Text

Description automatically generated

Team Pikachu

**CZ2006**

**SOFTWARE ENGINEERING**

**Created by:** Chen Zhengjie

**Updated by:** Everyone

**Last updated:** 08/04/2022

Team members:

Chen Zhengjie – U2023280A

Edison Koh - U2022330E

Kris Huang Wei - U2020746H

Liu Feng Hao – U2022839H

Fu Guan Qiao-U2022359H

Tan Li Ying- U2021711G

**Table of Contents**

**1**.[**Introduction**](#_heading=h.30j0zll) **4**

1.1 [Purpose of Product](#_heading=h.1fob9te) 4

1.2 [Intended Audience and Stakeholders](#_heading=h.3znysh7) 4

1.3 [Product scope](#_heading=h.516nkfpacg8o) 5

**2.**[**Overall Description**](#_heading=h.2et92p0) **5**

2.1 [Product Perspective](#_heading=h.tyjcwt) 5

2.2 OneTracker Features 6

2.3 [User Classes and Characteristics](#_heading=h.1t3h5sf) 7

2.4 [Operating Environment](#_heading=h.4d34og8) 7

2.5 [Design and Implementation Constraints](#_heading=h.2s8eyo1) 7

2.6 [User Documentation](#_heading=h.17dp8vu) 8

2.7 [Assumptions and Dependencies](#_heading=h.3rdcrjn) 8

[**3. External Interface Requirements**](#_heading=h.26in1rg) **9**

[3.1 User Interfaces (UI)](#_heading=h.lnxbz9) 9

[3.2 Hardware Interfaces](#_heading=h.35nkun2) 15

[3.3 Software Interfaces](#_heading=h.1ksv4uv) 15

[3.4 Communications Interfaces](#_heading=h.44sinio) 15

[**4. System Features and Functional Requirements**](#_heading=h.2jxsxqh) **1**6

[4.1 Login/out](#_heading=h.3j2qqm3) 16

4.2 Register 17

4.3 Stock 18

4.4 DeleteIngredient 18

4.5 ViewIngredient 19

4.6 Purchase 20

4.7 AddIngredient 21

4.8 Consume 22

4.9 DeleteFromPurchase 22

4.10 ViewHistory 23

[**5. Other Nonfunctional Requirements**](#_heading=h.1y810tw) **2**4

[5.1 Performance Requirements](#_heading=h.pq800mixpisw) 24

[5.2 Safety Requirements](#_heading=h.2xcytpi) 24

[5.3 Security Requirements](#_heading=h.1ci93xb) 24

[5.4 Software Quality Attributes](#_heading=h.3whwml4) 24

[5.5 Business Rules](#_heading=h.2bn6wsx) 25

[**6. Other Requirements**](#_heading=h.qsh70q) **2**5

# Introduction

## Purpose of Product

Our group’s web application enables users to keep track of the food items that were purchased as well as the food items that users have consumed. The application will return users how much calories they have consumed. The project is successful when users are able to monitor their purchase and consumption habits. This project is a tool to assist users in maintaining a healthier lifestyle.

## Intended Audience and Stakeholders

Our main target audience are Singaporeans who are determined to change their lifestyle and reduce their food wastage.

| Stakeholder | Purpose of SRS to stakeholder |
| --- | --- |
| Users | Documentation of problem solved using our group’s OneTracker web application  User may refer to 1.4 Product Scope |
| Developers | Documentation explaining design specification based on requirement of User Developers may refer to section 4 system function |

## Product scope

With the improvement in technology, purchasing groceries can be done easily through any device which is able to connect to the internet. As such, people do not check if they have already bought the item before or have similar items, before purchasing more. Bulk buying is also less of a hassle as delivery services ship products straight to their doors. Consequently, this leads to huge food wastage and people wasting their hard-earned money.

This increased convenience has also brought about another issue. People can just purchase the food they crave and consume it in unhealthy amounts. Without a way to track and monitor calorie intake, their cost on healthcare will significantly increase in the future.

OneTracker aims to help people manage their lifestyle by providing them with a useful assistant to track their purchases and consumption. It will be a one-stop application that records all past and recent purchases and consumption of food items.

# Overall Description

## Product Perspective

This product is an open-source web-based system implementing the client-server model. The OneTracker Application provides a tool for users to record food purchases, track consumption and inform users about their daily food intake. The user records the food into their account and can track the food nutrient information conveniently.

* 1. **OneTracker features**

Basic feature: Log-in and Registration

The system allows users to create an account with the application by using an email address. The user account will be created and user can login to access the unique features of OneTracker

Unique feature: Stock

OneTracker stock feature allows users to view the status of their food storage.

Unique feature: Purchase

OneTracker purchase feature allows users to search for items that they may have purchased. Spoonacular api will be used to return the item’s nutrient information. if the user confirms that the item purchase is the same as the one spoonacular returns, it can be added to the stock list.

Unique feature: Consume

The user can choose several items from their stock to consume. When the consume button is pressed, items of certain quantity selected for consumption by the user will be removed from the stock page and an entry in history will be added.

Basic feature: View history

This feature allows users to view all purchases and consumption done by the user. Users also have the option of filtering by purchases or consumption.

## User Classes and Characteristics

We believe users of OneTracker will be in one of the following classes

| Class 1 | People who need a helper to track and monitor their nutrients  intake from their consumption. |
| --- | --- |
| Class 2 | People who need a helper to track and monitor their food storage. |

## Operating Environment

Our application was developed to be a web-based application deployed on cloud. Django is used for the backend as it has many useful in-built libraries and functionalities which improve our backend logic significantly. Our simple and user-friendly frontend user interface was designed using Bootstrap’s styling capabilities. The database was imported to PostgreSQL from the django default database SQLite. Lastly, the whole entire application was deployed onto Cloud Platform Heroku. The application only needs users to have internet access to be able to open it on a browser of their choice.

## Design and Implementation Constraints

| Constraint 1 | Storage used cloud platform | Heroku only provides a limited amount of storage for free users. if application were to be scale up, payment for a different storage plan will be required |
| --- | --- | --- |
| Constraint 2 | Limited spoonacular api calls per api key | The free api only allows 150 calls a day per key. Exceeding will give a keyerror. it is also inefficient to switch keys consistently |
| Constraint 3 | Unable to get all specific user purchased food item | The Spoonacular database does not have all the food products. If a Singaporean user wants to add a uniquely Singapore item eg khong guan biscuit, spoonacular will not have data on it. |

## User Documentation

Certain parts of the SRS can be a useful documentation for users to understand the uses of each function on the UI. The application functions are named in a straightforward way and the UI is user friendly,we believe a user will understand the functionality of the app after a few tries. Thus, no further documentation besides the SRS will be published

## Assumptions and Dependencies

The following assumptions are made:

| Assumption 1 | External tools used (heroku and spoonacular) are the free version. Therefore, there is limitation on the storage space in heroku and limited api calls |
| --- | --- |
| Assumption 2 | Application functions are named in a straightforward way and the UI is user friendly. This is assumed and not been tested |
| Assumption 3 | People who want to use the app have basic knowledge on using the internet. Failure to use any internet browser would mean the application is inaccessible. |

# 3. External Interface Requirements

## 3.1 User Interfaces (UI)

For UI, it can be divided into two sets namely UI when user is authenticated and logged in and UI when user is not logged in.

**3.1.1 UI for User is not logged in**

The pages a user can access when not logged in is limited. They are only allowed to access the about page, login and register page. The navigation bar will also reflect their limited page access.

HomePage/AboutUs page

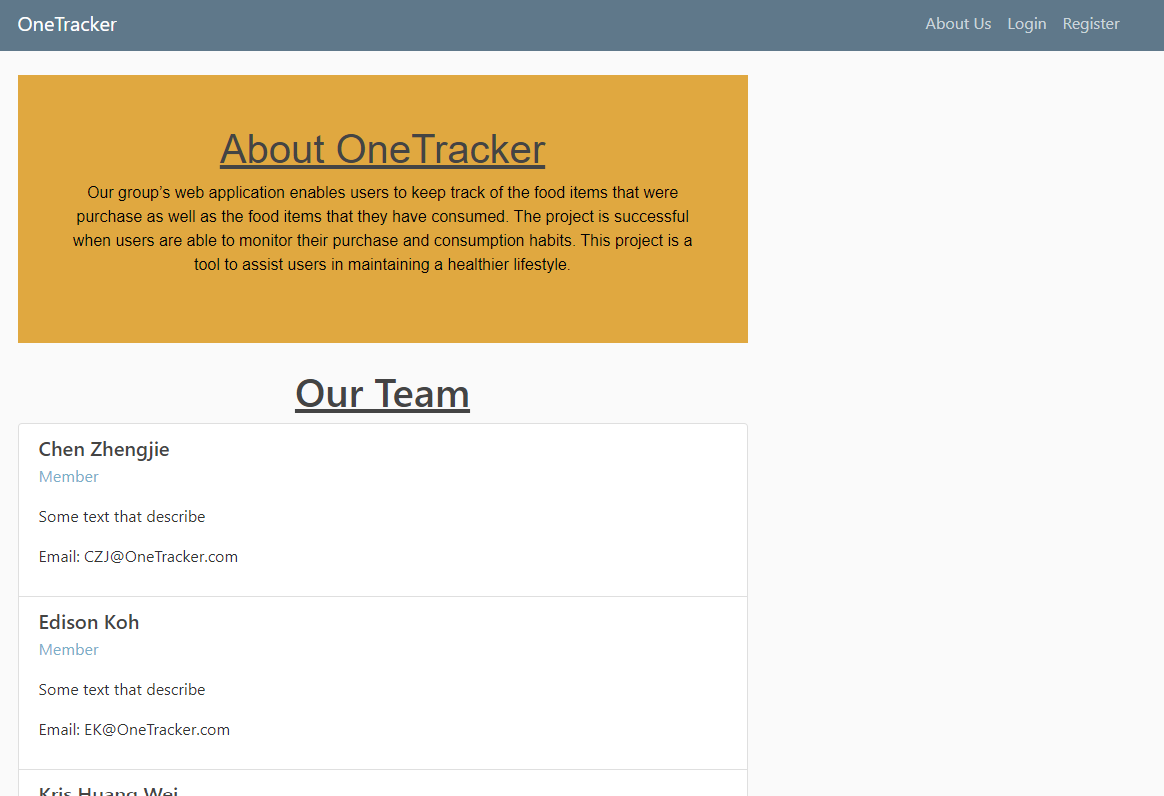


Fig 1: HomePage/AboutUs page

Users will enter the page shown in fig 1 when they enter OneTracker’s website url. Clicking on the about us page will redirect users to the same page.

Login page

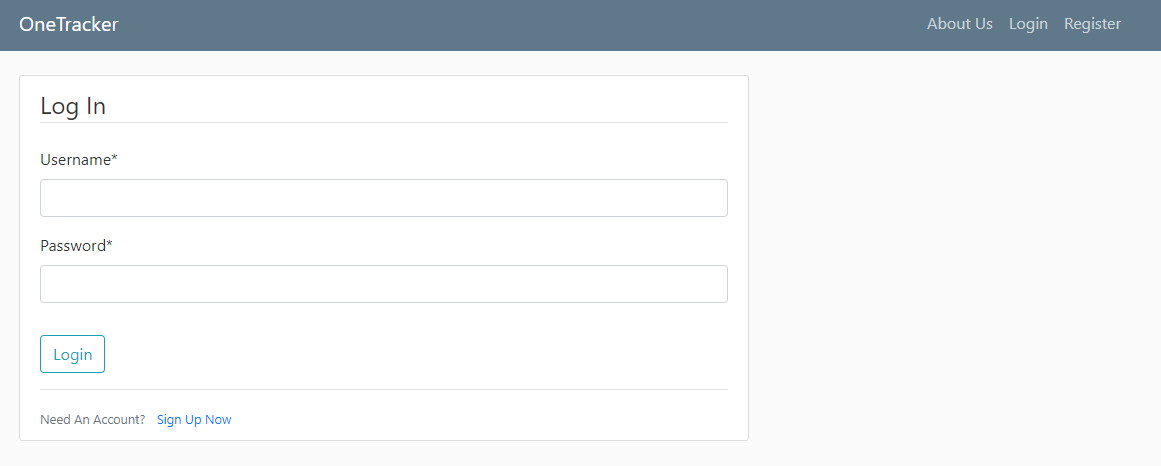


Fig 2: login page

By clicking login on the navigation bar, users will be redirected to the login page as shown in the fig 2.

Register page

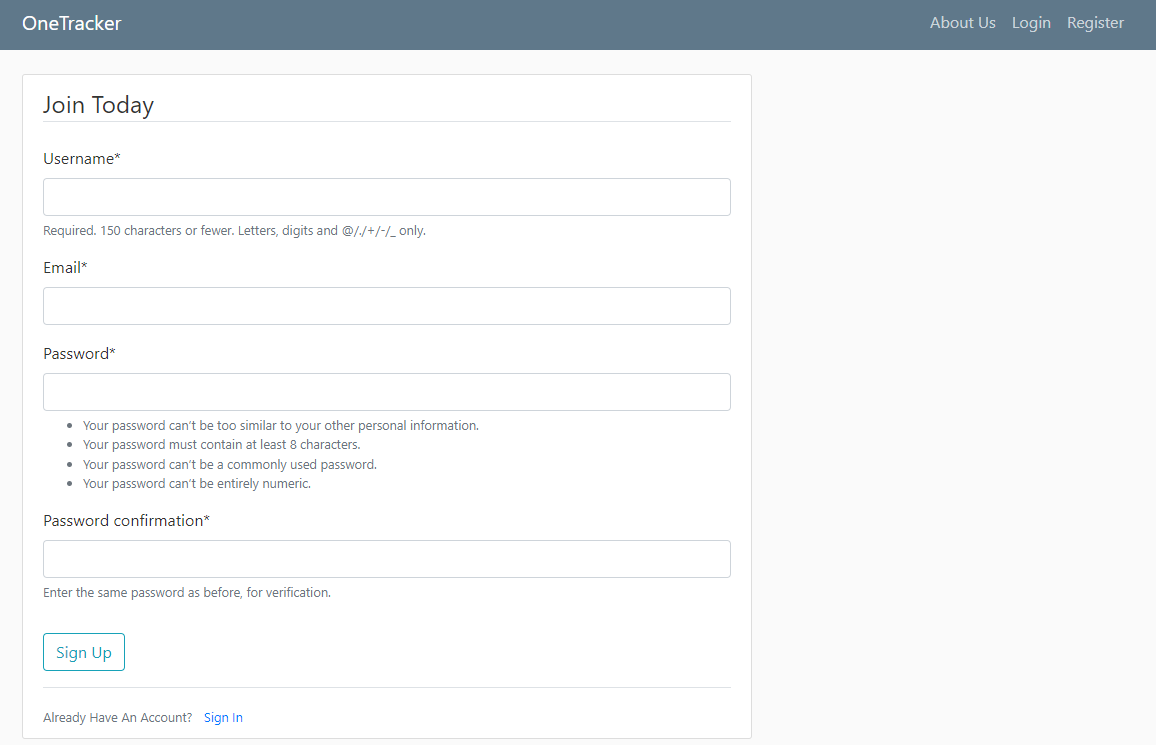


Fig 3: Register page

By clicking login on the navigation bar, users will be redirected to the register page as shown in the fig 3.

**3.1.2 UI for User is logged in**

The User can now access OneTracker’s more functional pages namely stock, purchase, consume, history. The navigation bar has also changed to reflect the change in page access.

Home Page when login

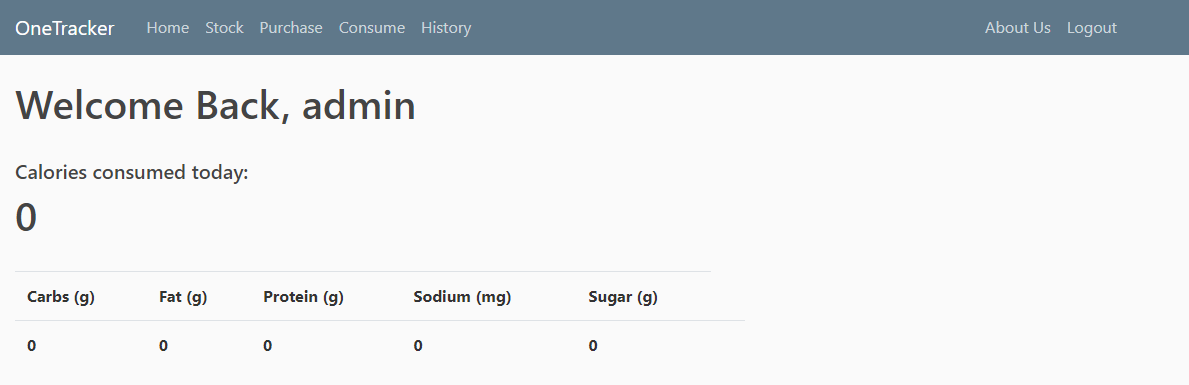


Fig 4: Home Page after successful login

Users will enter the page shown in fig 4 when they successfully login into their OneTracker’s account.

Stock Page

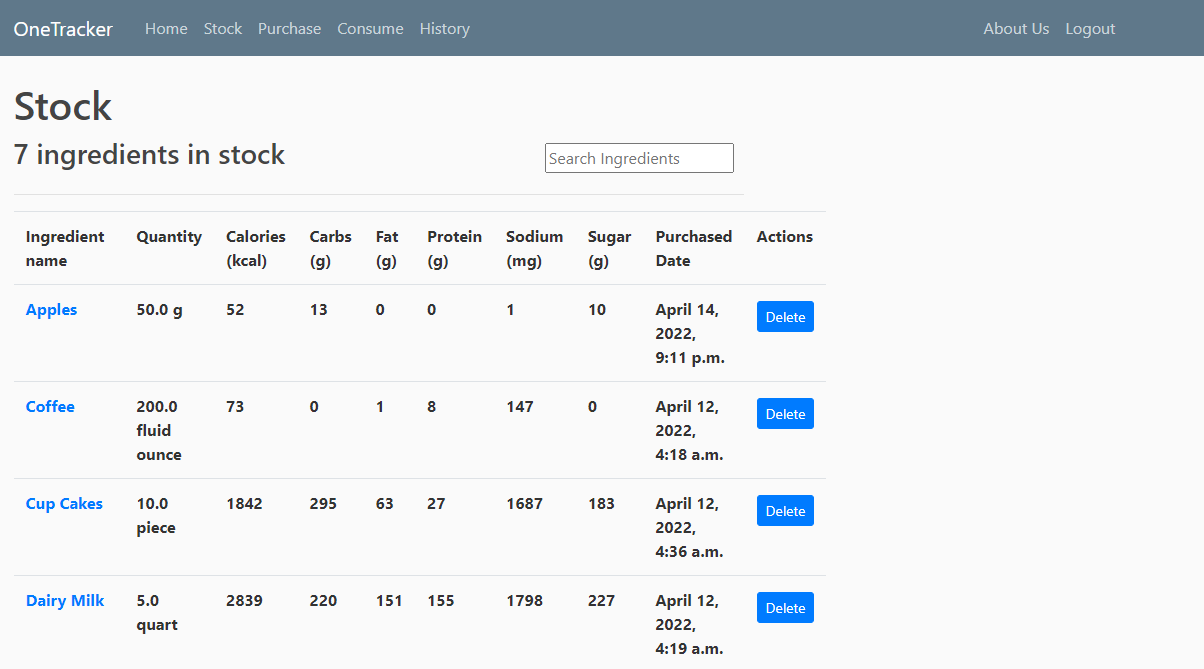


Fig 5: Stock page

The stock page displays the user’s food storage. It also gives the option to allow users to delete certain entries and look into and item more specifically.

Item view page

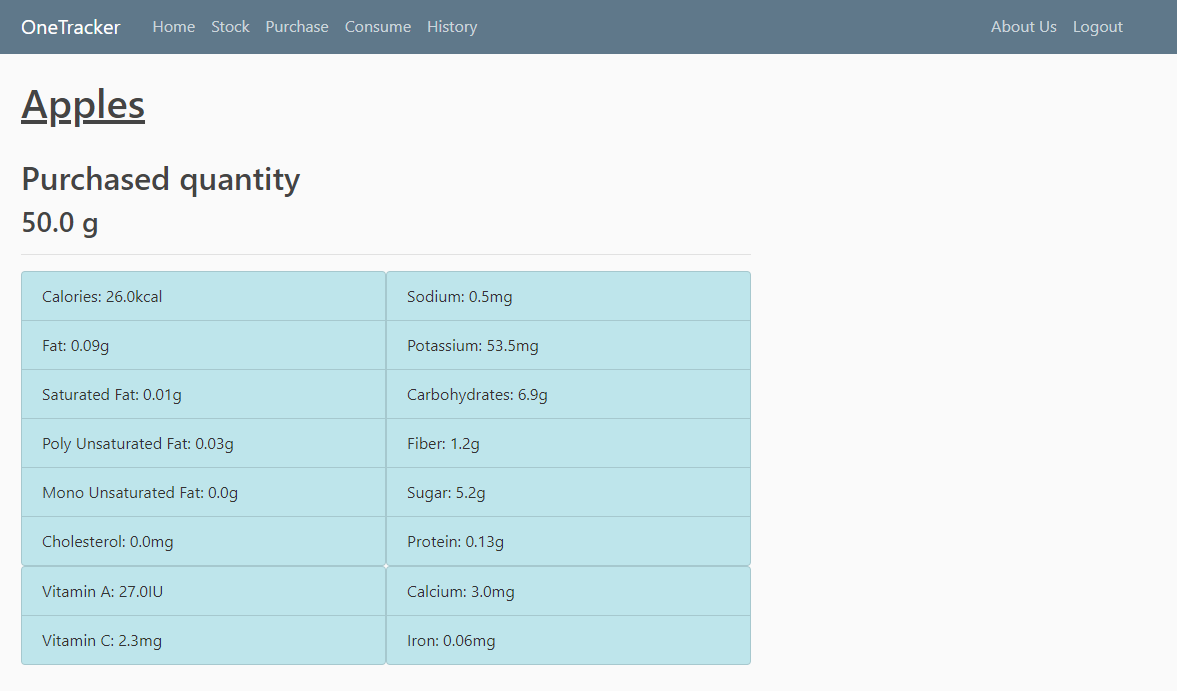


Fig 6: items view page

After clicking on an item in fig 5, the system will query spoonacular using the api to get even more specific nutrients information of the item selected from fig 5.

Purchase page

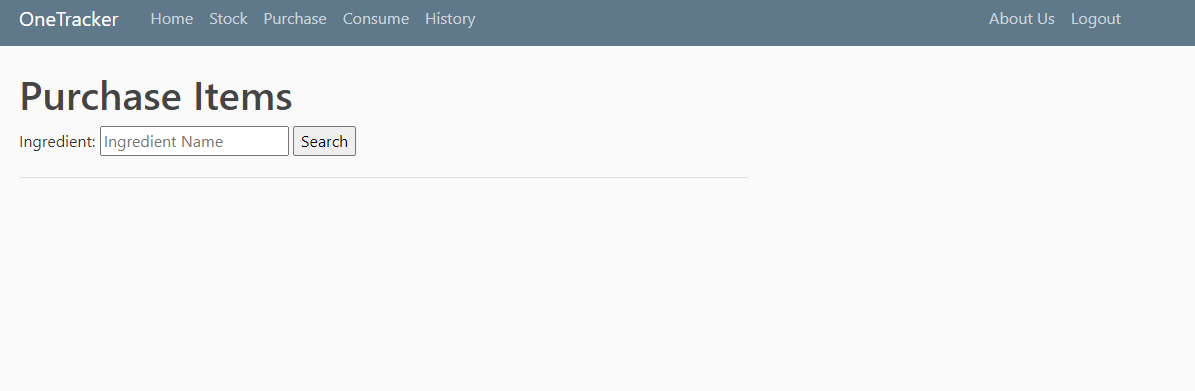


Fig 7: purchase page

Using the search bar, users can look for items that they have purchased.

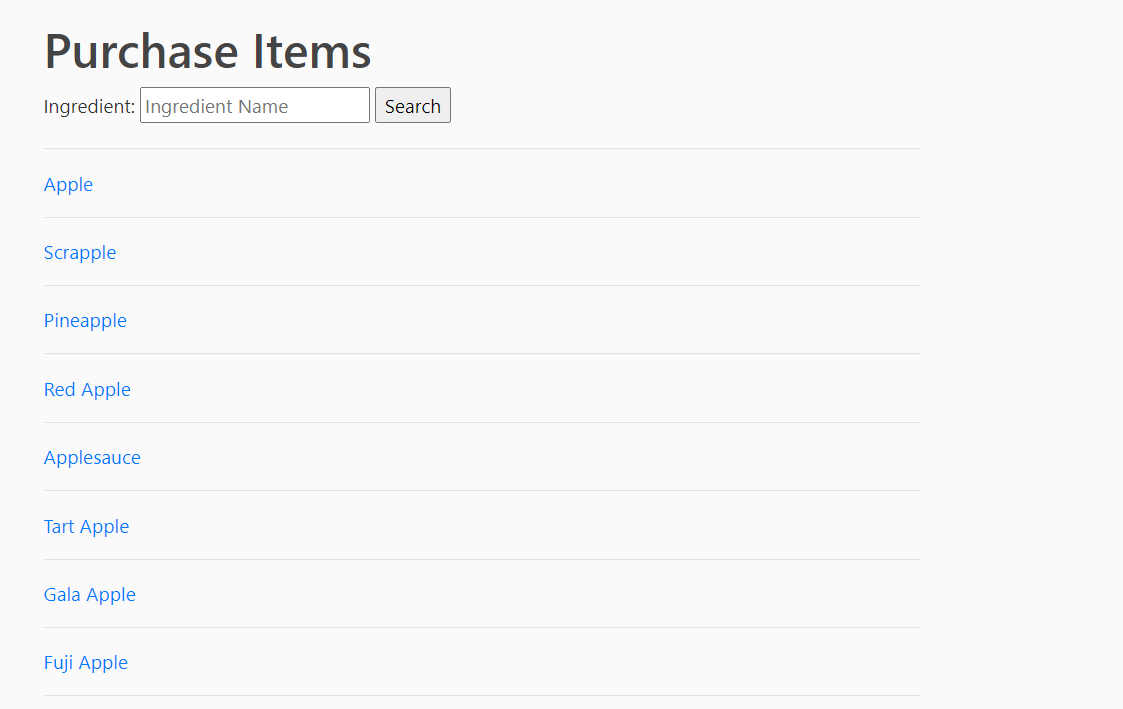


Fig 8: Purchase search successful

After a successful search using spoonacular api, the purchase page fig 8 will display all items in spoonacular database that match with the searched item.



Fig 9: Specific item view purchase page

By clicking on any item in the search list shown in fig 8, users will be redirected to fig 9. They can choose to add the item or go back to the search list in fig 8.

Consume Page

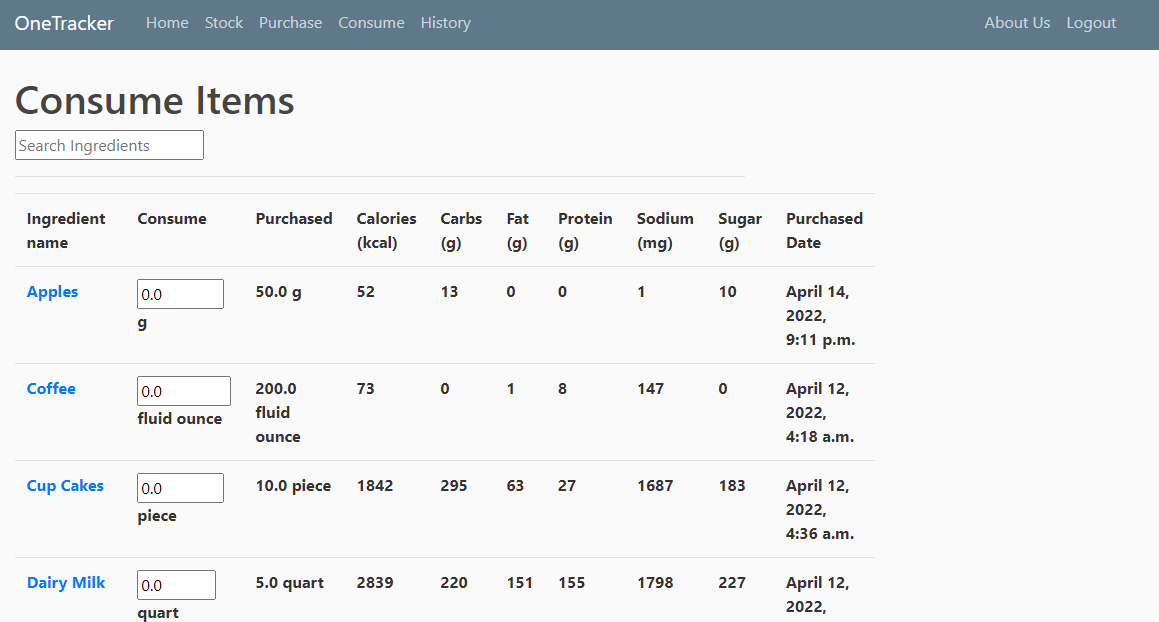




Fig 10: Consume page

The Consume page will show all items in the stock page. In addition, it has the functionality to remove items from the stock page and add entries in the history page when the user confirms by pressing the consume button at the bottom of the page.

History Page

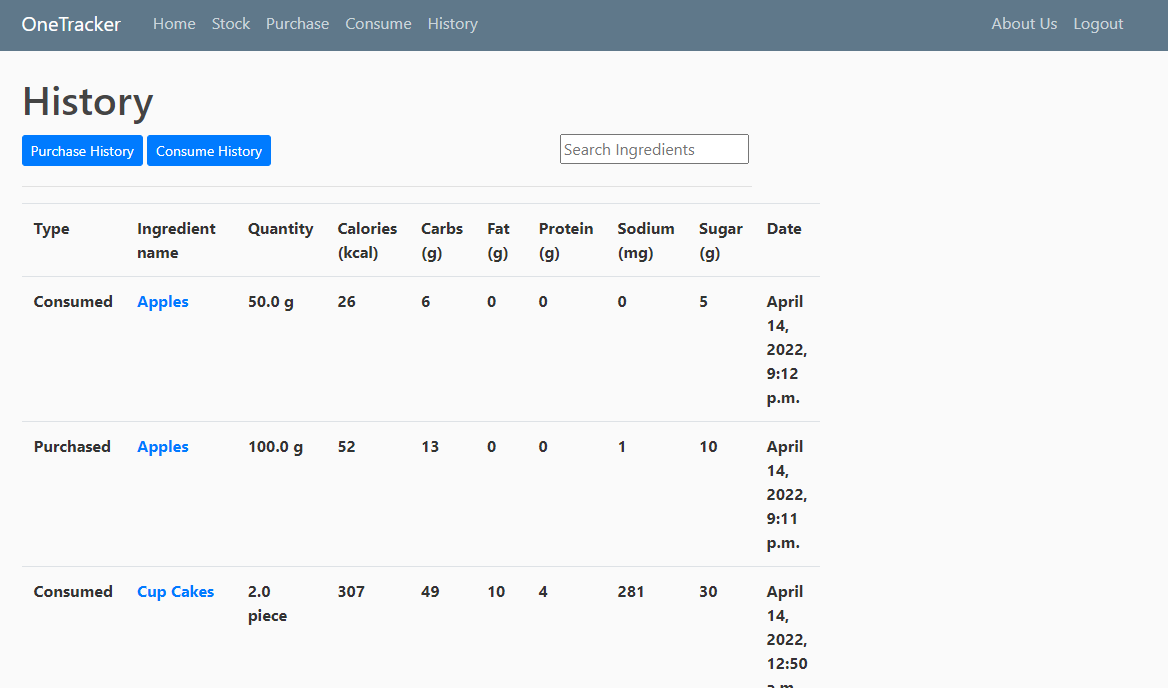


Fig 11: History page

The history page shows all past purchases and consumption of items. It has two buttons to get purchase records only or consumption records only. A search bar is all present for quick lookup of specific items.

## 3.2 Hardware Interfaces

The only hardware interface that is required will be the computer that the user is using to run our Onetracker web application.

## 3.3 Software Interfaces

Following are the software used:

| Software used | Description |
| --- | --- |
| Visual Studio Code | 1. Handles the importing of required django libraries.  2. Import certain in-built functionality for user related functions for instance login authentication |
| PostgresSQL | Database for managing food items data and user data |
| Cloud platform Heroku | Cloud service platform where the OneTracker is deployed on |

## 3.4 Communications Interfaces

Communications in OneTracker mostly HTTP get and post requests. The rest of the communications are in SQL to communicate with PostgreSQL database

# 4. System Features and Functional Requirements

## 4.1 Login/out

## 4.1.1 Description and Priority

Allows users to access an existing account previously created by entering the correct corresponding username and password.

Priority: High

## 4.1.2 Stimulus/Response Sequences

1. User clicks login on navigation bar in Fig 1 in section 3.1.1
2. Redirected to login page Fig 2 in section 3.1.1
3. User enters correct corresponding username and password to existing.
4. System verifies input from user
5. User is redirected to logged in home page shown in Fig 4 in section 3.1.2
6. When logged in user can click logout button on navigation bar to log out of account

## 4.1.2 Functional Requirements

REQ-1: User must be able to login to existing accounts

1.1 System must be able to verify login information with data stored in PostgreSQL

1.2 Prompted of any input fields is wrong

1.3 System must redirect user to logged in home page

REQ-2: User must be able to logout from existing accounts

2.1 System must be able to redirect user to logout successful page after user has clicked logout on navigation bar

**4.2 Register**

## 4.2.1 Description and Priority

Receives required user information and adds the new user into the database.

Priority: High

## 4.2.2 Stimulus/Response Sequences

1. User clicks onto the register button on the navigation bar
2. System redirects user to register page
3. User enter email, username and password and clicks register
4. System check if there is an existing account with the same email or username
5. System checks if password requirement is met
6. System informs user is account is created successfully
7. User is redirect to the login page

## 4.2.3 Functional Requirement

REQ-1 User must be able to create a new account

1.1 Username and email constraint must be met

1.1.1 Username and email must not belong to an existing account

1.2 Password constraint must be met

1.2.1 Password must have more than 7 character

1.2.2 Password must have a mix of alphanumeric characters

1.2.3 Password and confirm password fields must match

1.3 System must be able to add users information and create a new account with it

1.3.1 System must save all new user information into PostgreSQL

**4.3 Stock**

## 4.3.1 Description and Priority

Display all items user still has in their food storage

Priority: High

## 4.3.2 Stimulus/Response Sequences

1. User has logged in
2. User clicks on Stock on navigation bar
3. System retrieve all stock information from database
4. System display stock information on page

## 4.3.3 Functional Requirement

REQ-1 System must be able to display stock

1.1 System must be able to access database to retrieve user’s stock information

1.2 System must be able to display stock information in a organized manner

**4.4 DeleteIngredient**

## 4.4.1 Description and Priority

Delete an item in the stock table.

Priority: Medium

## 4.4.2 Stimulus/Response Sequences

1. User has logged in
2. User clicks on Stock on navigation bar
3. System retrieve all stock information from database
4. System display stock information on page
5. User clicks on delete button under the actions column

## 4.4.3 Functional Requirement

REQ-1 System must be able to delete stock item selected by user

1.1 System must be able to access database to delete stock item

1.2 System must be able to display the new stock information

**4.5 ViewIngredient**

## 4.5.1 Description and Priority

Using the ingredient id given by spoonacular to get detailed nutrient information.

Priority: High

## 4.5.2 Stimulus/Response Sequences

1. User has logged in
2. User clicks on Stock on navigation bar
3. System retrieve all stock information from database
4. System display stock information on page
5. User clicks on the item name in stock table
6. System get spoonacular ingredient id from database
7. System uses ingredient id and quantity purchase to api call spoonacular
8. Spoonacular returns detailed nutrient information.
9. System display it as shown in section 3.1.2 Fig 6

## 4.5.3 Functional Requirement

REQ-1 System must be able to query spoonacular database

1.1 System must be able to retrieve id and quantity from application’s database

1.2 System must be use valid api key and item information retrieve to query spoonacular

REQ-1 System must be able to display return spoonacular json file

1.1 System must be able process the json file

1.2 System must display data received from spoonacular on itemsview page

**4.6 Purchase**

## 4.6.1 Description and Priority

Using a string of a food item given by a user, the system will query spoonacular to get a list of items and their spoonacular ingredient id.

Priority: High

## 4.6.2 Stimulus/Response Sequences

1. User has logged in
2. User clicks on Purchase on navigation bar
3. User enter food item string into search bar
4. System will query spoonacular for list of matching item from spoonacular database
5. System will display items list as shown in section 3.1.2 Fig 8

## 4.6.3 Functional Requirement

REQ-1 System must be able to query spoonacular database and display results

1.1 System must be use valid api key and user input string to query spoonacular

1.2 System must be able retrieve results

1.3 System must be able to display list of match item from spoonacular

**4.7 AddIngredient**

## 4.7.1 Description and Priority

Adding a new item into stock

Priority: Hight

## 4.7.2 Stimulus/Response Sequences

1. Continued from 4.6 system feature
2. User clicks on item he or she purchase on the return list
3. system redirect user to a viewingredient page with an additional add button at the bottom of the page and a two input field for amount and units
4. User set amount and units. The default will be amount 100 and unit g
5. User clicks on add button to add item into stock
6. System updates stock according
7. System redirects User back to items list

## 4.7.3 Functional Requirement

REQ-1 System must be able add new item into stock

1.1 System must be able to save nutrient information into local database

1.2 System must also record quantity and units user input

1.3 System must be able retrieve results in stock table in the future

1.4 System must record this adding of item as purchase in history

**4.8 Consume**

## 4.8.1 Description and Priority

Display items in stock that user can consume and their balance quantity in food storage

Priority: Low

## 4.8.2 Stimulus/Response Sequences

1. User has logged in
2. User clicks on Consume on navigation bar
3. System will display items in stock that user can consume

## 4.8.3 Functional Requirement

REQ-1 System must be able to retrieve items in stock and display it correctly

**4.9 DeleteFromPurchase**

## 4.9.1 Description and Priority

Removing items from stock for consumption and adding consumption to history

Priority: Medium

## 4.9.2 Stimulus/Response Sequences

1. continue from consume 4.8
2. User can select the amount of each item consume and click consume button at the bottom to trigger the function
3. System will remove items selected by user from stock and add new entries in history
4. Logged in Home page will be updated to reflect nutrient intake information

## 4.9.3 Functional Requirement

REQ-1 System must be able to verify if item has enough quantity for user to consume

1.1 System must be able to compare user selected consume amount with quantity stated in stock table

1.2 System must raise error if user select quantity consume greater than amount of item balance in food storage

REQ-2 System must update all related database table correctly

**4.10 ViewHistory**

## 4.10.1 Description and Priority

Display past consumption and purchases records

Priority: Medium

## 4.10.2 Stimulus/Response Sequences

1. User has logged in
2. User clicks on History on navigation bar
3. System will display items that had been consume and purchased
4. User can filter according to consume or purchase

## 4.10.3 Functional Requirement

REQ-1 System must be able to retrieve history from database and display it correctly

# 5. Other Nonfunctional Requirements

## 5.1 Performance Requirements

* The application will perform accurately for all user queries.
* The purchase and consumption information must be updated within 5 minutes.
* 80% of the first users must be able to learn how to use various features of the application within 5 minutes.
* login verification process is done almost immediately
* retrieval of information and after displaying results is done within seconds

## 5.2 Safety Requirements

* The application shall not share the user’s account information and application usage information with third-party companies.
* The user shall not share their related verification information with others.

## 5.3 Security Requirements

* The system must hash the password created for the user's account.
* When users try to login, the system must compare the input password with the hashed password in the database.

## 5.4 Software Quality Attributes

* Data should be back up frequently
* System should be available 24/7
* If a reboot is required, full system functionality must be restored within 5 minutes
* The database can be replaced with any commercial product supporting standard SQL queries.
* Help messages must be displayed according to the user’s locale.

## 5.5 Business Rules

* Users register an account with name, valid email and password. A user can only edit their own information and cannot view other users information in the system. Only the administrator can manage the information of all users in the database.
* Users are authorized to access spoonacular API available in the system

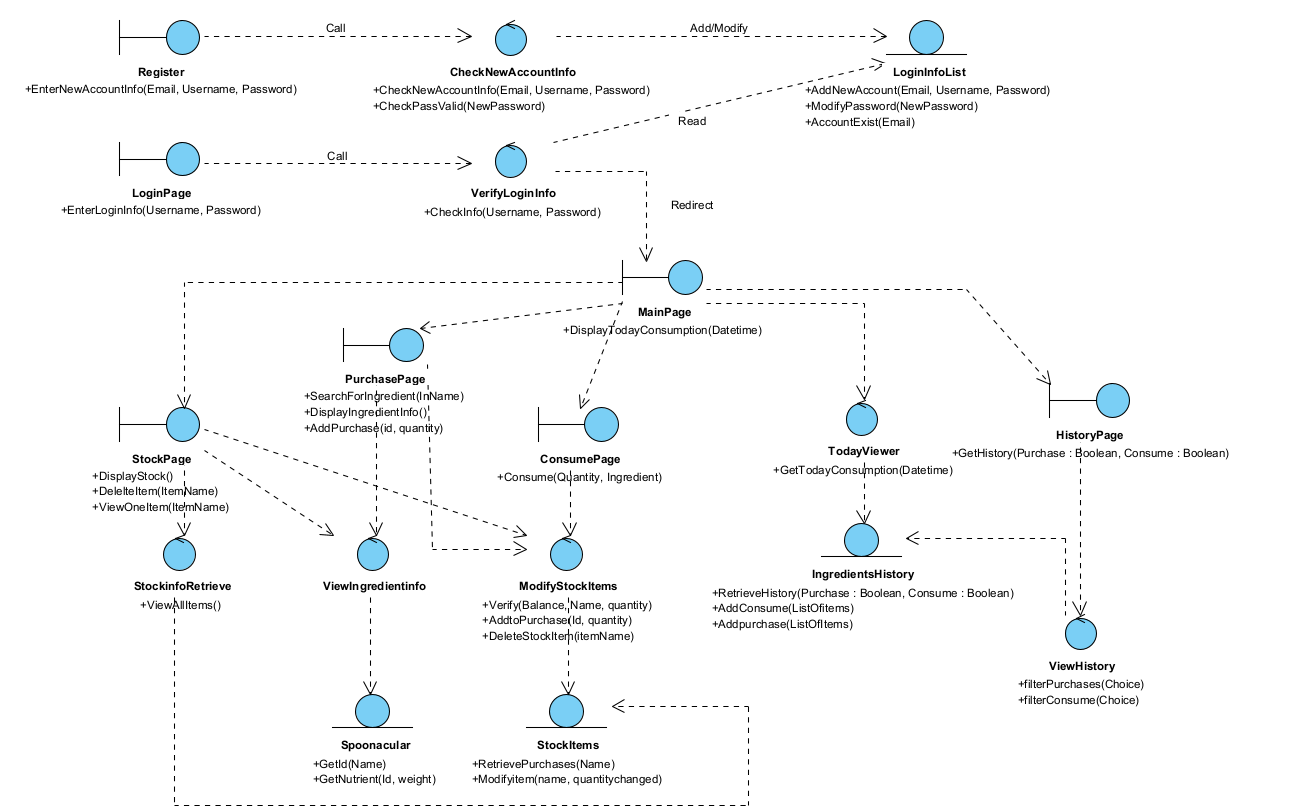
# 6. Other Requirements

**Appendix A: Glossary**

| No. | Term | Description |
| --- | --- | --- |
| 1. | User | People who have an existing Onetracker account. They have a unique username and email. |
| 2. | Stock | The function that retrieves all items in the food storage |
| 3. | food storage | The food items that the user previously purchased and were entered into the system. System had saved data into the database. |
| 4. | Ingredient | Ingredient, Food item and Item can be used interchangeably. They are the edible items that were purchased by the user. |
| 5. | Food item | Ingredient, Food item and Item can be used interchangeably. They are the edible items that were purchased by the user. |
| 6. | Item | Ingredient, Food item and Item can be used interchangeably. They are the edible items that were purchased by the user. |
| 7. | Nutrient information | Attributes: Calories, carbs, fat, sugar, sodium and protein that are save in the local database |
| 8. | Detail Nutrient information | The other food attributes that will be shown only when itemview is used |
| 9. | itemview | Function to query spoonacular to get all food attributes |
| 10. | Basic function/feature | Features that most application will have |
| 11. | Unique function/feature | Feature that should be unique to OneTracker |
| 12. | Priority ranking | With the absence of this system function how well will the application perform is the  criterion used for priority in section 4. |
| 13. | fields | Input boxes for user to enter related information |
| 14. | Password requirement | 1. Must have greater or equal to 8 characters  2. Must have a mix of alphanumeric and uppercase characters |
| 15. | units | Unit of measurement for ingredients. for instance grams(g), ounce, piece etc |
| 16 | History | Past purchases and consumption |
| 17. | Database | Where data is stored and retrieved from |
|  |  |  |

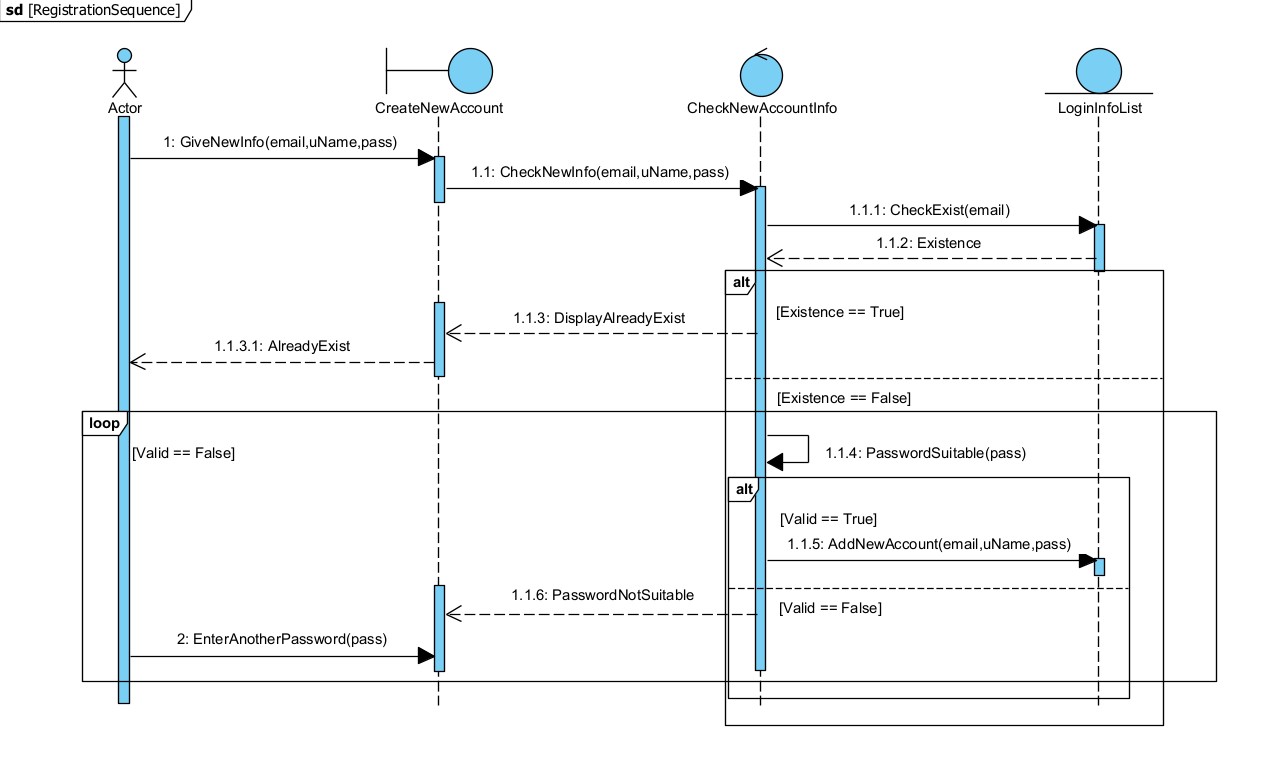
**Appendix B: Analysis Models**

Class diagram

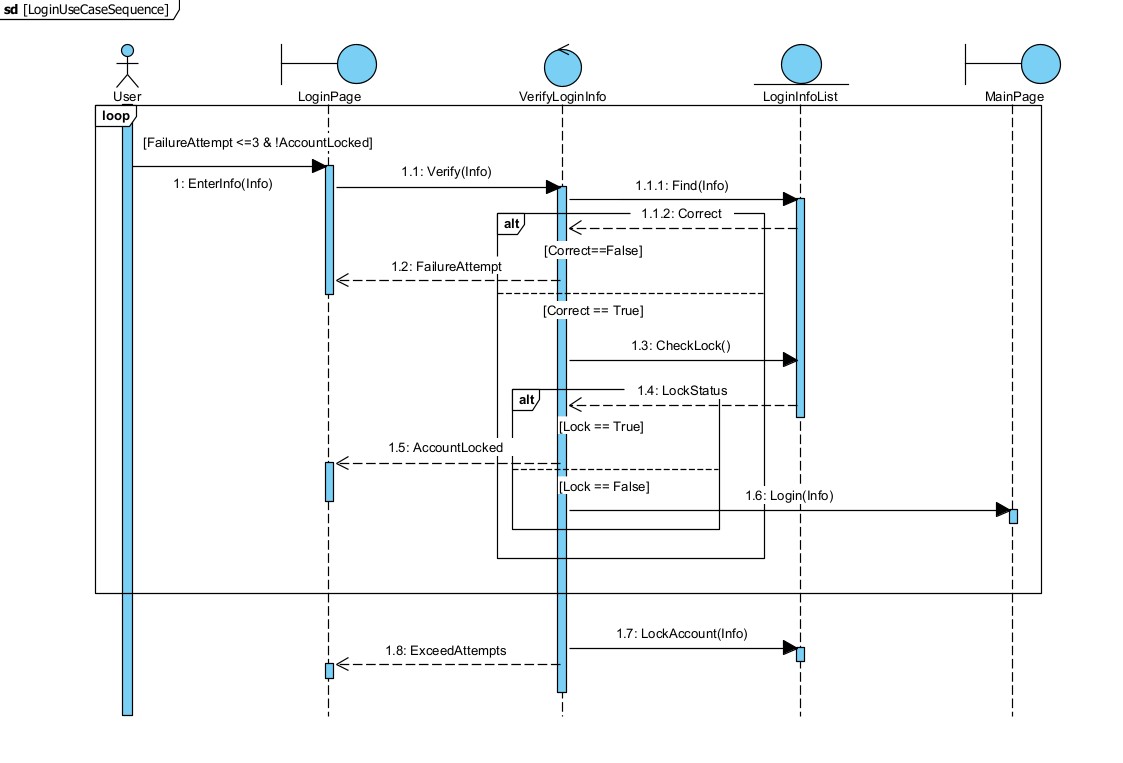
****

Sequence diagrams

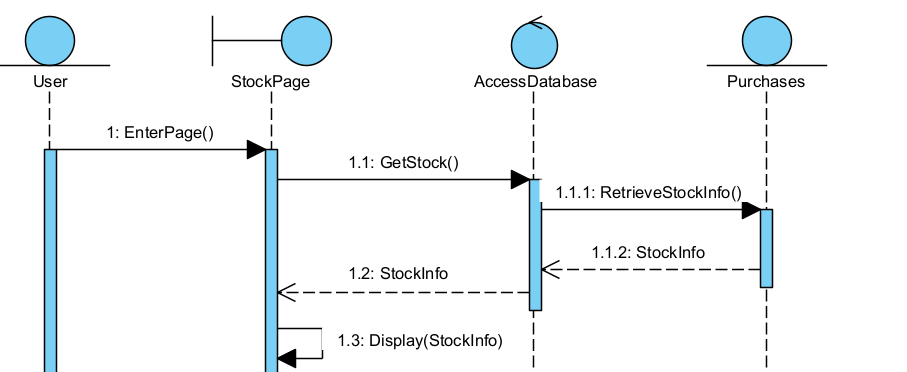
Registration

****

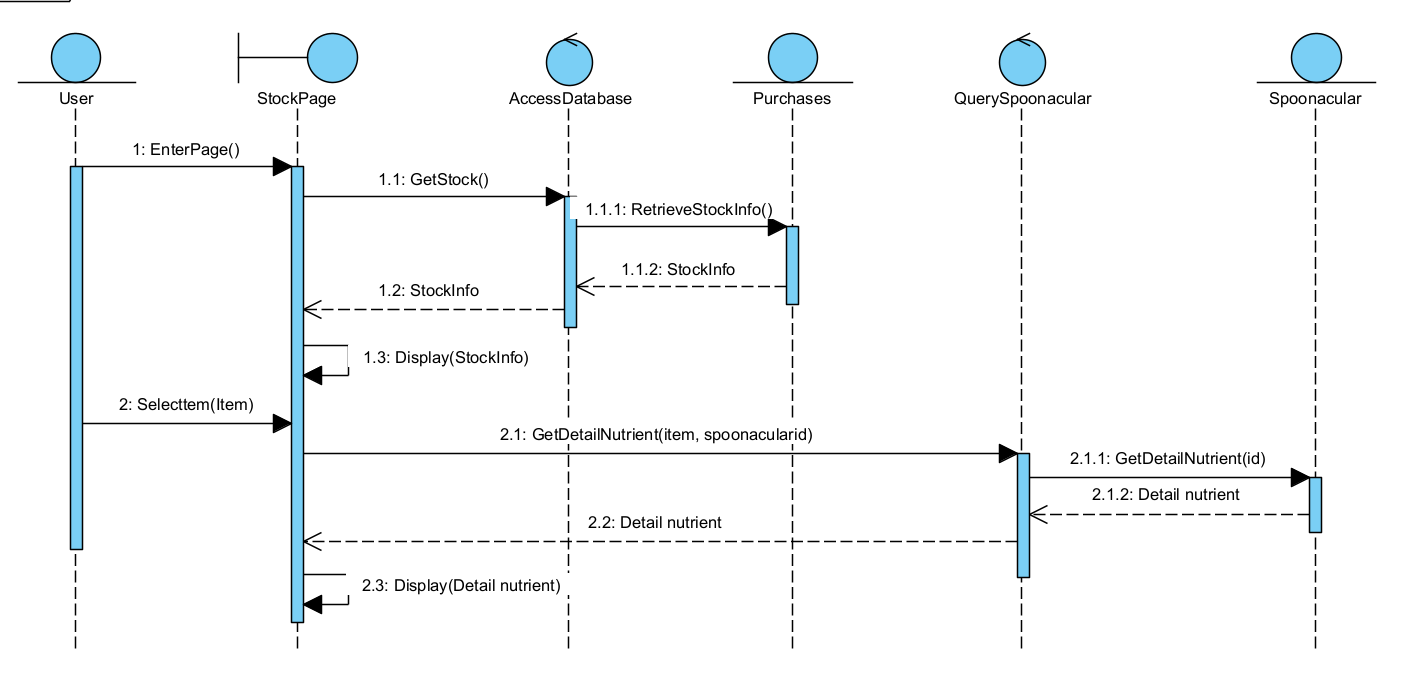
Login

****

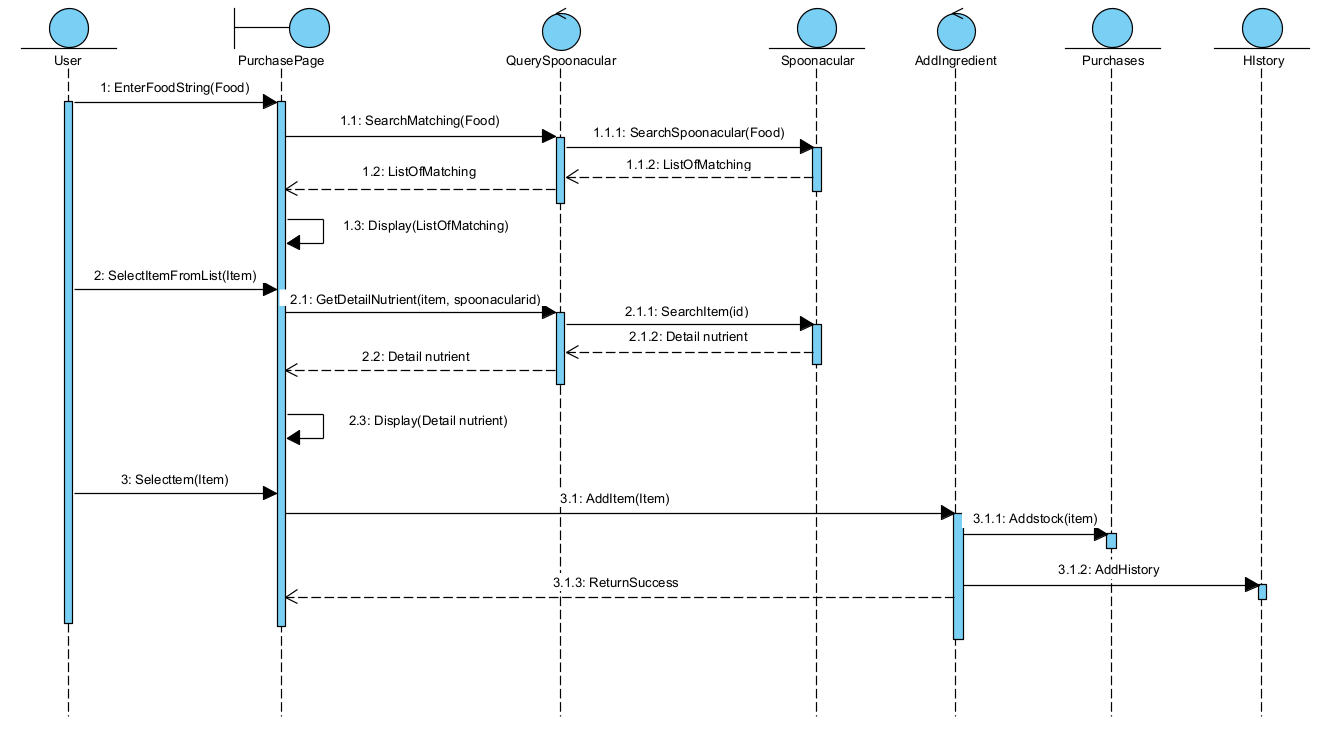
Stock



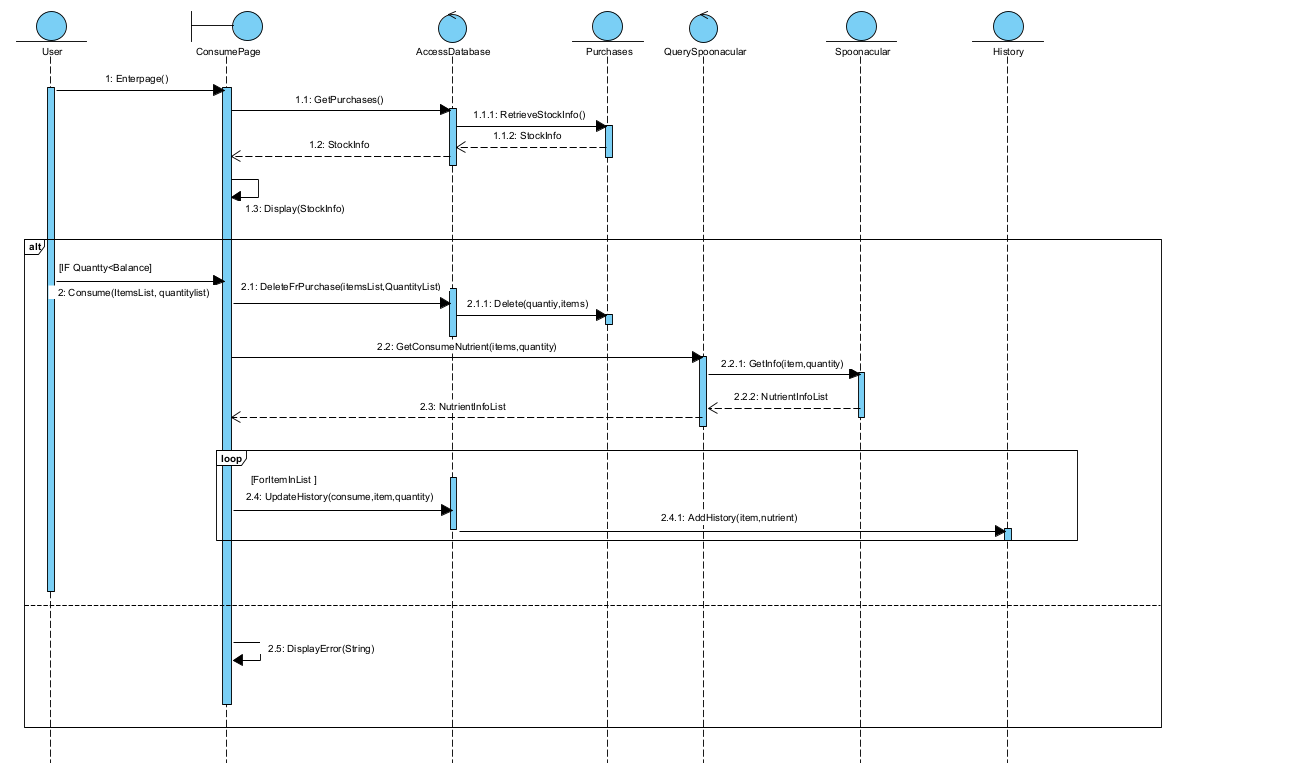
ViewIngredient



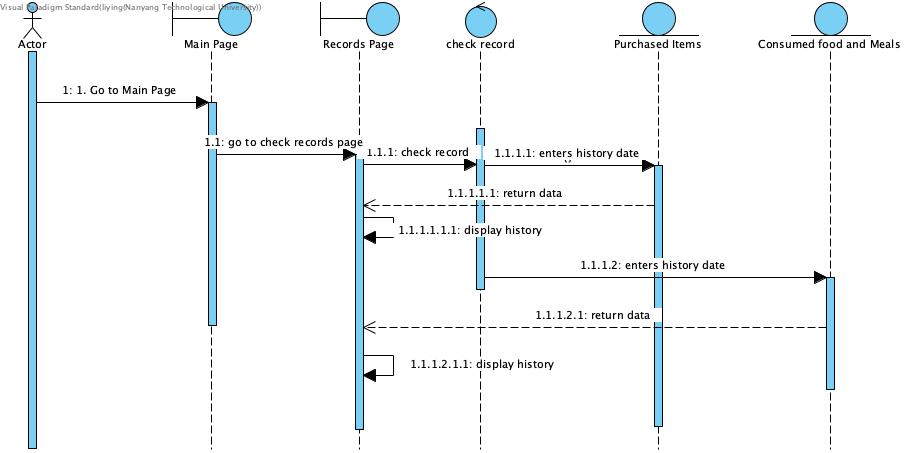
AddIngredient

****

DeleteFromPurchase

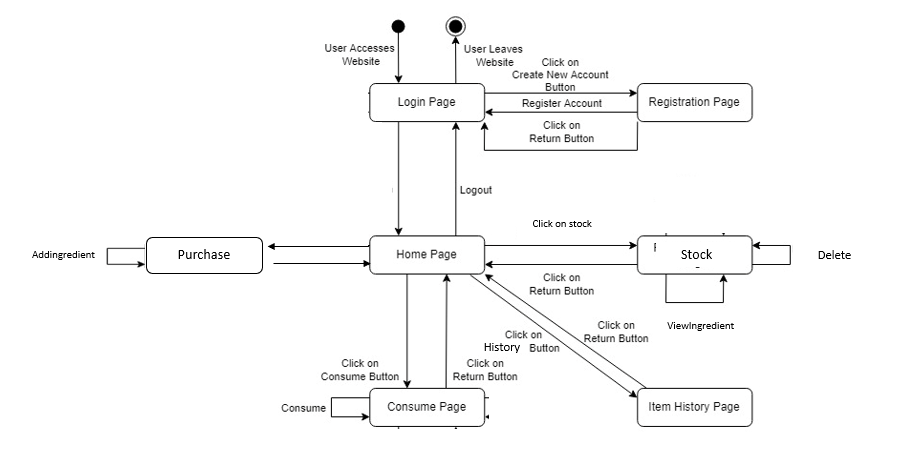


ViewHistory

****

**Appendix C: To Be Determined List**

**Dialogue map:**

**

**Use case description:**

Use Case 1: Login

| Use Case ID: | Use Case 1 | | |
| --- | --- | --- | --- |
| Use Case Name: | Login | | |
| Created By: |  | Last Updated By: |  |
| Date Created: | 24/03/2022 | Date Last Updated: |  |

| Actor: | User and System |
| --- | --- |
| Description: | This use case depicts how users login into an existing account. |
| Preconditions: | 1. User’s device should have access to the Internet(WIFI/4G)  2. User should have an existing account  3. User’s account should not be locked |
| Postconditions: | 1. User will login to existing account  2. User will be redirected to the main page |
| Priority: | High |
| Frequency of Use: | Very high |
| Flow of Events: | 1. User enters email or username and password  2. Clicks on the “login” button  3. System checks the validity of login information provided by the user  3.1 System checks validity of email or username  3.2 System checks lock status of accountem checks correctness of password  4. System redirects user to the main page |
| Alternate Flows: | AC 3.1: Invalid email or username  1. System will display an error message.  2. Number of failed attempts will increase by 1  2. Return to step 1.  3. User has to re-enter all login information.  AC 3.2: Password does not correspond to email or username  1. System will display an error message  2. Number of failed attempts will increase by 1  3. Return to step 1.  4. User has to re-enter the password. |
| Exceptions | 1. Invalid email or username entered by the user.  2. Wrong corresponding password to valid email or username  3. Account locked after three continuous failed login attempts |
| Includes: |  |
| Special Requirements: |  |
| Assumptions: |  |
| Notes and Issues: |  |

Use Case 2: Registration

| Use Case ID: | Use Case 2 | | |
| --- | --- | --- | --- |
| Use Case Name: | Registration | | |
| Created By: |  | Last Updated By: |  |
| Date Created: | 24/03/2022 | Date Last Updated: |  |

| Actor: | User and System |
| --- | --- |
| Description: | This use case depicts how a new user can create an account. |
| Preconditions: | 1. User’s device should have access to the Internet(WIFI/4G)  2. Email has not been used for an existing account |
| Postconditions: | 1. New user account information added to the database.  2. A verification email will be sent to the user. |
| Priority: | High |
| Frequency of Use: | Once |
| Flow of Events: | 1. User enters the registration page.  2. User enters email, username and password.  3. User clicks on the “create a new account” button.  4. System receives user information and check its validity  4.1 System checks validity of email and username  4.2 System checks password against password requirements  5. System adds new account created to accounts list in database  6. System display successful account creation notice |
| Alternate Flows: | AC 4.1: Email has been used for an existing account  1. System will display an error message  2. Return to step 1  2.AC 4.2: Password fails to meet password requirement  1. Return to step 2  2. User will have to enter a new password |
| Exceptions | 1. There is an existing account with the same email address .  2. Users input password does not meet password requirements:   * Minimum of 8 characters * Mix of alphanumeric characters |
| Includes: |  |
| Special Requirements: |  |
| Assumptions: |  |
| Notes and Issues: |  |

Use Case 3: Stock

| Use Case ID: | Use Case 3 | | |
| --- | --- | --- | --- |
| Use Case Name: | Stock | | |
| Created By: |  | Last Updated By: |  |
| Date Created: | 24/03/2022 | Date Last Updated: |  |

| Actor: | User and System |
| --- | --- |
| Description: | This use case depicts how the system records food purchased and how users can view the information. Simple retrieval of data from database |
| Preconditions: | 1. User’s device should have access to the Internet(WIFI/4G)  2. User is already logged into account |
| Postconditions: | 1. New record is updated to the user account successfully. |
| Priority: | High |
| Frequency of Use: | High |
| Flow of Events: | 1. User clicks on “Stock”  2. User is redirected to Stock page by system  3. System loads data of items balance in food storage  4. User can view the details of food items purchased  4.1 Nutrient information, date, item name and balance item will be shown.  4.2 User can click on item name to get the specific nutrient information (refer to viewingredient use case) |
| Alternate Flows: |  |
| Exceptions |  |
| Includes: |  |
| Special Requirements: |  |
| Assumptions: |  |
| Notes and Issues: |  |

Use Case 4: ViewIngredient

| Use Case ID: | Use Case 4 | | |
| --- | --- | --- | --- |
| Use Case Name: | ViewIngredient | | |
| Created By: |  | Last Updated By: |  |
| Date Created: | 24/03/2022 | Date Last Updated: |  |

| Actor: | User and System |
| --- | --- |
| Description: | This use case depicts how a user ask the system to query spoonacular to get the detail nutrient information of an item |
| Preconditions: | 1. User’s device should have access to the Internet(WIFI/4G)  2. User must be logged into account |
| Postconditions: | 1. Detail nutrient information of food items will be displayed |
| Priority: | High |
| Frequency of Use: | High |
| Flow of Events: | 1. User has logged in  2. User click on item in stock table or item in purchase item search result list  3. System get spoonacular ingredient id  4. System uses ingredient id and quantity purchase to api call spoonacular  5. Spoonacular return detailed nutrient information and display |
| Alternate Flows: |  |
| Exceptions |  |
| Includes: |  |
| Special Requirements: |  |
| Assumptions: |  |
| Notes and Issues: |  |

Use Case 5: Purchase

| Use Case ID: | Use Case 5 | | |
| --- | --- | --- | --- |
| Use Case Name: | Purchase | | |
| Created By: |  | Last Updated By: |  |
| Date Created: | 24/03/2022 | Date Last Updated: |  |

| Actor: | User and System |
| --- | --- |
| Description: | This use case depicts user can search for a item they had purchased or going to purchase |
| Preconditions: | 1. User’s device should have access to the Internet(WIFI/4G)  2. User must be logged into account |
| Postconditions: | 1. A list of items matching the user food item search input is displayed |
| Priority: | Medium |
| Frequency of Use: | High |
| Flow of Events: | 1. user is logged in  2. user clicks on purchase button  3. System redirects user to purchase page  4. user enters an item into search box  5. System queries spoonacular based on user item string  6. System displays result in the form of a list |
| Alternate Flows: | AC 5: fail to query  1. System will return an empty list |
| Exceptions |  |
| Includes: |  |
| Special Requirements: |  |
| Assumptions: |  |
| Notes and Issues: |  |

Use Case 6: AddIngredient

| Use Case ID: | Use Case 6 | | |
| --- | --- | --- | --- |
| Use Case Name: | AddIngredient | | |
| Created By: |  | Last Updated By: |  |
| Date Created: | 24/03/2022 | Date Last Updated: |  |

| Actor: | User and System |
| --- | --- |
| Description: | This use case depicts how user can add an item to stocks |
| Preconditions: | 1. User’s device should have access to the Internet(WIFI/4G)  2. User must be logged into account |
| Postconditions: | 1. stock table will be updated with the new item |
| Priority: | Medium |
| Frequency of Use: | High |
| Flow of Events: | 1. Continue from use case 5  2. user clicks items in purchase search list  3. System redirects user to itemsView page with an additional option to add item into stock  4. User clicks on add  5. System updates database and stock table will be update |
| Alternate Flows: |  |
| Exceptions |  |
| Includes: |  |
| Special Requirements: |  |
| Assumptions: |  |
| Notes and Issues: |  |

Use Case 7: DeletefromPurchase

| Use Case ID: | Use Case 7 | | |
| --- | --- | --- | --- |
| Use Case Name: | delete from purchase | | |
| Created By: |  | Last Updated By: |  |
| Date Created: | 24/03/2022 | Date Last Updated: |  |

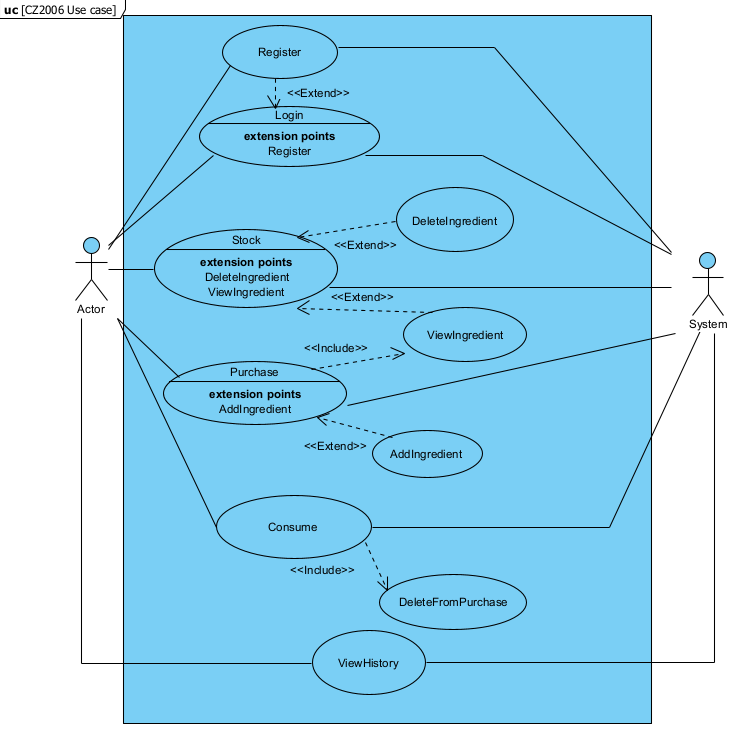
| Actor: | User and System |
| --- | --- |
| Description: | This use case depicts how a user consume certain quantity of items he or she has purchase |
| Preconditions: | 1. User’s device should have access to the Internet(WIFI/4G)  2. User must be logged into account |
| Postconditions: | 1. Stock balance will be reduced  2. History of consumption is recorded |
| Priority: | Medium |
| Frequency of Use: | Low |
| Flow of Events: | 1 .User clicks on “consume” in navigation bar  2, System redirects user to page  3. User select item and quantity user is going to consume  4. User press consume at bottom of page  5. System updated item quantity in stock table  6. This consumption will be recorded by the system |
| Alternate Flows: | AC 4: consumption amount larger than balance  1. Error message will be shown  2. Return to step 3 |
| Exceptions | 1. No food item was selected before the user clicks on the “finish” button |
| Includes: |  |
| Special Requirements: |  |
| Assumptions: |  |
| Notes and Issues: |  |

Use Case 8: View history

| Use Case ID: | Use Case 8 | | |
| --- | --- | --- | --- |
| Use Case Name: | View history | | |
| Created By: |  | Last Updated By: |  |
| Date Created: | 24/03/2022 | Date Last Updated: |  |

| Actor: | User and System |
| --- | --- |
| Description: | This use case depicts how the system can display purhase item history and consumption history |
| Preconditions: | 1. User’s device should have access to the Internet(WIFI/4G)  2. User must be logged into account |
| Postconditions: | 1. History is displayed on the page |
| Priority: | Low |
| Frequency of Use: | High |
| Flow of Events: | 1. User clicks on history on the logged in navigation bar  2. System retrieve records  3. Records will show items purchased and consumed and their nutrient information |
| Alternate Flows: |  |
| Exceptions |  |
| Includes: |  |
| Special Requirements: |  |
| Assumptions: |  |
| Notes and Issues: |  |

**Use Case Diagram:**

****