# SW Engineering CSC648/848 2019 – Team 102

# Milestone 1 – 2/25/2020

PROJECT – DIGITAL INVENTORY RECOLLECTION TERMINAL

COLLABERATORS

**Team Leader**

Sanchit Joshi

**SCRUM Master**

Aurpon Gupta

**Back End Leader**

Zhifan Cai

**Front End Leader**

Anthony Wong

**Git Master**

Matthew Davis

**Database** **Developer**

Daniel Godfrey

**Server Developer**

Christian Melendez

**Implementation Developer**

Dmitry Polozov

REVISIONS TABLE

|  |  |
| --- | --- |
| Version | Summary |
| 1 | Initial Document Creation |
|  |  |
|  |  |
|  |  |
|  |  |

1) Revised Data Definitions

# Overview

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| **Data Definitions** | | | | | |
| Collection "perishables" |  |  |  |  |  |
| Field | Collection | Data Type | Description | Example | Database |
| name | perishables | String | Itedentifier of a perishable food product that would be stored in a fridge |  | Mongo DB |
| calories | perishables | Double | Calories in a food item or recipe | 100 | Mongo DB |
| protein | perishables | Double | Protein in a food item or recipe | 12 | Mongo DB |
| sugar | perishables | Double | Sugar in a food item or recipe | 2 | Mongo DB |
| carbs | perishables | Double | Carbs in a food item or recipe | 19 | Mongo DB |
| fat | perishables | Double | Fat in a food item or recipe | 2 | Mongo DB |
|  |  |  |  |  |  |
| Collection "non-perishables" |  |  |  |  |  |
| Field | Collection | Data Type | Description | Example | Database |
| name | non-perishables | String | The name of the non-perishable good | "Paper Towels" | Mongo DB |
|  |  |  |  |  |  |
| Collection "users" |  |  |  |  |  |
| Field Name | Collection | Data Type | Description | Example | Database |
| username | users | String | Unique Identifier of App user | SanFranAmy7 | Mongo DB |
| family | users | Array | List of Users in a user family | {JohnM65,Kim43,  Matty} | Mongo DB |
| familyName | users | String | Name of Family | "Hernandez" | Mongo DB |
| profilePic | users | String(URL) | Photo of User displayed in home screen and when searching users |  | Mongo DB |
| recipes | users | Array of strings | List of recipes that were starred as favorites | "cobb salad" | Mongo DB |
| dailyNutrition | users | Array of objects | An array of documents with each days nutrition. |  | Mongo DB |
| dailyNutrition/dailyCalories | users | Double | Total calories consumed on a specific day | 1960 | Mongo DB |
| dailyNutrition/dailyCarbs | users | Double | Total carbs consumed on a specific day | 4 | Mongo DB |
| dailyNutrition/dailyProtein | users | Double | Total protein consumed on a specific day | 34 | Mongo DB |

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| dailyNutrition/dailyFat | users | Double | Total fat consumed on a specific day | 2 | Mongo DB |
| dailyNutrition/dailySugar | users | Double | Total sugar consumed on a specific day | 9 | Mongo DB |
| dietaryRestrictions | users | Array | List of Restrictions in diet | {Gluten Free, Vegan} | Mongo DB |
| restriction | users | String | Specific Restriction | "Gluten Free" | Mongo DB |
| notifications | users | Array of objects | Array of objects of notifications | "Milk about to expire" | Mongo DB |
| notifications/sender | users | String | Name of sender | "admin" | Mongo DB |
| notifications/text | users | String | Text of the notification | "Hello welcome to  Dirt" | Mongo DB |
| notifications/date | users | Double | Date in miliseconds since January 1st 1970 | 234830293239 | Mongo DB |
| workouts | users | Array of Objects | An array of different workouts and when they were done |  | Mongo DB |
| workouts/workoutName | users | String | The name of the workout | Running | Mongo DB |
| workouts/workoutDate | users | Double | The date of the workout | 20939200 | Mongo DB |
| workouts/caloriesBurned | users | Integer | Number of calories burned over a  certain period of time or after a  workout | 230 | Mongo DB |
| shoppingList | users | Array of strings | List of groceries to buy | {milk, eggs, cereal} | Mongo DB |
|  |  | String | Item on list | "milk" | Mongo DB |
|  |  |  |  |  |  |
| Collection "recipe" |  |  |  |  |  |
| Field Name | Collection | Data Type | Description | Example | Database |
| recipeName | recipe | String | The name of the recipe | "Teriyaki Chicken" | Mongo DB |
| ingredients | recipe | Array of Objects | An array of food items and their amount |  | Mongo DB |
| ingredients/ingredientAmount | recipe | Double | Amount of food item in ingredients in grams | 100 | Mongo DB |
| ingredients/ingredientName | recipe | String | Name of food item in ingredients | "Chicken" | Mongo DB |
| instructions | recipe | Array | Instructions, in order, for the recipe |  | Mongo DB |
| instructions/instruction | recipe | String | Instruction for a recipe | "Cook chicken in pan" | Mongo DB |
| recipeCalories | recipe | Double | Amount of calories in a recipe in grams | 200 | Mongo DB |
| recipeCarbs | recipe | Double | Amount of carbs in a recipe in grams | 2 | Mongo DB |

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| recipeProtein | recipe | Double | Amount of protein in a recipe in grams | 9 | Mongo DB |
| recipeFat | recipe | Double | Amount of fat in a recipe in grams | 34 | Mongo DB |
| recipeSugar | recipe | Double | Amount of sugar in a recipe in grams | 23 | Mongo DB |

2) Revised Functional Requirements

# User controlled functional requirements for Product (and priority 1-4)

1. Users must be able to freely add/remove items into the fridge without the use of the receipt (4)
2. Users must be able to create their own meal plans (2)
3. Users must be able to create shopping lists by selecting recipes (3)
4. Users must be able to create shopping lists in order to fulfill a meal plan (2)
5. Users must be able to ask for reports about their consumption, and must be able view the report from any nutrition category (how many calories, how much protein, how many carbs etc.) (4)
6. Users must be able to view the inventory/contents of their fridge at any given time (4)
   1. Users must be able to sort their inventory by nutrition categories. (2)
7. User must be able to search for recipes that make use of their current inventory (1)
8. User must be able to specify how much of a food item has been consumed (2)

**Autonomous functional requirements for Product ordered (and priority 1-4)**

1. Product must be able to enter contents of a receipt into the inventory by means of barcode scan (3)
2. Product must be able to enter contents of a receipt into the inventory by means of OCR (4)
3. Upon reading a receipt, Product must be able to discern the category of its contents by food/non-food, perishable/non-perishable, type of item(poultry, produce, dairy etc.), and quantities of items. (3)
4. Must be able to distinguish the type of item upon entry (3)
   1. Product must keep a record of the date of purchase/entry of every item (3)
   2. Product must keep a record of expiration of items if applicable (3)
5. Must be able to retrieve nutritional information about current inventory items by referencing a database of all Costco grocery items and their nutritional information (3)
6. Must be able to retrieve nutritional information of items specified in recipes or meal plans by referencing a database of all Costco grocery items and their nutritional information (2)
7. Product must allow the addition of other Users under control of the main User’s profile (1)

**Other functional requirements for Product**

1. Product must pull data of all Costco grocery items and their nutritional information (4)

3) UI Mockup and Storyboard

# Wire Frames

We have created two wireframe models to suit possible users. Each is comprised of different features that our product can offer to the user. Providing that our application is both suitable to their needs and easy to use.

The site is linked below. Using the arrow at the bottom or by clicking on the actual slide will bring you through the presentation. In addition to that, we have also included functioning components denoted by the pointer arrows on the slide which you can also use to forward the slides.

[https://www.figma.com/proto/7YPwRgr9CJwxXyMgnx1Red/UserStories?node-id=31%3A432&scaling=min-zoom](https://slack-redir.net/link?url=https%3A%2F%2Fwww.figma.com%2Fproto%2F7YPwRgr9CJwxXyMgnx1Red%2FUserStories%3Fnode-id%3D31%253A432%26scaling%3Dmin-zoom)

[https://www.figma.com/proto/7YPwRgr9CJwxXyMgnx1Red/UserStories?node-id=1%3A2&scaling=min-zoom](https://slack-redir.net/link?url=https%3A%2F%2Fwww.figma.com%2Fproto%2F7YPwRgr9CJwxXyMgnx1Red%2FUserStories%3Fnode-id%3D1%253A2%26scaling%3Dmin-zoom)

# GUI Design

We have extrapolated what we have viewed as workable components within the user stories and designed a prototype of a functional application for a phone web page. Using an iphone 8 as a template, we have made multiple walkthrough states.

The presentation displays the various functionalities of our web application step by step and explains in detail how to use it.

[https://www.figma.com/proto/VgN8nglI84osbv2ZuCrqn9/Prototype?node-id=251%3A5222&scaling=contain](https://slack-redir.net/link?url=https%3A%2F%2Fwww.figma.com%2Fproto%2FVgN8nglI84osbv2ZuCrqn9%2FPrototype%3Fnode-id%3D251%253A5222%26scaling%3Dcontain)

The next model will bring you to the starting screen of our web application as a full prototype equipped with all of the functionalities presented, with transition states included.

[https://www.figma.com/proto/VgN8nglI84osbv2ZuCrqn9/Prototype?node-id=191%3A2&scaling=scale-down](https://slack-redir.net/link?url=https%3A%2F%2Fwww.figma.com%2Fproto%2FVgN8nglI84osbv2ZuCrqn9%2FPrototype%3Fnode-id%3D191%253A2%26scaling%3Dscale-down)

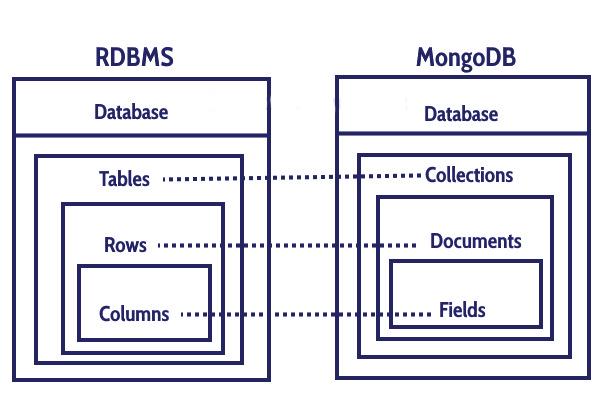
4) High Level Architecture and Database Organization

**Database Organization**

The main database we are using for our project will be a MongoDB database that uses noSQL. While SQL databases organize data relationally which can be visualized as a table, noSQL organizes data in collections and documents. In noSQL, a database entry does not require a specific scheme meaning documents held in a related collection can have different formats and data types. While our database has this functionality, the structure of our data allows us to still visual the data in a table as will be shown.

**General MongoDB**

MongoDB organizes information using 4 containers. Databases, Collections, Documents, and fields.



A MongoDB Documents may look something like this:

{

\_id = “9438jf0029f”,

name = “Daniel”,

height = 6.2,

age = 26,

schools = [“SFSU”, “CSM”, “CCSF”]

}

**D.I.R.T**

**General Description**

The bulk of the data for our product, Dirt, will be held in our MongoDB Atlas database, “fridge”. The nutritional data will be fetched from a third part open source database, FoodData Central, created by the USDA. Our “fridge” database will hold different collections which can be visualized as tables in a standard relational data table. Each user’s information will be stored into one document. This document will hold all information unique to that particular user. Another collection in “fridge” will be the “recipe” collection. Each document in “recipe” will represent one recipe that users can search through.

**User**

The collection “user” holds a bulk of the data for “Dirt”. While mongo databases create a unique object ID to be created with each document, each user will have their own unique username in the field “username”.

Name

In the field “name” the user's real name will be held. No need for uniqueness.

User photo

In field “profilePic”, a string containing the URL of a public hosted picture is held.

Fridge Contents

The field “fridge” holds an array of documents that denote a particular food in a user’s fridge with its name, expiration, and amount.

Food Name

The field “food” holds a string of the name of a particular food in a user’s fridge.

Expiration

The field “expiration” holds a long integer that denotes that date of expiration using milliseconds since January 1, 1970 GMT.

Amount of Food

The field “amount” holds the amount of food in grams as nutrition is calculated by 100 gram servings.

Receipts

The field “receipts” holds an array of URLs where photos of past receipts are hosted.

Recipes

The field “recipes” holds an array of strings with names of recipes that the user has favorited or created.

Daily Nutrition

A user can record the nutrients they consumed each day. Those values are held in the array “dailyNutrition” that holds documents, each representing a single day's nutrition.

Record Date

The field “date” hold the date for that days nutrition

Calories

The field “dailyCalories” holds the total of the calories consumed that day.

Carbs

The field “dailyCarbs” holds the total of the carbs consumed that day.

Protein

The field “dailyProtein” holds the total of the protein consumed that day.

Fat

The field “dailyFat” holds the total of the protein consumed that day.

Sugar

The field “dailySugar: holds the total of the sugar consumed that day.

**Recipes**

The collection “recipe” will be entirely dedicated to user created recipes. Users will be able to create recipes and save them. Nutritional values can be added manually or automatically calculated if none are added. Each document in “recipe” represents one recipe and the Nutritional values of the recipe.

Recipe Name

The field “recipeName” is a string of the user created name of the recipe.

Recipe Creator

The field “creator” will have the username of the user who created the recipe so that when a user searches through the user created recipes, their recipes are given first in the results

Access

The field “access” will mark whether the public has access or not.

Ingredients

The field “ingredients” will hold an array of documents that hold the ingredient name and the amount of the ingredient.

Ingredient Name

The field “ingredientName” will hold a string with the name of the ingredient.

Ingredient Amount

The field “ingredientAmount” will hold the amount of the ingredient in grams.

Recipe Calories

The field “recipeCalories” will hold a double of the amount of calories in the recipe.

Recipe Carbs

The field “recipeCarbs” will hold a double of the amount of carbs in the recipe.

Recipe Protein

The field “recipeProtein” will hold a double of the amount of protein in the recipe.

Recipe Fat

The field “recipeFat” will hold a double of the amount of fat in the recipe.

Recipe Sugar

The field “recipeSugar” will hold a double of the amount of sugar in the recipe.

**Functional Use of Database**

**User Profile**

When a person visits our site they will be prompted to create an account. A form will collect the users information and upload it to the database through a JSON object using a pre-built mongoose schema. The database will create a new document for every new user. Their username will be unique throughout the database.

**User Fridge Addition**

When a receipt is scanned the product name and amount will be gathered. Those values will be automatically added to the users “fridge” array in the users document. A user can also add items to their “fridge” manually through a page that prompts the user for the name of the food, the amount and the expiration date. This is packaged into a JSON object and added to the “fridge” array.

**Adding Food to Daily Nutrition**

A user can update their daily nutrition in the ways; directly by inputting an amount of calories, carbs, protein, sugar, and fat which would be added to the users “dailyNutrition” values, by searching for a meal or food and inputting the amount consumed, or by choosing from a food already in the users “fridge”. The search will query a third party site, FoodCentral Database, with the name of the food. That will return a JSON document with the nutritional values of the food. Those values are based on 100g of the found food item so a conversion using the amount of food consumed will take place before adding the nutritional values to the “dailyNutrition” array.

**Recipes**

When a user wants to create a recipe they will go to a page that will prompt them for the ingredients, the amounts per ingredients, and the instructions in order. The recipe will be either marked public or private by the user. All recipes, private and public, will be added to the “recipes” collection but will be marked private or public and will only be available for the creator of the recipe if private. The recipes document ID will be stored in an array in “recipes” in the users document.

**Perishables**

In order to differentiate a good that needs to be refrigerated and those that do not, a base food type will be derived from each refridgearable food good and checked against the “perishables” collection which is a list of all goods that require refrigeration. The foods in “perishables” will be without brand name, so instead of Heinz, the database would have a document called “ketchup”. If the product being scanned from a receipt does not have a type that appears in perishables it will automatically be deemed unrefrigerated.

**Third Party Databases**

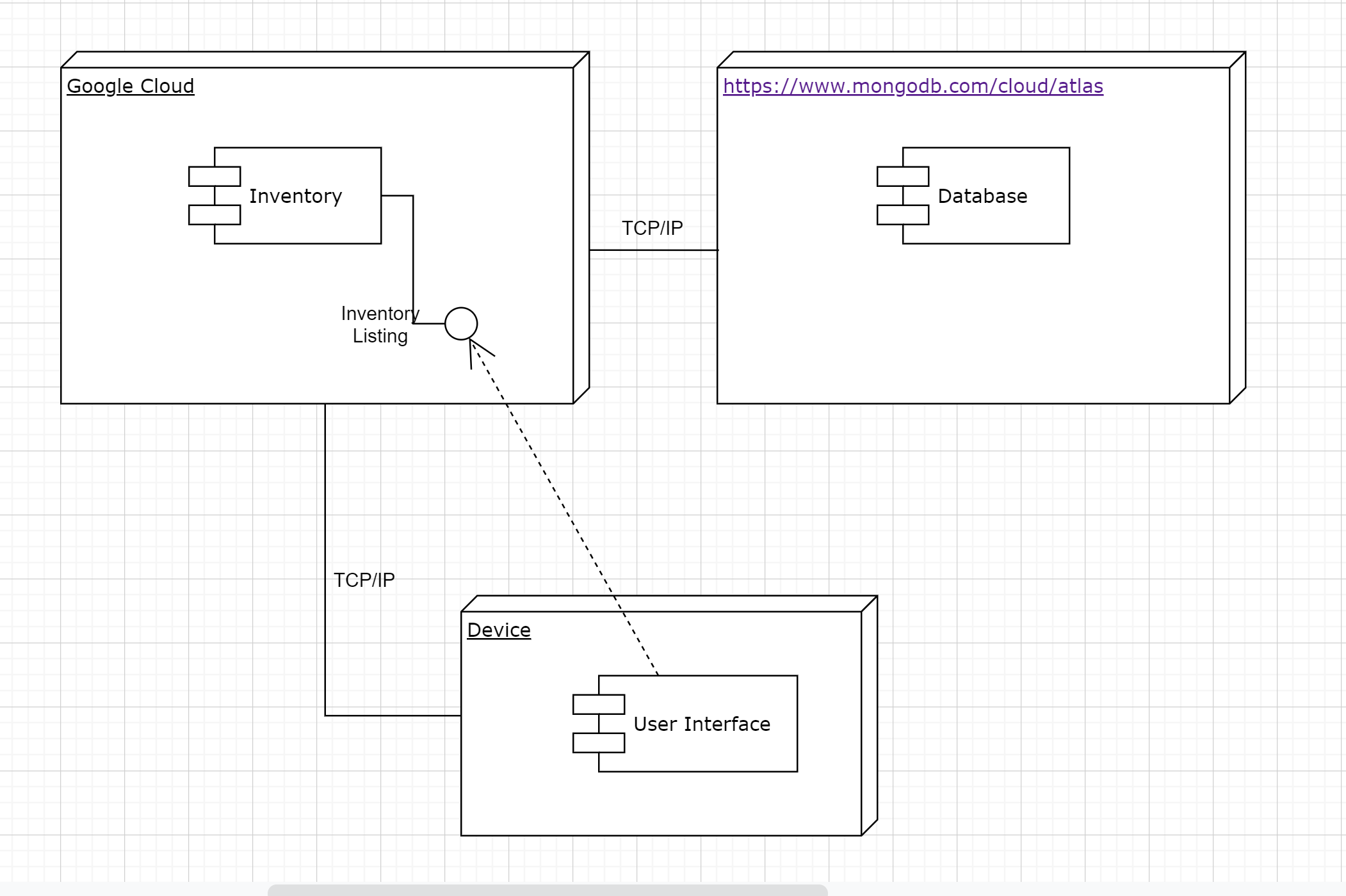
**FoodCentral Database**

FoodCentral Database is a database of both general food items and branded products. The database is curated by the USDA. The database returns information through url queries.

**Edamam**

Edamam is a recipe database that provides an API for querying instructions and ingredients for different meals. This database will be used along with our collection “recipe” with user created recipes. Results from both databases will appear to users when queried.

5) High Level UML Diagrams



6) Key Risks

**Risk: Inconsistent receipts between stores.**

Not all stores in a chain have consistent receipt markings for products, which can lead to issues with accessing outside databases for identifying perishable items. Our solution currently is to use a store that is consistent (Costco) while building up our own internal database of items that can correlate known shorthands to known items.

**Risk: Using outside databases.**

Using external databases and databases specifically from large companies can cause legal issues. Crediting where information comes from is a way to combat this, and slowly moving off of outside databases while we expand out own is another.

**Risk: Scheduling team meetings.**

As our team is made entirely of students, some of whom also work while attending classes, scheduling conflicts can arise quite easily. Good communication and time management from our Scrum Master has helped smooth out these conflicts and allowed for the finding of a standard meeting time, as well as additional meeting times when necessary.

7) Project Management

Our team members have dedicated tasks assigned for each milestone categorized under frontend, backend and server deployment. For every milestone we have an internal deadline and an external deadline i.e milestone submission to Teaching Assistant and Professor.

We have team meetings every Monday and Wednesday in which we discuss the progress of each task assigned and any issues- technical, skills or infrastructure related are addressed. If tasks have been completed, we review the work done over the meeting and provide feedback if anything needs to be changed.

For Milestone 2, there were three major tasks to complete namely UI mockups, Vertical Software prototype and UML diagrams.

**UI mockups:-**

UI mockups were designed using Figma and the front-end developer and lead self-organize the tasks and review the work on every internal deadline.

**Vertical software prototype:-**

The vertical software prototype was a major task in Milestone 2 and the backend lead assigned tasks individually and reviewed the tasks assigned at every internal deadline.

**UML diagrams:-**

When the backend team was developing the software prototype, the basic skeleton of the backend was developed which encompassed some basic classes and functions that we would need to complete the application prototype. So, we could develop class diagrams, component and deployment diagrams for the application.