Part 1:

JAYA

Nothing has changed.

Amber Mildenhall (apm2189), Jade Kaleel (jik2124), Yael Kelmer (yrk2107),

Alex Southwick (acs2313)

Language: Java, Platform: Mac, Repo: https://github.com/ambermildenhall/JAYA

Part 2:

Our service is an API called "Fridge API". Our API is geared towards users (individual people) and clients (applications and other services). The service can, in general, receive a list of ingredients that are in the user's fridge, return a modified list, and store the given list for future operations. Basic functionalities/features include the ability to read fridge contents, update fridge contents, add contents to the fridge, and delete contents from the fridge. Our API will also allow the user to create a "Core" ingredients list – a list of ingredients and their quantities that must be in their fridge at all times. When their fridge has reached a predetermined (by the user) amount of a "Core" ingredient, the API will alert the user to purchase more of it. Another functionality will be that, given the list of contents of their fridge, a user can request the API to return a recipe using only those ingredients. In addition, given a list of ingredients (recipe) and some preference (vegan, gluten-free, allergy, ingredient substitution) our API can return a modified list of ingredients that satisfies these requirements making the least changes. An additional functionality will be the ability for clients to see aggregate information about all of their users of the Fridge API, such as the most popular item in all fridges.

Various applications and services could utilize our API. For example, a supermarket app could use our fridge API to allow users to create lists of their fridge's contents and add "Core" ingredients to their shopping list. They could then tailor the ads they show in their app to the user based on what is in their fridge along with the aggregate data of their users to understand their customer's food habits and preferences. Other applications that can utilize our API are nutritional apps that calculate which food groups the user typically eats to determine which they need to eat

more/less of. Nutritional apps could also use the reminder function of our API to remind the user to buy more of a certain kind of food to stay within their nutritional goals. The recipe function of our API could be used by recipe websites to return recipes that users can make with the contents of their fridge. Another possibility could be a smart fridge that wants to recommend recipes to you based on the contents of your fridge.

Our service will accumulate data in the form of contents of users' fridges. The data will allow users to save the content of their fridge so they can view, update, delete, and add to it on demand. Our API will store the users' fridge contents, so even if the user restarts their application or website, the user's data will be stored. The data will also be used so that clients can view aggregated data about their users' most popular food items.

Part 3:

We will use Postman to create different calls/requests to the API. For example, a GET request to return the list of items in a user's fridge, POST to add an ingredient to a user's fridge, DELETE to remove an ingredient, and PUT to update an ingredient. A GET request could also be made to return the most popular ingredients in all user's fridges. In each of these example tests, we will compare the returned result or change to the expected result.

We will use Postman to make invalid API calls. We will check for these conditions in our code and send error messages to the users in these conditions that inform them of why the request was invalid and what they could do to fix it.

We will make calls to our API and check that the data stored in our database is aligned to what is expected at each step of the process. We can use an application such as DataGrip to manage our database, run queries, and view results.