**AppEtizer Object Oriented Analysis**

DSN

Evan Sturtevant, Kai McRoy, Emma Pinto, Jon Hopkins, and Jenna Waughen

CS 300 Software Engineering

Bill Thomas

3/8/2021

# Table Of Contents

[Noun Extraction](#_aevpdd1ot4f6) **3**

[Class Modeling](#_m93aw8q62vbh) **4**

[Scenarios](#_bfu45ert383m) **5**

[Sequence Diagrams](#_7shtjreelmir) **8**

[Use Case](#_8ksubbo0wx32) **10**

[Use Case Analysis](#_uiely315bys) **11**

[Use Case Realization](#_bh5lraaz4kbd) **12**

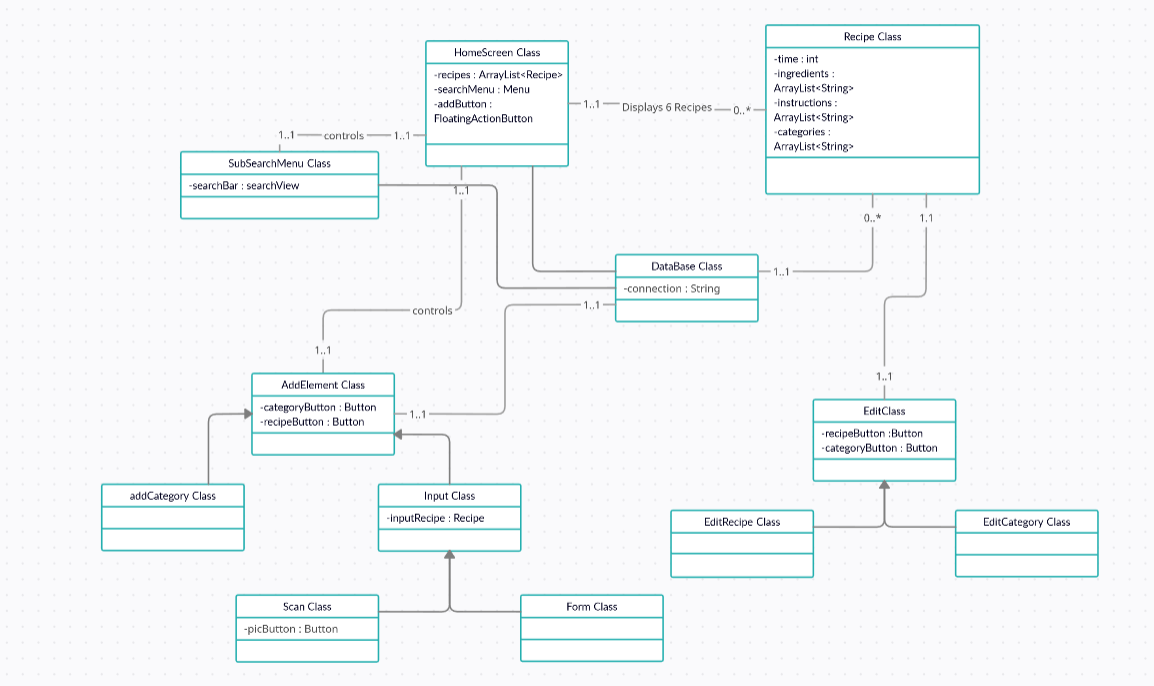
# Noun Extraction

AppEtizer is an individualized recipe **database** that allows for input, categorization, and deletion of **recipes**. The **app** will have two different ways to input their **recipe**, through manual inputs and scanning. It will ask for **information** on the **recipe** to which is then inputted by filling out a **form**. **Information** needed includes **total time** and **ingredients**. The **app** will have built in suggested **categories**  and a place to enter **categories**  to allow organization of **recipes**. Along with adding **categories**, **categories** may also be deleted.

The relevant nouns are highlighted in the above product description and yielded the following candidate classes: **Recipes**, **Categories**, **Input**, **Form**, and **Ingredients**. All but Ingredients and Categories have been implemented as classes in the design of the app, with these two being implemented as entity tables in the database.

Nouns that were not identified as candidate classes are: Database, Information, and Total Time. Information was determined to be a general noun and too vague for implication. Total Time is something that is more likely to be an attribute to the Recipe class than to be a class on its own. Database was determined to be its own entity, beyond the scope of the classes in java, though we are planning to implement a class in order to easily access and modify the database.

# Class Modeling



# Scenarios

Scenario #1 (User scans their recipe):

User A wishes to scan in a physical recipe into the app.

1. User A opens the app
2. User A selects the “Add” button
3. User A selects the “Recipe” option
4. User A selects the “Scan Recipe” option
5. User A uses devices camera to take an image of physical recipe
6. User A is prompted to check that the scanner has collected the proper information
7. The recipe is saved to the database
8. User A is returned to the AppEtizer home screen

Scenario #2 (User hand types recipe):

User B wishes to manually type a recipe into the app.

1. User B opens the app
2. User B selects the “Add” button
3. User B selects the “Recipe” option
4. User B selects the “Manually Input Recipe” option
5. User B is prompted to provide the ingredients, instructions, category for the recipe
6. The recipe is saved to the database
7. User B is returned to the AppEtizer home screen

Scenario #3 (User adds bad category name):

User C wishes to add a category but accidentally gives a recipe name instead of category name.

1. User C opens the app
2. User C selects the “Add” button
3. User C selects the “Category” option
4. User C is prompted to enter a category name
5. User C inputs a recipe name instead of a category name
6. User C saves category to database

User C realizes mistake

1. User C opens AppEtizer menu
2. User C selects the category with the incorrect name
3. In the category User C selects “More Options”
4. User C selects “Edit Category”
5. User C is prompted to to change the category name
6. User C inputs correct category name
7. User C saves category to database

Scenario #4 (User scans something that is not recipe):

User D wishes to add a recipe using the scan function but takes a picture of a cat instead of the recipe.

1. User D opens the app
2. User D selects the “Add” button
3. User D selects the “Recipe” option
4. User D selects the “scan recipe” option
5. User D uses devices camera to take an image of their cat
6. The app returns a message to User D that no information can be found from scan
7. User D is brought back to the device camera to take an image again
8. User D uses the the photo camera again taking the image of the recipe
9. User D is prompted to check that the scanner has collected the proper information
10. The recipe is saved to the database
11. User D is returned to the AppEtizer home screen

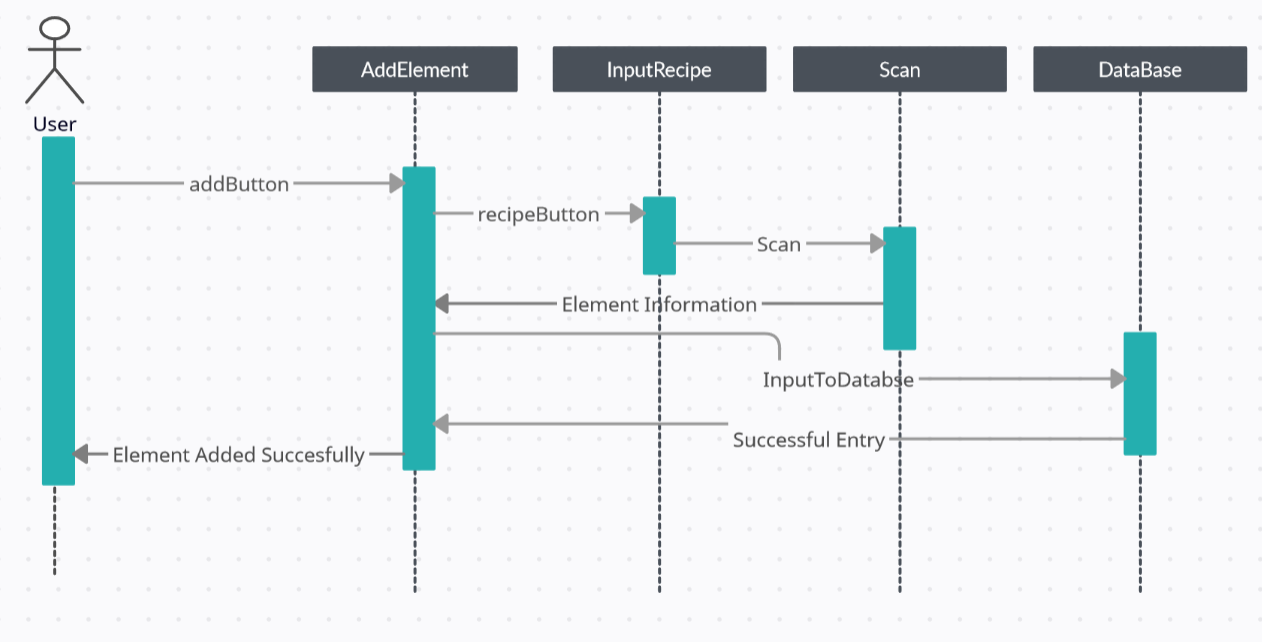
Scenario #5 (User searches and deletes a recipe):

User E wishes to use the app search function to search for a recipe and then delete the recipe.

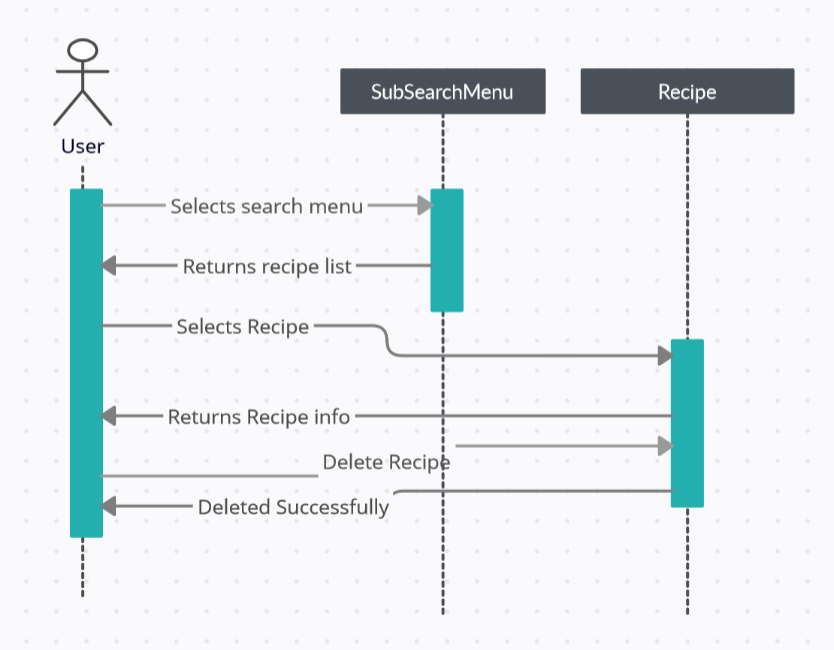
1. User E opens the app
2. User E taps on the search bar
3. User E enters the recipe name into the search bar
4. The app queries the database for the app with the corresponding name
5. The user is presented with the retrieved recipe
6. User E selects the recipe
7. In the recipe User E selects “More Options”
8. User E selects “Edit Recipe”
9. User E deletes recipe
10. Recipe is deleted from database
11. User E is returned to the AppEtizer home screen

# Sequence Diagrams

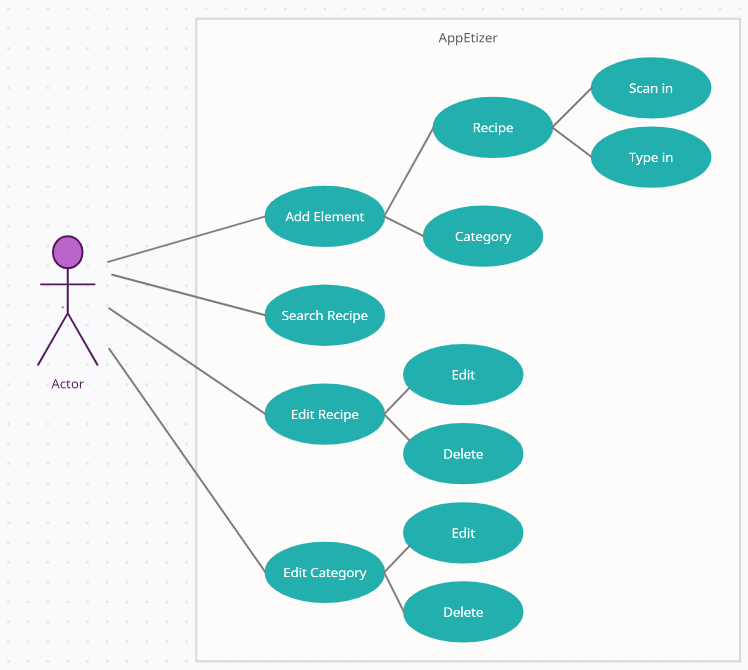
Inputting Recipe:



Search + Delete Recipe:



# Use Case

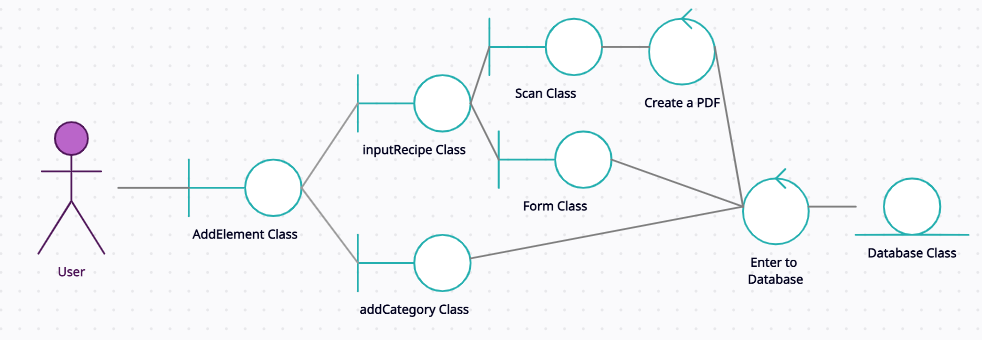


# Use Case Analysis

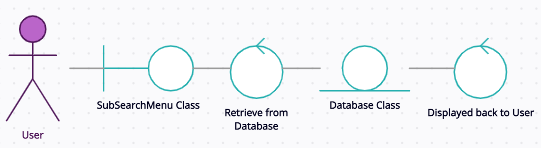
The use case has seen a few iterative changes since its original conception. Ultimately the diagram still represents the same thing as there haven’t been overarching design changes to the product itself. The changes mostly reflect refined insight of the inner workings of the classes which will constitute the app. More branches to the diagram have been added, this reflects the planned button layouts, that weren’t so clear at the beginning of the project. For example, now the delete recipe is embedded in the edit, meaning the user has to select a recipe, and then they can decide to delete or edit it.

# Use Case Realization

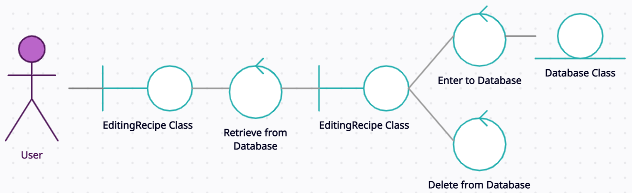
|  |
| --- |
| **Brief Description**  The *Add Element* use case enables the user to *input a recipe* into AppEtizer by either *Scan In* or *Type In* their desired recipe or to *input a category*. |
| **Step-by-Step Description**   1. User inputs a recipe.    1. User inputs recipe via scanning or manually typing and determines category/categories to put the recipe in.       1. User scans a recipe       2. User types in recipe.    2. Recipe is entered into the database. 2. User inputs a category.    1. User inputs a category name.       1. User inputs category name(s).    2. Category name(s) are entered into the database. |



|  |
| --- |
| **Brief Description**  The *Search Recipe* use case enables the user to search a recipe. |
| **Step-by-Step Description**   1. User searches for a recipe.    1. User clicks on desired recipe.    2. Recipe is retrieved from the database. |



|  |
| --- |
| **Brief Description**  The *Edit Recipe* use case enables the user to *edit* or *delete* a recipe that has been previously entered. |
| **Step-by-Step Description**   1. User edits a recipe.    1. Recipe is retrieved from the database.    2. User edits recipe.    3. Modified recipe is entered into the database. 2. User deletes a recipe.    1. Recipe is retrieved from the database.    2. User deletes recipe. |



|  |
| --- |
| **Brief Description**  The *Edit Category* use case enables the user to *edit* or *delete* a category that has been previously entered. |
| **Step-by-Step Description**   1. User edits a category.    1. Category is retrieved from the database.    2. User edits category.    3. Modified category is entered into the database. 2. User deletes a category.    1. Category is retrieved from the database.    2. User deletes category. |

