Student Code Online Review and Evaluation 2.0

TEAM: SHAMIK BERA, DOROTHY AMMONS, PATRICK KELLY, RAK ALSHARIF
ADVISOR/CLIENT: RAGHUVEER MOHAN

Table of Contents

Goals

Increase the capabilities of the current S.C.O.R.E. application

- Add collusion detection and AI detection to catch cheating and plagiarism
- Add roster importing and grading exporting
- Implement customizable rubrics for automated grading

Implement S.C.O.R.E into classrooms

- Allow current FIT CSE classes to use S.C.O.R.E
- Collect feedback from professors and students
- Make improvements based on suggestions

Motivations

Current progress

- Great progress has been made in our project
- We are on track to reach our goals

Our target

- Give professors and students an easier method to submit or create assignments

Approaches

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.

Approaches (P2)

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

Automated rubric based grading

- Custom rubrics that are used in auto-grading assignments

AI and collusion detection visualization

- Submissions that are tested for % AI used
- Submissions that are compared for collusion amongst each other
- Visuals to easily display these representations

Tools

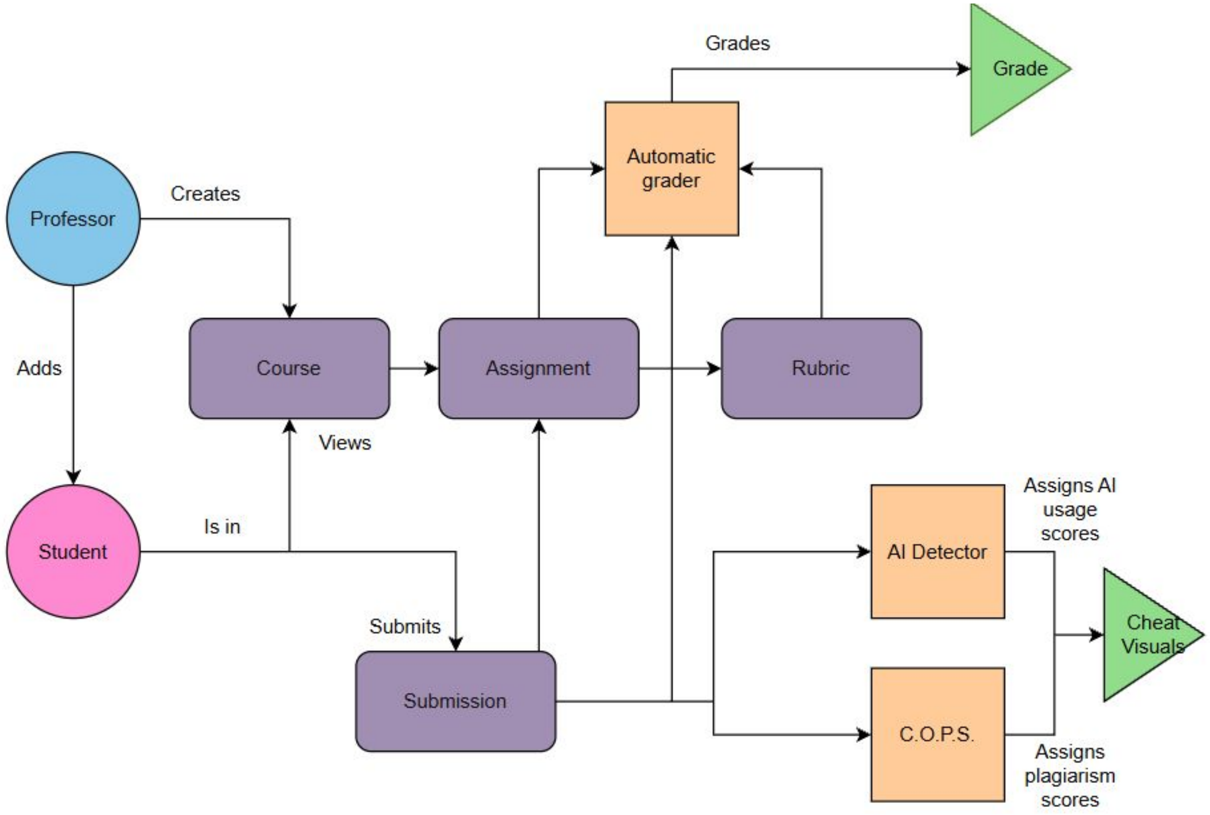
- COPS (Code Originality and Plagiarism System) for detecting similarity and potential collusion between student code submissions.
- Python for backend analysis, file processing, and detection logic.
- Flask for API endpoints that connect grading, detection tools, and the web interface.
- React for dynamically 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

- 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



Evaluation

- Speed
    - Measure how long file uploading is
    - Measure how long automated testing/grading is
- Accuracy
    - Ensure the autograder is grading test cases for submissions correctly
    - Ensure all rubric areas are accounting for the grade correctly
- Reliability
    - Test server capabilities in terms of multiple file submissions or exports at once
    - Test security protocols, including the deletion of data, data leaks, and system break-ins
- User Demo
    - Have professor(s) use the web app to create a course and assignment
    - Have students use the web app to make submissions and receive grades/feedback

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
COPS	50%	Create a matrix and cluster algorithm for the COPS program to be able to be read and visualized

Milestone 4

- 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

- 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

- 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%

Questions?