*Team Name: StockMyFoodBank*

*Team Members: Mitchell Cornett, Sally Nyambura, Sarah Murphree*

*Date: 03/23/2025*

*Project Title: StockMyFoodBank*

**Table of Contents**

Purpose — 2

System Summary — 2

System Scope (Full Version) — 3

Project Scope (Prototype) — 3

User Stories — 4

Use Case Scenario — 5

List of Steps — 6

Test Plan — 6

Use Cases — 10

Activity Diagram — 22

Class Diagram — 23

Sequence Diagram — 24

Basic Structure — 25

Risks and Assumptions — 26

API — 27

Wireframes — 28

State Diagram — 30

Class Responsibility Collaborator — 30

Task Object Responsibility — 32

Entity Relationship Diagram — 33

Nonfunctional Requirements — 34

User Acceptance Tests — 36

Unit Test Screenshots — 42

**Purpose**

A local food bank wants to gather information on the popularity of items which are currently stocked or may be stocked in the future. This information will be used to select items which are more popular and less likely to expire while still on the shelves. The food bank will manage the list of food items that are being voted on.

**System Summary**

The web application will allow clients to vote on how often they would select each food item when visiting a food bank. Instead of a simple Yes/No vote, clients will choose from multiple voting options: “Never,” “Yearly,” “Every 6 Months,” “Every Few Months,” “Monthly,” “Bi-Weekly,” and “Weekly.” Clients may also skip items without voting.

To enhance usability, clients can sort and filter the list of food items based on dietary restrictions/allergens (Dairy, Eggs, Nuts, Soy, Wheat, Shellfish, Fish, Sesame), and dietary preferences (Vegan, Pescatarian, Vegetarian, Keto, Paleo, Mediterranean, Low-Carb, Whole, Low-FODMAP, Low-Salt). A toggle will allow users to view all food items or only those that are free from their specified allergens. Food Items filtered out by allergen will be marked as “Never” and “Skip” for dietary preference filtering.

Clients will also be able to leave comments on food items to provide additional feedback, dietary concerns, or suggestions. Each user will have a profile where they can log in, save their personal information, and store dietary preferences for easier filtering in future sessions.

A publicly accessible results page will display aggregated vote counts for each food item, replacing voting buttons with numerical totals. Food bank managers can use this information to make informed inventory decisions. Additionally, administrators will have the ability to add or remove food items and may consider external factors such as cost, ease of transport, and availability when making final decisions.

This system ensures an efficient and user-friendly experience for clients while providing food bank managers with actionable insights to optimize inventory planning.

**System Scope (Full Version)**

The full version of the web app will have the same basic functionality as the prototype with several enhancements. Expanded voting options, instead of a simple Yes/No vote, users will be able to select how often they would choose each food item, with options “Never”, “Yearly”, “Every 6 Months”, “Every Few Months”, “Monthly”, “Bi-Weekly”, and “Weekly”. Ability to skip Items, users will have the option to skip an item without voting. Sorting & filtering, users can sort the list of food items and filter based on their allergens & restrictions (Dairy, Eggs, Nuts, Soy, Wheat, Shellfish, Fish, Sesame), and diets (Vegan, Pescatarian, Vegetarian, Keto, Paleo, Mediterranean, Low-Carb, Whole, LowFodMap, Low Salt). A toggle will allow users to view either all items or only items that are free of known allergens. Commenting system, users can leave comments on each item they vote on to provide additional feedback, dietary concerns, or suggestions. User profiles, users will be able to create a profile and login and save personal contact information. User preferences, dietary restrictions and preferences (such as vegetarian, vegan, gluten-free, and lactose-free) will be saved as user data to their profile for easier filtering in future sessions. Results page, the aggregated vote counts for each item will be displayed on a separate publicly accessible results page, replacing the voting buttons with numerical totals. Admin features, the food bank managers will be able to add or remove food items from the list and may consider external factors such as cost, ease of transport, and availability when making final decisions. This full version ensures a comprehensive user experience while providing food bank managers with actionable insights for inventory planning.

**Project Scope (Prototype)**

This version of the web app will have only basic functionality. Users will view a page split into two columns. One column will have a list of food items being considered by the food bank. The other column will have the question “Would you select this item during a visit?” with three response options next to each food item: “Yes,” “No,” and “Skip.” Users can select “Yes” or “No” for as many items as they wish, or choose to skip an item without voting. At the bottom of the page, there will be a freeform comment section where users can provide feedback or suggest dietary preferences, additional food items, or other relevant concerns. Submitting will increment a “Yes” or “No” counter for each item voted on. A separate page will display the results using the same format, with counts of Yes and No votes replacing Yes and No buttons and all comments submitted underneath.

**Project Details:**

**User Stories**

US1 - As a client, I want to visit the page to view the list of food items.

US2 - As a client, I want to view a webpage displaying all food items so that I can review and provide feedback.

US3 - As a food bank manager, I want to be able to navigate directly to the results page so that I can quickly view the overall food selection results.

US4 - As a client, I want to review each food item and select “Yes”, “No”, or “Skip” so that I can provide my feedback.

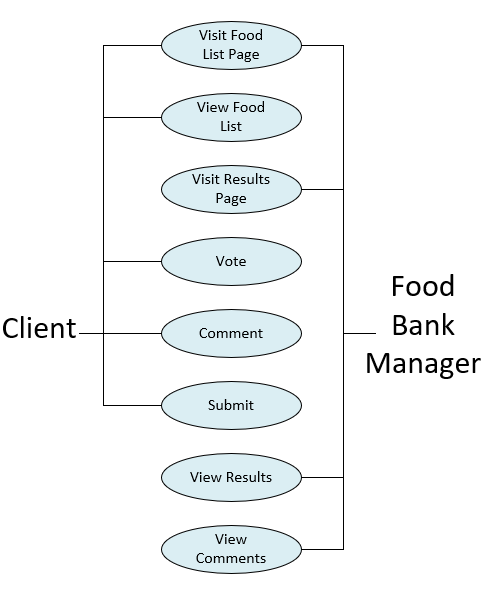
US5 - As a client, I want to enter freeform comments at the end of the page so that I can provide suggestions and feedback.

US6 - As a client, I want to submit my selections so that my votes are counted towards the final tally.

US7 - As a food bank manager, after the user clicks submit I want to view a summary page displaying the total counts of “Yes” and “No” votes for each food item so that I can analyze the results.

US8 - As a food bank manager, after the user clicks submit I want to review user comments at the bottom of the results page so that I can consider additional feedback and suggestions.

**Use Case Scenario**



**List of Steps**

FRUS1 - A user navigates to the web app via a provided link or website.

FRUS2 - A user will view the web page with all food items.

FRUS3 - A food bank manager can click on the results button, navigating straight to the results page.

FRUS4 - A user will review each food item, selecting “Yes”, “No”, or “Skip” next to each food Item.

FRUS5 - A user can enter in freeform comments at the end of the page for suggestions and feedback.

FRUS6 - A user will click submit at the bottom of the form, tallying up votes.

FRUS7 - A food bank manager, after results are submitted, will be able to view a page with the tallied vote counts of “Yes” and “No” next to each food item.

FRUS8 - A food bank manager, after submission, will view a page of food items with a list of comments at the bottom.

**Test Plan**

| Req. ID | Requirement | Acc. ID | Acceptance Criteria |
| --- | --- | --- | --- |
| FRUS1 | A user navigates to the web app via a provided link or website. | AC-1.1 | The client can access the web page via a provided link or website |
| AC-1.2 | The web page loads successfully without errors |
| AC-1.3 | The food selection page is displayed with a header and title. |
| AC-1.4 | The list of food items is visible to the client upon page load. |
| FRUS2 | A user will view the web page with all food items. | AC-2.1 | The webpage displays a list of food items. |
| AC-2.2 | Each food item has a name, description, and any relevant details. |
| AC-2.3 | The page layout is user-friendly and accessible. |
| AC-2.4 | Food items are arranged in a structured manner for easy review. |
| FRUS3 | A food bank manager can click on the results button, navigating straight to the results page. | AC-3.1 | A “Results” button is available on the main page for the food bank manager. |
| AC-3.2 | Clicking the “Results” button redirects the manager to the results page. |
| AC-3.3 | The results page loads correctly and displays aggregated data. |
| FRUS4 | A user will review each food item, selecting “Yes”, “No”, or “Skip” next to each food Item. | AC-4.1 | Each food item has three selection options: “Yes”, “No”, and “Skip.” |
| AC-4.2 | The client can select only one option per food item. |
| AC-4.3 | Clicking a selection updates the user's response dynamically. |
| AC-4.4 | The selection is retained unless the user changes it before submission. |
| FRUS5 | A user can enter in freeform comments at the end of the page for suggestions and feedback. | AC-5.1 | A text box is available at the bottom of the page for user comments. |
| AC-5.2 | The comment box allows freeform text input. |
| AC-5.3 | Comments are submitted along with the user’s food selections. |
| AC-5.4 | Comments are stored and visible in the results view for managers. |
| FRUS6 | A user will click submit at the bottom of the form, tallying up votes. | AC-6.1 | A “Submit” button is available at the bottom of the page. |
| AC-6.2 | Clicking the “Submit” button finalizes the user’s responses. |
| AC-6.3 | Upon submission, the system records the user’s selections. |
| AC-6.4 | The system prevents multiple submissions from the same user for a session. |
| FRUS7 | A food bank manager, after results are submitted, will be able to view a page with the tallied vote counts of “Yes” and “No” next to each food item. | AC-7.1 | The results page displays each food item with a count of “Yes” and “No” votes. |
| AC-7.2 | The results are updated dynamically after each submission. |
| AC-7.3 | The results are displayed in a list format for easy analysis. |
| FRUS8 | A food bank manager, after submission, will view a page of food items with a list of comments at the bottom. | AC-8.1 | The results page includes a section for user comments. |
| AC-8.2 | Comments are displayed in chronological order. |
| AC-8.3 | Only submitted comments appear on the results page. |

**Use Cases**

Use Case: UCFR1 - Navigate to Web Application

Traceability

* User Story: US-1
* Functional Step: FRUS-1
* Acceptance Criteria: FRUS1-AC1.1 to FRUS1-AC1.4

Description

This use case describes how a user navigates to the web application and accesses the food selection page.

Actors

* User (Client)
* System

Preconditions

* The user has access to a web browser.
* The web application is live and accessible.
* The user has a stable internet connection.

Postconditions

* The user successfully accesses the web application and views the homepage.

Steps

1. The user enters the web application's URL in the browser or clicks a provided link.
2. The system receives the request and loads the homepage.
3. The system verifies the application status and retrieves necessary resources.
4. The homepage is displayed with a header, title, and navigation options.
5. The system displays the food selection page with all available food items.

Alternative Steps

1. User Accesses the Web App via a Bookmark or Saved Link:
   * Instead of manually entering the URL, the user clicks a bookmarked link.
   * The system loads the homepage as usual.
2. User Revisits the Web App After a Previous Session:
   * If the user has visited the site before, the browser may auto-fill the URL.
   * The system may retrieve cached data for a faster experience.
3. User Arrives at the Page via a Search Engine:
   * The user searches for the web application on Google or another search engine.
   * The system loads the correct page when the user clicks the search result.

Exceptions

* Invalid URL Entered: The system displays a "Page Not Found" error if the user enters an incorrect URL.
* Network Issues Prevent Page Load: The system displays an error message indicating a connection issue.
* Server Downtime or Maintenance: The system displays a maintenance notice if the application is temporarily unavailable.

Nonfunctional Requirements

* The homepage must load within 3 seconds.
* The UI must follow accessibility standards for readability and navigation.
* The web application should support multiple browsers and devices.

Use Case: UCFR2 - View Webpage Displaying All Food Items

Traceability

* User Story: US-2
* Functional Step: FRUS-2
* Acceptance Criteria: FRUS2-AC2.1 to FRUS2-AC2.4

Description

This use case describes how a user accesses the food selection page and views all available food items along with their relevant details.

Actors

* User (Client)
* System

Preconditions

* The user has access to a web browser.
* The web application is live and accessible.

Postconditions

* The user successfully views a structured list of food items with names, descriptions, and relevant details.

Steps

1. The user navigates to the food selection page.
2. The system retrieves and loads the list of food items.
3. The system displays all food items with names and descriptions.
4. The system organizes the items in a structured and user-friendly format.
5. The user can scroll through the list to view all food items.

Alternative Steps

1. User Uses Pagination (If Available):
   * If there are many food items, the system divides them into multiple pages.
   * The user navigates through pages using "Next" and "Previous" buttons.

Exceptions

* Page Load Failure: If the food selection page fails to load, the system displays an error message.
* No Food Items Available: If there are no items in the database, the system displays a message stating that no food items are available at the moment.

Nonfunctional Requirements

* The webpage must load within 3 seconds.
* The layout must be accessible and easy to navigate on different devices (mobile, tablet, desktop).

Use Case: UCFR3 - Navigate to Results Page

Traceability

* User Story: US-3
* Functional Step: FRUS-3
* Acceptance Criteria: FRUS3-AC3.1 to FRUS3-AC3.1

Description

This use case describes how a food bank manager navigates directly to the results page to view the overall food selection results.

Actors

* Food Bank Manager
* System

Preconditions

* The user has access to a web browser.
* The web application is live and accessible.
* At least one vote has been recorded in the system.

Postconditions

* The food bank manager successfully views the results page, displaying aggregated vote counts for each food item.

Steps

1. The food bank manager navigates to the web application.
2. The system loads the homepage and displays the navigation menu.
3. The food bank manager clicks on the "Results" button.
4. The system retrieves the latest voting data.
5. The system loads and displays the results page, showing vote counts for each food item and all comments that have been submitted.

Alternative Steps

1. Manager Accesses Results Page via Direct Link:
   * Instead of navigating through the homepage, the manager enters the direct URL to the results page.
   * The system loads the results page without needing to go through the homepage.
2. Manager Refreshes the Results Page to View Updated Data:
   * If new votes have been submitted, the manager refreshes the page.
   * The system retrieves the latest data and updates the results.

Exceptions

* Results Page Fails to Load: The system displays an error message if there is a server issue.
* No Votes Available: The system displays a message stating that no votes have been recorded yet.

Nonfunctional Requirements

* The results page must load within 3 seconds.
* The data displayed must be up-to-date and dynamically refreshed when new votes are submitted.
* The UI should be structured for easy analysis, ensuring accessibility for all users.

Use Case: UCFR4 - Vote on Food Items

Traceability

* User Story: US-4
* Functional Step: FRUS-4
* Acceptance Criteria: FRUS4-AC4.1 to FRUS4-AC4.4

Description

This use case describes how a client reviews food items and selects "Yes", "No", or "Skip" to provide feedback on each item.

Actors

* User (Client)
* System

Preconditions

* The user has access to a web browser.
* The web application is live and accessible.
* The user can view the list of food items.

Postconditions

* The user’s voting selections are successfully recorded.

Steps

1. The user navigates to the food selection page.
2. The system displays a list of food items with voting options.
3. The user reviews each food item.
4. The user selects one option per food item:
   * "Yes" if they would take the item.
   * "No" if they would not take the item.
   * "Skip" if they do not wish to vote on the item.
5. The system records the selection in real time.
6. The selection remains unless the user changes it before submission.

Alternative Steps

1. User Changes Selection Before Submission:
   * The user modifies their choice for an item before submitting the final vote.
   * The system updates the recorded selection accordingly.
2. User Scrolls Through Items Instead of Voting Immediately:
   * The user chooses to review all items before selecting any options.
   * The system does not prompt the user to vote immediately.

Exceptions

* System Fails to Record Selection: If an option does not register, the system displays a retry message.
* User Selects More than One Option for an Item: The system prevents multiple selections per item and prompts the user to choose only one.

Nonfunctional Requirements

* Selections should be recorded dynamically without requiring a page refresh.
* The UI should ensure clear visibility of options for accessibility.
* The selection process should be responsive and work on various devices.

Use Case: UCFR5 - Enter Freeform Comments

Traceability

* User Story: US-5
* Functional Step: FRUS-5
* Acceptance Criteria: FRUS5-AC5.1 to FRUS5-AC5.4

Description

This use case describes how a client provides additional feedback or suggestions by entering freeform comments on the food selection page.

Actors

* User (Client)
* System

Preconditions

* The user has access to a web browser.
* The web application is live and accessible.
* The user is on the food selection page.

Postconditions

* The user's comments are successfully recorded and displayed in the results view.

Steps

1. The user scrolls to the bottom of the food selection page.
2. The system displays a text box labeled “Comments” or similar.
3. The user enters their freeform comment.
4. The system validates the comment input (e.g., ensuring it is not empty if required).
5. The user submits the comment along with their food selections.
6. The system stores the comment and associates it with the user’s submission.
7. The system displays the comment in the results view for food bank managers.

Alternative Steps

1. User Edits or Deletes a Comment Before Submission:
   * The user modifies or removes the comment before submitting the form.
   * The system updates the stored comment accordingly.
2. User Submits a Comment Without Voting on Food Items:
   * The user enters a comment without selecting “Yes” or “No” for food items.
   * The system allows submission of comments independently from votes.
3. User Exits the Page Before Submitting a Comment:
   * If the user navigates away, the system warns them that their comment will not be saved unless submitted.
4. User Enters a Long Comment (If Character Limit Exists):
   * If the comment exceeds the allowed character limit, the system prompts the user to shorten it.

Exceptions

* System Fails to Store the Comment: The system displays an error message and prompts the user to try again.
* User Tries to Submit an Empty Comment: The system prompts the user to enter valid text before submission.

Nonfunctional Requirements

* The comment box must support various input types (text, punctuation, emojis, etc.).
* Submission should be processed in under 3 seconds.
* The UI must ensure easy readability and accessibility.

Use Case: UCFR6 - Submit Food Selection Votes

Traceability

* User Story: US-6
* Functional Step: FRUS-6
* Acceptance Criteria: FRUS6-AC6.1 to FRUS6-AC6.4

Description

This use case describes how a client submits their voting selections for food items, finalizing their input.

Actors

* User (Client)
* System

Preconditions

* The user has access to a web browser.
* The web application is live and accessible.
* The user has reviewed and made selections for food items (Yes, No, or Skip).

Postconditions

* The user's selections are successfully recorded.
* The voting results update to reflect the user’s input.

Steps

1. The user reviews their selections on the food selection page.
2. The system displays a “Submit” button at the bottom of the page.
3. The user clicks the “Submit” button.
4. The system validates the selections and ensures all required fields are completed.
5. The system records the user’s selections in the database.
6. The system provides a confirmation message stating that the votes have been successfully submitted.

Alternative Steps

1. User Tries to Submit Without Making Any Selections:
   * The system prompts the user to make at least one selection before submitting.
2. User Edits Selections Before Submission:
   * The user modifies their choices before clicking "Submit."
   * The system updates the stored responses accordingly.
3. User Navigates Away Before Submission:
   * If the user leaves the page before submitting, their selections are lost unless auto-saved.

Exceptions

* Network Failure During Submission: The system displays an error message and suggests retrying when the connection is restored.
* System Fails to Record Votes: The system provides an error message and logs the issue for review.
* Session Timeout Before Submission: If the session expires, the system prompts the user to refresh the page and re-enter selections.

Nonfunctional Requirements

* Submission must be processed in under 3 seconds.
* The system must ensure data integrity and prevent duplicate submissions.
* The UI should provide clear feedback upon successful or failed submission.

Use Case: UCFR7 - View Voting Results Summary

Traceability

* User Story: US-7
* Functional Step: FRUS-7
* Acceptance Criteria: FRUS7-AC7.1 to FRUS7-AC7.3

Description

This use case describes how a food bank manager views the summarized voting results for each food item after users have submitted their votes.

Actors

* Food Bank Manager
* System

Preconditions

* The user has access to a web browser.
* The web application is live and accessible.
* At least one vote has been recorded in the system.

Postconditions

* The food bank manager successfully views the total counts of "Yes" and "No" votes for each food item.
* The manager can analyze voting results for inventory decision-making.

Steps

1. The food bank manager navigates to the web application.
2. The system displays the homepage with a navigation menu.
3. The manager clicks on the “Results” button to access the results page.
4. The system retrieves and loads the latest voting data.
5. The system displays the total number of "Yes" and "No" votes for each food item.
6. The manager can scroll through the list to review all food items.

Alternative Steps

1. Manager Refreshes the Page for Updated Results:
   * If new votes have been submitted, the manager refreshes the results page.
   * The system retrieves and displays the most recent data.

Exceptions

* No Votes Available Yet: The system displays a message stating that no votes have been recorded.
* Results Page Fails to Load: The system displays an error message and suggests reloading the page.
* Network Issues Prevent Data Retrieval: The system notifies the manager of a connection issue and suggests trying again later.

Nonfunctional Requirements

* The results page must load within 3 seconds.
* The displayed data should be dynamically updated when new votes are submitted.
* The interface must be structured for easy analysis and accessible across devices.

Use Case: UCFR8 - Review User Comments on Results Page

Traceability

* User Story: US-8
* Functional Step: FRUS-8
* Acceptance Criteria: FRUS8-AC8.1 to FRUS8-AC8.3

Description

This use case describes how a food bank manager reviews user-submitted comments on the results page to consider additional feedback and suggestions.

Actors

* Food Bank Manager
* System

Preconditions

* The user has access to a web browser.
* The web application is live and accessible.
* At least one comment has been submitted by a client.

Postconditions

* The food bank manager successfully views all user-submitted comments.
* The manager can analyze comments to understand client concerns, preferences, or suggestions.

Steps

1. The food bank manager navigates to the web application.
2. The system displays the homepage with a navigation menu.
3. The manager clicks on the “Results” button to access the results page.
4. The system retrieves and loads the latest voting results.
5. The system displays the total number of "Yes" and "No" votes for each food item.
6. The system displays a section at the bottom of the page with all submitted user comments.
7. The manager reviews the comments, which are displayed in chronological order.

Alternative Steps

1. Manager Scrolls Through Paginated Comments (If Many Exist):
   * If there are many comments, the system divides them into pages.
   * The manager clicks "Next" or "Previous" to navigate through the comments.

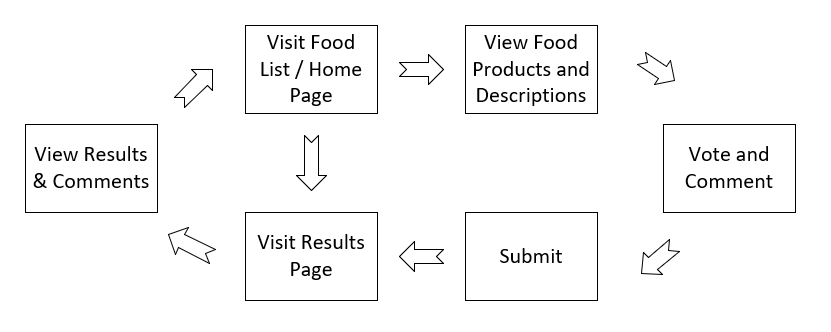
Exceptions

* No Comments Available Yet: The system displays a message stating that no user comments have been submitted.
* Results Page Fails to Load: The system displays an error message and suggests reloading the page.
* Network Issues Prevent Data Retrieval: The system notifies the manager of a connection issue and suggests trying again later.

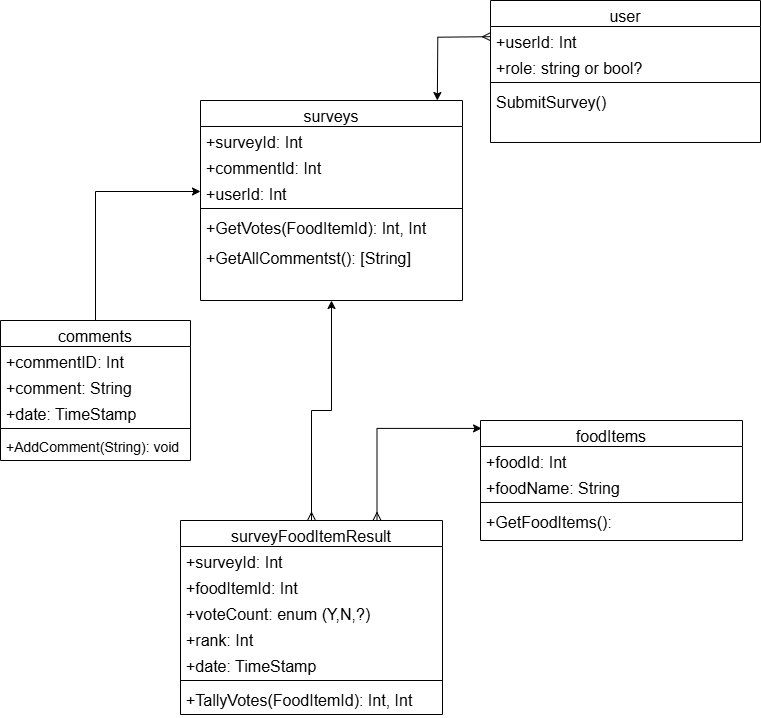
Nonfunctional Requirements

* The comments section must load within 3 seconds.
* The system must display comments in a structured, easy-to-read format.
* Filtering, searching, and sorting should function without requiring a page refresh.

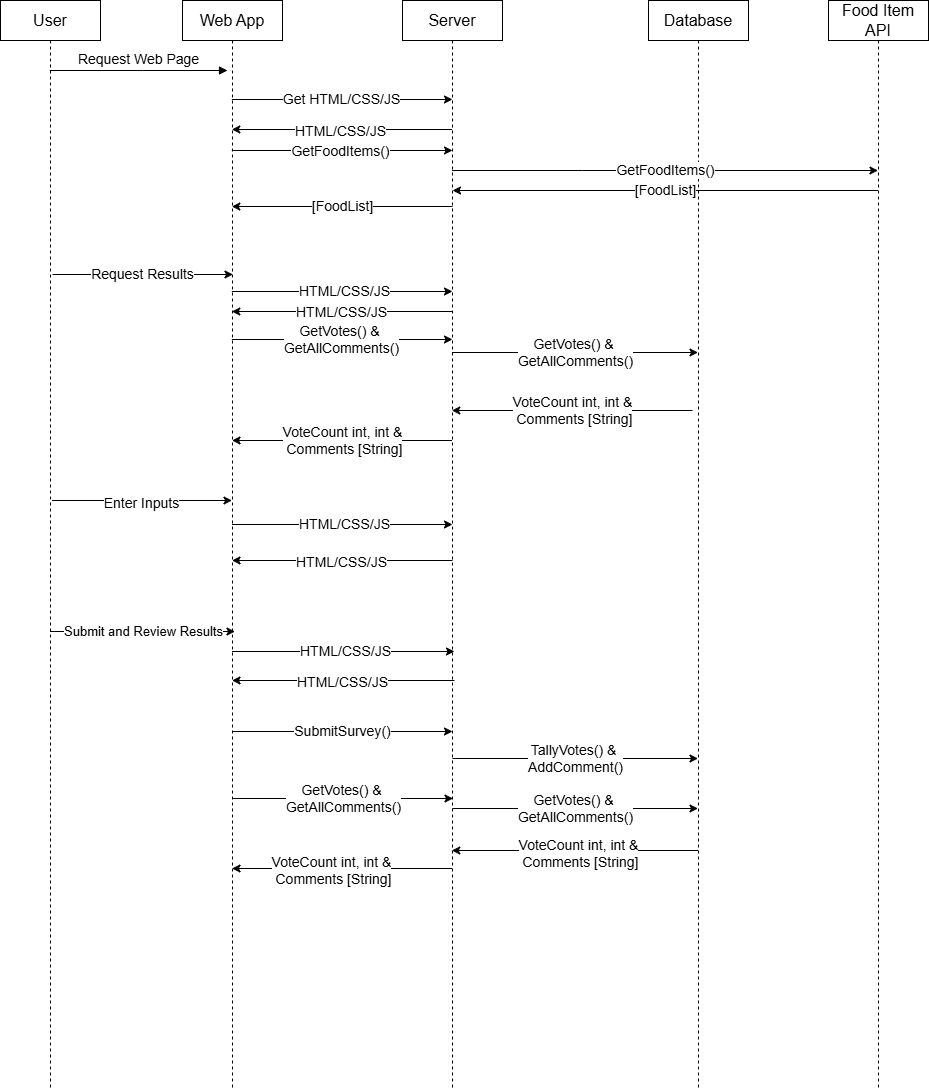
**Activity Diagram**



**Class Diagram:**

****

**Sequence Diagram:**

****

**Basic Structure:**

* Two Columns
  + Left Column: A list of food items being considered by the food bank
  + Right column: The question ‘would you select this item during a visit?” with “Yes”, “No”, “Skip” buttons for each item and a straight to results link at the top.
* Voting Mechanism:
  + Users can select "Yes", "No", “Skip” for as many food items as they wish.
  + They can skip items without voting.
  + When they submit their choices, the number of votes for "Yes" and "No" will increase accordingly.
  + A text box is at the bottom of the list for users to enter in additional comments.
* Displaying Results:
  + After submission, the results page will display the same list of food items, but instead of "Yes" and "No" buttons, there will be vote counts showing how many people selected "Yes" and how many selected "No" for each item, allowing food bank managers to see how popular each item is.
* Target Devices:
  + Web application is accessible on:
  + Phones
  + Tablets
  + Laptops
  + Other internet-connected devices
* ADA Compliance
  + Must be in compliance with ADA standards for web accessibility.
* Technological Constraints:
  + Will be using public APIs
  + Simple UI
  + Interacts with local datastore or a file system to store or retrieve data
* In Scope
  + Basic voting on food items
  + Displaying voting results (yes/no and frequency of answer)
  + Commenting on surveys
* Out of Scope:
  + Payment processing
  + Machine learning or advanced data analysis
  + Third party integrations beyond public APIs

**Risks and Assumptions:**

* Risks:
  + Time constraints for implementing all features before the deadline
* Assumptions:
  + Clients will have basic knowledge of using web apps
  + The food bank will provide a list of food items to be voted on

**API:**

Data is provided from the Spoonacular food API

Source Link: <https://spoonacular.com/food-api>

API Link:

[https://api.spoonacular.com/food/products/search?apiKey=b4fc3a18f1344fca9211962e6e93d321&query="meal"&minCalories=100&number=10](https://api.spoonacular.com/food/products/search?apiKey=b4fc3a18f1344fca9211962e6e93d321&query=%22meal%22&minCalories=100&number=10)

This API provides a list of products stocked in grocery stores. Our specific API link returns ten different meal items with at least 100 calories.

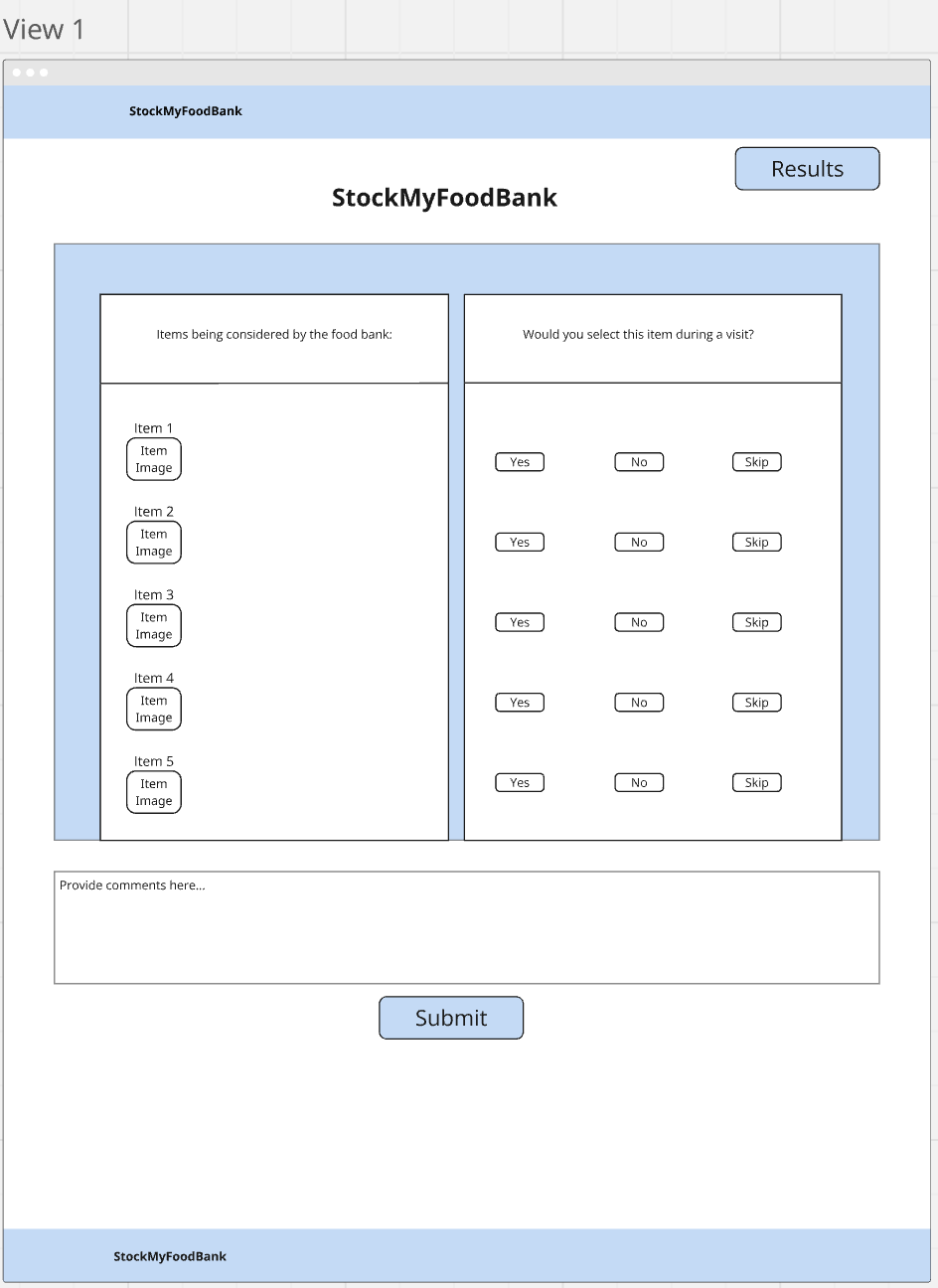
Data Model:

The API returns an array of product objects. We will be using the following fields from each product object.

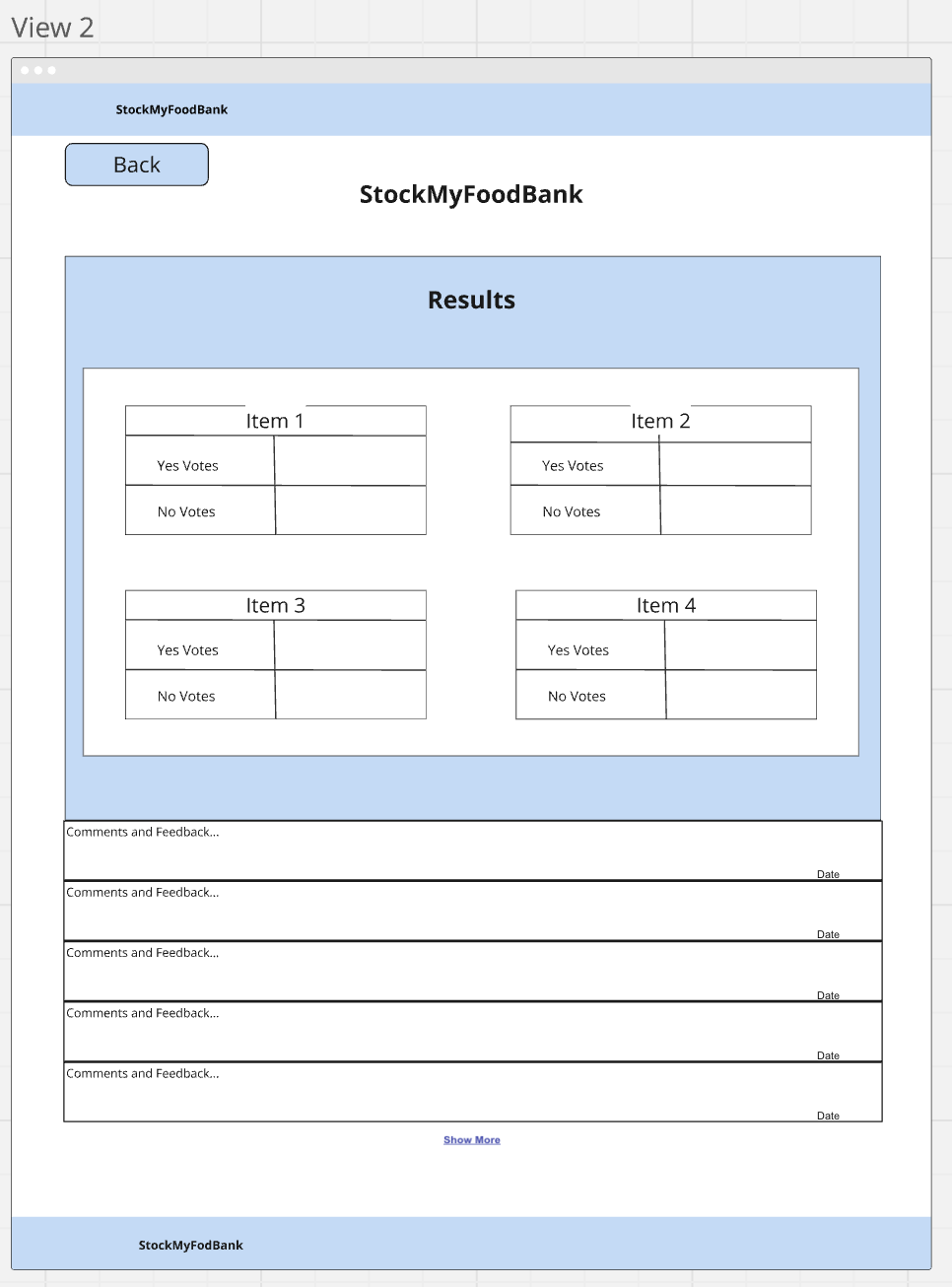
| Field | Data Type | Description |
| --- | --- | --- |
| id | integer | A unique identifier |
| title | string | A description of the product, sometimes including quantity or weight |
| image | string | A URL to an image of the food item |
| imageType | string | The file type of the image linked in the image field |

**Wireframes:**

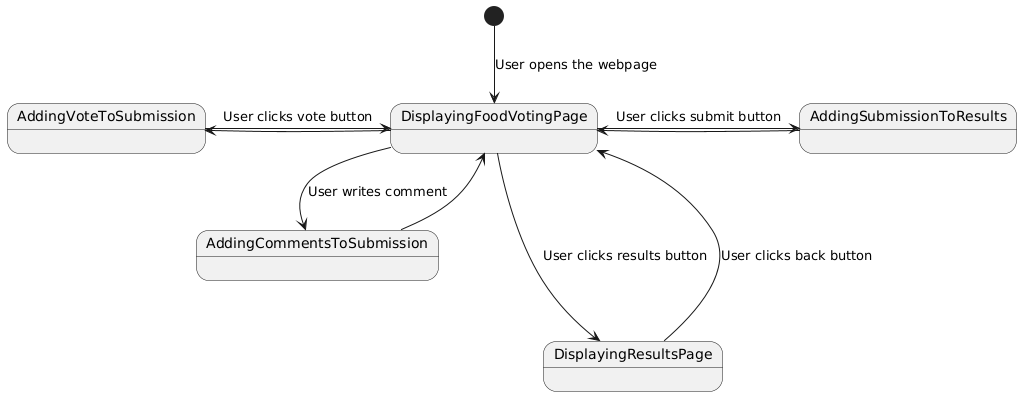
Survey/HomePage



Results Page:

****

**State Diagram:**



**Class Responsibility Collaborator:**

| User | |
| --- | --- |
| Responsibility | Collaborators |
| **userId**  **surveyIds**  **role** | **Survey** |

| FoodItem | |
| --- | --- |
| Responsibility | Collaborators |
| **foodId**  **foodName** | **SurveyFoodItemResult** |

| Surveys | |
| --- | --- |
| Responsibility | Collaborators |
| **surveyId**  **commentId**  **surveyFoodIdResultsId**  **userId** | **User**  **Comments**  **SurveyFoodItemResult** |

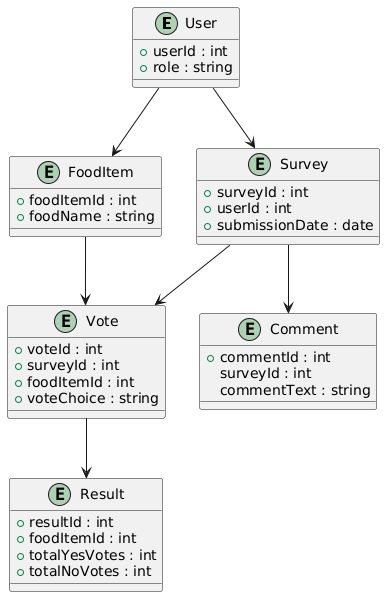
| SurveyFoodItemResults | |
| --- | --- |
| Responsibility | Collaborators |
| **SurveyId**  **FoodItemId**  **VoteCount**  **Rank**  **Date** | **Surveys**  **FoodItems** |

| Comments | |
| --- | --- |
| Responsibility | Collaborators |
| **CommentId**  **Comment**  **Date** | **Surveys** |

**Task Object Responsibility:**

| **Object Name** | **Responsibility** |
| --- | --- |
| **TallyVotes()** | **Aggregates and stores vote counts per food item**  **Updates vote count dynamically**  **Accepts foodItemId** |
| **GetAllComments()** | **Retrieves all submitted comments from users**  **Displays comments from surveys collected** |
| **GetVotes()** | **Retrieves individual votes for each food item**  **Provides vote data for analysis**  **Accepts foodItemId** |
| **AddComment()** | **Allows users to edit or delete comments before submission**  **Links comments to specific survey**  **Accepts a string** |
| **GetFoodItems()** | **Retrieves the list of available food items**  **Displays food item details (name)** |
| **SubmitSurvey()** | **Creates a survey submission** |

**Entity Relationship Diagram:**

****

**Nonfunctional Requirements**

| Req. ID | Nonfunctional requirement ID | Nonfunctional requirement | Test |
| --- | --- | --- | --- |
| FRUS1 | NFR1 | The web application should load within 3 seconds | Measure the time it takes to load the webpage under typical network conditions |
| FRUS2 | NFR2 | The system retrieves and displays all food items | Measure the time it takes to load the page with a specified number of food items |
| FRUS3 | NFR3 | The navigation from the current page to the results page should take less than 3 seconds | Click the results button and measure the time it takes to navigate to the results page |
| FRUS4 | NFR4 | The system only allows one selection of the options of “yes/no/skip” per food item and displays feedback of option selection | Selection of option per food item is recorded dynamically without requiring a page refresh |
| FRUS5 | NFR5 | The system supports storing and retrieving up to 5,000 characters of comments without data loss or performance issues. | Input a large comment of 5,000 characters to verify that the system can save the comment and retrieve it correctly |
| FRUS6 | NFR6 | The submission button should successfully process and record all votes in less than 3 seconds | Submit the form with multiple votes and measure the time it takes for the vote tallying process to complete |
| FRUS7 | NFR7 | The system’s results page should load with tallied votes in less than 5 seconds | Measure the load time of the results page after submission to ensure it takes less than 5 seconds |
| FRUS8 | NFR8 | The system’s results page displays food item comments within 3 seconds | Measure the load time of comments in results page and ensure that it takes less than 3 seconds and displays comments in chronological order |

**User Acceptance Tests**

Test ID: 01

Unit Tested: Food list page / homepage

Preconditions: User has an internet connection and a link to the web app

Environment: Web browser

Test Data: None

Steps: User clicks a link to the web app

Expected Results: Browser loads the food list page

Actual Results:

Pass/Fail:

Priority:

Severity:

Comments:

Test ID: 02

Unit Tested: Food item list

Preconditions: User has loaded the food list page

Environment: Web browser

Test Data: None

Steps: User scrolls or uses paging buttons to view the entire list of food items

Expected Results: The pre-defined number of food items are all visible

Actual Results:

Pass/Fail:

Priority:

Severity:

Comments:

Test ID: 03

Unit Tested: Voting buttons

Preconditions: All food items are visible on the food item list page

Environment: Web browser

Test Data: None

Steps: User looks to the right of each food item

Expected Results: Three buttons are next to each food item, a Yes button, No button, and Skip button

Actual Results:

Pass/Fail:

Priority:

Severity:

Comments:

Test ID: 04

Unit Tested: Voting buttons

Preconditions: Voting buttons are visible

Environment: Web browser

Test Data: None

Steps: User clicks one of the three voting buttons

Expected Results: The clicked voting button visually indicates that it has been selected

Actual Results:

Pass/Fail:

Priority:

Severity:

Comments:

Test ID: 05

Unit Tested: Voting buttons

Preconditions: Voting buttons visually indicate they have been selected

Environment: Web browser

Test Data: None

Steps: Click a voting button, then click a different voting button for the same food item

Expected Results: The first voting button should visually indicate it has unselected when the second voting button is clicked

Actual Results:

Pass/Fail:

Priority:

Severity:

Comments:

Test ID: 06

Unit Tested: Voting buttons

Preconditions: Voting buttons visually indicate they have been selected

Environment: Web browser

Test Data: None

Steps: Click a voting button, then click a voting button on a different food item

Expected Results: The voting button choice on the first food item should be maintained when the button for the second food item is clicked

Actual Results:

Pass/Fail:

Priority:

Severity:

Comments:

Test ID: 07

Unit Tested: Results button

Preconditions: User has loaded the food list page

Environment: Web browser

Test Data: None

Steps: User looks to the top right of the food list page

Expected Results: A results button is on the top right of the food list page

Actual Results:

Pass/Fail:

Priority:

Severity:

Comments:

Test ID: 08

Unit Tested: Results button

Preconditions: Results button is visible

Environment: Web browser

Test Data: None

Steps: User clicks the results button

Expected Results: The user is redirected to the results page

Actual Results:

Pass/Fail:

Priority:

Severity:

Comments:

Test ID: 09

Unit Tested: Results page

Preconditions: User has loaded the results page

Environment: Web browser

Test Data: Submitted surveys

Steps: User scrolls or uses paging buttons to view the entire list of results

Expected Results: All of the results are all visible, including Yes votes, No votes, and comments

Actual Results:

Pass/Fail:

Priority:

Severity:

Comments:

Test ID: 10

Unit Tested: Comments text box

Preconditions: User has loaded the food list page

Environment: Web browser

Test Data: None

Steps: User looks to the bottom of the food item list page

Expected Results: A text entry box is visible

Actual Results:

Pass/Fail:

Priority:

Severity:

Comments:

Test ID: 11

Unit Tested: Comments text box

Preconditions: Comments text box is visible

Environment: Web browser

Test Data: Sample text

Steps: Type in the comments text box

Expected Results: The comments text box allows text entry

Actual Results:

Pass/Fail:

Priority:

Severity:

Comments:

Test ID: 12

Unit Tested: Comments text box

Preconditions: Comments text box is visible

Environment: Web browser

Test Data: Sample text and submitted survey

Steps: Type in the comments text box, vote on at least one item, and submit the survey. Go to the results page.

Expected Results: The text typed in the comments text box should be visible as a comment on the results page

Actual Results:

Pass/Fail:

Priority:

Severity:

Comments:

Test ID: 13

Unit Tested: Submit button

Preconditions: User has loaded the food list page

Environment: Web browser

Test Data: None

Steps: User looks at the bottom of the food item list page

Expected Results: A submit button is visible below the comments text box

Actual Results:

Pass/Fail:

Priority:

Severity:

Comments:

Test ID: 14

Unit Tested: Submit button

Preconditions: Submit button is visible

Environment: Web browser

Test Data: Submitted survey

Steps: Vote on at least one food item and click submit. Attempt to vote on the same food item again

Expected Results: The food item cannot be voted on again after clicking submit

Actual Results:

Pass/Fail:

Priority:

Severity:

Comments:

Test ID: 15

Unit Tested: Submit button

Preconditions: Submit button is visible

Environment: Web browser

Test Data: Submitted survey

Steps: View the results page and record the number of votes on an item. On the food item list page, vote on that item. Submit the survey and return to the results page. Record the number of votes on the item

Expected Results: After submitting the survey, the number of votes on the item changes accordingly

Actual Results:

Pass/Fail:

Priority:

Severity:

Comments:

Test ID: 16

Unit Tested: Results page

Preconditions: User has loaded the results page

Environment: Web browser

Test Data: Submitted surveys

Steps: Submit multiple surveys with comments. Navigate to the results page

Expected Results: Comments are sorted in chronological order

Actual Results:

Pass/Fail:

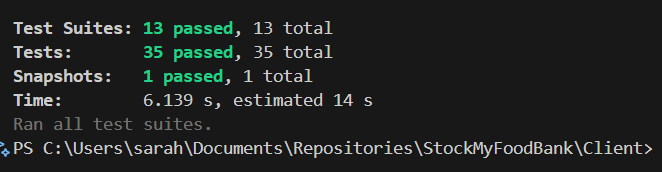
Priority:

Severity:

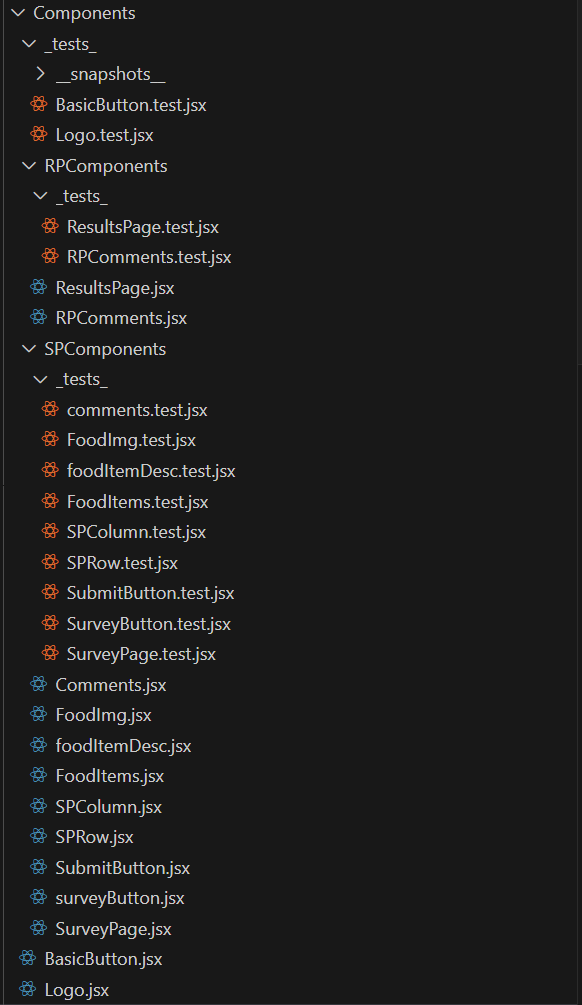
Comments:

**Unit Test Screenshots:**

Jest test:



Front end tests:



Back end tests:

