Version & Metadata

Version: v1.1
Title: PeerLearn — Technical Overview & Analysis
Project: PeerLearn (Personal project)
Author: Aleksandar Ivanov
Date: 2025-09-11 (Europe/Amsterdam)
Status: Draft for review
Change Log:

- v1.1 — Split from original master doc; restored all missing sections (data model, real-time flow, voice chat, security, build order, risks); added explicit source links.
- v1.0 — Initial analysis overview drafted from README.

1) Scope & Purpose

PeerLearn is a collaborative study platform where students create course-centric rooms with shared notes, flashcards/quizzes, chat/voice, and progress tracking (MVP details below). This document defines the MVP, architecture, do's/don'ts, data model, real-time flows, security essentials, delivery order, and risks — with every factual claim backed by a public source.

> **Constraint (project):** Backend must be C#/.NET; university prefers SQL, but MongoDB is preferred by the author. These are project constraints, not external facts.

2) Grounded Facts & Rationale

- **.NET 8 is current LTS; .NET 9 is STS** — .NET 8 (LTS) released Nov 14, 2023 with support through Nov 10, 2026; .NET 9 (STS) released Nov 12, 2024 and is supported 18 months to May 12, 2026. Microsoft support policy Lifecycle table What's new in .NET 9
- **ASP.NET Core is designed for building Web APIs** — official docs outline controllers & minimal APIs patterns. Web API overview Minimal APIs
- **SignalR provides real-time communication in ASP.NET Core** — bi-directional communication over WebSockets/other transports. SignalR overview JavaScript client @microsoft/signalr package
- **React remains widely used** — 2025 Stack Overflow survey shows React used by ~44.7% of all respondents, ~46.9% of professional developers (web frameworks & technologies section). SO Survey 2025 — Technology
- **Slate is a customizable rich-text editor** suitable for building Notion-like editors. Slate documentation

- **Yjs is a CRDT library enabling offline-friendly real-time collaboration**; official ecosystem recommends `y-websocket` for centralized auth, and also offers `y-webrtc`. Yjs docs y-websocket docs y-webrtc repo
- **WebRTC handles real-time audio/video and data channels** (via `RTCPeerConnection` + `getUserMedia`) and often requires STUN/TURN for NAT traversal. WebRTC API (MDN) getUserMedia (MDN) TURN (MDN) coturn project
- **MongoDB document model (schema-flexible) with official C#/.NET driver**; an EF Core provider for MongoDB also exists. MongoDB .NET/C# driver docs MongoDB EF Core provider docs
- **EF Core supports multiple providers including SQL Server & PostgreSQL** (satisfying SQL expectations). EF Core providers (Npgsql) NuGet SQL Server provider EF providers overview (MS Learn)
- **ACID transactions are a core property of relational databases** (e.g., PostgreSQL, SQL Server). PostgreSQL "About" (ACID since 2001) SQL Server transaction guide
- **JWT & REST security practices** — validate signature & standard claims; prefer HttpOnly cookies to mitigate token theft by JS; avoid storing session identifiers in `localStorage`. OWASP REST Security OWASP HttpOnly OWASP HTML5 Security — localStorage warning
- **OpenAI API & Hugging Face** can power flashcard/quiz generation (optional). OpenAI API reference Hugging Face Inference Endpoints

3) MVP Definition (4–8 weeks of core build; see timeline section for 4-month plan)

Must-have features (MVP):

1. **Rooms**: create/join/leave; list rooms you belong to. *(design decision)*
2. **Auth**: email/password with JWT (server-side validation of signature & claims; HttpOnly cookie recommended). OWASP REST Security HttpOnly
3. **Shared Notes**: rich-text editor using Slate bound to a Yjs document; sync via SignalR or `y-websocket`. Slate Yjs y-websocket SignalR
4. **Flashcards/Quizzes (basic)**: generate from note text via an AI endpoint; store decks & results. OpenAI API Hugging Face
5. **Progress tracking (starter)**: streaks, XP per study action; server-computed to avoid client tampering. *(design decision)*

Should-have (post-MVP): voice co-study, badges, room analytics, search. *(design decision)*

4) Architecture Overview

- **Backend:** ASP.NET Core Web API (.NET 8 LTS recommended during the 2025 academic year; upgrade path to .NET 9/10 later). Support policy

    - **Real-time:** SignalR hub for editor presence/awareness & chat updates. SignalR intro
    - **Auth:** ASP.NET Core Identity or JWT bearer; validate `iss,aud,exp`, etc.; deliver access token via HttpOnly cookie. JWT validation (OWASP) HttpOnly
    - **Persistence:**
        - **MongoDB** for notes, flashcards, quiz results, presence states. MongoDB .NET driver
        - **SQL (minimal)** if required by university for specific tables (e.g., audit/logs or user/profile) via EF Core SQL Server or PostgreSQL. EF Core SQL Server Npgsql EF Core ACID basics
- **Frontend:** React app with Slate editor; Yjs client & provider (`y-websocket` preferred for central auth). React usage stats 2025 Slate docs y-websocket docs
- **Optional voice:** WebRTC peer connections for room audio, with TURN for NAT traversal; feature-flagged. WebRTC API TURN coturn

5) Data Model (MVP slice)

MongoDB collection-oriented design; SQL tables can mirror selected entities if needed to meet coursework requirements. *(design decision)*

- **User**: `{ _id, email, password_hash, display_name, created_at, roles[] }` *(design decision; store passwords via ASP.NET Identity hashing.)* ASP.NET Core Identity
- **Room**: `{ _id, name, course_code, owner_id, member_ids[], created_at }` *(design decision)*
- **Note**: `{ _id, room_id, ydoc_snapshot?, last_updated_by, updated_at }` *(design decision)* Yjs docs
- **Flashcard**: `{ _id, room_id, note_id?, front, back, created_by, created_at }` *(design decision)*
- **QuizResult**: `{ _id, user_id, room_id, deck_id?, score, taken_at }` *(design decision)*
- **Presence (ephemeral)**: in-memory/Redis for cursors/users online; not persisted long-term. *(design decision)*

6) Real-Time Notes — Sync Flow (Slate + Yjs)

1. Editor operations apply to a **Slate** value and update the **Yjs** shared document via the `slate-yjs` binding. Slate Yjs
2. The Yjs doc syncs over a provider — **`y-websocket`** (central server for auth/cookies) or **`y-webrtc`** (peer-to-peer; trickier to auth/scale). y-websocket docs y-webrtc repo
3. Server relays awareness (selection/cursor presence) and persists periodic snapshots for recovery. *(design decision)*
4. SignalR broadcasts room events (joins, titles, badge unlocks) separate from editor ops. SignalR intro

Provider choice note: Yjs docs highlight `y-websocket` as a good choice when you need central auth/headers/cookies. y-websocket docs

7) Voice Chat (Optional, Post-MVP)

- Use **WebRTC** to capture mic streams (`navigator.mediaDevices.getUserMedia`) and connect peers via **RTCPeerConnection**; deploy **TURN** for users behind symmetric NATs. getUserMedia (MDN) WebRTC API TURN coturn
- Use SignalR for signaling (exchange SDP/ICE) or a small signaling endpoint. *(design decision)*

8) Security Essentials (MVP)

- **Auth tokens:** Validate JWT signature & claims on every request (`iss,aud,exp,nbf,iat`). OWASP REST Security
- **Token storage:** Prefer **HttpOnly** cookies over `localStorage` to reduce XSS-driven token theft; avoid storing session identifiers in `localStorage`. OWASP HttpOnly HTML5 Security: localStorage risk
- **Transport:** Enforce HTTPS (HSTS). *(design decision)*
- **Rate limiting:** Apply ASP.NET Core **RateLimiting** middleware to APIs & hubs. Rate limiting middleware
- **Data protection:** Persist and protect ASP.NET Core data protection keys (cert/KeyVault/NFS). Data Protection config Key lifetime
- **Input validation:** Use model validation attributes/FluentValidation. Model validation FluentValidation for ASP.NET

9) Build Order (Practical Sequencing)

1. **Auth & Users** (Identity/JWT + HttpOnly cookies) → seed admin. Identity
2. **Rooms CRUD** (MongoDB persistence). MongoDB .NET
3. **Notes MVP**: Slate editor + Yjs with `y-websocket` provider; add awareness cursors. Slate Yjs y-websocket
4. **Flashcards/Quizzes basic**: server endpoint calls OpenAI/HF; store decks/results. OpenAI Hugging Face
5. **Progress tracking (starter)**: XP events on server. *(design decision)*
6. **SignalR** for presence/room events and chat. SignalR intro
7. **Hardening**: rate limiting, Data Protection, logging, metrics. Rate limiting Data Protection

10) Do's & Don'ts (MVP)

Do

- Target **.NET 8 LTS** for stability during the academic window; plan an upgrade window later. Support policy

- Use **HttpOnly cookies** for access tokens, and validate JWTs server-side. OWASP HttpOnly REST Security
- Choose **`y-websocket`** for editor sync to simplify auth & headers. y-websocket docs
- Add **TURN** for voice. TURN (MDN)

Don't

- Store tokens in `localStorage` or expose them to JS if not necessary. HTML5 Security
- Depend on **`y-webrtc`** for auth-sensitive docs unless you've vetted signaling/auth; prefer `y-websocket`. y-websocket docs
- Over-index on premature optimization claims; benchmark real user flows when needed. *(design practice)*

11) Risks & Mitigations

- **Weak token storage or claim validation → account/session compromise.** Mitigate via HttpOnly cookies, strict JWT validation, short TTLs, refresh flow. OWASP HttpOnly REST Security
- **NAT traversal blocks voice** without TURN. Deploy coturn or hosted TURN. TURN (MDN) coturn
- **University SQL requirement** vs. app's MongoDB preference. Satisfy with hybrid persistence (e.g., users/audit in SQL via EF Core + content in MongoDB). Npgsql EF Core EF SQL Server MongoDB .NET driver
- **Real-time conflicts**: incorrect provider selection or missing snapshotting. Prefer `y-websocket`, schedule periodic snapshots. y-websocket docs

12) 4-Month Delivery Plan (from README)

- **Month 1:** Backend scaffolding (models, auth, CRUD for Rooms/Notes) + Frontend auth/dashboard. *(design plan)*
- **Month 2:** Real-time collaboration (SignalR/Yjs) + Flashcards & quizzes. *(design plan)*
- **Month 3:** Progress tracking + gamification; polish UI; docs & demo prep. *(design plan)*
- **Month 4 (buffer/QA):** Security hardening, metrics, TURN setup, performance fixes. *(design plan)*

13) References (direct links)

- .NET support & lifecycle:
  - https://dotnet.microsoft.com/en-us/platform/support/policy
  - https://learn.microsoft.com/en-us/lifecycle/products/microsoft-net-and-net-core
  - https://learn.microsoft.com/en-us/dotnet/core/whats-new/dotnet-9/overview

- ASP.NET Core Web API:
    - https://learn.microsoft.com/en-us/aspnet/core/web-api/?view=aspnetcore-9.0
    - https://learn.microsoft.com/en-us/aspnet/core/fundamentals/minimal-apis/overview?view=aspnetcore-9.0
- SignalR:
    - https://learn.microsoft.com/en-us/aspnet/core/signalr/introduction?view=aspnetcore-9.0
    - https://learn.microsoft.com/en-us/aspnet/core/signalr/javascript-client?view=aspnetcore-9.0
    - https://www.npmjs.com/package/%40microsoft/signalr
- React adoption stats:
    - https://survey.stackoverflow.co/2025/technology
- Slate editor:
    - https://docs.slatejs.org
- Yjs & providers:
    - https://docs.yjs.dev
    - https://docs.yjs.dev/ecosystem/connection-provider/y-websocket
    - https://github.com/yjs/y-webrtc
- WebRTC & media capture:
    - https://developer.mozilla.org/en-US/docs/Web/API/WebRTC_API
    - https://developer.mozilla.org/en-US/docs/Web/API/MediaDevices/getUserMedia
    - https://developer.mozilla.org/en-US/docs/Web/API/WebRTC_API/Protocols#turn
    - https://github.com/coturn/coturn
- MongoDB:
    - https://www.mongodb.com/docs/drivers/csharp/
    - https://www.mongodb.com/docs/efcore/current/
- EF Core providers (SQL):
    - https://www.nuget.org/packages/Microsoft.EntityFrameworkCore.sqlserver/
    - https://www.npgsql.org/efcore/
    - https://learn.microsoft.com/en-us/ef/core/providers/?tabs=dotnet-core-cli
- ACID references:
    - https://www.postgresql.org/about/
    - https://learn.microsoft.com/en-us/sql/relational-databases/sql-server-transaction-locking-and-row-versioning-guide?view=sql-server-ver17
- Security (OWASP):

    - https://cheatsheetseries.owasp.org/cheatsheets/REST_Security_Cheat_Sheet.html
    - https://owasp.org/www-community/HttpOnly
    - https://cheatsheetseries.owasp.org/cheatsheets/HTML5_Security_Cheat_Sheet.html
- AI endpoints:
    - https://platform.openai.com/docs/api-reference
    - https://huggingface.co/docs/inference-endpoints/index
