PeerLearn Labs

PeerLearn — Tech Stack Analysis (v2)

Date: 2025-09-25

1. Purpose

This document explains the rationale behind the chosen technologies for PeerLearn. It aligns university constraints with project needs.

2. Backend Choice — C# / ASP.NET Core Web API

ASP.NET Core provides long-term support, strong Web API patterns, real-time communication (SignalR), and security features such as Data Protection and Rate Limiting.

3. Frontend Choice — React

React is widely used and supported, integrates well with collaborative libraries such as Slate and Yjs, and offers a familiar developer experience.

4. Database Strategy — Hybrid SQL + MongoDB

SQL satisfies university requirements for relational modeling. MongoDB is chosen for collaborative notes and flashcards due to schema flexibility and efficient handling of CRDT-based updates.

5. Real-Time Collaboration Stack

Slate provides the editor, Yjs ensures conflict-free syncing, y-websocket handles transport, and SignalR complements collaboration with presence and chat.

6. Voice (Post-MVP)

WebRTC enables real-time voice; TURN is required for NAT traversal.

7. Risks & Trade-offs

Operational overhead of two datastores, SQL/NoSQL divergence, and security concerns with token handling. Mitigation strategies include clear separation of responsibilities and secure defaults.

8. Decision Matrix

Backend: ASP.NET Core (.NET 8 LTS)
Frontend: React
Collaboration: Slate + Yjs + y-websocket; SignalR
Data: MongoDB for content, SQL for audit and compliance

9. References

Microsoft .NET lifecycle, ASP.NET Core docs, SignalR, React adoption surveys, Yjs/Slate documentation, MongoDB and EF Core references, and OWASP guidance.