420-511-VA WEB SERVICES

WINTER 2026

TEAM PROJECT PROPOSAL

Adaptive Recipe & Fitness Service

A REST Web API for Intelligent Recipe Management and Workout Planning

Said Omar Becerra Chira,

Talia Muro

& Mariam Salim

VANIER COLLEGE

Teacher: Sleiman Rabah

February 11, 2026

Executive Summary

The Adaptive Recipe & Fitness Service is a REST-based web API that revolutionizes how users interact with recipes and fitness routines. Unlike traditional recipe services that simply display instructions, our service provides intelligent features including ingredient-based recipe search, automatic ingredient substitutions, measurement conversions, cost estimation, recipe scaling, and dietary restriction management.

The service addresses real-world problems: food waste from unused ingredients, inability to follow recipes due to missing ingredients, confusion over measurement units, and difficulty adapting recipes to dietary needs. By integrating with external APIs for nutritional data, pricing information, and workout demonstrations, the service creates a comprehensive cooking and fitness system.

Table of Contents

Tables

Table 1: Recipes

Fields: id (PK), name, description, author, culture, prep_time, cook_time, servings, difficulty_level, cuisine_type

Table 2: Ingredients

Fields: id (PK), name, category, avg_price_per_unit, price_currency, standard_unit, allergen_info (JSON), nutritional_info (JSON), is_perishable, shelf_life_days

Table 3: Recipe_Ingredients (Junction)

Fields: id (PK), recipe_id (FK), ingredient_id (FK), quantity, unit, is_optional, preparation_note, display_order

Table 4: Ingredient_Substitutions

Fields: id (PK), original_ingredient_id (FK), substitute_ingredient_id (FK), conversion_ratio, notes, substitution_type, confidence_score

Table 5: Unit_Conversions

Fields: id (PK), from_unit, to_unit, conversion_factor, ingredient_type, is_volume, is_weight, notes

Table 6: Dietary_Restrictions

Fields: id (PK), name, description, severity_level (preference/allergy/religious/ethical/medical)

Table 7: Recipe_Dietary_Compatibility

Fields: id (PK), recipe_id (FK), restriction_id (FK), is_compatible, notes

Table 8: Workouts

Fields: id (PK), name, description, intensity, duration, calories_burned_estimate, workout_area, equipment_needed (JSON)

Table 9: Exercises

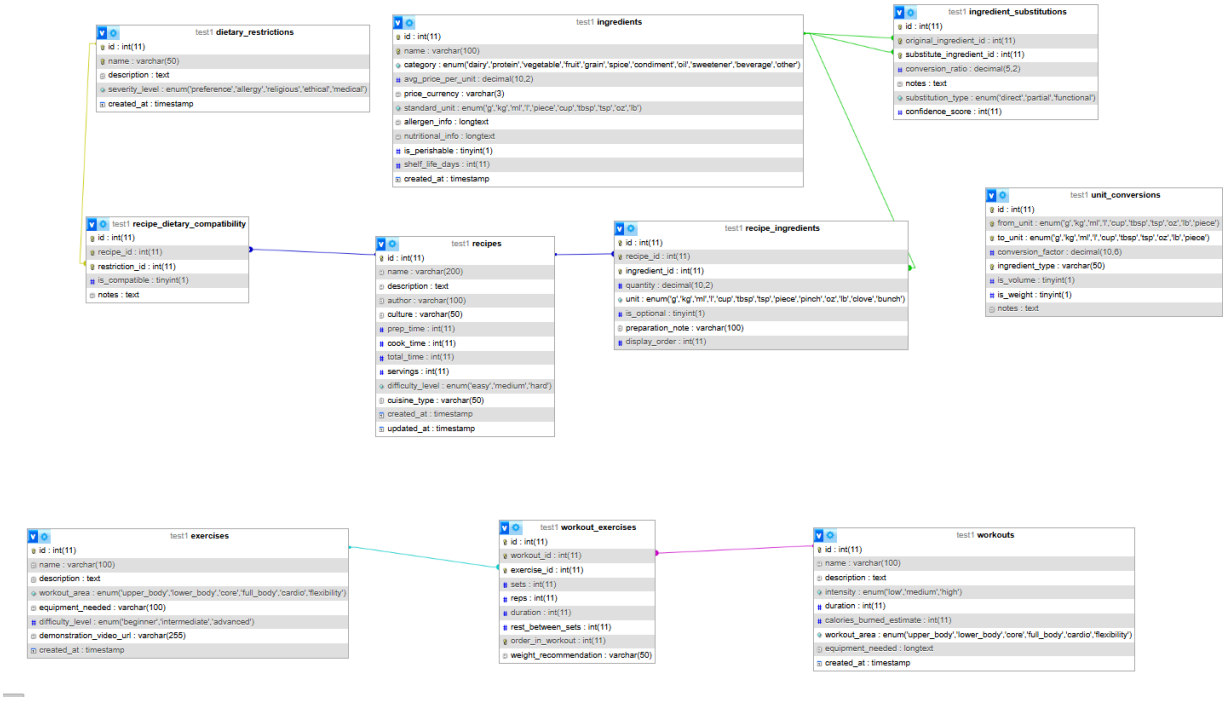
Fields: id (PK), name, description, workout_area, equipment_needed, difficulty_level, demonstration_video_url

Table 10: Workout_Exercises (Junction)

Fields: id (PK), workout_id (FK), exercise_id (FK), sets, reps, duration, rest_between_sets, order_in_workout, weight_recommendation

Entity-Relationship Diagram

See attached ER diagram (schema_with_relationships.png) showing all 10 tables with their fields, primary keys (PK), foreign keys (FK), and relationship connections. The diagram illustrates one-to-many relationships and junction tables for many-to-many associations.



Parent collection resources

Team Member	Table Name	Collection Resource URI
Talia	**Recipe**	**/recipes**
Mariam	**Ingredient**	**/ingredients**
Said	**Exercise**	**/exercises**

Sub-collections

Relationship	Sub-Collection URI
Recipes ↔ Ingredients	**/recipes/{recipe_id}/ingredients**
Workouts ↔ Exercises	**/workouts/{workout_id}/exercises**
Ingredients ↔ Substitutions	**/ingredients/{ingredient_id}/substitutions**

Filter Tables

Talia's collection: /recipes

#	Field Name	Data Type	Filter Parameter	Example Query
1	**name**	**VARCHAR**	**?name=Classic%20Spaghetti%20Carbonara**	**GET /recipes?name=Classic%20Spaghetti%20Carbonara**
2	**author**	**VARCHAR**	**?author=Chef%20Antonio%20Rossi**	**GET /recipes?author=Chef%20Antonio%20Rossi**
3	**total_time**	**INT**	**?total_time=25**	**GET /recipes?total_time=25**

4	**difficulty**	**ENUM**	**?difficulty=easy**	**GET /recipes?difficulty=easy**
5	**servings**	**INT**	**?servings=4**	**GET /recipes?servings=4**

Mariam's collection: /ingredients

#	Field Name	Data Type	Filter Parameters	Example Query
1	**Name**	**Varchar**	**?name=eggs**	**GET /ingredients?name=eggs**
2	**Category**	**Enum**	**?category=meat**	**GET /ingredients?category=meat**
3	**Is_perishable**	**Boolean**	**?is_perishable=true**	**GET /ingredients?is_perishable=true**
4	**Avg_price_per_unit**	**Decimal (10,2)**	**?avg_price_per_unit=20.00**	**GET /ingredients?avg_price_per_unit=20.00**
5	**allergen**	**varchar**	**?allergen=nuts**	**GET /ingredients?allergen=nuts**

Said's Collection: /exercises

#	Field Name	Data Type	Filter Parameters	Example Query

1	**name**	**VARCHAR**	**?name=push-up**	**GET /exercises?name=push-up**
2	**workout_area**	**VARCHAR**	**?workout_area=chest**	**GET /exercises?workout_area=chest**
3	**difficulty**	**ENUM**	**?difficulty=beginner**	**GET /exercises?difficulty=beginner**
4	**equipment**	**VARCHAR**	**?equipment=dumbbells**	**GET /exercises?equipment=dumbbells**
5	**description**	**TEXT**	**?description_contains=strenght**	**GET /exercises?description_contains=strenght**

Filter Types by Data Type

Data Type	Filter Parameter	Example Query
String	**Exact match, contains, startsWith, endsWith**	**?name=Apollo, ?name_contains=Mars**
Number	**Equals, min, max, range**	**?servings=4, ?minServings=2&maxServings=4**
Boolean	**Equals**	**?isOptional=true**
Enum	**Equals, in (multiple values)**	**?difficulty=easy**

Sorting Parameters

Talia's Collection: /recipes

Sortable Field	Sort Parameter Example
name	**?sort=name or ?sort=-name (desc)**
difficulty	**?sort=difficulty (easiest first)**
servings	**?sort=servings (smallest first)**

Mariam's Collection: /ingredients

Sortable Field	Sort Parameter Example
name	**?sort=name or ?sort=-name (desc)**
avg_price_per_unit	**?sort=avg_price_per_unit**
category	**?sort=category or ?sort=-categort (desc)**

Said's Collection: /exercises

Sortable Field	Sort Parameter Example
name	**?sort=name or ?sort=-name (desc)**
difficulty	**?sort=difficulty (beginner first)**
workout_area	**?sort=workout_area(alphabetical)**

Pagination Parameters

Parameter	Description	Example
page	**Page number (starts at 1)**	**?page=2**
limit	**Items per page**	**?limit=20**
offset	**Skip n items (alternative)**	**?offset=20**

HTTP Status Codes

Successful Status Codes (2xx)

Status Code	Name	When to Use
200	**OK**	**Successful GET, PUT, PATCH requests that return data**
201	**Created**	**Successful POST request that creates a new resource**
204	**No Content**	**Successful DELETE request (no response body)**

Client Error Status Codes (4xx)

Status Code	Name	When to Use
400	Bad Request	**Invalid request syntax, malformed JSON, validation errors**

401	**Unauthorized**	**Missing or invalid authentication credentials**
403	**Forbidden**	**Valid credentials but insufficient permissions**
404	**Not Found**	**Resource does not exist at the given URI**
405	**Method Not Allowed**	**HTTP method not supported for this endpoint**
409	**Conflict**	**Request conflicts with current state (e.g., duplicate entry)**
422	**Unprocessable Entity**	**Valid syntax but semantic errors (e.g., invalid field values)**

Server Error Status Codes (5xx)

Status Code	Name	When to Use
500	**Internal Server Error**	**Unexpected server-side error**
503	**Service Unavailable**	**Server temporarily unavailable (maintenance, overload)**