Design Document: Rubric Automation & Grading Integration

---

1. Context

We are integrating AI-powered rubric creation, rubric management, and automated grading into the main grading platform.
Legacy AI logic and prompt code is placed in a temporary folder within this repo.
Cursor:

- When AI rubric generation, AI grading, or rubric normalization/prompt logic is required, refer to the temp legacy folder for examples and structure.
- Do **not** copy legacy backend logic verbatim—adapt prompt strings and JSON parsing only.
- If you cannot find relevant logic, **ask me for clarification.**

---

2. Key Things to Remember / Cursor Tips

- **Rubric = Core Object:** Each assignment can have up to 3 rubrics (AI or manual), fully editable and switchable.
- **AI Integration:** Only prompt engineering, parsing, and logic for OpenAI API calls should be adapted from the temp legacy code.
- **Do not copy backend infrastructure or route structure from legacy—use only the working prompt/response sections and helpers.**
- **UI Consistency:** All rubric and grading interfaces must match the main web app's style.
- **Endpoints:** All endpoints should accept any file type, but image-based AI flows are currently proven to work.

- **If Cursor cannot find the referenced logic, it should ask me for the right file, location, or code snippet.**

---

3. Step-by-Step Implementation Plan

Step 1: Add Rubric Panel to Assignment Splash Page

Goal:
Add a right-hand panel for rubric management and creation.

Steps:

- Insert a new panel (right side, above student grades, near files section).
- Show dropdown of all existing rubrics (max 3 per assignment), with “Create AI Rubric” and “Create Manual Rubric” options.
- If a rubric is selected, show its structure (match main app UI).
- Place “Confirm for Assignment” button at the bottom—sets that rubric as active for grading.

Nuances:

- Rubrics must have a non-empty name to be saved.
- Allow delete/rename in dropdown.
- Block creation at 3 rubrics.

Cursor:
No legacy code reference needed for this step—UI logic only.

Step 2: Rubric Creation Workflow (Refer to Legacy Temp Folder for GPT-4 Prompt & Parsing Logic)

Goal:
Enable teachers to generate rubrics via AI or create/edit them manually.

Steps:

- On "Create AI Rubric", prompt for assignment file (image or PDF) and solution file.
- On AI generation, **refer to the temp legacy folder** for prompt structure, image preparation, and JSON parsing.
- **Copy/adapt only the GPT-4 prompt and parsing logic,** not the full backend implementation.
- On "Create Manual Rubric", open a rubric builder UI (add/edit sections, parts, items, point validation).
- After AI rubric is generated, display for review/edit (full manual override allowed).
- Require rubric to be named before saving.

Nuances:

- AI-generated rubrics should auto-validate point totals (see legacy helpers if needed).
- If Cursor cannot find prompt logic or needs more context, **ask me.**

Step 3: Rubric Display & Grading UI Integration

Goal:
Display the selected rubric in the grading modal, under the grade input/score section.

Steps:

- Add rubric viewer to the left column of grading modal (under “Grade Input”).
- Render rubric with collapsible sections/parts, color badges, indents—matching main app UI.
- Interactive: Show which items are satisfied (from AI), allow manual override by checking/unchecking items.

- On rubric change, update score and feedback (require manual Save/Publish).
- Auto-populate feedback box with AI-generated comments (editable).

Nuances:

- Feedback always editable before saving/publishing.
- Only the rubric marked “Confirmed for Assignment” is used for grading/feedback.
- If rubric is edited, must Save before it becomes selectable/active.

Cursor:
No legacy code reference needed here—UI only.

Step 4: Saving, Publishing, and Rubric Management

Goal:
Align rubric and grading save/publish workflow with existing platform logic.

Steps:

- “Save” saves as draft, “Publish” finalizes grades/feedback.
- Limit rubric storage to 3 per assignment; require delete to create a new one.
- Support rubric renaming and deletion from dropdown.

Nuances:

- No auto-save: only explicit Save/Publish.
- All rubric CRUD actions should trigger proper DB updates and UI refresh.

Cursor:
No legacy code reference needed—match main app’s save/publish workflow.

Step 5: AI Grading and Feedback Integration (Refer to Legacy Temp Folder for GPT-4 Grading Prompt & Parsing Logic)

Goal:
Integrate AI grading using the active rubric, and feedback population.

Steps:

- When grading, send student submission and active rubric to backend AI grading endpoint.
- **Refer to the temp legacy folder** for image preparation, GPT-4 grading prompt, JSON response parsing, score, and feedback breakdown.
- **Copy/adapt only the prompt and parsing logic,** not the full backend structure.
- Populate feedback textbox with AI summary (teacher can edit before publish).

Nuances:

- Grading only runs on the active rubric.
- Any rubric changes require re-running AI grading for new student submissions.

Cursor:

- If logic is missing or unclear, **ask me for the code or clarification.**

Step 6: Rubric Normalization and Helpers (Refer to Legacy Temp Folder as Needed)

Goal:
Normalize rubric data structure and point calculation as needed for grading UI and API.

Steps:

- If point normalization logic or helper functions are needed, **refer to the temp legacy folder** for sample implementations.

- Integrate point calculation/check into rubric creation/edit workflow.

Nuances:

- Only bring over helper functions if your existing app does not already handle point validation/normalization.

Cursor:

- Reference only as needed, and always check if a similar utility already exists in the current codebase.

---

4. Final Reminders for Cursor

- **Use only prompt and parsing logic from the temp legacy folder—never copy full backend routes or infrastructure.**
- Always prioritize **main LMS platform's database, API, and UI patterns.**
- If you cannot find referenced legacy logic, **pause and ask me for location or code snippet.**
- If simplification is possible, discuss first.
- Endpoints should be extensible for future file types.
-