### Internship Project Report

#### Project Title: Development of a Food Ordering Application

#### Prepared By:

Ms.Aziza Azim Khalfe

#### Internship Position:

Web Development Intern

#### Company/Organization:

Unified Mentor

### Abstract

This report documents the development of a food ordering application as part of my internship project. The primary objective of this project was to design and implement a user-friendly application using Python's Tkinter library, integrated with functionalities for ordering food, viewing the cart, and processing orders. The following sections outline the project objectives, technologies used, detailed implementation, results, and conclusions drawn from the project.

### Table of Contents

1. Introduction
2. Project Objectives
3. Technologies Used
4. Implementation
   1. Python Tkinter for GUI
   2. Image Handling with PIL
   3. Email and SMS Integration
5. Results and Discussion
6. Conclusion
7. References
8. Appendices

### Introduction

The introduction provides an overview of the project's purpose, scope, and significance, setting the context for the detailed sections that follow. The food ordering application serves as a practical application to enhance my understanding and proficiency in GUI development and integration of various functionalities.

### Project Objectives

1. Develop a user-friendly graphical interface using Tkinter.
2. Implement functionalities to view and select food items.
3. Allow users to filter food items by vegetarian and non-vegetarian options.
4. Enable users to add items to the cart with special requests.
5. Provide functionalities to view, edit, and remove items from the cart.
6. Integrate email and SMS capabilities for order summaries.

### Technologies Used

* **Python Tkinter**: For creating the graphical user interface.
* **PIL (Pillow)**: For handling and displaying images.
* **smtplib and email.mime**: For sending order summaries via email.
* **IntVar**: For tracking the state of checkboxes and radio buttons.

### Implementation

#### Python Tkinter for GUI

The main window of the application was created using Tkinter. The application allows users to view food items, select items based on their preferences (vegetarian or non-vegetarian), and add them to a cart.

#### Image Handling with PIL

The images of food items were handled using the PIL library. The images were resized and displayed dynamically based on user selections.

#### Email and SMS Integration

The application allows users to enter their email or phone number to receive an order summary. The email functionality was implemented using Python's smtplib and email.mime libraries. For SMS functionality, a placeholder for API integration is provided.

### Results and Discussion

The food ordering application successfully integrates various functionalities:

* Users can view and filter food items.
* Adding items to the cart with special requests is supported.
* Viewing, editing, and removing items from the cart is possible.
* Order summaries can be sent via email and SMS (placeholder for SMS API).

The project demonstrates effective use of Tkinter for GUI development and integrates additional libraries for enhanced functionality.

### Conclusion

This project successfully demonstrates the integration of Tkinter, PIL, and email functionalities to create a comprehensive food ordering application. The application is user-friendly and provides a practical solution for managing food orders, enhancing my skills in GUI development and integration.

### References

1. [Tkinter Documentation](https://docs.python.org/3/library/tkinter.html" \t "_new)
2. [Pillow (PIL Fork) Documentation](https://pillow.readthedocs.io/" \t "_new)
3. [smtplib Documentation](https://docs.python.org/3/library/smtplib.html" \t "_new)
4. [email.mime Documentation](https://docs.python.org/3/library/email.mime.html" \t "_new)

### Appendices

* **Appendix A**: Source Code

#### Source Code

import tkinter as tk

from tkinter import ttk, simpledialog, messagebox, IntVar

from PIL import ImageTk, Image

import smtplib

from email.mime.text import MIMEText

# Global cart list to store selected items

cart = []

cart\_window = None

# Dictionary to store food item prices

food\_prices = { "Burger": 5.99, "Pizza": 8.99, "Fries": 2.49, "Salad": 6.99, "Biryani": 10.99, "Pasta": 7.49,

"Sandwich": 4.99, "Sushi": 9.99, "Steak": 12.99}

# Create the main window

root = tk.Tk()

root.title("Food Ordering App")

# Function to update the displayed images based on the radio button selection

def update\_images():

selection = radio\_var.get()

# Clear existing images from the frame

for widget in food\_frame.winfo\_children():

widget.destroy()

filtered\_food\_items = []

if selection == 1: # Display vegetarian only

for food in food\_items:

if food["vegetarian"]:

filtered\_food\_items.append(food)

elif selection == 2: # Display non-vegetarian only

for food in food\_items:

if not food["vegetarian"]:

filtered\_food\_items.append(food)

else: # Display both

filtered\_food\_items = food\_items

display\_images(filtered\_food\_items)

# Function to display images

def display\_images(image\_list):

# Load and display images

for i, food in enumerate(image\_list):

image = Image.open(food["image\_path"])

image = image.resize((150, 100), Image.BILINEAR)

photo = ImageTk.PhotoImage(image)

label = tk.Label(food\_frame, image=photo)

label.image = photo

label.grid(row=i // 3, column=i % 3, padx=10)

# Bind the image label to a function that shows the details popup

label.bind("<Button-1>", lambda event, food=food: show\_details\_popup(food))

# Function to show details popup

def show\_details\_popup(food):

popup = tk.Toplevel()

popup.title("Food Details")

# Load and display image

image = Image.open(food["image\_path"])

image = image.resize((250, 200), Image.BILINEAR)

photo = ImageTk.PhotoImage(image)

label = tk.Label(popup, image=photo)

label.image = photo

label.pack()

# Display food name

name\_label = tk.Label(popup, text=food["name"], font=("Arial", 16, "bold"))

name\_label.pack(pady=10)

# Entry for quantity

quantity\_label = tk.Label(popup, text="Quantity:")

quantity\_label.pack()

quantity\_entry = tk.Entry(popup)

quantity\_entry.pack()

# Checkboxes for special requests

special\_requests\_label = tk.Label(popup, text="Special Requests:")

special\_requests\_label.pack()

# IntVars to track the state of checkboxes

extra\_cheese\_var = IntVar()

no\_onions\_var = IntVar()

extra\_sauce\_var = IntVar()

extra\_cheese\_checkbox = tk.Checkbutton(popup, text="Extra Cheese",variable=extra\_cheese\_var)

extra\_cheese\_checkbox.pack()

no\_onions\_checkbox = tk.Checkbutton(popup,text="No Onions",variable=no\_onions\_var)

no\_onions\_checkbox.pack()

extra\_sauce\_checkbox = tk.Checkbutton(popup, text="Extra Sauce",variable=extra\_sauce\_var)

extra\_sauce\_checkbox.pack()

# Add to cart button

add\_to\_cart\_button = tk.Button(popup,text="Add to Cart",command=lambda: add\_to\_cart(food, quantity\_entry.get(), extra\_cheese\_var.get(), no\_onions\_var.get(), extra\_sauce\_var.get()))

add\_to\_cart\_button.pack(pady=10)

# Close button

close\_button = tk.Button(popup, text="Close", command=popup.destroy)

close\_button.pack(pady=10)

# Function to add item to cart

def add\_to\_cart(food, quantity, extra\_cheese, no\_onions, extra\_sauce):

if quantity.isdigit() and int(quantity) > 0:

special\_requests = []

if extra\_cheese:

special\_requests.append("Extra Cheese")

if no\_onions:

special\_requests.append("No Onions")

if extra\_sauce:

special\_requests.append("Extra Sauce")

cart.append({

"name": food["name"],

"quantity": int(quantity),

"price": food\_prices[food["name"]],

"special\_requests": special\_requests

})

print( f"Added {quantity} {food['name']} to cart with special requests: {special\_requests}")

else:

print("Invalid quantity. Please enter a positive integer.")

# Function to view cart

def view\_cart():

global cart\_window

cart\_window = tk.Toplevel()

cart\_window.title("View Cart")

# Create a Treeview widget for the cart items

tree = ttk.Treeview(cart\_window,columns=('Name', 'Quantity', 'Price', 'Total Price'))

tree.heading('#0', text='ID')

tree.heading('Name', text='Name')

tree.heading('Quantity', text='Quantity')

tree.heading('Price', text='Price')

tree.heading('Total Price', text='Total Price')

if not cart:

# If cart is empty, display a message

no\_items\_label = tk.Label(cart\_window, text="Aren't you hungry? Add items quickly")

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

else:

# Display cart items

for i, item in enumerate(cart):

tree.insert('', 'end', text=i + 1,values=(item['name'], item['quantity'], item['price'],item['quantity'] \* item['price']))

# Add buttons to edit quantity, add, and remove items

edit\_button = tk.Button(cart\_window,text="Edit Quantity",command=lambda: edit\_quantity(tree))

edit\_button.pack(pady=5)

add\_button = tk.Button(cart\_window,text="Add Item",command=lambda: add\_item(tree))

add\_button.pack(pady=5)

remove\_button = tk.Button(cart\_window,text="Remove Item",command=lambda: remove\_item(tree))

remove\_button.pack(pady=5)

# Add checkout button

checkout\_button = tk.Button(cart\_window,text="Checkout",command=checkout,bg="black",fg="white")

checkout\_button.pack(pady=5)

tree.pack(expand=True, fill='both')

# Function to edit quantity of selected item

def edit\_quantity(tree):

selection = tree.selection()

if selection:

item\_id = tree.selection()[0]

try:

item\_index = int(tree.item(item\_id, 'text')) - 1

quantity\_input = simpledialog.askstring(

"Edit Quantity",

f"Enter new quantity for item {item\_id}: ",

parent=root)

if quantity\_input is not None and quantity\_input.isdigit() and int(

quantity\_input) > 0:

quantity = int(quantity\_input)

cart[item\_index]['quantity'] = quantity

tree.item(item\_id,

values=(cart[item\_index]['name'], quantity,

cart[item\_index]['price'],

quantity \* cart[item\_index]['price']))

else:

messagebox.showerror(

"Error", "Invalid quantity. Please enter a positive integer.")

except ValueError:

messagebox.showerror("Error", "Invalid item ID.")

# Function to add a new item to the cart

def add\_item(tree):

global cart, cart\_window

root.lift()

cart\_window.destroy()

# Function to remove selected item from the cart

def remove\_item(tree):

selection = tree.selection()

if selection:

item\_id = selection[0]

try:

item\_index = int(tree.index(item\_id)) - 1

del cart[item\_index]

tree.delete(item\_id)

except ValueError:

messagebox.showerror("Error", "Invalid item ID.")

global email\_entry, email\_popup

# Function to calculate total cost and generate summary

def checkout():

global email\_entry, email\_popup

# Create a popup window for email number entry

email\_popup = tk.Toplevel()

email\_popup.title("Enter Email Id or Phone Number")

email\_popup.geometry("300x250")

# Label and Entry for email number

email\_label = tk.Label(email\_popup,text="Enter your Email Id or Phone Number : ",font=("ariel", "8", "bold"))

email\_label.pack(pady=30)

email\_entry = tk.Entry(email\_popup)

email\_entry.pack()

emailb = tk.Button(email\_popup, text="Enter", command=process\_checkout)

emailb.pack()

def sending\_sms(summary, rec):

apiSecret = "23068fce54a81ddfb281dc429258bdcfb5f30dd5"

deviceId = "00000000-0000-0000-3dcc-f4835d980caa"

phone = f'+91{rec}'

message = summary

message = {

"secret": apiSecret,

"mode": "devices",

"device": deviceId,

"sim": 1,

"priority": 1,

"phone": phone,

"message": message

}

r = requests.post(url="https://www.cloud.smschef.com/api/send/sms",params=message)

# do something with response object

result = r.json()

print(result)

# Function to process checkout after entering email number

def process\_checkout():

global email\_entry, summary

total\_cost = sum(item['quantity'] \* item['price'] for item in cart)

summary = "Items Ordered:\n"

for item in cart:

summary += f"{item['name']} - Quantity: {item['quantity']}"

if 'toppings' in item:

summary += f", Toppings: {', '.join(item['toppings'])}"

elif 'special\_requests' in item:

summary += f", Special Requests: {', '.join(item['special\_requests'])}"

summary += "\n"

summary += f"Total Cost: ${total\_cost:.2f}\n"

summary += f"Email Id: {email\_entry.get()}"

# Display the summary

messagebox.showinfo("Checkout Summary", summary)

try: # Checking the user's input

int(email\_entry.get())

sending\_sms(summary, email\_entry.get())

except:

# Send email

SendingEmail()

def SendingEmail():

global email\_entry, summary

sender\_email = "abcd@gmail.com" # Use Your actual Gmail here

sender\_password = "hzey twbz pgjs qxzp" # Use your actual password here

recipient\_email = email\_entry.get() # Replace with recipient email

# Construct MIME message

message = MIMEText(summary)

message['From'] = sender\_email

message['To'] = recipient\_email

message['Subject'] = "Checkout Summary"

# Connect to SMTP server

with smtplib.SMTP('smtp.gmail.com', 587) as server:

server.starttls()

server.login(sender\_email, sender\_password)

server.sendmail(sender\_email, recipient\_email, message.as\_string())

print("Email sent successfully")

# Create a frame to contain the food items

food\_frame = tk.Frame(root)

food\_frame.pack(pady=20)

# List of food items with their image paths and vegetarian status

food\_items = [{

"name": "Burger",

"image\_path": "Images/pizza.png",

"vegetarian": False

}, {

"name": "Pizza",

"image\_path": "Images/pizza.png",

"vegetarian": False

}, {

"name": "Fries",

"image\_path": "Images/fries.png",

"vegetarian": True

}, {

"name": "Salad",

"image\_path": "Images/salad.png",

"vegetarian": True

}, {

"name": "Biryani",

"image\_path": "Images/biryani.png",

"vegetarian": False

}, {

"name": "Pasta",

"image\_path": "Images/pasta.png",

"vegetarian": True

}, {

"name": "Sandwich",

"image\_path": "Images/sandwich.png",

"vegetarian": True

}, {

"name": "Sushi",

"image\_path": "Images/sushi.png",

"vegetarian": False

}, {

"name": "Steak",

"image\_path": "Images/steak.png",

"vegetarian": False

}]

# Create radio variable

radio\_var = tk.IntVar(value=3) # Initially set to both

# Create radio buttons

veg\_radio = tk.Radiobutton(root,text="Vegetarian",variable=radio\_var,value=1,command=update\_images)

nonveg\_radio = tk.Radiobutton(root,text="Non-Vegetarian",variable=radio\_var,value=2,command=update\_images)

both\_radio = tk.Radiobutton(root,text="Both",variable=radio\_var,value=3,command=update\_images)

# Place radio buttons

veg\_radio.pack(side=tk.LEFT, padx=5)

nonveg\_radio.pack(side=tk.LEFT, padx=5)

both\_radio.pack(side=tk.LEFT, padx=5)

# Create View Cart button

view\_cart\_button = tk.Button(root,text="View Cart",command=view\_cart,bg="black",fg="white")

view\_cart\_button.pack(pady=10)

# Display images initially

update\_images()

# Run the main event loop

root.mainloop()