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 3

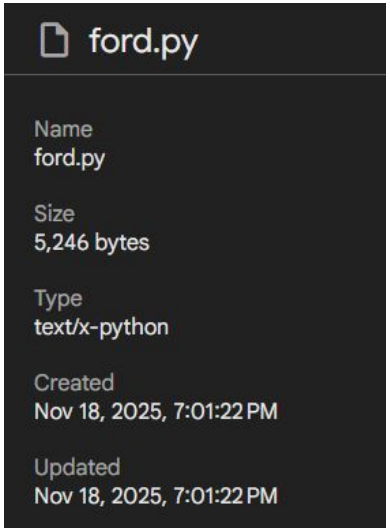
- Implement MOSS functionality into S.C.O.R.E. web application
- Implement LLM for AI detection into S.C.O.R.E. web application
- Set up Google Cloud Run to S.C.O.R.E. authentication
- Integrate import functionality for rosters into S.C.O.R.E.
- Integrate rubric functionality into S.C.O.R.E.
- Implement export functionality for grades into S.C.O.R.E.

Milestone 3 – Completion Matrix

Task	Dorothy	Patrick	Shamik	Rak	To Do
1. Finalize backend and databases	100%	0%	0%	0%	
2. Set up hosting with Google Cloud Run	0%	0%	0%	0%	Set up hosting
3. Add the LLM for AI detection to the web application	0%	0%	0%	50%	Add functionality to front end
4. Add the MOSS functionality to the web application	0%	70%	0%	0%	Connect MOSS to the students' assignment submissions
5. Add the rubric page and functionality	50%	0%	0%	0%	Fix database storage, connect to autotest files
6. Add the import functionality for rosters	0%	0%	60%	0%	Connect roster to the created course to see the added students
7. Add the export functionality for grades	0%	0%	0%	0%	Coincides with rubric functionality, add export CSV option for student grades after

Database fixes

- Fixed some of the retrieving and sending from/to the database in the endpoints
- Stored all large files (input files, export files, student submissions.. etc) in the firestore bucket

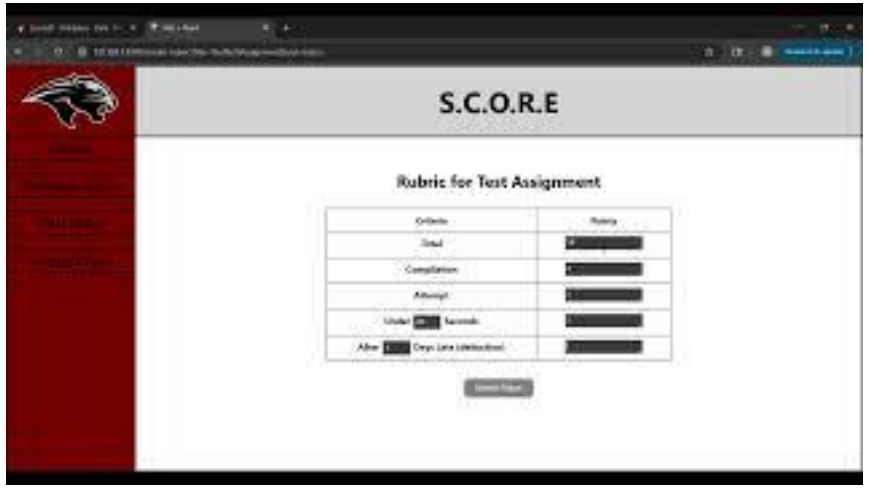


Rubric Page

- Added a “Points” input box for each test case
- Added a button for creating rubric within an assignment
- Added a rubric page
    - Rubric has 5 columns
        - Total: Total number of points for the assignment
        - Compilation: Points granted for successful code compilation
        - Attempt: Points granted for submitting anything
        - Runtime: “Under _ Seconds” Points granted for if the runtime is under a certain amount of time
        - Late Penalty: “After _ Days Late (deduction)” Points removed a given amount of days after due date

Rubric Page Demo/ What's left

- Database endpoint is already set up, however, rubrics are attached to assignments. This means an assignment needs to be created before adding the rubric. We need to figure out how to make this possible when rubric creations are completed before assignments are published



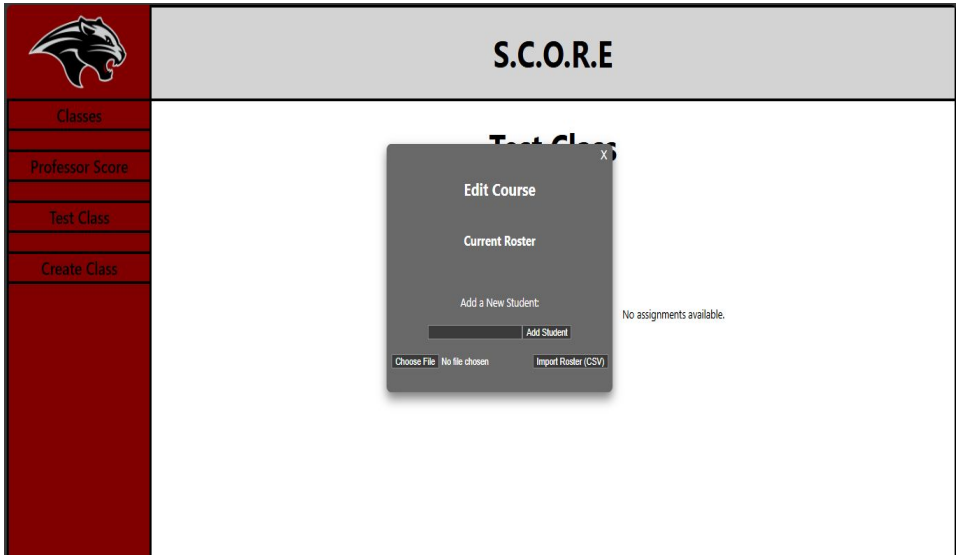
- We then need to connect the rubric inputs to the autotest files in order to get proper scores for student submissions

Roster functionality

- Added a file input that only accepts CSV format.
- Added a button for importing a roster once the CSV file has been selected.
- The header of the CSV contains student’s full name and their email.
- An existing button and input to add new students manually would also be inserted into the imported roster.

Roster functionality Demo/ What's left

- A database endpoint for importing roster is created in the backend, but it is embedded in the course page. Students can be added to the course either one by one or by importing a CSV file containing multiple students, which is done by editing the course. The CSV file must be attached before uploading, and then it is parsed row by row.



- Next, the student's name and email, saved from the database, need to connect to the created SCORE 2.0 classes.

LLM-Based AI Detection

- Integrated multiple pre-trained LLM models into SCORE 2.0 backend
  - RoBERTa-based classifier (HuggingFace)
  - GPT-based probability detector (OpenAI API)
    - `$env:OPENAI_API_KEY = "sk-..." <<<` **Secret Key**
- Cleaned and removed the previous detection modules
- Built a stable ensemble combining both detectors
- Standardized the AI probability output into a unified JSON format
- Added logging + error handling for reliability
- Validated the detection results on multiple student code samples
- Prepared the system for frontend integration in Milestone 4

MOSS Implementation Work

MOSS Integration – Backend Work

- Built a Python similarity analysis module and converted it into a backend route (/api/moss/demo)
- Added directory scanning, file comparison logic, and matrix generation
- Connected the tool into the existing Flask backend structure

API Development & Testing

- Added backend endpoint with proper HTTP methods
- Validated functionality using cURL and local sample submissions
- Fixed routing issues and JSON-return formats

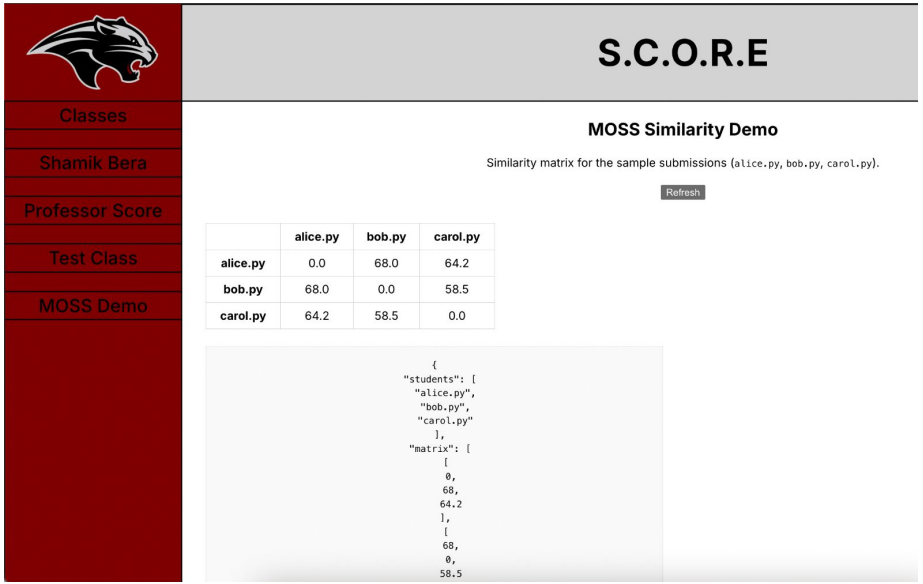
MOSS Frontend & System Integration

Integration With SCORE Web App

- Built a clean UI page for instructors to view similarity results
- Added dynamic table rendering and refresh functionality
- Successfully deployed the page into the compiled frontend bundle

Fixes and System-Level Improvements

- Rebuilt frontend assets using npm run build and linked them to Flask
- Debugged static file serving and fallback routes
- Prepared full-stack flow so future MOSS features can plug directly into assignment pages



Future Steps: Full MOSS Integration With Assignment Submissions

Next Steps – Phase 2 MOSS Integration

- Connect MOSS to real student submissions instead of sample files
- Automate similarity detection whenever students submit code to SCORE
- Store all similarity results in Firestore for instructor review
- Build a dedicated instructor dashboard panel for viewing comparisons

Milestone 4 – Task Matrix

Task	Dorothy	Patrick	Shamik	Rak
1. Fix rubric/assignment database clash	100%	0%	0%	0%
2. Connect rubric to autotest files	50%	0%	50%	0%
3. Add the export functionality for grades	100%	0%	0%	0%
4. Connect roster to the created courses	0%	10%	90%	0%
5. Integrate AI Detection (LLM) into the instructor dashboard	0%	0%	0%	100%
6. Set up hosting with Google Cloud Run	100%	0%	0%	0%
7. Connect MOSS to real student submission	0%	100%	0%	0%

Questions?