aijai: Technical Architecture & Feature Specification

1. System Overview & High-Level Architecture

aijai is a cloud-native, multi-tenant platform built on a microservices architecture. The core components serve three distinct user bases: **Learners**, **Owls** (Experts), and **Enterprise Administrators**.

Core Tenets:

- **Mobile-First:** The Learner experience is a **Progressive Web App (PWA)** for offline capability and app-like usability.
- **AI-Augmented, Not AI-Driven:** AI services assist in content generation and personalization, but human judgement is the final authority.
- **Data Isolation:** Strict tenant-based data segregation, especially for Proprietary Domains.

2. Core Technical Components

A. The Learner PWA (Progressive Web App)

- **Technology Stack:** React.js / Vue.js frontend, with a PWA manifest and service worker for offline caching of scenarios.
- **Key Features:**
    1. **Onboarding & Personalization Engine:**
        - Free-form text input analyzed by NLP to suggest initial domains.
        - Renders the personalized **7x7 Skills Matrix**.
    2. **Scenario Player:**
        - Presents scenario, records user ratings (1-5) for three options.
        - Calculates and displays percentage score based on variance from expert rating.
        - Reveals the **"Expert View"** using the **Hybrid Rationale** format (Scores + Integrated Narrative + Key Takeaway).
    3. **Progression Engine:**
        - Tracks running average score per domain.
        - Manages tiered unlocks (Foundation -> Intermediate -> Advanced) based on proficiency gates (e.g., 75% average).
        - Updates the visual "snail shell" progression on the matrix.
    4. **Gentle Social Benchmarking:**
        - Displays anonymized, aggregate data (e.g., "Your score was higher than 70% of your peers on this scenario").
        - Shows team/cohort averages without individual identification.

B. The Owl Portal (Web Application)

- **Technology Stack:** Separate web application (likely same frontend framework as PWA for code reuse, but different build).
- **Key Features:**
    1. **The Mission Board:**
        - Displays available scenario "briefs" (AI-generated from news harvest or editor-created).
        - Implements a "claim" system with temporary reservation to prevent duplication.

2. **AI-Powered Scenario Creation Studio:**
    - **AI Co-pilot:** Integrates with LLM APIs (e.g., OpenAI, Anthropic) to:
        - Suggest draft scenarios based on a brief or an Owl's raw input.
        - Generate three potential responses.
        - Act as a "Socratic interviewer" to draw out insights.
    - **Structured Form:** Guides the Owl through the **Hybrid Rationale** creation (Scores, Integrated Narrative, Key Takeaway).
    - **Similarity Checker:** Runs real-time checks against the existing library to flag potential duplicates.
3. **Impact Dashboard:**
    - Shows metrics: scenarios published, "Thank Yous" received, aggregate learner scores on their content.
4. **Community Features:** Access to Owl directories, messaging, and event information.

C. The Admin & Analytics Dashboard (Web Application)

- **Audience:** Enterprise L&D managers, aijai internal editors.
- **Key Features:**
    1. **Enterprise Analytics Engine:**
        - **Anonymized Aggregation:** Processes user data to generate heat maps, trend analyses, and cohort comparisons (by tenure, department, location).
        - **"Wisdom Gap" Identification:** Flags domains with high score variance.
    2. **Content Management System (CMS):**
        - For editors to review, edit, and approve scenarios.
        - Manages the "Mission Board" and domain taxonomy.
    3. **Tenant Management:** Manages enterprise clients, their users, and their Proprietary Domains.

D. Backend Microservices

A set of interconnected services handling specific domains of logic.

1. **User Service:** Authentication, authorization, profile management.
2. **Content Service:** Manages scenarios, domains, ratings, and scores. Core of the learning engine.
3. **AI Service:** Orchestrates calls to external LLM APIs for briefing, drafting, and similarity checks.
4. **Analytics Service:** Collects and aggregates anonymized data for reporting.
5. **Notification Service:** Handles push notifications, emails ("Thank you" alerts, weekly goals).

E. Data Layer

- **Primary Database:** PostgreSQL for structured data (users, scenarios, scores, domains).
- **Analytics Database:** A columnar database like Amazon Redshift or Snowflake for powering the analytics dashboard.
- **File Storage:** AWS S3 or equivalent for storing any rich media (e.g., if scenarios eventually include images).

3. Key Technical Features & Algorithms

A. The Scoring Algorithm

- `Scenario Score (%) = 100% - ( ( |U₁-E₁| + |U₂-E₂| + |U₃-E₃| ) / 12 ) * 100 )`
- Where `U` is the User's rating and `E` is the Expert's rating for each of the three options. The maximum possible difference is 12 (if the user is wrong by 4 points on all three), which is normalized to 0%.

B. Personalization & Recommendation Engine

- Analyzes a user's performance patterns to suggest new domains and scenarios that address their "soft spots."
- Prioritizes scenarios that have generated high "Bad News Insights" for similar learners.

C. Content Duplication Prevention

- **Vector Embeddings:** Each scenario's core dilemma is converted into a vector via an embedding model (e.g., from OpenAI).
- **Similarity Search:** A vector database (e.g., Pinecone) is queried to find existing scenarios with similar embeddings when an Owl drafts a new one.

4. AI Integration Points

- **Harvesting:** AI scans RSS feeds, news APIs for domain-specific trends to generate "briefs."
- **Drafting:** AI acts as a co-pilot for Owls, turning anecdotes into structured scenarios.
- **Onboarding:** NLP parses the learner's free-text narrative to suggest initial domains.
- **Moderation:** AI performs initial checks for content quality and guideline adherence before human editorial review.

5. Infrastructure & DevOps

- **Cloud Provider:** AWS, Google Cloud, or Azure.
- **Containerization:** Docker and Kubernetes for scalable, resilient deployment of microservices.
- **CI/CD Pipeline:** Automated testing and deployment.

This architecture supports the complex, multi-sided nature of the platform while remaining scalable and focused on the core user experience of judgement calibration. The technical features are directly mapped to the unique value propositions we've defined.