Lettuce Save - The Busy Student's Grocery Planner
Member Details - Group 18

Member 1: Piper Winder
- Role: User Experience
- GitHub Username: piper-winder

Member 2: Hilarion Christian Gunawan
- Role: Deadline coordinator
- GitHub Username: HilarionTech77

Member 3: Paolo Mota Marques
- Role: Quality Assurance Manager
- GitHub Username: motamarp

Member 4: Ahman Raines
- Role: Backend/Database
- GitHub Username: araines05

Link to GitHub Repository - https://github.com/motamarp/cs362-grocery-list-project

Communication
Microsoft Teams will be used for the majority of the group's communication, with emphasis on project delegation, management, and development. A group chat will be created as well for immediate logistical communications, so as not to spam documentation on Teams. Members are expected to respond within 24 hours of receiving a message, with reasonable exceptions. For program development, proper block commenting will be implemented along with consistent documentation whenever a commit is pushed. Lastly, we all agree to be respectful and honest with each other, holding each other accountable and asking for help when needed.

Handling Missed Deadlines
Ideally, having a team member with a specific role as deadline coordinator will help us make all of our deadlines. In the case that something might not be finished in time, all members are expected to reach out to other team members for assistance and a plan will be made.

Project Description

Abstract

Our project is a grocery-list app that not only helps the user in making meal plans based on their preferences and constraints, it also helps to be more convenient throughout their shopping journey. Instead of being frustrated when you cannot find the specific item you want in a store, our app has the feature that allows you to search which store has the specific brand of item you want to buy, price comparisons, and if it is in stock.

Goal

This system will address the problems that users face when creating and planning their grocery lists. This includes understanding what needs restocking, finding the best option to purchase, and meeting other miscellaneous requirements (such as dietary restrictions). Essentially, users of this app will be granted a more efficient and sustainable way to track groceries in a personalized fashion. This would serve to solve the issues of forgetting to purchase certain necessities, while also easing navigation of dietary restrictions and providing alternatives for items.

Current practice

Grocery list apps today generally serve the purpose of organization. Please review the list below for applications and their links. They allow you to add or remove, and search up items to develop a purchase plan. Presently, there are several existing grocery list applications that have their own differentiating features. Some of these include sharing lists with family members and friends, notifying the user exactly where aisle items are located, and providing cooking instructions for food recipes. Despite the numerous features of these applications, most do not provide a personalized use and instead serve the user base in general. As students ourselves, we have been a part of many discussions with peers on the challenges of finding cost-efficient and alternative items for personal necessities. According to a study done in 2022 about Prevalence of Food Insecurity at OSU Corvallis, there is an estimated food insecurity rate of 26.6% of student participants (Oregon State University Food Security Study). With this knowledge, creating an Oregon State focused grocery list making application aims to assist such students.

- Instacart
- Jow
- eMeals

Novelty

While many other grocery list apps are available, our system is specifically catered towards busy students who wish to eat healthy while both staying cost efficient and saving on time. Thus, we will provide price comparisons for each item in their list. Many grocery lists apps have options to create separate lists for each grocery store, such as Instacart, however, users are still required to parse out the lists themselves manually. We hope to have an interface that provides a comprehensive list that automatically sorts which stores to buy what from. As students ourselves, we know exactly what our targeted audience wants so we will make sure to incorporate our individual preferences throughout the design process.

Effects

This grocery list application will specifically be super beneficial for college students who wish to eat well balanced meals that are cost efficient but don't have the time or resources to schedule or plan them all on their own. As a student, I end up spending hours planning my shopping lists, determining what recipes are worthwhile, and finding the ingredients at a reasonable price. Our aim is to make grocery shopping and meal planning less of a hassle so that users can focus on things that actually matter to them like school or work.

Use Cases (Function Requirements)

Profile Modification

1. Actors: New user that wishes to meal plan with specific specifications
2. Triggers: User is prompted to create a profile
3. Preconditions: User wishes to use application, application is open, no current profile has been created
4. Post Conditions: User confirms profile specifications once all needed information is provided and preferences were selected, immediately brought to main dashboard
5. Success Scenario (Ideal)
    a. User opens application and is prompted to create a profile (clicks button)
    b. User inputs all needed information (age, sex, height, etc)
    c. User reviews all optional toggle specifications (build, budget, dietary restrictions, activity level, etc) and curates it to themselves
    d. User clicks the save profile button and has access to main dashboard
6. Variation of success scenarios:
    a. Default creation
        i. User opens application and is prompted to create a profile (clicks button)
        ii. User inputs all needed information (age, sex, height, etc)
        iii. User keeps all other specifications on their default setting
        iv. User clicks the save profile button and has access to main dashboard
    b. Change preferences
        i. User opens profile settings page
        ii. User reviews specifications and changes certain options to better fit their goals
        iii. User clicks the save profile button
7. Exceptions: Failure conditions
    a. Closing Window
        i. User opens application and is prompted to create a profile (clicks button)
        ii. User gets overwhelmed with requirements and leaves application

iii. All possible inputs are erased, no profile has been created

b. Missing Information
   i. User opens application and is prompted to create a profile (clicks button)
   ii. User inputs almost all specifications but age
   iii. User keeps all other specifications on their default setting
   iv. User clicks the save profile button but an error pops up due to missing information
   v. User is unable to create profile until required fields are full

Meal & Grocery Planning

1. Actors: User that needs to figure out what they will be eating for the upcoming week
2. Triggers: User clicks the "create meal plan" button
3. Preconditions: User profile exists, known recipes database exists
4. Postconditions: Meals have been planned for the week, ingredient list has been compiled and organized by store, estimate of total cost is displayed
5. Success Scenario:
   a. User decided to create meal plan (clicks create button)
   b. User reviews automated meals recommendations and makes swaps or modifications
   c. User reviews automated grocery list and makes swaps or modifications
   d. User saves meal plan
6. Extensions/variations
   a. Edit meal plan
      i. User decides to edit meal plan (clicks edit button)
      ii. User makes changes to either meals planned or ingredients list
      iii. User saves meal plan
7. Exceptions/Failures
   a. Closes app
      i. User decides to edit meal plan (clicks edit button)
      ii. User makes changes to either meals planned or ingredients list
      iii. User leaves app, no changes are saved

Logging

1. Actors: User wishes to interact with meal plan and grocery list (adding notes, if plan has been followed, recipe review)
2. Triggers: User clicks the "add notes" button
3. Preconditions: The user has already chosen a meal plan and grocery list
4. Postconditions: An actionable sticky note icon will appear next to the user's meal plan and or grocery list
5. Success Scenario:
   a. The user wishes to create a note (clicks a note)
   b. User is given a prompt on where this note is to be attached to (meal plan or grocery list)
   c. User is brought to "notes page" where they can type

d. User saves notes

6. Extensions/variations:
    a. User edits the notes:
        i. User presses the sticky note icon
        ii. User presses the edit button and change what they wrote
        iii. User saves edited notes
7. Exceptions/Failures
    a. Unsaved notes:
        i. User presses the sticky note icon
        ii. User save edited notes
        iii. The application doesn't register so it still presents the "old" notes instead of the new edited ones

Review History

1. Actors: User that wants to see the changes they have made for their meal plan / list
2. Triggers: User hits the "Review History" button from homepage
3. Preconditions: User made (and saved) a meal plan / grocery list before
4. Postconditions: The previously saved information is displayed to the user
5. Success Scenario
    a. User clicks the "Review History" button
    b. The system displays the information in the saved list(s)
    c. User selects one of the meal plans / lists to inspect for more detail
6. Extensions/variations
    a. The user restores a meal plan / list that is no longer being stored
    b. The system sets a selected list or meal plan as the current one
7. Exceptions/Failures
    a. No saved information (meal plan or grocery list exists)
    b. The system would display a message stating: "No history available"

Returning User

1. Actors: User returning to the application
2. Triggers: Application opens
3. Preconditions: User profile exists before opening application
4. Postconditions: User profile and most recent plan are loaded
5. Success Scenario
    a. User opens application
    b. Systems loads profile data
    c. System opens homepage with saved data
6. Extensions/Variations
    a. Login Information change
7. Exceptions/Failures
    a. User profile is not loaded correctly
    b. No saved account information was loaded
    c. User is unable to login

Non-functional Requirement

- Reliability: Application is able to compile and run 99% of the time with all proper data saved. Under normal circumstances, all the core components like the recipe / list upload and history review should be working without issue or delay of some kind.
- Usability: Users must be able to intuitively navigate the application, it will take first time users less than 5 minutes to create a generalized meal plan (no specifications). Users shouldn't need to take more than a couple seconds to understand what each button leads to in the application (e.g. review history). Users should be able to complete an account and understand the main functions of the app within 60 seconds of opening it. Source Regarding User Onboarding
- Performance: New pages must load within 3 seconds of opening or swapping. Additionally, any background work such as restocking suggestions shouldn't slow the application performance.
- Portability: Application can be accessible on multiple devices (phone, desktop, etc) with conventional methods. It should maintain the same general structure on the application no matter how it is accessed.

External Requirements

- Compliance and Data Integrity: Possible user input errors are expected and user is provided alerts in such cases
- Barrier to entry: The product is easily accessible for users to download, install, and run from the App Store, Google Play, and Windows.
- Documentation: System is well documented to allow new developers to maintain and modify it

Software Architecture

Major components

- Creating Profile / Returning User:
  - **Purpose/responsibility:** Provides user access to application with data that is specific to them
  - **Input specs:** Height, weight, age, gender, lifestyle, budget
  - **Output:** Aggregated biometrics stored, profile created
  - **Logic:** Compilation of biometrics to later generate recommendations (see recipe/list upload)
  - **Dependencies:** Review History, Uploads
- Recipe / List Upload & Display:
  - **Purpose/Responsibility:** Allows the user to upload grocery / recipe ingredient lists which are displayed on the home-page.
  - **Input specs:** Aggregate biometrics from profile, user adds item(s) to list based on suggestions / custom choices from search feature

        - **Output:** Information displayed on user's current grocery list, and meal planner
        - **Logic:** Take profile information and generate meal plan recommendations along with related grocery list
        - **Dependencies:** Grocery, Review History, Profile
    - Grocery / Ingredient Search:
        - **Purpose/Responsibility:** Allows users to look through possible ingredient alternatives and add them to their lists
        - **Input specs:** Ingredient name
        - **Output:** Compiled list of ingredients from created meal plan with changes implemented
        - **Logic:** Compare strings and filter
        - **Dependencies:** Recipe, Grocery store database(s)
    - Review History:
        - **Purpose/Responsibility:** Allows the user to see detailed information of previous meal plans and diet for every meal
        - **Input specs:** Stored information of past and current meal plans, review history button
        - **Output:** Selection of meal plans – Day of week – Meal Titles, Macronutrient display
        - **Logic:** Take recipe and grocery information to make timeline and calculate macronutrients
        - **Dependencies:** Grocery, recipe

Interfaces

- ZeroMQ will be used for communication between microservices.
    - ZMQ provides peer-to-peer communication, eliminating the need for a separate message broking service
    - ZMQ is universal; it works with all languages
    - Provides sockets to setup in each microservice for easy communication
- Alternative: RabbitMQ
    - Pros:
        - Provides a user-friendly UI for easier management
        - Supports multiple protocols
        - Popular; Lots of resources online regarding troubleshooting/setup
    - Cons:
        - Considered to have a steep learning curve
        - Degraded performance with large amounts of data
- React will also be used for frontend and backend communication.
    - Allows us to take our components and individually render them through the browser to create specified pages

Data Storage

- Stored Data:
  - Information provided for profile creation:
    - This data is collected from the user's input in the profile creation.
  - Recipe:
    - This data is collected from existing recipes. However, user can also upload their recipes
  - Grocery/ ingredients:
    - This data contains the information on ingredients that are uuto-generated based on the recipe data from the active meal_plan. It could also be manually modified when users use want to use alternative ingredients
- Database to be used:
  - **Django** – Backend framework with python, **React** – frontend framework. We will also potentially use **Scrapy** to attain data from websites to compile grocery data.
- Alternatives:
  - **Flask** – Backend / Frontend features.
    - Pros:
      - Readily available documentation for setting up a minimal application
      - Easy to integrate individual adaptions
    - Cons:
      - Not "batteries-included," which means it doesn't come with all necessary installations
      - No data and file management structure readily available
  - **Reflex** – Another potential frontend framework.
    - Pros:
      - No need for additional Javascript or other languages, all 100% python building
      - Offers database management features such as routing/state management
    - Cons:
      - Debugging complexity increased
      - "Newer" framework with potential issues that may not be fixed by conventional means
        - Less documentation

Software Design

Packages, Classes, Units of abstraction

Profile/returning user component

This component will manage user identity and biometric data. It's responsible for onboarding of users, and collecting/storing the information needed to make personalized recommendations later on.

- Backend
    - User
        - Authentication/user identification
    - UserProfile
        - User's statistics (height, weight, age, lifestyle, budget)
    - ProfileView
        - For creation/modification of profile data
- Frontend
    - React forms
        - CreateProfile
        - EditProfile
        - ProfileSummary

MealPlan component:

- Backend
    - week
- Frontend
    - editMeal
    -

GroceryList component:

- Backend
- Frontend

ReviewHistory component:

This component manages in storing the history of users meal plan and grocery list as well as the user's macronutrients

- Backend
- Frontend

Coding Guidelines

The majority of our code will be written in Python. We will specifically use PEP 8 – Style Guide for Python Code as our coding style guideline due to the easy readability with a hanging indent for continued lines, block comments for methods and classes, and snake case for variables.

Since JavaScript will really only be used for frontend interactions, we will follow the generic JavaScript Style Guide outlined by W3Schools. Key points to note is the less than 80 characters per line with proper indentation and bracketing for complex compound statements. However, for consistency we will continue to use snake case for variable and function names.

As one of our team roles, the quality assurance manager will quickly review the code prior to submission to allow for continuity. With proper documentation and commenting, we will be able to pinpoint any discrepancies and ultimately correct the mistake.

Team Process Description

Technical Approach
Since we are using an API gateway microservice pattern, we will create the base application framework and from there add in services. Each service will have its own database to interact with, and will act as the intermediary for all other services wishing to access the data. The API gateway will act as the main entry point for users, taking care of all major functions. Messages will be sent between services to handle the minute tasks within each major function.

To implement the code, we will specifically be using Django as our Python backend framework. It is open-source with much documentation and has foundational built-in simplified interactions with React, which we plan to use as our frontend framework. We are using React since it provides us foundational frontend functions to allow browser based visuals and easy user interactions. We plan to use Scrapy, which allows us to scrape data from other websites to compile grocery data.

External Feedback
At the beginning of week three we presented our app idea to the whole class. After the presentation, we reviewed the discussion post where students were able to provide their insights and questions to us. We have already implemented a few changes into our project description that were recommended from peers.

Our goal is to have a mainly functioning app by week 7, we then will begin testing and debugging the program. Testing will involve having random users attempt to download, operate, and navigate the app, through that experience we will then ask for feedback. We'll start with black-box testing and then progress with white-box testing shortly after.

Risks

1) A challenge that we foresee while developing this project is properly accessing grocery store's individual online resources with our application to provide seamless integration and up to date information. We hope to mitigate these risks by providing clear user instructions, finding previously constructed databases to access, and researching grocery store's online capabilities.

2) Another potential risk is being unable to implement every feature as stated in the presentation with the time constraint that we have. Currently there are multiple features that need to be stand-alone, and most applications could probably settle just having one or two of the main features presented. Given this fact, we decided to combat this by focusing on the main aspects of our project – that being the list generation, meal plan suggestions, and profile creation. We've also thoroughly planned out who is responsible for each of those sections.

3) Another issue we face at the moment is that not all of our members are too experienced with Python. Although the language can be intuitive if you know C++, remembering the syntax and other concepts may pose a challenge. To prevent this from affecting project completion, we plan on communicating frequently through Microsoft Teams and in our weekly Friday meetings about any struggles the team may be facing. This way, team members can address any concerns or questions about debugging or just Python syntax in general.

4) For this project, our work will mostly go through flip engineering servers which tend to go down a lot. This is especially true for the week before final tests and projects are typically due, as all other software engineering classes will also be using those servers. To combat this, the team is currently working on finding an alternative environment (such as a local Docker). This isn't yet finalized in any form, but we are working out how the team will proceed in case of this issue.

5) Finally, the chosen frontend/backend frameworks may be confusing for some team members. Our main choice, Django, isn't one most of the group members have much experience with. This could result in major problems with our software architectures that aren't easy to identify, and it's crucial that we check in with each other to make sure everything is flowing together smoothly. Despite having alternatives, we firmly believe that using Django would be best if possible. The team members will work together to research information and familiarize themselves with Django. If after a week or so, issues arise, then there will be a shift to one of our alternatives. We will know if there are major issues depending on if we're able to make it work successfully with **Scrapy**, which is how we'll attain some of our program's information.

Major Features:

We plan to implement healthy meal plan recommendations, with built in logic to display recipes that go together throughout the whole week to cut down on food waste. The meal plans will be customizable with specific differentiations such as how often one wishes to cook, if they are ok with leftovers, and how many servings they need each meal. Users will also be able to upload their own recipes via website link, PDF, or manual. Within the actual list itself, we will provide price comparisons for each ingredient needed along with which stores have the items in stock.

Essentially:

- Healthy Meal Plan
    - Customizable to each user's individual needs
    - Prioritizes minimizing food waste and affordability
- Ingredient finding
- User Recipe Uploads
- Store price comparisons

Stretch Goals:

- We hope to implement suggestive restocking through learning weekly grocery patterns, which will involve possible AI use.
- Another possible feature we hope to implement is a comprehensive dietary review, where the user can track their macros and supplements they are receiving through their food.

Documentation Plan

Through weekly reports and our project schedule we will be able to keep track of development at a high level. With the use of GitHub we will create branches for each individual component, where members will keep it up to date. Thankfully, git provides us logs where the date, the creator, and an individual message is provided with every commit. That is more than enough documentation, when paired with our block comment formatting. We are already using teams so we plan on setting up a pipeline notification through Teams so all members have easy access to git status updates.

Test Plan & Bugs

First we plan on doing individual unit testing for each component as they are being developed. Unit testing will mainly focus on validation testing, where or not the system satisfies the functional requirements. For example, profile creation will be tested for best case scenarios, along with edge cases (improper characters, missing input, etc). Then once two components are able to pass their unit testing, we will continue onto integration testing. Again, mainly focusing on validation testing but in use cases where both components are used. Since we are using a microservices architecture, test cases would involve the passing of information from one service to another. We also hope to do error testing for when one microservice breaks, and how the next service is able to react.

Throughout the whole development process, whenever bugs occur, we will create a GitHub issue to keep track of them. This will also allow us to document test cases that are created and added to the test suite. Once most of the application has been developed, we will move onto system testing. We will use the test suite we've compiled of previous issues to do regression testing upon the whole system, along with testing typical use cases and edge cases.

To ensure **Scrapy** works well with our frontend/backend choices, first the team will check that the chosen information is being pushed into our software. Once we have confirmation of this, we proceed with attempting to format the information into our separate features. Upon doing so, we check to see if the information remains while unit-testing specific features such as searching for groceries. This is the most crucial part, as several of our features require storing the chosen grocery lists. After validation from the individual tests, we'll then begin using integration testing to ensure information is pushed from one page to another.

Piper Winder – User Experience
Justification: This role ensures there is seamless integration with backend features and user interactions that are intuitive and pleasing to the eye. I am suited for this role due to my keen attention to detail, interest in continuity of product, and my lack of patience which translates well into thinking like a user.
Personal Schedule:

expected to be done by end of week	Milestone Goals
Week 3	- All use case scenarios and diagrams have been created - User flow diagram is outlined
Week 4	- Whole file system is setup with Django, React, and Github
Week 5	- Create individual feature test cases - History page setup - Grocery list can be automatically generated for default profile
Week 6	- Grocery list items can be modified
Week 7	- Grocery list generates other plans from specifications - REST SHOULD BE TESTING/DEBUGGING
Week 8	- External feedback

Week 9	- Project report

Paolo Mota Marques – Quality Assurance Manager
Justification: This role is set to make sure everything ties in together nicely after work is finished. In other words, a final review of the assignments/project before submission. As the person who created the Github repository and turned in the projects so far, this role feels most fitting for Paolo. He has also taken similar roles in previous projects.
Personal Schedule:

Week	What to Accomplish:	What will be worked on:
3	~~Weekly Report 1~~ ~~Repository setup~~	Automated Meal Planning
4	Weekly Report 2	Automated Meal Planning
5	Weekly Report 3 Automated Meal Planning	Logging system Make sure every feature works together in practice
6	Weekly Report 4 Everything is, in the most basic sense, working as expected together.	Integration Testing Fix integration bugs Logging System
7	Weekly Report 5 All integration bugs fixed	Integration Testing Fix integration bugs
8	Weekly Report 6	TBD (More than likely bug-fixes)
9	Weekly Report 7 All features tested and working	At this point in time, everything should be functional and final.

Ahman Raines – Backend/Database
Justification: This role is to ensure that all backend components of the project interact properly with each other, as well as with the front end of the program (Collaboration with User Experience). This encompasses management and testing of all server-side logic/core functionality of the project. Backend experience in past projects makes this a suitable position for Ahman.
Personal Schedule: Monday, Wednesday, Friday

Week	Goal	Accomplished
Week 4	- Website scraping, grocery store database compilation	
Week 5	- - Logging page created, user is able to add notes and reviews	- Recipe database interactions - Profile creation for recommendation reference
Week 6	- Review page created	
Week 7	- Grocery list modifications - Meal planning modifications - Review page with logging implemented	- Automated meal planing - Automated grocery list - Logging feature
Week 8	- Test implemented features - Test feature integration	Ideally everything has been implemented now and it is mainly testing and debugging
Week 9	- Project report	

Hilarion Christian Gunawan- Deadline coordinator
Justification: This role helps to make sure the team knows when assignments are due to have enough time to meet up and work on it together. Since us students are always busy with both classes and other assignments,the deadline coordinator will help remind the team of upcoming

deadlines so that each group member can also manage their own schedule to have time to work on the project which Hilarion is suitable for the job.

Personal Schedule: Wednesday- 1-3:30 pm
Free on Fridays and weekends

Week	What I want to get Done:	What I Start On:
3	~~Weekly Report 1~~ ~~Repo setup~~	
4	Weekly Report 2	Profile creation and modification
5	Weekly Report 3	Profile specification
6	Weekly Report 4	Unit testing for profile
7	Weekly Report 5	
8	Weekly Report 6	TBD (More than likely bug-fixes)
9	Weekly Report 7	At this point in time, everything should be functional and final.

Timeline

Week	Working On	Accomplished
Week 3	- User flow diagram - Project architecture (class diagrams, functions, etc) - Frontend design	- General project proposal - GitHub repository made - Requirements (Use Cases & Scenarios)
Week 4	- Code database interactions (recipes) - Code profile creation	- File structure created in github

	- Grocery store database collection	
Week 5	- Code automated meal planning - Create individual feature test cases - Logging feature - Review page setup	- Recipe database interactions - Profile creation for recommendation reference
Week 6	- Code grocery list - Create integrated test cases	
Week 7	- Test previously implemented features - Grocery list modifications - Meal planning modifications - Review page with logging implemented	- Automated meal planing - Automated grocery list - Logging feature
Week 8	- Test implemented features - Test feature integration	Ideally everything has been implemented now and it is mainly testing and debugging
Week 9	- Project report	