SDEV-435 Project Submission of Deliverables

Jessica Kovarik

11 AUG 2019

Champlain College

Jessica Kovarik

Dish Collection Tracker

SDEV-435 Project Proposal

11 AUG 2019

Champlain College



# Executive Summary

Vintage dishes are one popular collectible by some people. In fact, there are various online resources, virtual meeting places, and social media groups dedicated to collecting, selling, and learning about various lines of vintage dishes. Some collectors enjoy acquiring pieces related to a specific era or design, such as those who collect mid-century modern dishes. Others focus on collecting specific brands, such as collectors who own only Pyrex vintage dishes. And some collectors focus on a specific line made from a specific designer, such as someone who collects the “Charm” pattern from Anchor Hocking. For those collectors who collect pieces within a specific line, they often want to know the full product line that was produced. However, there are few places where a collector can obtain information about the pieces available while out shopping. An example of this desire can be seen on vintage Fiestaware and Pyrex social media sites, where users often inquire about if a piece seen at a store is vintage, actually made by the dish company they thought, or if the piece was produced in other colors. Although there are collector’s books and similar printed materials available documenting dish lines, users want information in the moment, while they are out shopping.

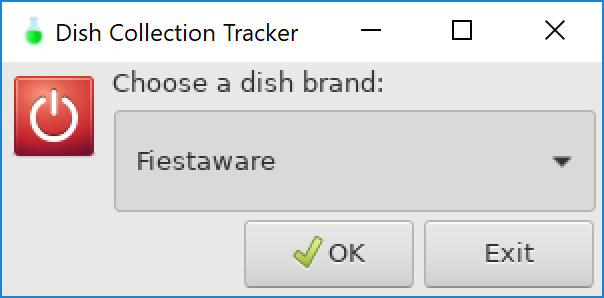
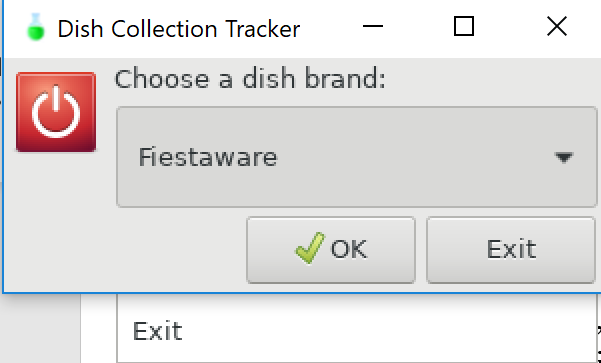
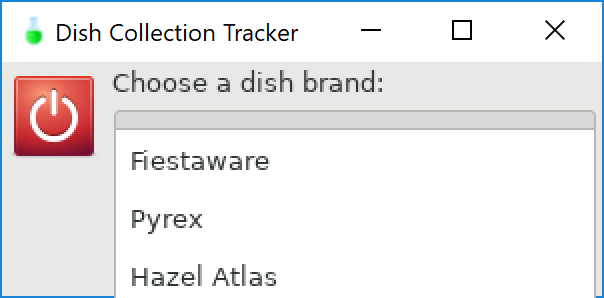
In addition to knowing the pieces available in a line of dishes, collectors often want a way to catalogue their personal collections. Sometimes it is difficult to remember how many teacup saucers one has when they are stacked in a display or stored away. Collectors also want a way to determine what pieces they do not have in their collection and are still searching for. In addition, being able to identify pieces while out shopping, without having to carry a book around, is also valuable to a collector.

Thus, Dish Collection Tracker will fill the need of users who collect vintage dishes. Dish Collection Tracker will be a program that provides vintage dish collectors a place to keep track of their personal collections by entering the pieces they currently own. Users can document piece history and notes about the piece. Users will also be able to look up all pieces available in a line of vintage dishes through a keyword search. Photos and text will show the full line of dishes, to help users identify pieces while out shopping. Dish Collection Tracker will first offer collection documentation for vintage Fiestaware and Pyrex. The information stored within the program will be accessible offline, which is key as some shopping venues have limited connectivity. Also, not every dish collector has a smart mobile device to access information online while shopping.

For this project, vintage dishes will refer to dishes that are no longer currently produced in the same size, color, and/or form. Vintage Fiestaware will be defined as Fiestaware produced in and after 1935 but before 1986 for this project by The Homer Laughlin China Company (The Homer Laughlin China Collector’s Association Guide, 2000). Dish Collection Tracker will include information about vintage milk glass Pyrex produced from 1947 to the 1980s (Vintage Pyrex 101: a guide to Pyrex, n.d.).

The demographics of target users for Dish Collection Tracker’s products are individuals who collect vintage dishes and use the internet and/or mobile devices. Intended users of these products may wish to either document their individual collections and/or learn about all pieces available in a dish line. Users may also want to determine which pieces they do not own in a specific collection of dishes. These users will be already be comfortable with technology or will be users interested in the use of technology to track vintage dish collections.

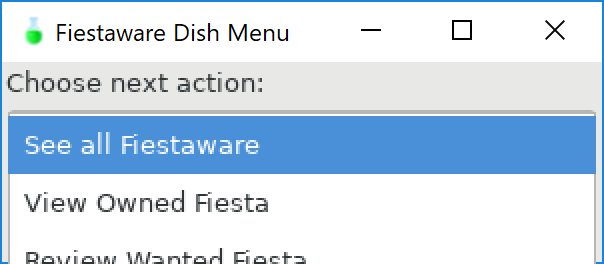
Start program | User starts Dish Collection Tracker

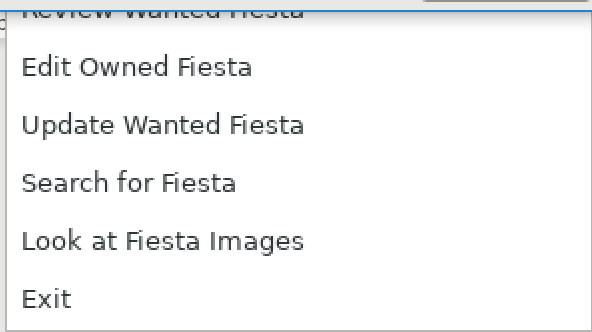
 

Once the program is launched, this is the first screen the user encounters. The user can choose from Fiestaware, Pyrex, and Hazel Atlas dish brands. User can exit using drop down menu (unable to capture Exit option in a screenshot) or “Exit” button in dialog. User selects “Ok” button or can hit “Enter” to proceed.

# Dish brand selected | Dish submenu presented to user

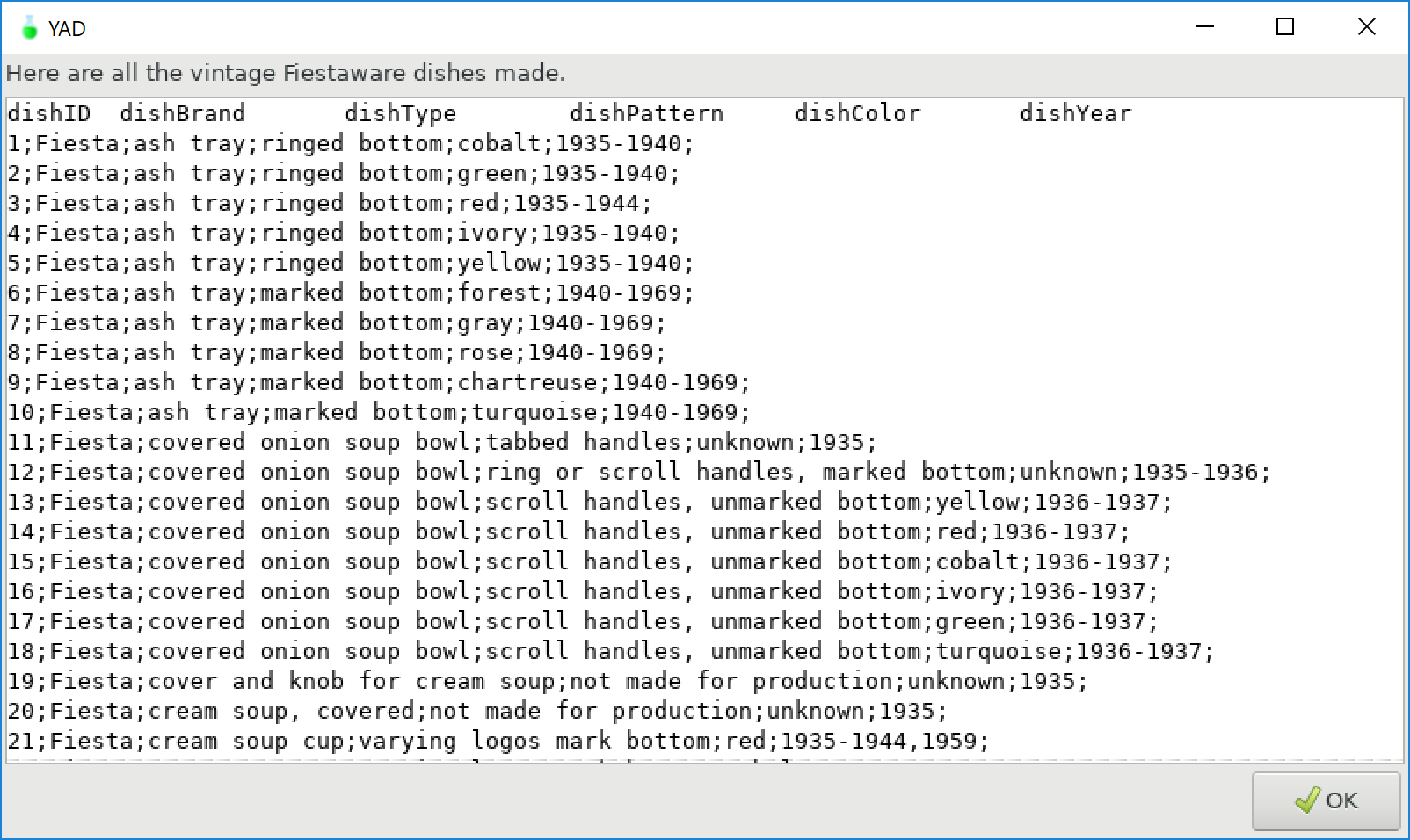
After the user selects the dish brand, the user is presented with the available actions (see all known dishes, view owned dishes, review wanted dishes, edit owned dishes, update wanted dishes, search for a piece, look at pictures, and exit).

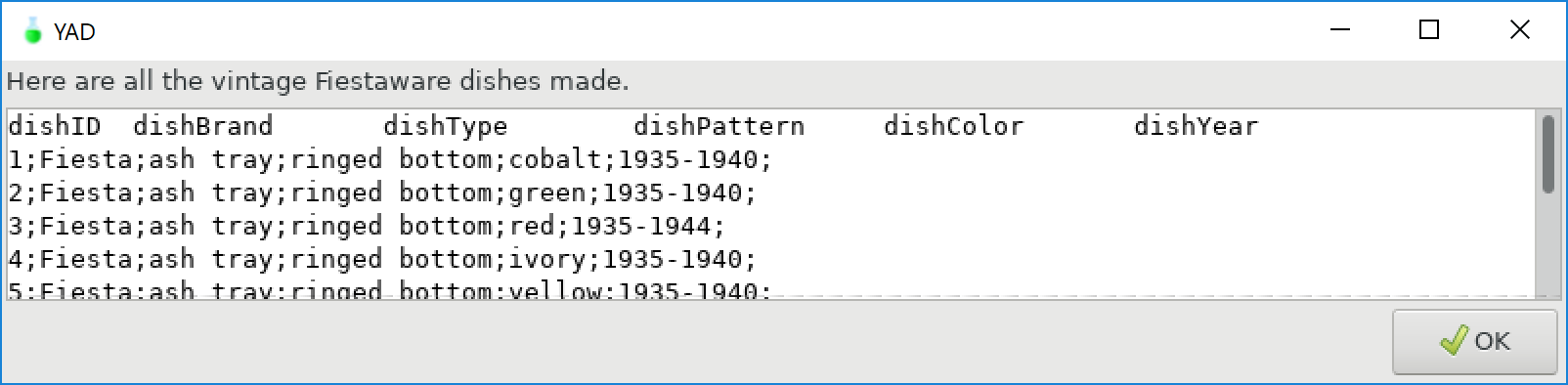


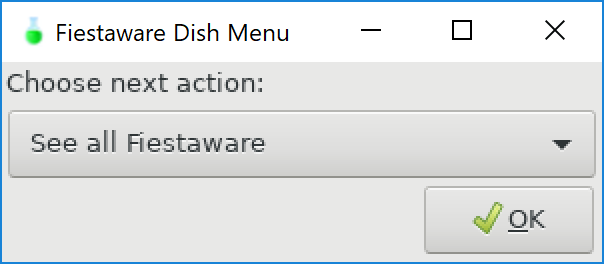


# See all dishes in a brand | Displays list of all known dishes

A text dialog is displayed with the dish brand the user selects. If the list is longer than the dialog size, a scrollbar appears. When “Ok” is selected, the user is brought back to “Choose next action:” menu.

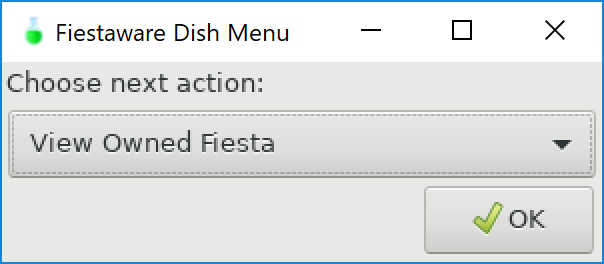
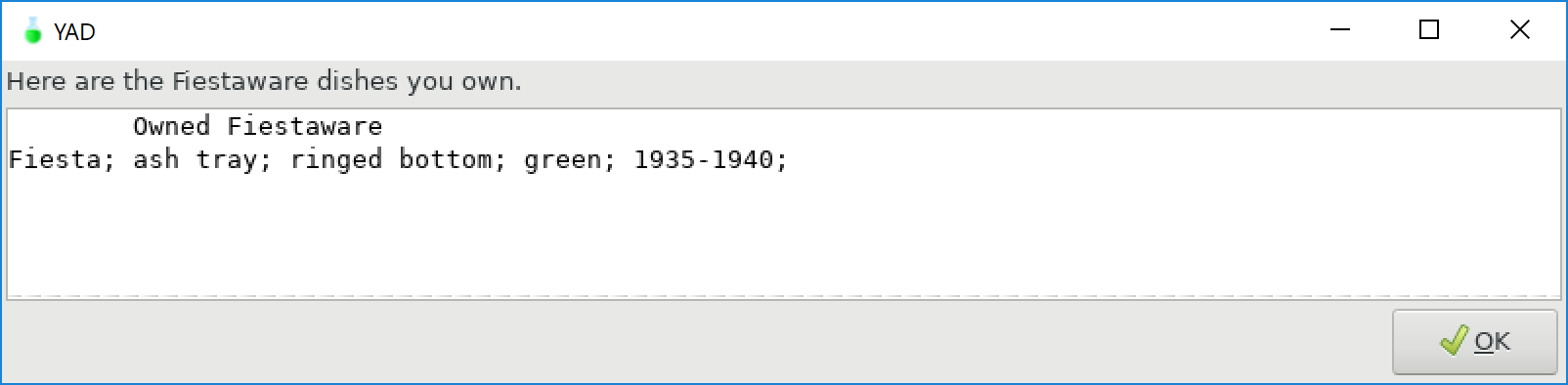






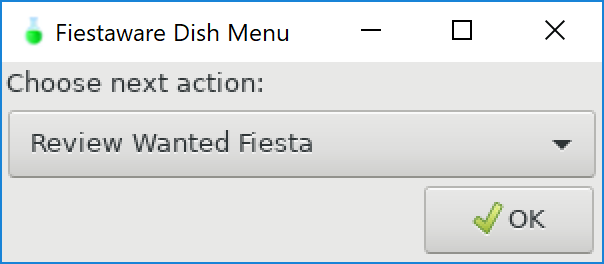
# View owned dishes | User can view all owned dishes of a brand

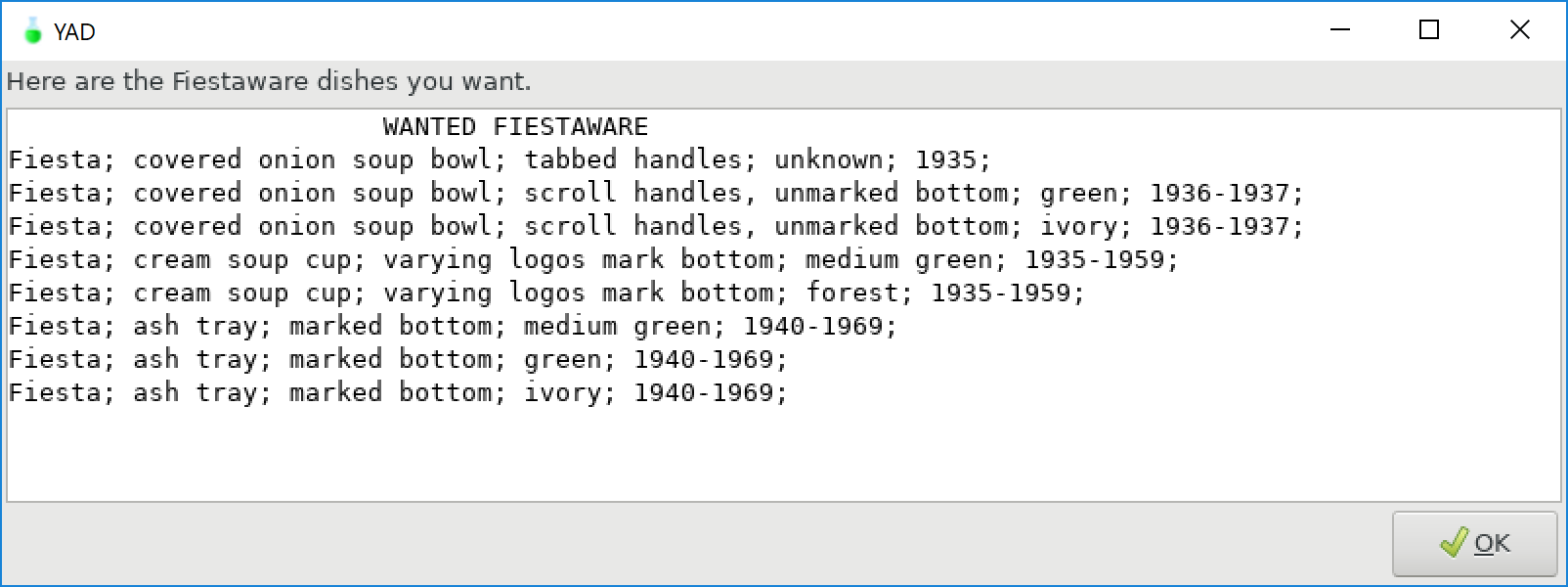
Use as many scenarios as needed to provide an overview of the system. The scenarios should give a screen shot and describe how the screen is to be used.

# Review wanted dishes | User can view all wanted dishes of a brand

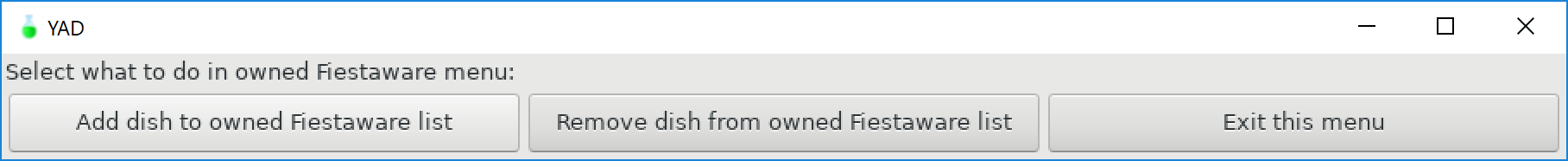
The user selects Review Wanted [dish brand]. The user is then presented with the corresponding wantedBrandDishes.txt file.

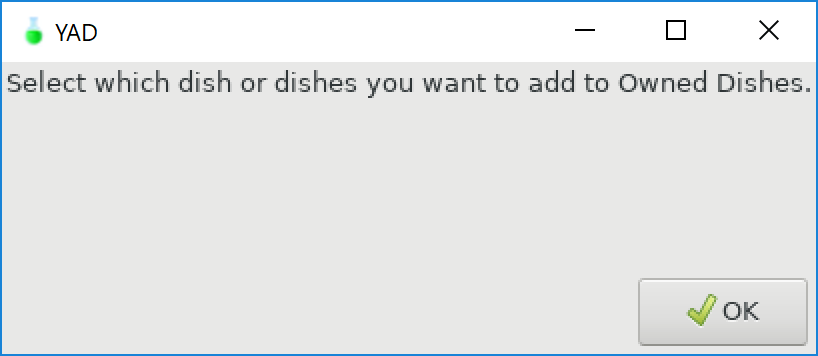


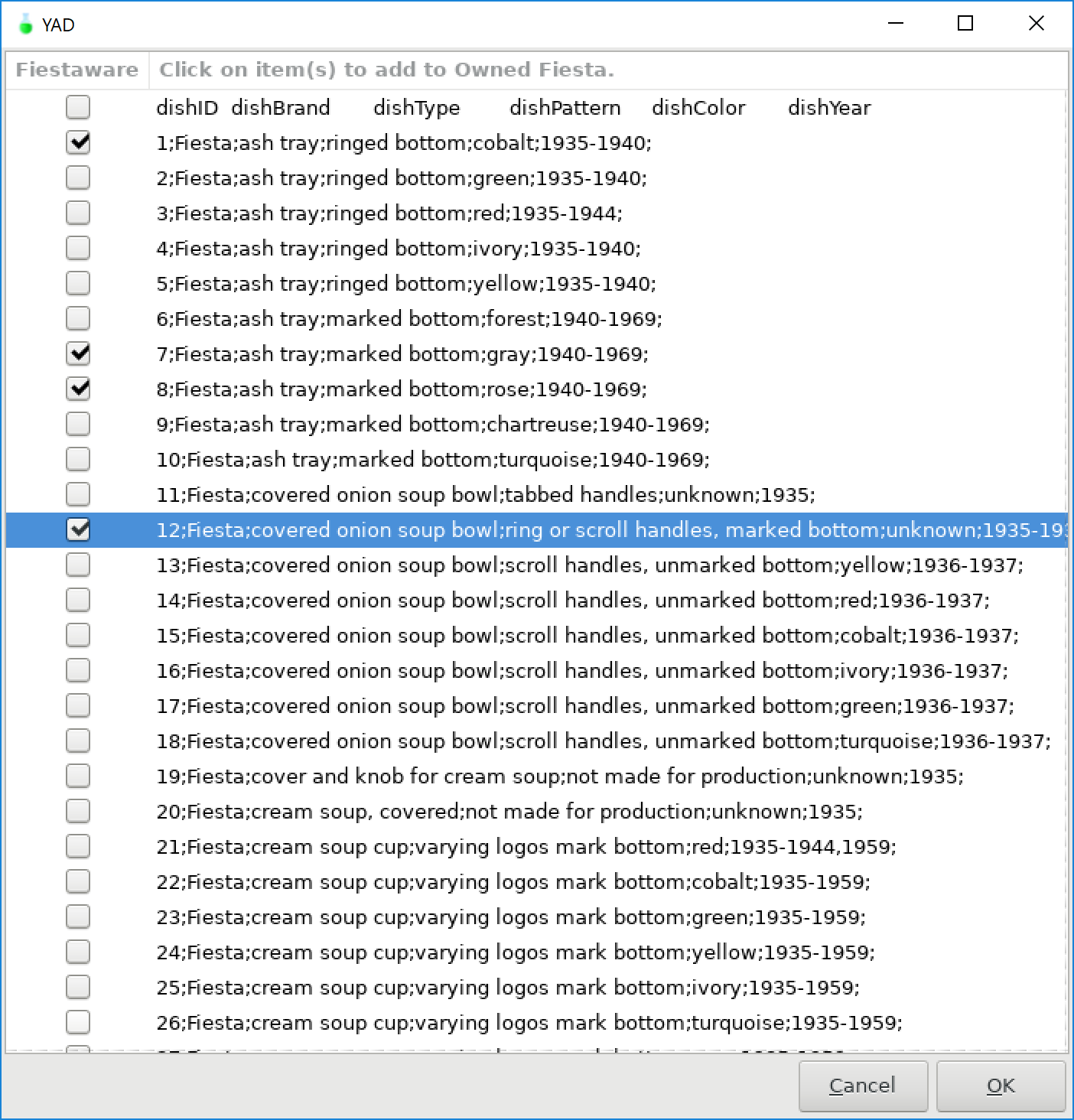


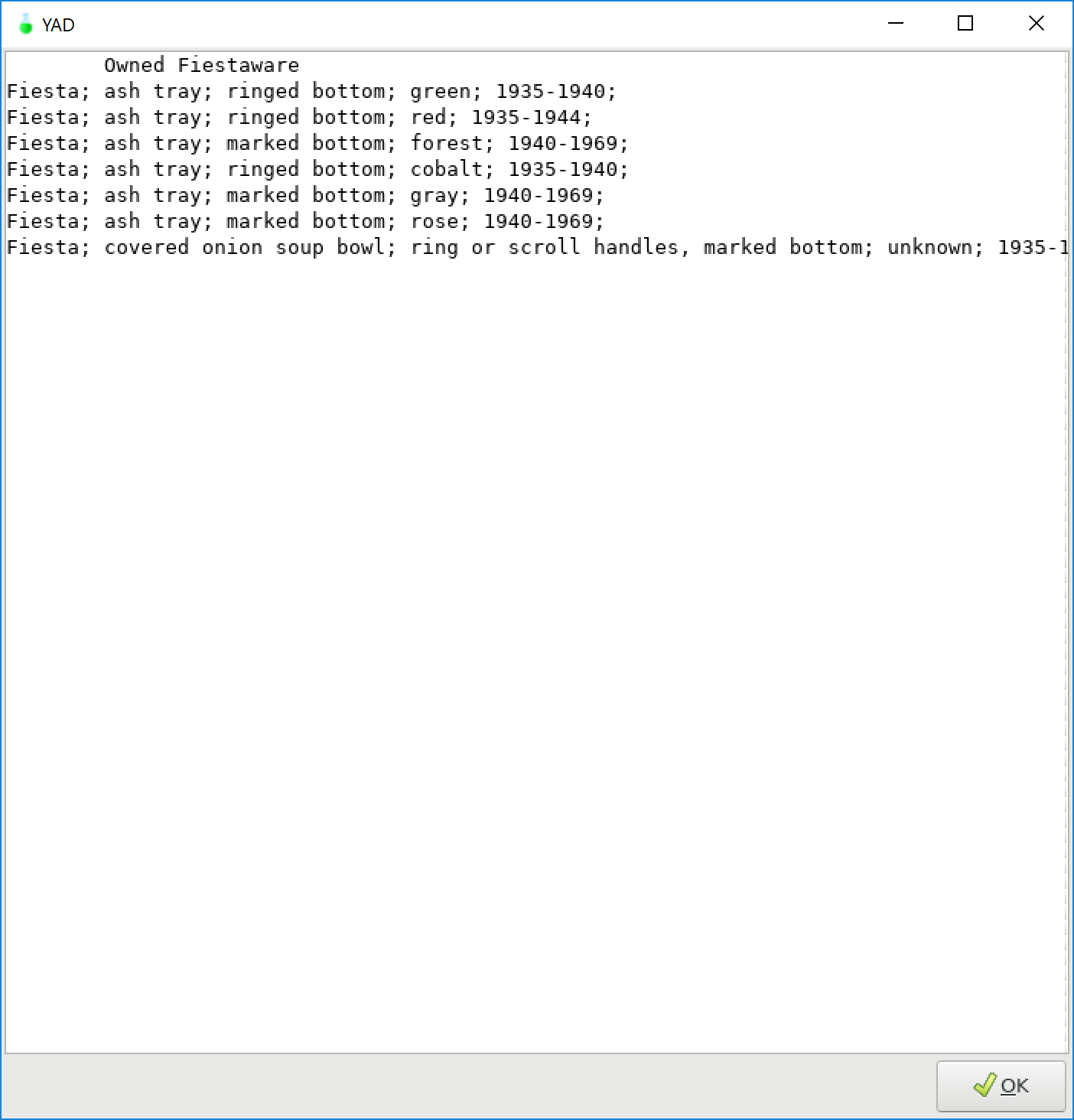
# Review owned dishes | Owned dishes menu is displayed

User is presented with owned dish menu (add dish to owned dish list, remove dish from owned list, or exit). Adding and removing dishes from the owned list involves making selections from a generated list. To add dishes, the user selects from the entire owned dish list. This preserves the syntax of the dish lists. Removing a dish is done by selecting dishes to remove from the owned dish list.



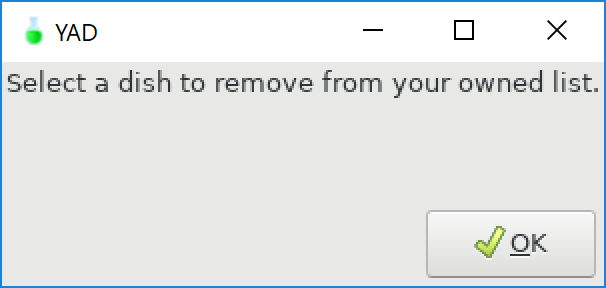


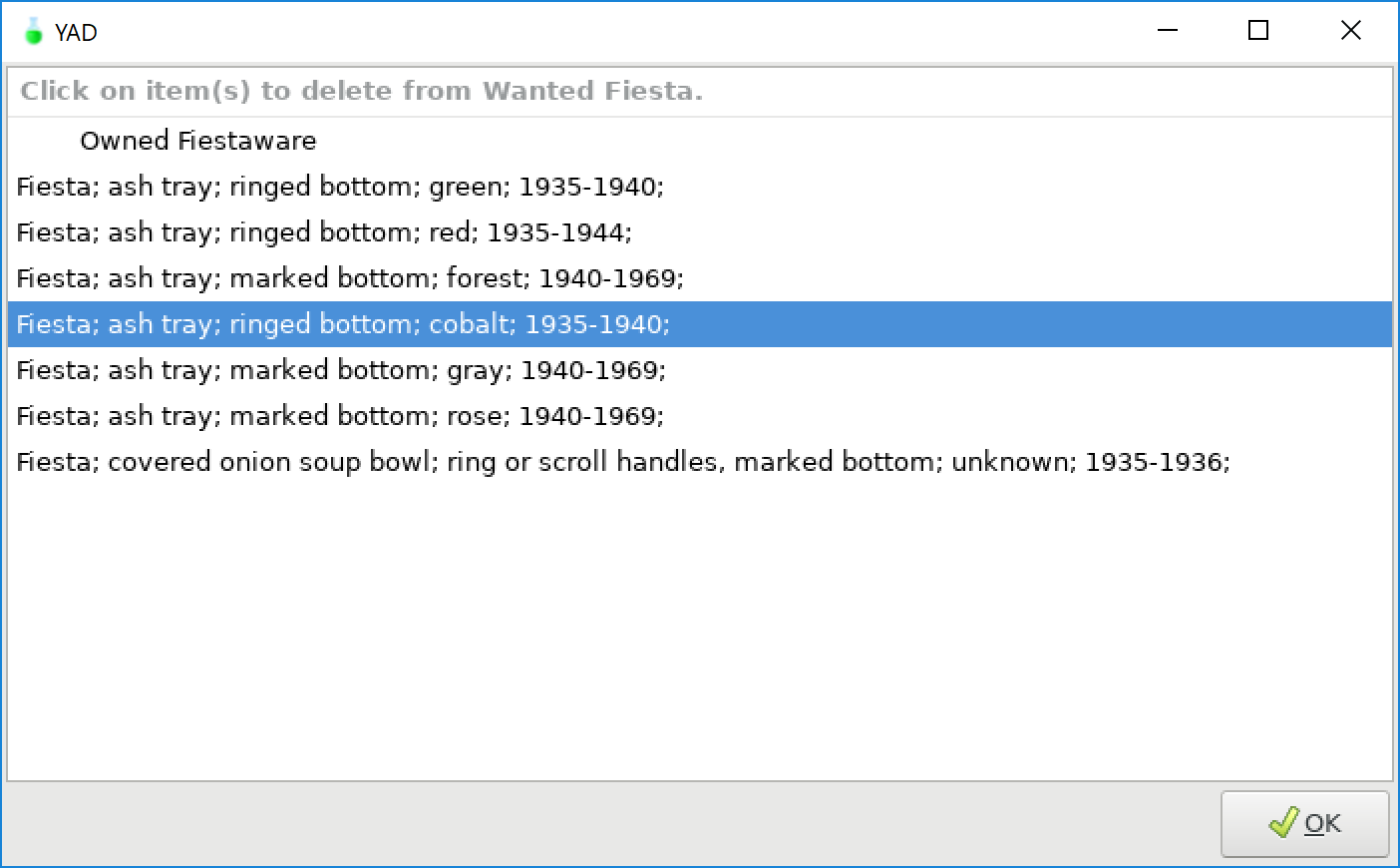


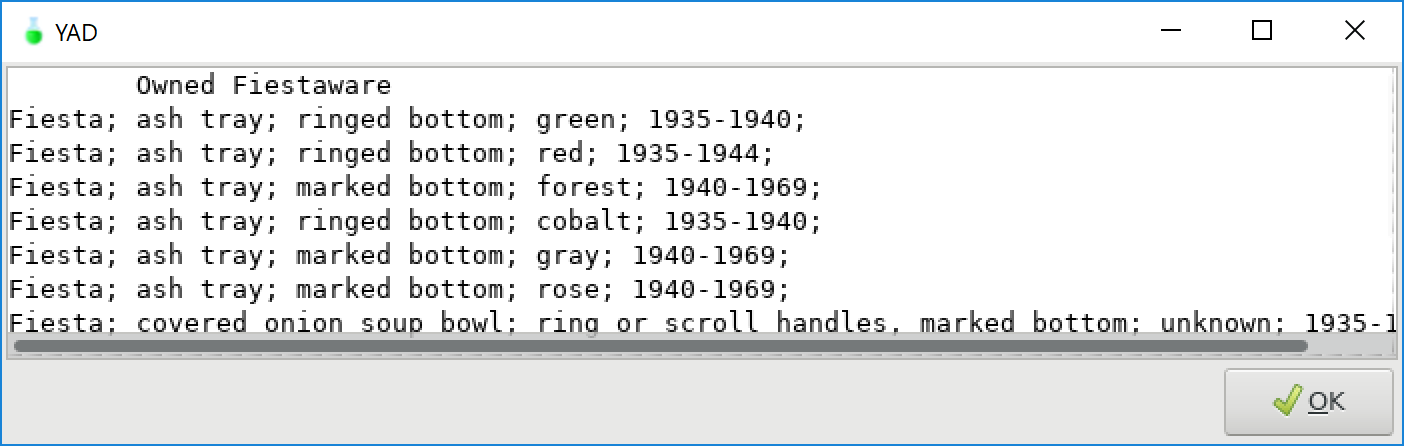


# Remove dish | User selects a dish to remove from wanted list

Once remove dish from owned list is selected, the user is presented with a dialog to select the dish to remove from the owned dish list. The user highlights the dish to remove using the cursor, then selects “Ok.” The dish is removed, and the newly updated owned dish list is presented to the user. This helps to verify the dish was removed as well as allow the user to see what dishes remain on the list.

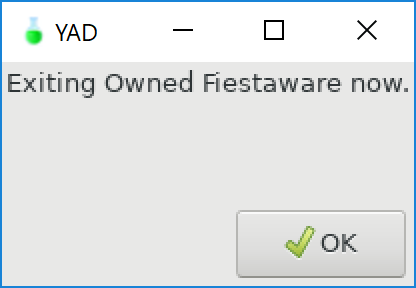






# Exit owned dishes | Exits from owned dish module

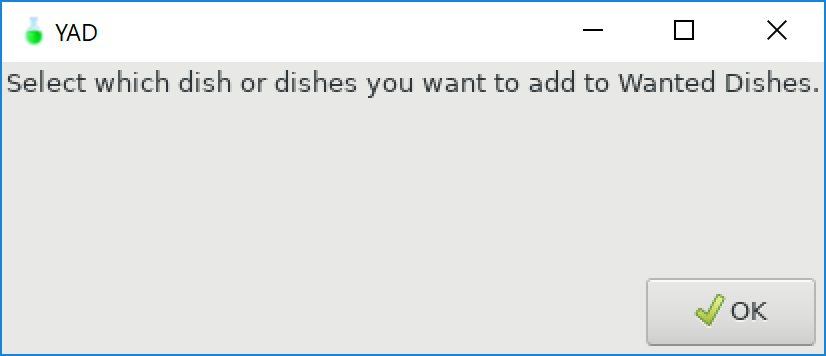
When user selects to exit owned dish menu, the below dialog appears. Then the user is presented with the dish brand submenu.

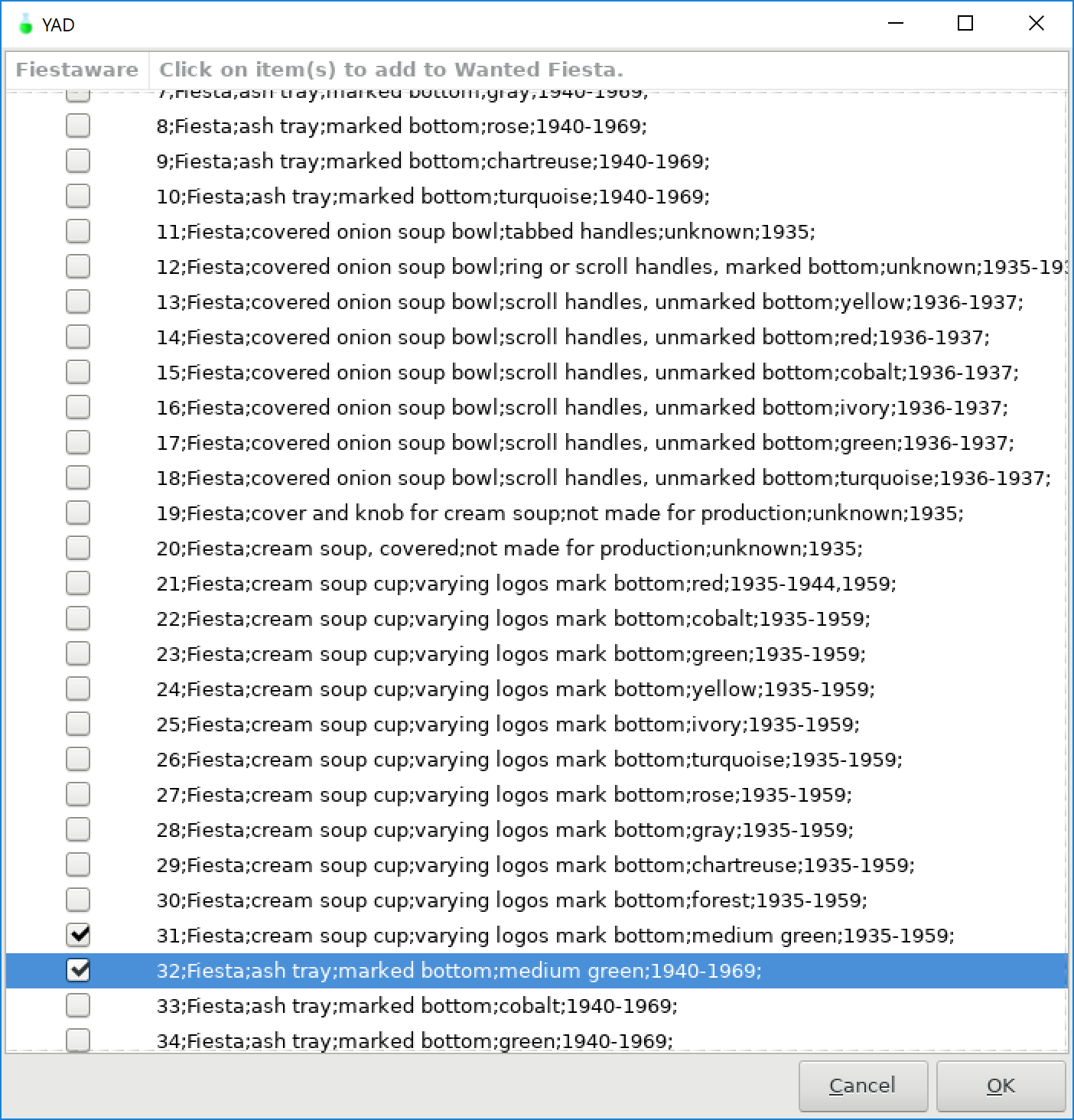


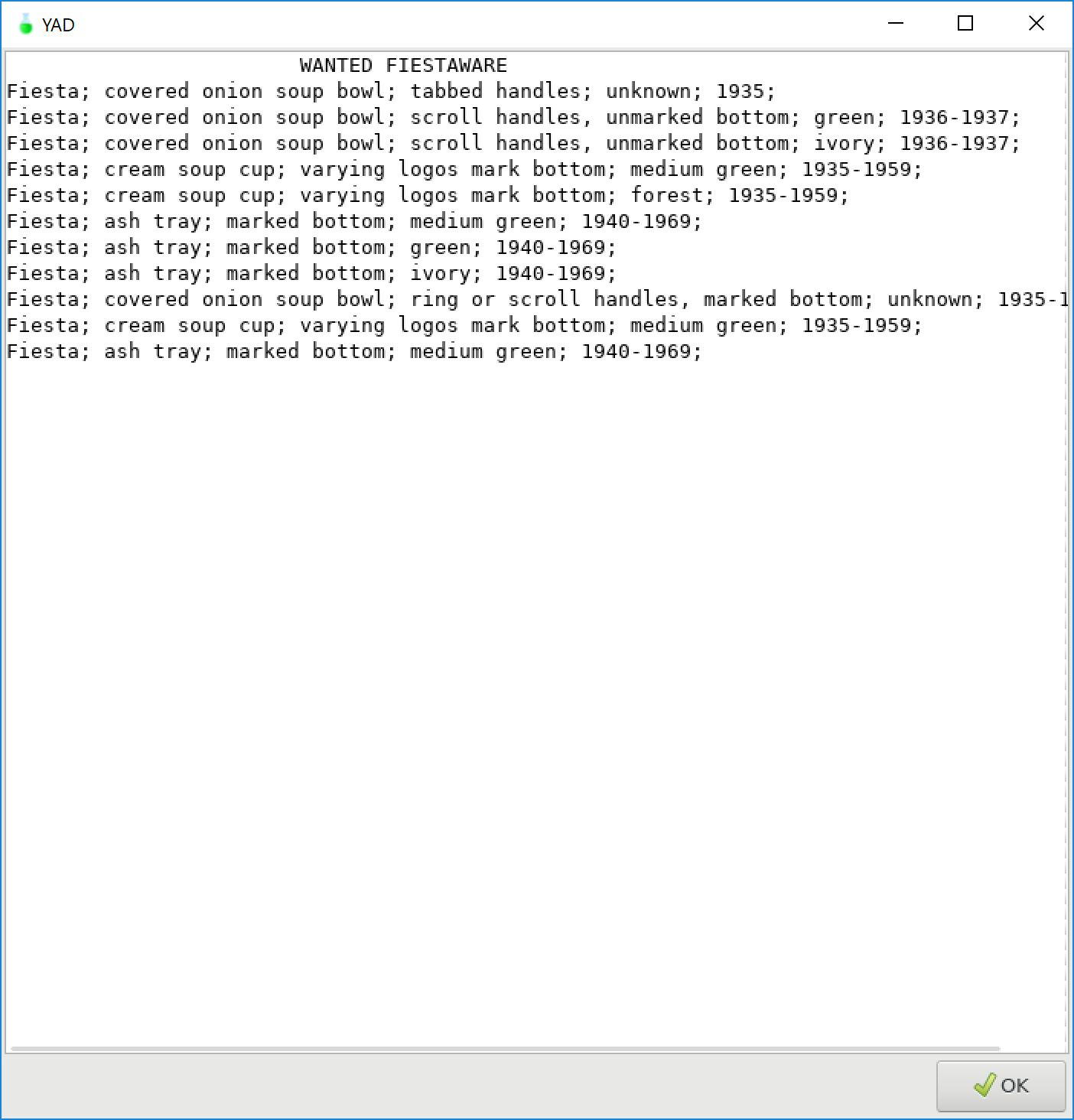
# Update wanted dishes | User goes to wanted dish submenu

User can select from adding dishes to the wanted list, removing dishes from the wanted list, and exiting the wanted dish submenu.



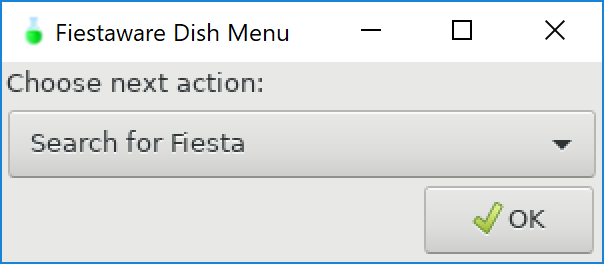


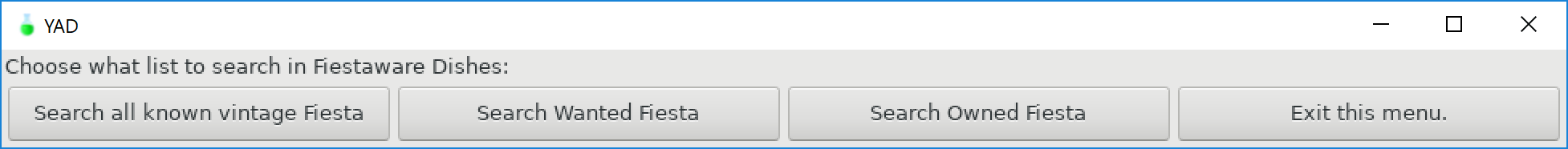




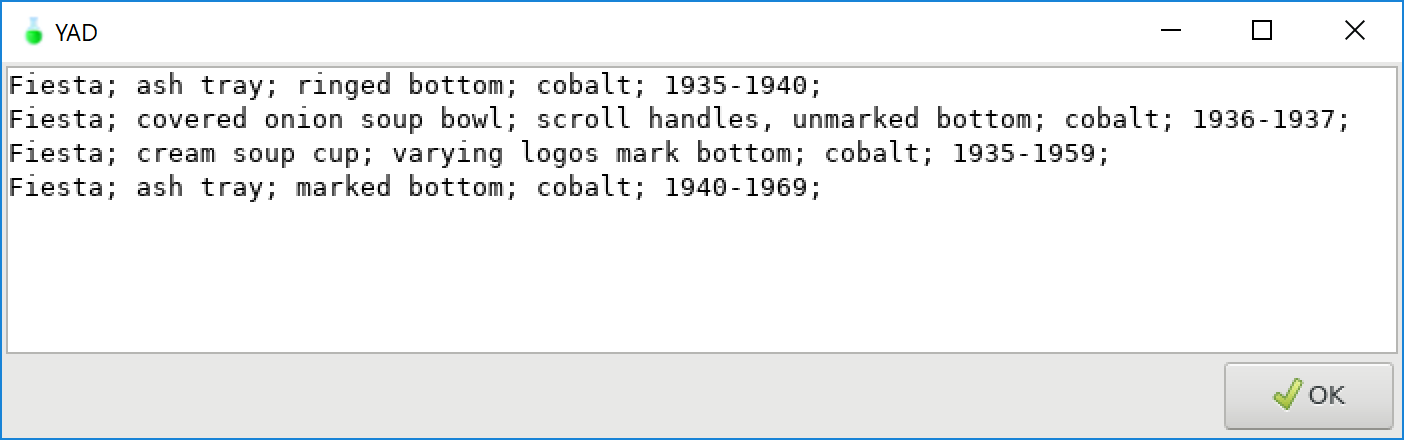
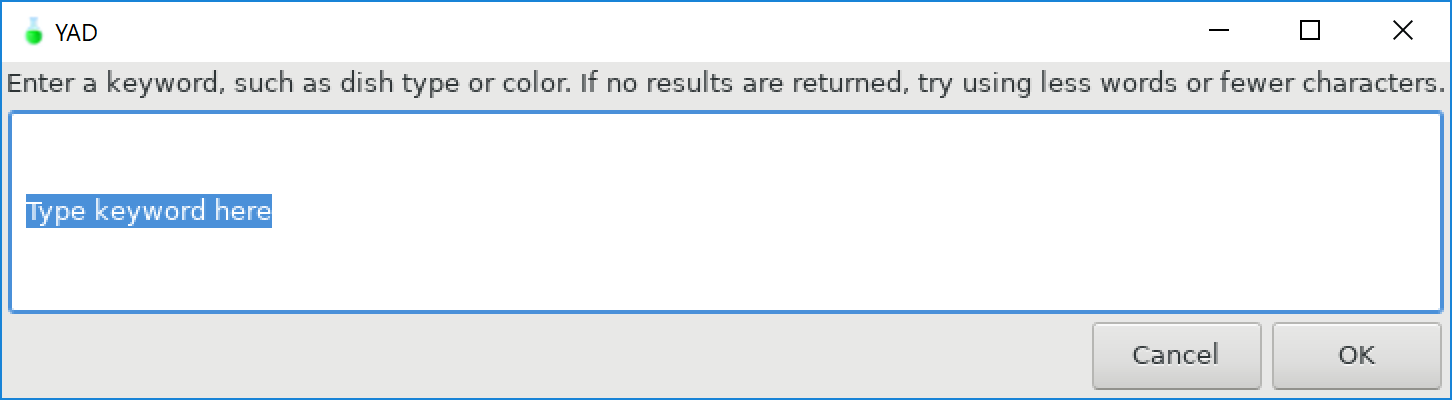
# Search for dishes | User is brought to search dishes submenu

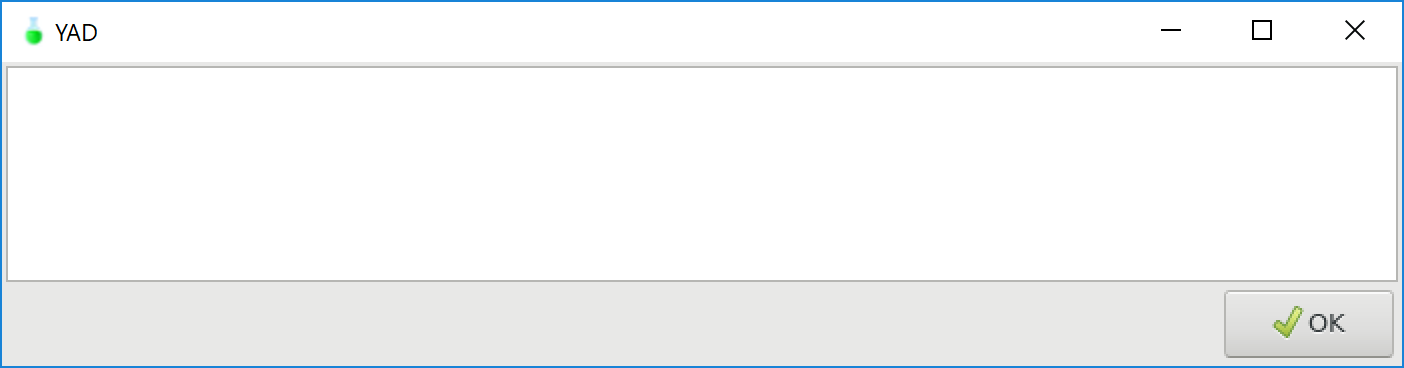
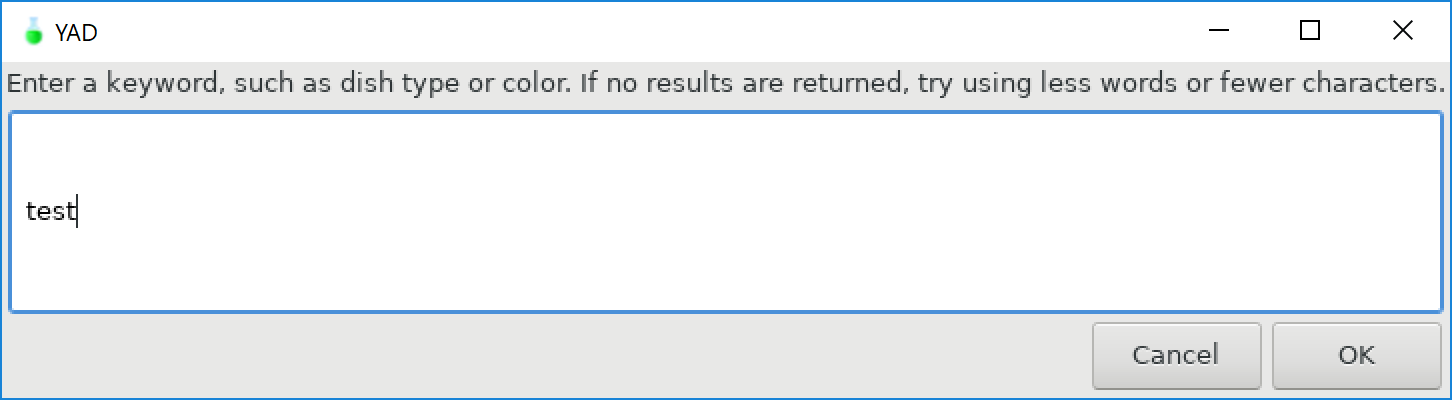
Within a dish brand submenu, the user can search for dishes. After select “Search for Brand Dishes” the user is then taken to the search dishes submenu. The user can search through all known pieces of a brand, search owned dish list, search wanted dish list, and exit the menu.

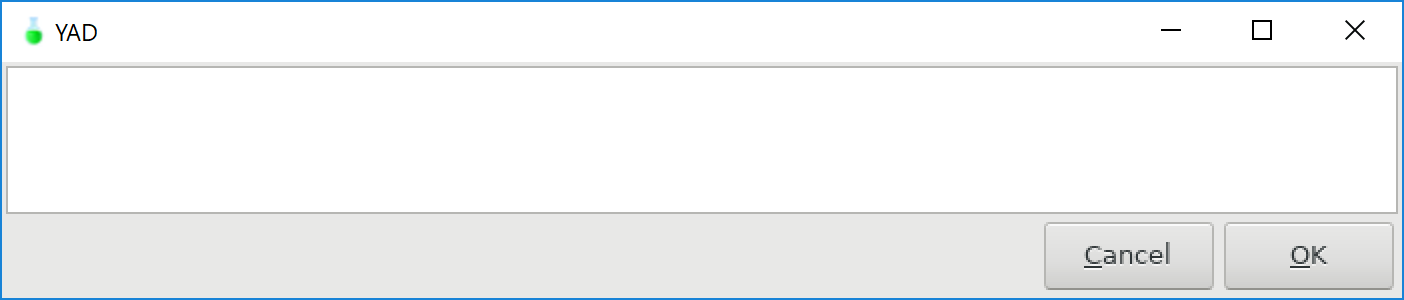
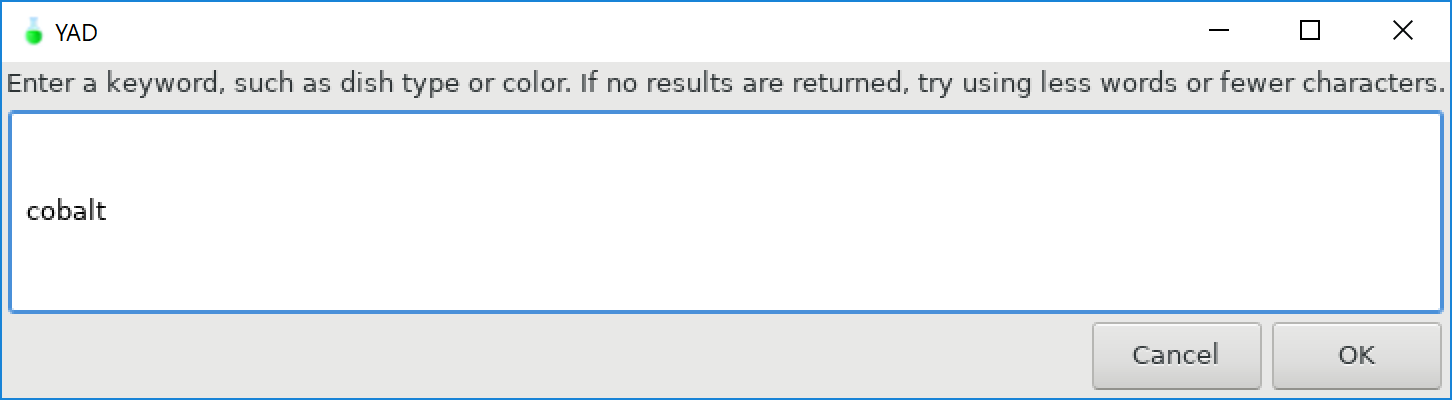


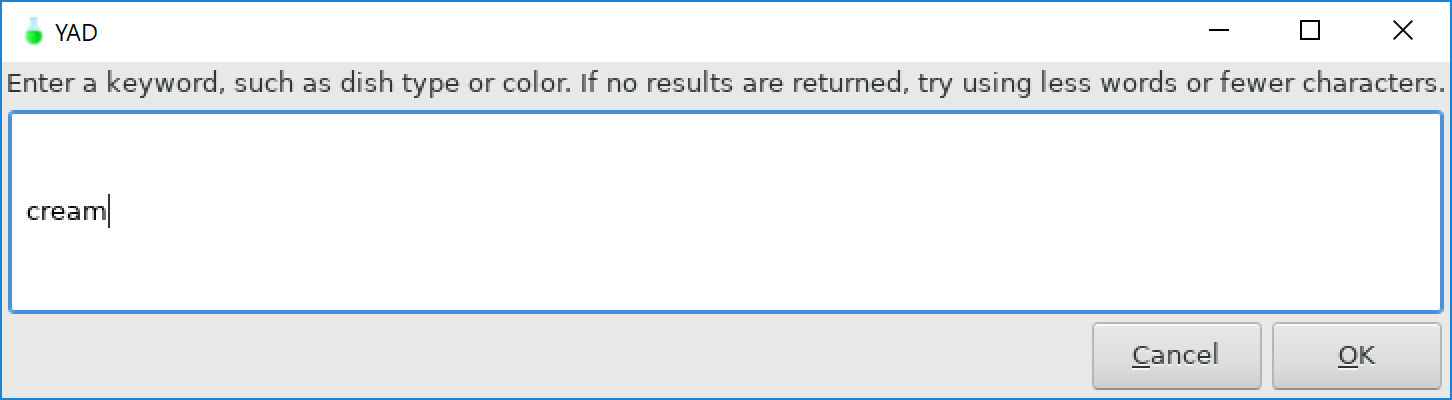
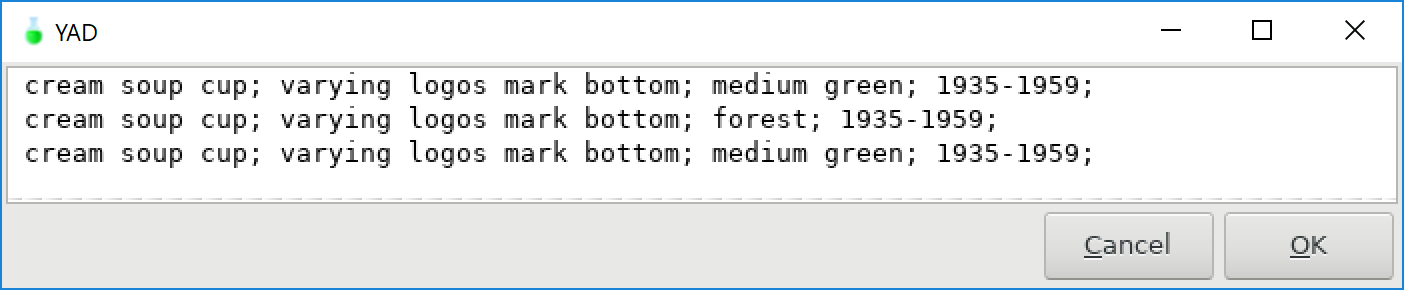


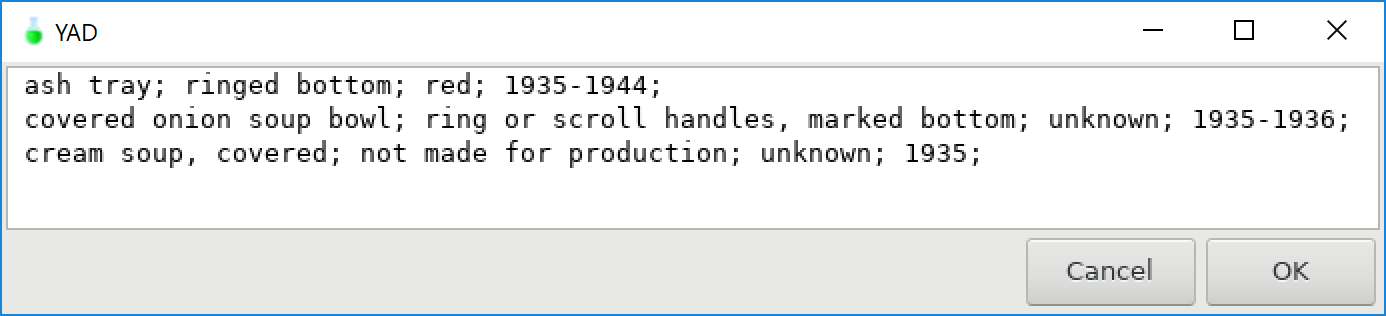
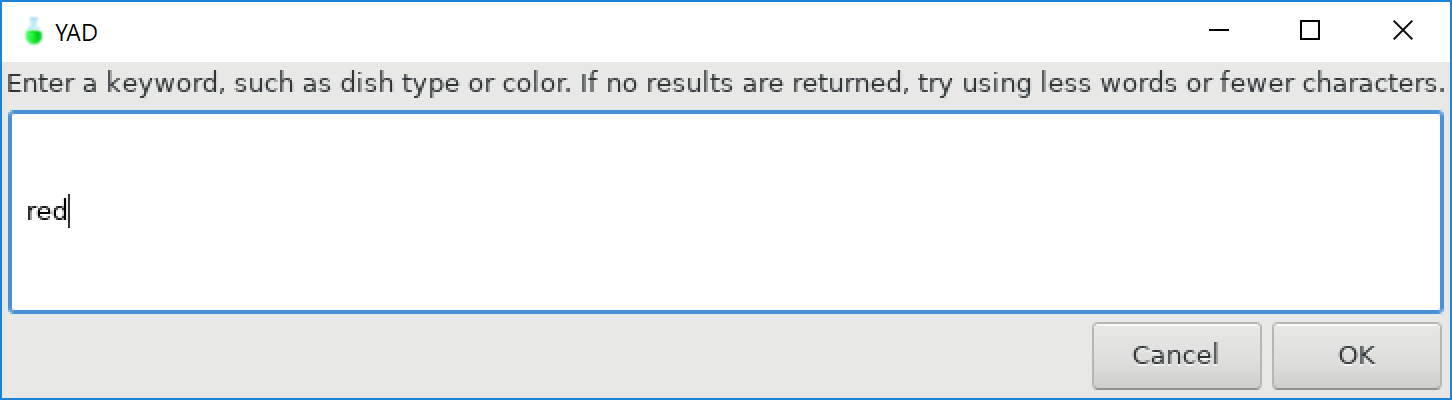
Search all dishes (keyword exists):

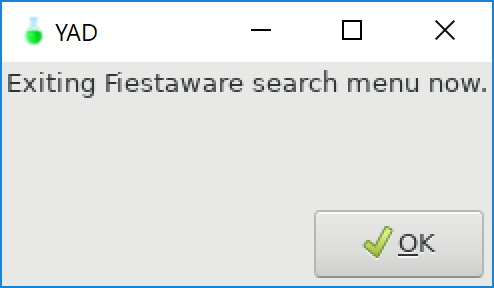
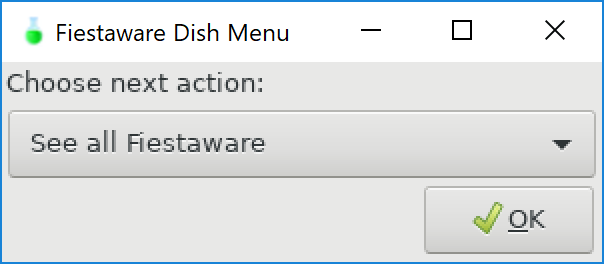


Search all dishes (keyword does not exist):

Search wanted dishes (keyword does not exist):

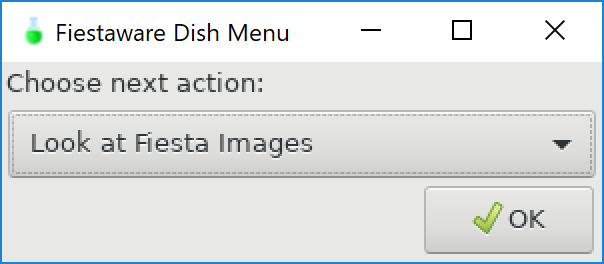
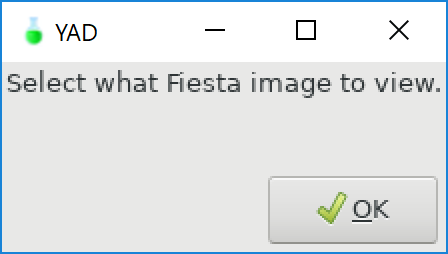
Search wanted dishes (keyword does exist): 

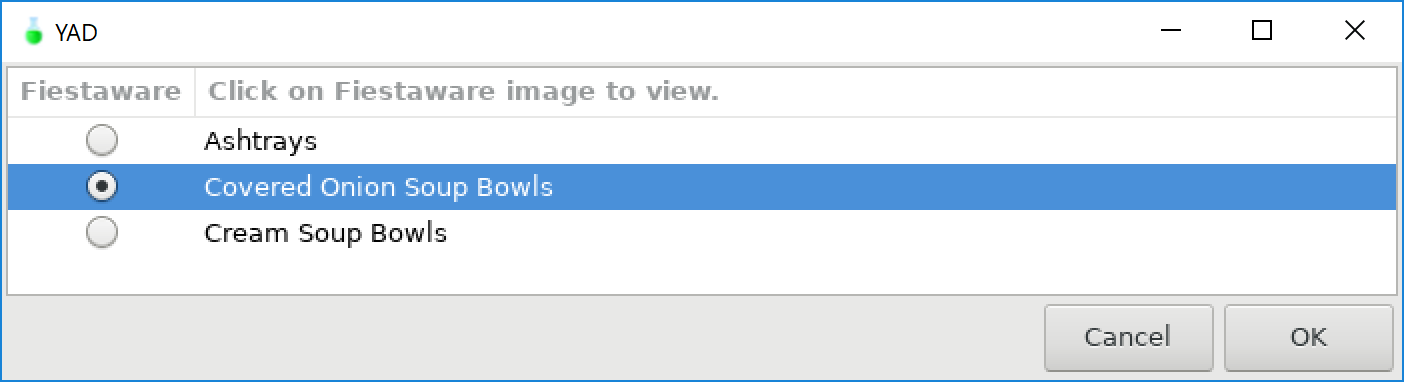
Search owned dishes:  


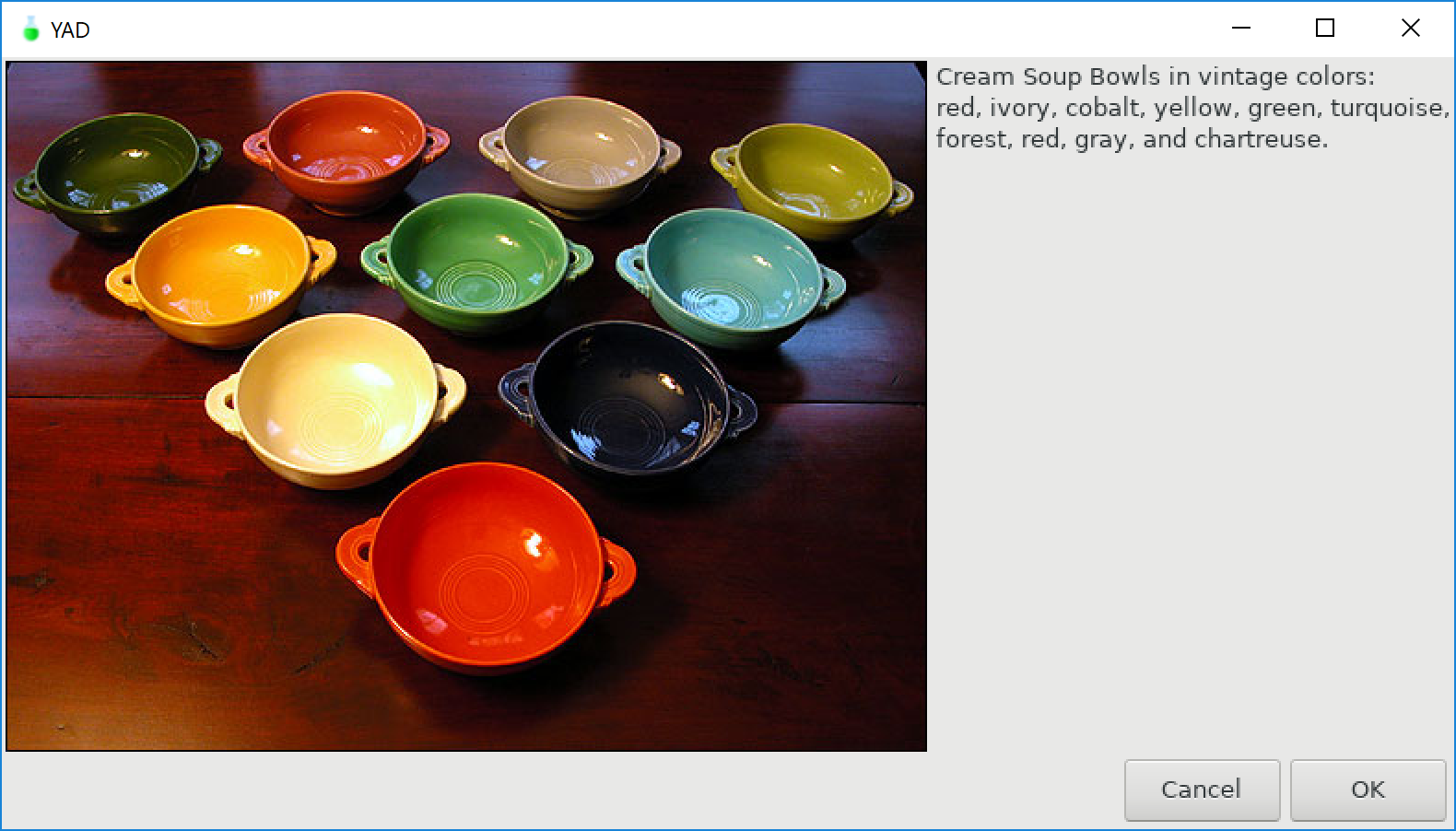
Exiting  
 

# Look at images | View existing images of specific dishes

Selecting the Look at Dish Brand Images module allows the user to view specific images. The program comes with some images, to aid in identification of dishes. The user is presented with a list of available images for a brand of dishes. The user selects an image and then that file is retrieved and displayed. After the user is done viewing the image, the user is brought back to the Dish Brand Submenu.

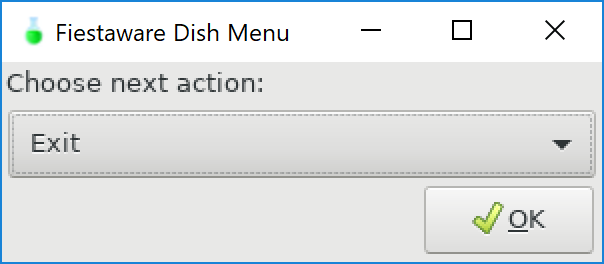
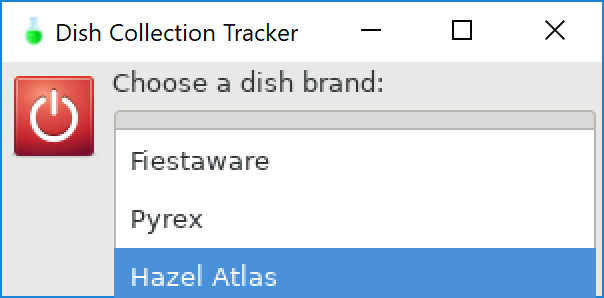
 





# Exit dish brand | Exit takes user back to dish brand selection menu

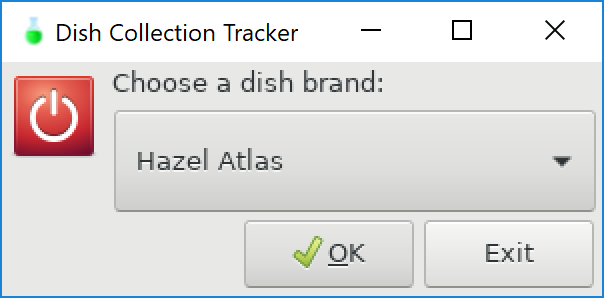
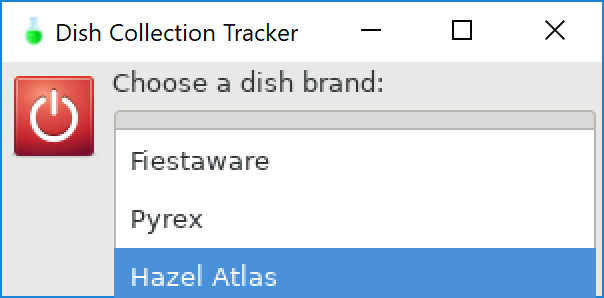
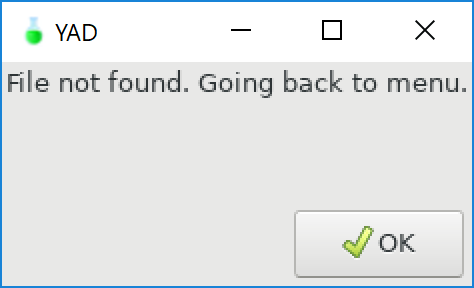
When the user wishes to exit a Dish Brand Submenu, the user selects Exit. This exits the while true do loop, and the user is taken back to the dish brand selection menu (Start Program scenario).

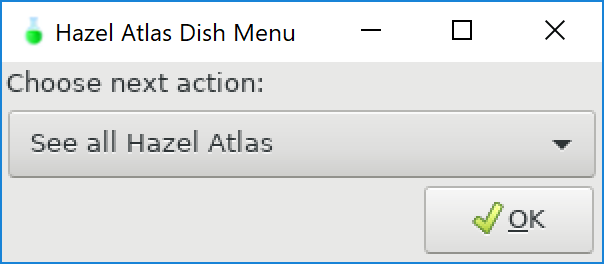
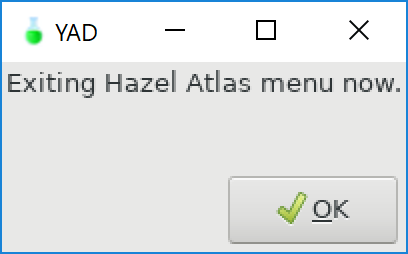
 

# File does not exist | User selects modules that execute error dialog

The Hazel Atlas Dish Submenu was designed to test that program performs as expected when files do not exist. This submenu contains all components and models as Fiestaware and Pyrex, expect the Look for Images module (due to time).

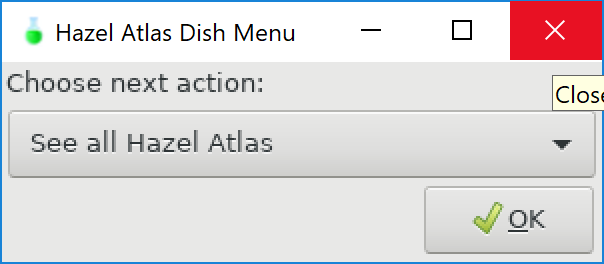
User selects Hazel Atlas Dish Submenu. User selects to view all dishes. User is presented with dialog stating the file was not found. The user is then taken back to the Hazel Atlas Dish Submenu.

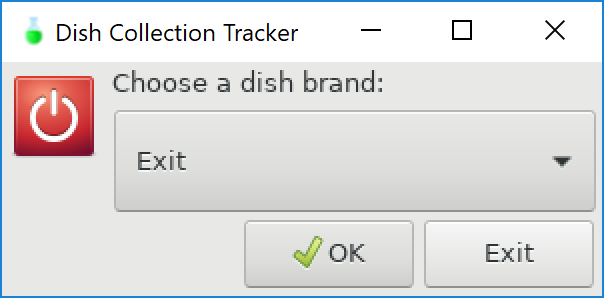
# Exit submenu| User wants to exist a Dish Brand Submenu

The user can select the “x” button in the upper right-hand corner of the yad dialog box to exit the program. The user can also select “esc” and exit.



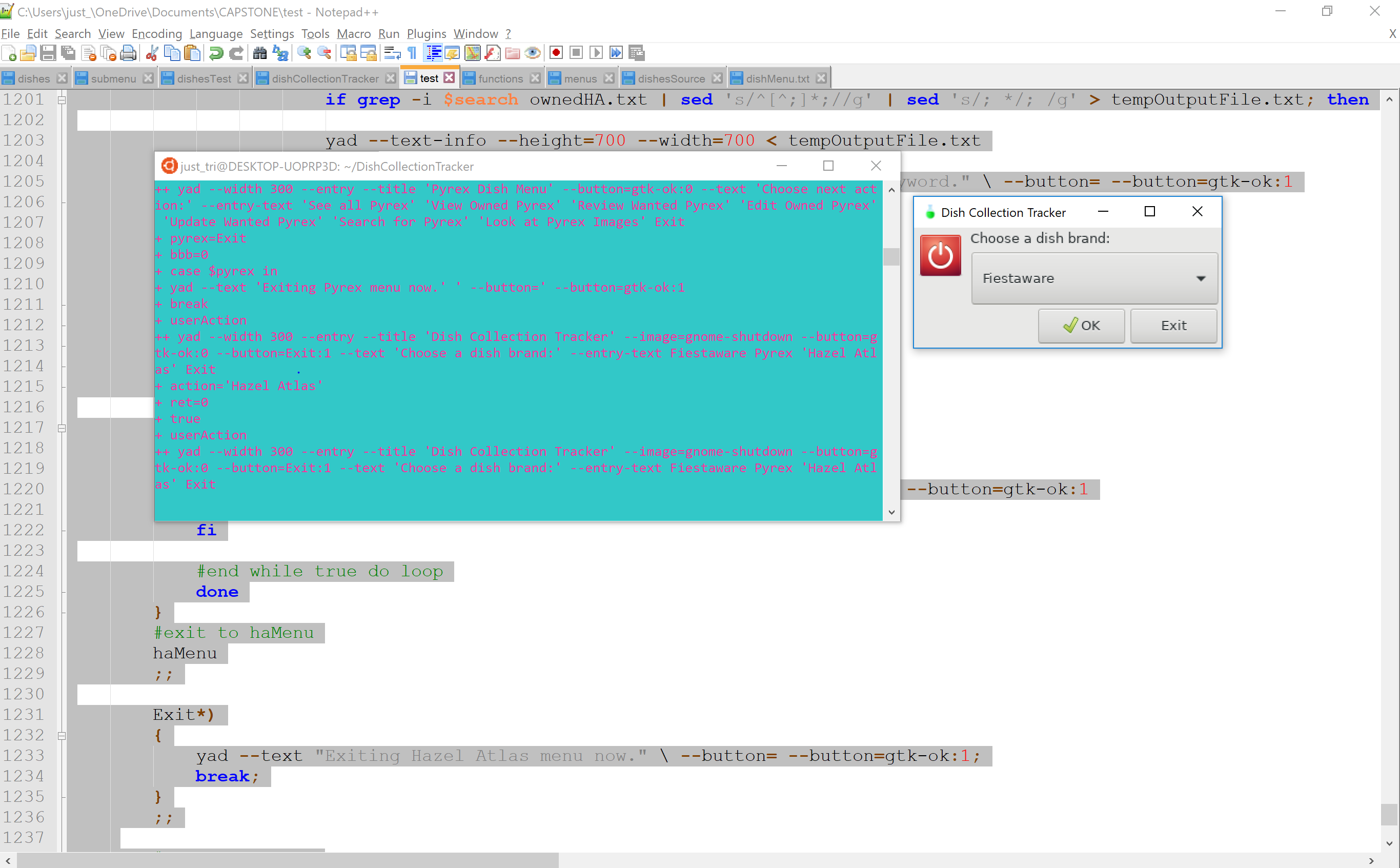
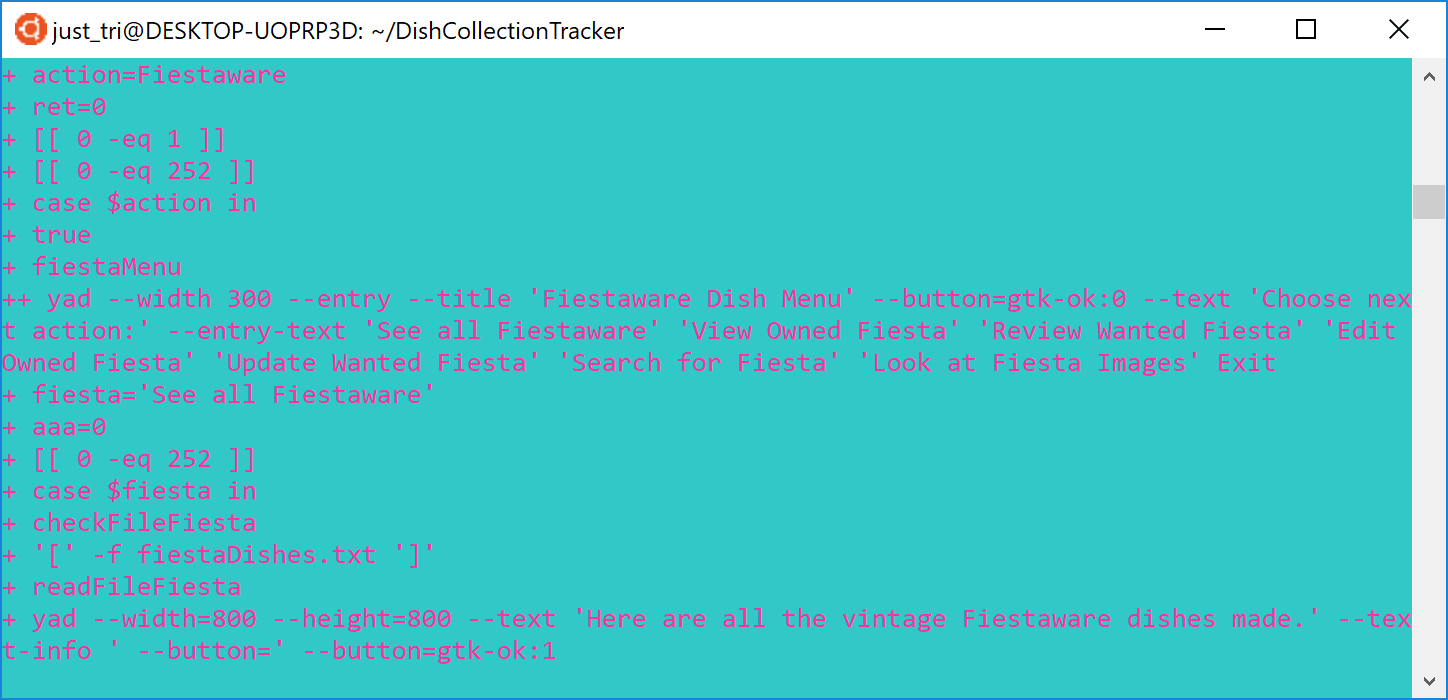
# Exit | User wants to exit Dish Collection Tracker

When the user is at the primary menu, the last option is to Exit. The user can select Exit from the drop down menu and select “OK” or the user can click the “Exit” button. Either selection will close Dish Collection Tracker. The user will then see a blinking cursor at the command line interface.

# Reports

Dish Collection Tracker does not have any built-in reporting systems. However, it should be noted that by executing the bash script with -x (i.e. bash -x dishCollectionTracker), the terminal shows the commands being executed and any values returned. This can be used to debug. This can be a useful tool for programmers supporting Dish Collection Tracker to determine where an error is occurring and possibly why.



In addition, another way to track what is occurring in the program is when data is returned to the user. For example, when a dish is added or removed from a list, the new list is shown. This allows the user to verify the action was executed as expected. Screenshots of this are included above, in the scenarios section.

# System Architecture

This program’s architecture consists of the user, the user’s device (recommended to be a Windows 10 device with Ubuntu), the bash script (dishCollectionTracker), and an X Server for Windows to graphically display the script (Xming is recommended). The user may prefer to have a wizard to setup the X Server, such as XLaunch. If the user does not have yad installed, that will also need to be installed in Ubuntu.

Each device requires the installation of all components. After all the files and required programs (yad and an X Server for Windows), then the user executes the bash script, dishCollectionTracker.

Once executed, the bask script interacts with the necessary files and programs. If the script cannot find a file, that information is displayed to the user. The script contains commands to edit and save files, based on the user’s input. The script also uses yad to display information graphically to the user, with the help of an X Server for Windows.

## System Architecture Diagram

The user’s device interacts with the program files, specifically the bash script to use Dish Collection Tracker. As the user uses their device (looks at the screen, uses the keyboard and/or mouse), that input is captured through X Server. X Server takes the input of where the user clicked to then interact with the bash script. Then the user’s input is executed from the bash script. The information then is sent to X Server, displayed to the user, and ready for the user’s input again.

User’s Device

X Server for Windows

Program Files

User’s Input

Bash Script

## Source Code Structure

Required files for installation of program are described in the below table.

|  |  |
| --- | --- |
| **Code Directory** | |
| **Directory** | **Usage** |
| DishCollectionTracker | Contains all source code for program. |
| Images | Folder that contains subfolders with dish brands. Holds images of various piece. |
| dishCollectionTracker | Contains all source code for program. |
| Images/Fiesta | Directory for Fiestaware images. |
| Images/Pyrex | Directory for Pyrex images. |
| FiestaImages.txt | Data storage file for list of available Fiestaware dishes to view images of in look at dishes module. |
| PyrexImages.txt | Data storage file for list of available Pyrex dishes to view images of in look at dishes module. |
| fiestaDishes.txt | Data storage file listing all known Fiestaware dishes (currently only a partial listing). |
| pyrexDishes.txt | Data storage file listing all known Pyrex dishes (currently only a partial listing). |
| ownedFiesta.txt | Data storage file for Fiestaware dishes user currently owns. This file is modified by user in program, adding and removing data. |
| ownedPyrex.txt | Data storage file for Pyrex dishes user currently owns. This file is modified by user in the program, adding and removing data. |
| wantedFiesta.txt | Data storage file for Fiestaware dishes user currently wants. This file is modified by user in program, adding and removing data. |
| wantedPyrex.txt | Data storage file for Pyrex dishes user currently wants. This file is modified by user in program, adding and removing data. |
| Ashtrays.jpg | Image file of Fiestaware ashtrays. |
| Covered\_Onion\_ Soupbowls.jpg | Image file of Fiestaware covered onion soup bowls. |
| Cream\_Soup\_Bowls.jpg | Image file of Fiestaware cream soup bowls. |
| Butterprint\_Cinderella\_ Bowls.jpg | Image file of Pyrex cinderella-style mixing bowls. |
| Primary\_Mixing\_Bowls. jpeg | Image file of Pyrex mixing bowls in primary colors. |
| Snowflake\_Various\_ Pieces.jpg | Image file of Pyrex dishes in snowflake pattern. |
| Terra.jpg | Image file of Pyrex dishes advertisement in terra pattern. |
| *Highlighted rows indicate directories containing source code.* | |

# Executables

The executable file of this program is the bash script, dishCollectionTracker. This bash script contains functions, modules, and commands to run the entire program and communicate with the data storage .txt files.

### Bash Script (dishCollectionTracker)

A bash script, that once executed by “bash dishCollectionTracker” at the command line, will launch the entire program.

# Code Architecture

Dish Collection Tracker is comprised of a bash script that the user executes. As an overview, once started, the script creates yad dialogs for the user to interact with the program. The script then takes the user input, navigates to the appropriate section of the code. If instructed, the script then can read and if needed, write, to a specified file. The script can also display select images to the user in a yad dialog. When the program is exited, the script is terminated.

## Data Store

The program is comprised of multiple switch menus. There is a menu to select the brand of dish. Once a brand is selected, another menu is presented to the user, displaying the options. These menus are functions. Data is stored in text files and in directories. When data in a text (.txt) file is needed, the data in the file is redirected to a temporary file. That temporary file may be further manipulated, if needed, such as adding extra spaces to display properly to the user. Then the data from the temporary file is displayed. The user can interact with the data through the yad GUI dialog. Any input captured from the user is then redirected to either another temporary file or results in the original .txt data file being manipulated.

Data is added or removed from the .txt files by grep commands and redirection of input and output. By manipulating the flow of data to and from the .txt files, users create individualized wanted dish lists and inventory dish lists. There are also .txt files that contain all dishes from a brand for users to view. Users also view the all dishes lists to select dishes they add to the wanted list. The program comes with the necessary .txt files. At this point in the program’s development, users cannot create .txt files. If a .txt file does not exist, a message displays indicating the file is not found and the user is brought back to a menu.

## User Defined Functions

User-defined functions are used for the main menu, dish submenus, reading files, checking if files exist, and displaying a message if a file does not exist. These functions are discussed under the classes section.

External Files & Data

As previously discussed above, .txt files and images (.jpg and .jpeg) files are contained as part of this program. These files have been documented above; however, as a reminder it is recommended to keep all the image files in the Image folder. Fiesta and Pyrex are subdirectories of Images. The .txt files and bash script are stored in the DishCollectionTracker directory. Images could be contained inside DishCollectionTracker, if the user wishes to install the program that way.

The below directories and files, have the following permissions, depending on if the user needs read and/or write access to it:

-rw-rw-r-- 1 just\_tri 51 Aug 3 23:08 FiestaImages.txt

-rw-rw-r-- 1 just\_tri 101 Aug 10 10:22 PyrexImages.txt

-rwxrwxrwx 1 just\_tri 40265 Aug 11 13:56 dishCollectionTracker

-rw-rw-rw- 1 just\_tri 2406 Aug 10 19:07 fiestaDishes.txt

-rw-rw-rw- 1 just\_tri 675 Aug 10 20:59 ownedFiesta.txt

-rw-rw-rw- 1 just\_tri 50 Aug 10 19:19 ownedPyrex.txt

-rw-rw-rw- 1 just\_tri 2389 Jun 23 12:07 pyrexDishes.txt

-rw-rw-rw- 1 just\_tri 797 Aug 10 21:01 wantedFiesta.txt

-rw-rw-rw- 1 just\_tri 18 Jul 6 23:02 wantedPyrex.txt

drwxrwxrwx 1 just\_tri 4096 Aug 3 19:35 Images

drwxrwxrwx 1 just\_tri 4096 Aug 3 22:29 Fiesta

drwxrwxrwx 1 just\_tri 4096 Aug 11 13:53 Pyrex

-rwxrwxrwx 1 just\_tri 36992 Aug 4 16:42 Butterprint\_Cinderella\_Bowls.jpg

-rwxrwxrwx 1 just\_tri 25888 Aug 4 16:39 Primary\_Mixing\_Bowls.jpeg

-rwxrwxrwx 1 just\_tri 171103 Aug 3 20:10 Snowflake\_Various\_Pieces.jpg

-rwxrwxrwx 1 just\_tri 105071 Aug 11 13:50 Terra.jpg

-rwxrwxrwx 1 just\_tri 10285 Aug 3 22:18 Ashtrays.jpg

-rwxrwxrwx 1 just\_tri 32311 Aug 3 22:19 Covered\_Onion\_SoupBowls.jpg

-rwxrwxrwx 1 just\_tri 53295 Aug 3 22:20 Cream\_Soup\_Bowls.jpg

Programming Language | Linux

This project was written in Linux, using Ubuntu version 18.04 LTS. A Windows 10 system was used while coding this project. The program is a bash shell script. Once executed, the user no longer uses the terminal to navigate through the program, but interacts through GTK+ dialog boxes.

Yad, which stands for yet another dialog, was used to develop GTK+ dialog boxes in shell scripts. Yad can interact with shell scripts and GTK+ dialogs to provide a graphical user interface (GUI) for the user to navigate the program after executing the program through the command line.

GTK+ dialog is an open-source widget that creates the GUI. GTK+ dialog is an object-oriented widget took licensed through GNU Lesser General Public License. However, to display yad dialogs, a X Window System is needed. Xming, was selected. Xming provides the applications, tools, and fonts to display yad GUIs. In order to launch Xming, XLaunch is used. XLaunch is a wizard that prompts the user through setup of XMing X server sessions.

References:  
*yad (1) – Linux man pages*. (n.d.). Retrieved Juy 31, 2019 from https://www.systutorials.com/docs/linux/man/1-yad/

*GTK*. (n.d.). Retrieved August 01, 2019 from https://en.wikipedia.org/wiki/GTK

*Xming*. (n.d.). Retrieved August 01, 2019 from https://en.wikipedia.org/wiki/Xming

User-defined Functions/Project Classes

Linux is not an object-oriented programing language. However, user-defined functions were used to abstract re-usable pieces of code. The project utilizes these functions:

### Read File | readFileFiesta ()

This function reads the fiestaDishes.txt file and displays the contents in a yad dialog box.

There is a function to read the pyrexDishes.txt and haDishes.txt files, too. In addition, there are read file functions for readFileWantedFiesta and readFileOwnedFiesta (as well as for Pyrex and Hazel Atlas). Thus, there are a total of nine read file functions.

#readFile functions to read .txt file

#Fiesta readFile functions

readFileFiesta ()

{

yad --width=800 --height=800 --text "Here are all the vintage Fiestaware dishes made." --text-info \ --button= --button=gtk-ok:1 < fiestaDishes.txt

}

readFileWantFiesta ()

{

yad --width=800 --height=800 --text "Here are the Fiestaware dishes you want." --text-info \ --button= --button=gtk-ok:1 < wantedFiesta.txt

}

readFileOwnFiesta ()

{

yad --width=800 --height=800 --text "Here are the Fiestaware dishes you own." --text-info \ --button= --button=gtk-ok:1 < ownedFiesta.txt

}

#Pyrex readFile functions

readFilePyrex ()

{

yad --width=800 --height=800 --text "Here are all the vintage milkglass Pyrex dishes made." --text-info \ --button= --button=gtk-ok:1 < pyrexDishes.txt

}

readFileWantPyrex ()

{

yad --width=800 --height=800 --text "Here are the Pyrex dishes you want." --text-info \ --button= --button=gtk-ok:1 < wantedPyrex.txt

}

readFileOwnPyrex ()

{

yad --width=800 --height=800 --text "Here are the Pyrex dishes you own." --text-info \ --button= --button=gtk-ok:1 < ownedPyrex.txt

}

#Hazel Atlas readFile functions

readFileHA ()

{

yad --width=800 --height=800 --text "Here are all the vintage Hazel Atlas dishes made." --text-info \ --button= --button=gtk-ok:1 < haDishes.txt

}

readFileWantHA ()

{

yad --width=800 --height=800 --text "Here are the Hazel Atlas dishes you want." --text-info \ --button= --button=gtk-ok:1 < wantedHA.txt

}

readFileOwnHA ()

{

yad --width=800 --height=800 --text "Here are the Hazel Atlas dishes you want." --text-info \ --button= --button=gtk-ok:1 < ownedHA.txt

}

#File not found function

fileNotFound ()

{

yad --text "File not found. Going back to menu." \ --button= --button=gtk-ok:1

}

### Check File | checkFileFiesta ()

This code checks if the fiestaDishes.txt file exists. To check the file, it uses readFileFiesta. If the file is not found, a second function is called, fileNotFound () (see below for information on this function).

Two other functions, checkFilePyrex () and checkFileHA () exist to read the corresponding dish files. All check file functions use the same fileNoteFound () function. In addition, checkOwnedFiesta () and checkWantedFiesta () also exist (and corresponding functions for other dish brands).

There are also total of nine functions that check if files exist and if so, read the file and if not, call fileNotFound ().

#checkFile functions check to ensure file exists before opening

#Fiesta functions

checkFileFiesta ()

{

if [ -f fiestaDishes.txt ]

then

readFileFiesta

else

fileNotFound

fi

}

checkWantedFiesta ()

{

if [ -f wantedFiesta.txt ]

then

readFileWantFiesta

else

fileNotFound

fi

}

checkOwnedFiesta ()

{

if [ -f ownedFiesta.txt ]

then

readFileOwnFiesta

else

fileNotFound

fi

}

#Pyrex checkFile functions

checkFilePyrex ()

{

if [ -f pyrexDishes.txt ]

then

readFilePyrex

else

fileNotFound

fi

}

checkWantedPyrex ()

{

if [ -f wantedPyrex.txt ]

then

readWantedPyrex

else

fileNotFound

fi

}

checkOwnedPyrex ()

{

if [ -f ownedPyrex.txt ]

then

readFileOwnPyrex

else

fileNotFound

fi

}

#Hazel Atlas checkFile functions

checkFileHA ()

{

if [ -f haDishes.txt ]

then

readFileHA

else

fileNotFound

fi

}

checkWantedHA ()

{

if [ -f wantedHA.txt ]

then

readFileWantHA

else

fileNotFound

fi

}

checkOwnedHA ()

{

if [ -f ownedHA.txt ]

then

readFileOwnHA

else

fileNotFound

fi

}

### File Not Found | fileNotFound ()

If a file is not found, a yad dialog box is displayed stating “File not found. Going back to menu.” This function is used within other functions and also in if/else loops.

#File not found function

fileNotFound ()

{

yad --text "File not found. Going back to menu." \ --button= --button=gtk-ok:1

}

### Dish Main Menu | userAction()

userAction () displays the main dish menu when called. This is necessary after breaking out of some cases, to ensure the user is directed back to the appropriate menu, until “Exit” is selected.

#Functions for user menus

#top level menu to select dish brand

userAction()

{

action=$(yad --width 300 --entry --title "Dish Collection Tracker" \

--image=gnome-shutdown \

--button="gtk-ok:0" --button="Exit:1" \

--text "Choose a dish brand:" \

--entry-text \

"Fiestaware" "Pyrex" "Hazel Atlas" "Exit")

ret=$?

}

### Dish Submenu | fiestaMenu ()

This function contains the secondary dish menu, as well as sets the variable for the user’s choice. pyrexMenu () and haMenu() also exist, serving the same purpose, but for the other dish brands.

#Fiestaware menu using yad to create drop down menu

fiestaMenu()

{

fiesta=$(yad --width 300 --entry --title "Fiestaware Dish Menu" \

--button="gtk-ok:0" \

--text "Choose next action:" \

--entry-text \

"See all Fiestaware" "View Owned Fiesta" "Review Wanted Fiesta" "Edit Owned Fiesta" "Update Wanted Fiesta" "Search for Fiesta" "Look at Fiesta Images" "Exit")

aaa=$?

}

#Pyrex secondary menu using yad for drop down menu

pyrexMenu()

{

pyrex=$(yad --width 300 --entry --title "Pyrex Dish Menu" \

--button="gtk-ok:0" \

--text "Choose next action:" \

--entry-text \

"See all Pyrex" "View Owned Pyrex" "Review Wanted Pyrex" "Edit Owned Pyrex" "Update Wanted Pyrex" "Search for Pyrex" "Look at Pyrex Images" "Exit")

bbb=$?

}

#Hazel Atlas secondary menu using yad to create drop down menu

haMenu()

{

ha=$(yad --width 300 --entry --title "Hazel Atlas Dish Menu" \

--button="gtk-ok:0" \

--text "Choose next action:" \

--entry-text \

"See all Hazel Atlas" "View Owned Hazel Atlas" "Review Wanted Hazel Atlas" "Edit Owned Hazel Atlas" "Update Wanted Hazel Atlas" "Search for Hazel Atlas" "Exit")

ccc=$?

}

Project Modules

Modules are used for procedural based code that does not require state data like class modules do. Complete the introduction to modules.

### Dish Brand Submenu | while true do switch statement menu

The Dish Brand Submenu for each dish brand is composed of a while true do menu. This menu uses a switch statement directed by the user’s input. For example, the Fiesta Brand Dish Submenu displays all the actions available related to Fiesta dishes. Until the user selects to exit the Fiesta menu, the user will continue to loop through the menu. This module consists of the following smaller menus: view all dishes, review wanted dishes, edit owned dishes, update wanted dishes, search for dishes, look at images, and exit.

The same Dish Submenu is reused for the Pyrex Submenu. The Hazel Atlas Submenu contains all but the ability to view images. This was done because the Hazel Atlas menu was designed to show the functionality of the program when a file is not found, so the image menu was not built due to time.

### Edit Owned Dishes Submenu | while true do, if/else statement, switch statement menu

The submenu to edit owned executes an if/else statement that is directed by the user’s input inside a do while true loop. This submenu is customized with brand-specific information (i.e. titles on yad dialogs), but otherwise is the same for each brand of dish.

If the file is not found, the if statement exits. Otherwise, if the file is found, the user is presented with a yad dialog to select to edit owned or wanted dishes (or to exit). This input then drives a switch statement to the appropriate action. Adding and removing dishes is somewhat similar, using grep and sed commands to edit the user’s choice and then redirecting that information to amend the appropriate .txt file.

### Update Wanted Dishes Submenu | while true do, if/else statement, switch statement

Similar to the Edit Owned Dishes Submenu, the Update Wanted Dishes Submenu is enclosed by a while true do loop. Then, an if/else statement is executed to determine what happens if the file does or does not exit. If the file exists to update, then a switch statement is executed, allowing the user to select what to do next. Until the user exits the Update Wanted Dishes section, this submenu is repeated.

### Search Dishes Submenu | while true do, if/then statement

To search all dishes, wanted dishes, and owned dishes of a brand, the Search Dishes Submenu is used. This menu uses a while true do command to loop through the menu until the user is done. User input will dictate while of the four if statements is executed, with the fourth if statement being to exit the program.

Within the if statements is a yad input box. This takes the user’s input and then through grep and sed commands, searches for the input in the appropriate file. The results are returned to the user, displayed in a yad textbox.

### Look Dishes Submenu | if/then statement

Viewing images occurs through the Look Dishes Submenu. If the .txt with images does not exist, the fileNotFound () function is executed. Othewise, the user is presented with a list of images to select, using radio buttons. Once selected, the image is displayed. The user is then brought back to the Dish Brand Submenu.

Program Start and End Flow

The user types “bash dishCollectionTracker” at the command line interface of Ubuntu. Then the user selects which dish brand to work with in the program. Within the dish brand submenu, the user can select various actions. The user can view all known dishes made of that brand, view owned dishes, or view wanted dishes. The use can also choose to add or remove dishes from the owned or wanted lists of that brand, which may be done after a piece is bought or sold from the user’s collection. The user could also search for specific dishes by keyword in the list of all dishes of that brand or in their own wanted, or owned dish lists. This functionality may be done if the user is out shopping or is trying to find out what known pieces exist in a brand. Each brand of dishes also contains some pre-selected images, which the user can select to view, aiding in piece identification. And lastly, the user can opt to exit a dish brand submenu.

Once done in a dish brand submenu, the user exits the dish brand submenu and is returned to the dish band menu. The user can result another dish brand or choose to exit the program.

User selects Exit. Returns to CLI.

Dish Collection Tracker GUI opens. User selects next action.

User executes bash script (bash dishCollectionTracker)

Exit displays message that user is going back to dish menu.

Look allows user to select from a list of available images to select one image to view.

Search option goes to another submenu. User can then search all dish list, owned dishes, or wanted dishes.

Edit and Update options display wanted or owned dish submenu. User can then add or remove dishes from respective list.

See, View, and Review display corresponding list of dishes.

User enters dish brand submenu

User selects from: See, View, Review, Edit, Update, Search, Look, Exit

User selects dish brand.

User enters dish brand submenu

No

User selects Exit?

Yes

Summary

Dish Collection Tracker is a set of files installed on the user’s device. The user should run Ubuntu on a Windows 10 machine for best results. An X Server for Windows will be needed to display the GUI. The user may prefer a wizard to setup the X Server, although any Unix installation should be able to support this program. And note, the user may need to install yad software for the program to run as intended.

Once the bash script is executed by typing “bash dishCollectionTracker” on the command line, the program will launch the first dialog. The user then interacts with the yad dialogs, through keyboard, mouse, and/or touchscreen input. The bash script utilizes do while true loops, switch statements, if/then, and if/else statements to display information to the user and take in user input. The user can read the data storage .txt files to display the content. The user can also view images through the dialogs. Any changes the user makes to the files (of owned and wanted dishes) is written to the file, thus saving changes so the user can view the updated .txt files when that menu option is selected.

# APPENDIX B (BUILD AND RELEASE PROCESS)

As new data is added to the .txt files (more dishes added to the fiestaDishes and pyrexDishes files for instance), an update will be required. To add the data, the programmer can utilize a text editor, such as Notepad++, to continue adding entries to the files. Once Fiestaware and Pyrex files are completed, then developing the Hazel Atlas all dish file is recommended.

As new images are sourced, resized, and uploaded to the appropriate Image directories, these files will also need to be pushed to the users.

Each update’s build will be the result of all new and updated files. A .txt file will also be prepared that documents all the changes. All these files will be made available on Dish Collection Tracker’s GitHub page. As there is currently no charge for this program, this free distribution method will be sufficient at this time.

Users, following the included update instructions, will upload the new files, replacing their existing files. For example, in the first update there will be more dishes added to fiestaDishes.txt and pyrexDishes.txt. The update can have new data storage file names, such as fistaDishes1.txt and pyrexDishes1.txt to represent the version, providing the bash script is also updated to read the new files. This way the user can just upload all new data storage files from each update. However, this would likely mean the bash script is also renamed. Since this program already requires advanced user abilities to install, it is realistic that the user would likely be able to understand and follow the directions to run the new script (i.e. bash dishCollectionTracker1).

It will be up to the users to check for new versions and updates of Dish Collection Tracker at this time. It is reasonable as the use of this program spreads in the future, keeping track of users’ emails in order to notify them of an update could be a realistic component of the build and release process.

# APPENDIX C (CLIENT INSTALLATION INSTRUCTIONS)

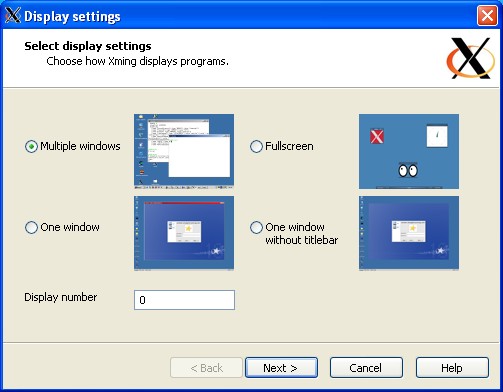
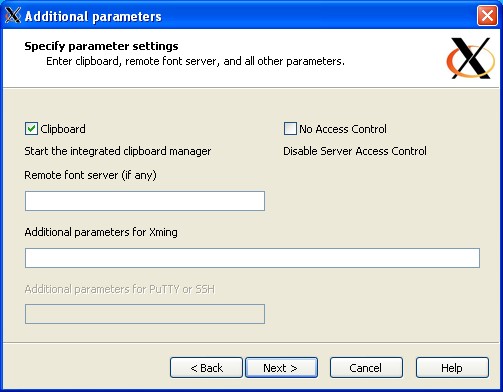
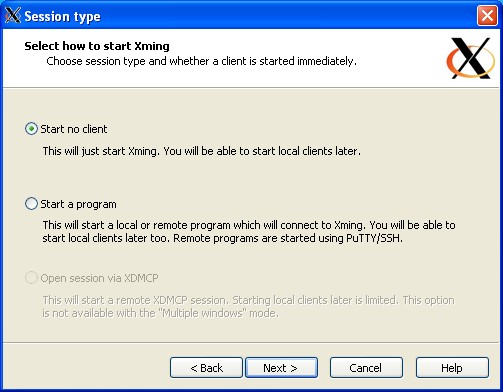
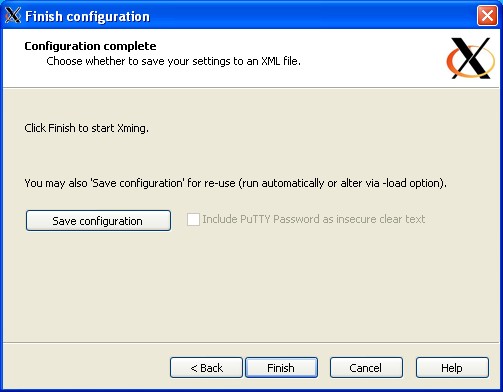
On a Windows machine, install Ubuntu if not already installed. Installation can be completed through Microsoft Windows App Store (<https://www.microsoft.com/en-us/store/apps/windows>). Search for Ubuntu App, selecting the newest version or the version the machine it will be installed on will support.

Once installed, launch Ubuntu. Next, install yad, if needed, by typing into the command line:  
 sudo apt update  
 sudo apt install yad

A X Server will need to be installed as well. Xming is recommended. If using Xming, it can be downloaded from here: <http://sourceforge.net/project/downloading.php?group_id=156984&filename=Xming-6-9-0-31-setup.exe>.

Instructions can be found online, at websites such as this one: <http://www.geo.mtu.edu/geoschem/docs/putty_install.html>.

If Xming is used, run XLaunch (which is a wizard to configure Xming). Follow the below steps in the wizard:

Next, download the Dish Collection Tracker files (located in DishCollectionTracker directory and Images directory on GitHub at: <https://github.com/Daisymoo/DishCollectionTracker>).

Note: Files cannot be moved directly from Windows Subsystem to Ubuntu.

1. First, install the files on the Windows machine, such as the C: Downloads.
2. Next, run Ubuntu. Navigate to the /mnt/c/Users/username folder. Select the location the Dish Collection Tracker files are in (i.e. /mnt/c/Users/username/Downloads).
3. Then move the Dish Collection Tracker files (mv DishCollectionTracker home/username/DishCollectionTracker and mv Images home/username/DishCollectionTracker). These files need to be installed in the user’s home location (i.e. home/user).
4. Next, execute the script (bash dishCollectionTracker).
5. If installed correctly, Dish Collection Tracker dish brand selection menu will appear.

# APPENDIX D (DEVELOPER SETUP INSTRUCTIONS)

Developers install the same components as the clients: Ubuntu (or other Linux/Unix environment), an X Server for Windows, and yad.

On a Windows machine, install Ubuntu if not already installed. Installation can be completed through Microsoft Windows App Store (<https://www.microsoft.com/en-us/store/apps/windows>). Search for Ubuntu App, selecting the newest version or the version the machine it will be installed on will support.

Once installed, launch Ubuntu. Next, install yad, if needed, by typing into the command line:  
 sudo apt update  
 sudo apt install yad

A X Server will need to be installed as well. Xming is recommended. If using Xming, it can be downloaded from here: <http://sourceforge.net/project/downloading.php?group_id=156984&filename=Xming-6-9-0-31-setup.exe>.

Instructions can be found online, at sites such as this one: <http://www.geo.mtu.edu/geoschem/docs/putty_install.html>. If Xming is used, run XLaunch (which is a wizard to configure Xming).

Next, the user needs to download the Dish Collection Tracker files (located in DishCollectionTracker directory and Images directory on GitHub at: <https://github.com/Daisymoo/DishCollectionTracker>). Files cannot be moved directly from Windows Subsystem to Ubuntu.

1. First, install these files on the Windows machine, such as the C: Downloads.
2. Next, run Ubuntu. Navigate to the /mnt/c/Users/username folder. Select the location the Dish Collection Tracker files are in (i.e. /mnt/c/Users/username/Downloads).
3. Then move the Dish Collection Tracker files (mv DishCollectionTracker home/username/DishCollectionTracker and mv Images home/username/DishCollectionTracker). These files need to be installed in the user’s home location (i.e. home/user).
4. Once all programs are installed and the Dish Collection Tracker files moved, execute the script (bash dishCollectionTracker).
5. The dish brand selection menu will appear, which can be navigated using the mouse, keyboard, and/or touchscreen input.

In addition to the above Client Setup Instructions, the developers may need to install a text editor in the Linux or Windows system. For developers who prefer to work from Linux, a text editor such as VI/VIM editor or nano could be used, allowing for interaction through the CLI or GUI. For developers who prefer to work in Windows programs such as Notepad++ or Brackets can be used. In Windows, as .txt files are edited, the new file has to be copied and then pasted into the Linux CLI.

Developers also need the ability to edit photos. A program such as Microsoft Photos, which is included on Windows 10 machines, can be used to resize photos.

For debugging the script, using the built-in Unix/Linux tools such as set -x to execute a script (bash -x dishCollectionTracker), can be useful. Developers working on the .txt data storage files will need access to dish history resources (i.e. websites or printed books) or access to a knowledgeable expert to add dish entries to the .txt files.

Lastly, developers need access to Dish Collection Tracker’s GitHub repository to upload new program files (<https://github.com/Daisymoo/DishCollectionTracker>). This is the location program files will be stored for clients (users) to access.