InBeta

Pantry
Software Design Specification

Cole Pendergraft, Colton Lobdell, Piper Sheldon Young, Ethan Pressley, Joe Cates

Table of Contents

1. System Overview

According to feedingamerica.org, approximately 40% of all food produced in America ends up wasted, which amounts to a staggering 108 billion pounds of food waste annually. As it stands, there are no applications or software solutions with sights set on this problem specifically. There are applications such as Yummly which provide individual users with very basic means of tracking the expiration dates of their food, but this feature is an afterthought, there are incredibly few applications purpose-built to directly tackle the issue of food waste in America.

This is where Pantry comes in, the first inventory management system purpose built to aid both individual Americans and large food-buying organizations with tracking food expiration dates and preventing food waste. Through Pantry, users will be able to easily track their food expiration dates and receive advice on how best to preserve or use food about to go bad. Further, the app will feature a Shopping List tool that communicates directly with the user's pantry, adding selected items into the shopping list automatically when they go bad or are used.

Pantry consists of 8 primary modules that are broken up into two categories, those that the user sees and interacts with directly and those that the user does not directly interface with. The user side is composed of 5 of the 8 modules, the Pantry List which contains the information about what products the user currently owns and the expected expiration date of each of these items, the Shopping List which contains both automatically-entered and manually-entered food items to be purchased by the user on their next shopping trip, the Menu/Navigation which acts as the transition point between the Shopping List, Pantry List, and the Settings screen, the User Input which is used to populate the Pantry List and the Shopping List, and the Search/Sort/Filter module which enables the user to quickly find specific items stored in either list. The components hidden to the user compose the other 3 modules, which are the Notifications System that works to reach out to the user through their phone's notification system and inform them about foods that are about to go bad, the Settings System which enables the user to edit various elements about how the system as a whole functions, and a database to store all the Shopping List and Pantry List information.

2. Software Architecture

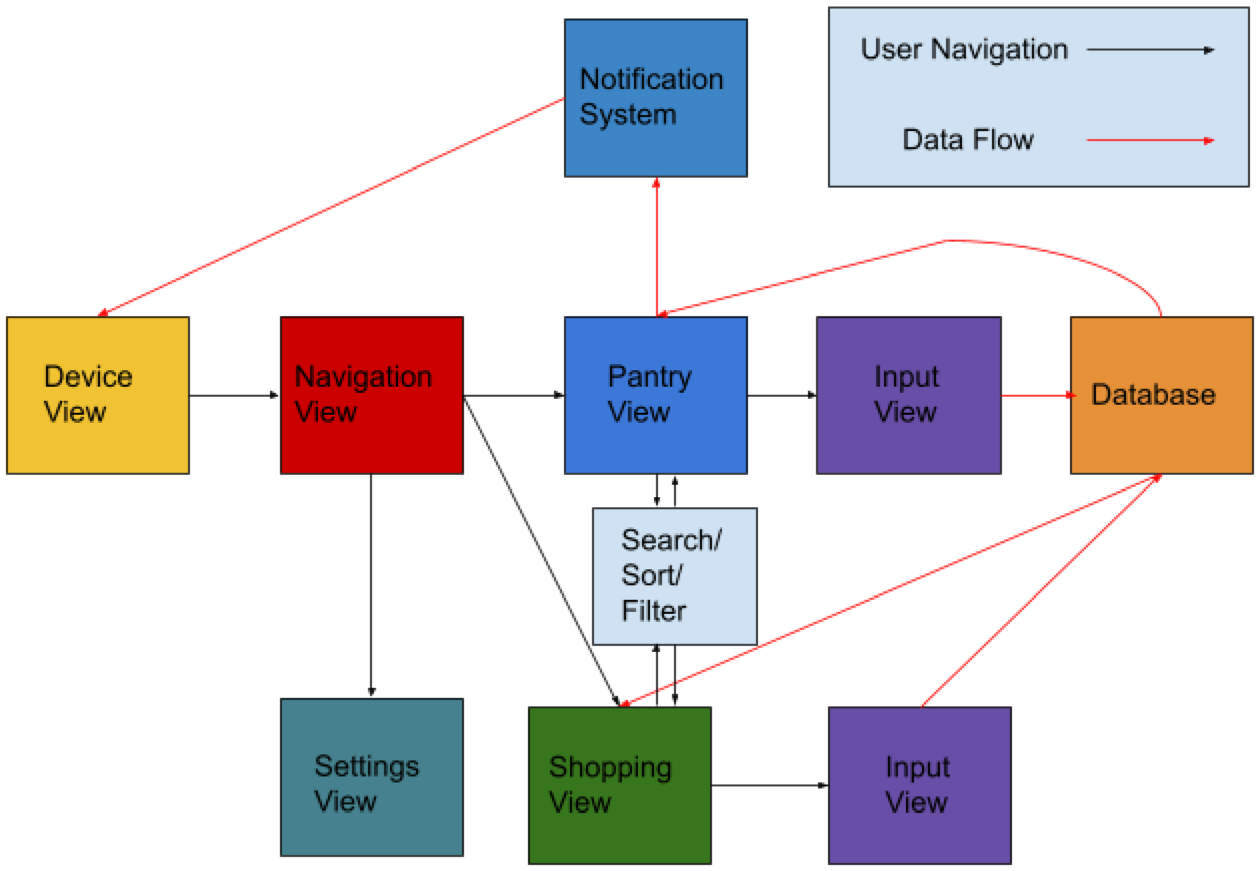
2.1 Components and their Functionality

- **User Side (Seen by user)**
  - Pantry List
    - The Pantry List stores a log of all the food items currently inside the user's pantry and the best-by date associated with each food item.
  - Shopping List
    - The Shopping List contains two distinct types of food objects, those that the user has manually written into their shopping list, and the elements

        - within the Pantry List that the user has selected to automatically populate the shopping list when that item expires or is consumed.
    - Input
        - The Input module contains all the functionality that enables the user to enter in food items into either their Pantry List or their Shopping List.
    - Search/Sort/Filter
        - The Search/Sort/Filter module enables the user to rapidly access items stored in either their Pantry List or their Shopping List either by searching it up directly or employing filters or sorting tags that restrict the elements displayed.
    - Menu/Navigation
        - The Menu module handles transitioning between the different model views, Pantry List, Shopping List, and Setting. The Menu will be accessible from anywhere in the app through a button that opens and closes the view.

- **Hidden Side (not seen by user)**
    - Notifications
        - The Notification System will be designed to communicate directly with the Pantry List, reading the associated best-by dates and notifying the user when something is about to go bad. Ideally, this notification will also include tips about how to extend the shelf life of the food item.
    - Settings
        - The Settings System will enable the user to interface directly with modular system features and change the application's operations to suit their liking. This will include such features as selecting when the user would prefer to receive notifications.
    - Database
        - A database with different tables to store relevant information. Examples of tables include a shopping list table and a pantry list table.

2.2 Architecture Diagram



2.3 Module Interaction

The face of Pantry is the Pantry List module which will be what is first seen when the app is opened. The Pantry List shows a categorized list of foods and products that are tracked by the Pantry app. Each element of the list will correspond to a number of backend database operations. Generally, each element of the list will correspond to a specific type of food or product that has its own identity within the database. From the Pantry List, users can change the number of a certain product they have stored and manually set values and characteristics about list items such as shelf life, behavior in the list, and adding new list items. All of these actions will communicate with the database on the backend to assure that what is displayed in the list corresponds with the values stored in the database. The Shopping List module will behave similarly, but by nature will serve a more temporary purpose. Items from the Pantry List can be added to the Shopping List, and even set to be automatically added when they have been used up. The Shopping List will also communicate with the database, allowing for easy transfer from Pantry to Shopping List and vice versa. The Input module will cooperate directly with both the Pantry and Shopping List modules, and will allow users to edit list items as well as create and insert new items.

The Settings module handles user modification of the app settings, such as how to handle automatic addition of items from the Pantry List to the Shopping List, or when app notifications will occur. These changes to settings will be stored in the database and affect the behavior of each module. The Menu module controls transitions between each module view of the Pantry app. The menu can be opened through a button available in any view, and holds a list of buttons that will transition the app to a new specified view. As mentioned, the Database module will actively communicate with each other module in the app. Items stored in the Pantry and Shopping lists will coordinate with similar entries in the database, and app settings will inform other modules how to update the database.

2.4 Rationale for Architecture

In order for this application to feasibly make an impact on food waste in the US it needs to be enticing to both the average consumer and larger organizations, and in response to this need the architecture was designed to prioritize user experience as well as functionality.

This architecture divides all modules into two distinct categories: the user side of things which encompasses anything that the user sees or interacts with directly, and the hidden side of the system that encompasses everything else. Since this is a mobile application, the vast majority of the work will be focused on creating an interactive environment that the user finds not only functional, but also easy to use. Most, if not all, the hidden functionality will serve to enable or improve the capabilities of these user-side elements, effectively improving user experience. This architecture places an emphasis on how the user interacts with the product which will enable the application to better address the needs and wants of consumers, ideally drawing the consumer in and encouraging use of the system. The more individuals who actively use the software system, the more of an impact the system will have on food waste in the US, so making the application easy to use and desirable is key.

2.5 Technologies Used

As it stands, Pantry is not going to require any external technologies in order to function. The application itself will be written in Kotlin through Android Studio, but will require no other dependencies in order to function. All database information will be stored locally on the user's phone so no external servers are needed, and all of the core functionality is self-contained so there is no need for any other technologies.

However, there will be a collection of communication-focused technologies used by the InBeta team in order to put together an optimal product. First and foremost is ClickUp, a tool to be used in facilitating a SCRUM based approach to building the application. Along with ClickUp, we also intend to use Google Drive as a document hub, Github as a source code hub, and Discord as a means of facilitating daily progress meetings to determine if we are on track during sprints.

3. Software Modules

3.1. Pantry List

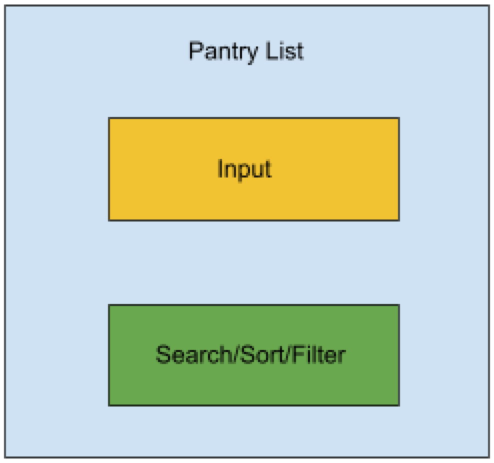
3.1.1. Primary role and function

The pantry list serves as one of the three primary points of communication between the application and the user. Accessible through the primary navigation menu, the pantry list will display a scrollable list of elements the user has entered into the pantry table of the database via the input module. In addition to being able to scroll through this list, the user will also have access to features that enable the list to be sorted or searched through in order to improve user access to information. The pantry list also serves to communicate with the notification module, notifying the user if an item in the pantry is about to go bad. Additionally, items in the pantry list that are one day off from the scheduled expiration date will be flagged to improve the speed at which users can access important information.

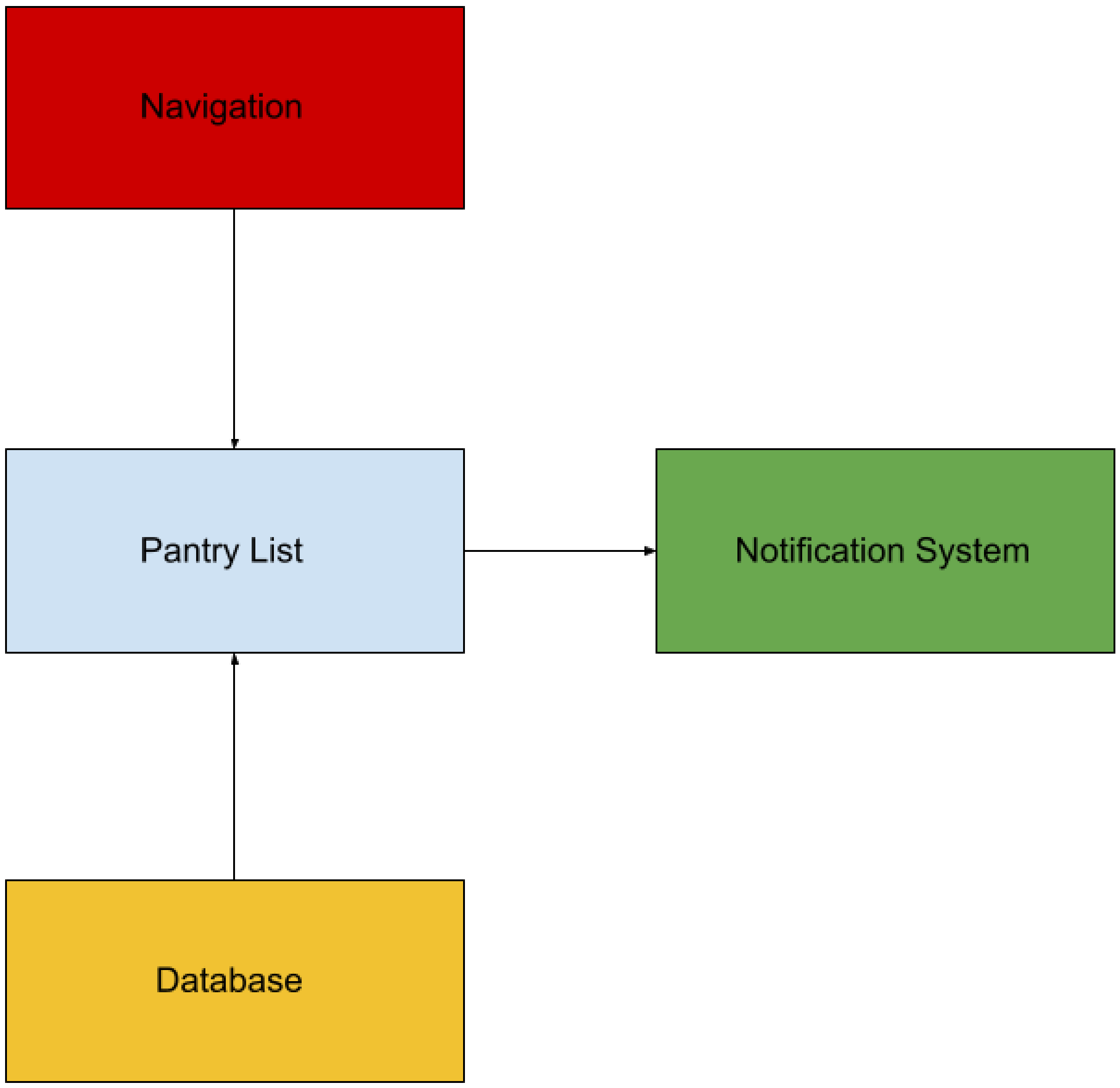
3.1.2. Interface

The pantry view will display the contents of the pantry table in the form of a scrollable list. At the top of this list will be a clickable "+" icon that brings up the input window, allowing users to input elements into the database directly from the pantry list. Next to this icon will be a "sort" option which, when selected, brings up a menu enabling the user to sort the list to only display elements matching a tag that is associated with the food element when it is input into the database. Next to this "sort" button will be a text bar that enables users to directly search up the item in question.

3.1.3. Static Model



3.1.4. Dynamic Model



3.1.5. Design Rationale

The pantry list would be designed to communicate the information of importance to the user in a simple and straightforward manner. This inspired the decision to use a list of items that can either be scrolled through, searched through, or sorted to make accessing information simple and require no explanation. The pantry list is the face of the application, and items within this list that are nearing expiration need to be displayed to the user first and foremost in order to keep them updated on the state of their inventory. This is why the pantry list, and only the pantry list, communicates with the application notification system to communicate expiration information to the user from outside the application.

3.2. Shopping List

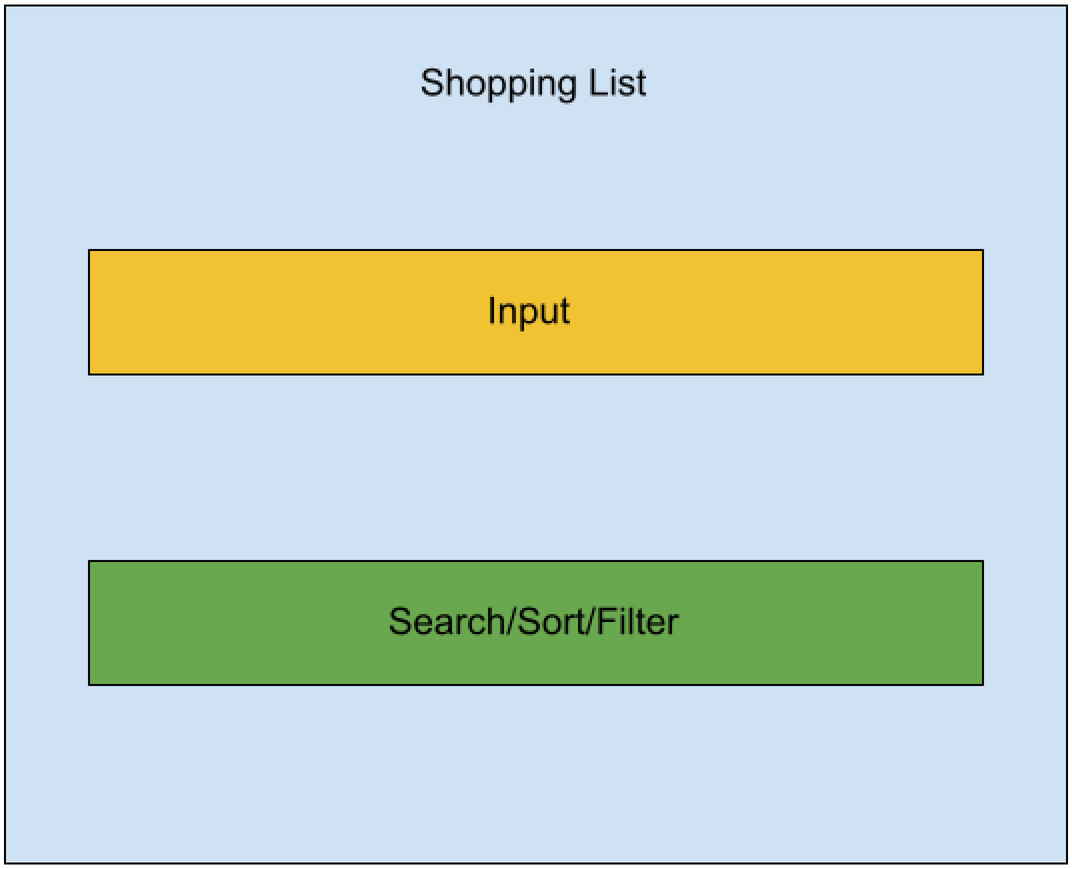
3.2.1. Primary role and function

The shopping list acts as the second primary point of contact between the application and the user. Like the pantry list, the shopping list will feature a scrollable list of food elements that have been added to the shopping table via the input module. The user will have the option to add elements to this list, as well as search through and sort it in much the same way as the pantry list. Further, the option exists to automatically populate this shopping list with food elements from the pantry list that have gone bad.

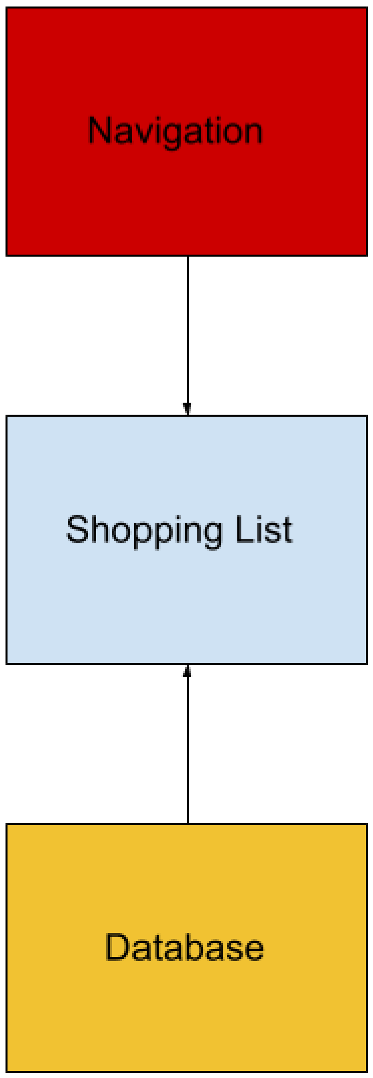
3.2.2. Interface

From an interface standpoint, the shopping view is identical to the pantry view. It too is a scrollable list of elements with associated categorization tags, and it will feature a "+" button that brings up the input module, a "sort" button which brings up a menu enabling users to sort the displayed list based on the tags, and a search bar which enables users to quickly find specific items.

3.2.3. Static Model



3.2.4. Dynamic Model



3.2.5. Design Rationale

The shopping list would be designed to communicate the information of importance to the user in a simple and straightforward manner. Like the pantry list, this inspired the decision to use a list of items that can either be scrolled through, searched through, or sorted to make accessing information simple and require no explanation. Unlike the pantry list, the shopping list does not need to communicate with the notification system as there is nothing that occurs in the shopping list the user needs to be aware of.

3.3. Input

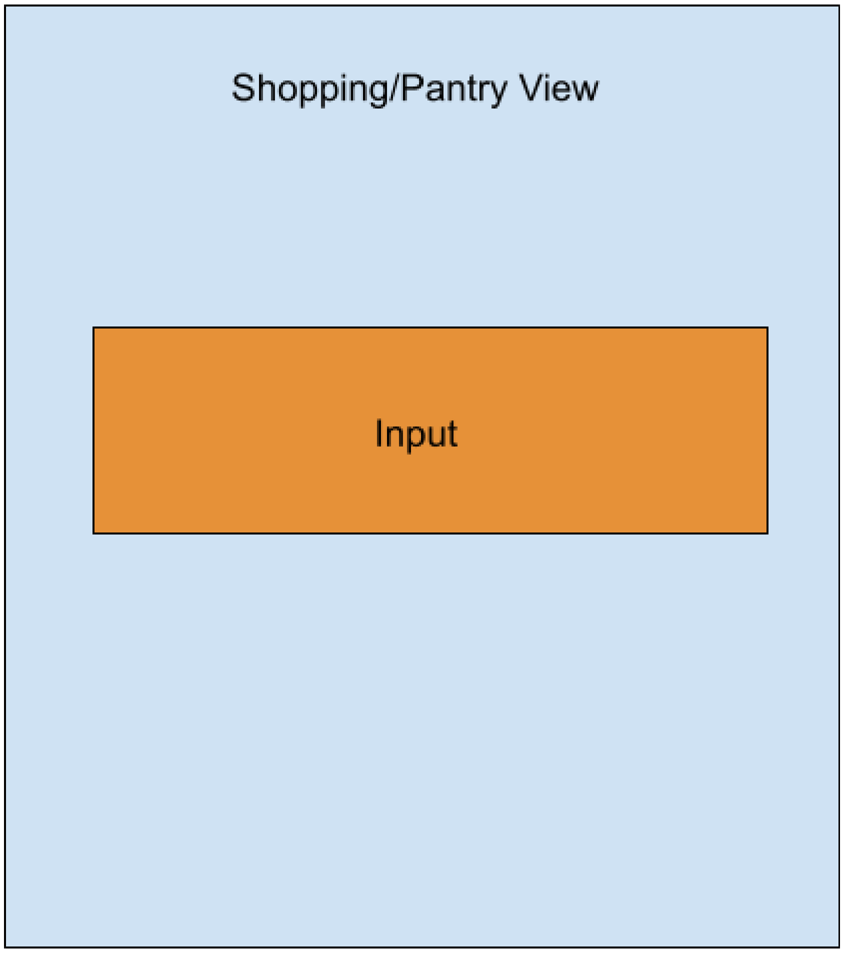
3.3.1. Primary role and function

The input module is associated with both the pantry list and the shopping list, and serves as the means through which users add elements into the respective list. The module is accessible via an icon on both of the two lists. As well as inserting food elements into the list itself, the input module also enables the user to associate “tags” with each item entered. These tags will be single-word descriptions of the food item such as “vegetable”, “meat”, or “poultry”. These tags will be utilized by the sorting functionality to enable users to sort the displayed list by tag.

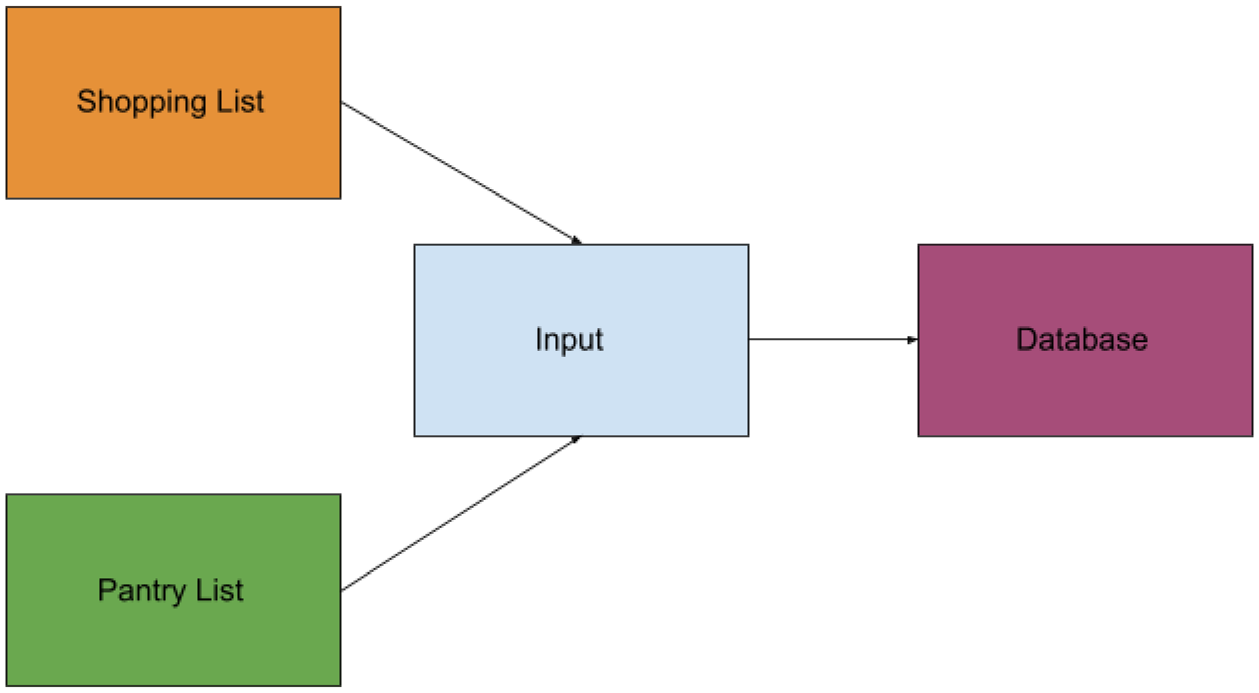
3.3.2. Interface

The input interface will be accessible by selecting the “+” button on one of the two lists. When clicked, the icon will open up a window that grants access to 3 different entry fields. First is a simple text entry field used to give the item a name. Below this will be a drop-down field enabling the user to associate pre-determined tags with the food item in question. Next to this drop down menu will be a date entry field for the user to specify the date upon which the item in question needs to be thrown away.

3.3.3. Static Model



3.3.4. Dynamic Model



3.3.5. Design Rationale

Like many other user-focused elements of the application, the input module is designed to be as simple as it possibly can be. Associated with each list is an icon with a "+" symbol, which will be recognizable to the user as the way in which elements are added to each list. The actual window will feature only 3 entry fields that are clearly titled so there is no confusion on the user side about how to add elements to each list.

3.4. Search/Sort/Filter

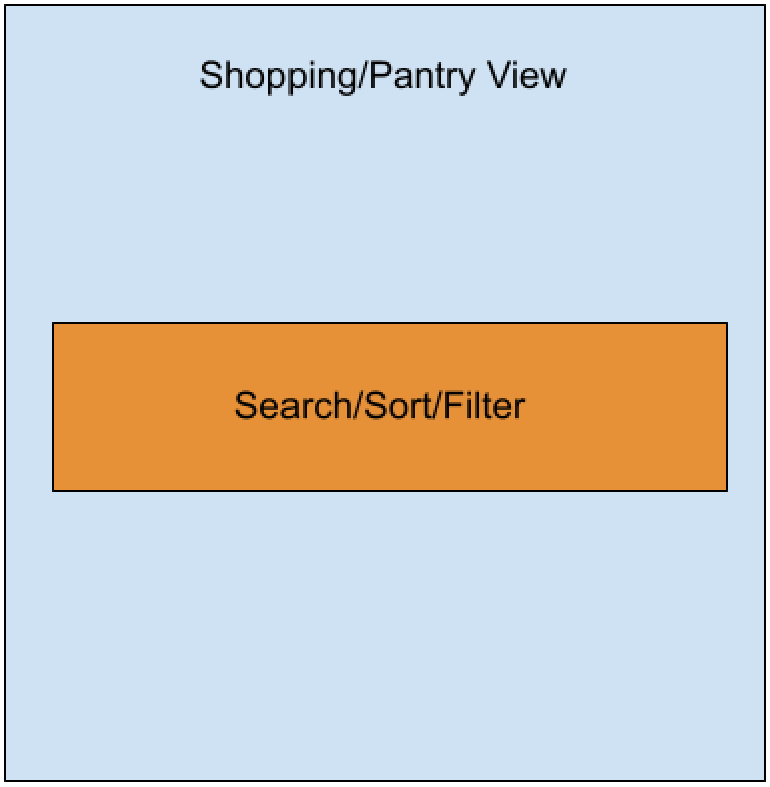
3.4.1. Primary role and function.

The Search/Sort/Filter module is associated with the two different lists much in the same way that the input module is. This module is directly accessible from each list view, and allows the user to decide what elements are displayed in the list by either using a text entry field to directly search up an exact item name or by sorting the list to only display elements with a given tag or tags. Additionally, the search/sort/filter module will enable the user to display food items nearing expiration to the pantry list.

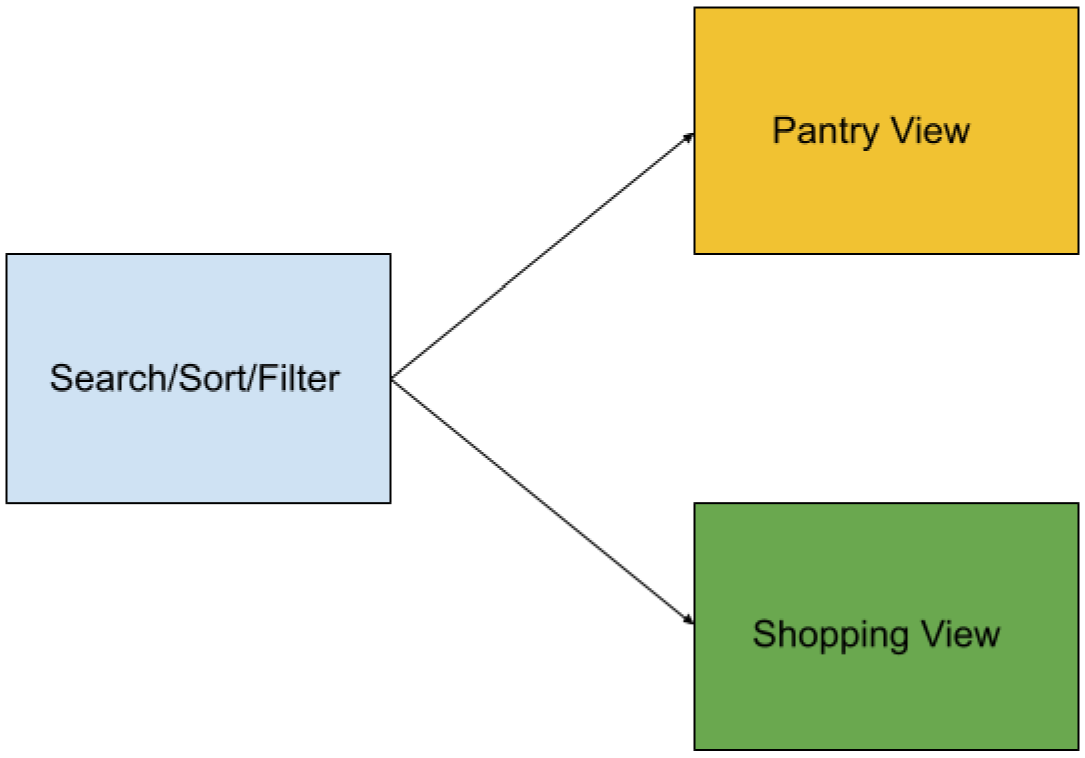
3.4.2. Interface

The interface for the search/sort/filter module is accessible through the two list views. Near the top of each list will be a button labeled "Sort" which, when clicked, opens up a window that allows users to select the tags that they would like to sort the list by, as well as an option to remove all currently selected tags. Upon exiting this window, the list will only display food items with at least one of the provided tags. Next to this button will be the text entry field through which the user can input specific food item names. When a name is searched, the list will display only items with names matching the search parameters without considering capitalization of letters. If an item not in the list is provided, the list will simply become empty.

3.4.3. Static Model



3.4.4. Dynamic Model



3.4.5. Design Rationale

The search/sort/filter module will be designed in order to improve the user's access to information displayed by the app, as scrolling a list of what could be hundreds of food items is not an ideal means of finding specific items. The module will feature multiple means of searching the list in order to provide functionality for different situations. The user will be capable of looking up particular items by name or sorting the list by specific categories depending on their need.

3.5. Menu/Navigation

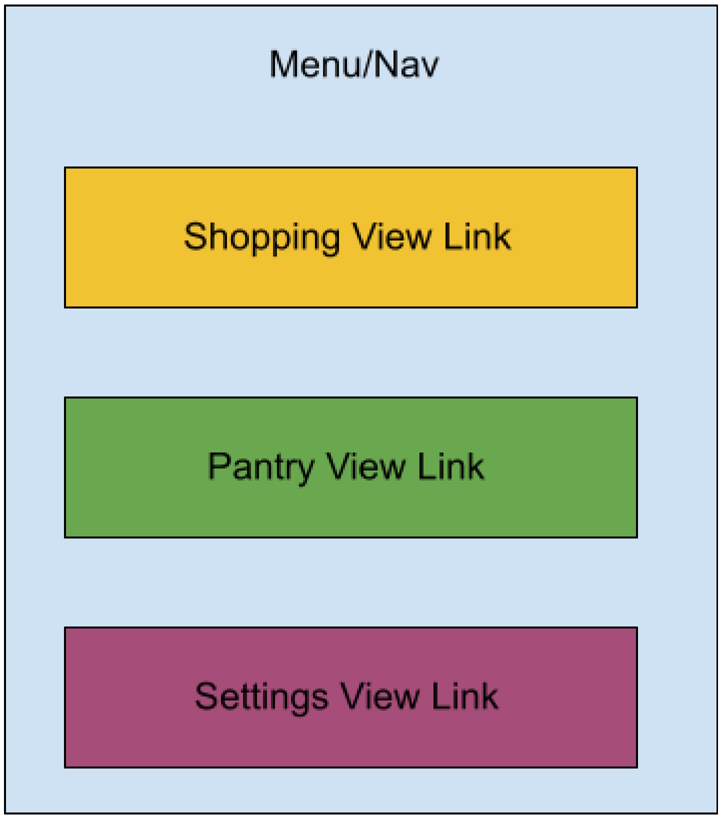
3.5.1. Primary role and function

The navigation menu provides the user with access to all other views in the app. From this navigation menu, the user will be able to select if they would like to transition from the current view to the shopping list view, the pantry list view, or the settings view.

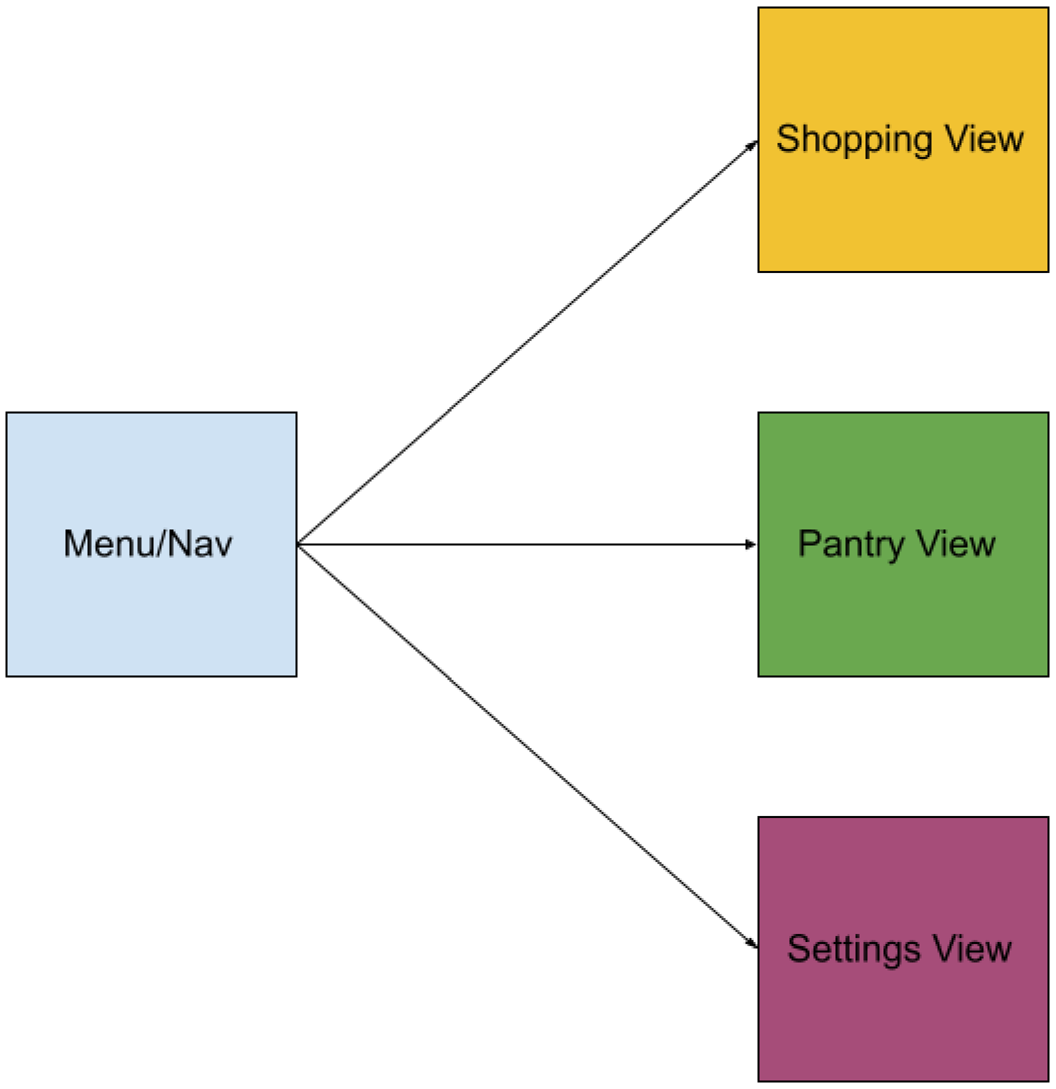
3.5.2. Interface

The navigation menu will take the form of a sliding "drawer" available to the user by clicking an icon in the top left of the application. Upon clicking this icon, the menu itself will slide out from the left side of the screen, displaying the 3 different view options for the user to navigate to. Clicking the icon again will cause the menu to retract back into the left side of the screen.

3.5.3. Static Model



3.5.4. Dynamic Model



3.5.5. Design Rationale

Any decent application has a centralized navigation system, a "home" for the user to return to that enables them to access all other elements of the application in an easy way. The menu/navigation module does just that for Pantry, it acts as the transition point between the current view and any other view the user wishes to access. Accessing these alternative views is a very simple process and will be familiar to any who have used applications with menus in the past, which will reduce the amount of instruction the user requires before the application can be properly used.

3.6. Notification System

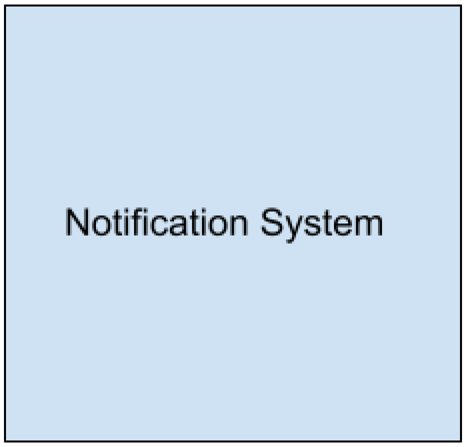
3.6.1. Primary role and function

The notification system serves as the third primary means of communication between the application and the user. The notification system interfaces directly with the pantry list in order to provide the user with push notifications when an item falls within a user-specified period time prior to expiration.

3.6.2. Interface

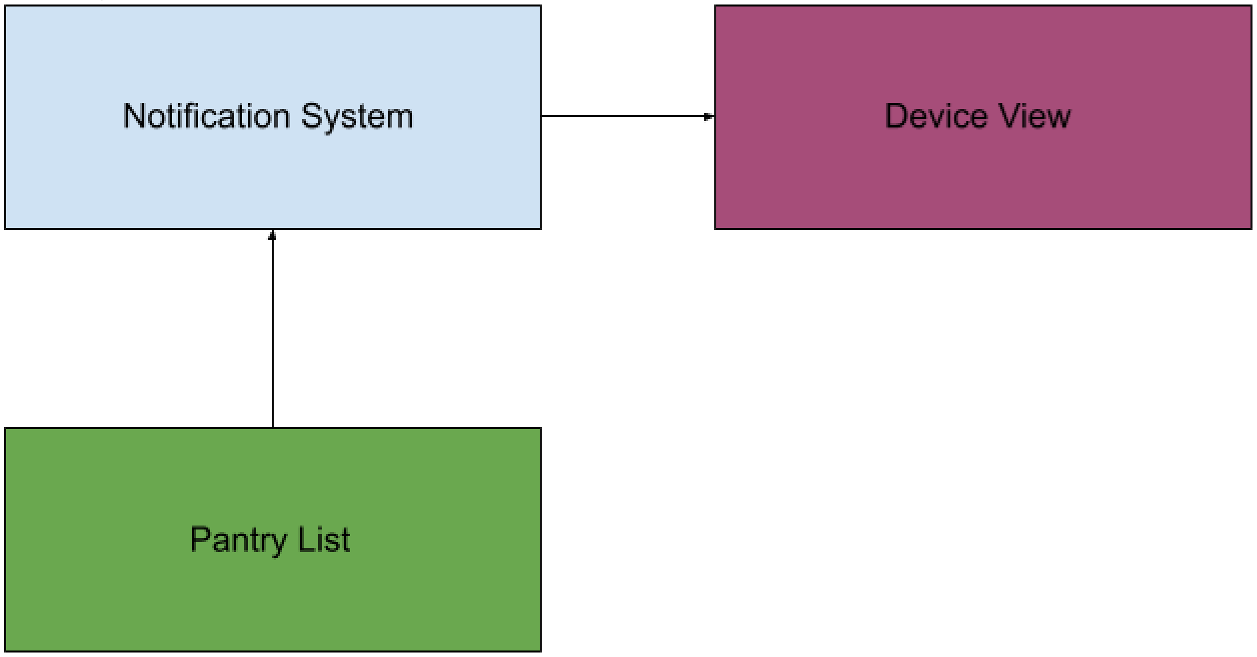
Not applicable, the user should have no direct interaction with the notification system.

3.6.3. Static Model



** The Notification System does not have any internal components to reference so it has an isolated static diagram.*

3.6.4. Dynamic Model



3.6.5. Design Rationale

The core of Pantry is its ability to aid the user in tracking food expiration dates, and in order to adequately keep the user informed a notification system that operates when the app runs in the background was an absolute necessity. The notification system is the means by which the user will be actively notified about items that are nearing expiration, and it will be up to the user as to how far in advance and how many times the system notifies them of the food item in question. The system will send push notifications to the user device containing the name of the item in question and the remaining time until said item expires, enabling the user to make a decision about how best to utilize the item before it goes bad.

3.7. Settings

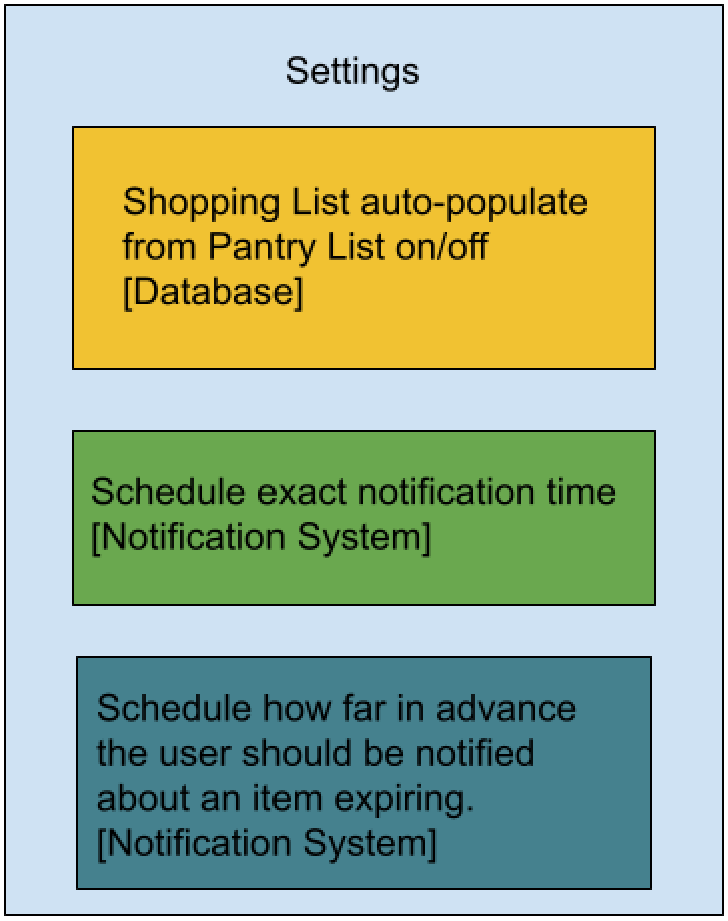
3.7.1. Primary role and function

The settings view enables the user to customize various aspects of their experience of the application. Currently, there are three main settings the user will have access to. First is the option to automatically populate the shopping list from expired pantry list items. It is common in the food service industry to routinely buy the same goods on a cycle, so it stands to reason that when an item currently in storage goes bad or is used it will need to be replaced with the same item, and shopping list auto population will immediately add items that have expired or have been used into the shopping list so the user is aware they need to order more. In an individual's home, however, it is more common to buy different weekly ingredients depending on the meals to be cooked that week, so it would be less useful for the shopping list to fill itself with items that have expired in the pantry list. The second setting is the option to determine how far in advance the notification system will notify the user of food nearing expiration, with the included option to specify multiple time periods. Meaning, a user could specify the notification system to notify them both 1 week or 3 days prior to expiration. The last setting the user has control over is the exact time of day that notifications will be sent. These choices operate in five minute intervals so a user can select times such as 3:30pm or 10:55am.

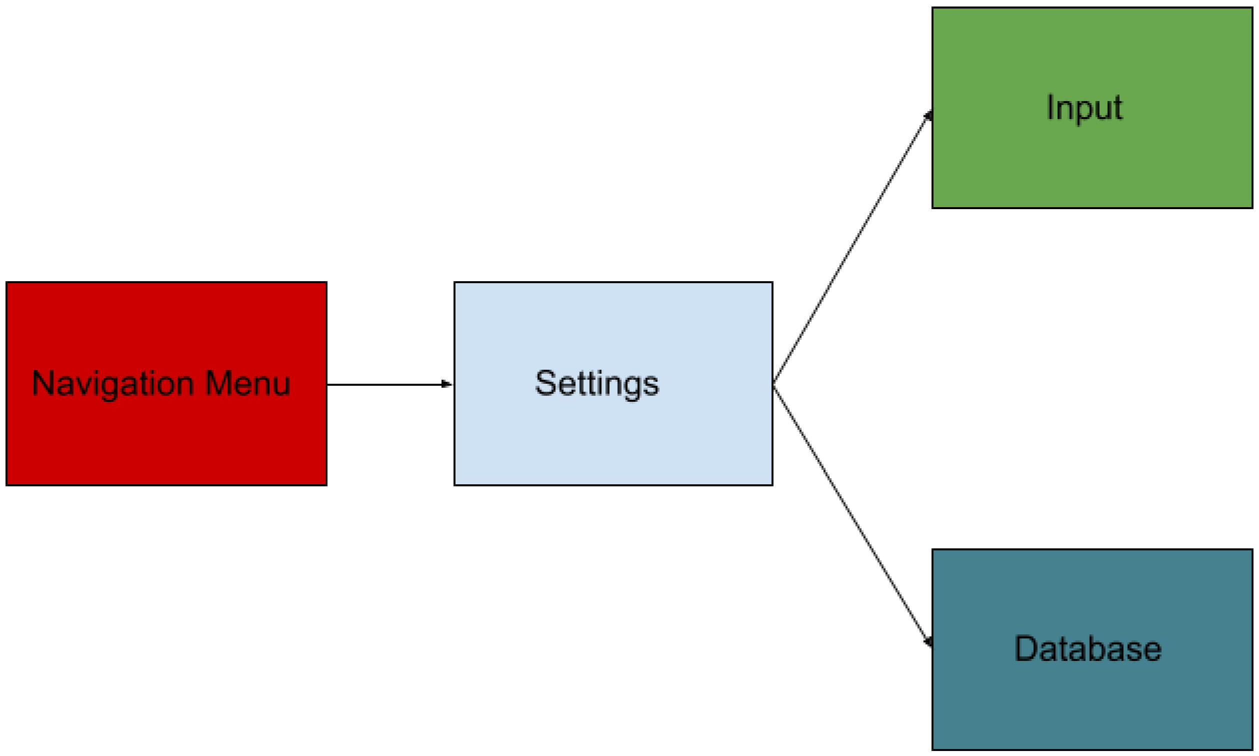
3.7.2. Interface

The settings view will feature one on/off switch attached to the title "Auto Populate Shopping List". Turning this switch on or off will in turn cause the respective setting to be turned on or off. Below this will be the text "Notification Period" followed by a drop down menu with various time periods such as "1 day" and "1 week". Lastly is another title "Notification time", which and a button labeled "Pick Time" which opens a view with a clock that the user can interact with, or if the user prefers the view can be switched to a text based view.

3.7.3. Static Model


Settings
Shopping List auto-populate from Pantry List on/off [Database]
Schedule exact notification time [Notification System]
Schedule how far in advance the user should be notified about an item expiring. [Notification System]

3.7.4. Dynamic Model



3.7.5. Design Rationale

Pantry is designed with restaurants and large food-buying organizations as the priority, but that doesn't mean it isn't a tool that could benefit individual consumers as well. That said, the individual consumer and larger enterprises have very different needs when it comes to the application, so a settings menu is needed in order to enable the user to customize app functionality to their liking. With the settings menu, the application becomes capable of satisfying two different classes of user without needing multiple versions by allowing for changes to how the shopping list receives information from the pantry list and limiting the amount of specific data that needs to be entered for individual users who don't want to take the time to add custom expiration dates to each food item. The settings view will be very accessible to every class of user, with each setting being very clearly titled and associated with a button in order to reduce the amount of instruction that the user needs to use the application.

3.8. Database

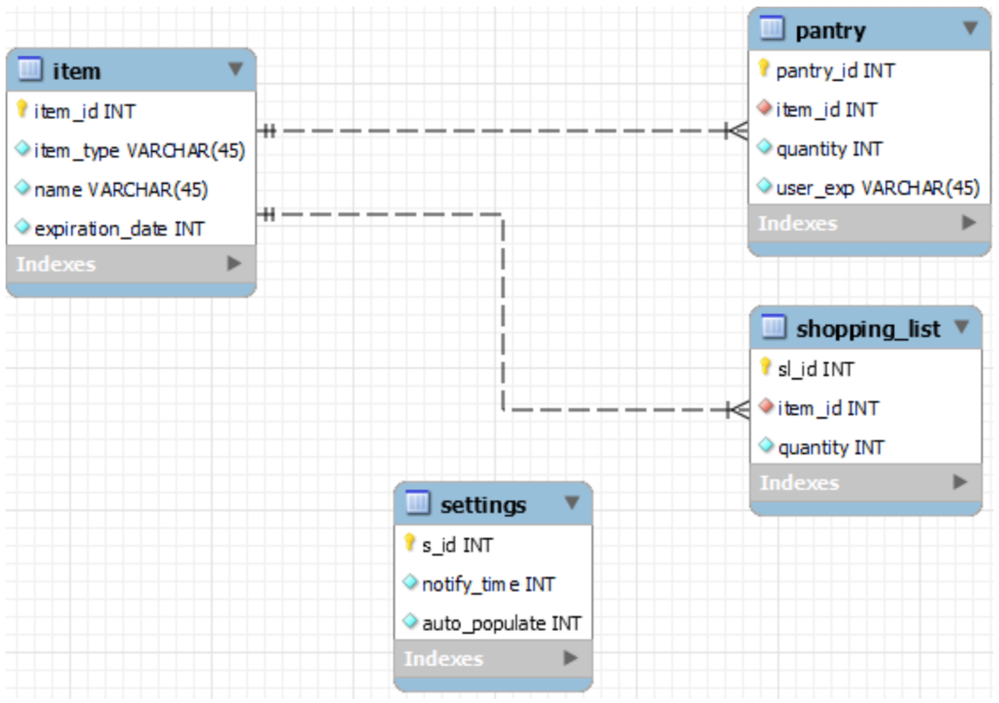
3.8.1. Primary role and function

The database operates to store all relevant data to be displayed in the pantry and shopping list views. The database interfaces directly with the input module, which acts as the only means to enter elements into the database.

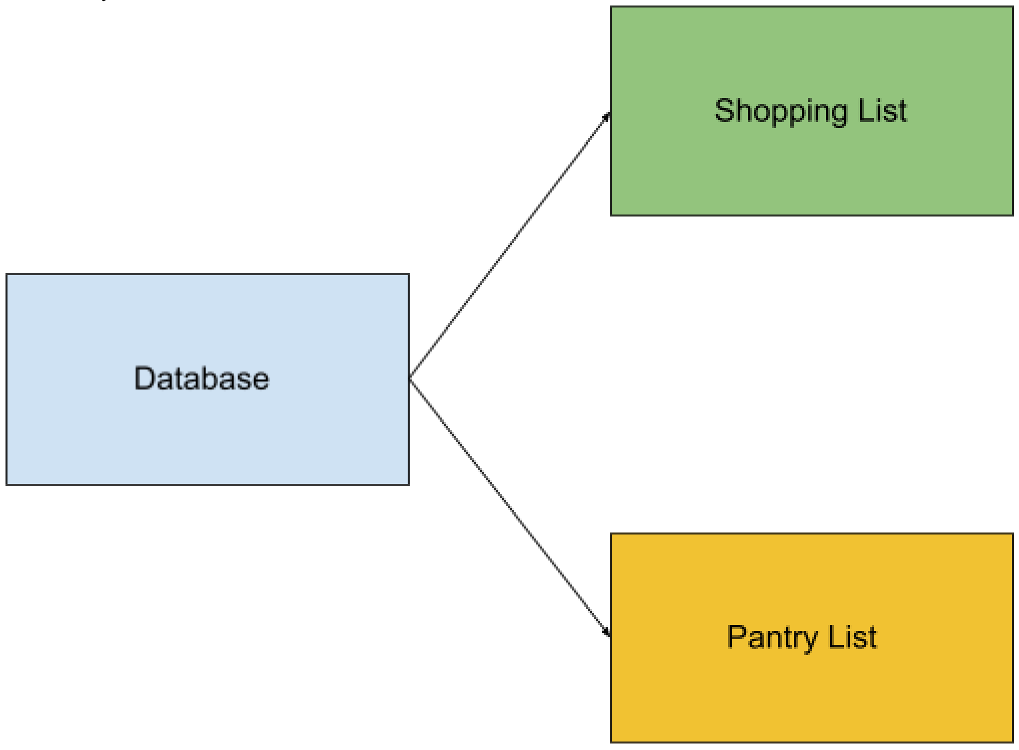
3.8.2. Interface

Not applicable, the user should have no direct interaction with the database.

3.8.3. Static Model



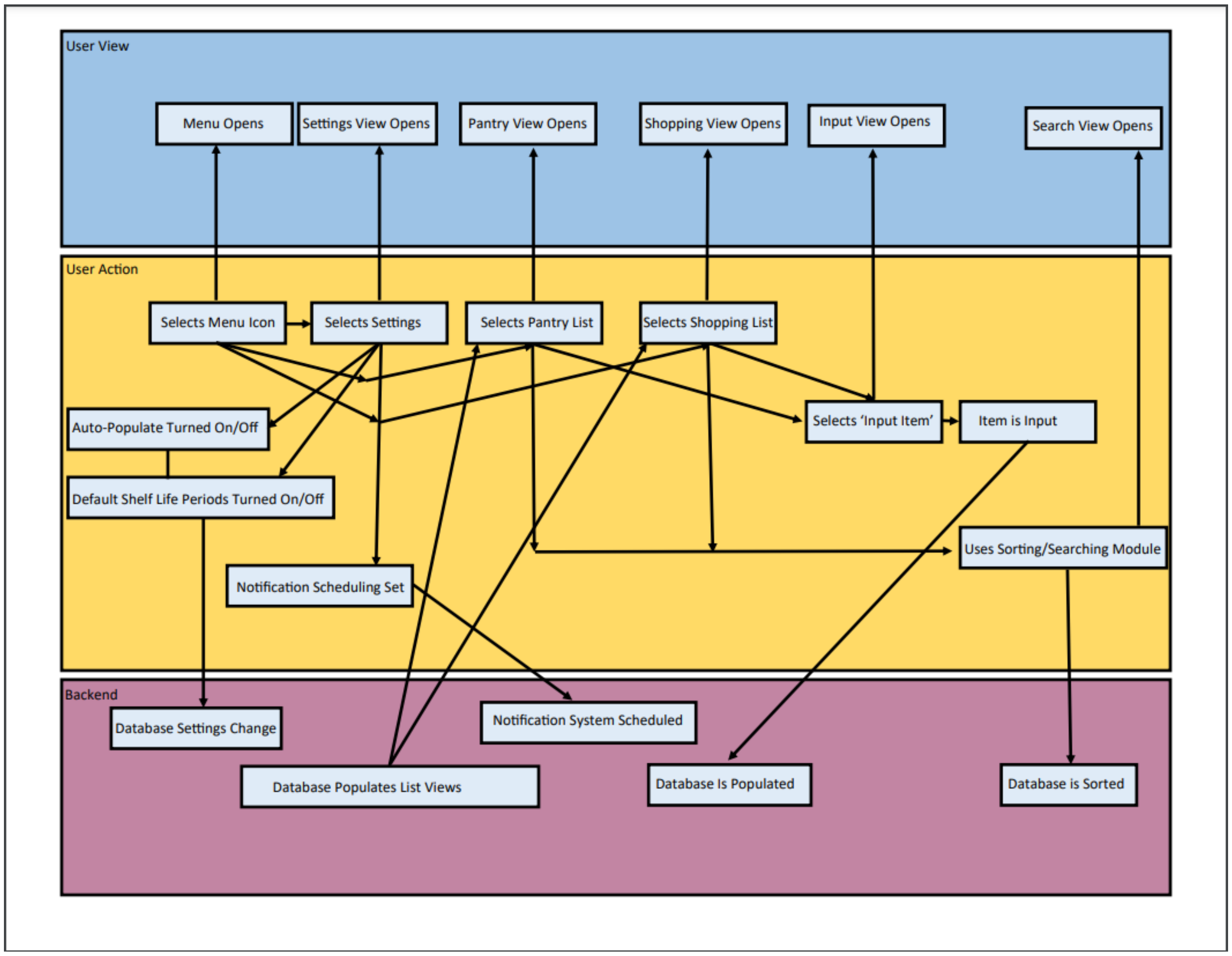
3.8.4. Dynamic Model



3.8.5. Design Rationale

Pantry stores a lot of custom information entered by the user, so a database will be a necessary component of Pantry's overall functionality. The database itself will be sectioned into three distinct tables to make the storing of information more straightforward, and these tables will communicate with each other to create additional functionality such as populating the shopping list with expired pantry list items. Sectioning off tables in this way will improve the ability of the InBeta team to build functionality that efficiently accesses the information stored in the database.

4. Dynamic Models of Operational Scenarios (Use Cases)



5. References

Data on Food Waste in America
https://www.feedingamerica.org/

6. Acknowledgements

We would like to acknowledge and thank Professor Juan Flores, Aktilek Zhumadil, and our fellow CIS 422 peers for being gracious, supportive, and insightful clients throughout the course of developing this product.