

A photograph of a dining table with several white plates and bowls containing various dishes, including salads, vegetables, and seafood. Hands are visible using silverware to eat. The image is dimly lit with a dark overlay.

# CSC207 Final Project **Meal Master**

Group 201

# API usage: EDAMAM

Usage: generate recipe by preference

Input:

```
String q;  
String[] diet;  
String[] health;  
String[] cuisineType;  
String[] mealType;  
String calories; // format: {MIN}-{MAX}  
String preparationTime; // format: 0-{MAX}
```

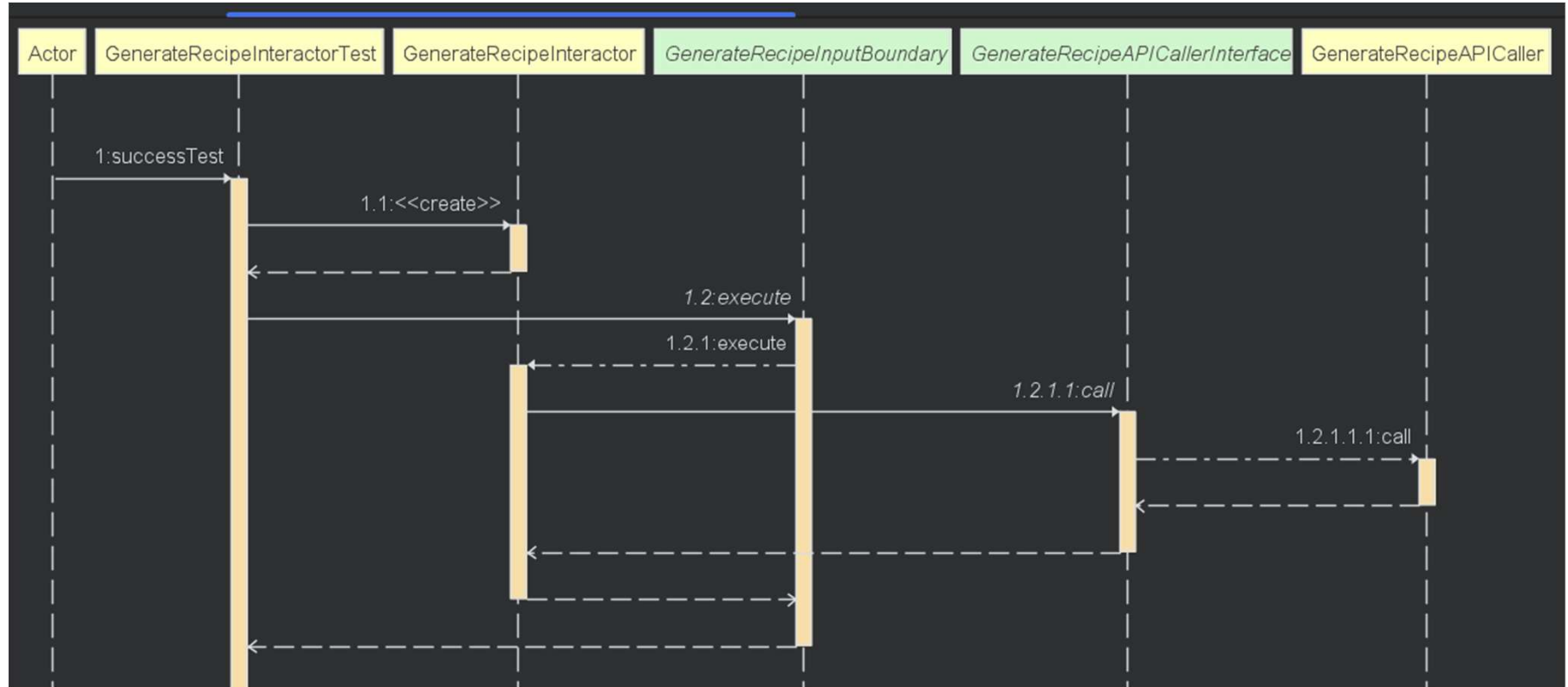
Example API call [here](#)

Output:

```
▼ recipe:  
  ▶ uri: "http://www.edamam.com/ontology/recipe/288ef27389f6fe82ae9c7c69"  
  ▶ label: "Beef goulash soup"  
  ▶ image: "https://edamam-product-i-d0fd842aa688ebab58fcabb0"  
  ▶ images: {...}  
  ▶ source: "BBC Good Food"  
  ▶ url: "http://www.bbcgoodfood.com/recipes/beef-goulash-soup"  
  ▶ shareAs: "http://www.edamam.com/recipe/high-fiber/100-20000-cal"  
  ▶ yield: 3  
  ▶ dietLabels: [...]  
  ▶ healthLabels: [...]  
  ▶ cautions: [...]  
  ▼ ingredientLines:  
    0: "1 tbsp rapeseed oil"  
    1: "1 large onion, halved and sliced"  
    2: "3 garlic cloves, sliced"  
    3: "200g extra lean stewing beef, finely diced"  
    4: "1 tsp caraway seeds"  
    5: "2 tsp smoked paprika"  
    6: "400g can chopped tomatoes"  
    7: "600ml beef stock"  
    8: "1 medium sweet potato, peeled and diced"  
    9: "1 green pepper, deseeded and diced"  
    10: "150g pot natural bio yogurt"  
    11: "good handful parsley, chopped"  
  ▶ ingredients: [...]  
  ▶ calories: 849.5175909000903  
  ▶ totalCO2Emissions: 85342.09895382394  
  ▶ co2EmissionsClass: "G"  
  ▶ totalWeight: 1792.1412975220628  
  ▶ totalTime: 75
```

# API usage: EDAMAM

## [Recipe Search API Documentation](#)



# API usage:



Usage: save user, recipe, and planner online

Input: String filepath e.g. "recipes.csv"

```
recipes.csv x
1 label,recipeUrl,imagePath,calories,preparationTime,yield,ingredients
2 White Pizza,https://food52.com/recipes/40095-white-pizza,https://eda
3 Chicken Ramen,http://norecipes.com/blog/chicken-ramen-recipe/,https:/
4 Beef stroganoff with herby pasta,https://www.bbcgoodfood.com/recipes/
5 Couscous-Stuffed Beef Tomatoes,http://www.bbcgoodfood.com/recipes/294
6 Chicken and Dumplings,https://food52.com/recipes/11995-chicken-and-du
7 Indomie Kuah,https://github.com/ChristofferTan/csc207-project,null,38
8 Chicken & sweetcorn ramen,https://www.bbcgoodfood.com/recipes/chicken
9 Asian chicken rice balls & broth,http://www.jamieoliver.com/recipes/c
10 Beef goulash soup,http://www.bbcgoodfood.com/recipes/beef-goulash-sou
11 Chicken Ramen Bowl,https://perduefoodservice.com/menu-ideas/chicken-r
12 Asian-Style Chicken and Rice,https://www.marthastewart.com/973886/as
13 Beef on a String,https://food52.com/recipes/16988-beef-on-a-string,ht
14 Indomie Goreng,github.com/ChristofferTan/csc207-project,github.com/e
15 Tofu Mushroom Ramen Soup,https://www.allrecipes.com/recipe/263060/tof
16 bebek goreng,https://,https://,100,30,20,donald bebek
17 Roasted beet and horseggram salad with feta,https://food52.com/recipes
18 Cheese Omelette,https://www.epicurious.com/recipes/food/views/cheese-
19 Ramen recipes,http://www.simply-gourmet.com/2015/12/ramen.html,https:
20 |
```

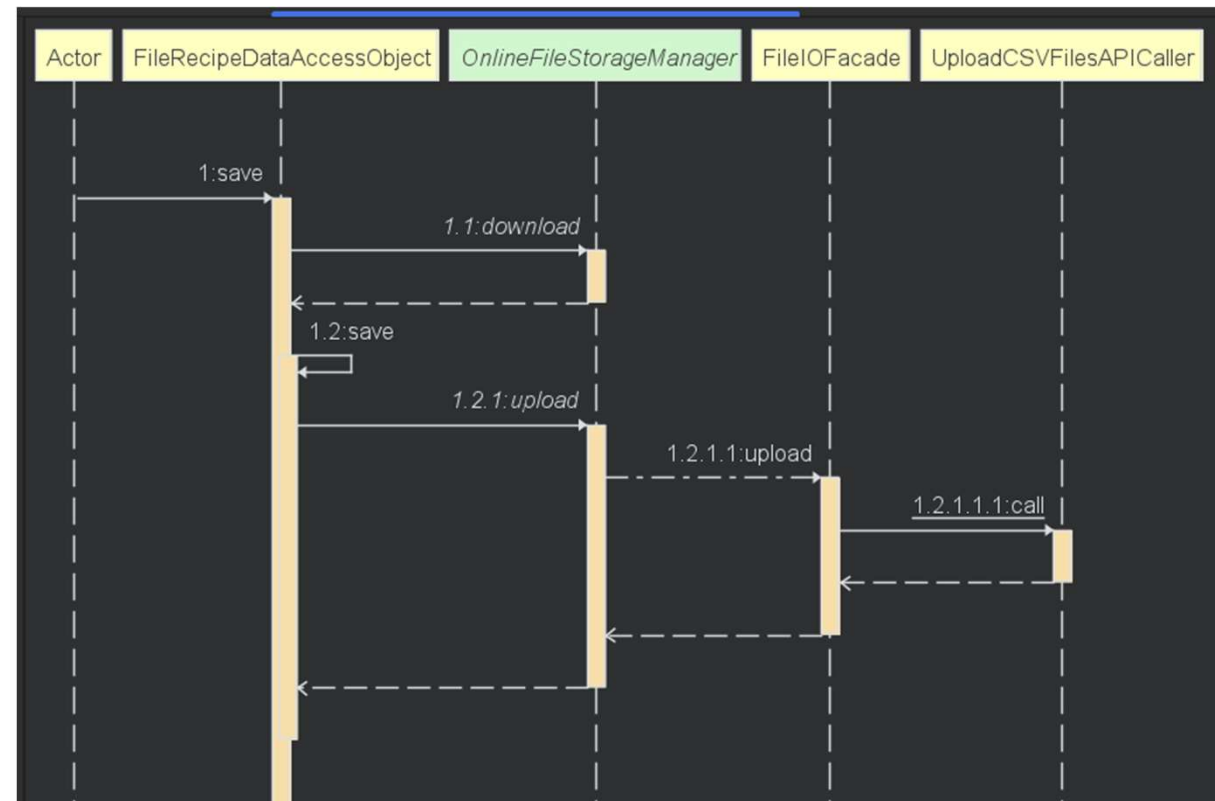
Output:

Example Value	Schema
<pre>{   "success": true,   "status": 0,   "id": "3fa85f64-5717-4562-b3fc-2c963f66afa6",   "key": "string",   "name": "string",   "link": "string",   "expires": "2023-12-06T06:04:01.624Z",   "expiry": "string",   "downloads": 0,   "maxDownloads": 0,   "autoDelete": true,   "size": 0,   "mimeType": "string",</pre>	

# API usage:



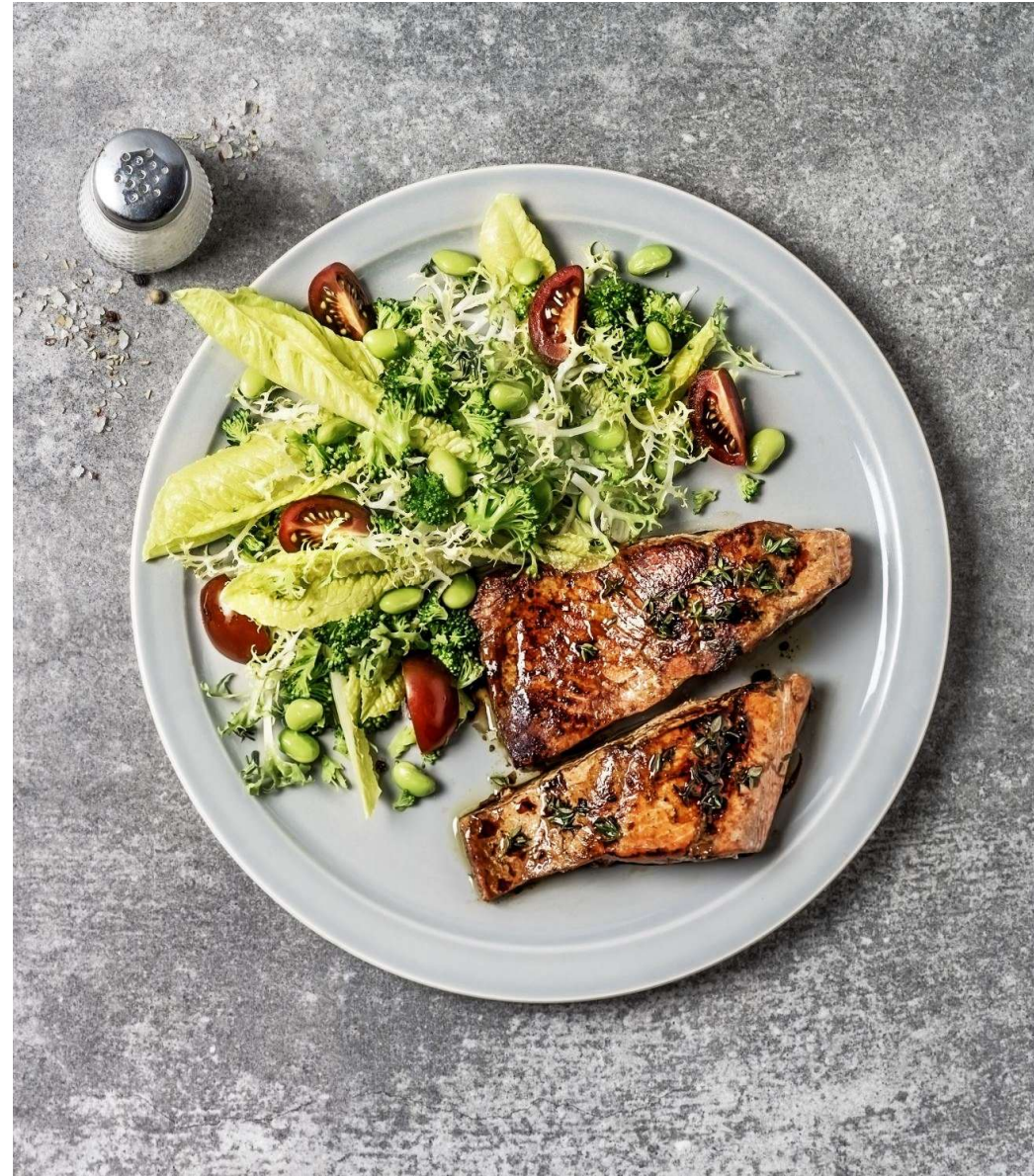
[file.io API Documentation](#)





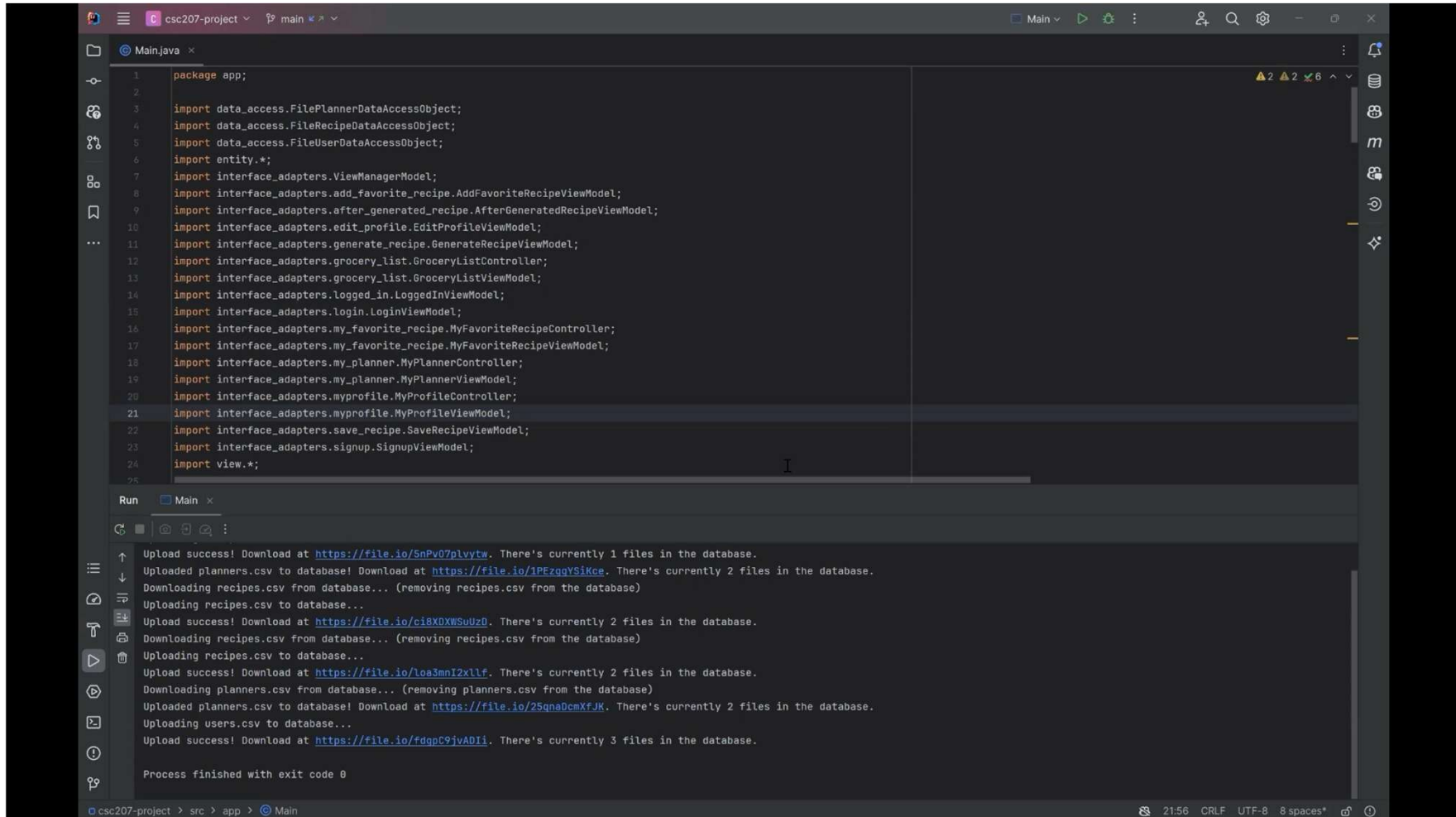
# Meal Master Specification

- **Generate Recipes:** Instantly create meal ideas to suit any taste.
- **Save to Planner:** Organize your meals for the week ahead.
- **Favorite Recipes:** Keep your most-loved recipes at your fingertips.
- **Grocery List:** Compile ingredients needed for the week with one click.
- **Calorie Tracking:** Recommendations for daily caloric needs and monitor your intake to meet health goals.



# Functionality Demonstration

<https://drive.google.com/file/d/1vkUSEyi4IIGu2cH8J7QLkMDiSc5Kid4O/view?pli=1>



The screenshot displays an IDE window for a project named 'csc207-project'. The main editor shows the 'Main.java' file with the following code:

```
1 package app;
2
3 import data_access.FilePlannerDataAccessObject;
4 import data_access.FileRecipeDataAccessObject;
5 import data_access.FileUserDataAccessObject;
6 import entity.*;
7 import interface_adapters.ViewManagerModel;
8 import interface_adapters.add_favorite_recipe.AddFavoriteRecipeViewModel;
9 import interface_adapters.after_generated_recipe.AfterGeneratedRecipeViewModel;
10 import interface_adapters.edit_profile.EditProfileViewModel;
11 import interface_adapters.generate_recipe.GenerateRecipeViewModel;
12 import interface_adapters.grocery_list.GroceryListController;
13 import interface_adapters.grocery_list.GroceryListViewModel;
14 import interface_adapters.logged_in.LoggedInViewModel;
15 import interface_adapters.login.LoginViewModel;
16 import interface_adapters.my_favorite_recipe.MyFavoriteRecipeController;
17 import interface_adapters.my_favorite_recipe.MyFavoriteRecipeViewModel;
18 import interface_adapters.my_planner.MyPlannerController;
19 import interface_adapters.my_planner.MyPlannerViewModel;
20 import interface_adapters.myprofile.MyProfileController;
21 import interface_adapters.myprofile.MyProfileViewModel;
22 import interface_adapters.save_recipe.SaveRecipeViewModel;
23 import interface_adapters.signup.SignupViewModel;
24 import view.*;
```

The terminal window at the bottom shows the following output:

```
Run Main
Upload success! Download at https://file.io/5nPV07plvytw. There's currently 1 files in the database.
Uploaded planners.csv to database! Download at https://file.io/1PEzggYSiKce. There's currently 2 files in the database.
Downloading recipes.csv from database... (removing recipes.csv from the database)
Uploading recipes.csv to database...
Upload success! Download at https://file.io/ci8XDxWSuUz0. There's currently 2 files in the database.
Downloading recipes.csv from database... (removing recipes.csv from the database)
Uploading recipes.csv to database...
Upload success! Download at https://file.io/loa3mn12xllf. There's currently 2 files in the database.
Downloading planners.csv from database... (removing planners.csv from the database)
Uploaded planners.csv to database! Download at https://file.io/25qnaDcmXfJK. There's currently 2 files in the database.
Uploading users.csv to database...
Upload success! Download at https://file.io/fdqpC9ivADIi. There's currently 3 files in the database.

Process finished with exit code 0
```

The status bar at the bottom indicates the project path 'csc207-project > src > app > Main', the time '21:56', and the encoding 'CRLF UTF-8 8 spaces'.

# SOLID Principle

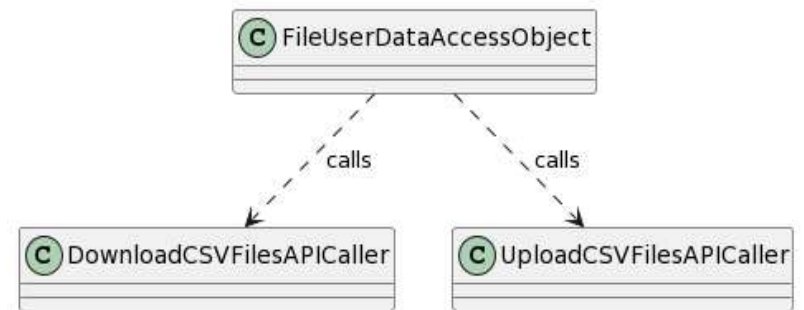
## 1. Single Responsibility Principle (SRP)



FileUserDAO, FileRecipeDAO, and FilePlannerDAO **delegates the responsibility** of downloading and uploading CSVs in a separate class.



These DAOs and the download/upload CSV class have only **one responsibility**.



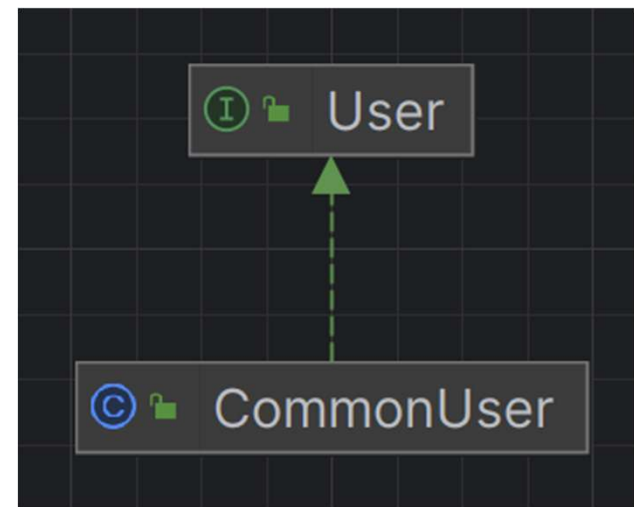


# SOLID Principle

---

## 2. Open-Closed Principle (OCP)

- User interface.
- The user interface is **open** to new implementations, so we can add new types of user by making them implement User.
- The user interface is **closed** in which we will not have to change it, and new user types will have to implement the methods in User.

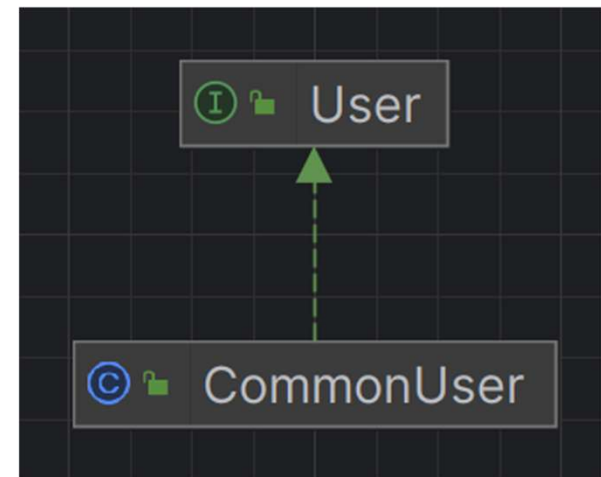


# SOLID Principle

---

## 3. Liskov-Substitution Principle (LSP)

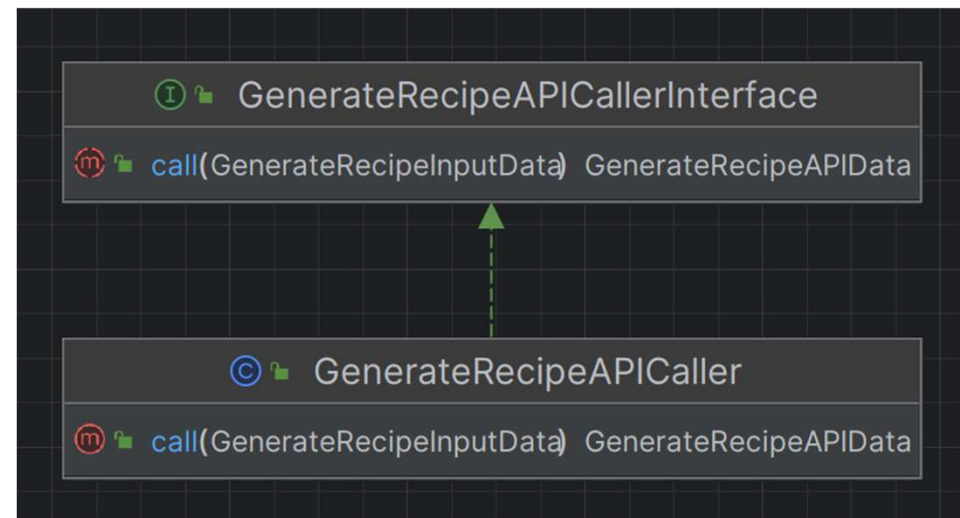
- CommonUser implements User.
- Our methods passes the User class as arguments.
- If we pass in CommonUser instead, the code will still run since CommonUser implements User.
- So User can be **replaced** by CommonUser, or other classes implementing it.



# SOLID Principle

## 4. Interface Segregation Principle (ISP)

- GenerateRecipeAPICaller, the caller for edamam API.
- Implements GenerateRecipeAPICallerInterface.
- Client does not depend on **methods they do not need** in the GenerateRecipeAPICaller.

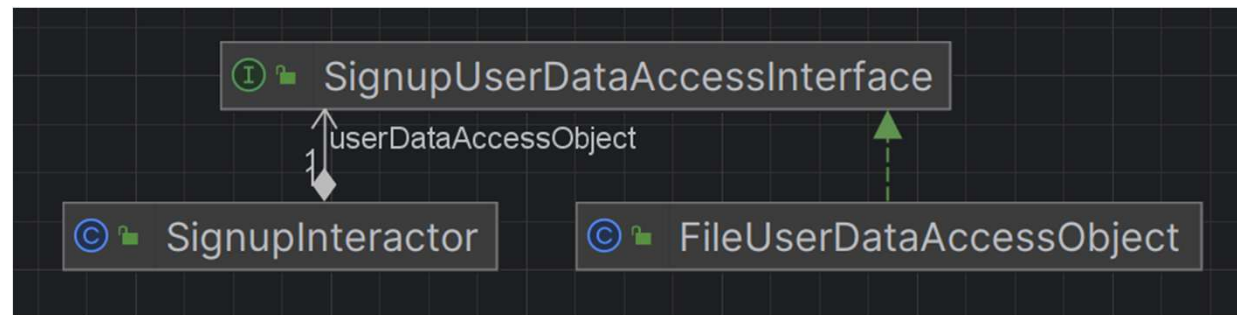


# SOLID Principle

---

## 5. Dependency Inversion Principle (DIP)

- The interactor for Sign Up, Log in, etc. **depends** on the use case data access interface not in FileUserDAO.



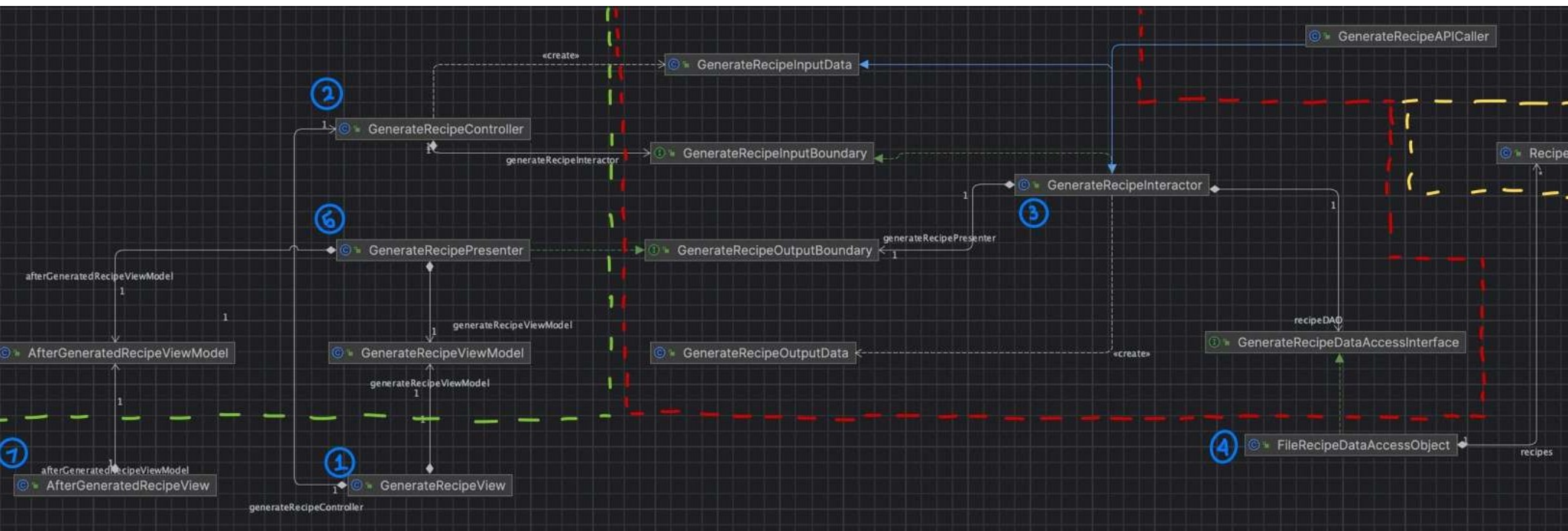


# Clean Architecture

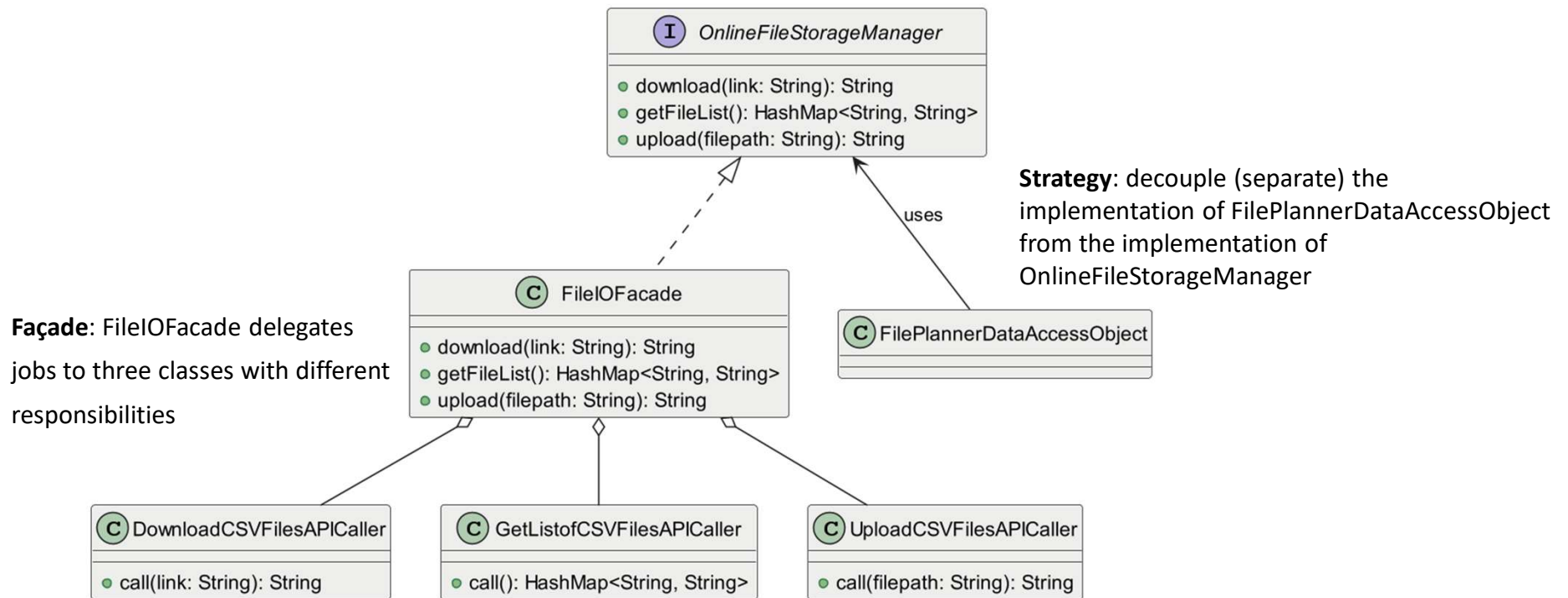
Interface adapters

Application Business Rules

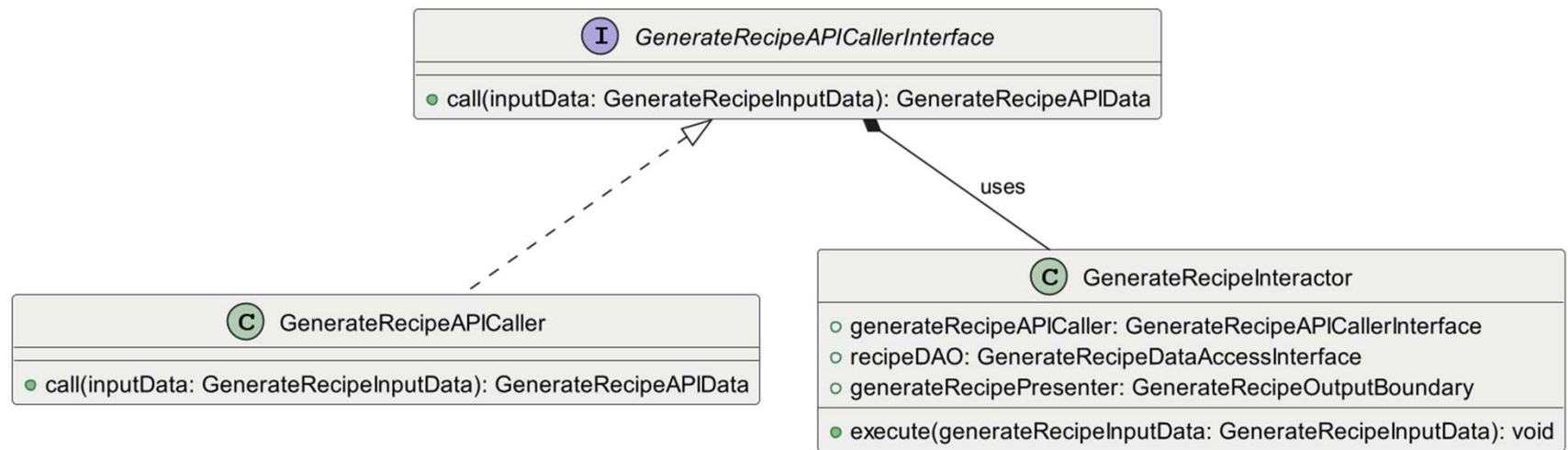
Enterprise Business Rules



# Design Pattern: Façade & Strategy



## Design Pattern: Strategy & Dependency Injection

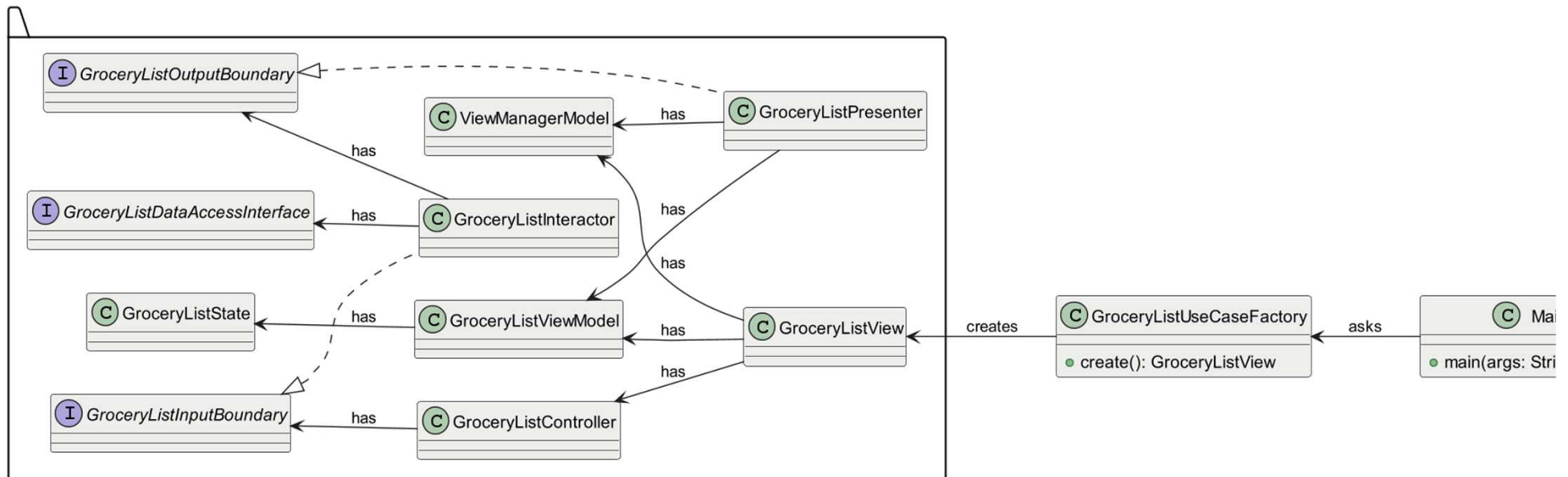


**Strategy:** can easily switch to another way of calling the API

**Dependency Injection:** avoid hard dependency by "injecting" an interface instead of concrete class.  
This allows subclasses as well

# Design Pattern: Factory

- **Factory:** obscure implementation details
- Use it without worrying about how it work





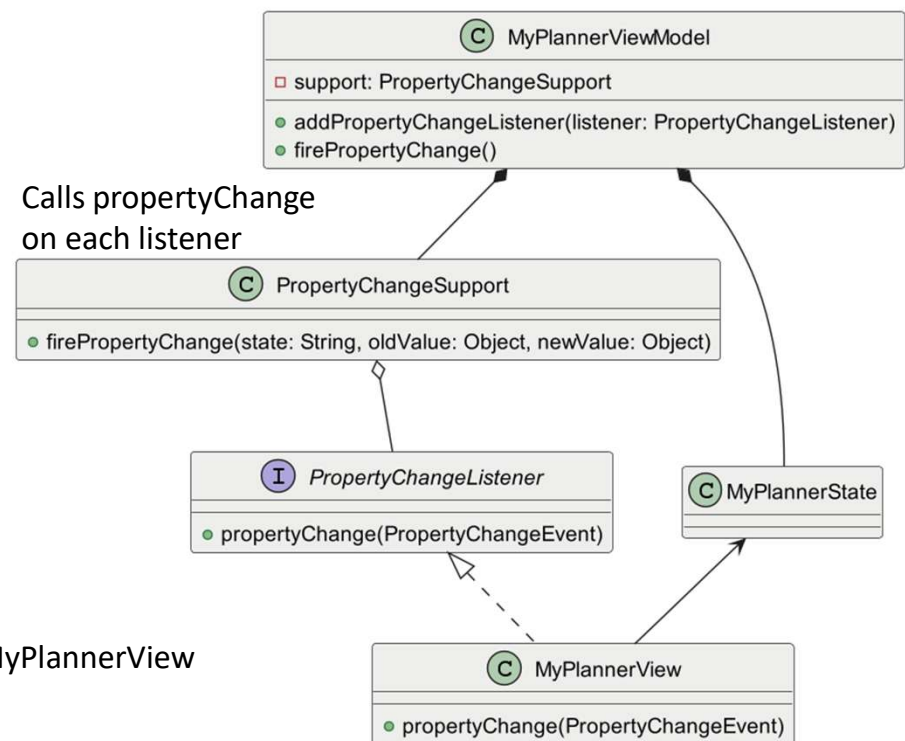
# Design Pattern: Builder

```
RequestBody requestBody = new MultipartBody.Builder()
    .setType(MultipartBody.FORM)
    .addFormDataPart(name: "file", csvFile.getName(),
    .addFormDataPart(name: "expires", API_EXPIRATION_D
    .build();

// Build the request
Request request = new Request.Builder()
    .url(API_URL)
    .post(requestBody)
    .addHeader(name: "accept", value: "application/json
    .addHeader(name: "Authorization", API_AUTHORIZATIO
    .build();
```

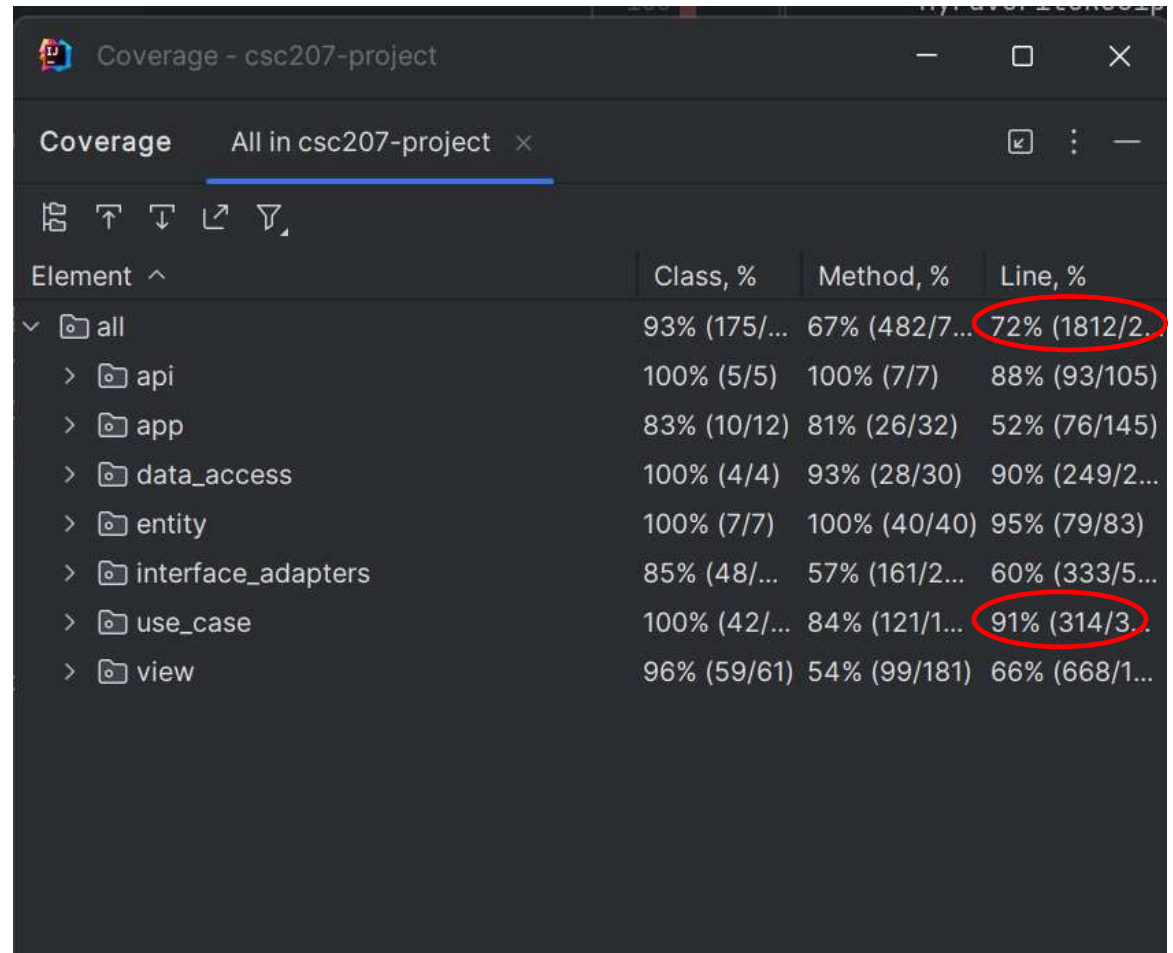
- **Builder:** create a complex Request object in a step-by-step fashion

# Design Pattern: Observer



# Testing

- **Use Case Interactors**
  - Achieved a robust **91% line coverage**.
  - Comprehensive testing of all use case interactors to ensure reliability.
- **Overall System Testing**
  - Attained an overall **72% line coverage**.
- **View Testing Approach**
  - Implemented automated testing by simulating JButton clicks and navigating through different views.
- **Areas Not Tested**
  - **Variable Outputs:** Components like GenerateRecipeAPICaller and Planner.
  - Many presenters never reach fail view.



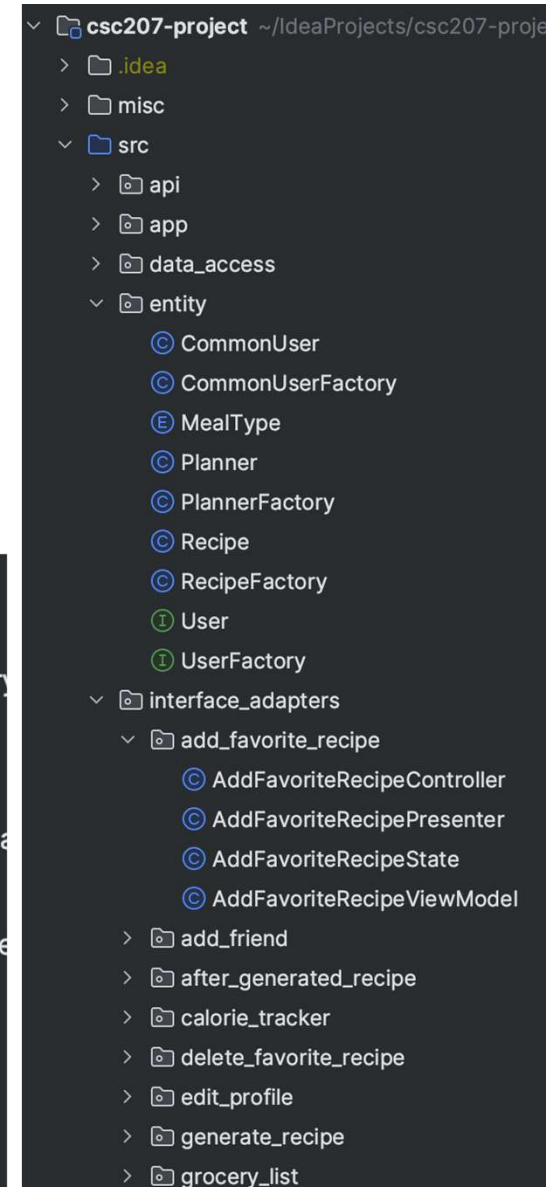
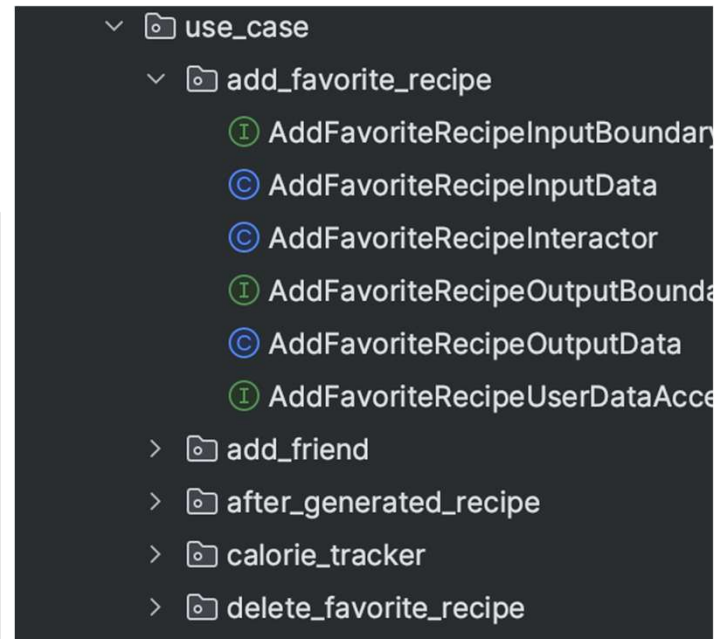
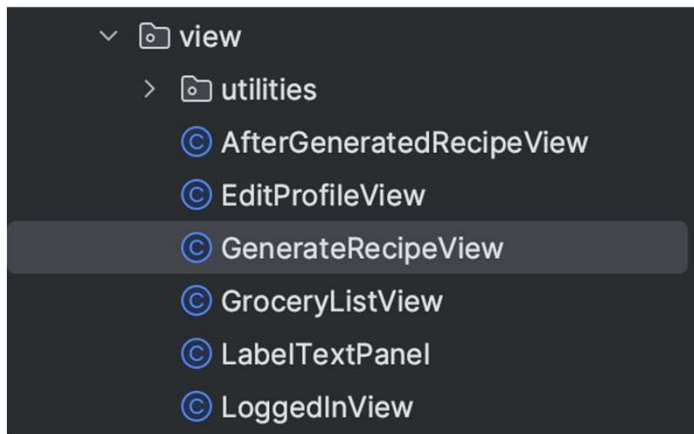
The screenshot shows a 'Coverage - csc207-project' window with a table of test results. The table has four columns: 'Element', 'Class, %', 'Method, %', and 'Line, %'. The 'Line, %' column contains values that are circled in red: 72% (1812/2...) for the 'all' element and 91% (314/3...) for the 'use\_case' element. The 'use\_case' element also has its 'Method, %' value circled in red.

Element ^	Class, %	Method, %	Line, %
✓ all	93% (175/...	67% (482/7...	72% (1812/2...
> api	100% (5/5)	100% (7/7)	88% (93/105)
> app	83% (10/12)	81% (26/32)	52% (76/145)
> data_access	100% (4/4)	93% (28/30)	90% (249/2...
> entity	100% (7/7)	100% (40/40)	95% (79/83)
> interface_adapters	85% (48/...	57% (161/2...	60% (333/5...
> use_case	100% (42/...	84% (121/1...	91% (314/3...
> view	96% (59/61)	54% (99/181)	66% (668/1...

# Code Organization

## Package by Layer of Clean Architecture

- Clear separation
- Flexibility and adaptability







---

## Further exploration

---

- **Social Feature**
  - Friends
  - Dietary advisor
- **Accessibility Improvements**
  - Goal tracking for eating disorders
  - Better accommodation for dietary restrictions



Thank You