Cheap Staples

System Design Specification

Jan Bermudez(jb), Garret Bunkers(gb), Dylan Hopper(dh), Tim Nadeau(tn), Patrick Rodriguez (pr)

Table of Contents

1. SDS Revision History

Date	Name	Revision Description
2/25/2023	Timothy Nadeau	Added sections: System Overview, Display Page Module
2/26/2023	Patrick Rodriguez	Added sections: Software Architecture
2/26/2023	Timothy Nadeau	Updated Software modules section names & figures, Added section: Dynamic Models (Use Cases)
2/28/2023	Timothy Nadeau	Updated file formatting
2/28/2023	Patrick Rodriguez	Added sections: LPID (MySQL Database)
2/28/2023	Jan Bermudez	Added sections: Web Scraper Module, Database Gathering Module

2. System Overview

Cheap Staples is a software designed with the intent to provide a good way for low-income people to find groceries they need at a good price. The system will allow the user to select from a choice of select foods and compare prices from store to store nearby.

This system will use various modules to acquire and display this data, including: a database, data processing module, user interface, and web scraper. The system will utilize a web scraper to find grocery prices from select stores, and an online database to store all the prices in. From there, the user can request the data on the database to be updated at any point, but may take a while whilst the scraper scrapes. Then whether or not the user updated the data, the system will use the user interface to pass an input to the data processing module which will grab the current data from within the database, and display it on their screen.

3. Software Architecture

The software architecture of *Cheap Staples* is decomposed into four modules:

1. **Grocery Store Web Scrapers:** The BeautifulSoup python library will aid in traversing the HTML code of different grocery store websites to check for the availability and price of a given ingredient to insert into the Location-Price-Ingredient database.
2. **Location-Price-Ingredient Database:** A MySQL database which holds available staple ingredients and prices for each grocery store location in Eugene, Oregon. It returns appropriate data entries to the Database Gathering Module defined by the user in the Ingredient Chooser and Location-Price Interface.
3. **Database Gathering Module:** Responsible for retrieving user specified ingredients to be displayed in the Ingredient Chooser and Location-Price Interface. For each ingredient it will also execute MySQL queries to retrieve appropriate location-price data entries from the Location-Price-Ingredient database to be displayed to the user.
4. **Ingredient Chooser and Location-Price Interface:** Interactive interface responsible for providing user valid options of ingredients to choose from and displaying the location and price that each user-defined ingredient is available at.

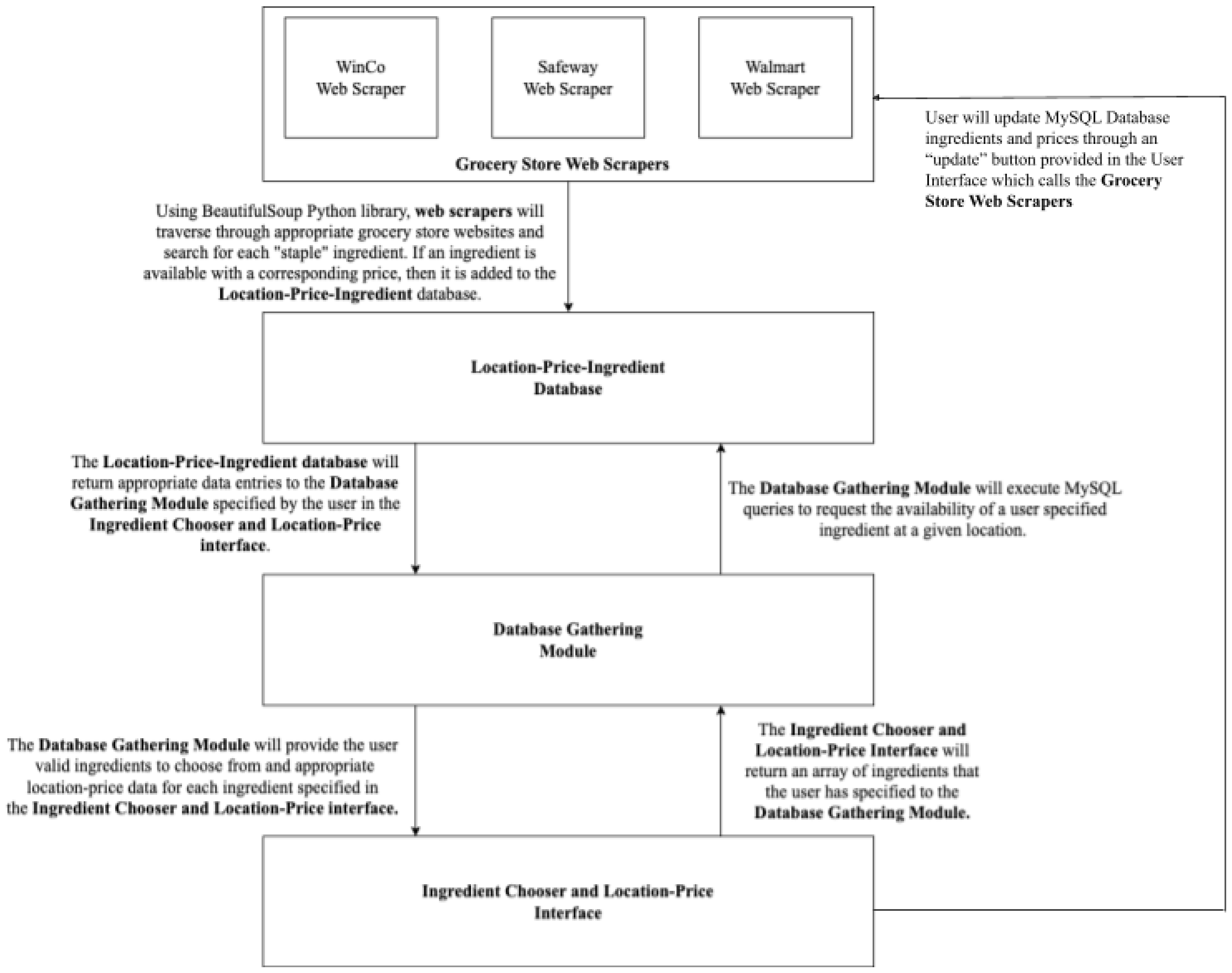
WinCo Web Scraper
Safeway Web Scraper
Walmart Web Scraper
Grocery Store Web Scrapers
User will update MySQL Database ingredients and prices through an "update" button provided in the User Interface which calls the Grocery Store Web Scrapers
Using BeautifulSoup Python library, web scrapers will traverse through appropriate grocery store websites and search for each "staple" ingredient. If an ingredient is available with a corresponding price, then it is added to the Location-Price-Ingredient database.
Location-Price-Ingredient Database
The Location-Price-Ingredient database will return appropriate data entries to the Database Gathering Module specified by the user in the Ingredient Chooser and Location-Price interface.
The Database Gathering Module will execute MySQL queries to request the availability of a user specified ingredient at a given location.
Database Gathering Module
The Database Gathering Module will provide the user valid ingredients to choose from and appropriate location-price data for each ingredient specified in the Ingredient Chooser and Location-Price interface.
The Ingredient Chooser and Location-Price Interface will return an array of ingredients that the user has specified to the Database Gathering Module.
Ingredient Chooser and Location-Price Interface

Figure 1: Modular diagram of *Cheap Staples* software architecture.

4. Software Modules

4.1 Grocery Store Web Scrapers

a. Role and primary function

The role and primary function of the grocery store scrapers is to collect the availability and price for the system's selected staple ingredients.

Web scrapers will traverse its respective grocery store website searching for staple ingredients

b. Interface to other modules

Interacts mainly with the Location-Price-Ingredient Database. For a given grocery store, the web scraper will format and input the name of the ingredient, the name of the grocery store, and its corresponding price into the database.

The scrapers will be activated via the user interface. When the user wants to update the ingredient information, the user will have the option to click a button on their screen to gather new information before any pricing and location information is displayed.

c. A dynamic model

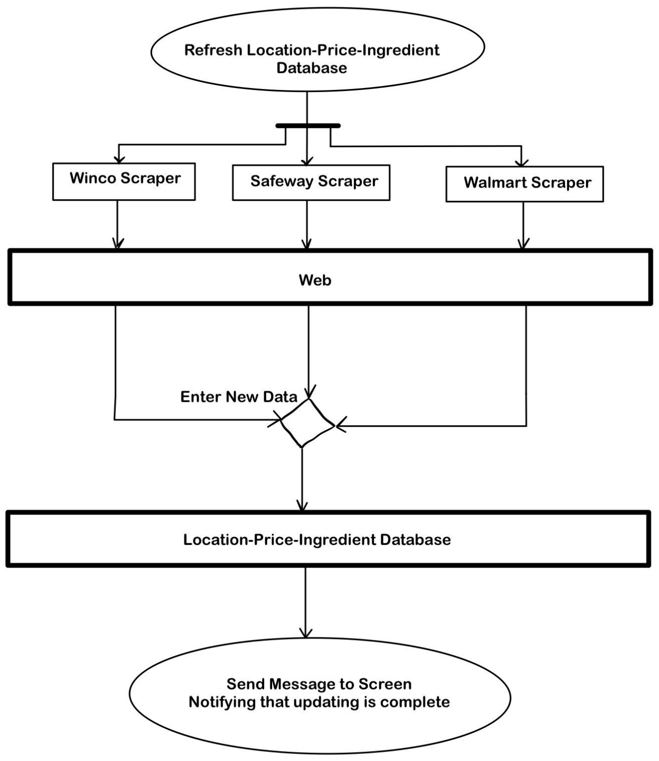
Refresh Location-Price-Ingredient Database
Winco Scraper
Safeway Scraper
Walmart Scraper
Web
Enter New Data
Location-Price-Ingredient Database
Send Message to Screen
Notifying that updating is complete

Figure 2: An activity diagram showing the "update ingredient database" use case

d. Design rationale

Manually traversing every grocery store's website to look for ingredients and inputting their respective values into the database would be tedious and cumbersome. The grocery web scraper is fundamental to streamline the process of finding affordable staple foods in Eugene, OR. The design choice of creating a web scraper for each individual grocery as opposed to a single universal one was made due to how the BeautifulSoup library works and the varying differences in each grocery store website's HTML. This trade-off was justified by time restrictions placed on the implementation of the system.

4.2 Ingredient Chooser and Location-Price Interface

a. Role and primary function

1) Allow the user to easily input their specifications
2) Take input from the user, and pass the given arguments to the data processing module to retrieve from the database
3) Display output to the user's screen

b. Interface to other modules

Communicates closely with the data processing module as it gives arguments for what it needs, then takes the output of the data processing module to display to the user's screen.

Can also send a ping to the scraper module to do a scraping and update data in the database

c. Static model

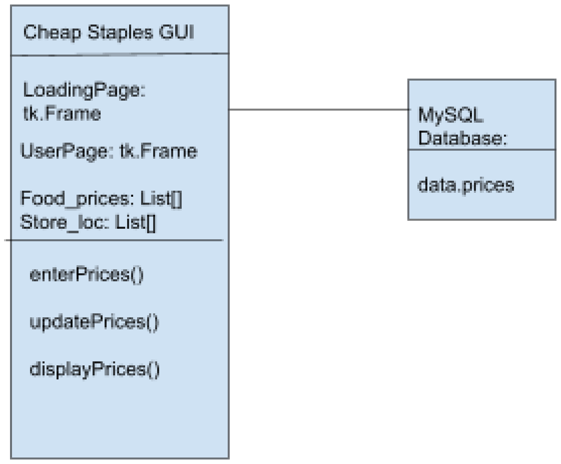

Figure 3: Static model of Display page module

d. A dynamic model

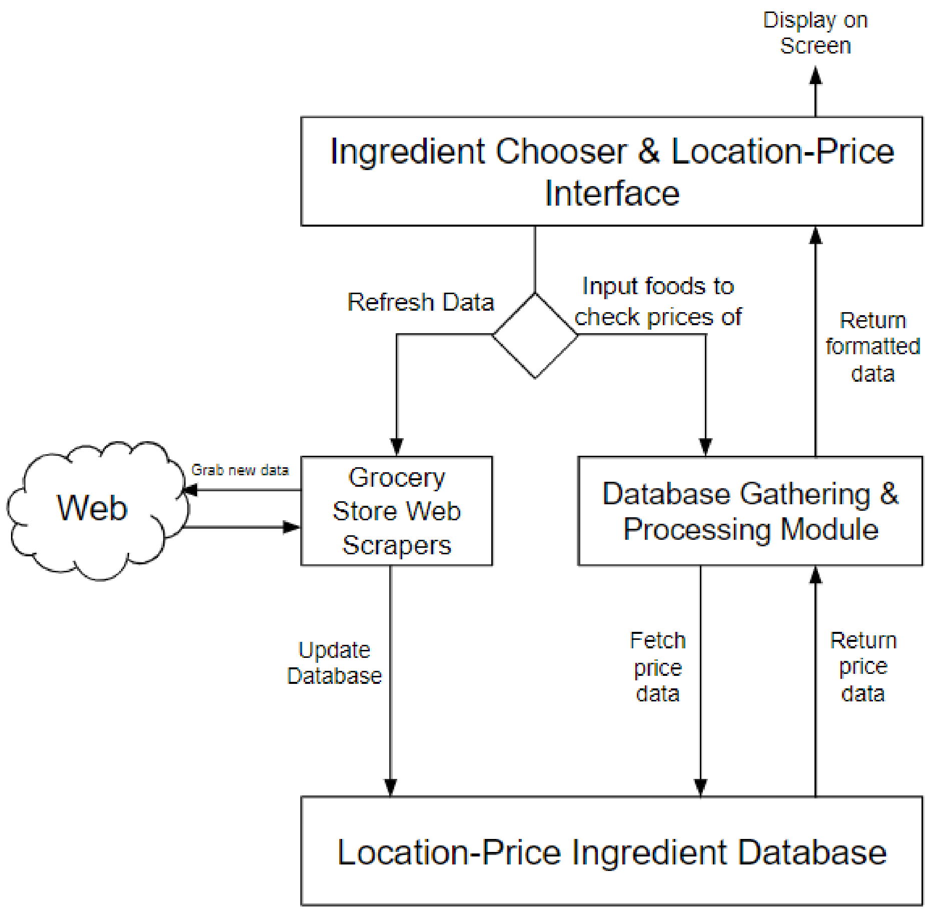
Display on Screen
Ingredient Chooser & Location-Price Interface
Refresh Data
Input foods to check prices of
Return formatted data
Web
Grab new data
Grocery Store Web Scrapers
Database Gathering & Processing Module
Update Database
Fetch price data
Return price data
Location-Price Ingredient Database

Figure 4: Dynamic model & behavior for the Display Page Module

e. Design rationale

Having a user interface makes inputting data easier and more intuitive for the user, increasing ease of use of the system overall.

4.3 Location-Price-Ingredient Database (MySQL)

a. Role and primary function

The Location-Price-Ingredient MySQL Database serves to contain all necessary information required to display the user *Cheap Staple's* available ingredients to choose from alongside their corresponding price and location.

b. Interface to other modules

The Location-Price-Ingredient Database will communicate with the Database Gathering Module by returning necessary data entries from executed MySQL queries to be formatted and displayed to the user.

The Grocery Store Web Scrapers will update the MySQL database upon first population and when a user presses the "update prices" button.

c. Static model

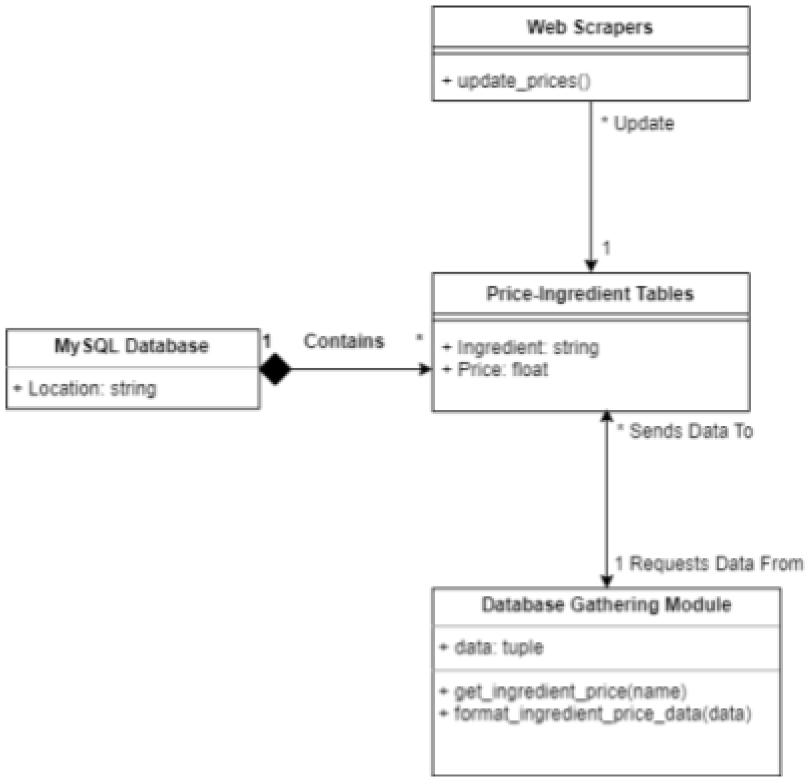

Figure 5: Class Diagram Describing Characteristics of Location-Price Ingredient MySQL Database and Interactions with Other Modules

d. A dynamic model

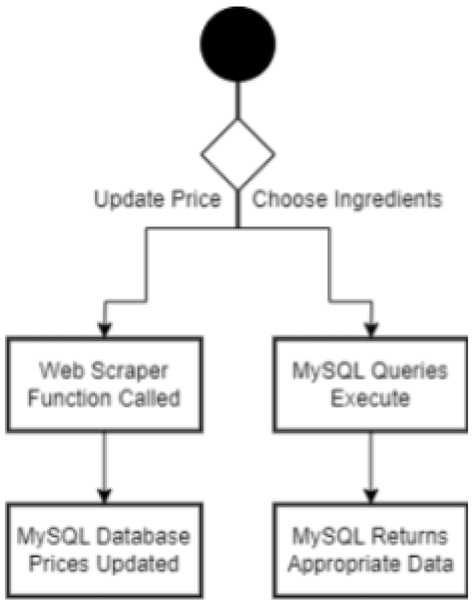
Update Price
Choose Ingredients
Web Scraper Function Called
MySQL Queries Execute
MySQL Database Prices Updated
MySQL Returns Appropriate Data

Figure 6: Activity Diagram Showcasing Different Behaviors of Location-Price-Ingredient Database

e. Design rationale

Having a database stored on the cloud would save a lot of time in computing and scraping. Since scraping can take minutes at a time, we want a solution that'll quickly load up the most recent data.

4.4 Database Gathering Module

a. Role and primary function

The role and primary function of the database gathering module is to act as an intermediary between the Ingredient Chooser and Location-Price Interface(ICLP) and location-price-ingredient(LPI) database.

The database gathering module serves to receive data to format it for ease of use, increasing modularity and comprehension of the system code.

b. Interface to other modules

The database gathering module will receive a list of user chosen ingredients from the Ingredient Chooser and Location-Price Interface to make queries to the LPI database.

The database gathering module will receive data entries from the LPI database which will then be used to construct "Table" objects for each selected item and be sent back to the ICLP Interface.

c. Static model

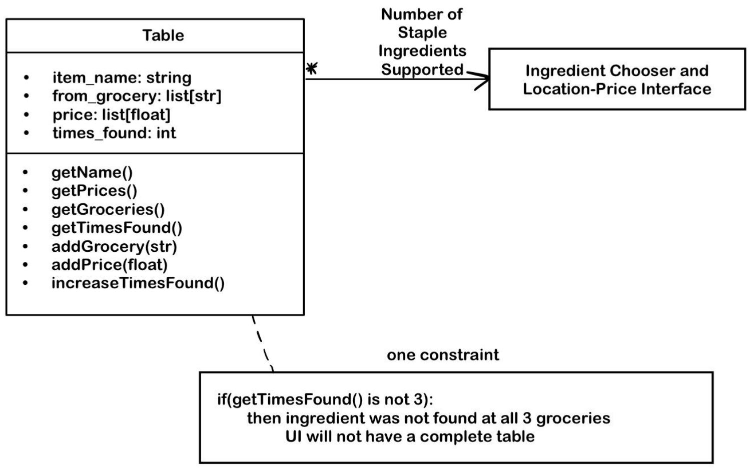
Table
item_name: string
from_grocery: list[str]
price: list[float]
times_found: int
getName()
getPrices()
getGroceries()
getTimesFound()
addGrocery(str)
addPrice(float)
increaseTimesFound()
*
Number of Staple Ingredients Supported
Ingredient Chooser and Location-Price Interface
one constraint
if(getTimesFound() is not 3):
then ingredient was not found at all 3 groceries
UI will not have a complete table

Figure 7: A class diagram for Table object

d. A dynamic model

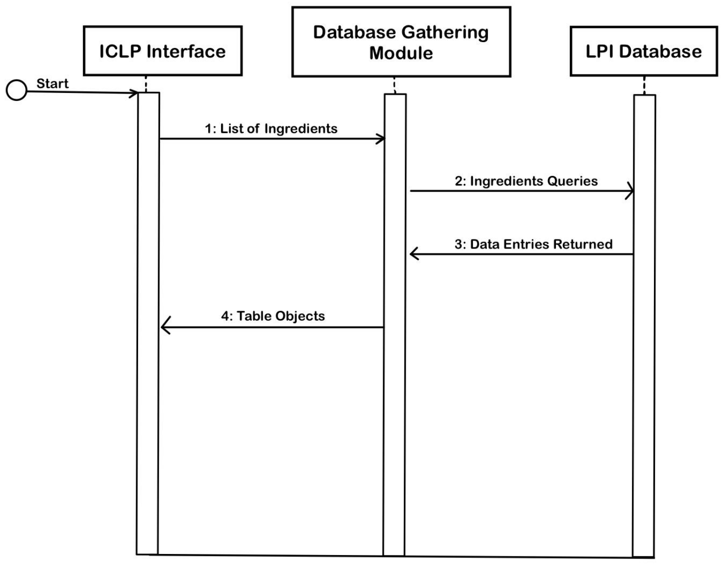

Figure 8. A sequence diagram for retrieving data from the LPI Database

e. Design rationale

The design choice to create the database gathering module was to separate out the processing of data from the creation of the ICLP interface and the database itself. This increases the similarity of components within a module making the code more comprehensible and allowing more members to work in parallel.

5. Dynamic Models of Operational Scenarios (Use Cases)

Use Case #1: User wants to update data in the Database

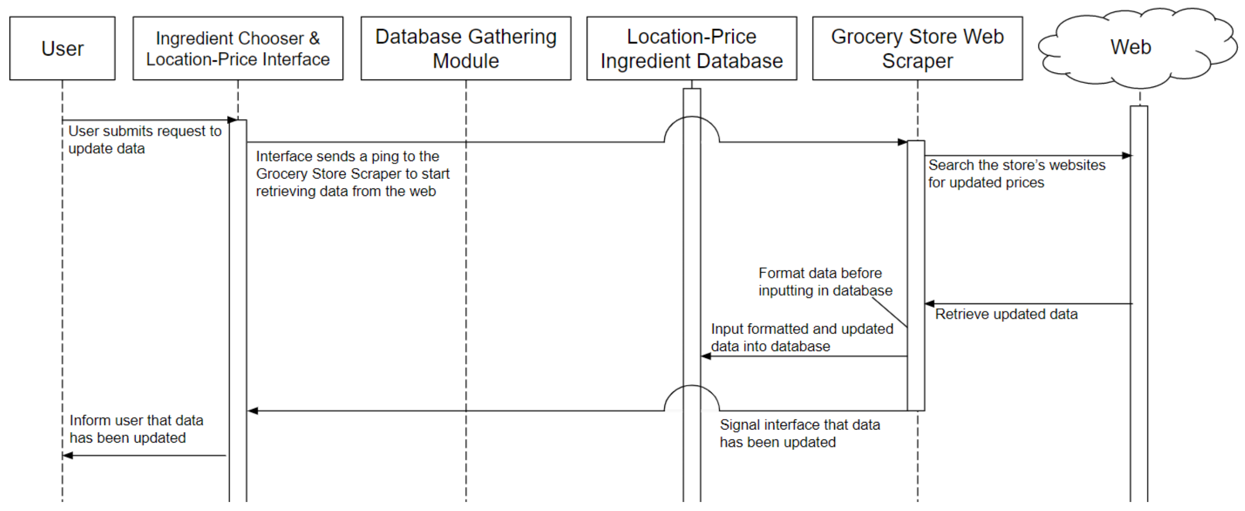
User
Ingredient Chooser & Location-Price Interface
Database Gathering Module
Location-Price Ingredient Database
Grocery Store Web Scraper
Web
User submits request to update data
Interface sends a ping to the Grocery Store Scraper to start retrieving data from the web
Search the store's websites for updated prices
Format data before inputting in database
Retrieve updated data
Input formatted and updated data into database
Inform user that data has been updated
Signal interface that data has been updated

Figure #9: Sequence diagram for updating the database with new data from the web

Use Case #2: User wants to view the cheapest price for a given item

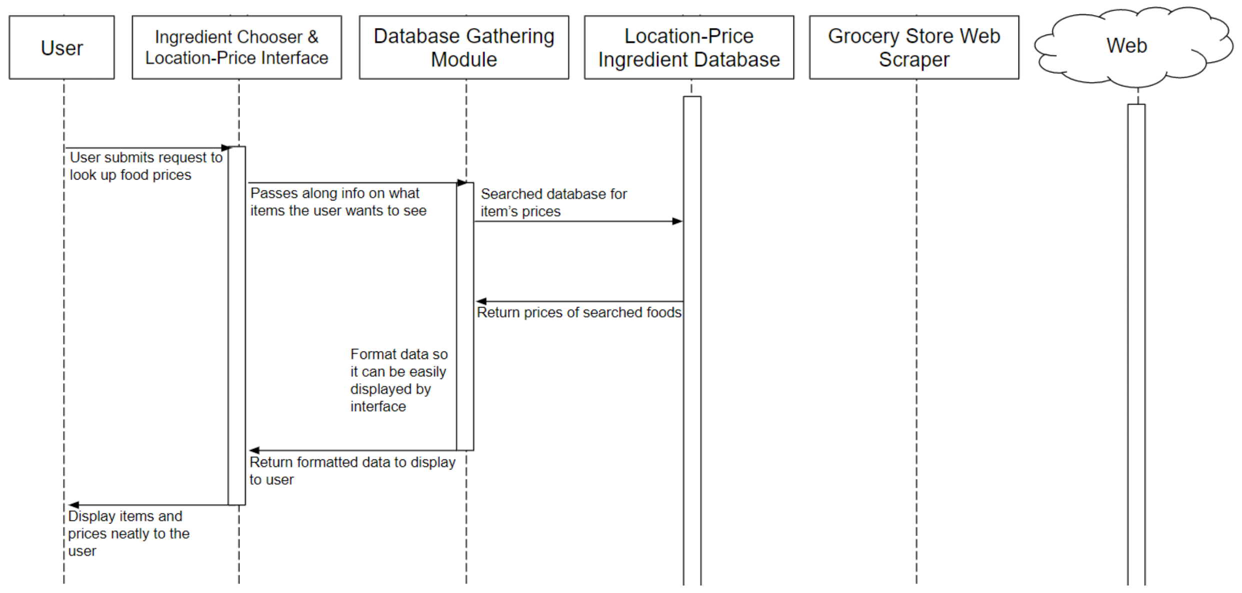
User
Ingredient Chooser & Location-Price Interface
Database Gathering Module
Location-Price Ingredient Database
Grocery Store Web Scraper
Web
User submits request to look up food prices
Passes along info on what items the user wants to see
Searched database for item's prices
Return prices of searched foods
Format data so it can be easily displayed by interface
Return formatted data to display to user
Display items and prices neatly to the user

Figure #10: Sequence Diagram for finding a cheap price for food within *Cheap Foods*

6. References

No references at this time.

7. Acknowledgements

This SDS document was based off of the SDS template document provided by professor Hornof at the University of Oregon.