Full Stack + RAG Assesssment

🔍 Objective

Build a **full-stack AI-powered fitness & nutrition assistant** that enables users to track workouts and meals, while receiving personalized, data-informed recommendations. The system should:

- **Capture structured user data** (profile, workouts, meals)
- **Store conversational history & vector embeddings** in Pinecone
- **Use OpenAI's models** to generate personalized workout and meal suggestions
- Be deployed using **AWS Amplify Gen 2 (Free Tier)**

💻 Tech Stack Overview

- **Frontend**: Next.js (React), MUI (preferred), Server-Side Rendering (SSR) enabled for performance
- **Backend**: AWS Lambda, DynamoDB, API Gateway or AppSync (GraphQL or REST)
- **Authentication**: AWS Cognito
- **Data Storage**:
  - **DynamoDB**: For user profile and structured logs (workout/meal)
  - **Pinecone**: For storing and retrieving OpenAI-generated vector embeddings
- **AI Integration**:
  - **OpenAI API (text-embedding-ada-002 + chat)** for embeddings and suggestions

✅ Deliverables

1. User Profile Page

- Fields: Age, Height, Weight, Fitness Goals
- Editable form that updates DynamoDB
- Embedding (text summary or concatenated fields) stored in Pinecone under user namespace

2. Workout & Meal Logging

- Logging forms for workouts and meals
  - Fields: Type, Time, Calories, Notes
- Upon submission:

- Stored in DynamoDB
- Lambda creates an embedding of the notes/summary
- Embedding is upserted into Pinecone under namespace = user_id
- Metadata to include: { type: 'meal' | 'workout', date, calories, etc. }

3. Ask Coach (Chatbot)

- Input: Free-form user prompt (e.g., “Suggest a post-workout meal”)
- Lambda function pipeline:
  - Embed user prompt via OpenAI
  - Query Pinecone for top-k similar past logs + user profile vector
  - Retrieve structured data (e.g., goals, weight) from DynamoDB
  - Augment prompt with retrieved embeddings + structured context
  - Send final prompt to OpenAI Chat API
  - Return personalized recommendation in natural language

4. Frontend Pages

- **Profile**: View/edit user data
- **Log Entry**: Form to log meals/workouts
- **Chat Interface**: Ask Coach interaction UI with streamed or static response
- Ensure mobile-friendly responsive design

🧠 AI & Embedding Setup

- Use text-embedding-ada-002 to embed:
  - Profile (1 vector per user)
  - Each log entry (meal or workout) individually
- Pinecone vector index:
  - Namespace = user_id
  - Metadata = { type, date, calories, etc. }
- Retrieval-Augmented Generation (RAG) Pattern:
  a. Embed user prompt
  b. Query Pinecone for related logs + profile
  c. Compile context (structured + retrieved)
  d. Send to OpenAI Chat API
  e. Display grounded, personalized recommendation

🔒 Security & Validation

- Use AWS Cognito to:
  - Handle signup/login
  - Protect frontend routes and API endpoints
- Backend API (GraphQL or REST):
  - Secure with Cognito authorizers or IAM
  - Validate all inputs server-side
- Frontend:
  - Use form validation for fields (age, weight, etc.)
  - Prevent malformed or unsafe inputs

✅ Testing Requirements

- Write **2–3 unit tests** for React components (e.g., form validation, chat response rendering)
- Write **at least 1 backend test** for a Lambda function
  - Example: Test that a mocked OpenAI response is correctly formatted and returned

📦 Submission Checklist

Your GitHub repo should include:

- Full source code (frontend + backend)
- README.md with:
  - 🗺️ Architecture diagram (can be hand-drawn or created in a tool like Lucidchart, Whimsical, or Excalidraw)
  - 📐 Pinecone index design (schema, namespace strategy, metadata fields)
  - ⚙️ AWS Amplify Gen 2 configuration (categories, auth, mock data)
  - 🧪 Instructions to run locally (amplify mock, npm run dev) and deploy
- Loom video (≤10 minutes) walkthrough of app

⏰ Timeline

Deadline: Please submit within **7 days** of receiving this assignment.