Test Plan

Team No Stress
12/15/2021
Daniel Ramirez (Team Leader)
Hunter Lewis
Luke Sunaoka
Matthew Quinn
Michelle Vo
Tyler Thorin

1 Overview

A project test plan is basically the planning and preparing for the testing phase of software development. The test plan should outline the major functions that need to be tested, what specific tests will be performed, best approach to take for the tests, what to and not to test, how the test should be performed, who is responsible for each test, what results should be expected, define and consider a successful or failed test, and the criteria or requirement that must be completed for the test during the testing phase. The test plan must also describe in detail the method, goals, approach, and whatever else should be used for each test.

2 Types of Software Testing

Although there are many more types of software testing in existence, our test plan will primarily revolve around performing these types of tests:

- Compatibility Testing
    - Compatibility testing ensures that the feature is compatible with different configurations of devices.
- Functional Testing
    - Functional testing tests the core functionality of the feature to ensure that it is functioning correctly.
- GUI Testing
    - GUI testing tests the usability and aesthetics of the user-interface relative to the application.
- Integration Testing
    - Integration testing ensures that an entire, integrated system meets a set of requirements and that the system functions properly.
- Non-Functional Testing
    - Non-Functional testing verifies the readiness of a system according to nonfunctional parameters that are not addressed in functional testing.
- Performance Testing
    - Performance testing tests the speed, stability, scalability, reliability and resources usage of a software application placed under a specific workload.
- Security Testing
    - Security testing tests to find any potential flaws, vulnerabilities and weaknesses in a software system to ensure that it is free from any threats or risks.
- Stress Testing
    - Stress testing tests a software system beyond normal operational capacity to test the results.

- System Testing
    - System testing tests the system as a whole. All modules and components are tested to see whether the system works as expected or not.
- Unit Testing
    - Unit testing functions by checking small pieces of code to ensure that the individual parts of a program work properly on their own thus speeding up testing strategies and reducing unnecessary tests.
- User Input Testing
    - User Input testing test to see if the input function of an application is working properly given a specific test case.

3 Assumptions and Dependencies/Constraints

Each test case lists its own assumptions and dependencies/constraints. However, several general assumptions and dependencies/constraints are listed here to avoid redundancy:

Assumptions:

- The user's device has a display
- The user's device supports input
- The user's device has an internet connection
- The application is running in the foreground

Dependencies/Constraints:

- Functional Web Browser that supports PWAs

4 Test Requirements

4.1 Create User Account

- **User Story:**
    - As a new user, I want to create an account so that I can use the service
- **Functional Requirement:**
    - The application must allow users to create new accounts by adding an email, username, and password
- **Non-Functional Requirement:**
    - The app will uphold certain security standards upon creation of account so that no user information will be breached.
- **Assumptions:**
    - User authentication on the database is already set up
    - Keyboard input is functional
- **Dependencies/Constraints:**
    - User Account Database
- **Testing Methods:**
    - **Functional Testing**
        - Create Account
            - Success Case:
                - The account information is valid and does not exist in the database

                - Failure Case:
                    - The account information already exists and should not be created
                - Execution Time:
                    - 10s
    - **Integration Testing**
        - Connect to Database
            - Success Case:
                - When creating an account, the user's information is sent and stored in the database
            - Failure Case:
                - The user's information does not get stored in the database
            - Execution Time:
                - 5s
    - **Security Testing**
        - Secure Password Verification
            - Success Case:
                - The user's password is a minimum of 8 characters and contains a combination of letters, numbers, and special characters
            - Failure Case:
                - The user's password is accepted when it's too weak
            - Execution Time:
                - 100ms
    - **User Input Testing**
        - Valid Inputs
            - Success Cases:
                - johnsmith@gmail.com is a valid email address
                - jamesbond007@outlook.com is a valid email address
            - Failure Cases:
                - notavalidemail$yahoo.com is an invalid email address
                - invalid.com is an invalid email address
            - Execution Time:
                - 100ms
    - **Non-Functional Testing**
        - Secure Account Creation
            - Success Case:
                - The user's account information is only accessible within the application
            - Failure Case:
                - The user's account information is accessible to external applications or services
            - Execution Time:
                - 500ms
- **Total Execution Time:**

        - 15.7s

4.2 Login Feature

- **User Story:**
    - I am not a registered user, but I would like to access the limited features as a guest user.
    - I have signed out of my account and I need to log back in.
    - I forgot my username and/or password and I would like to retrieve my account information.
- **Functional Requirement:**
    - This feature must allow the user to either sign in as a user or a guest user.
    - Guest users will have limited accessibility to the application compared to a registered account user.
    - Account users can access their account if their username/email and passwords match the one stored in the database.
    - If the user forgets their account information, they can retrieve it through email from the system if their email matches the one in the database.
- **Non-Functional Requirement:**
    - The app will save the login information of the user if they choose to save on the app.
- **Assumptions:**
    - User authentication on the database is already set up
    - Keyboard input is functional
- **Dependencies/Constraints:**
    - User Account Database
- **Testing Methods:**
    - **Functional Testing**
        - Logging into Account
            - Success Case
                - You have successfully logged in
            - Failure Case
                - Username/Email does not match the one in the user database.
                - Incorrect password.
            - Execution Time:
                - 1s
        - Retrieving Account Information
            - Success Case
                - Email is sent to the registered account email.
            - Failure Case
                - Email does not match the one in the user database.
            - Execution Time:
                - 5s
    - **Non-Functional Testing**
        - Saving Account Information
            - Success Case

                - After opting into it, the user does not have to login again after closing the app
            - Failure Case
                - The user must login again after every session
            - Execution Time:
                - 10s
- **Total Execution Time:**
    - 16s

4.3 Scan Barcode

- **User Story:**
    - I am curious about what is in this product and to find out, I need to scan the barcode
- **Functional Requirement:**
    - The application must allow for the scanning of barcodes
- **Non-Functional Requirement:**
    - The app will instruct the user on how they should adjust their camera so that the barcode can become recognized by the camera.
- **Non-Functional Requirement:**
    - Once a product has been scanned, the app will identify the contents of the product and make recommendations based on similar nutritional content.
- **Assumptions:**
    - Device has a functional camera
- **Dependencies/Constraints:**
    - Camera Accessibility
    - Barcode Lookup
- **Testing Methods**:
    - **Compatibility Testing**
        - Camera Opens on Multiple Platforms
            - Success Case:
                - The camera natively opens on iOS devices and Android devices, and uses a webcam on PC
            - Failure Case:
                - The camera does not work
            - Execution Time:
                - 1m
    - **Functional Testing**
        - Barcode Scanning
            - Success Case:
                - The barcode is captured using the camera and used to identify the product
            - Failure Case:
                - The barcode is not captured
            - Execution Time:
                - 10s
    - **Non-Functional Testing**

            - Camera Adjustment
                - Success Case:
                    - The app instructs the user to move the camera closer or further from the barcode
                - Failure Case:
                    - The app does not instruct the user when using the camera
                - Execution Time:
                    - 5s
- **Total Execution Time:**
    - 1m 15s

4.4 Manually Enter Product

- **User Story:**
    - The barcode scanner isn't working for some reason, but I still want to look this product up.
- **Functional Requirement:**
    - The application must allow users to enter the barcode manually
- **Non-Functional Requirement:**
    - Allow the user to upload a previously taken photo from their device to act as their ‘scan’.
- **Assumptions:**
    - Photo library is accessible
    - Keyboard input is functional
- **Dependencies/Constraints:**
    - Barcode Lookup
- **Testing Methods:**
    - **Functional Testing**
        - Barcode Lookup
            - Success Case:
                - Product is able to be found by entering a barcode
            - Failure Case:
                - Barcode is unable to be entered manually
            - Execution Time:
                - 10s
    - **Non-Functional Testing**
        - Photo Barcode Lookup
            - Success Case:
                - An imported photo is used to input the barcode
            - Failure Case:
                - The app does not allow a photo to be used to find the barcode
            - Execution Time:
                - 5s
    - **User Input Testing**
        - Valid Inputs

                - Success Cases:
                    - 845502021128
                    - 073141153980
                - Failure Cases:
                    - A83920
                    - 0123456789
                - Execution Time:
                    - 100ms
- **Total Execution Time:**
    - 15.1s

4.5 Define Chemical Ingredients

- **User Story:**
    - This product has a lot of long names in its ingredient list and I want to know what they are
- **Functional Requirement:**
    - Application must simplify chemical names in order to better understand terms
- **Non-Functional Requirement:**
    - All ingredients, nutrients, and vitamins will be categorized by their appropriate classification in alphabetical order.
- **Assumptions:**
    - Chemical ingredients are listed along with the product
- **Dependencies/Constraints:**
    - Chemical Ingredients Database
- **Testing Methods:**
    - **Functional Testing**
        - Chemical Ingredient Definitions
            - Success Case:
                - Calcium phosphate monobasic is defined as a “material of relatively high purity, as required for baking, is produced by treating calcium hydroxide with phosphoric acid”
            - Failure Case:
                - A definition is not found for calcium phosphate monobasic
            - Execution Time:
                - 500ms
    - **Non-Functional Testing**
        - Categorizing Ingredients
            - Success Case:
                - Calcium phosphate monobasic and ammonium bicarbonate are classified as acidity regulators
            - Failure Case:
                - Calcium phosphate monobasic and ammonium bicarbonate cannot be classified

                - Execution Time:
                    - 100ms
- **Total Execution Time:**
    - 600ms

4.6 Articles Based on Currently Scanned Product

- **User Story:**
    - I wish to visit the manufacturers website of the product I just scanned
- **Functional Requirement:**
    - Application must provide users a link for more information on the product they scanned
- **Non-Functional Requirement:**
    - Present the article based on the scanned product so that it fits on the user's browsers in a top-down scrollable format.
- **Assumptions:**
    - The product's manufacturer has a website containing information about the product
- **Dependencies/Constraints:**
    - Web Scraping
- **Testing Methods:**
    - **Functional Testing**
        - Health Articles
            - Success Case:
                - After scanning the barcode for a box of oreos, an article is shown that outlines the potential risks of the product
            - Failure Case:
                - No article is presented for the product
            - Execution Time:
                - 5s
    - **Non-Functional Testing**
        - Scrollable Article
            - Success Case:
                - The article is scaled to the device screen and can be scrolled through
            - Failure Case:
                - The article is not scaled nor scrollable
            - Execution Time:
                - 1s
- **Total Execution Time:**
    - 6s

4.7 SQL Database Access

- **User Story:**
    - I wish to see information on the product I just scanned
- **Functional Requirement:**
    - The database must be able to find and return information on the scanned product

- **Non-Functional Requirement:**
    - Results and information pulled from the database will be processed and shown to the user within 1.5 seconds.
- **Assumptions:**
    - The database already contains product information
- **Dependencies/Constraints:**
    - SQL Database
- **Testing Methods:**
    - **Functional Testing**
        - Accessible Product Information
            - Success Case:
                - The scanned product is accessible through the database
            - Failure Case:
                - The scanned product is inaccessible through the database
            - Execution Time:
                - 500ms
    - **Non-Functional Testing**
        - Fast Performance
            - Success Case:
                - The product is found and returned within 1.5 seconds
            - Failure Case:
                - The product is found and returned after 1.5 seconds
            - Execution Time:
                - 1.5s
    - **Performance Testing**
        - Rapid Performance
            - Success Case:
                - The database can be rapidly queried to return results quickly
            - Failure Case:
                - The database return results too slowly
            - Execution Time:
                - 1.5s
    - **Integration Testing**
        - Connect to Database
            - Success Case:
                - The database is accessible within the application
            - Failure Case:
                - The database is inaccessible within the application
            - Execution Time:
                - 500ms
- **Total Execution Time:**
    - 4s

4.8 Food Alternatives

- **User Story:**
  - I want to see similar products to what i'm looking at, but with lower calories
- **Functional Requirement:**
  - The application must be able to provide alternatives to the currently scanned food, with modified search parameters
- **Non-Functional Requirement:**
  - Provide visual representations of products recommended to users.
- **Assumptions:**
  - The product is identified using the camera
- **Dependencies/Constraints:**
  - Product Database
- **Testing Methods:**
  - **Functional Testing**
    - Alternative Cereals
      - Success Case:
        - After scanning a cereal box of Lucky Charms, the application provides an alternative healthier product like Cheerios
      - Failure Case:
        - An alternative product is never recommended
      - Execution Time:
        - 5s
  - **Non-Functional Testing**
    - Visual Representation
      - Success Case:
        - A picture of a box of Cheerios is shown to the user
      - Failure Case:
        - A picture of a box of Cheerios is not shown to the user
      - Execution Time:
        - 500ms
- **Total Execution Time:**
  - 5.5s

4.9 Food Flags

- **User Story:**
  - I want to avoid high fructose corn syrup in my products
- **Functional Requirement:**
  - The application must accept a list of flagged items, and alert the consumer when a product contains a flagged item
- **Non-Functional Requirement:**
  - Add a visual indicator by adding a checkmark on whatever the user has flagged.
- **Assumptions:**
  - Products contain a list of chemical ingredients

- **Dependencies/Constraints:**
    - Product Database
- **Testing Methods:**
    - **Functional Testing**
        - Flagging High Fructose Corn Syrup
            - Success Case:
                - If a product contains high fructose corn syrup, the user will be alerted
            - Failure Case:
                - The user is not alerted when a product contains high fructose corn syrup
            - Execution Time:
                - 500ms
    - **Non-Functional Testing**
        - Visual Indicator
            - Success Case:
                - The product containing high fructose corn syrup is given a checkmark to alert the user
            - Failure Case:
                - The product is not flagged with a checkmark
            - Execution Time:
                - 100ms
- **Total Execution Time:**
    - 600ms

4.10 Food Filter

- **User Story:**
    - I'm looking for new foods that are high in protein
- **Functional Requirement:**
    - Application must accept nutrients and provide the user options containing those
- **Non-Functional Requirement:**
    - Users will be able to select from a list of nutrients, vitamins, or ingredients listed in a top-down view listed in alphabetical order. Users will also be able to select or deselect every item listed through an optional button.
- **Assumptions:**
    - Products contain a list of chemical ingredients
- **Dependencies/Constraints:**
    - Product Database
- **Testing Methods:**
    - **Functional Testing**
        - Filtering Food Lists by Protein
            - Success Case:
                - Only food products that are high in protein remain in the list
            - Failure Case:

            - All food products remain in the list
        - Execution Time:
            - 500ms
    - **Non-Functional Testing**
        - Common Food Filters
            - Success Case:
                - A list of common food filters are shown to filter the current food list
            - Failure Case:
                - No recommended food filters are shown
            - Execution Time:
                - 100ms
- **Total Execution Time:**
    - 600ms

4.11 Personalized Daily Percentage Values (AMR)

- **User Story:**
    - I want an easier way to know how much of my daily nutrition is being taken up by a given product
- **Requirement:**
    - Application must calculate users AMR and store it. This value is used to modify nutrition labels' daily percentages.
- **Non-Functional Requirement:**
    - Calories will be displayed numerically alongside the recommended calorie consumption in a bar graph format alongside the 5 most recent items searched or scanned, listed and presented as a bar graph as well. The user's Active Metabolic Rate will be displayed numerically.
- **Assumptions:**
    - User's profile is already created with height, weight, and age
- **Dependencies/Constraints:**
    - AMR Calculator
- **Testing Methods:**
    - **Functional Testing**
        - Daily Percentage Value
            - Success Case:
                - The daily percentages on the product are adjusted to match the actual daily percentages for the user
            - Failure Case:
                - The daily percentages are not adjusted according to the user
            - Execution Time:
                - 500ms
    - **Non-Functional Testing**
        - Recommended Calorie Consumption
            - Success Case:
                - The calorie count for the product is shown relative to

the recommended daily calorie consumption for the user
                - Failure Case:
                    - The calorie count is not shown alongside the user's recommended calorie consumption
                - Execution Time:
                    - 500ms
- **Total Execution Time:**
    - 1s

4.12 Calorie Frame of Reference

- **User Story:**
    - I have a hard time visualizing how many calories a product has and wish there was a better way to understand it.
- **Functional Requirement:**
    - The application must plot products next to each other with their calories by serving to act as a visual aid.
- **Non-Functional Requirement:**
    - Add a bar graph that visually describes the product's calorie frame of reference when compared to other products.
- **Assumptions:**
    - Product lists the amount of calories it contains
- **Dependencies/Constraints:**
    - Graphing Functions
- **Testing Methods:**
    - **Functional Testing**
        - Calorie Comparison
            - Success Case:
                - For each product in a list, the product's calorie count is plotted next to the calorie count of the other products in the list
            - Failure Case:
                - The calorie count for the products are not plotted
            - Execution Time:
                - 1s
    - **Non-Functional Testing**
        - Bar Graph
            - Success Case:
                - Each product's calorie count is shown as a bar graph
            - Failure Case:
                - Each product's calorie count is not graphed
            - Execution Time:
                - 500ms
- **Total Execution Time:**
    - 1.5s

4.13 Average Product Rating

- **User Story:**
  - I want to see the average rating a user gives this product
- **Functional Requirement:**
  - Application must provide an average rating gathered from other websites
- **Non Functional Requirement:**
  - Show the user the rating on products visually represented by a star on screen.
- **Assumptions:**
  - Ratings are pulled from external websites
- **Dependencies/Constraints:**
  - Web Scraping
- **Testing Methods:**
  - **Functional Testing**
    - Product Rating
      - Success Case:
        - The product rating is calculated by averaging ratings gathered from external sources
      - Failure Case:
        - The product rating is not calculated
      - Execution Time:
        - 10s
  - **Non-Functional Testing**
    - Star Rating
      - Success Case:
        - The average rating is shown via filled or partially filled stars
      - Failure Case:
        - The average rating is not shown via stars
      - Execution Time:
        - 100ms
  - **Integration Testing**
    - Connecting to External Sources
      - Success Case:
        - The product is found on a website and stores the given rating to be averaged later
      - Failure Case:
        - The website cannot be accessed
      - Execution Time:
        - 10s
- **Total Execution Time:**
  - 20.1s

4.14 In-House Product Reviews

- **User Story:**
  - I wish to leave a review for this product

- **Functional Requirement:**
    - Users will be able to click on a scale of 1 to 5 stars to rate the app. They will also be provided the option to comment below these stars. Once their review has been made, users can submit the review by clicking the appropriate button.
- **Non-Functional Requirement:**
    - Add a chat filter algorithm that will parse a review for any flagged words or language and hide those reviews from appearing to users and their star rating will be discarded.
- **Assumptions:**
    - Product can be rated with stars
    - Keyboard input is functional
- **Dependencies/Constraints:**
    - Product Review Database
- **Testing Methods:**
    - **Functional Testing**
        - Internal Reviews
            - Success Case:
                - A review can be submitted alongside a star rating for the product
            - Failure Case:
                - The review cannot be submitted
            - Execution Time:
                - 2.5s
    - **Non-Functional Testing**
        - Explicit Filtering
            - Success Case:
                - Any reviews that contain explicit, inappropriate, or irrelevant words will be removed
            - Failure Case:
                - Explicit reviews are not discarded
            - Execution Time:
                - 100ms
    - **User Input Testing**
        - Character Limit
            - Success Case:
                - Character input is within a character limit of 300
            - Failure Case:
                - Character input exceeds character limit of 300
            - Execution Time:
                - 100ms
- **Total Execution Time:**
    - 2.7s

4.15 Feedback on Application Recommendations

- **User Story:**

        - I keep getting bad recommendations for products, and want to let the app makers know this isn't a good fit
- **Functional Requirement:**
    - The app must allow for users to rate how accurate recommendations are
- **Non-Functional Requirement:**
    - Ensure that the application will not prompt the user during awkward times or in ways that will negatively impact their experience.
- **Assumptions:**
    - The application recommends products to users
- **Dependencies/Constraints:**
    - Recommendation Algorithm
- **Testing Methods:**
    - **Functional Testing**
        - Recommendation Rating
            - Success Case:
                - The user can rate the accuracy of their product recommendations through a star rating system
            - Failure Case:
                - The user is unable to provide feedback on their product recommendations
            - Execution Time:
                - 2.5s
    - **Non-Functional Testing**
        - Recommendations of Related Products
            - Success Case:
                - After scanning a new product, a list of related products will be recommended.
            - Failure Case:
                - The user is prompted while scanning a new product with the camera
            - Execution Time:
                - 5s
- **Total Execution Time:**
    - 7.5s

4.16 Aggregated News

- **User Story:**
    - I want to see the latest news on health trends
- **Functional Requirement:**
    - Application must pull news headlines and link the user to the full article
- **Non-Functional Requirement:**
    - News will be presented to users from a top-down view with the top being the most recent news to be posted. News will be listed by their article title alongside the authors. Clicking on the article title will redirect the user to the website.
- **Assumptions:**

        - News articles are pulled from external websites
- **Dependencies/Constraints:**
    - Web Scraping
- **Testing Methods:**
    - **Functional Testing**
        - Aggregated News
            - Success Case:
                - News stories are provided from external sources and shown to the user collectively
            - Failure Case:
                - News stories are not shown to the user
            - Execution Time:
                - 5s
    - **Non-Functional Testing**
        - News Feed
            - Success Case:
                - An organized news feed is curated with links to the original articles
            - Failure Case:
                - News articles are shown randomly and without order
            - Execution Time:
                - 1s
    - **Integration Testing**
        - Connecting to External Sources
            - Success Case:
                - News headlines are pulled from multiple websites with links to the article
            - Failure Case:
                - External news headlines cannot be integrated into the news feed
            - Execution Time:
                - 10s
- **Total Execution Time:**
    - 16s

4.17 Personalized News

- **User Story:**
    - The news articles aren't matching my personal interests and I want them to be more targeted towards my needs
- **Functional Requirement:**
    - Application must have personalized news feed based on user data
- **Non-Functional Requirement:**
    - Allows users to flag certain news they are not interested in so that the news presented to the user can be adjusted to their preference accordingly.
- **Assumptions:**

        - News articles are pulled from external websites
- **Dependencies/Constraints:**
    - Personalized News Algorithm
- **Testing Methods:**
    - **Functional Testing**
        - Like/Dislike News
            - Success Case:
                - The user can like or dislike news articles to better adjust the news feed to their interests
            - Failure Case:
                - The user cannot provide feedback on news articles
            - Execution Time:
                - 500ms
    - **Non-Functional Testing**
        - Not Interested
            - Success Case:
                - The user can mark news items as not interested to remove similar news items from their feed
            - Failure Case:
                - The user cannot mark news items as not interested
            - Execution Time:
                - 500ms
- **Total Execution Time:**
    - 1s

4.18 User Data Collection

- **User Story:**
    - My data is important to me and I don't want it being collected
- **Functional Requirement:**
    - Application must allow users to opt out of all data collection
- **Non-Functional Requirement:**
    - Users will be able to switch on screen whether they would allow for their data to be shared.
- **Assumptions:**
    - User data is already being collected
- **Dependencies/Constraints:**
    - User Data Database
- **Testing Methods:**
    - **Functional Testing**
        - Opt Out
            - Success Case:
                - The user can opt out of user data collection by the application
            - Failure Case:
                - The user cannot opt out of user data collection
            - Execution Time:

                - 500ms
        - **Non-Functional Testing**
            - Opt Out Toggle
                - Success Case:
                    - In the settings, the user can view and toggle their data collection preferences
                - Failure Case:
                    - The toggle cannot be viewed or toggled
                - Execution Time:
                    - 100ms
- **Total Execution Time:**
    - 600ms

4.19 History

- **User Story:**
    - I scanned a product earlier and want to look at the details of it again
- **Requirement:**
    - Application must provide access to user history
- **Non-Functional Requirement:**
    - Users will be able to access the last 25 items they have scanned through a top-down scrolling window. The items will be organized from the most recent scanned product starting from the top. The items will be listed by their name**.**
- **Assumptions:**
    - Scanned products have been saved
- **Dependencies/Constraints:**
    - Product Database
- **Testing Methods:**
    - **Functional Testing**
        - Product History
            - Success Case:
                - The user can view previously scanned products in a list
            - Failure Case:
                - The scanned products are not saved by the application
            - Execution Time:
                - 1s
    - **Non-Functional Testing**
        - Organized History
            - Success Case:
                - The product history is sorted by most recently scanned products and lists up to 25 items
            - Failure Case:
                - The product history is sorted randomly and includes too many or too few items

                - Execution Time:
                    - 500ms
- **Total Execution Time:**
    - 1.5s

4.20 Main Menu

- **User Story:**
    - As a user I need a way to navigate through the various functions this product provides
- **Requirement:**
    - Application must have a menu for users to be able to access all elements of our application
- **Non-Functional Requirement:**
    - Main menu UI must be intuitive, visually pleasing, and functional.
- **Assumptions:**
    - The user interface is interactive
- **Dependencies/Constraints:**
    - Navigation Menu
- **Testing Methods:**
    - **Functional Testing**
        - Navigation Menu
            - Success Case:
                - The user can navigate through all parts of the application via a navigation menu
            - Failure Case:
                - The navigation menu does not navigate the user correctly
            - Execution Time:
                - 5s
    - **Non-Functional Testing**
        - Menu Aesthetics
            - Success Case:
                - The menu UI is simple, easy to use, and practical
            - Failure Case:
                - The menu UI is difficult to understand or use properly
            - Execution Time:
                - 100ms
    - **GUI Testing**
        - Menu GUI
            - Success Case:
                - The menu GUI scales properly to devices of varying screen sizes and operating systems
            - Failure Case:
                - The menu GUI does not scale or adapt to different devices
            - Execution Time:

    - 10s
- **Total Execution Time:**
    - 15.1s

5 Total Testing Time

Total Execution Time:

- *4.1 Create User Account*
  - 15.7s
- *4.2 Login Feature*
  - 16s
- *4.3 Scan Barcode*
  - 1m 15s
- *4.4 Manually Enter Product*
  - 15.1s
- *4.5 Define Chemical Ingredients*
  - 600ms
- *4.6 Articles Based on Currently Scanned Product*
  - 6s
- *4.7 SQL Database Access*
  - 4s
- *4.8 Food Alternatives*
  - 5.5s
- *4.9 Food Flags*
  - 600ms
- *4.10 Food Filter*
  - 600ms
- *4.11 Personalized Daily Percentage Values (AMR)*
  - 1s
- *4.12 Calorie Frame of Reference*
  - 1.5s
- *4.13 Average Product Rating*
  - 20.1s
- *4.14 In-House Product Reviews*
  - 2.7s
- *4.15 Feedback on Application Recommendations*
  - 7.5s
- *4.16 Aggregated News*
  - 16s
- *4.17 Personalized News*
  - 1s
- *4.18 User Data Collection*
  - 600ms
- *4.19 History*

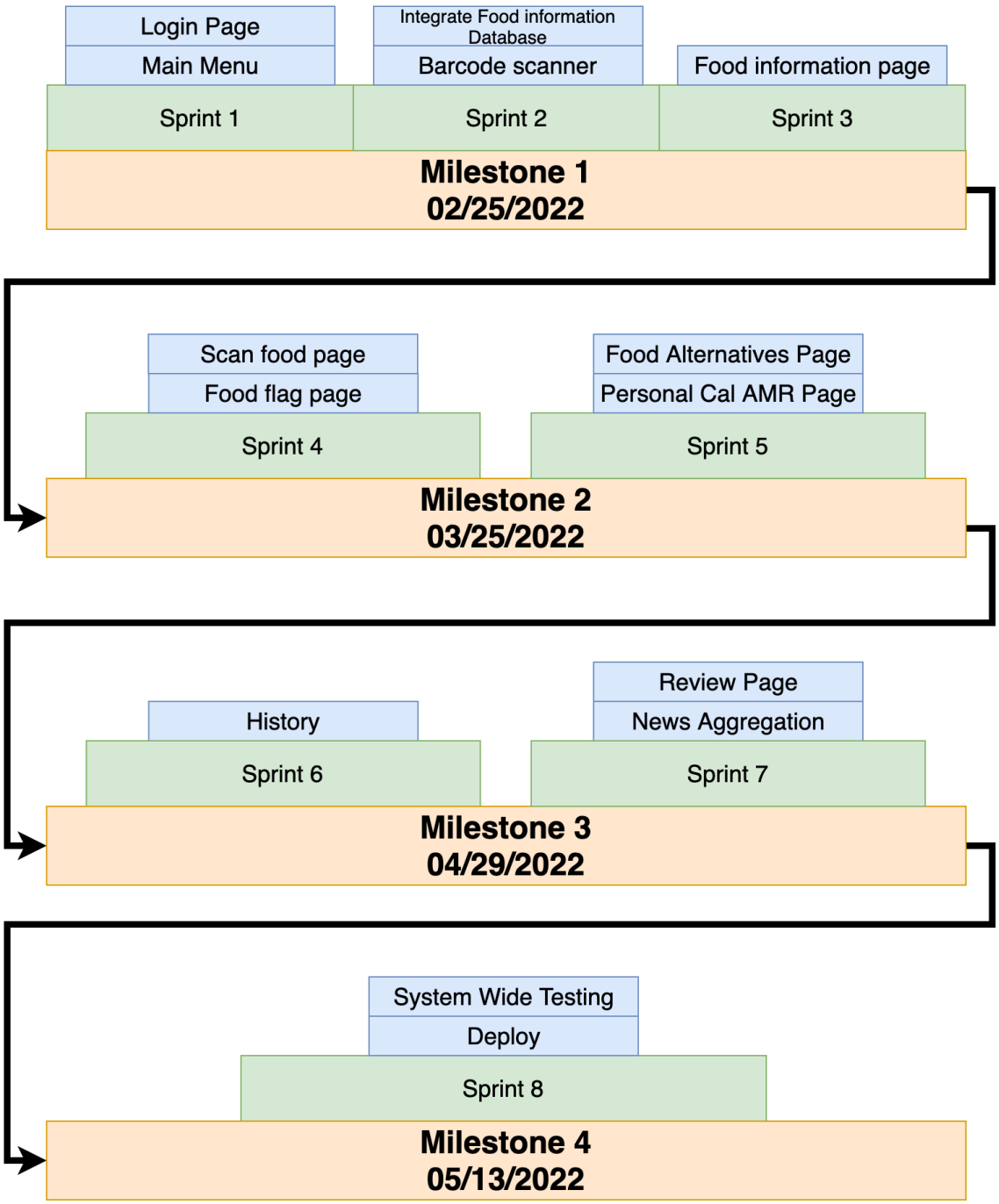
    - 1.5s
- *4.20 Main Menu*
    - 15.1s

Total Testing Time for System:

- 3m 26s 100ms

6 Timeline

The timeline for the testing plan is directly correlated to the sprint cycles during the software development process. As new features are developed, they are simultaneously tested according to the testing plan. Additionally, the final sprint has been dedicated to the deployment and final testing of the project. A roadmap for the development of the software features has been outlined below:



Each test requirement has been assigned to a sprint cycle with dates as follows:

- **Sprint 1**: 2/25/22 - 3/4/22
    - 4.1 | 2/25/22 - 2/28/22
    - 4.2 | 2/28/22 - 3/2/22
    - 4.20 | 3/2/22 - 3/4/22
- **Sprint 2**: 3/4/22 - 3/11/22
    - 4.3 | 3/4/22 - 3/7/22
    - 4.4 | 3/7/22 - 3/9/22
    - 4.7 | 3/9/22 - 3/11/22
- **Sprint 3**: 3/11/22 - 3/18/22
    - 4.5 | 3/11/22 - 3/14/22
    - 4.6 | 3/14/22 - 3/18/22
- **Sprint 4**: 4/4/22 - 4/11/22
    - 4.9 | 4/4/22 - 4/7/22
    - 4.10 | 4/7/22 - 4/11/22
- **Sprint 5**: 4/11/22 - 4/18/22
    - 4.8 | 4/11/22 - 4/14/22
    - 4.11 | 4/14/22 - 4/16/22
    - 4.12 | 4/16/22 - 4/18/22
- **Sprint 6**: 4/29/22 - 5/6/22
    - 4.18 | 4/29/22 - 5/3/22
    - 4.19 | 5/3/22 - 5/6/22
- **Sprint 7**: 5/6/22 - 5/13/22
    - 4.13 | 5/6/22 - 5/9/22
    - 4.14 | 5/9/22 - 5/10/22
    - 4.15 | 5/10/22 - 5/11/22
    - 4.16 | 5/11/22 - 5/12/22
    - 4.17 | 5/12/22 - 5/13/22

7 Automated Testing

Writing automated tests is a key component to the successful testing of any project. Although most of our software testing will be done manually, automated testing will be applied wherever it can be used in this project. To accomplish this, several testing tools will be implemented in the project.

- React Native Testing Library
    - Used to write maintainable tests for individual React Native components.
- Jest
    - Used to run tests and ensure correctness for JavaScript codebases.
- Selenium

    - Used to perform web application testing across a variety of devices and operating systems.
- Firebase Test Lab
    - Used to run integration testing between the core database and the application.