
Context Paragraph

We are building a modern assignment submission and grading portal for teachers, fully integrated with our LMS and gradebook. The goal is to allow teachers to view all students in a course, see each student's latest submission for an assignment, preview and download submissions, mark assignments as late/on-time (with auto-logic and manual override), assign per-student due dates, enter scores and feedback, and save/publish grades to the gradebook. All data and endpoints should remain consistent with our existing gradebook and assignment infrastructure. Submission history is preserved for audit but students only see the most recent result. Save/publish should always be bulk actions, and teachers can edit and republish as needed. We aim for clear, intuitive UX and a scalable backend for future teacher workflows.

General Tips for Cursor

- **Reuse existing gradebook endpoints and data models wherever possible for grades, feedback, late status, and student-assignment relations—this maintains a single source of truth.**
- **Keep UI/UX consistent with the rest of the platform (table styling, modals, button patterns, etc).**
- **Support bulk updates for grades/feedback/save/publish actions (single API call).**
- **Protect audit trail: all student submissions are permanent; no teacher delete.**
- **Ask for clarification if an endpoint, model, or frontend component is missing—generate it in the same structure as previous gradebook/assignment code.**

- **Late/on-time status should be determined by default due date (or student override), but is also manually adjustable by teacher.**

- **Teachers can enter grades/feedback even if a student hasn't submitted; students only see data when "Publish" is pressed.**

- **Use the same file preview/iframe modal system as elsewhere on the site.**

- **Ensure all permission checks: only teachers can see or modify grading table.**

Steps for Cursor

Step 1: Build Teacher Assignment Grading Table

Goal:
Create a responsive, user-friendly table on the assignment splash page (teacher view) showing all students, their submission status, lateness, grades, and feedback for the assignment.

To-dos:

- **Display all students in the course, even those with no submissions ("No Submission" in the table).**

- **Add sorting/filtering for: only submitted, first/last name, and late/on-time.**

- **"Submission" column:**

    - **Show the most recent submission if it exists.**

    - **Clicking “View All Submissions” opens a modal with list (date/time) and iframe preview; allow download of each.**

- **“Late/On-Time” column:**
    - **Auto-calculate using due date (global or per-student override) and most recent submission.**
    - **Allow teacher to manually override; changes persist even after further submissions, unless due date is changed again.**
- **“Due Date/Time” column:**
    - **Teacher can override due date/time for each student; if not, default to assignment due date.**
- **“Score” and “Feedback” columns:**
    - **Teachers can enter/edit these anytime, even without a submission.**
- **“Save Grades” and “Publish Grades” buttons (see below).**

Questions for Cursor:

- **What UI/table library (if any) is already in use? Follow existing styling/conventions.**
- **Is there already a component for bulk editable tables?**
- **Confirm the endpoint/data model for per-student due dates—create or extend as needed.**
- **Should late/on-time status calculation live in backend or frontend? (Follow current pattern.)**

Nuances:

- **Make sure changing a student's due date *retroactively* updates lateness for all past submissions for that student.**
- **Don't allow teachers to delete any submission file—show all for audit.**
- **Only the latest submission is visible by default; modal allows access to past submissions.**
- **Ensure save vs. publish logic matches the rest of the gradebook.**

Step 2: Implement Submission Preview Modal

Goal:
Allow teachers to view and download all past submissions from a student for this assignment, in a modal with a list + iframe preview.

To-dos:

- **Modal opens from main grading table (from "View All Submissions" button/icon in submission column).**
- **Show list of all past submissions with date/time.**
- **Clicking an entry previews that submission in the iframe; allow download.**
- **Modal includes "X" to close; highlights most recent submission on open.**

Questions for Cursor:

- **Is there a reusable modal/iframe component already?**

- **What's the best way to fetch all submissions for one student/assignment—reuse endpoint or create a new one if needed?**

Nuances:

- **Modals should be accessible and responsive.**
- **Audit trail: teachers see all, but cannot delete.**

Step 3: Grade/Feedback Entry, Save & Publish

Goal:
Allow teachers to enter/edit grades and feedback, save work-in-progress, and publish when ready.

To-dos:

- **Grades and feedback can be entered for any student (even without submission).**
- **"Save Grades" persists data privately (teachers only); can be used before and after publish.**
- **"Publish Grades" makes grades/feedback visible to students in the gradebook.**
- **After publishing, new edits require another "Publish" to update what students see.**
- **Both save and publish actions should use bulk endpoints.**

Questions for Cursor:

- **Reuse existing gradebook bulk update endpoints for grades, feedback, late status?**

- **Ensure frontend state syncs with backend responses for reliability.**

Nuances:

- **All grades, late status, feedback must stay in sync between assignment grading table and gradebook view.**

- **Publishing is a "snapshot"—students only see last published state.**

Step 4: Filtering, Sorting, and Table Usability

Goal:
Make the grading table easy to navigate for large classes.

To-dos:

- **Add filtering options for: submitted only, on-time/late, first/last name.**

- **Allow sorting on all columns.**

- **Table and modals should remain performant with large student lists.**

Questions for Cursor:

- **What filtering/sorting utilities are already available?**

- **If not, implement scalable methods (e.g., debounce, virtualization for large lists).**

Nuances:

- **Filter/sort state should persist if teacher navigates away and returns.**
- **UX polish: loading states, error handling, success feedback.**

Step 5: Data Consistency and Audit Trail

Goal:
Ensure every action (grade, feedback, lateness, submission) is fully auditable and cannot be deleted/overwritten.

To-dos:

- **Store all submissions permanently; teachers can view/download but not delete.**
- **Keep full history of grade/feedback changes for teacher view (students see only latest published).**
- **Use existing file storage mechanisms for submission files.**

Questions for Cursor:

- **Confirm all models/relations needed for audit/history; generate migration if missing.**

Nuances:

- **Protect audit trail from accidental or malicious edits.**
- **Plan for future reporting or export features if needed.**

Extra Guidance for Cursor

- **For any ambiguity in UI or endpoint, ask the user for clarification or follow the standard pattern already used in gradebook/assignment features.**

- **Keep code modular and add helpful comments.**

- **All new components, endpoints, and models should be consistent with prior work in both structure and naming.**