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

- Integrate the application with Canvas
- Add detection for cheating and plagiarism
- Improve the shell client for users off campus

Classroom deployment

- Allow current FIT CSE classes to use the web application
- Collect feedback from professors and students
- Make improvements based on suggestions

Motivations

With the current S.C.O.R.E. application, professors are unable to

- Enter grades directly to Canvas
- Check for plagiarism, AI usage, or collusion
- Add custom rubrics for grade penalties, such as late scores
- Access the platform with institutional standard security

Students are unable to

- Use the shell client off campus
- Access the platform with institutional standard security

Approach

Canvas Integration

Professor

- View Canvas rosters on S.C.O.R.E..
- Submit the S.C.O.R.E. grades and feedback directly to canvas
- Apply custom rubrics

Approach [Cont.]

Stanford MOSS Integration

Professor

- View every submission’s MOSS plagiarism score
- View similarities between S.C.O.R.E. submissions

Generative AI and Plagiarism Detection

Professor

- View similarity scores for every submission compared to generative AI
- View plagiarism scores

Approach [Cont.]

Security Authentication

All Users

- Access the platform through TRACKS

Shell client

Students

- Access the shell client off campus

Novel features/functionalities

Semi-automated rubric based grading

- The current web application and those similar do not allow for custom rubrics

Plagiarism, AI, and collusion detection visualization

- Kattis and other program problem platforms do not check for the use of AI, plagiarism, or similarities between submissions
- Very few applications, if any, present similarity scores for large volumes of submissions in a way that is both efficient and easy to interpret

Potential Tools

Full Stack:
- MongoDB
- Express
- React
- Node JS

API:
- Canvas
- MOSS
- TRACKS

Languages:
- Rust
- Python

Other third party software tools:
- Django: Front end
- AWS Cloud Services
- SFTP
- OAuth

Technical Challenges

- MOSS API integration
- Clustering algorithms for visualizing MOSS scores
- Canvas integration

Milestones 1, 2, and 3

Milestone 1 (Sept 29):

- Meet with previous team about their work for their project.
- Understand the current S.C.O.R.E application and their tools used.
- Research and compare new tools for MOSS API
- Meet with Dr. White to discuss clustering and visualization techniques
- Create a requirement document, design document, and test plan

Milestone 2 (Oct 27):

- Test MOSS AI and similarity detection
- Understand and test clustering algorithms
- Draft and test matrix views for visualizing MOSS data

Milestone 1, 2, and 3 [Cont.]

Milestone 3 (Nov 24):

- Implement the MOSS matrix views in S.C.O.R.E.
- Add TRACKS to the S.C.O.R.E authentication
- Integrate Canvas classes, rosters, assignments, and rubrics into S.C.O.R.E.
- Implement S.C.O.R.E. grades into Canvas
- Test those integrated features

Task Matrix for Milestone 1

Task	Dorothy	Patrick	Shamik	Rak
Familiarize with previous project	25%	25%	25%	25%
Research the old tools	25%	25%	25%	25%
Research new tools	5%; Clustering Research with White 20%	25%	5%; Clustering Research with White 20%	25%
Requirement Document	Finalize 45%	Finalize 45%	Draft 5%	Draft 5%
Design Document	Models 20%	Models 20%	Draft 15%	Models 45%
Test Plan	15%	15%	45%	25%