Student Code Online Review and Evaluation 2.0

Team: Shamik Bera, Dorothy Ammons, Patrick Kelly, Rak Alsharif

Advisor/Client: Raghuveer Mohan

Table of Content

Milestone 2

- Replace the backend with Flask and Firebase
- Create the CLI Client file to interact with the Google Cloud Run server from the command line terminal
- Create and test LLM for AI detections
- Create visuals for AI detection
- Build and test MOSS integration

Milestone 2 – Completion Matrix

Task	Dorothy	Patrick	Shamik	Rak	To Do
1. Replace frontend/backend with Flask and Firestore	80%	0%	0%	0%	Finish endpoints for processing test cases
2. Replace rust server with Python	0%	10%	70%	0%	Create the CLI Client file
3. Add AI detection page to website without functionality	0%	0%	0%	5%	Add the buttons and page(s) for the AI detection
4. Create and test LLM for AI detections	0%	0%	0%	90%	Continue refining accuracy and model evaluation
5. Create visuals for AI detections	0%	0%	0%	30%	Create visuals from the LLM results
6. Built and tested MOSS integration with Matrix	0%	100%	0%	0%	Integrate into grading pipeline for Milestone 3 and connect to frontend for visualization

Replacing the backend

This task involved replacing the Node.js and MongoDB backend files with Flask and Firestore.

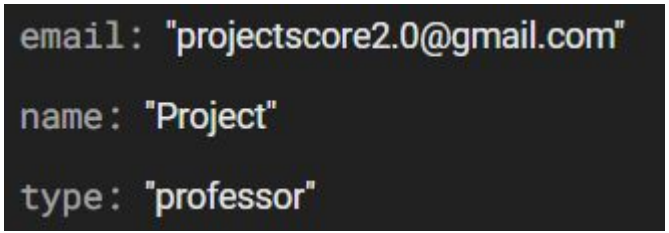
- Created the Firestore database and bucket
- Replaced all of the endpoints with Flask endpoints
- Connected the endpoints to the database
- Kept the file structure and interaction with React + Vite frontend the same
- Modified some of the front end files to save and send data in formats that worked more efficiently with Firestore

Replacing the backend (DEMO)

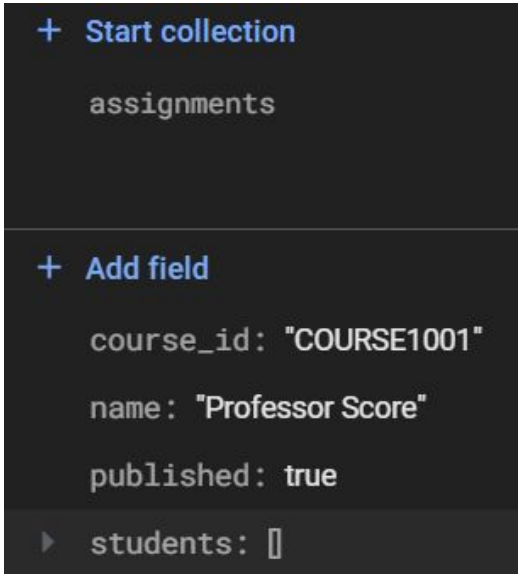
Firestore Database

The Firestore database currently stores users and courses separately, this is for debugging purposes only, so that each one of our group members can interact with every course and test collaboratively.

Users



Courses



Firestore Database [Cont]

Assignments

Test cases

Replacing Rust server with Python

- Converted the server in Rust into Python with Flask
- REST endpoints are being used instead of listening for raw TCP connections and parsing custom text commands.
- On port 12345, it uses Flask web server instead of the TcpListener.
- Connected the commands to the Backend

LLM for Detecting AI Usage

- Developed and tested the initial LLM-based AI detection module for S.C.O.R.E. (2.0).
- Built a Python model that analyzes code submissions and predicts how likely they are AI-generated.
- Implemented feature extraction functions that measure comment ratio, line length, indentation depth, and AI-related patterns.

LLM for Detecting AI Usage [Cont]

- Created and tested a Flask API endpoint (/detect/ai) to return probability and label (“AI” or “Human”).
- Prepared data structure and output format for future web integration and visualization.

MOSS Similarity Detection Prototype

Goal: Goal: Add a system that can compare student submissions and detect code similarities automatically.

- Built a Python tool that checks similarity between code files and creates a results matrix.
- Added a backend route (/api/moss/demo) so the website can request and show the results
- Tested the System with sample files which generated working percentage results

Outcome: The backend can now generate and and return similarity data that will be implemented into the main grading system

Moss Similarity Matrix Demo


```
{
  "matrix": [
    [
      0.0,
      68.0,
      64.2
    ],
    [
      68.0,
      0.0,
      58.5
    ],
    [
      64.2,
      58.5,
      0.0
    ]
  ],
  "students": [
    "alice.py",
    "bob.py",
    "carol.py"
  ]
}
```


- The demo output lists each file (*alice.py*, *bob.py*, *carol.py*) and a matrix of percentage values.
- Each number shows how similar one file's code is to another.
    - For example, *alice.py* and *bob.py* share 68% similarity, while *bob.py* and *carol.py* share 58.5%.
- A value of 0.0 means it's comparing the same file to itself.

Integration and Next Steps

Integration Work:

- Connected the MOSS tool into the backend structure and added a health check route for testing.
- Linked it with existing AutoTest and Auto Feedback setup for smoother future use.
- Verified all routes with live curl testing to confirm the backend is functioning properly.

Next Steps:

- Connect to the official MOSS API for real comparisons.
- Store similarity data in the database for instructors to review.
- Add a simple results page in the frontend to display the similarity table.

Milestone 3 – Task Matrix

Task	Dorothy	Patrick	Shamik	Rak
1. Finalize backend and databases	100%	0%	0%	0%
2. Set up hosting with Google Cloud Run	100%	0%	0%	0%
3. Add the LLM for AI detection to the web application	0%	0%	0%	100%
4. Add the MOSS functionality to the web application	0%	100%	0%	0%
5. Add the rubric page and functionality	25%	25%	25%	25%
6. Add the import functionality for grades	0%	0%	100%	0%
7. Add the export functionality for grades	0%	0%	100%	0%

Questions?