LineByLine: Platform Vision Document

Table of Contents

# LineByLine: A Platform for Real Language Acquisition

**Version 1.0** | October 2025  
*“Language learning that respects science, not engagement metrics.”*

## Table of Contents

1. [Executive Summary](#executive-summary)
2. [The Problem: Why Traditional Apps Fail](#the-problem)
3. [Our Solution: Evidence-Based Learning](#our-solution)
4. [Core Philosophy & Learning Principles](#core-philosophy)
5. [Technical Architecture](#technical-architecture)
6. [Module Ecosystem](#module-ecosystem)
   * [Vocabulary Module](#vocabulary-module)
   * [Speaking Module](#speaking-module)
   * [Writing Module](#writing-module)
   * [Reading Module](#reading-module)
   * [Listening Module](#listening-module)
7. [Design System: Quiet Immersion](#design-system)
8. [Gamification Strategy](#gamification-strategy)
9. [AI & Personalization](#ai-personalization)
10. [Roadmap](#roadmap)
11. [Success Metrics](#success-metrics)

## Executive Summary

**LineByLine** is a comprehensive language learning platform built on proven acquisition methods, not dopamine loops. Unlike Duolingo’s gamification-first approach, we prioritize:

* **Comprehensible Input** (Krashen’s i+1 hypothesis)
* **Spaced Repetition** (FSRS algorithm for optimal retention)
* **Output Practice** (speaking & writing with immediate feedback)
* **Real-world content** (not isolated sentences)
* **Community learning** (collaborative decks, shared progress)

### Tech Stack

| Layer | Technology | Purpose |
| --- | --- | --- |
| **Frontend** | Next.js (React) | Fast, SEO-friendly, PWA-ready |
| **Backend** | ASP.NET Core | Robust API, auth, data management |
| **ML/AI Engine** | Python (FastAPI + gRPC) | FSRS scheduling, speech processing |
| **Cache/Queue** | Redis | Session state, real-time features |
| **Database** | PostgreSQL | User data, decks, progress |
| **Deployment** | Docker + Compose | Consistent dev/prod environments |

### Key Differentiators

1. **Git-like deck versioning** — fork, commit, merge vocabulary sets
2. **FSRS 4.5 algorithm** — scientifically optimal review scheduling
3. **Modular skill system** — master one skill at a time
4. **Comprehensible input engine** — content adapts to your exact level
5. **Community-driven content** — subscribe to expert-curated decks

## The Problem: Why Traditional Apps Fail

### The Duolingo Trap

| What They Do | Why It Fails | Science Says |
| --- | --- | --- |
| **Gamification > Learning** | Streaks become the goal, not fluency | Extrinsic motivation fades; intrinsic motivation (understanding) lasts |
| **Isolated sentences** | “The apple is red” teaches nothing useful | Languages need context (usage-based theory) |
| **Translation exercises** | Thinking through L1 slows acquisition | Direct L2 → meaning mapping is faster (Krashen) |
| **Fixed review intervals** | Same timing for everyone | Individuals forget at different rates (Ebbinghaus curve varies) |
| **No output practice** | Passive recognition ≠ active recall | Output forces deeper processing (Swain’s Output Hypothesis) |

### What Learners Actually Need

Based on research from:

* **Stephen Krashen** (Input Hypothesis)
* **Paul Nation** (Vocabulary acquisition)
* **Merrill Swain** (Output Hypothesis)
* **Robert Bjork** (Desirable Difficulties)

**Successful acquisition requires:**

1. **Massive comprehensible input** (listening/reading at i+1 level)
2. **Spaced, interleaved practice** (not massed repetition)
3. **Forced output** (speaking/writing with feedback)
4. **Emotional engagement** (stories, not sentences)
5. **Autonomous learning** (learner controls content & pace)

## Our Solution: Evidence-Based Learning

### The LineByLine Method

┌─────────────────────────────────────────────────────────┐  
│ INPUT (60% of time) │  
│ ├─ Listening: Podcasts, dialogues (with/without subs) │  
│ └─ Reading: Graded texts, authentic articles │  
├─────────────────────────────────────────────────────────┤  
│ VOCABULARY (25% of time) │  
│ └─ FSRS-scheduled flashcards with context sentences │  
├─────────────────────────────────────────────────────────┤  
│ OUTPUT (15% of time) │  
│ ├─ Speaking: Shadowing, conversation practice │  
│ └─ Writing: Sentence construction, journaling │  
└─────────────────────────────────────────────────────────┘

### Why This Works

1. **Input-driven** — 60% focus on listening/reading builds intuition
2. **Context-rich** — vocabulary taught in sentences, not isolation
3. **Personalized spacing** — FSRS adapts to YOUR memory
4. **Output forces consolidation** — speaking/writing reveals gaps
5. **Intrinsic motivation** — progress measured in comprehension, not XP

## Core Philosophy & Learning Principles

### 1. Comprehensible Input (i+1)

**Definition:** Expose learners to content slightly above their current level.

**Implementation:**

* **Reading Module:** Texts auto-graded by known vocabulary %
  + Green: 98%+ known words (easy practice)
  + Yellow: 90-97% known (optimal learning zone)
  + Red: <90% known (frustration zone)
* **Listening Module:** AI adjusts speech rate (0.7x → 1.3x)
* **Vocabulary Module:** New words appear in familiar sentence structures

**Example:**

Current level: A2 (knows ~1500 words)  
  
✅ Good input: "Yesterday, María went to the market and bought fresh vegetables."  
❌ Bad input: "The photosynthetic process converts solar energy into chemical energy."

### 2. Spaced Repetition (FSRS Algorithm)

**Why not Anki’s SM-2?**

| Algorithm | Release | Strengths | Weaknesses |
| --- | --- | --- | --- |
| SM-2 (Anki) | 1988 | Simple, proven | Fixed formula, no personalization |
| **FSRS 4.5** | 2023 | Machine learning, adapts per user | Requires initial calibration |

**FSRS learns:**

* How quickly YOU forget specific card types
* Your optimal review intervals (not averages)
* Which cards need more frequent review

**Implementation:**

* Python service (@inclusive) handles scheduling
* Redis caches due cards for instant response
* ASP.NET backend (@personal-vocab) logs reviews

### 3. Comprehensible Output

**Theory (Swain, 1985):** Producing language (speaking/writing) helps learners:

* Notice gaps in knowledge
* Test hypotheses about grammar
* Develop fluency (automaticity)

**Our Output Tools:**

| Tool | Skill | Method |
| --- | --- | --- |
| **Shadowing** | Speaking | Repeat native audio with 0-1s delay |
| **Voice Journaling** | Speaking | 2-min daily audio diary (AI feedback) |
| **Sentence Mining** | Writing | Write 3 sentences using new vocabulary |
| **Conversation Simulator** | Speaking | AI partner adapts to your level |

### 4. Shadowing: The Ultimate Pronunciation Tool

**What is Shadowing?**

Listen to audio (speech, podcast, dialogue) and simultaneously repeat what you hear—matching:

* Pronunciation
* Intonation
* Rhythm
* Emotion

**Cognitive Benefits:**

1. **Phonological loop training** — improves working memory
2. **Prosody acquisition** — natural speech patterns
3. **Automaticity** — reduces cognitive load over time

**Implementation:**

User flow:  
1. Select audio (30s-2min clip)  
2. Listen 1x without speaking (comprehension check)  
3. Shadow 3x:  
 - First: Focus on pronunciation  
 - Second: Match rhythm/intonation  
 - Third: Add emotion/naturalness  
4. Record yourself (optional)  
5. AI compares your audio to native (pronunciation score)

**Content Library:**

* Movie/TV dialogues (Friends, Breaking Bad, etc.)
* TED Talk snippets
* Podcast clips (news, storytelling)
* Shadowing-specific lessons (minimal music, clear speech)

### 5. Desirable Difficulties (Bjork)

**Principle:** Learning should feel challenging, not easy.

**What We Avoid:**

* ❌ Multiple choice (too easy, encourages guessing)
* ❌ Translation crutches (prevents direct L2 thinking)
* ❌ Immediate hints (robs opportunity to struggle)

**What We Embrace:**

* ✅ Production (type/speak the answer)
* ✅ Interleaved practice (mix vocab, grammar, listening)
* ✅ Delayed feedback (think before seeing answer)

## Technical Architecture

### System Overview

┌──────────────────────────────────────────────────────────────┐  
│ FRONTEND (Next.js) │  
│ ┌────────────┐ ┌────────────┐ ┌────────────┐ │  
│ │ Vocabulary │ │ Speaking │ │ Reading │ │  
│ │ Module │ │ Module │ │ Module │ + 2 more │  
│ └────────────┘ └────────────┘ └────────────┘ │  
└────────────────────────┬─────────────────────────────────────┘  
 │ REST/WebSocket  
┌────────────────────────▼─────────────────────────────────────┐  
│ BACKEND (ASP.NET Core) │  
│ • User auth (JWT) │  
│ • Deck management (CRUD, versioning) │  
│ • Progress tracking (XP, streaks, badges) │  
│ • Social features (leaderboards, friends) │  
└────────────┬──────────────────────────┬──────────────────────┘  
 │ │  
 │ gRPC │ Redis pub/sub  
 │ │  
┌────────────▼──────────────┐ ┌───────▼──────────────────────┐  
│ AI ENGINE (Python) │ │ REDIS │  
│ • FSRS scheduling │ │ • Session cache │  
│ • Speech-to-text │ │ • Real-time updates │  
│ • Text difficulty calc. │ │ • Due card queue │  
│ • Pronunciation scoring │ │ │  
└───────────────────────────┘ └──────────────────────────────┘  
 │  
 │ PostgreSQL  
┌────────────▼──────────────┐  
│ DATABASE │  
│ • Users, decks, cards │  
│ • Review history (FSRS) │  
│ • Progress logs │  
└───────────────────────────┘

### Service Responsibilities

#### 1. Frontend (Next.js)

**Location:** line-by-line-ui/

**Features:**

* Server-side rendering for SEO
* PWA (offline study mode)
* Real-time updates via WebSocket
* Responsive design (mobile-first)

**Key Pages:**

/dashboard → Module selector, daily goal  
/vocabulary/study → Flashcard review (FSRS)  
/vocabulary/browse → Deck marketplace  
/speaking/shadow → Shadowing practice  
/reading/library → Graded texts  
/profile/stats → Progress graphs, streaks

#### 2. Backend (ASP.NET Core)

**Location:** personal-vocab/

**Endpoints:**

/api/auth → Register, login, JWT refresh  
/api/decks → CRUD, fork, commit, merge  
/api/cards → Add, edit, bulk import  
/api/study → Get due cards, submit review  
/api/progress → XP, badges, streaks  
/api/social → Friends, leaderboards  
/api/content → Texts, audio files (CDN proxy)

**Database Schema (Key Tables):**

Users → id, email, xp, streak\_count, settings  
Decks → id, owner\_id, parent\_deck\_id (for forks), visibility  
Cards → id, deck\_id, front, back, context\_sentence  
Reviews → id, card\_id, user\_id, rating, fsrs\_state  
Subscriptions → user\_id, deck\_id (following public decks)  
Content → id, type (text/audio), difficulty, language

#### 3. AI Engine (Python + gRPC)

**Location:** inclusive/

**Services:**

1. **FSRS Scheduler**
   * Calculates next review date per card
   * Updates card parameters (difficulty, stability)
   * Library: open-spaced-repetition/py-fsrs
2. **Speech Processing**
   * Speech-to-text (Whisper API or Azure)
   * Pronunciation scoring (phoneme comparison)
   * Voice activity detection
3. **Content Analysis**
   * Text difficulty scoring (based on frequency lists)
   * Sentence mining (extract example sentences)
   * Vocabulary extraction

**gRPC Proto Definition:**

service FSRSService {  
 rpc ScheduleCard(CardState) returns (NextReview);  
 rpc BatchSchedule(CardList) returns (ReviewSchedule);  
}  
  
service SpeechService {  
 rpc TranscribeAudio(AudioData) returns (Transcript);  
 rpc ScorePronunciation(PronunciationRequest) returns (Score);  
}

## Module Ecosystem

### Navigation Structure

┌─────────────────────────────────────────┐  
│ SIDEBAR (Persistent) │  
│ ┌─────────────────────────────────┐ │  
│ │ 📚 Vocabulary │ │  
│ │ 🎤 Speaking │ │  
│ │ ✍️ Writing │ │  
│ │ 📖 Reading │ │  
│ │ 🎧 Listening │ │  
│ │ ──────────────────── │ │  
│ │ 📊 Progress │ │  
│ │ 👥 Community │ │  
│ │ ⚙️ Settings │ │  
│ └─────────────────────────────────┘ │  
└─────────────────────────────────────────┘

## Vocabulary Module

### Overview

The foundation of language learning—spaced repetition flashcards powered by FSRS.

### Features

#### 1. Deck Management

**Personal Decks:**

* Create custom decks (e.g., “Business Spanish”, “Anime Vocabulary”)
* Import from CSV, Anki (.apkg), or text files
* Tag cards (grammar, slang, formal, etc.)

**Public Deck Marketplace:**

* Browse community-created decks
* Preview first 10 cards before subscribing
* See creator’s study stats (credibility)

**Git-like Versioning:**

linebyline/spanish-1000-words (master)  
├─ kamil/spanish-1000-words-modified (fork)  
│ └─ Added 50 regional words  
│ └─ Fixed 3 translation errors  
└─ maria/spanish-1000-words-business (fork)  
 └─ Specialized for business contexts

**Actions:**

* **Fork:** Copy deck to your account
* **Commit:** Save changes with message (“Added medical terms”)
* **Pull Request:** Suggest changes to original deck
* **Merge:** Original creator accepts your changes

#### 2. Study Session

**Review Flow:**

1. Show card front (e.g., "La manzana")  
2. User recalls meaning (mental effort)  
3. Flip card → see back ("The apple" + example sentence)  
4. Self-rate difficulty:  
 - Again (complete blank)  
 - Hard (struggled to recall)  
 - Good (recalled correctly)  
 - Easy (instant recognition)  
5. FSRS calculates next review (30m → 3d → 2w → 4mo...)

**Card Types:**

| Type | Front | Back | Use Case |
| --- | --- | --- | --- |
| **Vocab → Meaning** | L2 word | L1 definition + sentence | Recognition |
| **Meaning → Vocab** | L1 definition | L2 word (type it) | Production |
| **Audio → Text** | Audio clip | Written word | Listening |
| **Image → Vocab** | Picture | L2 word | Direct association (no L1) |
| **Cloze** | “I \_\_\_\_ to school” | “I **went** to school” | Grammar/usage |

**Contextual Learning:** Every card includes:

* Example sentence (shows word in use)
* Audio pronunciation (native speaker)
* Mnemonic hints (optional, user-created)

#### 3. Advanced Features

**Sentence Mining:**

* Import sentences from reading/listening content
* Auto-generate flashcards from highlighted words
* AI suggests best context sentence

**Leech Detection:**

* Cards reviewed 8+ times without progress → flagged
* Suggests: rewrite card, add mnemonic, or suspend

**Heatmap Progress:**

GitHub-style contribution graph:  
  
 M T W T F S S  
 ██ ██ ░░ ██ ██ ██ ██ ░░ ← Week 1  
 ██ ██ ██ ██ ░░ ██ ██ ← Week 2  
  
Dark = >20 cards reviewed  
Light = 1-20 cards  
Empty = 0 cards (streak broken!)

### Technical Implementation Details

#### System Architecture

**Technology Stack:**

* **Backend Framework:** ASP.NET Core 9.0 (C#)
* **Database:** PostgreSQL with Entity Framework Core
* **Caching Layer:** Redis (StackExchange.Redis)
* **AI/ML Service:** Python gRPC service (FSRS algorithm)
* **Authentication:** JWT Bearer tokens
* **Data Mapping:** AutoMapper
* **API Documentation:** Swagger/OpenAPI

**Service Dependencies:**

// Core services registered in Program.cs  
builder.Services.AddScoped<CardService>();  
builder.Services.AddScoped<DeckService>();  
builder.Services.AddScoped<DeckSubscriptionService>();  
builder.Services.AddScoped<StudySessionService>();  
builder.Services.AddScoped<SettingsService>();  
  
// Redis for session state and queue management  
builder.Services.AddSingleton<IConnectionMultiplexer>(sp =>  
{  
 var configuration = builder.Configuration.GetSection("Redis")["ConnectionString"];  
 return ConnectionMultiplexer.Connect(configuration!);  
});  
  
// gRPC client for FSRS scheduling  
builder.Services.AddGrpcClient<VocabService.VocabServiceClient>(o =>  
 o.Address = new Uri(builder.Configuration.GetSection("GrpcAddresses")["Inclusive"]!)  
);

#### API Endpoints & Functionality

**Base Path:** /api/v1/

##### Deck Management (/api/v1/deck)

| Endpoint | Method | Purpose | Request Body | Response |
| --- | --- | --- | --- | --- |
| /deck | POST | Create new deck | CreateDeckRequestDto | DeckDto |
| /deck/{deckId} | GET | Get deck details | - | DeckDto |
| /deck/{deckId} | PUT | Update deck metadata | UpdateDeckRequestDto | DeckDto |
| /deck/{deckId} | DELETE | Delete deck (owner only) | - | 200 OK |
| /deck/my-decks | GET | Get user’s owned/subscribed decks | - | List<DeckDto> |
| /deck/public | GET | Browse public deck marketplace | - | List<DeckDto> |
| /deck/{deckId}/fork | POST | Fork public deck to personal library | - | DeckDto (forked copy) |

**Example: Create Deck**

POST /api/v1/deck  
Authorization: Bearer {jwt\_token}  
Content-Type: application/json  
  
{  
 "title": "Spanish Business Vocabulary",  
 "description": "Essential terms for business meetings",  
 "isPublic": true  
}

**Example: Fork Deck**

POST /api/v1/deck/abc-123-def/fork  
Authorization: Bearer {jwt\_token}  
  
Response:  
{  
 "id": "xyz-456-ghi",  
 "title": "Fork of Spanish Business Vocabulary",  
 "description": "Essential terms for business meetings",  
 "isPublic": false,  
 "ownerId": "user-789",  
 "parentDeckId": "abc-123-def",  
 "createdDate": "2025-11-01T10:30:00Z",  
 "cardCount": 150,  
 "isSubscribed": false  
}

##### Card Management (/api/v1/card)

| Endpoint | Method | Purpose | Request Body | Response |
| --- | --- | --- | --- | --- |
| /card | POST | Add card to deck | CreateCardRequestDto | CardDto |
| /card/{cardId} | GET | Get card with progress | - | CardDto |
| /card/{cardId} | PUT | Update card content | UpdateCardRequestDto | CardDto |
| /card/{cardId} | DELETE | Delete card (creator only) | - | 200 OK |
| /card/{deckId}/cards | GET | Get all cards in deck | - | List<CardDto> |
| /card/all | GET | Get all user’s cards (table view) | - | List<CardTableRowDto> |
| /card/due | GET | Check if cards due | Query params | { hasDueCards: bool } |
| /card/next/{sessionId} | GET | Get next card in study session | - | CardDto |
| /card/review/{cardId} | POST | Submit review (FSRS update) | ReviewCardRequestDto | ReviewResponseDto |

**Query Parameters for /card/due:**

* deckId (optional): Filter by specific deck
* sortBy (optional): "createdDate" | "deckId" | "nextReviewDate"
* skill (optional): "Reading" | "Writing" | "Speaking" | "Listening"
* mode (optional): "Learn" | "Review" | "Cram"

**Example: Submit Review**

POST /api/v1/card/review/card-123  
Authorization: Bearer {jwt\_token}  
Content-Type: application/json  
  
{  
 "quality": 3 // 1=Again, 2=Hard, 3=Good, 4=Easy  
}  
  
Response:  
{  
 "card": {  
 "id": "card-123",  
 "front": "La manzana",  
 "back": "The apple",  
 "progress": {  
 "due": "2025-11-05T10:00:00Z", // Next review date  
 "stability": 5.2,  
 "difficulty": 6.8,  
 "state": 2, // Review state  
 "reps": 5 // Total repetitions  
 }  
 }  
}

##### Study Sessions (/api/v1/study)

| Endpoint | Method | Purpose | Request Body | Response |
| --- | --- | --- | --- | --- |
| /study/start | POST | Initialize study session | StartSessionRequestDto | { sessionId: guid } |
| /study/end/{sessionId} | POST | End session, get statistics | - | EndStudySessionResponseDto |
| /study/{sessionId} | GET | Get session details | - | StudySessionDto |
| /study/{sessionId}/check | GET | Check if session active | - | { active: bool } |
| /study/{sessionId}/stats | GET | Get live session statistics | - | CardStatsDto |

**Study Session Lifecycle:**

1. START SESSION → Creates session, populates Redis queues  
 ├─ Fetches due cards (FSRS Due < Tomorrow)  
 ├─ Fetches new cards (up to DailyNewCardLimit)  
 ├─ Separates into 3 queues: learning, new, review  
 └─ Returns sessionId  
  
2. GET NEXT CARD → Pop from priority queue  
 ├─ Priority: Learning > New > Review  
 ├─ Checks daily limits (new/review)  
 └─ Returns CardDto or null (session complete)  
  
3. SUBMIT REVIEW → gRPC call to FSRS service  
 ├─ Calculate next review date  
 ├─ Update UserCardProgress (difficulty, stability)  
 ├─ Increment daily counters  
 ├─ Re-queue if due today (learning → review transition)  
 └─ Update session stats (Redis cache)  
  
4. END SESSION → Finalize metrics  
 ├─ Save session stats to DB  
 ├─ Clear Redis queues  
 └─ Return summary (totalCardsReviewed, averageQuality)

**Example: Start Session**

POST /api/v1/study/start  
Authorization: Bearer {jwt\_token}  
Content-Type: application/json  
  
{  
 "deckId": "deck-123"  
}  
  
Response:  
{  
 "sessionId": "session-789"  
}

**Example: Get Session Stats**

GET /api/v1/study/session-789/stats  
Authorization: Bearer {jwt\_token}  
  
Response:  
{  
 "newCount": 8, // Cards user hasn't seen  
 "learningCount": 5, // Cards in learning phase (multiple reviews today)  
 "reviewCount": 23 // Cards due for review  
}

##### Deck Subscriptions (/api/v1/deck-subscription)

| Endpoint | Method | Purpose | Response |
| --- | --- | --- | --- |
| /deck-subscription/{deckId} | POST | Subscribe to public deck | DeckSubscriptionDto |
| /deck-subscription/{deckId} | DELETE | Unsubscribe from deck | 200 OK |
| /deck-subscription | GET | Get all subscribed decks | List<DeckSubscriptionDto> |

**Subscription vs. Fork:**

* **Subscribe:** Follow public deck, receive updates (read-only)
* **Fork:** Create personal copy, full edit permissions, no updates

##### Deck Versioning (/api/v1/deck-version)

| Endpoint | Method | Purpose | Response |
| --- | --- | --- | --- |
| /deck-version/deck/{deckId} | GET | Get version history | List<DeckVersionDto> |

**Version tracking enables:**

* Commit messages (“Added 50 medical terms”)
* Change attribution (ModifiedByUserId)
* Rollback capability (SnapshotJson)
* Pull request workflow (future feature)

#### Data Models

##### Core Entities

**Deck Entity:**

public class Deck  
{  
 public Guid Id { get; set; }  
 public string Title { get; set; }  
 public string? Description { get; set; }  
 public string? ImageUrl { get; set; }  
 public bool IsPublic { get; set; }  
 public Guid OwnerId { get; set; }  
 public Guid? ParentDeckId { get; set; } // For forked decks  
 public DateTime CreatedDate { get; set; }  
 public DateTime? LastReviewedDate { get; set; }  
 public string[] Tags { get; set; } // ["Spanish", "Business"]  
  
 // Navigation properties  
 public virtual Deck? ParentDeck { get; set; }  
 public virtual ICollection<Card> Cards { get; set; }  
 public virtual ICollection<DeckVersion> Versions { get; set; }  
 public virtual ICollection<DeckSubscription> Subscriptions { get; set; }  
}

**Card Entity:**

public class Card  
{  
 public Guid Id { get; set; }  
 public string Front { get; set; } // L2 word or phrase  
 public string Back { get; set; } // L1 translation/definition  
 public string? Hint { get; set; } // Mnemonic or grammar tip  
 public string? MediaUrl { get; set; } // Audio/image URL  
 public SkillType Skill { get; set; } // Reading/Writing/Speaking/Listening  
 public Guid DeckId { get; set; }  
 public DateTime CreatedDate { get; set; }  
 public Guid CreatorId { get; set; }  
  
 // Navigation properties  
 public virtual Deck Deck { get; set; }  
 public virtual ICollection<UserCardProgress> Progress { get; set; }  
}  
  
public enum SkillType  
{  
 Reading,  
 Writing,  
 Speaking,  
 Listening  
}

**UserCardProgress (extends FSRS Card):**

public class UserCardProgress : DotnetFSRS.Card  
{  
 public Guid Id { get; set; }  
 public Guid UserId { get; set; }  
 public Guid CardId { get; set; }  
 public int LastQuality { get; set; } // 1-4 (Again/Hard/Good/Easy)  
  
 // Inherited from FSRS Card:  
 // - Due (DateTime): Next review date  
 // - Stability (float): Memory strength  
 // - Difficulty (float): Card intrinsic difficulty (0-10)  
 // - State (enum): New, Learning, Review, Relearning  
 // - Reps (int): Total review count  
 // - Lapses (int): Times user clicked "Again"  
 // - LastReview (DateTime): Last review timestamp  
  
 public virtual Card Card { get; set; }  
}

**StudySession Entity:**

public class StudySession  
{  
 public Guid Id { get; set; }  
 public Guid UserId { get; set; }  
 public Guid DeckId { get; set; }  
 public DateTime StartTime { get; set; }  
 public DateTime? EndTime { get; set; }  
 public double AverageQuality { get; set; } // Mean quality (1-4)  
 public int TotalCardsReviewed { get; set; }  
  
 public virtual ICollection<ReviewedCard> ReviewedCards { get; set; }  
}  
  
public class ReviewedCard  
{  
 public Guid Id { get; set; }  
 public Guid StudySessionId { get; set; }  
 public Guid CardId { get; set; }  
 public int Quality { get; set; }  
 public DateTime ReviewedAt { get; set; }  
  
 public virtual StudySession StudySession { get; set; }  
 public virtual Card Card { get; set; }  
}

**UserSettings Entity:**

public class UserSettings  
{  
 public Guid Id { get; set; }  
 public Guid UserId { get; set; }  
 public int DailyNewCardLimit { get; set; } = 10;  
 public int DailyReviewLimit { get; set; } = 100;  
 public int NewCardsCompletedToday { get; set; }  
 public int ReviewsCompletedToday { get; set; }  
 public int RolloverHour { get; set; } = 4; // Day starts at 4 AM  
 public DateTime LastResetDate { get; set; }  
 public LearningMode PreferredMode { get; set; } // Learn/Review/Cram  
}  
  
public enum LearningMode  
{  
 Learn, // Only new cards (up to DailyNewCardLimit)  
 Review, // Prioritize reviews, then new cards (default)  
 Cram // All cards in deck, ignore FSRS (for testing)  
}

**DeckVersion Entity:**

public class DeckVersion  
{  
 public Guid Id { get; set; }  
 public Guid DeckId { get; set; }  
 public string ChangeDescription { get; set; } // Commit message  
 public Guid ModifiedByUserId { get; set; }  
 public DateTime ModifiedDate { get; set; }  
 public string? SnapshotJson { get; set; } // Full deck state (optional)  
  
 public virtual Deck Deck { get; set; }  
}

**DeckSubscription Entity:**

public class DeckSubscription  
{  
 public Guid Id { get; set; }  
 public Guid UserId { get; set; }  
 public Guid DeckId { get; set; }  
 public DateTime SubscribedDate { get; set; }  
  
 public virtual Deck Deck { get; set; }  
}

#### FSRS Integration (gRPC)

**Proto Definition (Vocab.proto):**

syntax = "proto3";  
package vocab;  
  
service VocabService {  
 rpc ReviewCard (ReviewRequest) returns (ReviewResponse) {}  
}  
  
message ReviewRequest {  
 Card card = 1; // Current FSRS state  
 int32 quality = 2; // 1-4 (Again/Hard/Good/Easy)  
 google.protobuf.Int32Value review\_duration = 3; // Seconds spent reviewing  
}  
  
message ReviewResponse {  
 Card card = 1; // Updated FSRS state  
 ReviewLog review\_log = 2;  
}  
  
message Card {  
 int32 state = 1; // 0=New, 1=Learning, 2=Review, 3=Relearning  
 google.protobuf.Int32Value step = 2; // Current step in learning phase  
 google.protobuf.FloatValue stability = 3; // Memory stability  
 google.protobuf.FloatValue difficulty = 4; // Intrinsic difficulty (0-10)  
 google.protobuf.Timestamp due = 5; // Next review date  
 google.protobuf.Timestamp last\_review = 6;  
}  
  
message ReviewLog {  
 int32 rating = 2;  
 google.protobuf.Timestamp review\_datetime = 3;  
 google.protobuf.Int32Value review\_duration = 4;  
}

**FSRS State Machine:**

NEW → (First review)  
 ├─ Again (1) → LEARNING (step 1, due in 1 min)  
 ├─ Hard (2) → LEARNING (step 1, due in 5 min)  
 ├─ Good (3) → LEARNING (step 2, due in 10 min)  
 └─ Easy (4) → REVIEW (due in 4 days)  
  
LEARNING → (Subsequent reviews within day)  
 ├─ Again → LEARNING (reset to step 1)  
 ├─ Hard → LEARNING (same step, shorter interval)  
 ├─ Good → LEARNING (next step)  
 └─ Easy → REVIEW (graduate early)  
  
REVIEW → (Mature cards)  
 ├─ Again → RELEARNING (due in 10 min)  
 ├─ Hard → REVIEW (interval × 1.2)  
 ├─ Good → REVIEW (interval × 2.5)  
 └─ Easy → REVIEW (interval × 5.0)  
  
RELEARNING → (Lapsed cards)  
 ├─ Again → RELEARNING (step 1)  
 ├─ Hard → RELEARNING (same step)  
 ├─ Good → RELEARNING (next step)  
 └─ Easy → REVIEW (graduate)

**gRPC Call Flow:**

// In CardService.SubmitReviewAsync()  
var reviewRequest = new ReviewRequest  
{  
 Card = \_mapper.Map<Vocab.Card>(progress), // Map UserCardProgress → gRPC Card  
 Quality = request.Quality // 1-4  
};  
  
var reviewResponse = await \_vocabServiceClient.ReviewCardAsync(reviewRequest);  
  
progress = \_mapper.Map(reviewResponse.Card, progress); // Update UserCardProgress  
await \_dbContext.SaveChangesAsync();

**FSRS Advantages:**

* **Personalized:** Learns user’s forgetting curve (stability parameter adapts)
* **Efficient:** Fewer reviews for same retention vs. SM-2
* **Research-backed:** Based on 20,000+ users’ review data

#### FSRS Configuration & Parameters (Anki-Compatible)

**Current Configuration Location:** inclusive/config.json

{  
 "server\_port": 40051,  
 "fsrs": {  
 "request\_retention": 0.9,  
 "maximum\_interval": 36500,  
 "w": [  
 0.4072, 1.1829, 3.1262, 15.4722, 7.2102, 0.5316, 1.0651, 0.0234, 1.616,  
 0.1544, 1.0824, 1.9813, 0.0953, 0.2975, 2.2042, 0.2407, 2.9466, 0.5034,  
 0.6567  
 ],  
 "enable\_fuzz": true,  
 "enable\_short\_term": true  
 }  
}

##### FSRS Parameters Explained

**1. request\_retention (Default: 0.9)**

* **What it is**: Target retention rate (90%)
* **Anki equivalent**: Desired retention in deck options
* **Effect**: Higher values = shorter intervals (more frequent reviews)
* **Range**: 0.7 - 0.97
* **Recommendation**:
  + 0.9 (90%) - Balanced (default, same as Anki)
  + 0.85 (85%) - Fewer reviews, longer intervals
  + 0.95 (95%) - More reviews, better retention

**2. maximum\_interval (Default: 36500 days)**

* **What it is**: Maximum number of days between reviews (~100 years)
* **Anki equivalent**: Maximum interval in deck options
* **Effect**: Caps how long a card can be scheduled for
* **Recommendation**:
  + 36500 - Effectively unlimited (default)
  + 365 - Maximum 1 year between reviews
  + 180 - Maximum 6 months for professional knowledge

**3. w (Weights Array)**

* **What it is**: 19 algorithm parameters that define the scheduling behavior
* **Anki equivalent**: FSRS-5 algorithm weights
* **Current values**: Optimized default parameters from FSRS research
* **Source**: Based on analysis of millions of Anki reviews
* **DO NOT MODIFY** unless you have custom optimization data

**Weight Parameters (for reference):**

w[0-3]: Initial stability for Again, Hard, Good, Easy on new cards  
w[4-7]: Initial difficulty for Again, Hard, Good, Easy  
w[8]: Difficulty decay rate  
w[9]: Stability increase for Easy bonus  
w[10]: Hard penalty  
w[11-14]: Stability factors for ratings  
w[15-16]: Retrievability factors  
w[17-18]: Advanced scheduling parameters

**4. enable\_fuzz (Default: true)**

* **What it is**: Randomizes review times slightly (±5%)
* **Anki equivalent**: Same as Anki’s fuzzing
* **Benefits**:
  + Prevents many cards from being due on the same day
  + More natural review distribution
  + Reduces workload spikes
* **Example**: 10-day interval becomes 9.5-10.5 days randomly

**5. enable\_short\_term (Default: true)**

* **What it is**: Optimizes scheduling for short-term learning
* **Benefits**:
  + Better handling of learning phase cards
  + More accurate predictions for new cards
  + Improved graduation timing
* **Keep enabled** for best Anki-like experience

##### Key FSRS Concepts

**1. Stability**

* Represents how well you know the card
* Higher stability = longer intervals
* Increased by successful reviews
* Decreased by failures
* Range: 0.1 days to maximum\_interval

**2. Difficulty**

* Inherent difficulty of the card
* Based on your historical performance
* Range: 1 (easy) to 10 (hard)
* Affects interval calculations
* Updates gradually over time

**3. Retrievability**

* Probability you’ll recall the card
* Calculated from stability and time elapsed
* Drives the scheduling algorithm
* Target: request\_retention (e.g., 90%)

##### Interval Examples

**New Card First Review:**

Rating Again: Learning (10 minutes)  
Rating Hard: Learning (10 minutes)  
Rating Good: Learning → 1 day  
Rating Easy: Review → 4 days (graduate immediately)

**Learning Card (1 day interval):**

Rating Again: Back to step 1 (10 minutes)  
Rating Hard: Same step or slight increase  
Rating Good: Graduate to Review (2-3 days)  
Rating Easy: Graduate to Review (4 days)

**Review Card (10 days interval):**

Rating Again: Relearning (10 minutes)  
Rating Hard: Review (5-8 days)  
Rating Good: Review (20-30 days)  
Rating Easy: Review (25-40 days)

**Review Card (100 days interval):**

Rating Again: Relearning (10 minutes)  
Rating Hard: Review (50-80 days)  
Rating Good: Review (200-300 days)  
Rating Easy: Review (250-400 days)

##### Comparison with Anki FSRS

| Feature | Anki FSRS | LineByLine | Status |
| --- | --- | --- | --- |
| Algorithm | FSRS v5.0 | FSRS v6.1.0 | ✅ Better |
| Card States | 4 states | 4 states | ✅ Match |
| Quality Ratings | 1-4 | 1-4 | ✅ Match |
| Retention Target | 0.9 (90%) | 0.9 (90%) | ✅ Match |
| Maximum Interval | 36500 | 36500 days | ✅ Match |
| Fuzzing | ✅ Yes | ✅ Yes | ✅ Match |
| Short-term | ✅ Yes | ✅ Yes | ✅ Match |
| Learning Steps | Manual | FSRS-optimized | ✅ Better |
| Graduation | Manual | Automatic | ✅ Better |

##### Customizing FSRS Parameters

**Conservative Learning (More Reviews):**

{  
 "request\_retention": 0.95,  
 "maximum\_interval": 180,  
 "enable\_fuzz": true,  
 "enable\_short\_term": true  
}

**Aggressive Learning (Fewer Reviews):**

{  
 "request\_retention": 0.85,  
 "maximum\_interval": 36500,  
 "enable\_fuzz": true,  
 "enable\_short\_term": true  
}

**Exam Preparation (Short-term):**

{  
 "request\_retention": 0.95,  
 "maximum\_interval": 30,  
 "enable\_fuzz": false,  
 "enable\_short\_term": true  
}

**Language Learning (Balanced):**

{  
 "request\_retention": 0.9,  
 "maximum\_interval": 365,  
 "enable\_fuzz": true,  
 "enable\_short\_term": true  
}

##### When to Change Parameters

**✅ Safe to Change:**

* request\_retention - Adjust based on workload
* maximum\_interval - Based on material type
* enable\_fuzz - Personal preference

**⚠️ Advanced Users Only:**

* w (weights) - Only if you have custom optimization data
* Requires analysis of thousands of reviews
* Use FSRS optimizer: https://github.com/open-spaced-repetition/fsrs-optimizer

**❌ Don’t Change:**

* enable\_short\_term - Keep enabled (better learning)

##### Applying Configuration Changes

**1. Edit Configuration:**

# Edit inclusive/config.json  
nano inclusive/config.json

**2. Restart FSRS Service:**

# Using Docker  
docker-compose restart inclusive  
  
# Or rebuild if needed  
docker-compose up -d --build inclusive

**3. Verify Changes:**

# Check logs  
docker-compose logs inclusive | tail -n 50  
  
# Should show: "Server starting on port 40051..."

##### Monitoring & Optimization

After 1000+ reviews, you can analyze your data:

1. Export your review history
2. Use FSRS optimizer: https://github.com/open-spaced-repetition/fsrs-optimizer
3. Get custom weights optimized for YOUR learning patterns
4. Update w array in config.json

**Expected Results (90% retention):**

* **New cards**: 10-15 reviews to reach 100+ day intervals
* **Learning phase**: 1-3 sessions
* **Workload**: ~10 reviews per 100 cards per day (steady state)
* **Retention**: ~90% accuracy on reviews

##### Troubleshooting

**Cards reviewing too often:** → Decrease request\_retention to 0.85

**Cards reviewing too rarely (forgetting):** → Increase request\_retention to 0.95

**Too many cards due on same day:** → Ensure enable\_fuzz is true

**New cards graduating too quickly/slowly:** → Keep defaults, FSRS auto-optimizes

**Very old cards (years):** → Adjust maximum\_interval if needed

#### Redis-Based Session Management

**Why Redis?**

* **Speed:** Sub-millisecond queue operations (LPUSH/RPOP)
* **Session isolation:** Each study session has dedicated queues
* **Expiration:** Auto-cleanup after 24 hours (prevents memory leak)
* **Real-time stats:** Instant card count updates without DB queries

**Queue Structure:**

Redis Keys per Session:  
├─ new:{sessionId} → List of new cards (JSON)  
├─ learning:{sessionId} → List of learning cards (JSON)  
├─ review:{sessionId} → List of review cards (JSON)  
└─ stats:{sessionId} → Cached stats (JSON, TTL=24h)  
  
Example:  
new:abc-123-def → ["{ 'id': 'card-1', 'front': 'hello', ... }", ...]

**Session Initialization:**

// StudySessionService.StartSessionAsync()  
1. Check user settings (DailyNewCardLimit, DailyReviewLimit)  
2. Query PostgreSQL:  
 - New cards: Cards without UserCardProgress  
 - Due cards: UserCardProgress.Due < Tomorrow  
3. Separate due cards by state:  
 - Learning: State = Learning or Relearning  
 - Review: State = Review  
4. Push to Redis:  
 - LPUSH new:{sessionId} [card1, card2, ...]  
 - LPUSH learning:{sessionId} [card3, card4, ...]  
 - LPUSH review:{sessionId} [card5, card6, ...]  
5. Cache stats:  
 - SETEX stats:{sessionId} 86400 { newCount, learningCount, reviewCount }

**Card Retrieval (Priority System):**

// StudySessionService.GetNextCardAsync()  
Priority: Learning > New > Review  
  
1. Try RPOP learning:{sessionId}  
 ├─ If card returned:  
 │ ├─ Check: Due < Tomorrow && ReviewsCompletedToday < DailyReviewLimit  
 │ └─ Return card OR retry next queue  
 └─ If empty, try next queue  
  
2. Try RPOP new:{sessionId}  
 ├─ Check: NewCardsCompletedToday < DailyNewCardLimit  
 └─ Return card OR retry next queue  
  
3. Try RPOP review:{sessionId}  
 ├─ Check: Due < Tomorrow && ReviewsCompletedToday < DailyReviewLimit  
 └─ Return card OR return null (session complete)  
  
4. If null:  
 ├─ Set session.EndTime = NOW  
 └─ Clear Redis queues

**Dynamic Re-Queuing:**

// After review submission  
If (card state changed && next review due today):  
 If (Learning → Review):  
 LPUSH review:{sessionId} card  
 Else If (Review → Learning):  
 LPUSH learning:{sessionId} card

**Session Stats Caching:**

// Cached to avoid expensive COUNT queries  
{  
 "newCount": 8,  
 "learningCount": 5,  
 "reviewCount": 23  
}  
  
// Updated after each review:  
- NewCount decrements when new card reviewed  
- LearningCount increments/decrements based on state transitions  
- ReviewCount decrements when review card completed

**Session Cleanup:**

// StudySessionService.EndSessionAsync()  
1. Save session stats to PostgreSQL:  
 - TotalCardsReviewed  
 - AverageQuality (mean of all ratings 1-4)  
 - ReviewedCards (audit trail)  
2. Clear Redis keys:  
 - DEL new:{sessionId}  
 - DEL learning:{sessionId}  
 - DEL review:{sessionId}  
 - DEL stats:{sessionId}

#### Daily Limits & Rollover Logic

**User Settings:**

* DailyNewCardLimit: Max new cards per day (default: 10)
* DailyReviewLimit: Max reviews per day (default: 100)
* RolloverHour: Day starts at this hour UTC (default: 4 AM)
  + Why 4 AM? Users study late night; rollover at midnight would interrupt sessions

**Counter Reset Logic:**

private void ResetDailyCounters(UserSettings settings)  
{  
 var (startOfToday, \_) = GetDayBoundaries(settings);  
  
 if (settings.LastResetDate < startOfToday)  
 {  
 settings.NewCardsCompletedToday = 0;  
 settings.ReviewsCompletedToday = 0;  
 settings.LastResetDate = DateTime.UtcNow;  
 }  
}  
  
private (DateTime, DateTime) GetDayBoundaries(UserSettings settings)  
{  
 DateTime now = DateTime.UtcNow;  
 DateTime startOfToday = now.Date.AddHours(settings.RolloverHour);  
  
 // If before rollover hour, "today" is actually yesterday's session  
 if (now < startOfToday)  
 {  
 startOfToday = startOfToday.AddDays(-1);  
 }  
  
 DateTime startOfTomorrow = startOfToday.AddDays(1);  
 return (startOfToday, startOfTomorrow);  
}

**Example:**

RolloverHour = 4  
Current Time: Nov 1, 2025 02:30 AM UTC  
  
GetDayBoundaries():  
 startOfToday = Oct 31, 2025 04:00 AM UTC  
 startOfTomorrow = Nov 1, 2025 04:00 AM UTC  
  
Cards due: UserCardProgress.Due < Nov 1, 04:00 AM  
// This means cards due at 3:59 AM on Nov 1 are still "due today"

**Limit Enforcement:**

// In SubmitReviewAsync()  
bool isNewCard = progress == null;  
  
if (isNewCard && settings.NewCardsCompletedToday >= settings.DailyNewCardLimit)  
 throw new InvalidOperationException("Daily new card limit reached");  
  
if (!isNewCard && settings.ReviewsCompletedToday >= settings.DailyReviewLimit)  
 throw new InvalidOperationException("Daily review limit reached");  
  
// After successful review:  
if (isNewCard)  
 settings.NewCardsCompletedToday++;  
else  
 settings.ReviewsCompletedToday++;

#### Learning Modes

**1. Review Mode (Default)**

LearningMode.Review:  
 Priority: Due reviews > New cards  
 Limits: Respect DailyNewCardLimit + DailyReviewLimit  
 Queue order: learning → new → review

**Use case:** Balanced learning (maintain existing knowledge + add new)

**2. Learn Mode**

LearningMode.Learn:  
 Priority: New cards only  
 Limits: DailyNewCardLimit (ignore reviews)  
 Queue order: new only

**Use case:** Rapid vocabulary building (beginner phase)

**3. Cram Mode**

LearningMode.Cram:  
 Priority: All cards in deck  
 Limits: None (ignore FSRS due dates + daily limits)  
 Sort by: CreatedDate or DeckId

**Use case:** Pre-exam review, deck preview

**Switching Modes:**

PUT /api/v1/settings  
{  
 "preferredMode": "Cram"  
}

#### Database Schema (PostgreSQL)

**Key Tables:**

-- Users managed by auth service (not shown)  
  
-- Decks  
CREATE TABLE Decks (  
 Id UUID PRIMARY KEY,  
 Title VARCHAR(255) NOT NULL,  
 Description TEXT,  
 ImageUrl VARCHAR(500),  
 IsPublic BOOLEAN DEFAULT FALSE,  
 OwnerId UUID NOT NULL,  
 ParentDeckId UUID, -- FK to Decks.Id (for forks)  
 CreatedDate TIMESTAMP DEFAULT NOW(),  
 LastReviewedDate TIMESTAMP,  
 Tags VARCHAR(50)[], -- PostgreSQL array  
 FOREIGN KEY (ParentDeckId) REFERENCES Decks(Id) ON DELETE SET NULL  
);  
  
-- Cards  
CREATE TABLE Cards (  
 Id UUID PRIMARY KEY,  
 Front VARCHAR(500) NOT NULL,  
 Back VARCHAR(500) NOT NULL,  
 Hint TEXT,  
 MediaUrl VARCHAR(500),  
 Skill INT NOT NULL, -- Enum: 0=Reading, 1=Writing, 2=Speaking, 3=Listening  
 DeckId UUID NOT NULL,  
 CreatedDate TIMESTAMP DEFAULT NOW(),  
 CreatorId UUID NOT NULL,  
 FOREIGN KEY (DeckId) REFERENCES Decks(Id) ON DELETE CASCADE  
);  
  
-- User Card Progress (FSRS state)  
CREATE TABLE UserCardProgress (  
 Id UUID PRIMARY KEY,  
 UserId UUID NOT NULL,  
 CardId UUID NOT NULL,  
  
 -- FSRS fields  
 Due TIMESTAMP NOT NULL,  
 Stability FLOAT,  
 Difficulty FLOAT,  
 State INT NOT NULL, -- 0=New, 1=Learning, 2=Review, 3=Relearning  
 LastReview TIMESTAMP,  
 Reps INT DEFAULT 0,  
 Lapses INT DEFAULT 0,  
 ElapsedDays INT DEFAULT 0,  
 ScheduledDays INT DEFAULT 0,  
  
 LastQuality INT DEFAULT 0,  
  
 FOREIGN KEY (CardId) REFERENCES Cards(Id) ON DELETE CASCADE,  
 UNIQUE(UserId, CardId)  
);  
CREATE INDEX idx\_user\_due ON UserCardProgress(UserId, Due);  
  
-- Study Sessions  
CREATE TABLE StudySessions (  
 Id UUID PRIMARY KEY,  
 UserId UUID NOT NULL,  
 DeckId UUID NOT NULL,  
 StartTime TIMESTAMP NOT NULL,  
 EndTime TIMESTAMP,  
 AverageQuality FLOAT DEFAULT 0,  
 TotalCardsReviewed INT DEFAULT 0,  
 FOREIGN KEY (DeckId) REFERENCES Decks(Id) ON DELETE CASCADE  
);  
  
CREATE TABLE ReviewedCards (  
 Id UUID PRIMARY KEY,  
 StudySessionId UUID NOT NULL,  
 CardId UUID NOT NULL,  
 Quality INT NOT NULL,  
 ReviewedAt TIMESTAMP NOT NULL,  
 FOREIGN KEY (StudySessionId) REFERENCES StudySessions(Id) ON DELETE CASCADE,  
 FOREIGN KEY (CardId) REFERENCES Cards(Id) ON DELETE CASCADE  
);  
  
-- User Settings  
CREATE TABLE UserSettings (  
 Id UUID PRIMARY KEY,  
 UserId UUID NOT NULL UNIQUE,  
 DailyNewCardLimit INT DEFAULT 10,  
 DailyReviewLimit INT DEFAULT 100,  
 NewCardsCompletedToday INT DEFAULT 0,  
 ReviewsCompletedToday INT DEFAULT 0,  
 RolloverHour INT DEFAULT 4,  
 LastResetDate TIMESTAMP DEFAULT NOW(),  
 PreferredMode INT DEFAULT 1 -- 0=Learn, 1=Review, 2=Cram  
);  
  
-- Deck Subscriptions  
CREATE TABLE DeckSubscriptions (  
 Id UUID PRIMARY KEY,  
 UserId UUID NOT NULL,  
 DeckId UUID NOT NULL,  
 SubscribedDate TIMESTAMP DEFAULT NOW(),  
 FOREIGN KEY (DeckId) REFERENCES Decks(Id) ON DELETE CASCADE,  
 UNIQUE(UserId, DeckId)  
);  
  
-- Deck Versions (Git-like commits)  
CREATE TABLE DeckVersions (  
 Id UUID PRIMARY KEY,  
 DeckId UUID NOT NULL,  
 ChangeDescription VARCHAR(500),  
 ModifiedByUserId UUID NOT NULL,  
 ModifiedDate TIMESTAMP DEFAULT NOW(),  
 SnapshotJson TEXT, -- Optional full deck snapshot  
 FOREIGN KEY (DeckId) REFERENCES Decks(Id) ON DELETE CASCADE  
);

#### Performance Optimizations

**1. Eager Loading (Avoid N+1 Queries):**

var cards = await \_dbContext.Cards  
 .Include(c => c.Deck)  
 .Include(c => c.Progress.Where(p => p.UserId == userId))  
 .ToListAsync();

**2. Pagination (Future):**

var decks = await \_dbContext.Decks  
 .Where(d => d.IsPublic)  
 .Skip((page - 1) \* pageSize)  
 .Take(pageSize)  
 .ToListAsync();

**3. Redis Caching:**

* Session card queues (avoid repeated DB queries)
* Session stats (updated after each review, not recomputed)

**4. Index Optimization:**

CREATE INDEX idx\_user\_due ON UserCardProgress(UserId, Due);  
CREATE INDEX idx\_deck\_public ON Decks(IsPublic) WHERE IsPublic = TRUE;  
CREATE INDEX idx\_card\_deck ON Cards(DeckId);

**5. Transaction Isolation:**

// In SubmitReviewAsync() - prevent race conditions  
using var transaction = await \_dbContext.Database.BeginTransactionAsync();  
try  
{  
 // Update progress  
 // Update settings counters  
 // Re-queue card if needed  
 await \_dbContext.SaveChangesAsync();  
 await transaction.CommitAsync();  
}  
catch  
{  
 await transaction.RollbackAsync();  
 throw;  
}

#### Error Handling & Validation

**Common Error Scenarios:**

1. **Invalid Review Quality:**

if (request.Quality < 1 || request.Quality > 4)  
 throw new ArgumentException("Quality must be 1-4");

1. **Daily Limit Exceeded:**

if (settings.NewCardsCompletedToday >= settings.DailyNewCardLimit)  
 throw new InvalidOperationException("Daily new card limit reached");

1. **Card Not Due:**

if (progress.Due >= startOfTomorrow)  
 throw new InvalidOperationException("Card not due today");

1. **Unauthorized Access:**

var deck = await \_dbContext.Decks  
 .FirstOrDefaultAsync(d => d.Id == deckId &&  
 (d.IsPublic || d.OwnerId == userId || d.Subscriptions.Any(s => s.UserId == userId)));  
if (deck == null)  
 return Unauthorized();

1. **Session Mismatch:**

if (card.DeckId != activeSession.DeckId)  
 throw new InvalidOperationException("Card does not belong to active session's deck");

**Logging Strategy:**

\_logger.LogInformation("User {UserId} started session {SessionId} on deck {DeckId}",  
 userId, sessionId, deckId);  
  
\_logger.LogWarning("Session {SessionId}: Invalid card JSON or deck mismatch", sessionId);  
  
\_logger.LogError(ex, "Error fetching next card for session {SessionId}", sessionId);

#### Future Enhancements

**1. Bulk Import:**

POST /api/v1/card/bulk  
{  
 "deckId": "deck-123",  
 "cards": [  
 { "front": "hello", "back": "hola" },  
 { "front": "goodbye", "back": "adiós" }  
 // ... up to 1000 cards  
 ]  
}

**2. Card Contributions (Pull Requests):**

POST /api/v1/contribution  
{  
 "deckId": "public-deck-123",  
 "cardId": "card-456",  
 "suggestedChanges": {  
 "front": "corrected spelling",  
 "back": "better translation"  
 },  
 "rationale": "The current translation is outdated"  
}

**3. Deck Analytics:**

GET /api/v1/deck/deck-123/analytics  
  
Response:  
{  
 "totalSubscribers": 1240,  
 "averageRetention": 0.87, // 87% cards retained after 30 days  
 "completionRate": 0.65, // 65% users complete all cards  
 "popularCards": [ // Most favorited/bookmarked  
 { "cardId": "...", "votes": 89 }  
 ]  
}

**4. AI-Generated Hints:**

POST /api/v1/card/card-123/generate-hint  
  
Response:  
{  
 "hint": "Think of 'manzana' like 'Manzanita' (little apple brand)"  
}

**5. Voice Recording for Cards:**

POST /api/v1/card/card-123/upload-audio  
Content-Type: multipart/form-data  
  
Response:  
{  
 "mediaUrl": "https://cdn.linebyline.com/audio/card-123.mp3"  
}

#### Security Considerations

**1. JWT Authentication:**

* All endpoints require [Authorize] attribute
* Token expiration: 24 hours (configurable)
* Refresh token mechanism (separate endpoint)

**2. Authorization Checks:**

// Deck ownership  
var deck = await \_dbContext.Decks  
 .FirstOrDefaultAsync(d => d.Id == deckId && d.OwnerId == userId);  
  
// Card access (owner, subscriber, or public)  
var card = await \_dbContext.Cards  
 .Include(c => c.Deck)  
 .FirstOrDefaultAsync(c => c.Id == cardId &&  
 (c.Deck.IsPublic ||  
 c.Deck.OwnerId == userId ||  
 c.Deck.Subscriptions.Any(s => s.UserId == userId)));

**3. Rate Limiting (TODO):**

// Prevent abuse of public deck creation  
builder.Services.AddRateLimiter(options =>  
{  
 options.AddFixedWindowLimiter("deck-creation", opt =>  
 {  
 opt.Window = TimeSpan.FromHours(1);  
 opt.PermitLimit = 5; // Max 5 decks per hour  
 });  
});

**4. Input Validation:**

// Sanitize user input (prevent XSS)  
deck.Title = HtmlEncoder.Default.Encode(request.Title);  
card.Front = HtmlEncoder.Default.Encode(request.Front);

**5. SQL Injection Prevention:**

* Entity Framework parameterizes all queries automatically
* Never use raw SQL with string concatenation

## Speaking Module

### Overview

Move from silent learner to confident speaker through progressive output practice.

### Tools

#### 1. Shadowing Studio

**Purpose:** Master pronunciation, intonation, and rhythm.

**Content Types:**

* **Dialogues:** Movie/TV scenes (2-person conversations)
* **Monologues:** TED Talks, speeches (1-person)
* **Music:** Songs with clear lyrics (karaoke mode)
* **Podcasts:** News clips, storytelling

**Interface:**

┌────────────────────────────────────────────┐  
│ 🎬 Friends S01E01 - Coffee Shop Scene │  
│ ──────────────────────────────────────────│  
│ │  
│ [====●================] 0:32 / 1:45 │  
│ │  
│ 🔊 "How you doin'?" │  
│ 👂 Listen │ 🎤 Shadow │ 📊 Compare │  
│ │  
│ Transcript (Toggle): │  
│ Rachel: "How you doin'?" │  
│ Joey: "How YOU doin'?" │  
└────────────────────────────────────────────┘

**Shadowing Modes:**

1. **Chorus Mode:** Audio + your voice simultaneously
2. **Delay Mode:** Hear phrase → repeat 1s later
3. **Solo Mode:** Muted audio, you fill the silence

**AI Feedback:**

* Pronunciation score (0-100)
* Highlighted mispronounced phonemes
* Rhythm match (visual waveform comparison)

#### 2. Conversation Simulator

**Purpose:** Practice real dialogues without human pressure.

**How It Works:**

AI Persona: Maria (Spanish teacher, friendly, patient)  
  
Maria: "Hola! ¿Cómo estás hoy?"  
 ↓  
[You speak your response via microphone]  
 ↓  
AI transcribes + analyzes your Spanish  
 ↓  
Maria: "¡Bien! Escuché que dijiste 'estoy cansado'.  
 ¿Trabajaste mucho hoy?"

**Difficulty Levels:**

* **A1:** Simple Q&A (name, hobbies, weather)
* **B1:** Roleplay (restaurant, shopping, directions)
* **C1:** Debate topics (politics, philosophy, ethics)

**Adaptive Conversation:**

* AI detects hesitation → simplifies next question
* AI detects fluency → increases complexity

#### 3. Voice Journaling

**Purpose:** Build speaking confidence through daily reflection.

**Prompt Examples:**

* “Describe your day in 2 minutes.”
* “What’s a childhood memory?”
* “Convince me to visit your hometown.”

**AI Analysis:**

* Speech rate (words per minute)
* Filler words (um, uh, like) frequency
* Grammar errors (auto-detected)
* Vocabulary diversity (unique words used)

**Progress Graph:**

Speaking Fluency Over 30 Days  
  
150 wpm ┤ ╭─●  
 │ ╭───╯  
100 wpm ┤ ╭──●──╯  
 │ ╭──●───╯  
 50 wpm ┤ ●─●───╯  
 └────────────────────────────  
 Day 1 10 20 30

## Writing Module

### Overview

Develop written fluency from sentence construction to essay composition.

### Tools

#### 1. Sentence Builder

**Purpose:** Learn grammar patterns through guided construction.

**Exercise Types:**

**Scrambled Sentences:**

Words: [el, manzana, comí, ayer, la]  
Build: "Ayer comí la manzana."  
Feedback: ✅ Perfect! (Subject-Verb-Object order)

**Fill-in-the-Blank:**

"Gestern \_\_\_\_ ich einen Apfel." (essen)  
Answer: "Gestern aß ich einen Apfel."  
Feedback: ✅ Correct past tense (Präteritum)

**Translation (Input → Output):**

L1: "I would have gone if I had known."  
L2: [User types French translation]  
AI: Checks grammar + suggests alternatives

#### 2. Composition Studio

**Purpose:** Write longer texts (paragraphs, essays) with AI feedback.

**Workflow:**

1. Choose topic: "Describe your ideal weekend"  
2. Write freely (200-500 words)  
3. Submit for analysis  
4. Receive feedback:  
 - Grammar errors (highlighted)  
 - Vocabulary suggestions (synonyms for overused words)  
 - Structure critique (intro-body-conclusion)  
5. Revise + resubmit

**AI Feedback Example:**

Original: "I like very much pizza. Is delicious."  
  
Feedback:  
❌ Word order: "I like pizza very much." (adverb placement)  
❌ Article missing: "It is delicious."  
💡 Richer vocabulary: "I adore pizza. It's delectable."

#### 3. Daily Writing Challenges

**Purpose:** Build writing habit through micro-tasks.

**Examples:**

* **Monday:** Write 3 sentences using today’s vocabulary
* **Wednesday:** Describe this image in 50 words (photo provided)
* **Friday:** Write a letter to your future self (100 words)

**Gamification:**

* 7-day writing streak = +500 XP
* Monthly challenge winner = Badge + featured on homepage

## Reading Module

### Overview

Massive comprehensible input through graded texts and authentic content.

### Features

#### 1. Library (Graded Readers)

**Content Categories:**

* Short stories (5-10 min reads)
* News articles (adapted for learners)
* Book chapters (Harry Potter, classics in L2)
* Blog posts (travel, tech, lifestyle)

**Difficulty Grading:**

AI analyzes text:  
- Known vocabulary % (based on your vocab cards)  
- Sentence complexity (clauses per sentence)  
- Grammar structures (tenses, subjunctive, etc.)  
  
Result:  
🟢 Comfortable (98%+ known words)  
🟡 Learning Zone (90-97% known)  
🔴 Challenging (<90% known)

**Reading Interface:**

┌─────────────────────────────────────────────┐  
│ 📖 El Principito - Capítulo 1 │  
│ Level: B1 | 📊 92% known words │  
├─────────────────────────────────────────────┤  
│ │  
│ Cuando yo tenía seis años vi una vez una │  
│ magnífica [lámina] en un libro... │  
│ ▲ │  
│ └─ Click word → see definition │  
│ + Add to deck (one tap) │  
│ │  
│ [Previous] [Bookmark] [Next] │  
└─────────────────────────────────────────────┘

**Interactive Features:**

* **Click-to-define:** Hover any word → instant translation
* **Add to deck:** Save unknown words → auto-creates flashcard
* **Text-to-speech:** Hear entire text (adjustable speed)
* **Progress tracking:** Bookmarks, reading time, words learned

#### 2. Sentence Mining Tool

**Purpose:** Extract vocabulary from authentic content YOU care about.

**Workflow:**

1. Paste text (article, lyrics, book passage)  
2. AI highlights words not in your deck  
3. Select words to learn  
4. AI generates flashcards with:  
 - Word + definition  
 - Original sentence (context)  
 - Audio pronunciation  
5. Cards added to deck "Mined from [Source Name]"

**Example:**

Input text: "Le chat noir dormait sur le canapé."  
  
Unknown words: canapé  
  
Generated card:  
Front: "canapé"  
Back: "sofa / couch"  
Context: "Le chat noir dormait sur le canapé."  
Audio: [🔊 native speaker]

#### 3. Parallel Texts

**Purpose:** Read L2 text with optional L1 translation side-by-side.

**Interface:**

┌──────────────────┬──────────────────┐  
│ Spanish (L2) │ English (L1) │  
├──────────────────┼──────────────────┤  
│ Era una noche │ It was a dark │  
│ oscura y │ and stormy │  
│ tormentosa. │ night. │  
│ │ │  
│ [Hidden by default → │  
│ Click to reveal if stuck] │  
└──────────────────┴──────────────────┘

**Best Practice:**

* Read L2 paragraph first (comprehension test)
* Only check L1 if completely lost
* Goal: Wean off L1 over time

## Listening Module

### Overview

Train your ear through progressive listening challenges—from scripted dialogues to authentic podcasts.

### Tools

#### 1. Dialogue Practice

**Purpose:** Master conversational listening.

**Content:**

* Coffee shop orders
* Doctor appointments
* Job interviews
* Casual friend chats

**Exercise Flow:**

1. Listen to 30s dialogue (no text)  
2. Answer comprehension questions:  
 - "What did Maria order?"  
 - "Why is Carlos late?"  
3. Listen again with transcript  
4. Repeat until 80%+ comprehension

**Difficulty Adjustments:**

* Speed: 0.7x (beginner) → 1.0x (intermediate) → 1.3x (advanced)
* Accent: Standard → regional dialects
* Background noise: Clean audio → café ambiance

#### 2. Podcast Library

**Purpose:** Consume authentic content for advanced learners.

**Categories:**

* **Learner Podcasts:** Slow speech, clear enunciation (Coffee Break Spanish, etc.)
* **Native Content:** True crime, comedy, news (real-world listening)

**Features:**

* Transcript toggle (on/off)
* Adjustable playback speed
* Bookmarking (save position + timestamp note)
* Vocabulary extraction (mine words from episodes)

**Recommended Strategy:**

Beginner (A1-A2):  
 └─ Learner podcasts only (100% comprehension goal)  
  
Intermediate (B1-B2):  
 └─ 70% learner, 30% native content (focus on gist)  
  
Advanced (C1-C2):  
 └─ 100% native content (full comprehension + idioms)

#### 3. Dictation Exercises

**Purpose:** Test listening precision.

**How It Works:**

1. Hear sentence: "Je voudrais un café, s'il vous plaît."  
2. Type what you heard (no replay)  
3. Submit  
4. AI compares:  
 ✅ Correct: 100%  
 ⚠️ Minor typo: "Je voudrais un cafe" (95%, missing accent)  
 ❌ Significant error: "Je voodray un café" (60%)  
5. Replay + see correct text

**Difficulty Levels:**

* **L1:** Single sentences (8-12 words)
* **L2:** Short paragraphs (50 words)
* **L3:** News clips (200 words, normal speed)

## Design System: Quiet Immersion

### Philosophy

**“Learning should feel like breathing — effortless, private, and deeply satisfying.”**

**Goals:**

1. **Zero eye strain** → Dark mode default, low blue light
2. **Zero distraction** → Single-focus layouts, no clutter
3. **Zero friction** → Keyboard shortcuts, instant load times
4. **Maximum immersion** → Floating cards, spatial depth

### Color Palette

| Role | Color | Hex | Usage |
| --- | --- | --- | --- |
| **Background** | Void Black | #0a0a0a | Main canvas |
| **Card Base** | Deep Indigo Glass | rgba(30, 27, 75, 0.6) | Content containers |
| **Text (Primary)** | Frost White | #f8fafc | Body text |
| **Text (Secondary)** | Slate Gray | #94a3b8 | Labels, metadata |
| **Accent (Success)** | Soft Cyan | #67e8f9 | Correct answers, progress |
| **Accent (Warm)** | Muted Amber | #fbbf24 | XP, achievements |
| **Accent (Error)** | Soft Coral | #fb7185 | Wrong answers (not harsh red) |
| **Borders** | Dark Slate | #1e293b | Subtle dividers |

**Glassmorphism Effect:**

.card {  
 background: rgba(30, 27, 75, 0.6);  
 backdrop-filter: blur(12px);  
 border: 1px solid rgba(255, 255, 255, 0.1);  
 box-shadow: 0 8px 32px rgba(0, 0, 0, 0.4);  
}

### Typography

**Font Stack:**

body {  
 font-family: "Inter", -apple-system, BlinkMacSystemFont, "Segoe UI", sans-serif;  
 font-size: 16px;  
 line-height: 1.6;  
 font-weight: 400;  
}  
  
h1,  
h2,  
h3 {  
 font-family: "Clash Display", "SF Pro Rounded", sans-serif;  
 font-weight: 600;  
}

**Scale:**

* **Body:** 16px (mobile), 18px (desktop)
* **Small:** 14px (labels, metadata)
* **H3:** 20px (card titles)
* **H2:** 28px (page headers)
* **H1:** 36px (hero text)

**Multilingual Support:**

* **Arabic:** Line-height 1.8 (taller for diacritics)
* **CJK:** Use Noto Sans CJK (Chinese/Japanese/Korean)
* **RTL languages:** Auto-flip layout with dir="rtl"

### Layout System

**Grid:**

.container {  
 max-width: 480px; /\* Mobile-first \*/  
 margin: 0 auto;  
 padding: 1.5rem;  
}  
  
@media (min-width: 768px) {  
 .container {  
 max-width: 720px; /\* Tablet \*/  
 }  
}  
  
@media (min-width: 1200px) {  
 .container {  
 max-width: 1080px; /\* Desktop: Side-by-side modules \*/  
 }  
}

**Card Anatomy:**

.card {  
 border-radius: 20px;  
 padding: 1.5rem;  
 margin-bottom: 1.5rem;  
 transition: transform 0.2s ease;  
}  
  
.card:hover {  
 transform: translateY(-4px);  
 box-shadow: 0 12px 48px rgba(0, 0, 0, 0.6);  
}

**Spacing Scale:**

0.25rem → 4px (tight elements)  
0.5rem → 8px (form fields)  
1rem → 16px (related items)  
1.5rem → 24px (cards, sections)  
3rem → 48px (major sections)

### UI Components

#### 1. Flashcard (Vocabulary Module)

<div className="flashcard">  
 <div className="flashcard-front">  
 <h2>La manzana</h2>  
 <button className="audio-btn">🔊</button>  
 </div>  
  
 <div className="flashcard-back">  
 <p className="definition">The apple</p>  
 <p className="context">"Comí una manzana ayer."</p>  
  
 <div className="rating-buttons">  
 <button className="rating rating-again">Again</button>  
 <button className="rating rating-hard">Hard</button>  
 <button className="rating rating-good">Good</button>  
 <button className="rating rating-easy">Easy</button>  
 </div>  
 </div>  
</div>

**Visual:**

┌─────────────────────────────────┐  
│ │  
│ La manzana 🔊 │  
│ │  
└─────────────────────────────────┘  
 ⬇️ (flip animation)  
┌─────────────────────────────────┐  
│ The apple │  
│ "Comí una manzana ayer." │  
│ │  
│ [Again] [Hard] [Good] [Easy] │  
└─────────────────────────────────┘

#### 2. Progress Vine (Persistent UI Element)

**Concept:** A growing vine (SVG) on the left sidebar that visualizes long-term progress.

<svg id="progress-vine" width="60" height="100%">  
 <path  
 d="M30,0 Q40,100 30,200 Q20,300 30,400"  
 stroke="#67e8f9"  
 stroke-width="3"  
 fill="none"  
 stroke-dasharray="1000"  
 stroke-dashoffset="800"  
 <!-- Animates as user progresses -->  
 />  
  
 <!-- Leaves = milestones (10 cards, 50 cards, 100 cards) -->  
 <circle cx="30" cy="100" r="8" fill="#10b981" />  
 <circle cx="30" cy="200" r="8" fill="#fbbf24" />  
</svg>

**Growth Logic:**

* Vine height = Total XP / 100
* Leaf appears every 1000 XP
* Vine color shifts: Cyan (beginner) → Emerald (intermediate) → Gold (advanced)

#### 3. AI Avatar (Bottom-Right Corner)

**Purpose:** Subtle companion that reacts to user actions.

**Appearance:**

* 2.5D stylized face (SVG-based)
* Minimal features (eyes, smile)
* Soft glow around avatar

**Reactions:** | Event | Avatar Response | |——-|—————-| | Correct answer | Eyes close briefly (contentment), smile widens | | Wrong answer | Head tilts (empathy), no judgment | | Streak milestone | Subtle sparkle animation | | Session complete | Gentle nod, eyes glow cyan |

**Implementation:**

<div className="ai-avatar">  
 <svg viewBox="0 0 100 100">  
 <!-- Eyes (animate based on state) -->  
 <circle cx="35" cy="40" r="8" fill="#67e8f9" className="eye-left" />  
 <circle cx="65" cy="40" r="8" fill="#67e8f9" className="eye-right" />  
  
 <!-- Smile (path morphs) -->  
 <path  
 d="M30,60 Q50,70 70,60"  
 stroke="#f8fafc"  
 stroke-width="3"  
 fill="none"  
 className="smile"  
 />  
 </svg>  
</div>

#### 4. Daily Goal Widget

┌─────────────────────────────────────┐  
│ Today's Goal: 20 cards │  
│ │  
│ ████████████░░░░░░ 60% (12/20) │  
│ │  
│ 🔥 3-day streak │  
└─────────────────────────────────────┘

### Micro-Interactions

**1. Card Flip Animation:**

.flashcard {  
 transition: transform 0.6s;  
 transform-style: preserve-3d;  
}  
  
.flashcard.flipped {  
 transform: rotateY(180deg);  
}

**2. Correct Answer Pulse:**

@keyframes success-pulse {  
 0% {  
 transform: scale(1);  
 }  
 50% {  
 transform: scale(1.05);  
 box-shadow: 0 0 20px #67e8f9;  
 }  
 100% {  
 transform: scale(1);  
 }  
}  
  
.card.correct {  
 animation: success-pulse 0.3s ease;  
}

**3. Haptic Feedback (Mobile):**

// On correct answer  
navigator.vibrate(30); // Short tick  
  
// On wrong answer  
navigator.vibrate([50, 100, 50]); // Double tap  
  
// On milestone  
navigator.vibrate([100, 50, 100, 50, 200]); // Celebration pattern

**4. Spatial Audio:**

const audio = {  
 correct: new Audio("/sounds/chime-soft.mp3"), // 440Hz, 0.2s  
 incorrect: new Audio("/sounds/thud-soft.mp3"), // 220Hz, 0.3s  
 milestone: new Audio("/sounds/bell-warm.mp3"), // Harmonious chord  
};  
  
audio.correct.volume = 0.3; // Subtle, not jarring

### Accessibility

**WCAG AAA Compliance:**

* **Color contrast:** 4.5:1 minimum (body text)
* **Focus indicators:** 3px cyan outline on interactive elements
* **Keyboard navigation:** Tab through all actions
* **Screen reader:** ARIA labels on all icons/buttons

**Reduce Motion:**

@media (prefers-reduced-motion: reduce) {  
 \* {  
 animation-duration: 0.01ms !important;  
 transition-duration: 0.01ms !important;  
 }  
}

**Font Scaling:**

* Respect user’s OS font size setting
* Allow in-app text size adjustment (Settings)

### Performance Targets

| Metric | Target | Why |
| --- | --- | --- |
| **First Contentful Paint** | < 1.2s | User sees content fast |
| **Largest Contentful Paint** | < 2.5s | Main content loaded |
| **Time to Interactive** | < 3.0s | User can act quickly |
| **JavaScript Bundle** | < 150KB | Fast on 3G networks |
| **Image Optimization** | WebP, lazy load | Reduce bandwidth |

**Optimization Strategies:**

* Code splitting per module (load Vocabulary separate from Speaking)
* Service worker for offline caching
* CDN for audio/image assets
* Database query caching (Redis)

## Gamification Strategy

### Philosophy

**“Gamification should reward learning, not just logging in.”**

**What We Avoid:**

* ❌ Daily login bonuses (encourages empty sessions)
* ❌ Pay-to-skip mechanics (undermines learning)
* ❌ Public shaming (leaderboards showing “worst” performers)

**What We Embrace:**

* ✅ Mastery-based XP (more XP for difficult cards)
* ✅ Intrinsic rewards (seeing your vine grow)
* ✅ Collaborative goals (study with friends)

### XP System

**XP Sources:**

| Action | XP Earned | Rationale |
| --- | --- | --- |
| Review card (Good) | 10 XP | Base reward |
| Review card (Hard) | 15 XP | More effort = more reward |
| Review card (Easy) | 5 XP | Too easy = less learning |
| Complete 20-card session | 50 XP | Consistency bonus |
| 7-day streak maintained | 200 XP | Building habit |
| Shadowing exercise (>80% score) | 30 XP | Output practice |
| Write 100-word composition | 40 XP | Deep engagement |
| Read article (5+ min) | 25 XP | Input practice |
| Create public deck (>10 subscribers) | 500 XP | Community contribution |

**XP Decay Prevention:**

* No XP loss for missing days (no punishment)
* Bonus XP for returning after 3+ day break (encouragement)

**Level System:**

Level 1: 0 - 500 XP (Seedling 🌱)  
Level 2: 500 - 1,500 XP (Sprout 🌿)  
Level 5: 5,000 - 10,000 XP (Tree 🌳)  
Level 10: 25,000 - 50,000 XP (Forest 🌲)

### Streaks

**Definition:** Consecutive days with at least 1 completed study session.

**Visualization:**

🔥 14-day streak  
  
Week 1: ✅✅✅✅✅✅✅  
Week 2: ✅✅✅✅✅✅✅  
Today: ← 2 hours left to maintain streak!

**Streak Freezes:**

* Earn 1 freeze per 7-day streak
* Use freeze to skip 1 day without breaking streak
* Max 2 freezes stored

**Milestones:**

* 7 days: +200 XP, “Week Warrior” badge
* 30 days: +1000 XP, “Monthly Maestro” badge
* 100 days: +5000 XP, “Centurion” badge, custom avatar frame

### Badges

**Categories:**

**1. Consistency Badges:**

* 🌟 **First Steps:** Complete 1st study session
* 🔥 **Week Warrior:** 7-day streak
* 📅 **Monthly Maestro:** 30-day streak
* 💎 **Century Club:** 100-day streak

**2. Mastery Badges:**

* 📚 **Vocab Novice:** Review 100 cards
* 📖 **Vocab Scholar:** Review 1,000 cards
* 🎓 **Vocab Master:** Review 10,000 cards
* 🗣️ **Pronunciation Pro:** 50 shadowing exercises (>80% score)
* ✍️ **Writer’s Guild:** Write 10,000 words total

**3. Social Badges:**

* 👥 **Deck Creator:** Publish 1st public deck
* ⭐ **Community Star:** Your deck reaches 100 subscribers
* 🤝 **Study Buddy:** Complete 10 co-study sessions with friends

**4. Special Badges:**

* 🌙 **Night Owl:** Study session at 2-4am
* 🌍 **Polyglot Path:** Study 3+ languages in platform
* 🏆 **Leaderboard Legend:** Top 10 in monthly XP

**Display:**

* Profile page: Showcase up to 6 favorite badges
* Badge cabinet: View all earned badges (grayed out = locked)

### Leaderboards

**Types:**

**1. Global Leaderboard (Monthly):**

Rank User XP This Month  
─────────────────────────────────  
 1 🥇 Sarah 15,240 XP  
 2 🥈 Miguel 14,890 XP  
 3 🥉 Yuki 13,750 XP  
...  
 42 You 8,120 XP

**2. Friends Leaderboard:**

Among your 12 friends:  
  
 1 Alex 3,450 XP  
 2 You 3,210 XP  
 3 Jamie 2,980 XP

**3. Language-Specific:**

Top Spanish Learners (This Week):  
  
 1 Carlos 2,100 XP  
 2 You 1,850 XP

**Privacy:**

* Opt-out option (hide from global leaderboard)
* Only show first name + last initial (e.g., “Sarah M.”)
* Friends leaderboard requires mutual follow

### Progress Visualization

**1. Module Progress Rings:**

Dashboard shows 5 rings (one per module):  
  
 Vocabulary Speaking Writing Reading Listening  
 88% 45% 23% 67% 34%

**2. Language Proficiency Estimate:**

Spanish Proficiency: B1 (Intermediate)  
  
📊 Breakdown:  
- Vocabulary: 1,842 words known (B1: 1,500-2,500)  
- Grammar: 78% accuracy on B1 structures  
- Listening: 85% comprehension on B1 audio  
- Speaking: Estimated B1 (based on AI conversations)  
  
🎯 Next milestone: B2 (need 658 more words)

**3. Time Investment Graph:**

Study Time This Month: 14h 32m  
  
 Hours  
 8 ┤ ╭───●  
 6 ┤ ╭─╯  
 4 ┤ ╭─╯  
 2 ┤─╯  
 0 └──────────────  
 Week 1 2 3 4

## AI & Personalization

### Learning Profile

**System tracks:**

* **Vocabulary size** (per language)
* **Grammar competency** (which structures mastered)
* **Preferred content** (topics: sports, tech, food, etc.)
* **Optimal study time** (morning person vs. night owl)
* **Review patterns** (FSRS difficulty parameters)

**Adaptive Content Recommendations:**

"Based on your progress, we recommend:  
- 📖 Reading: 'El Alchemista' (90% known vocabulary)  
- 🎧 Podcast: 'Coffee Break Spanish' Episode 42  
- 📝 Writing: Practice present subjunctive (your weak area)

### AI Features

**1. Smart Review Ordering:**

* Mix old + new cards (not all hard cards at end)
* Interleave topics (vocabulary + grammar, not blocks)
* Priority to cards near “forgetting threshold”

**2. Difficulty Auto-Adjustment:**

* Text too easy (>98% comprehension) → suggest harder texts
* Speaking exercises too fast → auto-slow to 0.8x

**3. Personalized Study Plans:**

Your goal: B2 Spanish in 6 months  
  
Weekly plan:  
- Monday/Wed/Fri: 30 min vocabulary review  
- Tuesday/Thursday: 20 min speaking practice  
- Weekends: 40 min reading + listening

**4. AI Tutor (Future Feature):**

* Ask questions: “When do I use ‘ser’ vs. ‘estar’?”
* Get explanations in your native language
* Request custom exercises for weak areas

## Roadmap

### Phase 1: MVP (Q4 2025) ✅ In Progress

**Core Features:**

* ✅ User authentication (register, login, JWT)
* ✅ Vocabulary module (FSRS flashcards)
* ✅ Deck management (create, fork, commit)
* ✅ Basic gamification (XP, streaks, levels)
* 🚧 Speaking module (shadowing only)
* 🚧 Reading module (graded texts library)

**Tech Debt:**

* Finalize gRPC communication (Python ↔ ASP.NET)
* Docker compose for local dev environment
* CI/CD pipeline (GitHub Actions)

### Phase 2: Enhanced Modules (Q1 2026)

**New Features:**

* Listening module (podcast library, dictation)
* Writing module (sentence builder, composition studio)
* Speaking: Conversation simulator (AI partner)
* Social features (friends, leaderboards, profile pages)

**Design:**

* Full “Quiet Immersion” UI implementation
* AI avatar reactions
* Progress vine animation
* Mobile PWA (offline mode)

### Phase 3: Community & Content (Q2 2026)

**Marketplace:**

* Public deck discovery (search, filters, ratings)
* Deck versioning UI (commit history, diffs, merge)
* Creator analytics (subscriber count, review stats)

**User-Generated Content:**

* Import tool (Anki, CSV, Quizlet)
* Bulk sentence mining (paste article → auto-generate cards)
* Image/audio uploads for custom cards

**Monetization (Optional):**

* Premium tier ($5/mo):
  + Unlimited decks (free: 5 decks max)
  + Advanced stats (granular progress graphs)
  + Priority AI features (faster pronunciation scoring)
* Keep core learning free forever

### Phase 4: Advanced AI (Q3 2026)

**AI Tutor:**

* Chat with AI in target language (Socratic method)
* Grammar explanations on-demand
* Personalized study plan generator

**Content Intelligence:**

* Auto-generate decks from YouTube videos (extract subtitles)
* Podcast transcription + vocabulary extraction
* Adaptive text difficulty (AI rewrites texts to your level)

**Speech Tech:**

* Real-time pronunciation feedback during shadowing
* Accent training (British vs. American English)
* Emotion detection (are you confident or hesitant?)

### Phase 5: Mobile Apps (Q4 2026)

**Native iOS/Android:**

* Offline study mode (sync when online)
* Widgets (daily goal progress, streak counter)
* Push notifications (gentle reminders, not spam)
* Voice-first interface (study while commuting)

**Wearables:**

* Apple Watch: Quick reviews during breaks
* Smart speakers: “Alexa, quiz me on Spanish vocab”

## Success Metrics

### Learning Outcomes (Primary)

| Metric | Target | Measurement |
| --- | --- | --- |
| **Vocabulary retention** | 85%+ after 30 days | FSRS data (cards due vs. cards remembered) |
| **User-reported fluency gain** | +1 CEFR level per 6 months | Self-assessment survey |
| **Active study rate** | 60% of users study 3+ days/week | Session logs |

### Engagement (Secondary)

| Metric | Target | Measurement |
| --- | --- | --- |
| **7-day retention** | 50% | Users who return after 1 week |
| **30-day retention** | 30% | Users still active after 1 month |
| **Average session length** | 15-20 minutes | Time from start to end of study session |
| **Daily active users (DAU)** | Grow 10% month-over-month | Login events per day |

### Community (Tertiary)

| Metric | Target | Measurement |
| --- | --- | --- |
| **Public decks created** | 1,000+ in Year 1 | Deck publish events |
| **Deck forks** | 20% of public decks forked | Fork action count |
| **Average deck rating** | 4.2+ / 5 stars | User ratings (after studying deck) |

### Technical (Infrastructure)

| Metric | Target | Current |
| --- | --- | --- |
| **API response time** | < 200ms (p95) | Monitor with Datadog/Grafana |
| **Uptime** | 99.5% | Exclude planned maintenance |
| **Mobile load time** | < 2s (LCP) | Lighthouse CI scores |

## Conclusion

**LineByLine** is not just another language app—it’s a **scientifically-grounded learning ecosystem** that respects users’ time, intelligence, and goals.

### What Makes Us Different

1. **Science over engagement hacks** — FSRS, comprehensible input, output practice
2. **Modular mastery** — Focus on one skill deeply, not shallow “gamified” lessons
3. **Community-driven content** — Git-like deck versioning, public marketplace
4. **Privacy-first design** — No data selling, no manipulative streaks
5. **Beautiful, calming UI** — “Quiet Immersion” design for distraction-free learning

### Our Promise

**“We succeed when you become fluent, not when you open the app 100 days in a row.”**

## Appendices

### A. Glossary of Language Learning Terms

| Term | Definition |
| --- | --- |
| **CEFR** | Common European Framework of Reference (A1 → C2) |
| **Comprehensible Input** | Language slightly above current level (i+1) |
| **Comprehensible Output** | Producing language to test hypotheses |
| **FSRS** | Free Spaced Repetition Scheduler (ML-based algorithm) |
| **Shadowing** | Repeating audio simultaneously to improve pronunciation |
| **SRS** | Spaced Repetition System (generic term) |
| **i+1** | Input at “current level + 1 difficulty” |

### B. Recommended Reading

* **“The Natural Approach”** by Stephen Krashen
* **“How Languages Are Learned”** by Patsy Lightbown
* **“Make It Stick”** by Peter Brown (learning science)
* **“Fluent Forever”** by Gabriel Wyner (practical methods)

### C. Technical Documentation Links

* **FSRS Algorithm:** [open-spaced-repetition/fsrs4anki](https://github.com/open-spaced-repetition/fsrs4anki)
* **gRPC Protocol:** proto/fsrs\_service.proto (internal repo)
* **API Endpoints:** docs/api-reference.md (internal repo)
* **Design System:** docs/design-system.md (internal repo)

**Last Updated:** October 31, 2025  
**Document Owner:** LineByLine Product Team  
**Contact:** [project repository]

*This is a living document. As we learn from users and iterate on features, we’ll update this vision accordingly.*