**RMS OF FASSOS**

**PROBLEM STATEMENT :-**

The Restaurant Management System is a comprehensive software application designed to streamline various operations within a restaurant establishment. Developed using Python programming language and the Tkinter library for creating the Graphical User Interface (GUI) and use Sql for Data management , this system aims to provide an efficient and user-friendly solution for managing orders, generating bills, tracking sales, and maintaining customer records.

**Project Description:-**

At the core of the application lies a robust user authentication system. Upon launching the program, users are prompted with a login window where they must enter valid credentials to gain access to the main application. This security measure ensures that only authorized personnel can interact with the system, protecting sensitive data and maintaining confidentiality.

Once authenticated, users are presented with the main application window, which is divided into three distinct sections. The header section displays the restaurant's name and the current date and time, providing a welcoming and informative interface. The menu section is the heart of the application, where users can input the desired quantities of various food items categorized into beverages, snacks, main courses, and desserts. This section allows for easy order placement and management, ensuring accurate tracking of customer preferences.

The control section is a versatile collection of buttons that provide access to various functionalities within the system. The "RESET" button allows users to clear all entered quantities and start a new order from scratch, ensuring a fresh slate for each customer. The "RATE CARD" button opens a separate window displaying a comprehensive list of available food items and their corresponding prices, enabling users to quickly reference pricing information.

One of the key features of the system is the ability to calculate the total cost of an order. By clicking the "TOTAL COST" button, the application meticulously computes the cost of each item based on the entered quantities and predefined rates. It then factors in applicable taxes and service charges, presenting the user with a detailed breakdown of the total bill amount.

When a customer is ready to settle the bill, the "GENERATE BILL" button initiates the billing process. The system prompts the user to enter the customer's name and phone number, ensuring accurate record-keeping and facilitating future communication with the customer if needed. Upon providing the customer details, a separate window is displayed, presenting a professionally formatted bill that includes the order details, item quantities, rates, taxes, service charges, and the total amount payable.

Notably, the application integrates with a SQLite database named "db\_saleReports.db" to store and retrieve critical data. This database houses tables for sales reports, order details, customer information, item rates, and user authentication credentials. As each bill is generated, the corresponding sales record is securely stored in the database, allowing for comprehensive sales reporting and analysis.

Speaking of sales reporting, the system offers a powerful visualization tool powered by the Matplotlib library. By clicking the "SALES REPORT" button, a new window opens, displaying a bar chart that graphically represents the sales figures for each customer. This visual representation provides valuable insights into sales patterns, customer preferences, and revenue generation, aiding in informed decision-making and strategic planning.

The application also incorporates various other functionalities to enhance the overall user experience. For instance, the "NAAN" button opens a separate window showcasing the available naan varieties and their corresponding prices, catering to the specific needs of the restaurant's menu offerings.

Throughout the development process, emphasis has been placed on code modularity and maintainability. The codebase is organized into functions and modules, each responsible for specific tasks such as cost calculation, bill generation, sales reporting, database operations, and GUI event handling. This modular approach not only enhances code readability and maintainability but also facilitates future enhancements and additions to the system.

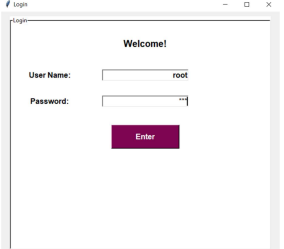
Furthermore, the Tkinter library has been instrumental in creating the visually appealing and intuitive GUI components, such as windows, frames, labels, entry fields, and buttons. The thoughtful layout and design of these components ensure a seamless and user-friendly experience, minimizing the learning curve for both staff and customers.

Overall, the Restaurant Management System is a robust and comprehensive solution tailored to the unique needs of the restaurant industry. By integrating order management, billing, sales reporting, and customer management into a single application, it streamlines operations, enhances efficiency, and provides valuable insights for informed decision-making. With its user-friendly interface, robust database integration, and modular codebase, this system empowers restaurant owners and staff to deliver exceptional service while maintaining a competitive edge in the ever-evolving hospitality landscape.

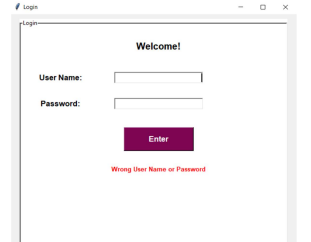
**FUNCTIONALITY AND MODULE :-**

**LOGIN SCREEN:**

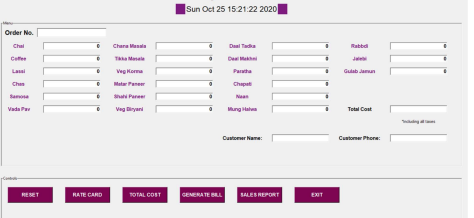
1. When user enters, correct user name and password, access is given to the main interface –



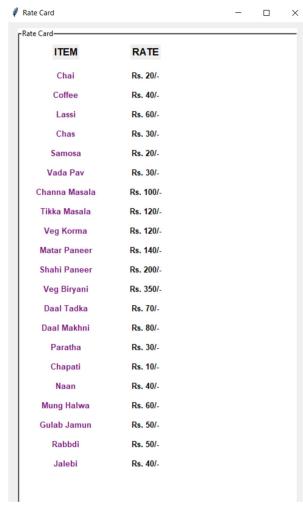
1. When user enters, wrong user name or password (proper message is displayed on the screen) –



Main interface-

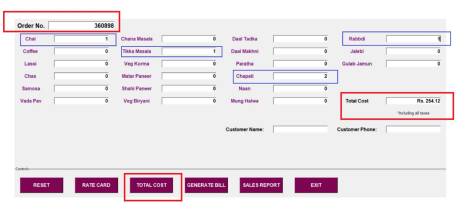


Rate card-



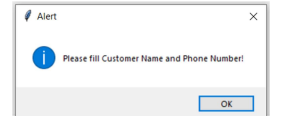
Total Cost Function (on clicking ‘TOTAL COST’, after entering meal quantity)

When user enters meal quantity and clicks on “TOTAL COST” button, then a unique order no. gets generated, along with that, the total cost of meal gets calculated and displayed on “Total Cost” entry box

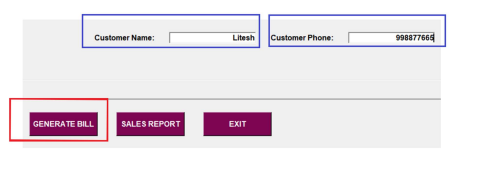


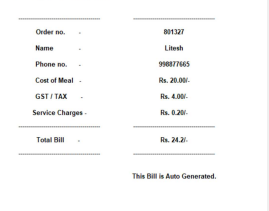
Message Box –

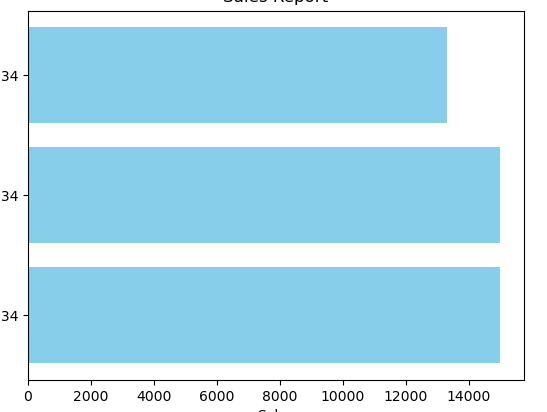
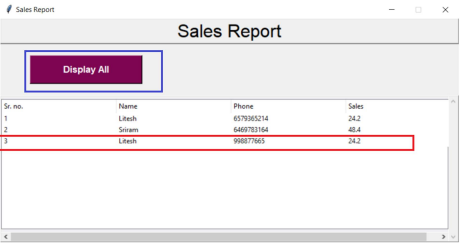
When user clicks on “Generate Bill” Button, without entering customer’s name and phone number then, message box gets displayed with proper error message.



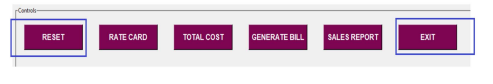
Bill Generator-





Sales Report-

Reset and exit button-



**Libraries/Packages/modules used – tkinter**

def: Tkinter library is used to create graphical user interface (GUI).

 from functools partial

def: Partial functions allow us to fix a certain number of arguments of a

function and generate a new function.

 from tkinter messagebox

def: The messagebox module is used to display the message boxes in the

python applications.

 ttk from tkinter.ttk

def: We are using this module in our project to create tree view.

 random

def: To generate random numbers.

 time

def: To get local time of the computer, and store it into variable.

 sqlite3

def: To connect and store records to the database. And to retrieve record

from database.

Functions used in the project:

 app()

def: app function is the main function in the program, which is having the

main source code, and this function is having other functions listed below:

 Cost()

def: This function is fetching values from entry boxes and calculating the

cost of meal, and after calculating cost, it returning those values.

 destroy()

def: This function is destroying the root window.

 reset()

def: This function is resetting the values of entry boxes.

 customer()

def: This function is fetching value from customer name and phone

number entry boxes, and displaying message if entry boxes is not filled.

 Database()

def: This function is making connection with database and creating table

if it does not exist.

 insert(cname, cphone, csales)

def: This function is taking three arguments and storing the values of

these arguments into the database.

 Rate()

def: This function is generating rate card (price card) in a new window.

 bill()

def: This function is collecting the values returned by Cost function and

using those values to generate bill in a separate window.

 report()

def: This function is creating separate window to display sales report.

 populateView()

def: This function is fetching values from the database and inserting

those values in tree view.

 validateLogin()

def: This function is validating login and calling app function (main

function) if entered user name and password is correct.