ST10027949

Ted Ngobeni

PROG6211

Portfolio Of Evidence

Contents

Premiss: The Recipe Application

The recipe application is an application that allows a user to create recipes. These recipes are made up of ingredients as well as steps as to how the recipe is prepared. All this information is input by the user, from how many ingredients there are to the name of the recipe. The creation process is entirely up to the user.

Main Window .xaml:

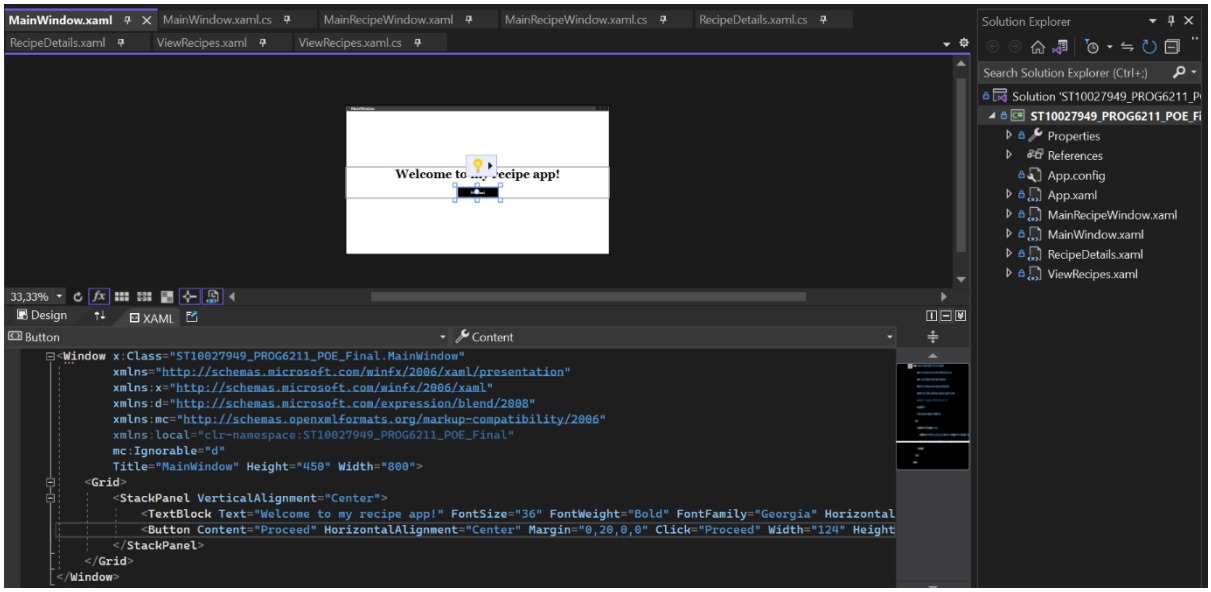

Figure 1: The amin window the user sees when entering the application.

When you first launch the application, you will be greeted by the MainWindow. This window has a simple and welcoming design, featuring a large TextBlock that displays the message “Welcome to my recipe app!” in a bold, Georgia font. The text is horizontally centered on the screen to catch your attention.

Below the welcome message, you will see a Button labeled “Proceed”. This button is also horizontally centered and has a black background with white text. When you click on this button, the MainWindow will trigger an event called “Proceed” which is defined in the code-behind file.

When the “Proceed” event is triggered, the application will create and display a new window called MainRecipeWindow. This window is where you can interact with the main features of the recipe app.

Code-behind file:

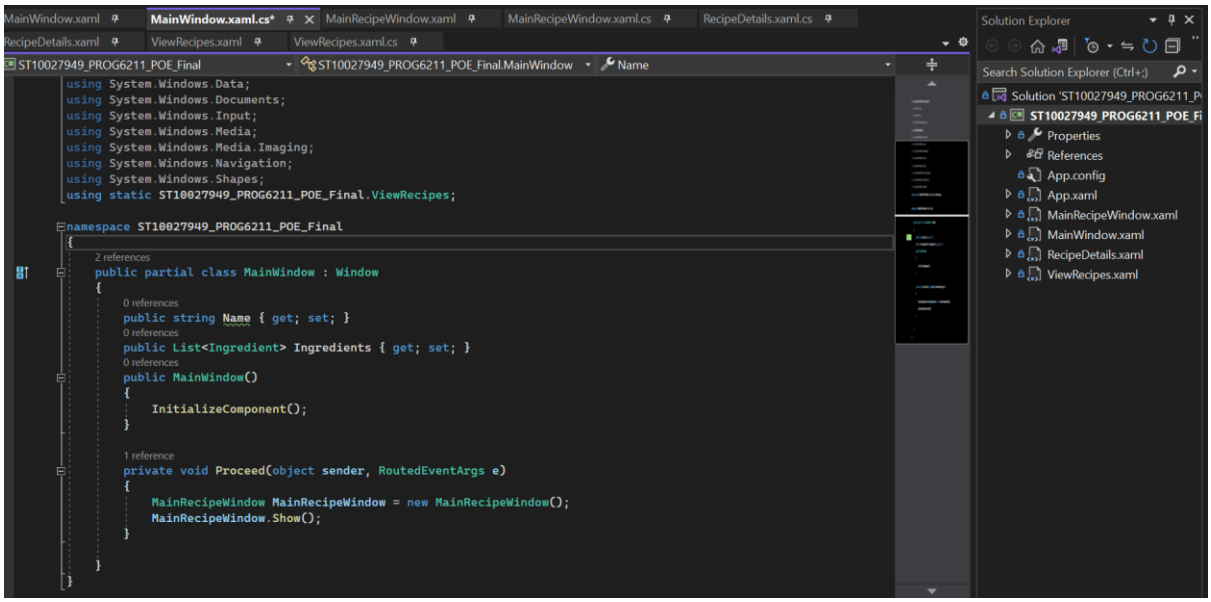

Figure 2:Code behind file of the main window

When you first launch the application, you will be greeted by the MainWindow. This window has a simple and welcoming design, featuring a large TextBlock that displays the message “Welcome to my recipe app!” in a bold, Georgia font. The text is horizontally centered on the screen to catch your attention.

Below the welcome message, you will see a Button labeled “Proceed”. This button is also horizontally centered and has a black background with white text. When you click on this button, the MainWindow will trigger an event called “Proceed” which is defined in the code-behind file.

When the “Proceed” event is triggered, the application will create and display a new window called MainRecipeWindow. This window is where you can interact with the main features of the recipe app.

Main Recipe Window:

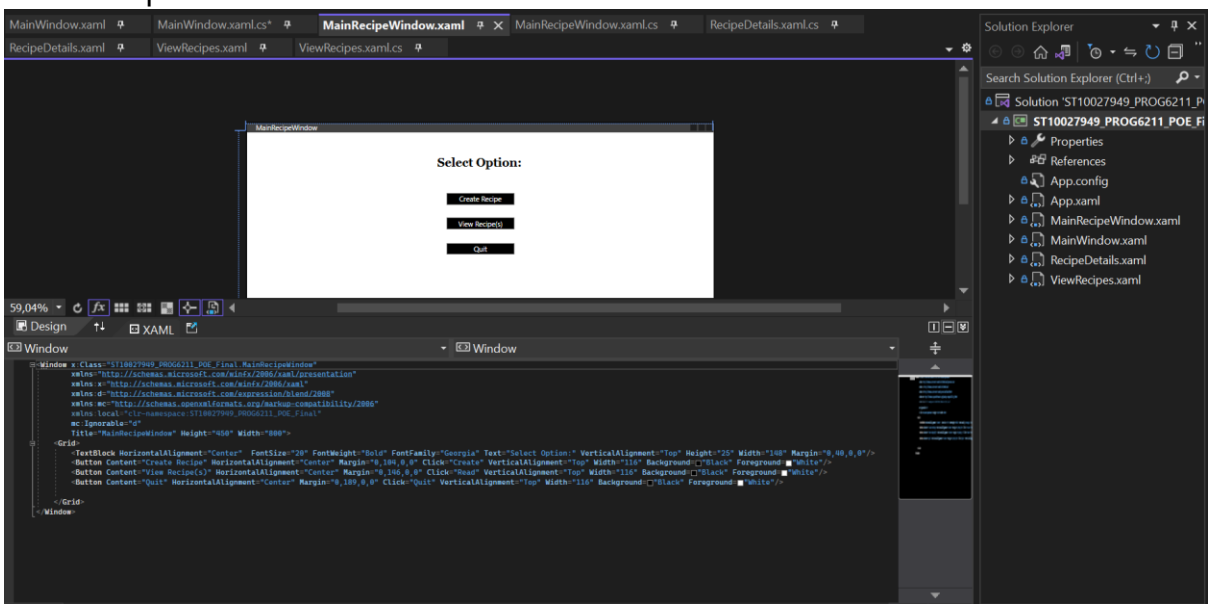

Figure 3: Main Recipe Window code and window

When you navigate to the MainRecipeWindow, you will see a window with a Grid layout. At the top of the window, there is a TextBlock that displays the text “Select Option:” in a bold, Georgia font. This text is horizontally centered on the screen and serves as a heading for the options below it.

Below the heading, there are three Button elements that are also horizontally centered on the screen. The first button is labeled “Create Recipe” and, when clicked, will trigger an event called “Create” which should be defined in the code-behind file. The second button is labeled “View Recipe(s)” and, when clicked, will trigger an event called “Read” which should also be defined in the code-behind file. The third button is labeled “Quit” and, when clicked, will trigger an event called “Quit” which should be defined in the code-behind file.

All three buttons have a black background with white text and are of equal width. They are vertically aligned from top to bottom with some margin between them.

Code-behind File:

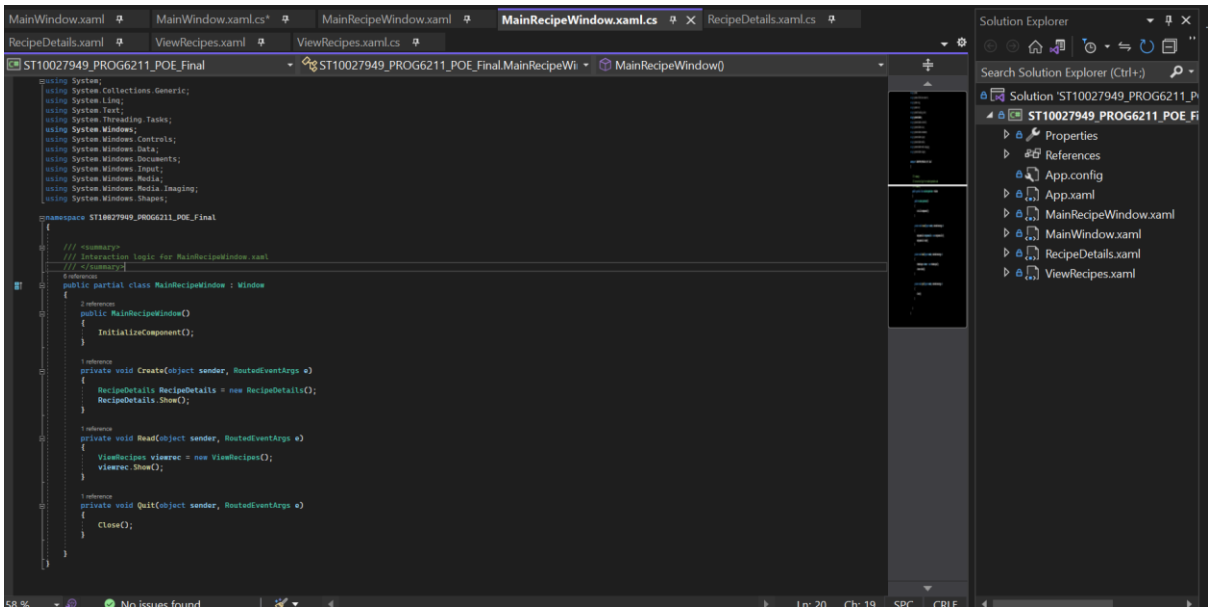

Figure 4: Code-behind file

The MainRecipeWindow class has a constructor that is called when the window is first created. Inside the constructor, the InitializeComponent() method is called to initialize all of the UI components that were defined in the XAML file.

The MainRecipeWindow class also has three event handlers for the “Create”, “Read”, and “Quit” events that are triggered when the corresponding buttons are clicked.

When the “Create” button is clicked, the Create method is called. This method creates a new instance of the RecipeDetails class and calls its Show() method to display the window. This suggests that when you click on the “Create Recipe” button, you will be taken to a new window where you can create a new recipe.

When the “Read” button is clicked, the Read method is called. This method creates a new instance of the ViewRecipes class and calls its Show() method to display the window. This suggests that when you click on the “View Recipe(s)” button, you will be taken to a new window where you can view existing recipes.

When the “Quit” button is clicked, the Quit method is called. This method simply calls the Close() method to close the MainRecipeWindow. This suggests that when you click on the “Quit” button, you will exit the MainRecipeWindow.

Recipe Details Window:

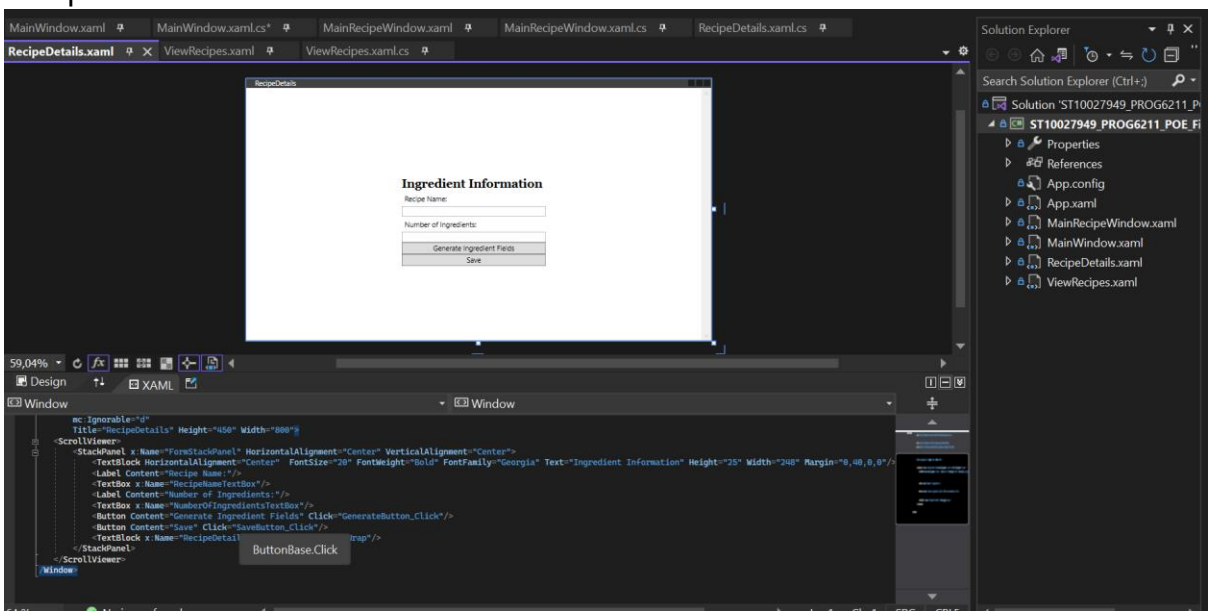

Figure 5: Recipe Details Window

When you navigate to the RecipeDetails window, you will see a window with a ScrollViewer that contains a StackPanel. The StackPanel is horizontally and vertically aligned to the center of the window and contains several elements.

At the top of the StackPanel, there is a TextBlock that displays the text “Ingredient Information” in a bold, Georgia font. This text is horizontally centered on the screen and serves as a heading for the form below it.

Below the heading, there are two pairs of Label and TextBox elements. The first pair consists of a Label with the content “Recipe Name:” and a TextBox with the name “RecipeNameTextBox”. The second pair consists of a Label with the content “Number of Ingredients:” and a TextBox with the name “NumberOfIngredientsTextBox”.

Below these elements, there are two Button elements. The first button is labeled “Generate Ingredient Fields” and, when clicked, will trigger an event called “GenerateButton_Click” which should be defined in the code-behind file. The second button is labeled “Save” and, when clicked, will trigger an event called “SaveButton_Click” which should also be defined in the code-behind file.

At the bottom of the StackPanel, there is a TextBlock with the name “RecipeDetailsBlock” that has text wrapping enabled. This element can be used to display additional information or feedback to the user.

Code-behind File:

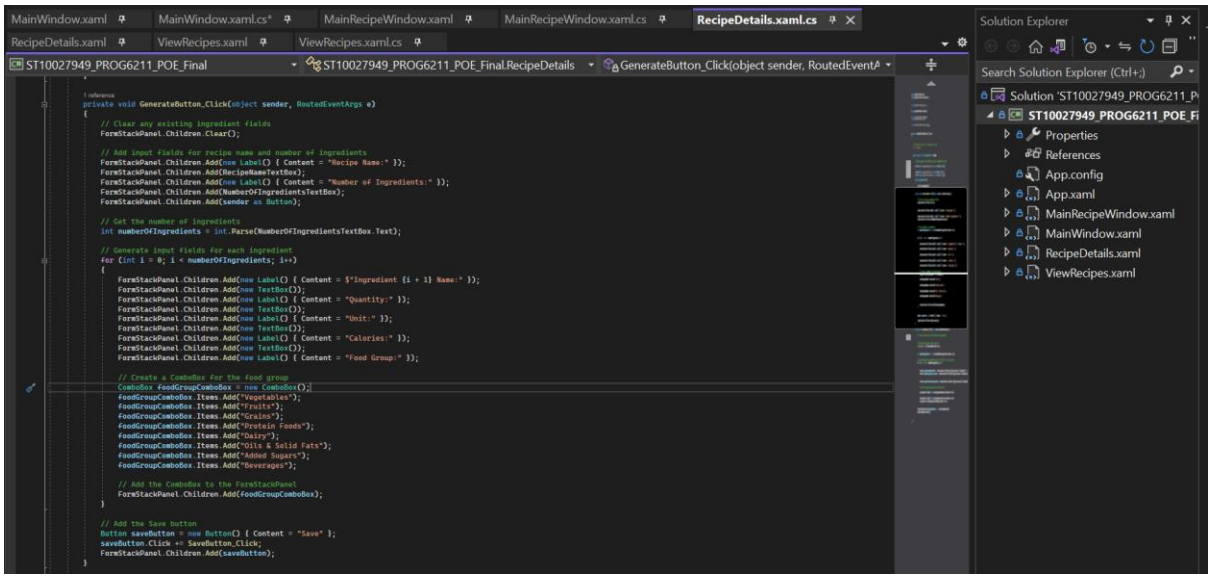

Figure 6:Code behind file 1

The RecipeDetails class has several private fields that are used to store information about the ingredients in a recipe. These fields are generic collections of type List<string> or List<double> and are used to store the names, quantities, measurement units, calories, and food groups of the ingredients.

The RecipeDetails class has a constructor that is called when the window is first created. Inside the constructor, the InitializeComponent() method is called to initialize all of the UI components that were defined in the XAML file.

The RecipeDetails class also has an event handler for the “GenerateButton_Click” event that is triggered when the “Generate Ingredient Fields” button is clicked.

When this button is clicked, the GenerateButton_Click method is called. This method first clears any existing elements from the FormStackPanel and then adds input fields for the recipe name and number of ingredients. It also adds the “Generate Ingredient Fields” button back to the FormStackPanel.

Next, the method retrieves the number of ingredients from the NumberOfIngredientsTextBox and uses this value to generate input fields for each ingredient. For each ingredient, a set of Label and TextBox elements are added to the FormStackPanel for the ingredient name, quantity, unit, calories, and food group. For the food group, a ComboBox is created with several predefined options and added to the FormStackPanel.

Finally, a “Save” button is created and added to the FormStackPanel. This button has an event handler attached to it that will call the SaveButton_Click method when clicked.

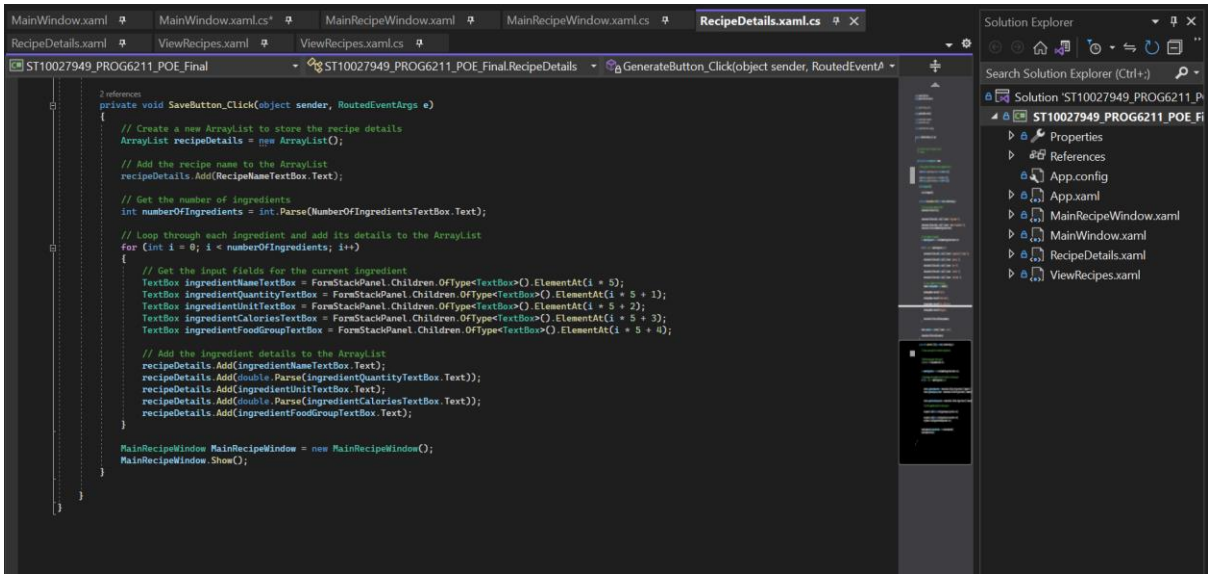

Figure 7: Code behind file 2

The `SaveButton_Click` method is an event handler for the "Save" button's click event in the `RecipeDetails` window of your WPF application.

When the "Save" button is clicked, this method is called. Inside the method, a new `ArrayList` is created to store the details of the recipe. The recipe name is retrieved from the `RecipeNameTextBox` and added to the `ArrayList`.

Next, the method retrieves the number of ingredients from the `NumberOfIngredientsTextBox` and uses this value to loop through each ingredient and add its details to the `ArrayList`. For each ingredient, the method retrieves the corresponding input fields from the `FormStackPanel` and adds their values to the `ArrayList`.

After all of the ingredient details have been added to the `ArrayList`, a new instance of the `MainRecipeWindow` class is created and its `Show()` method is called to display the window. This suggests that when you click on the "Save" button, you will be taken back to the `MainRecipeWindow`.

It's important to note that while this method collects and stores the recipe details in an `ArrayList`, it does not appear to do anything with this data. The `ArrayList` is not saved or passed to any other part of the application, so it is unclear how this data is intended to be used.

View Recipe Window:

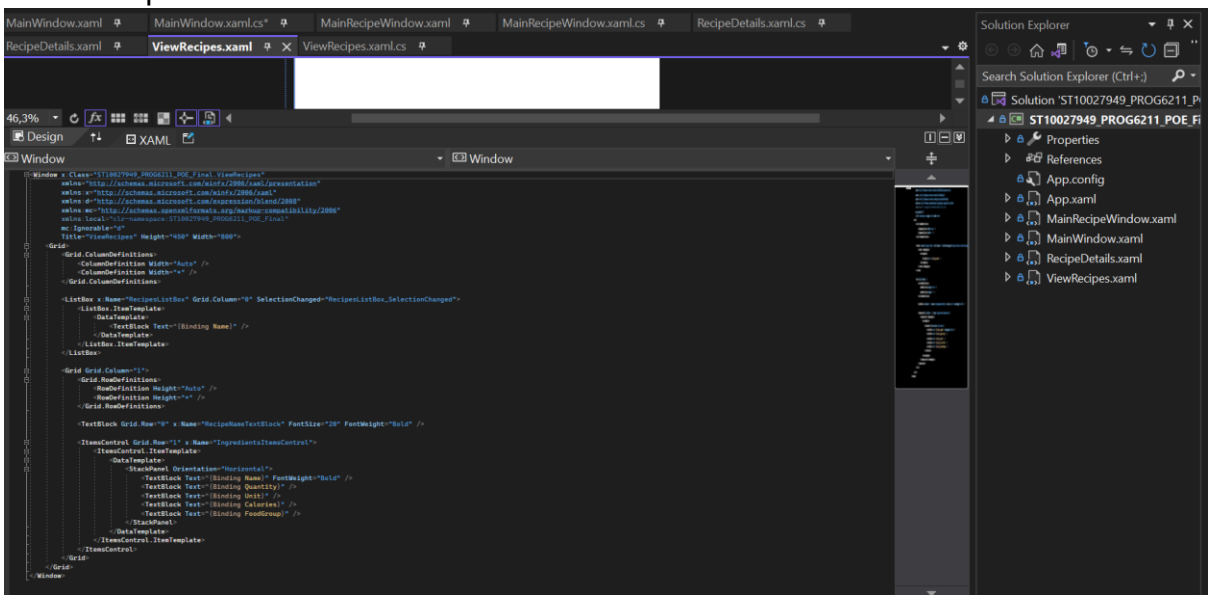

Figure 8: View Recipe Window

When you navigate to the `ViewRecipes` window, you will see a window with a `Grid` layout that has two columns. The first column is set to automatically size to its content, while the second column takes up the remaining space.

In the first column, there is a `ListBox` with the name "RecipesListBox". This `ListBox` has an `ItemTemplate` defined that specifies how each item in the `ListBox` should be displayed. In this case, each item is displayed as a `TextBlock` that binds its text to the `Name` property of the item.

In the second column, there is another `Grid` with two rows. The first row is set to automatically size to its content, while the second row takes up the remaining space.

In the first row of this nested `Grid`, there is a `TextBlock` with the name "RecipeNameTextBlock". This `TextBlock` has a large font size and bold font weight and can be used to display the name of the selected recipe.

In the second row of this nested `Grid`, there is an `ItemsControl` with the name "IngredientsItemsControl". This `ItemsControl` has an `ItemTemplate` defined that

specifies how each item in the control should be displayed. In this case, each item is displayed as a horizontal `StackPanel` that contains several `TextBlock` elements. These `TextBlock` elements bind their text to various properties of the item, such as its name, quantity, unit, calories, and food group.

The `ViewRecipes` window also has an event handler attached to the "SelectionChanged" event of the "RecipesListBox". This event handler should be defined in the code-behind file and will be called when the user selects an item from the `ListBox`.

Code-behind File:

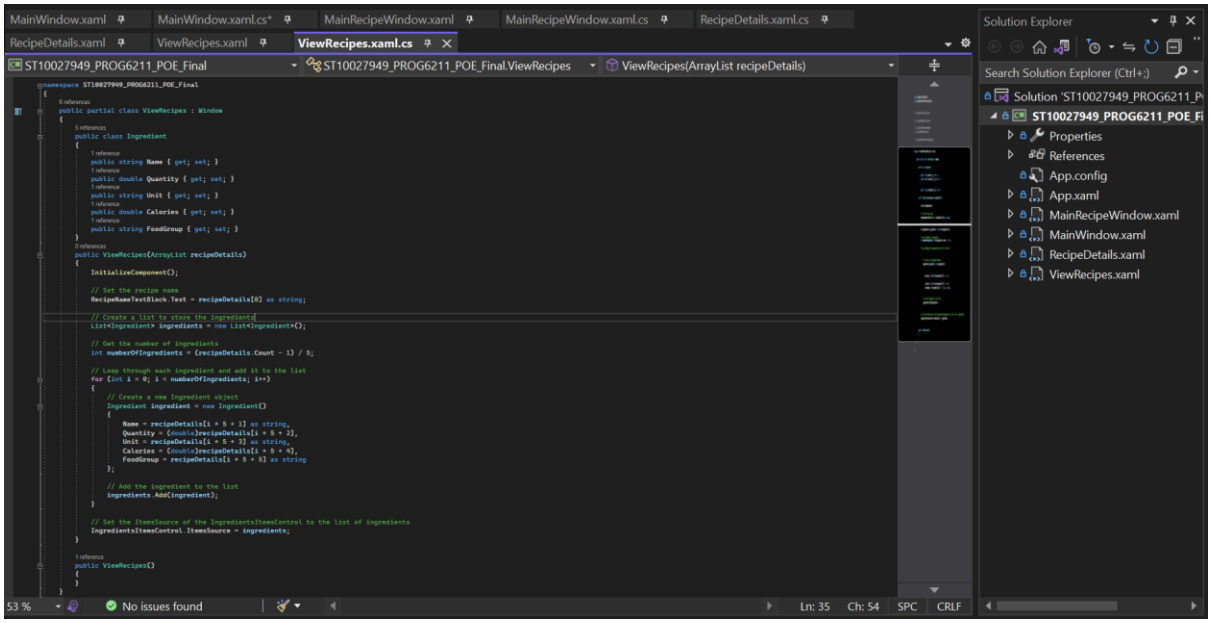

Figure 9: Code Behind File for view recipe

The `ViewRecipes` class has a nested `Ingredient` class that is used to represent an ingredient in a recipe. This class has several public properties for the name, quantity, unit, calories, and food group of the ingredient.

The `ViewRecipes` class has two constructors. The first constructor takes an `ArrayList` of recipe details as a parameter, while the second constructor is parameterless.

When the first constructor is called with an `ArrayList` of recipe details, it initializes the UI components and then sets the text of the "RecipeNameTextBlock" to the first element in the `ArrayList`, which should be the recipe name.

Next, the constructor creates a `List<Ingredient>` to store the ingredients in the recipe. It calculates the number of ingredients based on the size of the `ArrayList` and then loops through each ingredient to create a new `Ingredient` object and add it to the list.

After all of the ingredients have been added to the list, the constructor sets the `ItemsSource` property of the "IngredientsItemsControl" to this list. This will cause the `ItemsControl` to display each ingredient using the `DataTemplate` defined in the XAML file.

It's important to note that while this constructor takes an `ArrayList` of recipe details as a parameter, it is not clear how this data is intended to be passed to the constructor. The second, parameterless constructor does not appear to do anything with this data.