Introduction

The system will be able to let any user whether they are busy college students or health conscious individuals keep track of both their health and budget simultaneously. It contains visually appealing pages which allow users to navigate seamlessly through each section of their health and budget journey including: a goals page which the user can manually update depending on their fitness or financial goals for a specific week or month, a calories page which will allow the user to add what they eat to a diary which will monitor all of that food's macronutrients, a page to contain a budget which will be displayed on the main page of the website which keeps track of the user's food purchases, a page for allergy input as to warn the user whether a certain suggested food contains a potentially harmful food allergen, a weight tracking page to visually display a user's weight throughout their time in their fitness journey, and finally a page to create an account for the website.

Value Proposition: "Keep track of your finances and health without worrying about the stresses of daily life."

MVP: Budget calculator, weight graphic, account creation, meal tracker, health/fitness or budget goals, macronutrient calculator,

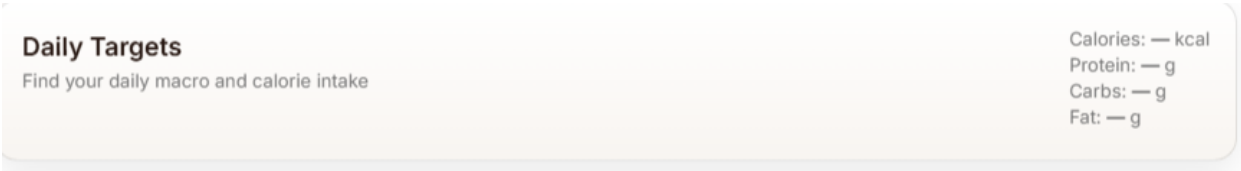
Repository Link: https://github.com/loganb7869/CS386-Project-Repo

Implemented Requirements

2.1 Logan Bankert

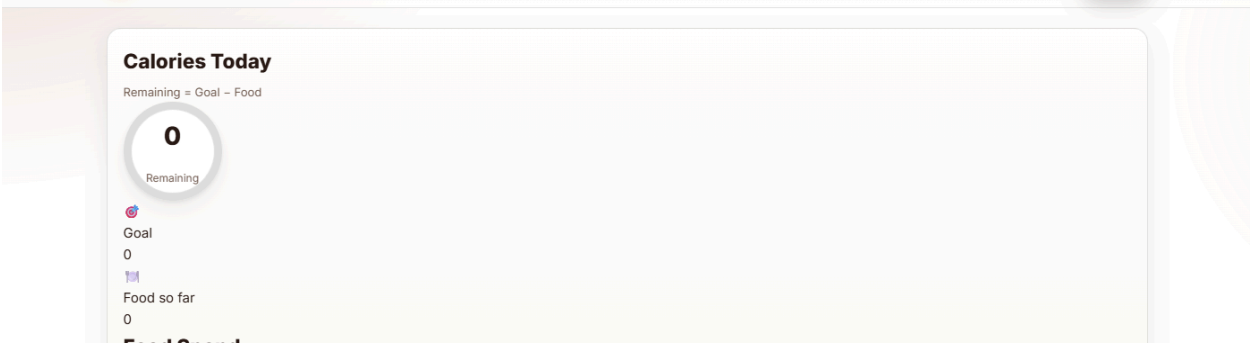
Requirement 1

1. Requirement(User story from Repo)**: As a student athlete, I want to keep track of my carb intake so that I can carb-load more effectively.**
2. Issue: https://github.com/loganb7869/CS386-Project-Repo/issues/3
3. Pull request: https://github.com/loganb7869/CS386-Project-Repo/pull/19
4. Implemented by: Logan Bankert
5. Approved by: Rita Bolanos
6. Automated Testing: https://github.com/loganb7869/CS386-Project-Repo/blob/ard584-unittests/Unittest_2.1.A
7. Visual Evidence:



Requirement 2

1. Requirement(User story from Repo): **As a busy student, I want to keep track of my calorie intake so that I eat enough to not be tired during class.**
2. Issue: https://github.com/loganb7869/CS386-Project-Repo/issues/4
3. Pull request: https://github.com/loga nb7869/CS386-Project-Repo/pull/20
4. Implemented by: Logan Bankert
5. Approved by: Rita Bolanos
6. Automated Testing: https://github.com/loganb7869/CS386-Project-Repo/blob/ard584-unittests/Unittest_2.1.B
7. Visual Evidence:

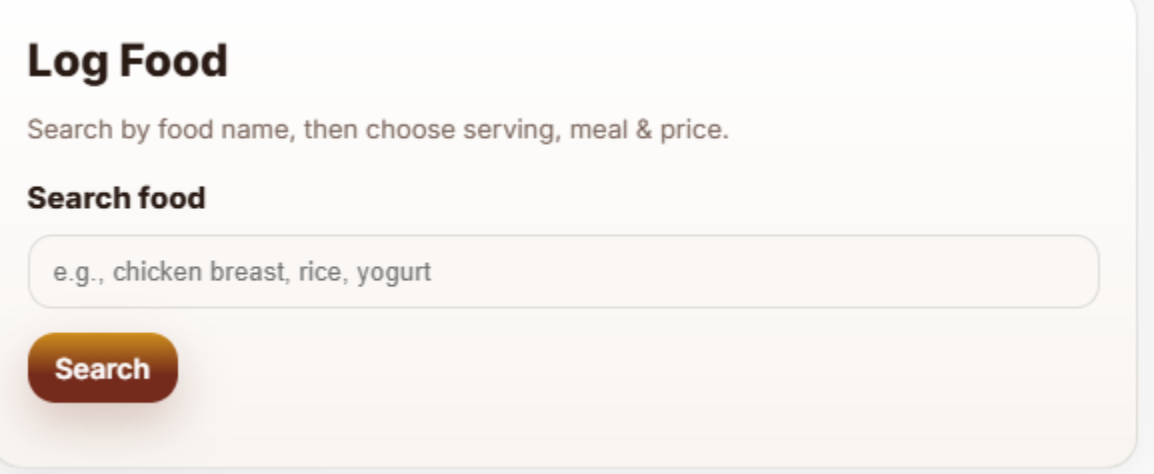


2.2 Rita Bolanos

Requirement 1

1. Requirement(User story from Repo): **As a student, I want to save frequently eaten foods so that I can easily access them.**
2. Issue: https://github.com/loganb7869/CS386-Project-Repo/issues/5
3. Pull request: https://github.com/loganb7869/CS386-Project-Repo/pull/21
4. Implemented by: Rita Bolanos
5. Approved by: Rita Bolanos
6. Automated Testing: https://github.com/loganb7869/CS386-Project-Repo/blob/ard584-unittests/Unittest_2.2.A

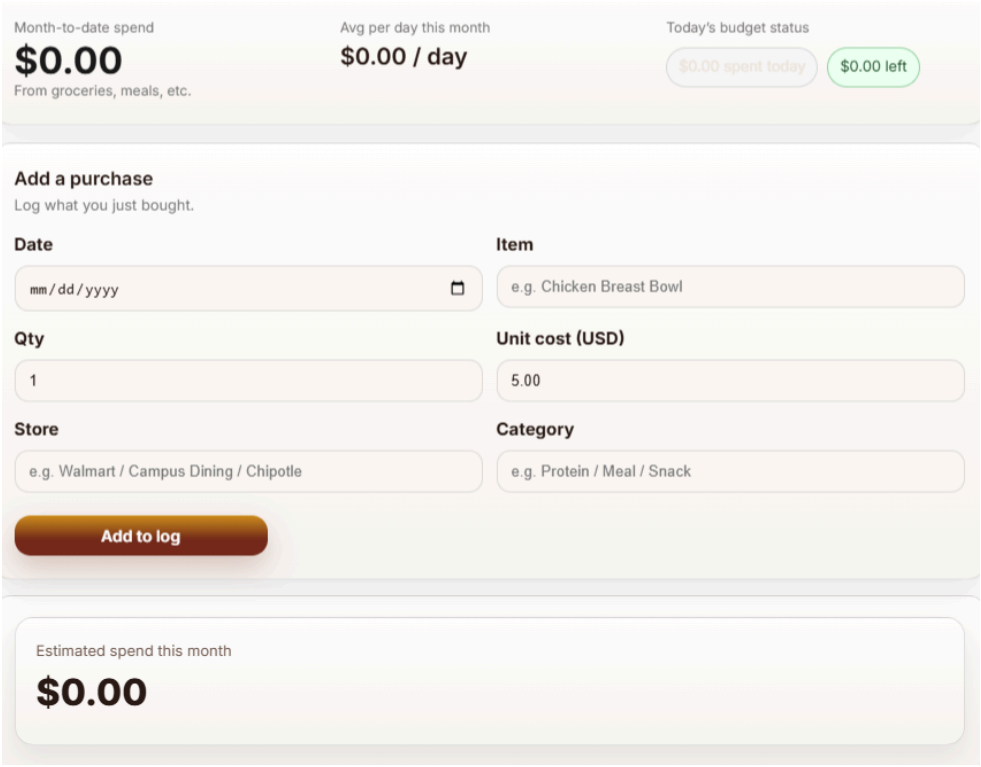
7. Visual Evidence:



Requirement 2

1. Requirement(User story from Repo): **As a budget conscious student, I want to see the cost per food item so that I can stay within my budget.**
2. Issue: https://github.com/loganb7869/CS386-Project-Repo/issues/6
3. Pull request: https://github.com/loganb7869/CS386-Project-Repo/pull/22
4. Implemented by: Rita Bolanos
5. Approved by: Rita Bolanos
6. Automated Testing: https://github.com/loganb7869/CS386-Project-Repo/blob/ard584-unittests/Unittest_2.2.B

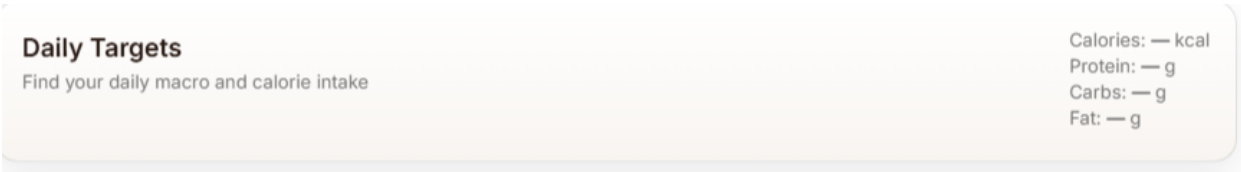
7. Visual Evidence:



2.3 Annaliese Dedmore

Requirement 1

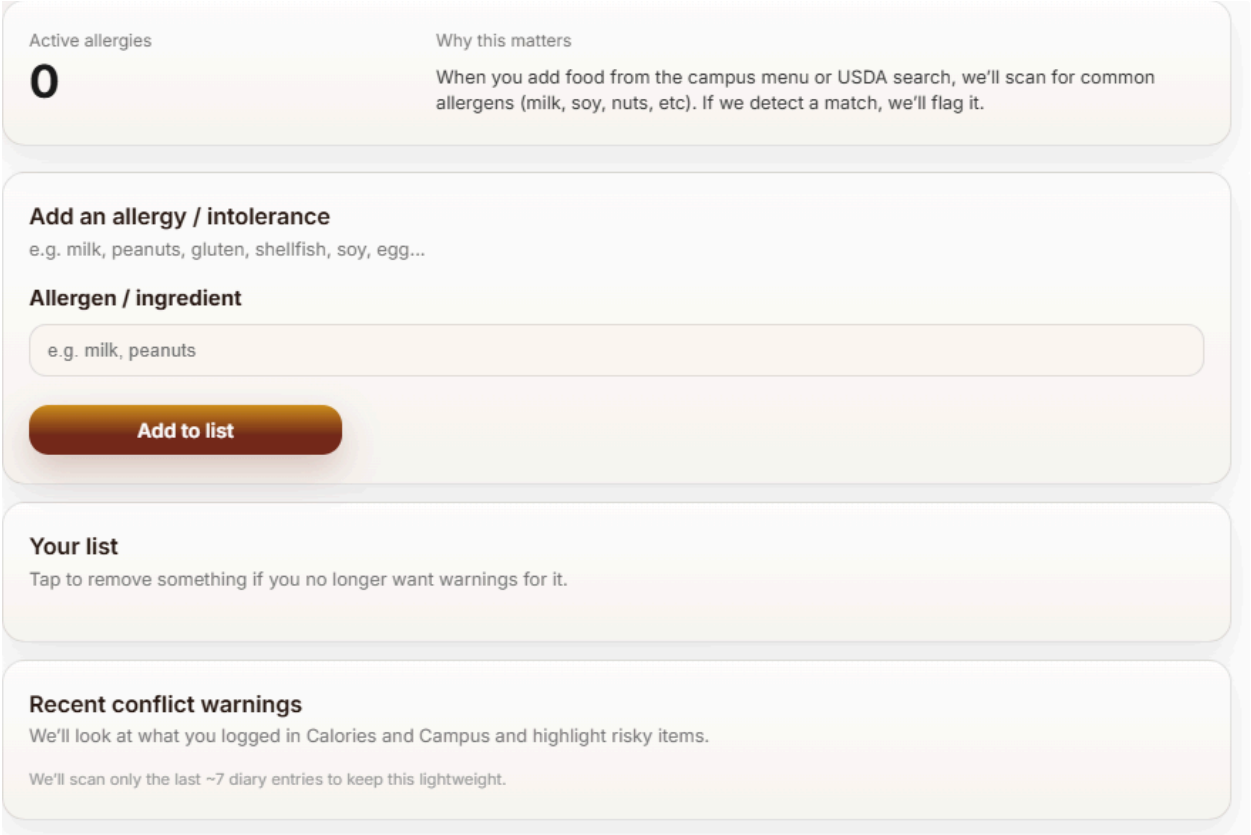
1. Requirement(User story from Repo): **As a lazy student, I want to be able to track my total targeted macros vs. logged macros.**
2. Issue: https://github.com/loganb7869/CS386-Project-Repo/issues/7
3. Pull request: https://github.com/loganb7869/CS386-Project-Repo/pull/23
4. Implemented by: Annaliese Dedmore
5. Approved by: Rita Bolanos
6. Automated Testing: https://github.com/loganb7869/CS386-Project-Repo/blob/ard584-unittests/Unittest_2.3.A
7. Visual Evidence:



Requirement 2

1. Requirement(User story from Repo): **As a health conscious student, I want to be able to track all my allergies to make sure I remember what to avoid.**

2. Issue: https://github.com/loganb7869/CS386-Project-Repo/issues/8
3. Pull request: https://github.com/loganb7869/CS386-Project-Repo/pull/24
4. Implemented by: Annaliese Dedmore
5. Approved by: Rita Bolanos
6. Automated Testing: https://github.com/loganb7869/CS386-Project-Repo/blob/ard584-unittests/Unittest_2.3.B
7. Visual Evidence:

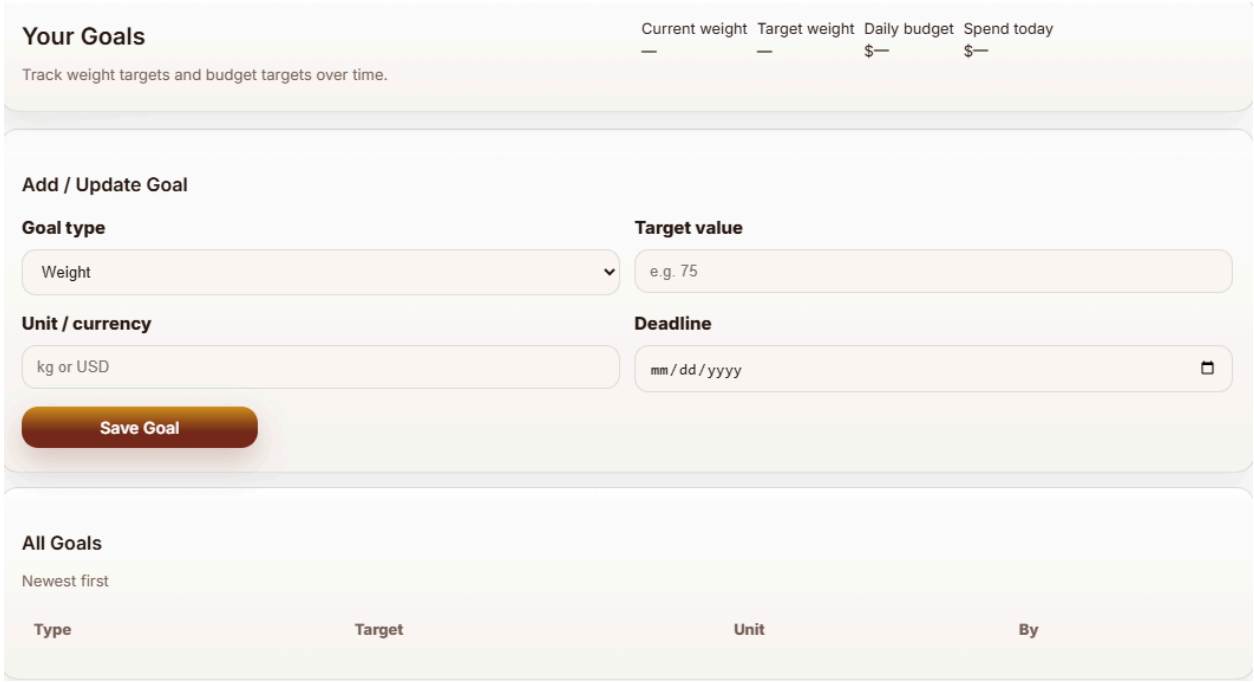


2.4 Isidro Marquez

Requirement 1

1. Requirement(User story from Repo): **As a student committed to improving my personal health, I want to be able to view all of my health details and goals all in one place.**
2. Issue: https://github.com/loganb7869/CS386-Project-Repo/issues/9
3. Pull request: https://github.com/loganb7869/CS386-Project-Repo/pull/25
4. Implemented by: Isidro Marquez
5. Approved by: Rita Bolanos
6. Automated Testing: N/A

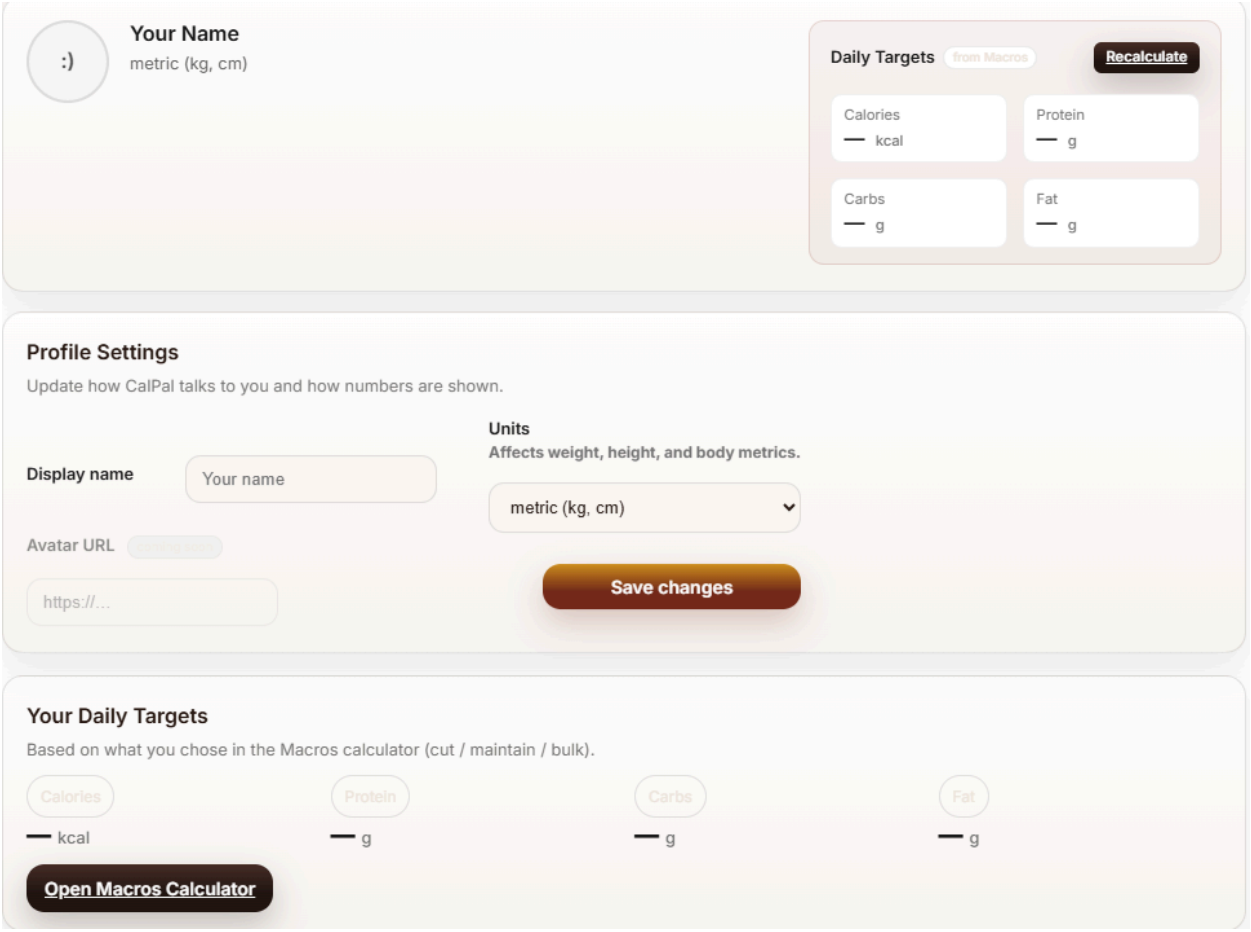
7. Visual Evidence:



Requirement 2

1. Requirement(User story from Repo): **As a newly-independent student, I want to be able to track all of my health changes as I adapt to college life.**
2. Issue: https://github.com/loganb7869/CS386-Project-Repo/issues/10
3. Pull request: https://github.com/loganb7869/CS386-Project-Repo/pull/26
4. Implemented by: Isidro Marquez
5. Approved by: Rita Bolanos
6. Automated Testing: https://github.com/loganb7869/CS386-Project-Repo/blob/ard584-unittests/Unittest_2.4.B

7. Visual Evidence:

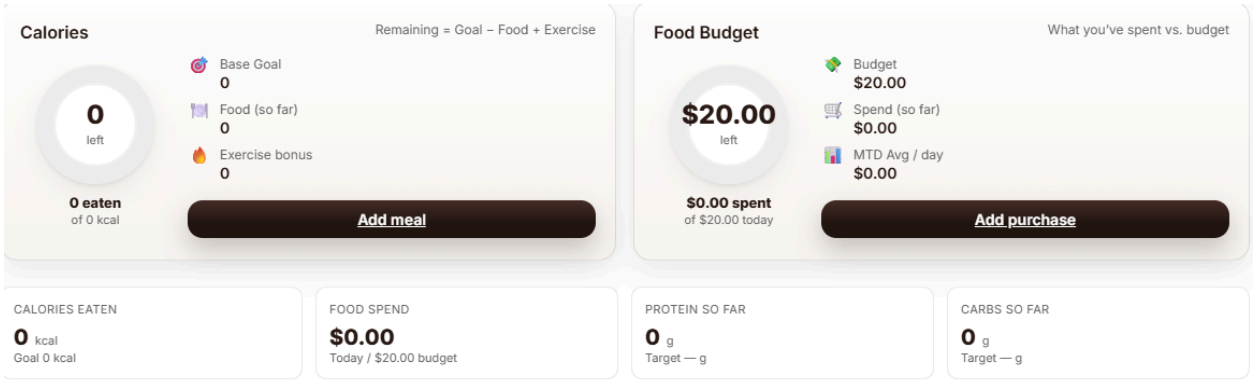


2.5 Nile Ham

Requirement 1

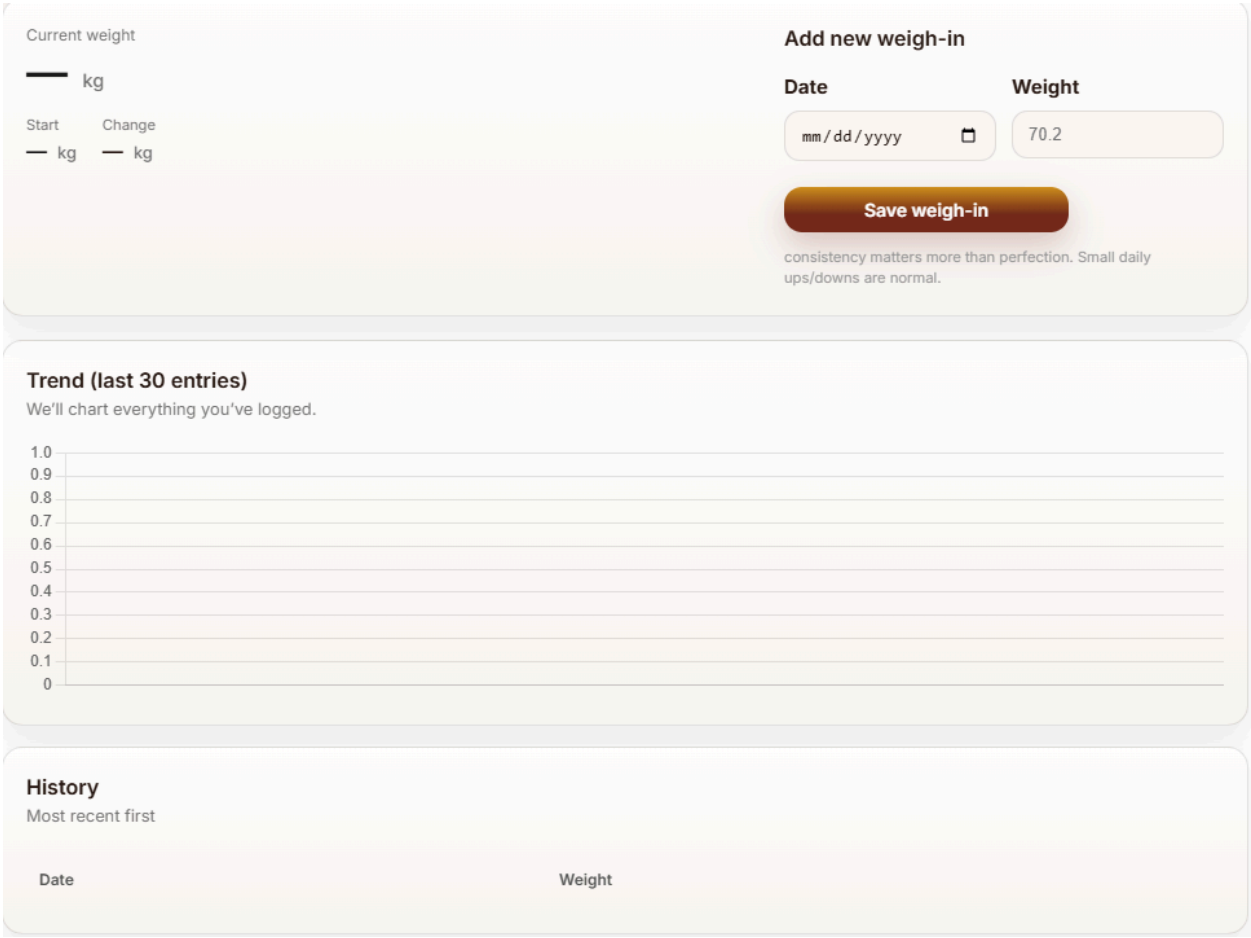
1. Requirement(User story from Repo): **As someone who enjoys eating healthy and saving money, I want to be able to track what I eat and spend at the same time.**
2. Issue: https://github.com/loganb7869/CS386-Project-Repo/issues/11
3. Pull request: https://github.com/loganb7869/CS386-Project-Repo/pull/27
4. Implemented by: Nile Ham
5. Approved by: Rita Bolanos
6. Automated Testing: N/A

7. Visual Evidence:



Requirement 2

1. Requirement(User story from Repo): **As someone who frequently does intense workouts, I want to be able to track what I eat throughout the year depending on what my goals are.**
2. Issue: https://github.com/loganb7869/CS386-Project-Repo/issues/12
3. Pull request: https://github.com/loganb7869/CS386-Project-Repo/pull/28
4. Implemented by: Nile Ham
5. Approved by: Rita Bolanos
6. Automated Testing: https://github.com/loganb7869/CS386-Project-Repo/blob/ard584-unittests/Unittest_2.5.B



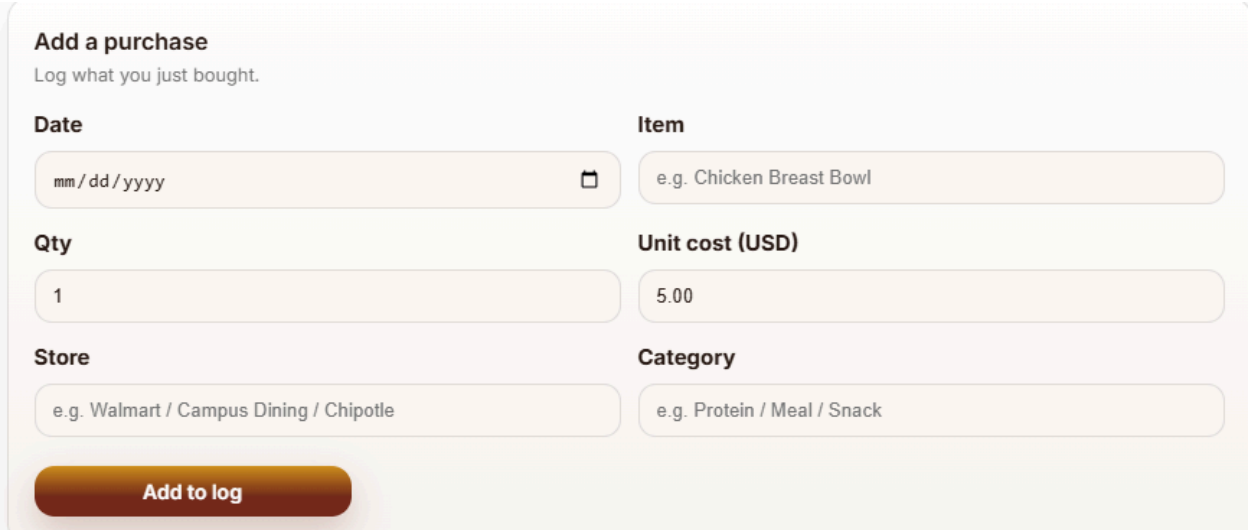
7. Visual Evidence:

2.6 Luke Flaker

Requirement 1

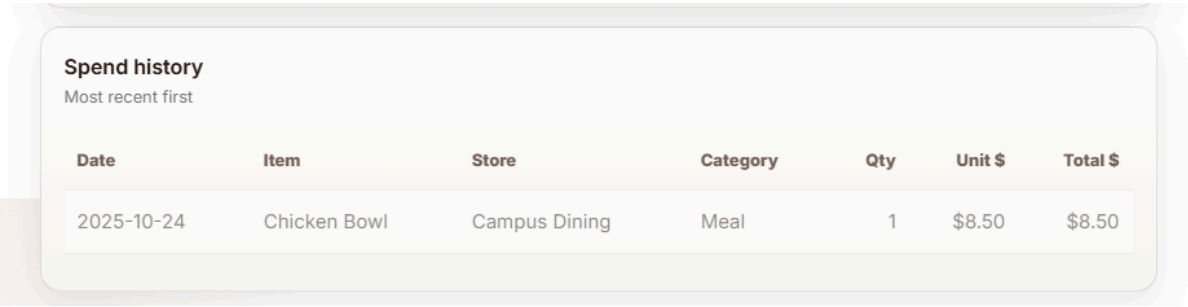
1. Requirement(User story from Repo): **As a freshman college student with a limited dining plan, I want to be able to juggle my spending on other meals alongside using my dining plan to ensure I have enough until the semester ends.**
2. Issue: https://github.com/loganb7869/CS386-Project-Repo/issues/13
3. Pull request: https://github.com/loganb7869/CS386-Project-Repo/pull/29
4. Implemented by: Luke Flaker
5. Approved by: Rita Bolanos
6. Automated Testing: https://github.com/loganb7869/CS386-Project-Repo/blob/ard584-unittests/Unittest_2.6.A

7. Visual Evidence:



Requirement 2

1. Requirement(User story from Repo): **As a busy college student, I want to be able to keep track of what I spend at the grocery so as to not go over my weekly budget.**
2. Issue: https://github.com/loganb7869/CS386-Project-Repo/issues/14
3. Pull request: https://github.com/loganb7869/CS386-Project-Repo/pull/30
4. Implemented by: Luke Flaker
5. Approved by: Rita Bolanos
6. Automated Testing: https://github.com/loganb7869/CS386-Project-Repo/blob/ard584-unittests/Unittest_2.6.B
7. Visual Evidence:



Automated Testing

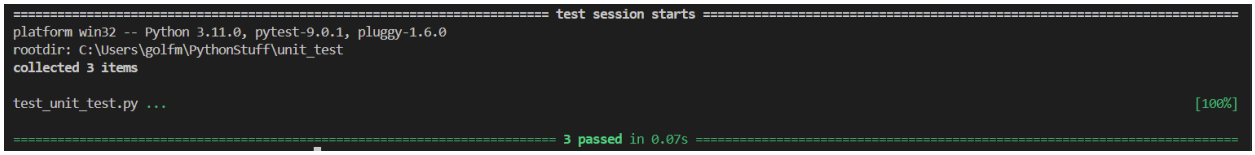
3.1 Unit tests

1. Testing Framework: PyTest
2. Test Location: https://github.com/loganb7869/CS386-Project-Repo/tree/main/tests/d6tests/unit
3. Test Example:

a. https://github.com/loganb7869/CS386-Project-Repo/blob/main/tests/d6tests/unit/test_unit_test.py
b. https://github.com/loganb7869/CS386-Project-Repo/blob/main/tests/d6tests/unit/unit_test.py
c. The test goes over the three different types of budget input from the user: no input, inputted budget, and unrecognized input such as letters instead of numbers. For each test, it stores the state of the user input and runs through each test to check for if the input is valid or not, then runs the asserts debug statement for PyTest to ensure that all the tests pass successfully.
d. Code snippet for test:

```python
class TestFinalPriceCalculator:
    def test_Default(self):
        state = {}
        result = getDailyBudgetGoal(state)
        assert result == 5.00

    def test_CurrentBudget(self):
        state = {"daily_budget": 35.00}
        result = getDailyBudgetGoal(state)
        assert result == 35.00

    def test_Invalid(self):
        state = {"daily_budget": "N/A"}
        result = getDailyBudgetGoal(state)
        assert result == 5.00
```

4. Test Results:



3.2 Integration

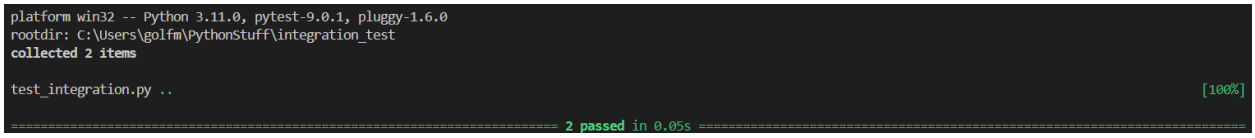
1. Testing Framework: PyTest
2. Test Location: https://github.com/loganb7869/CS386-Project-Repo/tree/main/tests/d6tests/integration
3. Test Example:
    a. https://github.com/loganb7869/CS386-Project-Repo/blob/main/tests/d6tests/integration/test_integration.py
    b. https://github.com/loganb7869/CS386-Project-Repo/blob/main/tests/d6tests/integration/unit_test.py
    c. Expanding off of the unit test, I added a function test which tests for the budget being updated when a user adds food to their diary. It includes a case where if the user hasn't inputted their budget, the default budget will be used to subtract the food price from the default budget. Finally, it also has a case where the user

has a predefined budget which takes the price of the meal, subtracts it from the budget, and returns the remaining budget after the diary is updated.

d. Code snippet for test:

```python
class TestIntegration:
    def test_budget_remaining_default(self):
        state = {}
        unit_cost = 5.00
        quantity = 1
        remaining_money = getRemainingBudget(state, unit_cost, quantity)

        assert remaining_money == 15.00

    def test_updated_budget(self):
        state = {"daily_budget": 100.00}
        unit_cost = 5.00
        quantity = 10
        remaining_money = getRemainingBudget(state, unit_cost, quantity)

        assert remaining_money == 50.00
```

4. Test Results:

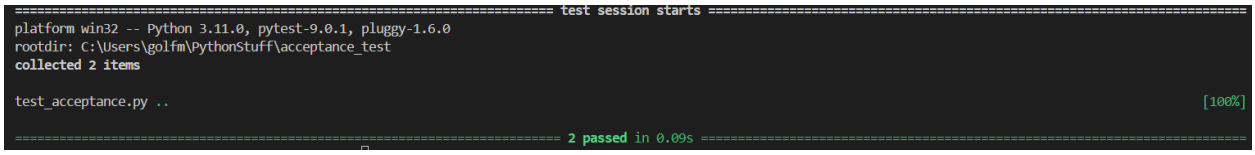


3.3 Acceptance

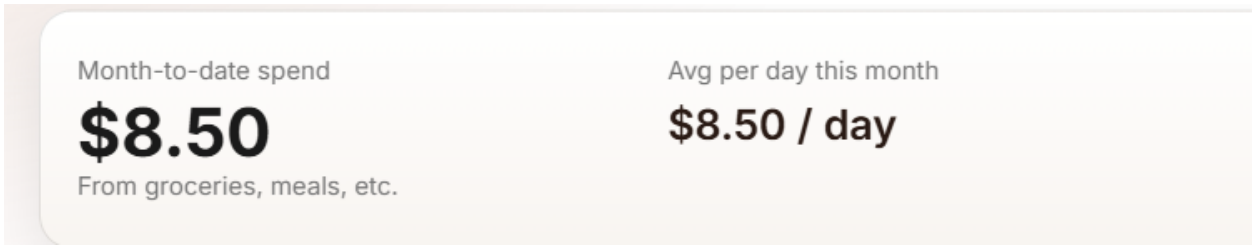
1. Testing tools: PyTest
2. Test Location: https://github.com/loganb7869/CS386-Project-Repo/tree/main/tests/d6tests/acceptance
3. Test Example:
    a. A user has a default budget of $20 for a certain day. They go to the store and purchase a chicken rice bowl for $8.50. The user adds this purchase to their diary which then updates the remaining budget for that day as $11.50.
    b. 1. User purchases a chicken rice bowl for $8.50
    2. User adds purchase to their website diary
    3. System recognizes the price inputted by the user
    4. System updates the default budget by subtracting the price of the meal from the default daily budget
    5. System displays the remaining budget to the user
    c. Feature completion confirmation: budget is updated to represent the correct remaining balance and the system will be able to receive more from the user.
    d. https://github.com/loganb7869/CS386-Project-Repo/blob/main/tests/d6tests/acceptance/test_acceptance.py
    e. Code snippet for test:

```python
class TestAcceptance:
    def test_default_budget_purchase(self):
        state = {}
        budget = getDailyBudgetGoal(state)

        assert budget == 20.00

        unit_cost = 8.50
        quantity = 1
        spent = addCost(unit_cost, quantity)

        assert spent == 8.50

        remaining = getRemainingBudget(state, unit_cost, quantity)

        assert remaining == 11.50

    def test_custom_budget_purhcase(self):
        state = {"daily_budget": 50.00}
        budget = getDailyBudgetGoal(state)

        assert budget == 50.00

        unit_cost = 12.99
        quantity = 2
        spent = addCost(unit_cost, quantity)

        assert spent == 25.98

        remaining = getRemainingBudget(state, unit_cost, quantity)

        assert remaining == 24.02
```

4. Test Results:



Comparison to actual functionality in website for default purchase:



System Demonstration

System demonstration video Link:
https://drive.google.com/file/d/1GO_v38C0vqwuC1jas5AfckQJfrtzexwf/view?usp=sharing

AI-Assisted Code Quality Review

5.1 AI Interaction

1. Tool Used: ChatGPT 4, Google Gemini
2. Prompt Engineering:

    a. Analyze the provided source code and generate a comprehensive report focused on maintainability. Your analysis should cover architecture (identify structural issues that may hinder future modification or extension), design (evaluate design choices with respect to readability, modularity, cohesion, and coupling), and code quality (detect maintainability concerns such as inconsistent naming, code duplication, overly complex methods, lack of separation of concerns, and unclear logic). For each issue, provide specific, concrete examples, including folder names, class names, function names, variable names, or actual code snippets. Focus entirely on the source code and its structure—do not recommend external tools, libraries, or frameworks. Analyze the entire codebase, ignoring only code within comments. Do not ask follow-up questions; produce a complete, detailed, and specific analysis based solely on the provided code. Be thorough, precise, and grounded in software engineering best practices.
3. Conversation Link:
    a. Logan Bankert link: https://chatgpt.com/share/692377a1-f4ec-8008-a9e0-0e761f2b105c
    b. Rita Bolanos link: https://chatgpt.com/share/692388d6-c690-8009-8dda-59173e6b3e0d
    c. Nile Ham link: Gemini conversation link not available
    d. Annaliese Dedmore link: Gemini conversation link not available
    e. Isidro Marquez link: https://chatgpt.com/share/6923f624-70d0-800d-9c7f-d6c98bc69034
    f. Luke Flaker: https://chatgpt.com/share/692388f3-c698-8012-a28d-ff5ed7e2060c

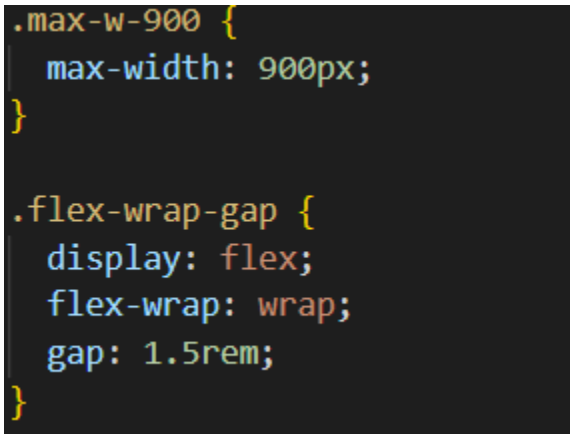
5.2 AI Interaction

Logan Bankert

In reference to the ‘code.html’ file:

1. AI Feedback: Architecture (structural issues that hinder extension) - HTML contains lots of inline styling and inline event wiring, mixing structure, presentation and behavior. This tightly couples markup with styling and JS behavior. The file links to css/main.css and js/app.js, but many presentation rules and event handlers are embedded directly in the HTML. Examples: 1. Inline styles on containers and elements (repeated throughout) 2. Inline event handlers / behaviour wiring. **Why it hurts:** Makes it hard to change styling without touching HTML, or to replace the JS implementation without combing through markup. Also leads to duplicated style declarations and inconsistent UI when multiple developers edit different files.
2. Assigned Reviewer: Logan Bankert
3. Severity Assessment: Low
4. Decision: Approved
5. Explanation: After working with the suggestion generated by the AI, corrections were made to both the ‘cost.html’ and ‘main.css’ file which helped to reduce the excessive use

of the ‘style’ element in the HTML file itself. An example of these changes are displayed in this screenshot, with the HTML classes being updated in light of these changes.



```
.max-w-900 {
  max-width: 900px;
}

.flex-wrap-gap {
  display: flex;
  flex-wrap: wrap;
  gap: 1.5rem;
}
```

1. AI Feedback: No componentization or templates - The spending table contains a single static sample row in the tbody, but there is no <template> element or clear structure for how rows are created/inserted. **Why it hurts:** Lack of template/component pattern increases duplication (every new row creation needs manual DOM code), reduces testability, and raises risk of markup inconsistencies when rendering dynamic data.
2. Assigned Reviewer: Logan Bankert
3. Severity Assessment: Low
4. Decision: Rejected
5. Explanation: While implementing templates inside of our code can dramatically improve readability, this implementation may potentially create more issues with how JavaScript handles the DOM manipulation for row and column creation.

1. AI Feedback: Inconsistent naming conventions (readability and predictability) - Different naming styles are mixed: 1. CamelCase for IDs: mtdSpendNumber, avgPerDay, todaySpendPill, todayRemainPill, costOut 2. Kebab-case for classes: topbar, nav-tabs, brand-left, right-area. 3. Abbreviated input name cat instead of category: <input name="cat" type="text" ...> while the table has the "Category" header. **Why it hurts:** Mixing conventions increases cognitive load and mistakes when writing/selecting DOM hooks in JS. Shortened names like cat are ambiguous and reduce readability.
2. Assigned Reviewer: Logan Bankert
3. Severity Assessment: Low
4. Decision: Rejected
5. Explanation: While maintaining consistency in naming conventions is an important part of our jobs as developers, it ultimately is unimportant in reference to the greater majority of the program itself. As long as the naming conventions remain partially consistent in the code and everything is referenced correctly, it comes down to the developer to ensure that these defined conventions are properly implemented.

1. AI Feedback: Labels & form semantics: partial usage - HTML uses <label> wrapping inputs, which is good, but the inputs lack id attributes and the labels do not use for

attributes. **Why it hurts:** Screen readers and automated testing often rely on explicit for/id pairs. Wrapping labels works but combining both consistently is best practice for clarity and future DOM re-arrangements.

2. Assigned Reviewer: Logan Bankert
3. Severity Assessment: Low
4. Decision: Rejected
5. Explanation: While this suggestion generated may be best practice for labels and form semantic syntax, the AI mentions that the way we had implemented the label wrapping works fine, but it may have some potential problems when it comes to readability and testability. Due to this, the team decided that correcting this syntax would be best left the way it is, as to prevent ambiguous definitions and id pairs.

Rita Bolanos

In reference to the 'macros.html' file:

1. AI Feedback: Excessive inline styles: You're using inline styles everywhere (e.g., style="display:flex;..."). Why it's a problem: Harder to maintain, Impossible to reuse, Bloats HTML, Can't be minified or cached like CSS files. Fix: Move all those rules to your stylesheet (main.css).
2. Assigned Reviewer: Rita Bolanos
3. Severity Assessment: Medium
4. Decision: Approved
5. Explanation: While this problem is not of high priority, since a priority for our team is functionality over all else, organizing and making the code easier to read is important to make modifying and debugging the code easier.

1. AI Feedback: Hardcoded element IDs everywhere. Good for simple pages, but you're scaling to multiple tools (macros, allergies, weight, cost). Better to namespace them (e.g., macro-, weight-) or use data-* attributes if you grow the functionality.
2. Assigned Reviewer: Rita Bolanos
3. Severity Assessment: Low
4. Decision: Rejected
5. Explanation: Due to this critique mostly affecting potential expansion of the program which is not planned at this time since the current features and functionality of the website is mostly up to our team's standards.

1. AI Feedback: The layout will work but some flex rows wrap awkwardly. Because you use flex-wrap: wrap without breakpoints, rows can collapse in unexpected ways on narrow screens. Use CSS media queries instead of inline flex rules.
2. Assigned Reviewer: Rita Bolanos
3. Severity Assessment: Medium
4. Decision: Approved
5. Explanation: While adding mobile functionality is less important to our team then making sure the website works on desktop, allowing for mobile accessibility is still important to

our team, and as such we will work on debugging how the website displays on mobile devices.

1. AI Feedback: LocalStorage for health data: You're storing calorie/macro targets in localStorage. That's fine for a lightweight personal app, but: It persists forever. Any script on the origin can read it. If you expand into a login-based app, consider server-side storage.
2. Assigned Reviewer: Rita Bolanos
3. Severity Assessment: Low
4. Decision: Rejected
5. Explanation: While we had originally had plans to create a login style website, we decided against it for various reasons, therefore I believe leaving the data storage strategy as it is is a more feasible goal for the scope of our project.

Annaliese Dedmore

In reference to allergies.html

Feedback 1

1. AI Feedback: AI's original suggestion text: Issue: Extensive use of inline styles (style="..."). Suggestion: Move all presentational styling into the external stylesheet (css/main.css) by creating semantic utility or component classes. The HTML should focus solely on structure and content. **Why it hurts**: (Tight Coupling) It couples presentation logic directly to the structure. This makes it impossible to globally change the application's look or layout without editing multiple HTML files. Developers must hunt through HTML elements to find and update styling, violating the Don't Repeat Yourself (DRY) principle
2. Assigned Reviewer: Annaliese Dedmore
3. Severity Assessment: Medium
4. Decision: Rejected
5. Explanation: A primary reason as to why these changes are being rejected is due to the fact that it will conflict with corresponding files and how they are written. Additionally, the code being written this way may violate principles however it does not cause any known issues, making it not necessary to alter.

Feedback 2

1. AI Feedback: Issue: Inline JavaScript event handlers (onclick, onsubmit). Suggestion: Remove all inline JavaScript event handlers and handle all interaction logic using unobtrusive JavaScript within the external file (js/app.js). This improves maintainability and makes debugging easier. **Why it hurts**: (Logic Fragility) It splits application logic between two files (HTML and JS). If the JavaScript file is updated, the developer must remember to check and potentially update corresponding inline handlers in the HTML, leading to brittle, hard-to-debug code and an increased chance of missed updates.
2. Assigned Reviewer: Annaliese Dedmore
3. Severity Assessment: Low
4. Decision: Rejected

5. Explanation: Similarly to feedback 1, refactoring the code may cause problems within other aspects of it. Furthermore, the JS code may require inlines in order to execute in the desired manner due to whatever is required for the specific environment.

Feedback 3

1. AI Feedback: Issue: Inconsistent application of margin/padding, mixing utility classes (mt-3) with inline styles (margin-bottom:1rem;). Suggestion: Adopt a consistent set of standardized spacing utility classes (e.g., following a scale like 1/2/3/4) and ensure they are used exclusively for all margins and padding. **Why it hurts**: (Visual Inconsistency & Cognitive Load) Developers have to memorize both a spacing utility scale (mt-3 means Xvalue) and custom pixel/rem values. This inconsistency slows down development and increases the likelihood of creating unintentional visual inconsistencies across the application.
2. Assigned Reviewer: Annaliese Dedmore
3. Severity Assessment: Low
4. Decision: Approved
5. Explanation: The primary reason for approving this suggestion is because it provides consistency within the visual aspect of the application. It also approves scalability.

Feedback 4

1. AI Feedback: Issue: Hardcoded, placeholder elements (like the example pill) that JavaScript needs to target and manage. Suggestion: Use data- attributes to semantically mark elements intended for dynamic manipulation (rendering, removal, or updating) by JavaScript. This clarifies the contract between HTML and the frontend script. **Why it hurts**: (Increased Code Complexity & Risk) JavaScript has to rely on specific class names or IDs that might be reused for styling or structure. Using data- attributes creates a clear contract that an element is being targeted for behavior/data, reducing the risk of accidentally breaking JS functionality by changing a class used primarily for CSS styling.
2. Assigned Reviewer: Annaliese Dedmore
3. Severity Assessment: Low
4. Decision: Rejected
5. Explanation: The way that the code is hardcoded still ensures that the code works, meaning that changing this will be relatively unnecessary as refactoring would be extensive yet the changes will be incredibly minimal.

Nile Ham

In reference to weight.html file

Feedback 1

1. AI Feedback: The architecture of the weight.html file presents a significant maintainability concern primarily due to the lack of clear separation of concerns between structure, presentation, and behavior. Ex: Inline and Embedded Styles, Unnecessary HTML Structure for Layout, Tight Coupling of HTML and JavaScript. **Why it hurts:** it forces developers to edit hundreds of lines of HTML to make a single visual change, instead of updating one spot in the CSS file.
2. Assigned Reviewer: Nile Ham
3. Severity Assessment: Low

4. Decision: Rejected
5. Explanation: While this is something that is important to change. Due to the current stage we are at, its priority is very low and only really affects the visual changes, not really the main code base itself. When it comes down to it, once we are finished with everything else, we can edit the css sheet and html files to be easier to change, but as of now this change is not needed and only provides looks rather than function.

Feedback 2

1. AI Feedback: The design of the form and button elements shows decent initial modularity with classes like .btn and .form-grid, but the overall design is compromised by element-specific customizations. Ex: Low Cohesion in the Button Handler, Implicit Styling Through Layout Classes, Redundant Style Specification. **Why it hurts**: scattering the logic; a button's function is split between the HTML's onclick attribute and the external JavaScript file, making it hard to track event handling.
2. Assigned Reviewer: Nile Ham
3. Severity Assessment: low
4. Decision: Rejected
5. Explanation: Once again this isn't important to the code function. Changing this would be far more time consuming for something that just changes the readability of the code rather than a core issue or function. Another reason is that as this is a team project styles will be different so adding what fits best is the better option. This just means there needs to be comments explaining more on that part of the code.

Feedback 3

1. AI Feedback: Code quality issues are centered on inconsistencies in styling and lack of self-documenting code in certain areas. Ex.Inconsistent Styling Method, Unclear Element Roles. **Why it hurts:** Makes development unpredictable. A developer doesn't know if they should look in the CSS file, a utility class, or an inline style to change a component's look, wasting time.
2. Assigned Reviewer: Nile Ham
3. Severity Assessment: Low
4. Decision: Rejected
5. Explanation: Readability is something you can start with, but isn't necessary and something that can be fixed later when there is a fully working project. The more people working on the code the more style and techniques that are used, there is bound to be some confusion until the end. If there are still confusing parts using comments would be much more efficient then restructuring the code for it to be clearer on where to look.

Feedback 4

1. AI Feedback: A key maintainability issue stems from the tight coupling of generic element tags and specific semantic styling within the HTML, necessitating repeated and verbose inline styles instead of using dedicated, descriptive CSS classes. **Why it hurts:** violates the principle of Don't Repeat Yourself (DRY) and separates concerns poorly, leading to code that is time-consuming to update and error-prone.
2. Assigned Reviewer :Nile Ham
3. Severity Assessment: Medium
4. Decision: Approved

5. Explanation: This is something that is being worked on and will be changed later on as the code enters the final stages as this is more of a problem with getting different sections to work on and bring them all together. Which help with production of the code rather than readability.

Luke Flaker

In reference to the 'goals.html' file:

1. AI Feedback: Duplicate script inclusion risking double initialization. The javascript initialization is referenced in <head> with defer and then it is referred again before <body> without defer. This causes the browser to execute the file twice. **Why this hurts:** Can be harder to reuse markup for other pages. Duplicate script inclusion can create many duplicates of code such as event listeners, intervals, state loads, and more. This can negatively interfere with the behavior of the code.
2. Assigned Reviewer: Luke Flaker
3. Severity Assessment: Medium
4. Decision: Approved
5. Explanation: Since we already use defer to reference javascript, there is no benefit to using script at the bottom of body. Defer is cleaner, handles ordering better, and is more consistent across browsers.

1. AI Feedback: Inconsistent naming and prefixes. ID's use the prefix _g for goal summary, while there are other ID's that are full words. This creates inconsistency.
2. Assigned Reviewer: Luke Flaker
3. Severity Assessment: low
4. Decision: rejected
5. Explanation: While our ID's are not the most consistent, they are functional and properly named to match their function. We also are not planning to grow the codebase, or develop more pages so there is no benefit to changing the variable names.

1. AI Feedback: Inline CSS styles reduce cohesion of CSS. This separates the appearance between many pages and elements. **Why it Hurts:** Inline styles can't be overwritten and can only be used for one element. This makes it harder to make any styling changes.
2. Assigned Reviewer: Luke Flaker
3. Severity Assessment: Low
4. Decision: Rejected
5. Explanation: Although Ai is correct about inline styling being inefficient for making changes, our team is satisfied with the design and layout of our application and we don't plan on making any big styling changes.

1. AI Feedback: Mixed responsibilities: markup contains behavior details. Our HTML page has behavior definitions. For example: <form class="form-grid mt-3" id="goalsForm" onsubmit="return false">. Behaviors should be defined in Javascript and content should

be defined in HTML. **Why it hurts:**It can become hard to track behavior when scattered across multiple pages, which can make debugging more difficult. Inline JS is also not reusable.

2. Assigned Reviewer: Luke Flaker
3. Severity Assessment: medium
4. Decision: Accepted
5. Explanation: in order to be able to debug our application, it is best if we have our functionality contained in Javascript. This will help create clean markup.

5.3 Individual Reflection

Completed:

1. Logan Bankert
2. Rita Bolanos
3. Annaliese Dedmore
4. Nile Ham
5. Isidro Marquez

(Fill out this questionnaire: https://qualtrics.nau.edu/jfe/form/SV_6Dtu3QGYSVFa6IC)

Retrospective

Overall during the course of this project, our team learned a lot about more soft skills that we had less experience in than simple coding. For example, our team learned a lot about how to adapt to and solve problems regarding communication, scheduling and other development issues outside of programming. On the programming side however, we also gained experience in managing a git repository, creating automated tests, and managing code updates and changes. The obstacles our team faced–especially during the early stages of development–were mostly communication related as our team struggled to manage meeting times and between meeting communication, however these issues became easier to resolve over time as the team became more used to working together on the project. Another obstacle we struggled with initially was scale, as when the project was first discussed we were unsure of whether we wanted the website to be more general or more specific to NAU, in the end we decided our website would be more general but with more limited functionality then originally proposed, removing things like the websites use of outside databases to help users shop for meals that hit their health goals. If we could expand the project adding features such as this would be our first step, then potentially developing the website into a mobile application would be our next expansion to make the software more accessible. Another potential expansion would be more styling and visual elements, particularly featuring the mascot more prominently and adding related theming. To conclude this project has been a great learning experience to work on, and the team agrees that while we encountered many challenges during development, that the project was fun to work on and resulted in a final product that the team could be proud of.