# Affamato

**Team**:

Canvas Group: **Falcon**

**Members**:


- Cameron Clark: cameron.clark0821@utexas.edu \\ github.com/cameronclark0821
- Justin Henry: justinhenry@utexas.edu \\ github.com/justinhenry
- Alex Issa: alex.issa32@utexas.edu \\ github.com/alexissa32
- Julia Rebello: julialrebello@utexas.edu \\ github.com/JLRebello
- Samir Riad: sriad123@utexas.edu \\ github.com/sriad123
- Rooshi Patidar: rooshipatidar@utexas.edu \\ github.com/rooshimadethis


**URL** to Github/Gitlab repo and shared Google docs:

- https://github.com/alexissa32/Affamato
- https://drive.google.com/drive/folders/0ANUp-cLx6lnZUk9PVA

**Vision**:

What recipes can I make with the food I have? How long until this food expires? What recipes can I make that are rich in Vitamin C that I already have most of the ingredients for? Affamato has the answers.

Our vision is for Affamato to be a smart-cooking and health assistant. Affamato is a web application that helps users do a multitude of food-related tasks. It allows users to create grocery lists, track their food inventory and perishability, and recommend recipes based on many different categories. The purpose of Affamato is to help people save money, reduce their food waste, organize their food stock, and broaden their palates. There are websites that encompass partial functionality of our product, but this will be the first to combine and expand these ideas. *Fridge Pal* and *myfitnesspal* are the two closest concepts to ours. *Fridge Pal* is an inactive app, but it implemented pantry and perishability tracking, shopping lists, and a database of recipes. The *myfitnesspal* app is widely used; however, it is primarily a fitness and nutrition tracking app.

Affamato will be a much more detailed pantry and food tracker. Users will have information on item expiration, tips on storage, disposal, and kitchen-care, and recipe recommendations based on what food the user already has. Affamato will not only recommend recipes based on soon-to-be expired items or current inventory, but also categorically offer options based on desired cuisine, nutrition, and dietary restrictions. Food items will also have individual nutrition information available.

Affamato
Powered by Falcon Group

Our short-term goal is to implement the inventory, recipe, and tips components of the website. User waste reduction, organization, and money saving are the priorities. Think Spotify for recipes:

1. You listen to music and like/dislike it so it recommends new music. ↔ You have food in your inventory and some of it will expire soon, so Affamato recommends recipes you'd like that would use those items.
2. You are bored of your playlists and listen to one of Spotify's categorical music playlists. ↔You are bored of your recipes, and choose new ones based on categories like cuisine, different diets, dietary/allergy restrictions, and/or nutrition goals.
3. You save songs you like. ↔ You save recipes you like.

Long term, we would like to expand our food nutrition component into the overall health tracking sphere. We will have daily nutrition and exercise tracking, where Affamato will recommend things based on whatever goals a user has. Think Spotify for recipes plus *myfitnesspal*.

Alternatives:

*myfitnesspal* - Nutrition and exercise tracker.
*Fridge Pal* (inactive) - Organizes food inventory, has generic recipes.
*Epicurious* - Recipes.
*Fresh Box* - Proper food storage, expiration date notifications.

None of these alternatives are as expansive and detailed as our app. Most were also released circa 2012. We think in 2019, people are much more accepting of technology helping them do things, so the site should have a better chance of success.

**Competitive analysis**:

"*In the United States, food waste is estimated at between 30-40 percent of the food supply. This estimate, based on estimates from USDA's Economic Research Service of 31 percent food loss at the retail and consumer levels, corresponded to approximately 133 billion pounds and $161 billion worth of food in 2010.*" - United States Department of Agriculture (https://www.usda.gov/oce/foodwaste/faqs.htm)

Affamato aims to provide a way for users to keep track of their food inventory, so that they can use most, if not all, of the food they buy. This is an important goal that benefits us all, as users will save money and food waste will be reduced. In addition, Affamato also offers recipes based on on-hand or soon-to-be expiring items, which exposes users to new recipes that include familiar ingredients. Affamato is exciting because users who know different ways to cook with their ingredients are more likely to actually use them. Affamato will also provide tips on storage, disposal, cooking and general kitchen care, which will educate users on how best to preserve, cook with, and dispose of their food items, as well as care for their expensive pots and pans. No other application we've seen encompasses all these features.


**Data Sources, Scraping, and Database**:

1. https://www.programmableweb.com/category/all/apis?keyword=nutrition (API database)
2. https://spoonacular.com/ Spoonacular API (For recipes and nutrition facts)
3. https://developers.google.com/custom-search/v1/overview (For perishability facts)
4. http://api2.bigoven.com/web/console (For recipes, needs an account)
5. Additional sources may be needed/we may need to build out certain information for the database. Ex. The tips section for disposal and kitchen-care might be something we need to do ourselves.

We will make calls to these APIs with different food items, search the results for different facts that we want to store, and then store them in the database.

In terms of infrastructure and database, we will use GCP for our Cloud Platform as it has a better PaaS structure than AWS. In terms of the actual database, we will probably use a SQL DB (MySQL), over NoSQL (Would probably use MongoDB), but we aren't sure about that.

Our database will store information based on the following:

1. **User** - Will have a username, password, GroceryList, Inventory, list of favorite recipes, settings, etc.
2. **GroceryList** - Will contain FoodItems to be purchased and quantities of those items.
3. **Inventory** - Will contain FoodItems, quantity, and their time to expiration.
4. **Recipes** - Will contain FoodItems+quantities and a list of instructions to complete the recipe.
5. **FoodItems** - Will contain information on a food item. A general time to expiration and all nutrition information.
6. **Tips** - Storage, disposal, cooking, and kitchen-care. Can pop-up throughout the site based on keywords on the current page.

**Requirements**:

1. Save user's money.
    a. As a thrifty consumer, I would like to cut down on grocery shopping by efficiently using my groceries, so I am not re-purchasing items unnecessarily.
2. Reduce food waste.
    a. As a green consumer, I would like to reduce the amount of food I waste by tracking when items expire so that I may use all of the perishables that I purchase.
3. Calorie counter: lose weight, focusing on calories.
    a. As someone who would like to lead a healthier lifestyle, I would like to have quick access to the nutritional information of the food I buy.
4. Tries new recipes (For example, wanting cuisine type X).
    a. As an adventurous foodie, I would like to try new recipes.
5. Bulking: gain weight, focusing on maximizing volume of protein and healthy fats, getting more with less so one can fit it in the fridge/only need to get groceries once a week.


   a. As a bodybuilder, I would like to increase my muscle mass by consuming the right foods at a great enough volume.
6. Trying to follow a specific diet - keto, vegan, kosher, etc.
   a. As a someone who has dietary and health restrictions, I would like to have easy access to what foods I can eat, and what recipes I can make with them.

**Formal Use Case 1:**
Goal: User wants to see what recipes they can make with the food in their pantry
Primary actor: User
Precondition: User is at the login screen
Success end condition: List of potential recipes is displayed
Failure end condition: List of potential recipes is not displayed
Trigger: User logs into system

1. User enters account and password
2. User reaches home page
3. User enters Recipes Search Page
4. User selects “Search Recipes by Inventory”
5. User receives relevant recipes

Extensions:
1a. Password is incorrect
  1a1. System returns user to login screen
  1a2. User backs out or tries again
5a. Recipe not Found
  5a1. Systems notifies user that recipe was not found
  5a2. System returns user to Recipes Search Page

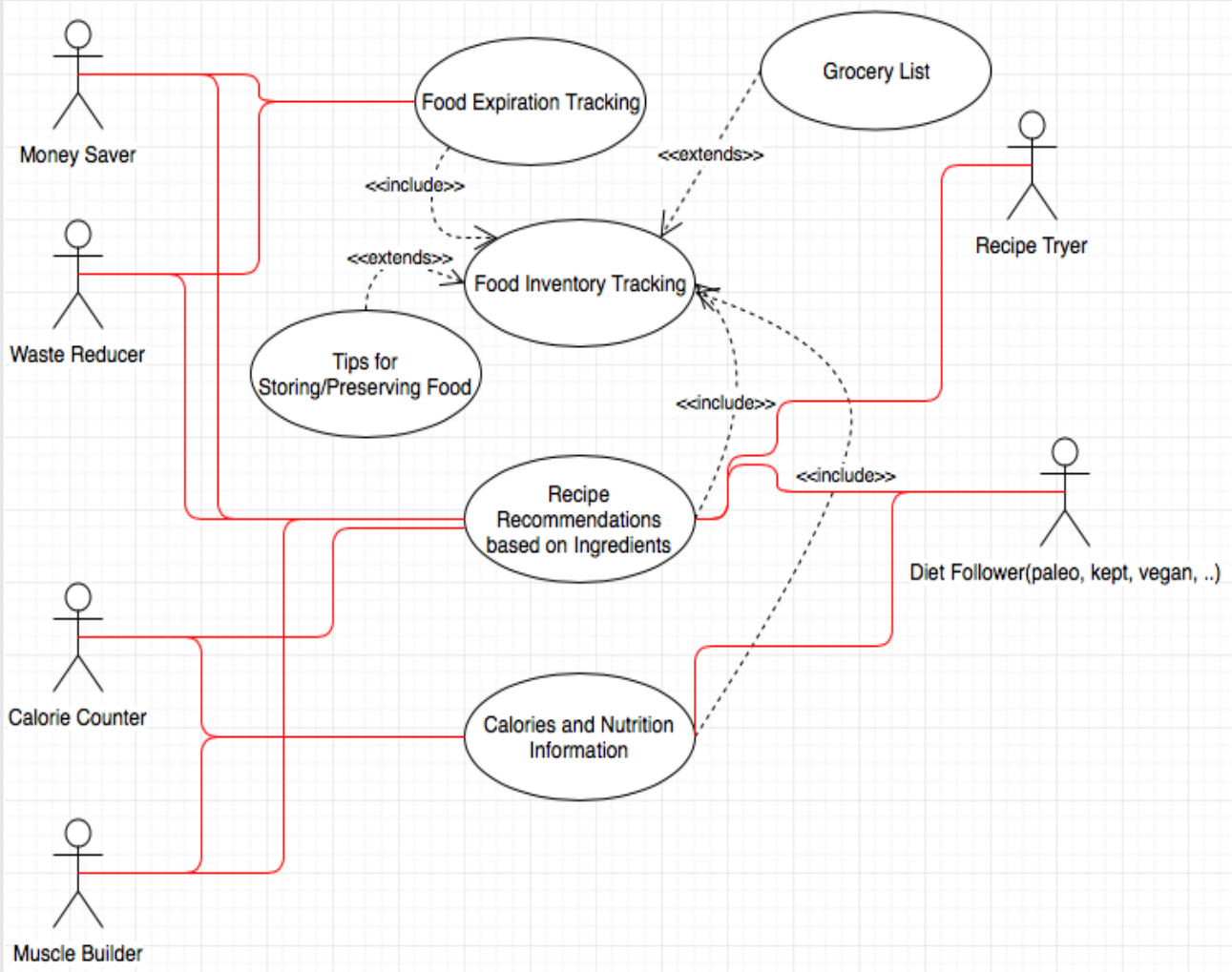
**Formal Use Case 2:**
Goal: User wants to add a food item to their pantry to see its shelf life
Primary actor: User
Precondition: User is at the login screen
Success end condition: Shelf life of food item displayed
Failure end condition: Shelf life of food item not displayed
Trigger: User logs into system

1. User enters account and password
2. User reaches home page
3. User enters inventory page
4. User enters food item and quantity
5. Item and its information including shelf life and nutrition are displayed in inventory

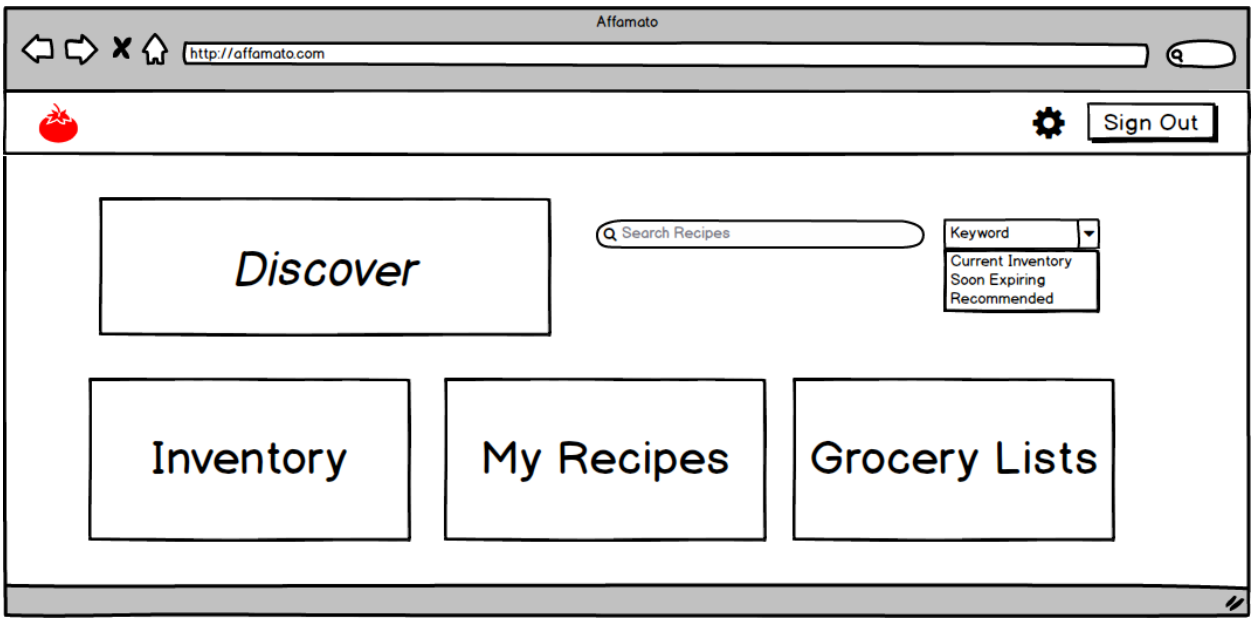
Extensions:
1a. Password is incorrect

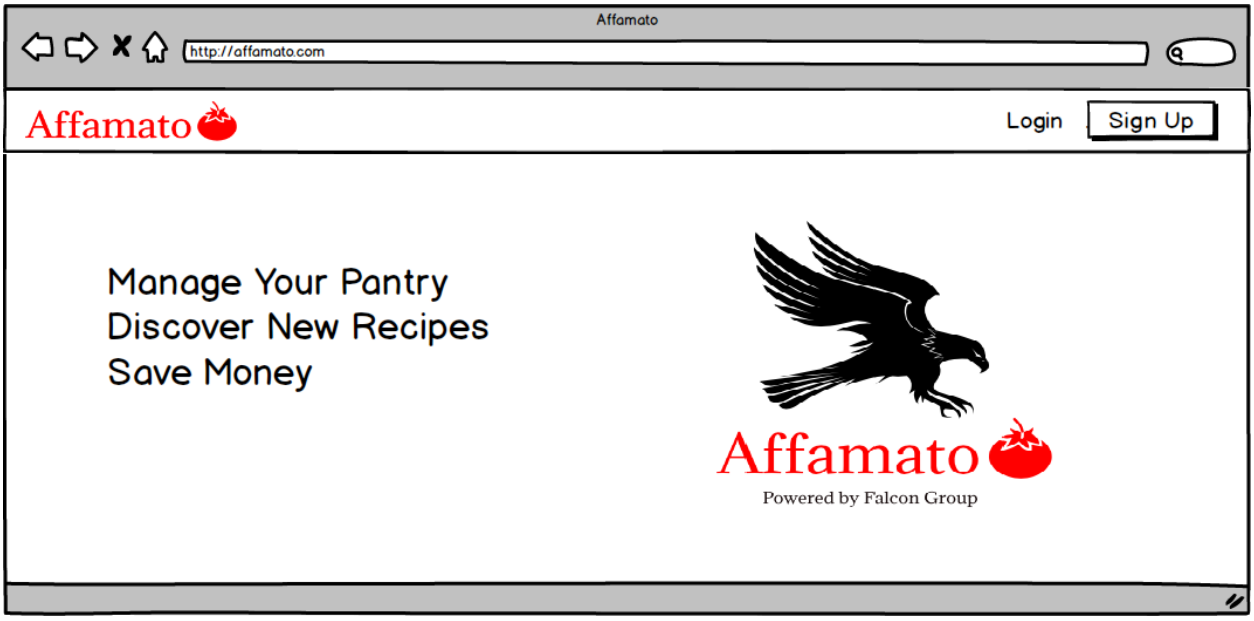
1a1. System returns user to login screen
1a2. User backs out or tries again
4a. Item not found
4a1. System notifies user that item was not found
4a2. System returns user to inventory page

**User Diagram**:



**Interface**:


Affamato
http://affamato.com
Sign Out
Discover
Search Recipes
Keyword
Current Inventory
Soon Expiring
Recommended
Inventory
My Recipes
Grocery Lists


Affamato
http://affamato.com
Affamato
Login
Sign Up
Manage Your Pantry
Discover New Recipes
Save Money
Affamato
Powered by Falcon Group


**Planning and Scheduling**:

Phase I - due Feb 26

- At least 5 user stories on issue boards. Est. 5 hours.
- Some data collection - Will use APIs listed above. Est. 15-20 hours.
    - Basic database running on GCP with some information stored
    - Team understands how to use database and query with Postman
    - Created basic Objects in Java to be stored in database
    - Start scraping (Find scrape tools on Github and learn how to use them)
    - A static, deployed site with at least 5 pages hosted on GCP. Est. 10 hours.
    - Barebones UI built from DHTML and reactjs
    - Basic static pages for: Login, search results page, grocery list page, my recipes, and my inventory
    - Team needs to learn about basics of APIs, GCP, SQL and Database stuff, Postman, DHTML and reactjs, Mocha, Selenium and Github. Will probably require 6 hours per person
    - Stats derived on the About page, dynamically, from GitHub. Est. an hour.
    - Basic UI tests written with Selenium. Est. 5 hours.
    - URL from a hostname provider (Namecheap). Will take a couple minutes.
    - Report. Will take 5-10 hours, updating as we go.

Phase II - due March 12

- At least 5 additional user stories on issue boards. Est. 5 hours.
- Collection of a lot of data from your sources and storing in Database. Est. 20 hours.
    - Automated scraping to retrieve food items and associated info
    - Start scraping recipes
    - Users have an account and login stored in the database
    - All objects have functional instances: user, food items, pantry, recipes, etc.

- Refinement of the API. Est. 5-10 hours.
- Creating a dynamic website with many pages hosted on GCP. Est. 20 hours.
    - Login page is fully functional
    - Pantry page with food information is fully functional
    - Basic working search algorithm
    - Results on search page (recipes or food items) are fully functional
    - GUI framework is dynamic with a sleek and modern look

- Most unit tests written for the API with Postman. Est. 5 hours.
- Most unit tests for JS written with Mocha. Est. 5 hours.
- Most GUI tests written with Selenium. Est. 5 hours.
- Report refinement. Est. 5-10 hours, updating as we go.

Phase III - due April 9

- At least 5 more user stories –put them on your issue boards. Est. 5 hours.
- Finish data collection- all recipes, food, and relevant info is scraped and stored in the database. Est. 10 hours.


- Refine your dynamic site with many pages hosted on GCP. Est. 20 hours.
  - All pages are in their final form in both appearance and functionality
  - Search algorithm works for complex scenarios
- Refine ALL tests. Est. 10 hours.
- Add to and refine the technical report. Est. 10-20 hours, updating as we go.

Phase IV - due April 30

- Development is IDEALLY completed in phase III.
- Refactor, apply design patterns. Est. 5-10 hours.
- Finish the final project report. Est. 20 hours.
- Catch up with something if stuff gets behind. Time TBD.
- Expand on calorie/exercise tracking in accordance with our long term vision for a lifestyle app if we have lots of extra time. Time TBD.
- Create a presentation and put it on GitHub as a pdf (not required until you present). Est. 20 hours.

## Tools, software, frameworks:

- **Postman**: https://www.getpostman.com/ (For interacting with HTTP APIs to the DB)
- **SQL** (PostgreSQL or MySQL) vs NoSQL DB (MongoDB): https://medium.com/xplenty-blog/the-sql-vs-nosql-difference-mysql-vs-mongodb-32c9980e67b2 (Database stuff); https://www.sqlalchemy.org/ (SQL language instructions)
- **Github**: https://www.github.com (Collaboration and version control)
- **DHTML**: https://getbootstrap.com/ (All things website and UI)
- **Reactjs**: https://reactjs.org/ (More UI stuff)
- **Google Cloud Platform**: https://cloud.google.com/ (Cloud/DB hosting)
- **Java and JUnit**: https://www.oracle.com/java/
- **API Tools**: http://flask.pocoo.org/
- **RESTful API info**: https://en.wikipedia.org/wiki/Representational_state_transfer
- **Domain Name**: https://www.namecheap.com/
- **Mocha**: https://mochajs.org/ (Javascript testing)
- **Selenium**: https://www.seleniumhq.org/ (Static site and GUI testing)
- **Communication**: Slack, GroupMe, planitpoker.com

## Feasibility:

- Barcode scanning has become standard in these kinds of applications, but since this is a web application, we are not including this.
- Data collection/Cost excessive: Using api's that we can access as much as we need. Example: the foodacular api had a limit of 50 (free) requests per day, and we would likely need more than that to flesh out a 'recipe book'. We noticed that UT uses this api so we hope we can get access to more requests/day.
- Additionally the third person APIs above may not be reliable.
- Lack of familiarity with APIs, DHTML, DBs, Cloud providers, testing frameworks, building a website, any other tools we need, etc. We have a lot of new tools that most or all of the group members don't know how to use. Lack of familiarity means we could create more unnecessary bugs, write less efficient code, not properly test our components, etc.


- Time limit/too many features/can't deliver on performance - We have lots of functionality in our project and it may be hard to implement ALL of it over the semester. We are prioritizing the functionality of the pantry, recipes, and associated nutrition and expiration information. The health app tracking we mention in the vision is the secondary goal.
- The search and recommendation algorithms are going to be the trickiest part of what the application can actually do. Writing a brute force algorithm isn't too bad, but would be very slow, and no one likes slow.