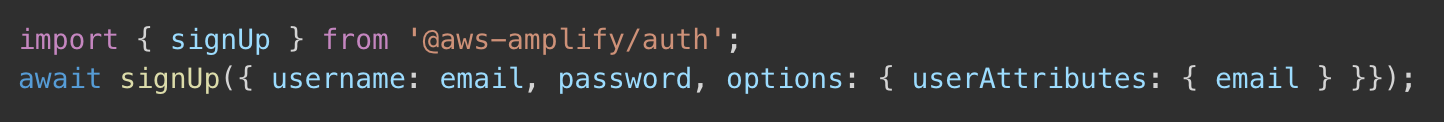
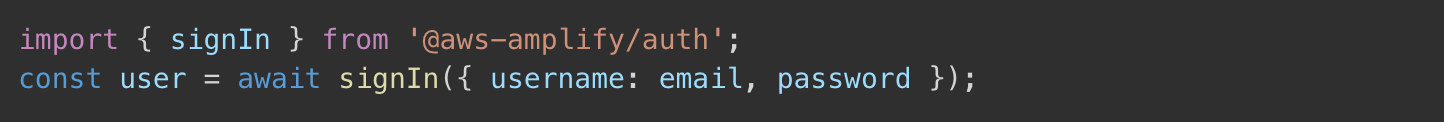
1. **Authentication APIs**
   1. AWS Cognito – Sign‑Up  
      **Purpose**  Create a new account and seed the user pool.  
      **Method**  POST (AWS Cognito SDK / Amplify Auth).  
      **Process**  Takes *email + password* → returns JWT + Cognito ID; the app stores that ID locally and in DynamoDB for cross‑reference.

**Key Code**



* 1. AWS Cognito – Login  
     **Purpose**  Authenticate existing users.  
     **Method**  POST (AWS Cognito SDK / Amplify Auth).  
     **Process**  Email + password → Cognito verifies → returns fresh JWT tokens (ID, Access, Refresh).

**Key Code** 

1. **AWS Amplify GraphQL**

|  |  |  |
| --- | --- | --- |
| Endpoint | Purpose | Key Code |
| CreateOnboardingData | Store first‑time quiz results ( equipment, fitness type) | await client.models.OnboardingData.create({...}) |
| UpdateOnboardingData | Let returning users tweak their quiz answers | await client.models.OnboardingData.update({...}) |
| ListOnboardingData | Retrieve the logged‑in user’s record | await client.models.OnboardingData.list({ filters:{ userID:{ eq:userID }}}) |

**How it works:** AWS Amplify automatically generates GraphQL API endpoints to interact with DynamoDB.

1. **ExerciseDB via RapidAPI**

Fetch Exercises by Target Muscle  
**Purpose**  Pull a big pool of candidate moves.  
**Method**  GET /exercises/target/{muscle} (RapidAPI key in headers).

**Key Code** A black screen with text

AI-generated content may be incorrect.

1. **Nutrition Recommendation APIs (Spooncular)**

Fetch Meal Suggestions  
 **Purpose**  Serve balanced meals that align with the user’s calorie/macronutrient targets.  
 **Method**  GET /mealplanner/generate (API Key as query param).  
 **Process**  The app sends *targetCalories* & *diet prefs* → receives a JSON list of recipes.

*Implementation mirrors the ExerciseDB flow—wrap in fetch(), parse JSON, then store in app state.*