**Deliverable #3 Define Business Problem**

**Introduction:**

Team: CoffeeBreadSnek (Yes, this is correct, based on Discord usernames)

Members: Jakob Hoveland, Morgan Du Bois, Anna Rowena Waldron

Project: Inventory and Recipe Finder for Grocery Store

**Business Problem:**

The problem that we are trying to solve is a recipe finder and inventory finder for a grocery store. Many stores have an app or a function on their website that allows customers to find where various groceries are located within the store. Few, however, also have the functionality of finding recipes that grocery items could be put into. Many customers are not able to determine if a specific store has a necessary ingredient for a recipe they are making for sale, or if that product is in stock. This leads to frustrated customers not being able to find what they want, or having to travel to multiple locations and wasting time. This may cause the business to lose valuable customers.

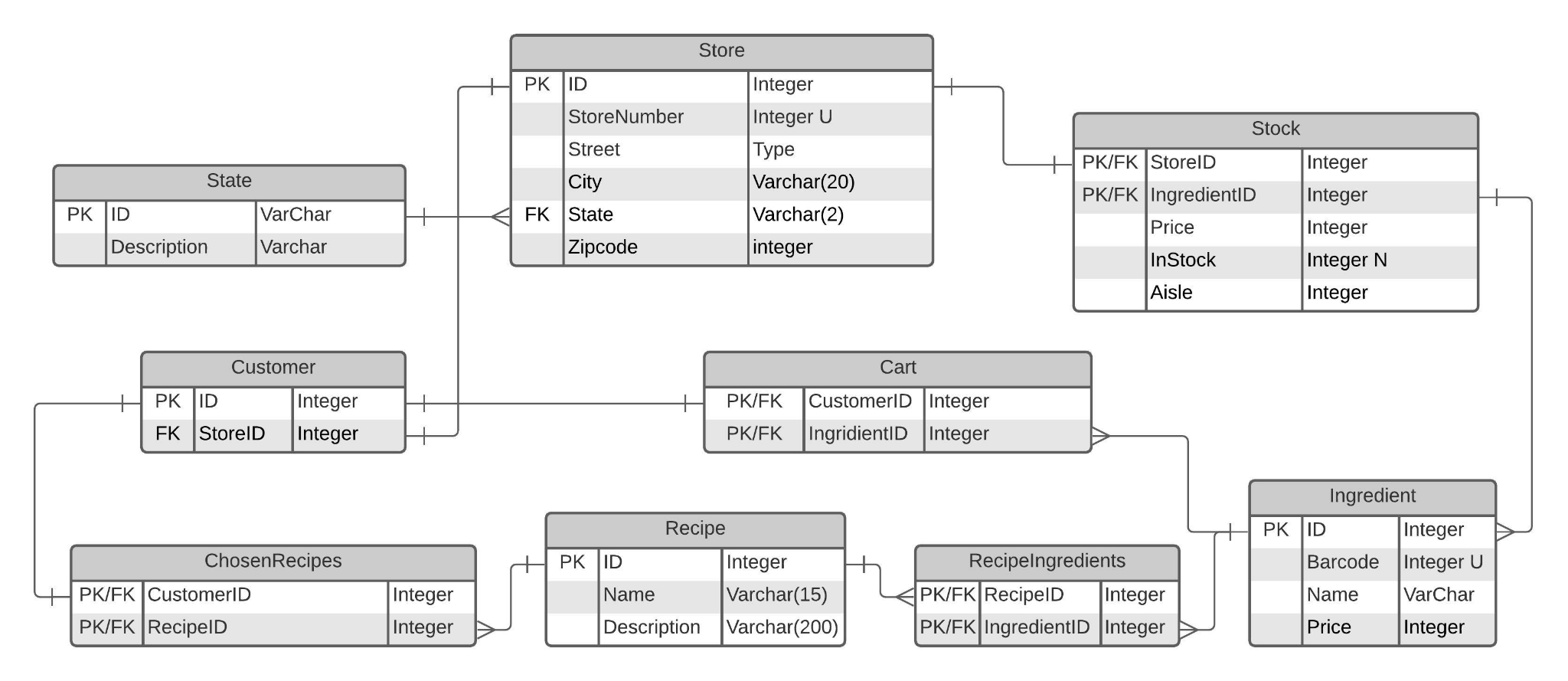
**High Level Interface:**

This app will be a tool for users and will not store information after the app is closed; no accounts will be made for the users and no actual purchases will be made through the app. It will be able to return a list of recipes that contain the food items the customer has chosen which are currently saved in the customer’s digital shopping cart. The customer may also select recipes instead of food items. The app will then provide a list of store locations within the given zip code given by the user, and if no store is within that zip code, it will return a list of all stores. Once a store has been selected, a list of items in the recipe selected is returned to the user, and information about those stock items (price, stock, location, etc.).

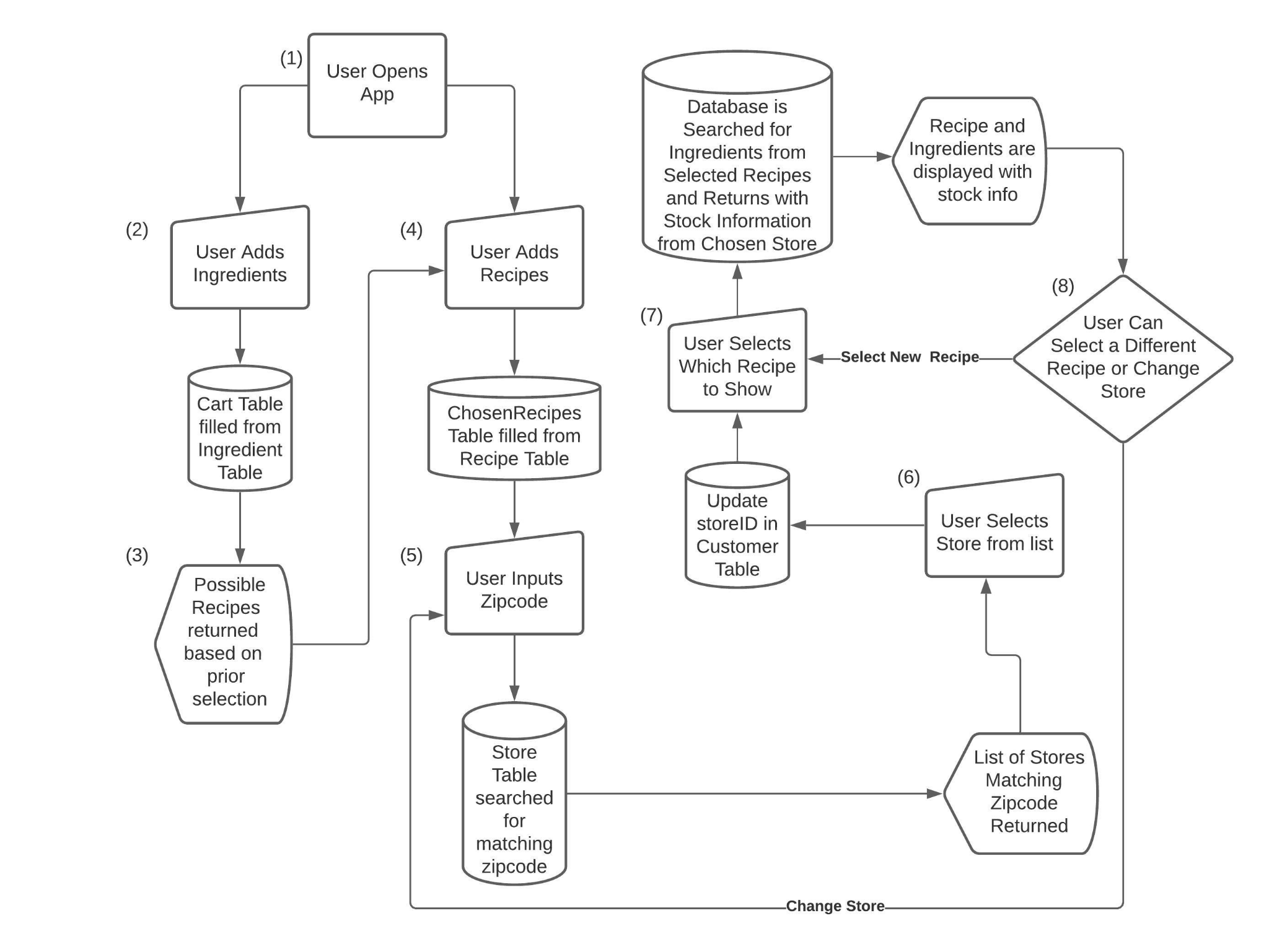
**User Interactions:**

* First, the customer inputs the ingredients they want to use in a recipe. Ingredients are first typed but a drop down bar is shown with potential options of that ingredient from a list of ingredients (e.g. onion would have a drop down bar of red onion, yellow onion, walla walla onion, ect.). A customer can also instead choose to put the name of a recipe in the search bar and a similar list will be shown with similar or like recipes, in that way selecting recipes instead of individual ingredients.
* Second, the customer is provided a list of recipes using the ingredients already provided by the customer, or the customer is provided a list of recipes based on their recipe selection from step 1.
* Third, the customer Inputs a zip code and will be shown a drop down bar of store addresses within that zipcode to choose from.
* Fourth, the customer can choose a recipe, which will return a list of the remaining ingredients that the customer did not list in addition to the listed items
* Fifth, the customer is shown the list of all ingredients in the selected recipe(s) (seperated for each recipe) with their corresponding store address, aisle number, price, and whether it is in stock or not
* Sixth, the customer can change their store selection to see if another store has the products that they are searching for by inputting a different zip code, and then selecting a different store

**Schema Design**



**User Interface Design**

****

**Refer to flow chart**

1. The user opens the application, and is presented with two search bars; one for inputting ingredients (2) and one for inputting recipes (4).
2. If the user uses the ingredient search bar, The user selects ingredients by typing the names of the ingredients they have or want to be used in a recipe (e.g. onion would have a drop down bar of red onion, yellow onion, walla walla onion, ect.). The user can add multiple ingredients but only one ingredient is searched for at a time. As every ingredient is added, the Cart table is updated with a new entry. These ingredients are used to find corresponding recipes in (3).
3. The user input is searched against the recipes table using a drop down bar of matching items to show the user possible recipe matches. This is accomplished by finding all entries in the Recipe Table that have ingredients that match the Cart table.
4. If the user uses the recipe search bar, the user can search for a specific recipe by name, or continue to add recipes based upon their ingredient search. When the user selects a recipe, the ChosenRecipe table is updated with a new entry linking it to the chosen Recipe table entry.
5. The user then inputs a zip code. A list of stores that are within that zip code is returned to the screen for the user to choose from. If there are no stores within the searched zip code, a message indicating that there are no stores in that location will be returned and a list of all stores will be given to the user. The input of zip code is searched against Store.zipcode for any matches, which are then returned.
6. From the returned lists of stores, the user can select a store. The list the user is given is of stores within that zip code along with information on where that store is located. The application then continues to (7). When a user selects a store, Customer.StoreID is updated with the Store.ID from the chosen table.
7. The selected recipes of the user are printed, and the user may then choose a recipe. Once a recipe is selected, the ingredient list for that recipe is displayed, along with the selected stores information on those ingredients. Based on the value stored in Customer.StoreID, a table is returned containing the information from the Ingredient table, joined onto the stock table with ingredientID, where Stock.StoreID is equal to Customer.StoreID.
8. Finally if the user finds that the store they have chosen does not have the items they want in stock, they can then change the store by clicking a change store button, or they can return to the list of selected recipes and find the stock information for the ingredients of another recipe by clicking a back to recipes button. Either the screen is cleared and the steps taken in the database at (7) are taken again, or the program loops back to (5) and continues from there.