Case Study Challenge | Web Development

Project: "SyncScript" – Collaborative Research & Citation Engine

Business Context and Case Overview

In academia and professional research, information is often siloed. While tools exist for writing, there is a gap in **collaborative, real-time verification and source sharing**. **SyncScript** is a platform where researchers can collaboratively build "Knowledge Vaults"—shared repositories of verified sources, annotated PDFs, and cross-referenced citations.

As a full-stack engineer, your mission is to architect a system that transitions from a personal bookmarking tool to a high-concurrency collaborative engine capable of handling complex data relationships and instant updates.

Key Challenges

- **Concurrency Control:** Managing multiple users editing the same "Knowledge Vault" simultaneously without data loss.
- **Complex Data Relationships:** Linking authors, publications, and user annotations across a sprawling database.
- **File Integrity & Security:** Ensuring that uploaded research remains immutable and accessible only to authorized collaborators.

Your goal is to build a high-performance marketplace for information. You are tasked with moving beyond a simple CRUD app to create a system that handles **real-time synchronization**, **complex access levels**, and **data integrity** for thousands of concurrent interactions.

The Mission: The Unified Scale-Up

Unlike a traditional pilot, you must deliver a system that is architected for national scale from day one. You are responsible for both the backend service and the core frontend.

A. Core Functionality & Data Modeling

- **Knowledge Vaults:** Users must be able to create "Vaults" to house research.
- **Resource Management:** Support full CRUD operations for Sources (URLs, Titles) and Annotations (Notes).
- **Complex Relationships:** Implement a relational database (e.g., PostgreSQL) to manage many-to-many relationships between researchers and their shared vaults.

B. Security & Collaboration

- **Identity & Access:** Implement Authentication and **Role-Based Access Control (RBAC)**.
  - *Owner:* Full control over the Vault.
  - *Contributor:* Can add/edit research.
  - *Viewer:* Read-only access.
- **Protection:** Integrate Request Throttling and Rate-Limiting to prevent API abuse.

C. Cloud Integration & Real-Time Performance

- **File Management:** Integrate a **Cloud Storage service (e.g., AWS S3 or Cloudinary)** for PDF uploads and research whitepapers. This replaces local storage to ensure file immutability and handle the complexity of signed URLs and multipart uploads.

- **Instant Updates:** Use **WebSockets** or similar technology to ensure that when one researcher adds a source, it appears instantly on the screens of all active collaborators.
- **Automated Notifications:** Use a managed service (e.g., **Pusher or Twilio**) to trigger browser or SMS notifications when a new "Contributor" is added to a Vault.
- **Optimization & Auditability:** Implement **Caching (e.g., Redis)** for high-traffic research and maintain immutable logs of all status changes to ensure research integrity.

Submission Guidelines

Your solution must demonstrate critical thinking regarding system evolution.

1. **Code Repository:** A functional GitHub repo with a clear directory structure.
2. **README Documentation:** Must include design decisions, data model diagrams, and assumptions made.
3. **Live Demo:** A 7-minute functional walkthrough of the product, followed by a Q&A session with the judges.

Evaluation Rubric

Category	Weight	Excellent (9-10)	Good (6-8)	Needs Improvement (0-5)
System Architecture & Data Modeling	**25%**	Database schema is highly optimized with clear many-to-many relationships; handles complex data integrity with ease.	Schema is functional and relational but may lack optimization for high-scale queries.	Poorly defined entities; relies on flat files or local storage without a scaling plan.
Real-Time & Performance	**25%**	Flawless WebSocket integration and Cloud Storage usage; effective use of Redis caching and notification queues.	Real-time features work but show latency; caching or cloud storage is implemented but not strategically used.	No real-time synchronization; system relies purely on manual refreshing or local storage.

Security & Access Control	**20%**	Robust JWT auth and granular RBAC (Owner/Contributor/Viewer); effective rate-limiting and audit logs.	Basic authentication and role separation are present but may have minor security gaps.	Missing authentication or roles; system is vulnerable to basic API abuse.
User Experience (UX/UI)	**15%**	Frontend is responsive and adapts views dynamically based on user roles and permissions.	Functional and clean UI; consistent navigation but lacks advanced role-based dynamic elements.	Difficult to navigate; UI does not reflect backend permissions or real-time state changes.
Creativity & Edge	**15%**	Implements advanced features like auto-citation, AI metadata extraction, or complex conflict resolution.	Includes minor "bonus" features beyond the core brief to improve usability.	Meets minimum requirements only; no attempt at original problem-solving.