**Sensa Learn: Required Features & Tech Stack**

This stack is designed to integrate seamlessly with the existing PBL architecture, primarily leveraging Amazon Bedrock for AI-driven features and Amazon RDS for data storage.

**1. Feature: Chapter Complexity Score**

* **What it does:** As described in the design prompt, the Sensa Learn Dashboard shows a complexity score ("High," "Medium," "Low") for each chapter. This score is calculated automatically after the main concept map is generated.
* **How it works:** The back-end will analyze the graph structure of each chapter's concept map. The score is determined by a formula that considers the number of keywords (nodes), the number of relationships (edges), and the density of cross-chapter connections.
* **Tech Stack:**
  + **Compute:** The calculation logic will be a **Python** function within the existing **AWS Fargate** processing worker. It runs as the final step of the document analysis pipeline.
  + **Storage:** The calculated score (e.g., "High") is stored as metadata within the chapter\_concept\_maps JSONB object in **Amazon RDS**.

**2. Feature: Core Concept Summarization**

* **What it does:** When a user enters the Analogy View, the system first presents a clean, concise summary of the complex topic, as specified in the design.
* **How it works:** The back-end retrieves all the keywords and defined relationships for the selected chapter from RDS. It then makes a dedicated API call to Claude 3.5 Sonnet, instructing it to synthesize this structured data into a digestible paragraph.
* **Tech Stack:**
  + **AI Model:** **Amazon Bedrock (Claude 3.5 Sonnet)** for its strong summarization capabilities.
  + **API Endpoint:** A new endpoint, such as GET /sensa-learn/chapter/{chapter\_id}/summary, will be created in the **FastAPI** application.

**3. Feature: Personalized Analogy Generation**

* **What it does:** This is the core feature of Sensa Learn. The system generates several analogy options to simplify the core concept, tailored to the user's personal profile (age, location, interests).
* **How it works:** This requires a sophisticated, context-rich call to the LLM. The back-end sends the core concept summary *along with* the user's profile data. The prompt explicitly instructs the AI to use the profile data to create relevant analogies and to provide a "Personalization Hint" explaining its reasoning.
* **Tech Stack:**
  + **AI Model:** **Amazon Bedrock (Claude 3.5 Sonnet)**.
  + **API Endpoint:** GET /sensa-learn/chapter/{chapter\_id}/analogies. This endpoint will query both the chapter data and the user's profile before calling Bedrock.
  + **Database (Amazon RDS):** The user\_profiles table (see below) is essential for providing the personalization context.

**4. Feature: User Profile & Feedback System**

* **What it does:** The system needs to collect and store user preferences to power the analogy generation, as well as gather feedback on the quality of those analogies.
* **How it works:**
  + Standard database tables will store user profile information and feedback.
  + API endpoints will allow the front-end to create, read, and update this information.
* **Tech Stack:**
  + **Database (Amazon RDS):**
    - A user\_profiles table will be added to store fields like age\_range, location, and a JSONB field for interests.
    - A new analogy\_feedback table will be created to log user ratings ("thumbs up/down") for each analogy, linking user\_id, the chapter, and the analogy content. This data is crucial for future model fine-tuning.
  + **API (FastAPI on Fargate):**
    - PUT /profile: An endpoint for the front-end to save the user's learning preferences.
    - POST /feedback/analogy: An endpoint to record the user's ratings.