

**Project Title: Personalized Recipe Book to Bring An Engaging Experience to People who Enjoy Cooking.**

Author(s): Eric Gu, Abigail Boleng, Yu-Hsiang Huang

Northeastern University

EECE 2140. Computing Fundamentals for Engineers Fall 2024

Date: 11/22/24

**Abstract**

This project develops a Recipe Book Application aimed at simplifying the process of recipe management and providing personalized meal suggestions based on available ingredients. Utilizing Python for backend logic, Tkinter for the graphical user interface (GUI), and Pillow (PIL) for image handling, the application is built on Object-Oriented Programming (OOP) principles. Core components include classes for recipes, ingredients, and recipe management, designed to allow users to edit, add, and remove recipe data efficiently. The application reads from and writes to recipe files, offering a dynamic and customizable cooking experience. Key features include an ingredient-based recipe search, the ability to edit recipe details, and basic image integration for recipe visuals. Limitations in time, user compatibility, and technical complexity hindered further development of the GUI, but planned improvements involve refining search functionality, enhancing the editing process, and allowing for more advanced image management. The project highlights the integration of fundamental programming concepts with practical, user-centered design, demonstrating a scalable solution for recipe tracking and meal planning. Future enhancements could include expanded search filters and advanced user interface features to improve usability and user engagement.

# Contents

1. [Introduction](#_bookmark0) 2
   1. [Background](#_bookmark1) 2
   2. [Problem Statement](#_bookmark2) 2
   3. [Objectives](#_bookmark3) 2
   4. [Scope](#_bookmark4) 2
2. [Technical Approaches and Code UML](#_bookmark5) 3
   1. [Development Environment](#_bookmark6) 3
   2. [Data Collection and Preparation](#_bookmark7) 3
   3. [Implementation Details](#_bookmark8) 3
3. [Project Demonstration](#_bookmark9) 4
   1. [Screenshots and Code Snippets](#_bookmark10) 4
4. [Discussion and Future Work](#_bookmark11) 5
5. [Conclusion](#_bookmark12) 6

**Chapter 1**

# Introduction

## Background

Our project uses GUI and the Python language to create user interface for users to make a recipe book. We use data structures including Strings, Lists, and Integers.

## Problem Statement

Creating a Recipe Book that users can be personalized through creating a user interface.

## Objectives

Objectives:

Be able to enter new recipes with ingredients and steps.

Be able to edit and delete recipes

Be able to search recipes through selected keywords.

Be able to sort recipes in alphabetical order for clear navigation.

## Scope

A GUI is covered in the project to present the user interface. Python is the coding language used to make this project. All data sent to a text file so that data from previous use can be saved into the user’s next use. Things that are not included are a return button on the user interface and the mood aspect of the recipe has not been added yet.

**Chapter 2**

# Technical Approaches and Code UML

## Development Environment

The Recipe Tracker Application was developed using Python as the primary programming language. The graphical user interface (GUI) was designed using **Tkinter**, which provided a framework for creating interactive elements such as windows, buttons, labels, and entry fields, enabling seamless user interaction. The **Pillow (PIL)** library was utilized for image handling, allowing for the inclusion and manipulation of recipe visuals to enhance the user experience. The application was structured using Object-Oriented Programming (OOP) principles, which facilitated the creation of modular and reusable components, including classes for Recipe, Ingredients, Recipe Manager, and GUI. This approach ensured efficient data handling and functionality within the software.

Core data structures, such as lists, strings, and integers, were used to store and process recipe information, including ingredient details and recipe steps. Development was carried out in an integrated development environment (IDE) like **VS Code**, which provided essential features such as debugging tools, syntax highlighting, and version control to streamline the workflow. This robust development environment allowed for effective testing and iterative improvements, ensuring the application’s scalability, compatibility, and maintainability.

## Data Collection and Preparation

The database we only have in our code file is the Recipe\_Database.txt. It stores multiple recipe data both found online and also made by us. Resources referenced online for database creation includes allrecipes.com and bbcgoodfood.com. We chose to use a text file due to its simplicity and easy for editing and storage. The formatting of the recipe's title, ingredients, and steps were kept organized and allows efficient read and write into the file. The reading and writing functions ensures data integrity when processing the database when the system interacts with users(add recipe, edit recipe, delete recipe, etc).

## Implementation Details

**Step 1: Project Planning and Requirements Gathering**

* Literature Review: Inspiration from some videos when it came to the design of our GUI. Guidance for Editing File & Reading Data from File [1][2][3][4].
* We defined the primary purpose of the project: to create a recipe management application where users can add, view, edit, and delete recipes.
* We identified core functionalities, such as storing recipes, managing ingredients, and maintaining user data persistently via file storage.
* Initialized all data and class creation to utilize object-oriented programming and store data in unique data structures.
* Utilized file reading and writing capabilities to access stored data for routine access by user.
* We selected Python as the programming language, with Tkinter for the GUI and Pillow (PIL) for image handling. A part of our initial pseudocode can be seen in Diagram 1. Additionally, our initial vision for the user interface setup can be seen in Diagram 2.
* Our project aim is to create a simple and versatile recipe-managing app that has features that can add, edit, view, and delete recipes. The decision to use Tkinter allows users to interact with our software more easily. With a robust database and efficient management, it would align user expectations of convenience and practicality.

A diagram of steps and ingredients

Description automatically generated

Diagram 1. Initial Pseudocode for Recipe Class Instantiation

A diagram of a recipe

Description automatically generated

Diagram 2. Initial Vision for the User Interface Setup using GUI

**Step 2: Setting Up the Classes and OOP Design**

* Adopted Object-Oriented Programming (OOP) to create modular and reusable components. Encapsulating ingredient and recipe attributes with their own class provides easier maintenance and edit.
* Key Classes Defined:
* There are 4 major classes in the program: Ingredient class, Recipe class, RecipeManager class, and RecipeApp class. The Ingredient class is a building block for the Recipe class, and it includes necessary data: ingredient name, quantity, and unit. The RecipeManager class mostly deals with the text file database by loading recipes from the text file and saving recipes to the text file. The RecipeApp class is the major class that integrates the user interface with tkinter. It allows users to add new recipes to the database, search for a specific stored recipe, edit a specific recipe, and delete unwanted recipes.
* Algorithms for Getter/Setter Methods:
  + Implemented encapsulation using private attributes (self.\_attribute) with getter and setter methods for controlled access.

**Step 3: Implementing File Handling for Persistent Data**

* Choose a plain text file (Recipe\_Database.txt) for storing recipes persistently.
* Developed functions to read from the file during program initialization and write to the file upon updates:
  + load\_recipes(): Reads the file and splits recipe data with a separator to populate the in-memory recipe\_list.
  + save\_recipes(): Writes the current state of recipe\_list back to the file.
* Applied file-handling techniques with error handling to ensure smooth operation even if the file is missing or empty.

**Step 4: Designing the User Interaction**

* We used GUI, which is event-driven, to allow users to interact with the program until they choose to exit.
* Algorithm for Input Handling:
  + Prompt the user for actions (e.g., add a recipe, exit the program).
  + The interaction between user and application is manage through the button click of GUI that triggers a specific function of the button.
  + Handle invalid inputs gracefully by prompting the user again.

**Step 5: Managing Recipe Data**

* Stored recipes in an in-memory list for efficient processing during runtime.
* Added checks to prevent duplicate entries before appending a new recipe.
* The RecipeManager class is responsible to manage the in-memory list of recipes. It keeps this list always connected with the database text file by loading recipes when being called and saving any changes made immediately by user input. This keep the system running persistently through the user interaction.
* Algorithm for Adding Recipes:
  + Prompt the user for a recipe name.
  + Check if the name already exists in recipe list.
  + If not, append the name and update the file.

**Step 6: Optimizing the Code for Modularity and Readability**

* Modularized the code by separating tasks into functions:
  + main(): Orchestrates the overall program flow.
  + load\_recipes() and save\_recipes(): Handle data persistence.
* Using several classes is efficient to distribute and classify responsibilities. It also makes the code more readable. The Ingredient class deals with the details of each individual ingredient whereas Recipe class manages collections of ingredients combined with corresponding steps to make the recipe. The RecipeManager class deals with the database(saving, loading, editing). The RecipeApp class is in charge of the GUI. It ensures a smooth interaction between system and user through Tkinter.
* Used descriptive function and variable names for better readability.
* Organized the flow to allow for easy expansion, such as future GUI integration.

**Step 7: Testing and Debugging**

* Ran the program with various inputs, including edge cases, such as:
  + Adding duplicate recipes.
  + Running the program without the file present.
  + Exiting the loop with invalid or incomplete inputs.
  + Providing invalid inputs for the entry boxes.
* Debugged issues related to file handling, duplicate detection, and list management.

**Step 8: Finalizing and Documenting**

* Documented key components and functions for future reference or enhancement.
* Provided comments explaining the purpose and functionality of each class, method, and function.

**Methodologies and Algorithms Implemented**

1. **Object-Oriented Programming (OOP)**:
   * Encapsulation 🡪 Used private attributes with getters and setters for controlled access.
   * Composite Relation between Classes à Ingredient class is used in Recipe class without applying inheritance but aggregation. This allows components to be more simple while also being reusable.
2. **File I/O Algorithms**:
   * **File Reading** à The function reads in file and applies the split(‘---\n’) to separate the text into blocks to allow further recipe processing.
   * **File Writing** à Opened the file in write mode to save all recipes at the end of the session, overwriting the file with the updated list.
3. **Validation and Input Handling**:
   * Ensured user inputs were validated at each step to avoid program crashes or invalid data.
   * Implemented simple prompts and error-checking loops for robust interaction.
4. **Efficient Data Management**:
   * Used Python lists to store recipes in memory, which provided quick and easy operations like appending and searching.

**Chapter 3**

# Project Demonstration

## Screenshots and Code Snippets

Our personal recipe book gives a user-friendly experience with the tkinter-created GUI, shown in Figures 1 to 11 below. This allows users to search, add, and manage recipes easily and productively. Figure 1 shows the "Search Recipe" scene that allows users to enter either a recipe name or an ingredient name. If a match is found, all recipe windows that match that input will appear. Figure 2 depicts the case when a recipe name in the database has been searched, showing full information about the recipe name, required ingredients, and preparation steps. When users want to see what they make from ingredients found in their fridge, they can input ingredient names such as sugar, shown in Figure 3 in the search. The system will then show all recipes that contain sugar for the users to view. The next tab is the "Add Recipe" tab, shown in Figure 4, which allows users to add their own recipes to the database. Figure 5 demonstrates how a user will enter the details of a yogurt recipe to be written into the database, including title name, ingredient contents, and steps. For additional management of the recipes, the user can use the Manage Recipes tab, containing functions such as viewing all recipe titles, editing recipes, deleting recipes, and sorting recipes, as shown in Figure 6. Figure 7 shows how a user will be directed when they click the Edit Recipe button. He/she will first enter a recipe name to edit and then update all the details (title, ingredient contents, and steps). A confirmation prompt is shown in the last step of the edit recipe function to notify and ensure that the users have edits entered correctly. When the user wants to delete/remove a recipe from the database, they will use the delete recipe button shown in Figure 9. By entering the name of the recipe and confirming the deletion, the data will be removed completely from the database. The sort button illustrated in Figure 10 keeps the database organized by having all recipe titles sorted alphabetically. Finally, when the users want to view the entire recipe stored in the database, they can use the view all recipes button in Figure 11 to view all recipe titles through a scrollbar. Changes made, such as sorting or deleting a recipe, will be all reflected in this function as it reads the updated database. In general, the system gives sufficient features to achieve a user’s basic needs and experience of a personalized recipe book.

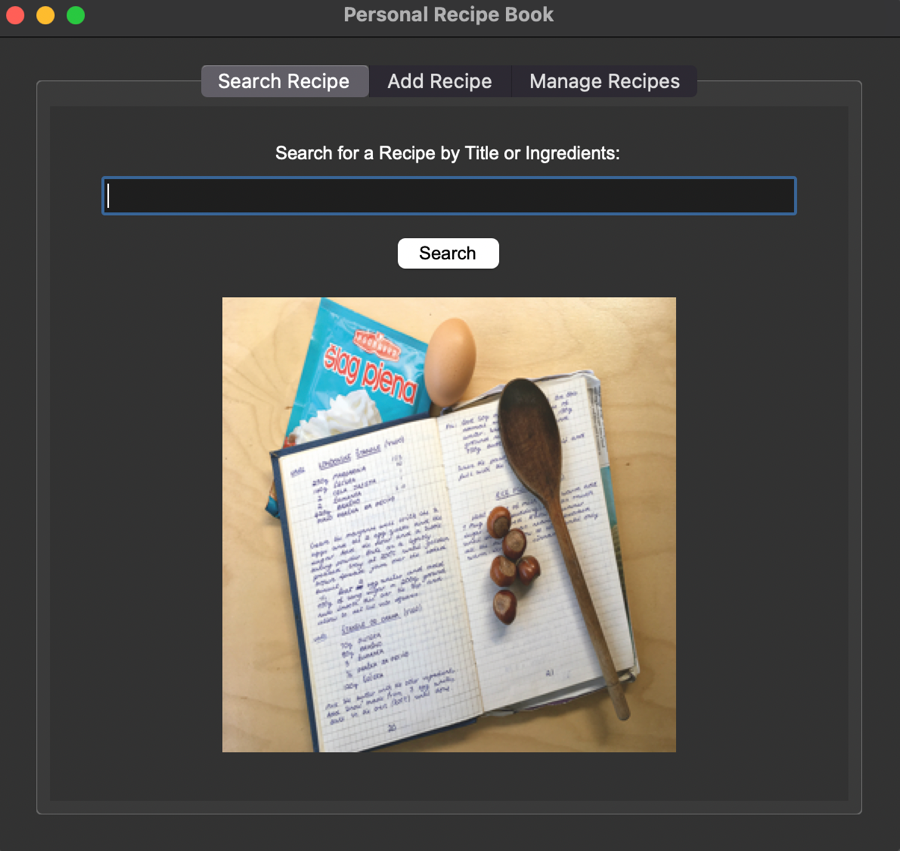


Figure 1. The search recipe tab

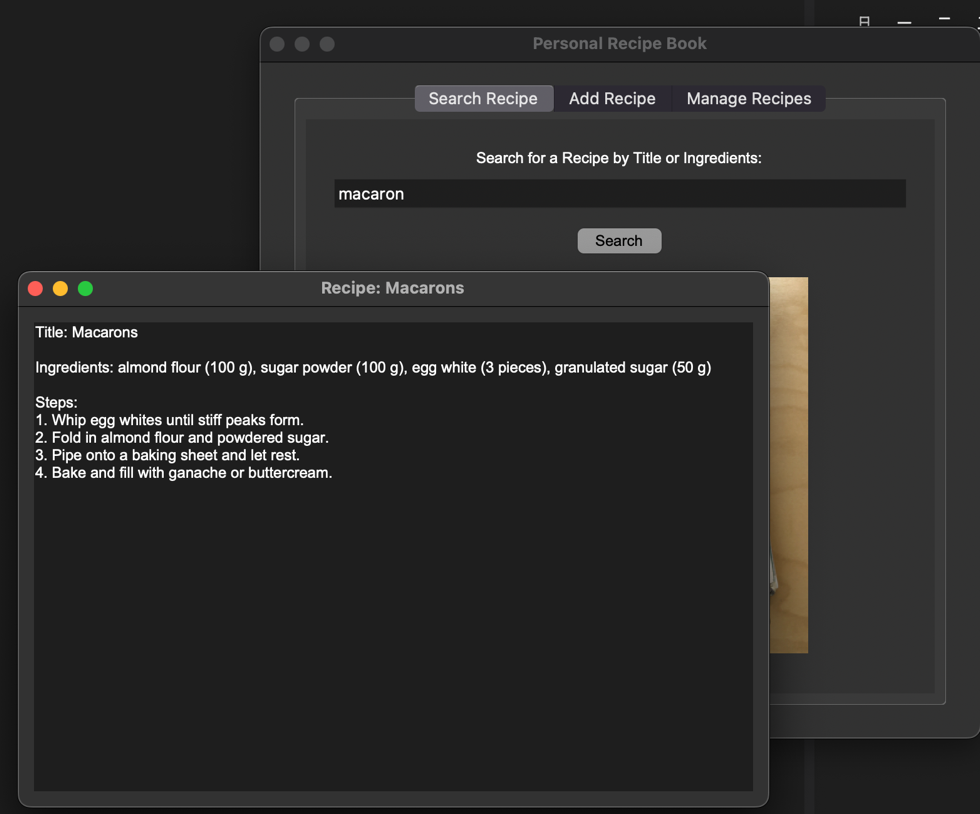


Figure 2. Recipe display window when users search for “macaron,” a recipe title

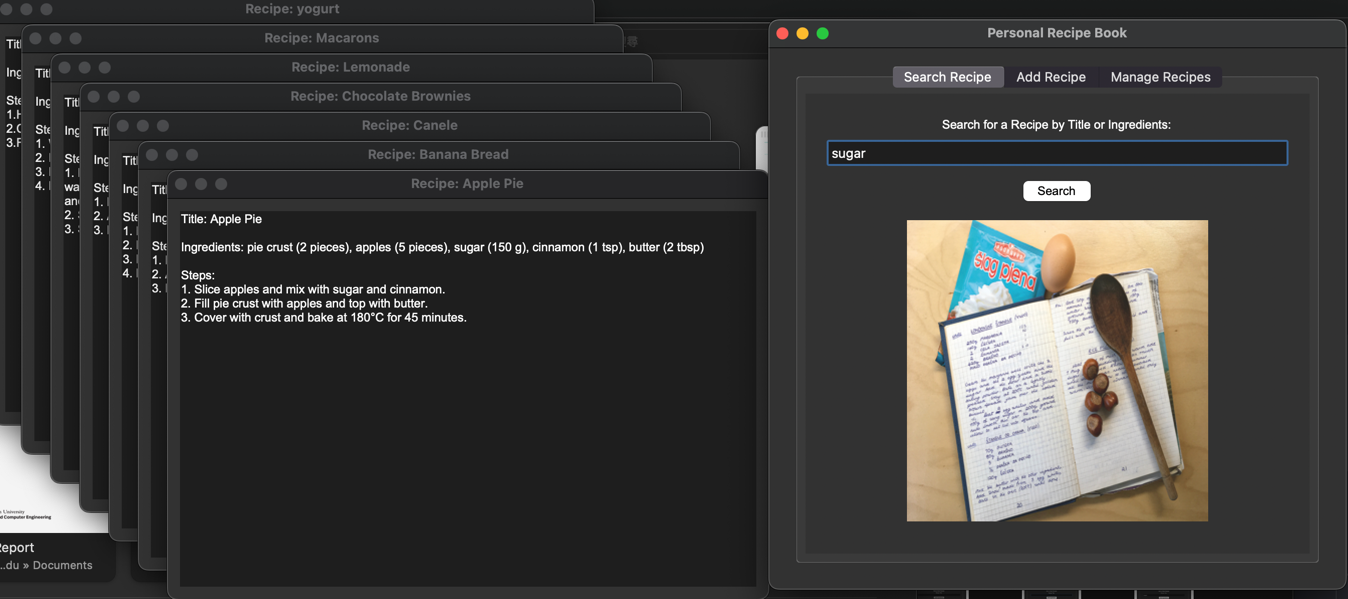


Figure 3. All recipes displayed when users search for “sugar,” an ingredient

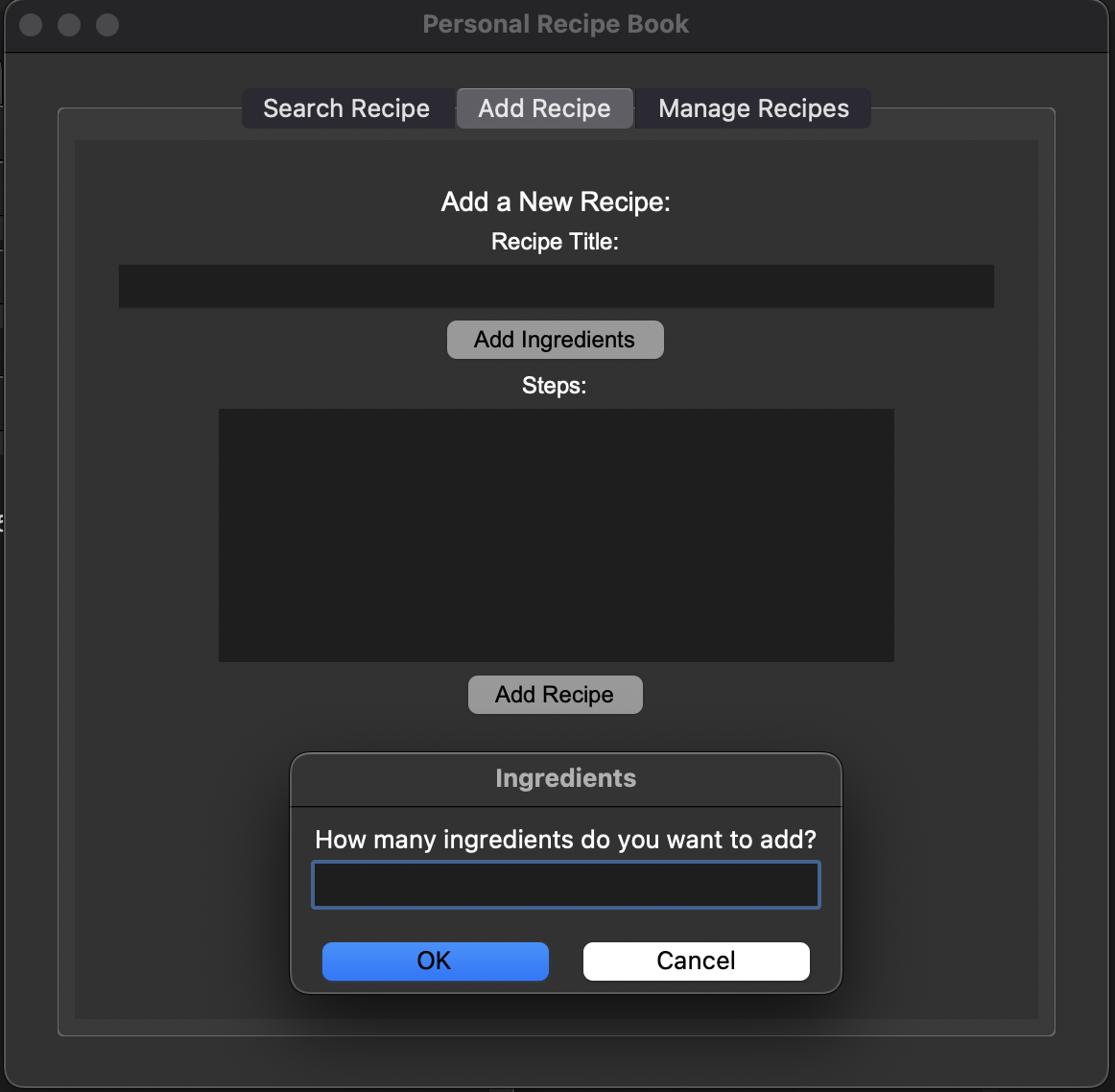


Figure 4. Add recipe tab

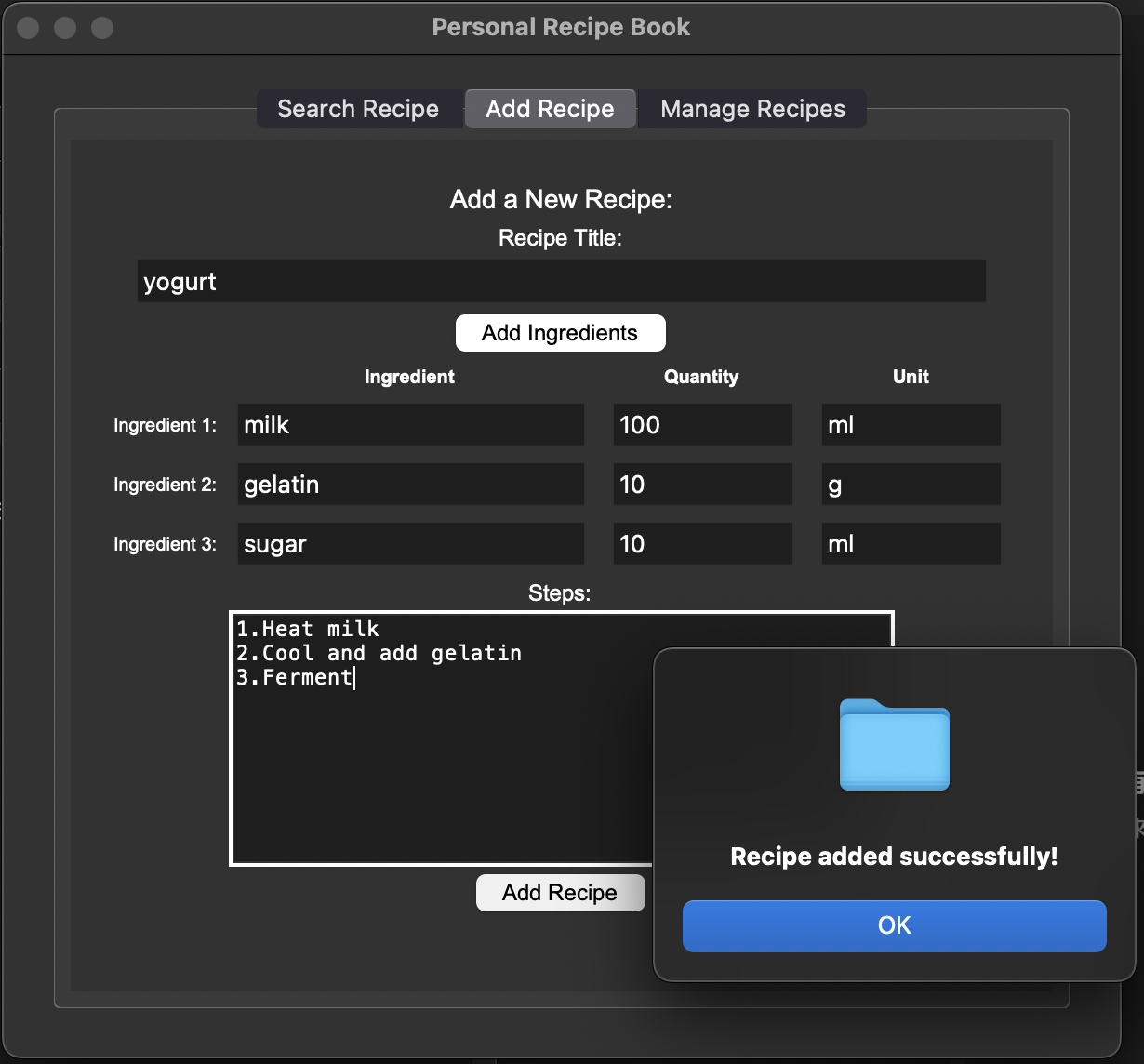


Figure 5. Demonstration of adding a recipe “yogurt”



Figure 6. Manage recipes tab with 4 functions

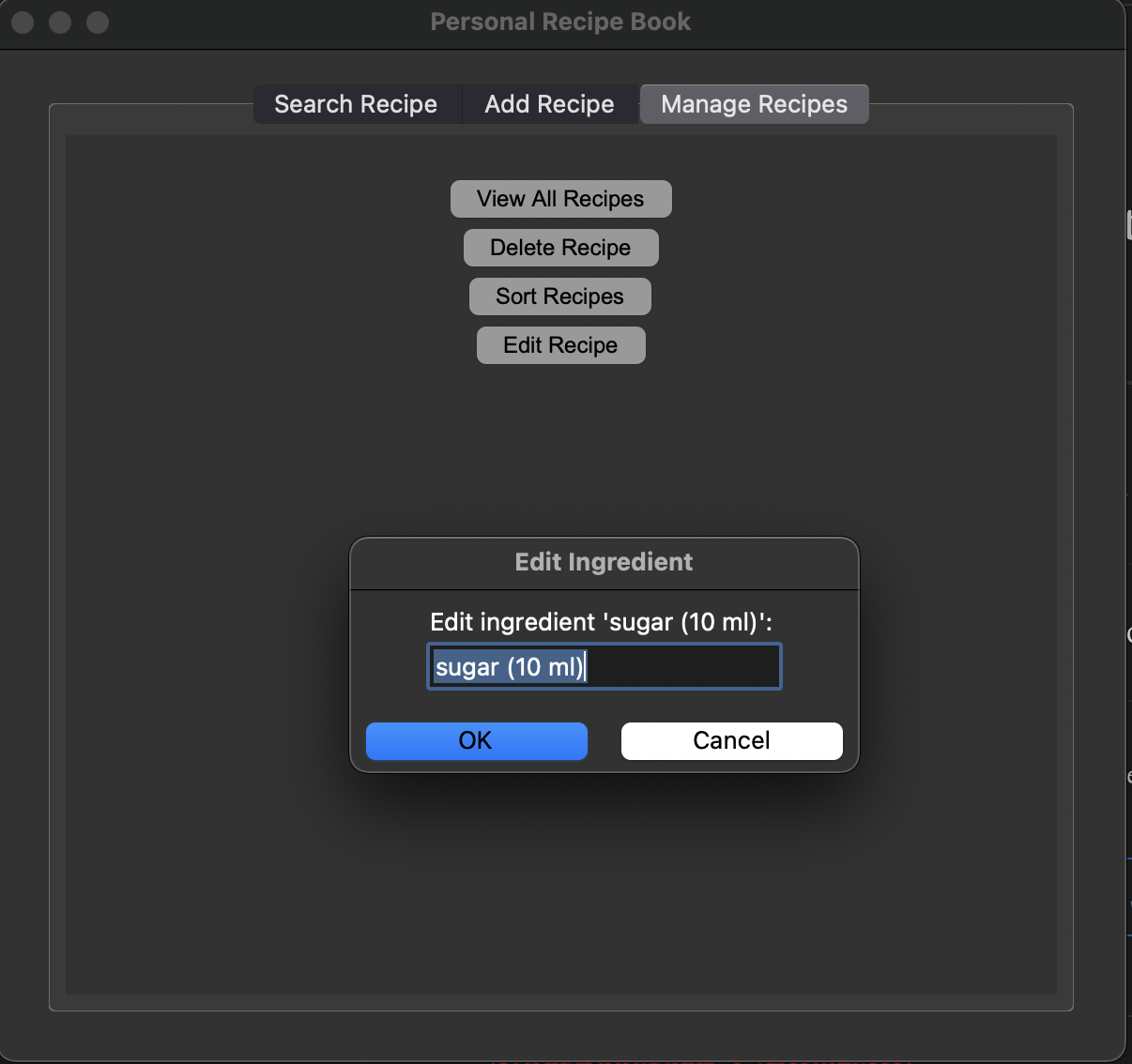


Figure 7. Edit recipe function demonstration

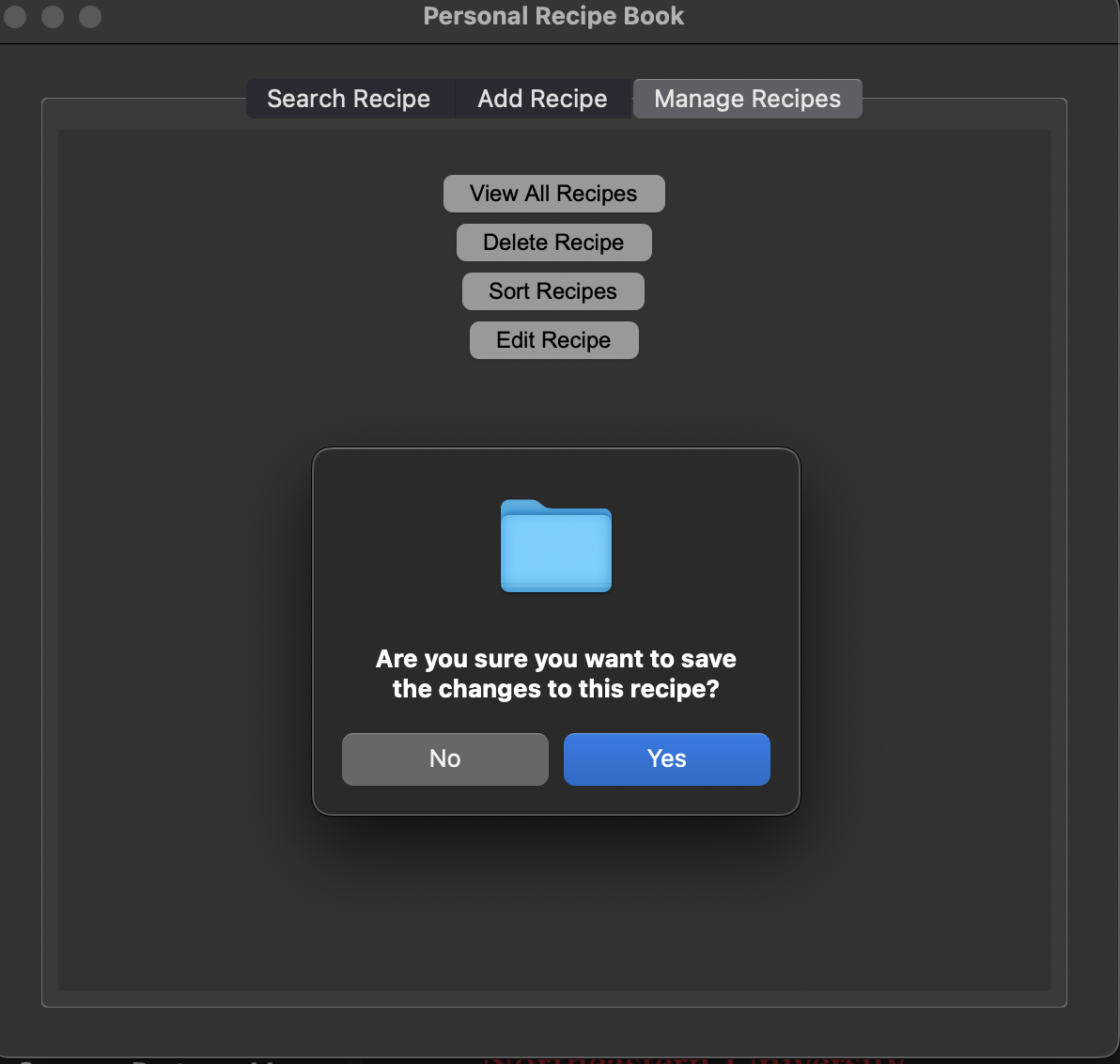


Figure 8. Double confirmation of edit recipe

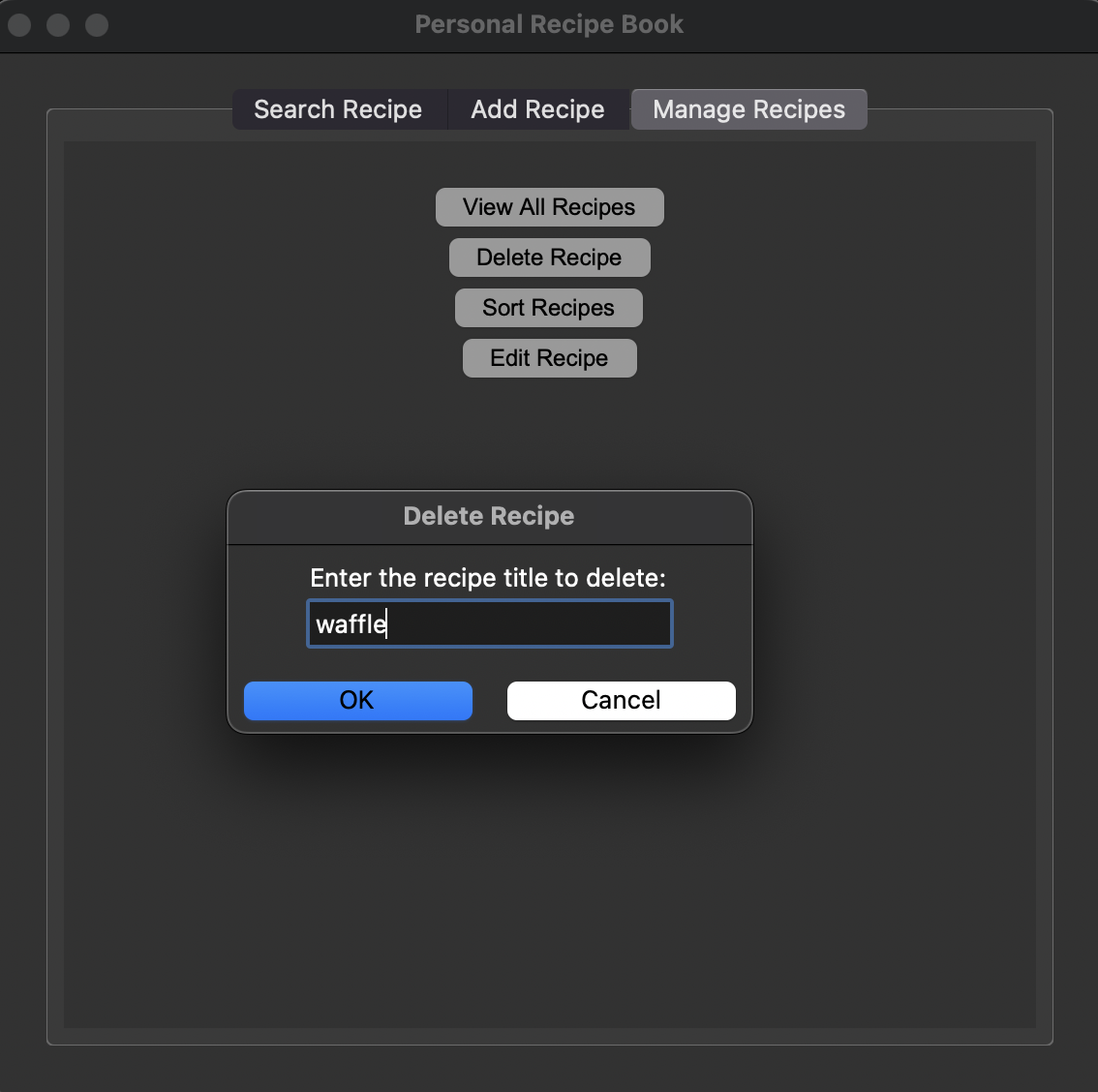


Figure 9. Delete recipe function demonstration



Figure 10. Sort recipe function demonstration

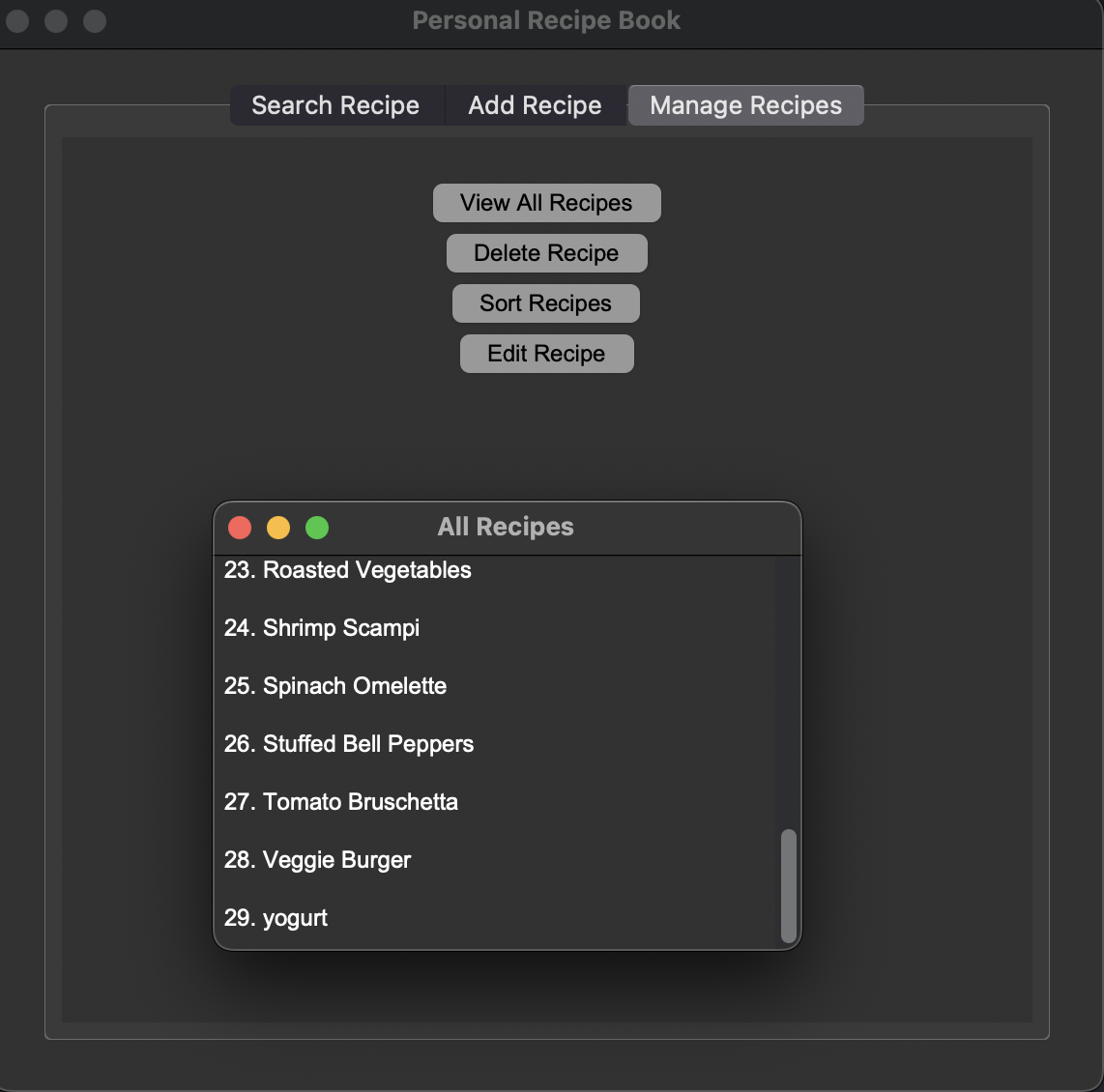


Figure 11. View all recipes function with data being sorted and waffle being deleted

**Chapter 4**

# Discussion and Future Work

The Recipe Book Application effectively addresses the need for managing and discovering recipes based on available ingredients, offering users a streamlined way to organize and modify their recipes. Built with Python, Tkinter, and Pillow, the application provides key features such as ingredient-based search and the ability to edit recipes, which enhances convenience for users looking to plan meals with what they have at home. However, while the project met its initial goals, several limitations emerged. The GUI remains simple due to time and knowledge constraints, limiting the complexity of interactivity and user experience. Additionally, manual recipe entry can be cumbersome, potentially deterring some users, and the image integration feature is not fully developed, which reduces the app’s visual appeal and user engagement. Social features, such as the ability to connect with friends and share recipes, were also not included in the current version, which limits the app’s potential as a community-driven platform. Despite these challenges, the application demonstrates a solid application of Object-Oriented Programming (OOP) principles and offers a valuable tool for home cooks and food enthusiasts.

To improve the user interface, we have made several recommendations. These include enhancing the GUI with dynamic updates, advanced search filters, and more interactive elements. Adding social features would encourage further user engagement, allowing users to share recipes and connect with others. The ability to upload recipes in PDF form would save users time and improve convenience, while advanced image management would allow users to add photos of their dishes, enhancing the app's functionality. Furthermore, integrating cloud synchronization would allow users to access their recipes across multiple devices, ensuring that their data is always available and secure. By addressing these limitations and incorporating these improvements, the Recipe Tracker Application has the potential to become a more comprehensive and user-friendly platform for culinary enthusiasts.

**Chapter 5**

# Conclusion

All the objectives outlined for the Recipe Tracker Application were successfully achieved, resulting in a functional and user-friendly software tool. We were able to create an application that allows users to efficiently enter their recipes, store the data within the system, and access it whenever needed. Users can edit and delete recipes, providing them with the flexibility to customize and manage their collections dynamically. The implementation of a graphical user interface (GUI) using Tkinter added an intuitive layer to the application, enhancing usability and making the software accessible to individuals with varying levels of technical expertise.

Through this project, we gained significant experience and a deeper understanding of Python programming, particularly in the context of GUI development. We explored foundational concepts of Object-Oriented Programming (OOP) and effectively utilized libraries like Pillow (PIL) for image handling. This project also provided hands-on experience in managing data structures such as lists, strings, and integers, which were crucial for the seamless operation of the application.

While the application met its objectives, it also highlighted areas for future improvement, such as expanding the GUI’s complexity, integrating additional features like PDF uploads and recipe image support, and enabling social sharing capabilities. These potential enhancements, combined with our accomplishments, underscore the project’s value as a stepping stone in software development. The experience gained through this project not only allowed us to deliver a functional tool for recipe management but also equipped us with skills and insights for tackling more advanced development challenges in the future.

# Bibliography

[1]

“Tkinter PhotoImage,” *Python Tutorial - Master Python Programming For Beginners from Scratch*, Oct. 23, 2021. https://www.pythontutorial.net/tkinter/tkinter-photoimage/ (accessed Nov. 21, 2024).

‌

[2]

“W3Schools.com,” *W3schools.com*, 2024. https://www.w3schools.com/python/python\_file\_write.asp (accessed Nov. 21, 2024).

[3]

“W3Schools.com,” *W3schools.com*, 2024. https://www.w3schools.com/python/python\_ref\_list.asp (accessed Nov. 18, 2024).

[4]

GeeksforGeeks. “Scrollable Frames in Tkinter.” *GeeksforGeeks*, 4 Mar. 2021, www.geeksforgeeks.org/scrollable-frames-in-tkinter/. (accessed Nov. 10, 2024).

‌

‌