JUNE 27, 2024

PROG6221 POE

LWANDLE CHAUKE
ST10380788

THE INDEPENDENT INSTITUTE OF EDUCATION

MODULE NAME:	MODULE CODE:
PROGRAMMING 2A	PROG6221

ASSESSMENT TYPE: POE (PAPER)

TOTAL MARK ALLOCATION: 300 MARKS

TOTAL HOURS: A minimum of 35 HOURS is suggested to complete this assessment.

By submitting this assignment, you acknowledge that you have read and understood all the rules as per the terms in the registration contract, in particular the assignment and assessment rules in The IIE Assessment Strategy and Policy (IIE009), the intellectual integrity and plagiarism rules in the Intellectual Integrity and Property Rights Policy (IIE023), as well as any rules and regulations published in the student portal.

INSTRUCTIONS:

1. ***No material may be copied from original sources, even if referenced correctly, unless it is a direct quote indicated with quotation marks. No more than 10% of the assignment may consist of direct quotes.***
2. ***Make a copy of your assignment before handing it in.***
3. *Assignments must be typed unless otherwise specified.*
4. *Begin each section on a new page.*
5. *Follow all instructions on the PoE cover sheet.*
6. *This is an individual assignment.*

ACADEMIC HONESTY DECLARATION

Please complete the Academic Honesty Declaration below.
Please note that your assessment will not be marked, and you will receive 0% if you have not completed ALL aspects of this declaration.

Declaration

	SIGN
I have read the assessment rules provided in this declaration.	[illegible]
This assessment is my own work.	[illegible]
I have not copied any other student's work in this assessment.	[illegible]
I have not uploaded the assessment question to any website or App offering assessment assistance.	[illegible]
I have not downloaded my assessment response from a website.	[illegible]
I have not used any AI tool without reviewing, re-writing, and re-working this information, and referencing any AI tools in my work.	[illegible]
I have not shared this assessment with any other student.	[illegible]
I have not presented the work of published sources as my own work.	[illegible]
I have correctly cited all my sources of information.	[illegible]
My referencing is technically correct, consistent, and congruent.	[illegible]
I have acted in an academically honest way in this assessment.	[illegible]

PART 1:

CODE:

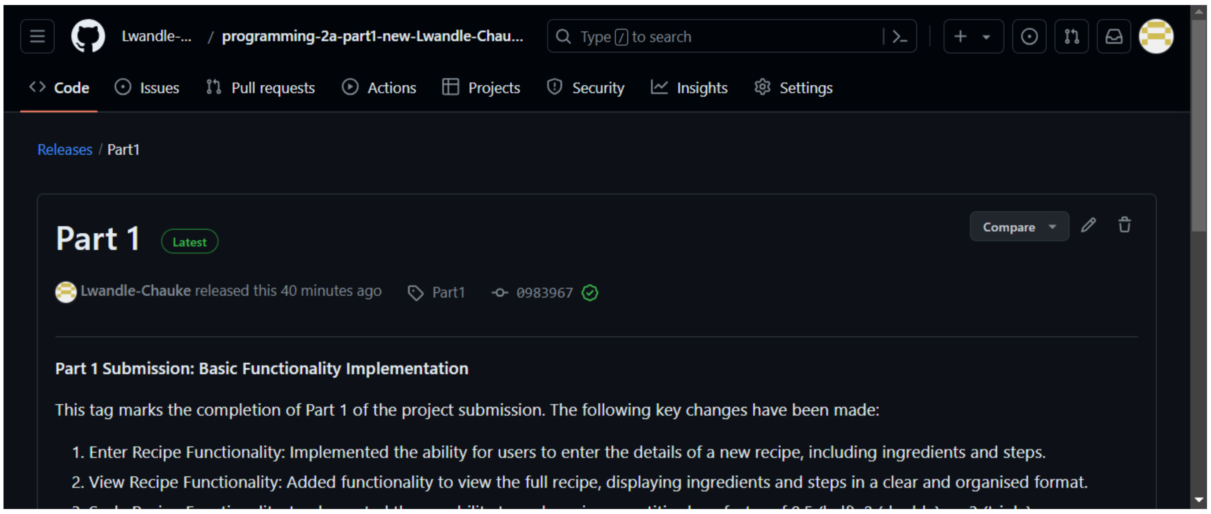
```
using System;

class MyRecipeApp
{
    static string[] ingredients;
    static double[] quantities;
    static string[] units;
    static string[] steps;

    static void Main(string[] args)
    {
        Console.WriteLine("Welcome to My Recipe App!");

        int previousChoice = -1;
        // Display the menu options after each user interaction
        DisplayOptions();

        // Main loop for interacting with the user
        while (true)
        {
            Console.Write("\nEnter your choice: ");
            int choice;
            try
            {
                choice = Convert.ToInt32(Console.ReadLine());
            }
            catch (FormatException)
            {
                Console.WriteLine("Invalid choice. Please enter a number.");
                continue;
            }

            // Switch statement to handle user choices
            switch (choice)
            {
                case 1:
                    EnterRecipe();
                    break;
                case 2:
                    if (ingredients == null)
                    {
                        Console.WriteLine("No recipe has been created.");
                        EnterRecipe();
                    }
                    else
                    {
                        ViewRecipe();
                    }
                    break;
                case 3:
                    ScaleRecipe();
                    break;
                case 4:
                    ResetQuantities();
                    break;
                case 5:
```

```
                ClearData();
                break;
            case 6:
                Console.WriteLine("Thank you for using Recipe App!");
                return;
            default:
                Console.WriteLine("Invalid choice. Please try again.");
                break;
        }

        if (choice != 4)
        {
            previousChoice = choice;
        }

        // Display the menu options after each user interaction
        DisplayOptions();
    }
}

// Method to enter a new recipe
static void EnterRecipe()
{
    Console.WriteLine("\nEnter the details for the recipe");

    Console.Write("Number of ingredients: ");
    int numIngredients = Convert.ToInt32(Console.ReadLine());

    ingredients = new string[numIngredients];
    quantities = new double[numIngredients];
    units = new string[numIngredients];

    for (int i = 0; i < numIngredients; i++)
    {
        Console.Write("Ingredient {0}: ", i + 1);
        ingredients[i] = Console.ReadLine();

        Console.Write("Quantity: ");
        quantities[i] = Convert.ToDouble(Console.ReadLine());

        Console.Write("Unit of measurement: ");
        units[i] = Console.ReadLine();
    }

    Console.Write("\nNumber of steps: ");
    int numSteps = Convert.ToInt32(Console.ReadLine());

    steps = new string[numSteps];

    Console.WriteLine("\nSteps:");
    for (int i = 0; i < numSteps; i++)
    {
        Console.Write("Step {0}: ", i + 1);
        steps[i] = Console.ReadLine();
    }

    Console.WriteLine("\nRecipe entered successfully!");
}

// Method to view the full recipe
static void ViewRecipe()
{
        Console.WriteLine("\n------------------------------ MY RECIPE ----------
--------------------");
        Console.WriteLine("\nINGREDIENTS");
        for (int i = 0; i < ingredients.Length; i++)
        {
            Console.WriteLine("{0}. {1} - {2} {3}", i + 1, ingredients[i],
quantities[i], units[i]);
        }

        Console.WriteLine("\n*****");
        Console.WriteLine("\nSTEPS");
        for (int i = 0; i < steps.Length; i++)
        {
            Console.WriteLine("Step {0}: {1}", i + 1, steps[i]);
        }
    }

    // Method to scale the recipe quantities
    static void ScaleRecipe()
    {
        if (ingredients == null)
        {
            Console.WriteLine("No recipe has been created.");
            EnterRecipe();
            return;
        }

        Console.WriteLine("\nSCALE");
        Console.WriteLine("1. Half");
        Console.WriteLine("2. Double");
        Console.WriteLine("3. Triple");

        Console.Write("\nEnter your choice: ");
        int choice;
        try
        {
            choice = Convert.ToInt32(Console.ReadLine());
        }
        catch (FormatException)
        {
            Console.WriteLine("Invalid choice. Please enter a number.");
            return;
        }

        switch (choice)
        {
            case 1:
                ScaleQuantities(0.5);
                Console.WriteLine("Recipe scaled to half.");
                break;
            case 2:
                ScaleQuantities(2);
                Console.WriteLine("Recipe scaled to double.");
                break;
            case 3:
                ScaleQuantities(3);
                Console.WriteLine("Recipe scaled to triple.");
                break;
            default:
                Console.WriteLine("Invalid choice. Please try again.");
                break;
        }
    }
    // Method to scale the quantities by a given factor
    static void ScaleQuantities(double scale)
    {
        for (int i = 0; i < quantities.Length; i++)
        {
            quantities[i] *= scale;
        }
    }

    // Method to reset the quantities to their original values
    static void ResetQuantities()
    {
        if (ingredients == null)
        {
            Console.WriteLine("No recipe has been created.");
            EnterRecipe();
            return;
        }

        InitialiseRecipe();
        Console.WriteLine("\nQuantities reset to original values.");
    }

    // Method to clear all recipe data
    static void ClearData()
    {
        InitialiseRecipe();
        Console.WriteLine("\nData cleared. Enter a new recipe to continue.");
    }

    // Method to reset all recipe arrays to null
    static void InitialiseRecipe()
    {
        ingredients = null;
        quantities = null;
        units = null;
        steps = null;
    }

    // Method to display the menu options
    static void DisplayOptions()
    {
        Console.WriteLine("\nMENU");
        Console.WriteLine("1. Enter a new recipe");
        Console.WriteLine("2. View recipe");
        Console.WriteLine("3. Scale recipe");
        Console.WriteLine("4. Reset quantities");
        Console.WriteLine("5. Clear data");
        Console.WriteLine("6. Exit");
    }
}
```

Link to GitHub:

https://github.com/Lwandle-Chauke/programming-2a-part1-new-Lwandle-Chauke

TAGS:


Releases / Part1

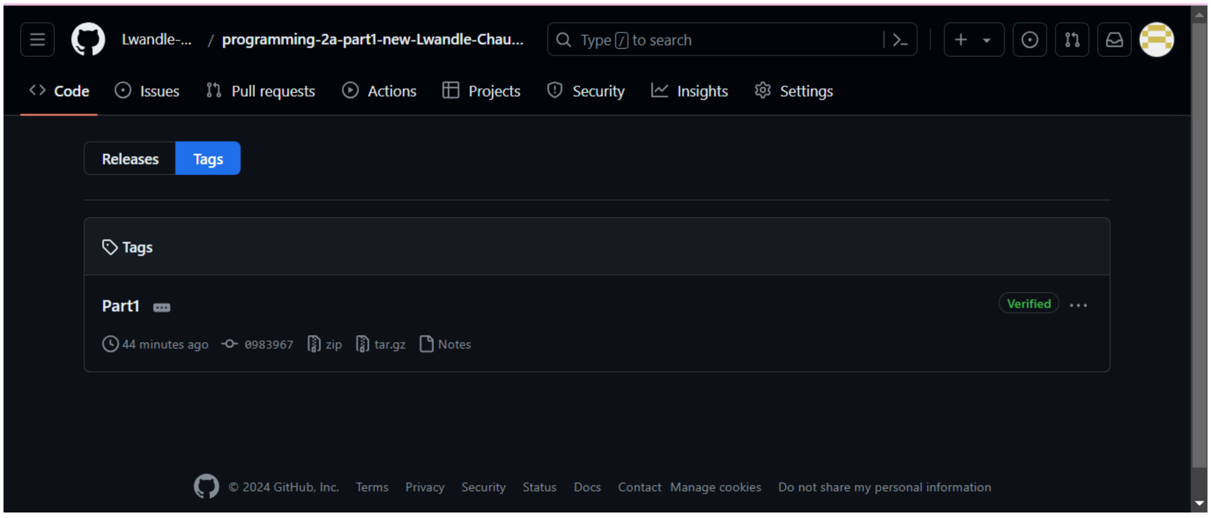
Part 1 Latest

Lwandle-Chauke released this 40 minutes ago · Part1 · 0983967

Part 1 Submission: Basic Functionality Implementation

This tag marks the completion of Part 1 of the project submission. The following key changes have been made:

1. Enter Recipe Functionality: Implemented the ability for users to enter the details of a new recipe, including ingredients and steps.
2. View Recipe Functionality: Added functionality to view the full recipe, displaying ingredients and steps in a clear and organised format.

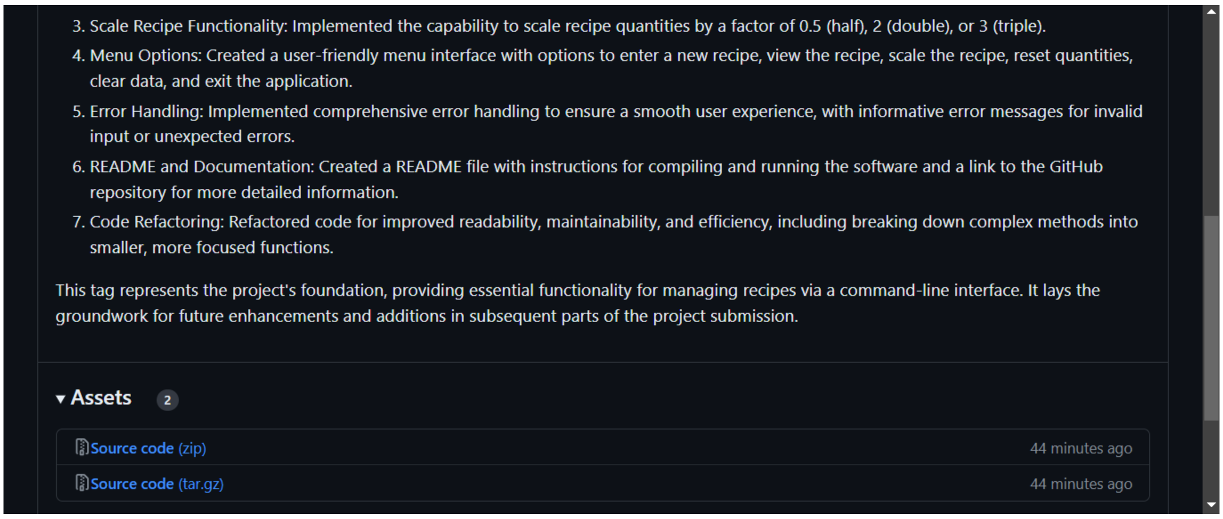

Releases | Tags

Tags

Part1 · Verified

44 minutes ago · 0983967 · zip · tar.gz · Notes

© 2024 GitHub, Inc. Terms Privacy Security Status Docs Contact Manage cookies Do not share my personal information



3. Scale Recipe Functionality: Implemented the capability to scale recipe quantities by a factor of 0.5 (half), 2 (double), or 3 (triple).
4. Menu Options: Created a user-friendly menu interface with options to enter a new recipe, view the recipe, scale the recipe, reset quantities, clear data, and exit the application.
5. Error Handling: Implemented comprehensive error handling to ensure a smooth user experience, with informative error messages for invalid input or unexpected errors.
6. README and Documentation: Created a README file with instructions for compiling and running the software and a link to the GitHub repository for more detailed information.
7. Code Refactoring: Refactored code for improved readability, maintainability, and efficiency, including breaking down complex methods into smaller, more focused functions.

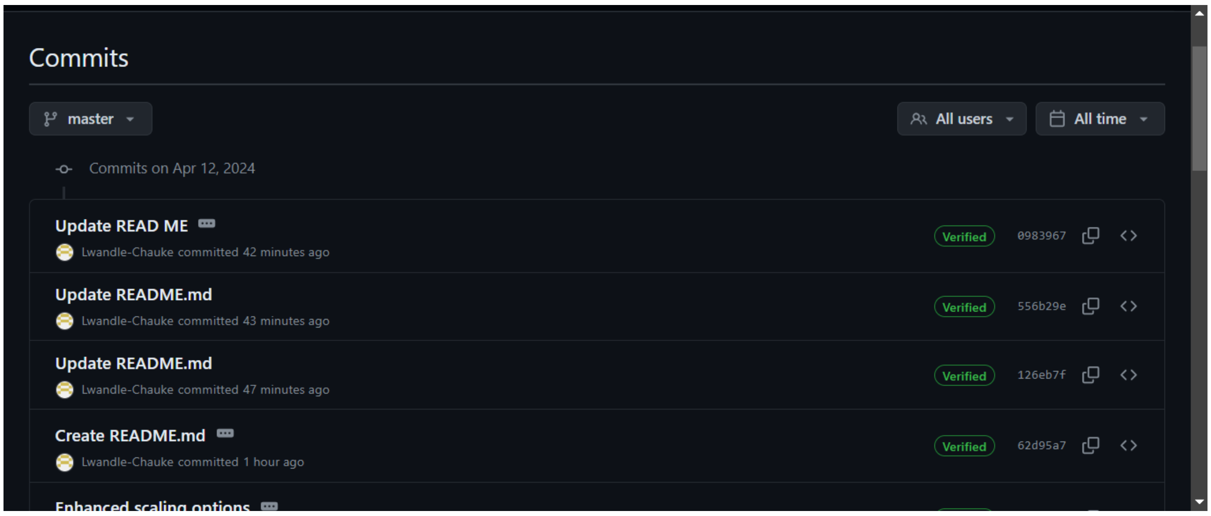
This tag represents the project's foundation, providing essential functionality for managing recipes via a command-line interface. It lays the groundwork for future enhancements and additions in subsequent parts of the project submission.

▾ Assets 2

Source code (zip) 44 minutes ago

Source code (tar.gz) 44 minutes ago

COMMITS:

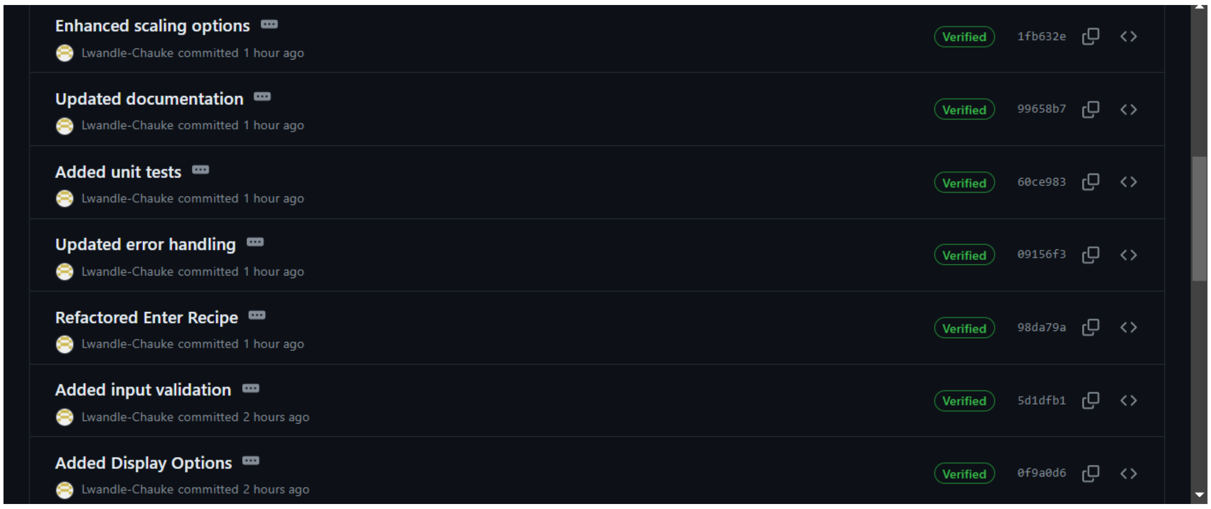


Commits

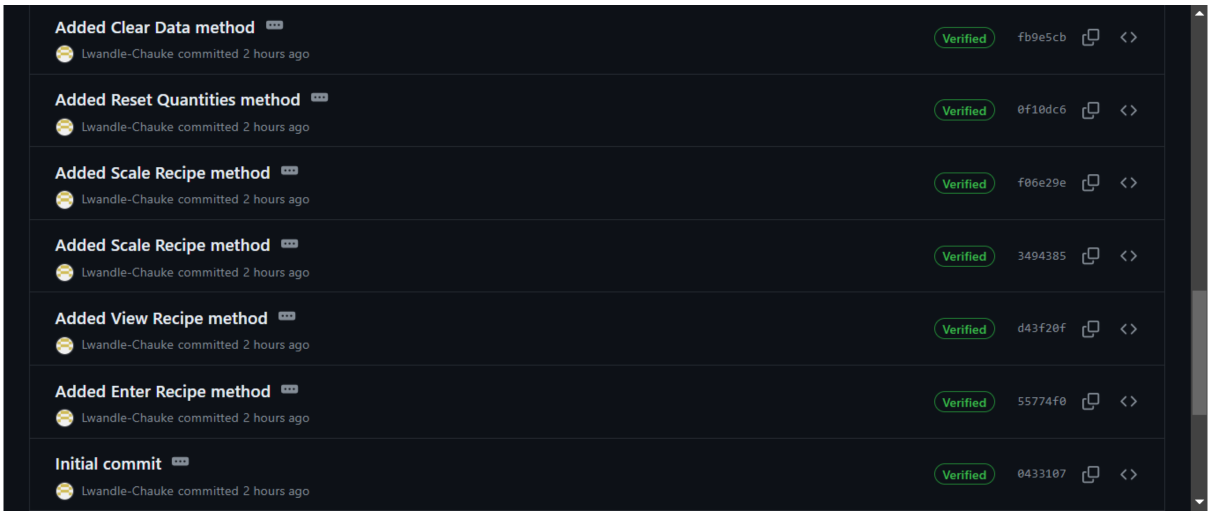
master | All users | All time

Commits on Apr 12, 2024

Update READ ME Lwandle-Chauke committed 42 minutes ago	Verified	0983967
Update README.md Lwandle-Chauke committed 43 minutes ago	Verified	556b29e
Update README.md Lwandle-Chauke committed 47 minutes ago	Verified	126eb7f
Create README.md Lwandle-Chauke committed 1 hour ago	Verified	62d95a7



Enhanced scaling options Lwandle-Chauke committed 1 hour ago	Verified	1fb632e
Updated documentation Lwandle-Chauke committed 1 hour ago	Verified	99658b7
Added unit tests Lwandle-Chauke committed 1 hour ago	Verified	60ce983
Updated error handling Lwandle-Chauke committed 1 hour ago	Verified	09156f3
Refactored Enter Recipe Lwandle-Chauke committed 1 hour ago	Verified	98da79a
Added input validation Lwandle-Chauke committed 2 hours ago	Verified	5d1dfb1
Added Display Options Lwandle-Chauke committed 2 hours ago	Verified	0f9a0d6



Added Clear Data method
Lwandle-Chauke committed 2 hours ago — Verified fb9e5cb

Added Reset Quantities method
Lwandle-Chauke committed 2 hours ago — Verified 0f10dc6

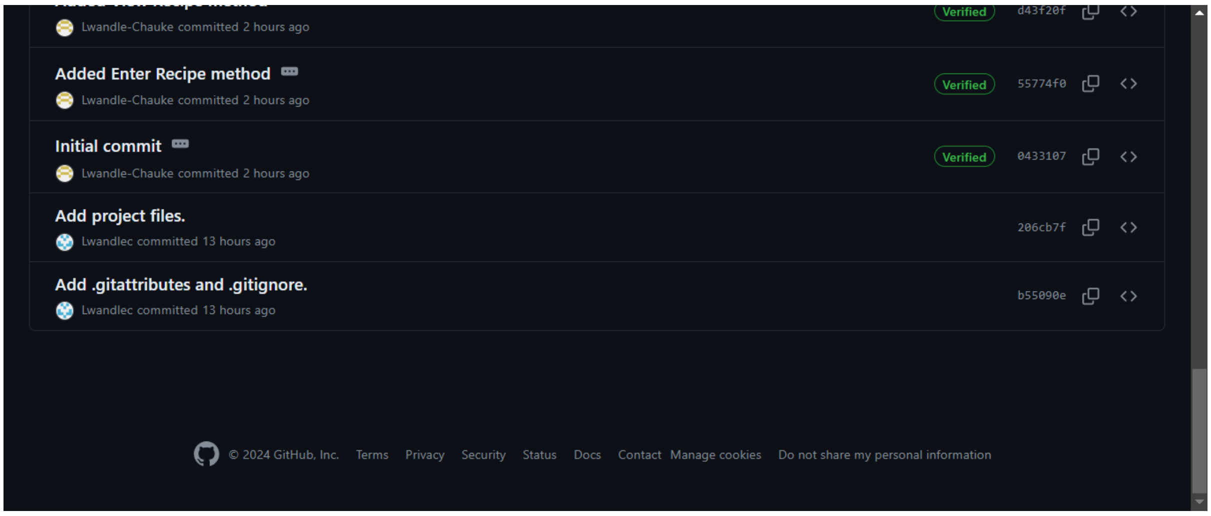
Added Scale Recipe method
Lwandle-Chauke committed 2 hours ago — Verified f06e29e

Added Scale Recipe method
Lwandle-Chauke committed 2 hours ago — Verified 3494385

Added View Recipe method
Lwandle-Chauke committed 2 hours ago — Verified d43f20f

Added Enter Recipe method
Lwandle-Chauke committed 2 hours ago — Verified 55774f0

Initial commit
Lwandle-Chauke committed 2 hours ago — Verified 0433107



Lwandle-Chauke committed 2 hours ago — Verified d43f20f

Added Enter Recipe method
Lwandle-Chauke committed 2 hours ago — Verified 55774f0

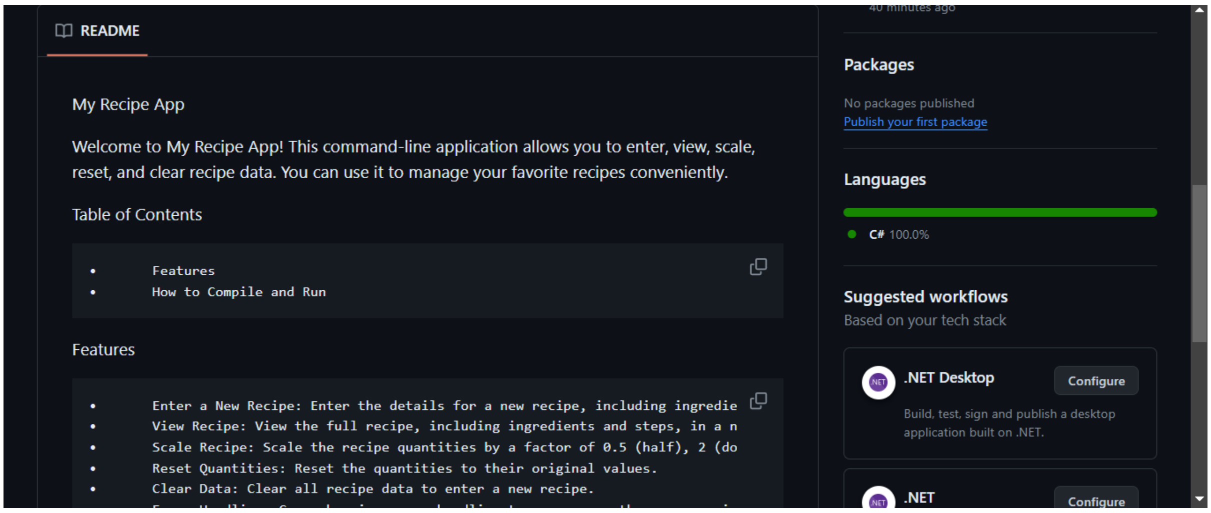
Initial commit
Lwandle-Chauke committed 2 hours ago — Verified 0433107

Add project files.
Lwandlec committed 13 hours ago — 206cb7f

Add .gitattributes and .gitignore.
Lwandlec committed 13 hours ago — b55090e

© 2024 GitHub, Inc. Terms Privacy Security Status Docs Contact Manage cookies Do not share my personal information

READ ME



README

My Recipe App

Welcome to My Recipe App! This command-line application allows you to enter, view, scale, reset, and clear recipe data. You can use it to manage your favorite recipes conveniently.

Table of Contents

```
•     Features
•     How to Compile and Run
```

Features

```
•     Enter a New Recipe: Enter the details for a new recipe, including ingredie
•     View Recipe: View the full recipe, including ingredients and steps, in a n
•     Scale Recipe: Scale the recipe quantities by a factor of 0.5 (half), 2 (do
•     Reset Quantities: Reset the quantities to their original values.
•     Clear Data: Clear all recipe data to enter a new recipe.
```

40 minutes ago

Packages

No packages published
Publish your first package

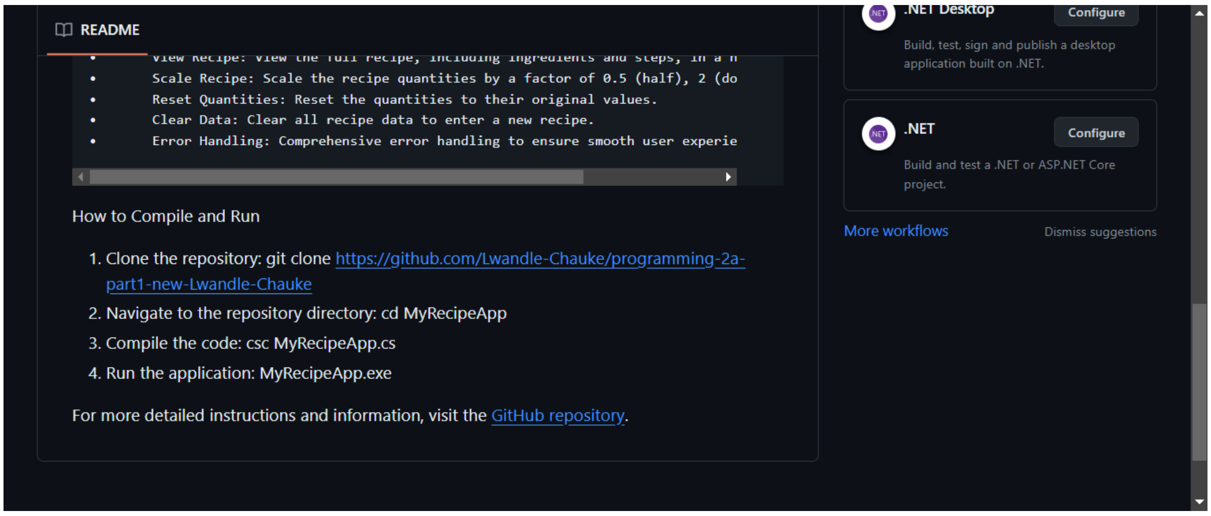
Languages

C# 100.0%

Suggested workflows
Based on your tech stack

.NET Desktop — Configure
Build, test, sign and publish a desktop application built on .NET.

.NET — Configure


README
.NET Desktop
Configure
Build, test, sign and publish a desktop application built on .NET.
.NET
Configure
Build and test a .NET or ASP.NET Core project.
More workflows
Dismiss suggestions

```
•      View Recipe: View the full recipe, including ingredients and steps, in a n
•      Scale Recipe: Scale the recipe quantities by a factor of 0.5 (half), 2 (do
•      Reset Quantities: Reset the quantities to their original values.
•      Clear Data: Clear all recipe data to enter a new recipe.
•      Error Handling: Comprehensive error handling to ensure smooth user experie
```

How to Compile and Run

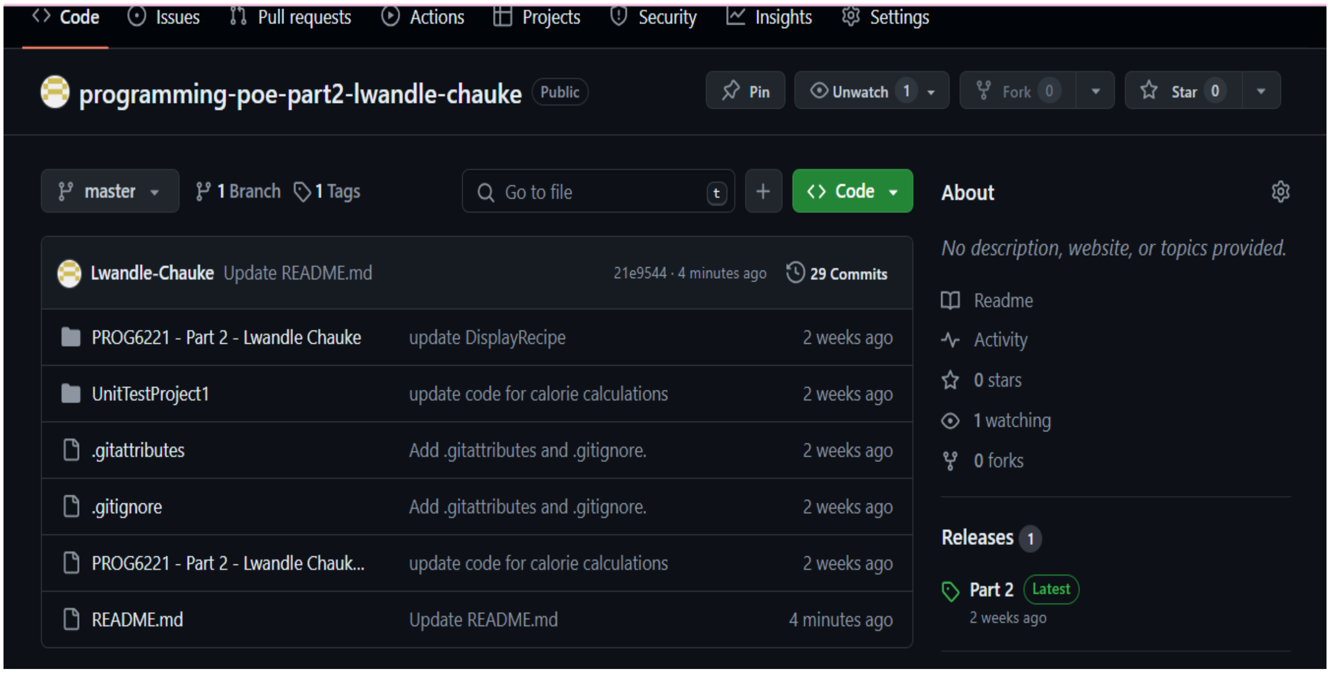
1. Clone the repository: git clone https://github.com/Lwandle-Chauke/programming-2a-part1-new-Lwandle-Chauke
2. Navigate to the repository directory: cd MyRecipeApp
3. Compile the code: csc MyRecipeApp.cs
4. Run the application: MyRecipeApp.exe

For more detailed instructions and information, visit the GitHub repository.

PART 2:

GitHub Repository link:

https://github.com/Lwandle-Chauke/programming-poe-part2-lwandle-chauke



No description, website, or topics provided.

Lwandle-Chauke Update README.md 21e9544 · 4 minutes ago 29 Commits

PROG6221 - Part 2 - Lwandle Chauke	update DisplayRecipe	2 weeks ago
UnitTestProject1	update code for calorie calculations	2 weeks ago
.gitattributes	Add .gitattributes and .gitignore.	2 weeks ago
.gitignore	Add .gitattributes and .gitignore.	2 weeks ago
PROG6221 - Part 2 - Lwandle Chauk...	update code for calorie calculations	2 weeks ago
README.md	Update README.md	4 minutes ago

Readme File:

My Recipe App README

Overview

The Recipe App is a command-line application written in C# that allows users to manage recipes. It provides functionality for adding new recipes, viewing existing ones, scaling recipes, and displaying detailed information about each recipe. Additionally, it includes a feature to warn users if the total calories of a recipe exceed 300.

Link to GitHub Repository

https://github.com/Lwandle-Chauke/programming-poe-part2-lwandle-chauke

Features

- **Add New Recipes:** Users can enter details for a new recipe, including ingredients and steps.
- **View Existing Recipes:** Users can view a list of existing recipes and select one to view its details.
- **Scale Recipes:** Recipes can be scaled by a given factor, adjusting the quantities of all ingredients accordingly.
- **Calorie Warning:** If the total calories of a recipe exceed 300, the user receives a warning message.
- **Clear Data:** Users can clear the current recipe data and start fresh.

Getting Started

1. **Clone or Download the Repository:** Clone or download the repository to your local machine.
2. **Open the Solution:** Open the solution file in Visual Studio or any C# IDE.
3. **Build and Run:** Build the solution and run the application.

Usage

- Upon launching the application, users are presented with a menu with options to:
  1. Enter a new recipe
  2. View existing recipes
  3. Scale a recipe
  4. Clear recipe data
  5. Exit the application
- Follow the prompts to perform the desired action.
- When entering a new recipe, provide details such as the recipe name, ingredients (name, quantity, unit, calories, and food group), and steps.
- If the total calories of a recipe exceed 300, a warning message is displayed.

Requirements

- .NET Framework or .NET Core installed on your machine.
- Access to a command-line interface for input and output.

Contributing

Contributions to the Recipe App are welcome! If you encounter any issues or have suggestions for improvements, please open an issue or submit a pull request on the GitHub repository.

License

This project is licensed under the [MIT License](LICENSE).

Changes Based on Lecturer's Feedback

1. Improved Error Handling:

   - Added input validation to ensure users enter valid data for ingredients and steps.

2. Enhanced Recipe Display:

   - Improved the layout of the recipe display for better readability, ensuring a neat format.

3. Recipe Scaling:

   - Improved the scaling functionality to correctly handle unit conversions, such as converting tablespoons to cups.

4. Confirmation Before Clearing Data:

   - Added a prompt to confirm before clearing existing recipe data to prevent accidental deletions.

5. Refined Class Structure:

   - Refactored the code to enhance the logical structure of classes, ensuring better encapsulation and separation of concerns.

6. Use of Dynamic Lists:

   - Replaced static arrays with dynamic lists for storing ingredients and steps, allowing for more flexible and efficient data management.

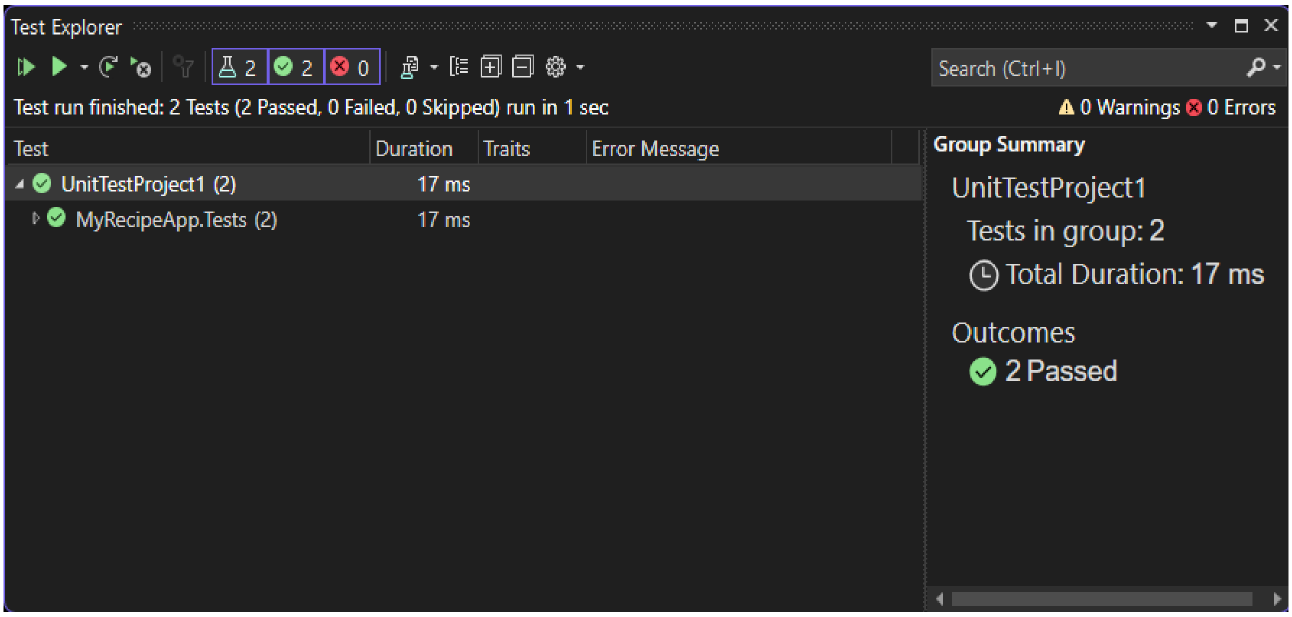
7. Improved Documentation:

   - Added detailed comments and documentation throughout the code to enhance readability and maintain coding standards.

These changes improve the functionality, usability, and structure of the application, addressing the feedback provided and ensuring a better user experience.

Unit Tests:

```
using Microsoft.VisualStudio.TestTools.UnitTesting;
using System;
using System.Collections.Generic;

public class Recipe
{
    public string Name { get; set; }
    public List<Ingredient> Ingredients { get; set; } = new
List<Ingredient>();

    public double CalculateTotalCalories()
    {
        double totalCalories = 0;
        foreach (var ingredient in Ingredients)
        {
            totalCalories += ingredient.Calories;
        }
        return totalCalories;
    }
}

// Ingredient.cs
public class Ingredient
{
    public string Name { get; set; }
    public double Quantity { get; set; }
    public string Unit { get; set; }
    public double Calories { get; set; }
    public string FoodGroup { get; set; }
}

// RecipeTests.cs
namespace MyRecipeApp.Tests
{
    [TestClass]
    public class RecipeTests
    {
        [TestMethod]
        public void TestCalculateTotalCalories_SimpleRecipe()
        {
            // Arrange
            Recipe recipe = new Recipe();
            recipe.Ingredients.Add(new Ingredient { Calories = 100 });
            recipe.Ingredients.Add(new Ingredient { Calories = 200 });

            // Act
            double totalCalories = recipe.CalculateTotalCalories();

            // Assert
            Assert.AreEqual(300, totalCalories);
        }

        [TestMethod]
        public void TestCalculateTotalCalories_ComplexRecipe()
        {
            // Arrange
            Recipe recipe = new Recipe();
            recipe.Ingredients.Add(new Ingredient { Calories = 50 });
            recipe.Ingredients.Add(new Ingredient { Calories = 100 });
            recipe.Ingredients.Add(new Ingredient { Calories = 200 });

            // Act
            double totalCalories = recipe.CalculateTotalCalories();

            // Assert
            Assert.AreEqual(350, totalCalories);
        }
    }
}
```



```
Test run finished: 2 Tests (2 Passed, 0 Failed, 0 Skipped) run in 1 sec        0 Warnings  0 Errors

Test                          Duration   Traits   Error Message
UnitTestProject1 (2)          17 ms
  MyRecipeApp.Tests (2)       17 ms

Group Summary
UnitTestProject1
  Tests in group: 2
  Total Duration: 17 ms
Outcomes
  2 Passed
```

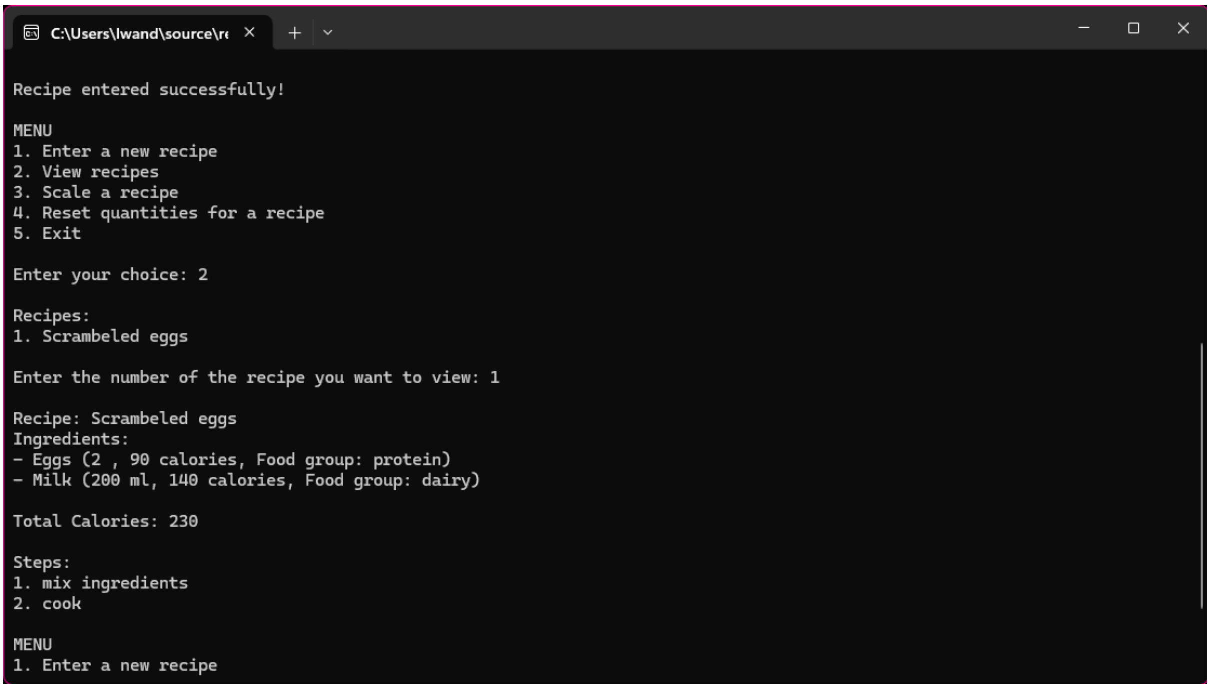
Code:

```
using System;
using System.Collections.Generic;

// Class representing an ingredient
class Ingredient
{
    // Ingredient properties
    public string Name { get; set; }
    public double Quantity { get; set; }
    public string Unit { get; set; }
    public double Calories { get; set; }
    public string FoodGroup { get; set; }
}

// Class representing a recipe
class Recipe
{
    public string Name { get; set; }
    public List<Ingredient> Ingredients { get; set; }
    public List<string> Steps { get; set; }
    public delegate void CalorieNotificationHandler(Recipe recipe);
    public event CalorieNotificationHandler CalorieNotification;

    // Method to calculate the total calories of the recipe
    public double CalculateTotalCalories()
    {
        double totalCalories = 0;
        foreach (var ingredient in Ingredients)
        {
            totalCalories += ingredient.Calories;
        }
        return totalCalories;
    }
    // Method to check if the total calories exceed 300 calories to trigger
notification
    public void CheckCalories()
    {
        if (CalculateTotalCalories() > 300)
        {
            CalorieNotification?.Invoke(this);
        }
    }

    // Method to scale the quantities of ingredients
    public void ScaleQuantities(double factor)
    {
        foreach (var ingredient in Ingredients)
        {
            ingredient.Quantity *= factor;
        }
    }

    // Method to reset the quantities to their original values
    public void ResetQuantities()
    {
        // Restore original quantities
        foreach (var ingredient in Ingredients)
        {
            ingredient.Quantity = ingredient.Quantity;
        }
    }
}

class MyRecipeApp
{
    static List<Recipe> recipes = new List<Recipe>();

    static void Main(string[] args)
    {
        Console.WriteLine("Welcome to My Recipe App!");


        DisplayOptions(); // Display the menu options


        while (true) // Loop to keep the program running until the user
chooses to exit
        {
            Console.Write("\nEnter your choice: ");
            int choice = Convert.ToInt32(Console.ReadLine());


            switch (choice)
            {
                case 1:
                    EnterRecipe(); // Option to enter a new recipe
                    break;
                case 2:
                    ViewRecipes(); // Option to view existing recipes
                    break;
                case 3:
                    ScaleRecipe(); // Option to scale a recipe
                    break;
                case 4:
                    ResetQuantities(); // Option to reset recipe quantities
                    break;
                case 5:
                    Console.WriteLine("Thank you for using Recipe App!");
                    return; // Exit the program
                default:
                    Console.WriteLine("Invalid choice. Please try again.");
                    break;
            }


            DisplayOptions(); // Display the menu options again
        }
    }
```

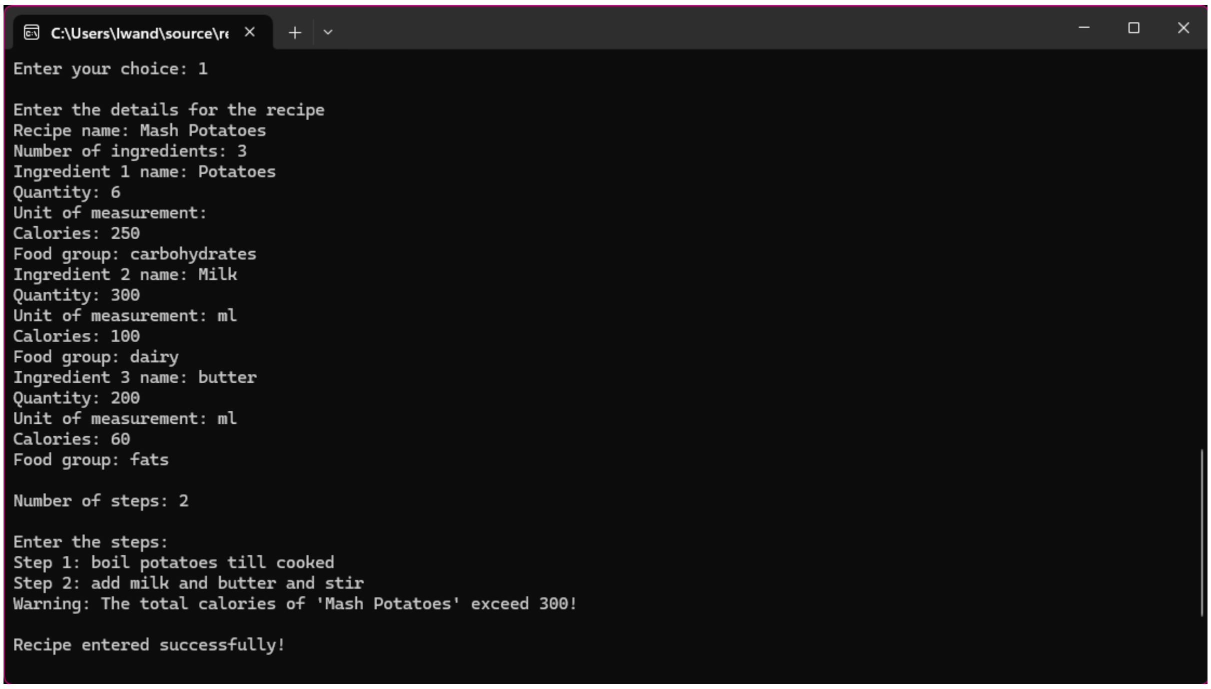
```
// Method to enter a new recipe
static void EnterRecipe()
{
    Recipe recipe = new Recipe();
    recipe.Ingredients = new List<Ingredient>();
    recipe.Steps = new List<string>();

    Console.WriteLine("\nEnter the details for the recipe");

    Console.Write("Recipe name: ");
    recipe.Name = Console.ReadLine();

    Console.Write("Number of ingredients: ");
    int numIngredients = Convert.ToInt32(Console.ReadLine());

    for (int i = 0; i < numIngredients; i++)
    {
        Ingredient ingredient = new Ingredient();

        Console.Write("Ingredient {0} name: ", i + 1);
        ingredient.Name = Console.ReadLine();

        Console.Write("Quantity: ");
        ingredient.Quantity = Convert.ToDouble(Console.ReadLine());

        Console.Write("Unit of measurement: ");
        ingredient.Unit = Console.ReadLine();

        Console.Write("Calories: ");
        ingredient.Calories = Convert.ToDouble(Console.ReadLine());

        Console.Write("Food group: ");
        ingredient.FoodGroup = Console.ReadLine();

        recipe.Ingredients.Add(ingredient);
    }
```

```
        Console.Write("\nNumber of steps: ");
        int numSteps = Convert.ToInt32(Console.ReadLine());

        Console.WriteLine("\nEnter the steps:");
        for (int i = 0; i < numSteps; i++)
        {
            Console.Write("Step {0}: ", i + 1);
            recipe.Steps.Add(Console.ReadLine());
        }

        recipes.Add(recipe);
        recipe.CalorieNotification += Recipe_CalorieNotification;
        recipe.CheckCalories();

        // Sort recipes by name
        recipes.Sort((x, y) => string.Compare(x.Name, y.Name));

        Console.WriteLine("\nRecipe entered successfully!");
    }

    // Method to handle calorie notification
    static void Recipe_CalorieNotification(Recipe recipe)
    {
        Console.WriteLine($"Warning: The total calories of '{recipe.Name}'
exceed 300!");
    }

    // Method to view existing recipes
    static void ViewRecipes()
    {
        if (recipes.Count == 0)
        {
            Console.WriteLine("No recipes available.");
            return;
        }
        Console.WriteLine("\nRecipes:");

        // Display recipes with numeric identifiers and names
        for (int i = 0; i < recipes.Count; i++)
        {
            Console.WriteLine($"{i + 1}. {recipes[i].Name}");
        }

        Console.Write("\nEnter the number of the recipe you want to view: 
");
        int recipeNumber = Convert.ToInt32(Console.ReadLine());

        if (recipeNumber > 0 && recipeNumber <= recipes.Count)
        {
            DisplayRecipe(recipes[recipeNumber - 1]);
        }
        else
        {
            Console.WriteLine("Invalid recipe number.");
        }
    }

    // Method to display the details of a recipe, including total calories
    static void DisplayRecipe(Recipe recipe)
    {
        Console.WriteLine("\nRecipe: " + recipe.Name);
        Console.WriteLine("Ingredients:");
        double totalCalories = 0; // Variable to store total calories
        foreach (var ingredient in recipe.Ingredients)
        {
            totalCalories += ingredient.Calories; // Add calories of each 
ingredient to total
            Console.WriteLine("- {0} ({1} {2}, {3} calories, Food group: 
{4})", ingredient.Name, ingredient.Quantity, ingredient.Unit, 
ingredient.Calories, ingredient.FoodGroup);
        }
        Console.WriteLine("\nTotal Calories: " + totalCalories); // Display 
total calories
```

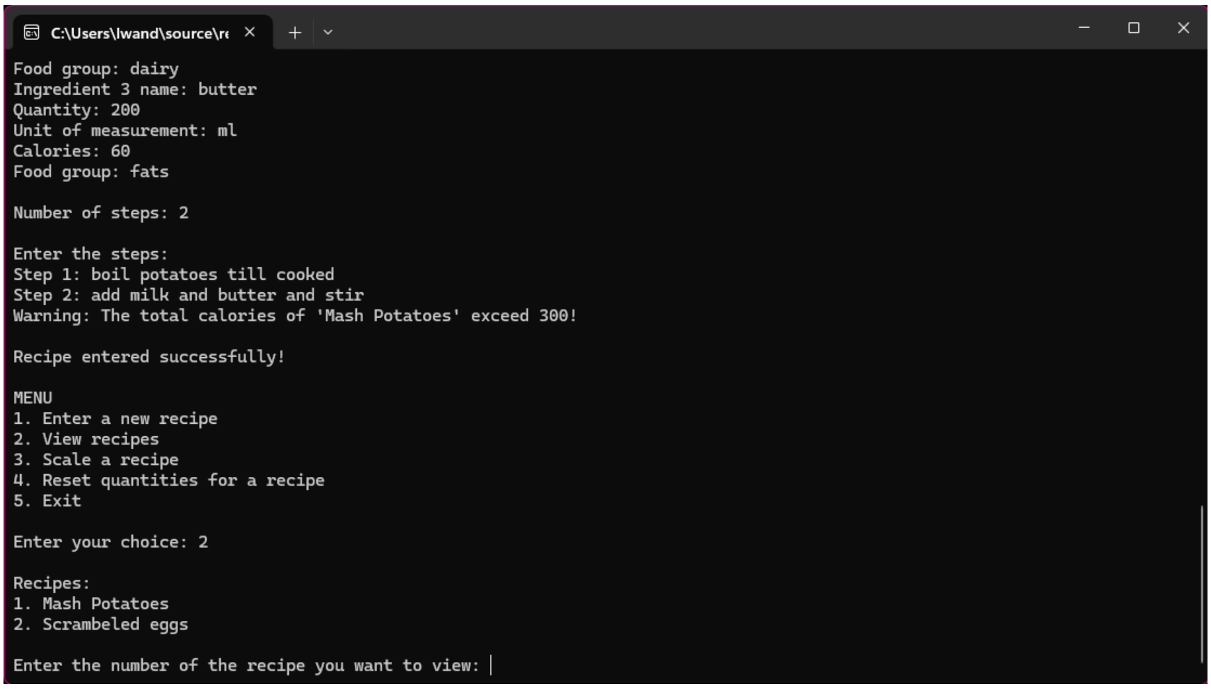
```
        Console.WriteLine("\nSteps:");
        for (int i = 0; i < recipe.Steps.Count; i++)
        {
            Console.WriteLine("{0}. {1}", i + 1, recipe.Steps[i]);
        }
    }



    // Method to scale a recipe
    static void ScaleRecipe()
    {
        if (recipes.Count == 0)
        {
            Console.WriteLine("No recipes available.");
            return;
        }

        Console.WriteLine("\nRecipes:");

        // Display recipes with numeric identifiers and names
        for (int i = 0; i < recipes.Count; i++)
        {
            Console.WriteLine($"{i + 1}. {recipes[i].Name}");
        }

        Console.Write("\nEnter the number of the recipe you want to scale: 
");
        int recipeNumber = Convert.ToInt32(Console.ReadLine());

        if (recipeNumber > 0 && recipeNumber <= recipes.Count)
        {
            Console.WriteLine("Choose scaling factor:");
            Console.WriteLine("1. Half");
            Console.WriteLine("2. Double");
            Console.WriteLine("3. Triple");
            Console.Write("Enter your choice: ");
            int choice = Convert.ToInt32(Console.ReadLine());

            Recipe selectedRecipe = recipes[recipeNumber - 1];

            switch (choice)
            {
                case 1:
                    selectedRecipe.ScaleQuantities(0.5);
                    Console.WriteLine("Recipe scaled to half.");
                    break;
                case 2:
                    selectedRecipe.ScaleQuantities(2);
                    Console.WriteLine("Recipe scaled to double.");
                    break;
                case 3:
                    selectedRecipe.ScaleQuantities(3);
                    Console.WriteLine("Recipe scaled to triple.");
                    break;
                default:
                    Console.WriteLine("Invalid choice. Please try again.");
                    break;
            }
        }
        else
        {
            Console.WriteLine("Invalid recipe number.");
        }
    }

    // Method to reset quantities of a recipe
    static void ResetQuantities()
    {
        if (recipes.Count == 0)
        {
            Console.WriteLine("No recipes available.");
            return;
        }

        Console.WriteLine("\nRecipes:");

        // Display recipes with numeric identifiers and names
        for (int i = 0; i < recipes.Count; i++)
        {
            Console.WriteLine($"{i + 1}. {recipes[i].Name}");
        }

        Console.Write("\nEnter the number of the recipe you want to reset
quantities for: ");
        int recipeNumber = Convert.ToInt32(Console.ReadLine());

        if (recipeNumber > 0 && recipeNumber <= recipes.Count)
        {
            Recipe selectedRecipe = recipes[recipeNumber - 1];
            selectedRecipe.ResetQuantities();
            Console.WriteLine("Quantities reset for the selected recipe.");
        }
        else
        {
            Console.WriteLine("Invalid recipe number.");
        }
    }

    // Method to display menu options
    static void DisplayOptions()
    {
        Console.WriteLine("\nMENU");
        Console.WriteLine("1. Enter a new recipe");
        Console.WriteLine("2. View recipes");
        Console.WriteLine("3. Scale a recipe");
        Console.WriteLine("4. Reset quantities for a recipe");
        Console.WriteLine("5. Exit");
        }

}
```



```
Recipe entered successfully!

MENU
1. Enter a new recipe
2. View recipes
3. Scale a recipe
4. Reset quantities for a recipe
5. Exit

Enter your choice: 2

Recipes:
1. Scrambeled eggs

Enter the number of the recipe you want to view: 1

Recipe: Scrambeled eggs
Ingredients:
- Eggs (2 , 90 calories, Food group: protein)
- Milk (200 ml, 140 calories, Food group: dairy)

Total Calories: 230

Steps:
1. mix ingredients
2. cook

MENU
1. Enter a new recipe
```

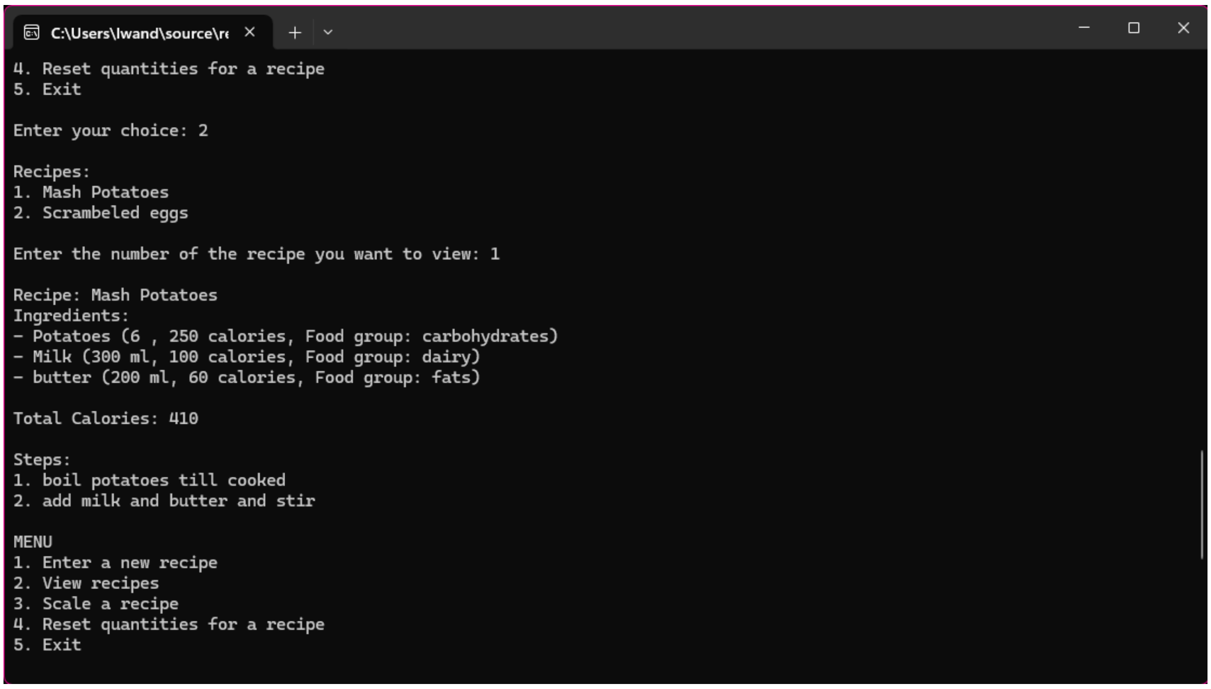


```
Enter your choice: 1

Enter the details for the recipe
Recipe name: Mash Potatoes
Number of ingredients: 3
Ingredient 1 name: Potatoes
Quantity: 6
Unit of measurement:
Calories: 250
Food group: carbohydrates
Ingredient 2 name: Milk
Quantity: 300
Unit of measurement: ml
Calories: 100
Food group: dairy
Ingredient 3 name: butter
Quantity: 200
Unit of measurement: ml
Calories: 60
Food group: fats

Number of steps: 2

Enter the steps:
Step 1: boil potatoes till cooked
Step 2: add milk and butter and stir
Warning: The total calories of 'Mash Potatoes' exceed 300!

Recipe entered successfully!
```



C:\Users\lwand\source\re

```
Food group: dairy
Ingredient 3 name: butter
Quantity: 200
Unit of measurement: ml
Calories: 60
Food group: fats

Number of steps: 2

Enter the steps:
Step 1: boil potatoes till cooked
Step 2: add milk and butter and stir
Warning: The total calories of 'Mash Potatoes' exceed 300!

Recipe entered successfully!

MENU
1. Enter a new recipe
2. View recipes
3. Scale a recipe
4. Reset quantities for a recipe
5. Exit

Enter your choice: 2

Recipes:
1. Mash Potatoes
2. Scrambeled eggs

Enter the number of the recipe you want to view:
```



C:\Users\lwand\source\re

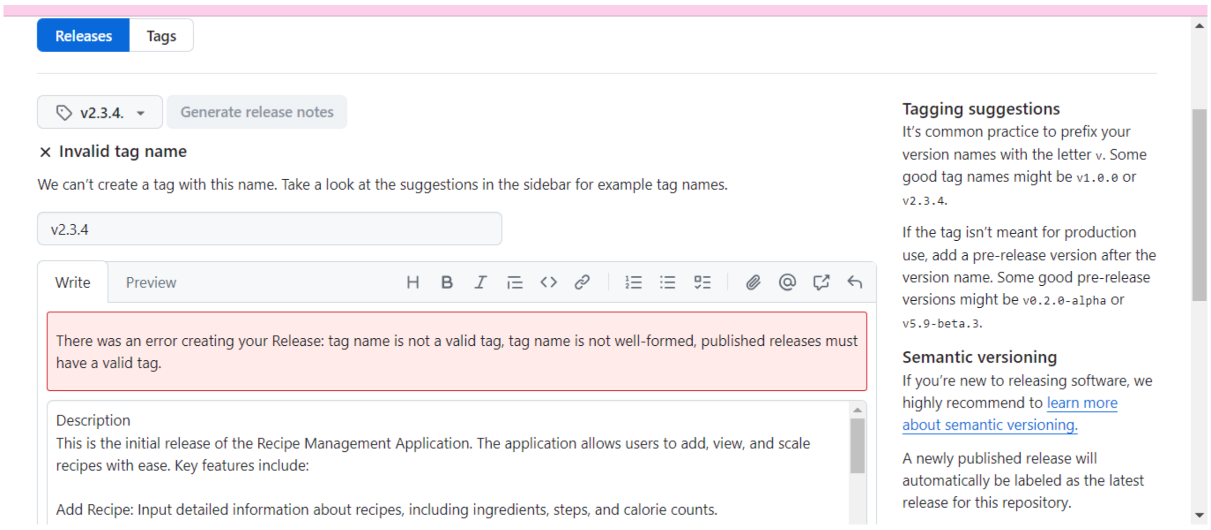
```
4. Reset quantities for a recipe
5. Exit

Enter your choice: 2

Recipes:
1. Mash Potatoes
2. Scrambeled eggs

Enter the number of the recipe you want to view: 1

Recipe: Mash Potatoes
Ingredients:
- Potatoes (6 , 250 calories, Food group: carbohydrates)
- Milk (300 ml, 100 calories, Food group: dairy)
- butter (200 ml, 60 calories, Food group: fats)

Total Calories: 410

Steps:
1. boil potatoes till cooked
2. add milk and butter and stir

MENU
1. Enter a new recipe
2. View recipes
3. Scale a recipe
4. Reset quantities for a recipe
5. Exit
```

PART 3:

GitHub Repository Link:

https://github.com/IIEMSA/programming-2a-poe-Lwandle-Chauke

Release Tags:



× Invalid tag name

We can't create a tag with this name. Take a look at the suggestions in the sidebar for example tag names.

v2.3.4

There was an error creating your Release: tag name is not a valid tag, tag name is not well-formed, published releases must have a valid tag.

Description
This is the initial release of the Recipe Management Application. The application allows users to add, view, and scale recipes with ease. Key features include:

Add Recipe: Input detailed information about recipes, including ingredients, steps, and calorie counts.

Tagging suggestions
It's common practice to prefix your version names with the letter `v`. Some good tag names might be `v1.0.0` or `v2.3.4`.

If the tag isn't meant for production use, add a pre-release version after the version name. Some good pre-release versions might be `v0.2.0-alpha` or `v5.9-beta.3`.

Semantic versioning
If you're new to releasing software, we highly recommend to learn more about semantic versioning.

A newly published release will automatically be labeled as the latest release for this repository.

Description

This is the initial release of the Recipe Management Application. The application allows users to add, view, and scale recipes with ease.

Key features include:

- Add Recipe: Input detailed information about recipes, including ingredients, steps, and calorie counts.
- View Recipes: Browse and view saved recipes in alphabetical order.
- Scale Recipe: Scale the quantity of ingredients by factors of 0.5, 2, or 3.

Features

Add Recipe:

Enter the recipe name, number of ingredients, ingredient details (name, food group, calories), and preparation steps.

Save recipes with a simple click.

Automatic notification if total calories exceed 300.

View Recipes:

Display all recipes in alphabetical order.

Click on a recipe to view detailed information.

Scale Recipe:

Select a recipe to scale.

Choose a scaling factor (0.5, 2, or 3).

Confirm scaling and receive a success message.

Improvements and Fixes

Logic Enhancements: Improved the logic for scaling recipes and handling ingredient details based on lecture feedback.

User Interface: Enhanced the UI for better user experience, including a pink background and white buttons for a clean and intuitive design.

Getting Started

To get started with the Recipe Management Application:

Clone the Repository:

bash

Copy code

```
git clone https://github.com/yourusername/recipe-management-app.git

cd recipe-management-app
```

Build and Run the Application:

Open the project in Visual Studio.

Build the solution and run the application.

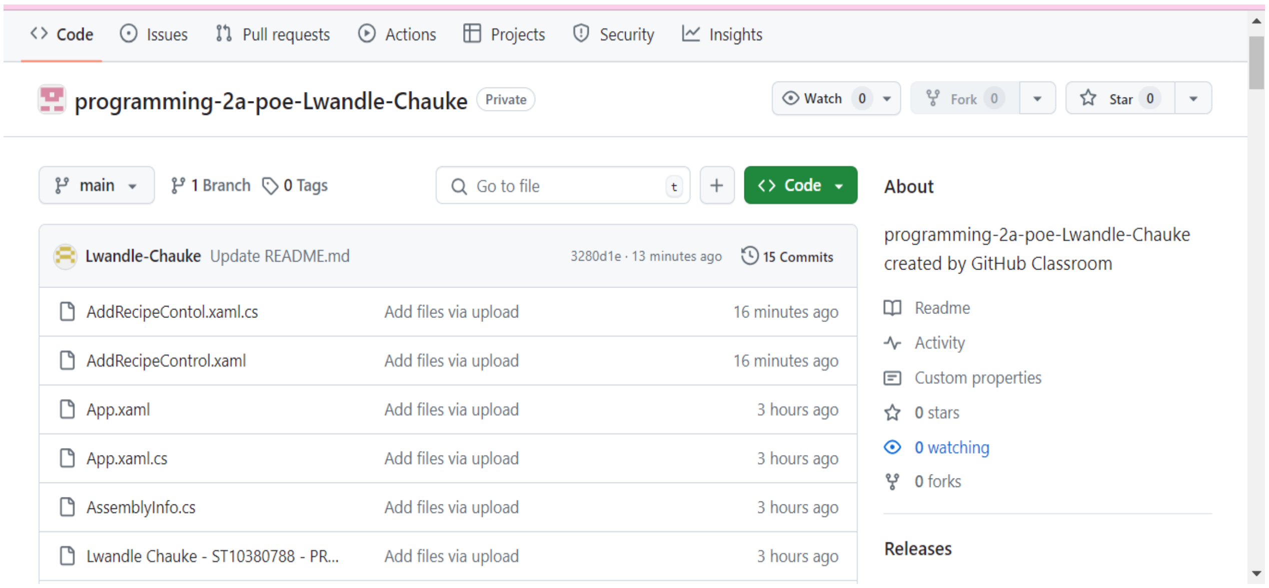
Contact

For any questions, issues, or contributions, please reach out:

Email: support@recipeapp.com

GitHub Issues: https://github.com/yourusername/recipe-management-app/issues

Thank you for using the Recipe Management Application!



programming-2a-poe-Lwandle-Chauke Private

main · 1 Branch · 0 Tags

Lwandle-Chauke Update README.md · 3280d1e · 13 minutes ago · 15 Commits

AddRecipeContol.xaml.cs	Add files via upload	16 minutes ago
AddRecipeControl.xaml	Add files via upload	16 minutes ago
App.xaml	Add files via upload	3 hours ago
App.xaml.cs	Add files via upload	3 hours ago
AssemblyInfo.cs	Add files via upload	3 hours ago
Lwandle Chauke - ST10380788 - PR...	Add files via upload	3 hours ago

About

programming-2a-poe-Lwandle-Chauke created by GitHub Classroom

Readme
Activity
Custom properties
0 stars
0 watching
0 forks

Releases

READ ME:

README for Recipe Management Application

Table of Contents

Introduction

Welcome to the Recipe Management Application! This application is designed to help users manage their recipes efficiently, including adding new recipes, viewing existing recipes, and scaling ingredients.

Features

- Add new recipes with ingredient details and preparation steps.

- View and sort recipes in alphabetical order.

- Scale recipes by 0.5, 2, or 3 times the original quantities.

System Requirements

- Operating System: Windows 7 or later

- .NET Framework: 4.7.2 or later

- RAM: 2 GB or more

- Disk Space: 100 MB of free space

- Screen Resolution: 1024x768 or higher

Installation

1. Download the Installer:

- Obtain the installer from the provided source or download link.

2. Run the Installer:

- Double-click the downloaded installer file.

- Follow the on-screen instructions to complete the installation.

- Accept the license agreement and choose the installation directory.

3. Finish Installation:

- Once the installation is complete, click 'Finish'.

- The application should now be installed on your system.

Usage

Add Recipe

1. Launch the Application:

- Open the application via the desktop shortcut or Start Menu.

2. Select 'Add Recipe':

- Click the 'Add Recipe' button from the main menu.

3. Enter Recipe Details:

- Input the recipe name, number of ingredients, ingredient details (name, food group, calories), number of steps, and step descriptions.

4. Save Recipe:

- Click the 'Save Recipe' button to save the recipe to your collection.

View Recipes

1. **Select 'View Recipes'**:

- Click the 'View Recipes' button from the main menu.

2. Browse Recipes:

- Recipes are displayed in alphabetical order.

- Click on a recipe to view more details.

Scale Recipe

1. Select 'Scale Recipe':

- Click the 'Scale Recipe' button from the main menu.

2. Choose Recipe and Scale Factor:

- Select a recipe from the drop-down list.

- Choose a scale factor (0.5, 2, or 3) by clicking the corresponding button.

3. Confirm Scaling:

- A message box will confirm that the recipe has been successfully scaled.

Troubleshooting

- Application Not Launching:

- Ensure your system meets the requirements.

- Reinstall the application if necessary.

- Error Messages:

- Ensure all fields are filled correctly when adding a recipe.

- Restart the application if errors persist.

- Recipes Not Saving:

- Check that all required fields are completed.

- Ensure there is sufficient disk space.

Feedback and Contributions

We welcome feedback and contributions to improve the Recipe Management Application. If you have suggestions, encounter bugs, or wish to contribute to the project, please contact us at support@recipeapp.com.

Lecture Feedback

Based on feedback from our lectures, we have made several improvements to the application:

- Logic Fixes: We have refined the logic in part 2 of the code to ensure better performance and user experience. Specific issues related to recipe scaling and ingredient management have been addressed.

- User Interface Enhancements: Based on user feedback, we have updated the UI to be more intuitive and visually appealing, including adding clear instructions and warning messages for calorie limits.

License

This project is licensed under the MIT License. See the LICENSE file for more information.

Contact

For any inquiries or support requests, please contact:

- Email: support@recipeapp.com

- Phone: 123-456-7890