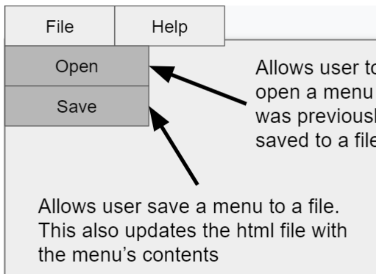
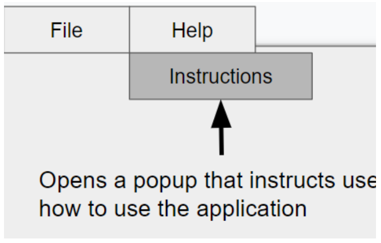
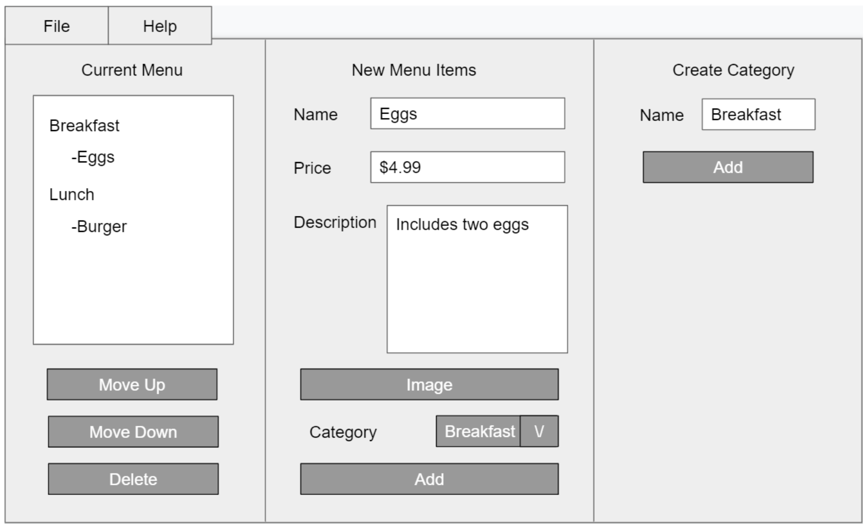
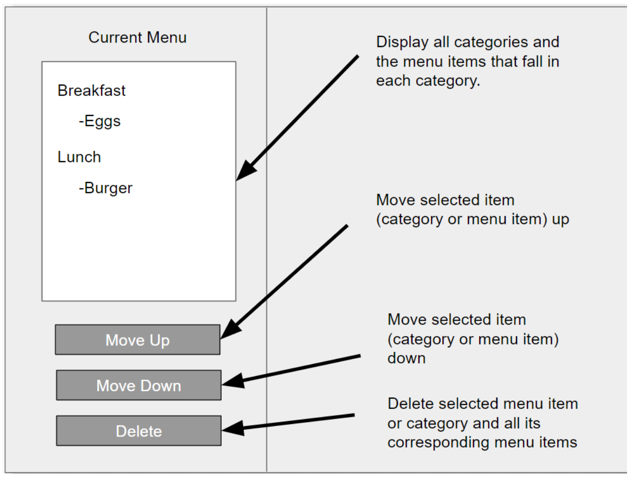
Criterion B: Design

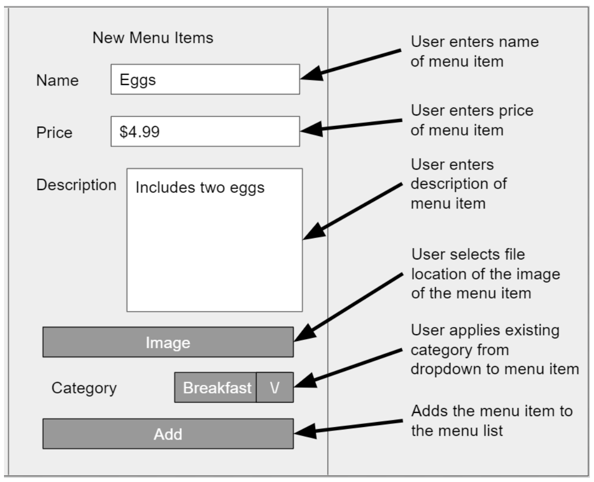
GUI sketch

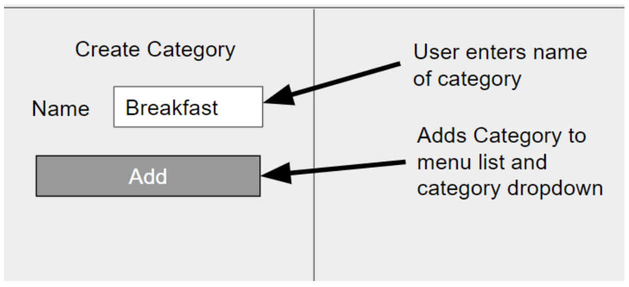

File
Help
Open
Save
Allows user to open a menu was previousl saved to a file
Allows user save a menu to a file. This also updates the html file with the menu's contents



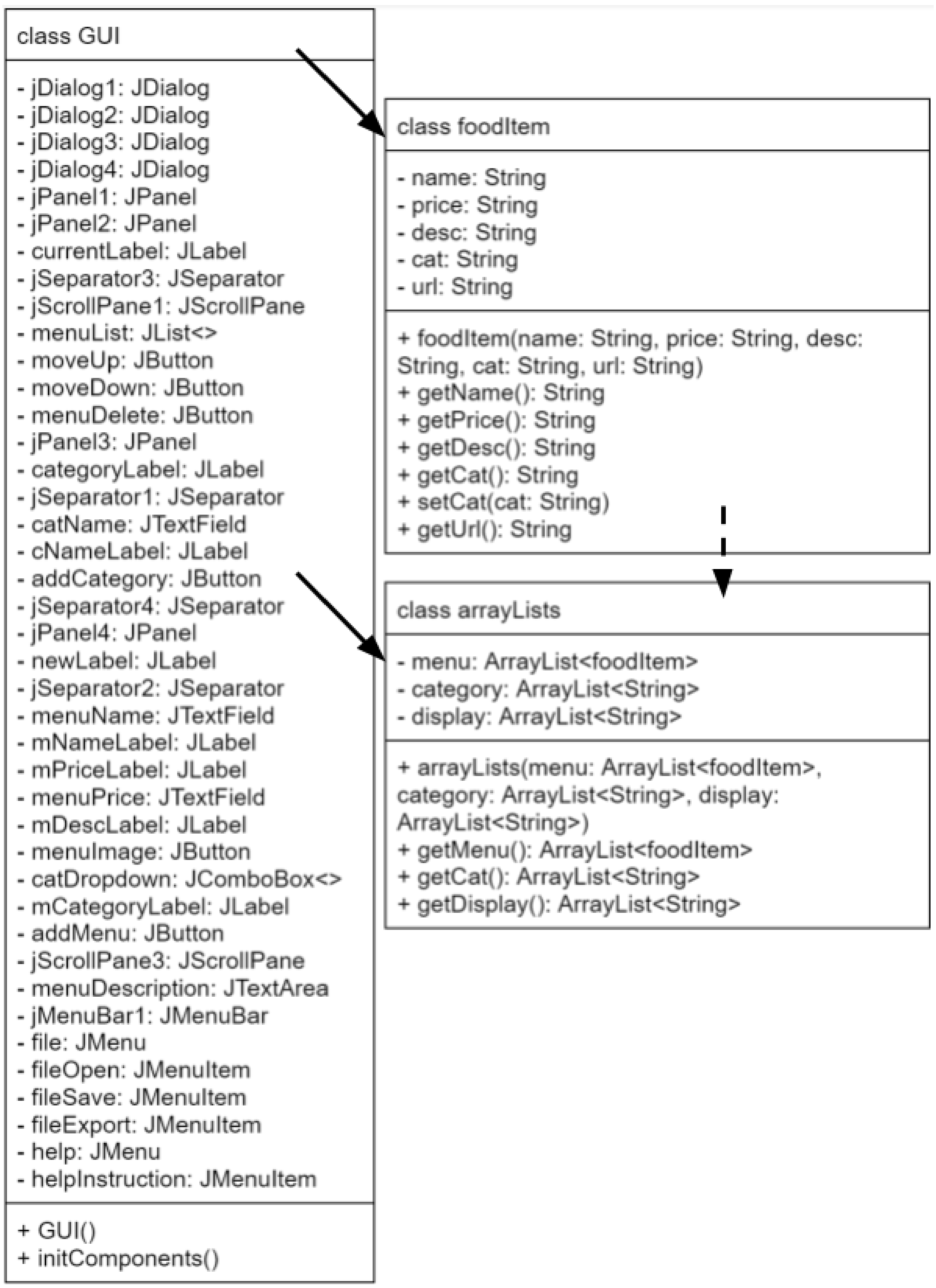



Current Menu
Breakfast
-Eggs
Lunch
-Burger
Move Up
Move Down
Delete
Display all categories and the menu items that fall in each category.
Move selected item (category or menu item) up
Move selected item (category or menu item) down
Delete selected menu item or category and all its corresponding menu items


New Menu Items
Name
Eggs
Price
$4.99
Description
Includes two eggs
Image
Category
Breakfast
V
Add
User enters name of menu item
User enters price of menu item
User enters description of menu item
User selects file location of the image of the menu item
User applies existing category from dropdown to menu item
Adds the menu item to the menu list


Create Category
Name
Breakfast
Add
User enters name of category
Adds Category to menu list and category dropdown

Class diagrams



```
class GUI
- jDialog1: JDialog
- jDialog2: JDialog
- jDialog3: JDialog
- jDialog4: JDialog
- jPanel1: JPanel
- jPanel2: JPanel
- currentLabel: JLabel
- jSeparator3: JSeparator
- jScrollPane1: JScrollPane
- menuList: JList<>
- moveUp: JButton
- moveDown: JButton
- menuDelete: JButton
- jPanel3: JPanel
- categoryLabel: JLabel
- jSeparator1: JSeparator
- catName: JTextField
- cNameLabel: JLabel
- addCategory: JButton
- jSeparator4: JSeparator
- jPanel4: JPanel
- newLabel: JLabel
- jSeparator2: JSeparator
- menuName: JTextField
- mNameLabel: JLabel
- mPriceLabel: JLabel
- menuPrice: JTextField
- mDescLabel: JLabel
- menuImage: JButton
- catDropdown: JComboBox<>
- mCategoryLabel: JLabel
- addMenu: JButton
- jScrollPane3: JScrollPane
- menuDescription: JTextArea
- jMenuBar1: JMenuBar
- file: JMenu
- fileOpen: JMenuItem
- fileSave: JMenuItem
- fileExport: JMenuItem
- help: JMenu
- helpInstruction: JMenuItem
+ GUI()
+ initComponents()
```

```
class foodItem
- name: String
- price: String
- desc: String
- cat: String
- url: String
+ foodItem(name: String, price: String, desc:
String, cat: String, url: String)
+ getName(): String
+ getPrice(): String
+ getDesc(): String
+ getCat(): String
+ setCat(cat: String)
+ getUrl(): String
```

```
class arrayLists
- menu: ArrayList<foodItem>
- category: ArrayList<String>
- display: ArrayList<String>
+ arrayLists(menu: ArrayList<foodItem>,
category: ArrayList<String>, display:
ArrayList<String>)
+ getMenu(): ArrayList<foodItem>
+ getCat(): ArrayList<String>
+ getDisplay(): ArrayList<String>
```

Input data

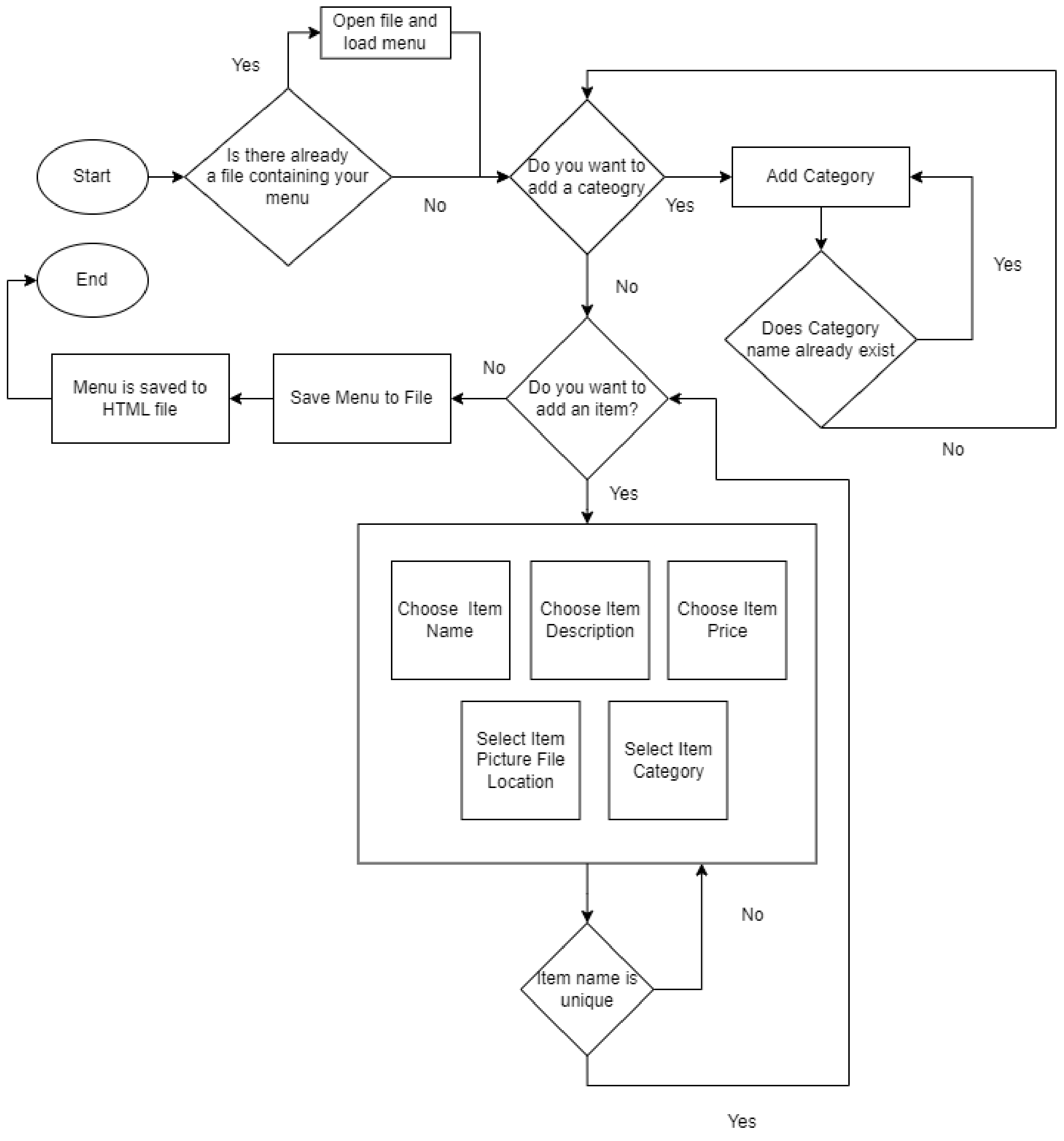
1. Category name	
Ex: Name: Breakfast	Ex: Name: Lunch
Limitations for creating category: ● A new category cannot be added if one with the same name already exists	
2. Menu item information, including name, price, description, category and location of image	
Ex: Name: Eggs Price: $5.99 Description: Two sunny side up eggs. Category: Breakfast Image: /img/eggs.jpg	Ex: Name: Burger Price: $13.99 Description: Double patty burger Category: Lunch Image: /img/burger.jpg
Limitations for creating menu item: ● A new item cannot be added if one with the same name already exists ● The category must be selected from a pre existing category ● The file location for the image must be selected	
Comment The user can rearrange the order of the menu items, which will change the output data	

Output data

1. One txt file containing the data of the menu. This can be opened later to be edited by the application.

2. One html file that can be opened to display a website reflecting the menu that was created.

Process description


Start
Is there already a file containing your menu
Yes
Open file and load menu
No
Do you want to add a cateogry
Yes
Add Category
Does Category name already exist
Yes
No
No
End
Do you want to add an item?
No
Save Menu to File
Menu is saved to HTML file
Yes
Choose Item Name
Choose Item Description
Choose Item Price
Select Item Picture File Location
Select Item Category
Item name is unique
No
Yes

Pseudocode for methods

```
moveUp():
  1. Check if the selected index in the menuList JList is less than or equal to 0
  2. If the selected index is less than or equal to 0, do nothing and return
  3. If the selected index is greater than 0, swap the selected item with the item above it in the list
  4. Select the moved item in the list
  5. Return

moveDown():
  1. Check if the selected index in the menuList JList is greater than or equal to the last index in the list
  2. If the selected index is greater than or equal to the last index, do nothing and return
  3. If the selected index is less than the last index, swap the selected item with the item below it in the list
  4. Select the moved item in the list
  5. Return

addMenu():
 1: Check if catDropdown has no selected item
 2: If no item is selected, display warning message to select a category
 3: Check if selectedImage is null
 4: If selectedImage is null, display warning message to select an image and return mcdlist
 5: Iterate through mcdlist.getMenu()
 6: If an element in mcdlist.getMenu() has the same name as menuName.getText(), display warning message that a food item already has that name and return mcdlist
 7: Set sourceFile equal to selectedImage and set imagefilepath equal to the path of selectedImage
 8: Set currentDirectory to the current working directory and set destinationFile equal to a file in the assets/img folder with the name of sourceFile
 9: Try to copy sourceFile to destinationFile, replacing any existing file
 10: If an IOException occurs, display the error message
 11: Add a new foodItem object to mcdlist.getMenu() with the values menuName.getText(), menuPrice.getText(), menuDescription.getText(), the selected item in catDropdown, and imagefilepath
 12: Call updateMenuList with mcdlist as an argument
 13: Set selectedImage to null
 14: Return mcdlist
```

menuDelete():

```
1. Check if an item is selected in the menu list. If not, do nothing.
2. Check if the selected item is a category. If it is, remove it and all the menu items in that
   category from the arrayLists object.
3. If the selected item is a menu item, remove it from the arrayLists object.
4. Update the menu list with the updated arrayLists object.
```

updateMenuList():

```
1. Clear the display list
2. For each category in the mcdlist object:
   a. Add the category to the display list
   b. For each menu item in the mcdlist object:
         i. If the menu item belongs to the current category, add it to the display list
3. Set the model of the menu list to the display list
```

fileOpen():

```
1. Create a FileInputStream to read the object from a file
2. Create a file chooser object
3. Show the file chooser and get the user's response
4. If the user selected a file, set the object to the object in its contents
5. Get the selected file
6. Create an ObjectInputStream to read the object from the file
7. Read the object from the file
8. Catch any IOExceptions or ClassNotFoundExceptions that may occur
9. Remove all prexisting categories in the dropdown
10. Add the categories from the new menu to the dropdown
11. Update the menu list
```

fileSave():

```
1. Initialize a JFileChooser object named "fileChooser"
2. Set the file selection mode of "fileChooser" to only allow directories to be selected
3. Show the file chooser and store the user's response in a variable "result"
   If the user selected a file:
   a. Initialize a File object named "selectedDirectory" to the selected file
   b. Initialize a File object named "newFile" with the selectedDirectory as its parent
   directory and the name "menu.txt"
   c. If "newFile" does not exist, create it
   d. Open an ObjectOutputStream on "newFile" and store it in a variable "oos"
   e. Write the object "mcdlist" to the output stream
   f. Close the output stream
4. g. Call the HTMLedit function with "mcdlist" as the input
```

```
HTMLedit():
    1.  Set the value of the currentDirectory variable to the current working directory.
    2.  Initialize the selectedFile variable to the file at the currentDirectory + the file separator +
        "untitled.html".
    3.  Initialize the reader variable to a new BufferedReader with the selectedFile as its
        argument.
    4.  Declare the line and key variables.
    5.  Initialize the html variable to a new StringBuilder.
        While the reader can read a line, do the following:
        a. Set the value of line to the next line read by the reader.
        b. If line contains key[0], do the following:
    6.  i. For each element in mcdlist.getCat(), do the following:
            1.  Append the element to html.
                Append line to html.
                c. Else, append line to html.
    7.  Close the reader.
    8.  Initialize the writer variable to a new PrintWriter with the selectedFile as its argument.
    9.  Print the html to the selectedFile using the writer.
    10. Close the writer.
    11. Re-initialize the reader variable to a new BufferedReader with the selectedFile as its
        argument.
    12. Re-initialize the html variable to a new StringBuilder.
        While the reader can read a line, do the following:
        a. Set the value of line to the next line read by the reader.
        b. If line contains key[1], do the following:
    13.  For each element in mcdlist.getCat(), do the following:
            1.  Append the element to html.
                For each element in mcdlist.getMenu(), do the following:
                    a. If the element's category is equal to the current element in
                    mcdlist.getCat(), append it to html.
                    c. Else, append line to html.
    14. Close the reader.
    15. Re-initialize the selectedFile variable to the file at the currentDirectory + the file
        separator + "index.html".
    16. Re-initialize the writer variable to a new PrintWriter with the selectedFile as its argument.
    17. Print the html to the selectedFile using the writer.
    18. Close the writer.
```

Schedule for developing the product
Program will be divided in two sections: the menu generator, and the html editor; The menu generator will generate an object containing all menu items with descriptions in order. The html editor will make changes to the website.

GUI (3 weeks)	Html editor (1 week)
1. Set up the project structure and create the GUI class 2. Implement the moveUp and moveDown buttons 3. Implement the delete button 4. Implement the addCategory button 5. Implement the addMenu button 6. Implement the fileOpen and fileSave menu items 7. Implement the fileExport menu item 8. Implement the helpInstruction menu item 9. Test the code and debug any issues 10. Finalize the code and prepare for deployment.	1. Identify the location in the HTML file where the new data will be appended. 2. Write the code to open the HTML file and read it line by line. 3. Write the code to search for the target line in the HTML file and store its location. 4. Write the code to append the new data from the mcdlist to the target line in the HTML file. 5. Test the new method to verify that it works as expected. 6. Refine the code as needed based on the results of the testing. 7. Integrate the new method into the existing code and verify that it works correctly with the rest of the application. 8. Test the entire application to ensure that the new method does not cause any issues or break existing functionality.

Ideas for Expansion

1. Add the ability to edit existing menu items and categories. This could include adding form fields and buttons for updating the name, price, description, and image of a menu item, as well as the ability to change the category that a menu item belongs to.
2. Implement search functionality to allow users to search for menu items by name, price, or category. This could include adding a search bar and search button to the user interface, as well as implementing the logic for filtering the menu items based on the search criteria.
3. Implement functionality for creating and managing multiple menus, such as the ability to create new menus, switch between existing menus, and delete menus. This could include adding options to the user interface for managing multiple menus, as well as implementing the logic for creating, switching, and deleting menus.
4. Add the ability to import and export menu items and categories from and to external file formats, such as CSV or JSON.

Test Plan

Action Test	**Way of testing and result**
Verify that the GUI is displayed correctly.	Test case: Launch the application and verify that the user interface is displayed correctly, with all the buttons and labels in their expected positions. result: The user interface is displayed correctly
Verify that the moveUp and moveDown buttons function properly, including the inability to move items out of their respective categories, and being allowed to change category order	Test case: Add several menu items to the menuList, including items in different categories. Select a menu item and click on the moveUp or moveDown buttons. result: The selected menu item moves up or down in the list as expected. The menu item cannot be moved out of its category, and the category order can be changed by moving items between categories.
Verify that the delete button removes the selected menu item from the menuList	Test case: Add several menu items to the menuList. Select a menu item and click on the delete button. result: The selected menu item is removed from the menuList. result: The selected menu item is removed from the menuList
Verify that the addCategory button adds a new category to the catDropdown	Test case: Enter a new category name in the text field and click on the addCategory button. result: The new category appears in the catDropdown.
Verify that the addMenu button adds a new menu item to the menuList with variables such as ‘name’, ‘description’, ‘price’, ‘picture’ and ‘category’	Test case: Enter the values for the name, description, price, picture, and category of a new menu item in the corresponding text fields and click on the addMenu button. result: The new menu item is added to the menuList with the specified values.
Verify that the fileOpen menu item opens a file and displays the contents in the menuList	Test case: Click on the fileOpen menu item and select a file to open. result: The contents of the selected file are displayed in the menuList.
Verify that the recently opened menu file is affected by buttons	Test case: Open a menu file, make changes to the menuList using the buttons (e.g. add or delete menu items), and click on the fileSave menu item to save the changes. result: The changes made to the menuList are saved to the opened menu file.
Verify that the fileSave menu item saves the current state	Test case: Make changes to the menuList and click on the fileSave menu item. Select a location and file name to save the menuList.

of the menuList to a file where the client chooses	result: The current state of the menuList is saved to the selected file.
Verify that the fileExport menu item exports the current state of the menuList to a PDF file	Test case: Make changes to the menuList and click on the fileExport menu item. Select a location and file name to export the menuList. result: The current state of the menuList is exported to a PDF file with the specified name and location.
Verify that the helpInstruction menu item displays the instructions for using the GUI in a new window.	Test case: Click on the helpInstruction menu item. Result: A new window is opened with the instructions for using the GUI.
Verify that the category dropdown is updated alongside the addition and deletion of categories	Test case: Add and delete categories using the addCategory and delete buttons, and verify that the changes are reflected in the catDropdown. Result: The catDropdown is updated to reflect the changes made to the categories.
Verify that the current menu list will automatically group the menu items under their corresponding category	Test case: Add menu items with different categories to the menuList and verify that the items are grouped under their corresponding categories. result: The menu items are grouped under their corresponding categories in the menuList.
Verify that info messages will appear on the screen in the case of data-entry errors	Test case: Attempt to add a menu item with invalid data (eg. same data as a previous item) and verify that an error message appears on the screen. result: An error message appears on the screen indicating the data-entry error. (eg. ‘A menu item with this name already exists’)
Verify that the html file is edited according to categories and food items	Test case: Add categories and food items to the menu list and verify that the corresponding changes are made to the HTML file. result: The HTML file is edited to reflect the changes made to the menu list, including the addition and deletion of categories and food items.
Verify that the website is accurate to the menu list created by the client	Test case: Add categories and food items to the menu list and verify that the changes are reflected on the website. result: The website is updated to reflect the changes made to the menu list, including the addition and deletion of categories and food items.