**Initial Architecture**

**Team Number**: 15

**Team Members**: Audrey Pino, Tony Czajka, Harvey Ji, Aiden Frevert, Nick Anaya

**Project Name**: CookIt

**Project Synopsis**: Mobile social media app for logging used recipes and finding recipes based on popularity, favorite chefs, dietary restrictions, and owned ingredients.

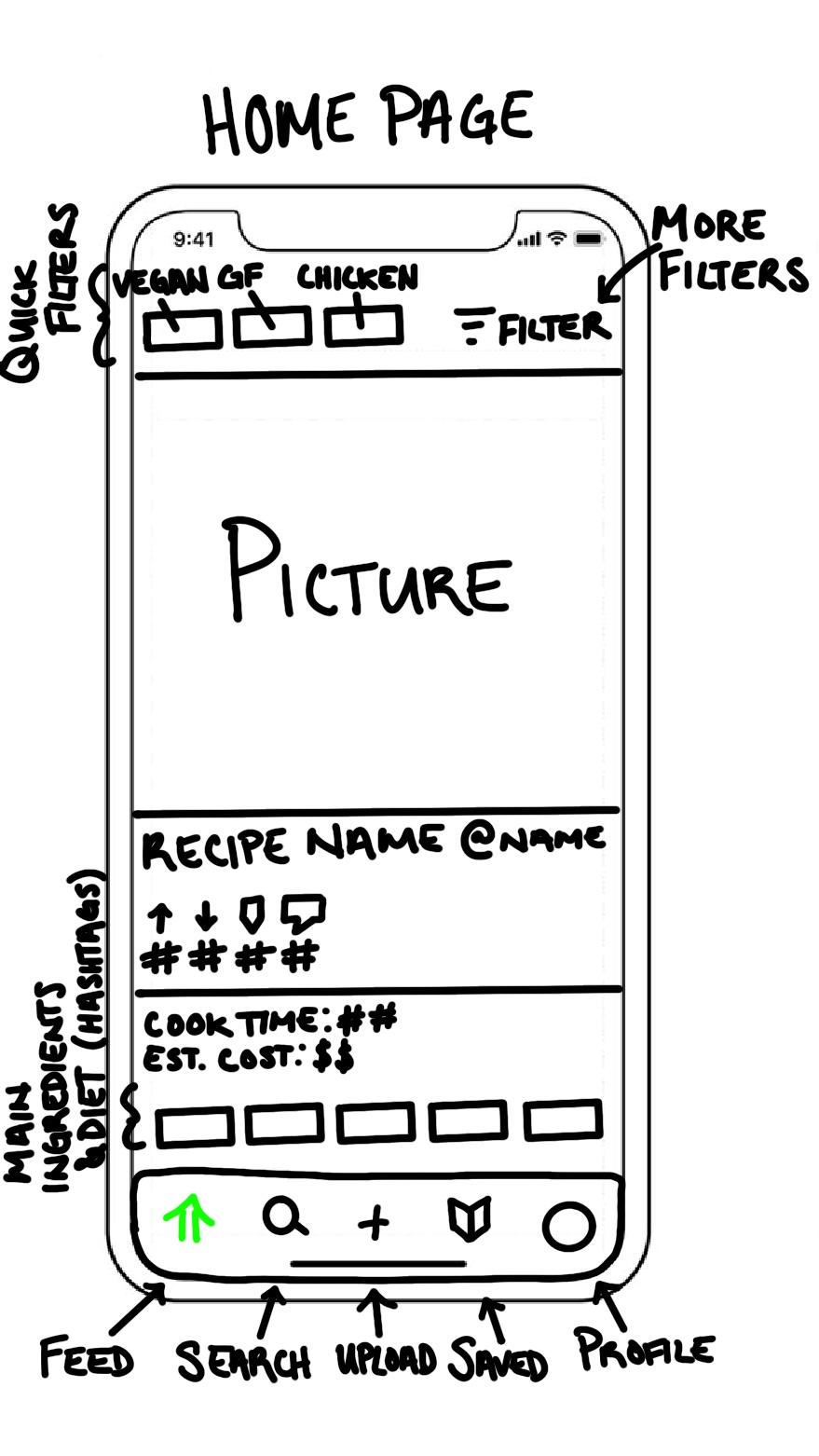
## Architecture:

# How The Software Works

Our application provides a quick and engaging way for users to discover and share recipes. The app introduces a social aspect to cooking by taking elements inspired by Instagram and Tik Tok and applying them to an application where users can share and save recipes with friends and professionals. The ability to quickly save recipes, edit them, add them to lists, and leave reviews drastically improves the user experience for finding meal ideas. Functionality for users to filter their search for recipes based on certain ingredients they have, dietary restrictions, or category of meal enables users to find a usable recipe quickly, without having to sift through tons of unwanted recipes. The app has a feature where you can upload your pantry so the app knows what food you have to use, and searches can be automatically performed based on ingredients you have. Placing multiple recipes into a list allows users to instantly generate a shopping list of all the ingredients they will need to make the recipes in that list. Meal prep is also made easy with the app. The calendar function allows for quick drag and drop of recipes to their associated days so users can easily plan out their meals for the week. Our app is heavily focused on how the user interacts and moves within the app. Swiping movements and shortcuts are essential to the user experience in the app. Quickly swiping through recipe options like tik-tok / tinder ensures the user is able to quickly parse through potential recipes, and with a different directional swipe can easily save to a list of recipes (to make) the next week or another custom list. A separate swipe can tell the algorithm to not recommend recipes like this one, and another touch can quickly bring up reviews and ratings of the recipe. All these features are aimed to improve upon the current process for finding recipes which is to sift through food blogs on google which are often littered with ads and are overly complicated. In the future, the goal for the app is to be able to integrate grocery stores with the shopping list feature so that with the click of a button the user can order the necessary ingredients online from the app.

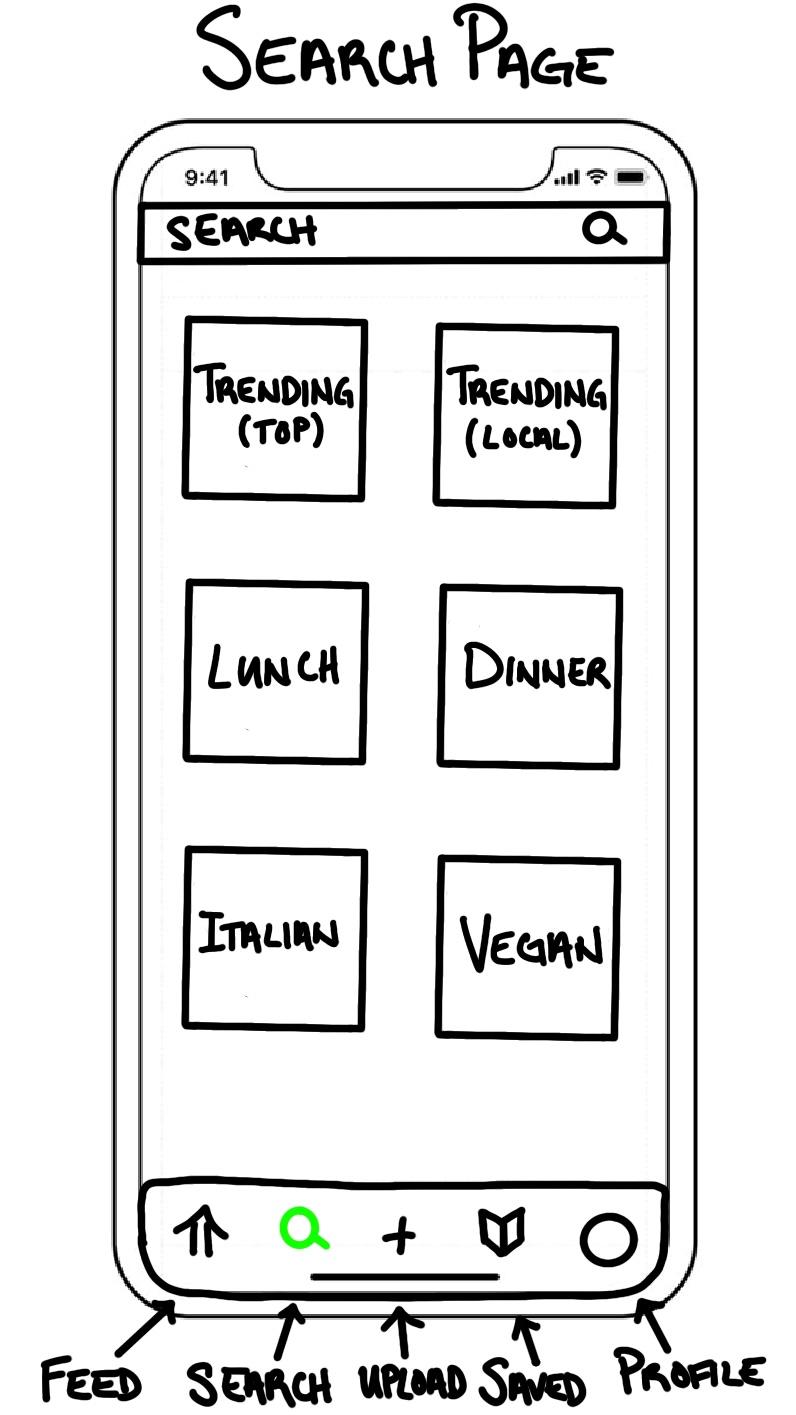
# UI Design

To begin, the home page of the app will be the area in which users discover recipes. The pages header will contain a few quick add filters which can be customized by the user as well as a filter button that has a list of other filters a user can add to their feed. The feed will display one recipe at a time, allowing for the user to see a picture of the meal as well as a thumbnail that includes the recipes name, number of up votes, down votes, saves, comments, cook time, estimated cost, and the main ingredients used in the recipe. This view aims to give the user a quick idea of the meal. If the user wants to know more about the recipe, they are easily able to expand the details section to reveal the full list of ingredients as well as the recipe instructions. The user can swipe left for a new recipe or they can swipe up to save the recipe.



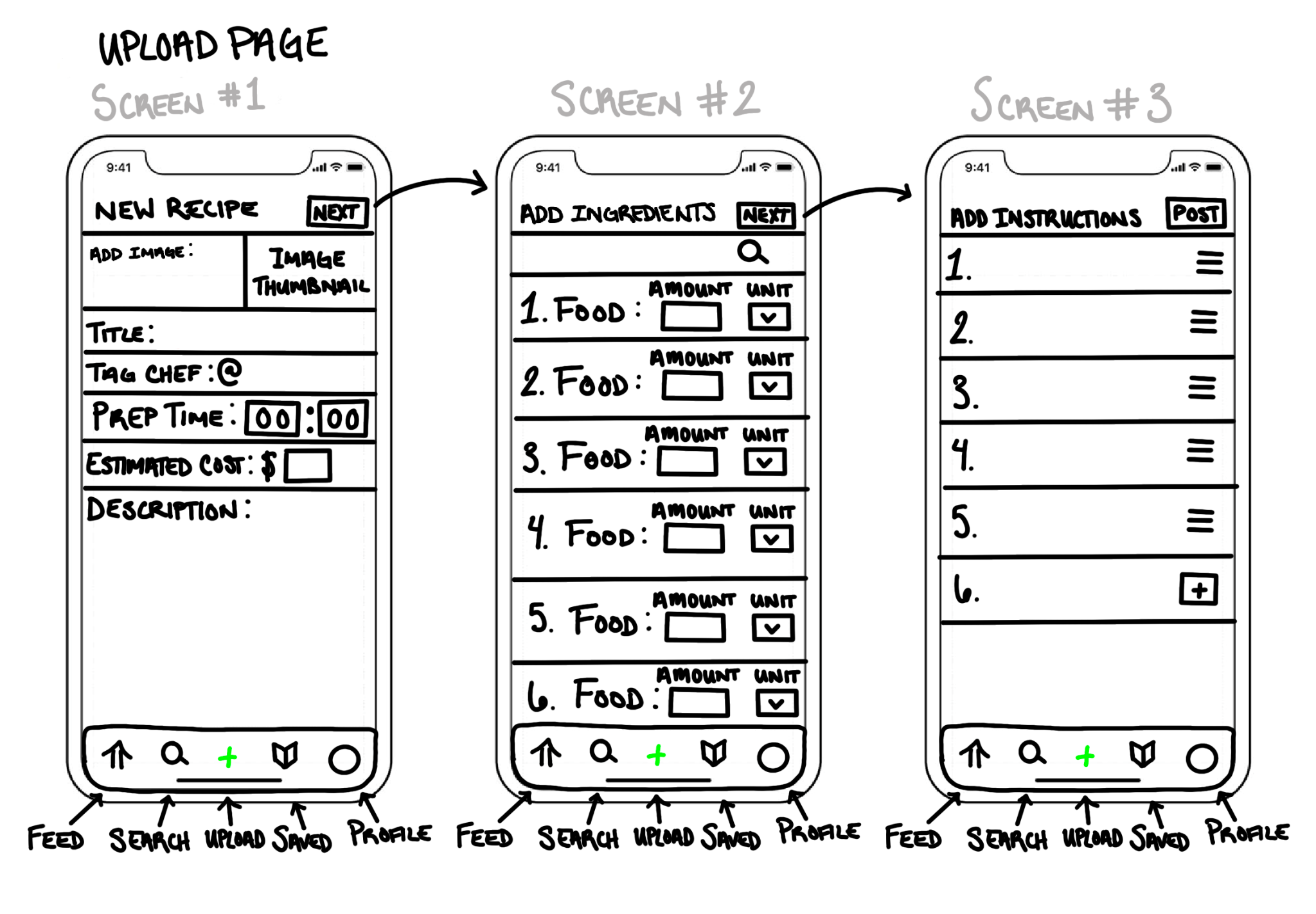
*Figure 1: Homepage wireframe*

To further discover new recipes, the user can proceed to the search page. Here, the user is able to search for specific chefs, recipes by titles, diet, or by specific ingredient. There are also a few quick links on this page that will take the user to the top trending recipes, local trending recipes, as well as trending lunch, breakfast, and dinner recipes. These quick links will eventually cater towards the individual user by recommending quick links based on their commonly used filters.



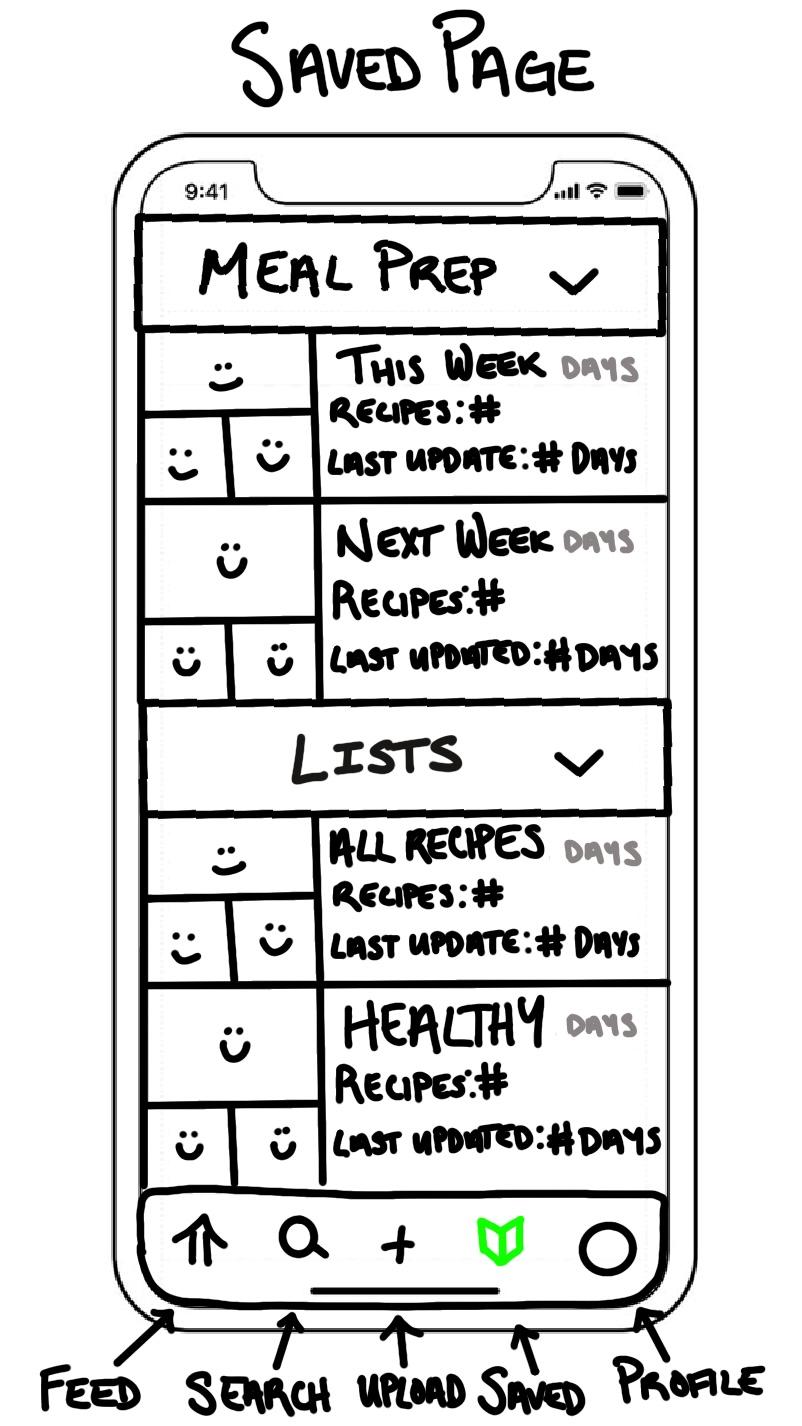
*Figure 2: Search page wireframe*

Next, the user is able to post their own recipe on the upload page. Here they will upload the picture for the recipe and add all of the recipes ingredients and instructions. The first step to this process will be uploading a thumbnail picture, and inputting the recipes title, collaborators, preparation time, estimated cost, and short description. The next button will take them to the ingredients pages where they will be able to add each ingredient and quantity. The final step will be adding the instructions to preparing the meal.



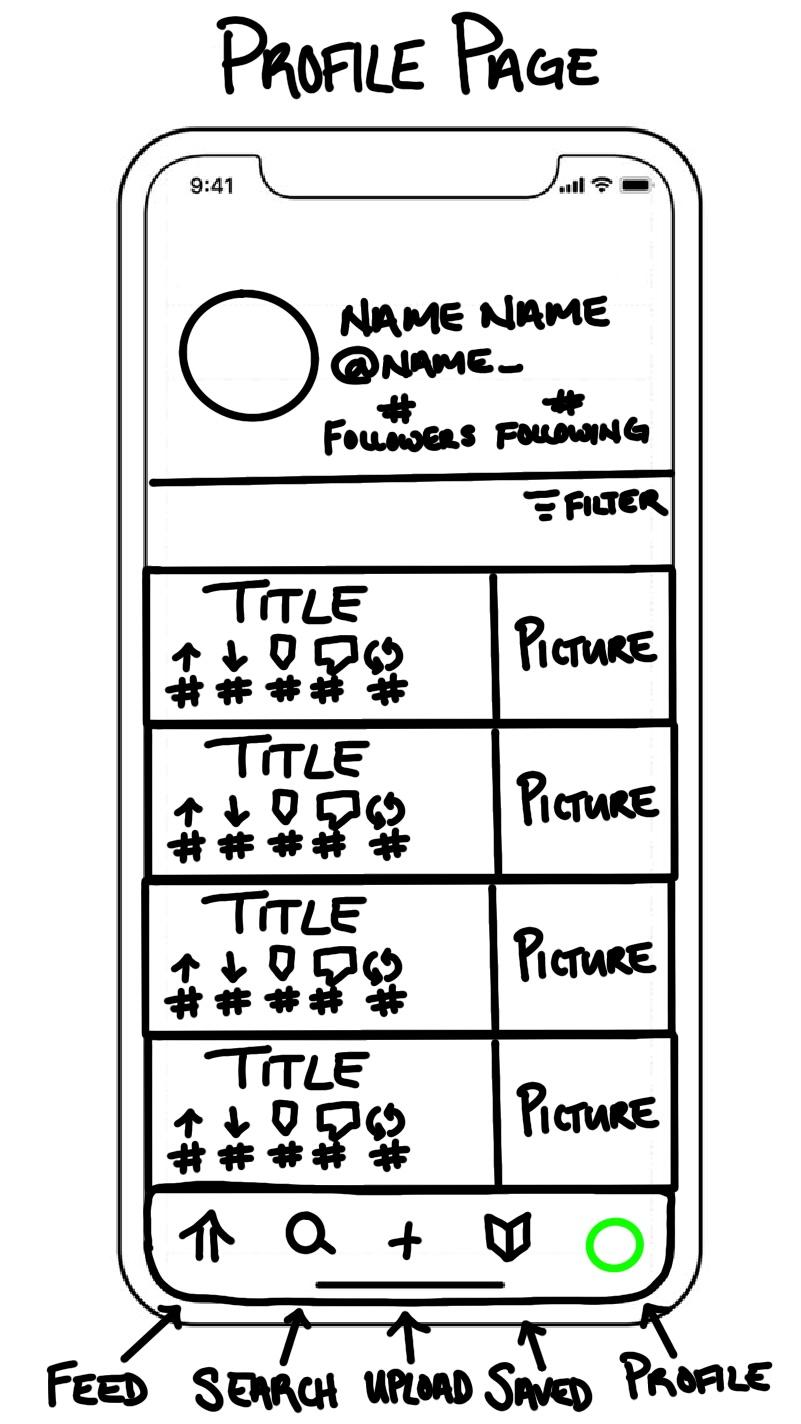
*Figure 3: Upload page wireframe*

The saved page of the app will include two dropdown menus: Meal Prep and Lists. The meal prep drop down will allow users to add saved recipes to a weekly calendar. The lists drop down allows users to sort their saved recipes to different lists. This function allows users to quickly find recipes that they had previously saved and organize them based on their liking.



*Figure 4: Saved page wireframe*

Finally, each user has their own profile page. The profile page consists of the recipes that the user has uploaded. This page functions as an easy way for users to access their personal list of recipes, making it quick and easy for the user to find and remake recent meals. Further, each recipe on the user's profile has icons indicating the number of up votes, down votes, saves, comments, as well as reused. The reused statistic is only visible on the recipe owner's profile, allowing the user to see how many people have tried their recipe.



*Figure 5: Profile page wireframe*

# Backend

The backend will implement a layered architecture using .Net 7, entity framework, and SQL Server. It will consist of 3 layers as defined below.

*API*: This layer will act as the entry point into the backend. When the frontend sends a request, it will hit the corresponding controller which will then route it to the next layer. Configuration, logging setup, auth, etc will also be handled in this layer.

*Business*: The business layer will handle any logic, data manipulation, or specific business functionality that should be kept away from the API layer.

*Data Access:* This layer is solely responsible for communicating with the database. This is where we can find all of the SQL queries in their corresponding repositories, our raw data models, and config for our ORM.

*Shared (Not a layer):* This will act as a bridge between all of the above layers. This is where our filter objects (request) and DTOs (response) will reside.

NOTE: Each layer is dependent on its downstream neighbor. That being said, the API layer should not know about the Data Access layer (dependency is not transitive).