

CODELAB II

ASSESSMENT 2: Data Driven Application

**Lecturer : Ms.Lavanya Mohan**

|  |  |
| --- | --- |
| **Student’s Name** | Naomi Chellsea Espiritu |
| **Roll No** | 2022-211 |
| **Github Repository Name** | Assessment-2---Advance-Programming- |
| **Github Repository Link** | https://github.com/naomichellsea/Assessment-2---Advance-Programming- |
| **Repository Screen Shot** | Screenshot 2025-01-05 at 8.33.40 PM |

**Data Driven Application - Naomeal**

**Your data driven application must be accompanied by a Development Document of 1000 words. This development document should consist of the following elements (296)**

**GITHUB REPOSITORY :** https://github.com/naomichellsea/Assessment-2---Advance-Programming-

**MEALDB API :** https://www.themealdb.com/api.php

**Brief: A short description of your application including which API you have selected. You should also provide a link to your Github repository.**

Naomeal is an application designed to suggest users their meal when they are unsure about what to cook or eat.

The app uses (“https://www.themealdb.com/api.php”) to provide random meal suggestions, suggests meals based on given categories in the app, and allow users to search for meals based on ingredients they have on their house, fridge, kitchen etc.

My app is an interactive where users can explore various meals without thinking meal ideas which are is a common problems for humans.

**Project Plan: Project plan that breaks the project down into key milestones (e.g planning, design, development & testing) and tasks. This plan should allocate the estimated time to complete each of the listed tasks.**

1. Planning:

* Understand the MealDB API and plan how to integrate it into my application.
* Create a plan for the UI design and functionality, using a trial and error approach.

Estimated Time: 1 day

1. Design:

* Design the user interfrace wireframe.
* Integrate the functional components such as buttons, input field, cards on grid layout etc.

Estimated Time: 2 day

1. Development:

* Implement the core functions including random meal, meal categories, and search by ingredients.
* Check to ensure everything is working smoothly.

Estimated Time: 3 days

1. Testing:

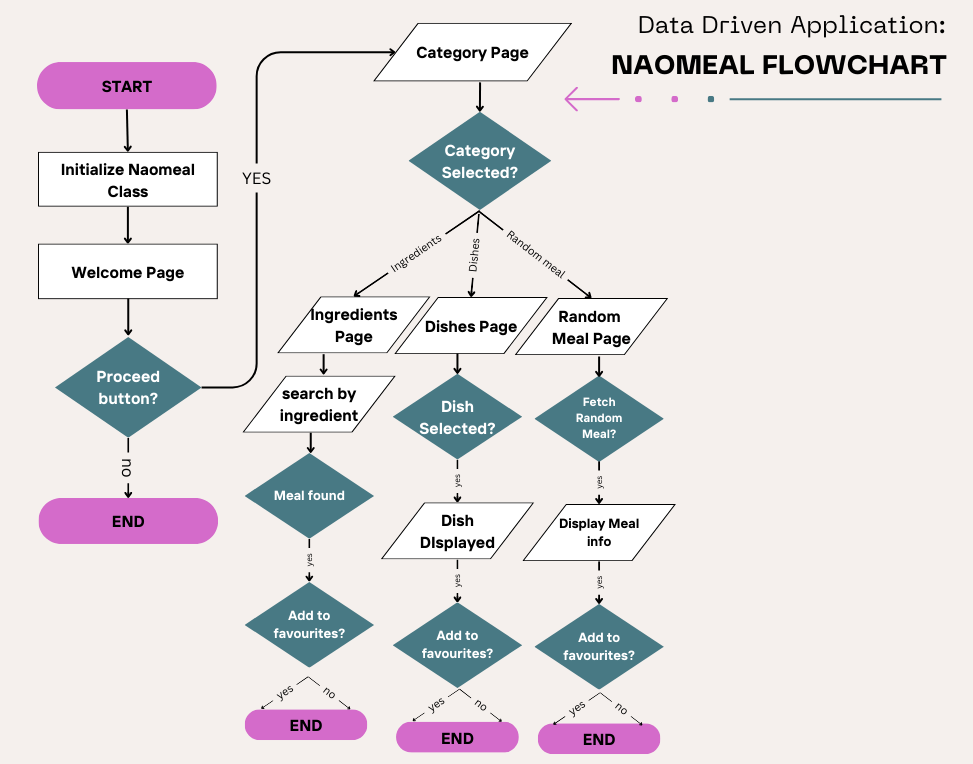
* Test the random meal feature, meal category selection, and ingredient search functionality. Essentially testing the final application.
* Identify and fix any errors, both failures and successes.

Estimated Time: 1 day

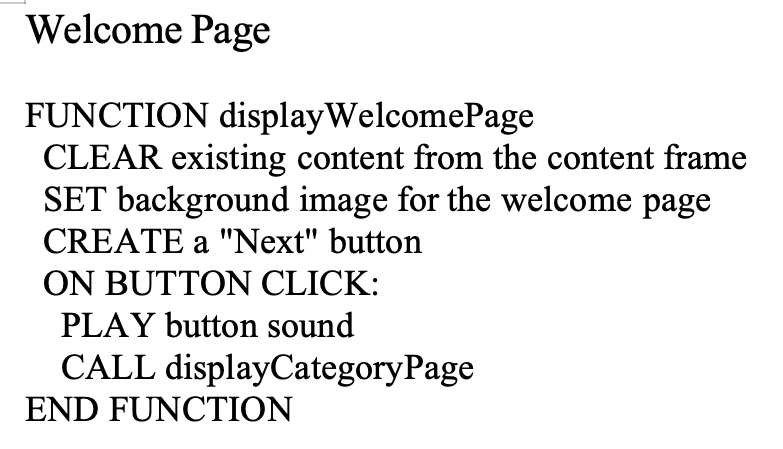
**Evidence of design: This may include but is not limited to: specification list of your program requirements, Flowchart, Pseudo-Code, Wireframes, UML Data Structure Diagram. A minimum expectation for this section is the inclusion of a wireframe for the design of the GUI.**

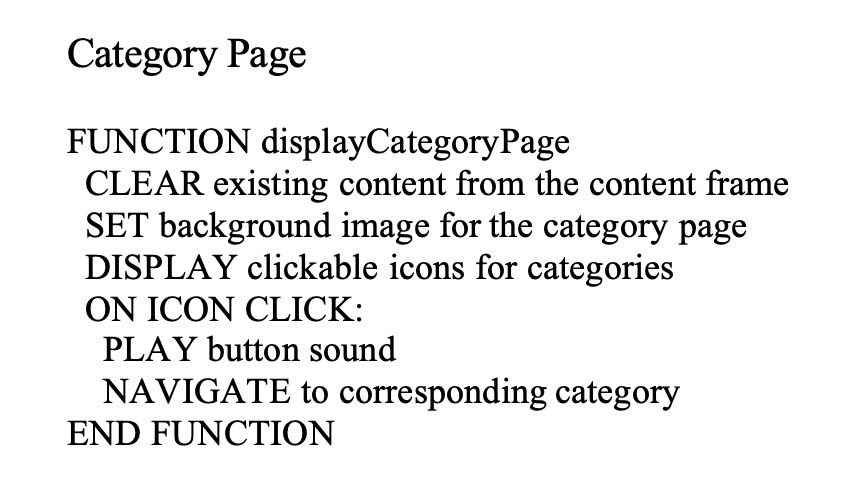
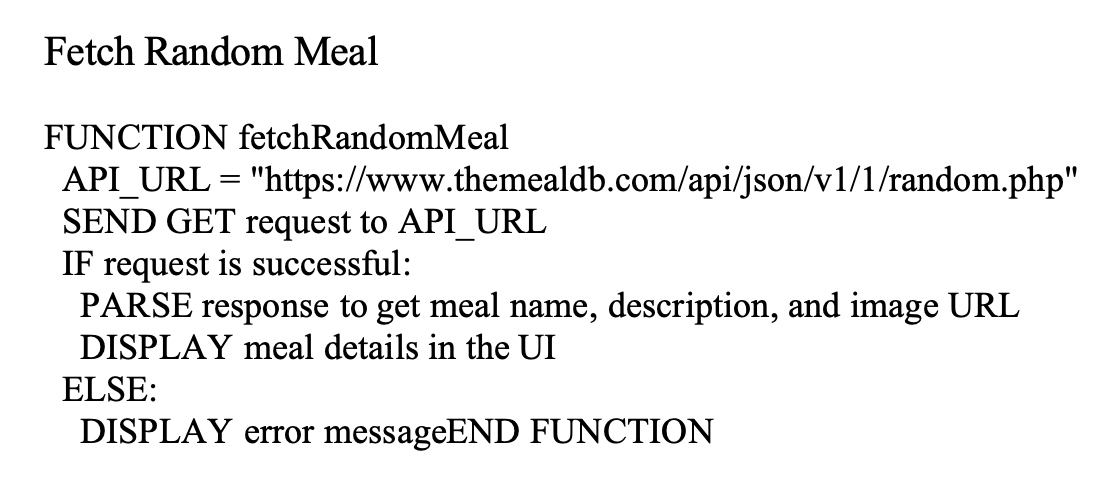
***Wireframe:***

***Flowchart:***

**

***Pseudo-Code Examples:***

******

******

**Technical Description & Walkthrough: In this section you should provide a link to a video which includes a walkthrough of your program running as well as a technical breakdown of your code.**

**This technical breakdown should explain how the key features of your program have been implemented via code. The video technical description & walkthrough contributes to the overall word count. The anticipated length of the video is 3-5 minutes.**

**Walkthrough Link:** <https://drive.google.com/drive/folders/1qxeUYpc49NCg-PR87TvqPUbkxoYEEA4D?usp=sharing>

**Technical Breakdown:**

My application of Naomeal used Tkinter library for GUI elements that are essentials for my app. Therefore, this is how the main window is set up:

self.root.title("Naomeal")

self.root.geometry("850x750")

self.bg\_color = "#ffffff" #this will initialize background color and theme

self.content\_frame = tk.Frame(self.root, bg=self.bg\_color)

self.content\_frame.pack(fill="both", expand=True)

1. Background Music Initialization

#function used to initialize and play background music

def bg\_music(self):

pygame.mixer.init()

music\_file = "/Users/naomichellsea/Advance Programming/Assessment 2 - Naomeal/Evidence of Design/resto jazz.mp3"

pygame.mixer.music.load(music\_file)

pygame.mixer.music.set\_volume(0.2) #can adjust the set volume (0.0 to 1.0)

pygame.mixer.music.play(-1) #will play music in a loop

The bg\_music() function uses pygame to play the sound of jazz track I have added to add ambiance to the app.

1. Welcome Page as the First Page

#First page for welcoming the users to my app

def welcomepage(self):

self.welcomepagebg() # This will set the background image for the page

startbutton = tk.Button(self.content\_frame, text="Next", command=self.categorypage,

font=("Helvetica", 16), bg="#6b0707", fg="#6b0707", relief="flat")

startbutton.place(relx=0.21, rely=0.868, anchor="center")

The welcomepage() function used the first screen with a background image that I also customized welcomepagebg() and a button leading to the main page of categories.

1. Adaptive Background Control

def welcomepagebg(self):

image\_path = "/Users/naomichellsea/Advance Programming/Assessment 2 - Naomeal/Evidence of Design/home.png"

try:

backimg = Image.open(image\_path)

backimg = backimg.resize((900, 750), Image.LANCZOS) #have to resize the image to fit the window

backimg = ImageTk.PhotoImage(backimg)

#adds the background label and place it in the content frame

bglabel.lower() #Lower the background label to the back of other widgets

Background images are dynamically added across different screens and resized using PIL. For example, the welcomepagebg() function handles the background for the first page.

I used bglabel.lower() to position the background behind the other widgets, ensuring nothing gets obstructed.

1. Category Navigation

def categoryicons(self):

#Creates three clickable image buttons in a zigzag layout.

icon\_paths = [

"/Users/naomichellsea/Advance Programming/Assessment 2 - Naomeal/Evidence of Design/1.png",

"/Users/naomichellsea/Advance Programming/Assessment 2 - Naomeal/Evidence of Design/2.png",

"/Users/naomichellsea/Advance Programming/Assessment 2 - Naomeal/Evidence of Design/3.png",

]

commands = [

self.randommeal,

self.dishespagetwo,

self.ingredientmeal,

]

button.bind("<Enter>", lambda e, btn=button, path=path: self.iconcategoryhover(btn, path, (560, 310)))

button.bind("<Leave>", lambda e, btn=button, path=path: self.iconcategoryhover(btn, path, (550, 300)))

The categoryicons() function displays three clickable images to navigate between pages. I used images as icons because the PNG icons didn’t work in my case, so I had to implement it this way.

It ended up being a success, and I also added a hover animation for a smoother user experience.

1. Random Meal Fetch

def randommealapi(self):

# etch a random meal from the API

try:

url = "https://www.themealdb.com/api/json/v1/1/random.php"

response = requests.get(url, timeout=10) #adds a timeout for better error handling

response.raise\_for\_status()

def resultsmeal(self, mealname, mealdescr, mealimg):

self.meal\_label = tk.Label(self.mealframe, text=f"{mealname}\n\n{mealdescr}",

The randommealapi() fetches the data from TheMealDB API to display the results that has been fetched with resultsmeal().

1. Category Based Dishes

def dishesapi(self, category):

url = f"https://www.themealdb.com/api/json/v1/1/filter.php?c={category}"

try:

response = requests.get(url) #get the request from the selected category in my app

meals = response.json().get('meals', []) #If no meals were return, empty list will go.

return meals

This feature allows the users to explore meals by category such as chicken, or beef then display them in a grid layout. get request retrieves the meal details and the method returns a list of meal.

1. Ingredient Search

def ingredientmealapi(self, ingredient):

api\_url = f"https://www.themealdb.com/api/json/v1/1/filter.php?i={ingredient}"

try:

response = requests.get(api\_url)

def displayresult(self, data):

for widget in self.resultsarea.winfo\_children():

widget.destroy()

This feature lets users enter their desired ingredient to find meals that match. The request.get() function sends a request to the API, filtering meals based on the given ingredient.

The results are displayed in a scrollable frame, self.resultsarea, allowing users to easily view multiple meals fetched from the API.

**Critical Reflection: An open and detailed evaluation of your application that notes what is compelling about the work, what could be improved, and what you need to learn to make these improvements.**

#### ****What Works Well****

My app, Naomeal, features an interactive design, which is key to providing a great user experience and keeping users engaged. The seamless API integration works perfectly with the app's features.

The design of the app, which utilizes third-party applications, complements the GUI content well. Hover effects, button sounds, and background music create an exciting and interesting ambiance for users.

The divided structure of my program makes my whole code clean and easy to manage. Additionally, error handling ensures a hassle-free experience for users.

I intentionally didn't include a back button because my app is designed for one-time use. If users want to access another feature, they'll need to return to the start screen.

#### ****What Could Be Improved****

The design is quite simple, and there’s room for improvement. For instance, enhancing accessibility by adjusting font styles, sizes, and button colors could make the app visually more appealing. Also, message box could be improve as well. API gave me a hard time in the making as it was very slow loading.

**Regarding my coding techniques, I’m still developing my skills in building GUI applications using Python, and there's certainly room for growth.**

#### ****What to learn next****

#### To improve the app and my coding experience, I should learn more about:

* Animations for more smoother and better looking.
* Better error handling, especially with buttons eg. in button colours as much as I try to change the colour, it is not implementing.
* Accessibility design for more interactive and aesthetically pleasing application.
* Advanced error handling.

**Appendix: A copy of your code should be included in an appendix at the end of your documentation. To provide this simply copy and paste the code from each of your project files (do not provide the code in screenshots).**

"""

Created on Tue Dec 17 00:27:21 2024

@author: naomichellsea

"""

"""

Assessment 2 - DDA

Naomeal

For this assignment you are tasked with developing an application that makes use of

data retrieved from an API. Your application should aim to demonstrate a range of programming

techniques introduced over the course of CodeLab I and CodeLab II, including use of functions

and where appropriate object oriented programming and GUI.

The final application should be delivered via a functioning interactive GUI built using Tkinter.

This GUI should allow the user to interact via mouse and/or keyboard input.

"""

import tkinter as tk

from tkinter import messagebox

import requests

from PIL import Image, ImageTk

import pygame

import io

import os

class Naomeal:

def \_\_init\_\_(self, root):

self.root = root

self.root.title("Naomeal")

self.root.geometry("850x750")

self.bg\_color = "#ffffff" #background colour for main gui page

self.card\_color = "#f7f7f7"

self.categoryicons() #background image for different pages

self.favorites = [] #empty list to store favourite meals

self.content\_frame = tk.Frame(self.root, bg=self.bg\_color) #add frame to hold content for smooth transitions

self.content\_frame.pack(fill="both", expand=True)

self.categorypagebg()

self.welcomepage()

self.welcomepagebg()

self.card\_color = "#ffffff"

self.meal\_label = None

self.bg\_music() #background music

#function to add background music

def bg\_music(self):

"""Initialize and play background music."""

pygame.mixer.init()

music\_file = "/Users/naomichellsea/Advance Programming/Assessment 2 - Naomeal/Evidence of Design/resto jazz.mp3"

pygame.mixer.music.load(music\_file)

pygame.mixer.music.set\_volume(0.2) #volume to control the music

pygame.mixer.music.play(-1) #play the music in loop

#First page for welcoming the users to my app

def welcomepage(self):

"""Sets the background image and creates the homepage with a 'Proceed' button."""

self.welcomepagebg() #background image for this page

#clear previous content to go to next page

for widget in self.content\_frame.winfo\_children():

widget.destroy()

#proceed button clickable to move to next page

startbutton = tk.Button(self.content\_frame, text="Next", command=self.categorypage,

font=("Helvetica", 16), bg="#6b0707", fg="#6b0707", relief="flat")

startbutton.bind("<Button-1>", lambda e, cmd=self.categorypage: [self.buttonsound(), cmd()])

startbutton.place(relx=0.21, rely=0.868, anchor="center")

def welcomepagebg(self):

"""Sets the background image for the current page."""

image\_path = "/Users/naomichellsea/Advance Programming/Assessment 2 - Naomeal/Evidence of Design/home.png"

try:

backimg = Image.open(image\_path)

backimg = backimg.resize((900, 750), Image.LANCZOS) #resize the image to fit the gui window

backimg = ImageTk.PhotoImage(backimg)

#background label to place it in frame

bglabel = tk.Label(self.content\_frame, image=backimg)

bglabel.place(relwidth=1, relheight=1) #ensures it covers the whole frame

#prevents garbage collection

self.content\_frame.bg\_image = backimg #store the image reference

bglabel.lower() #put the background to the back of other widgets so it overlap

except FileNotFoundError:

print(f"No Background Image {image\_path}")

#Second page for letting the users choose their desired catergory

def categorypage(self):

"""Directly navigates to the next page content."""

#Clear the current content

for widget in self.content\_frame.winfo\_children():

widget.destroy()

self.categorypagebg() #set the bg image for next page

self.categoryicons() #clickable image button for pages.

def categorypagebg(self):

"""Sets the background image within the content frame."""

image\_path = "/Users/naomichellsea/Advance Programming/Assessment 2 - Naomeal/Evidence of Design/main.png"

try:

bg\_image = Image.open(image\_path)

bg\_image = bg\_image.resize((900, 750), Image.LANCZOS) #resizing the image to fit the window

bg\_image = ImageTk.PhotoImage(bg\_image)

#create the background label and place it to content frame

background\_label = tk.Label(self.content\_frame, image=bg\_image)

background\_label.place(relwidth=1, relheight=1) #ensures image covers the entire frame

self.root.bg\_image = bg\_image #prevents garbage collection

background\_label.lower() #put the background to the back of other widgets so it overlap

except FileNotFoundError:

print(f"No Background Image {image\_path}")

def categoryicons(self):

#Creates three clickable image buttons in a zigzag layout.

icon\_paths = [

"/Users/naomichellsea/Advance Programming/Assessment 2 - Naomeal/Evidence of Design/1.png",

"/Users/naomichellsea/Advance Programming/Assessment 2 - Naomeal/Evidence of Design/2.png",

"/Users/naomichellsea/Advance Programming/Assessment 2 - Naomeal/Evidence of Design/3.png",

]

commands = [

self.randommeal,

self.dishespagetwo,

self.ingredientmeal,

]

positions = [(0.13, 0.45), (0.78, 0.62), (0.25, 0.97)]

for i, (path, command, pos) in enumerate(zip(icon\_paths, commands, positions)):

try:

icon = Image.open(path).resize((550, 300), Image.LANCZOS)

icon = ImageTk.PhotoImage(icon)

button = tk.Label(self.root, image=icon, cursor="hand2", bd=0, relief="flat")

button.image = icon #store image references

button.place(relx=pos[0], rely=pos[1], anchor="center")

button.bind("<Button-1>", lambda e, cmd=command: [self.buttonsound(), cmd()])

button.bind("<Enter>", lambda e, btn=button, path=path: self.iconcategoryhover(btn, path, (560, 310)))

button.bind("<Leave>", lambda e, btn=button, path=path: self.iconcategoryhover(btn, path, (550, 300)))

except FileNotFoundError:

print(f"No Icon Image {path}")

#Third page for letting the users explore random meal

def randommeal(self):

self.clearswidgets() #Clear previous widgets

image\_path = "/Users/naomichellsea/Advance Programming/Assessment 2 - Naomeal/Evidence of Design/wang.png" #Set background image

try:

bgimg = Image.open(image\_path)

bgimg = bgimg.resize((900, 750), Image.LANCZOS) #resizing the image

bgimg = ImageTk.PhotoImage(bgimg)

bglabel = tk.Label(self.root, image=bgimg)

bglabel.place(relwidth=1, relheight=1)

self.root.bgimg = bgimg #prevents garbage collection

bglabel.lower()

except FileNotFoundError:

print(f"No Background Image {image\_path}")

#Meal card frame, creates new frame everytime.

self.mealframe = tk.Frame(self.root, bg=self.card\_color, relief="raised", bd=3)

self.mealframe.place(relx=0.5, rely=0.53, anchor="center", width=700, height=500)

#random meal "search" button

searchbtn = tk.Button(self.mealframe, text="Make Me Yummy Meal!", font=("Helvetica", 12), bg="#6b0707", command=self.searchclicked)

searchbtn.pack(side="left", padx=20, pady=10)

#add to favourites button

self.favbtn = tk.Button(self.mealframe, text="My Favourite", font=("Helvetica", 12), bg="#6b0707", state="disabled", command=self.myfav)

self.favbtn.pack(side="right", padx=20, pady=10)

def searchclicked(self):

#called when user search random meal button is clicked

mealname, mealdescr, mealimg = self.randommealapi()

if mealname:

#update the meal card with the meal details from the API

self.resultsmeal(mealname, mealdescr, mealimg)

self.mealname = mealname #storing the meal name for later

self.favbtn.config(state="normal") #enable the add to favourties button from disable.

else:

messagebox.showerror("Failed to make you a meal ;< ")

def myfav(self, mealname):

#adds the current meal to favourites lists

if hasattr(self, "mealname") and self.mealname: #ensures meal name exits in the frame

if self.mealname not in self.favorites:

self.favorites.append(self.mealname)

messagebox.showinfo("Favorites", f"'{self.mealname}' added to favorites! ❤️ ")

else:

messagebox.showinfo("Favorites", f"'{self.mealname}' is already in your favorites.")

else:

messagebox.showerror("No meal has been selected ;/")

def resultsmeal(self, mealname, mealdescr, mealimg):

# Create a new meal label every time to avoid issues with destroying/recreating

if hasattr(self, 'meal\_label') and self.meal\_label: #creates new meal label everytime to avoid destroying

self.meal\_label.destroy() #destroy old label to create new one

self.meal\_label = tk.Label(self.mealframe, text=f"{mealname}\n\n{mealdescr}",

bg=self.card\_color, font=("Helvetica", 12), wraplength=480)

self.meal\_label.pack(pady=20)

try:

mealpic = Image.open(requests.get(mealimg, stream=True).raw)

mealpic = mealpic.resize((150, 150), Image.LANCZOS)

mealpic = ImageTk.PhotoImage(mealpic)

image\_label = tk.Label(self.mealframe, image=mealpic)

image\_label.image = mealpic #prevents garbage collection

image\_label.pack(side="top", pady=10)

except Exception as e:

print(f"Error loading image: {e}")

def randommealapi(self):

#fetch random meal from API

try:

url = "https://www.themealdb.com/api/json/v1/1/random.php"

response = requests.get(url, timeout=10) #timeout before error handling

response.raise\_for\_status() #it raise and error for HTTP issues

data = response.json()

#extract meak details from API

meal = data['meals'][0]

mealname = meal['strMeal']

mealdescr = meal['strInstructions']

mealimg = meal['strMealThumb']

return mealname, mealdescr, mealimg

except Exception as e:

print(f"Error fetching meal data: {e}")

return None, None, None

#4th page for user to choose their desired type of dishes.

def dishespagetwo(self):

self.clearswidgets()

image\_path = "/Users/naomichellsea/Advance Programming/Assessment 2 - Naomeal/Evidence of Design/samp.png" #bg image for this new frame

try:

bgimgtwo = Image.open(image\_path)

bgimgtwo = bgimgtwo.resize((900, 750), Image.LANCZOS) #LANCZOS instead of ANTIALIAS

bgimgtwo = ImageTk.PhotoImage(bgimgtwo)

background\_label = tk.Label(self.root, image=bgimgtwo)

background\_label.place(relwidth=1, relheight=1)

self.root.bgimgtwo = bgimgtwo #prevents garbage collection

background\_label.lower()

except FileNotFoundError:

print(f"No Background Image{image\_path}")

dishes = ["Beef", "Chicken", "Dessert", "Pasta", "Seafood", "Vegetarian"]

#this is the category cards in grid layout

cards = [] #storing card reference for later use

for i, category in enumerate(dishes):

card = tk.Frame(self.root, bg=self.card\_color, relief="raised", bd=2, highlightthickness=1)

card.place(relx=(0.2 + (i % 3) \* 0.3), rely=(0.2 + (i // 3) \* 0.3), width=200, height=200)

#rounded corners for shadow effect

card.configure(highlightbackground="#ccc", highlightcolor="#ccc", relief="flat")

card["borderwidth"] = 2 #visible border for the card

#load the PNG image

icon = "/Users/naomichellsea/Advance Programming/Assessment 2 - Naomeal/Evidence of Design/food.png" #replace with you image file path

icnimg = Image.open(icon)

icnimg = icnimg.resize((80, 80), Image.LANCZOS) #resize the image if needed

icnpic = ImageTk.PhotoImage(icnimg)

#creating label to display the image

iconplace = tk.Label(card, image=icnpic, bg=self.card\_color)

iconplace.image = icnpic

iconplace.pack(pady=10)

label = tk.Label(card, text=category, bg=self.card\_color, font=("Helvetica", 14, "bold"))

label.pack(side="top", pady=10)

#button with hover animation

btn = tk.Button(

card,

text="Explore",

bg="#A02334",

fg="#FFAD60",

font=("Helvetica", 12),

command=lambda cat=category: self.dishesclicked(cat)

)

btn.pack(side="bottom", pady=10)

cards.append(card) #save references for later use

def dishesclicked(self, category):

#handles the users click options on a dishesclicked.

meals = self.dishesapi(category) #fetch filtered meals for the selected category.

self.clearswidgets() #Clear the current content

self.dishesresultspg(meals, category) #display results in grid frame.

self.dishesani() #apply animations for new page

def dishesresultspg(self, meals, category):

rows = 3 #3 rows in the grid

columns = 3 #3 columns in the grid

dishescard = []

#meal cards and display in grid layout form

for i, meal in enumerate(meals):

row = i // columns

col = i % columns

card = tk.Frame(self.root, bg=self.card\_color, relief="raised", bd=2, highlightthickness=1) #creates the card of meal

card.grid(row=row, column=col, padx=10, pady=10)

card.configure(highlightbackground="#ccc", highlightcolor="#ccc", relief="flat") #rounded corners with shadow

card["borderwidth"] = 2 #visble border of the card

label = tk.Label(card, text=meal['strMeal'], bg=self.card\_color, font=("Helvetica", 14, "bold"))

label.pack(side="top", pady=10)

btn = tk.Button(card, text="Explore", bg="#6b0707", fg="white", font=("Helvetica", 12), command=lambda meal=meal: self.explorebutton(meal))

btn.bind("<Button-1>", lambda e, cmd=self.categorypage: [self.buttonsound(), cmd()])

btn.pack(side="bottom", pady=10)

dishescard.append(card) #save references for later use

def explorebutton(self, meal):

"""Handle when the user clicks to explore a meal."""

print(f"Exploring meal: {meal['strMeal']}")

#use the API to get the detailed meal data using the meal

meal\_id = meal.get('idMeal') #Extract the meal id

if not meal\_id:

print("Error: Meal ID not found!")

return

#get meal details using API meal ids

mealdetails = self.dishesapi(meal\_id)

if mealdetails:

print(f"Fetched ! {mealdetails}")

else:

#message box for the errors to notify

root = tk.Tk()

root.withdraw() #hiding the root window

messagebox.showerror("Error", "No Details Found")

def dishesapi(self, category):

url = f"https://www.themealdb.com/api/json/v1/1/filter.php?c={category}"

try:

response = requests.get(url)

response.raise\_for\_status()

meals = response.json().get('meals', [])

return meals

except requests.exceptions.RequestException as e:

print(f"Error fetching data: {e}")

self.show\_error\_message("Failed to load meals. Please try again later.")

return []

def dishesani(self, options):

"""Display options in a 5-column grid with slide-in animations."""

self.clearswidgets() #to clear the existing widgets in the exising window

container = tk.Frame(self.root, bg=self.bg\_color)

container.pack(fill="both", expand=True)

#display options in 5 column grid

for i, option in enumerate(options):

card = tk.Frame(container, bg=self.card\_color, relief="raised", bd=2)

card.grid(row=i // 5, column=i % 5, padx=3, pady=3, sticky="nsew")

#content to the card such as meal name

option\_name = option.get("strMeal", "Unknown")

label = tk.Label(card, text=option\_name, bg=self.card\_color, font=("Helvetica", 12, "bold"))

label.pack(pady=2)

#explore button that are clickable

explore\_button = tk.Button(card, text="Explore", command=lambda opt=option: self.explorebutton(opt))

explore\_button.pack(pady=2)

#slide in animation in the card

self.root.after(100 \* i, lambda widget=card: self.transitionsin(widget))

def transitionsin(self, widget, duration=500):

"""Animates a widget sliding in and ensures it stays in its final position."""

widget.update\_idletasks() #to update the widget geometry

x\_start = -200 #starting position

y\_start = widget.winfo\_y()

x\_end = widget.winfo\_x()

steps = 20 #the number of animation steps

dx = (x\_end - x\_start) / steps #distance to move per step

def slide\_step(step):

"""Move the widget one step closer to its final position."""

nonlocal x\_start

if step <= steps:

x\_start += dx

widget.place(x=int(x\_start), y=y\_start) #adjust widgets gradually

self.root.after(duration // steps, lambda: slide\_step(step + 1))

else:

widget.place(x=x\_end, y=y\_start) #ensures it stays at the final position because of animation

#to start the animation

slide\_step(0)

#5th page for user to enter their existing ingredients to make dishes.

def ingredientmeal(self):

self.clearswidgets()

image\_path = "/Users/naomichellsea/Advance Programming/Assessment 2 - Naomeal/Evidence of Design/festival.png" #sets another background image for this frame

try:

bgimgthree = Image.open(image\_path)

bgimgthree = bgimgthree.resize((900, 750), Image.LANCZOS) #resize the image if needed

bgimgthree = ImageTk.PhotoImage(bgimgthree)

bglabelthree = tk.Label(self.root, image=bgimgthree)

bglabelthree.place(relwidth=1, relheight=1)

self.root.bgimgthree = bgimgthree #prevents garbage collection

bglabelthree.lower()

except FileNotFoundError:

print(f"No Background Image {image\_path}")

#search bar user can use

entersearch = tk.Label(self.root, text="Enter Ingredient:", font=("Helvetica", 16), bg=self.bg\_color)

entersearch.place(relx=0.2, rely=0.2, anchor="center")

ingredentry = tk.Entry(self.root, font=("Helvetica", 14))

ingredentry.place(relx=0.5, rely=0.2, width=300, anchor="center")

searchbtn = tk.Button(self.root, text="Search", font=("Helvetica", 12), command=lambda: self.ingredientmealapi(ingredentry.get()))

searchbtn.bind("<Button-1>", lambda e, cmd=self.categorypage: [self.buttonsound(), cmd()])

searchbtn.place(relx=0.8, rely=0.2, anchor="center")

#results frame area

self.resultsarea = tk.Frame(self.root, bg=self.bg\_color)

self.resultsarea.place(relx=0.5, rely=0.5, anchor="center", width=600, height=400)

#canvas for the results frame

frame = tk.Canvas(self.resultsarea, bg=self.bg\_color)

frame.pack(side=tk.LEFT, fill=tk.BOTH, expand=True)

#scroll bar for the results frame

scrollbar = tk.Scrollbar(self.resultsarea, orient=tk.VERTICAL, command=frame.yview)

scrollbar.pack(side=tk.RIGHT, fill=tk.Y)

#configure the canvas with scrollbar

frame.configure(yscrollcommand=scrollbar.set)

#bind canvas with scrollbar

frame.bind("<Configure>", lambda e: frame.configure(scrollregion=frame.bbox("all")))

def displayresult(self, data):

"""Display results in the results frame."""

for widget in self.resultsarea.winfo\_children(): #clear orevious resukts in frane

widget.destroy()

meals = data.get("meals", []) #check if there is a meal

if not meals:

no\_results\_label = tk.Label(self.resultsarea, text="No results found", font=("Helvetica", 14), bg=self.bg\_color)

no\_results\_label.pack(pady=10)

else:

for idx, item in enumerate(meals): #Display the meal recipe

recipe\_name = item.get("strMeal", "No name")

recipe\_image\_url = item.get("strMealThumb", "")

#show the recipe name

recipe\_label = tk.Label(self.resultsarea, text=recipe\_name, font=("Helvetica", 14), bg=self.bg\_color)

recipe\_label.grid(row=idx, column=0, pady=5, padx=10, sticky="w")

if recipe\_image\_url:

try:

img\_response = requests.get(recipe\_image\_url)

img\_response.raise\_for\_status() #to make sure that request is successful

img\_data = io.BytesIO(img\_response.content) #it converts bytes into a file-like object

img = Image.open(img\_data)

img = img.resize((100, 100), Image.LANCZOS)

img = ImageTk.PhotoImage(img)

img\_label = tk.Label(self.resultsarea, image=img, bg=self.bg\_color)

img\_label.image = img

img\_label.grid(row=idx, column=1, pady=5, padx=10)

except requests.exceptions.RequestException as e:

print(f"Error image fetch {e}")

except Exception as e:

print(f"Error image loading {e}")

#example onyl to d isplay each recipe

for idx, item in enumerate(data.get("recipes", [])):

recipe\_name = item.get("name", "No name")

recipe\_label = tk.Label(self.resultsarea, text=recipe\_name, font=("Helvetica", 14), bg=self.bg\_color)

recipe\_label.grid(row=idx, column=0, pady=5, padx=10, sticky="w")

def ingredientmealapi(self, ingredient):

api\_url = f"https://www.themealdb.com/api/json/v1/1/filter.php?i={ingredient}"

try:

response = requests.get(api\_url)

if response.status\_code == 200:

data = response.json()

self.displayresult(data)

else:

print(f"Error API fetch {response.status\_code}")

except requests.exceptions.RequestException as e:

print(f"Error API request{e}")

#end of function pages

#general functions

def clearswidgets(self):

for widget in self.root.winfo\_children(): #clears all the widgets from the screen

widget.destroy()

def buttonsound(self):

"""Plays a sound effect when a button is clicked."""

btnsoundpath = "/Users/naomichellsea/Advance Programming/Assessment 2 - Naomeal/Evidence of Design/click-21156.mp3"

try:

pygame.mixer.init()

pygame.mixer.Sound(btnsoundpath).play()

except Exception as e:

print(f"Error playing sound: {e}")

def iconcategoryhover(self, button, path, size):

"""Handles hover effect for buttons."""

try:

icon = Image.open(path).resize(size, Image.LANCZOS)

button.image = ImageTk.PhotoImage(icon)

button.configure(image=button.image)

except FileNotFoundError:

print(f"Hover icon not found at {path}")

if \_\_name\_\_ == "\_\_main\_\_":

root = tk.Tk()

app = Naomeal(root)

root.mainloop() #to start the tkinter with all the functions.