MEALIFY: AN IOS MEAL PLANNING AND RECIPE SEARCH APP

by

JOSEPH TO

A THESIS

Presented to the Department of Computer Science
and the Robert D. Clark Honors College
in partial fulfillment of the requirements for the degree of
Bachelor of Science

May 2024

An Abstract of the Thesis of

Joseph To for the degree of Bachelor of Science
in the Department of Computer Science to be taken June 2024

Title: Mealify: An iOS Meal Planning and Recipe Search App

Approved: *Eric Will, Ph.D.*
Primary Thesis Advisor

Mealify addresses the challenges of rising grocery prices and the need for complex recipes by providing an innovative meal-planning app experience that combines extensive recipe options with real-time pricing information from nearby stores. Designed with practicality, Mealify streamlines recipe searches, allowing users to find recipes based on their dietary preferences and budget. This paper goes through the process of developing an iOS app, including the initial ideation, technologies used, and the front and backend logic of each view throughout the app. After the first iteration of Mealify, a case study was conducted with other similar apps on the market to compare the performance in supporting a specific use case. Mealify showed strengths in recipe searching capabilities by providing the ideal recipe that fits the use case constraints. The results also revealed opportunities for improving the efficiency of grocery acquisitions and cooking guidance. Ultimately, this project lays the groundwork for advancing recipe-based price searching, a crucial step towards empowering users to make more informed decisions about their meal planning and recipe selection.

Acknowledgements

First and foremost, I would like to express my deepest appreciation to my thesis committee. My primary thesis advisor, Eric Wills, has been an exceptional mentor throughout the tenure of this project. His guidance has been invaluable, and his approachability has made our interactions not only productive but also enjoyable. Eric has consistently provided insightful feedback and served as a reliable sounding board for many of my ideas. Additionally, I am grateful to my CHC representative and prospectus professor, Anita Chari, for her invaluable contributions during the brainstorming phases of this project and her guidance in bringing this all to fruition. Her expertise and support have been instrumental in shaping the trajectory of my thesis. A final thanks goes to my family, friends, and partner. I could not have asked for a better support system. Your unwavering encouragement has meant everything to me throughout this journey.

Table of Contents

List of Figures

Introduction

With the increasing popularity of varying diet plans, many individuals endure the process of finding a suitable recipe, which includes identifying the necessary ingredients and following instructions on how to cook the meal. While the market offers various meal planning and recipe apps to simplify this task, the rising cost of groceries has become a hindrance, with prices surpassing the average household income. How can we create a similar experience to existing meal planning and recipe apps in the market while considering the escalating grocery prices?

Mealify is an innovative meal-planning app designed to enhance the recipe-searching experience. With an extensive range of filtering options, users can discover recipes that align with their dietary preferences and budgetary constraints. Unlike other apps on the iOS app store, Mealify offers a vast selection of recipes from various sources across the web. To further assist users in their meal preparation journey, Mealify leverages the user's location data to provide real-time pricing information for each ingredient at nearby stores. This feature empowers users to plan their grocery shopping effectively by targeting convenience and affordability. During the cooking process, Mealify can seamlessly convert the recipe to the desired number of servings across different units of measurement. This functionality ensures users can adapt recipes to their specific needs and quantities. In addition to its practical features, Mealify boasts a visually pleasing user interface that maintains a consistent template for all recipes. This cohesive design enhances the user experience, allowing for seamless navigation and a visually appealing presentation of each recipe. With its comprehensive filtering options, real-time pricing information, conversion capabilities, and visually pleasing interface, Mealify seeks to revolutionize the meal-planning app experience.

As I brainstormed the type of app to develop, practicality was a key criterion. I aimed to create something not only to streamline my own life but also to benefit others. Weekly, I invest significant time searching for recipes that cater to my cravings and dietary requirements. Yet, even after gathering ingredients, recipe instructions often prove cumbersome due to their formalities. Mealify seeks to simplify this process for everyone.

Background

The Origin of Mobile Applications

The history of mobile apps dates to the early 1990s with the release of the first mobile phone IBM Simon. IBM Simon was the first introduction to a mobile graphical user interface (GUI). The limited computing power and lack of standardization across devices hindered the development of a thriving app ecosystem. Hereafter, there have been multiple innovations within the industry, which played an immense role in mobile app popularity today.

In the early 2000s, mobile phones using Java-based apps became popular, with companies like Nokia and BlackBerry launching app marketplaces for their devices (Koohang and Paliszkiewicz 2012). However, the fragmentation of the mobile phone market made it challenging for developers to create apps compatible with multiple devices, which led to the eventual introduction of the iOS platform in 2007 and the App Store in 2008. The App Store allowed developers to create and distribute apps for the iOS platform, allowing users to search for and download apps from a centralized location (Chaffey 2019).

The success of the App Store led to the creation of similar app stores by other companies like Google Play. Nowadays, mobile apps have become an integral part of people's lives, with over 218 billion app downloads in 2020 (Statista 2021). Mobile Apps have transformed various industries, including transportation, communication, and entertainment. With the rise of mobile technology and the increasing availability of smartphones, mobile app usage will continue to grow and shape societal practices.

In the App Store today, many available apps assist with meal planning and recipe finding. The notable apps on the market that receive excellent user feedback are Mealime Meal Plans & Recipe, Cooklist: Pantry Meal Recipes, and Intent Meal Planner & Recipes.

Competing Apps

Mealime is a meal-planning app that offers a variety of features to help users plan and prepare healthy meals (Mealime Meal Plans Inc., 2023). The app produces personalized meal plans based on the user's dietary preferences, fitness goals, and food allergies. Users can search for customized meal plans from a recipe database and create an ingredient list based on the recipe selected. Regarding the recipe searching and cooking experience, the user can search for recipes with various search terms that include for you, suggestions, cuisine, calories, price per serving, diets, and dish types. Users can even take it a step further by signing up for the pro memberships that give them access to new recipes every month, the creation of their recipes, nutrition tracking, advanced recipe filters, and a meal plan history. Each recipe provides the user with the basic recipe information, the cookware required, the ingredients, and the cooking instructions. Overall, Mealime helps users find a fitting meal plan, creates a grocery list with the option to send grocery delivery apps for orders, and has a community section where users can interact and share their recipes.

Cooklist is a meal-planning app that helps users create meal plans based on the ingredients they have already purchased (Cooklist Inc., 2023). Users add the ingredients from their pantry, which allows the app to track what the user will need to buy for the selected recipe. Based on the user pantry, the app generates suggested recipes but also leaves the option for users to find recipes on their own. The user can search for recipes by title and filter the results with a pro membership through an extensive number of options that include ingredients, time, price per serving, diets, allergens, dislikes, course and cuisine, calories, equipment and method, ingredient count, and a sort by option. Each recipe provides general information along with ingredient substitutions, reviews, nutritional facts, FAQs regarding the recipe, and related recipes. The app

also offers a shopping page for users to search for ingredients readily available at nearby grocery stores. Cooklist differs from other apps in the market by striving to reduce overall food waste and providing a more effortless grocery shopping experience.

Intent is another meal-planning app that helps users plan and prepare healthy meals (Excipient, Inc., 2023). Upon opening the app, the user answers questions regarding recipe preferences and allergens, grocery methods, and more that tailor the app to fit the desired experience. Intent offers a variety of features to help users customize their meal plans based on their dietary preferences and fitness goals. The recipes Intent offers come from a database with over 1,500 healthy recipes. For recipe searching, the user can enter a cuisine, recipe type, ingredients, or diet. With the free version of Intent, the user is limited to filtering by time length and only has access to certain recipes. Each recipe has the basic information, reviews, a dedicated note section for the user, and an edit option (assuming they have the premium version). The app tracks the user's daily calorie intake and provides nutritional information for each recipe. Users can link their accounts to nearby grocery stores to transfer over shopping lists for delivery services. Users can also receive suggestions for ingredient substitutions and connect with other Intent users through the community forum. Intent redefines meal planning by providing users with a wealth of features like personalized recipe recommendations, dietary customization, nutritional tracking, grocery list integration, and a vibrant community forum, all aimed at facilitating healthier eating habits and streamlined meal preparation.

Mealify

Mealify incorporates in place features from prominent meal planning and recipe apps while introducing its unique functionalities. First and foremost, Mealify differs from competitors in the specific target audience. Mealify does not intend to push a healthier lifestyle toward users

and will not restrain the recipes shown based on whether they are healthy. Instead, Mealify strives to influence budget-friendly recipes that are easy to follow. Mealify will emulate the feature of filtering recipes, with a few differences in filtering options. Unlike most competitors, the advanced features that Mealify offers will not be behind the membership paywall and will be free for all users. Mealify caters to users who have a clear vision of their recipe preferences and culinary objective. The first rendition of Mealify will not incorporate any user preferences that restrict the recipe responses. All the features are fully disclosed to the user to take control of their recipe searching experience.

Methodology

Scrum

In any domain of software development work, whether in the industry or a smaller-scale project, following a project management framework is optimal. Throughout the Mealify project, the plan is to adhere to the Scrum agile project management framework (Van Der Hoek 2023), with adjustments, considering this is a personal rather than a team project. Embracing Scrum facilitates the management and iterative delivery of complex projects, emphasizing adaptability and the delivery of high-value increments in short timeframes.

Prior to development, Scrum starts with creating a list of features that Mealify will incorporate. This process begins with identifying common research questions that provide a purpose and goal for this project. Putting oneself into the users' perspective to brainstorm the requirements they may demand from this app, the priority levels of each feature, and when they will get done in the development process are planned and laid out. In Scrum, this terminology denotes a backlog, encompassing all features, their completion requirements, and an estimated time frame for each feature. Creating a well-defined backlog in the Scrum framework, informed by user perspective and research, is crucial as it sets the roadmap for development, prioritizing features and establishing estimated time frames, ensuring alignment with user demands and project goals throughout the development process.

When the development process begins, Scrum specifically requires sprints, which designate periods in time for development. Since this project will coincide with other obligations, sprints offer flexible time periods to focus on software development and achieve significant progress. Each sprint aims to complete a portion of the product backlog, and at the end of the sprints, the team reviews their accomplishments. In a thesis's case, weekly meetings

with the primary thesis advisor serve as sprint overview sessions. After each sprint (weekly meetings), the advisor is presented with the accomplishments from the recent sprint, a software demonstration of the results, and goals for the next sprint. These sprints persist until the completion of the product backlog or until time allows. Adhering to the Scrum framework aligns well with Mealify's requirements, as it's a personal project guided by an expert. Maintaining a product backlog facilitates effective tracking of ongoing features and enables strategic time management and consistent progress in development endeavors.

Version Control

Version control is a crucial aspect of maintaining Mealify's code integrity. Establishing a GitHub repository at the project's outset facilitates organized tracking of modifications. Appending descriptive commit messages and pushing updates to the main branch will ensure transparency and accountability throughout development. Additionally, version control enables effortless rollback of faulty commits, enhancing the project's stability and reliability. If changes fail to meet quality standards or pass further testing, abstaining from commits ensures code consistency and prevents potential issues later.

Xcode and Swift

The integrated development environment (IDE) to develop an iOS app and any other macOS software is called Xcode. Xcode offers developers a cohesive workflow encompassing user interface design, coding, testing, and debugging. Within Xcode, Swift is the programming language developed by Apple, known for its speed, safety, and ease of use, making it ideal for building applications across various Apple platforms. Specifically for the user interface, Mealify uses SwiftUI, which is a toolkit by Apple designed to simplify and streamline the process of building user interfaces for applications across Apple platforms. Many of the apps developed by

Apple and other developers use SwiftUI, hence why many of the controls in Mealify will look like those in other applications. SwiftUI provides access to a wide array of visually appealing controls, minimizing the attention required to craft the user interface for Mealify.

Data Storage

Regarding data storage, Mealify's current version does not require a traditional relational database like SQLite3 or Firebase. The only data that needs to be stored are recent recipes, saved recipes, and the desired selected Kroger location. To store any non-relational data, like those in Mealify, Apple created a UserDefaults class that saves key-value pairings that hold simple data types such as strings, numbers, and booleans. The app stores any saved data locally on the user's device and retains it until the deletion of the app. Future iterations of Mealify may require the implementation of a database to add on features like user accounts, forums, and more. While UserDefaults efficiently handles Mealify's current requirements, the potential need for a more robust database in future iterations underscores the app's scalability and adaptability to evolving user needs.

APIs

Mealify will utilize recipes sourced from the Spoonacular Recipe API, an application programming interface (API) that grants access to an extensive database of culinary recipes and related information. Spoonacular provides various endpoints, which are specific URLs that act as a gateway to access and interact with resources or services provided by an API, defining the functionality available and the methods to manipulate data. The endpoints Mealify incorporates includes searching for recipes and obtaining recipe details. All endpoints are API GET methods, whose purpose is to retrieve information from the server. Each endpoint incorporates optional query parameters for refining searches, and Mealify will integrate most of these parameters into

its recipe search filtering functionality. The free developer version of the Spoonacular Recipe API limits Mealify to retrieving a maximum of ten recipes per API call. However, in the future, as Mealify evolves, upgrading to a higher-tier API version to access more recipes per search will be straightforward and will maintain the same logic utilized in the backend. Choosing the Spoonacular Recipe API enables Mealify to tap into plethora culinary resources, offering users extensive recipe options and customizable search functionalities.

Mealify utilizes the official Kroger API for real-time grocery pricing. Access to official grocery store APIs requires approval through an application detailing the intended usage. Since Mealify received approval only from Kroger, it exclusively utilizes this grocery API. The application interacts with two endpoints: one for finding nearby locations and the other for product search. To access these endpoints, Mealify must obtain an authorization token and pass it as a query parameter in both the location and product endpoints. The location search functionality requires a 5-digit zip code as input and returns information about nearby stores. Similarly, the product search functionality accepts a search term for products, optionally accompanied by a brand, and returns the relevant products found. By opting for the official Kroger API as its primary source for real-time grocery pricing, Mealify guarantees reliability and consistency in providing users with accurate information regarding nearby stores and product availability.

Results

Navigation Sidebar

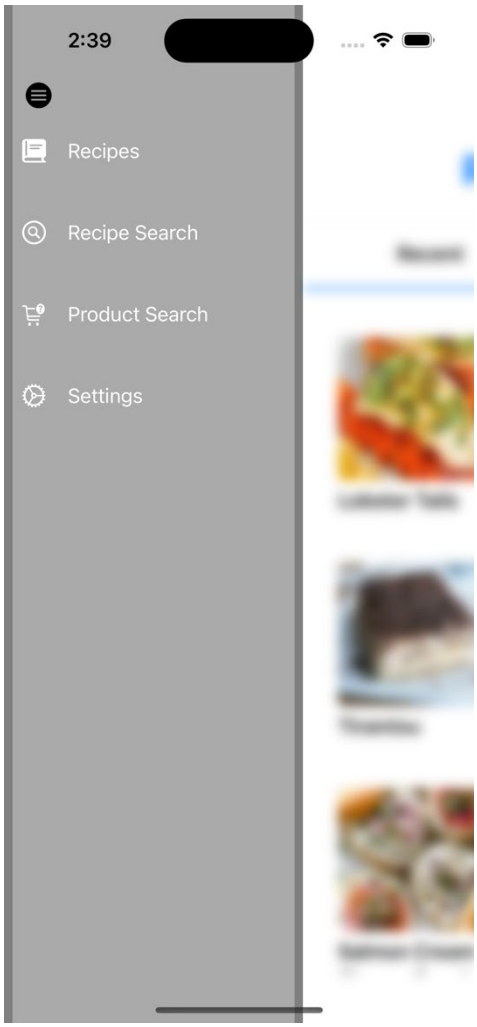

Figure 1: Navigation Sidebar User Interface

In the app's interface, the Navigation sidebar is the primary means for users to navigate between various views. It offers four distinct options: recipes, recipe search, product search, and settings. When users choose any of these options, they are taken directly to the corresponding page. The Navigation sidebar remains easily accessible from any page by tapping the blue triple bar icon positioned at the top left of each of these destinations.

On the front-end, the Navigation sidebar appears as a light gray bar that smoothly slides out from the left side of the screen. It takes up 60% of the screen's width, extends to the full length, and blurs out the parent view, ensuring a clear and focused visual of the view. Each option in the Navigation sidebar features an SF symbol paired with a text label. Upon selecting an option, the sidebar collapses, seamlessly guiding the user to their desired destination.

Meanwhile, on the backend, the Navigation sidebar's functionality is managed through multiple navigation links, controlling the user's navigation experience. This process involves setting the destination parameter to the relevant view instance, facilitating a seamless transition to the desired section of the app.

Recipes

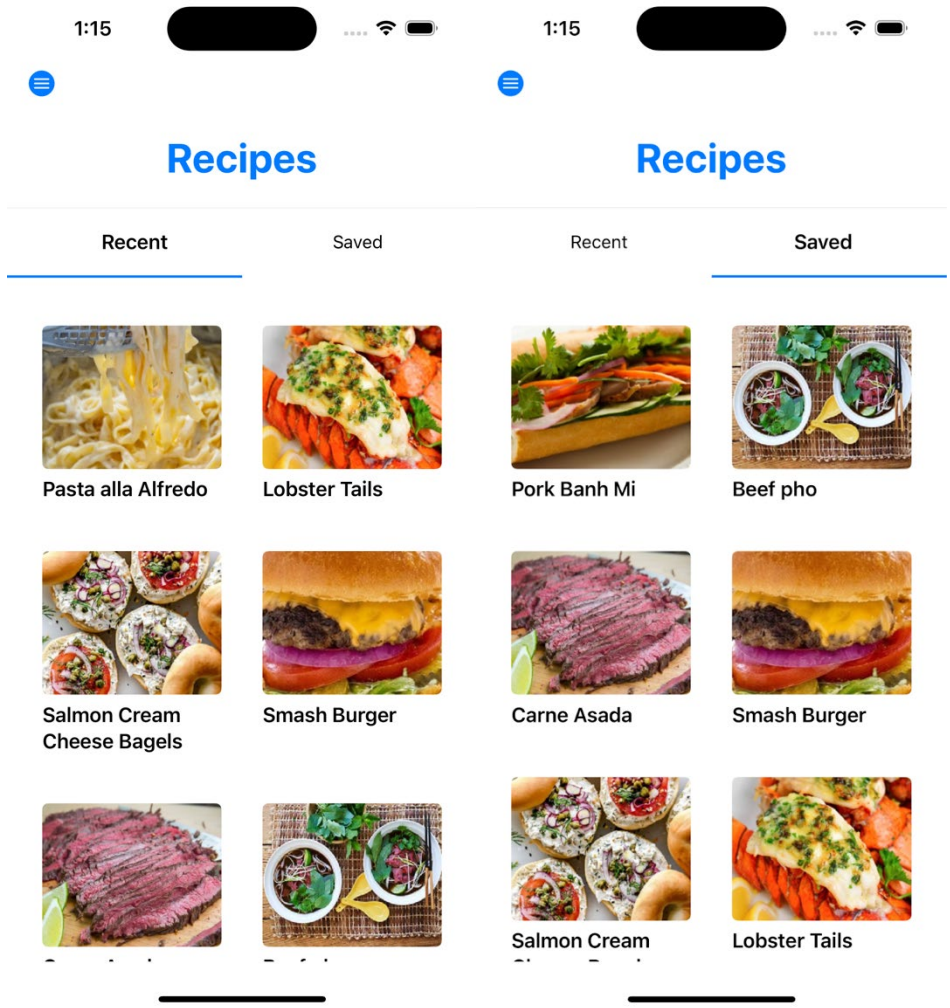

Figure 2 and 3: Recipes User Interface

These are the two tabs within the Recipes view: Recent and Saved. Figure 2 shows the recent tab. Figure 3 shows the saved tab.

The Recipes view serves as the landing screen for Mealify. Upon opening the app, Mealify presents the recent recipes tab as the default and showcases the ten most recently viewed recipes. Users can switch to the saved tab to view any recipes the user bookmarked, and from both tabs, users can select a recipe to access its details. The recent recipes tab enables users to revisit recipes they are still considering or in the process of completing. This tab proves handy if

users navigate elsewhere in the app or after app closure. Saved recipes allow users to access recipes they have previously enjoyed and wish to keep for future reference.

Regarding the user interface, the title "Recipes" appears prominently at the top in a blue bolded title font, like other view titles throughout the rest of Mealify. The tab text is bolded to differentiate between tabs and underlined with a blue bar, with the Recent tab initially highlighted in bold to signify the active tab. Clicking on a tab switches the view, accompanied by a bolded animation on the tab name and the blue bar sliding under the respective tab name. Recipes are presented in a two-column scroll, each displaying the recipe picture and title.

Recent and saved recipes are stored in JSON format to the user's defaults when users navigate into a recipe and/or save a recipe. Upon entering the Recipes view, Mealify retrieves the corresponding user's defaults data for both recent and saved recipes and decodes them from JSON into a list of recipe structs. It then iterates through each recipe list to display the contained recipes as a navigation link, each with the recipe image displayed by an async image, which when clicked, redirects the user to the respective Recipe Detail view. If there are no recipes stored in either tab, the recipe display remains empty.

Recipe Search

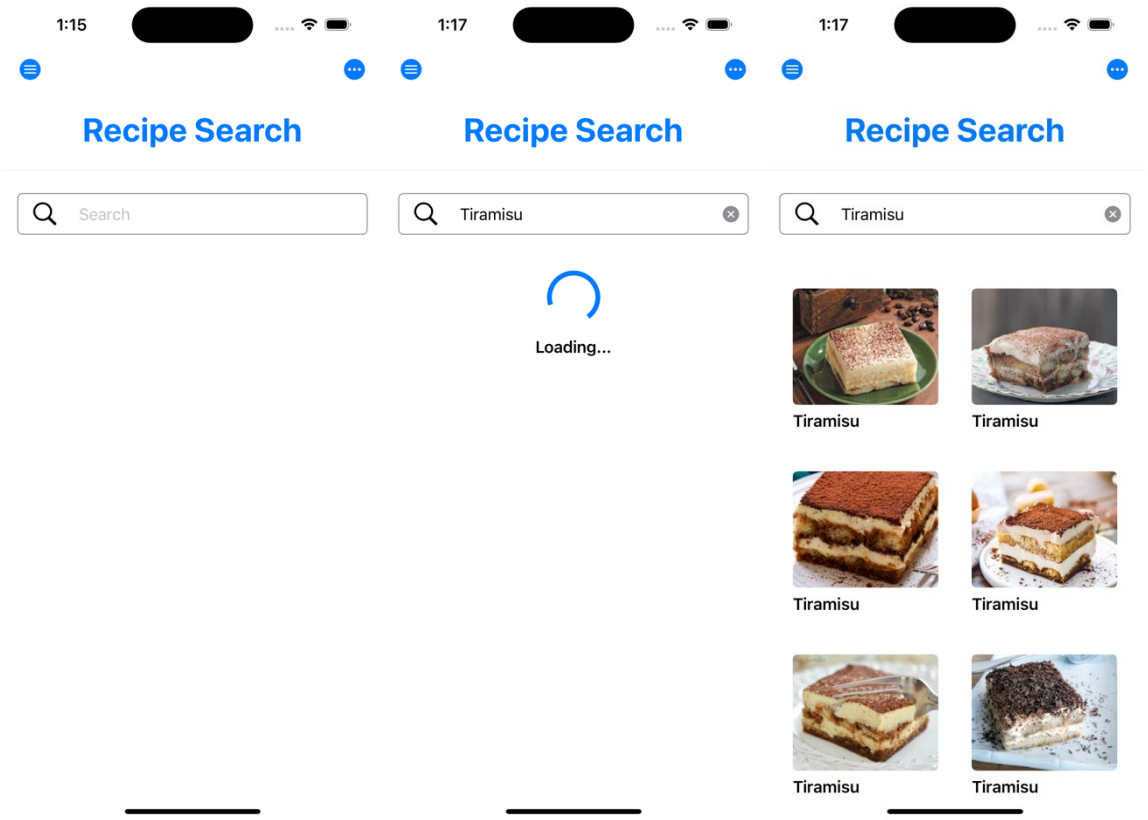

Figure 4, 5, and 6: Recipe Search User Interface

These three images follow the chronological order use of the Recipe Search view. Figure 4 is the initial view, prior to any search. Figure 5 shows the loading animation when the user prompts a search. Figure 6 shows an example result of a search.

The Recipe Search view is the centerpiece for where users can take on their recipe-searching journey. The search bar takes in recipe terms inputted by the user and returns the Spoonacular Recipe API response by displaying the desired recipes. With these results, users can browse through the list of recipes until they find the one that best suits their preferences.

On the frontend, the Recipe Search view follows a similar template by displaying a title up top, indicating the current view, the navigation sidebar icon on the top left, and a blue three-dot icon located on the top right (details for this view are in the following section). The search bar consists of a magnifying glass SF symbol and a Swift text field, all outlined by a black

rectangle with rounded corners. When clicking on the text field, the user is prompted to type in a search term and can clear the current search term by tapping on the "x" icon on the right side of the search bar. After the user commits their search term by pressing enter, an animated loading icon will appear below the search bar and then slowly transition into the resulting recipes in a two-column display, following the format of the Recipes view, users can click on the recipe card to access the corresponding recipe details.

Within the backend infrastructure, when the user enters a search term, a function activates, sending a request to the Recipe API with the specified term included in the query. The function decodes the API's JSON response into a list of recipe structs, subsequently presented in a format akin to the Recipe view. As for the clear button, its icon is visible only when the text field is populated. Upon clicking the "x" icon, the search bar text field’s text promptly clears, causing the icon to vanish.

Filters

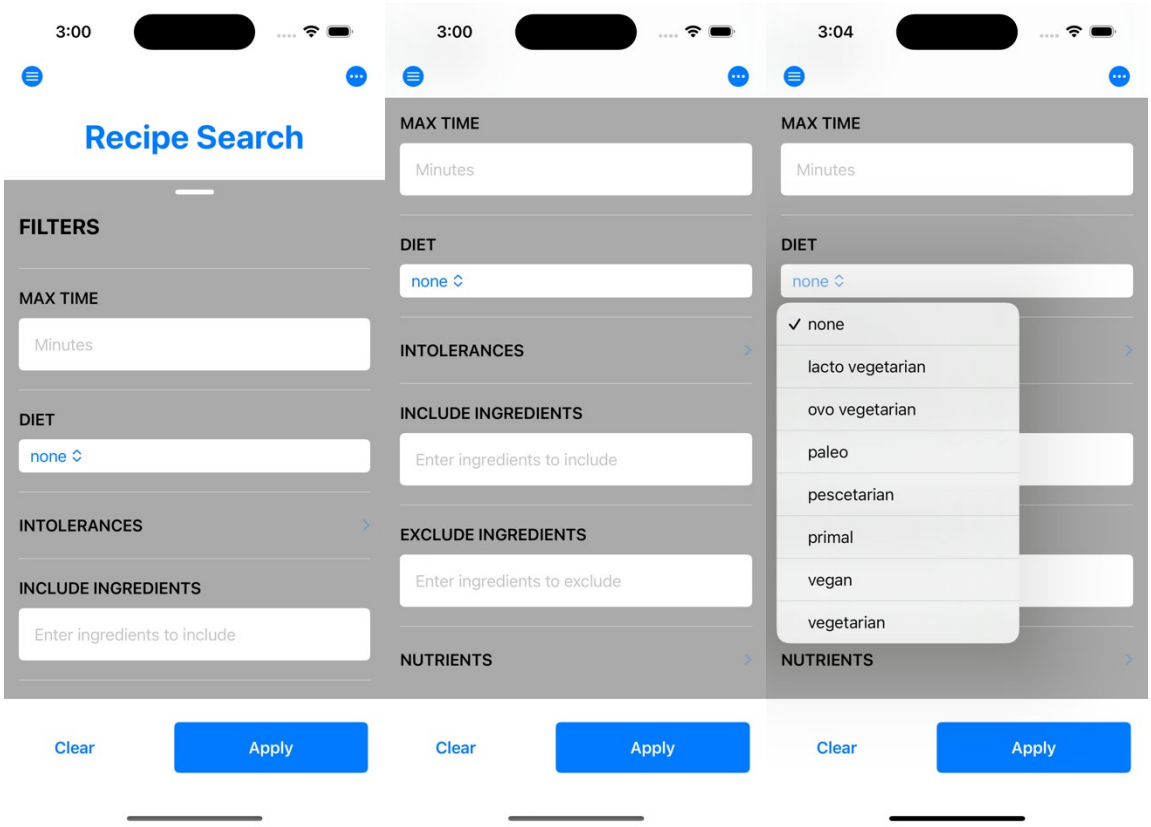

Figure 7, 8, and 9: Filter Side view User Interface Part I

Figures 7 and 8 show the initial interface of the Filter Side view, prior to any expansions. Figure 9 is the diet picker when expanded.

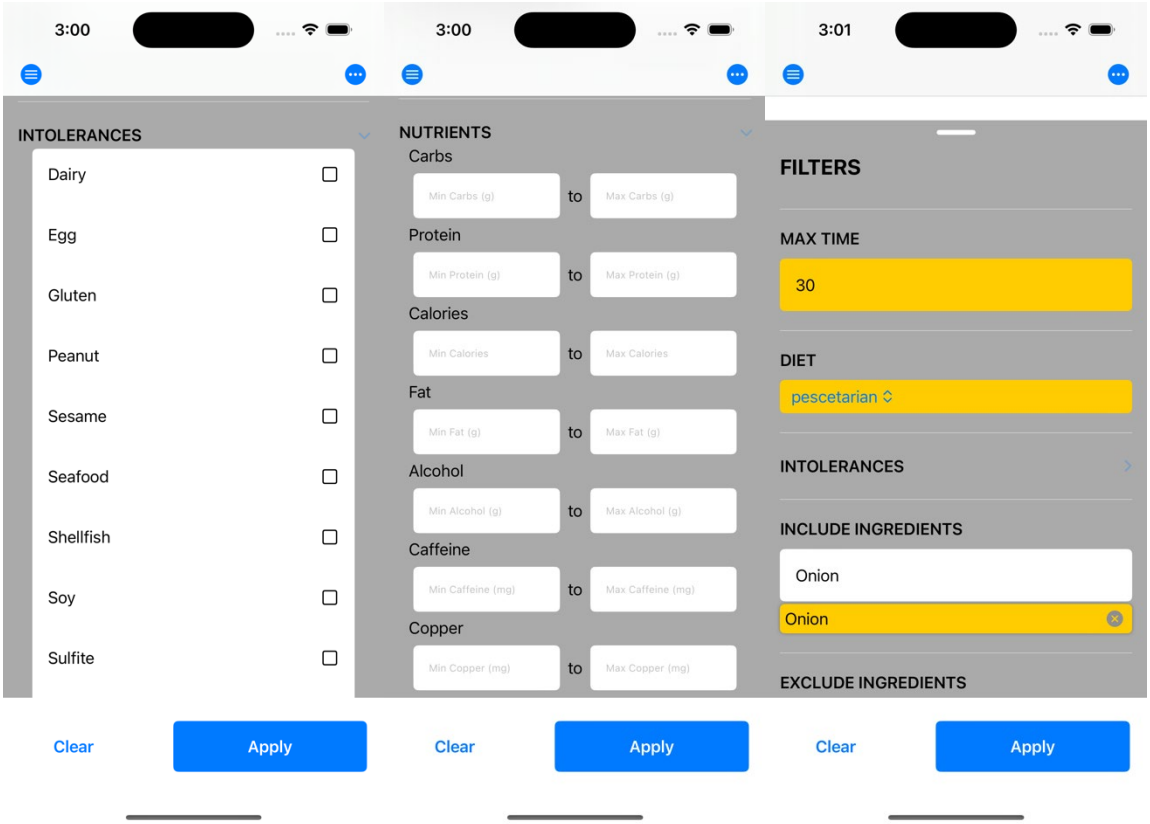

Figure 10, 11, and 12: Filter Side view User Interface Part II

Figure 10 is the dropdown view of the intolerances. Figure 11 is the start of the dropdown view for the nutrients. Figure 12 is the Filter Side view when inputted with example values.

After the user's initial entry of a search term, the Filter Side view offers a range of optional filters to refine the search results. Accessible via a tap on the blue triple-dot icon positioned at the screen's top-right corner, this feature presents a comprehensive array of search parameters. These encompass filters such as maximum cooking time, dietary preferences (including pescetarian, lacto-vegetarian, ovo-vegetarian, vegan, paleo, primal, and vegetarian), as well as intolerances (such as dairy, egg, gluten, peanut, sesame, seafood, shellfish, soy, sulfite, tree nut, and wheat). Additionally, users can specify the inclusion or exclusion of ingredients and set minimum and maximum thresholds for various nutrients (including carbohydrates, protein, calories, fat, alcohol, caffeine, copper, calcium, choline, cholesterol, fluoride, saturated fat, as well as a plethora of vitamins and minerals). Once users have input their desired filters, tapping

the "Apply" button will trigger the display of updated search results tailored to their refined query.

Upon tapping the Filter Side view icon, a light gray bar smoothly slides up from the bottom of the screen, spanning the width and taking up 60% of the height. At the top, a "FILTERS" title indicates the current view, followed by various filtering options. Each option has its own section with clear titles. The maximum cooking time feature allows users to input minutes through a text field. Dietary preferences are selected via a dropdown Swift picker, and intolerances are chosen similarly with a dropdown Swift disclosure group. Users can input inclusion and exclusion ingredients through a text field, with entered terms appearing as cards below. Nutrient filtering begins collapsed, then expands to reveal individual nutrients with paired text field inputs for setting minimum and maximum amounts. To differentiate the active and inactive filters, all active controls turn from white to yellow.

As anticipated, the Filter Side view modifies the existing search query used in the Search view. Each filtering option is stored in a designated variable, and upon tapping the apply button, the same search recipe function is invoked, with an optional parameter encompassing all the filtering variables. For each variable, if present, its URL name and value are appended to the query, and a request is sent to obtain a new refined list of recipes. Conversely, upon tapping the clear button, all filtering variables are reset, ensuring subsequent query calls commence with the baseline query. This action resets all filtering controls to their initial state in the user interface returning all their values to empty and their controls from yellow to white.

Recipe Detail

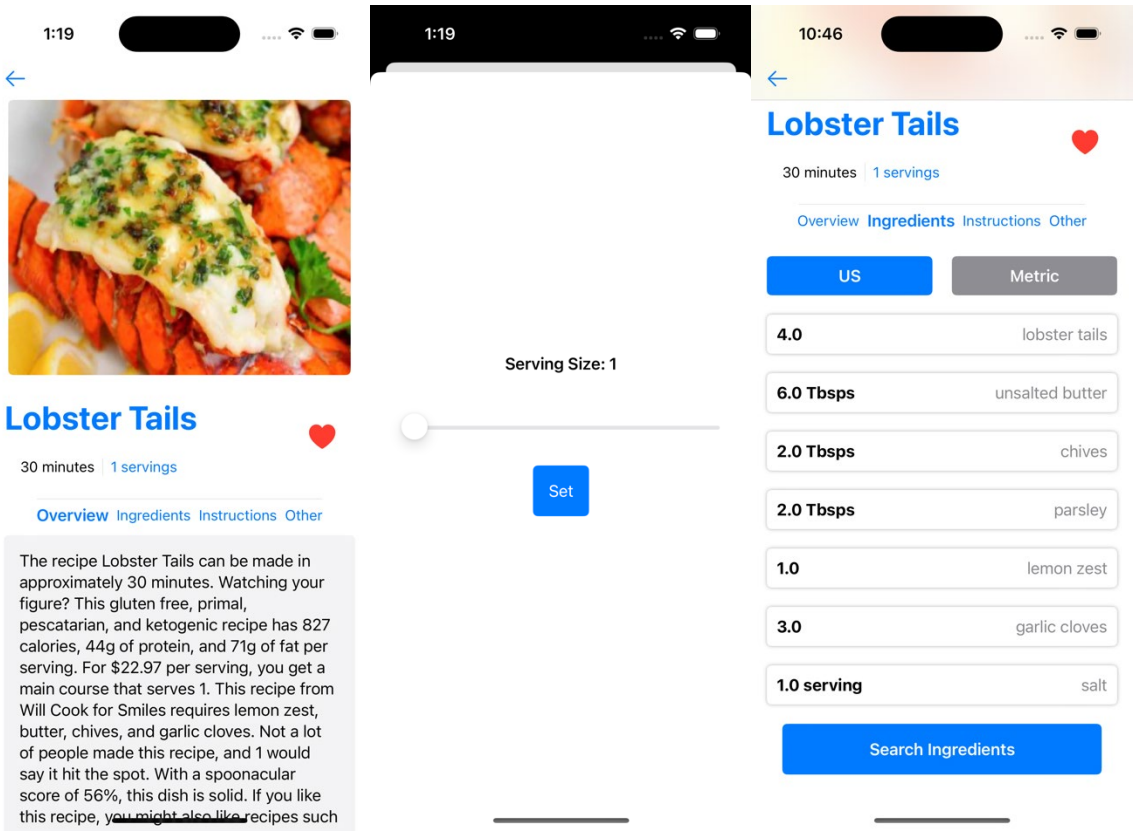

Figure 13, 14, and 15: Recipe Detail view User Interface Part I

Figure 13 is the initial view when opening a recipe. Figure 14 is the serving size adjustment popover. Figure 15 is an example of a recipe's ingredient list.

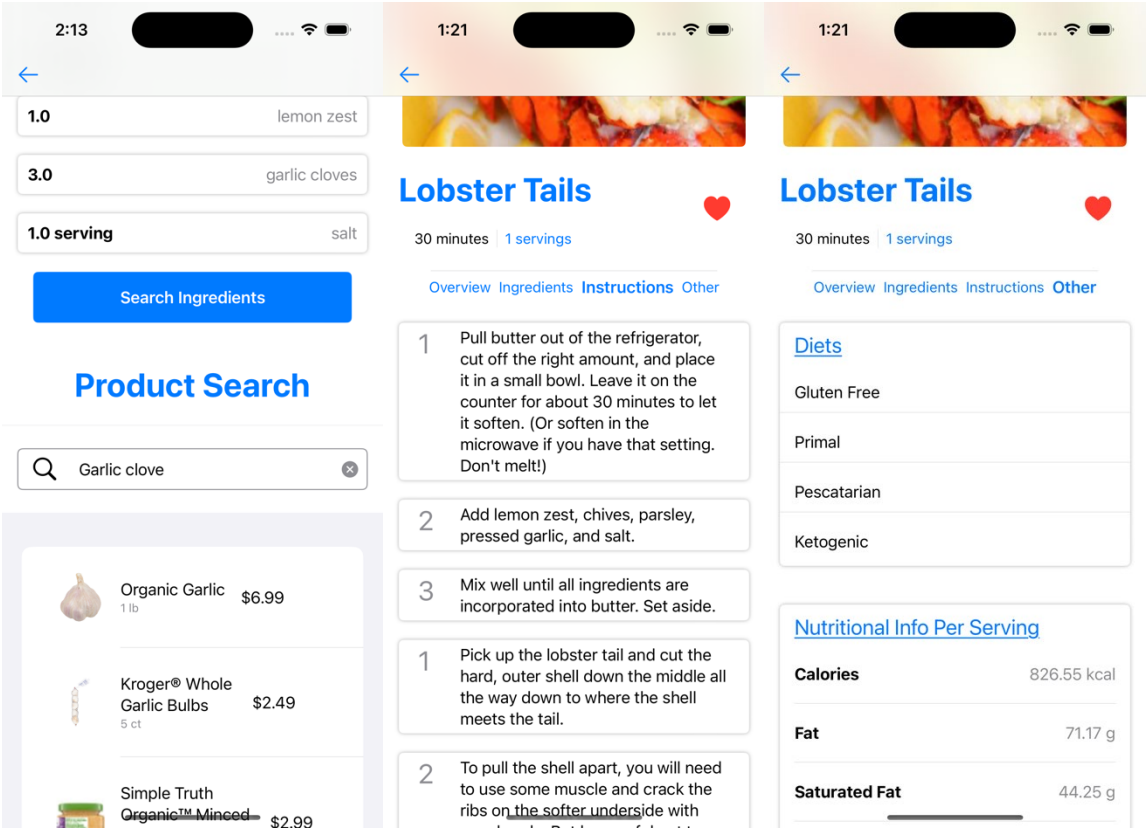

Figure 16, 17, and 18: Recipe Detail view User Interface Part II

Figure 16 is the extendable product search under the ingredients tab with an example search result. Figure 17 is the instructions tab. Figure 18 is the other tab.

Within the Recipe Detail view, users are provided with a comprehensive overview of the recipe, offering a thorough exploration of its ingredients, instructions, and additional details. Organized into four tabs—Overview, Ingredients, Instructions, and Other—the interface ensures a seamless browsing experience of this plethora of information. Each tab includes elements like the recipe image, title, servings, and a “heart” button for saving recipes. Serving adjustments are made by tapping on the serving text. The Overview tab provides a summary, the ingredients tab lists components with amounts and an extendable Product Search view, and the Instructions tab offers step-by-step guidance. The Other tab covers dietary preferences and nutritional info. Mealify displays the Recipe Detail view whenever a user taps on a recipe card.

Upon entering this view, users encounter an enlarged image of the recipe visible on the recipe card, accompanied by its bolded title below. Positioned at the top right of the screen is a left-pointing arrow icon, serving as the back button to return to the previous view. Beneath the title, users find details such as the estimated preparation and cook time, the current serving size, and a heart icon for saving the recipe. Tapping the serving size button triggers a popover to slide over the view, presenting a picker initialized to the original number of servings from the recipe. Users can adjust the slider to any serving amount from 1 to 20 and then tap the set button to confirm the serving amount and return to the Recipe Detail view. Adjacent to the serving size button is the heart icon for saving recipes, initially appearing unfilled. Tapping this icon turns the heart red, indicating that the recipe has been saved.

Below the page divider, the view features four tabs, with the overview tab highlighted in bold to indicate the current tab. The overview tab presents a summary text of the recipe within a Swift stack with a gray background, enhancing readability. In the ingredients tab, the ingredient list is displayed, accompanied by two buttons labeled "US" and "Metric" at the top for users to switch between measurement systems, with the currently selected system highlighted in blue. Each ingredient is listed in outlined stacks, with the ingredient name in bold on the left and the measurement amount on the right. Beneath this list, a large button labeled "Search Ingredients" opens the Product Search view (details provided in the subsequent section) below it, offering users seamless access like that from the Navigation sidebar. The instructions tab follows a similar format, presenting steps in outlined stacks with the step number on the left in large text and the description on the right. Finally, the others tab displays applicable diets in a stack, with each diet listed vertically and separated by dividers. The nutritional information stack is

positioned directly below, with each nutrient highlighted in bold on the left and its corresponding amount on the right, also separated by dividers.

When accessing the Recipe Detail view, the initializer requires a passed-in recipe structure to load the required information. Within this initializer, Mealify sets the current serving size to the original amount provided by the recipe, and the system checks if the recipe exists in the user defaults saved recipe list. If it does, the application fills in the saved heart icon. When the user opts to save a recipe, Mealify triggers a save function that encodes the recipe structure into JSON and stores it in the user's defaults saved recipe structure list. Depending on the user's desired ingredient measuring systems and serving size, the corresponding ingredient amounts are extracted from the recipe structure, which stores both US and Metric measurements. The system then divides these amounts by the original serving size to obtain the unit amount and then multiplies them by the selected serving size set by the serving size slider. When the user taps on the "Search Products" button in the ingredients tab, the Product Search view is summoned with a true boolean argument, indicating that the view opens as a child view. This action disables certain features, such as the toolbar icon. Additionally, the frame for this view adjusts to a larger size, specifically when opened from the Recipe Detail view.

Product Search

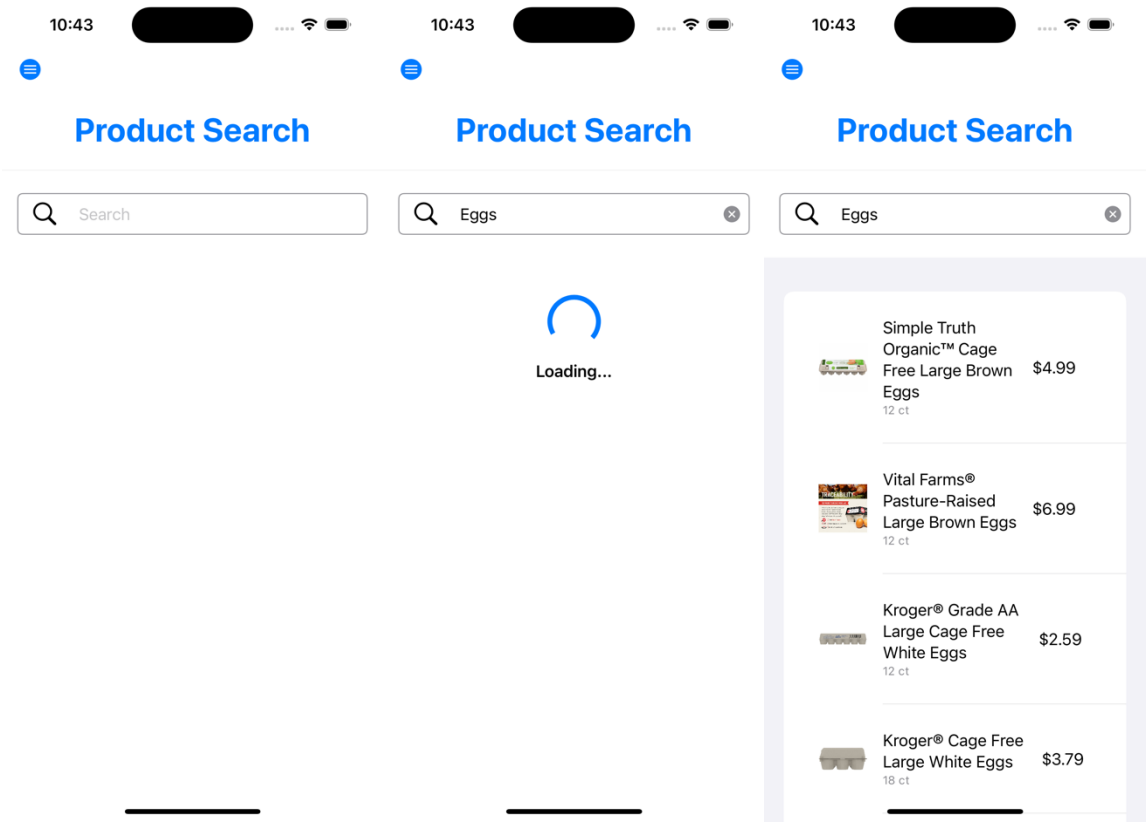

Figure 19, 20, and 21: Product Search view User Interface

These three images follow the chronological order use of the Product Search. Figure 19 is the initial view, prior to any search. Figure 20 shows the loading animation when the user prompts a search. Figure 21 shows the results for an example search.

The Product Search feature serves to locate ingredients for recipes at nearby Kroger locations, allowing users to check availability and pricing. Users can access this feature through the Navigation sidebar or directly from the ingredients list in the Recipe Detail view. Users determine the Kroger location used for the search in the Settings view. Currently a work in progress, this view offers a glimpse of its future potential. Mealify currently has active applications for other grocery APIs like Walmart. Once Mealify integrates multiple grocery APIs, it will provide real-time price comparisons for products available at nearby stores and offer a price per serving for each recipe.

The interface of the Product Search mirrors that of the Recipe Search. It begins with a title displayed above, followed by a search bar featuring a text field enclosed in a cornered rectangle, a search symbol, and a clear button. After initiating a search, the loading progress indicator appears, followed by the resulting products displayed in a list of stacks containing product images, descriptions, sizing, and pricing.

In the background, when a search term is submitted, Mealify calls a function that takes the search term and user data to make a GET API call to Kroger's product search. The user data, holding the desired location ID, serves as a function parameter. The API call follows a similar process to other API calls in Mealify: the search term is appended to the base URL query and then executed. The function decodes the JSON response from the API into defined product structures before passing them back to the Product Search view for display. However, the JSON response does not include product images, only URLs to access them. Therefore, when displaying the products, Mealify accesses the images via URL and converts the response to a UI Image via initiation of a URL session.

Settings

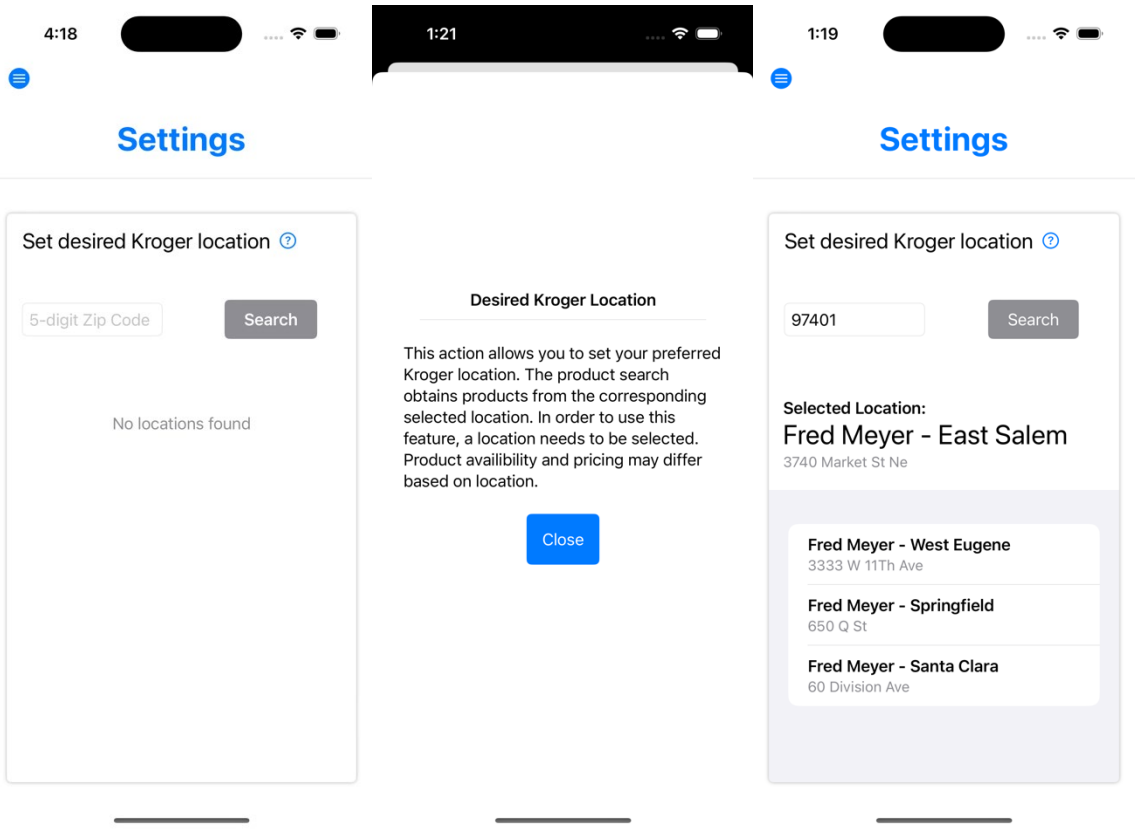

Figure 22, 23, and 24: Settings view User Interface

Figure 22 is the initial Settings view, prior to any location set. Figure 23 is the popover explanation for the set desired Kroger location setting. Figure 24 are example results for an inputted zip code.

The Settings view currently houses the location setting for Kroger product search. It displays the currently selected location if the user has already set one. To set or change the location, users input a 5-digit zip code into the provided field, and upon tapping the search button, nearby Kroger locations to that zip code will appear. Users can select an option from the list to set the location. This one setting is made separate from the other with the anticipation of further expansion. The Settings view will accommodate various upcoming settings, from user profiles and preferences to future options.

On the frontend, the Settings view adheres to the standard format: a blue bolded title at the top, followed by a divider, and then the location setting displayed in a rounded stack with a subtle shadow effect. On top of this stack, the setting title accompanies a blue question icon. Tapping this icon opens a popover explaining the setting's functionality. To close the popover, users tap the blue close button and return to the Settings view. Below the setting title, a text field for entering the zip code and a search button are available. If the user has a set location, the view displays its name and address after the "Selected location:" text; otherwise, it shows "No Locations Found." After inputting a 5-digit zip code, a list view appears below, displaying nearby locations, and selecting one updates the selected location text.

When users enter a zip code in the text field, the system invokes a function to retrieve the location. This function validates the zip code before making an API call with the appended zip code query. The function decodes the resulting JSON into a list of location structures and then returns this list to the Settings view for display in the list view. When the user taps on a respective location, the system saves the corresponding location ID to the user defaults.

Discussion

To assess Mealify's effectiveness in meeting user needs, we will conduct a comparative case study with other notable meal planning and recipe search apps previously mentioned in the background section. This study will simulate a typical user scenario that aligns with Mealify's target audience. The assessment will commence with app initialization, progressing to recipe discovery via search terms, filters, and recipe quality assessment. Subsequently, we will navigate through the steps of obtaining the groceries, following recipe instructions, monitoring nutrient intake, and addressing any other potential edge cases. We will analyze and compare the support provided by each app during these stages, offering valuable insights into Mealify's performance during its initial iteration. This comparative analysis serves as a crucial milestone for Mealify, illuminating both areas of success and improvement. Furthermore, it offers predictive insights into Mealify's potential market performance if launched in its current state.

The desired recipe and filtering constraints will be uniform for all apps. The criteria include an affordable gluten-free burrito bowl with a maximum preparation time of 40 minutes. It should contain chicken breasts, quinoa, and cilantro and exclude white rice and spinach. Also, the recipe must not exceed 50 grams of carbohydrates, should have at least 30 grams of protein, and a maximum of 60 milligrams of cholesterol.

Mealime

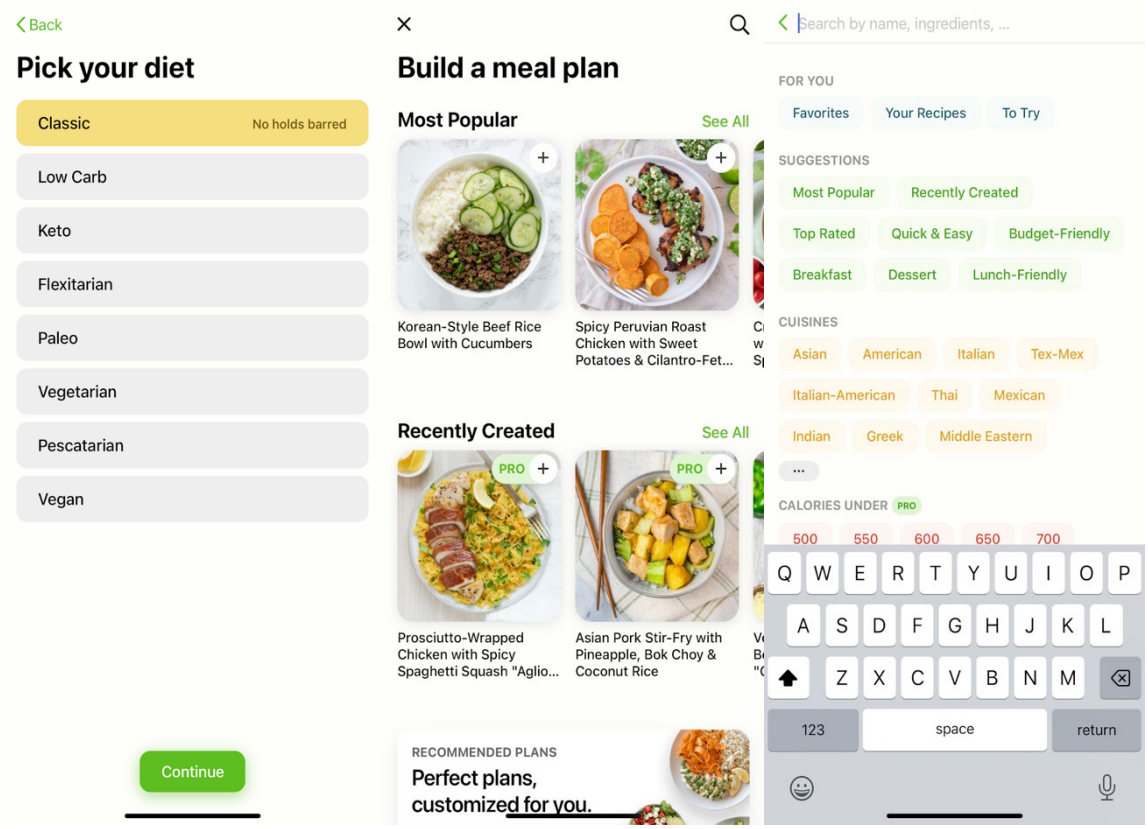

Figure 25, 26, and 27: Mealime use case User Interface Part I

These three images show a glimpse of the first few steps when completing the use case in Mealime. Figure 25 is one of the prompted questions when initially launching the app. Figure 26 is the starting view when building a meal plan. Figure 27 is the starting view when completing a search.

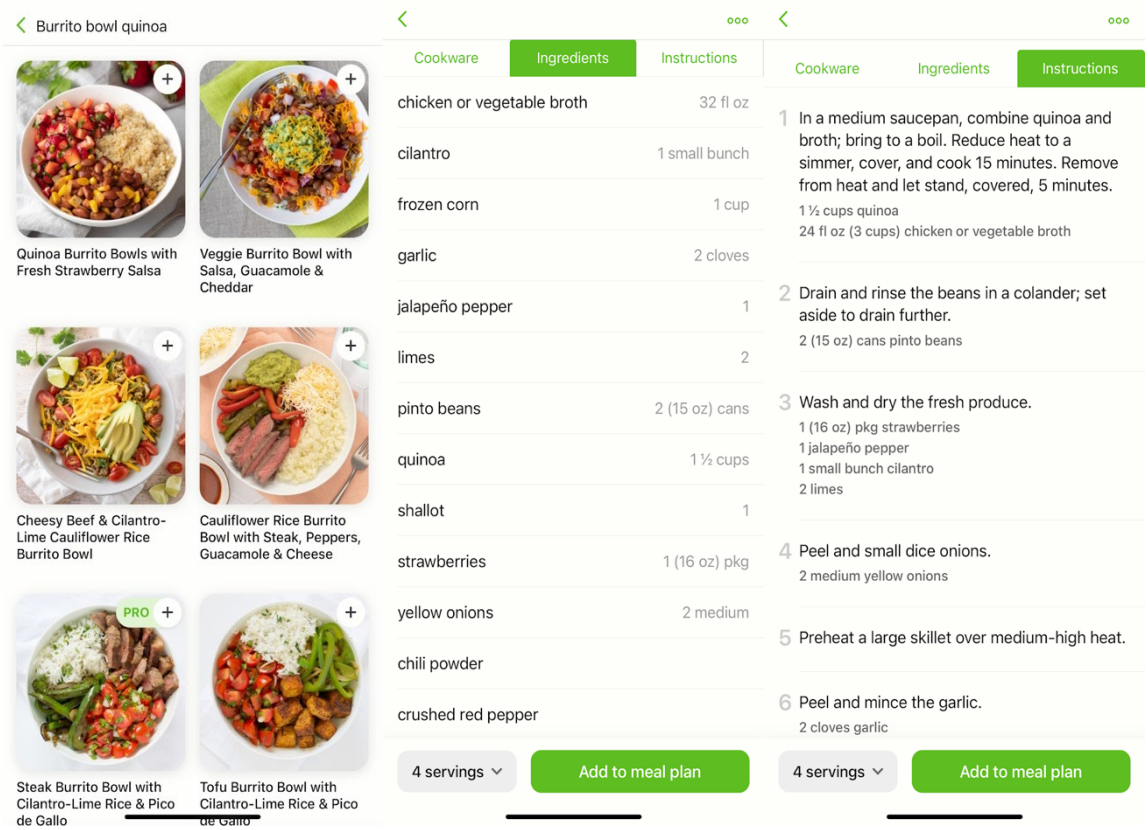

Figure 28, 29, and 30: Mealime Use Case User Interface Part II

These three images show a glimpse of the next few steps when completing the use case in Mealime. Figure 28 are the results for the use case search. Figure 29 is the ingredient list for the selected recipe. Figure 30 is the step-by-step instructions for the selected recipe.

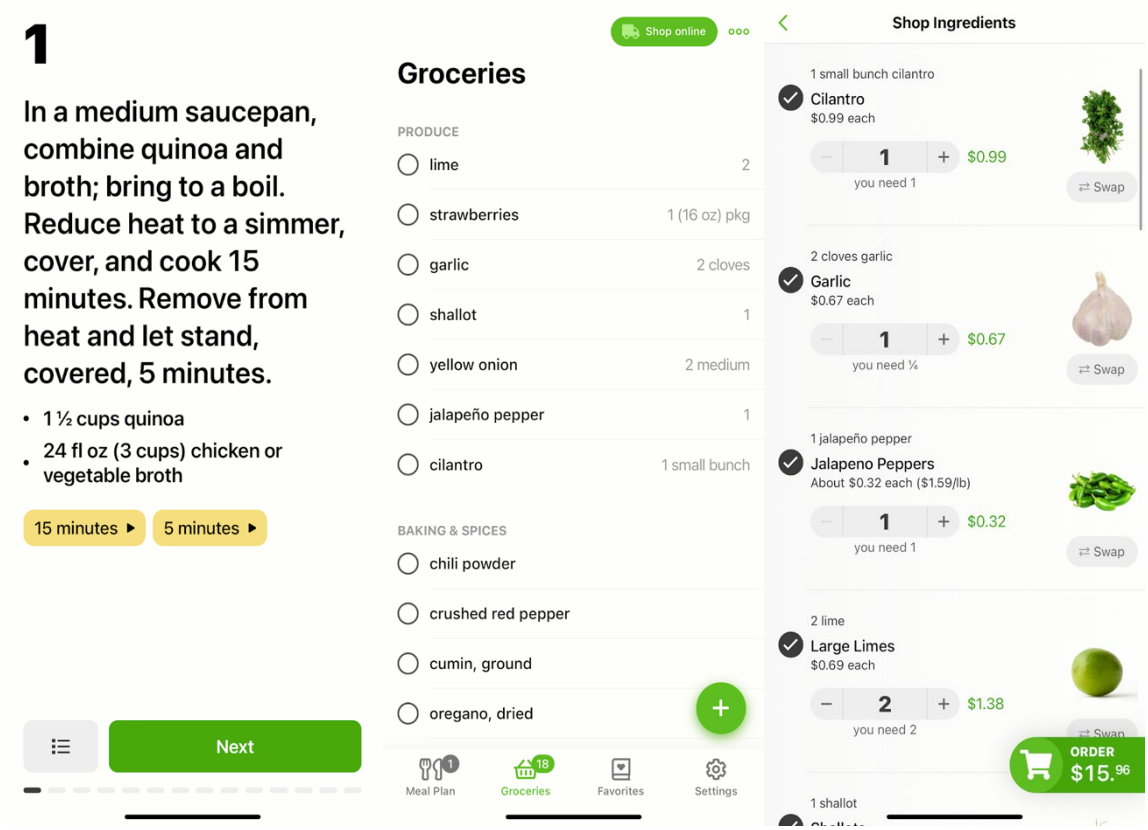

Figure 31, 32, and 33: Mealime Use Case User Interface Part III

These three images show a glimpse of the final steps when completing the use case in Mealime. Figure 31 is the first instruction step in the cooking mode. Figure 32 is the grocery list created from the meal plan. Figure 33 is the grocery cart for the selected grocery store.

Mealime was the first app tested in this study and overall performed well for this scenario as it caters to a similar audience as Mealify. Upon startup, Mealime provides app descriptions and prompts users with questions for customization. For this scenario, preferences included a classic diet, gluten allergy, no dislikes, and four servings per meal. This approach of setting constraints at the outset is logical for personalization. However, adjustments to preferences such as dislikes require navigating to a different view than the search function. After setup and account creation, I initiated the recipe search by accessing the meal plan page, starting a plan, and selecting the search icon. Due to filtering features being exclusive to pro members, only gluten-allergen filters were available. Hence, finding a suitable recipe depended entirely on

manually inspecting ingredient lists. Similarly, nutrient information and ingredient exclusions were limited to pro membership. Among the free recipes, the closest match was a quinoa burrito bowl lacking chicken and featuring a strawberry salsa, which failed to meet the protein requirement. To help acquire ingredients, users can add them to a grocery list within Mealime and link a grocery store account for streamlined purchasing. This aspect of the process was efficient and straightforward. Following ingredient acquisition, the user can follow cooking instructions using Mealime's detailed steps and cooking view with timers, simplifying the process. However, tracking nutrient information requires manual calculation, as pro membership is necessary to access this feature. The main challenge of this use case in Mealime lies in recipe searching, with no options meeting all constraints and limited access to filters. Despite this, Mealime excelled in grocery shopping and cooking instructions, emphasizing the importance of these aspects in the process.

Intent

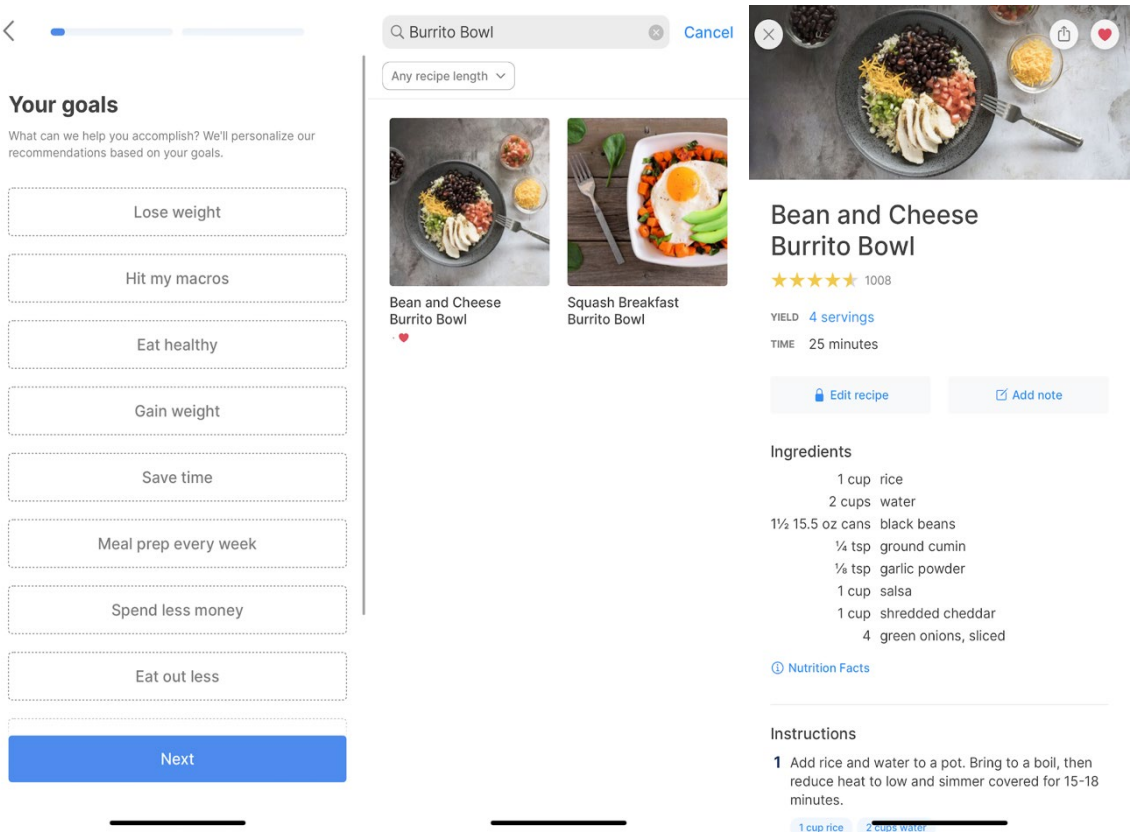

Figure 34, 35, and 36: Intent Use Case User Interface Part I

These three images show a glimpse of the first few steps when completing the use case in Intent. Figure 34 is one of the prompted questions when initially launching the app. Figure 35 are the search results for the use case. Figure 36 are the details for the selected recipe.

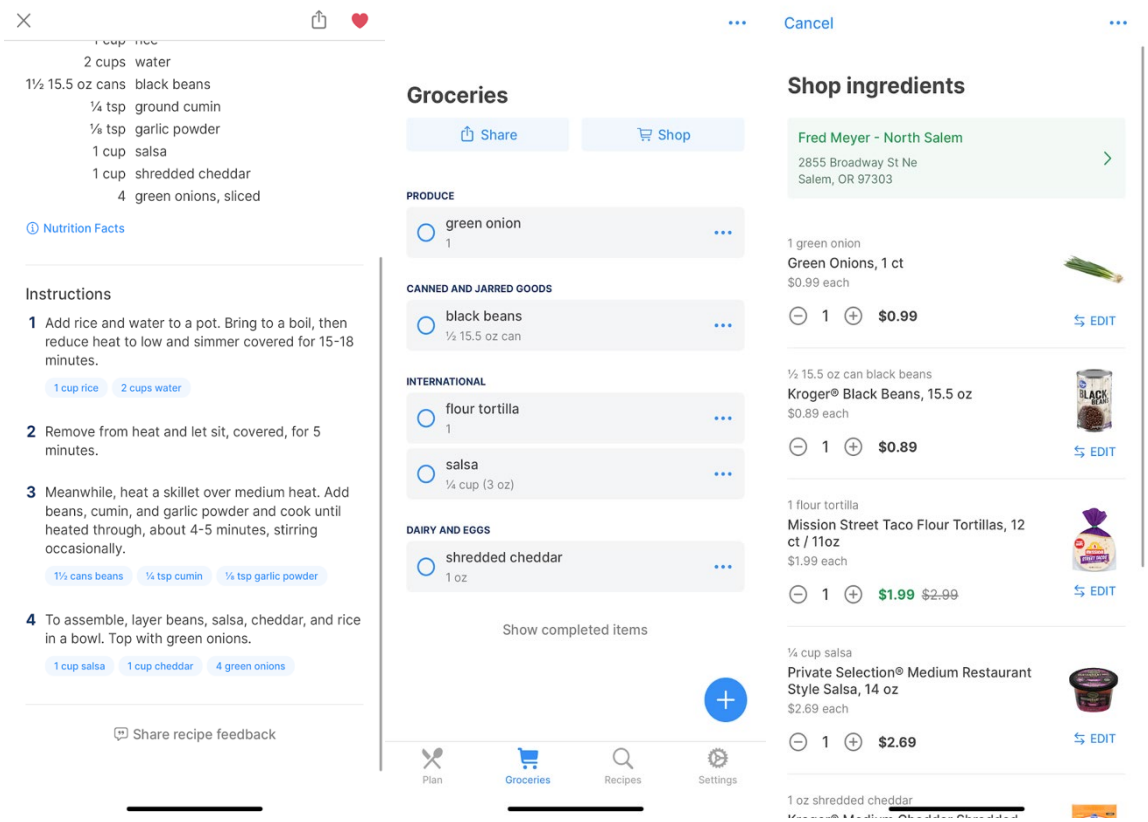

Figure 37, 38, and 39: Intent Use Case User Interface Part II

These three images show the next few steps when completing the use case in Intent. Figure 37 are the step-by-step instructions for the selected recipe. Figure 38 is the grocery list for the selected recipe. Figure 39 is the converted shopping list when sending the grocery list over to a third-party grocery app.

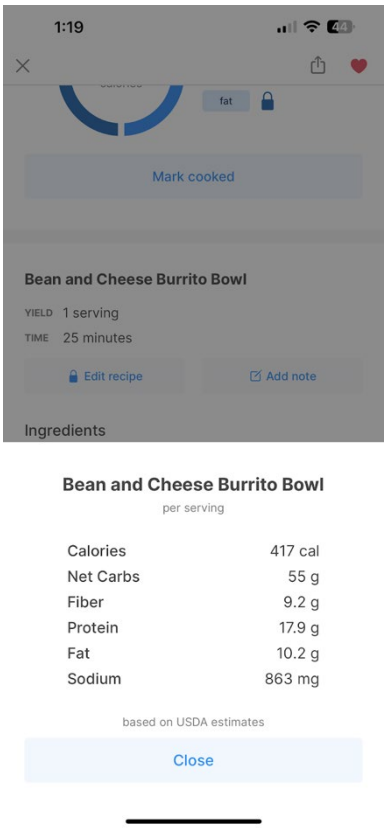

Figure 40: Intent Use Case User Interface Part III

This image is the last step when completing the use case in Intent and shows the nutritional information for the selected recipe.

Intent, the other app evaluated in this scenario, delivered a performance like Mealime, with some slight distinctions. Upon opening the app, the app prompts users with questions regarding goals, nutritional preferences, diets, intended meals to prepare, restrictions, food dislikes, grocery shopping preferences, and more. For this scenario, I set the goals to try new recipes, the diet to be balanced, meals to lunch and dinner, and restrictions to gluten. To initiate the recipe search, I navigated to the recipe tab via the navigation menu, entered the search term "burrito bowl," and disregarded the only available free filter, which was preparation time, since the only two options were 15 and 30 minutes maximum. The search results were disappointing, with only two options for a burrito bowl, the better of which was the beans and cheese burrito bowl. While the cooking time met the scenario requirement, the ingredients lacked chicken

breast, quinoa, or cilantro. Intent offers limited nutritional information for each recipe; based on the provided information, this recipe contained insufficient protein and excessive carbohydrates. After manually calculating the cholesterol, it met the requirement. To obtain the grocery lists, Intent requires users to add a recipe to a meal plan. Once the user confirms the meal plan, it prompts for ingredients the user might already have and then generates a bulleted grocery list organized into sections such as produce, canned goods, international items, and dairy. Users can also link to a nearby grocery store account, generating a list of ingredients with the store's products and pricing for easy purchasing. This feature closely resembles Mealime's, and the organized bulleted grocery list simplifies in-store shopping. For recipe preparation, instructions are listed on the same page as the recipe information, displaying ingredient amounts for each step. These steps are overall straightforward and succinct. The final step in this scenario is nutrient tracking. While Intent provides select nutrient counts such as calories, net carbs, fiber, protein, fat, and sodium, users must manually calculate other nutrient amounts. From these observations, Intent showed weaknesses in finding the desired recipe due to limited variety and filtering options. It also lacked comprehensive nutrient information for each recipe. However, Intent offered the best experience among the three apps in terms of grocery list features, organization, streamlined third-party app integration, and solid step-by-step instructions.

Mealify

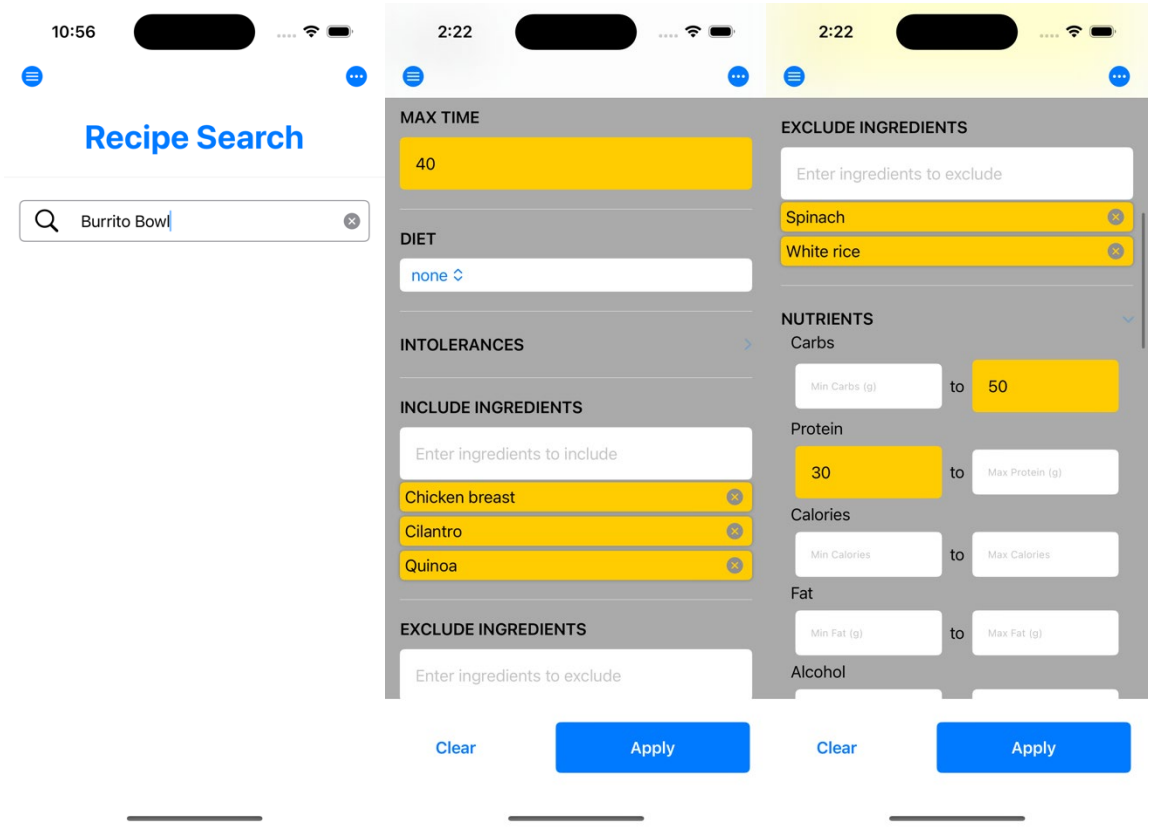

Figure 41, 42, and 43: Mealify Use Case User Interface Part I

These three images show a glimpse of the first few steps when completing the use case in Mealify. Figure 41 is the initial Recipe Search view with the use case search term. Figures 42 and 43 shows the applied filtering constraints to the search.

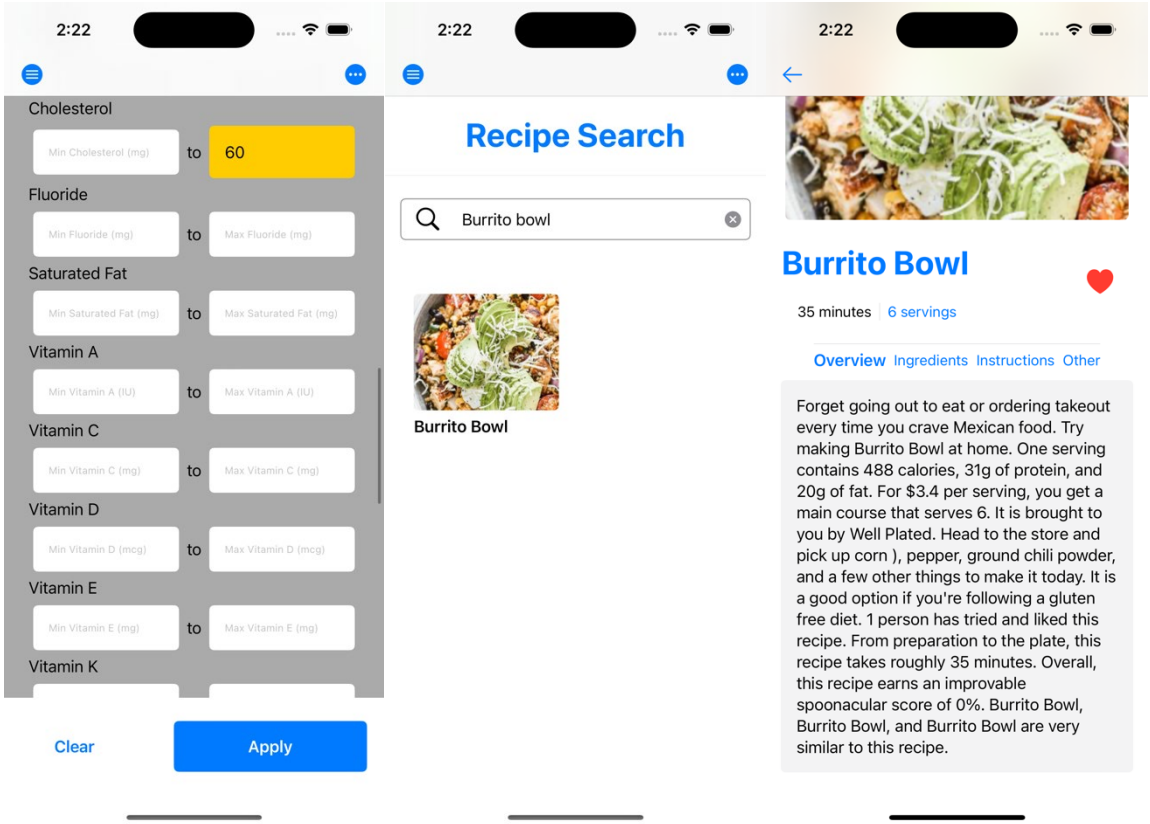

Figure 44, 45, and 46: Mealify Use Case User Interface Part II

These three images show a glimpse of the next few steps when completing the use case in Mealify. Figure 44 is the last search filtering constraint applied. Figure 45 is the search result after applying the filters. Figure 46 is the overview summary for the selected recipe.

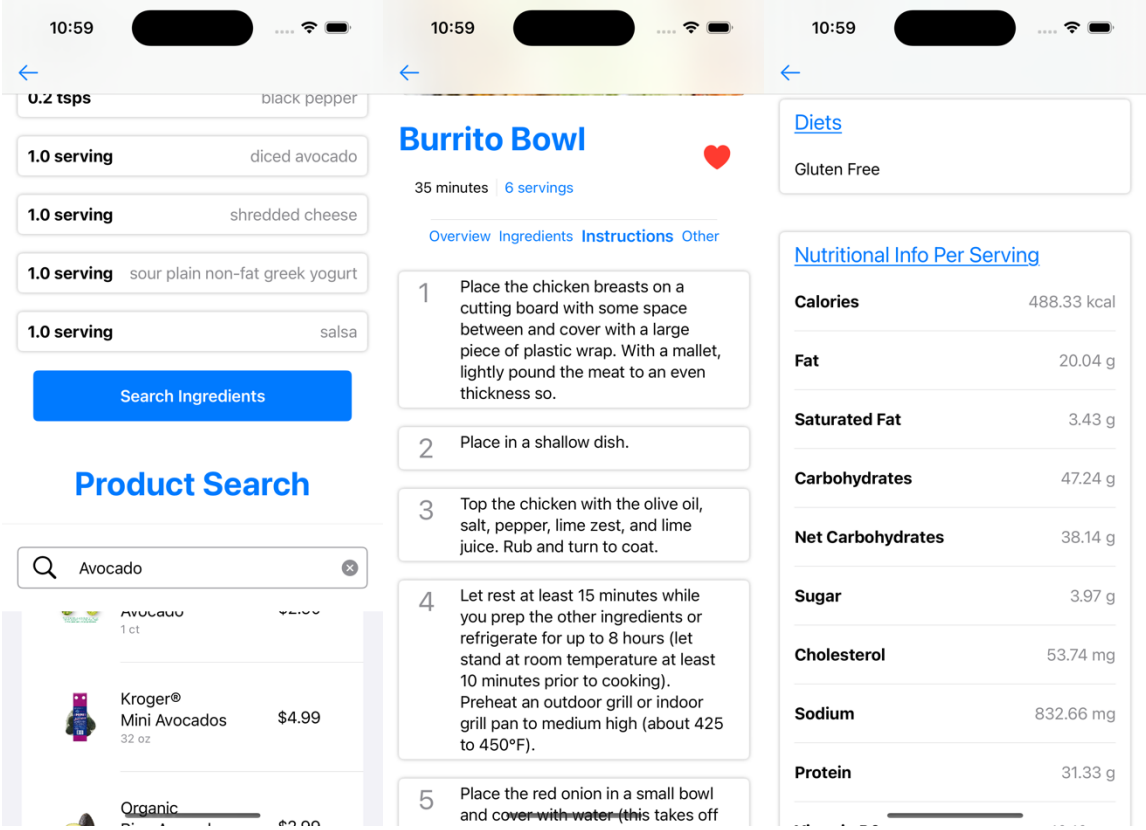

Figure 47, 48, and 49: Mealify Use Case User Interface Part III

These three images show a glimpse of the final steps when completing the use case in Mealify. Figure 47 shows part of the ingredients list with a search for one of the products. Figure 48 are the step-by-step instructions for the selected recipe. Figure 49 is the other tab for the recipe, which shows the applicable diets and nutritional information.

Regarding Mealify, it excelled in areas aligned with the specific use case while adequately addressing other aspects of the process. Upon the initial launch of Mealify, there are no setup questions or account creation; instead, users are taken directly to the landing page. To access the recipe search tab, I opened the navigation sidebar and navigated to this view. Entering the search term "burrito bowl" in the search bar, I opened the filters to apply all the required constraints. Upon running the search, one recipe titled "Burrito Bowl" appeared. This recipe provides a summary overview, including details such as the $3.40 price per serving and other enticing information. This recipe met all the use case requirements as the preparation time is 35

minutes, the ingredient list includes chicken breasts, quinoa, and cilantro, and excludes white rice and spinach. Moreover, the recipe nutrient amounts indicate 47g of carbohydrates, 31g of protein, and 54g of cholesterol per serving. Therefore, Mealify offers an advanced method to search for precise recipes while providing access to a variety of options that meet user needs. To view the list of ingredients, I tapped on the ingredients tab, where there is also an extendable product search to locate ingredients at nearby Kroger locations. For each required product, I found all the matches at my nearby Fred Meyer with their real-time prices. However, there was no way to keep a checklist of the ingredients in Mealify or an option to transfer the list to my Fred Meyer app, so I completed this part of the process manually outside of the app. When time for cooking, the instructions are listed under the corresponding tab and are easy to follow, with each step listed from top to bottom. Unlike the other two apps, Mealify did not show ingredient amounts at each step. To see the amounts, the user must switch between tabs. To keep track of the nutrient levels in the recipe, Mealify offers a comprehensive list in the other tab that includes all the necessary nutrients I may need to monitor.

Overall, Mealify stood out as the most adept at providing recipes that align with the specific constraints of the user's needs. However, the other competitor apps highlighted areas for improvement to further enhance Mealify's instructions, grocery shopping experience, and user preference features. Despite this, considering the time and effort invested in each aspect of Mealify, its performance at every step of the use case met expectations.

Conclusion

In its present state, Mealify represents an innovative solution tailored to meet the needs of individuals seeking assistance in meal planning and recipe searching. It offers users the ability to discover precisely the recipes they desire within their budgetary constraints. Drawing from Mealify's developmental structure and its comparative performance with similar iOS apps in the market, it can be deduced that the current progress aligns with the established standards within this domain. This progress also serves as the foundation for the integration of groundbreaking future functionalities.

When it comes to advancing current practices in Mealify, API usage remains at the forefront. The backend logic allows for future integration of a recipe API that offers a wider variety in recipe options and a multitude of other grocery APIs that are available to developers. These incremental changes will enhance the current functionalities of Mealify immensely.

Looking ahead, Mealify’s trajectory offers numerous avenues for improvement. It is best to focus on test cases for the current functionalities. Implementing extensive test cases ensures stability and reliability when further modifying code. Then, price per serving with real time grocery API pricing is the next priority since Mealify holds most of the data needed for this feature. Moreover, offering flexible sorting options for recipes and products boosts the usability of the recipe and product searches. Expanding Mealify’s capability to include comprehensive meal and grocery planning features promotes better meal organization and preparation. The introduction of user profiles facilitates personalization and cross-device access. By considering these potential directions in subsequent versions, Mealify is poised to meet the evolving needs of its users, solidifying its position as one of the leading meal planning and recipe search apps in the marketplace.

Bibliography

Apple Inc. "Designing for iOS - Platforms - Human Interface Guidelines - Design - Apple Developer." Accessed n.d. https://developer.apple.com/design/human-interface-guidelines/platforms/designing-for-ios/.

Apple Inc. "Planning Your iOS App." Apple Developer. Accessed April 17, 2023. https://developer.apple.com/ios/planning/.

“AsyncImage.” n.d. Apple Developer Documentation. Accessed May 3, 2024. https://developer.apple.com/documentation/swiftui/geometryreader/.

“Button.” n.d. Apple Developer Documentation. Accessed May 3, 2024. https://developer.apple.com/documentation/swiftui/button/.

“Color.” n.d. Apple Developer Documentation. Accessed May 3, 2024. https://developer.apple.com/documentation/swiftui/color./.

Cooklist Inc. "Cooklist: Pantry Meals Recipes." Apple App Store, Vers. 1.83.1 (2023). Accessed May 9, 2023. https://apps.apple.com/us/app/cooklist-pantry-meals-recipes/id1352600944.

“CornerRadius.” n.d. Apple Developer Documentation. Accessed May 3, 2024. https://developer.apple.com/documentation/quartzcore/calayer/1410818-cornerradius/.

“DataTask(With:completionHandler:).” n.d. Apple Developer Documentation. Accessed May 3, 2024. https://developer.apple.com/documentation/foundation/urlsession/1407613-datatask/.

“DisclosureGroup.” n.d. Apple Developer Documentation. Accessed May 3, 2024. https://developer.apple.com/documentation/swiftui/disclosuregroup/.

“Divider.” n.d. Apple Developer Documentation. Accessed May 3, 2024. https://developer.apple.com/documentation/swiftui/divider/.

Excipient, Inc. "Intent Meal Planner & Recipes." Apple App Store, Vers. 3.30 (2023). Accessed May 9, 2023. https://apps.apple.com/us/app/intent-meal-planner-recipes/id1488552222.

“GemoetryReader.” n.d. Apple Developer Documentation. Accessed May 3, 2024. https://developer.apple.com/documentation/swiftui/geometryreader/.

Glamazdina, Yulya. “The Agile Software Development Life Cycle: What Is Agile SDLC and How to Use It?” The Agile Software Development Life Cycle: What is Agile SDLC and How to Use It? | Brocoders blog about software development. Accessed May 3, 2024. https://brocoders.com/blog/agile-software-development-life-cycle/#:~:text=There%20are%205%20phases%20within,Custom%20priority.

“HStack.” n.d. Apple Developer Documentation. Accessed May 3, 2024. https://developer.apple.com/documentation/swiftui/hstack/.

“Images(_:).” n.d. Apple Developer Documentation. Accessed May 3, 2024. https://developer.apple.com/documentation/swiftui/shader/argument/image(_:)/.

“JSONDecoder.” n.d. Apple Developer Documentation. Accessed May 3, 2024. https://developer.apple.com/documentation/foundation/jsondecoder/.

“JSONEncoder.” n.d. Apple Developer Documentation. Accessed May 3, 2024. https://developer.apple.com/documentation/foundation/jsonencoder./.

“JSONSerialization.” n.d. Apple Developer Documentation. Accessed May 3, 2024. https://developer.apple.com/documentation/foundation/jsonserialization/.

“LazyVGrid.” n.d. Apple Developer Documentation. Accessed May 3, 2024. https://developer.apple.com/documentation/swiftui/lazyvgrid/.

“List.” n.d. Apple Developer Documentation. Accessed May 3, 2024. https://developer.apple.com/documentation/swiftui/list/.

Mealime Meal Plans Inc. "Mealime Meal Plans & Recipes." Apple App Store, Vers. 4.16.6 (2023). Accessed May 9, 2023. https://apps.apple.com/us/app/mealime-meal-plans-recipes/id1079999103.

“NavigationLink.” n.d. Apple Developer Documentation. Accessed May 3, 2024. https://developer.apple.com/documentation/swiftui/shader/argument/image(_:)/.

“Padding.” n.d. Apple Developer Documentation. Accessed May 3, 2024. https://developer.apple.com/documentation/tvml/padding/.

“Picker.” n.d. Apple Developer Documentation. Accessed May 3, 2024. https://developer.apple.com/documentation/swiftui/picker/.

“Popover.” n.d. Apple Developer Documentation. Accessed May 3, 2024. https://developer.apple.com/documentation/swiftui/presentationadaptation/popover/.

“Rectangle.” n.d. Apple Developer Documentation. Accessed May 3, 2024. https://developer.apple.com/documentation/swiftui/rectangle/.

“Scrollable.” n.d. Apple Developer Documentation. Accessed May 3, 2024. https://developer.apple.com/documentation/appkit/nscell/1534125-scrollable/

“ScrollView.” n.d. Apple Developer Documentation. Accessed May 3, 2024. https://developer.apple.com/documentation/swiftui/scrollview./.

“SF Symbols.” n.d. Apple Developer Documentation. Accessed May 3, 2024. https://developer.apple.com/sf-symbols/.

“Shadow(_:).” n.d. Apple Developer Documentation. Accessed May 3, 2024. https://developer.apple.com/documentation/swiftui/shapestyle/shadow(_:)-swift.method/.

"SideBar." n.d. Apple Developer Documentation. Accessed May 3, 2024. https://developer.apple.com/documentation/swiftui/searchfieldplacement/sidebar/.

"Spacer." n.d. Apple Developer Documentation. Accessed May 3, 2024. https://developer.apple.com/documentation/swiftui/spacer/.

"Text." n.d. Apple Developer Documentation. Accessed May 3, 2024. https://developer.apple.com/documentation/swiftui/text/.

"TextField." n.d. Apple Developer Documentation. Accessed May 3, 2024. https://developer.apple.com/documentation/swiftui/textfield/.

"Toolbars." n.d. Apple Developer Documentation. Accessed May 3, 2024. https://developer.apple.com/design/human-interface-guidelines/toolbars#.

"URL." n.d. Apple Developer Documentation. Accessed May 3, 2024. https://developer.apple.com/documentation/foundation/url/.

"URLQueryItem." n.d. Apple Developer Documentation. Accessed May 3, 2024. https://developer.apple.com/documentation/foundation/urlqueryitem/.

"URLRequest." n.d. Apple Developer Documentation. Accessed May 3, 2024. https://developer.apple.com/documentation/foundation/urlrequest./.

"UserData." n.d. Apple Developer Documentation. Accessed May 3, 2024. https://developer.apple.com/documentation/foundation/jsonencoder/.

"UserDefaults." n.d. Apple Developer Documentation. Accessed May 3, 2024. https://developer.apple.com/documentation/foundation/userdefaults/.

"VStack." n.d. Apple Developer Documentation. Accessed May 3, 2024. https://developer.apple.com/documentation/swiftui/vstack/.

"WithAnimation(_:_:)." n.d. Apple Developer Documentation. Accessed May 3, 2024. https://developer.apple.com/documentation/swiftui/withanimation(_:_:)/.

"ZStack." n.d. Apple Developer Documentation. Accessed May 3, 2024. https://developer.apple.com/documentation/swiftui/zstack/.