RecipeApp

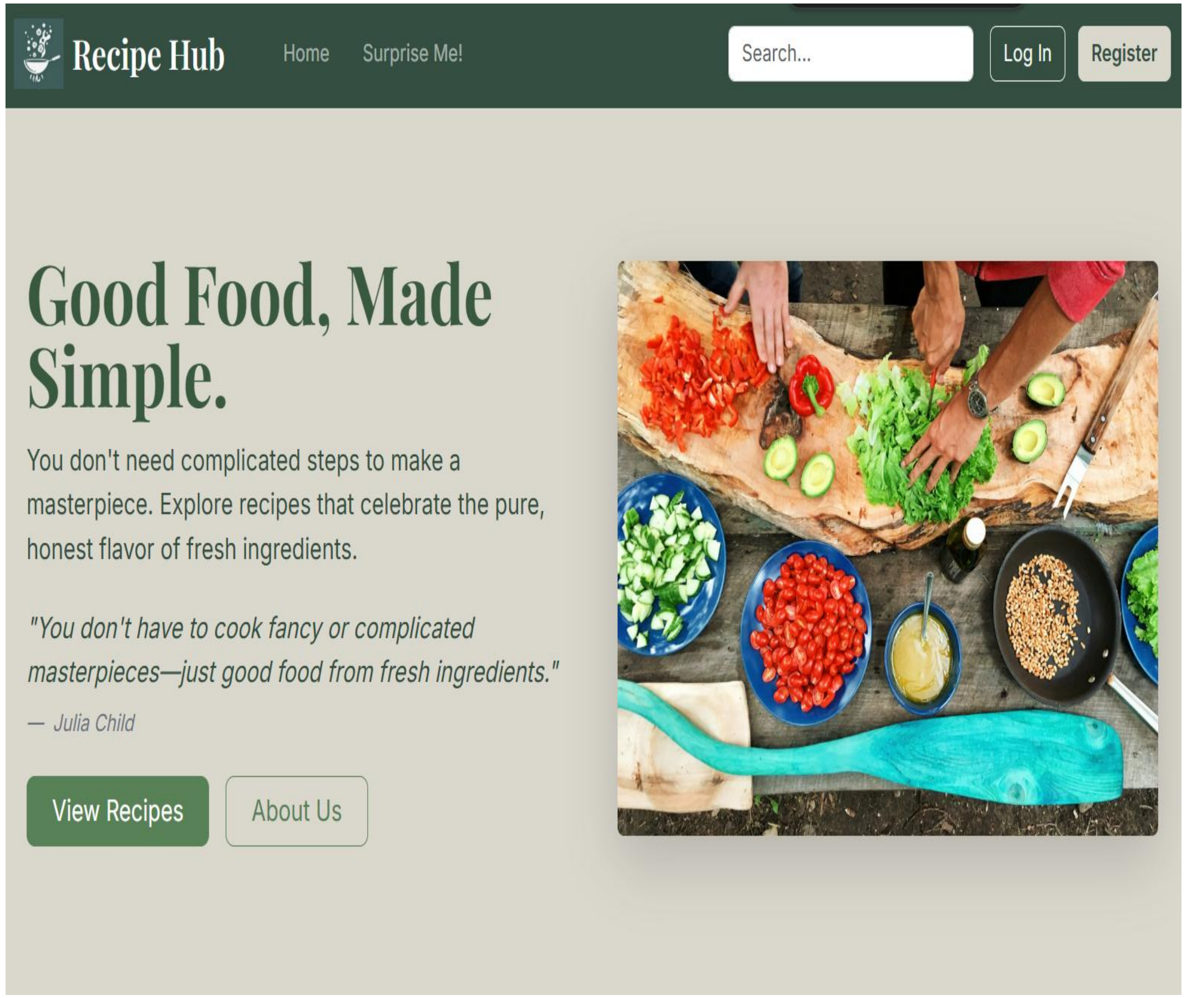
A Modern Recipe Management Solution

---By Minh Triet Vu and Peng Yang ---

John Abbott College FSD-14

2025

Background & Overview


Recipe Hub
Home
Surprise Me!
Search...
Log In
Register

Good Food, Made Simple.

You don't need complicated steps to make a masterpiece. Explore recipes that celebrate the pure, honest flavor of fresh ingredients.

"You don't have to cook fancy or complicated masterpieces—just good food from fresh ingredients."

— Julia Child

View Recipes
About Us

Context:

Cooking enthusiasts often struggle to find, organize, and store recipes from multiple sources.
Our project aims to centralize recipe management, allowing users to browse, import, and manage meals efficiently.

Problem Solved:

Provides a unified system for recipe storage, search, and categorization, solving data fragmentation and limited API use in existing apps.

Solution Overview

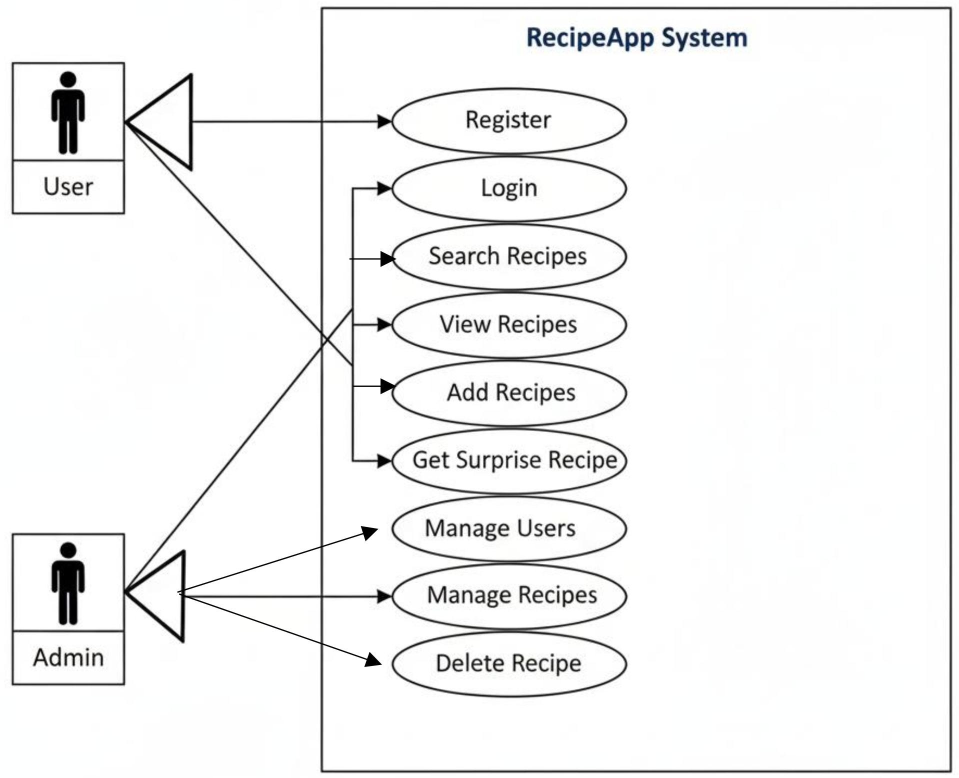
Our Web Application Includes:

- User Registration & Login (Spring Boot + MySQL)
- Admin Dashboard for managing users and recipes
- Recipe import via external API (TheMealDB)
- Category & Cuisine filtering
- Bootstrap-based responsive UI

User Perspective:

- Simple UI for exploring and importing meals
- Role-based access control
- Browse a dynamic list of recipes with category and cuisine filters
- View detailed recipe pages with instructions and nutritional information

RecipeApp Use Case Diagram (Implemeted Features)



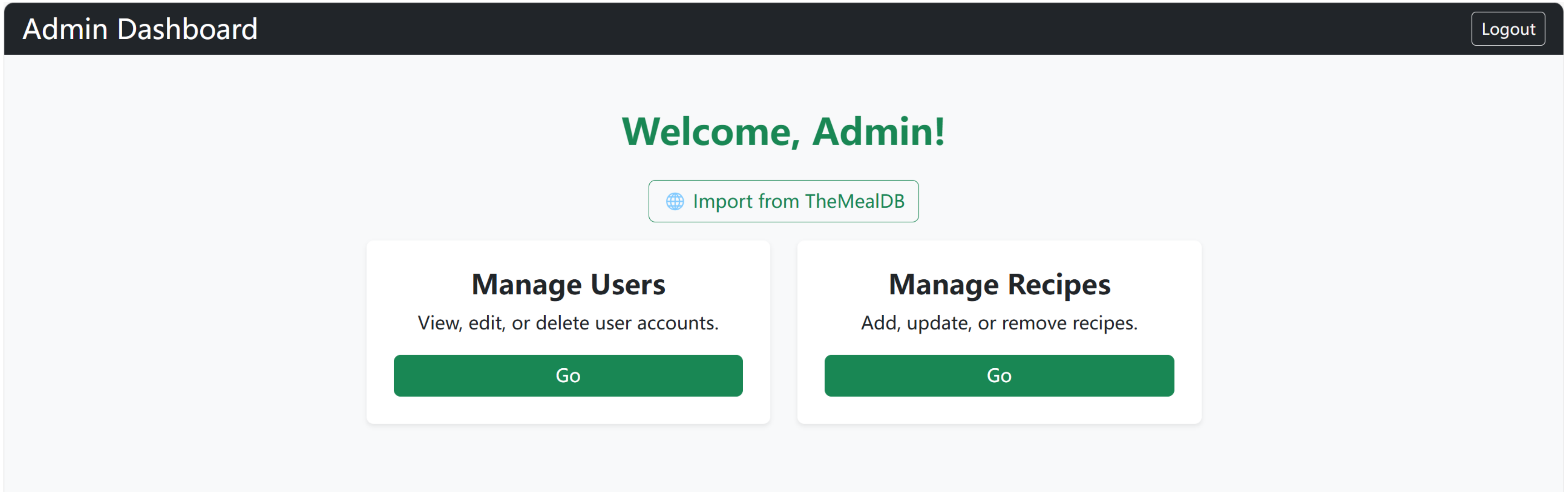
Features (User & Admin)

User Features:

- View and search recipes
- Manage personal recipes
- Explore imported meals

Admin Features:

- Manage users (promote, delete)
- Import recipes from TheMealDB API
- Edit or delete existing recipes


Admin Dashboard

Welcome, Admin!

Manage Users

View, edit, or delete user accounts.

Manage Recipes

Add, update, or remove recipes.

Database Design Overview

Database: MySQL (Relational)

Main Entities:

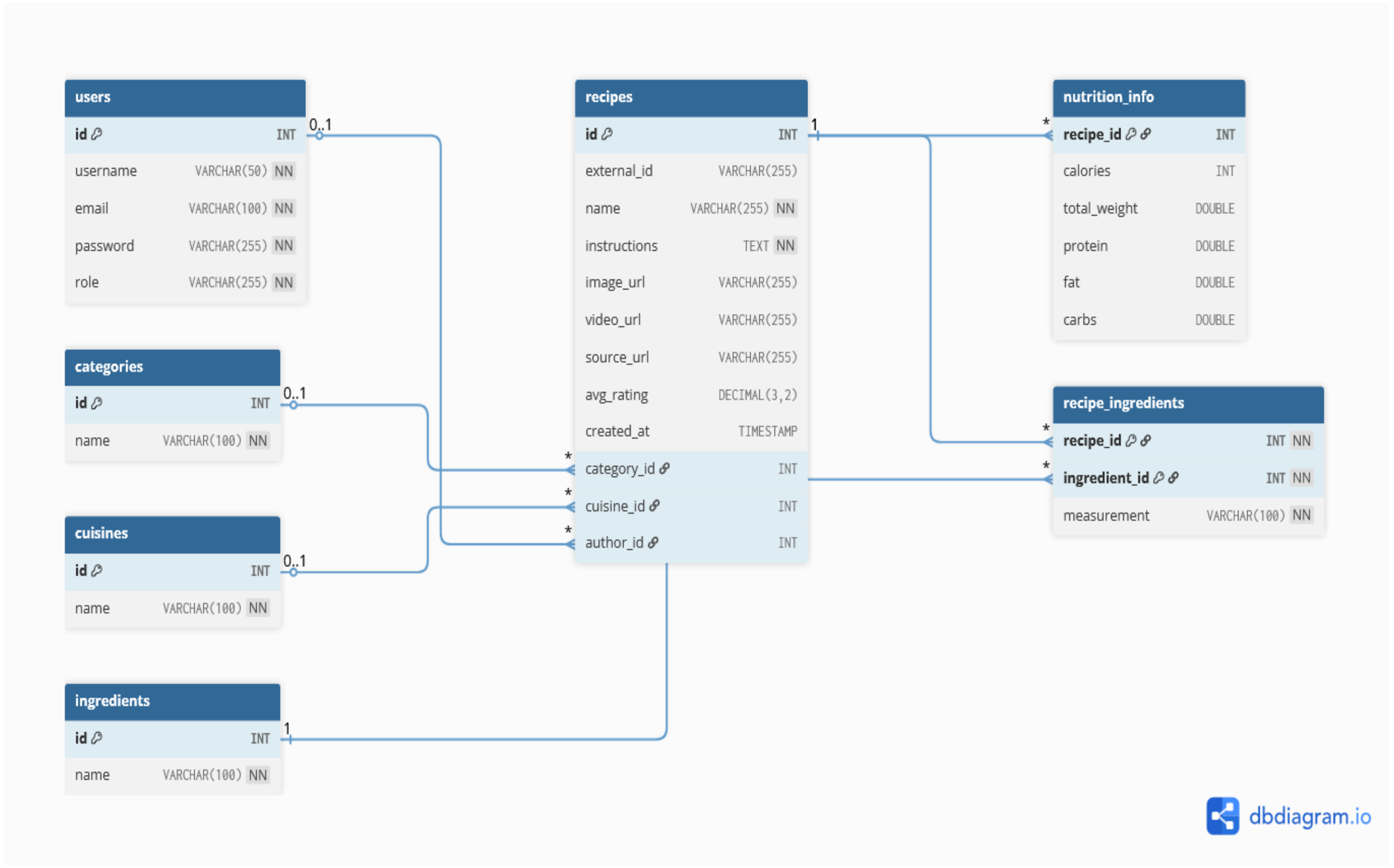
- Users — authentication and roles

- Recipes — main content table

- Categories & Cuisines — classification

- Ingredients — detailed recipe info

- Recipe_Ingredients — bridge for many-to-many link



users		
id	INT	
username	VARCHAR(50)	NN
email	VARCHAR(100)	NN
password	VARCHAR(255)	NN
role	VARCHAR(255)	NN

categories		
id	INT	
name	VARCHAR(100)	NN

cuisines		
id	INT	
name	VARCHAR(100)	NN

ingredients		
id	INT	
name	VARCHAR(100)	NN

recipes		
id	INT	
external_id	VARCHAR(255)	
name	VARCHAR(255)	NN
instructions	TEXT	NN
image_url	VARCHAR(255)	
video_url	VARCHAR(255)	
source_url	VARCHAR(255)	
avg_rating	DECIMAL(3,2)	
created_at	TIMESTAMP	
category_id	INT	
cuisine_id	INT	
author_id	INT	

nutrition_info	
recipe_id	INT
calories	INT
total_weight	DOUBLE
protein	DOUBLE
fat	DOUBLE
carbs	DOUBLE

recipe_ingredients		
recipe_id	INT	NN
ingredient_id	INT	NN
measurement	VARCHAR(100)	NN

Database Relationships

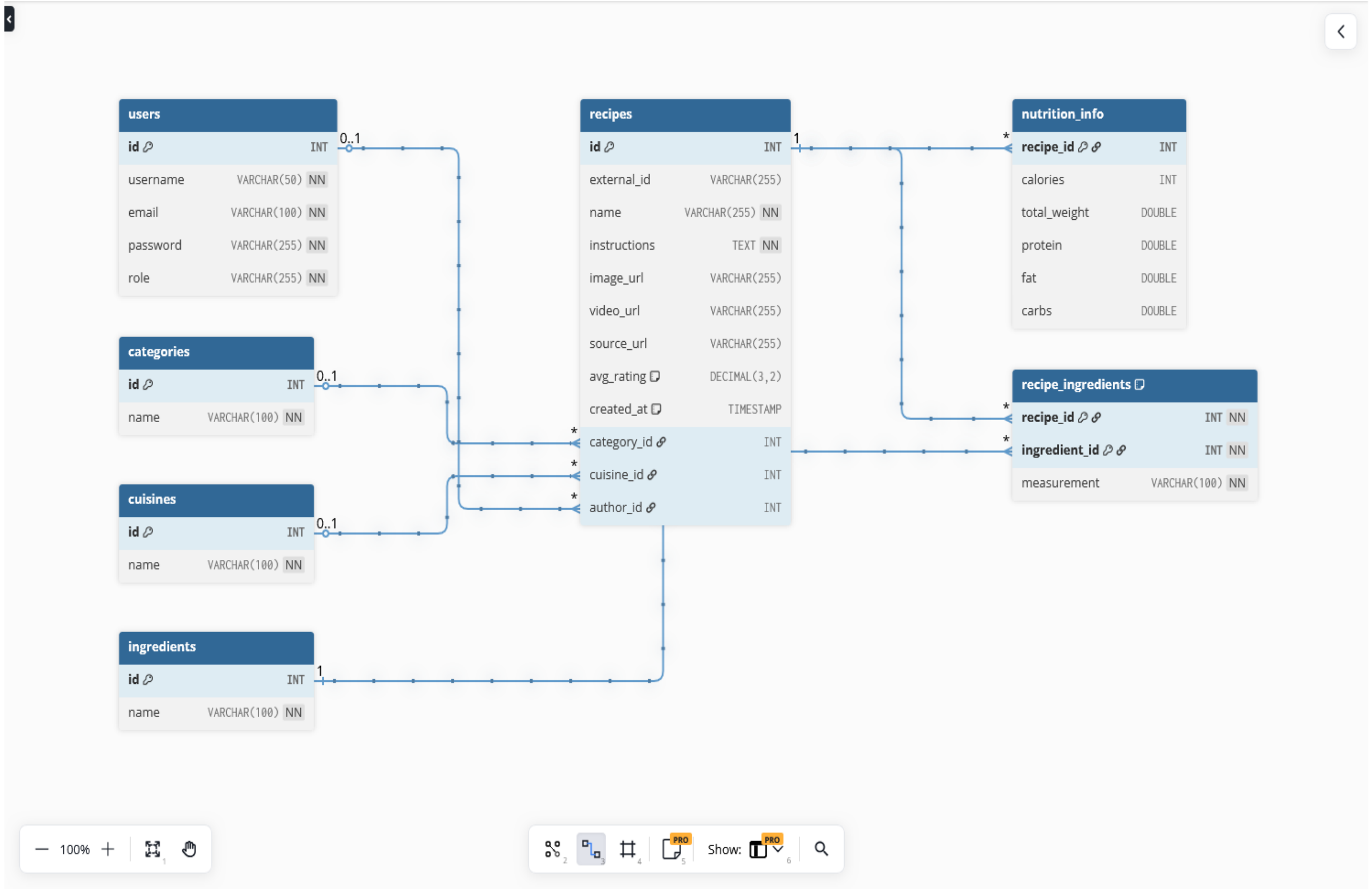
Key Relationships:

- One User → Many Recipes
- One Category → Many Recipes
- One Cuisine → Many Recipes
- Many Ingredients ↔ Many Recipes

(through Recipe_Ingredients)

Data Integrity:

- Primary/foreign keys with cascading deletes



users

id	INT	
username	VARCHAR(50)	NN
email	VARCHAR(100)	NN
password	VARCHAR(255)	NN
role	VARCHAR(255)	NN

categories

id	INT	
name	VARCHAR(100)	NN

cuisines

id	INT	
name	VARCHAR(100)	NN

ingredients

id	INT	
name	VARCHAR(100)	NN

recipes

id	INT	
external_id	VARCHAR(255)	
name	VARCHAR(255)	NN
instructions	TEXT	NN
image_url	VARCHAR(255)	
video_url	VARCHAR(255)	
source_url	VARCHAR(255)	
avg_rating	DECIMAL(3,2)	
created_at	TIMESTAMP	
category_id	INT	
cuisine_id	INT	
author_id	INT	

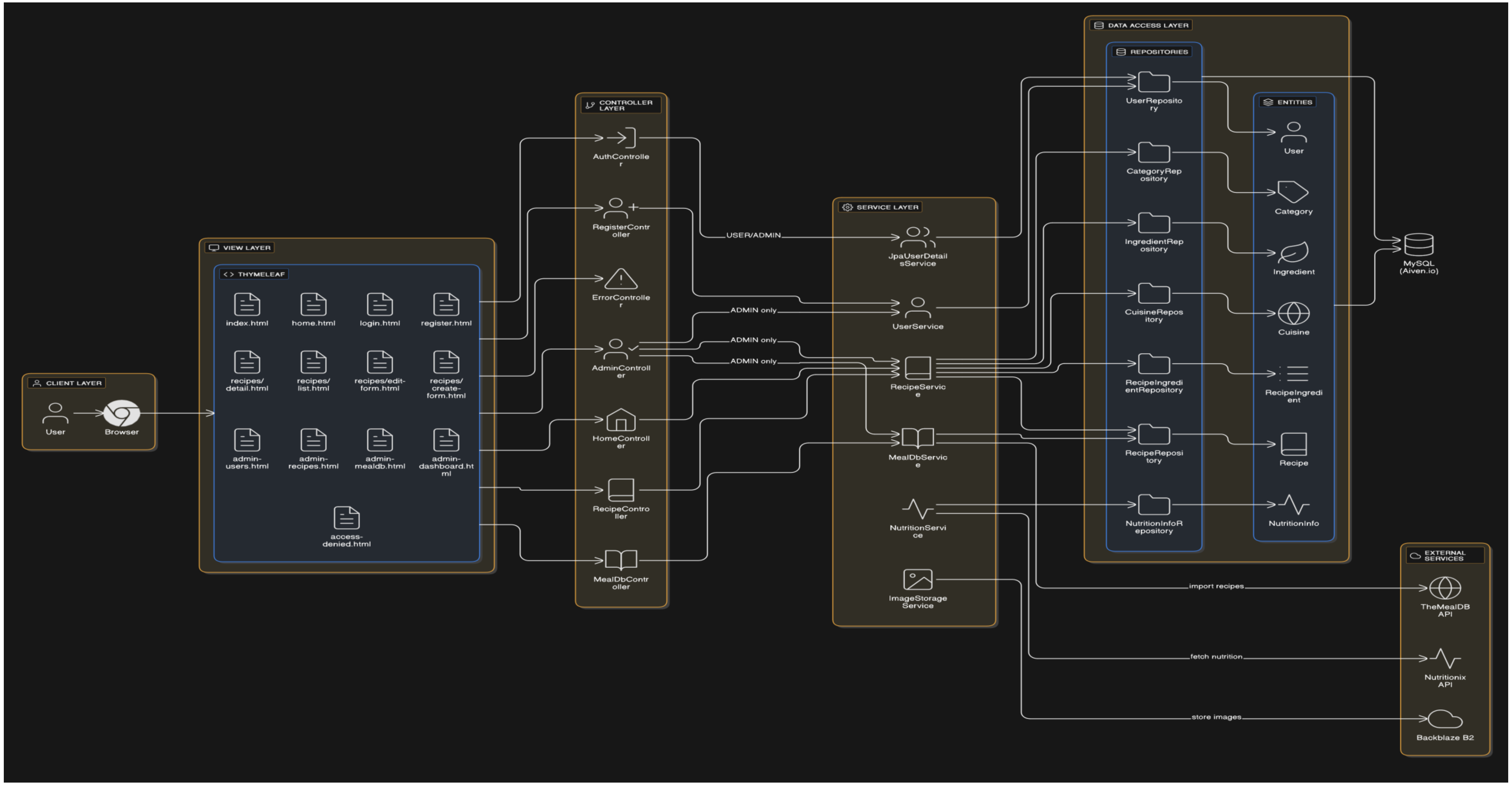
nutrition_info

recipe_id	INT
calories	INT
total_weight	DOUBLE
protein	DOUBLE
fat	DOUBLE
carbs	DOUBLE

recipe_ingredients

recipe_id	INT	NN
ingredient_id	INT	NN
measurement	VARCHAR(100)	NN

System Architecture



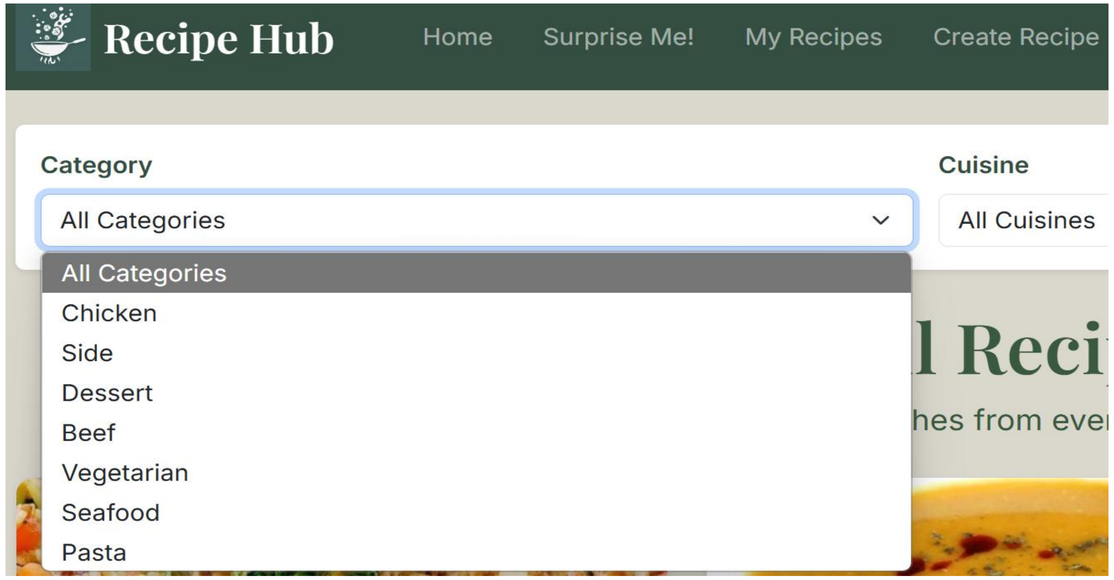
TheMealDB API Integration

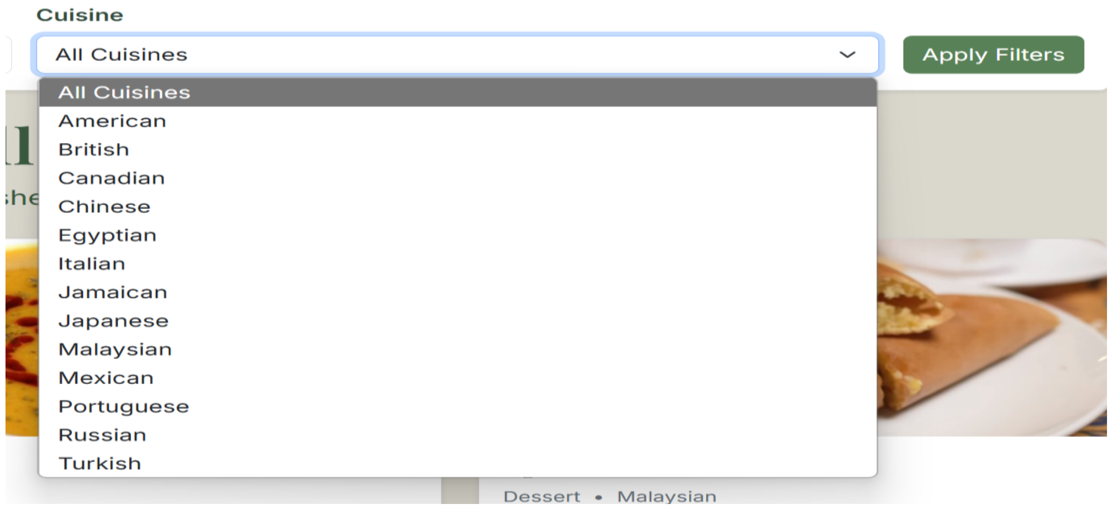
Goal: Let Admins import recipes directly from TheMealDB.

Endpoints Used:
- /list.php?c=list → Categories
- /list.php?a=list → Areas
- /filter.php?c=Seafood → Meals by category
- /lookup.php?i=52772 → Meal details by ID

Tech Used: Spring WebClient (non-blocking)

```
String response = webClient.get()
    .uri("/filter.php?c=" + category)
    .retrieve()
    .bodyToMono(String.class)
    .block();
```



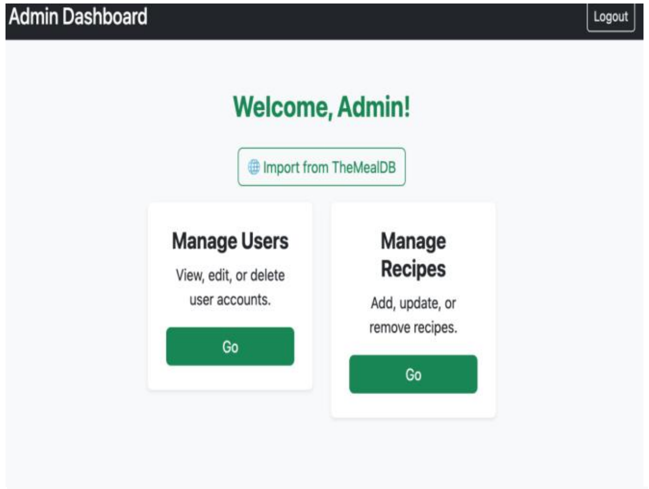


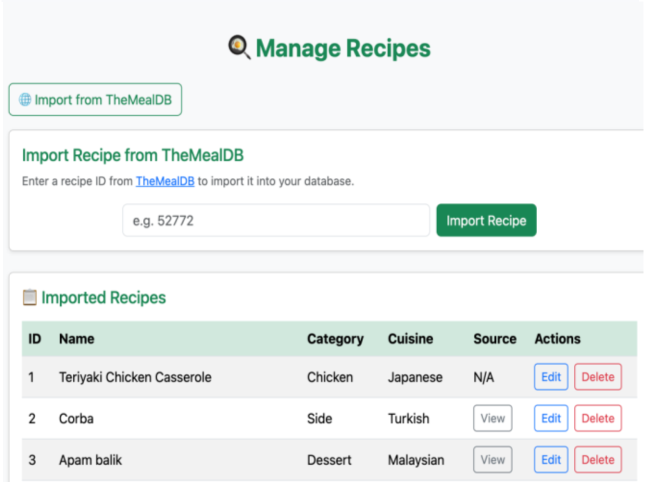
Admin Dashboard & Import Workflow

The admin home page provides three main actions — Manage Users, Manage Recipes, and Import Recipes. 2. App fetches meals from TheMealDB.

• Manage Users: Promote, demote, or delete users easily.
• Manage Recipes: Edit, view, or delete imported recipes.
• Recipes can also be imported by entering an ID or browsing from TheMealDB API. 4. Clicks "Import" → Recipe stored in local DB.

Admins can choose a Category or Area and load meals directly from TheMealDB — a one-click import workflow.





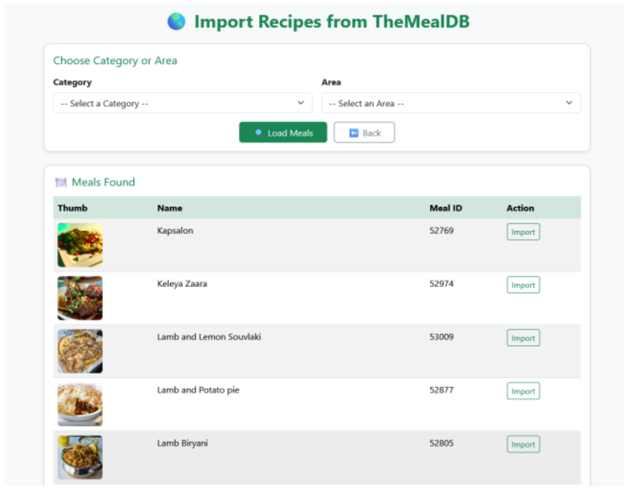
Import Recipe from TheMealDB

Enter a recipe ID from TheMealDB to import it into your database.

e.g. 52772

Imported Recipes

ID	Name	Category	Cuisine	Source	Actions
1	Teriyaki Chicken Casserole	Chicken	Japanese	N/A	Edit Delete
2	Corba	Side	Turkish	View	Edit Delete
3	Apam balik	Dessert	Malaysian	View	Edit Delete



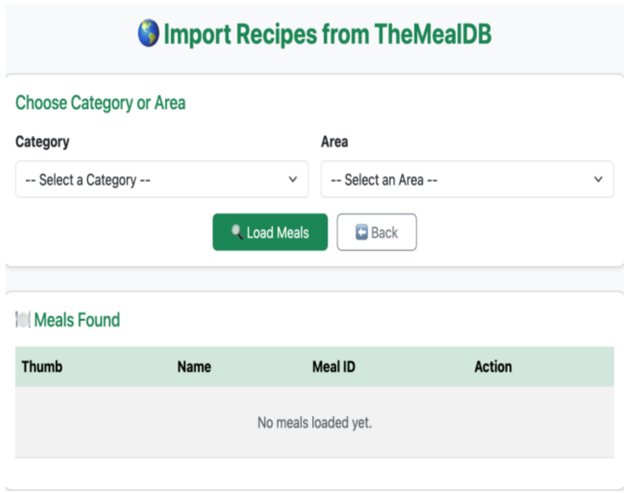
Choose Category or Area

Category: -- Select a Category --

Area: -- Select an Area --

Meals Found

Thumb	Name	Meal ID	Action
	Kapsalon	52769	Import
	Keleya Zaara	52974	Import
	Lamb and Lemon Souvlaki	53009	Import
	Lamb and Potato pie	52877	Import
	Lamb Biryani	52805	Import



Choose Category or Area

Category: -- Select a Category --

Area: -- Select an Area --

Meals Found

Thumb	Name	Meal ID	Action
No meals loaded yet.			

Challenges & Solutions

Challenge	Solution
Dynamic Ingredient List	Use javascript to add and remove input fields, clear and re-add strategy in RecipeService
Duplicate imports	Checked via existsByExternalId()
Deleting Cloud Storage Images	Added logic to the RecipeService that, upon recipe deletion or image update, calls the ImageStorageService to explicitly delete the old file from the B2 bucket.
Admin deletion protection	Prevented root admin removal
Alerts disappearing too fast	Use boostrap warning class
Integrating S3-Compatible Storage	Configured the AWS S3 client with an endpointOverride in our B2StorageConfig to point to the Backblaze URL.

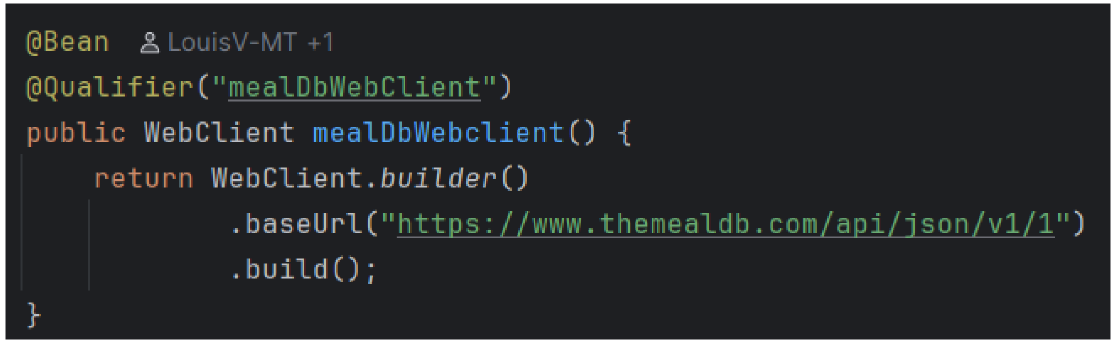
Version Control & Teamwork

Collaboration Tools:

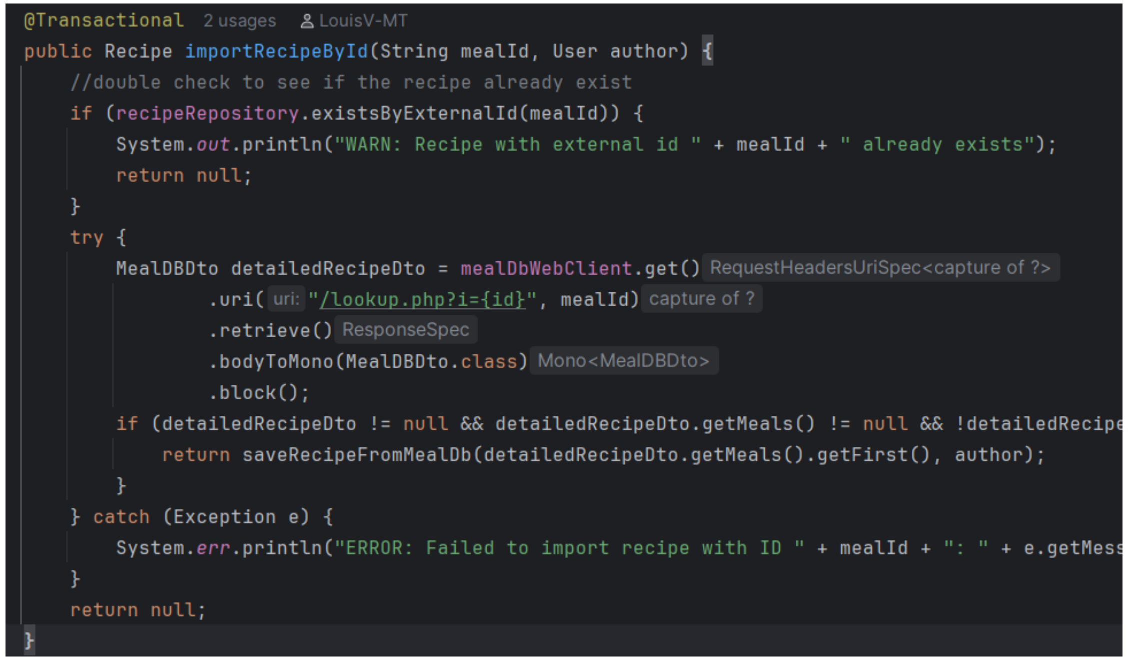
- Git & GitHub for code versioning
- Branch workflow (feature/mealdb-import-ui, feature/register)
- GitHub Pull Requests for review
- Microsoft Teams for discussion
- Trello for task management

What We Learned (API Integration)

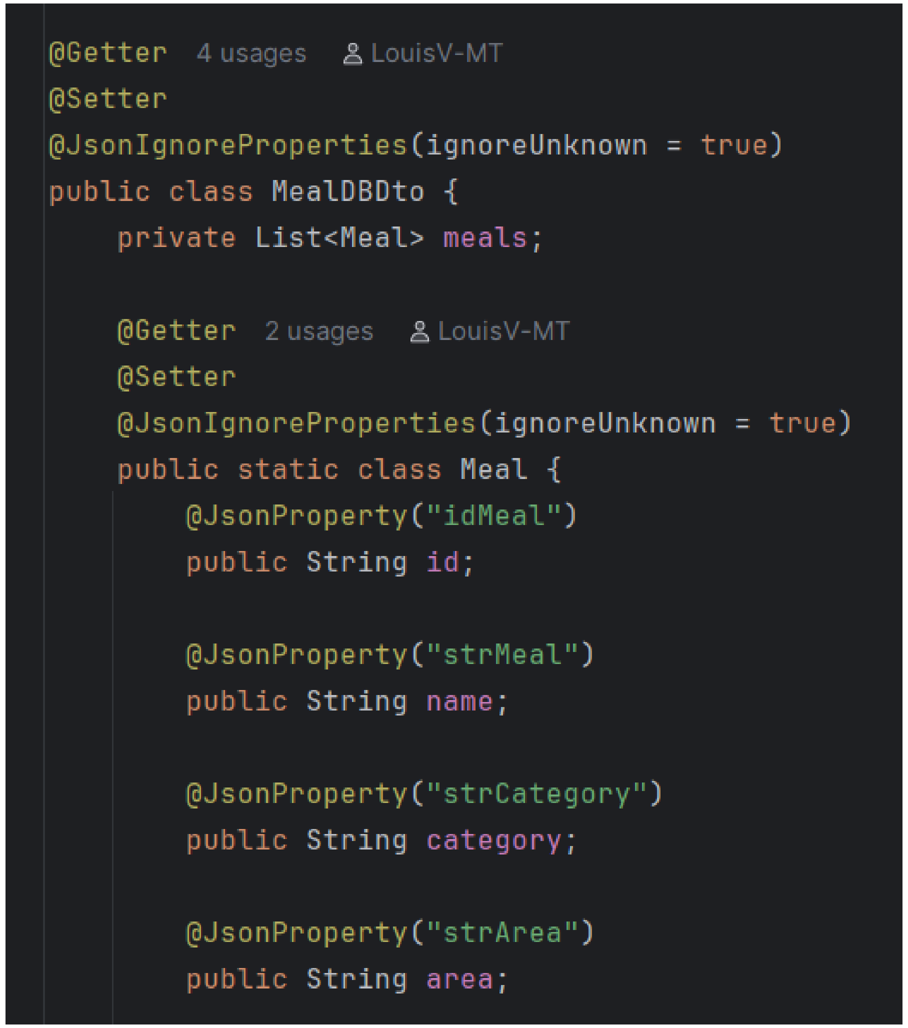
Map JSON to a Java Object with WebClient



```java
@Bean  LouisV-MT +1
@Qualifier("mealDbWebClient")
public WebClient mealDbWebclient() {
    return WebClient.builder()
            .baseUrl("https://www.themealdb.com/api/json/v1/1")
            .build();
}
```



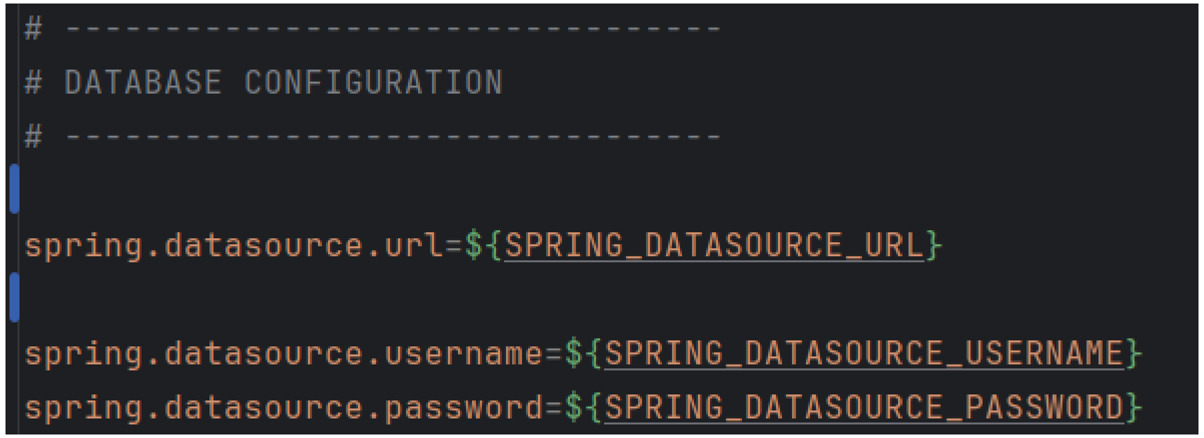
```java
@Transactional  2 usages  LouisV-MT
public Recipe importRecipeById(String mealId, User author) {
    //double check to see if the recipe already exist
    if (recipeRepository.existsByExternalId(mealId)) {
        System.out.println("WARN: Recipe with external id " + mealId + " already exists");
        return null;
    }
    try {
        MealDBDto detailedRecipeDto = mealDbWebClient.get() RequestHeadersUriSpec<capture of ?>
                .uri( uri: "/lookup.php?i={id}", mealId) capture of ?
                .retrieve() ResponseSpec
                .bodyToMono(MealDBDto.class) Mono<MealDBDto>
                .block();
        if (detailedRecipeDto != null && detailedRecipeDto.getMeals() != null && !detailedRecipe
            return saveRecipeFromMealDb(detailedRecipeDto.getMeals().getFirst(), author);
        }
    } catch (Exception e) {
        System.err.println("ERROR: Failed to import recipe with ID " + mealId + ": " + e.getMess
    }
    return null;
}
```



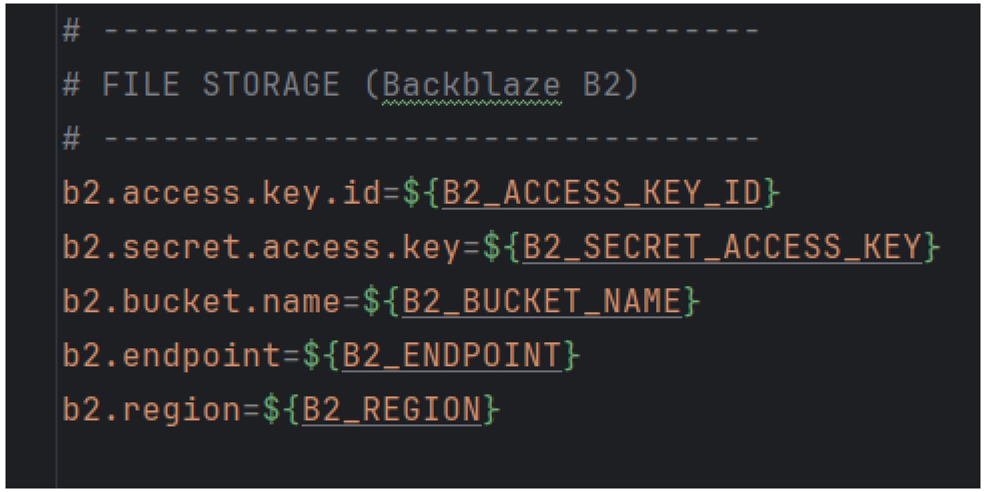
```java
@Getter  4 usages  LouisV-MT
@Setter
@JsonIgnoreProperties(ignoreUnknown = true)
public class MealDBDto {
    private List<Meal> meals;

    @Getter  2 usages  LouisV-MT
    @Setter
    @JsonIgnoreProperties(ignoreUnknown = true)
    public static class Meal {
        @JsonProperty("idMeal")
        public String id;

        @JsonProperty("strMeal")
        public String name;

        @JsonProperty("strCategory")
        public String category;

        @JsonProperty("strArea")
        public String area;
```

What We Learned (Working with deployment)

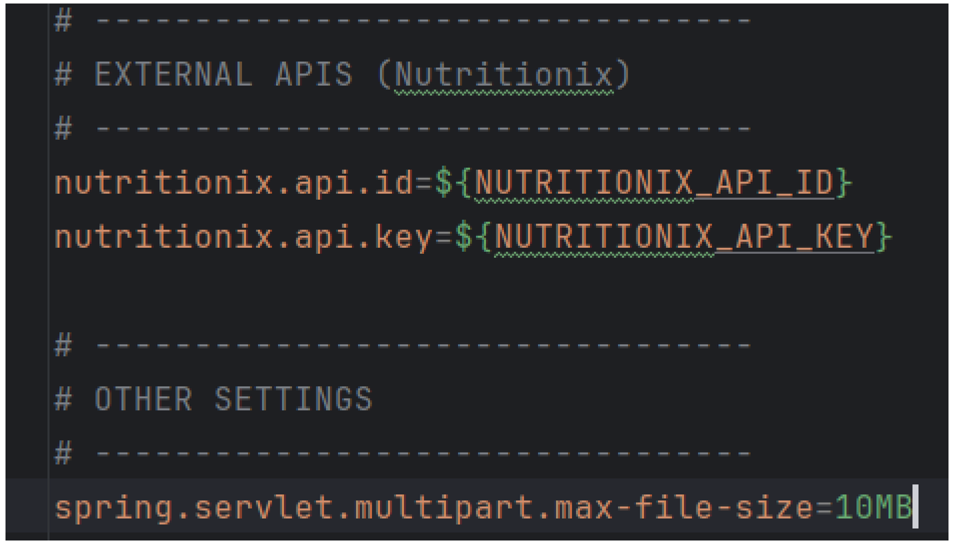
Connect Spring Boot App to external services



```
# ---------------------------------
# DATABASE CONFIGURATION
# ---------------------------------

spring.datasource.url=${SPRING_DATASOURCE_URL}

spring.datasource.username=${SPRING_DATASOURCE_USERNAME}
spring.datasource.password=${SPRING_DATASOURCE_PASSWORD}
```



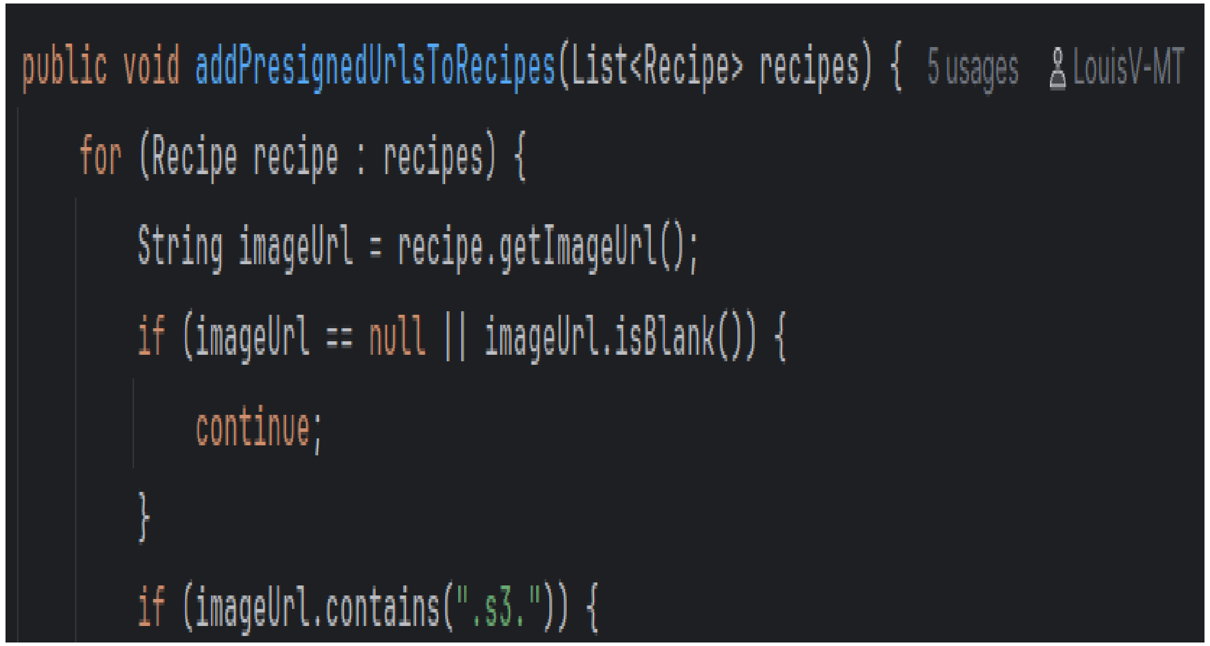
```
# ---------------------------------
# FILE STORAGE (Backblaze B2)
# ---------------------------------
b2.access.key.id=${B2_ACCESS_KEY_ID}
b2.secret.access.key=${B2_SECRET_ACCESS_KEY}
b2.bucket.name=${B2_BUCKET_NAME}
b2.endpoint=${B2_ENDPOINT}
b2.region=${B2_REGION}
```



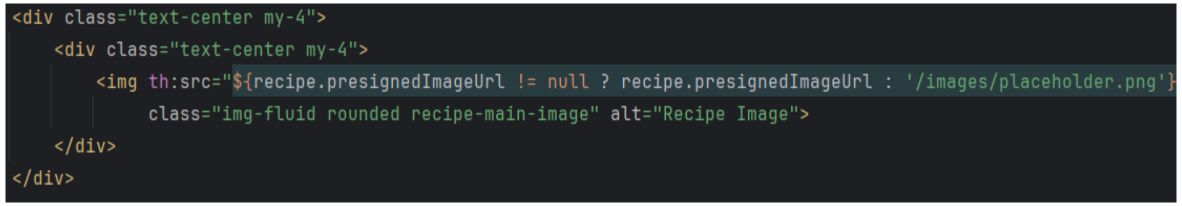
```
# ---------------------------------
# EXTERNAL APIS (Nutritionix)
# ---------------------------------
nutritionix.api.id=${NUTRITIONIX_API_ID}
nutritionix.api.key=${NUTRITIONIX_API_KEY}

# ---------------------------------
# OTHER SETTINGS
# ---------------------------------
spring.servlet.multipart.max-file-size=10MB
```

What We Learned (Working with deployment)

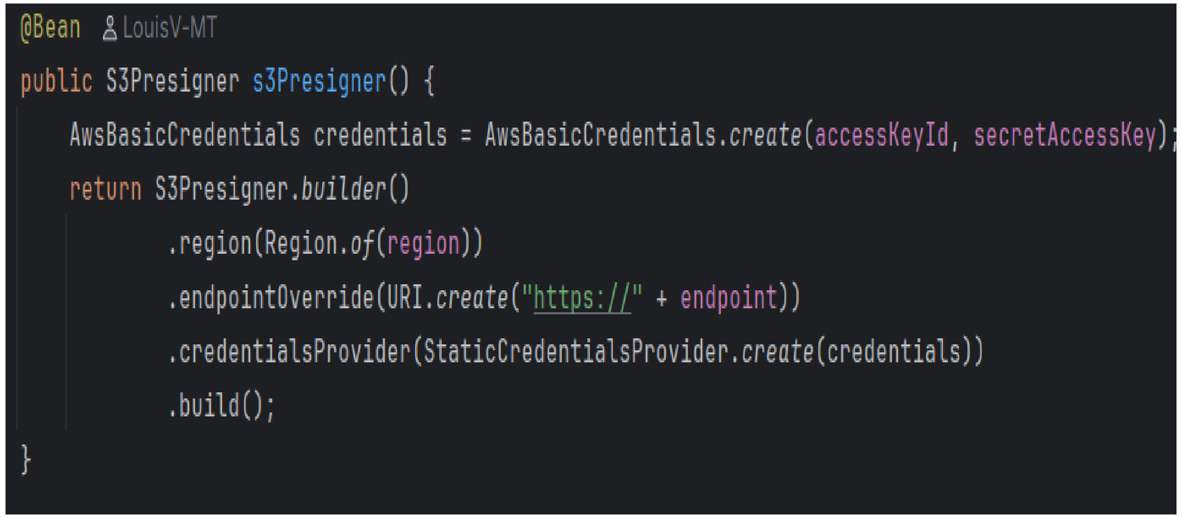
Presigned link with s3 bucket



```java
public void addPresignedUrlsToRecipes(List<Recipe> recipes) {  5 usages  LouisV-MT
    for (Recipe recipe : recipes) {
        String imageUrl = recipe.getImageUrl();
        if (imageUrl == null || imageUrl.isBlank()) {
            continue;
        }
        if (imageUrl.contains(".s3.")) {
```



```html
<div class="text-center my-4">
    <div class="text-center my-4">
        <img th:src="${recipe.presignedImageUrl != null ? recipe.presignedImageUrl : '/images/placeholder.png'}
             class="img-fluid rounded recipe-main-image" alt="Recipe Image">
    </div>
</div>
```



```java
@Bean  LouisV-MT
public S3Presigner s3Presigner() {
    AwsBasicCredentials credentials = AwsBasicCredentials.create(accessKeyId, secretAccessKey);
    return S3Presigner.builder()
            .region(Region.of(region))
            .endpointOverride(URI.create("https://" + endpoint))
            .credentialsProvider(StaticCredentialsProvider.create(credentials))
            .build();
}
```

What We Learned (Team Workflow)

Team Practices:

- Merge often, commit small changes
- Review code before pushing
- Used Git branching strategy (feature branches, pull requests)
- Learned to resolve merge conflicts collaboratively
- Improved communication via regular code reviews and stand-ups

Future work

- Implement User Reviews and Ratings
- Add Favorite Recipes
- Develop Meal Planning
- Email Meal planner to user email

Summary

Achievements:

- Built a complete full-stack recipe management system
- Integrated with external REST API
- Created responsive, user-friendly interface
- Ensured data consistency and security

Result:

A functional, extensible app built through teamwork and learning.

Thank You

“A project built with passion for food and technology.”

Team Members: Vu Minh Triet , Yang Peng

Instructor: Gregory Prokopski