**Stage 2: Development Report**

Michael Johnston

(1800329)

Table of Contents

[Introduction 3](#_Toc37854499)

[User Documentation 3](#_Toc37854500)

[Business Model Implementation 3](#_Toc37854501)

[Does the Business Model Match the Design? 3](#_Toc37854502)

[Inheritance and Associations 7](#_Toc37854503)

[Scope Operators 7](#_Toc37854504)

[Setters, Getters and Properties 7](#_Toc37854505)

[Constructors 7](#_Toc37854506)

[Method and Attribute Correct Coding 7](#_Toc37854507)

[User Interface Implementation 9](#_Toc37854508)

[Does the User Interface Match the Design? 9](#_Toc37854509)

[User Interface Controls, Validation and Error Prevention 19](#_Toc37854510)

[Coding of Appropriate Events and Links to the Business Model 19](#_Toc37854511)

[Usage of Unfamiliar Libraries and Constructs 23](#_Toc37854512)

[Microsoft.Data.Sqlite library and SQLitePCLRaw.bundle\_winsqlite3 package 23](#_Toc37854513)

[Context Menu Construct 24](#_Toc37854514)

[Saving, Loading, Writing and Reading to Files 24](#_Toc37854515)

[Ordering List Item Objects by Property 25](#_Toc37854516)

[Returning Database Entries from a Given Range of Dates 25](#_Toc37854517)

[Error Handling 25](#_Toc37854518)

[User Interface Error Handling and Validation 25](#_Toc37854519)

[Business Model Error Handling 27](#_Toc37854520)

[Internal Documentation 28](#_Toc37854521)

[Testing 29](#_Toc37854522)

[Testing Strategy 29](#_Toc37854523)

[White Box Testing 30](#_Toc37854524)

[Black Box Testing 33](#_Toc37854525)

[Scenario 1 34](#_Toc37854526)

[Scenario 2 36](#_Toc37854527)

[Scenario 3 37](#_Toc37854528)

[Scenario 4 38](#_Toc37854529)

[References 40](#_Toc37854530)

# Introduction

This document is intended to serve as evidence for the development of the Mixing Desk app in that it will showcase the various aspects of the app’s implementation such as:

* Business model implementation (Does the code match the design as was originally envisioned? Does the code make proper use of coding constructs, operators, inheritance, setters, getters etc?)
* View Model implementation (Does the UI match the original design? Are UI controls being properly used? Is input being validated? Event handler usage and links between the view model and business model)
* Unfamiliar library and/or coding constructs usage
* Error handling
* Internal documentation
* Testing
* User documentation

# User Documentation

A user guide has been written on the operation of the Mixing Desk app in the attached file called “mixingDeskUserGuideMichaelJohnston”. It is strongly recommended that you read that before continuing as this guide will give you an in depth understanding of how the app works and how a user might use the app.

# Business Model Implementation

We will now provide evidence as to the litany of important aspects a business model should contain.

## Does the Business Model Match the Design?

In the attached file businessModelUMLClassDiagram.pdf the original UML business model is shown. The implementation is very similar to this model having largely the same relationships between classes. Shown below in figures 1-4 are the four classes implemented.

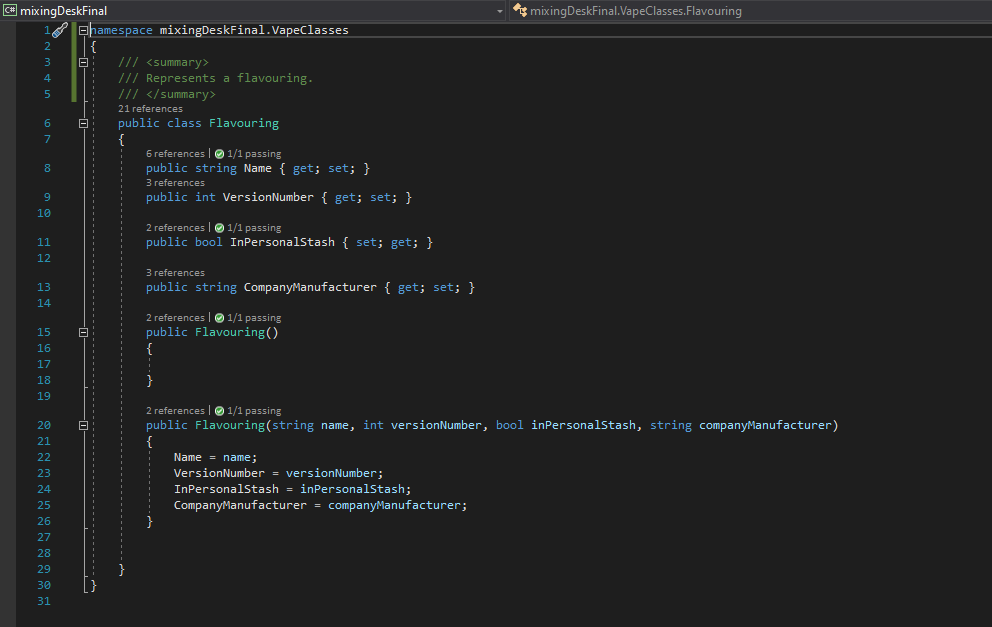


Figure : Flavouring class

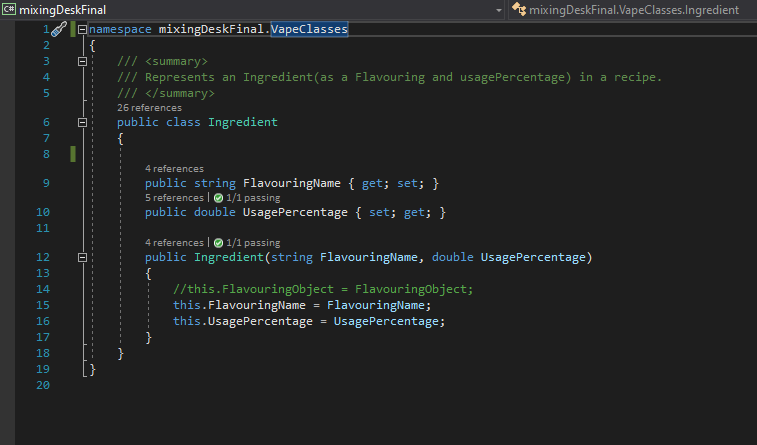


Figure : Ingredient class

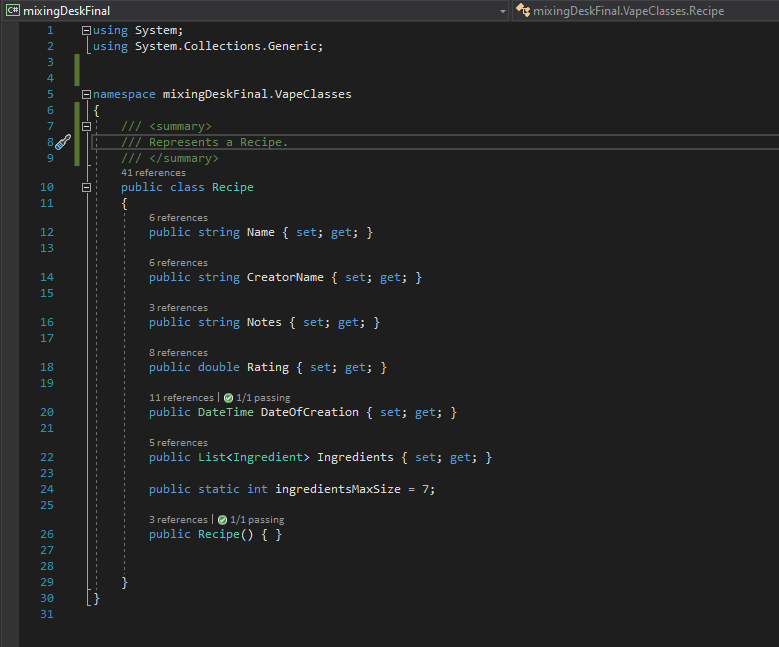


Figure : Recipe class

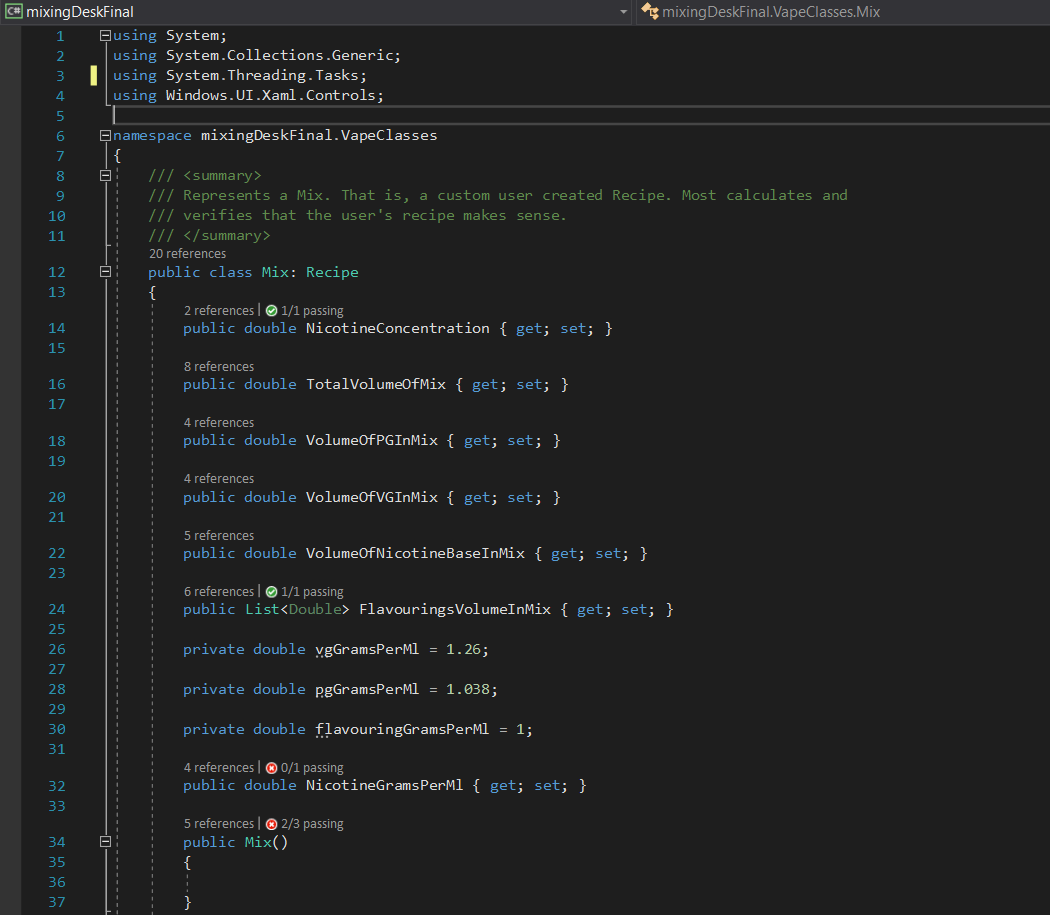


Figure : Mix class

From these figures we can see that most of the class members are the same as was intended in the UML diagram. The idNumber fields in each class were unnecessary in this implementation. Most of the setters and getters stated in the UML are also still present in the form of sets and gets on the properties of the classes. The Ingredient class is also slightly different in that it is not so strongly tied to the flavouring class in that the flavouring component of the Ingredient class is instead just a string.

Furthermore, there is DataAccess class in this prototype. This class deals with all of the database reading, writing and updating. This class could not be included in the original design because at the time this developer didn’t know how to integrate Sqlite with a UWP project.

In conclusion the code matches the design very closely.

## Inheritance and Associations

In figure 4 you will notice that the Mix class inherits from the Recipe class. This is an appropriate use of inheritance given that a Mix is simply a custom user recipe and has many of the same attributes, name, date of creation, notes, an ingredients list. The Mix class specializes in that it deals with calculation of the user’s custom recipe or calculation of any general recipe.

In figure 3 we can see that the Recipe class holds a list of Ingredient objects. This is an appropriate usage of an association which vastly simplifies the information that any given Recipe object is holding.

## Scope Operators

In figures 1-4 we can also see the usage of scope operators such as the public members of the Recipe class. The properties Ingredients, Rating, Notes, Name etc. must be public in order to be used outside the class by databinding into lists etc. This is the same for the Ingredient and Company class. The Mix class has some private operators which only the Mix class needs to know about. The public properties don’t seem to need to be public though they were very useful in testing and could be useful in later iterations. All event handlers are private.

## Setters, Getters and Properties

In figures 1-4 we can see many Properties all of which have the option to set and get them. This is necessary in order to read and alter their values in the code. This is an appropriate use of setters, getters and Properties.

## Constructors

In figures 1-4 we can also see all of the constructors in the business model. In the Flavouring class in figure 1 we see the use of an overloaded constructor.

## Method and Attribute Correct Coding

Given that you have by now tried running the Mixing Desk application it is safe to assume that the methods and attributes have been coded correctly in that they do not throw any unresolved exceptions. This is also further demonstrated in the Testing section later in this document. In figures 5-7 below, examples of correctly coded methods are shown.



Figure : PGVGValidator method in Mix class

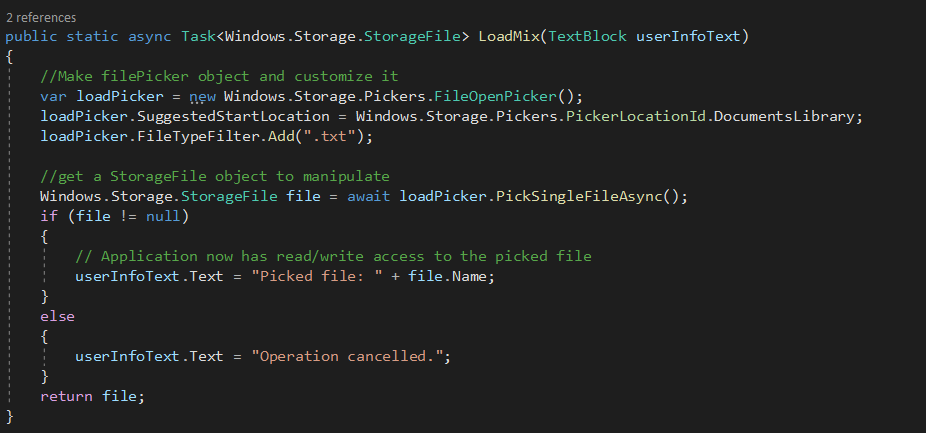


Figure : LoadMix method in Mix class

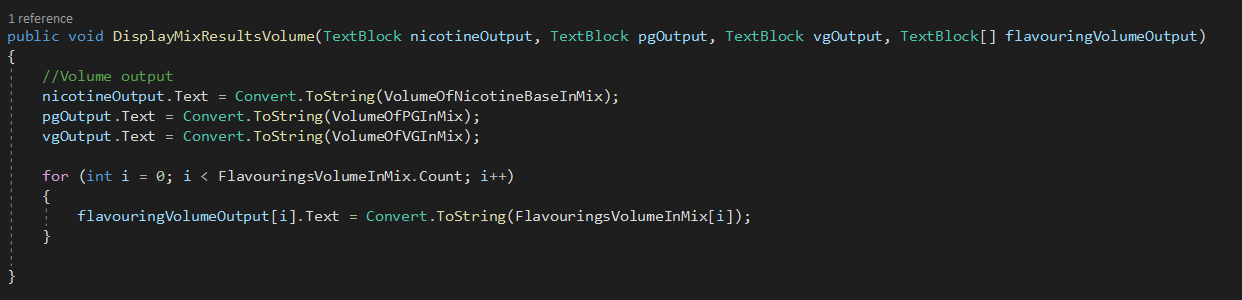


Figure : DisplayMixResultsVolume method in Mix class

# User Interface Implementation

## Does the User Interface Match the Design?

We will now show the various pages that were designed for the Mixing Desk app and compare them to the final implemented solution in figures 8-19.

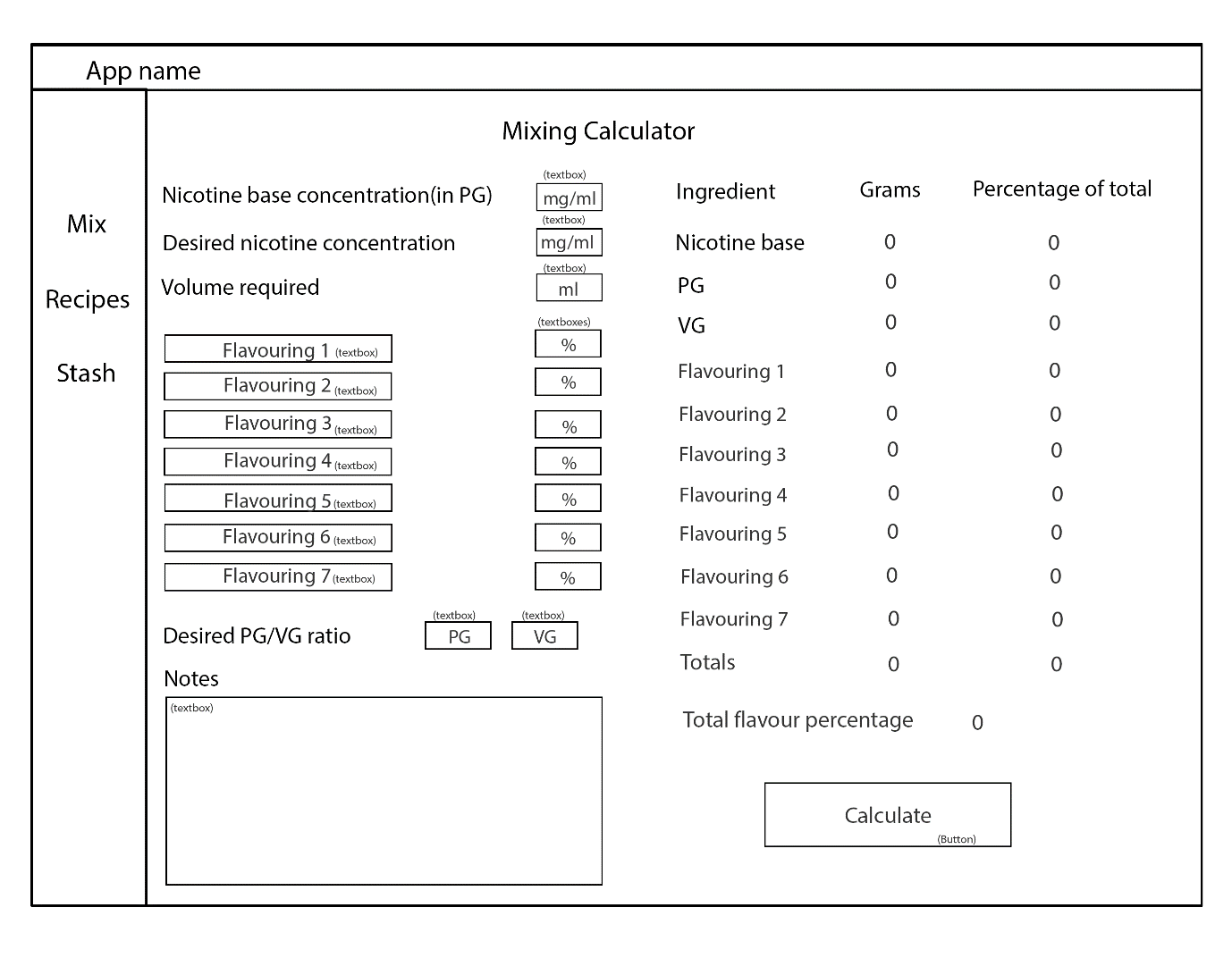


Figure : Mixing Calculator page wireframe design



Figure : Mixing Calculator page in final prototype.

The Mixing Calculator page in the final prototype application is very similar to the wireframe design. All that is added is a volume output column (because some people do mix e-liquid that way) and a button for saving and loading recipes.

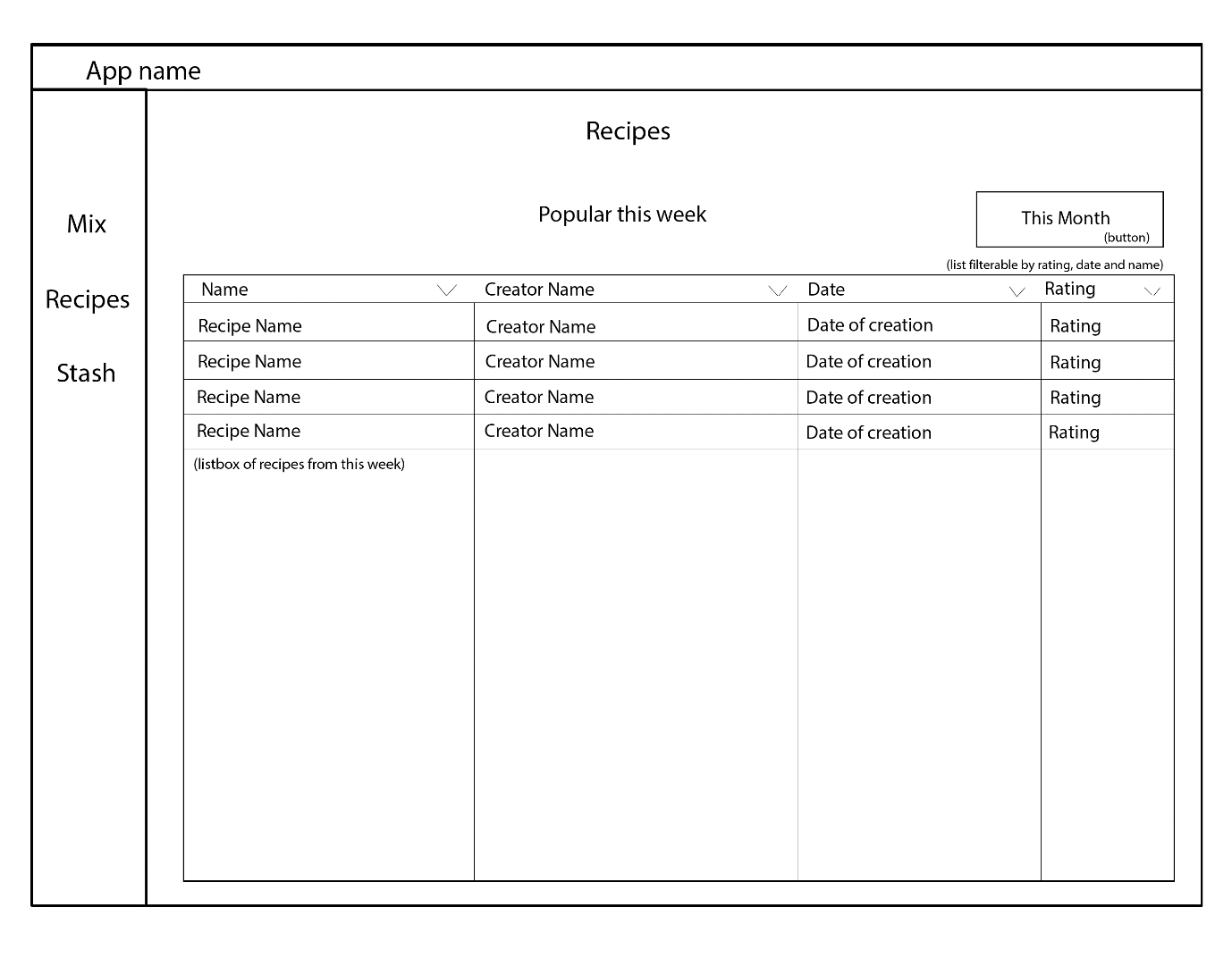


Figure : Recipes This Week page wireframe design.



Figure : Recipes This Week page in final prototype.

The Recipes This Week page is very similar to the original design though the column headers were not possible to implement because list views in UWP do not have the option for column headers built in. Therefore, pseudo column headers were added, and the Date and Rating headers do have the expected functionality of sorting the list. The same is true for the Recipes This Month and Recipes This Year pages.

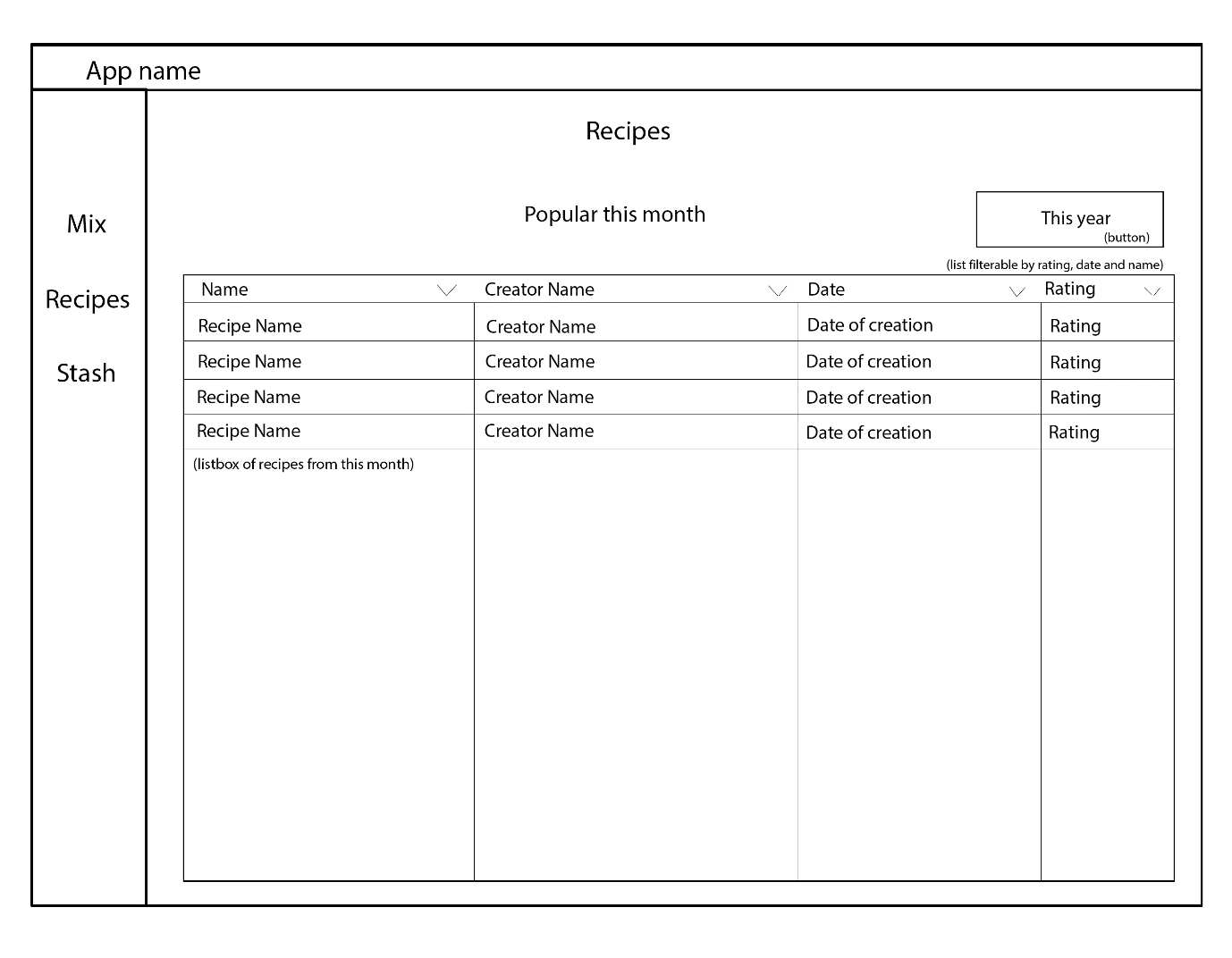


Figure : Recipes This Month page wireframe design.

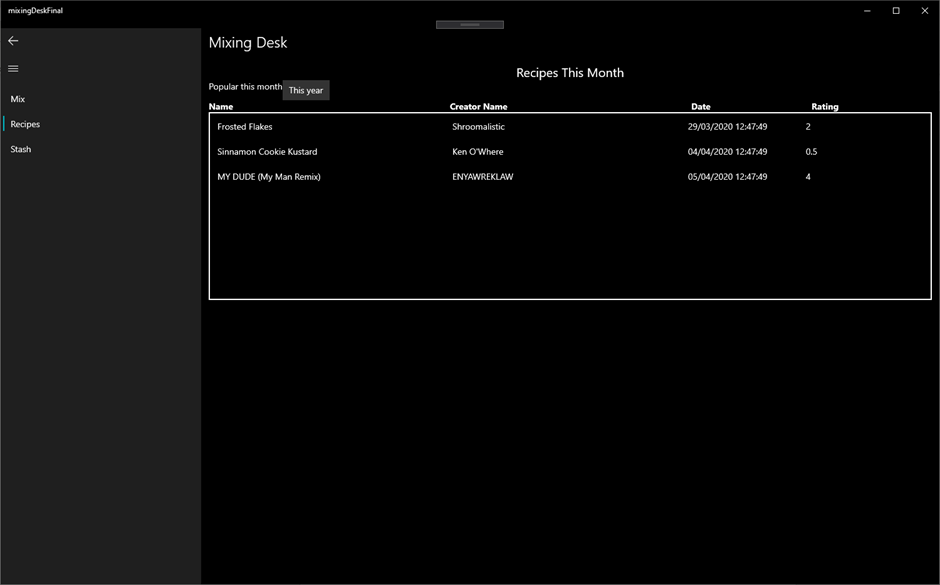


Figure : Recipes This Month page in final prototype.

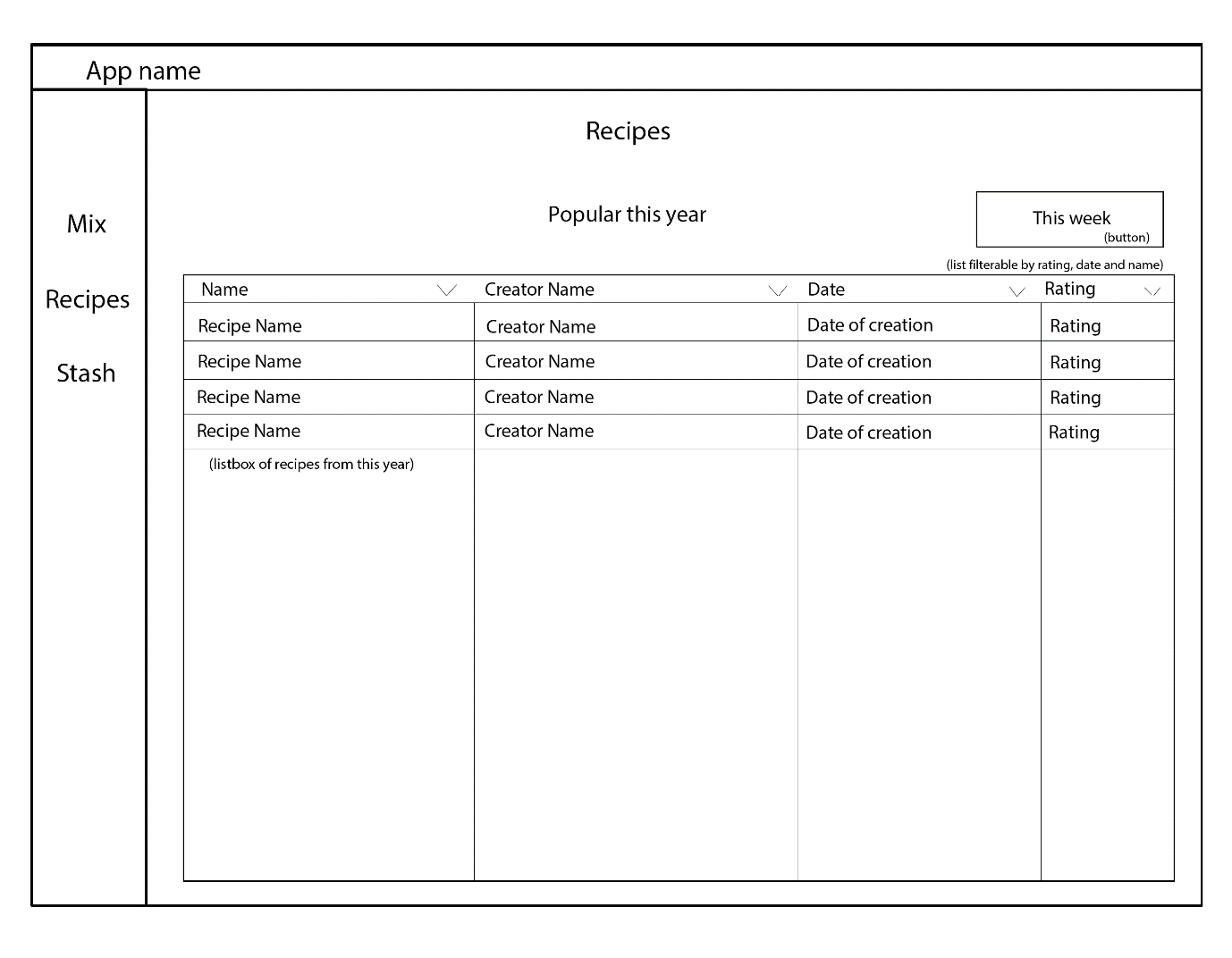


Figure : Recipes This Year page wireframe design.

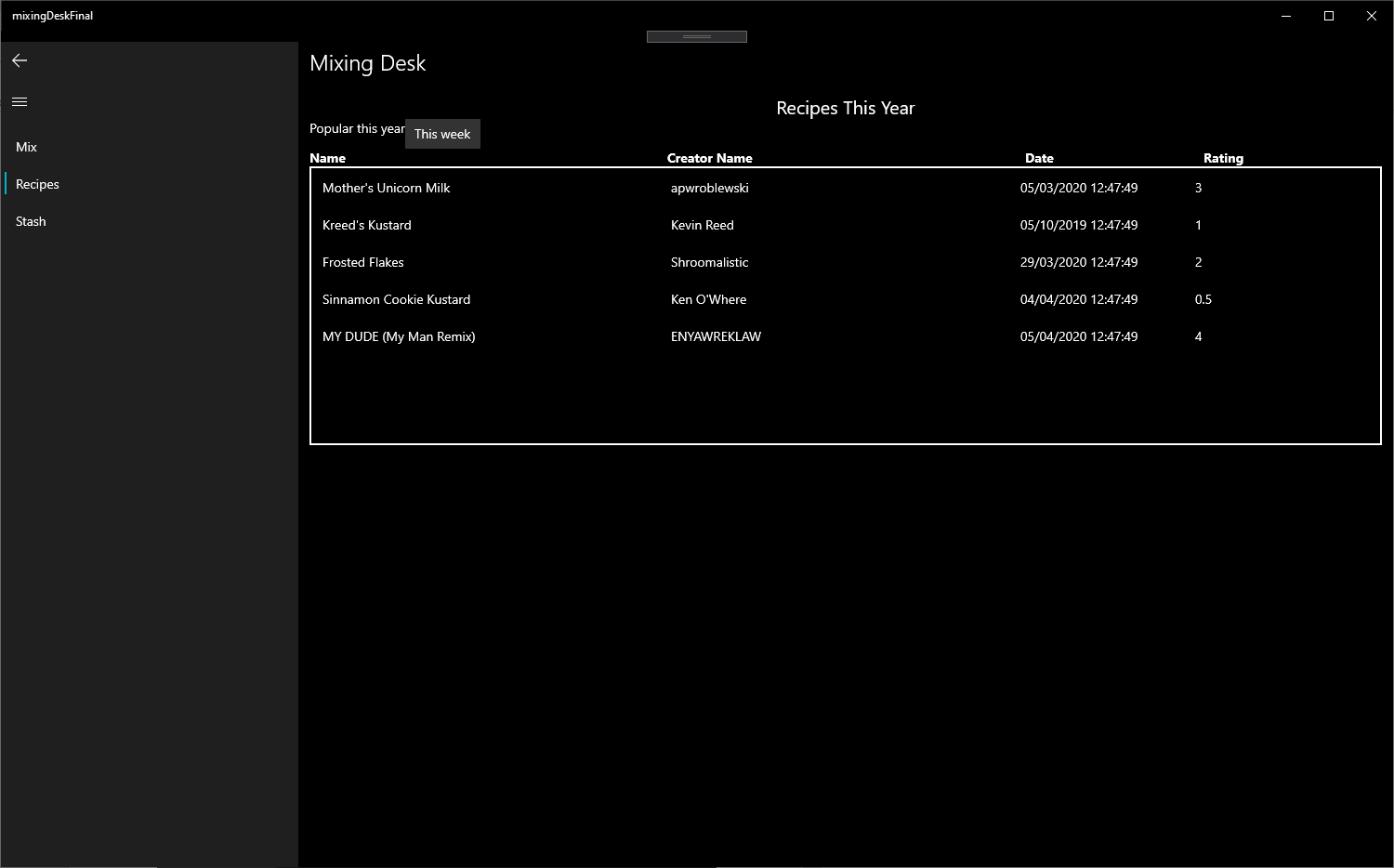


Figure : Recipes This Year page in final prototype.

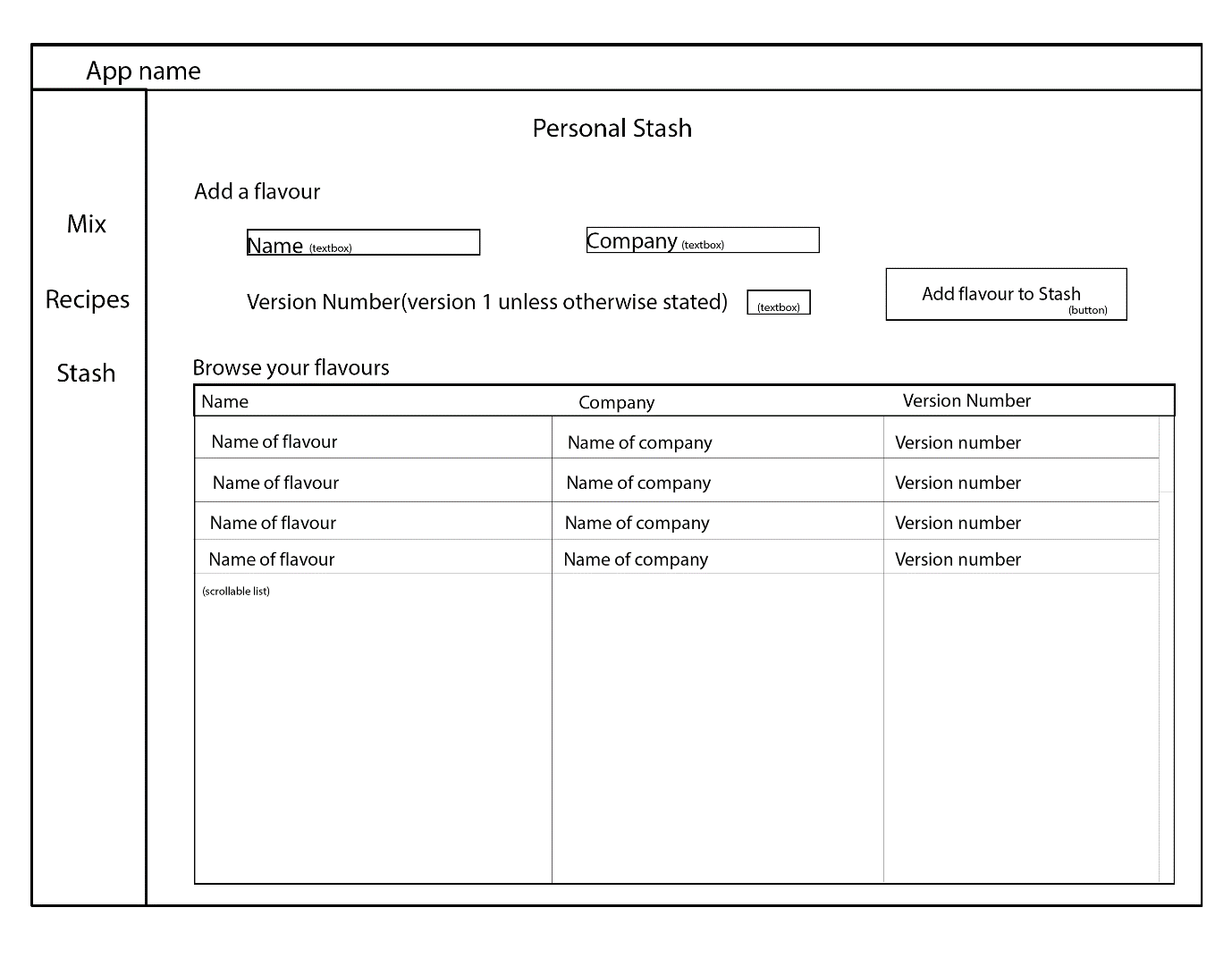


Figure : Personal Stash page wireframe design.

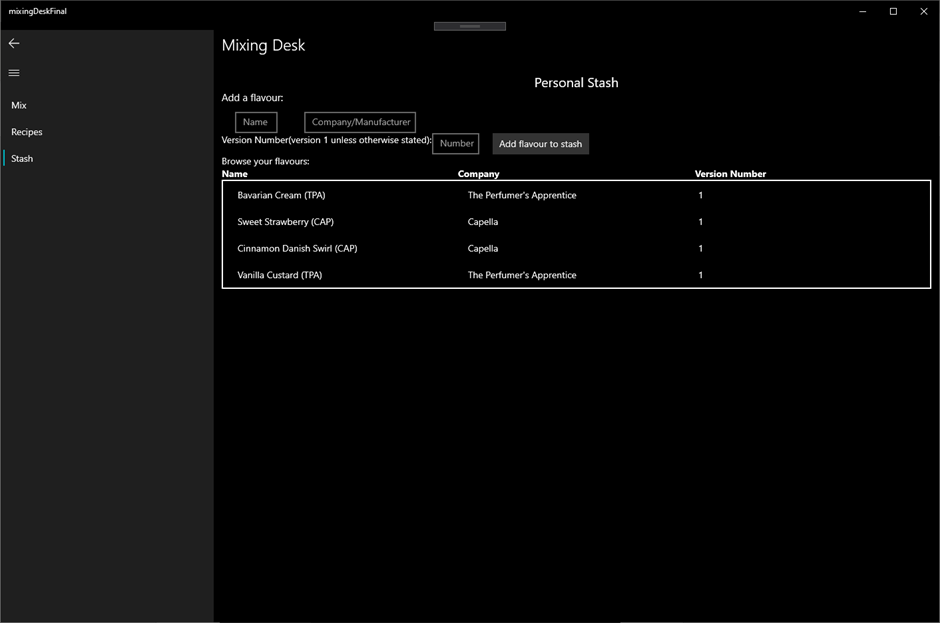


Figure : Personal Stash page in final prototype.

The Personal Stash page is also very similar to the original design. Again, pseudo column headers were added. These do not sort though.

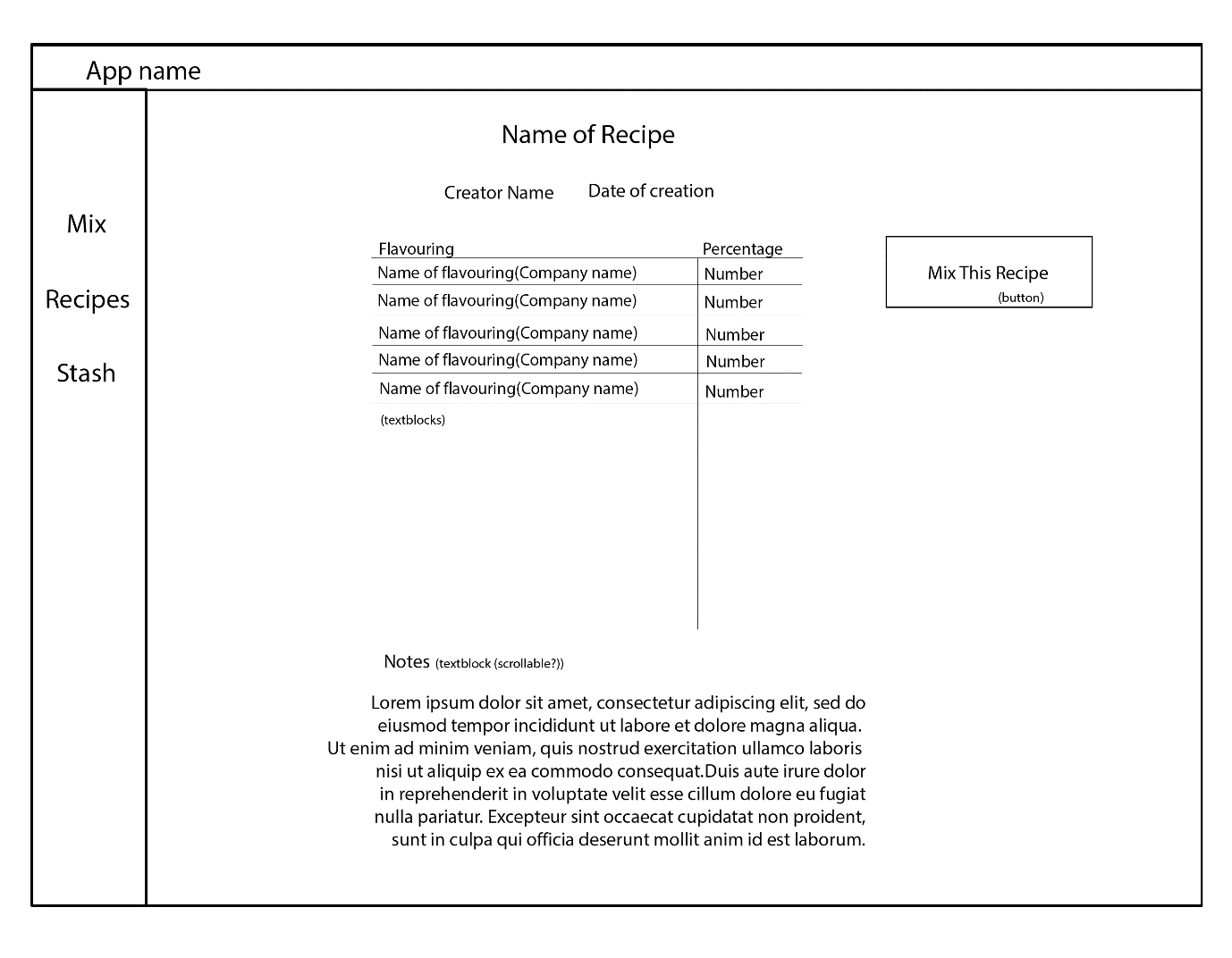


Figure : Specific Recipe page wireframe design.

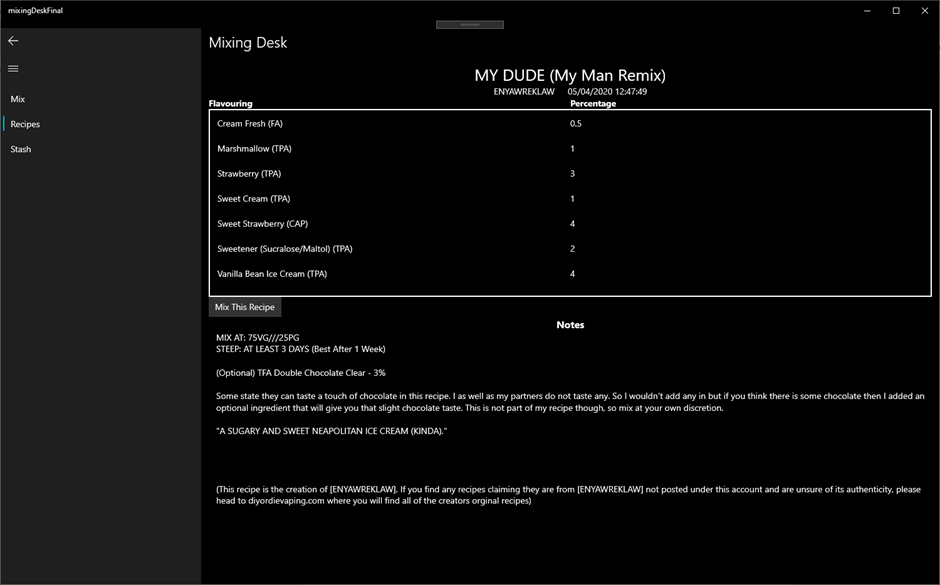


Figure 19: Specific Recipe page in final prototype.

In conclusion the UI largely matches the design.

## User Interface Controls, Validation and Error Prevention

In both the Mixing Calculator and Personal Stash page user input is validated in many UI controls. These are all textboxes and as can be seen in figures 20 and 21, this validation utilises the TextBoxRegex extension from Microsoft.Toolkit.Uwp.UI.Extensions.

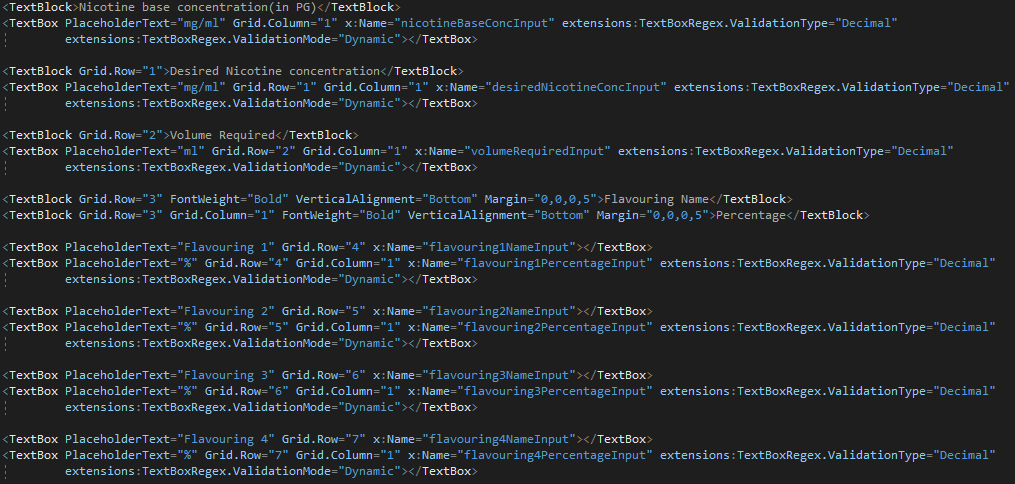


Figure 20: Mixing Calculator text box validation.

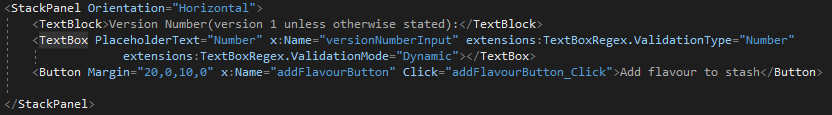


Figure 21: Personal Stash text box validation.

## Coding of Appropriate Events and Links to the Business Model

We will now discuss some event handlers some of which connect to the business model.

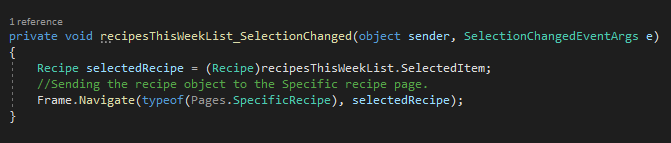


Figure 22: Recipes This Week selectionChanged event handler.

In figure 22 above, this event handler runs when a user clicks a list item within the list on the Recipes This Week page. This event handler will navigate the view to the Specific Recipe page taking with it a Recipe object.

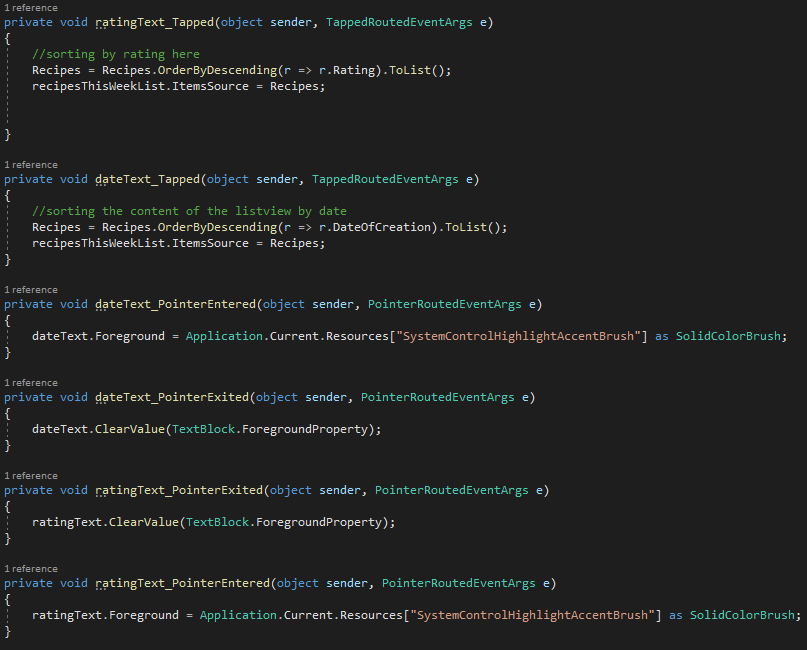


Figure 23: Mouse-over and Tapped event handlers in Recipes This Week page.

In figure 23 above, event handlers are shown from the Recipes This Week page. These include event handlers which re-order the list when a TextBlock is clicked and mouse-over events (PointerEntered and PointerExited) which change the text’s foreground property (its colour) to that of the system highlight colour and then back to what it was. Figure 24 shows this in action.

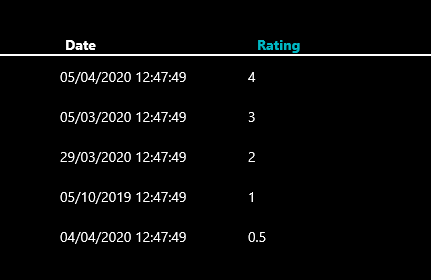


Figure 24: Mouse-over event changing the colour of the TextBlock.

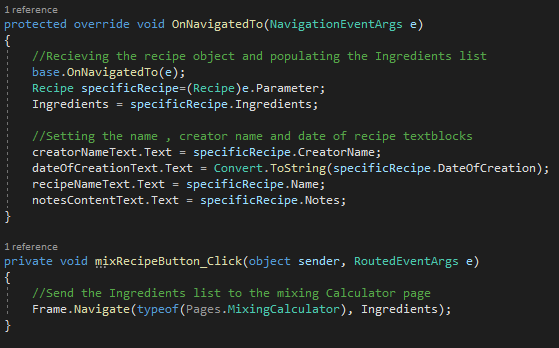


Figure 25: OnNavigatedTo and mixRecipeButton\_Click event handlers in the Specific Recipe page.

In Figure 25 we see the OnNavigatedTo event handler in the Specific Recipe page. This method runs whenever a specific recipe item is selected from a list on the Recipe pages and the frame is navigated to this page. This method populates the various UI elements of this page using the Recipe object’s properties which it was passed as it navigated to this page.



Figure 26: LoadRecipeClick event handler in Mixing Calculator page.

In figure 26 above, the LoadRecipeClick event handler is responsible for loading recipe information from a .txt file.

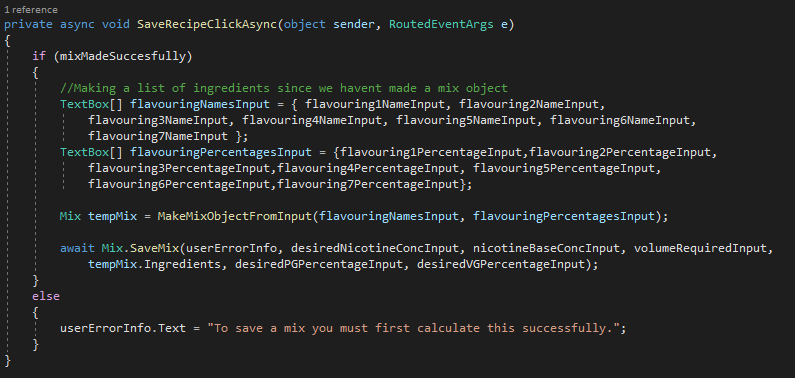


Figure 27: SaveRecipeClickAsync event handler in the Mixing Calculator page.

Figure 27 above shows the SaveRecipeClickAsync event handler responsible for saving a user created recipe to a .txt file.

# Usage of Unfamiliar Libraries and Constructs

This section details the usage of some of the unfamiliar libraries that were employed in the development of the Mixing Desk app. A full Research Log is included in the file researchLog.txt.

## Microsoft.Data.Sqlite library and SQLitePCLRaw.bundle\_winsqlite3 package

The Microsoft.Data.Sqlite library and SQLitePCLRaw.bundle\_winsqlite3 package were used in this prototype in order to allow the Mixing Desk app to access the version of Sqlite that is installed with Windows (Norm Estabrook, 2018). This is shown in figure 28 below in the Data Access Class.

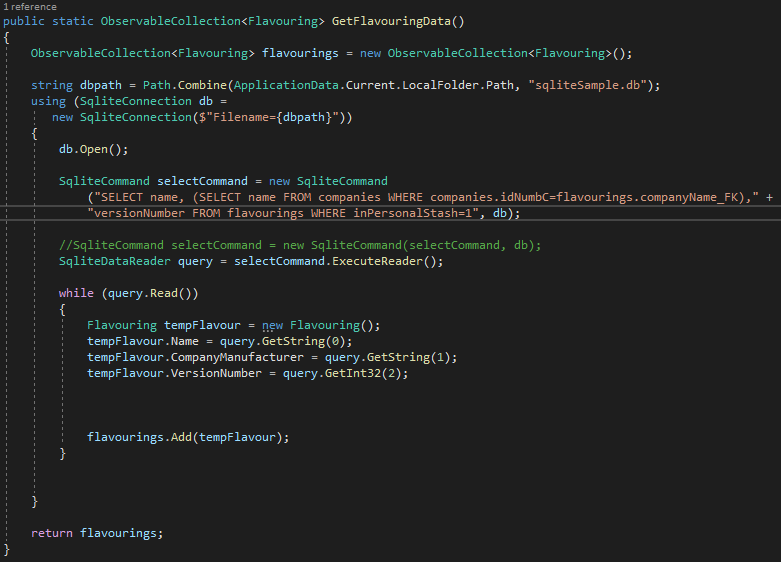


Figure 8: GetFlavouringData method in DataAccess class.

## Context Menu Construct

Context menus were implemented on the Mixing Calculator page and the Personal Stash page (Jim Walker, 2019). See figure 29 below.

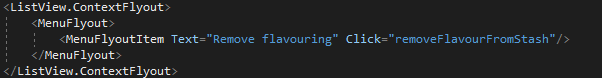


Figure 29: Context MenuFlyout construct in Personal Stash xaml page.

## Saving, Loading, Writing and Reading to Files

In order to save custom user created mixes or recipes, this developer had to research how files are saved (Michael Satran M. J., Save a file with a picker, 2018), loaded (Michael Satran M. J., Open files and folders with a picker, 2018) and how to read and write to them (Michael Satran L. H., 2018). See figure 27.

## Ordering List Item Objects by Property

To allow the user to order the Recipe pages lists by rating or date this developer had to research how to order objects by their properties (Schmelter, 2014). See figure 23.

## Returning Database Entries from a Given Range of Dates

In order to show recipes created from within one week or one month or one year from the apps opening, this developer had to research how to return database entries from a given range of dates (Thunder, 2010). Figure 30 below.

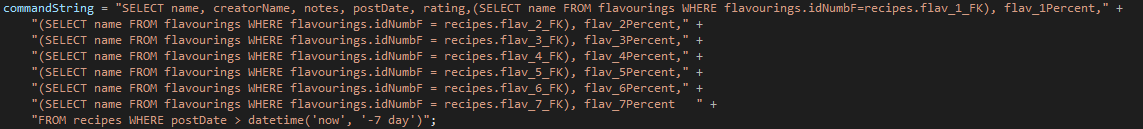


Figure 30: Sqlite query for a given range of dates in GetRecipeData method in DataAccess class.

# Error Handling

## User Interface Error Handling and Validation

Shown again for convenience is the validation on the text boxes in the Mixing Calculator page.

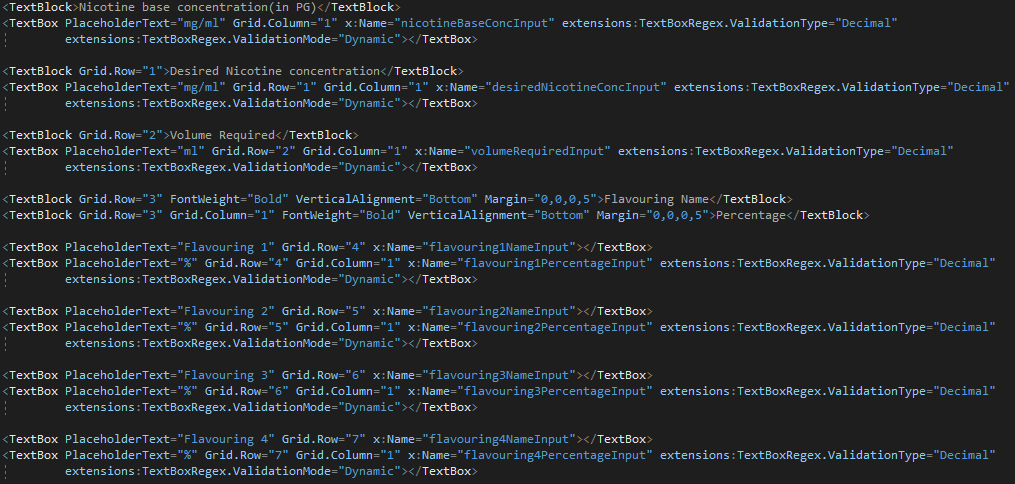


Figure 31: Mixing Calculator text box validation.

There is also further validation (checking for missing input) done for these text boxes when the user clicks the “Calculate” button shown below.

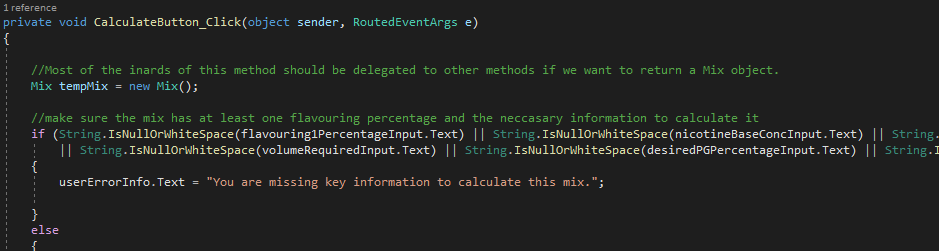


Figure 32: CalculateButton\_Click event handler checking for incorrect or missing input.

On the Personal Stash page, the user can also input data. Show below is more validation on the input that needs to be validated.

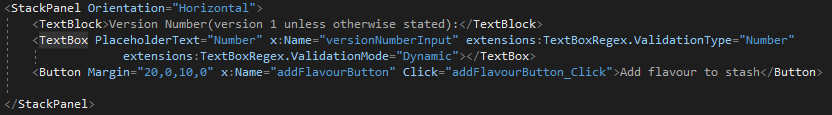


Figure 33: Personal Stash text box validation.

This is again checked for missing input shown below.

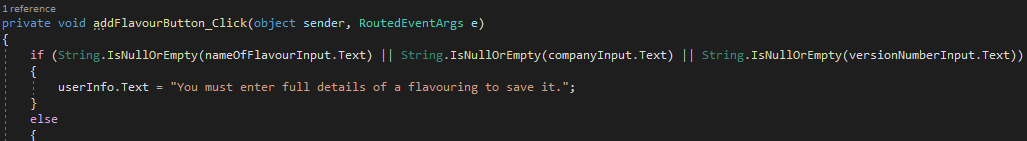


Figure 34: addFlavourButton\_Click event handler checking user input.

## Business Model Error Handling

In figure 33 below we see the CalculateMixInVolume method in the Mix class checks the user’s input to see if it is empty.

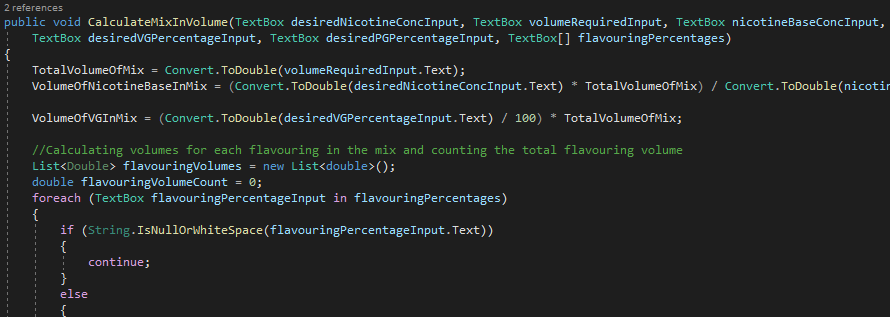


Figure : CalculateMixInVolume method in Mix class.

In figure 34 below we see the PGVGValidator method in the Mix class checks if the user’s input in two textboxes adds up to 100. If it doesn’t, it throws an exception.

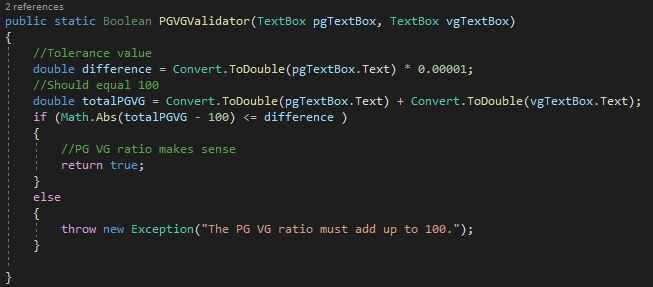


Figure : PGVGValidator method in the Mix class.

This exception is caught in the CalculateButton\_Click event handler shown below though the code is to long to show here properly. See the business code listing.

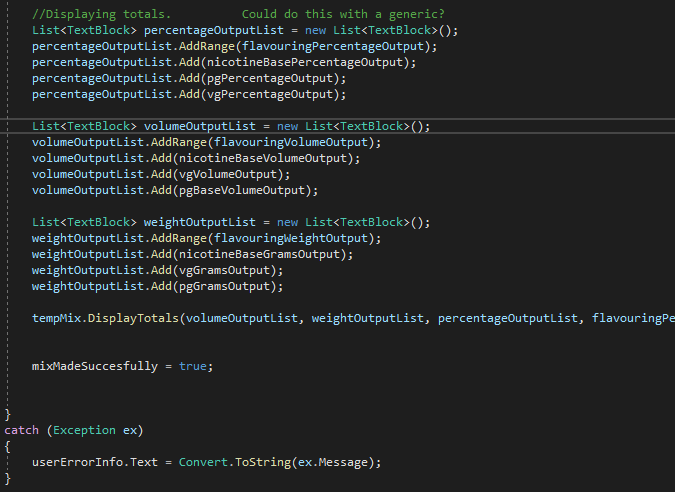


Figure 37: CalculateButton\_Click event handler catch block.

This catch block isn’t ideal (being so general) but this developer didn’t have the time to create any custom exceptions.

# Internal Documentation

See the code listing files included with this report.

# Testing

The overall development of this application has roughly followed the Rapid Application Development (RAD) methodology. (That is, client requirements were researched and established, the business and view model were designed, testing will take place in the manner described below, and then the finished prototype application will be presented to the client). We will now describe the testing strategy that will be used in the development stage.

## Testing Strategy

The testing of the Mixing Desk application in its current iteration will be performed with both white and black box testing. The white box testing will be performed with unit tests on all methods (except setters and getters) in the business model classes. These test runs will be documented and evaluated in this report. The black box testing (functionality testing) will take the form of test cases with scenarios performed on the software and the resulting outputs and software reaction documented.

## White Box Testing

Below in figures 38-44 you will see the output from unit-test test runs on all of the business model classes except for the DataAccess class. They are cropped for readability. Most require no comment.

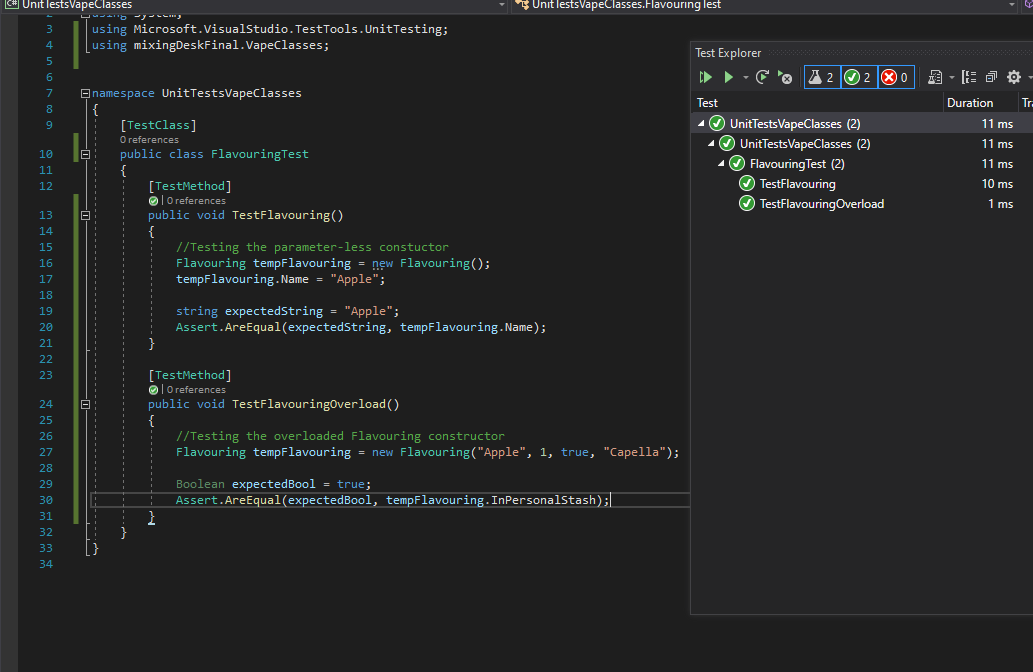


Figure 38: FlavouringTest Class test run.

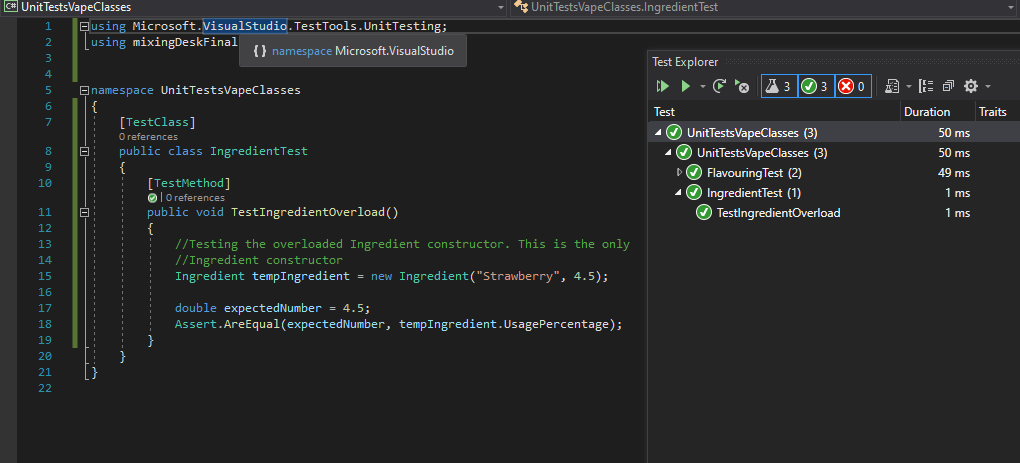


Figure 39: IngredientTest Class test run

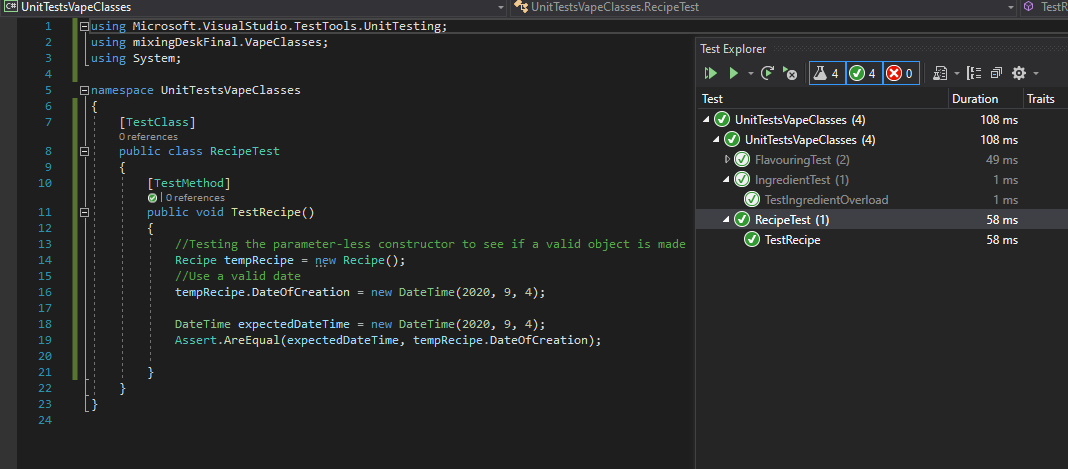


Figure : RecipeTest Class test run.

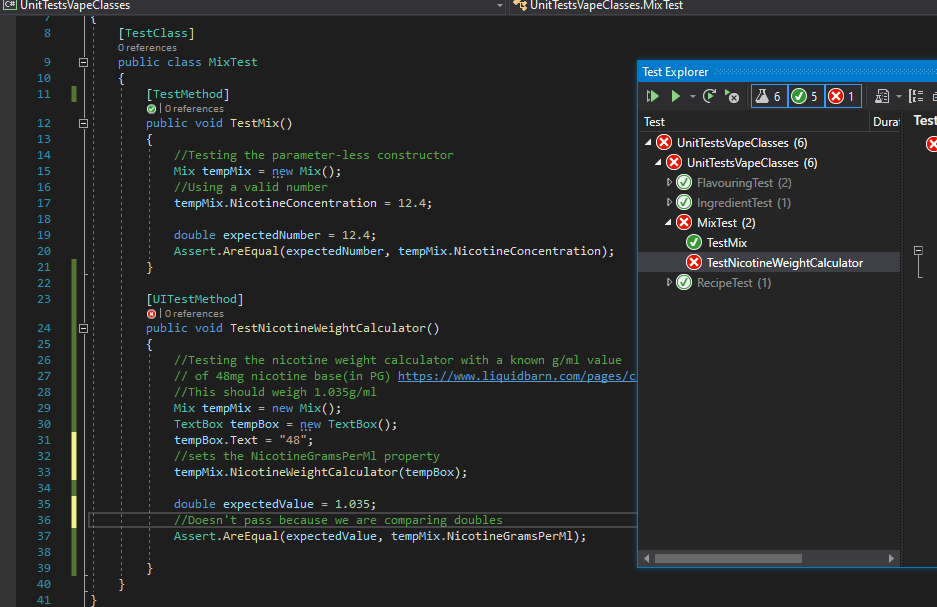


Figure 41: MixTest Class test run. TestNicotineWeightCalculator failing.

In figure 41 above you can see in the MixTest class that the test method TestNicotineWeightCalculator method fails. It fails because expected value and the produced value from NicotineWeightCalculator are very similar but not identical. Figure 42 below shows the detailed error output and how close the values are.

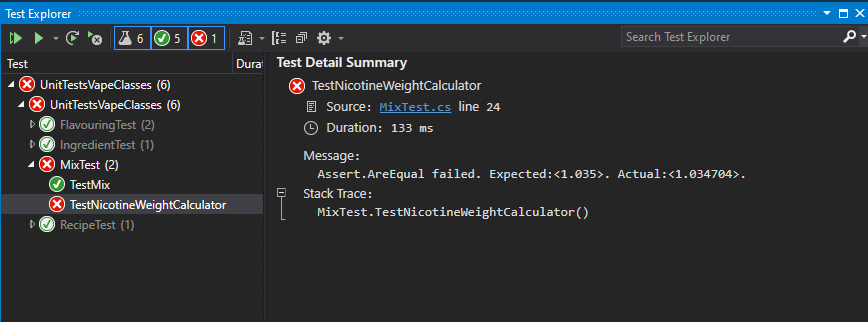


Figure 42: TestNicotineWeightCalculator failure details.

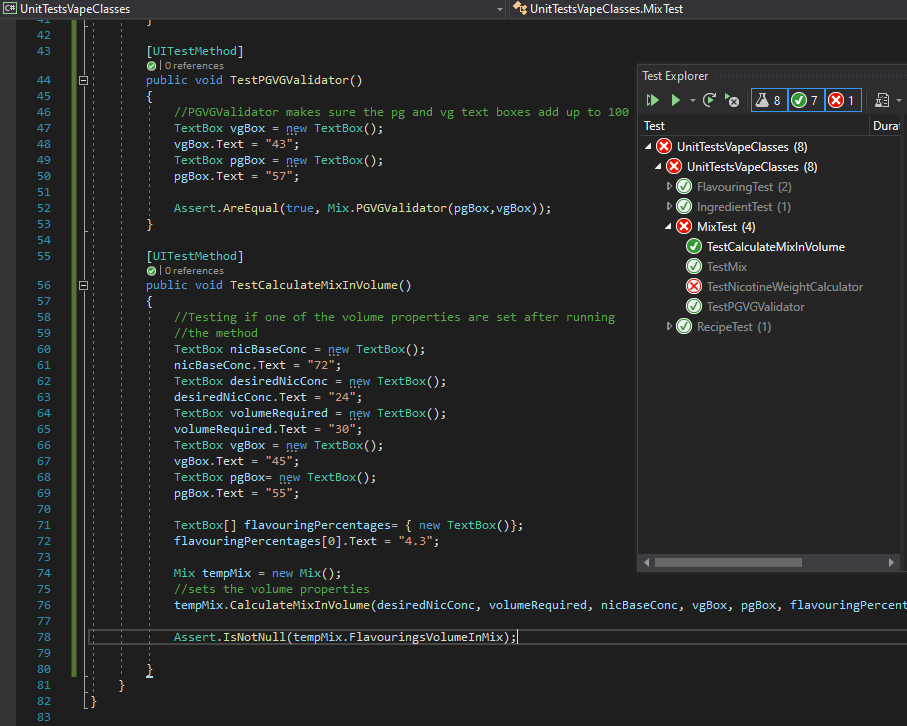


Figure 43: TestPGVGValidator and TestCalculateMixInVolume in MixTest passing.

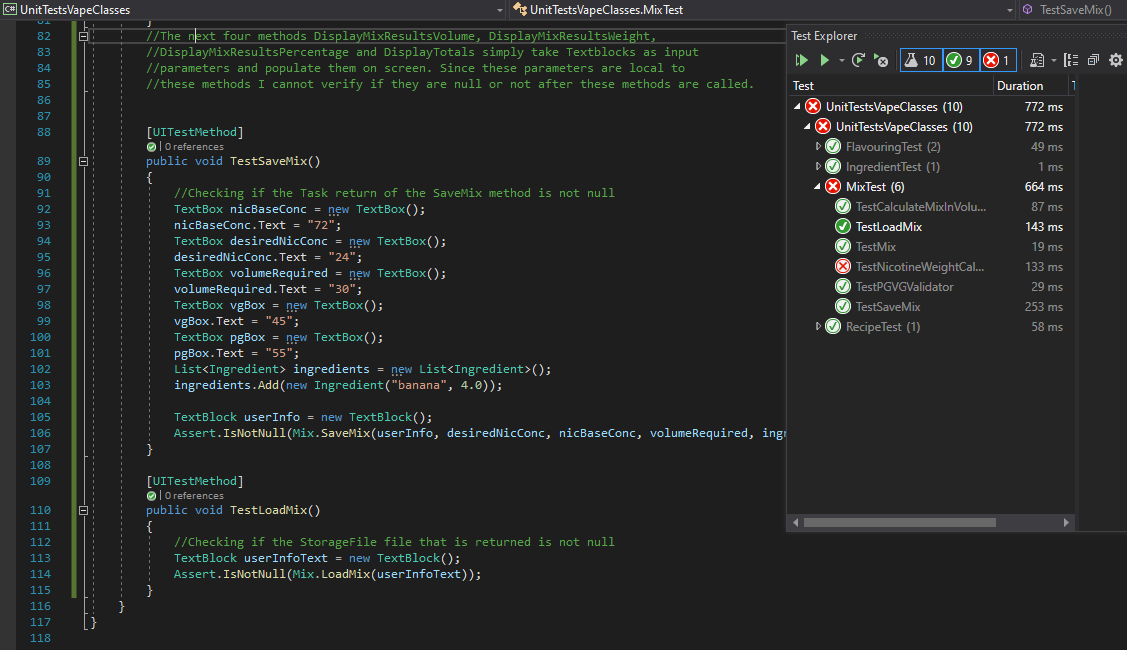


Figure 44: TestSaveMix and TestLoadMix in MixTest both passing.

It is worth noting here that there are four other methods in the Mix class. DisplayMixResultsVolume, DisplayMixResultsWeight, DisplayMixResultsPercentage and DisplayTotals. These methods have a return type of void, do not alter any of the Mix classes properties and simply alter text blocks in the Mixing Calculator page. They therefore cannot be tested.

## Black Box Testing

To test the software in a black-box manner test cases were developed (in the tables below) and the software reaction was documented.

### Scenario 1

|  |  |  |
| --- | --- | --- |
|  | **Action (Stimulus)** | **Software Reaction** |
| 1. | User wants to add one flavouring to their personal stash and remove two other flavourings. | System presents the user with relevant labelled input text boxes in the “Stash” section:   * Name * Company/Manufacturer * Version Number |
| 2. | User enters data into the text boxes. | The system verifies that the user has input data into all 3 text boxes (else it will display the message “You must enter full details of a flavouring to save it.”). The system then refreshes the list to show the newly added flavour. |
| 3. | User wants to delete two flavourings from the list. |  |
| 4. | User right clicks on the flavouring they wish to delete and then selects the option “Remove flavouring”. They do this for each flavouring they wish to delete. | The system updates the list, removing the selected flavouring. |

|  |  |  |  |
| --- | --- | --- | --- |
|  | **Test Scenario** | **Values** | **Software Reaction** |
| 1. | Name, Company/Manufacturer and Number text boxes are blank. | Name=blank  Company/Manufacturer=blank  Number=blank | PASSED-System is waiting for user input |
| 2. | User enters a value for the Name field and leaves the rest blank and clicks the “Calculate” button. | Name=”Banana Cream”  Company/Manufacturer=blank  Number=blank | PASSED-System displays the message “You must enter full details of a flavouring to save it.” |
| 3. | User enters values for all three fields. | Name= ”Banana Cream”  Company/Manufacturer=TPA  Number=V2 | FAILED-System throws an exception expecting only numbers. (will be corrected) |
| 4. | User enters valid values for all three fields. | Name= ”Banana Cream”  Company/Manufacturer=”TPA”  Number=2 | PASSED-System accepts the input and updates the list to display it. |
| 5. | User selects a flavouring in the list to delete by right clicking it. | N/A | PASSED-System displays flyout menu with “Remove flavouring” option. |
| 6. | User selects the “Remove flavouring” option. | N/A | PASSED-System removes the selected item from the list. |

### Scenario 2

|  |  |  |
| --- | --- | --- |
|  | **Action (Stimulus)** | **Software Reaction** |
| 1. | User wants to browse recipes and examine the details of on the recipes. | System presents the |
| 2. | User navigates to the Recipes section | System presents the user with the first of three Recipe pages showing recipes filtered from different time periods on each page with a button to navigate to the next recipe page. Recipes are shown in a list on each page. Also shown is a back button. |
| 3. | User browses the recipe pages and then selects a recipe to view | System navigates to a page showing the specific details of the selected recipe including the recipe title, creator name, list of ingredients and their usage percentages, notes on the recipe and a button labelled “Mix This Recipe”. |

We will not give a detailed breakdown of this scenario because there are no input values to analyse.

### Scenario 3

|  |  |  |
| --- | --- | --- |
|  | **Action (Stimulus)** | **Software Reaction** |
| 1. | User wants to mix a recipe they have found in the Recipes section |  |
| 2. | User finds the recipe and its details they would like to mix. User clicks the “Mix This Recipe” button | System navigates the user to the Mixing Calculator page inputting the flavouring names and their corresponding usage percentages in the appropriate textboxes. System inputs commonly used numbers into the fields for nicotine base strength, desired nicotine strength, volume required and the desired PG/VG ratio fields for convenience. |
| 3. | User wants to calculate the mix as it appears in the given fields presents. |  |
| 4. | User clicks the “Calculate” button. | System calculates the proportions of the mix showing the results in percentage, weight and volume measurements. The system also checks that the user’s mix makes mathematical sense ensuring that at least one flavouring at a non-zero percentage is used. It also checks that the PG/VG ratio adds up to 100%. |

We will not give a detailed breakdown of this scenario because there are no input values to analyse.

### Scenario 4

|  |  |  |
| --- | --- | --- |
|  | **Action (Stimulus)** | **Software Reaction** |
| 1. | User wants to make their own mix and save it for future reference. |  |
| 2. | User inputs data into the given text boxes. | System accepts appropriate user input validating it as it is entered only allowing numbers in fields where numbers are expected. |
| 3. | User clicks the “Calculate” button. | System calculates the proportions of the mix showing the results in percentage, weight and volume measurements. The system also checks that the user’s mix makes mathematical sense ensuring that at least one flavouring at a non-zero percentage is used. It also checks that the PG/VG ratio adds up to 100%. |

|  |  |  |  |
| --- | --- | --- | --- |
|  | **Test Scenario** | **Values** | **Software Reaction** |
| 1. | The fields Nicotine base concentration,  Desired Nicotine concentration, Volume Required and desired PG/VG ratio fields are auto filled with values. The fields Flavouring1-Flavouring7, the seven corresponding percentage fields and the Notes field are blank. | Nicotine base concentration=72,  Desired Nicotine concentration=18,  Volume Required=30,  Desired PG=50,  Desired VG =50 | PASSED- System is awaiting user input. |
| 2. | The user enters the following values and clicks the “Calculate” button. | Nicotine base concentration=72,  Desired Nicotine concentration=18,  Volume Required=30,  Desired PG=50,  Desired VG =30,  Flavouring1=”Vanilla Custard”,  Flavouring1Percentage= 5 | PASSED-System dispays the message “The PG VG ratio must add up to 100”. |
| 3. | The user alters the values to the following and clicks the “Calculate” button. | Nicotine base concentration=72,  Desired Nicotine concentration=18,  Volume Required=30,  Desired PG=50,  Desired VG =50,  Flavouring1=”Vanilla Custard”,  Flavouring1Percentage= blank | PASSED-System displays the message “You are missing key information to calculate this mix.” |
| 4. | The user alters the values to the following and clicks the “Calculate” button. | Nicotine base concentration=72,  Desired Nicotine concentration=18,  Volume Required=blank,  Desired PG=50,  Desired VG =50,  Flavouring1=”Vanilla Custard”,  Flavouring1Percentage= 5 | PASSED-System maintains the message “You are missing key information to calculate this mix.” |
| 5. | The user alters the values to the following and clicks the “Calculate” button. | Nicotine base concentration=72,  Desired Nicotine concentration=18,  Volume Required=30,  Desired PG=50,  Desired VG =50,  Flavouring1=”Vanilla Custard”,  Flavouring1Percentage= 5 | PASSED-System performs the calculation and displays the results(in ml):  Nicotine base 11.25  PG 9  VG 22.5  Vanilla Custard 2.25  Total 45  Total flavour percentage 5 |

# References

Jim Walker, M. J. (2019, April 19). *Menus and context menus*. Retrieved from Microsoft Docs: https://docs.microsoft.com/en-us/windows/uwp/design/controls-and-patterns/menus

Michael Satran, L. H. (2018, December 19). *Create, write, and read a file*. Retrieved from Microsoft Docs: https://docs.microsoft.com/en-us/windows/uwp/files/quickstart-reading-and-writing-files

Michael Satran, M. J. (2018, December 19). *Open files and folders with a picker*. Retrieved from Microsoft Docs: https://docs.microsoft.com/en-us/windows/uwp/files/quickstart-using-file-and-folder-pickers

Michael Satran, M. J. (2018, December 19). *Save a file with a picker*. Retrieved from Microsoft Docs: https://docs.microsoft.com/en-us/windows/uwp/files/quickstart-save-a-file-with-a-picker

Norm Estabrook, M. S. (2018, November 30). *Use a SQLite database in a UWP app*. Retrieved from Microsoft Docs: https://docs.microsoft.com/en-us/windows/uwp/data-access/sqlite-databases

Schmelter, T. (2014, July 31). *Order a list of objects by it's DateTime property*. Retrieved from Stack Overflow: https://stackoverflow.com/questions/25053887/order-a-list-of-objects-by-its-datetime-property

Thunder. (2010, February 22). *sqlite select with condition on date*. Retrieved from Stack Overflow: https://stackoverflow.com/questions/2309227/sqlite-select-with-condition-on-date