.

AI and collusion detection visualization

The previous edition of the application and similar code problem websites do not check answers for similarities to other submissions or to AI. Additionally, they lack efficient visualization of these detections for an educator.

Algorithms and Tools: Potentially Useful Algorithms and Software Tools

- COPS (Code Originality and Plagiarism System) for detecting similarity and potential collusion between student code submissions.
- Python for backend analysis, file processing, and detection logic.
- Flask for API endpoints that connect grading, detection tools, and the web interface.
- React for dynamically displaying grades, similarity results, and AI detection data.
- Google Cloud Run for scalable deployment and classroom usage.
- Firestore and Cloud Storage for managing submission files, grading data, and metadata.
- Git/GitHub for version control and team collaboration.
- Python-based feature extraction for source code analysis
- Flask API endpoint for AI probability scoring

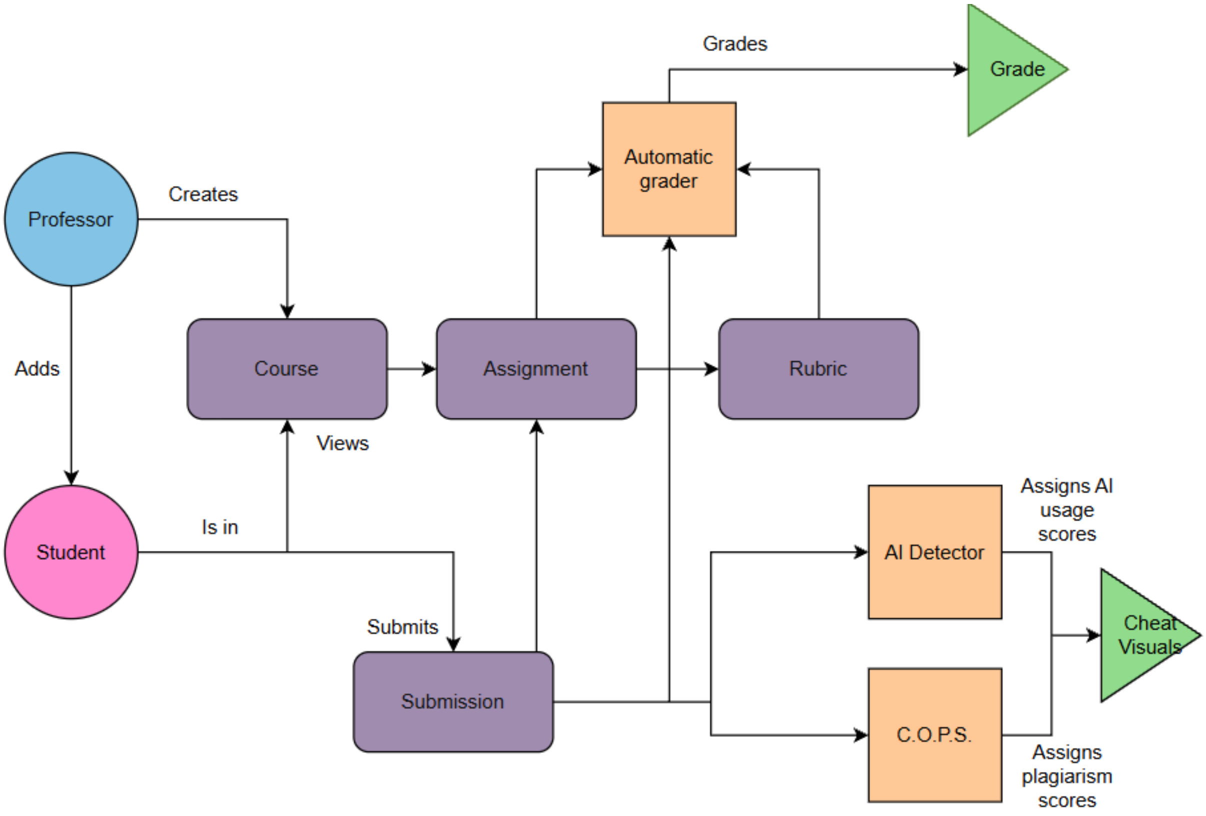
Technical Challenges

A major technical challenge is accurately detecting plagiarism, collusion, and AI usage without producing misleading results. Student code often differs in formatting and style, which makes it difficult to distinguish between legitimate similarities and actual copying.

- Designing plagiarism and AI detection systems that minimize false positives while still identifying meaningful similarities in student code.
- Integrating multiple features (autograding, rubrics, AI detection, and COPS) into a single, stable platform.
- Visualizing similarity and detection results in a clear and useful manner for instructors.
- Ensuring the system scales well for large classes while maintaining performance and data security.

Design

System Architecture Diagram



Evaluation

Speed:

During assignment submission in the student view, we will measure how long it takes for the submission to go through. This includes the time needed to upload files and run automated test cases. Ensuring fast, reliable submission times is crucial during larger classes and during peak submission periods near deadlines.

Accuracy:

The accuracy will be measured by creating a sample assignment, test cases, and a rubric, along with sample code to submit to that assignment. Then, we ensure that the code passes the correct number of test cases and that the auto grade meets the requirements in the rubric. Additionally, we will compare automated grading results with manually graded submissions to confirm their consistency and correctness.

Reliability:

Measured by repeatedly submitting assignments under various conditions and ensuring that the system has consistent performance and accurate results. This includes submitting multiple assignments simultaneously to check for server stability, such as running test cases, applying rubrics, and recording grades without errors or interruptions. The system should be able to operate dependably across different workloads and classroom environments.

User Demo:

Have the professor log in to the SCORE 2.0 platform, create a class, import a roster full of students for that class, and create an assignment with a rubric and test cases. Then, have any students on the roster log in to SCORE 2.0 to access the class and the newly published assignment. That student will submit the code, view the pass/fail test cases, and see the grade based on the rubric. Upon student's submission, the professor will analyze the similarity between AI detection and collusion detection.

Progress Summary

Module/feature	Completion %	To do
Roster Importing	50%	Force Canvas style spreadsheets for importing

Grade Exporting	0%	Create an export button and subsystem to export a csv in the Canvas gradebook style
Automatic Rubric Based Grading	50%	Connect the rubric system to the auto grader
AI Detection	60%	Improve accuracy, integrate results into professor dashboard
Collusion Detection	50%	Needs to be implemented into the submission sections so it can automatically pull and detect.
COPS	50%	Create a matrix and cluster algorithm for the COPS program to be able to be read and visualized

Milestone 4 (Feb 23):

- Complete Automatic Grading Rubric
- Complete Google Cloud Run Hosting
- Complete Importing Roster
- Integrate AI detection results into the submission workflow
- Evaluate AI detection accuracy using sample student submissions
- Connect AI detection output to the web interface for professor review
- Create a Cluster Algorithm for COPS with visualization

Milestone 5 (Mar 30):

- Refine AI detection model based on testing results
- Improve interpretability of AI probability scores for instructors
- Release SCORE 2.0 into classrooms
- Collect feedback from users
- Address reliability issues (multiple users at once, large data, security breaches)
- Conduct evaluation and analyze results
- Create poster for Senior Design Showcase
- Complete COPS integration

Milestone 6 (Apr 20):

- Finalize AI detection integration into the complete system

- Conduct evaluation and analyze results
- Create user manual/developer manual
- Create demo video
- Make any final touches

Task Matrix for Milestone 4

Task	Dorothy	Patrick	Shamik	Rak
Rubric Autograder Completion	100%	0%	0%	0%
Complete Google Cloud Run Hosting	100%	0%	0%	0%
Import Roster Completion	0%	0%	100%	0%
AI Detection Integration & Testing	0%	0%	0%	100%
Complete COPS Matrix	0%	100%	0%	0%

Approval from Faculty Advisor

"I have discussed with the team and approved this project plan. I will evaluate the progress and assign a grade for each of the three milestones."

Signature: ________________________________ Date: ________