Cedrik Julianne M. Ocampo

CS103 Finals Project (C++, C#, Python)

In creating my final project for Computer Programming 2, I intend to make a simple but effective program that being the ***Feedback/Review System For A Restaurant***.

I started off making the original code in C++ as it is my comfort pick in creating codes. I find C++ very understandable to use and I have the most experience in using, I just enjoy it overall when making codes.

In creating codes and viewing the outputs in the terminal, I like it to be orderly and neat when displaying. This is the whole code that I started with. It will be my basis for the other two codes that are in different languages.

C++ Code Breakdown

```
#include <iostream>
#include <vector>
#include <string>
#include <limits>

using namespace std;
```

Starting off simple with the header files, as you can see we have **<iostream>** for the input and output functions, **<vector>** so we can use dynamic arrays to basically store the reviews that are going to be inputted by the users, **<string>** handles the names and feedback, **<limits>** is used for the error handling.

```
struct Review {
    string name;
    string feedback;
    int rating;
};
```

Using struct can basically group together related data like the name, feedback and ratings inputted. It acts as a container of the data and makes it into one unit making it easier to understand.

```
void addReview(vector<Review>& reviews) {
    Review newReview;
    cout << "==================================\n";
    cout << "---------Leave a Review-----------\n\n";
    cout << "Enter your name: ";
    cin.ignore();
    getline(cin, newReview.name);
    cout << "Enter your feedback: ";
    getline(cin, newReview.feedback);

    while (true) {
        cout << "Enter your rating (1-5): ";
        cin >> newReview.rating;
        if (cin.fail() || newReview.rating < 1 || newReview.rating > 5) {
            cin.clear();
            cin.ignore(numeric_limits<streamsize>::max(), '\n');
            cout << "Invalid rating. Please enter a number between 1 and 5." <<
endl;
        }
        else {
            break;
        }
    }

    reviews.push_back(newReview);
}
```

This function acts as I would call a collector that it **gathers the user inputs to create a review** and adds it to the vector<Review>, It asks the user to enter their name and feedback then asks the rating from (1-5) using a loop so that it has an error handling on it so it wont bug out the program.

```
void displayReviews(const vector<Review>& reviews) {
    if (reviews.empty()) {
        cout << "No reviews yet." << endl;
        return;
    }

    double totalRating = 0;
    cout << "==================================\n";
    cout << "------------Reviews-------------\n\n";
    for (const auto& review : reviews) {
        cout << "Name: " << review.name << endl;
        cout << "Feedback: " << review.feedback << endl;
        cout << "Rating: " << review.rating << "/5" << endl;
        cout << "---------------------------------" << endl;
        totalRating += review.rating;
    }

    double averageRating = totalRating / reviews.size();
    cout << "Average Rating: " << averageRating << "/5" << endl;
}
```

This function just **displays the overall reviews** that were stored to the program, I also added an average rating of the reviews to get an overall rating. It also has a feature that for example if there are no reviews made yet it will display a notice.

```
int main() {
    vector<Review> reviews;
    int choice;

    do {
        cout << "==================================\n";
        cout << "--------Restaurant Review---------\n\n";
        cout << "1. Add Review" << endl;
        cout << "2. Display Reviews" << endl;
        cout << "3. Exit\n\n";
        cout << "--------(Input no.: 1 - 3)--------\n";
        cout << "Enter your choice: ";
        cin >> choice;

        if (cin.fail() || choice < 1 || choice > 3) {
            cin.clear(); // clear the error flag
            cin.ignore(numeric_limits<streamsize>::max(), '\n'); // discard invalid
input
            cout << "Invalid choice. Please enter a number between 1 and 3." <<
endl;
            continue;
        }

        switch (choice) {
        case 1:
            addReview(reviews);
            break;
        case 2:
            displayReviews(reviews);
            break;
        case 3:
            cout << "Exiting..." << endl;
            break;
        }
    } while (choice != 3);

    return 0;
}
```

I would call this the menu area of the code where it asks the user what they want to do, if they want to leave a review, see the reviews or just exit the program. Just like the other parts where the user has to enter an input, it has error handling if input is not the proper choice.

C# Code Breakdown

For C# it was pretty basic to see the counterparts of the original code, obvious ‘cause both are derived from C. The 2nd code needed a few tweaks here and there and eventually got it working just like the first program.

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

class Program
{
```

Much like in C++ the `using System` is like `<iostream>` this basically makes the input and output of the program work. Then `using System.Collections.Generic` allows the use of generic collections like `List<T>`.

```
struct Review
{
    public string Name;
    public string Feedback;
    public int Rating;
}
```

As I said the basis for my code is the C++ one, so this will also have a struct to group the data.

```
static void AddReview(List<Review> reviews)
{
    Console.WriteLine("==================================");
    Console.WriteLine("---------Leave a Review----------\n");
    Review newReview;

    Console.Write("Enter your name: ");
    newReview.Name = Console.ReadLine();

    Console.Write("Enter your feedback: ");
    newReview.Feedback = Console.ReadLine();

    while (true)
    {
        Console.Write("Enter your rating (1-5): ");
        if (int.TryParse(Console.ReadLine(), out newReview.Rating) &&
newReview.Rating >= 1 && newReview.Rating <= 5)
        {
            break;
        }
        else
        {
            Console.WriteLine("Invalid rating. Please enter a number between 1 and
5.");
        }
    }
    reviews.Add(newReview);
```

```
}
```

Collects the user inputs to create a new review.

```
static void DisplayReviews(List<Review> reviews)
{
    if (reviews.Count == 0)
    {
        Console.WriteLine("No reviews yet.");
        return;
    }

    double totalRating = 0;
    Console.WriteLine("==================================");
    Console.WriteLine("-------------Reviews-------------\n");
    foreach (var review in reviews)
    {
        Console.WriteLine($"Name: {review.Name}");
        Console.WriteLine($"Feedback: {review.Feedback}");
        Console.WriteLine($"Rating: {review.Rating}/5");
        Console.WriteLine("----------------------------------");
        totalRating += review.Rating;
    }

    double averageRating = totalRating / reviews.Count;
    Console.WriteLine($"Average Rating: {averageRating:F1}/5");
}
```

Displays all the reviews stored in the reviews list and calculates the average rating.

```
static void Main()
{
    List<Review> reviews = new List<Review>();
    int choice;

    do
    {
        Console.WriteLine("==================================");
        Console.WriteLine("--------Restaurant Review---------\n");
        Console.WriteLine("1. Add Review");
        Console.WriteLine("2. Display Reviews");
        Console.WriteLine("3. Exit\n");
        Console.WriteLine("--------(Input no.: 1 - 3)--------");
        Console.Write("Enter your choice: ");

        if (int.TryParse(Console.ReadLine(), out choice))
        {
            switch (choice)
            {
                case 1:
                    AddReview(reviews);
                    break;
                case 2:
                    DisplayReviews(reviews);
```

```
                    break;
                case 3:
                    Console.WriteLine("Exiting...");
                    break;
                default:
                    Console.WriteLine("Invalid choice. Please try again.");
                    break;
            }
        }
        else
        {
            Console.WriteLine("Invalid choice. Please enter a number between 1 and
3.");
        }
    } while (choice != 3);
}
```

main menu of the program, allowing users to interact with the review system through a menu.

Python Code Breakdown

When making the Python version, i kind of had a hard time in making it because it's a language that I am not familiar with or have experience outside of the activities made in class. I watched a few videos to understand some basic syntax of it, used the internet alot to help me construct it. I still had fun making this version even if I'm not used to it even if its a lot simpler in terms of the syntax.

```
class Review:
    def __init__(self, name, feedback, rating):
        self.name = name
        self.feedback = feedback
        self.rating = rating
```

Defines a Review class to encapsulate review data: name, feedback, and rating.
The __init__ method initializes the attributes when a Review object is created.

```
def add_review(reviews):
    print("==================================")
    print("---------Leave a Review-----------\n")
    name = input("Enter your name: ")
    feedback = input("Enter your feedback: ")

    while True:
        try:
            rating = int(input("Enter your rating (1-5): "))
            if 1 <= rating <= 5:
                break
            else:
                print("Invalid rating. Please enter a number between 1 and 5.")
```

```
        except ValueError:
            print("Invalid input. Please enter a valid number.")

    reviews.append(Review(name, feedback, rating))
```

Purpose:
Handles the collection and validation of a new review from the user.

Steps:
Prompts the user for their name and feedback.
Uses a while loop to validate the rating input:
Ensures the input is an integer between 1 and 5.
Provides user-friendly error messages for invalid input.
Adds the validated review to the reviews list as a Review object.

Why Use try-except?:
Handles non-numeric inputs for rating gracefully, preventing the program from crashing.

```
def display_reviews(reviews):
    if not reviews:
        print("No reviews yet.")
        return

    total_rating = 0
    print("==================================")
    print("-------------Reviews-------------\n")
    for review in reviews:
        print(f"Name: {review.name}")
        print(f"Feedback: {review.feedback}")
        print(f"Rating: {review.rating}/5")
        print("----------------------------------")
        total_rating += review.rating

    average_rating = total_rating / len(reviews)
    print(f"Average Rating: {average_rating:.1f}/5")
```

Purpose:
Displays all stored reviews and calculates the average rating.

Steps:
Checks if the reviews list is empty and exits early if no reviews are available.
Iterates over each Review object in the reviews list to display its attributes.
Computes and displays the average rating, formatted to one decimal place.

Key Features:
Uses Python string interpolation (f-strings) for clean and readable output.
Provides summary statistics (average rating) for better insights.

```
def main():
    reviews = []
    while True:
        print("==================================")
        print("--------Restaurant Review---------\n")
        print("1. Add Review")
        print("2. Display Reviews")
        print("3. Exit\n")
        print("--------(Input no.: 1 - 3)--------")
        try:
            choice = int(input("Enter your choice: "))
            if choice == 1:
                add_review(reviews)
            elif choice == 2:
                display_reviews(reviews)
            elif choice == 3:
                print("Exiting...")
                break
            else:
                print("Invalid choice. Please try again.")
        except ValueError:
            print("Invalid choice. Please enter a number between 1 and 3.")
```

Purpose:

Implements the main program logic, allowing users to navigate through the menu system.
Steps:
Displays a menu with three options: Add Review, Display Reviews, or Exit.
Uses try-except to validate user input, ensuring the choice is numeric.
Routes the user's choice to the corresponding function (add_review, display_reviews) or exits the program.
Why Use a Menu System?:

Provides a structured and user-friendly interface for interacting with the program.

```
if __name__ == "__main__":
    main()
```

Purpose:

Ensures that the main function runs only when the script is executed directly, not when imported as a module.