**AppEtizer Specification Document**

DSN

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

CS 300 Software Engineering

Bill Thomas

3/1/2021

# Table of Contents

Acceptance Criteria . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . 2

Solution Strategy. . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . 3

Informal Specification . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . 3

Computerization. . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . 4

Details of the Data Flows. . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . 5

Logic of the Processes. . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . 6

Blocking Factors . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . 8

Hardware Requirements . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . 9

Input-Output Specifications . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . 10

Sizing . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . .10

ER Diagrams . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . 11

# Acceptance Criteria

The following list are features of AppEtizer.

* Users can add recipes
  + Users can manually type out recipes
  + Users can scan recipes
* Users can modify saved recipes
* Users can delete recipes
* Users can place recipes in a specific category
* Users can create categories
* Users can modify categories
* Users can delete categories
* Users can search recipes
* Users can “star” recipes
* Recipe database stores recipes for individual users
* Recipe database deletes individual user recipes
* Recipe database can identify user recipes
* Recipe database allows for all recipes to be viewed but limits modification to user defined recipes.

# 

# Solution Strategy

AppEtizer will be built on and will use the following database:

* We will use Android Studio to develop our app user interface
* We will create a SQLite database to store recipes

# Informal Specification

* Categories are displayed on the left hand side of the screen.
* If certain ingredients are in a recipe, it will display if the recipe is vegan, vegetarian, lactose free, etc. if all of the recipes contain the proper ingredients.
* When a user adds recipes by hand, the app will save the information into the database to be stored.
* If a user scans in a recipe, the app will use a library to extract the attributes of the recipe (ingredients, directions, etc.) and put them into the database.
* If a recipe is tagged with a category, that recipe will show up under that category in the app and will be sorted into that category in the database.

# 

# Computerization

AppEtizer will contain processes that are all online. Batch processing does not make sense in this case. AppEtizer primarily obtains and exports information through a database. These queries and inputs will not be a lot of data. For example, most recipes are only one or two pages long. The information stored on these will not be too much to transfer to and back from a database. All the information will not need to be displayed at once as well. When the recipe list is on the screen, the user only needs specific highlights of information (Cook Time, Title, Frequency, Categories). Only when the recipe is selected is the rest of the data presented from the user, prompting a query for that information. Since AppEtizer is not retrieving all information for all recipes at once, it makes the online processes simple and small.  
 A batch system would not fall in line with AppEtizer’s requirements. The user must be able to select their recipe soon after they input it. With a batch processing system, the user’s inputs will not be reflective in the database until certain time intervals. With an online system, this problem is alleviated, making the actions performed by the user immediately reflected on the app.

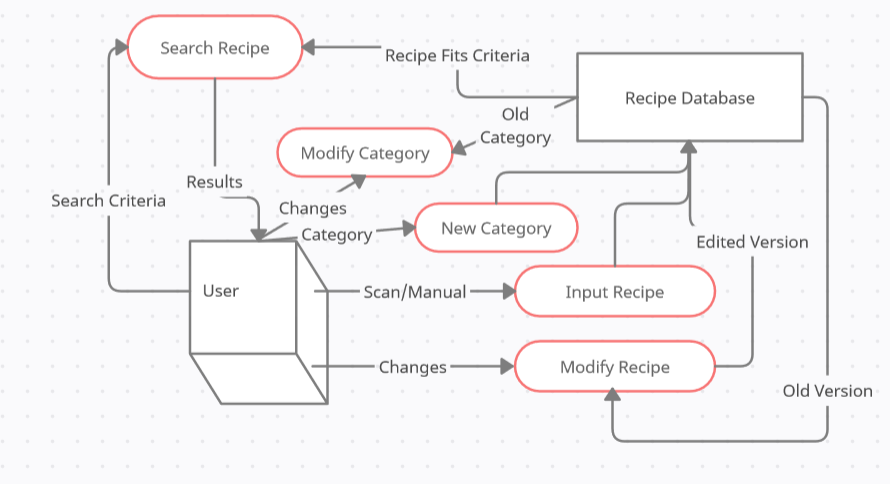
# Details of the Data Flows

The following is a Data Flow Diagram which demonstrates how data will pass through the app. The User has several ways they can provide data to the app through inputting recipes, modifying recipes, inputting categories, modifying categories, and providing search criteria. The only place data is being stored is in the recipe database. The database can provide data in a few situations:

(1) the user searches for a recipe, in which case the database returns recipes that match that description for the user to choose;

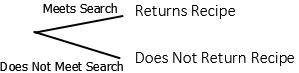
(2) the user requests to modify a recipe, in which case the database returns the current saved version;

(3) the user requests to modify a category, in which case the database returns the current saved category.



# Logic of the Processes

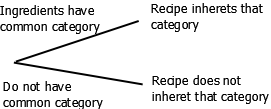
The **search recipe** process only returns recipes that meet the user’s search. If the recipe does not fall in line with the search criteria, it won’t be returned to the user.



The **modify category** process takes the old category and replaces it with the new category.

The **new category** process takes a new category imputed by the user and submits it to the database as well as the recipe it is being added on.

The **input recipe** process takes the scan or information from the user and pushes the new recipe to the database. If all of the ingredients in the recipe have a common category, then the recipe itself inherits that category.



The **modify recipe** process changes an aspect of the recipe. This could be any of the attributes of the recipe, as well as the ingredients.

# Blocking Factors

The database management system we are using is SQLite with Android Studio.

The tables with their attributes in our database are as follows:

This is also referenced with our ER diagram.

The bold attributes are the primary keys of the tables.

The italics are foreing keys of other tables.

* Recipe
  + **ID**
  + Name
  + Instructions
  + Image
* Ingredients
  + **Name**
* Category
  + **Name**
* recipeUses
  + ***Id***
  + ***ingName***
* recipeHas
  + ***Id***
  + ***catName***
* ingsHave
  + ***ingName***
  + ***catName***

# Hardware Requirements

The only hardware needed to run our app are phones that the app runs on. We are using Android Studio with an API of 26 to build our app so the phones that work with Android 8.0 and up will be able to run our app.

The hardware the client has, which is an Android phone, will be adequate for the input and output volumes our app will use. The client will not need to purchase any other hardware in order to use our app.

# Input-Output Specifications

For this application inputs will be taken in as scanned pdf form or user input text. When users input text they will be required to fill out the spaces corresponding to directions and ingredients. When users scan their recipes they will be asked to scan a physical copy of their recipe into the app to be saved as a pdf in the database. The scanned pdfs will be read for by the system so that the Ingredients and directions can be taken out of the original pdf. Outputs of the application will display all the recipes as a standardized format.

# Sizing

* 3 entities in ER Diagram
* 5 processes in Data Flow Diagram
* Each inputted file can be a range of 0 MB to 5 MB
* The volume of input depends on the frequency of how much the user uses the app. The volume will be higher when the user first downloads the app, but will become less frequent as the user finds new recipes.

# ER Diagram

Recipe {**ID**, Name, Instructions, Image}

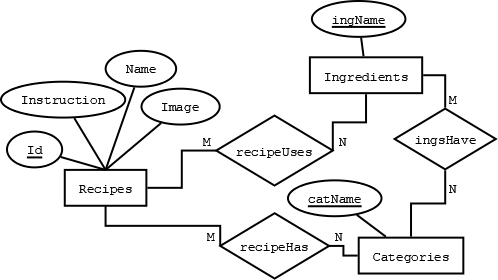
Ingredients {**Name**}

Category {**Name**}

recipeUses{***Id***, ***ingName***}

recipeHas{***Id***, ***catName***}

ingsHave{***ingName***, ***catName***}



The three entities within the database are recipes, ingredients, and categories. Both recipes and ingredients have categories, though their categories may differ slightly. A category is effectively going to be a search term, and most of the will apply to recipes. Examples are types of cuisine: Italian, Indian, Chinese...; type of meal: entree, appetizer, desert…; or meal classifications: vegan, vegetarian, lactose free. The classifications are what would apply to ingredients as well, for example lettuce is vegan, vegetarian, and lactose free among other things. A recipe that has all its ingredients sharing a category also falls into said category, though that logic will largely be decided beyond the scope of the database.