Amy cardona

CIS129

05/19/2025

Final Project: PreGrade: Estimate Your Assignment Score Before You Submit

---

Introduction to the Topic

- **Topic:** PreGrade - A Self-Grading Tool for Students
- **Why It Matters:** Students often submit work without knowing how well they met expectations. Waiting for feedback can cause stress and delay improvement.

Problem Identified:

Students often submit assignments without knowing how well they've met the criteria. Waiting for a grade can be stressful, especially if feedback is delayed or unclear. Many students also don't fully understand rubrics or how their work is scored.

- Lack of clarity in rubrics and expectations.
- No immediate way to estimate assignment quality.
- Delayed feedback can lead to repeated mistakes in future assignments.

Existing Solutions

- **Grammarly/QuillBot:** Focus on grammar, not full assignment evaluation.
- **Google Classroom Rubrics:** For teachers, not students.
- **Turnitin:** Gives feedback, but only after submission.

Gap: No tool offers *student-first* grading before submitting work.

Real-World Application

- Helps students write with intent.
- Teaches students to understand rubrics.
- Encourages revision and growth.
- Can be used by tutors, study groups, and writing centers.

Software Design Proposal - PreGrade

- **Goal:** Let students select an assignment type, review the rubric, self-score (or use AI suggestions), and get an estimated grade.
- **Inputs:** Assignment type, uploaded work or written input, rubric selection.
- **Outputs:** Estimated grade, score breakdown, revision tips.

Key Features

- Rubric templates for essays, lab reports, presentations, coding projects, and research papers.
- Self-scoring
- Upload teacher's rubric as a PDF and auto-extract criteria.
- Revision suggestions per rubric category.
- Exportable feedback PDF.
- **Live Grade Widget:** An Optional feature that monitors your assignment tab and gives real-time updates on your current estimated grade as you write. It can be turned on/off

by the user.

Pseudocode

```
# 1. Get user input on assignment type
INPUT assignment_type

# 2. Load appropriate rubric template
LOAD corresponding_rubric(assignment_type)

# 3. Handle optional custom rubric upload
IF uploaded_rubric_file:
    # Use OCR to extract grading criteria and their weights
    EXTRACT criteria and weights using OCR
    BUILD custom_rubric()

# 4. Iterate over each grading criterion
FOR each_criteria IN rubric:
    DISPLAY criteria_description
    # Let user self-score OR use AI to generate score
    GET user_score OR auto_score(criteria, user_input)

# 5. Optional: Enable Live Grade Widget
IF live_widget_enabled:
    # Continuously monitor active assignment writing tab
    MONITOR active_assignment_tab()
    # Reanalyze the assignment content in real-time
    CONTINUOUSLY re-analyze text
    # Update the user's current estimated grade
    UPDATE estimated_grade_display()

# 6. Final calculation and feedback
CALCULATE total_score
DISPLAY estimated_grade_and_feedback
```

User Interaction

- **Step 1:** Choose assignment type (essay, lab, project, etc.).

- **Step 2:** Upload or paste assignment.
- **Step 3:** Upload rubric PDF or use a built-in template.
- **Step 4:** View rubric and self-score or use AI help.
- **Step 5:** See estimated grade and suggestions.
- **Step 6:** Optionally activate Live Grade Widget for real-time feedback.
- **Step 7:** Save or export results.

Open Questions/Reflection

- How do we ensure students don't just rely on PreGrade instead of actual review?
- Can rubric extraction from PDF be made accurate across all formats?
- Will the live widget be too distracting or helpful for different writing styles?

Citations

- "Grammarly: Free Writing AI Assistance." *Grammarly*, www.grammarly.com.
- "Use Rubrics for Assignments." *Google for Education Help*, Google, support.google.com/edu/classroom/answer/9335069?hl=en.
- "Turnitin." *Turnitin*, www.turnitin.com.
- "Writing Center." *The University of North Carolina at Chapel Hill*, writingcenter.unc.edu.
- "Use Autograding." *GitHub Docs*, GitHub, https://docs.github.com/en/education/manage-coursework-with-github-classroom/teach-with-github-classroom/use-autograding.
- "Acardona5's Profile." *Codewars*, https://www.codewars.com/users/acardona5.
- "Specs — Gradescope Autograders Documentation." *Gradescope*, https://gradescope-autograders.readthedocs.io/en/latest/specs/.
- "Python Support — Gradescope Autograders Documentation." *Gradescope*, https://gradescope-autograders.readthedocs.io/en/latest/python/.
- *Gradescope Python Autograder Sample Project.* GitHub, https://github.com/gradescope/autograder_samples/tree/master/python/src.

- *Calculator.py - Gradescope Python Example.* GitHub, https://raw.githubusercontent.com/gradescope/autograder_samples/master/python/src/solution/calculator.py.
- Forsythe, George E. *Automatic Machine Grading Programs*. Stanford University, 1964. *Association for Computing Machinery*, https://doi.org/10.1145/800257.808930.

```
# 1. Get user input on assignment type
assignment_type = input("Select assignment type (essay, lab, project,
etc.):")

# 2. Load appropriate rubric template
rubric = load_corresponding_rubric(assignment_type)

# 3. Optional: Handle custom rubric upload
if uploaded_rubric_file:
    # Extract criteria and weights using OCR from uploaded rubric
    criteria, weights =
extract_criteria_and_weights_OCR(uploaded_rubric_file)
    rubric = build_custom_rubric(criteria, weights)

# 4. Iterate over each grading criterion
for criterion in rubric:
    display(criterion.description)
    # Optionally let the user score or use AI evaluation
    score = get_user_score_or_auto_score(criterion, user_input)
    criterion.score = score

# 5. Optional: Enable Live Grade Widget
if live_widget_enabled:
    # Monitor user's assignment editing tab
    while editing_assignment_tab_active():
        assignment_text = capture_current_assignment_text()
        updated_scores = reanalyze_text(assignment_text, rubric)
        update_estimated_grade_display(updated_scores)

# 6. Final calculation and feedback
total_score = calculate_total_score(rubric)
feedback = generate_feedback(rubric)
display(total_score, feedback)
```