Nutrition Tracker

Software Requirements Specification

Version 1

10/5/2018

Allen Poon, Brendan Lestander,
Kyle Horner, Sai Konduru

Software Engineers

Prepared for
CS1530: Software Engineering

Revision History

Date	Description	Author	Comments
10/5/2018	Version 1	Team	Sprint 0 Finalization

Document Approval

The following Software Requirements Specification has been accepted and approved by the following:

Signature	Printed Name	Title	Date
	<Your Name>	Software Eng.	

Table of Contents

1. Introduction

1.1 Purpose

Nutrition Tracker is an app to track daily caloric intake and provide an overview of it. The target audience is everyone.

1.2 Scope

1. Nutrition Tracker
2. Allow the user to keep track of their daily nutrition
3. The app is meant to help the user stay on track dietary wise and track their progress to reach their weight goals. It is also meant to help them be more aware of what they're eating daily.

1.3 Definitions, Acronyms, and Abbreviations

NND - National Nutrient Database

2. General Description

2.1 Product Perspective

Nutrition Tracker has competition out there such as MyFitnessPal, which is another popular diet tracker. The way Nutrition Tracker differs is through the simple layout of the app and the ability to generate graphs to provide an overview of the user's progress.

2.2 Product Functions

- Search for food item
- Add a food item to the daily intake
- Update current weight
- Input weight goal
- View progress graphs for caloric intake and weight

2.3 Constraints, Assumptions and Dependencies

- Constraints
    - Can only make 3600 requests per hour to NND REST API
- Assumptions
    - User has an internet connection
- Dependencies
    - Firebase
    - Google Auth service
    - NND REST API
    - SQLite
    - Flutter

3. Specific Requirements

3.1 External Interface Requirements

3.1.1 Software Interfaces

3.2 Functional Requirements

3.2.1 User Searches for Foods

User can search for foods that they ate or plan to eat. This will result in either a total view of the nutritional facts about that food item or allow the user to enter the nutrition facts.

3.2.2 User Can put foods into Categories

Any food that the user searches can be added into four separate categories: breakfast, lunch, dinner and snack.

3.2.3 Show Error message with options when Food item not found

Anytime the user searches for a food item that isn't found in the database, the application will show an error message saying "Food Item not found" and will give the user the option to make a custom food item.

3.2.4 User can add Custom Food Item

Anytime the user searches for a food item that isn't found in the database, the application will allow the user to add a custom food item. App will prompt the user for the name of the food item and will ask for the nutrition facts of the item. This information will be stored locally.

3.2.5 User can see Nutrition Facts

When the user searches for food Item from the National Nutrition Database, the application will show them the nutrition facts about the food item.

3.2.6 User can set Height and Weight

When user first signs up for the application they can set their current height and weight. These numbers can be changed at any time.

3.2.7 User can set caloric goals

If the user does not want to use the recommended caloric goals they have the option to input their own value.

3.2.8 User can set weight goals

The user is able to input a weight goal, and this weight goal is factored into the user's generated caloric goal. This can be changed at any time by the user.

3.2.9 User can login through Google accounts

The user is able to use an existing Google account to login to the app.

3.2.10 User can log out at any point in the app

All screens of the app will allow the user to log out.

3.2.11 User can view Progress on Caloric goals on a weekly and daily basis via a line graph

User can navigate to a caloric goal graph page, where they will be able to select a daily or weekly graph. The graph will have calories on the y-axis and time on the x-axis.

3.2.12 User can view weight goals progress on a weekly to monthly basis via a line graph

User can navigate to a weight goal graph page, where they will be able to select a weekly or daily graph. The graph will have their recorded weight on the y-axis and the date it was recorded on the x-axis.

3.2.13 User can take a before and after picture

The user will be able to store before and after pictures of themselves so they can see their weight loss progress.

3.2.14 User can view the weight and height units in either imperial or metric

Users will be able to navigate to a screen where they can select which unit they would prefer to use. This will affect all weight and height values that are displayed.

3.3 Use Cases

1 Priority = Lowest
5 Priority = Highest

3.3.1 Search Food Item

Use Case ID:	SEARCH		
Use Case Name:	Search Food Item		
Created By:		Last Updated By:	
Date Created:	9/16/2018	Date Last Updated:	10/5/2018

Actor:	User
Description:	User searches for a food item using the in app search feature. The search returns an item found in either the custom food list or the National Nutrition Database.
Preconditions:	User is logged in, clicks the menu button, and clicks the search button.
Postconditions:	User is given the result of the search in the form of a list. User can select a food item from the list of results. Once selected user can see the nutrition facts of the selected item.
Priority:	5
Frequency of Use:	Estimated about 10 times per day (depends on the user's diet)
Normal Course of Events	User is logged in, clicks the menu button, clicks the search button, enters the name of a food item, and hits the search. The app searches the local custom food items list, then searches the

	custom food list when food item not found in NND. The app then returns results from custom list or NND.
Alternative Courses:	Food item isn't found in both the local SQLite database and NND database. User is given the option to enter a custom food entry or search for a new food item.
Exceptions:	Unable to connect to NND.
Includes:	
Special Requirements:	App queries the local SQLite database after querying the NND database through the REST API
Assumptions:	None
Notes and Issues:	None

3.3.2 User Log Out

Use Case ID:	LOGOUT		
Use Case Name:	User Log Out		
Created By:		Last Updated By:	
Date Created:	10/3/2018	Date Last Updated:	10/5/2018

Actor:	User
Description:	User logs out of the app using a google account
Preconditions:	User is logged in.
Postconditions:	User is signed out of their account. User is shown login screen.
Priority:	3
Frequency of Use:	Estimated about 3 times per day
Normal Course of Events	User clicks menu and selects log out. Request is sent to Google and user is logged out. User will view the login screen.
Alternative Courses:	

Exceptions:	Google Client doesn't allow user to log out.
Includes:	None
Special Requirements:	None
Assumptions:	None
Notes and Issues:	None

3.3.3 View graph

Use Case ID:	VIEW_GRAPH		
Use Case Name:	View graph		
Created By:		Last Updated By:	
Date Created:	9/16/2018	Date Last Updated:	10/5/2018

Actor:	User
Description:	User clicks on the "View Graph" button to view the caloric and weight goal graphs.
Preconditions:	User is logged in, clicks on the menu button, clicks the "View Graph" button
Postconditions:	User is taken to a page where two graphs are displayed for caloric and weight progress.
Priority:	3
Frequency of Use:	1 time per week
Normal Course of Events	User is logged in, clicks on the menu button, clicks the "View Graph" button, app will calculate the caloric and weight progress graphs by pulling the string data from Firebase and perform string arithmetic to pull the numeric data out, then plot the two graphs with that data.

Alternative Courses:	User failed to enter weight data. App will redirect to the enter weight screen with an error message saying that a graph cannot be displayed before the user enters the wait.
Exceptions:	None
Includes:	None
Special Requirements:	Minimize the number of times the graphs are calculated
Assumptions:	User logs their food intake and weight progress.
Notes and Issues:	Come up with a way to cache the graph data to reduce the number of times it is calculated

3.3.4 Enter current weight

Use Case ID:	INPUT_WEIGHT		
Use Case Name:	Enter current weight		
Created By:		Last Updated By:	
Date Created:	9/16/2018	Date Last Updated:	10/5/2018

Actor:	User
Description:	User inputs their current weight to log their progress.
Preconditions:	User is logged in, clicks the menu button, clicks the “Stats” button.
Postconditions:	User’s old weight is appended to the “Archive Weight” key/pair value in Firebase, newly entered weight is recorded under the “Current Weight” key/pair value.
Priority:	3
Frequency of Use:	1 time per day
Normal Course of Events	User is logged in, clicks the menu button, clicks the “Stats” button, user enters current weight and hits “Submit”. User’s old weight is appended to the “Archive Weight” key/pair value in Firebase, newly

	entered weight is recorded under the “Current Weight” key/pair value.
Alternative Courses:	No negative values nor letters for the input.
Exceptions:	Unable to connect to Firebase.
Includes:	None
Special Requirements:	None
Assumptions:	None
Notes and Issues:	None

3.3.5 Enter current height

Use Case ID:	INPUT_HEIGHT		
Use Case Name:	Enter current height		
Created By:		Last Updated By:	
Date Created:	9/19/2018	Date Last Updated:	10/5/2018

Actor:	User
Description:	User inputs their current height.
Preconditions:	User is logged in, clicks the menu button, clicks the “Stats” button.
Postconditions:	User’s newly entered height is recorded under the “Current Height” key/pair value.
Priority:	1
Frequency of Use:	1 time (more if growing)
Normal Course of Events	User is logged in, clicks the menu button, clicks the “Stats” button, user enters current height and hits “Submit”. User’s newly entered height is recorded under the “Current Weight” key/pair value.
Alternative Courses:	No negative values nor letters for the input.

Exceptions:	Unable to connect to Firebase.
Includes:	None
Special Requirements:	None
Assumptions:	None
Notes and Issues:	None

3.3.6 User Login

Use Case ID:	LOGIN		
Use Case Name:	User Login		
Created By:		Last Updated By:	
Date Created:	9/19/2018	Date Last Updated:	10/5/2018

Actor:	User
Description:	User logs in to the app using a google account
Preconditions:	No user is logged in.
Postconditions:	User is signed in to their account. The user's historic weight and caloric data is loaded from the database. The home page is shown.
Priority:	5
Frequency of Use:	Estimated about 3 times per day
Normal Course of Events	User opens the app, inputs their credentials and submits them. This is sent to Google sign-in, which authenticates and returns whether the credentials are valid. If so the user is signed in and shown their homepage. If the credentials are not valid then the app returns to the login screen.
Alternative Courses:	User enters invalid credentials, causing sign-in to fail. An error message is displayed to that effect.
Exceptions:	Unable to connect to google.

Includes:	None
Special Requirements:	None
Assumptions:	None
Notes and Issues:	None

3.3.7 User enters caloric goal

Use Case ID:	INPUT_CAL_GOAL		
Use Case Name:	User enters caloric goal		
Created By:		Last Updated By:	
Date Created:	9/19/2018	Date Last Updated:	10/5/2018

Actor:	User
Description:	User inputs their caloric goal.
Preconditions:	User is logged in, clicks the menu button, clicks the “Stats” button.
Postconditions:	Caloric goal is updated for the user.
Priority:	2
Frequency of Use:	About 1 time per month
Normal Course of Events	User logs in to their account, opens the menu, and selects “Stats”. User inputs their caloric goal as an integer. The caloric goal is sent to Firebase. User is returned to home page.
Alternative Courses:	User enters a value that is not an integer.
Exceptions:	Unable to connect to Firebase.
Includes:	None
Special Requirements:	None
Assumptions:	None
Notes and Issues:	None

3.3.8 User enters weight goal

Use Case ID:	INPUT_WEIGHT_GOAL		
Use Case Name:	User enters weight goal		
Created By:		Last Updated By:	
Date Created:	9/19/2018	Date Last Updated:	10/5/2018

Actor:	User
Description:	User inputs whether their goal is to gain or lose weight.
Preconditions:	User is logged in, clicks menu button,and clicks “Stats” button.
Postconditions:	User’s “Goal” value is updated to Yes if they select gain, or No if they select lose.
Priority:	2
Frequency of Use:	About 1 time per month
Normal Course of Events	User logs in to their account, opens the menu, and selects “Stats”. User’s goal variable is updated in Firebase.. User is returned to home page.
Alternative Courses:	User enters a value that is not an integer.
Exceptions:	Unable to connect to Firebase.
Includes:	None
Special Requirements:	None
Assumptions:	None
Notes and Issues:	None

3.3.9 User adds custom food

Use Case ID:	ADD_CUSTOM_FOOD		
Use Case Name:	User adds custom food.		
Created By:		Last Updated By:	
Date Created:	9/19/2018	Date Last Updated:	10/5/2018

Actor:	User
Description:	User creates a food item not found in the NND and saves it.
Preconditions:	User is logged in, clicks the menu button, clicks search, searches for an invalid food item and clicks “add custom food item” option.
Postconditions:	Newly created food item will be saved locally along with all other custom food items created on this device.
Priority:	4
Frequency of Use:	About 1 time per month
Normal Course of Events	User logs in to their account, searches for an invalid food item, Selects “Add Custom Food Item”. User then inputs the name and calorie value associated with the item. User is returned to home page.
Alternative Courses:	User attempts submit without filling out the form.
Exceptions:	Unable store on local device.
Includes:	None
Special Requirements:	None
Assumptions:	User has easy access to the caloric information of the custom food item.
Notes and Issues:	Invalid food item is defined as food item not found in the custom food list and the National Nutrition Database.

3.3.10 View nutrition facts

Use Case ID:	VIEW_NUTRITION		
Use Case Name:	View nutrition facts		
Created By:		Last Updated By:	
Date Created:	9/19/2018	Date Last Updated:	10/5/2018

Actor:	User
Description:	User views nutrition facts for a food item
Preconditions:	User is logged in, clicks the menu button, clicks the “Search” button, clicks a food item
Postconditions:	User is displayed the nutrition facts for a food item
Priority:	4
Frequency of Use:	Once per search
Normal Course of Events	User is logged in, clicks the menu button, clicks the “Search” button, clicks a food item, nutrition facts are pulled from NND or custom list and is displayed to the user
Alternative Courses:	None
Exceptions:	None
Includes:	None
Special Requirements:	None
Assumptions:	Search feature works as intended
Notes and Issues:	None

3.3.11 User takes picture for before/after photo

3.3.12 User adds food to his current day caloric intake

3.3.13 User selects a category for a certain food

3.3.14 User changes units from pounds to kilograms or vice versa

3.4 Classes / Objects

3.4.1 User

3.4.1.1 Attributes

- Int CurrentHeight
- Int CurrentWeight
- Map<Date, int> ArchiveWeight
- Photo CurrentPhoto
- Photo BeforePhoto
- Int Goal
- Bool Metric
- DailyCal cal

3.4.1.2 Methods

- getCurrentHeight()
- setCurrentHeight(int newHeight)
- getCurrentWeight()
- setCurrentWeight(int newWeight)
- getCurrentPhoto()
- setCurrentPhoto(Image newPhoto)
- getGoal()
- setGoal(int newGoal)
- getMetric()
- setMetric(bool Metrict)
- getArchiveWeight()
- getBeforePhoto()
- getDailyCal()
- addTodaysCal(int day, FoodItem ft)
- toString()

3.4.2 FoodItem

3.4.2.1 Attributes

- String name
- Int calories
- Int protein
- Int carbs
- Int fat

3.4.2.2 Methods

- getName()
- getCalories()
- getMacros()
    - Returns Protein, carbs, and fat

3.4.3 Graph

3.4.3.1 Methods

- showWeightGraph(User user)
- showCalorieGraph(User user)

3.4.4 RestApiNND

Object responsible for communicating with NND

3.4.1.1 Attributes

- String apiKey;
- JSON Response

3.4.1.2 Methods

- getResponse(String searchParam)
- parseResponse()

3.4.5 Get

3.4.5.1 Attributes

- String foodName

3.4.5.2 Methods

- toString()

3.4.6 Photo

3.4.6.1 Attributes

- Image currentPhoto;

3.4.6.2 Methods

- toString()

3.4.6 DailyCal

3.4.6.1 Attributes

- FoodItem[] dailyFood

3.4.6.2 Methods

- addFoodItem(FoodItem food)
- getDailyCal(str day)

3.4.6 CustomList

3.4.6.1 Attributes
- Database sqlLite

3.4.6.2 Methods
- searchCustomList(String searchParam)
- addCustomItem(FoodItem fi)

3.5 Non-Functional Requirements

3.5.1 Performance

3.5.1.1 Average search time is less than 1 second
When a user searches for a food item the back-end NND search takes less than 1 second. This doesn't account for any latency in the users connection.

3.5.1.2 Graphs dynamically adjust axis values to best display data
When the user changes graphs they will automatically adjust the axis values so that the graph is not too small or large. This will ensure that users are able to gather useful information from the graph features.

3.5.2 Reliability

3.5.2.1 MTBF value greater than 24 hours
The mean time between failures of the app is greater than 24 hours. This ensures that users are not constantly experiencing crashes.

3.5.3 Availability

3.5.3.1 Daily feed will be cached into a local database until an update is made
The users daily nutrition data will be stored locally so the user can access this information without an internet connection.

3.5.3.2 Greater than 99% uptime
The user should be able to access all functions of the app at least 99% of the time as long as they have internet connection. As the app is relatively simple there should be few reasons why it would not function properly.

3.5.4 Security

3.5.4.1 Allow users to save login credentials for quick access

Users are able to select a checkbox so that the app remembers their credentials. These credentials must be encrypted and stored securely.

3.5.5 Maintainability

3.5.5.1 Application Testability

The application is developed in such a way that allows for ease of testing.

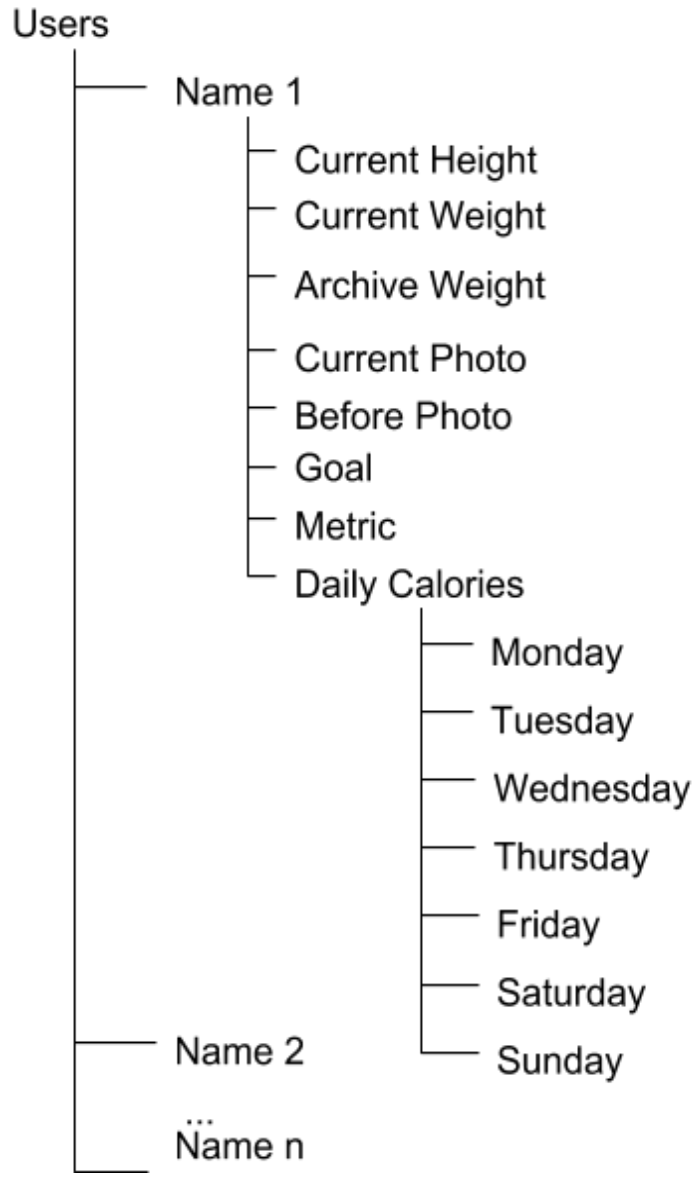
3.5.6 Portability

3.5.6.1 Both Android and iOS apps are supported

By developing the app with Flutter, Android and iOS apps can be created concurrently. This allows the user flexibility in how they can use the app.

3.6 Logical Database Requirements

The implementation will contain two databases, one remote database that will store all user information, and a local database that will store custom food items and a cache of current day feed, that have been added by the user. The user database will be created using Firebase, it is non-relational, and the data will be accessed using google accounts. The data structure is represented by the image below.



The local custom food database will be relational and use SQLite. It will contain one table with the food item names and all nutrition data that the user enters. The data will be accessed by SQL queries whenever a search is performed.
Another table will be added to the SQLite database to contain the current day feed. This will contain the same information as the other table but be smaller and be purged multiple times a day to update the info on Firebase.

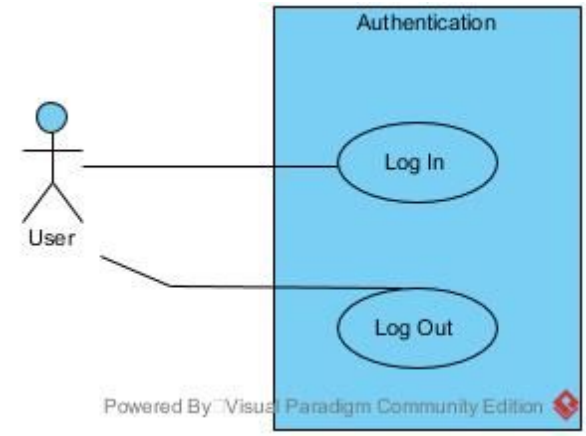
3.7 Other Requirements

Catchall section for any additional requirements.

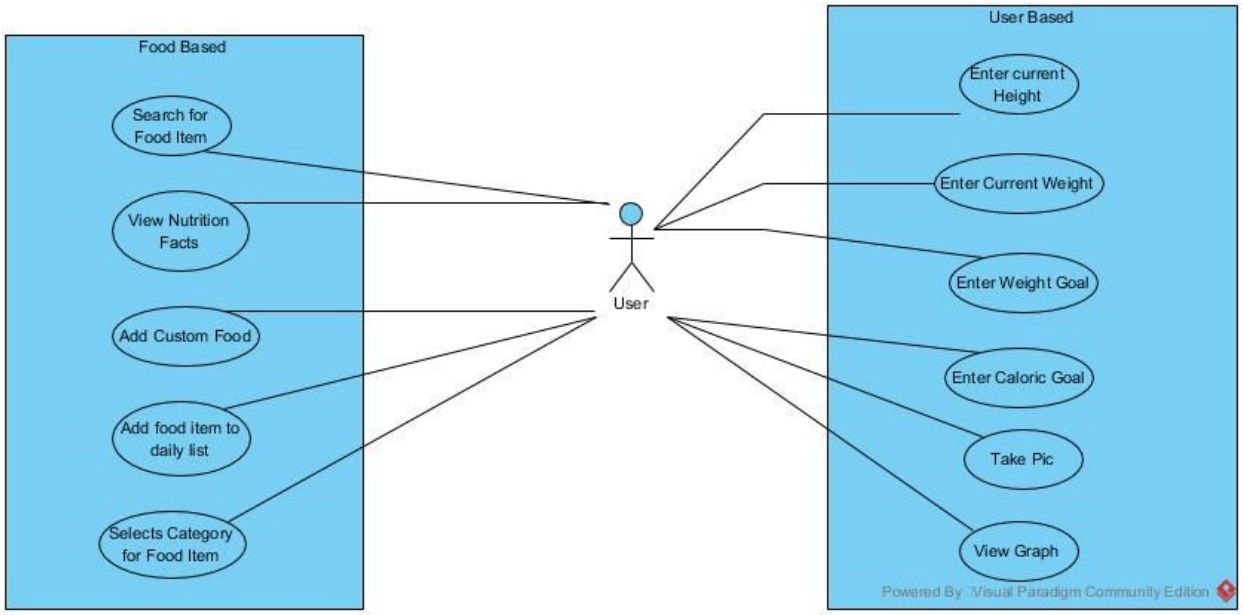
4. Analysis Models

4.1 Use Case Diagrams

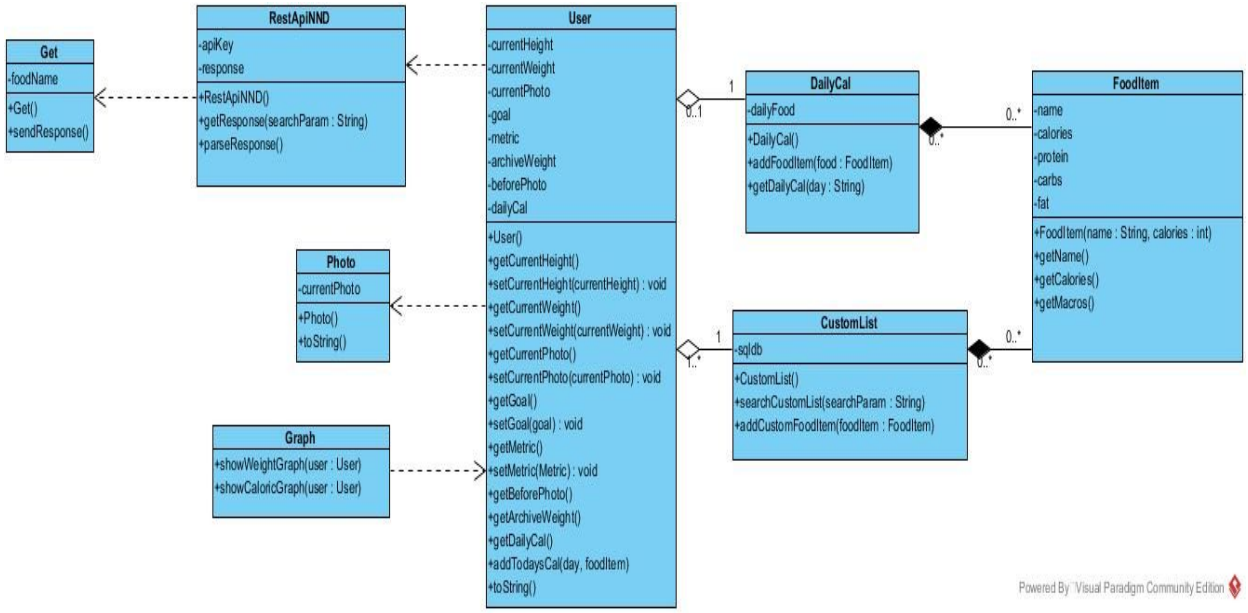
Log In



General Use

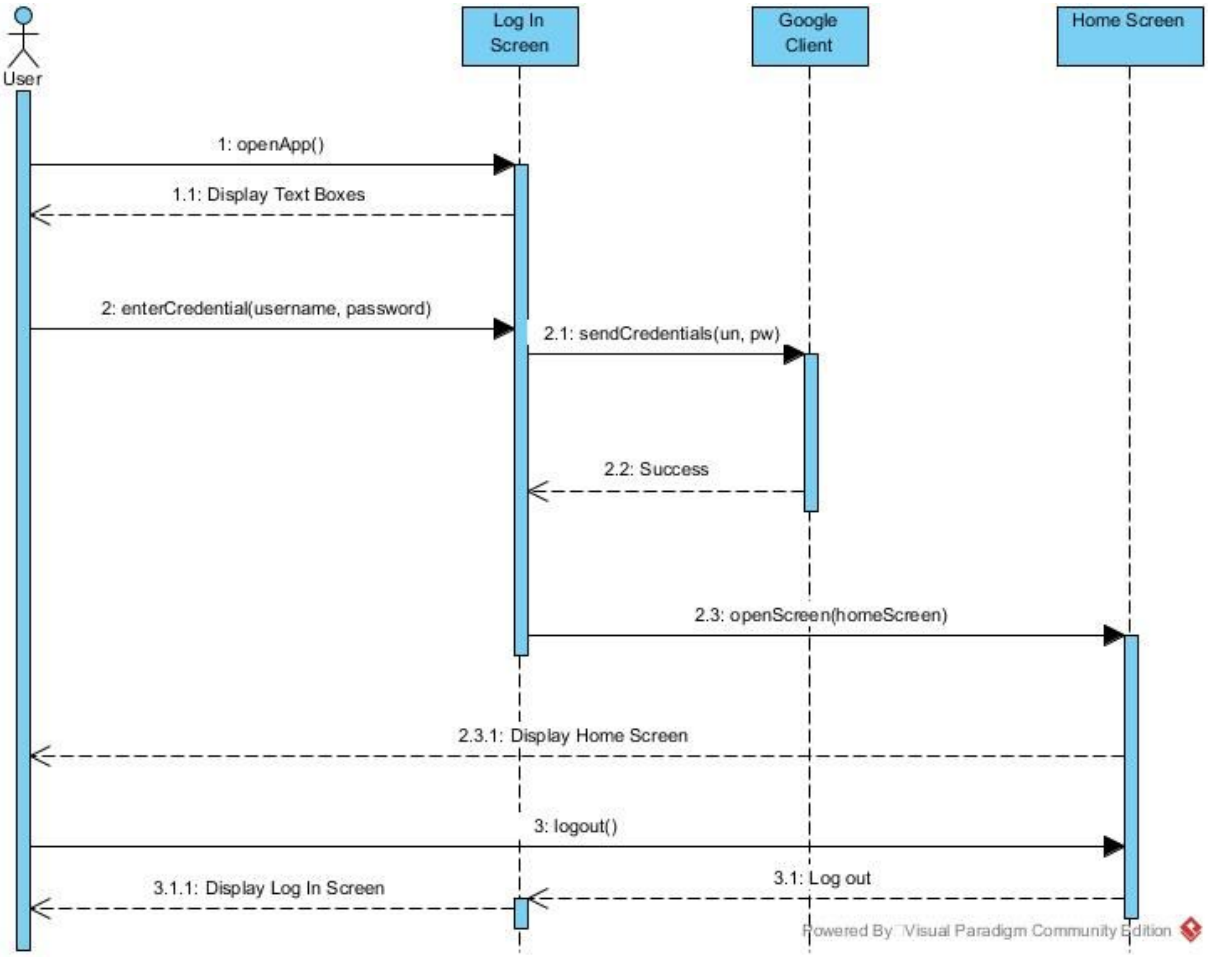


4.2 Class Diagrams

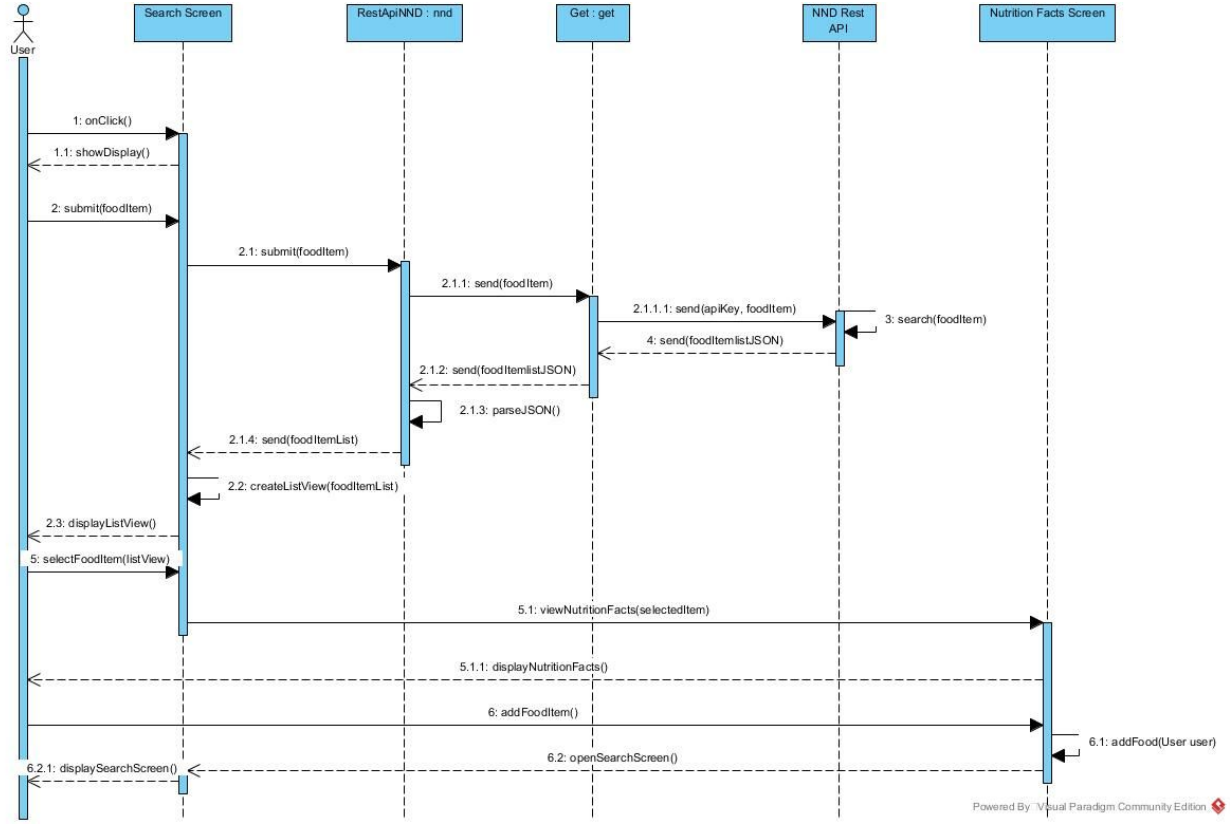


4.3 Sequence Diagrams

Log In

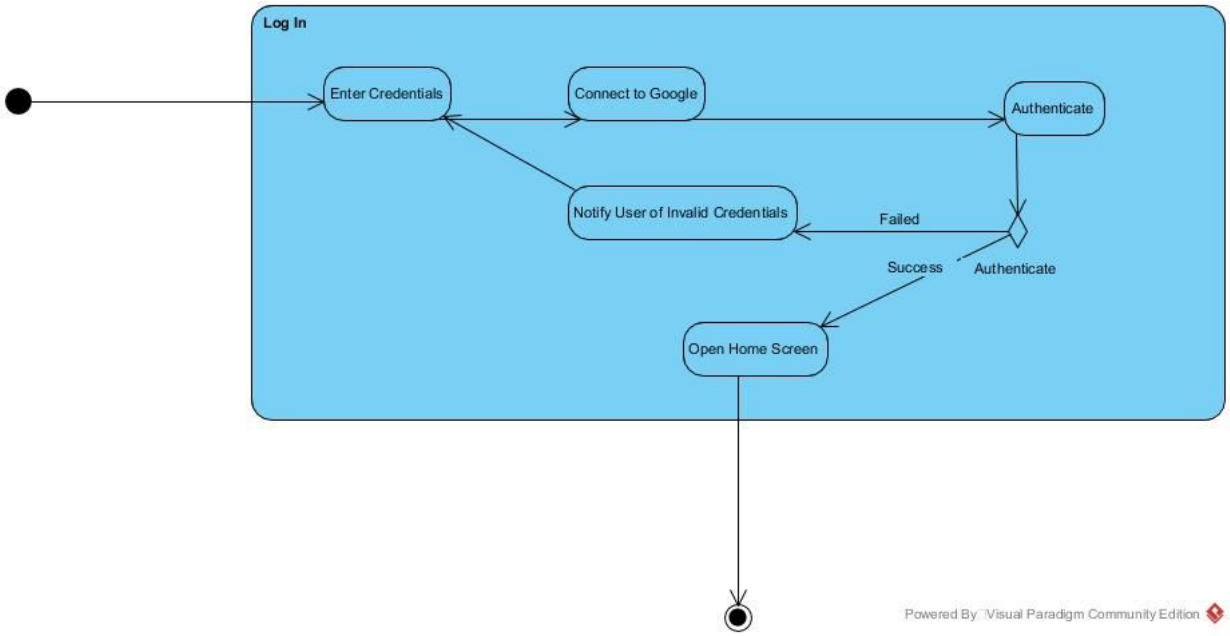


Search/Add Food Item

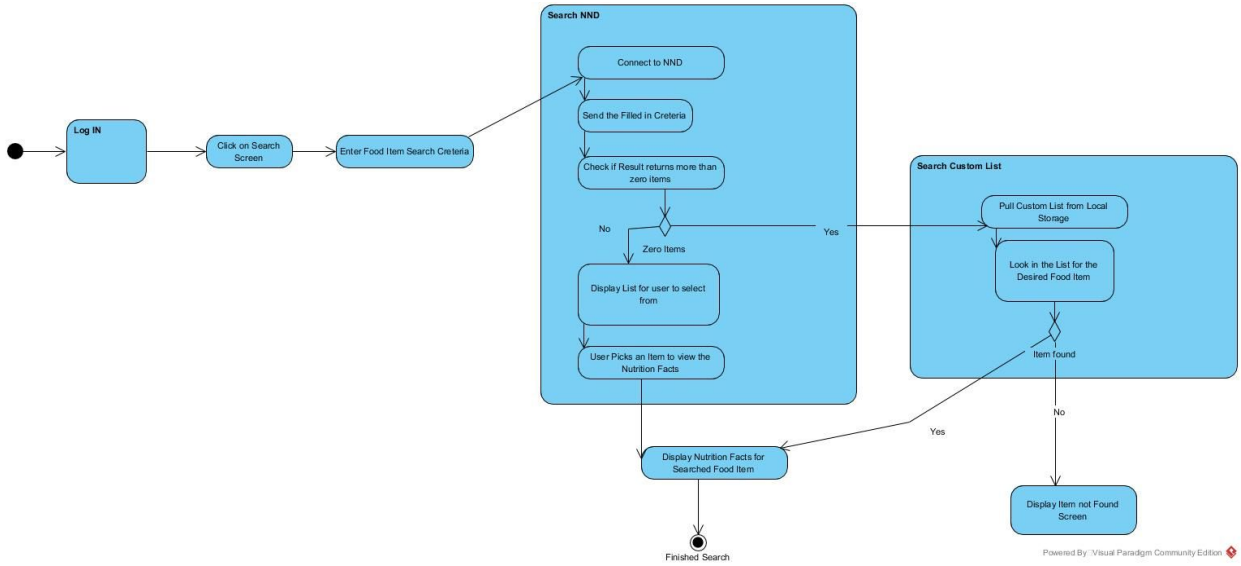


4.4 Activity Diagrams

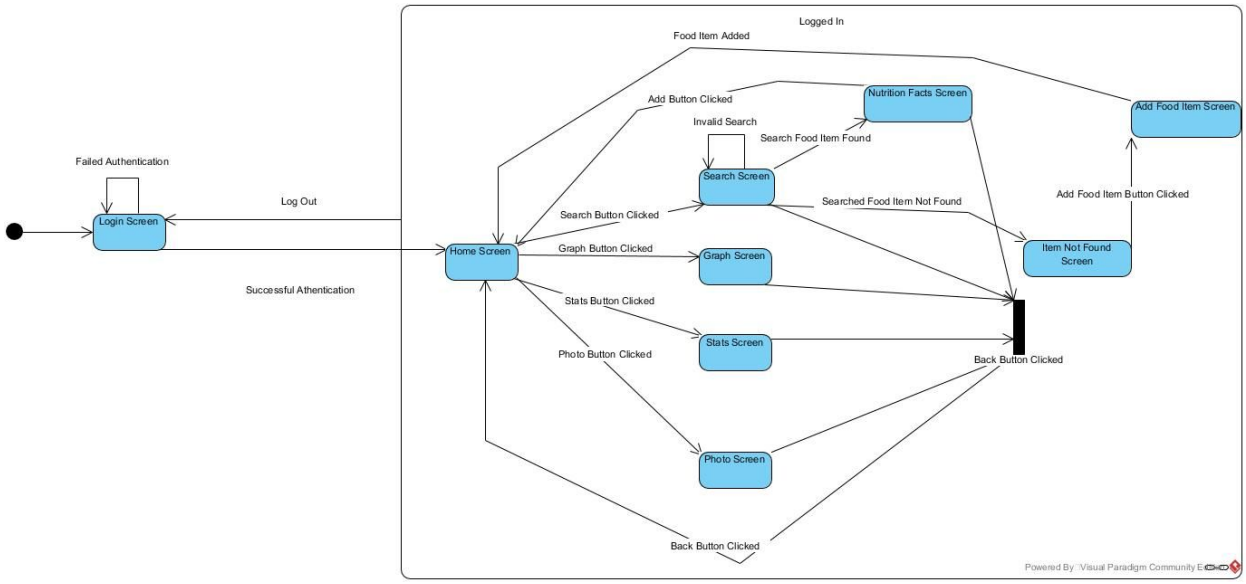
Log In



Search



4.5 State-Transition Diagrams (STD)



5. Change Management Process

Updates to the SRS can come from any member of the group, and must be approved by a majority(3) of the group members. Updates can be added directly to the document with a comment until they are approved by the group.