Christine Grosso

cgrosso@bu.edu

Abstract

A Class Model, an Activity Diagram and Pseudocode for a single method in the App.

Detailed design of homebot robot’s eetz app

Assignment 6—October 17, 2019

Table of Contents

[Introduction 1](#_Toc22024968)

[Assumptions 2](#_Toc22024969)

[Class Model (updated) 3](#_Toc22024970)

[Activity Diagram 4](#_Toc22024971)

[Pseudocode for method eetz.confirmIngredients() 5](#_Toc22024972)

[Appendix A: Use-Cases 7](#_Toc22024973)

[Use-Case #1 7](#_Toc22024974)

[Use-Case #2 9](#_Toc22024975)

[Appendix B: Sequence Diagram (updated slightly) 11](#_Toc22024976)

[Appendix C: GUI Sketch 12](#_Toc22024977)

[Works Cited 13](#_Toc22024978)

# Introduction

This paper explores the implementation of the method:

confirmIngredients()

in the <<Design>>:EetzApp class for the *homeBot eetz* App. As specified in the Pseudocode section, the method is called with a :Recipe object calling argument, and returns a boolean value.

Presented in the body of this paper are:

* a (revised from previous) Class Model [see Appendix A-1, Use-Case 1; and Appendix B, Sequence Diagram],
* an Activity Diagram,
* and Pseudocode

for the method. The Find-Recipe Use-Case is the scenario for which the Class Model is designed. Several preconditions are set up before the call to the method, and the boolean result of the method will determine whether it is possible to make the recipe or not, from ingredients already at home. These pre- and post- conditions are explained in the next section.

# Assumptions

Note to Developer: specific names of objects and variables (in green) are included here only for clarity in describing function calls and object interfaces.

**System Preconditions:**

* Instantiation examples of :EetzApp (upon initial system start-up):
* eetz = new :EetzApp
* larder = new :HomeInventory
* myBox = new :RecipeStash
* tini = new :Cook
* bob = new :Cook
* shu = new :Cook

**Method Preconditions** (which are Use-Case **Conditions** that happen before the method is called)**:**

* Instantiation example of :RecipesList:
  + possibleRecipes = new :RecipesList
* :GoogleAssistant takes each recipe it finds in the cloud and passes it, via its sendRecipeToBot() method, to eetz. The main() method in eetz calls the :Recipe class to:
  + create/construct a new :Recipe object (example: newRecipe = new :Recipe)
  + translate the recipe information from the :GoogleAssistant into the necessary format to store in newRecipe, which includes the newRecipe object calling the :Ingredient class to instantiate :Ingredient objects for the newRecipe’s ingredients attribute, which is a list.
  + store the recipe information in newRecipe.
* After performing the actions in the above bullet, and since one of the filters chosen by, say, tini, was “I don’t want to shop” (see Appendix C, GUI App), eetz calls:

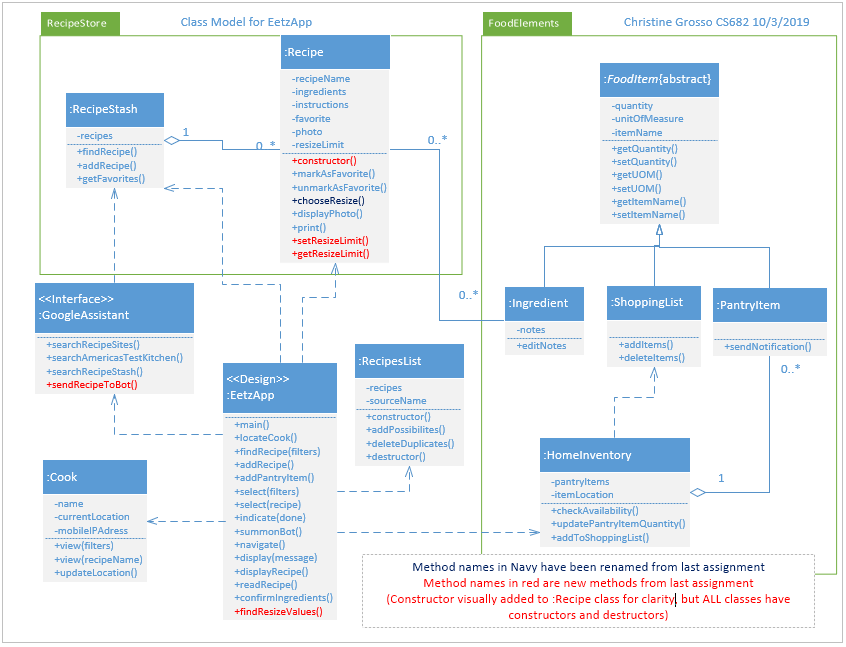
eetz.confirmIngredients(newRecipe) // THIS IS THE METHOD BEING MODELED

// this method returns a boolean value

**Postconditions:**

* If the boolean value returned = false, newRecipe is deleted.
* If the boolean value returned = true, eetz.addRecipe(newRecipe) is called to add newRecipe to possibleRecipes.

# Class Model (updated)



# Activity Diagram



# Pseudocode for method eetz.confirmIngredients()

(As mentioned above in Assumptions section, specific names of objects and variables [in green] are included here only for clarity in describing function calls and object interfaces. **Names can be whatever the developer prefers.**)

This method will determine if there are sufficient pantry items in the home inventory to enable a given recipe to be made without first going to the grocery store. If there ARE sufficient pantry items to make at least half of a recipe and up to four times a recipe, the newRecipe.resizeLimit attribute will be updated with a value in the range of {0.5, 2.0, 3.0, 4.0} (these values may change in a future release).

|  |  |
| --- | --- |
| **Method Call**: | eetz.confirmIngredients(newRecipe) |
| Calling argument type: | :Recipe object |
| Return type: | boolean |
|  | True = enough ingredients in homeInventory to make at least half of the recipe |
|  | False = not enough ingredients in homeInventory to make at least half of the recipe |
| Note: | If True, the maximum available multiple (up to 4) will be stored in the attribute newRecipe.resizeLimit. |

**Other methods called by this method:**

|  |  |
| --- | --- |
| **Method Call**: | larder.checkAvailability(newRecipe.ingredients[index]) |
| Calling argument type: | :Ingredient object |
| Return type: | float |
| Note: | The floating point number returned will be in the range {0.0,…}. The value will indicate how many times more, if any, this ingredient exists in the homeInventory, compared to the amount needed for the recipe. |

|  |  |
| --- | --- |
| **Method Call**: | eetz.findResizeValues(multiple) |
| Calling argument type: | float in the range {>0.0,…} |
| Return type: | float in the range {0.0, 0.5, 1.0, 2.0, 3.0, 4.0} |
| Note: | THESE VALUES MAY CHANGE IN A FUTURE RELEASE.  This method will translate a positive floating-point number into a multiple, acceptable to the system, of how many times a recipe can be increased, based on a pre-defined scale. |

|  |  |
| --- | --- |
| **Method Call**: | newRecipe.setResizeLimit(resizeFactor) |
| Calling argument type: | float in the range {0.5, 1.0, 2.0, 3.0, 4.0} |
| Return type: | none |
| Note: | sets the resizeLimit attribute in the :Recipe object that will allow a :Cook object to choose a valid resize option for the recipe. |

Note: object instances and variables in green, method calls in red, comments in purple.

//BEGIN

// newRecipe.ingredients – some kind of list containing indexed :Ingredient objects, which

// would possibly be indexed something like: 0 - length(newRecipe.ingredients)-1

multiple = 4.0 // max number of recipe size increases (determined by min ingredient in pantry)

index = 0 // initialize index to whatever your list needs it to be

//loop through each ingredient in newRecipe. Call the checkAvailability() method in larder to //see how many times the recipe-amount of that ingredient exists in larder. If there is 0 of //something, we are done.

while multiple > 0 and index < length(newRecipe.ingredients)

//while we didn’t encounter an ingredient we don’t have on-hand, AND we haven’t yet

// checked every ingredient in the recipe…

multiple = min(multiple, larder.checkAvailability(newRecipe.ingredients[index]) // METHOD

// CALL

// note to developer: implement the “find min” function any way you want.

index++ // increment ingredients index by 1

endWhile

if multiple = 0, return FALSE // an ingredient in the recipe is not in the homeInventory

else resizeFactor = findResizeValues(multiple) // get a “resize factor” from METHOD CALL

if resizeFactor = 0, return FALSE // multiple was < 0.5; newRecipe object will be destructed

else newRecipe.setResizeLimit(resizeFactor) // METHOD CALL (resizeFactor argument will

// update the resizeLimit attribute in the newRecipe object

return TRUE

# Appendix A: Use-Cases

## Use-Case #1

Key: entity class, non-entity, attribute, method

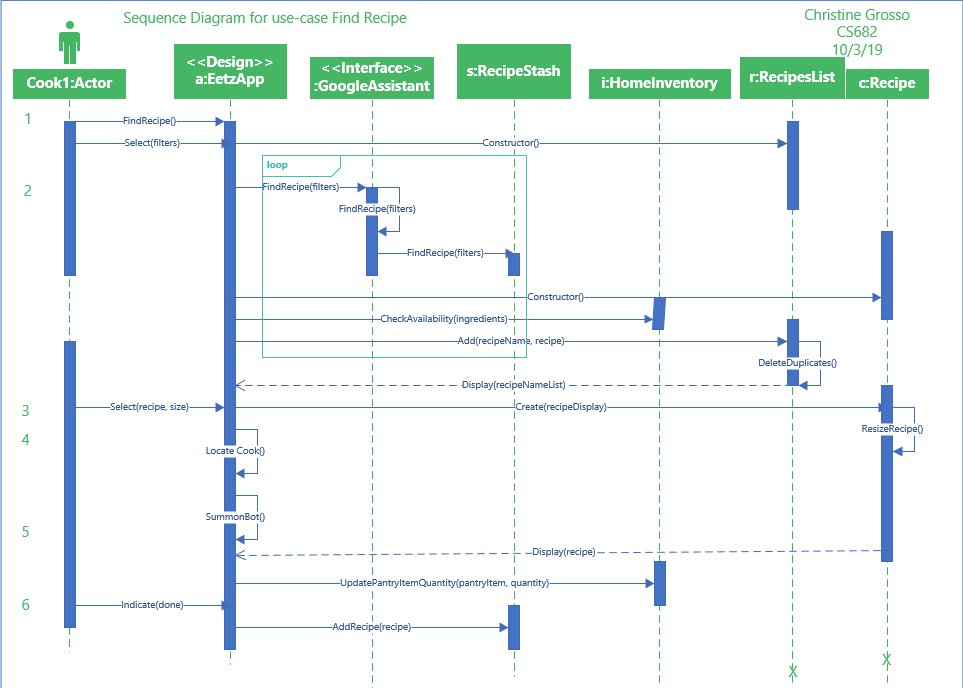
|  |  |  |
| --- | --- | --- |
| Use-Case Name | Find Recipe | |
| Actor | Cook1 | |
| Description | This use-case describes the event of Cook1-- known to *homeBot* (see Appendix B-1)--finding a recipe for what Cook1 can make for dinner, based on “I need an idea.” Food inventory has previously been updated with *homeBot* object-recognition capabilities (See Appendix B-1). The use-case will end when Cook1 indicates “Done.” | |
| Pre-condition | Cook1 has opened *eetz* app on mobile phone. Screen displays, “Are you hungry, <name of Cook1>? Letz *eetz*!!” and a menu of options. | |
| Step # | Actor | System |
| 1 | “Find Recipe” is preselected on options list, and Cook1 touches “Go” | System displays menu of Find Recipe filters for Cook1 to view. Preselected filters are: “I need an idea,” “Main dish,” “I don’t want to shop,” “Prep time < 30 mins.” Non-preselected filter is: “Ingredient” with an associated text box. |
| 2 | Cook1 touches “Go” | System finds recipes in cloud and Recipe Stash, checks availability of Home Inventory, adds recipes to Recipe List, and deletes duplicates. |
| 3 |  | System displays recipe names for Cook1 to view. |
| 4 | Cook1 scrolls through choices, and selects a Recipe (such as: “Chicken Cacciatore”) and a resize option. | System resizes recipe **Invalid source specified.**, locates Cook1, summons *homeBot* and displays recipe. |
| 5 |  | *homeBot* displays Recipe on screen, navigates to Cook1, reads recipe instructions to Cook1, and updates pantry items. |
| 6 | Cook1 indicates “Done” | System adds recipe to Recipe Stash and displays farewell message. |
| Alternate Courses | Alt Step 2 (Actor/System): Cook1 deselects “I don’t want to shop” and touches “Go.” System responds with additional choices.  Alt Step 2 (Actor/System): Cook1 deselects “Prep-time < 30 min” and touches “Go.” System responds with different choices.  Alt Step 2 (Actor/System): Cook1 selects “Ingredient”; enters, for example, “Coconut milk” in text box; and touches “Go.” System responds with Recipes containing coconut milk as an ingredient.  Alt Step 4/5 (System/Actor): System resizes Recipe and determines Cook1 is not home. Cook1 arrives home and summons *homeBot.* | |
| Implementation Constraints | * *homeBot* searches Recipes from personal Recipe stash as well as from integrated cloud services (See Appendix B-5). * The recipe stash for the *homeBot* system resides in the cloud so that it can be accessed from mobile devices integrated with the *homeBot* system. * Mobile device’s IP address and GPS location are used by homeBot to greet Cook1 personally and determine Cook1’s location, respectively (See Appendices B-1 and B-5).\* | |

## Use-Case #2

Key: entity class, non-entity, attribute, method

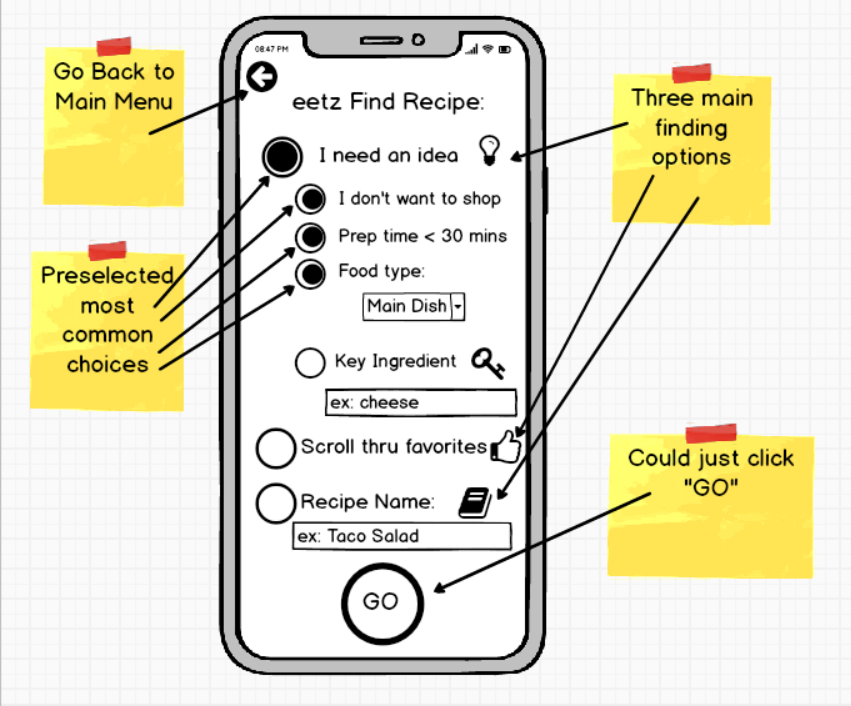
|  |  |  |
| --- | --- | --- |
| Use-case name | Add Recipe | |
| Actor | Cook2 | |
| Description | This use-case describes the event of Cook2 adding a Recipe to the *homeBot* Recipe stash. The use-case will end when Cook2 indicates “Done.” | |
| Pre-condition | Cook2 has asked *homeBot* to open *eetz* app, and then selected “Add Recipe” from the main menu. The *homeBot* recognizes Cook2 (see Appendix B-1) and asks, “Are you hungry, <name of Cook2>? How can I help you?”(See Appendices B-5 and B-7.) Food inventory has previously been entered with *homeBot* object-recognition capabilities (See Appendices B-1). The use-case ends when Cook2 indicates Cook2 is “Done.” | |
| Step # | Actor | System |
| 1 | Cook2 asks the *homeBot* to add a Recipe, such as Grandma’s Meatballs, to Recipe stash. | System says, “Sure. Would you like me to find one with that name, or do you want to scan it in?” |
| 2 | Cook2 responds with “I want to scan it in.” | System responds with, “Ok. I will wait to receive your scanned Recipe,” and waits. (See Appendix B-6.) |
| 3 | Cook2 scans Recipe with mobile device and sends to *homeBot.* | System responds with, “It looks like this Recipe will take more than 30 minutes to prepare. Is that correct, <name of Cook2>?” |
| 4 | Cook2 agrees. | System adds recipe to Recipe Stash and responds with, “Ok. I will add this Recipe to your recipe stash, <name of Cook2>. Would you like to add another Recipe, or cook something, or add to the pantry?” |
| 5 | Cook2 says, “No, thank you,” and indicates “Done.” | System responds with, “You’re welcome. Please let me know if you need anything else.” |
| Alternate Courses | Alt Step 2 (Actor/System): Cook2 responds with, “I want you to find me a Recipe with that name.” System finds a Recipe from the cloud, adds it to Cook2’s stash, and informs Cook2.  Alt Step 5 (Actor/System): Cook2 says, “Yes, I would like to make this Recipe now.” System responds with, “Ok. I will help you.” *homeBot* displays Recipe on screen, guides Cook2 through instructions in Recipe, and answers verbal questions. | |
| Implementation Constraints | The recipe stash for the *homeBot* system resides in the cloud so that it can be accessed from mobile devices integrated with the *homeBot* system. | |

# Appendix B: Sequence Diagram (updated slightly)



# Appendix C: GUI Sketch

Mockup of Find Recipe menu screen on iPhone:



(Sketch inspired by ideas from Module 3 in Week 3 Assignment Preview by Prof. Jack Polnar, 9/18/2019.) (Braude, 2019)

(Grosso, 2019)

# Works Cited

Braude, E. P. (2019, March 19). MET CS 682--Module 3: Assignment 3--Requirements. Boston, MA, USA.

Grosso, C. (2019). *Requirements for homeBot Robot eetz App--Assignment 3.* Boston, MA, USA: Boston University.

|  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- |
|  | **D** | **C-** | **C+** | **B-** | **B+** | **A** | **Your Letter Grade** |
| **I Clarity** | Disorganized or hard-to-understand | | Satisfactory but some parts of the submission are disorganized or hard to understand | Generally organized and clear | Very clear, organized and persuasive presentation of ideas and designs | Exceptionally clear, organized and persuasive presentation of ideas and designs | **A+** |
|
|
| **ii Technical Soundness** | Little understanding of, or insight into material technically | | Some understanding of material technically | Overall understanding of much material technically | Very good overall understanding of technical material, with some real depth | Excellent, deep understanding of technical material and its inter-relationships | **A+** |
|
|
| **iii Thoroughness & Coverage** | Hardly covers any of the major relevant issues | | Covers some of the major relevant issues | Reasonable coverage of the major relevant area | Thorough coverage of almost all of the major relevant issues | Exceptionally thorough coverage of all major relevant issues | **A+** |
|
|
| **iv Relevance** | Mostly unfocused | Focus is off topic or on insubstantial or secondary issues | Only some of the content is meaningful and on topic | Most or all of the content is reasonably meaningful and on-topic | All of the content is reasonably meaningful and on-topic | All of the content is entirely relevant and meaningful | **A** |
|
|
| **v Utilization of resources or UML tools** | No useful use of notes, text(s), Web or UML tools with incorrect details or applicability | | Some useful use of notes, text(s Web or UML tools with mostly correct details or applicability | Fairly good use of notes, text(s Web or UML tools with correct details or applicability | Very good use of notes, text(s), Web or UML tools with correct details or applicability | Excellent use of notes, text(s), Web or UML tools with entirely correct details or applicability | **A-** |
|

# You rock this course. Nice work! Good luck on the final. Total 96!