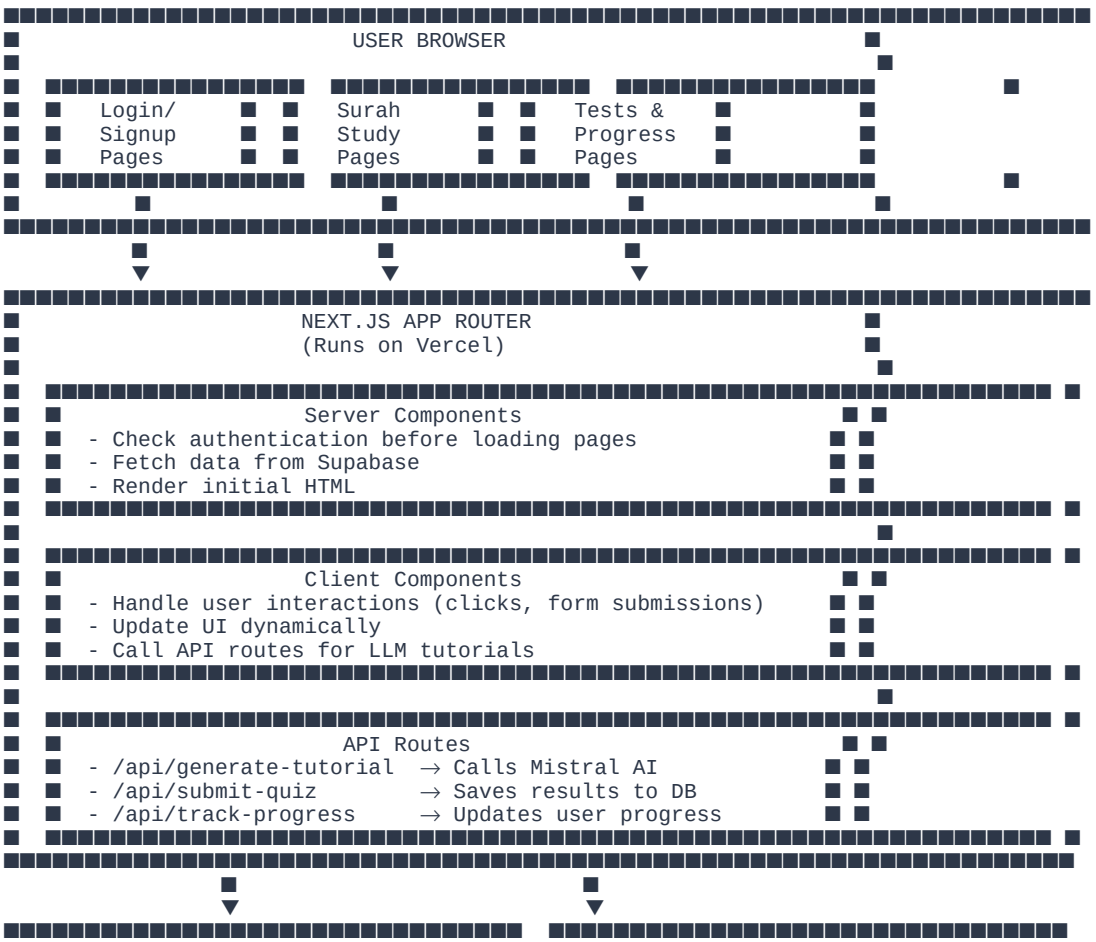
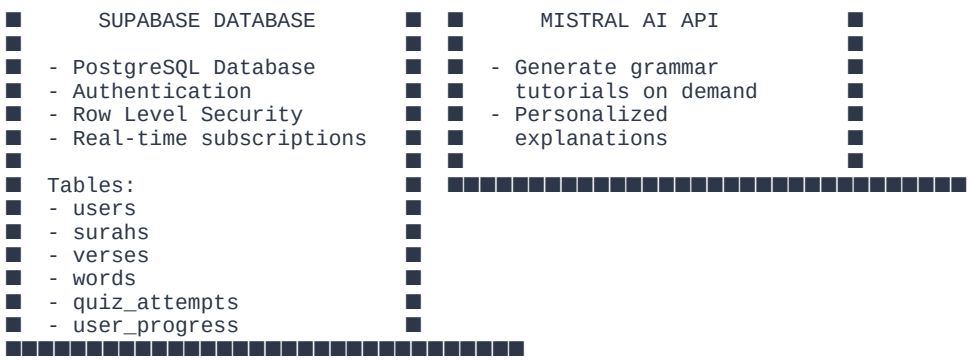
Quranic Arabic Learning Platform

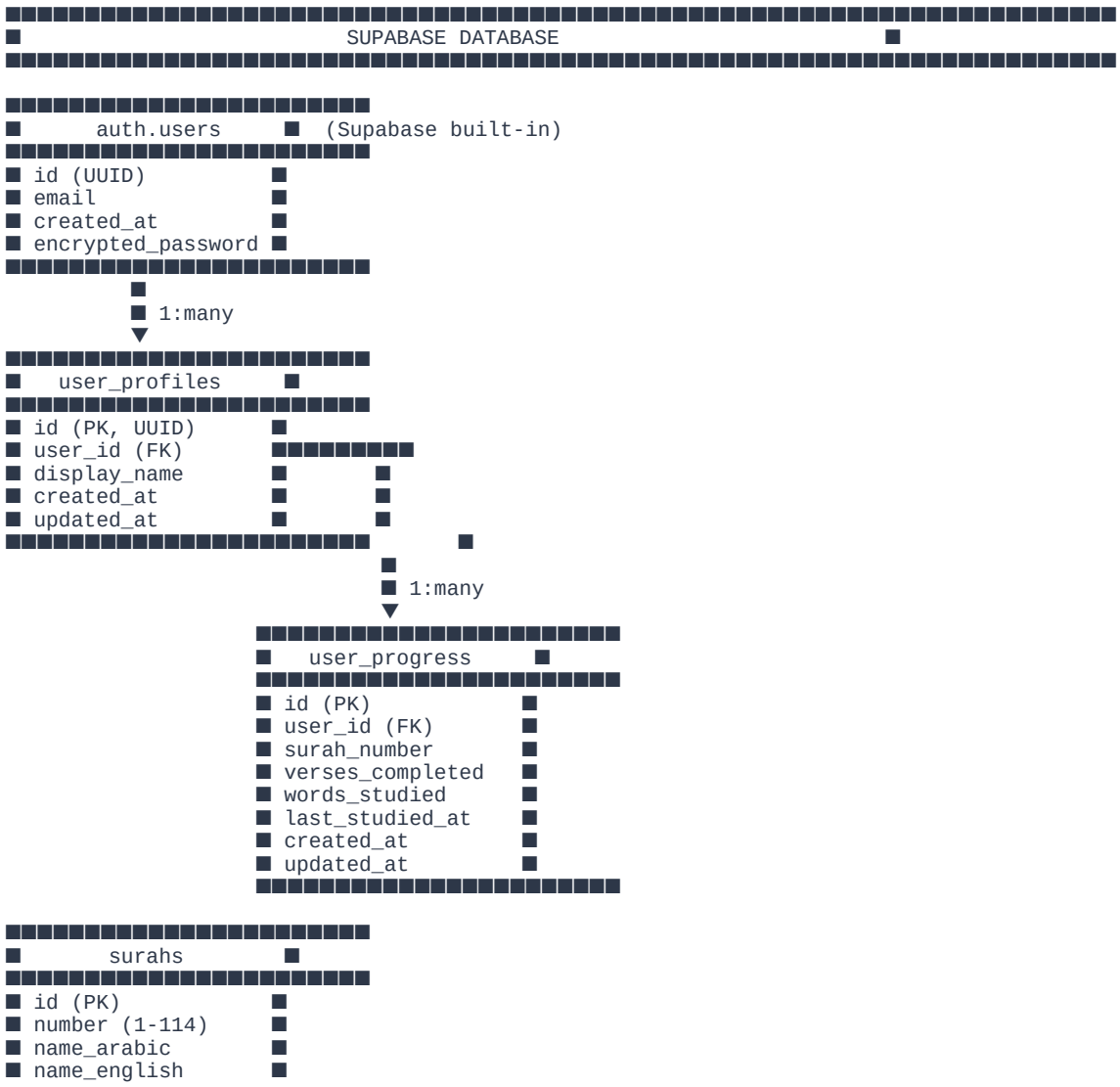
Architecture Documentation

1. High-Level System Architecture





2. Database Schema




name_translation
revelation_place
(Meccan/Medinan)
total_verses
created_at
1:many
verses
id (PK)
surah_id (FK)
verse_number
text_arabic
text_simple
(without diacritics)
translation_en
created_at
1:many
words
id (PK)
verse_id (FK)
position (1, 2, 3...)
text_arabic
text_simple
transliteration
translation_en
root
lemma
grammar (JSONB)
← Contains all grammatical info
{
"pos": "noun",
"case": "nominative",
"number": "singular",
"gender": "masculine",
"form": "I",
"tense": "perfect",
...
}
audio_url
created_at
quiz_questions
id (PK)
surah_number
question_type
(vocab, translation, grammar)
question_text
correct_answer
options (JSONB)
word_id (FK)
(optional - if question is about a specific word)
created_at
many:many
quiz_attempts
id (PK)
user_id (FK)
question_id (FK)
user_answer
is_correct
attempted_at
time_taken_ms

```
■■■■■■■■■■■■■■■■■■■■■■■■
■  chapter_progress    ■  (For grammar learning pathway)
■■■■■■■■■■■■■■■■■■■■■■■■
■ id (PK)              ■
■ user_id (FK)         ■
■ chapter_id           ■  (ch1, ch2, ch3... from haywood-chapters.js)
■ status               ■  (not_started, in_progress, completed)
■ quiz_score           ■
■ completed_at         ■
■ created_at           ■
■ updated_at           ■
■■■■■■■■■■■■■■■■■■■■■■■■
```

Key Schema Design Decisions:

1. **JSONB for Grammar Data**: The `words.grammar` field uses JSONB to store flexible grammatical information. This allows for different word types to have different grammatical properties without requiring separate tables.

2. **Separate User Progress Tables**: `user_progress` tracks Quranic content progress, while `chapter_progress` tracks grammar chapter completion.

3. **Quiz Architecture**: Questions are pre-generated and stored. User attempts are tracked separately, allowing for analytics and adaptive learning.

3. Component Architecture

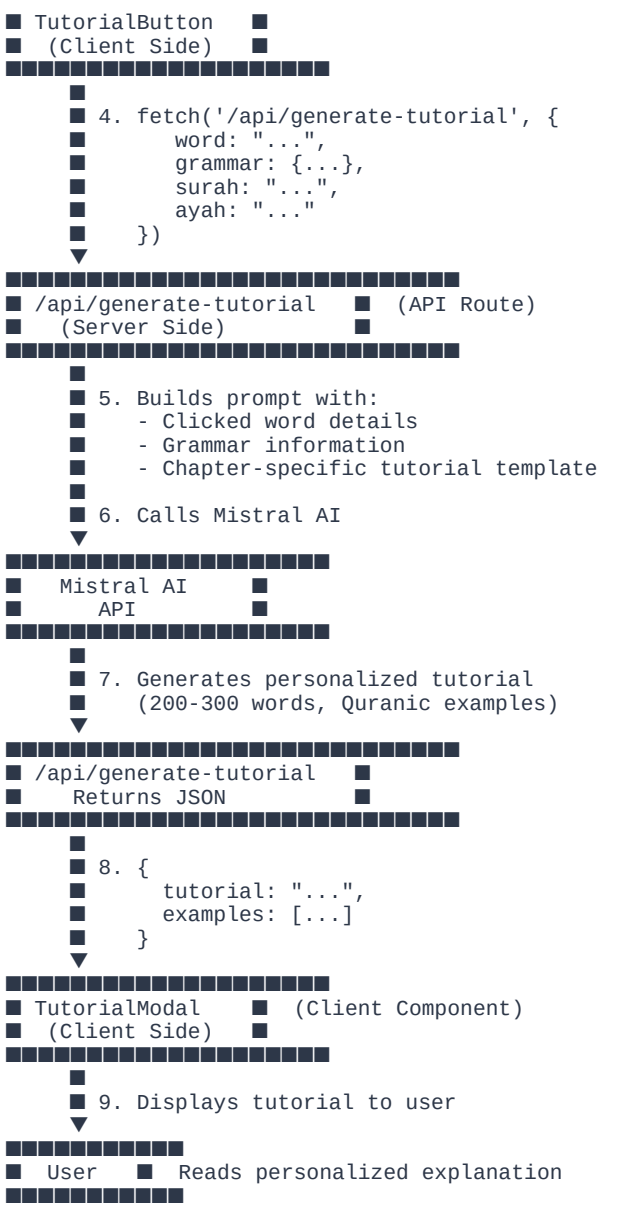
```
app/
■■■ layout.tsx                    ← Root layout (navigation, auth provider)
■
■■■ page.tsx                      ← Homepage / Landing
■
■■■ auth/
■   ■■■ login/
■   ■   ■■■ page.tsx              ← Login page (client component)
■   ■■■ signup/
■       ■■■ page.tsx              ← Signup page (client component)
■
■■■ dashboard/
■   ■■■ page.tsx                  ← User dashboard (server component)
■   ■■■ layout.tsx                ← Dashboard layout (nav sidebar)
■
■■■ surah/
■   ■■■ page.tsx                  ← Surah list (server component)
■   ■■■ [number]/
■       ■■■ page.tsx              ← Surah reader (server component)
■       ■■■ components/
■           ■■■ VerseDisplay.tsx  ← Shows verses (client)
■           ■■■ WordCard.tsx      ← Click to see word details (client)
■           ■■■ GrammarPanel.tsx  ← Shows grammar info (client)
■
■■■ grammar/
■   ■■■ page.tsx                  ← Grammar chapters list
■   ■■■ [chapterId]/
■       ■■■ page.tsx              ← Chapter content
■       ■■■ components/
■           ■■■ TutorialButton.tsx   ← Trigger AI tutorial
■           ■■■ TutorialModal.tsx    ← Display tutorial
■
■■■ quiz/
■   ■■■ page.tsx                  ← Quiz type selection
```

```
■   ■■■ [surahNumber]/
■       ■■■ vocab/
■       ■   ■■■ page.tsx          ← Vocabulary quiz
■       ■■■ translation/
■       ■   ■■■ page.tsx          ← Translation quiz
■       ■■■ components/
■           ■■■ QuizCard.tsx      ← Question display (client)
■           ■■■ QuizResults.tsx   ← Results & recommendations
■
■■■ progress/
■   ■■■ page.tsx                  ← User progress dashboard
■
■■■ api/
■   ■■■ generate-tutorial/
■   ■   ■■■ route.ts              ← POST: Generate AI tutorial
■   ■■■ submit-quiz/
■   ■   ■■■ route.ts              ← POST: Save quiz attempt
■   ■■■ track-progress/
■   ■   ■■■ route.ts              ← POST: Update user progress
■   ■■■ audio/
■       ■■■ [wordId]/
■           ■■■ route.ts          ← GET: Stream audio for word
■
■■■ components/
■   ■■■ AuthButton.tsx            ← Login/Logout button
■   ■■■ Navigation.tsx            ← Main navigation
■   ■■■ AudioPlayer.tsx           ← Audio playback component
■
■■■ lib/
    ■■■ supabase/
    ■   ■■■ server.ts             ← Server-side Supabase client
    ■   ■■■ client.ts             ← Client-side Supabase client
    ■■■ mistral.ts                ← Mistral AI integration
    ■■■ utils/
        ■■■ grammarHelpers.ts     ← Parse grammar data
        ■■■ quizHelpers.ts        ← Quiz generation logic
```

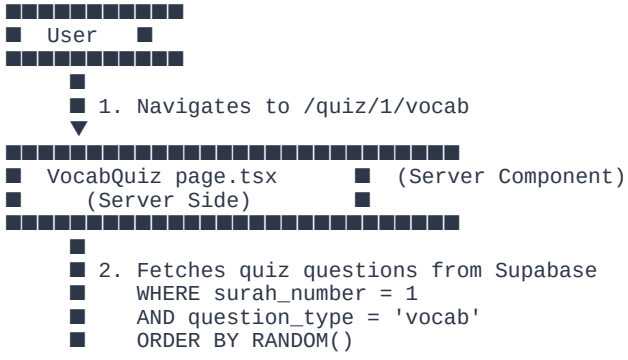
4. Data Flow Diagrams

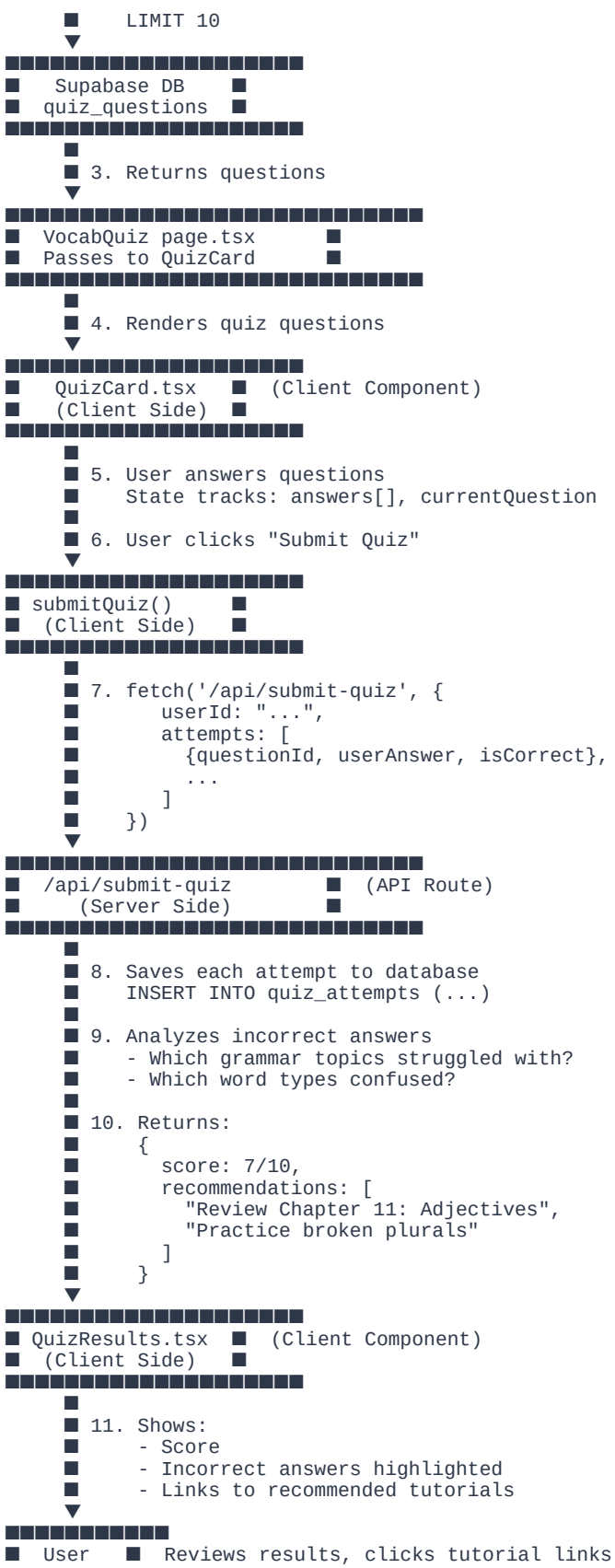
A. User Studies a Word

```
■■■■■■■■■■■■
■  User    ■
■■■■■■■■■■■■
     ■
     ■ 1. Clicks on Arabic word
     ▼
■■■■■■■■■■■■■■■■■■■■■■
■  WordCard.tsx     ■  (Client Component)
■  (Client Side)   ■
■■■■■■■■■■■■■■■■■■■■■■
     ■
     ■ 2. Displays:
     ■    - Arabic text
     ■    - Transliteration
     ■    - Translation
     ■    - [Tutorial] button
     ■    - [Audio] button
     ■
     ■ 3. User clicks [Tutorial]
     ▼
■■■■■■■■■■■■■■■■■■■■■■
```



B. User Takes a Quiz




LIMIT 10
Supabase DB
quiz_questions
3. Returns questions
VocabQuiz page.tsx
Passes to QuizCard
4. Renders quiz questions
QuizCard.tsx
(Client Component)
(Client Side)
5. User answers questions
State tracks: answers[], currentQuestion
6. User clicks "Submit Quiz"
submitQuiz()
(Client Side)
7. fetch('/api/submit-quiz', {
userId: "...",
attempts: [
{questionId, userAnswer, isCorrect},
...
]
})
/api/submit-quiz
(API Route)
(Server Side)
8. Saves each attempt to database
INSERT INTO quiz_attempts (...)
9. Analyzes incorrect answers
- Which grammar topics struggled with?
- Which word types confused?
10. Returns:
{
score: 7/10,
recommendations: [
"Review Chapter 11: Adjectives",
"Practice broken plurals"
]
}
QuizResults.tsx
(Client Component)
(Client Side)
11. Shows:
- Score
- Incorrect answers highlighted
- Links to recommended tutorials
User
Reviews results, clicks tutorial links

5. Authentication Flow

Protected Route Pattern

Every protected page follows this pattern:

```
// app/dashboard/page.tsx (Server Component)
import { createClient } from '@/lib/supabase/server';
import { redirect } from 'next/navigation';

export default async function DashboardPage() {
  const supabase = await createClient();
  const { data: { user } } = await supabase.auth.getUser();

  if (!user) {
    redirect('/auth/login');
  }

  // User is authenticated, render page
  return <div>Welcome, {user.email}!</div>;
}
```

6. Tutorial Generation Flow (Detailed)


3. Loads chapter template from haywood-chapters.js
→ Get chapter.tutorialPrompt for Chapter 4
4. Replaces placeholders in prompt:
{word} → "■■■■■■■■■■"
{translation} → "the praise"
{surah} → "1"
{ayah} → "1"
{grammarDetails} → "nominative case, definite"
5. Final prompt sent to Mistral:
"You are a friendly Arabic teacher explaining nominative case. The student clicked on '■■■■■■■■■■' (al-■amdu) from Surah 1, Ayah 1. This word is in nominative case because it's the subject of the sentence. Explain nominative case using this example, then show 2-3 more Quranic examples..."
Sends to AI
Mistral AI API
Generates tutorial following prompt structure:
1. Simple definition
2. Explanation using clicked word
3. 2-3 additional Quranic examples
4. Memory tip
Returns: ~250 word conversational explanation
Returns generated text
/api/generate-tutorial/route.ts
Returns JSON:
{
tutorial: "The nominative case (■■■■■■) shows...",
status: "success"
}
Response sent back
TutorialModal.tsx (Client Component)
Displays:
Tutorial: Nominative Case
The nominative case (■■■■■■) shows that a word is the subject - the doer of the action or the topic being discussed. Let's look at the word you clicked: ■■■■■■■■■■ (al-■amdu) "the praise."
Notice the ■ (damma) at the end? That's the nominative case marker...
[More Quranic examples with references]
Tip: Think "subject = up" - nominative uses damma which sounds like "u" (up!)

7. Testing System Architecture

```
┌──────────────────────────────────────────────────────────────────────────┐
│                        Quiz Generation System                            │
└──────────────────────────────────────────────────────────────────────────┘

QUIZ TYPES:

1. Vocabulary Quiz (Arabic → English)
   ┌──────────────────┐
   │   ████████   │
   └──────────────────┘
   ┌──────────────────┐
   │ A) house    │
   │ B) book     │  ✓ Correct
   │ C) door     │
   │ D) pen      │
   └──────────────────┘

2. Translation Quiz (English → Arabic)
   ┌──────────────────────┐
   │ "the book"      │
   └──────────────────────┘
   ┌──────────────────────┐
   │ A) ██████████     │
   │ B) ███████████    │  ✓ Correct
   │ C) ████████     │
   │ D) ██████████    │
   └──────────────────────┘

3. Grammar Quiz
   ┌──────────────────────────────────────────┐
   │ What case is ██████████ in?          │
   │ (from: ██████████ ████████)            │
   └──────────────────────────────────────────┘
   ┌──────────────────────────────────────────┐
   │ A) Nominative  ✓ Correct             │
   │ B) Genitive                          │
   │ C) Accusative                        │
   └──────────────────────────────────────────┘

ADAPTIVE QUIZ ALGORITHM:

┌──────────────────────────────────────────────────────────────────────────┐
│  User Performance Tracking                                               │
└──────────────────────────────────────────────────────────────────────────┘

After each quiz:
1. Calculate score per grammar topic
2. Identify weak areas
3. Weight future questions toward weak areas

Example:
User scores:
- Nouns (general): 90%
- Nominative case: 60%  ← Weak area
- Verbs: 85%
- Broken plurals: 40%   ← Weak area

Next quiz generation:
- 40% questions on weak areas (nominative + broken plurals)
- 40% questions on medium areas
- 20% questions on strong areas (review/confidence)

┌──────────────────────────────────────────────────────────────────────────┐
│  Recommendation Engine                                                   │
└──────────────────────────────────────────────────────────────────────────┘

Based on quiz results:
1. Map incorrect answers to grammar chapters
2. Recommend specific tutorials
3. Track progress over time

Example output after quiz:
┌──────────────────────────────────────────────────────────────┐
```

```
Your Score: 7/10

Recommended Study:
• Chapter 4: Declension of Nouns (3 errors)
• Chapter 6: Broken Plural (2 errors)

[Start Tutorial] [Retake Quiz] [View Progress]
```

Key Architectural Principles

1. **Separation of Concerns**

• **Server Components**: Fetch data, check auth, render initial HTML
• **Client Components**: Handle interactions, manage state
• **API Routes**: Business logic, external API calls

2. **Data Security**

• Row Level Security (RLS) in Supabase ensures users only see their own progress
• Authentication checked on both client and server
• Sensitive operations (quiz submission, progress tracking) go through API routes

3. **Performance Optimization**

• Server-side rendering for initial page load (fast)
• Client-side interactivity for smooth UX
• Tutorial caching (optional): Store generated tutorials to avoid repeated AI calls

4. **Scalability**

• Stateless API routes (can be scaled horizontally)
• Database-backed state (progress, quiz results)
• LLM rate limiting to control costs

5. **Mobile-First Design**

• Responsive components
• Touch-friendly interactions
• Optimized for on-the-go learning

Technology Stack Summary

Layer	Technology	Purpose
Frontend Framework	Next.js 14+ (App Router)	React framework with SSR
Styling	Tailwind CSS	Utility-first CSS
Database	Supabase (PostgreSQL)	Data storage + auth
Authentication	Supabase Auth	User management
LLM	Mistral AI	Tutorial generation
Hosting	Vercel	Deployment platform
State Management	React Server Components + Client State	Minimize client-side state
Arabic Text	Amiri Font (24px)	Optimal Quranic text rendering
Audio	Browser Audio API	Word pronunciation

Next Steps for Implementation

1. **Phase 1**: Authentication + Database Setup
- Set up Supabase project
- Create auth pages
- Test login/logout flow

2. **Phase 2**: Core Content Display
- Import Al-Fatiha data
- Create surah reader page
- Implement word click → details display

3. **Phase 3**: Grammar Tutorials
- Set up Mistral AI integration
- Create tutorial generation API
- Test with various grammar points

4. **Phase 4**: Quiz System
- Generate quiz questions
- Build quiz interface
- Implement scoring and recommendations

5. **Phase 5**: Progress Tracking
- User dashboard
- Progress visualization
- Adaptive learning algorithm

Each phase builds on the previous, allowing for incremental testing and learning.