Building an Assignment Workflow Portal with React.js and Node.js

Overview

You are tasked with creating a portal for teachers and students with both front-end and back-end components using React.js and Node.js.

The portal will include a single login screen for both teachers and students. After authentication, the user will be redirected to their role-specific dashboard:

- Teacher Dashboard → Create and manage assignments.
- Student Dashboard → View and submit answers for published assignments.

The goal is to implement a workflow-driven feature where assignments move through defined states, and students actively participate by submitting answers. This ensures the task goes beyond simple CRUD operations.

Requirements

1. Single Login Page with Role-Based Redirection

- Implement a single login screen for both teachers and students.
- User inputs email and password.
- Backend verifies credentials and responds with: • JWT token • Role (teacher or student)
- Based on the role: • Teacher → Redirect to the Teacher Dashboard. • Student → Redirect to the Student Dashboard.
- Handle login errors and display clear error messages (e.g., invalid credentials).

---

2. Teacher Dashboard – Assignment Management

- Teachers manage the full lifecycle of assignments.
- Each assignment must have: • Title • Description • Due Date • Status (Default: Draft)
- State transitions for assignments:

Draft → Published → Completed

• Draft: Editable and deletable. • Published: Visible to students for submission (cannot be deleted). • Completed: Locked after review, no further changes allowed.

- Teachers can view all student submissions for each assignment, including: • Student Name • Submitted Answer • Submitted Date • Optional: Teachers can mark submissions as reviewed.

---

3. Student Dashboard – Viewing and Submitting Assignments

- Students can view only Published assignments.
- For each assignment, a student can: • Submit their answer (text-based). • View their submitted answer after submission. • Cannot edit the submission once it's sent.
- Each student can submit only one answer per assignment.

---

4. Assignment Listing & Filtering

Teacher View:

- Filter assignments by status: • Draft • Published • Completed

Student View:

- Automatically filtered to show only Published assignments.

Technology Requirements

Front-End (React.js)

- Use React.js for building the front-end components.
- Implement state management using Redux or Context API.
- Use a single login page for both roles.
- Implement role-based UI rendering: • Show only relevant options and pages for each role.
- Styling: TailwindCSS or Material UI.
- Ensure proper form validation and error handling on the client side.

- Should use the latest and good coding practices.

Back-End (Node.js + Express.js)

- Use Node.js with Express.js for building RESTful APIs.

Submission Guidelines

- Create separate repositories for: • Front-end (React.js) • Back-end (Node.js)
- Share links to both repositories once completed.
- Each repository must include a README file with: • Steps to set up and run locally. • Any additional notes or assumptions made.

Additional Information

- Focus on workflow and access control, not just CRUD.
- Enforce security best practices: • Protect all teacher-only routes. • Validate inputs both client-side and server-side.
- Keep the design simple yet responsive.
- Bonus points for: • Preventing submissions after the assignment's due date. • Adding pagination to the assignment list. • Dashboard analytics for teachers (e.g., number of submissions per assignment).

Example Flow

1. Login Flow:

- User logs in through the same login screen.
- Backend returns role (teacher or student).
- Frontend redirects based on role.

2. Teacher Flow:

- Creates assignment → Draft
- Publishes assignment → Published

- Students submit answers.
- After review, teacher marks it as → Completed.

3. Student Flow:

- Logs in and sees only Published assignments.
- Submits one answer per assignment.
- Views submitted answer but cannot edit it.