**1. Bối cảnh đề tài**

Bạn đang làm **phần mềm quản lý học liệu cho hệ thống học tập trực tuyến (LMS)**. Thầy yêu cầu rất rõ một ý cốt lõi:

“Đừng nghĩ theo kiểu: admin, giảng viên, sinh viên là 3 bảng. Hãy nghĩ: **Users – Roles – Actions**. Mỗi người dùng phải có tài khoản, được gán vào các nhóm (roles), và mỗi role có các thao tác (actions). Quan hệ đều là N–N.”

Tức là: thay vì cứng hóa “Admin” hay “Teacher”, ta thiết kế **RBAC linh hoạt**. Sau này thêm “Người nhập học liệu”, “Quản lý khoa”, “CTV nội dung” không phải sửa code nhiều.

Ngoài ra thầy còn nhắc: “giảng viên không chắc là người nhập nội dung, người nhập thường soạn bằng Word → phải có cơ chế đưa file DOC/DOCX → HTML và phân quyền xem/sửa theo từng học liệu”. Bạn lại đang có **module convert DOC → HTML** từ trước → ta tận dụng.

**2. Mục tiêu hệ thống (phiên bản hiện tại)**

1. **Xác thực + phân quyền theo action** (RBAC 3 lớp):
   * Users
   * Roles
   * Actions
   * Bảng trung gian: UserRoles, RoleActions, UserActions (cấp quyền riêng), cộng thêm “phân quyền trên từng học liệu” → MaterialUserPerm.
2. **Hiển thị menu động theo user**:
   * “Mỗi trang chính là một Action”
   * Menu lấy từ DB, không hard-code ở frontend.
3. **Quản lý học liệu** ở mức cơ bản:
   * xem danh sách học liệu đã xuất bản
   * upload / thêm học liệu (chỉ người có quyền)
   * quản lý học liệu theo người được phân công
4. **Tách backend – frontend** để sau này làm UI hiện đại.

**3. Công nghệ bạn đang dùng / đã chốt**

**Backend:**

* .NET Framework **4.5.1**
* **ASP.NET Web API** (kiểu cổ điển, chạy được trên VS 2013)
* Kết nối SQL Server bằng **Ado.NET thuần** (SqlConnection, SqlCommand)
* JWT tự ký (không dùng Identity)
* Refresh token lưu DB (bảng lms.AuthRefreshTokens)
* Tự viết attribute kiểm quyền [RequireAction("...")]

**Database (SQL Server):**

* Schema chính: **lms**
* Bảng cốt lõi:
  + lms.Users
  + lms.Roles
  + lms.Actions
  + lms.UserRoles
  + lms.RoleActions
  + lms.UserActions
  + lms.Materials
  + lms.MaterialUserPerm (phân quyền tới từng học liệu)
  + View: lms.v\_MenuByUser
  + Hàm: lms.fn\_GetEffectiveActions(@UserId), lms.fn\_UserHasAction(@UserId, @ActionCode)
* Có liên kết thêm sang DB convert cũ (Documents, Sections, Paragraphs, Cells) qua schema conv (tùy).

**Frontend:**

* **React + Vite**
* **JavaScript** (không dùng TS)
* **axios** + interceptor
* **react-router-dom**
* CORS bật từ backend

**4. Backend hiện tại đã làm gì**

**4.1. Xác thực**

* API: POST /api/auth/login
  + Nhận username + password
  + Kiểm tra trong lms.Users
  + Đã chuyển sang mô hình **hash + salt (PBKDF2)**: nếu user còn mật khẩu cũ (plain) thì **login lần đầu sẽ tự “nâng cấp” sang hash**.
  + Sinh **access token (JWT ~ 15 phút)**.
  + Sinh **refresh token (7 ngày)** lưu vào lms.AuthRefreshTokens.
  + Trả accessToken trong body, **refreshToken trong cookie HttpOnly + Secure**.
* API: POST /api/auth/refresh
  + Đọc cookie refreshToken
  + Kiểm tra DB: còn hạn, chưa revoke
  + Trả accessToken mới
* API: POST /api/auth/logout
  + Xoá cookie refresh
  + (tùy chọn) đánh dấu revoke token trong DB

**4.2. Kiểm tra quyền**

* Bạn đã viết attribute kiểu:

[RequireAction("MATERIAL\_UPLOAD")]

* Attribute này:
  + đọc userId từ JWT
  + gọi SQL:
  + SELECT lms.fn\_UserHasAction(@uid, @ac)
  + nếu trả 0 → 403.
* Hàm SQL lms.fn\_GetEffectiveActions(@UserId) tính quyền **từ role + từ user + trừ deny**.

**4.3. Menu động**

* View SQL: lms.v\_MenuByUser join từ UserRoles → RoleActions → Actions để lấy ra *những action có IsMenu=1* của đúng user.
* API: GET /api/menu/my
  + Đã code trong Web API
  + Trả về danh sách action (actionId, code, name, path, parentId, sortOrder, icon)
  + Frontend dùng để vẽ sidebar.

**4.4. Materials**

* API public: GET /api/materials/published?skip=0&take=20
  + Lấy từ lms.Materials WHERE Status=3
  + Để sinh viên / khách xem
  + Có [AllowAnonymous]
* API quản lý:
  + GET /api/materials/manage → chỉ user được phân công trong lms.MaterialUserPerm mới thấy
  + POST /api/materials/upload → có [RequireAction("MATERIAL\_UPLOAD")]
  + DELETE /api/materials/{id} → có [RequireAction("MATERIAL\_DELETE")]

**4.5. JWT handler / authorize**

* Bạn đã đăng ký handler trong WebApi để mọi request đi qua đều kiểm tra JWT.
* Controller cần bảo vệ được gắn [Authorize], còn endpoint nào public thì [AllowAnonymous].

=> Tức là **backend đã chạy được một vòng đời: login → gọi API có quyền → chặn nếu không đủ quyền.** Đó là lõi của đề tài.

**5. Frontend hiện tại đã làm gì**

**5.1. Cấu trúc đã đề xuất**

src/

api/ ← axios + auth + menu + materials

hooks/ ← useAuth, useMenu

components/ ← Sidebar, MenuTree, ProtectedRoute, ActionGate (tùy)

pages/ ← Dashboard, Login, MaterialsPublished, MaterialsUpload, MaterialsManage, Users, Roles, Actions, NotFound

layout/ ← AppLayout

App.jsx

config.js

**5.2. Luồng đang chạy**

1. User mở web → nếu chưa login → hiện Login.jsx.
2. Login gọi POST /api/auth/login → nhận accessToken → lưu vào biến trong JS → gọi tiếp /api/menu/my.
3. Menu trả về đúng **menu theo user** → render sidebar.
4. Người dùng bấm vào từng menu → React Router chuyển route → mở page tương ứng.
5. Nếu đang gọi API mà token hết hạn → axios interceptor tự gọi /api/auth/refresh vì cookie đã có sẵn → gọi lại request.

**5.3. Các page mẫu đã có**

* Dashboard
* MaterialsPublished (gọi API thật)
* MaterialsUpload (gọi API thật và test phân quyền)
* MaterialsManage (gọi API thật và test phân quyền theo từng học liệu)
* Users / Roles / Actions (placeholder để map với DB)

→ Tức là frontend đã **đúng tinh thần “mỗi trang chính là một action”** như lời thầy.

**6. Hướng phát triển tiếp (roadmap chi tiết)**

Tôi sẽ chia theo kiểu công việc để bạn đưa vào Cursor dễ:

**(1) Hoàn thiện bảo mật backend**

* Bật global exception filter để tất cả lỗi trả { ok:false, message:"..." }
* Bật CORS đúng domain FE
* Đưa JWT secret vào Web.config (không hard-code)
* Thêm job (SP hoặc Agent) dọn refresh token hết hạn
* Thêm index:
* CREATE UNIQUE INDEX IX\_Actions\_Code ON lms.Actions(ActionCode);
* CREATE INDEX IX\_Materials\_Status\_Updated ON lms.Materials(Status, UpdatedAt DESC);
* CREATE INDEX IX\_MaterialUserPerm\_User ON lms.MaterialUserPerm(UserId, MaterialId);

**(2) Chuẩn hoá học liệu**

* Ràng buộc Materials.Status = (1:Draft, 2:Pending, 3:Published)
* Thêm API: GET /api/materials/{id}
* Thêm API: PUT /api/materials/{id} (có [RequireAction("MATERIAL\_EDIT")])
* Thêm API: GET /api/materials/byclass/{classId} nếu bạn làm quan hệ lớp học → học liệu

**(3) Kết nối module convert DOC → HTML**

* Khi upload học liệu, nếu file là DOCX → push vào bảng/conversion job
* Lưu đường dẫn HTML đã convert vào lms.Materials
* Chỉ cấp quyền EDIT cho vai trò “người nhập nội dung”

**(4) Frontend**

* Tách riêng form login (đã có)
* Thêm trang xem chi tiết học liệu (đọc HTML do backend trả)
* Thêm guard ẩn nút theo ActionCode (client-side)
* Thêm trang quản trị: Users / Roles / Actions gọi API thật (khi bạn viết xong API tương ứng)

**(5) Báo cáo / giải trình với thầy**

* Vẽ lại ERD chỉ còn: **Users, Roles, Actions + bảng N–N + Materials + MaterialUserPerm**
* Mô tả luồng: “user đăng nhập → lấy actions → sinh menu → request → kiểm tra quyền ở DB → trả data”
* Nói rõ: “không cần tạo bảng Admins, Teachers, Students riêng — vì đó chỉ là role”

**Tóm nhanh cho Cursor**

Dự án: LMS – Phần mềm quản lý học liệu cho hệ thống học tập trực tuyến.  
Yêu cầu chính: phân quyền động theo mô hình Users–Roles–Actions (N–N), mỗi trang chính là một action. Backend .NET 4.5.1 Web API, SQL Server, JWT + refresh token bằng HttpOnly cookie, attribute kiểm quyền gọi hàm SQL lms.fn\_UserHasAction. Frontend React (Vite, JS) gọi /api/menu/my để render menu theo user, dùng axios interceptor để tự refresh access token. Đã có API materials (published, manage, upload) và API auth đầy đủ. Cần sinh thêm CRUD cho materials, users, roles, actions và bổ sung UI tương ứng.

Thế là xong. Giờ bạn đẩy qua Cursor, bảo “sinh thêm page /materials/detail/:id + gọi API /api/materials/{id} + dùng layout có Sidebar này”, nó sẽ hiểu liền