Planning Meeting

Database Planning

Existing System: -SSH camera, generate random data in java → put into database
-SSH Cloud Database
-Web API → will be simulated in the terminal

The process of how the app would work. Going from the SSH camera – to the cloud – to the app.

If we were to do it like that – when would we update the app? Once opened or daily.
Initially agreed on when opened. However, realised that if lots of users – it will slow it down but also for future implementations would be better. When an item is added to the fridge , we will work out the quality then.

How on earth is the data stored from the fridge camera? – Update Ben read the document it was actually really simple. The data will be stored as a fridge_ingrediant table and then a cloud table.

Within the fridge table we will include:

“tenants_fridge_item”:
tenants_id
Fridge_item_id
quantity
date_time
quality_rating

“fridge_item”:
Fridge_item_id
Price
Name
Estimated_Shelf_Life

“tenants”:
tenants_id
tenants_name

“recipe”:
recipe_id
recipe_name
recipe_instructions

(Possibly)” Recipe Log Book:
The idea is that we don’t want to be repeating the same recipes every time to the same user so it would include new fields with recipe_id and recipe_rating. But as this isn’t a key feature we have decided to save it for if we have extra time at the end.

Calculations Planning

Changing the calculations as we have come across some problems. The intention is to use the items that are going out of date rather than prioritising the fresher items (which is what the calculations do as of now)
New conditions being implemented too.

M = (Sum of U/ N)

M: Recipe match rating, ranging from 0-100%
n: Number of fridge ingredients
U: If ingredient unavailable or Q=0 then 0
else, U = 1 – Q
Q: Quality of the ingredient, ranging from 0-100%

Q = (1 - Days since captured/Estimated Shelf Life + 1)

User-Interface Plan

A table of recipes. Ordered by the M number
(Possibly allowing them to see their ingredients with quality scores)

Work Plan

Database Unit(Docker): Josh
Manual population of items: Arman
Recipes: Ben
Logic to randomly generate items: Arman
Logic to calculate Quality (and put data into database): Jamal
Logic to get it out the database and calculate and M value: Jamal
Web API: Josh
JavaFX: Ben