Step 0: Set Context in Cursor

Paste this summary as your project context:

vbnet
CopyEdit

```
Building an LMS Gradebook MVP. Gradebook is accessed from the main
topbar.
Upon entry, users see a sidebar of courses (like Discussions).
Teachers can manage grade sections (categories/weights), assignments,
student grades, and set late penalties.
Students can view their own grades, weights, and run “What-If”
scenarios.
Backend uses Prisma models and Express API. Frontend uses
Next.js/React in /src/app/gradebook/, with role-specific and shared
components.
All business logic for grade calculation is shared and modular for
future extensibility.
```

Step 1: Schema & Backend Scaffolding

1.1. Update Prisma Schema

- **File:** `prisma/schema.prisma`
    - Add models: `GradeSection`, `Assignment`, `Grade`, update `Course` with `latePenalty`.
    - (Optional: Enum for assignment types for future extensibility.)

1.2. Generate and Migrate

- Run `npx prisma generate` and `npx prisma migrate dev`.

1.3. Scaffold Backend Routes

- **File:** `apps/api/src/routes/gradebook.ts`
    - Set up Express router.
    - Endpoints:
        - `GET /api/gradebook/courses` (user courses)
        - `CRUD /api/gradebook/:courseId/sections`
        - `CRUD /api/gradebook/:courseId/assignments`
        - `GET/POST /api/gradebook/:courseId/grades`
        - `GET/PUT /api/gradebook/:courseId/late-penalty`
        - `GET /api/gradebook/:courseId/students`
    - Add middleware for auth/permissions.
    - Stub handlers initially—return mock data or success for now.

Step 2: Frontend Scaffolding

2.1. Setup Next.js Route and Context

- **Folder:** `apps/web/src/app/gradebook/`
    - Create `page.tsx` for Gradebook entry point.
    - Create `GradebookContext.tsx` for shared state.
    - Add `types.ts` for TypeScript interfaces (Course, GradeSection, Assignment, Grade, etc.).
    - Add `api.ts` for API calls.

2.2. Wire Up Context and Initial Layout

- In `page.tsx`, wrap main content in `GradebookContext`.
- **Render:** topbar Gradebook button, sidebar placeholder, main content placeholder.

Step 3: Sidebar and Course Selector

3.1. Sidebar UI

- **File:** `GradebookSidebar.tsx`
    - Fetch and display user's courses.
    - On course select, update context with selected course.
- **Integrate** in `page.tsx` so sidebar always appears after Gradebook is selected.

Step 4: Main Content Area and Routing

4.1. Main Content Split

- **File:** `GradebookMain.tsx`
    - Switches between teacher/student view based on context/role.
- **Teacher folder:** `/Teacher/`
    - `GradeSections.tsx` (category management UI)
    - `Assignments.tsx` (assignment CRUD UI)
    - `StudentGrades.tsx` (grade entry, averages, student selector)

- **Student folder:** `/Student/`
    - `MyGrades.tsx` (view own grades)
    - `WhatIfTool.tsx` (hypothetical calculator)

Step 5: Implement Grade Section Management (Teacher)

5.1. GradeSections Component

- **File:** `/Teacher/GradeSections.tsx`
    - Display/add/edit/delete categories.
    - Use sliders/up-down for % input (enforce sum=100).
    - Update live total.
    - Fetch/save via `/sections` endpoint.

Step 6: Late Penalty (Teacher)

6.1. LatePenalty UI

- Add late penalty edit form under sections management.
- GET/PUT from `/late-penalty` endpoint.
- Store in context for grade calculation.

Step 7: Assignment CRUD (Teacher)

7.1. Assignments Component

- **File:** `/Teacher/Assignments.tsx`
    - List, add, edit, delete assignments.
    - Assignment form: name, due date, section dropdown, optional comment.
    - Validate section/category on save.
    - CRUD with `/assignments` endpoint.

Step 8: Student Grades and Entry (Teacher)

8.1. StudentGrades Component

- **File:** `/Teacher/StudentGrades.tsx`
    - Teacher-only table: student selector, grade entry for each assignment, submitted date, status (late/on time, allow override), comments.
    - Show average for each assignment.
    - Fetch/save via `/grades` endpoint.

Step 9: Student View (MyGrades + What-If Tool)

9.1. MyGrades Component

- **File:** `/Student/MyGrades.tsx`
    - Read-only table: assignments, grades, categories, late status, comments.
    - Display section weights (view-only).

    - Fetch from `/grades` endpoint.

9.2. WhatIfTool Component

- **File:** `/Student/WhatIfTool.tsx`
    - Modal/dialog for hypothetical grades.
    - On change, recalculate and show new total using shared grade logic.

Step 10: Shared Grade Calculation Logic

10.1. Calculation Utility

- **File:** `/lib/gradebook.ts`
    - Export function to calculate overall grade from assignments, sections, weights, late penalties.
    - Use in both Teacher and Student components (and for What-If Tool).

Step 11: UI Polish, Accessibility, Responsive

11.1. Accessibility/Responsive

- Review: semantic HTML, ARIA, keyboard navigation.
- Ensure layouts work on tablet/mobile.

11.2. UI Components

- **Folder:** `/components/gradebook/` (shared components: sliders, tables, modals, etc.)

Step 12: Finalize & Test

12.1. Test with sample data for both roles.

12.2. Edge cases: weights, missing assignments, late penalties.

12.3. Bugfix and polish.