Student Code Online Review and Evaluation 2.0

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

Table of Contents

- Milestone 1
- Milestone 1 - Completion Matrix
- Software Testing Plan
- Software Specification Requirements
- Software Design Document
- Milestone 2 - Task Matrix

Milestone 1

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

Milestone 1 - Completion Matrix

Task	Completion %	Dorothy	Patrick	Shamik	Rak	To do
1. Investigate new tools	100%	60%	40%	0%	0%	
2. Investigate old tools	70%	50%	20%	0%	0%	Familiarize ourselves with rust and MongoDB
3. Investigate current system	100%	70%	20%	10%	0%	
4. Requirement Document	100%	80%	0%	20%	0%	
5. Test Plan	100%	5%	5%	90%	0%	
6. Design Document	100%	95%	0%	5%	0%	

Technical Tools

Flask -

Frontend/Backend

Controls requests and deliverables

Firebase -

Cloud database containing user data, submissions data, assignments data, etc.

Cloud Storage -

Cloud space to hold program files

Technical Tools [Cont]

Google Cloud Run -

Always free version

Hosts website and processes HTTPS connections

CLI Client -

Processes commands from the terminal to the Flask API (in the Google Cloud Run Container) via HTTPS

Technical Challenges

Resolved -

Canvas API

- Instead, we will allow imports of CSV files from Canvas for rosters and export CSV files for grades to be uploaded to Canvas

Unresolved -

MOSS API

- We are waiting on access to the API to begin working with it

Clustering Algorithms

- We have a better understanding of how we want our MOSS scores to be visualized but have yet to work with Professor White on determining a good algorithm

Software Requirements Specifications

Functional Requirements

- Import Rosters
  - Upload a CSV from a Canvas roster to add all the student names to the roster of the created SCORE(2.0) class
- Export Grades
  - Professors export student grades for a particular assignment to CSV file, for upload to Canvas
- MOSS Similarity Detection
  - A button that can run the MOSS API across submissions and set similarity score thresholds
  - A matrix will all the similarity detections between students will be displayed or available for download and a cluster graph will be generated
- AI Detection
  - Probability of each submission generated by AI is predicted by hard coded LLM and those above the selected threshold will be displayed in a table
- Custom Rubrics
  - Each test case is worth a set number of points
  - Points can be dedicated for runtime, compilation, and attempt out of a selected total points
  - Points can be deducted for late submissions

Interface Requirements

HTTPS

- All users can connect with command line operations to interact with SCORE 2.0 platform
- Students can log into SCORE(2.0) through terminal to navigate classes and assignments, submit their code, and receive feedback
- Professors can also log into SCORE(2.0) through terminal to upload rosters, export grades, and add or remove assignments and classes

Web App

- SCORE 2.0 brings all changes to the web application relative to professor’s views and functionalities
- Professors have the ability to click on “detect similarities” and “detect AI” button along with exporting grades and creating rosters

Security Requirements

- User Authentication
  - Has to be authenticated through Google OAuth which uses TRACKS
- CLI Connection
  - File that connects commands from the terminal to the Flask system via HTTPS
- Contarization
  - Ensures that code will run in isolated containers to prevent interfering with main server processes
- Data Deletion
  - Removing assignment or class deletes all data related to submissions or rosters

Software Testing Plan

Functional Test

- Covers all the functional requirements with test cases in details
- Each test case demonstrates the professor of using SCORE 2.0 application through the terminal via HTTPS connection and web application
- Test cases also shows scenarios where the professors puts an incorrect input and it would display an error message with rejections from the system

User Test

- HTTPS Connection

  - The professors log into the system through terminal where the professors can select, add, and remove a class, import roasters, and export grades in a CSV format.

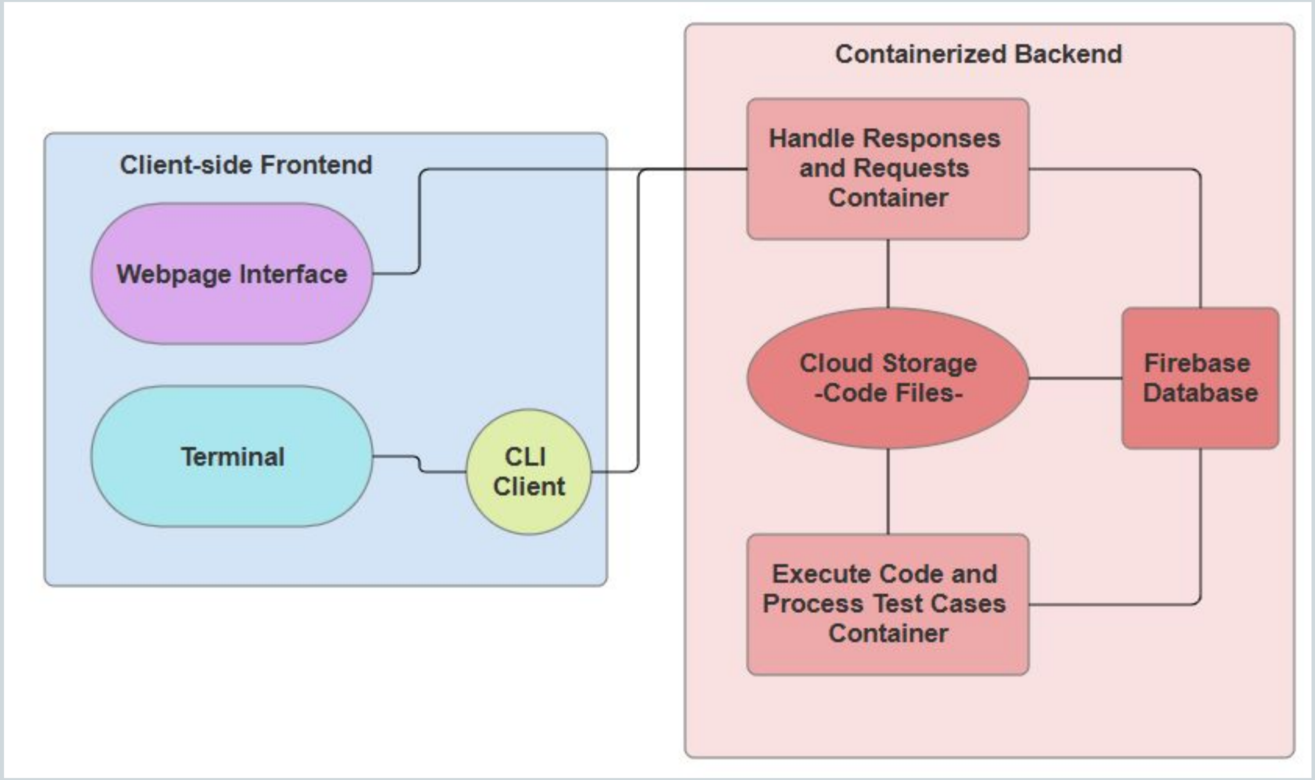
  - The students also log into the system through terminal to open the existing class, open up the posted assignments and submit their code files with test cases and feedback.

- Web Application

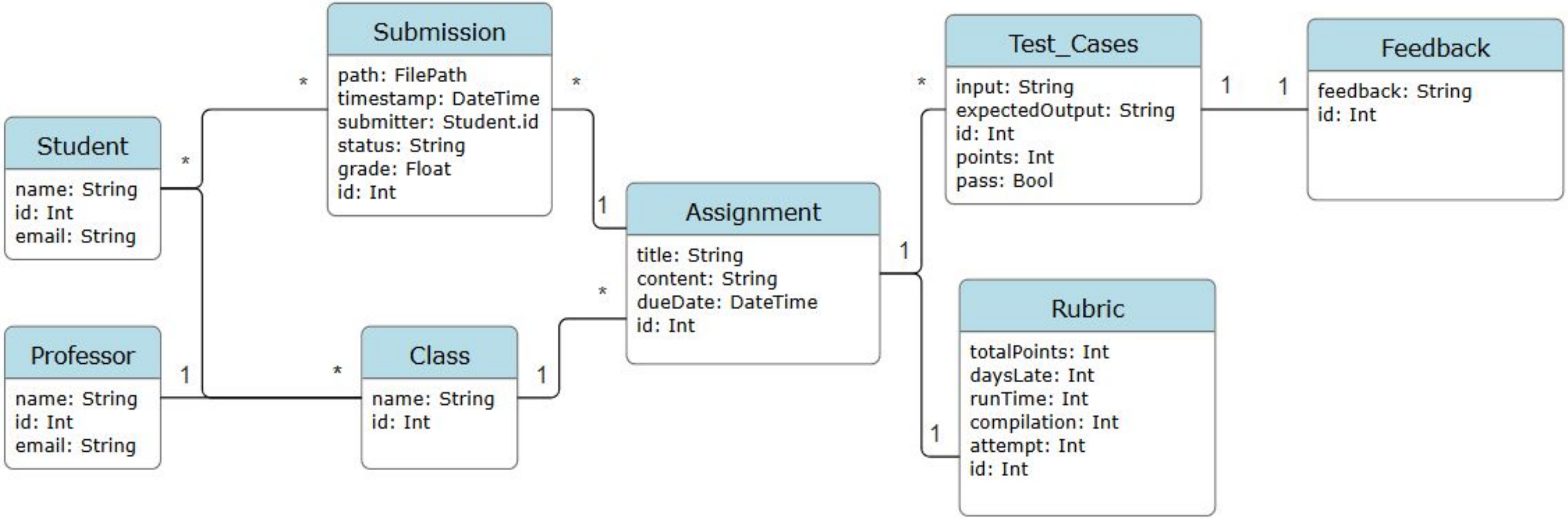
  - The professors log into the platform to import rosters on the class page and create rubrics on assignments, detect MOSS similarities and detect AI on the students' submissions.

Software Design Document

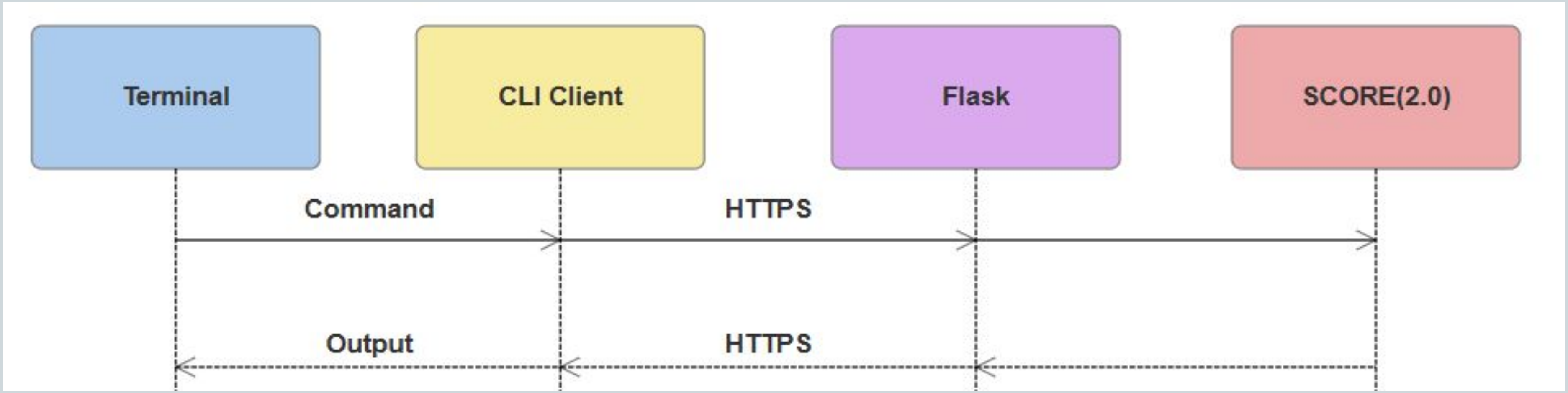
System Architecture


Containerized Backend
Client-side Frontend
Webpage Interface
Terminal
CLI Client
Handle Responses and Requests Container
Cloud Storage -Code Files-
Firebase Database
Execute Code and Process Test Cases Container

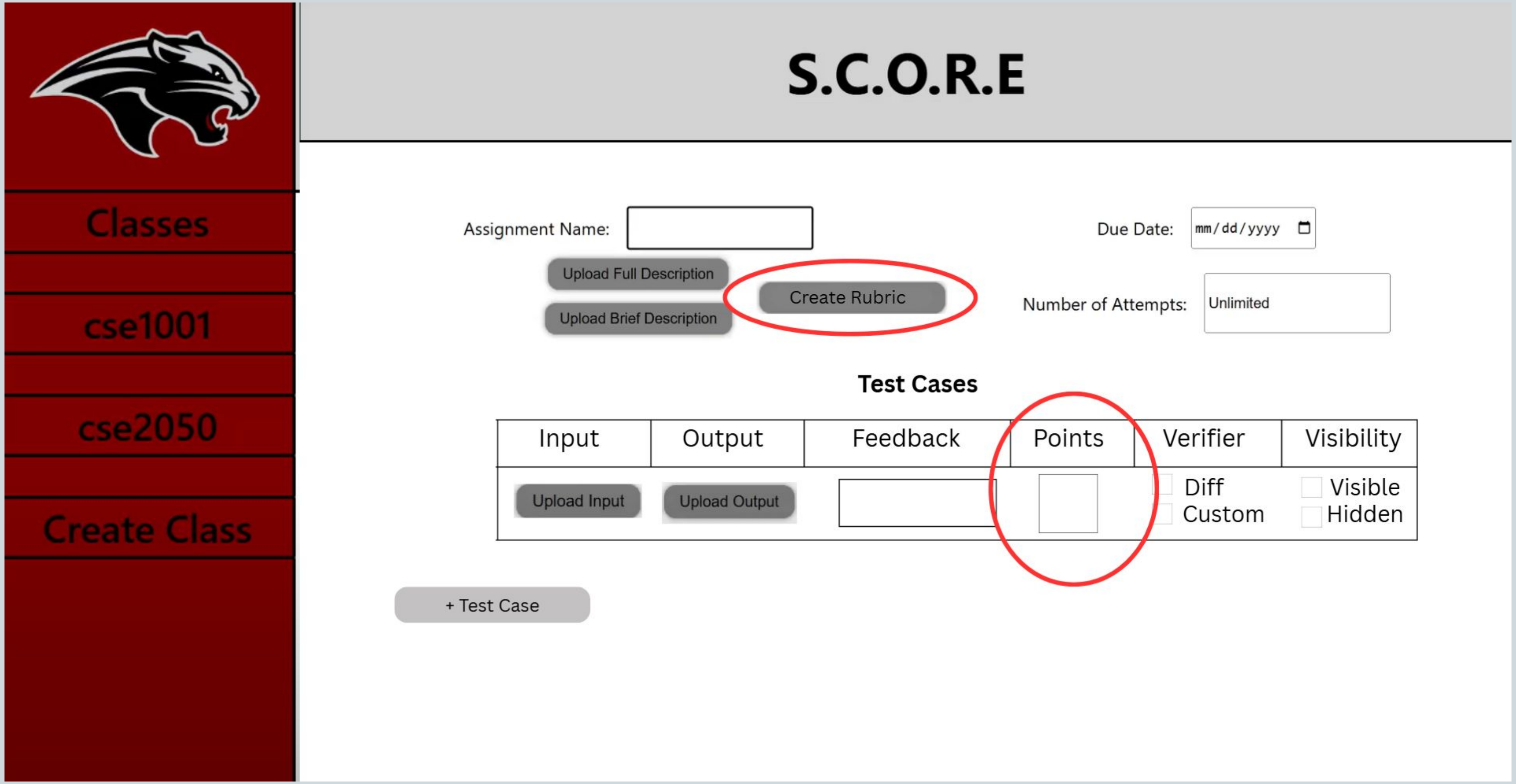
UML - Generalized Database



Terminal-Side System Architecture



Mockup - Rubric Addition



Assignment Name:

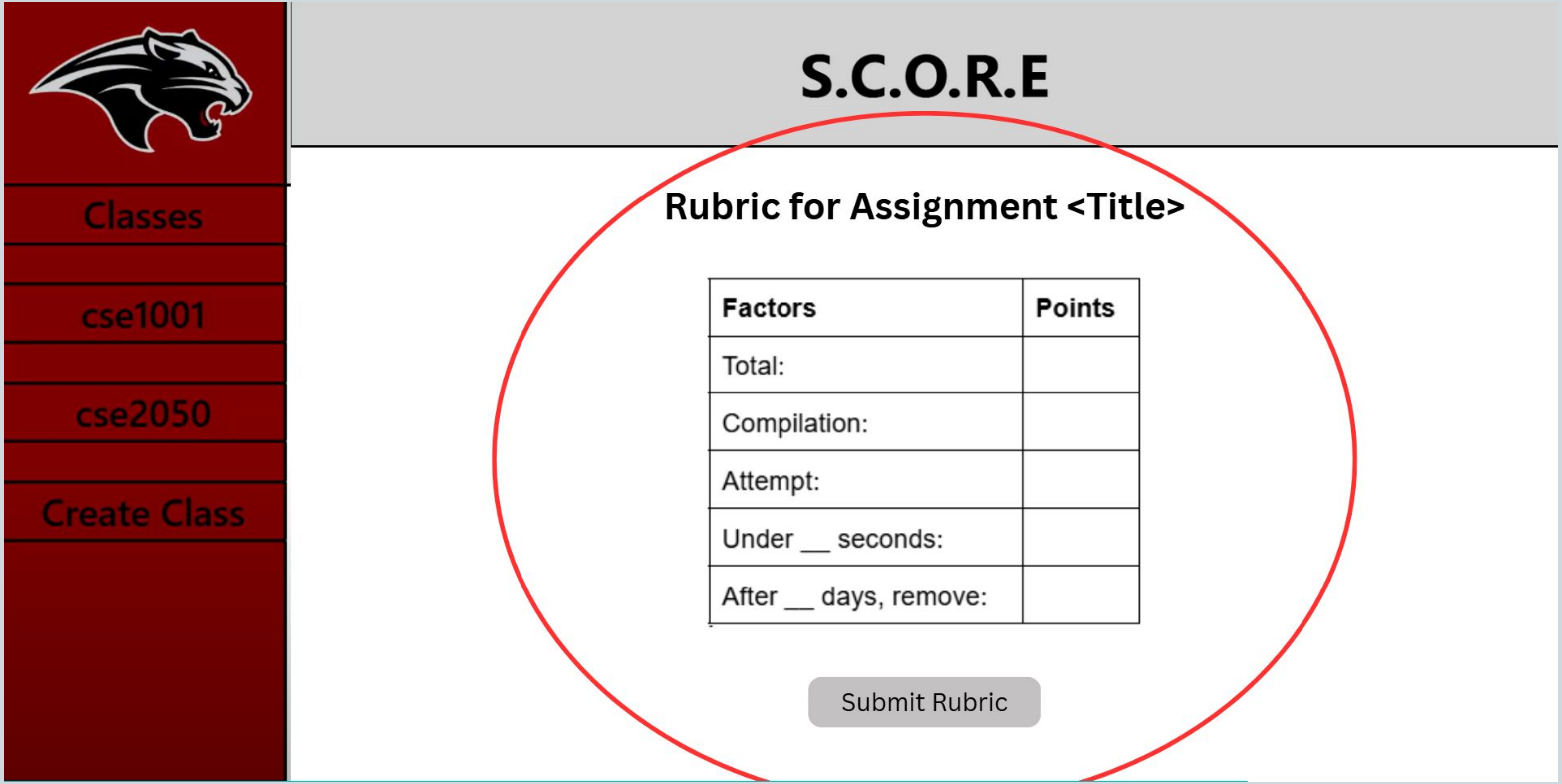
Due Date: mm/dd/yyyy

Number of Attempts: Unlimited

Test Cases

Input	Output	Feedback	Points	Verifier	Visibility
Upload Input	Upload Output			Diff Custom	Visible Hidden

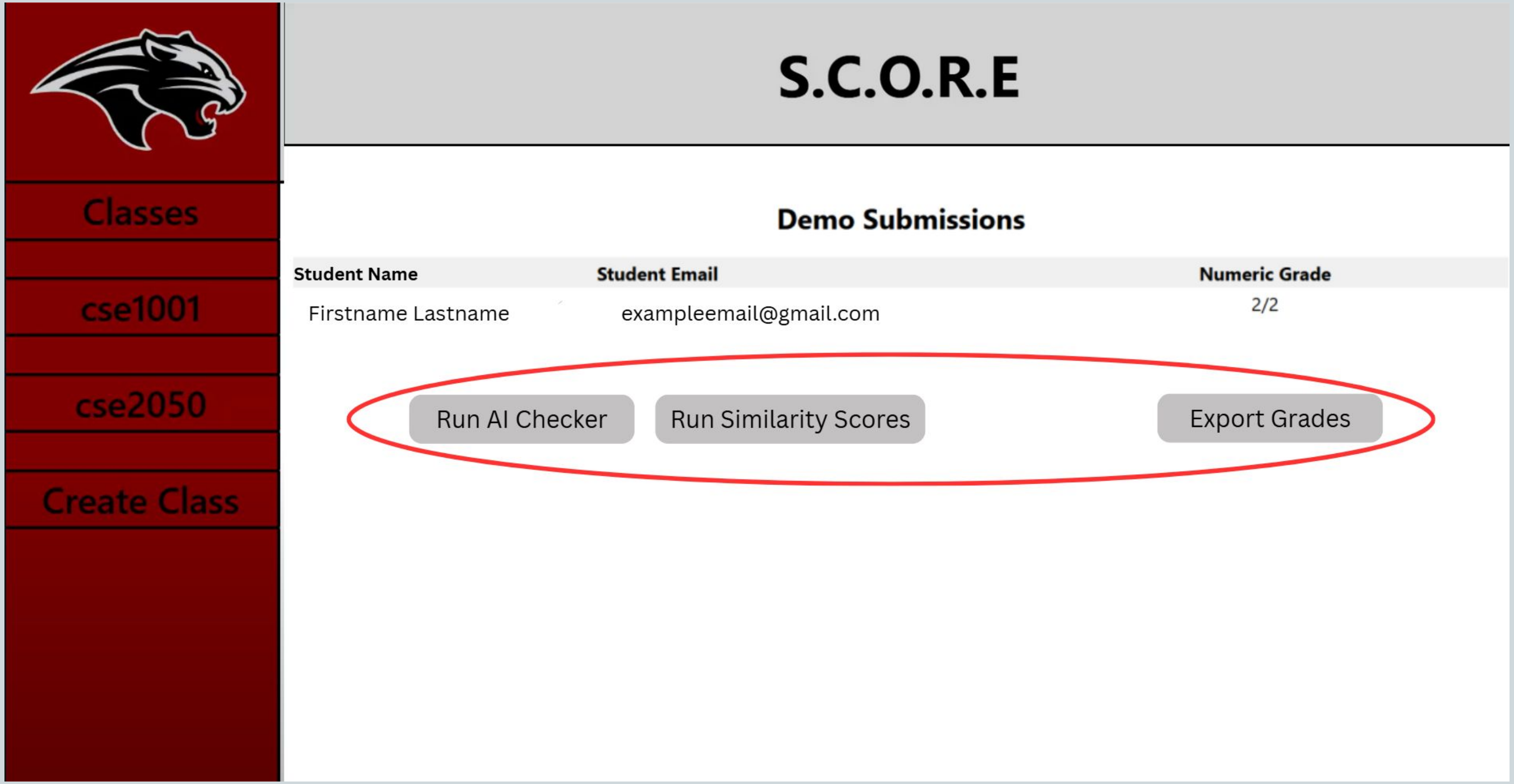
Mockup - Rubric Addition Pt.2



Rubric for Assignment <Title>

Factors	Points
Total:	
Compilation:	
Attempt:	
Under __ seconds:	
After __ days, remove:	

Mockup - Detect AI, Detect Collusion, Export Grades Features

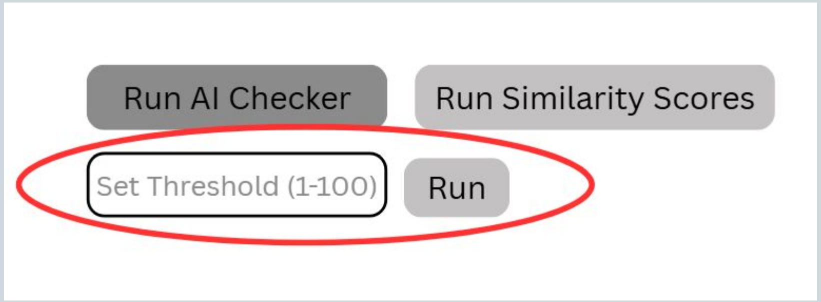


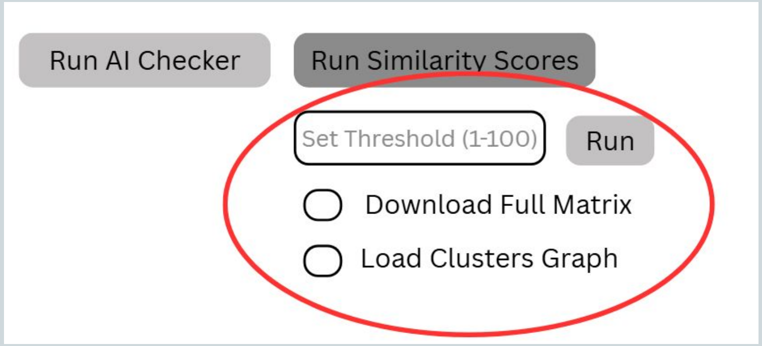
S.C.O.R.E

Demo Submissions

Student Name	Student Email	Numeric Grade
Firstname Lastname	exampleemail@gmail.com	2/2

Mockup - Detect AI, Detect Collusion, Export Grades Features Pt.2





Milestone 2 - Task Matrix

Task	Dorothy	Patrick	Shamik	Rak
1. Replace frontend/backend with Flask and MySQL	100%	0%	0%	0%
2. Replace rust server with Python	0%	0%	100%	0%
3. Add MOSS page to website without functionality	0%	0%	0%	100%
4. Test MOSS detections	0%	100%	0%	0%
5. Determine and test clustering algorithms	25%	25%	25%	25%

Questions?