SHRI VILEPARLE KELAVANI MANDAL’S

**SHRI BHAGUBHAI MAFATLAL POLYTECHNIC**

**RESTAURANT MANAGEMENT**

**Tanay Rambhia**

**Muskaan Sawant**

**Shubam Jadhav**

**Course Name: Programming Python**

**Code: PRP198918**

**Semester: 4**

**Program: Information Semester**

**Roll No: T008, T017, T059**

**ACKNOWLEDGEMENT**

I would like to express my sincere gratitude and appreciation to all those who have contributed to the successful completion of this mini Python project.

First and foremost, I would like to thank our esteemed faculty members for their guidance and support throughout this project. Their expertise, encouragement, and valuable feedback played a crucial role in shaping our understanding of Python programming and its application.

I am immensely grateful to my project supervisor, for their unwavering guidance, patience, and continuous support. Their insightful suggestions and constructive criticism greatly enhanced the quality of our project. I am indebted to them for their valuable time and efforts invested in reviewing and refining our work.

I would also like to extend my heartfelt appreciation to my fellow classmates and project team members. Their dedication, collaboration, and collective effort made this project a truly enriching experience. The brainstorming sessions, exchange of ideas, and collaborative problem-solving significantly contributed to the success of our project.

**PROBLEM STATEMENT**

The objective of this project is to develop a restaurant management system that will enable managers and workers to effectively handle a variety of areas of their businesses. Using Tkinter, the system will offer a user-friendly graphical user interface (GUI) and link to a MySQL database to store and retrieve information on restaurants.

The system should take into account the following needs:

**User Authentication**: Implement a login mechanism to verify users, including restaurant owners and personnel. Based on user roles, different degrees of access should be offered.

**Menu**: Users should have the ability to add, edit, and remove menu items. Each menu item has to include information

**Inventory management:** Enable users to manage restaurant inventory by monitoring ingredient and supply stock levels. When stock levels are low, the system need to send out notifications.

**Wage Management:** It helps businesses track and manage employee wages, including calculating payroll, managing wage rates, and tracking hours worked, and generating reports.

**Database Connectivity**: Establishing a connection to a MySQL database will allow you to store and retrieve information about your business, including the menu, orders, reservations, inventory, and personnel information.

**FEATURES**

**Menu Management:**

* Create and manage the restaurant's menu items, including their names, descriptions, prices, and categories.
* Allow for adding, updating, and deleting menu items.
* Provide a user-friendly interface for menu customization.

**Order Management:**

Allow customers to place orders, either in-person or online.

**Inventory management:**

Enable users to manage restaurant inventory by monitoring ingredient and supply stock levels. When stock levels are low, the system need to send out notifications.

**Wage Management:**

It helps businesses track and manage employee wages, including calculating payroll, managing wage rates, and tracking hours worked, and generating reports.

**LITERATURE SURVEY (NEW TECHNOLOGIES/TOOLS LEARNED)**

In Restaurant Management System, Literature Survey (new Technologies/Tools learned) you can explore various technologies and tools that can enhance the development and functionality of the system like:

1. GUI (Tkinter)
2. Database Connectivity (MySQL)
3. Various Modules like:
   1. OS Module
   2. Date and Time Module
   3. File Module
   4. Random Module
4. Functions
5. Classes and Interfaces

**SRS FOR RESTAURANT MANAGEMENT**

**TABLE OF CONTENT**

1. **INTRODUCTION**
   1. Purpose \*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*
   2. Document Conventions \*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*
   3. Intended Audience and Reading Suggestion \*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*
   4. Project Scope \*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*
   5. Reference \*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*
2. **OVERALL DESCRIPTION**
   1. Product Perspective \*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*
   2. Working of Product \*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*
   3. User Class and Characteristics \*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*
   4. Operating Environment \*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*
   5. Design and Implementation Constraints \*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*
   6. Assumption Dependencies \*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*
3. **SYSTEM FEATURES**
   1. Design and Priority \*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*
   2. Stimulus/Response Sequences \*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*
4. **EXTERNAL INTERFACE REQUIREMENTS**
   1. User Interface \*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*
   2. Hardware Interface \*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*
   3. Software Interface \*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*
   4. Communication Interface \*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*
5. **NONFUNCTIONAL REQUIREMENTS**
   1. Performance Requirements \*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*
      1. ER-Diagram \*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*
      2. Normalization \*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*
   2. Safety Requirements \*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*
   3. Security Requirements \*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*
   4. Software Quality Attributes \*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*
6. **INTRODUCTION**
   1. **PURPOSE**

The purpose of this document is to go the restaurant management system and order food.

**1.2 DOCUMENT CONVENTIONS**

This document uses the following conventions:

DDB Distributed Database

ER Entity Relation

* 1. **INTENDED AUDIENCE AND READING SUGGESTIONS**

This project is a prototype for the Restaurant Management. This has been implemented under the guidance of a college professor. This project is useful for all Restaurant owners who have a Restaurants.

**1.4 PROJECT SCOPE**

The scope of this project is very high as it reduces the paperwork by a huge amount. As well as it is a user-friendly system. The system is based on a relational database. We will have a database server supporting hundreds of users around the world as well as their activity on this system. Above all, we hope to provide a comfortable user experience along with the best pricing available.

1. **OVERALL DESCRIPTION**

**2.1 PRODUCT PERSPECTIVE**

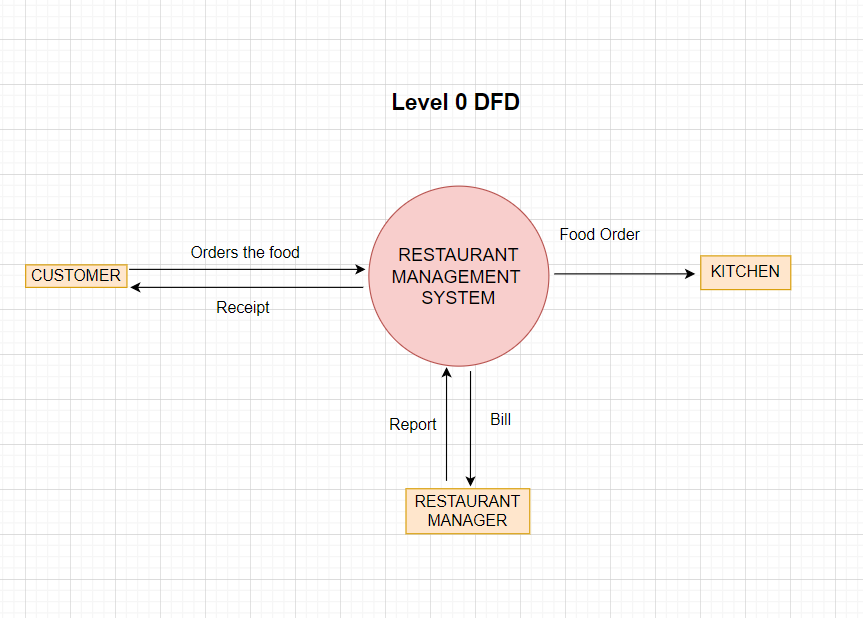
A distributed Restaurant database stores the following information:

* Customer details
* Money Details
* Order Details
* Food Details
* Inventory Details
* Workers Payroll
* Bill
* Receipt

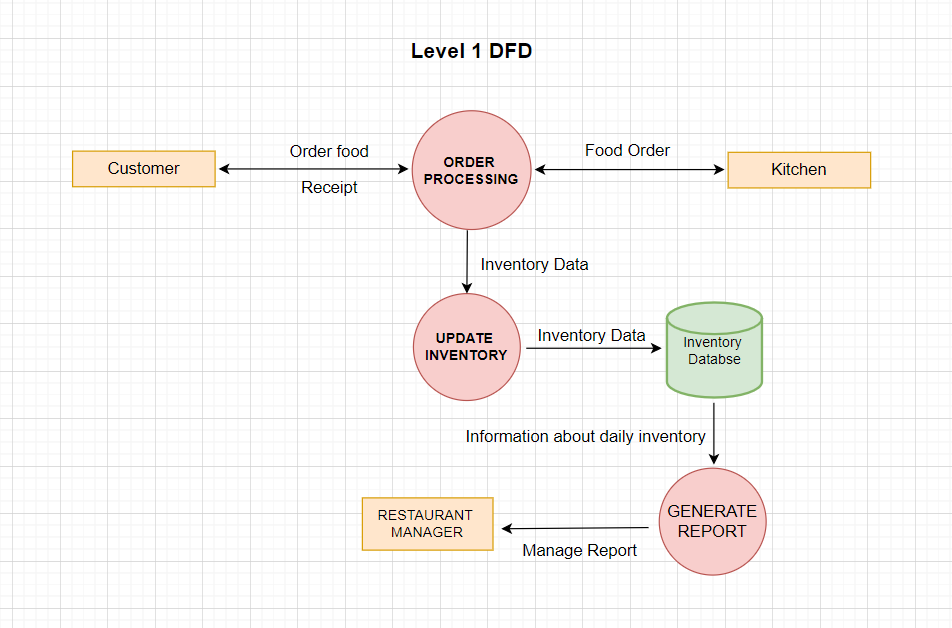
* 1. **SYSTEM DESIGN**

Data Flow Diagrams (DFD):

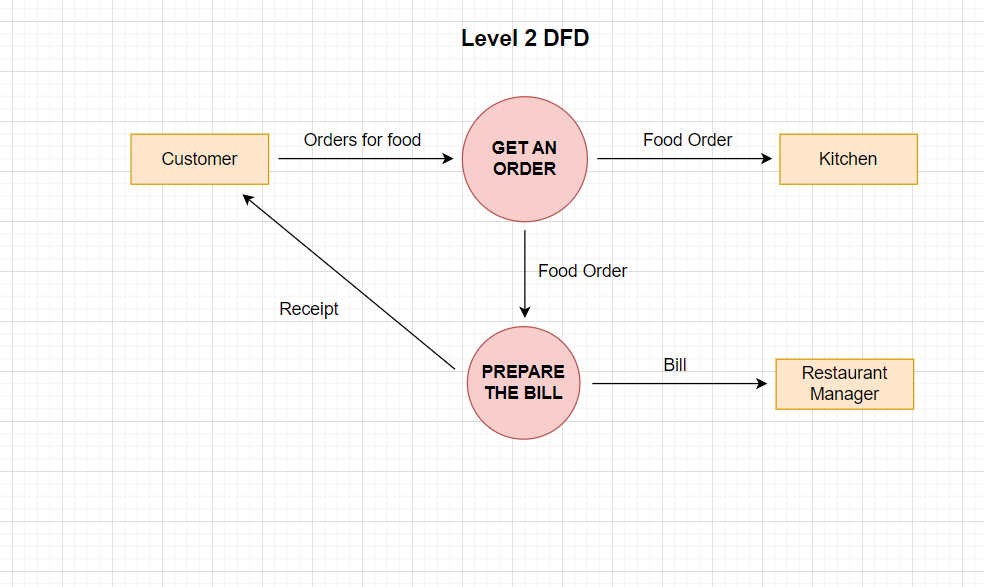
**Level 0:**



**Level 1:**

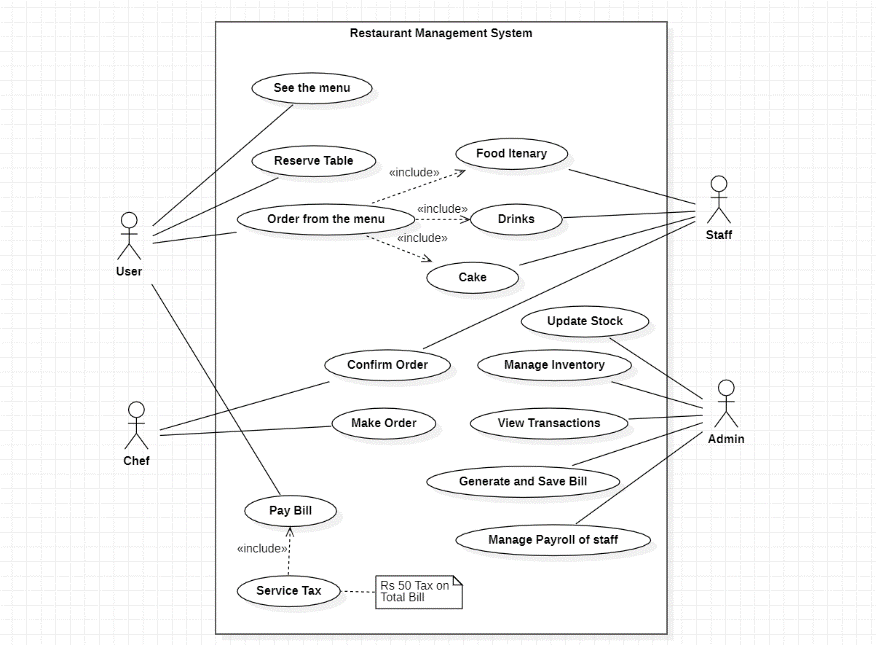
****

**Level 2:**

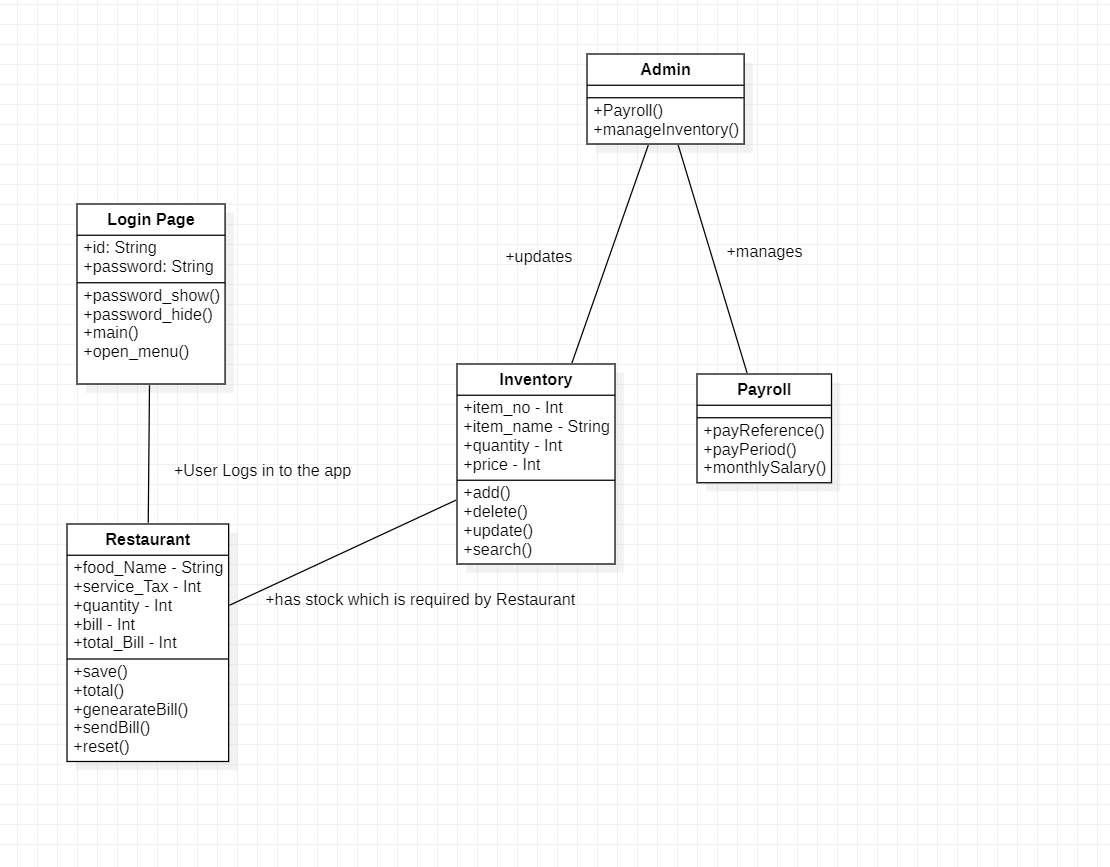
****

UML Diagrams:

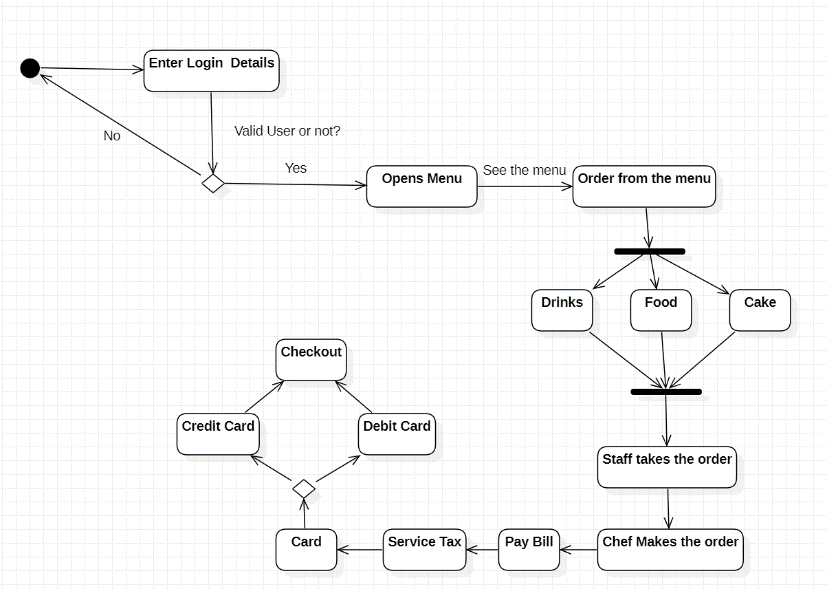
**Use Case:**

****

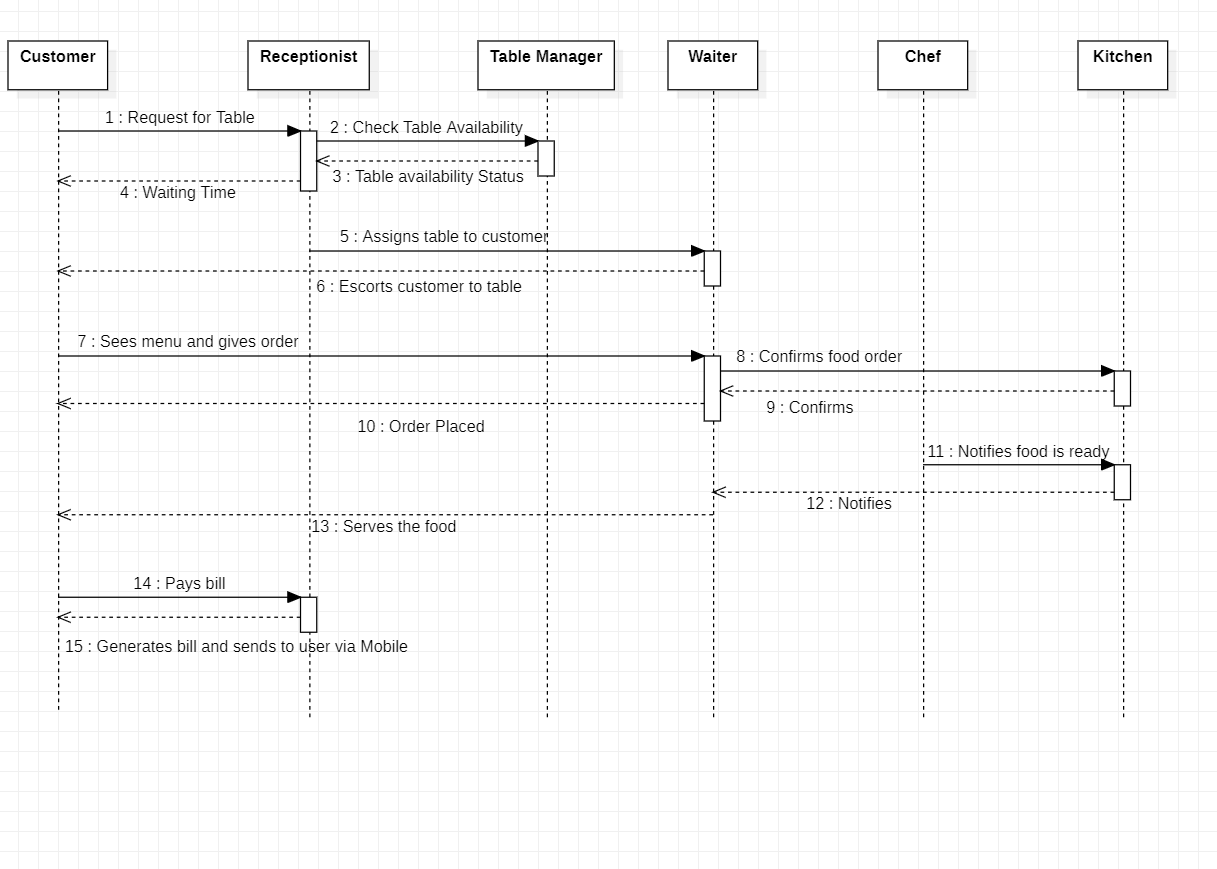
**Class Diagram:**

****

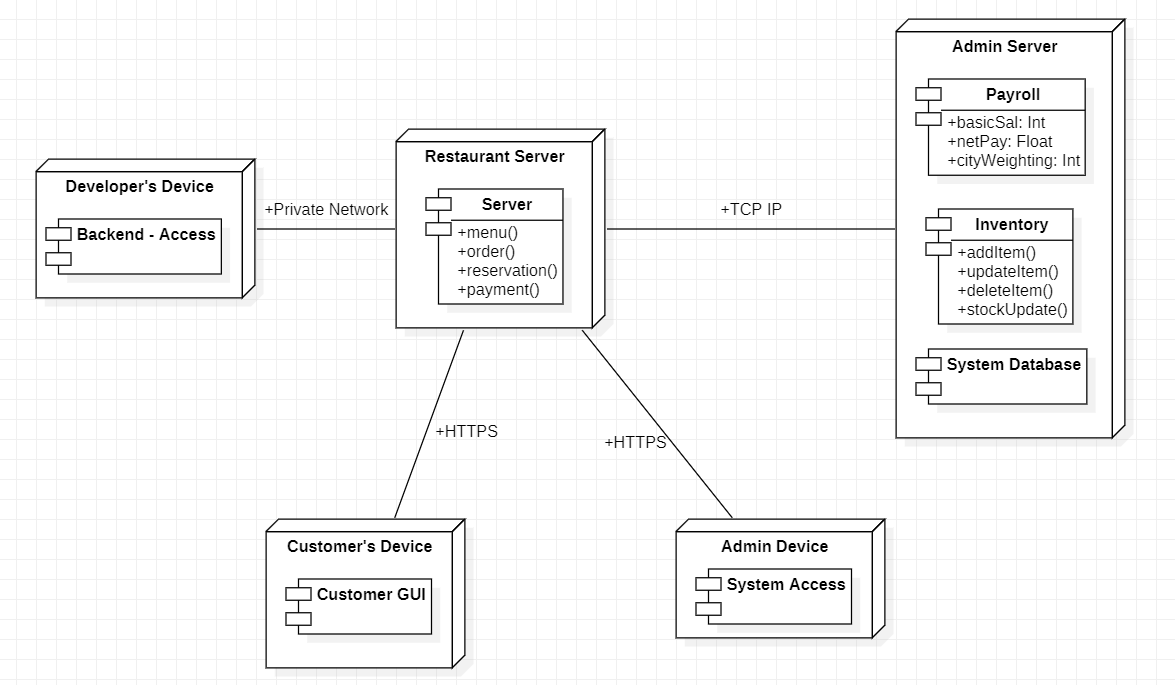
**Activity Diagram:**

****

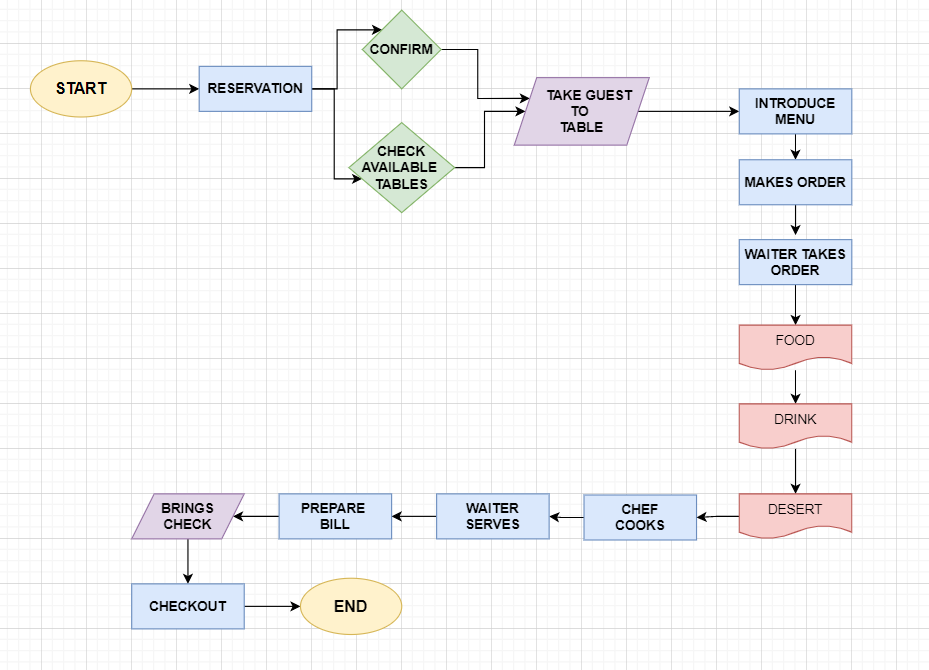
**Sequence Diagram:**

****

**Deployment Diagram:**

****

Flow Chart:



* 1. **USER CLASS AND CHARACTERISTICS**

As the project is desktop-based so the main role is of admin instead of a customer. The customer has to just visit the shop and order food. Here the user is admin.

The admin has the following features:

* Managing User Accounts
* Monitoring the system
* Performing Maintainace Task
* Responding to Security Issues
  1. **OPERATING ENVIRONMENT**

The operating environment of Restaurant Management is listed below:

* SQL Database
* OS: Windows 8
* Platform: Sublime Text and Xampp
  1. **DESIGN AND IMPLEMENTATION CONSTRAINTS**

How the response for applications 1 and 2 will be generated. Assuming these are global quire

* 1. **ASSUMPTION DEPENDENCIES**

Let us assume that this is a Restaurant, here the admin part will remain the same, but the customer can also buy a food at any time using this project.

1. **SYSTEM FEATURES**

**3.1 DESCRIPTION AND PRIORITY**

The Restaurant Management System maintains the food. Of course, this project has a higher priority as it reduces the paperwork by a huge amount, and as well as there is no fear of losing information or data.

**3.2 STIMULUS/RESPONSE SEQUENCES**

* Enter User details
* Shows report of User
* Bill Printed
* Daily and Monthly Wages

1. **EXTERNAL INTERFACE REQUIREMENTS**

**4.1 USER INTERFACE**

* Front-end software: Sublime Text

**4.2 BACKEND INTERFACE**

* Back-end software: Xampp

**4.3 HARDWARE INTERFACE**

* Windows with python already installed
* Internet connection

**4.4 SOFTWARE INTERFACE**

Following are the software used for the Restaurant Management System

|  |  |
| --- | --- |
| **Software Used** | **Description** |
| Operating system | We have chosen Windows operating system for its best support and user-friendliness. |
| Java | To implement the project we have chosen Python language for its more interactive support. |

**4.5 COMMUNICATION INTERFACES**

This project is supported in windows machine with the Sublime Text module. We are using simple electronic forms for Restaurant Management System

1. **NONFUNCTIONAL REQUIREMENTS**

**5.1 PERFORMANCE REQUIREMENTS**

The steps involved to perform the implementation of the Restaurant Management system database are as follows:

**5.1.1 ER-DIAGRAM**

The E-R Diagram constitutes a technique for representing the logical structure of a database pictorially. This analysis is then used to organize data as a relation, normalizing relation and finally obtaining a relation database.

* **ENTITIES:** Specifies distinct real-world items in an application.
* **PROPERTIES/ATTRIBUTES:** Specifies properties of an entity and relationships.
* **RELATIONSHIPS:** Connects entities and represents meaningful dependencies between them.

**5.1.2 NORMALIZATION**

The basic objective of normalization is to reduce redundancy which means that information is to be stored only once. Storing information several times leads to wastage of storage space and an increase in the total size of the data stored.

If a database is not properly designed it can give rise to modification anomalies. Modification anomalies arise when data is added to, changed, or deleted from a database table. Similarly, in traditional databases as well as improperly designed relational databases, data redundancy can be a problem. These can be eliminated by normalizing a database.

Normalization is the process of breaking down a table into smaller tables. So that each table deals with a single theme. There are three different kinds of modifications of anomalies and formulated the first, second, and third normal forms (3NF) is considered sufficient for most practical purposes. It should be considered only after a thorough analysis and a complete understanding of its implications.

**5.2 SAFETY REQUIREMENTS**

If there is extensive damage to a wide portion of the database due to catastrophic failure, such as a disk crash, the recovery method restores a past copy of the database that was backed up to archival storage (typically tape) and reconstructs a more current state by reapplying or redoing the operations of committed transactions from the backed-up log, up to the time of failure.

**5.3 SECURITY REQUIREMENTS**

Security systems need database storage just like many other applications. However, the special requirements of the security market mean that vendors must choose their database partner carefully.

**5.4 SOFTWARE QUALITY ATTRIBUTES**

* **CORRECTNESS**: All the expense and the income amount and date should be registered correctly into the database.
* **MAINTAINABILITY:** The database should be regularly backup through several RAID levels for recovery from failure.
* **USABILITY:** The database should be large enough to store the data of all of its customers whowant to use the app

**GANTT CHART (TIME CHART)**

Restaurant Management System:

**Task 1: Feasibility Study**

Start Date: [1/24/2023]

End Date: [2/11/2023]

Duration: [18 Days]

**Task 2: Requirement Analysis**

Start Date: [2/12/2023]

End Date: [2/21/2023]

Duration: [10 Days]

**Task 3: GUI Design**

Start Date: [2/22/2023]

End Date: [3/2/2023]

Duration: [10 Days]

**Task 4: Coding (Functionality Implementation)**

Start Date: [3/3/2023]

End Date: [3/18/2023]

Duration: [15 Days]

**Task 5: Integration and Testing**

Start Date: [3/18/2023]

End Date: [4/25/2023]

Duration: [38 Days]

**Task 6: Installations**

Start Date: [4/26/2023]

End Date: [5/4/2023]

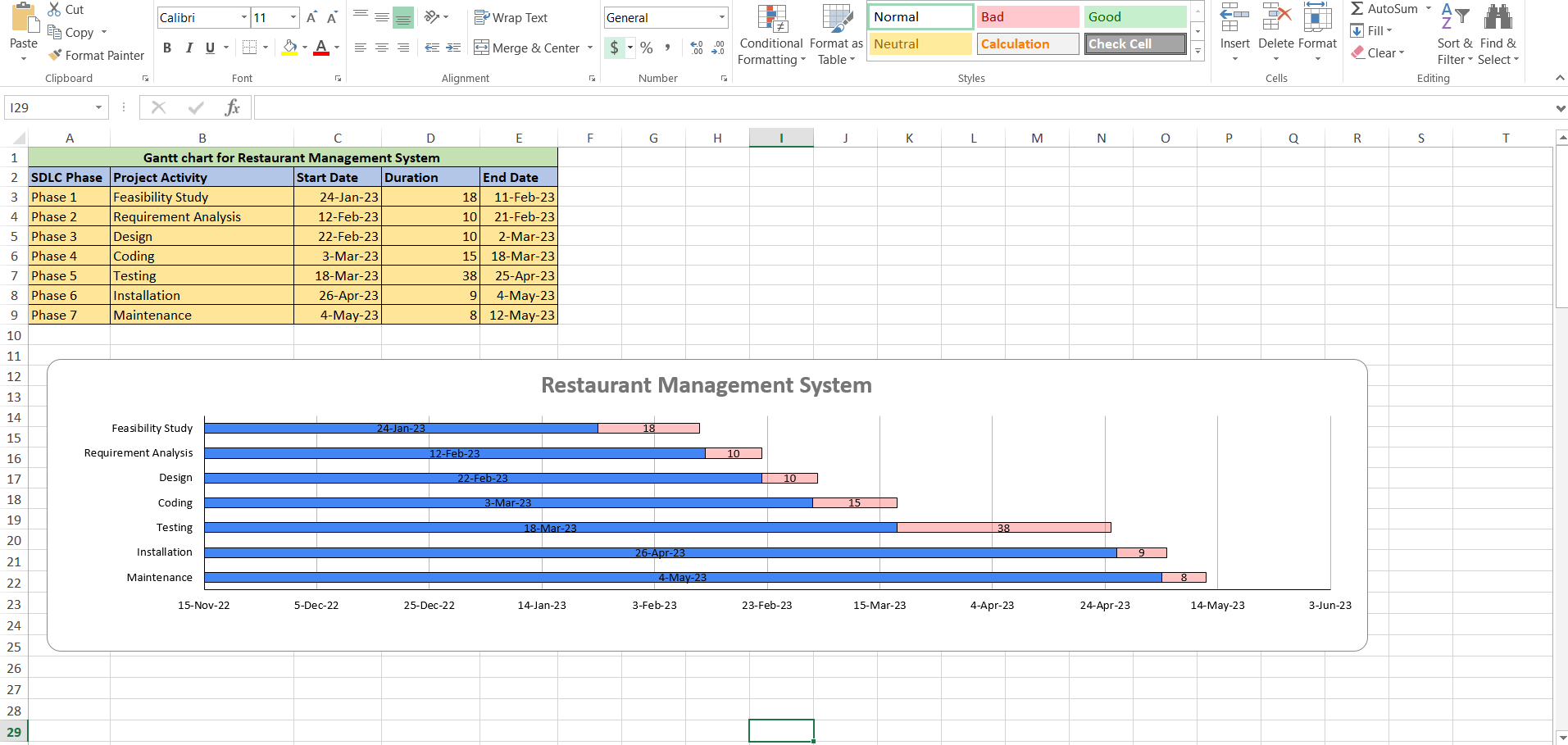
Duration: [9 Days]

**Task 7: Maintainace**

Start Date: [5/4/2023]

End Date: [5/12/2023]

Duration: [8 Days]



**MODULE DESCRIPTION WITH IMPLEMENTATION DATE (DELIVERABLES SCHEDULE)**

1. Module: User Authentication and Access Control

- Description: Implement user login and access control features to secure the system.

- Implementation Date: [24/1/23]

2. Module: Menu

- Description: Develop features for adding, editing, and deleting menu items. Include options for categorization, pricing, and displaying menu items.

- Implementation Date: [12/2/23]

3. Module: Order Management

- Description: Implement order placement, tracking, and management functionalities. Include features like adding items to the order, modifying quantities, and calculating the total price.

- Implementation Date: [22/2/23]

4. Module: Table Management

- Description: Develop features for managing restaurant tables, including table reservation, availability status, and table allocation to customers.

- Implementation Date: [3/3/23]

5. Module: Inventory Management

- Description: Implement functionalities to track inventory items, their quantities, and automatic stock update upon order placement or item consumption.

- Implementation Date: [18/3/23]

6. Module: Database Connectivity

- Description: Establish database connectivity to store and retrieve data related to menu items, orders, tables, and inventory.

- Implementation Date: [26/4/23]

**SYSTEM TESTING**

**LoginPage.py**

from tkinter import \*

from PIL import ImageTk, Image

from subprocess import Popen

class LoginPage:

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

self.window = window

self.window.geometry('1166x718')

self.window.resizable(10, 10)

self.window.state('zoomed')

self.window.title('Login Page')

# ========================================================================

# ============================background image============================

# ========================================================================

self.bg\_frame = Image.open('C:\\Users\\user\\Desktop\\Tanay\\images\\background1.png')

photo = ImageTk.PhotoImage(self.bg\_frame)

self.bg\_panel = Label(self.window, image=photo)

self.bg\_panel.image = photo

self.bg\_panel.pack(fill='both', expand='yes')

# ====== Login Frame =========================

self.lgn\_frame = Frame(self.window, bg='#040405', width=950, height=600)

self.lgn\_frame.place(x=200, y=70)

# ========================================================================

# ========================================================

# ========================================================================

self.txt = "WELCOME"

self.heading = Label(self.lgn\_frame, text=self.txt, font=('yu gothic ui', 25, "bold"), bg="#040405",

fg='white',

bd=5,

relief=FLAT)

self.heading.place(x=80, y=30, width=300, height=30)

# ========================================================================

# ============ Left Side Image ================================================

# ========================================================================

self.side\_image = Image.open('C:\\Users\\user\\Desktop\\Tanay\\images\\vector.png')

photo = ImageTk.PhotoImage(self.side\_image)

self.side\_image\_label = Label(self.lgn\_frame, image=photo, bg='#040405')

self.side\_image\_label.image = photo

self.side\_image\_label.place(x=5, y=100)

# ========================================================================

# ============ Sign In Image =============================================

# ========================================================================

self.sign\_in\_image = Image.open('C:\\Users\\user\\Desktop\\Tanay\\images\\hyy.png')

photo = ImageTk.PhotoImage(self.sign\_in\_image)

self.sign\_in\_image\_label = Label(self.lgn\_frame, image=photo, bg='#040405')

self.sign\_in\_image\_label.image = photo

self.sign\_in\_image\_label.place(x=620, y=130)

# ========================================================================

# ============ Sign In label =============================================

# ========================================================================

self.sign\_in\_label = Label(self.lgn\_frame, text="Sign In", bg="#040405", fg="white",

font=("yu gothic ui", 17, "bold"))

self.sign\_in\_label.place(x=650, y=240)

# ========================================================================

# ============================username====================================

# ========================================================================

self.username\_label = Label(self.lgn\_frame, text="Username", bg="#040405", fg="#4f4e4d",

font=("yu gothic ui", 13, "bold"))

self.username\_label.place(x=550, y=300)

self.username\_entry = Entry(self.lgn\_frame, highlightthickness=0, relief=FLAT, bg="#040405", fg="#6b6a69",

font=("yu gothic ui ", 12, "bold"), insertbackground = '#6b6a69')

self.username\_entry.place(x=580, y=335, width=270)

self.username\_line = Canvas(self.lgn\_frame, width=300, height=2.0, bg="#bdb9b1", highlightthickness=0)

self.username\_line.place(x=550, y=359)

# ===== Username icon =========

self.username\_icon = Image.open('C:\\Users\\user\\Desktop\\Tanay\\images\\username\_icon.png')

photo = ImageTk.PhotoImage(self.username\_icon)

self.username\_icon\_label = Label(self.lgn\_frame, image=photo, bg='#040405')

self.username\_icon\_label.image = photo

self.username\_icon\_label.place(x=550, y=332)

# ========================================================================

# ============================login button================================

# ========================================================================

self.lgn\_button = Image.open('C:\\Users\\user\\Desktop\\Tanay\\images\\btn1.png')

photo = ImageTk.PhotoImage(self.lgn\_button)

self.lgn\_button\_label = Label(self.lgn\_frame, image=photo, bg='#040405')

self.lgn\_button\_label.image = photo

self.lgn\_button\_label.place(x=550, y=450)

self.login = Button(self.lgn\_button\_label, text='LOGIN',command = self.open\_main, font=("yu gothic ui", 13, "bold"), width=25, bd=0,

bg='#3047ff', cursor='hand2', activebackground='#3047ff', fg='white')

self.login.place(x=20, y=10)

# ========================================================================

# ============================password====================================

# ========================================================================

self.password\_label = Label(self.lgn\_frame, text="Password", bg="#040405", fg="#4f4e4d",

font=("yu gothic ui", 13, "bold"))

self.password\_label.place(x=550, y=380)

self.password\_entry = Entry(self.lgn\_frame, highlightthickness=0, relief=FLAT, bg="#040405", fg="#6b6a69",

font=("yu gothic ui", 12, "bold"), show="\*", insertbackground = '#6b6a69')

self.password\_entry.place(x=580, y=416, width=244)

self.password\_line = Canvas(self.lgn\_frame, width=300, height=2.0, bg="#bdb9b1", highlightthickness=0)

self.password\_line.place(x=550, y=440)

# ======== Password icon ================

self.password\_icon = Image.open('C:\\Users\\user\\Desktop\\Tanay\\images\\password\_icon.png')

photo = ImageTk.PhotoImage(self.password\_icon)

self.password\_icon\_label = Label(self.lgn\_frame, image=photo, bg='#040405')

self.password\_icon\_label.image = photo

self.password\_icon\_label.place(x=550, y=414)

# ========= show/hide password ==================================================================

self.show\_image = ImageTk.PhotoImage \

(file='C:\\Users\\user\\Desktop\\Tanay\\images\\show.png')

self.hide\_image = ImageTk.PhotoImage \

(file='C:\\Users\\user\\Desktop\\Tanay\\images\\hide.png')

self.show\_button = Button(self.lgn\_frame, image=self.show\_image, command=self.show, relief=FLAT,

activebackground="white"

, borderwidth=0, background="white", cursor="hand2")

self.show\_button.place(x=860, y=420)

def open\_main(self):

Popen(["python", "RestaurantMenu.py"])

def show(self):

self.hide\_button = Button(self.lgn\_frame, image=self.hide\_image, command=self.hide, relief=FLAT,

activebackground="white"

, borderwidth=0, background="white", cursor="hand2")

self.hide\_button.place(x=860, y=420)

self.password\_entry.config(show='')

def hide(self):

self.show\_button = Button(self.lgn\_frame, image=self.show\_image, command=self.show, relief=FLAT,

activebackground="white"

, borderwidth=0, background="white", cursor="hand2")

self.show\_button.place(x=860, y=420)

self.password\_entry.config(show='\*')

def page():

window = Tk()

LoginPage(window)

window.mainloop()

if \_\_name\_\_ == '\_\_main\_\_':

page()

**RestaurantMenu.py**

from tkinter import \*

from tkinter import filedialog,messagebox

import random

import time

import requests

#Functions

def reset():

textReceipt.delete(1.0,END)

e\_daal.set('0')

e\_roti.set('0')

e\_sabji.set('0')

e\_fish.set('0')

e\_kebab.set('0')

e\_chawal.set('0')

e\_mutton.set('0')

e\_paneer.set('0')

e\_chicken.set('0')

e\_lassi.set('0')

e\_coffee.set('0')

e\_faluda.set('0')

e\_roohafza.set('0')

e\_shikanji.set('0')

e\_jaljeera.set('0')

e\_masalatea.set('0')

e\_badammilk.set('0')

e\_coldrinks.set('0')

e\_kitkat.set('0')

e\_oreo.set('0')

e\_apple.set('0')

e\_vanilla.set('0')

e\_banana.set('0')

e\_brownie.set('0')

e\_pineapple.set('0')

e\_chocolate.set('0')

e\_blackforest.set('0')

textroti.config(state=DISABLED)

textdaal.config(state=DISABLED)

textsabji.config(state=DISABLED)

textfish.config(state=DISABLED)

textkebab.config(state=DISABLED)

textpaneer.config(state=DISABLED)

textchicken.config(state=DISABLED)

textmutton.config(state=DISABLED)

textchawal.config(state=DISABLED)

textlassi.config(state=DISABLED)

textcoffee.config(state=DISABLED)

textjaljeera.config(state=DISABLED)

textroohafza.config(state=DISABLED)

textshikanji.config(state=DISABLED)

textbadammilk.config(state=DISABLED)

textmasalatea.config(state=DISABLED)

textfaluda.config(state=DISABLED)

textcolddrinks.config(state=DISABLED)

textoreo.config(state=DISABLED)

textapple.config(state=DISABLED)

textkitkat.config(state=DISABLED)

textvanilla.config(state=DISABLED)

textbanana.config(state=DISABLED)

textbrownie.config(state=DISABLED)

textpineapple.config(state=DISABLED)

textchocolate.config(state=DISABLED)

textblackforest.config(state=DISABLED)

var1.set(0)

var2.set(0)

var3.set(0)

var4.set(0)

var5.set(0)

var6.set(0)

var7.set(0)

var8.set(0)

var9.set(0)

var10.set(0)

var11.set(0)

var12.set(0)

var13.set(0)

var14.set(0)

var15.set(0)

var16.set(0)

var17.set(0)

var18.set(0)

var19.set(0)

var20.set(0)

var21.set(0)

var22.set(0)

var23.set(0)

var24.set(0)

var25.set(0)

var26.set(0)

var27.set(0)

costofdrinksvar.set('')

costoffoodvar.set('')

costofcakesvar.set('')

subtotalvar.set('')

servicetaxvar.set('')

totalcostvar.set('')

def send():

if textReceipt.get(1.0,END)=='\n':

pass

else:

def send\_msg():

message=textarea.get(1.0,END)

number=numberfield.get()

auth='woVHAjOGldMsPhnT7gS6XRIi4cYr0ym3FZkEWfKv9Qxauq8J2DHDWus7AqZKnkeXlVzQJa3fIRrp925S'

url='https://www.fast2sms.com/dev/bulk'

params={

'authorization':auth,

'message':message,

'numbers':number,

'sender-id':'FSTSMS',

'route':'p',

'language':'english'

}

response=requests.get(url,params=params)

dic=response.json()

result=dic.get('return')

if result==True:

messagebox.showinfo('Send Successfully','Message sent succesfully')

else:

messagebox.showerror('Error','Something went wrong')

root2=Toplevel()

root2.title("Send Bill")

root2.config(bg='red4')

root2.geometry('485x620+50+50')

logoImage=PhotoImage(file='sender.png')

label=Label(root2,image=logoImage,bg='red4')

label.pack(pady=5)

numberLabel=Label(root2,text='Mobile Number',font=('arial',18,'bold underline'),bg='red4',fg='white')

numberLabel.pack(pady=5)

numberfield=Entry(root2,font=('helvetica',22,'bold'),bd=3,width=24)

numberfield.pack(pady=5)

billLabel = Label(root2, text='Bill Details', font=('arial', 18, 'bold underline'), bg='red4', fg='white')

billLabel.pack(pady=5)

textarea=Text(root2,font=('arial',12,'bold'),bd=3,width=42,height=14)

textarea.pack(pady=5)

textarea.insert(END,'Receipt Ref:\t\t'+billnumber+'\t\t'+date+'\n\n')

if costoffoodvar.get() != '0 Rs':

textarea.insert(END, f'Cost Of Food\t\t\t{priceofFood}Rs\n')

if costofdrinksvar.get() != '0 Rs':

textarea.insert(END, f'Cost Of Drinks\t\t\t{priceofDrinks}Rs\n')

if costofcakesvar.get() != '0 Rs':

textarea.insert(END, f'Cost Of Cakes\t\t\t{priceofCakes}Rs\n')

textarea.insert(END, f'Sub Total\t\t\t{subtotalofItems}Rs\n')

textarea.insert(END, f'Service Tax\t\t\t{50}Rs\n')

textarea.insert(END, f'Total Cost\t\t\t{subtotalofItems + 50}Rs\n')

sendButton=Button(root2,text='SEND',font=('arial',19,'bold'),bg='white',fg='red4',bd=7,relief=GROOVE

,command=send\_msg)

sendButton.pack(pady=5)

root2.mainloop()

def save():

if textReceipt.get(1.0,END)=='\n':

pass

else:

url=filedialog.asksaveasfile(mode='w',defaultextension='.txt')

if url==None:

pass

else:

bill\_data=textReceipt.get(1.0,END)

url.write(bill\_data)

url.close()

messagebox.showinfo('Information','Your Bill Is Succesfully Saved')

def receipt():

global billnumber,date

if costoffoodvar.get() != '' or costofcakesvar.get() != '' or costofdrinksvar.get() != '':

textReceipt.delete(1.0,END)

x=random.randint(100,10000)

billnumber='BILL'+str(x)

date=time.strftime('%d/%m/%Y')

textReceipt.insert(END,'Receipt Ref:\t\t'+billnumber+'\t\t'+date+'\n')

textReceipt.insert(END,'\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\n')

textReceipt.insert(END,'Items:\t\t Cost Of Items(Rs)\n')

textReceipt.insert(END,'\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\n')

if e\_roti.get()!='0':

textReceipt.insert(END,f'Roti\t\t\t{int(e\_roti.get())\*10}\n\n')

if e\_daal.get()!='0':

textReceipt.insert(END,f'Daal\t\t\t{int(e\_daal.get())\*60}\n\n')

if e\_fish.get()!='0':

textReceipt.insert(END,f'Margeritta Pizza\t\t\t{int(e\_fish.get())\*100}\n\n')

if e\_chawal.get() != '0':

textReceipt.insert(END, f'Vada Pav:\t\t\t{int(e\_chawal.get()) \* 30}\n\n')

if e\_sabji.get() != '0':

textReceipt.insert(END, f'Sabji:\t\t\t{int(e\_sabji.get()) \* 50}\n\n')

if e\_paneer.get() != '0':

textReceipt.insert(END, f'Paneer Chilli:\t\t\t{int(e\_paneer.get()) \* 100}\n\n')

if e\_kebab.get() != '0':

textReceipt.insert(END, f'Burger:\t\t\t{int(e\_kebab.get()) \* 40}\n\n')

if e\_chicken.get() != '0':

textReceipt.insert(END, f'Paneer Tikka Roll:\t\t\t{int(e\_chicken.get()) \* 120}\n\n')

if e\_mutton.get() != '0':

textReceipt.insert(END, f'Schezwan Fried Rice:\t\t\t{int(e\_mutton.get()) \* 120}\n\n')

if e\_lassi.get() != '0':

textReceipt.insert(END, f'Lassi:\t\t\t{int(e\_lassi.get()) \* 50}\n\n')

if e\_coffee.get() != '0':

textReceipt.insert(END, f'Coffee:\t\t\t{int(e\_coffee.get()) \* 40}\n\n')

if e\_faluda.get() != '0':

textReceipt.insert(END, f'Faluda:\t\t\t{int(e\_faluda.get()) \* 80}\n\n')

if e\_shikanji.get() != '0':

textReceipt.insert(END, f'Shikanji:\t\t\t{int(e\_shikanji.get()) \* 30}\n\n')

if e\_jaljeera.get() != '0':

textReceipt.insert(END, f'Jaljeera:\t\t\t{int(e\_jaljeera.get()) \* 40}\n\n')

if e\_roohafza.get() != '0':

textReceipt.insert(END, f'Roohafza:\t\t\t{int(e\_roohafza.get()) \* 60}\n\n')

if e\_masalatea.get() != '0':

textReceipt.insert(END, f'Masala Chai:\t\t\t{int(e\_masalatea.get()) \* 20}\n\n')

if e\_badammilk.get() != '0':

textReceipt.insert(END, f'Badam Milk:\t\t\t{int(e\_badammilk.get()) \* 50}\n\n')

if e\_coldrinks.get() != '0':

textReceipt.insert(END, f'Cold Drinks:\t\t\t{int(e\_coldrinks.get()) \* 80}\n\n')

if e\_oreo.get() != '0':

textReceipt.insert(END, f'Oreo:\t\t\t{int(e\_oreo.get()) \* 400}\n\n')

if e\_apple.get() != '0':

textReceipt.insert(END, f'Apple:\t\t\t{int(e\_apple.get()) \* 300}\n\n')

if e\_kitkat.get() != '0':

textReceipt.insert(END, f'Kitkat:\t\t\t{int(e\_kitkat.get()) \* 500}\n\n')

if e\_banana.get() != '0':

textReceipt.insert(END, f'Banana:\t\t\t{int(e\_banana.get()) \* 450}\n\n')

if e\_brownie.get() != '0':

textReceipt.insert(END, f'Brownie:\t\t\t{int(e\_brownie.get()) \* 800}\n\n')

if e\_pineapple.get() != '0':

textReceipt.insert(END, f'Pineapple:\t\t\t{int(e\_pineapple.get()) \* 620}\n\n')

if e\_chocolate.get() != '0':

textReceipt.insert(END, f'Chocolate:\t\t\t{int(e\_chocolate.get()) \* 700}\n\n')

if e\_blackforest.get() != '0':

textReceipt.insert(END, f'Black Forest:\t\t\t{int(e\_blackforest.get()) \* 550}\n\n')

if e\_vanilla.get() != '0':

textReceipt.insert(END, f'Vanilla:\t\t\t{int(e\_vanilla.get()) \* 550}\n\n')

textReceipt.insert(END,'\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\n')

if costoffoodvar.get()!='0 Rs':

textReceipt.insert(END,f'Cost Of Food\t\t\t{priceofFood}Rs\n\n')

if costofdrinksvar.get() != '0 Rs':

textReceipt.insert(END,f'Cost Of Drinks\t\t\t{priceofDrinks}Rs\n\n')

if costofcakesvar.get() != '0 Rs':

textReceipt.insert(END,f'Cost Of Cakes\t\t\t{priceofCakes}Rs\n\n')

textReceipt.insert(END, f'Sub Total\t\t\t{subtotalofItems}Rs\n\n')

textReceipt.insert(END, f'Service Tax\t\t\t{50}Rs\n\n')

textReceipt.insert(END, f'Total Cost\t\t\t{subtotalofItems+50}Rs\n\n')

textReceipt.insert(END,'\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\n')

else:

messagebox.showerror('Error','No Item Is selected')

def totalcost():

global priceofFood,priceofDrinks,priceofCakes,subtotalofItems

if var1.get() != 0 or var2.get() != 0 or var3.get() != 0 or var4.get() != 0 or var5.get() != 0 or \

var6.get() != 0 or var7.get() != 0 or var8.get() != 0 or var9.get() != 0 or var10.get() != 0 or\

var11.get() != 0 or var12.get() != 0 or var13.get() != 0 or var14.get() != 0 or var15.get() != 0 or \

var16.get() != 0 or var17.get() != 0 or var18.get() != 0 or var19.get() != 0 or var20.get() != 0 or \

var21.get() != 0 or var22.get() != 0 or var23.get() != 0 or var24.get() != 0 or var25.get() != 0 or\

var26.get() != 0 or var27.get() != 0:

item1=int(e\_roti.get())

item2=int(e\_daal.get())

item3=int(e\_fish.get())

item4 = int(e\_sabji.get())

item5 = int(e\_kebab.get())

item6 = int(e\_chawal.get())

item7 = int(e\_mutton.get())

item8 = int(e\_paneer.get())

item9 = int(e\_chicken.get())

item10 = int(e\_lassi.get())

item11 = int(e\_coffee.get())

item12 = int(e\_faluda.get())

item13 = int(e\_shikanji.get())

item14 = int(e\_jaljeera.get())

item15 = int(e\_roohafza.get())

item16 = int(e\_masalatea.get())

item17 = int(e\_badammilk.get())

item18 = int(e\_coldrinks.get())

item19 = int(e\_oreo.get())

item20 = int(e\_apple.get())

item21 = int(e\_kitkat.get())

item22 = int(e\_vanilla.get())

item23 = int(e\_banana.get())

item24 = int(e\_brownie.get())

item25 = int(e\_pineapple.get())

item26 = int(e\_chocolate.get())

item27 = int(e\_blackforest.get())

priceofFood=(item1\*10)+(item2\*60)+(item3\*100)+(item4\*50)+ (item5\*40) + (item6 \* 30) + (item7 \* 120) \

+ (item8 \* 100) + (item9 \* 120)

priceofDrinks=(item10\*50)+(item11\*40)+ (item12 \* 80) + (item13 \* 30) + (item14 \* 40) + (item15 \* 60) \

+ (item16 \* 20) + (item17 \* 50) + (item18 \* 80)

priceofCakes=(item19\*400)+(item20\*300)+ (item21 \* 500) + (item22 \* 550) + (item23 \* 450) + (item24 \* 800) \

+ (item25 \* 620) + (item26 \* 700) + (item27 \* 550)

costoffoodvar.set(str(priceofFood)+ ' Rs')

costofdrinksvar.set(str(priceofDrinks)+ ' Rs')

costofcakesvar.set(str(priceofCakes)+ ' Rs')

subtotalofItems=priceofFood+priceofDrinks+priceofCakes

subtotalvar.set(str(subtotalofItems)+' Rs')

servicetaxvar.set('50 Rs')

tottalcost=subtotalofItems+50

totalcostvar.set(str(tottalcost)+' Rs')

else:

messagebox.showerror('Error','No Item Is selected')

def roti():

if var1.get()==1:

textroti.config(state=NORMAL)

textroti.delete(0,END)

textroti.focus()

else:

textroti.config(state=DISABLED)

e\_roti.set('0')

def daal():

if var2.get()==1:

textdaal.config(state=NORMAL)

textdaal.delete(0,END)

textdaal.focus()

else:

textdaal.config(state=DISABLED)

e\_daal.set('0')

def fish():

if var3.get()==1:

textfish.config(state=NORMAL)

textfish.delete(0,END)

textfish.focus()

else:

textfish.config(state=DISABLED)

e\_fish.set('0')

def sabji():

if var4.get() == 1:

textsabji.config(state=NORMAL)

textsabji.focus()

textsabji.delete(0, END)

elif var4.get() == 0:

textsabji.config(state=DISABLED)

e\_sabji.set('0')

def kebab():

if var5.get() == 1:

textkebab.config(state=NORMAL)

textkebab.focus()

textkebab.delete(0, END)

elif var5.get() == 0:

textkebab.config(state=DISABLED)

e\_kebab.set('0')

def chawal():

if var6.get() == 1:

textchawal.config(state=NORMAL)

textchawal.focus()

textchawal.delete(0, END)

elif var6.get() == 0:

textchawal.config(state=DISABLED)

e\_chawal.set('0')

def mutton():

if var7.get() == 1:

textmutton.config(state=NORMAL)

textmutton.focus()

textmutton.delete(0, END)

elif var7.get() == 0:

textmutton.config(state=DISABLED)

e\_mutton.set('0')

def paneer():

if var8.get() == 1:

textpaneer.config(state=NORMAL)

textpaneer.focus()

textpaneer.delete(0, END)

elif var8.get() == 0:

textpaneer.config(state=DISABLED)

e\_paneer.set('0')

def chicken():

if var9.get() == 1:

textchicken.config(state=NORMAL)

textchicken.focus()

textchicken.delete(0, END)

elif var9.get() == 0:

textchicken.config(state=DISABLED)

e\_chicken.set('0')

def lassi():

if var10.get() == 1:

textlassi.config(state=NORMAL)

textlassi.focus()

textlassi.delete(0, END)

elif var10.get() == 0:

textlassi.config(state=DISABLED)

e\_lassi.set('0')

def coffee():

if var11.get() == 1:

textcoffee.config(state=NORMAL)

textcoffee.focus()

textcoffee.delete(0, END)

elif var11.get() == 0:

textcoffee.config(state=DISABLED)

e\_coffee.set('0')

def faluda():

if var12.get() == 1:

textfaluda.config(state=NORMAL)

textfaluda.focus()

textfaluda.delete(0, END)

elif var12.get() == 0:

textfaluda.config(state=DISABLED)

e\_faluda.set('0')

def shikanji():

if var13.get() == 1:

textshikanji.config(state=NORMAL)

textshikanji.focus()

textshikanji.delete(0, END)

elif var13.get() == 0:

textshikanji.config(state=DISABLED)

e\_shikanji.set('0')

def jaljeera():

if var14.get() == 1:

textjaljeera.config(state=NORMAL)

textjaljeera.focus()

textjaljeera.delete(0, END)

elif var14.get() == 0:

textjaljeera.config(state=DISABLED)

e\_jaljeera.set('0')

def roohafza():

if var15.get() == 1:

textroohafza.config(state=NORMAL)

textroohafza.focus()

textroohafza.delete(0, END)

elif var15.get() == 0:

textroohafza.config(state=DISABLED)

e\_roohafza.set('0')

def masalatea():

if var16.get() == 1:

textmasalatea.config(state=NORMAL)

textmasalatea.focus()

textmasalatea.delete(0, END)

elif var16.get() == 0:

textmasalatea.config(state=DISABLED)

e\_masalatea.set('0')

def badammilk():

if var17.get() == 1:

textbadammilk.config(state=NORMAL)

textbadammilk.focus()

textbadammilk.delete(0, END)

elif var17.get() == 0:

textbadammilk.config(state=DISABLED)

e\_badammilk.set('0')

def colddrinks():

if var18.get() == 1:

textcolddrinks.config(state=NORMAL)

textcolddrinks.focus()

textcolddrinks.delete(0, END)

elif var18.get() == 0:

textcolddrinks.config(state=DISABLED)

e\_coldrinks.set('0')

def oreo():

if var19.get() == 1:

textoreo.config(state=NORMAL)

textoreo.focus()

textoreo.delete(0, END)

elif var19.get() == 0:

textoreo.config(state=DISABLED)

e\_oreo.set('0')

def apple():

if var20.get() == 1:

textapple.config(state=NORMAL)

textapple.focus()

textapple.delete(0, END)

elif var20.get() == 0:

textapple.config(state=DISABLED)

e\_apple.set('0')

def kitkat():

if var21.get() == 1:

textkitkat.config(state=NORMAL)

textkitkat.focus()

textkitkat.delete(0, END)

elif var21.get() == 0:

textkitkat.config(state=DISABLED)

e\_kitkat.set('0')

def vanilla():

if var22.get() == 1:

textvanilla.config(state=NORMAL)

textvanilla.focus()

textvanilla.delete(0, END)

elif var22.get() == 0:

textvanilla.config(state=DISABLED)

e\_vanilla.set('0')

def banana():

if var23.get() == 1:

textbanana.config(state=NORMAL)

textbanana.focus()

textbanana.delete(0, END)

elif var23.get() == 0:

textbanana.config(state=DISABLED)

e\_banana.set('0')

def brownie():

if var24.get() == 1:

textbrownie.config(state=NORMAL)

textbrownie.focus()

textbrownie.delete(0, END)

elif var24.get() == 0:

textbrownie.config(state=DISABLED)

e\_brownie.set('0')

def pineapple():

if var25.get() == 1:

textpineapple.config(state=NORMAL)

textpineapple.focus()

textpineapple.delete(0, END)

elif var25.get() == 0:

textpineapple.config(state=DISABLED)

e\_pineapple.set('0')

def chocolate():

if var26.get() == 1:

textchocolate.config(state=NORMAL)

textchocolate.focus()

textchocolate.delete(0, END)

elif var26.get() == 0:

textchocolate.config(state=DISABLED)

e\_chocolate.set('0')

def blackforest():

if var27.get() == 1:

textblackforest.config(state=NORMAL)

textblackforest.focus()

textblackforest.delete(0, END)

elif var27.get() == 0:

textblackforest.config(state=DISABLED)

e\_blackforest.set('0')

root=Tk()

root.geometry('1270x690+50+50')

root.resizable(1,1)

root.title('Restaurant Management System ')

root.config(bg='firebrick4')

topFrame=Frame(root,bd=10,relief=RIDGE,bg='firebrick4')

topFrame.pack(side=TOP)

labelTitle=Label(topFrame,text='Restaurant Management System',font=('arial',30,'bold'),fg='yellow',bd=9,

bg='red4',width=51)

labelTitle.grid(row=0,column=0)

#frames

menuFrame=Frame(root,bd=10,relief=RIDGE,bg='firebrick4')

menuFrame.pack(side=LEFT)

costFrame=Frame(menuFrame,bd=4,relief=RIDGE,bg='firebrick4',pady=10)

costFrame.pack(side=BOTTOM)

foodFrame=LabelFrame(menuFrame,text='Food',font=('arial',19,'bold'),bd=10,relief=RIDGE,fg='red4',)

foodFrame.pack(side=LEFT)

drinksFrame=LabelFrame(menuFrame,text='Drinks',font=('arial',19,'bold'),bd=10,relief=RIDGE,fg='red4')

drinksFrame.pack(side=LEFT)

cakesFrame=LabelFrame(menuFrame,text='Cakes',font=('arial',19,'bold'),bd=10,relief=RIDGE,fg='red4')

cakesFrame.pack(side=LEFT)

rightFrame=Frame(root,bd=15,relief=RIDGE,bg='red4')

rightFrame.pack(side=RIGHT)

calculatorFrame=Frame(rightFrame,bd=1,relief=RIDGE,bg='red4')

calculatorFrame.pack()

recieptFrame=Frame(rightFrame,bd=4,relief=RIDGE,bg='red4')

recieptFrame.pack()

buttonFrame=Frame(rightFrame,bd=3,relief=RIDGE,bg='red4')

buttonFrame.pack()

#Variables

var1=IntVar()

var2=IntVar()

var3=IntVar()

var4=IntVar()

var5 = IntVar()

var6 = IntVar()

var7 = IntVar()

var8 = IntVar()

var9 = IntVar()

var10 = IntVar()

var11 = IntVar()

var12 = IntVar()

var13 = IntVar()

var14 = IntVar()

var15 = IntVar()

var16 = IntVar()

var17 = IntVar()

var18 = IntVar()

var19 = IntVar()

var20 = IntVar()

var21 = IntVar()

var22 = IntVar()

var23 = IntVar()

var24 = IntVar()

var25 = IntVar()

var26 = IntVar()

var27 = IntVar()

e\_roti=StringVar()

e\_daal=StringVar()

e\_sabji = StringVar()

e\_chawal = StringVar()

e\_fish = StringVar()

e\_mutton = StringVar()

e\_kebab = StringVar()

e\_chicken = StringVar()

e\_paneer = StringVar()

e\_lassi=StringVar()

e\_coffee = StringVar()

e\_faluda = StringVar()

e\_shikanji = StringVar()

e\_roohafza = StringVar()

e\_jaljeera = StringVar()

e\_masalatea = StringVar()

e\_badammilk = StringVar()

e\_coldrinks = StringVar()

e\_oreo=StringVar()

e\_kitkat = StringVar()

e\_vanilla = StringVar()

e\_apple = StringVar()

e\_blackforest = StringVar()

e\_banana = StringVar()

e\_brownie = StringVar()

e\_pineapple = StringVar()

e\_chocolate = StringVar()

costoffoodvar=StringVar()

costofdrinksvar=StringVar()

costofcakesvar=StringVar()

subtotalvar=StringVar()

servicetaxvar=StringVar()

totalcostvar=StringVar()

e\_roti.set('0')

e\_daal.set('0')

e\_sabji.set('0')

e\_fish.set('0')

e\_kebab.set('0')

e\_chawal.set('0')

e\_mutton.set('0')

e\_chicken.set('0')

e\_paneer.set('0')

e\_lassi.set('0')

e\_coffee.set('0')

e\_faluda.set('0')

e\_roohafza.set('0')

e\_shikanji.set('0')

e\_jaljeera.set('0')

e\_masalatea.set('0')

e\_badammilk.set('0')

e\_coldrinks.set('0')

e\_kitkat.set('0')

e\_banana.set('0')

e\_pineapple.set('0')

e\_apple.set('0')

e\_chocolate.set('0')

e\_oreo.set('0')

e\_blackforest.set('0')

e\_brownie.set('0')

e\_vanilla.set('0')

##FOOD

roti=Checkbutton(foodFrame,text='Roti',font=('arial',18,'bold'),onvalue=1,offvalue=0,variable=var1

,command=roti)

roti.grid(row=0,column=0,sticky=W)

daal=Checkbutton(foodFrame,text='Daal',font=('arial',18,'bold'),onvalue=1,offvalue=0,variable=var2

,command=daal)

daal.grid(row=1,column=0,sticky=W)

fish=Checkbutton(foodFrame,text='Margeritta Pizza',font=('arial',18,'bold'),onvalue=1,offvalue=0,variable=var3

,command=fish)

fish.grid(row=2,column=0,sticky=W)

sabji=Checkbutton(foodFrame,text='Sabji',font=('arial',18,'bold'),onvalue=1,offvalue=0,variable=var4

,command=sabji)

sabji.grid(row=3,column=0,sticky=W)

kebab=Checkbutton(foodFrame,text='Burger',font=('arial',18,'bold'),onvalue=1,offvalue=0,variable=var5

,command=kebab)

kebab.grid(row=4,column=0,sticky=W)

chawal=Checkbutton(foodFrame,text='Vada Pav',font=('arial',18,'bold'),onvalue=1,offvalue=0,variable=var6

,command=chawal)

chawal.grid(row=5,column=0,sticky=W)

mutton=Checkbutton(foodFrame,text='Schezwan Fried Rice',font=('arial',18,'bold'),onvalue=1,offvalue=0,variable=var7,

command=mutton)

mutton.grid(row=6,column=0,sticky=W)

paneer=Checkbutton(foodFrame,text='Paneer Chilli',font=('arial',18,'bold'),onvalue=1,offvalue=0,variable=var8

,command=paneer)

paneer.grid(row=7,column=0,sticky=W)

chicken=Checkbutton(foodFrame,text='Paneer Tikka Roll ',font=('arial',18,'bold'),onvalue=1,offvalue=0,variable=var9

,command=chicken)

chicken.grid(row=8,column=0,sticky=W)

#Entry Fields for Food Items

textroti=Entry(foodFrame,font=('arial',18,'bold'),bd=7,width=6,state=DISABLED,textvariable=e\_roti)

textroti.grid(row=0,column=1)

textdaal=Entry(foodFrame,font=('arial',18,'bold'),bd=7,width=6,state=DISABLED,textvariable=e\_daal)

textdaal.grid(row=1,column=1)

textfish=Entry(foodFrame,font=('arial',18,'bold'),bd=7,width=6,state=DISABLED,textvariable=e\_fish)

textfish.grid(row=2,column=1)

textsabji = Entry(foodFrame, font=('arial', 18, 'bold'), bd=7, width=6, state=DISABLED, textvariable=e\_sabji)

textsabji.grid(row=3, column=1)

textkebab = Entry(foodFrame, font=('arial', 18, 'bold'), bd=7, width=6, state=DISABLED, textvariable=e\_kebab)

textkebab.grid(row=4, column=1)

textchawal = Entry(foodFrame, font=('arial', 18, 'bold'), bd=7, width=6, state=DISABLED, textvariable=e\_chawal)

textchawal.grid(row=5, column=1)

textmutton = Entry(foodFrame, font=('arial', 18, 'bold'), bd=7, width=6, state=DISABLED, textvariable=e\_mutton)

textmutton.grid(row=6, column=1)

textpaneer = Entry(foodFrame, font=('arial', 18, 'bold'), bd=7, width=6, state=DISABLED, textvariable=e\_paneer)

textpaneer.grid(row=7, column=1)

textchicken = Entry(foodFrame, font=('arial', 18, 'bold'), bd=7, width=6, state=DISABLED, textvariable=e\_chicken)

textchicken.grid(row=8, column=1)

#Drinks

lassi=Checkbutton(drinksFrame,text='Lassi',font=('arial',18,'bold'),onvalue=1,offvalue=0,variable=var10

,command=lassi)

lassi.grid(row=0,column=0,sticky=W)

coffee=Checkbutton(drinksFrame,text='Coffee',font=('arial',18,'bold'),onvalue=1,offvalue=0,variable=var11

,command=coffee)

coffee.grid(row=1,column=0,sticky=W)

faluda=Checkbutton(drinksFrame,text='Faluda',font=('arial',18,'bold'),onvalue=1,offvalue=0,variable=var12

,command=faluda)

faluda.grid(row=2,column=0,sticky=W)

shikanji=Checkbutton(drinksFrame,text='Shikanji',font=('arial',18,'bold'),onvalue=1,offvalue=0,variable=var13

,command=shikanji)

shikanji.grid(row=3,column=0,sticky=W)

jaljeera=Checkbutton(drinksFrame,text='Jaljeera',font=('arial',18,'bold'),onvalue=1,offvalue=0,variable=var14

,command=jaljeera)

jaljeera.grid(row=4,column=0,sticky=W)

roohafza=Checkbutton(drinksFrame,text='Roohafza',font=('arial',18,'bold'),onvalue=1,offvalue=0,variable=var15

,command=roohafza)

roohafza.grid(row=5,column=0,sticky=W)

masalatea=Checkbutton(drinksFrame,text='Masala Tea',font=('arial',18,'bold'),onvalue=1,offvalue=0,variable=var16

,command=masalatea)

masalatea.grid(row=6,column=0,sticky=W)

badammilk=Checkbutton(drinksFrame,text='Badam Milk',font=('arial',18,'bold'),onvalue=1,offvalue=0,variable=var17

,command=badammilk)

badammilk.grid(row=7,column=0,sticky=W)

colddrinks=Checkbutton(drinksFrame,text='Cold Drinks',font=('arial',18,'bold'),onvalue=1,offvalue=0,variable=var18

,command=colddrinks)

colddrinks.grid(row=8,column=0,sticky=W)

#entry fields for drink items

textlassi = Entry(drinksFrame, font=('arial', 18, 'bold'), bd=7, width=6, state=DISABLED, textvariable=e\_lassi)

textlassi.grid(row=0, column=1)

textcoffee = Entry(drinksFrame, font=('arial', 18, 'bold'), bd=7, width=6, state=DISABLED, textvariable=e\_coffee)

textcoffee.grid(row=1, column=1)

textfaluda = Entry(drinksFrame, font=('arial', 18, 'bold'), bd=7, width=6, state=DISABLED, textvariable=e\_faluda)

textfaluda.grid(row=2, column=1)

textshikanji = Entry(drinksFrame, font=('arial', 18, 'bold'), bd=7, width=6, state=DISABLED, textvariable=e\_shikanji)

textshikanji.grid(row=3, column=1)

textjaljeera = Entry(drinksFrame, font=('arial', 18, 'bold'), bd=7, width=6, state=DISABLED, textvariable=e\_jaljeera)

textjaljeera.grid(row=4, column=1)

textroohafza = Entry(drinksFrame, font=('arial', 18, 'bold'), bd=7, width=6, state=DISABLED, textvariable=e\_roohafza)

textroohafza.grid(row=5, column=1)

textmasalatea = Entry(drinksFrame, font=('arial', 18, 'bold'), bd=7, width=6, state=DISABLED,textvariable=e\_masalatea)

textmasalatea.grid(row=6, column=1)

textbadammilk = Entry(drinksFrame, font=('arial', 18, 'bold'), bd=7, width=6, state=DISABLED, textvariable=e\_badammilk)

textbadammilk.grid(row=7, column=1)

textcolddrinks = Entry(drinksFrame, font=('arial', 18, 'bold'), bd=7, width=6, state=DISABLED, textvariable=e\_coldrinks)

textcolddrinks.grid(row=8, column=1)

#Cakes

oreocake=Checkbutton(cakesFrame,text='Oreo',font=('arial',18,'bold'),onvalue=1,offvalue=0,variable=var19

,command=oreo)

oreocake.grid(row=0,column=0,sticky=W)

applecake=Checkbutton(cakesFrame,text='Apple',font=('arial',18,'bold'),onvalue=1,offvalue=0,variable=var20

,command=apple)

applecake.grid(row=1,column=0,sticky=W)

kitkatcake=Checkbutton(cakesFrame,text='Kitkat',font=('arial',18,'bold'),onvalue=1,offvalue=0,variable=var21

,command=kitkat)

kitkatcake.grid(row=2,column=0,sticky=W)

vanillacake=Checkbutton(cakesFrame,text='Vanilla',font=('arial',18,'bold'),onvalue=1,offvalue=0,variable=var22

,command=vanilla)

vanillacake.grid(row=3,column=0,sticky=W)

bananacake=Checkbutton(cakesFrame,text='Banana',font=('arial',18,'bold'),onvalue=1,offvalue=0,variable=var23

,command=banana)

bananacake.grid(row=4,column=0,sticky=W)

browniecake=Checkbutton(cakesFrame,text='Brownie',font=('arial',18,'bold'),onvalue=1,offvalue=0,variable=var24

,command=brownie)

browniecake.grid(row=5,column=0,sticky=W)

pineapplecake=Checkbutton(cakesFrame,text='Pineapple',font=('arial',18,'bold'),onvalue=1,offvalue=0,variable=var25

,command=pineapple)

pineapplecake.grid(row=6,column=0,sticky=W)

chocolatecake=Checkbutton(cakesFrame,text='Chocolate',font=('arial',18,'bold'),onvalue=1,offvalue=0,variable=var26

,command=chocolate)

chocolatecake.grid(row=7,column=0,sticky=W)

blackforestcake=Checkbutton(cakesFrame,text='Black Forest',font=('arial',18,'bold'),onvalue=1,offvalue=0,variable=var27

,command=blackforest)

blackforestcake.grid(row=8,column=0,sticky=W)

#entry fields for cakes

textoreo = Entry(cakesFrame, font=('arial', 18, 'bold'), bd=7, width=6, state=DISABLED, textvariable=e\_oreo)

textoreo.grid(row=0, column=1)

textapple = Entry(cakesFrame, font=('arial', 18, 'bold'), bd=7, width=6, state=DISABLED, textvariable=e\_apple)

textapple.grid(row=1, column=1)

textkitkat = Entry(cakesFrame, font=('arial', 18, 'bold'), bd=7, width=6, state=DISABLED, textvariable=e\_kitkat)

textkitkat.grid(row=2, column=1)

textvanilla = Entry(cakesFrame, font=('arial', 18, 'bold'), bd=7, width=6, state=DISABLED, textvariable=e\_vanilla)

textvanilla.grid(row=3, column=1)

textbanana = Entry(cakesFrame, font=('arial', 18, 'bold'), bd=7, width=6, state=DISABLED, textvariable=e\_banana)

textbanana.grid(row=4, column=1)

textbrownie = Entry(cakesFrame, font=('arial', 18, 'bold'), bd=7, width=6, state=DISABLED, textvariable=e\_brownie)

textbrownie.grid(row=5, column=1)

textpineapple = Entry(cakesFrame, font=('arial', 18, 'bold'), bd=7, width=6, state=DISABLED, textvariable=e\_pineapple)

textpineapple.grid(row=6, column=1)

textchocolate = Entry(cakesFrame, font=('arial', 18, 'bold'), bd=7, width=6, state=DISABLED, textvariable=e\_chocolate)

textchocolate.grid(row=7, column=1)

textblackforest = Entry(cakesFrame, font=('arial', 18, 'bold'), bd=7, width=6, state=DISABLED,textvariable=e\_blackforest)

textblackforest.grid(row=8, column=1)

#costlabels & entry fields

labelCostofFood=Label(costFrame,text='Cost of Food',font=('arial',16,'bold'),bg='firebrick4',fg='white')

labelCostofFood.grid(row=0,column=0)

textCostofFood=Entry(costFrame,font=('arial',16,'bold'),bd=6,width=14,state='readonly',textvariable=costoffoodvar)

textCostofFood.grid(row=0,column=1,padx=41)

labelCostofDrinks=Label(costFrame,text='Cost of Drinks',font=('arial',16,'bold'),bg='firebrick4',fg='white')

labelCostofDrinks.grid(row=1,column=0)

textCostofDrinks=Entry(costFrame,font=('arial',16,'bold'),bd=6,width=14,state='readonly',textvariable=costofdrinksvar)

textCostofDrinks.grid(row=1,column=1,padx=41)

labelCostofCakes=Label(costFrame,text='Cost of Cakes',font=('arial',16,'bold'),bg='firebrick4',fg='white')

labelCostofCakes.grid(row=2,column=0)

textCostofCakes=Entry(costFrame,font=('arial',16,'bold'),bd=6,width=14,state='readonly',textvariable=costofcakesvar)

textCostofCakes.grid(row=2,column=1,padx=41)

labelSubTotal=Label(costFrame,text='Sub Total',font=('arial',16,'bold'),bg='firebrick4',fg='white')

labelSubTotal.grid(row=0,column=2)

textSubTotal=Entry(costFrame,font=('arial',16,'bold'),bd=6,width=14,state='readonly',textvariable=subtotalvar)

textSubTotal.grid(row=0,column=3,padx=41)

labelServiceTax=Label(costFrame,text='Service Tax',font=('arial',16,'bold'),bg='firebrick4',fg='white')

labelServiceTax.grid(row=1,column=2)

textServiceTax=Entry(costFrame,font=('arial',16,'bold'),bd=6,width=14,state='readonly',textvariable=servicetaxvar)

textServiceTax.grid(row=1,column=3,padx=41)

labelTotalCost=Label(costFrame,text='Total Cost',font=('arial',16,'bold'),bg='firebrick4',fg='white')

labelTotalCost.grid(row=2,column=2)

textTotalCost=Entry(costFrame,font=('arial',16,'bold'),bd=6,width=14,state='readonly',textvariable=totalcostvar)

textTotalCost.grid(row=2,column=3,padx=41)

#Buttons

buttonTotal=Button(buttonFrame,text='Total',font=('arial',14,'bold'),fg='white',bg='red4',bd=3,padx=5,

command=totalcost)

buttonTotal.grid(row=0,column=0)

buttonReceipt=Button(buttonFrame,text='Receipt',font=('arial',14,'bold'),fg='white',bg='red4',bd=3,padx=5

,command=receipt)

buttonReceipt.grid(row=0,column=1)

buttonSave=Button(buttonFrame,text='Save',font=('arial',14,'bold'),fg='white',bg='red4',bd=3,padx=5

,command=save)

buttonSave.grid(row=0,column=2)

buttonSend=Button(buttonFrame,text='Send',font=('arial',14,'bold'),fg='white',bg='red4',bd=3,padx=5,command=send)

buttonSend.grid(row=0,column=3)

buttonReset=Button(buttonFrame,text='Reset',font=('arial',14,'bold'),fg='white',bg='red4',bd=3,padx=5,

command=reset)

buttonReset.grid(row=0,column=4)

#textarea for receipt

textReceipt=Text(recieptFrame,font=('arial',12,'bold'),bd=3,width=42,height=14)

textReceipt.grid(row=0,column=0)

root.mainloop()

**Admin.py**

from tkinter import \*

from subprocess import Popen

def open\_inventory():

Popen(["python", "Inventory.py"])

def open\_payroll():

Popen(["python", "Payroll.py"])

# Creating a window

win = Tk()

win.geometry("860x620")

win.minsize(width=560, height=520)

win.title("Restaurant Management System")

# Creating a title and menu frame

titleFrame = Frame(win, bg="#005B96")

titleFrame.place(relx=0.0, rely=0.0, relwidth=1.0, relheight=0.15)

menuFrame = Frame(win, bg="#6497B1")

menuFrame.place(relx=0.0, rely=0.15, relwidth=1.0, relheight=0.85)

# Creating a title heading

heading = Label(titleFrame, text="Welcome to Bhagubhai Restaurant", font=("Arial", 26, "bold"), bg="#005B96", fg="white")

heading.place(relx=0.5, rely=0.5, anchor=CENTER)

heading2 = Label(menuFrame, text="Admin's Dashboard", font=("Arial", 26, "bold"), bg="#005B96", fg="white")

heading2.place(relx=0.5, rely=0.1, anchor=CENTER)

# Creating a menu

adbk = Button(menuFrame, text="Payroll", command=open\_payroll, bg="#E83C3C", fg="white", font=("Arial", 16), width=15)

adbk.place(relx=0.5, rely=0.3, anchor=CENTER)

dltbk = Button(menuFrame, text="Inventory", command=open\_inventory, bg="#FFCC33", fg="white", font=("Arial", 16), width=15)

dltbk.place(relx=0.5, rely=0.5, anchor=CENTER)

win.mainloop()

**Inventory.py**

from tkinter import \*

import mysql.connector

root = Tk()

class InventoryManagement(Frame):

# Creates constructor for main frame of application

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

Frame.\_\_init\_\_(self)

self.master.title('Inventory Management')

self.grid()

self.items = []

root.geometry("650x450")

self.itemCount = len(self.items)

self.conn = mysql.connector.connect(host="localhost",user="root",password="",database="inventory\_db")

self.cursor = self.conn.cursor()

# Create the inventory table if it doesn't exist

self.create\_inventory\_table()

# Lines 23 - 36 are top of application, search feature labels/entry/buttons

Label(self, text='Search (Item Number): ').grid(row=0,column=1, padx=6, pady=20, sticky=E)

self.\_box1 = IntVar()

self.\_input = Entry(self, width=20, textvariable=self.\_box1)

self.\_input.grid(row=0, column=2, padx=8, pady=20, sticky=W)

self.btn1 = Button(self, text='Search',

command=self.searchInventory)

self.btn1.grid(row=0, column=3, padx=8, pady=20, sticky=W)

self.btn2 = Button(self, text='Reset', command=self.clearSearch)

self.btn2.grid(row=0, column=4, padx=4, pady=20, sticky=W)

# Lines 40 - 45 is the main text area for inventory display

self.scroll = Scrollbar(self)

self.scroll.grid(row=3, column=4)

self.text = Text(self, width=60, height=10, wrap=WORD,

yscrollcommand=self.scroll.set)

self.text.grid(row=3, column=0, columnspan=5, padx=20, pady=20)

self.scroll.config(command=self.text.yview)

# Lines 49 - 75 are labels/entry boxes for new/edit item entry

Label(self, text='Item Number ').grid(row=6, column=0, padx=6,

pady=6, sticky=W)

self.\_box2 = StringVar()

self.\_input1 = Entry(self, width=20, textvariable=self.\_box2)

self.\_input1.grid(row=6, column=1, padx=8, pady=10, sticky=E)

Label(self, text='Item Name ').grid(row=6, column=2, padx=6,

pady=6, sticky=E)

self.\_box3 = StringVar()

self.\_input = Entry(self, width=20, textvariable=self.\_box3)

self.\_input.grid(row=6, column=3, padx=8, pady=10, sticky=E)

Label(self, text='On Hand ').grid(row=10, column=0, padx=6,

pady=6, sticky=E)

self.\_box4 = StringVar()

self.\_input = Entry(self, width=20, textvariable=self.\_box4)

self.\_input.grid(row=10, column=1, padx=8, pady=10, sticky=W)

Label(self, text='Price ').grid(row=10, column=2, padx=6,

pady=6, sticky=E)

self.\_box5 = StringVar()

self.\_input = Entry(self, width=20, textvariable=self.\_box5)

self.\_input.grid(row=10, column=3, padx=8, pady=10)

# Lines 79 - 88 are buttons for corresponding functions to add/edit/delete items from text area

self.btn3 = Button(self, text='Add Item', command=self.addItem)

self.btn3.grid(row=11, column=1, padx=5, pady=20, sticky=W)

self.btn4 = Button(self, text='Edit Item',

command=self.editItem)

self.btn4.grid(row=11, column=2, padx=5, pady=20, sticky=W)

self.btn4 = Button(self, text='Delete Item',

command=self.deleteItem)

self.btn4.grid(row=11, column=3, padx=5, pady=20, sticky=W)

# Lines 91 - 98 inserts headers into text area and sets focus to Item Number entry box

self.text.insert(END, 'Item Number' + '\t\t' + 'Item Name'

+ '\t\t' + 'On Hand' + '\t\t' + 'Price'

+ '\t\t')

self.text.insert(END,

'------------------------------------------------------------'

)

self.text.configure(state="disabled")

self.\_input1.focus\_set()

def create\_inventory\_table(self):

# Create the inventory table if it doesn't exist

create\_table\_query = '''

CREATE TABLE IF NOT EXISTS inventory (

item\_number INT AUTO\_INCREMENT PRIMARY KEY,

item\_name VARCHAR(255),

on\_hand INT,

price DECIMAL(10, 2)

)

'''

self.cursor.execute(create\_table\_query)

self.conn.commit()

def addItem(self):

self.text.configure(state="normal")

self.text.delete(1.0, END)

self.text.insert(END, 'Item Number' + '\t\t' + 'Item Name'

+ '\t\t' + 'On Hand' + '\t\t' + 'Price'

+ '\t\t')

self.text.insert(END,

'------------------------------------------------------------'

)

items = self.items

iNum = self.\_box2.get()

iName = self.\_box3.get()

oHand = self.\_box4.get()

iPrice = self.\_box5.get()

if (iNum != '' and iName != '' and oHand != '' and iPrice != ''):

record = {

0: iNum,

1: iName,

2: oHand,

3: iPrice,

}

items.append(record)

for item in items:

self.text.insert(END, item[0] + '\t\t' + item[1] + '\t\t'

+ item[2] + '\t\t' + item[3] + '\t\t')

else:

self.text.delete(1.0, END)

self.text.insert(END, 'Error: One or more fields have been left blank.')

self.\_box2.set('')

self.\_box3.set('')

self.\_box4.set('')

self.\_box5.set('')

self.\_input1.focus\_set()

self.text.configure(state="disabled")

# Insert the item into the inventory table

insert\_query = '''

INSERT INTO inventory (item\_name, on\_hand, price)

VALUES (%s, %s, %s)

'''

values = (iName, oHand, iPrice)

self.cursor.execute(insert\_query, values)

self.conn.commit()

def searchInventory(self):

self.text.configure(state="normal")

self.text.delete(1.0, END)

self.text.insert(END, 'Item Number' + '\t\t' + 'Item Name'

+ '\t\t' + 'On Hand' + '\t\t' + 'Price'

+ '\t\t')

self.text.insert(END,

'------------------------------------------------------------'

)

searchVal = str(self.\_box1.get())

for item in self.items:

if item[0] == searchVal:

self.text.insert(END, item[0] + '\t\t' + item[1]

+ '\t\t' + item[2] + '\t\t' + item[3]

+ '\t\t')

self.text.configure(state="disabled")

# Simple function attached to reset button to clear the search box

search\_query = '''SELECT \* FROM inventory WHERE item\_number = %s '''

values = (searchVal,)

self.cursor.execute(search\_query, values)

items = self.cursor.fetchall()

for item in items:

self.text.insert(END, item[0] + '\t\t' + item[1]

+ '\t\t' + str(item[2]) + '\t\t' + str(item[3])

+ '\t\t')

def clearSearch(self):

self.\_box1.set('')

''' editItem() function clears the entry boxes to prevent errors. It then grabs the search box value and compares

to the list of dictionaries. If the dictionary's item number matches the value it inserts the value of the

dictionary into the entry boxes for editing. '''

def editItem(self):

self.text.configure(state="normal")

self.\_box2.set('')

self.\_box3.set('')

self.\_box4.set('')

self.\_box5.set('')

items = self.items

searchVal = str(self.\_box1.get())

for item in items:

if item[0] == searchVal:

self.items.remove(item)

self.\_box2.set(item[0])

self.\_box3.set(item[1])

self.\_box4.set(item[2])

self.\_box5.set(item[3])

self.\_box1.set('')

self.\_input1.focus\_set()

self.text.configure(state="disabled")

update\_query = '''

UPDATE inventory SET item\_name = %s, on\_hand = %s, price = %s

WHERE item\_number = %s

'''

values = (iName, oHand, iPrice, searchVal)

self.cursor.execute(update\_query, values)

self.conn.commit()

# Simple function to delete dictionary with item number that matches the search box value

def deleteItem(self):

self.text.configure(state="normal")

self.text.delete(1.0, END)

self.text.insert(END, 'Item Number' + '\t\t' + 'Item Name'

+ '\t\t' + 'On Hand' + '\t\t' + 'Price'

+ '\t\t')

self.text.insert(END,

'------------------------------------------------------------'

)

items = self.items

searchVal = str(self.\_box1.get())

for item in items:

if item[0] == searchVal:

self.items.remove(item)

for item in items:

self.text.insert(END, item[0] + '\t\t' + item[1] + '\t\t'

+ item[2] + '\t\t' + item[3] + '\t\t')

self.\_box1.set('')

self.text.configure(state="disabled")

# Delete the item from the inventory table

delete\_query = '''

DELETE FROM inventory WHERE item\_number = %s

'''

values = (searchVal,)

self.cursor.execute(delete\_query, values)

self.conn.commit()

def close\_connection(self):

# Close the database connection

self.cursor.close()

self.conn.close()

print("Database connection closed.")

def mainloop(self):

root.mainloop()

self.close\_connection()

def main():

InventoryManagement().mainloop()

main()

**Payroll.py**

from tkinter import \*

import random

import time

import datetime

from tkinter import messagebox

payroll = Tk()

payroll.geometry("1350x650")

payroll.resizable(0, 0)

payroll.title("Payroll Wages")

def exit():

payroll.destroy()

def reset():

EmployeeName.set("")

Address.set("")

Reference.set("")

EmployerName.set("")

City.set("")

Basic.set("")

OverTime.set("")

GrossPay.set("")

NetPay.set("")

Tax.set("")

PostCode.set("")

Gender.set("")

PayDate.set("")

Pension.set("")

StudenLoan.set("")

NIPayment.set("")

Deducations.set("")

TaxPeriod.set("")

NINumber.set("")

NICode.set("")

TaxablePay.set("")

PensionablePay.set("")

OtherPaymentDue.set("")

def PayRef():

PayDate.set(time.strftime("%d/%m/%Y"))

refPay = random.randint(20000, 709467)

refPaid = ("PR" + str(refPay))

Reference.set(refPaid)

NIPay = random.randint(20000, 559467)

NIPaid = ("NI" + str(NIPay))

NINumber.set(NIPaid)

def PayPeriod():

i = datetime.datetime.now()

TaxPeriod.set(i.month)

NCode = random.randint(1200, 3467)

CodeNI = ("NICode" + str(NCode))

NICode.set(CodeNI)

def MonthlySalary():

if Basic.get() == "":

BS = 0

else:

try:

BS = float(Basic.get())

except ValueError:

messagebox.showinfo("Error", "Wrong values!!! Use numbers.")

Basic.set("")

if City.get() == "":

CW = 0

else:

try:

CW = float(City.get())

except ValueError:

messagebox.showinfo("Error", "Wrong values!!! Use numbers.")

City.set("")

if OverTime.get() == "":

OT = 0

else:

try:

OT = float(OverTime.get())

except ValueError:

messagebox.showinfo("Error", "Wrong values!!! Use numbers.")

OverTime.set("")

MTax = ((BS + CW + OT) \* 0.3)

TTax = "Rs", str('%.2f' % ((MTax)))

Tax.set(TTax)

M\_StudenLoan = ((BS + CW + OT) \* 0.02)

MM\_StudenLoan = "Rs", str('%.2f' % ((M\_StudenLoan)))

StudenLoan.set(MM\_StudenLoan)

M\_Pension = ((BS + CW + OT) \* 0.012)

MM\_Pension = "Rs", str('%.2f' % ((M\_Pension)))

Pension.set(MM\_Pension)

M\_NIPayment = ((BS + CW + OT) \* 0.021)

MM\_NIPayment = "Rs", str('%.2f' % ((M\_NIPayment)))

NIPayment.set(MM\_NIPayment)

Deduct = MTax + M\_Pension + M\_StudenLoan + M\_NIPayment

Deducat\_Payment = "Rs", str('%.2f' % ((Deduct)))

Deducations.set(Deducat\_Payment)

NetPayAfter = ((BS + CW + OT) - Deduct)

NetAfter = "Rs", str('%.2f' % ((NetPayAfter)))

NetPay.set(NetAfter)

Gross\_Pay = "Rs", str('%.2f' % (BS + CW + OT))

GrossPay.set(Gross\_Pay)

TaxablePay.set(TTax)

PensionablePay.set(MM\_Pension)

OtherPaymentDue.set("0.00")

EmployeeName = StringVar()

Address = StringVar()

Reference = StringVar()

EmployerName = StringVar()

City = StringVar()

Basic = StringVar()

OverTime = StringVar()

GrossPay = StringVar()

NetPay = StringVar()

Tax = StringVar()

PostCode = StringVar()

Gender = StringVar()

PayDate = StringVar()

Pension = StringVar()

StudenLoan = StringVar()

NIPayment = StringVar()

Deducations = StringVar()

TaxPeriod = StringVar()

NINumber = StringVar()

NICode = StringVar()

TaxablePay = StringVar()

PensionablePay = StringVar()

OtherPaymentDue = StringVar()

textInput = StringVar()

Tops=Frame(payroll, width=1350, height=50, bd=16, relief="raise")

Tops.pack(side=TOP)

LF=Frame(payroll, width=700, height=650, bd=12, relief="raise")

LF.pack(side=LEFT)

RF=Frame(payroll, width=600, height=650, bd=12, relief="raise")

RF.pack(side=RIGHT)

#==================

lblTitle = Label(Tops, font=('arial', 50, 'bold'), text="Payroll Wages", fg="Steel blue", bd=10, anchor="w")

lblTitle.grid(row=0, column=0)

#==================

InsideLF=Frame(LF, width=700, height=100, bd=8, relief="raise")

InsideLF.pack(side=TOP)

InsideLFL=Frame(LF, width=325, height=400, bd=8, relief="raise")

InsideLFL.pack(side=LEFT)

InsideLFR=Frame(LF, width=325, height=400, bd=8, relief="raise")

InsideLFR.pack(side=RIGHT)

#==================

InsideRF=Frame(RF, width=600, height=200, bd=8, relief="raise")

InsideRF.pack(side=TOP)

InsideRFL=Frame(RF, width=300, height=400, bd=8, relief="raise")

InsideRFL.pack(side=LEFT)

InsideRFR=Frame(RF, width=300, height=400, bd=8, relief="raise")

InsideRFR.pack(side=RIGHT)

#==================Left Side

lblEmployeeName = Label(InsideLF, font=('arial', 12, 'bold'), text="Employee Name", fg="Steel blue", bd=10, anchor="w")

lblEmployeeName.grid(row=0, column=0)

txtEmployeeName = Entry(InsideLF, font=('arial', 12, 'bold'), bd=20, width=54, bg="powder blue", justify="left", textvariable = EmployeeName)

txtEmployeeName.grid(row=0, column=1)

lblAddress = Label(InsideLF, font=('arial', 12, 'bold'), text="Address", fg="Steel blue", bd=10, anchor="w")

lblAddress.grid(row=1, column=0)

txtAddress = Entry(InsideLF, font=('arial', 12, 'bold'), bd=20, width=54, bg="powder blue", justify="left", textvariable = Address)

txtAddress.grid(row=1, column=1)

lblReference = Label(InsideLF, font=('arial', 12, 'bold'), text="Reference", fg="Steel blue", bd=10, anchor="w")

lblReference.grid(row=2, column=0)

txtReference = Entry(InsideLF, font=('arial', 12, 'bold'), bd=20, width=54, bg="powder blue", justify="left", textvariable = Reference)

txtReference.grid(row=2, column=1)

lblEmployerName = Label(InsideLF, font=('arial', 12, 'bold'), text="Employer Name", fg="Steel blue", bd=10, anchor="w")

lblEmployerName.grid(row=3, column=0)

txtEmployerName = Entry(InsideLF, font=('arial', 12, 'bold'), bd=20, width=54, bg="powder blue", justify="left", textvariable = EmployerName)

txtEmployerName.grid(row=3, column=1)

#----------------------Left Left Side

lblCity = Label(InsideLFL, font=('arial', 12, 'bold'), text="City Weighting", fg="Steel blue", bd=10, anchor="w")

lblCity.grid(row=0, column=0)

txtCity = Entry(InsideLFL, font=('arial', 12, 'bold'), bd=10, width=18, bg="powder blue", justify="right", textvariable = City)

txtCity.grid(row=0, column=1)

lblBasic = Label(InsideLFL, font=('arial', 12, 'bold'), text="Basic Salary", fg="Steel blue", bd=10, anchor="w")

lblBasic.grid(row=1, column=0)

txtBasic = Entry(InsideLFL, font=('arial', 12, 'bold'), bd=10, width=18, bg="powder blue", justify="right", textvariable = Basic)

txtBasic.grid(row=1, column=1)

lblOverTime = Label(InsideLFL, font=('arial', 12, 'bold'), text="Over Time", fg="Steel blue", bd=10, anchor="w")

lblOverTime.grid(row=2, column=0)

txtOverTime = Entry(InsideLFL, font=('arial', 12, 'bold'), bd=10, width=18, bg="powder blue", justify="right", textvariable = OverTime)

txtOverTime.grid(row=2, column=1)

lblGrossPay = Label(InsideLFL, font=('arial', 12, 'bold'), text="Gross Pay", fg="Steel blue", bd=10, anchor="w")

lblGrossPay.grid(row=3, column=0)

lblGrossPay = Entry(InsideLFL, font=('arial', 12, 'bold'), bd=10, width=18, bg="powder blue", justify="right", textvariable = GrossPay)

lblGrossPay.grid(row=3, column=1)

lblNetPay = Label(InsideLFL, font=('arial', 12, 'bold'), text="Net Pay", fg="Steel blue", bd=10, anchor="w")

lblNetPay.grid(row=4, column=0)

lblNetPay = Entry(InsideLFL, font=('arial', 12, 'bold'), bd=10, width=18, bg="powder blue", justify="right", textvariable = NetPay)

lblNetPay.grid(row=4, column=1)

#-------------------Left Right Side

lblTax = Label(InsideLFR, font=('arial', 12, 'bold'), text="Tax", fg="Steel blue", bd=10, anchor="w")

lblTax.grid(row=0, column=0)

txtTax = Entry(InsideLFR, font=('arial', 12, 'bold'), bd=10, width=18, bg="powder blue", justify="right", textvariable = Tax)

txtTax.grid(row=0, column=1)

lblPension = Label(InsideLFR, font=('arial', 12, 'bold'), text="Pension", fg="Steel blue", bd=10, anchor="w")

lblPension.grid(row=1, column=0)

txtPension = Entry(InsideLFR, font=('arial', 12, 'bold'), bd=10, width=18, bg="powder blue", justify="right", textvariable = Pension)

txtPension.grid(row=1, column=1)

lblStudenLoan = Label(InsideLFR, font=('arial', 12, 'bold'), text="Studen Loan", fg="Steel blue", bd=10, anchor="w")

lblStudenLoan.grid(row=2, column=0)

txtStudenLoan = Entry(InsideLFR, font=('arial', 12, 'bold'), bd=10, width=18, bg="powder blue", justify="right", textvariable = StudenLoan)

txtStudenLoan.grid(row=2, column=1)

lblNIPavment = Label(InsideLFR, font=('arial', 12, 'bold'), text="Insurance Pavment", fg="Steel blue", bd=10, anchor="w")

lblNIPavment.grid(row=3, column=0)

txtNIPavment = Entry(InsideLFR, font=('arial', 12, 'bold'), bd=10, width=18, bg="powder blue", justify="right", textvariable = NIPayment)

txtNIPavment.grid(row=3, column=1)

lblDeducations = Label(InsideLFR, font=('arial', 12, 'bold'), text="Deducations", fg="Steel blue", bd=10, anchor="w")

lblDeducations.grid(row=4, column=0)

txtDeducations = Entry(InsideLFR, font=('arial', 12, 'bold'), bd=10, width=18, bg="powder blue", justify="right", textvariable = Deducations)

txtDeducations.grid(row=4, column=1)

#==================Right Side

lblPostCode = Label(InsideRF, font=('arial', 12, 'bold'), text="Post Code", fg="Steel blue", bd=10, anchor="w")

lblPostCode.grid(row=0, column=0)

txtPostCode = Entry(InsideRF, font=('arial', 12, 'bold'), bd=10, width=50, bg="powder blue", justify="right", textvariable = PostCode)

txtPostCode.grid(row=0, column=1)

lblGender = Label(InsideRF, font=('arial', 12, 'bold'), text="Gender", fg="Steel blue", bd=10, anchor="w")

lblGender.grid(row=1, column=0)

txtGender = Entry(InsideRF, font=('arial', 12, 'bold'), bd=10, width=50, bg="powder blue", justify="right", textvariable = Gender)

txtGender.grid(row=1, column=1)

#----------------------------

lblPayDate = Label(InsideRFL, font=('arial', 12, 'bold'), text="Pay Date", fg="Steel blue", bd=10, anchor="w")

lblPayDate.grid(row=0, column=0)

txtPayDate = Entry(InsideRFL, font=('arial', 12, 'bold'), bd=10, width=18, bg="powder blue", justify="left", textvariable = PayDate)

txtPayDate.grid(row=0, column=1)

lblTaxPeriod = Label(InsideRFL, font=('arial', 12, 'bold'), text="Tax Period", fg="Steel blue", bd=10, anchor="w")

lblTaxPeriod.grid(row=1, column=0)

txtTaxPeriod = Entry(InsideRFL, font=('arial', 12, 'bold'), bd=10, width=18, bg="powder blue", justify="left", textvariable = TaxPeriod)

txtTaxPeriod.grid(row=1, column=1)

lblNINumber = Label(InsideRFL, font=('arial', 12, 'bold'), text="Insurance Number", fg="Steel blue", bd=10, anchor="w")

lblNINumber.grid(row=2, column=0)

txtNINumber = Entry(InsideRFL, font=('arial', 12, 'bold'), bd=10, width=18, bg="powder blue", justify="left", textvariable = NINumber)

txtNINumber.grid(row=2, column=1)

lblNICode = Label(InsideRFL, font=('arial', 12, 'bold'), text="Insurance Code", fg="Steel blue", bd=10, anchor="w")

lblNICode.grid(row=3, column=0)

txtNICode = Entry(InsideRFL, font=('arial', 12, 'bold'), bd=10, width=18, bg="powder blue", justify="left", textvariable = NICode)

txtNICode.grid(row=3, column=1)

lblTaxablePay = Label(InsideRFL, font=('arial', 12, 'bold'), text="Taxable Pay ", fg="Steel blue", bd=10, anchor="w")

lblTaxablePay .grid(row=4, column=0)

txtTaxablePay = Entry(InsideRFL, font=('arial', 12, 'bold'), bd=10, width=18, bg="powder blue", justify="left", textvariable = TaxablePay)

txtTaxablePay .grid(row=4, column=1)

lblPensionablePay = Label(InsideRFL, font=('arial', 12, 'bold'), text="Pensionable Pay", fg="Steel blue", bd=10, anchor="w")

lblPensionablePay.grid(row=5, column=0)

txtPensionablePay = Entry(InsideRFL, font=('arial', 12, 'bold'), bd=10, width=18, bg="powder blue", justify="left", textvariable = PensionablePay)

txtPensionablePay.grid(row=5, column=1)

lblOtherPaymentDue = Label(InsideRFL, font=('arial', 12, 'bold'), text="Other Payment Due", fg="Steel blue", bd=10, anchor="w")

lblOtherPaymentDue.grid(row=6, column=0)

txtOtherPaymentDue = Entry(InsideRFL, font=('arial', 12, 'bold'), bd=10, width=18, bg="powder blue", justify="left", textvariable = OtherPaymentDue)

txtOtherPaymentDue.grid(row=6, column=1)

#----------------------

btnWagePayment = Button(InsideRFR, padx=8, pady=8, fg="black", font=('arial', 12, 'bold'), width=14,

text="Wage Paymant", bg="sky blue", command=MonthlySalary).grid(row=0, column=0)

btnReset = Button(InsideRFR, padx=8, pady=8, fg="black", font=('arial', 12, 'bold'), width=14,

text="Reset System", bg="sky blue", command=reset).grid(row=1, column=0)

btnPayRef = Button(InsideRFR, padx=8, pady=8, fg="black", font=('arial', 12, 'bold'), width=14,

text="Pay Reference", bg="sky blue", command=PayRef).grid(row=2, column=0)

btnPayCode = Button(InsideRFR, padx=8, pady=8, fg="black", font=('arial', 12, 'bold'), width=14,

text="Pay Code", bg="sky blue", command=PayPeriod).grid(row=3, column=0)

btnExit = Button(InsideRFR, padx=8, pady=8, fg="black", font=('arial', 12, 'bold'), width=14,

text="Exit", bg="sky blue", command=exit).grid(row=4, column=0)

payroll.mainloop()

**SYSTEM TESTING**

The following are some essential components of a Python software testing project for restaurant management:

1. Unit Testing: It includes evaluating distinct parts or units of code, such as methods or functions, to make sure they execute properly when used alone. For instance, you may test the procedures in charge of adding or removing menu items, figuring out costs, or changing bookings.

2. Integration Testing: This kind of testing is concerned with confirming that various software components interact and communicate properly. You would test how different modules, including menu management, order processing, and customer management, operate together in the restaurant management project.

3. Functional testing: This entails evaluating the programme in light of its functionality. You would evaluate the software's ability to carry out all the anticipated tasks, including adding new menu items, taking orders, creating invoices, and handling reservations, in the context of the restaurant management mini project.

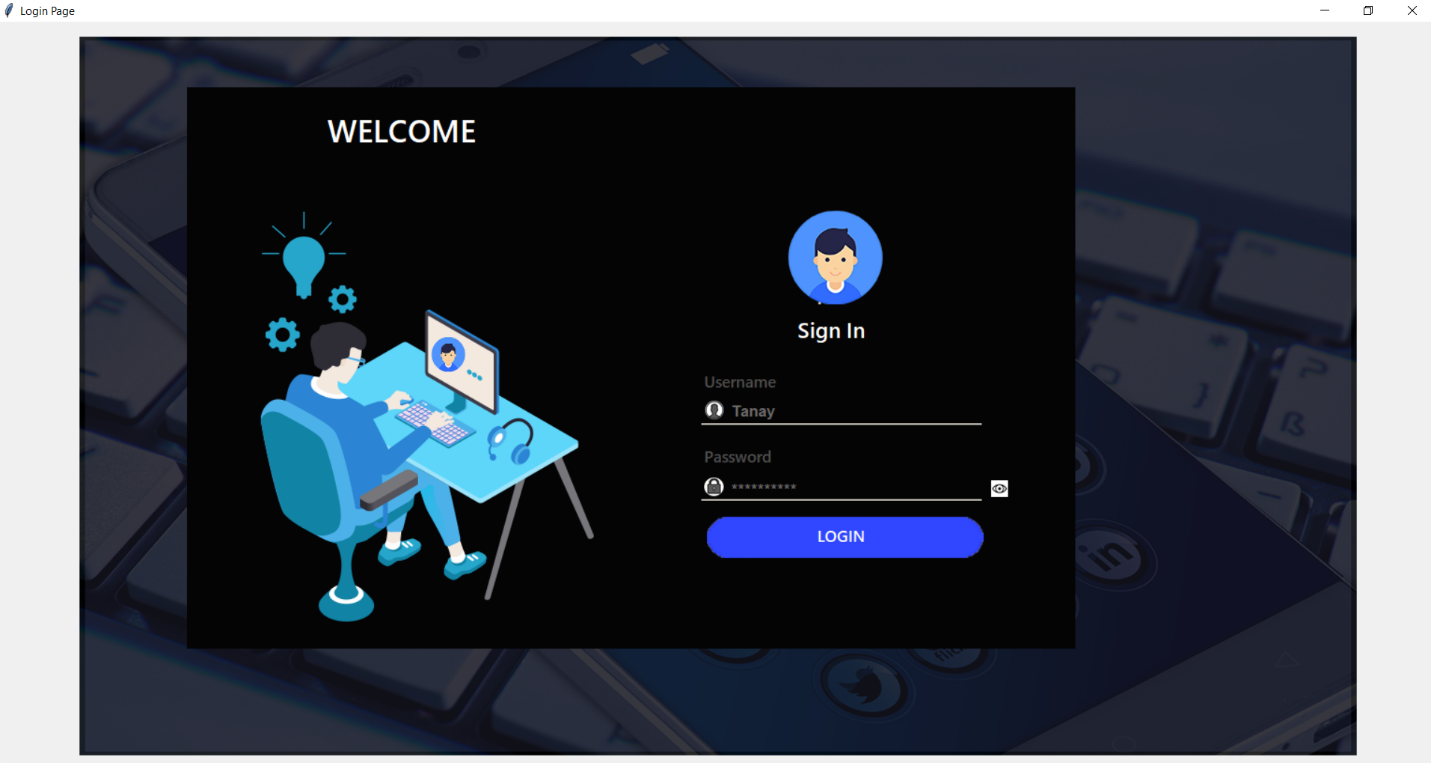
4. User Interface (UI) Testing: This sort of testing looks to see if the software's user interface is simple to use, intuitive, and reliable. You would examine if buttons, menus, forms, and other UI components behave as expected and meet your needs.

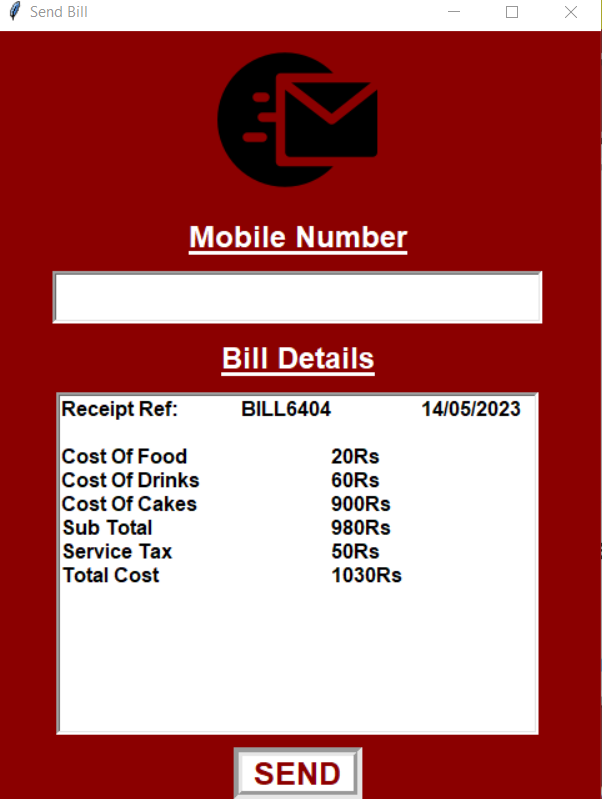
5. Performance testing: It involves determining how well the software performs and responds under various circumstances, such as managing a high volume of orders or concurrent users. You may test how the software manages a high number of concurrent orders or reservations for the restaurant management project.

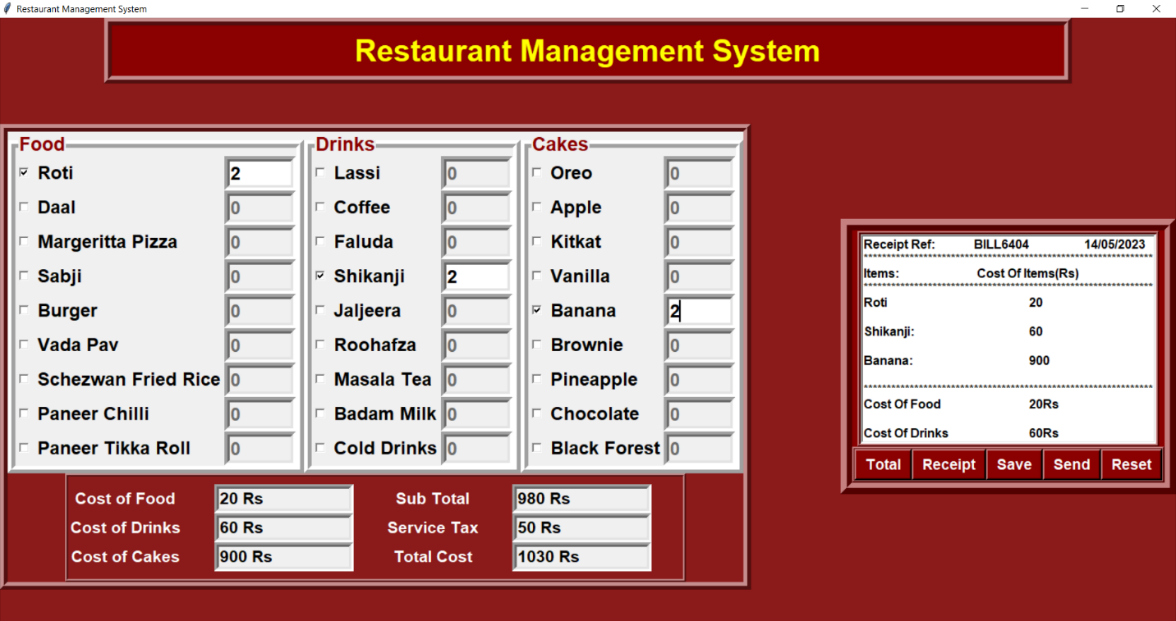
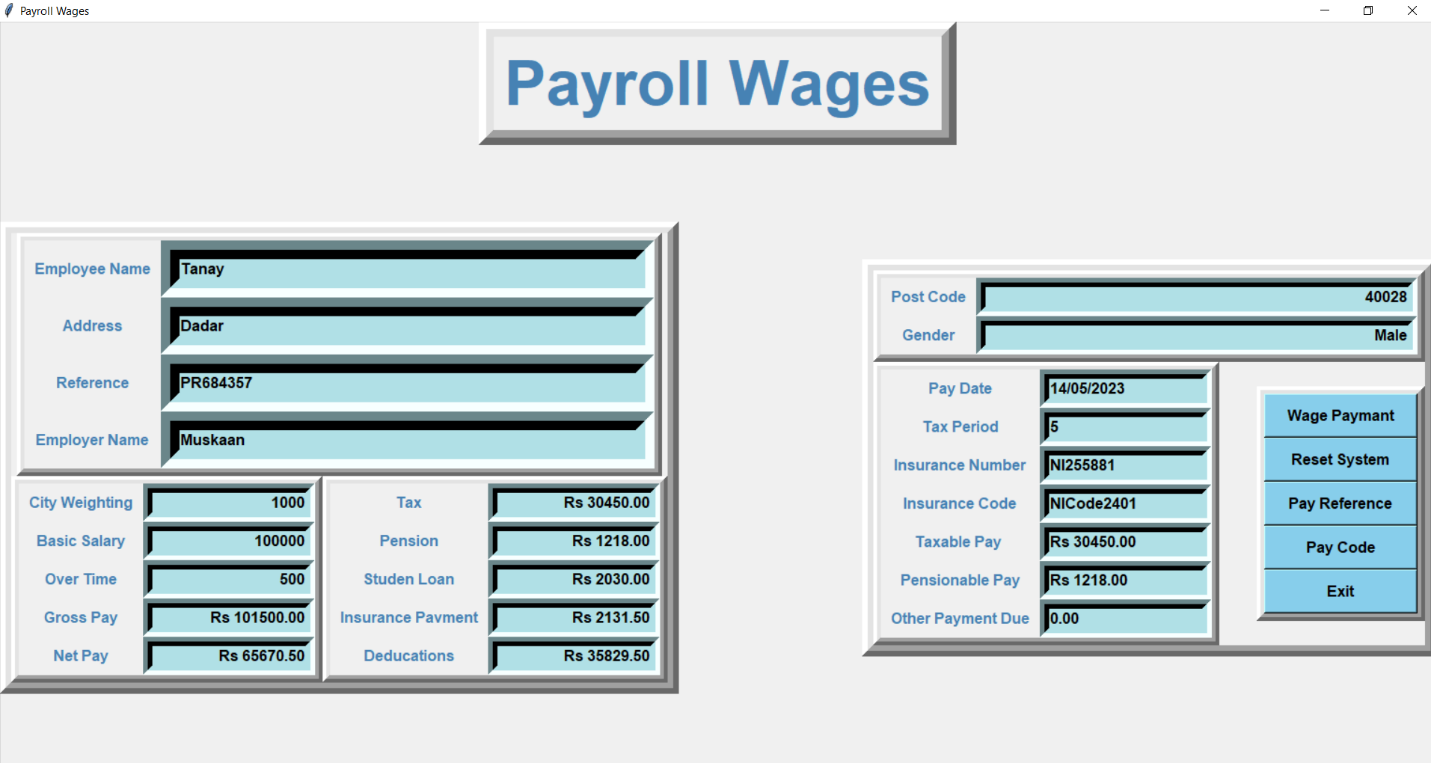
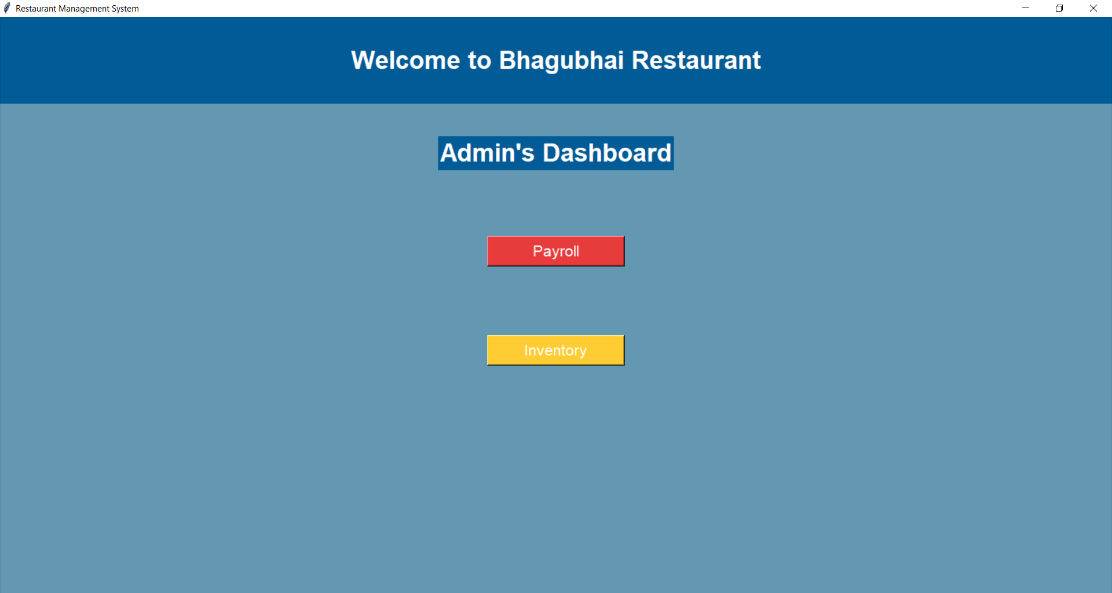
6. Security Testing: This category of testing is concerned with finding and fixing software security flaws. You may check, for instance, if the programme properly manages user authentication and authorisation, safeguards sensitive data, and restricts unauthorised access.

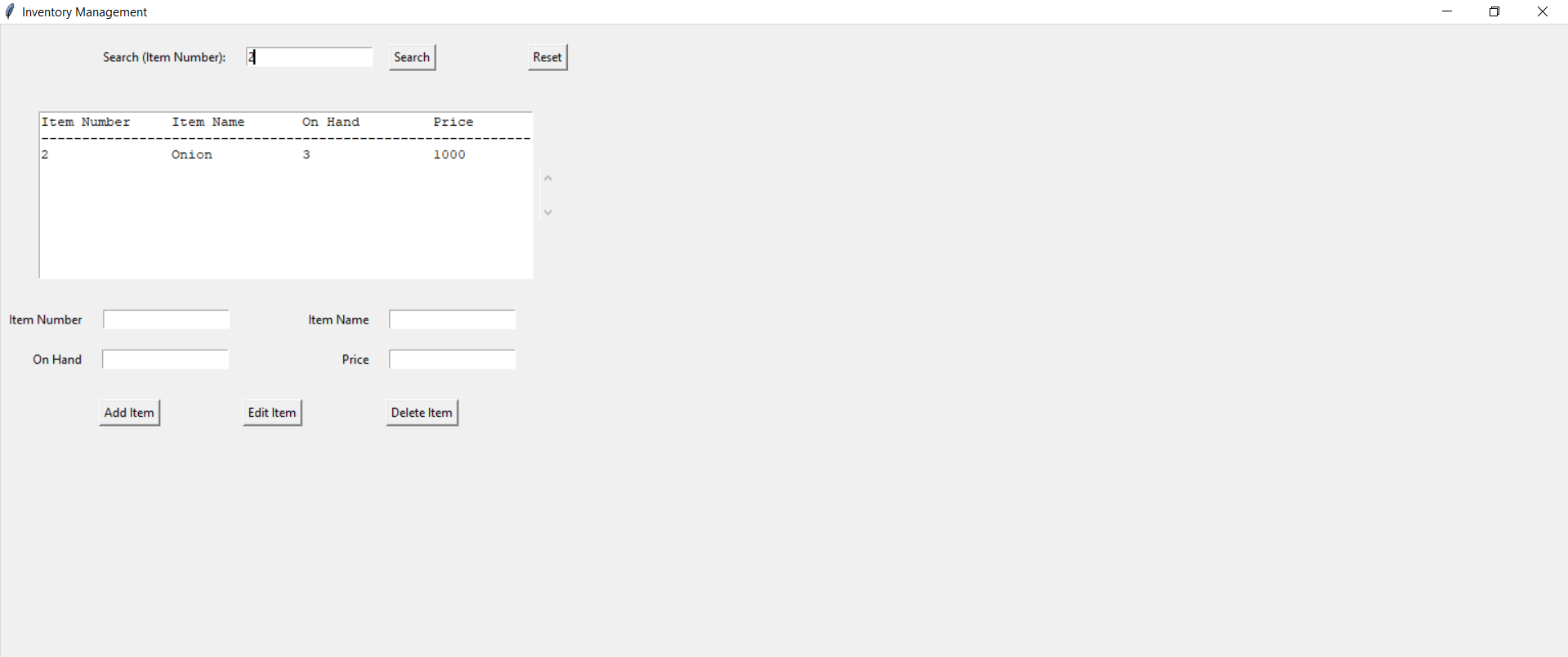
7. Regression testing: It is the process of retesting previously tested software components to make sure that subsequent additions or modifications haven't resulted in any new problems or broken functionality. This is essential each time you update.

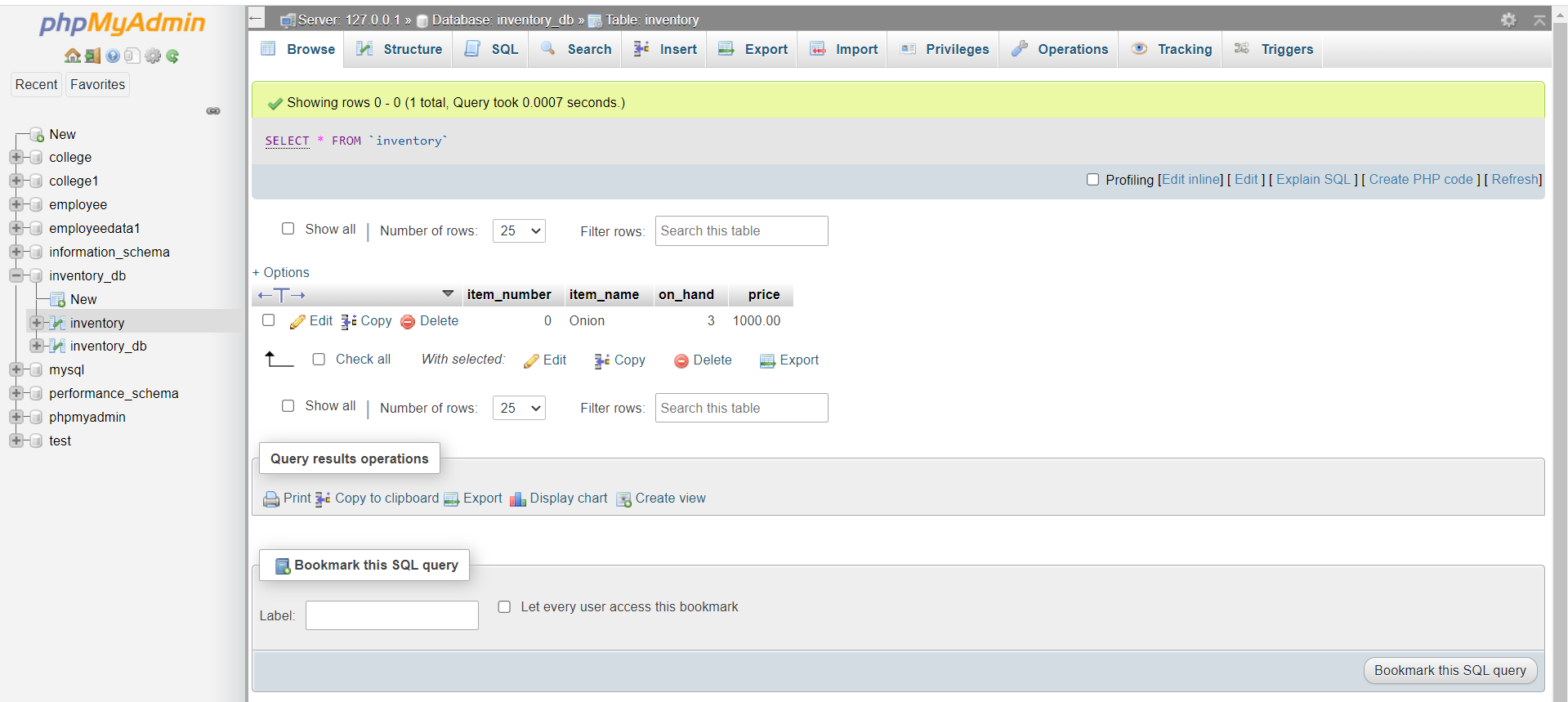
**RESULTS**

****

****

****

****

****

**FUTURE SCOPE**

The features that could not be added are array of objects and Multi-Threading. In the near future these concepts will be added in an appropriate way.

**CONCLUSION**

From this Mini Project I learned the various Python concept like Inheritance, Exception Handling, Packages, basic Class and objects, Basic Object Oriented Programming concepts, IO handling, File handling, GUI (Tkinter), Database Connectivity (MySQL) and String Functions. This helped me to strengthen the core Python concepts.

**REFERENCES**

1. <https://www.javatpoint.com/nlp-analysis-of-restaurant-reviews>
2. <https://www.geeksforgeeks.org/python-nlp-analysis-of-restaurant-reviews/>
3. <https://www.geeksforgeeks.org/inventory-management-with-json-in-python/>
4. <https://stackoverflow.com/questions/22492118/payroll-program-for-python-using-multiple-functions-and-return-function>
5. Python Essential Reference.