AviLingo — Complete Product Bible

Three Perspectives, One Product

This document covers everything you need to know about building AviLingo from:

1. **Product Owner/Manager** — What to build and why
2. **ICAO/Aviation English SME** — Domain expertise and content requirements
3. **Senior Backend Engineer** — How to build it right

PART 1: PRODUCT OWNER/MANAGER PERSPECTIVE

1.1 Product Vision

One-Liner

> "The fastest path from broken aviation English to ICAO Level 4+ certification"

Vision Statement

AviLingo helps non-native English speaking pilots pass their mandatory ICAO English proficiency tests through daily bite-sized practice combining AI-powered speech feedback, real ATC audio comprehension, and spaced repetition vocabulary learning.

Problem Statement

Problem	Evidence	Impact
ICAO English is mandatory but prep is fragmented	Pilots use 5+ resources (LiveATC, YouTube, PDFs, tutors)	Wasted time, inconsistent progress
Existing solutions are expensive	Courses cost $300-500, tutors $50-100/hr	Barrier for CIS/Asian pilots
No mobile-first solution exists	Competitors are web-based or desktop	Pilots can't practice during commute/layovers
Speaking practice is hardest to get	Need human partner or expensive tutor	#1 failure point in ICAO tests
Pilots don't know if they're ready	No realistic mock tests	Test anxiety, surprise failures

Success Metrics (North Star)

Metric	Definition	Target (Year 1)
Primary: Pass Rate	% of users who report passing ICAO test	>85%
Secondary: DAU/MAU	Daily engagement ratio	>40%
Revenue: MRR	Monthly recurring revenue	$10,000
Growth: Signups	New waitlist/registrations per month	1,000

1.2 User Personas

Persona 1: Rustam (Primary Target)

```
Demographics:
- 26 years old, First Officer
- Uzbekistan Airways, based in Tashkent
- Intermediate English (Level 3-4)
- Earns ~$1,500/month

Context:
- Passed Level 4 three years ago, revalidation due in 4 months
- Last time barely passed, nervous about retesting
- Studies during commute (metro) and layovers
- Wife and baby at home, limited time

Goals:
- Pass Level 4 comfortably, ideally get Level 5
- Practice speaking without embarrassment
- Fit study into 15-20 min daily

Pain Points:
- Can't afford $500 courses
- Embarrassed to practice speaking with colleagues
- Doesn't know if he's improving
- Forgets vocabulary between study sessions

Willingness to Pay: $10-20/month, up to $100 one-time
```

Persona 2: Dilnoza (Secondary Target)

```
Demographics:
- 22 years old, Cadet
- Uzbekistan Airways Flight Academy
- Pre-intermediate English (Level 2-3)
- No income (family supports)

Context:
- Needs Level 4 to graduate and get hired
- Has 8 months until test
- Full-time student, more study time available
- Competitive with classmates

Goals:
- Pass Level 4 on first attempt
- Build real aviation vocabulary
- Understand fast ATC communications

Pain Points:
- Never heard real ATC before
- Classroom English ≠ cockpit English
- No way to practice pronunciation alone
- Doesn't know what test is really like

Willingness to Pay: $5-10/month (family pays)
```

Persona 3: Viktor (Tertiary Target)

```
Demographics:
- 38 years old, Captain
- Air Astana, based in Almaty
- Upper-intermediate (Level 5)
- Earns ~$8,000/month

Context:
- Has Level 5, wants to maintain/improve
- Uses English daily on international routes
- Wants to sound more professional
- Considering instructor role later

Goals:
- Polish pronunciation
- Learn advanced vocabulary
- Stay sharp between revalidations
```

Pain Points:
- Too busy for courses
- Existing apps too basic for his level
- Wants something he can do in hotel rooms

Willingness to Pay: $20-30/month, $200+ lifetime

1.3 User Stories & Requirements

Epic 1: Vocabulary Learning

```
US-101: As a pilot, I want to learn aviation vocabulary with flashcards
        so that I can expand my technical English.

        Acceptance Criteria:
        - Cards show term, definition, example sentence
        - Audio pronunciation for each term
        - Swipe right (know) / left (don't know)
        - Spaced repetition algorithm schedules reviews

US-102: As a pilot, I want to browse vocabulary by category
        so that I can focus on my weak areas.

        Categories:
        - Aircraft systems
        - Weather terminology
        - Navigation
        - Emergencies
        - ATC phraseology
        - Airport operations

US-103: As a pilot, I want to see my vocabulary progress
        so that I know how much I've learned.

        Show:
        - Total terms learned
        - Terms due for review
        - Mastery percentage per category
```

Epic 2: Listening Comprehension

```
US-201: As a pilot, I want to listen to ATC audio clips
```

so that I can train my ear for real communications.

Acceptance Criteria:
- Audio player with play/pause/rewind
- Speed control (0.75x, 1x, 1.25x)
- Various accents (American, British, Indian, etc.)
- Transcript reveal after attempt

US-202: As a pilot, I want comprehension questions after each clip
so that I can verify I understood correctly.

Question types:
- Multiple choice
- Fill in the blank (callsign, altitude, heading)
- True/False

US-203: As a pilot, I want to filter exercises by difficulty
so that I can progress gradually.

Levels:
- Beginner: Slow, clear, standard phrases
- Intermediate: Normal speed, some accents
- Advanced: Fast, heavy accents, non-standard situations

Epic 3: Speaking Practice

US-301: As a pilot, I want to practice standard phraseology
so that I can respond correctly to ATC.

Flow:
1. Hear ATC instruction
2. Record my readback
3. See transcription
4. Compare to correct response
5. Get pronunciation feedback

US-302: As a pilot, I want to describe aviation pictures
so that I can practice for the oral exam.

Flow:
1. See aviation scenario image
2. 2-minute timer starts
3. Record description
4. Get AI feedback on vocabulary, grammar, fluency

US-303: As a pilot, I want AI conversation practice

```
    so that I can simulate the oral exam.

    Scenarios:
    - Weather discussion
    - Emergency situations
    - Flight planning
    - Unusual events
```

Epic 4: Mock Testing

```
US-401: As a pilot, I want to take a full mock ICAO test
    so that I know if I'm ready.

    Format (mirrors real test):
    - Part 1: Listening comprehension (10 min)
    - Part 2: Picture description (5 min)
    - Part 3: Role play - ATC communication (10 min)
    - Part 4: Interview questions (5 min)

US-402: As a pilot, I want to see my mock test results
    so that I can identify weak areas.

    Scoring on 6 ICAO criteria:
    - Pronunciation (1-6)
    - Structure (1-6)
    - Vocabulary (1-6)
    - Fluency (1-6)
    - Comprehension (1-6)
    - Interaction (1-6)
```

Epic 5: Progress & Motivation

```
US-501: As a pilot, I want daily practice streaks
    so that I stay motivated to study.

US-502: As a pilot, I want push notifications reminding me to practice
    so that I don't forget.

US-503: As a pilot, I want to see my predicted ICAO level
    so that I know when I'm ready for the test.
```

1.4 Feature Prioritization (MoSCoW)

MVP (Must Have) — Week 1-6

Feature	User Story	Effort	Impact
User auth (email/password)	-	3 days	Baseline
Vocabulary flashcards	US-101	5 days	High
Spaced repetition	US-101	2 days	High
Category browsing	US-102	2 days	Medium
Listening exercises (20)	US-201	5 days	High
Comprehension questions	US-202	3 days	High
Basic speaking recording	US-301	4 days	High
Speech-to-text	US-301	2 days	High
Progress tracking	US-103, US-503	3 days	Medium
Payments (RevenueCat)	-	3 days	Critical

Phase 2 (Should Have) — Week 7-10

Feature	User Story	Effort	Impact
AI pronunciation feedback	US-301	5 days	High
Picture description	US-302	4 days	High
More listening content (+30)	US-201	5 days	Medium
Difficulty filtering	US-203	2 days	Medium
Daily streaks	US-501	2 days	Medium
Push notifications	US-502	2 days	Medium

Phase 3 (Could Have) — Week 11-16

Feature	User Story	Effort	Impact
AI conversation partner	US-303	10 days	High
Full mock test	US-401	7 days	High

Feature	User Story	Effort	Impact
Mock test scoring	US-402	5 days	High
Offline mode	-	5 days	Medium
Leaderboards	-	3 days	Low

Won't Have (This Version)

- Video content
- Live tutoring marketplace
- Flight school admin dashboard (B2B)
- Social features (friends, chat)

1.5 Product Roadmap

```
2025 Q1 (Jan-Mar): Foundation
├── Week 1-6: MVP development
├── Week 7-8: Beta launch (100 users)
├── Week 9-12: Iterate based on feedback
└── Milestone: 500 users, $1K MRR

2025 Q2 (Apr-Jun): Growth
├── AI features (conversation, advanced feedback)
├── Mock test functionality
├── Content expansion (100 listening exercises)
├── Russian localization
└── Milestone: 2,000 users, $5K MRR

2025 Q3 (Jul-Sep): Scale
├── Flight school B2B features
├── Additional languages (Chinese, Arabic)
├── Partnership with testing centers
└── Milestone: 5,000 users, $15K MRR

2025 Q4 (Oct-Dec): Expansion
├── ATC controller product variant
├── Cabin crew English
├── Advanced certifications (ATPL prep)
└── Milestone: 10,000 users, $30K MRR
```

1.6 Risk Assessment

Risk	Probability	Impact	Mitigation
Low user acquisition	Medium	High	Start with warm network (CIS pilots)
Users don't convert to paid	Medium	High	Validate pricing with beta users early
AI speech feedback inaccurate	Medium	Medium	Use established APIs (Whisper), iterate
Content quality insufficient	Low	High	Partner with actual ICAO examiner for review
Competitor copies idea	Low	Medium	Speed to market + CIS language advantage
App store rejection	Low	Medium	Follow guidelines, no restricted content

1.7 Competitive Positioning

```
              High Price
                  │
                  │
    ICAO4U ●        │        ● CaptainPilot
    (€449)        │         ($300-500)
                  │
                  │
Low ──────────────────────────────────────┼────────────────────────────────────── High
Tech                │                Tech
                  │
                  │
  Basic Apps ●        │        ● AviLingo
  ($5-20)           │         ($9-19)
                  │
                  │
              Low Price

AviLingo Positioning:
"Modern tech (AI, mobile) at accessible price"
```

PART 2: ICAO/AVIATION ENGLISH SME PERSPECTIVE

2.1 ICAO Language Proficiency Requirements — Deep Dive

Regulatory Background

```
Source: ICAO Doc 9835 AN/453
"Manual on the Implementation of ICAO Language Proficiency Requirements"

Key Points:
- Implemented March 2008 after fatal accidents involving miscommunication
- Applies to ALL pilots and controllers in international operations
- Minimum Level 4 (Operational) required
- Tests both standard phraseology AND plain English
- Revalidation required: Level 4 = every 3-4 years, Level 5 = every 6 years
```

The Six ICAO Assessment Criteria (Detailed)

1. PRONUNCIATION

```
What It Means:
- Intelligibility of speech
- Accent doesn't block understanding
- Stress, rhythm, intonation patterns

Level 4 Requirement:
"Pronunciation, stress, rhythm and intonation are influenced by
the first language or regional variation but only SOMETIMES
interfere with ease of understanding."

Common Failure Points (CIS Pilots):
- "th" sounds → "s" or "z" (think → sink)
- "w" sounds → "v" (west → vest)
- Flat intonation (sounds robotic)
- Word stress errors (dePARture vs DEparture)
- Numbers: "tree" vs "three", "fife" vs "five"

Training Approach:
- Audio comparison (model vs user)
- Phoneme-level feedback
- Shadow reading exercises
- Focus on aviation-critical sounds
```

2. STRUCTURE (Grammar)

What It Means:
- Basic grammatical structures
- Sentence patterns
- Verb tenses, word order

Level 4 Requirement:
"Basic grammatical structures and sentence patterns are used creatively and are usually well controlled. Errors may occur, particularly in unusual or unexpected circumstances, but rarely interfere with meaning."

Common Failure Points:
- Article errors ("prepare for the landing" vs "prepare for landing")
- Verb tense confusion in reports
- Word order in questions
- Preposition errors ("fly to heading" vs "fly heading")

Training Approach:
- Pattern drills with aviation context
- Error correction exercises
- Common mistake awareness

3. VOCABULARY

```
What It Means:
- Range of aviation terminology
- Plain English vocabulary
- Ability to paraphrase when needed

Level 4 Requirement:
"Vocabulary range and accuracy are usually sufficient to
communicate effectively on common, concrete and work-related
topics. Can often paraphrase successfully when lacking
vocabulary in unusual or unexpected circumstances."

Vocabulary Categories Required:

A. Standard Phraseology (~200 terms)
   - "Cleared for takeoff"
   - "Say again"
   - "Unable"
   - "Standby"
   - "Affirm/Negative"
   - "Roger"
   - "Wilco"

B. Aircraft Systems (~100 terms)
   - Aileron, elevator, rudder
   - Hydraulics, pneumatics, electrical
   - APU, GPU, engine components
   - Flight instruments

C. Weather (~80 terms)
   - METAR/TAF terminology
   - Turbulence descriptions
   - Visibility, ceiling, clouds
   - Icing conditions

D. Navigation (~60 terms)
   - VOR, NDB, ILS, RNAV
   - Waypoints, airways
   - Approach types
   - Holding patterns

E. Emergencies (~50 terms)
   - Mayday, Pan-Pan
   - Fire, smoke, fumes
   - Medical emergencies
   - Fuel emergencies
   - Hijack codes
```

F. Airport Operations (~50 terms)
  - Taxiways, aprons, gates
  - Ground equipment
  - De-icing
  - FOD

Training Approach:
- Spaced repetition flashcards
- Context-based learning (not just definitions)
- Paraphrasing exercises

4. FLUENCY

What It Means:
- Speaking rate (not too slow, not too fast)
- Minimal hesitation
- Natural flow

Level 4 Requirement:
"Produces stretches of language at an appropriate tempo.
There may be occasional loss of fluency on transition from
rehearsed or formulaic speech to spontaneous interaction,
but this does not prevent effective communication."

Common Failure Points:
- Long pauses while thinking
- "Um", "uh" fillers
- Repeating same phrase multiple times
- Speaking too slowly (sounds uncertain)
- Speaking too fast (becomes unclear)

Training Approach:
- Timed speaking exercises
- Reduce thinking time gradually
- Practice transitions from standard to plain English

5. COMPREHENSION

What It Means:
- Understanding various accents
- Understanding fast speech
- Understanding non-standard situations

Level 4 Requirement:
"Comprehension is mostly accurate on common, concrete and
work-related topics when the accent or variety used is
sufficiently intelligible for an international community
of users. When confronted with a linguistic or situational
complication, comprehension may be slower or require
clarification strategies."

Comprehension Challenges:
- American vs British vs Indian accents
- Fast controller speech
- Clipped transmissions (radio static)
- Non-standard phraseology
- Unexpected instructions

Training Approach:
- Real ATC audio from multiple regions
- Variable speed playback
- Degraded audio quality exercises
- "Say again" practice (knowing when/how to ask)

6. INTERACTION

```
What It Means:
- Back-and-forth communication
- Clarification strategies
- Checking understanding

Level 4 Requirement:
"Responses are usually immediate, appropriate, and informative.
Initiates and maintains exchanges even when dealing with an
unexpected turn of events. Deals adequately with apparent
misunderstandings by checking, confirming, or clarifying."

Key Skills:
- Readback/hearback cycle
- "Say again" requests
- Confirming critical information
- Clarifying ambiguity
- Reporting problems clearly

Training Approach:
- Role-play exercises (pilot-ATC)
- Scenario-based conversations
- Practice handling unexpected situations
```

2.2 ICAO Test Format (What Users Face)

Test Structure

```
Total Duration: 25-40 minutes
Format: Live interview with certified examiner (or computer-based)

PART 1: LISTENING COMPREHENSION (10 minutes)
 ├─── Listen to 6-8 ATC audio clips
 ├─── Answer questions about each
 ├─── Various accents and speeds
 └─── Includes non-routine situations

PART 2: PICTURE DESCRIPTION (5 minutes)
 ├─── Shown aviation-related image
 ├─── Describe what you see (2 minutes)
 ├─── Examiner asks follow-up questions
 └─── Images show: emergencies, weather, procedures, incidents

PART 3: ROLE PLAY (10 minutes)
```

```
├──── Simulate pilot-ATC communication
├──── Routine scenarios (clearances, departures)
├──── Non-routine scenarios (emergencies, deviations)
└──── Must use correct phraseology + plain English

PART 4: INTERVIEW (5-10 minutes)
├──── Open conversation about aviation topics
├──── Your flying experience
├──── Hypothetical situations
├──── Opinions on aviation safety
└──── Tests natural conversation ability
```

Scoring

```
Each of 6 criteria scored 1-6
Overall level = LOWEST score among all criteria

Example:
- Pronunciation: 4
- Structure: 5
- Vocabulary: 4
- Fluency: 4
- Comprehension: 5
- Interaction: 3  ← This becomes overall level

Result: LEVEL 3 (Fail for international operations)

Implication: App must train ALL criteria, not just vocabulary
```

2.3 Content Requirements for App

Vocabulary Database (500 terms minimum)

```
Structure per term:
{
  "term": "go-around",
  "phonetic": "/ˈɡoʊ.əˌraʊnd/",
  "part_of_speech": "noun/verb",
  "definition": "A maneuver where the pilot aborts landing and climbs away",
  "aviation_context": "Commanded by ATC or initiated by pilot when landing is unsafe",
  "example_atc": "United 472, go around, traffic on runway",
  "example_response": "Going around, United 472",
  "common_errors": "Confusing with 'missed approach' (different procedure)",
  "audio_url": "vocab/go-around.mp3",
  "category": "procedures",
  "difficulty": 2,
  "icao_criteria": ["vocabulary", "comprehension"]
}
```

Listening Exercise Database (50 exercises minimum)

```
Structure per exercise:
{
  "id": "LSN-042",
  "title": "JFK Departure Clearance with Amendment",
  "audio_url": "listening/jfk-clearance-042.mp3",
  "duration_seconds": 45,
  "transcript": "Delta 1492, cleared to Atlanta...",
  "accent": "American - New York",
  "speed": "fast",
  "difficulty": 3,
  "category": "clearance_delivery",
  "scenario_type": "routine",
  "questions": [
    {
      "type": "multiple_choice",
      "question": "What is the cleared altitude?",
      "options": ["5,000", "15,000", "FL350", "FL250"],
      "correct": "5,000",
      "explanation": "Initial altitude is 5,000, expect FL350 after departure"
    }
  ],
  "teaching_points": [
    "Listen for 'maintain' vs 'expect'",
    "Initial altitude vs cruise altitude difference"
  ]
}

Categories needed:
- Clearance delivery (10)
- Ground control (8)
- Tower (10)
- Departure (8)
- En route/Center (6)
- Approach (8)
- Emergencies (5)
- Weather deviations (5)
```

Picture Description Bank (30 images minimum)

Image categories:
1. Aircraft situations (10)
   - Takeoff with bird strike
   - Landing in crosswind
   - Engine fire on ground
   - Gear not extended
   - De-icing operation

2. Cockpit scenarios (10)
   - Warning lights illuminated
   - Weather radar showing cells
   - Low fuel indication
   - Instrument failure
   - Smoke in cockpit

3. Airport/ground scenes (10)
   - FOD on runway
   - Emergency vehicles responding
   - Ground collision
   - Adverse weather conditions
   - Runway incursion situation

Per image, provide:
- Image file
- Description of what's shown
- Key vocabulary to use
- Sample Level 4 response
- Sample Level 5 response
- Follow-up questions examiner might ask

Speaking Scenarios (20 minimum)

Scenario types:

A. Standard Phraseology Drills
   - Readback clearances
   - Request altitude change
   - Report position
   - Acknowledge instructions

B. Emergency Communications
   - Declare Mayday
   - Report engine failure

```
  - Medical emergency
  - Fuel emergency

C. Non-routine Situations
  - Request deviation for weather
  - Report traffic conflict
  - Unable compliance
  - Request priority handling

Structure per scenario:
{
  "id": "SPK-015",
  "title": "Declaring Medical Emergency",
  "category": "emergency",
  "difficulty": 3,
  "setup": "You are captain of flight ABC123. A passenger has collapsed
           and is unresponsive. You need to divert to nearest suitable airport.",
  "atc_prompt_audio": "ABC123, go ahead with your message",
  "expected_elements": [
    "Pan-Pan or Mayday declaration",
    "Nature of emergency (medical)",
    "Souls on board",
    "Fuel remaining",
    "Request (divert, medical assistance)"
  ],
  "sample_response": "Pan-Pan, Pan-Pan, Pan-Pan. ABC123, we have a medical
                     emergency. Passenger unconscious. Request immediate
                     divert to nearest suitable airport. 180 souls on board,
                     4 hours fuel remaining. Request medical assistance on arrival.",
  "scoring_rubric": {
    "vocabulary": "Uses 'Pan-Pan', 'medical emergency', 'souls on board'",
    "structure": "Clear, logical sequence of information",
    "fluency": "Calm delivery despite urgency",
    "interaction": "Provides all necessary information proactively"
  }
}
```

2.4 Pedagogical Approach

Learning Principles

```
1. SPACED REPETITION
  - Vocabulary retention requires repeated exposure
  - Intervals increase as mastery improves
```

```
   - SM-2 algorithm (same as Anki)
   - Review due cards daily

2. CONTEXTUAL LEARNING
   - Words learned in isolation are forgotten
   - Always pair terms with aviation scenarios
   - Audio context preferred over text-only

3. ACTIVE RECALL
   - Passive reading doesn't build skills
   - Force user to produce (speak, type)
   - Test before teaching (retrieval practice)

4. IMMEDIATE FEEDBACK
   - Correct errors instantly
   - Show model answer for comparison
   - Explain why something is wrong

5. PROGRESSIVE DIFFICULTY
   - Start with slow, clear audio
   - Add accents, speed, complexity
   - Match content to user's current level

6. INTERLEAVING
   - Mix different topics in sessions
   - Don't do all vocabulary, then all listening
   - Brain works harder = better retention
```

Daily Practice Structure

```
Optimal session: 15-20 minutes

WARM-UP (3 min)
  └─── Review 10 due vocabulary cards

LISTENING (5 min)
  └─── 1-2 ATC audio exercises
  └─── Comprehension questions

SPEAKING (5 min)
  └─── 2-3 phraseology drills
  └─── OR 1 picture description

REVIEW (2 min)
```

```
└──── See progress summary
└──── Preview tomorrow's focus
```

2.5 Critical Content Accuracy Requirements

What Must Be 100% Correct

```
❗ STANDARD PHRASEOLOGY
  - Exact ICAO wording required
  - "Cleared for takeoff" NOT "cleared to takeoff"
  - Source: ICAO Doc 9432 (Radiotelephony Manual)

❗ EMERGENCY PROCEDURES
  - Mayday format must be correct
  - Pan-Pan format must be correct
  - Transponder codes (7500, 7600, 7700)

❗ PHONETIC ALPHABET
  - Alpha, Bravo, Charlie... (exact)
  - Number pronunciation (tree, fife, niner)

❗ REGULATORY INFORMATION
  - Don't make claims about specific authority requirements
  - Link to official sources when possible
```

What Can Be Simplified

```
✓ Grammar explanations (simplify for non-native speakers)
✓ Scenario contexts (can be generic)
✓ Cultural examples (adapt for CIS audience)
✓ Difficulty progressions (pedagogical freedom)
```

PART 3: SENIOR BACKEND ENGINEER PERSPECTIVE

3.1 System Architecture Overview

```
┌──────────────────────────────────────────────────────────────
│                 CLIENT LAYER                 │
├──────────────────────────────────────────────────────────────
```

```
│  ┌──────────────────────┐  ┌──────────────────────┐
┌──────────────────────────────────────────┐  │
│  │  iOS App   │  │ Android App │  │  Web App (Future)   │  │
│  │  (Expo)   │  │  (Expo)   │  │ (React/Next.js)   │  │
│  └──────────┬──────────┘  └──────────┬──────────┘
└──────────────────────┬──────────────────────┘  │
│     └──────────────────────────┼──────────────────────────────────────┘       │
│            │                │
│            ▼                │
│        REST API / GraphQL          │
└────────────────────────────────────────────────────────────────────────────────

           │
           ▼
┌────────────────────────────────────────────────────────────────────────────────
│          API GATEWAY                │
├────────────────────────────────────────────────────────────────────────────────
│ • Rate limiting (100 req/min per user)          │
│ • JWT validation                   │
│ • Request logging                  │
│ • CORS handling                    │
└────────────────────────────────────────────────────────────────────────────────
           │
           ▼
┌────────────────────────────────────────────────────────────────────────────────
│         APPLICATION LAYER               │
├────────────────────────────────────────────────────────────────────────────────
│                          │
│  ┌──────────────────────────────┐  ┌──────────────────────────────┐
┌──────────────────────────────┐  │
│  │  Auth Service  │  │ Content Service │  │ Progress Service │  │
│  │         │  │         │  │         │  │
│  │ • Signup/Login │  │ • Vocabulary  │  │ • Streaks    │  │
│  │ • JWT tokens  │  │ • Listening  │  │ • Stats    │  │
│  │ • OAuth     │  │ • Speaking  │  │ • Achievements │  │
│  │ • Password   │  │ • Mock tests  │  │ • Level calc  │  │
│  └──────────────┬──────────────┘  └──────────────┬──────────────┘
└──────────────┬──────────────┘  │
│     │        │        │     │
│  ┌──────────────┴──────────────┐  ┌──────────────┴──────────────┐
┌──────────────┴──────────────┐  │
│  │ Speech Service │  │  AI Service   │  │ Payment Service │  │
│  │         │  │         │  │         │  │
│  │ • Audio upload │  │ • Feedback gen │  │ • RevenueCat  │  │
│  │ • Whisper STT  │  │ • Conversation │  │ • Entitlements │  │
│  │ • Scoring    │  │ • Scoring    │  │ • Webhooks    │  │
│  └──────────────────────────────┘  └──────────────────────────────┘
└──────────────────────────────┘  │
```

```
│                                        │
└──────────────────────────────────────────────────────────────────────────────────────────────────────────────────
                │
                ▼
┌──────────────────────────────────────────────────────────────────────────────────────────────────────────────────
│              DATA LAYER                        │
├──────────────────────────────────────────────────────────────────────────────────────────────────────────────────
│                                        │
│  ┌────────────────────────────────────────┐   ┌────────────────────────────────────────┐
┌────────────────────────────────────────┐   │
│  │  PostgreSQL   │  │     Redis      │  │  Cloudflare R2  │  │
│  │              │  │              │  │              │  │
│  │ • Users        │  │ • Sessions     │  │ • Audio files  │  │
│  │ • Progress     │  │ • Cache        │  │ • Images       │  │
│  │ • Content      │  │ • Rate limits  │  │ • User uploads │  │
│  │ • Submissions  │  │ • Streaks      │  │              │  │
│  └────────────────────────────────────────┘   └────────────────────────────────────────┘
└────────────────────────────────────────┘   │
│                                        │
└──────────────────────────────────────────────────────────────────────────────────────────────────────────────────
                │
                ▼
┌──────────────────────────────────────────────────────────────────────────────────────────────────────────────────
│              EXTERNAL SERVICES                         │
├──────────────────────────────────────────────────────────────────────────────────────────────────────────────────
│  ┌──────────────────────────────┐  ┌──────────────────────────────┐  ┌──────────────────────────────┐
│
│  │ OpenAI API  │  │ Claude API  │  │ RevenueCat  │           │
│  │ (Whisper)   │  │ (Feedback)  │  │ (Payments)  │           │
│  └──────────────────────────────┘  └──────────────────────────────┘  └──────────────────────────────┘
│
│  ┌──────────────────────────────┐  ┌──────────────────────────────┐  ┌──────────────────────────────┐
│
│  │ ElevenLabs  │  │   Sentry    │  │  Mixpanel   │           │
│  │ (TTS)       │  │  (Errors)   │  │ (Analytics) │           │
│  └──────────────────────────────┘  └──────────────────────────────┘  └──────────────────────────────┘
│
└──────────────────────────────────────────────────────────────────────────────────────────────────────────────────
```

3.2 Database Schema (Complete)

```
sql
```

```
-- ============================================================
-- USERS & AUTH
-- ============================================================

CREATE TABLE users (
    id UUID PRIMARY KEY DEFAULT gen_random_uuid(),
    email VARCHAR(255) UNIQUE NOT NULL,
    password_hash VARCHAR(255) NOT NULL,

    -- Profile
    display_name VARCHAR(100),
    native_language VARCHAR(50),  -- 'russian', 'uzbek', 'chinese'
    current_icao_level INT,       -- 0-6, 0 = not tested yet
    target_icao_level INT DEFAULT 4,
    test_date DATE,               -- When is their test scheduled?

    -- Subscription (RevenueCat handles details)
    subscription_tier VARCHAR(20) DEFAULT 'free',  -- free, basic, pro, lifetime
    subscription_expires_at TIMESTAMP,
    revenuecat_id VARCHAR(255),

    -- Metadata
    timezone VARCHAR(50) DEFAULT 'UTC',
    notifications_enabled BOOLEAN DEFAULT true,
    created_at TIMESTAMP DEFAULT NOW(),
    updated_at TIMESTAMP DEFAULT NOW(),
    last_active_at TIMESTAMP DEFAULT NOW()
);

CREATE INDEX idx_users_email ON users(email);
CREATE INDEX idx_users_revenuecat ON users(revenuecat_id);

-- ============================================================
-- VOCABULARY
-- ============================================================

CREATE TABLE vocabulary_terms (
    id UUID PRIMARY KEY DEFAULT gen_random_uuid(),
    term VARCHAR(255) NOT NULL,
    phonetic VARCHAR(255),
    definition TEXT NOT NULL,
    example_sentence TEXT,
    example_atc TEXT,             -- ATC context example
    common_errors TEXT,
    category VARCHAR(100) NOT NULL,
    difficulty INT DEFAULT 1 CHECK (difficulty BETWEEN 1 AND 5),
```

```sql
    audio_url VARCHAR(500),
    created_at TIMESTAMP DEFAULT NOW()
);

CREATE INDEX idx_vocab_category ON vocabulary_terms(category);
CREATE INDEX idx_vocab_difficulty ON vocabulary_terms(difficulty);

-- User's progress per vocabulary term (Spaced Repetition)
CREATE TABLE vocabulary_progress (
    id UUID PRIMARY KEY DEFAULT gen_random_uuid(),
    user_id UUID NOT NULL REFERENCES users(id) ON DELETE CASCADE,
    term_id UUID NOT NULL REFERENCES vocabulary_terms(id) ON DELETE CASCADE,

    -- SM-2 Algorithm fields
    ease_factor FLOAT DEFAULT 2.5,
    interval_days INT DEFAULT 1,
    repetitions INT DEFAULT 0,
    next_review_at TIMESTAMP DEFAULT NOW(),
    last_reviewed_at TIMESTAMP,

    -- Stats
    times_correct INT DEFAULT 0,
    times_incorrect INT DEFAULT 0,

    UNIQUE(user_id, term_id)
);

CREATE INDEX idx_vocab_progress_user ON vocabulary_progress(user_id);
CREATE INDEX idx_vocab_progress_next_review ON vocabulary_progress(user_id, next_review_at);

-- ============================================================
-- LISTENING EXERCISES
-- ============================================================

CREATE TABLE listening_exercises (
    id UUID PRIMARY KEY DEFAULT gen_random_uuid(),
    title VARCHAR(255) NOT NULL,
    description TEXT,
    audio_url VARCHAR(500) NOT NULL,
    transcript TEXT NOT NULL,
    duration_seconds INT NOT NULL,

    -- Categorization
    category VARCHAR(100) NOT NULL,  -- clearance, tower, emergency, etc.
    accent VARCHAR(100),             -- american, british, indian, etc.
    speed VARCHAR(20),               -- slow, normal, fast
    difficulty INT DEFAULT 1 CHECK (difficulty BETWEEN 1 AND 5),
    scenario_type VARCHAR(50),       -- routine, non_routine, emergency

    -- Teaching content
    teaching_points JSONB,           -- Array of strings

    -- Metadata
    is_premium BOOLEAN DEFAULT false,
    created_at TIMESTAMP DEFAULT NOW()
);

CREATE INDEX idx_listening_category ON listening_exercises(category);
CREATE INDEX idx_listening_difficulty ON listening_exercises(difficulty);

-- Questions for listening exercises
CREATE TABLE listening_questions (
    id UUID PRIMARY KEY DEFAULT gen_random_uuid(),
    exercise_id UUID NOT NULL REFERENCES listening_exercises(id) ON DELETE CASCADE,
    question_order INT NOT NULL,
    question_type VARCHAR(50) NOT NULL,  -- multiple_choice, fill_blank, true_false
    question_text TEXT NOT NULL,
    options JSONB,                   -- For multiple choice
    correct_answer TEXT NOT NULL,
    explanation TEXT,
    created_at TIMESTAMP DEFAULT NOW()
);

CREATE INDEX idx_questions_exercise ON listening_questions(exercise_id);

-- User's listening exercise attempts
CREATE TABLE listening_attempts (
    id UUID PRIMARY KEY DEFAULT gen_random_uuid(),
    user_id UUID NOT NULL REFERENCES users(id) ON DELETE CASCADE,
    exercise_id UUID NOT NULL REFERENCES listening_exercises(id) ON DELETE CASCADE,

    score_percent INT,               -- 0-100
    answers JSONB,                   -- {question_id: user_answer}
    completed BOOLEAN DEFAULT false,
    time_spent_seconds INT,

    created_at TIMESTAMP DEFAULT NOW()
);

CREATE INDEX idx_listening_attempts_user ON listening_attempts(user_id);

-- ============================================================
-- SPEAKING EXERCISES
-- ============================================================
CREATE TABLE speaking_scenarios (
    id UUID PRIMARY KEY DEFAULT gen_random_uuid(),
    title VARCHAR(255) NOT NULL,
    scenario_type VARCHAR(50) NOT NULL,  -- phraseology, picture, conversation
    category VARCHAR(100),               -- emergency, routine, etc.
    difficulty INT DEFAULT 1 CHECK (difficulty BETWEEN 1 AND 5),

    -- Content
    instructions TEXT NOT NULL,
    setup_text TEXT,                    -- Scenario context
    atc_prompt_audio_url VARCHAR(500),   -- What ATC says
    atc_prompt_text TEXT,

    -- For picture descriptions
    image_url VARCHAR(500),

    -- Expected response info
    expected_elements JSONB,             -- Key things to include
    sample_response TEXT,
    scoring_rubric JSONB,

    is_premium BOOLEAN DEFAULT false,
    created_at TIMESTAMP DEFAULT NOW()
);

CREATE INDEX idx_speaking_type ON speaking_scenarios(scenario_type);
CREATE INDEX idx_speaking_category ON speaking_scenarios(category);

-- User's speaking submissions
CREATE TABLE speaking_submissions (
    id UUID PRIMARY KEY DEFAULT gen_random_uuid(),
    user_id UUID NOT NULL REFERENCES users(id) ON DELETE CASCADE,
    scenario_id UUID NOT NULL REFERENCES speaking_scenarios(id) ON DELETE CASCADE,

    -- Audio
    audio_url VARCHAR(500) NOT NULL,     -- User's recording
    duration_seconds INT,

    -- Transcription
    transcript TEXT,                    -- From Whisper

    -- AI Feedback
    ai_feedback JSONB,                  -- Detailed feedback
    scores JSONB,                       -- {pronunciation: 4, fluency: 3, ...}
    overall_score INT,
    -- Metadata
    processing_status VARCHAR(20) DEFAULT 'pending',  -- pending, processing, completed, failed
    created_at TIMESTAMP DEFAULT NOW()
);

CREATE INDEX idx_speaking_submissions_user ON speaking_submissions(user_id);
CREATE INDEX idx_speaking_submissions_status ON speaking_submissions(processing_status);

-- ============================================================
-- MOCK TESTS
-- ============================================================

CREATE TABLE mock_tests (
    id UUID PRIMARY KEY DEFAULT gen_random_uuid(),
    title VARCHAR(255) NOT NULL,
    description TEXT,
    duration_minutes INT DEFAULT 30,

    -- Test sections configuration
    sections JSONB NOT NULL,
    /* Structure:
    {
      "listening": {"exercise_ids": [...], "duration_minutes": 10},
      "picture": {"scenario_ids": [...], "duration_minutes": 5},
      "roleplay": {"scenario_ids": [...], "duration_minutes": 10},
      "interview": {"questions": [...], "duration_minutes": 5}
    }
    */

    is_premium BOOLEAN DEFAULT true,
    created_at TIMESTAMP DEFAULT NOW()
);

-- User's mock test attempts
CREATE TABLE mock_test_attempts (
    id UUID PRIMARY KEY DEFAULT gen_random_uuid(),
    user_id UUID NOT NULL REFERENCES users(id) ON DELETE CASCADE,
    test_id UUID NOT NULL REFERENCES mock_tests(id) ON DELETE CASCADE,

    -- Results
    section_scores JSONB,              -- Per-section scores
    criteria_scores JSONB,             -- 6 ICAO criteria scores
    overall_level INT,                 -- 1-6

    -- Timing
    started_at TIMESTAMP,
    completed_at TIMESTAMP,
    -- Detailed responses stored separately
    responses JSONB,

    status VARCHAR(20) DEFAULT 'in_progress',  -- in_progress, completed, abandoned
    created_at TIMESTAMP DEFAULT NOW()
);

CREATE INDEX idx_mock_attempts_user ON mock_test_attempts(user_id);

-- ============================================================
-- PROGRESS & GAMIFICATION
-- ============================================================

CREATE TABLE daily_progress (
    id UUID PRIMARY KEY DEFAULT gen_random_uuid(),
    user_id UUID NOT NULL REFERENCES users(id) ON DELETE CASCADE,
    date DATE NOT NULL,

    -- Activity counts
    vocab_reviewed INT DEFAULT 0,
    vocab_learned_new INT DEFAULT 0,
    listening_completed INT DEFAULT 0,
    speaking_completed INT DEFAULT 0,
    practice_minutes INT DEFAULT 0,

    -- Points/XP
    xp_earned INT DEFAULT 0,

    UNIQUE(user_id, date)
);

CREATE INDEX idx_daily_progress_user_date ON daily_progress(user_id, date);

-- Streak tracking
CREATE TABLE streaks (
    id UUID PRIMARY KEY DEFAULT gen_random_uuid(),
    user_id UUID UNIQUE NOT NULL REFERENCES users(id) ON DELETE CASCADE,

    current_streak INT DEFAULT 0,
    longest_streak INT DEFAULT 0,
    last_practice_date DATE,

    updated_at TIMESTAMP DEFAULT NOW()
);

-- Achievements
```

```sql
CREATE TABLE achievements (
    id UUID PRIMARY KEY DEFAULT gen_random_uuid(),
    code VARCHAR(50) UNIQUE NOT NULL,  -- 'first_lesson', 'week_streak', etc.
    name VARCHAR(100) NOT NULL,
    description TEXT,
    icon_url VARCHAR(500),
    xp_reward INT DEFAULT 0
);

CREATE TABLE user_achievements (
    id UUID PRIMARY KEY DEFAULT gen_random_uuid(),
    user_id UUID NOT NULL REFERENCES users(id) ON DELETE CASCADE,
    achievement_id UUID NOT NULL REFERENCES achievements(id) ON DELETE CASCADE,
    earned_at TIMESTAMP DEFAULT NOW(),

    UNIQUE(user_id, achievement_id)
);

-- ============================================================
-- NOTIFICATIONS
-- ============================================================

CREATE TABLE push_tokens (
    id UUID PRIMARY KEY DEFAULT gen_random_uuid(),
    user_id UUID NOT NULL REFERENCES users(id) ON DELETE CASCADE,
    token VARCHAR(500) NOT NULL,
    platform VARCHAR(20) NOT NULL,      -- ios, android
    created_at TIMESTAMP DEFAULT NOW(),

    UNIQUE(user_id, token)
);

-- ============================================================
-- ANALYTICS / AUDIT
-- ============================================================

CREATE TABLE user_events (
    id UUID PRIMARY KEY DEFAULT gen_random_uuid(),
    user_id UUID REFERENCES users(id) ON DELETE SET NULL,
    event_type VARCHAR(100) NOT NULL,   -- 'lesson_completed', 'subscription_started'
    event_data JSONB,
    created_at TIMESTAMP DEFAULT NOW()
);

CREATE INDEX idx_events_user ON user_events(user_id);
CREATE INDEX idx_events_type ON user_events(event_type);
CREATE INDEX idx_events_created ON user_events(created_at);
```

3.3 API Design

Authentication Endpoints

```yaml
POST /api/v1/auth/register
  Request:
    email: string (required)
    password: string (min 8 chars)
    native_language: string (optional)
  Response:
    user: User object
    access_token: JWT
    refresh_token: JWT

POST /api/v1/auth/login
  Request:
    email: string
    password: string
  Response:
    user: User object
    access_token: JWT
    refresh_token: JWT

POST /api/v1/auth/refresh
  Request:
    refresh_token: string
  Response:
    access_token: JWT

POST /api/v1/auth/logout
  Headers: Authorization: Bearer <token>
  Response: 204 No Content

POST /api/v1/auth/forgot-password
  Request:
    email: string
  Response: 200 OK (always, for security)
```

User Endpoints

```yaml
GET /api/v1/users/me
  Headers: Authorization: Bearer <token>
  Response:
    id: uuid
    email: string
    display_name: string
    native_language: string
    current_icao_level: int
    target_icao_level: int
    subscription_tier: string
    subscription_expires_at: datetime
    stats:
      current_streak: int
      total_vocab_learned: int
      total_practice_minutes: int
      predicted_level: int

PATCH /api/v1/users/me
  Headers: Authorization: Bearer <token>
  Request:
    display_name: string (optional)
    native_language: string (optional)
    target_icao_level: int (optional)
    test_date: date (optional)
    notifications_enabled: boolean (optional)
  Response: Updated User object
```

Vocabulary Endpoints

```yaml
```

```
GET /api/v1/vocabulary
 Headers: Authorization: Bearer <token>
 Query params:
  category: string (optional, filter by category)
  difficulty: int (optional, 1-5)
  limit: int (default 50)
  offset: int (default 0)
 Response:
  items: VocabularyTerm[]
  total: int
  categories: string[] (available categories)

GET /api/v1/vocabulary/:id
 Response: VocabularyTerm object with full details

GET /api/v1/vocabulary/review-queue
 Headers: Authorization: Bearer <token>
 Query params:
  limit: int (default 20)
 Response:
  items: VocabularyTerm[] (due for review, sorted by priority)
  total_due: int

POST /api/v1/vocabulary/:id/review
 Headers: Authorization: Bearer <token>
 Request:
  quality: int (0-5, SM-2 quality rating)
  # 0 = complete blackout
  # 1 = incorrect, but remembered upon seeing answer
  # 2 = incorrect, but answer seemed easy to recall
  # 3 = correct with serious difficulty
  # 4 = correct after hesitation
  # 5 = perfect response
 Response:
  next_review_at: datetime
  interval_days: int
  ease_factor: float
```

Listening Endpoints

```yaml
```

```
GET /api/v1/listening
  Headers: Authorization: Bearer <token>
  Query params:
    category: string (optional)
    difficulty: int (optional)
    completed: boolean (optional, filter by completion status)
    limit: int (default 20)
  Response:
    items: ListeningExercise[] (without transcript)
    total: int

GET /api/v1/listening/:id
  Headers: Authorization: Bearer <token>
  Response:
    exercise: ListeningExercise (full details)
    questions: ListeningQuestion[]
    user_attempts: ListeningAttempt[] (previous attempts)

POST /api/v1/listening/:id/submit
  Headers: Authorization: Bearer <token>
  Request:
    answers: { question_id: answer }
    time_spent_seconds: int
  Response:
    score_percent: int
    correct_answers: { question_id: correct_answer }
    explanations: { question_id: explanation }
    xp_earned: int
```

Speaking Endpoints

yaml

```yaml
GET /api/v1/speaking/scenarios
  Headers: Authorization: Bearer <token>
  Query params:
    type: string (phraseology, picture, conversation)
    category: string (optional)
    difficulty: int (optional)
  Response:
    items: SpeakingScenario[]

GET /api/v1/speaking/scenarios/:id
  Response: SpeakingScenario with full details

POST /api/v1/speaking/scenarios/:id/submit
  Headers: Authorization: Bearer <token>
  Content-Type: multipart/form-data
  Request:
    audio: file (webm, m4a, mp3)
  Response:
    submission_id: uuid
    status: "processing"

GET /api/v1/speaking/submissions/:id
  Headers: Authorization: Bearer <token>
  Response:
    submission: SpeakingSubmission
    # If status = completed:
    transcript: string
    ai_feedback: object
    scores: { pronunciation: int, fluency: int, ... }
    sample_response: string (for comparison)

# Webhook-style polling or WebSocket for real-time status
GET /api/v1/speaking/submissions/:id/status
  Response:
    status: "pending" | "processing" | "completed" | "failed"
    progress_percent: int (if processing)
```

Progress Endpoints

yaml

```
GET /api/v1/progress/daily
  Headers: Authorization: Bearer <token>
  Query params:
    start_date: date (optional, default 30 days ago)
    end_date: date (optional, default today)
  Response:
    days: DailyProgress[]
    streak: { current: int, longest: int }
    totals: { vocab: int, listening: int, speaking: int, minutes: int }

GET /api/v1/progress/stats
  Headers: Authorization: Bearer <token>
  Response:
    vocab:
      total_learned: int
      mastery_percent: int
      by_category: { category: { learned: int, total: int } }
    listening:
      completed: int
      average_score: int
    speaking:
      submissions: int
      average_score: int
    predicted_icao_level: int
    readiness_percent: int

GET /api/v1/progress/achievements
  Headers: Authorization: Bearer <token>
  Response:
    earned: Achievement[]
    available: Achievement[] (not yet earned)
```

Payment Endpoints

yaml

```
POST /api/v1/payments/verify
  Headers: Authorization: Bearer <token>
  Request:
    platform: "ios" | "android"
    receipt: string (from RevenueCat)
  Response:
    subscription_tier: string
    expires_at: datetime
    is_active: boolean

GET /api/v1/payments/entitlements
  Headers: Authorization: Bearer <token>
  Response:
    tier: string
    features: string[]
    expires_at: datetime
    is_trial: boolean
```

3.4 Audio Processing Pipeline

```
┌──────────────────────────────────────────────────────────────
│                    AUDIO UPLOAD FLOW                         │
└──────────────────────────────────────────────────────────────

1. CLIENT RECORDS AUDIO
   ├── Format: WebM (Chrome) or M4A (iOS native)
   ├── Sample rate: 16kHz minimum
   ├── Max duration: 3 minutes
   └── Max file size: 10MB

2. UPLOAD TO BACKEND
   ├── POST /api/v1/speaking/scenarios/:id/submit
   ├── Content-Type: multipart/form-data
   ├── Validate: file type, size, duration
   └── Return: submission_id, status: "processing"

3. STORE RAW AUDIO
   ├── Upload to Cloudflare R2
   ├── Path: /audio/{user_id}/{submission_id}.webm
   └── Set expiration: 90 days

4. QUEUE PROCESSING JOB
   ├── Add to Redis queue (or use background worker)
   └─── Job: { submission_id, audio_url, scenario_id }

5. TRANSCRIPTION (Worker)
   │
   ├─── Download audio from R2
   │
   ├─── Convert to WAV 16kHz mono (ffmpeg)
   │    └─── ffmpeg -i input.webm -ar 16000 -ac 1 output.wav
   │
   ├─── Send to OpenAI Whisper API
   │    POST https://api.openai.com/v1/audio/transcriptions
   │    {
   │      "model": "whisper-1",
   │      "file": audio_file,
   │      "language": "en",
   │      "response_format": "verbose_json",
   │      "timestamp_granularities": ["word"]
   │    }
   │
   └─── Store transcript in database

6. AI FEEDBACK GENERATION (Worker)
   │
   ├─── Build prompt with:
   │    ├─── Scenario context
   │    ├─── User transcript
   │    ├─── Expected elements
   │    └─── Scoring rubric
   │
   ├─── Send to Claude API
   │    POST https://api.anthropic.com/v1/messages
   │    {
   │      "model": "claude-sonnet-4-20250514",
   │      "messages": [{"role": "user", "content": prompt}],
   │      "max_tokens": 1000
   │    }
   │
   ├─── Parse structured feedback:
   │    {
   │      "scores": {
   │        "pronunciation": 4,
   │        "structure": 4,
   │        "vocabulary": 5,
   │        "fluency": 3,
   │        "comprehension": 4,
   │        "interaction": 4
   │      },
    │      "overall_level": 4,
    │      "strengths": ["Good use of standard phraseology", ...],
    │      "improvements": ["Reduce hesitation pauses", ...],
    │      "corrected_version": "...",
    │      "detailed_feedback": "..."
    │    }
    │
    └──── Store feedback in database

7. NOTIFY CLIENT
    ├──── Update submission status to "completed"
    ├──── Send push notification (optional)
    └──── Client polls or receives WebSocket update
```

AI Feedback Prompt Template

python

```
FEEDBACK_PROMPT = """You are an ICAO Aviation English examiner providing feedback
on a pilot's speaking exercise.

SCENARIO:
{scenario_description}

EXPECTED ELEMENTS TO INCLUDE:
{expected_elements}

PILOT'S RESPONSE (transcribed):
"{user_transcript}"

SAMPLE CORRECT RESPONSE:
"{sample_response}"

Evaluate the pilot's response and provide:

1. SCORES (1-6 scale for each, where 4 = ICAO Operational Level):
   - Pronunciation: Is speech clear and understandable despite accent?
   - Structure: Are sentences grammatically correct?
   - Vocabulary: Is aviation terminology used appropriately?
   - Fluency: Is speech smooth with appropriate pace?
   - Comprehension: Did they address the scenario correctly?
   - Interaction: Would this be effective communication with ATC?

2. OVERALL ICAO LEVEL (1-6, based on lowest criterion score)

3. STRENGTHS (2-3 specific things done well)

4. AREAS FOR IMPROVEMENT (2-3 specific suggestions with examples)

5. CORRECTED VERSION (how a Level 5 pilot might say it)

Respond in JSON format:
{
  "scores": {
    "pronunciation": <int>,
    "structure": <int>,
    "vocabulary": <int>,
    "fluency": <int>,
    "comprehension": <int>,
    "interaction": <int>
  },
  "overall_level": <int>,
  "strengths": [<string>, ...],
  "improvements": [<string>, ...],
  "corrected_version": <string>,
  "detailed_feedback": <string>
}

Be encouraging but honest. Use simple English in feedback since the user
is a non-native speaker. Focus on actionable improvements."""
```

3.5 Spaced Repetition Algorithm (SM-2)

```
python
```

```python
"""
SM-2 Algorithm Implementation for Vocabulary Learning
Based on: https://www.supermemo.com/en/archives1990-2015/english/ol/sm2
"""

from datetime import datetime, timedelta
from dataclasses import dataclass
from typing import Tuple


@dataclass
class ReviewResult:
    next_review_at: datetime
    new_interval: int  # days
    new_ease_factor: float
    new_repetitions: int


def calculate_next_review(
    quality: int,  # 0-5 rating from user
    current_interval: int,  # current interval in days
    current_ease_factor: float,  # current E-Factor (default 2.5)
    current_repetitions: int,  # number of successful repetitions
) -> ReviewResult:
    """
    Calculate the next review date based on SM-2 algorithm.

    Quality ratings:
    0 - Complete blackout, no memory
    1 - Incorrect, but remembered upon seeing answer
    2 - Incorrect, but answer seemed easy to recall
    3 - Correct response with serious difficulty
    4 - Correct response after hesitation
    5 - Perfect response, instant recall
    """

    # If quality < 3, reset repetitions (failed recall)
    if quality < 3:
        new_repetitions = 0
        new_interval = 1  # Review tomorrow
        new_ease_factor = max(1.3, current_ease_factor - 0.2)
    else:
        # Successful recall
        new_repetitions = current_repetitions + 1

        # Calculate new interval
        if new_repetitions == 1:
            new_interval = 1
        elif new_repetitions == 2:
            new_interval = 6
        else:
            new_interval = round(current_interval * current_ease_factor)

        # Update ease factor
        # EF' = EF + (0.1 - (5 - q) * (0.08 + (5 - q) * 0.02))
        new_ease_factor = current_ease_factor + (
            0.1 - (5 - quality) * (0.08 + (5 - quality) * 0.02)
        )
        new_ease_factor = max(1.3, new_ease_factor)  # Minimum EF is 1.3

    next_review_at = datetime.utcnow() + timedelta(days=new_interval)

    return ReviewResult(
        next_review_at=next_review_at,
        new_interval=new_interval,
        new_ease_factor=new_ease_factor,
        new_repetitions=new_repetitions
    )


# Usage example:
# User knows the word well (quality = 4)
# result = calculate_next_review(
#     quality=4,
#     current_interval=6,
#     current_ease_factor=2.5,
#     current_repetitions=2
# )
# result.new_interval might be 15 days
```

3.6 Security Considerations

Authentication

```python
```

```python
# JWT Configuration
JWT_CONFIG = {
    "algorithm": "HS256",
    "access_token_expire_minutes": 60,
    "refresh_token_expire_days": 30,
    "secret_key": os.environ["JWT_SECRET_KEY"], # 256-bit random
}

# Password Hashing
from passlib.context import CryptContext
pwd_context = CryptContext(schemes=["bcrypt"], deprecated="auto")

def hash_password(password: str) -> str:
    return pwd_context.hash(password)

def verify_password(plain: str, hashed: str) -> bool:
    return pwd_context.verify(plain, hashed)
```

Rate Limiting

python

```python
# Redis-based rate limiting
RATE_LIMITS = {
    "auth": {"requests": 5, "window_seconds": 60},      # Login attempts
    "api": {"requests": 100, "window_seconds": 60},     # General API
    "audio_upload": {"requests": 10, "window_seconds": 60}, # Speaking submissions
    "ai_feedback": {"requests": 20, "window_seconds": 60},  # AI calls
}

async def check_rate_limit(user_id: str, limit_type: str) -> bool:
    key = f"rate_limit:{limit_type}:{user_id}"
    limit = RATE_LIMITS[limit_type]

    current = await redis.incr(key)
    if current == 1:
        await redis.expire(key, limit["window_seconds"])

    return current <= limit["requests"]
```

Input Validation

python

```python
from pydantic import BaseModel, EmailStr, validator

class UserCreate(BaseModel):
    email: EmailStr
    password: str

    @validator('password')
    def password_strength(cls, v):
        if len(v) < 8:
            raise ValueError('Password must be at least 8 characters')
        if not any(c.isupper() for c in v):
            raise ValueError('Password must contain uppercase letter')
        if not any(c.isdigit() for c in v):
            raise ValueError('Password must contain a digit')
        return v

class AudioUpload(BaseModel):
    file_size: int
    duration_seconds: int
    mime_type: str

    @validator('file_size')
    def validate_size(cls, v):
        if v > 10 * 1024 * 1024:  # 10MB
            raise ValueError('File too large')
        return v

    @validator('duration_seconds')
    def validate_duration(cls, v):
        if v > 180:  # 3 minutes
            raise ValueError('Recording too long')
        return v

    @validator('mime_type')
    def validate_type(cls, v):
        allowed = ['audio/webm', 'audio/mp4', 'audio/mpeg', 'audio/m4a']
        if v not in allowed:
            raise ValueError('Invalid audio format')
        return v
```

3.7 Infrastructure & DevOps

Deployment Architecture

yaml

```yaml
# Railway deployment (recommended for MVP)

Production:
  Backend:
    - FastAPI app (2 instances)
    - Auto-scaling based on CPU
    - Health check: /health

  Database:
    - PostgreSQL 15
    - 1GB RAM minimum
    - Daily backups

  Cache:
    - Redis 7
    - 256MB RAM

  Workers:
    - Background job processor
    - 1 instance (scale as needed)

Environment Variables:
  - DATABASE_URL
  - REDIS_URL
  - JWT_SECRET_KEY
  - OPENAI_API_KEY
  - ANTHROPIC_API_KEY
  - CLOUDFLARE_R2_ACCESS_KEY
  - CLOUDFLARE_R2_SECRET_KEY
  - REVENUECAT_API_KEY
  - SENTRY_DSN
```

Monitoring & Observability

python

```
# Sentry for error tracking
import sentry_sdk
from sentry_sdk.integrations.fastapi import FastApiIntegration

sentry_sdk.init(
    dsn=os.environ["SENTRY_DSN"],
    integrations=[FastApiIntegration()],
    traces_sample_rate=0.1,
    environment=os.environ.get("ENVIRONMENT", "development"),
)

# Structured logging
import structlog

logger = structlog.get_logger()

# Usage
logger.info(
    "speaking_submission_completed",
    user_id=user_id,
    scenario_id=scenario_id,
    score=score,
    processing_time_ms=processing_time,
)
```

Cost Estimation (Monthly)

```
Infrastructure (Railway):
├── Backend (2 instances)    $20
├── PostgreSQL              $10
├── Redis                   $5
└── Subtotal                $35

External Services:
├── OpenAI Whisper          $50-200 (based on usage)
│   └── ~$0.006/minute of audio
│   └── 1000 submissions × 1 min = $6
│   └── Budget for 10,000 = $60
│
├── Anthropic Claude        $30-100
│   └── ~$0.003/1K input + $0.015/1K output
│   └── 1000 feedback requests ≈ $30
│
├── Cloudflare R2           $5-15
│   └── First 10GB free
│     └─── $0.015/GB after
│
├─── RevenueCat              $0 (free under $2.5K MRR)
│
└─── Subtotal                $85-315

TOTAL: ~$120-350/month at MVP scale
```

3.8 Testing Strategy

```
python
```

```
# Test structure
tests/
├── unit/
│   ├── test_spaced_repetition.py
│   ├── test_scoring.py
│   └── test_validators.py
├── integration/
│   ├── test_auth_flow.py
│   ├── test_vocabulary_api.py
│   ├── test_listening_api.py
│   └── test_speaking_api.py
├── e2e/
│   └── test_user_journey.py
└── fixtures/
    ├── audio_samples/
    └── mock_responses/

# Example unit test
def test_spaced_repetition_quality_5():
    """Perfect recall should increase interval significantly."""
    result = calculate_next_review(
        quality=5,
        current_interval=6,
        current_ease_factor=2.5,
        current_repetitions=2
    )
    assert result.new_interval == 15  # 6 * 2.5
    assert result.new_ease_factor == 2.6  # Increased
    assert result.new_repetitions == 3

def test_spaced_repetition_quality_1():
    """Failed recall should reset to 1 day."""
    result = calculate_next_review(
        quality=1,
        current_interval=30,
        current_ease_factor=2.5,
        current_repetitions=5
    )
    assert result.new_interval == 1
    assert result.new_repetitions == 0
    assert result.new_ease_factor == 2.3  # Decreased
```

SUMMARY: What To Build First

MVP Checklist (6 Weeks)

```
Week 1: Foundation
□ Expo project setup
□ FastAPI backend skeleton
□ PostgreSQL schema (core tables only)
□ User auth (register, login, JWT)
□ Basic navigation UI

Week 2: Vocabulary
□ Seed 100 vocabulary terms
□ Flashcard UI with audio
□ SM-2 spaced repetition
□ Review queue endpoint
□ Progress tracking

Week 3: Listening
□ Create 20 listening exercises
□ Audio player component
□ Comprehension questions
□ Score calculation
□ Cloudflare R2 setup

Week 4: Speaking (Basic)
□ Audio recording component
□ Upload to backend
□ Whisper integration
□ Display transcription
□ Basic comparison UI

Week 5: Speaking (AI)
□ Claude feedback integration
□ Scoring display
□ Feedback UI
□ Picture description exercises

Week 6: Polish & Launch
□ RevenueCat integration
□ Paywall implementation
□ Push notifications
□ Streak logic
□ Bug fixes
□ TestFlight / Beta release
```

You now have the complete blueprint. Build it. 🛫