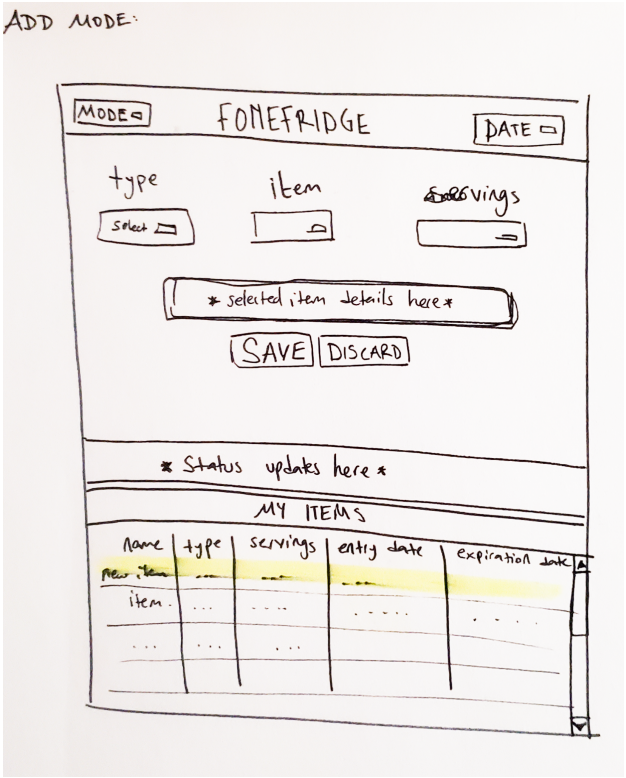
FONEFRIDGE DEVELOPER'S DOC

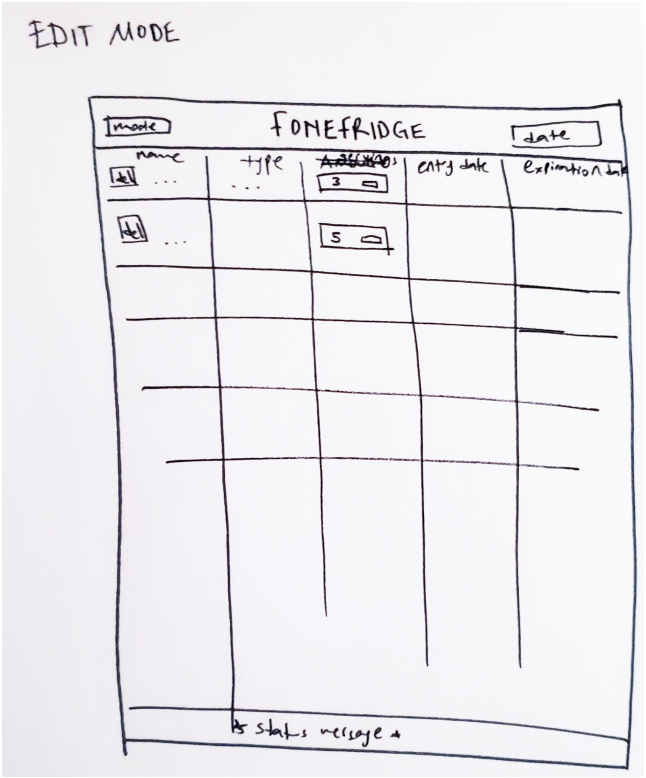
Repo: https://github.com/Pinanana/FoneFridge_HCI584.git (add_mode_run.py OR edit_mode_run.py)

Overview

Food waste has been a huge issue for climate change and budgeting. However there are some things we can do about it individually. Many people (as far as my questionnaire in the fall 2020 semester showed) struggle to keep track of their food stock when they are busy and they have little time to cook for themselves. As a result, many people throw away food items that they bought with the hope to cook them throughout the week but end up rotting in the deep corners of their fridges. For this problem, this product notifies people before the expiration date of the food items so that they can plan their meals accordingly.

Specs Highlight





Overall, the design has not been changed drastically from these wireframes. The main change is in the edit mode. Initial design had delete buttons next to items and dropdown menus inside treeview cells. But it was not possible to put other widgets inside treeview cells. So, this version shows those widgets when an item is double clicked.

Please check https://docs.google.com/document/d/1IlilmCk78RWJlrdzTOD4b8Mm8ZHxR1qIYWS111JakjI/edit?usp=sharing for the full project specifications document.

UI Flow

1. The user puts the new food items in the fridge.
2. Then the user enters the items in the FoneFridge desktop application, by running `add_mode_run.py` file. (`edit_mode_run.py` file is the edit mode screen.)
3. In the add mode, the user first selects the date from the calendar widget so that they are presented with the notification that shows expired and about to expire items before entering anything.
4. User searches for the item type.
5. After selecting a type, the item name dropdown shows the items from that type and the user can select an item name.
6. The user selects the serving count of their item from the servings dropdown menu.
7. Once the user enters an amount, the entry is saved to the `user_items.csv` file when clicked on the “SAVE” button.
8. If the user is not happy with the item, they can change the dropdown selections or click the “DISCARD” button to empty all of the selections.
9. There are status messages right under save and discard buttons so if there is a problem with saving the item, the user is informed.
10. The user can see their items at the bottom of the window in add mode.
11. If the user wants to delete or change the servings amount, they can go to the edit mode and double click on the item.
12. Once double clicked, the bottom part of the window gets populated by delete button and amount changing dropdown.
13. The user gets pop ups when they want to delete or edit to prevent user error.

Code Flow

Add Mode (add_mode_run.py):

The add mode is a class called Fonefridge. It has change_mode, calendar_get, show_the _item, generate_item_dropdown, fact_check, add_to_item, save_item, update_treeview, clear_all, and notification_trigger functions.

__init__:
The __init__ function has all the graphic Tk functions. The canvas parts are separated in the code with comment lines and the part name.

General: In the general part the .csv files are called with the pandas .read_csv() function. Then the frame is packed.

Top Ribbon: This is the very top part of the GUI. It has a frame to hold everything together, the main title of the application, mode switching button (change_mode function), and

the calendar widget that works with the DateEntry() function from Tkinter (calendar_get function).

Add Mode Upper Half: In this part, all the adding functions are located. It has a frame that takes up half the window called frame_middle. In this frame there is a main title that has a warning at first but once a calendar entry is made it turns into “search item”. Underneath it are the three item searching dropdown menus that are created with Combobox() functions and their labels above them. The first one is food_type_dropdown and this one creates the food_names_dropdown values with the generate_item_dropdown function. The Food_names_dropdown is the second one and the last one is servings_dropdown. Right under the dropdown menus there is a button to preview a selected item called preview_button. That button triggers the show_the_item function that populates the labels inside the hidden frame called frame_results underneath the preview button. There are two buttons under the frame_results. One of them is the save_button that saves the entry by first going through the fact_check function so that the user can fix any missing information. The other button is the discard_button that clears out the dropdown menus with clear_all function. Finally, at the very bottom of the frame_middle we have the message_box that has a label to communicate with the user about the status and errors.

Add Mode Display Half: This is the lower half of the canvas. The frame called frame_bottom contains the “your items” title and the treeview. In the style aspect of the treeview widget, there were some problems at first. If you have the wrong Tk version, you may face the same issues. Please check the “known issue” section in this document (next section) for it. The treeview background is set to the canvas background. Then the column names and widths are defined frome line 200 to 214. After that the values are called from user_items.csv file and sorted out from most to least recent. Finally a vertical scrollbar is placed to the left side.

change_mode:
This function closes add_mode_run.py and runs edit_mode_run.py file.

calendar_get:
This function grabs the date selected from the DateEntry widget. Then it clears the messages requesting a date entry from the user. Finally it calls for notifications with notification_trigger function.

show_the_item:
This function previews a selected item. First it goes through an if loop to check if a name is selected. If no type or name is selected a message is printed. If there is a name selected, the item is found by pandas .query() function. Then the information about that item is printed in a humanly readable paragraph inside the frame_resuts’ labels.

generate_item_dropdown:
This function is triggered when a type is selected from food_type_dropdown. This function searches the popular_items_library.csv file with pandas .query() function and returns all the titles with that type in a list. Then this list becomes the values for the food_names_dropdown.

fact_check:
Saving an item is designed to be a two step process. This is the first step of saving. This function is basically an if loop that checks if the user gave all the data the next step needs to save an entry. If any of the information is missing, a warning message is provided in the message_box. If all the necessary information is provided, we continue with the save_item function to actually save the item to user_items.csv file.

save_item:
This is the second step where we actually commit to save the item the user wants to save. First the item from the popular_items_library.csv file is found by .query() with the selected item name. Then the expiration and notification spans from the library are used to calculate the actual dates with datetime.timedelta() function. Then a dictionary is created from the entries and the calculated expiration and notification dates. All this data is appended to the df_user copy of the data frame then it is written to user_items.csv with the .to_csv() function. Finally the treeview is updated with update_treeview function and the dropdown menus are cleared with clear_all function.

update_treeview:
After adding an item, the user items preview is no longer up to date. So this function adds the new item with a tag that highlights it. Since it is just to show, the row is not pulled out from the user_items.csv file but created with all the collected data and changing the entry_date format to string.

clear_all:
This function clears the dropdown menus by setting their values to nothing.

notification_trigger:
Once a date is selected from the DateEntry widget, this function is run. It locates all the items that are expired and which items are about to expire. Then shows them to the user in the frame_results area. To find the items the .loc[] function is used. It works with strings so the entry_date is converted to string. To get the items that are about to expire, first the items that have a notify date before or equals to entry_date are located. Then the expired items are located by comparing their expiration dates to entry_date. After getting both lists, a simple "not in" statement, the about to expire items are located. Finally the items are presented to the user in the middle section by configuring the labels.

Edit Mode (edit_mode_run.py):

The edit mode is a class called Edit with change_mode, edit_tools, activate_button, delete_button, change_amount_button, delete_item, change_item_incsv, update_treeview, close_1, and close_2 functions.

__init__:

In __init__ function all the GUI visual elements are placed. The canvas parts are separated in the code with comment lines and the part name.

General: It is the same as the add mode. In the general part the .csv files are called with the pandas .read_csv() function. Then the frame is packed.

Top Ribbon: This is the very top part of the GUI. It has a frame to hold everything together, the main title of the application, and a mode switching button (change_mode function). It doesn't have the calendar widget like the add mode so today is set to datetime.today().

Edit Body: This section is the main section of the edit mode. Very top of the frame_edit we have a title and a sub-title. Editing works with double clicking on items so the users are requested to doubleclick on an item. Under the titles, we find the treeview widget code. First the style things are placed. Same as add mode, please check the "known issue" section if there is a mismatch in Tk version. Under the style things, the treeview columns are created. The today is set to today so that the "expired", "notified" and "others" tags can work. After tag configurations the user_items.csv data is separated into expired, notified, and rest of the items so that the tags can be used in corresponding chunks. The expired items are located by comparing today to expiration dates. The notified items are located by first getting all the items that are notified then the expired items are taken out by pandas .concat() function. Finally, the rest of the items are located by comparing today to notified dates. After that we have a vertical scrollbar. Finally the treeview doubleclick action is binded to edit_tools function.

Edit Buttons: The bottom_frame is an empty frame with a label requesting from the user to doubleclick on an item. When an item is selected, this frame is populated with buttons and a dropdown menu.

change_mode:
This function closes edit_mode_run.py and runs add_mode_run.py file.

edit_tools:
This function populates the bottom_frame with buttons and a dropdown menu. This function also finds the selected item .selection() from the user_items.csv file by first .query() with the title then loc[] with the entry date of that selected item. Then stores the index number of that selected item in the index_select_number variable to use in other functions. The title that was requesting a selection is configured to "now editing … that added on … :" and the buttons are placed underneath ist. The delete_but triggers delete_button function. The serv_but triggers change_amount_button function but it's disablet at first. A serv_drop dropdown selection activates the serv_but with activate_button function.

activate_button:
Once an amount is selected, the state of serv_but is changed from "disabled" to "normal".

delete_button:
Since anyone can accidentally click on buttons, there is a user error preventing pop up called pop_up_del when clicked on the delete_but. It asks the user if they want to delete that item, presents a delete button that triggers delete_item function, and a cancel button that triggers close_1 function.

change_amount_button:
With the same user error prevention intention, once the user clicks on serv_but the pop_up_amount window appears. In this window the user is asked if they want to change the amount from the original amount to the new amount by getting the amount from treeview selection and Combobox selection. There is also an “OK” button that triggers change_amount_incsv function and “CANCEL” button that triggers close_2 function.

delete_item:
If the user clicks to the delete button in the pop_up_del window, the selected item is dropped from the pandas copy of the .csv file. Then the user_items.csv file is overwritten with the pandas to_csv() function. Since changing the .csv file is not going to change the treeview, the treeview is updated by the update_treeview function. Then the bottom_frame is cleared to the condition before a selection is made by deleting the buttons and changing the label to "Please double click on the item you want to edit.". Finally the pop up is closed.

change_item_incsv:
When the user clicks OK in the pop_up_amount window, this function is triggered. The amount is changed by locating the item by index_select_number and “amount” column. After changing the pandas copy, the changed data frame is overwritten to user_items.csv file with the .to_csv() function. After that, like in the delete_item function, the treeview is updated by the update_treeview function. Then the bottom_frame is cleared to the condition before a selection is made by deleting the buttons and changing the label to "Please double click on the item you want to edit.". Finally the pop up is closed.

update_treeview:
First the existing treeview items are cleared by .get_children() and .delete() functions. Then the data from user_items.csv file is separated into expired, notified, and rest of the items just like in the __init__ function. (The same code)

close_1:
If the user wants to keep their item, they click “CANCEL”. The pop_up_del window is closed and nothing is changed.

close_2:
If the user wants to keep their item, they click “CANCEL”. The pop_up_amount window is closed and nothing is changed.

Known Issue

If you don’t have Tk version 8.6.10 or higher, then the treeview style will not work properly. It is a minor issue in terms of functionality but it will look ugly. One way to fix that is to copy and paste this piece of code under __init__:

```
def fixed_map(self, e):
    return [elm for elm in self.style_tw.map('Treeview', query_opt=self) if elm[:2] !=
('!disabled', '!selected')]
```

Also copy this line and comment out the existing line with self.style_tw.map(... (line 191 in add_mode_run.py and line 72 in edit_mode_run.py).

```
self.style_tw.map('Treeview', foreground=self.fixed_map('foreground'),
background=self.fixed_map('background'))
```

WARNING! If you make these changes, the treeview will no longer highlight the selected line but the functionality will not be affected by these changes and the background will have coherent colors with the overall interface.

- If there are items with same name and same entry date in the user_items.csv file, there can be problems in the edit mode.

Future Work

This is a very basic application that can get complex with how much data we want to store. For example, the items don't need to be stored in the fridge but also in the freezer, cupboard, or pantry. This data can be added as a column in the popular items library and the expiration spans can change according to the location.

This application can also store the deleted items as a shopping list in the future.

The servings amount entry can be a two step entry with selecting the type of amount (weight, volume, or count also metric or imperial) then the number.

add_to_item:
This can be the middle step function of saving an item. If the item the user wants to save exists with the same entry date, the new item's servings amount is simply added to the existing one. First the items with the same name are found with the .query() function then the items with the same entry date are located with the .loc[] function. If the result of this search is not an empty data frame, then the new entry's servings amount (servings_dropdown.get()) is added to the existing one. Then this change is overwritten in the user_items.csv file. Then the treeview is updated but the user sees the new item as a separate entry. Then the dropdown menus are cleared with the clear_all function. If there is no such item, the save_item function runs. If the user adds the same item with the same entry date without closing the window, they are added two times. BUT the .mainloop() function prevents anything to be saved to the .csv file before closing the window. So the pop_up window is created and destroyed just to make it work. BUT that also didn't work. That can be a nice addition to the FoneFridge application in the future.