JAYA

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

Alex Southwick (acs2313)

Part 1:

October 22nd, 2021

Part 2:

After our meeting with our TA we decided to reduce our scope from providing ingredient substitutions and recipes to just supplying recipes based on the ingredients in users' fridges. We will continue to have the alert functionality that alerts the user when they are missing "core" ingredients from their fridge. Every time the user calls the API to add or detract something from their fridge, our API will check if their fridge still meets their core requirements and alert the user if it doesn't. Our API will still save the user's fridge contents and persistent data. Other users will be unable to see each other's fridges and clients such as apps will not be able to see other clients' data.

As for clients that will use our API, our service can be used for nutritional applications that would like to suggest recipes for their users. It can also be used in supermarkets' apps that would like to help customers create their shopping list based on the contents of their fridge. An example of a user story would be a supermarket that wants to tailor their advertisement to their customers. The developer of the supermarket application would use our API to collect the aggregate data of what is in their user's fridges and what recipes they are using to target which advertisements their users see in their app.

User stories:

- As a supermarket, I want customers' fridge content so that I can advertise products to users better and make more informed purchasing decisions. My conditions of satisfaction are being able to know what my shoppers have in their fridges currently and what core ingredients my customers are lacking. A valid input would be a request for aggregate data of all of their user's fridges. The output would return the data of their user's fridges. An invalid input would be attempting to access the data of other customers who are not using their app. This would be prevented by auth0 authentication.
- As a nutritional application, I want recipe suggestions based on fridge content, so that my users can create healthy meals without needing to leave their home. My conditions of satisfaction are being able to return recipes to my user based on the ingredients in their fridge. Valid input would be a request for the contents of a particular user's fridge who is a member of the requesting app. The resulting output would be a list of recipes that utilize these ingredients. An example of invalid input would look like attempting to access a user's fridge that is not associated with the nutritional app. This would be prevented by auth0 authentication.
- As a smart fridge, I want to know the number and type of ingredients in my owner's fridge so that I can inform my owner when they need to buy new ingredients. My conditions of satisfaction are being able to alert my owner when certain ingredients count is less than the given threshold. A valid input would be a detraction of an ingredient from the user's fridge. This would trigger a function that would produce an output of an alert to the user that their fridge is low on a certain item. An invalid input would be a detraction of an item not in the user's fridge. This would produce an error that our API would return to the user and request that the user correct their input.

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 also do these tests from the perspectives of several different potential clients that can be identified using tools such as passcodes/keys.

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.

Part 4:

We will be using CheckStyle as our style checker for Java, SonarLint as our static analysis bug finder, JUnit as our test runner, and CodeCover as our coverage tracker as it is already integrated into JUnit. We will also be utilizing an outside API, Spoonacular, in order to have a list of recipes at hand when recommending meals a potential user can cook with what is already available within their fridge.