**CAPSTONE PROJECT REPORT**

(Project Term January-May 2023)

## *RESTAURANT MANAGEMENT SYSTEM*

Submitted by

**Jaswanth Singh Kumar Lankadasu Registration Number: 12104841**

**Project Group Number: K21UPA0 8**

**Course Code: INT216**

Under the Guidance of

**Waseem Ud Din Wani**

# School of Computer Science and Engineering



**PAC Form**

**DECLARATION**

We hereby declare that the project work entitled Restaurant Management System is an authentic record of my own work carried out as requirements of Capstone Project for the award of B. Tech degree in School of Computer Science and Engineering from Lovely Professional University, Phagwara, under the guidance of Waseem Ud Din Wani, during April to May 2023. All the information furnished in this capstone project report is based on my own intensive work and is genuine.

Project Group Number: K21UPA08

Name of Student: Jaswanth Singh Kumar Lankadasu

Registration Number: 12104841

Jaswanth Singh Kumar Lankadasu

Date: 30/04/2023

**CERTIFICATE**

This is to certify that the declaration statement made by the student is correct to the best of my knowledge and belief. He have completed this Capstone Project under my guidance and supervision. The present work is the result of their original investigation, effort, and study. No part of the work has ever been submitted for any other degree at any University. The Capstone Project is fit for the submission and partial fulfillment of the conditions for the award of B. Tech degree in School of Computer Science and Engineering from Lovely Professional University, Phagwara.

**Signature and Name of the Mentor**

**Designation**

**School of Computer Science and Engineering,**

Lovely Professional University,

Phagwara, Punjab.

Date:

**ACKNOWLEDGEMENT**

**TABLE OF CONTENTS**

Inner first page………………………………………………………………………. (i)

PAC form…………………………………………………………………………… (ii)

Declaration…...…………………………………………………………………….. (iii)

Certificate……..…….……………………………………………………………… (iv)

Acknowledgement…………………...…………………………………………....... (v)

Table of Contents……………………...…………………………………………… (vi)

**1. INTRODUCTION 1**

## *1.1. SECOND-LEVEL SUBHEADING 1*

## *1.2. ANOTHER SECOND-LEVEL SUBHEADING 1*

## *1.2.1. THIRD-LEVEL SUBHEADING 1*

# 1. INTRODUCTION

A Restaurant Management System is a software application that helps restaurant owners manage their business operations more efficiently. The system provides an easy-to-use interface for managing menu items, order information, among other things.

Restaurant Management System has the potential to greatly improve the efficiency and profitability of restaurants. By automating many routine tasks, your software can help restaurant owners to save time and reduce costs. Moreover, the system can provide valuable insights into customer behaviour, helping restaurant owners to optimize their menus, prices, and marketing strategies.

## 1.1. SECOND-LEVEL SUBHEADING

This is the second-level subheading of the first section. The second-level subheading uses the formatting style Heading 2. It is left aligned, boldface and single spaced, and it advances the text after it by one line. Second level subheadings are in Title Case (The first letters of principal words must be capitalized).

## 1.2. ANOTHER SECOND-LEVEL SUBHEADING

The format of this subheading is the same with the first one. The purpose of this subheading is to show you that if you have a subheading of a certain level, you must have more than one. The rationale is that you cannot have a list of only one item.

## 1.2.1. THIRD-LEVEL SUBHEADING

The third-level subheading uses the formatting style Heading 3. It uses the same formatting with the second-level subheading except that for the third-level subheading, only the first letter of the first word and proper nouns are capitalized (Sentence case).

### 1.2.2. ANOTHER THIRD-LEVEL SUBHEADING

cannot have a list of only one shows that if you have a subheading of a certain level, you must have more than one. The rationale is that you cannot have a list of only one item.

# 2. PROFILE OF THE PROBLEM

The restaurant industry is a highly competitive sector, and in order to remain successful, restaurant owners need to focus on improving customer experience and streamlining their operations. However, traditional methods of managing restaurant orders and billing can be inefficient and prone to errors, leading to longer wait times and decreased customer satisfaction. Furthermore, manual record-keeping can be time-consuming and difficult to manage, making it challenging for restaurant owners to track their finances and make informed decisions about their business.

The development of a restaurant management system can address these challenges by automating the ordering and billing process, improving accuracy, and reducing wait times. Additionally, a well-designed restaurant management system can provide restaurant owners with real-time data about their business.

The scope of this study is to design and develop a restaurant management system that meets the specific needs of restaurant owners and their customers. The system should be user-friendly, reliable, and secure, providing a simple and efficient way for customers to place orders and receive bills, and for restaurant.

# 3. EXISTING SYSTEM

## 3.1. INTRODUCTION

The existing system for managing restaurant orders and billing is typically a manual process involving handwritten orders, physical menus, and paper receipts. This process is prone to errors and can result in delays, which can negatively impact the customer experience.

## 3.2. EXISTING SOFTWARE

There are many restaurant management software available in the market, but they may not be suitable for all types of restaurants. For example, some software may be too expensive or too complex for small-scale restaurants. Additionally, many of these software may have features that are not required by a particular restaurant, which can result in unnecessary costs and complexity.

## 3.3. DFD FOR PRESENT SYSTEM

A data flow diagram (DFD) for the present system can be created to illustrate the flow of data and processes involved in the traditional method of managing orders and billing. The DFD can help identify inefficiencies in the existing system and provide insights into how the new system can be designed.

## 3.4. WHAT'S NEW IN THE SYSTEM TO BE DEVELOPED

The new restaurant management system will be a software-based solution that automates the process of order taking, food preparation, and billing. The new system will offer several advantages over the existing system, including:

* Improved accuracy: The system will reduce errors in the order taking and billing process, which can lead to more accurate bills and fewer customer complaints.
* Faster service: The system will streamline the ordering and billing process, resulting in faster service and reduced waiting times for customers.
* Better inventory management: The system will enable restaurant owners to track inventory levels in real-time, which can help them optimize their ordering and reduce waste.
* Enhanced customer experience: The system will provide a more modern and convenient ordering experience for customers, which can result in higher customer satisfaction and repeat business.

# 4. PROBLEM ANALYSIS

## 4.1. PRODUCT DEFINITION

The product definition for your restaurant management system includes the following features:

* A user-friendly interface that enables customers to place their orders easily.
* A database to store menu items.
* A billing system that calculates the total cost of the order and generates a bill for the customer.
* A reset button that enables the user to clear the order list and start over.
* A close button that allows the user to exit the application.
* Error handling and validation checks to prevent input errors and ensure accuracy..

## 4.2. FEASIBILITY ANALYSIS

A feasibility analysis is a crucial step in determining the viability of the project. This analysis assesses the technical, operational, and economic feasibility of the project. Technical feasibility evaluates whether the required technology is available and can be implemented effectively. Operational feasibility assesses whether the proposed system can be integrated with existing systems and whether it will be accepted by the users. Economic feasibility determines whether the project is financially viable and whether it can generate a sufficient return on investment.

The feasibility analysis for your restaurant management system project includes the following considerations:

* Technical feasibility: Can the system be developed with the available resources and technology (Tkinter and Python)?
* Operational feasibility: Will the system be easy to use and integrate with existing restaurant processes?
* Economic feasibility: Is the project cost-effective and will the benefits outweigh the costs?

## *4.3. PROJECT PLAN*

The project plan outlines the scope, timeline, resources, and deliverables of the project. This plan includes a detailed breakdown of the project tasks, timelines, dependencies, and milestones. It also identifies the roles and responsibilities of the project team and outlines the communication and risk management plan.

# 5. SOFTWARE REQUIREMENT ANALYSIS

## 5.1. INTRODUCITON

The software requirement analysis involves identifying the hardware and software requirements of the system.

## 5.1.1. Hardware Requirements

Any system with minimum of 4GB RAM and 128GB ROM which can run at least VS Code or Python or any other applications that can able to run the python code freely.

## 5.1.2. Software Requirements

Any software that is capable of running the python application is fine. Any OS can be fine.

## 5.2. GENERAL DESCRIPTION

The restaurant management system is designed to provide an easy-to-use interface for restaurant customers and staff. The system will allow customers to view the menu, place orders, and receive bills.

## 5.3. SPECIFIC REQUIREMENTS

The specific requirements outline the functional and non-functional requirements of the system, such as the user interface, database management, and security features.

## 5.3.1. User Interface

The user interface of the system should be intuitive and easy to use for both customers and staff. The system should be able to display the menu items in a visually appealing way, with images and descriptions of each item. Customers should be able to add items to their order, view their order details, and receive their bills with ease.

## 5.3.2. Security

The system should be secure and protect sensitive data such as customer information, order details, and payment information.

## 5.3.3. Compatibility

The system should be compatible with different hardware and software configurations. The system should be able to run on different operating systems and browsers and should be compatible with different screen sizes and resolutions.

# 6. DESIGN

## 6.1. SYSTEM DESIGN

The system design involves creating a high-level architecture and defining the modules and components of the system.

## 6.2. DESIGN NOTATIONS

Design notations, such as flowcharts and UML diagrams, can be used to illustrate the system design.

## 6.3. DETAILED DESIGN

The detailed design involves creating a low-level design of the system, including the algorithms, data structures, and programming languages to be used.

## 6.4. FLOWCHARTS

## 6.5. PSEUDO CODE

**7. TESTING**

## 7.1. FUNCTIONAL TESTING

## 7.2. STRUCTURAL TESTING

## 7.3. LEVELS OF TESTING

## 7.4. TESTING THE PROJECT

# 8. IMPLEMENTATION

## 8.1. IMPLEMENTATION OF THE PROJECT

## 8.2. CONVERSION PLAN

## 8.3. POST-IMPLEMENTATION AND SOFTWARE MAINTENANCE

# 9. PROJECT LEGACY

## 9.1. CURRENT STATUS OF THE PROJECT

## 9.2. REMAINING AREAS OF CONCERN

## 9.3. TECHNICAL AND MANGERIAL LESSONS LEARNT

# 10. USER MANUAL

# 11. SOURCE CODE

from tkinter import\*

from tkinter import messagebox

from PIL import ImageTk, Image

from tkinter import ttk # ttk(the tree view) is used to create the tables

window = Tk()

# -------------------- window title --------------------

window.title("Restaurant Management System")

# -------------------- display geomentry --------------------

window.geometry("1400x800")

# --------------- Data ---------------------

breakfast\_menu = [

['idli',30],

['dosa',40],

['poha',40],

['poori',40],

['upma',30],

['paneer\_paratha',50]]

lunch\_menu = [

['rice',40],

['roti',20],

['chicken',120],

['paneer',80],

['mutton',100],

['biryani',120]]

snacks\_menu = [

['samosa',15],

['panipuri',20],

['chaat',20],

['momos',50],

['thukpa',30],

['sandal',20]

]

sb = [] # used to store the values of the spinbox values

# ------------------ Table ------------------------

# treeview

table = ttk.Treeview(window, columns=('items','price','qty','total\_price'), show="headings") # show="heading" shows the headings inleft side

# setting col width

table.column("items", width = 100)

table.column("qty", width = 80)

table.column("price", width =80 )

table.column("total\_price", width = 80)

table.heading('items', text='Items')

table.heading('qty',text="Qty")

table.heading('price', text='Price' )

table.heading('total\_price',text='Total' )

table.place(x=900,y=100)

# -------------------- label to display Name --------------------

name = Label(window, text="Restaurant Management System By Jaswanth", font=("arial","26"), fg="Green")

name.place(x=450,y=10)

# -------------------- Radiobutton --------------------

#variables

v=IntVar()

#function for radio button

item1\_breakfast = Image.open("imgs/idli.png")

item1\_breakfast = item1\_breakfast.resize((150,150))

idli = ImageTk.PhotoImage(item1\_breakfast)

item1 = Button(window, image=idli)

item1.place(x=100,y=150)

item2\_breakfast = Image.open("imgs/dosa.png")

item2\_breakfast = item2\_breakfast.resize((150,150))

dosa = ImageTk.PhotoImage(item2\_breakfast)

item2 = Button(window, image=dosa)

item2.place(x=300,y=150)

item3\_breakfast = Image.open("imgs/poha.png")

item3\_breakfast = item3\_breakfast.resize((150,150))

poha = ImageTk.PhotoImage(item3\_breakfast)

item3 = Button(window, image=poha)

item3.place(x=500,y=150)

item4\_breakfast = Image.open("imgs/poori.png")

item4\_breakfast = item4\_breakfast.resize((150,150))

poori = ImageTk.PhotoImage(item4\_breakfast)

item4 = Button(window, image=poori)

item4.place(x=100,y=430)

item5\_breakfast = Image.open("imgs/upma.png")

item5\_breakfast = item5\_breakfast.resize((150,150))

upma = ImageTk.PhotoImage(item5\_breakfast)

item5 = Button(window, image=upma)

item5.place(x=300,y=430)

item6\_breakfast = Image.open("imgs/paneer\_paratha.png")

item6\_breakfast = item6\_breakfast.resize((150,150))

paneer\_paratha = ImageTk.PhotoImage(item6\_breakfast)

item6 = Button(window, image=paneer\_paratha)

item6.place(x=500,y=430)

# # lunch images

iteam1\_lunch = Image.open("imgs/rice.png")

iteam1\_lunch = iteam1\_lunch.resize((150,150))

rice = ImageTk.PhotoImage(iteam1\_lunch)

iteam2\_lunch = Image.open("imgs/roti.jpeg")

iteam2\_lunch = iteam2\_lunch.resize((150,150))

roti = ImageTk.PhotoImage(iteam2\_lunch)

iteam3\_lunch = Image.open("imgs/chicken.jpeg")

iteam3\_lunch = iteam3\_lunch.resize((150,150))

chicken = ImageTk.PhotoImage(iteam3\_lunch)

iteam4\_lunch = Image.open("imgs/Paneer.jpeg")

iteam4\_lunch = iteam4\_lunch.resize((150,150))

Panner = ImageTk.PhotoImage(iteam4\_lunch)

iteam5\_lunch = Image.open("imgs/mutton.jpeg")

iteam5\_lunch = iteam5\_lunch.resize((150,150))

mutton = ImageTk.PhotoImage(iteam5\_lunch)

iteam6\_lunch = Image.open("imgs/Biryani.jpeg")

iteam6\_lunch = iteam6\_lunch.resize((150,150))

Biryani = ImageTk.PhotoImage(iteam6\_lunch)

# snacks images

iteam1\_snacks = Image.open("imgs/samosa.png")

iteam1\_snacks = iteam1\_snacks.resize((150,150))

samosa = ImageTk.PhotoImage(iteam1\_snacks)

iteam2\_snacks = Image.open("imgs/panipuri.png")

iteam2\_snacks = iteam2\_snacks.resize((150,150))

panipuri = ImageTk.PhotoImage(iteam2\_snacks)

iteam3\_snacks = Image.open("imgs/chaat.jpeg")

iteam3\_snacks = iteam3\_snacks.resize((150,150))

chaat = ImageTk.PhotoImage(iteam3\_snacks)

iteam4\_snacks = Image.open("imgs/momos.jpeg")

iteam4\_snacks = iteam4\_snacks.resize((150,150))

momos = ImageTk.PhotoImage(iteam4\_snacks)

iteam5\_snacks = Image.open("imgs/thukpa.png")

iteam5\_snacks = iteam5\_snacks.resize((150,150))

thukpa = ImageTk.PhotoImage(iteam5\_snacks)

iteam6\_snacks = Image.open("imgs/sandal.jpeg")

iteam6\_snacks = iteam6\_snacks.resize((150,150))

sandal = ImageTk.PhotoImage(iteam6\_snacks)

# label or food items names

item1\_name = Label(window, text='Idli - Rs.30', font=("",16))

item1\_name.place(x=110,y=310)

item2\_name = Label(window, text='Dosa - Rs.40', font=("",16))

item2\_name.place(x=310,y=310)

item3\_name = Label(window, text='Poha - Rs.40', font=("",16))

item3\_name.place(x=510,y=310)

item4\_name = Label(window, text='Poori - Rs.40', font=("",16))

item4\_name.place(x=110,y=590)

item5\_name = Label(window, text='Upma - Rs.30', font=("",16))

item5\_name.place(x=310,y=590)

item6\_name = Label(window, text='Paneer Paratha - Rs.50', font=("",16))

item6\_name.place(x=510,y=590)

def show\_items(val):

if (val == 1):

# buttons to display items of breakfast

item1["image"] = idli

item2["image"] = dosa

item3["image"] = poha

item4["image"] = poori

item5["image"] = upma

item6["image"] = paneer\_paratha

item1\_name['text'] ='Idli - Rs.30'

item2\_name['text'] ='Dosa - Rs.40'

item3\_name['text'] ='Poha - Rs.40'

item4\_name['text'] ='Poori - Rs.40'

item5\_name['text'] ='Upma - Rs.30'

item6\_name['text'] ='Paneer Paratha - Rs.50'

elif (val == 2):

# buttons to display items of breakfast

item1["image"] = rice

item2["image"] = roti

item3["image"] = chicken

item4["image"] = Panner

item5["image"] = mutton

item6["image"] = Biryani

item1\_name['text'] ='Rice - Rs.40'

item2\_name['text'] ='Roti - Rs.20'

item3\_name['text'] ='Chicken - Rs.120'

item4\_name['text'] ='Panner - Rs.80'

item5\_name['text'] ='Mutton - Rs.100'

item6\_name['text'] ='Biryani - Rs.120'

elif (val == 3):

# buttons to display items of breakfast

item1["image"] = samosa

item2["image"] = panipuri

item3["image"] = chaat

item4["image"] = momos

item5["image"] = thukpa

item6["image"] = sandal

item1\_name['text'] ='Samosa - Rs.15'

item2\_name['text'] ='Panipuri - Rs.20'

item3\_name['text'] ='Chaat - Rs.20'

item4\_name['text'] ='Momos - Rs.50'

item5\_name['text'] ='Thukpa - Rs.30'

item6\_name['text'] ='Sandal - Rs.20'

sb\_reset()

# for breakfast

breakfast = Radiobutton(window, text='Breakfast', value=0, variable=v, font=('Times',20,'normal'), command=lambda: show\_items(1))

breakfast.place(x=120,y=70)

# for lunch

lunch = Radiobutton(window, text='Lunch/Dinner', value=1, variable=v,font=('Times',20,'normal'), command=lambda: show\_items(2))

lunch.place(x=310,y=70)

# for dinner

dinner = Radiobutton(window, text='Snacks', value=2, variable=v,font=('Times',20,'normal'), command=lambda: show\_items(3))

dinner.place(x=520,y=70)

# -------------- Spinbox to get the items -----------------

sb1 = Spinbox(window,from\_=0,to\_=10,font=('Times',20,'normal'),width=2)

sb1.place(x=150,y=350)

sb.append(sb1)

sb2 = Spinbox(window,from\_=0,to\_=10,font=('Times',20,'normal'),width=2)

sb2.place(x=350,y=350)

sb.append(sb2)

sb3 = Spinbox(window,from\_=0,to\_=10,font=('Times',20,'normal'),width=2)

sb3.place(x=550,y=350)

sb.append(sb3)

sb4 = Spinbox(window,from\_=0,to\_=10,font=('Times',20,'normal'),width=2)

sb4.place(x=150,y=630)

sb.append(sb4)

sb5 = Spinbox(window,from\_=0,to\_=10,font=('Times',20,'normal'),width=2)

sb5.place(x=350,y=630)

sb.append(sb5)

sb6 = Spinbox(window,from\_=0,to\_=10,font=('Times',20,'normal'),width=2)

sb6.place(x=550,y=630)

sb.append(sb6)

# funtion to reset the spinbox values

def sb\_reset():

for i in range(7):

sb[i].config(textvariable = IntVar(value=0))

# ------------------ Add Bill --------------------

total = 0

final\_total = 0

# funtion to add the bills

def add\_bill():

global v, total, table

if v.get() == 0:

for i in range(6):

if int(sb[i].get()) != 0:

table.insert(parent='',index=END, values=(breakfast\_menu[i][0], breakfast\_menu[i][1], sb[i].get(), (int(sb[i].get())\*breakfast\_menu[i][1])))

total += (int(sb[i].get())\*breakfast\_menu[i][1])

if v.get() == 1:

for i in range(6):

if int(sb[i].get()) != 0:

table.insert(parent='',index=END, values=(lunch\_menu[i][0], lunch\_menu[i][1], sb[i].get(), (int(sb[i].get())\*lunch\_menu[i][1])))

total += (int(sb[i].get())\*lunch\_menu[i][1])

if v.get() == 2:

for i in range(6):

if int(sb[i].get()) != 0:

table.insert(parent='',index=END, values=(snacks\_menu[i][0], snacks\_menu[i][1], sb[i].get(), (int(sb[i].get())\*snacks\_menu[i][1])))

total += (int(sb[i].get())\*snacks\_menu[i][1])

sb\_reset()

addbill = Button(window, text="Add Bill", font=("",20), command=add\_bill)

addbill.place(x=100,y=700)

# -------------------- Get Bill --------------------

# funtion to cal total amount

cost = Label(window, text="", font=("Times",24))

tax = Label(window, text="", font=("Times",24))

total\_cost = Label(window, text="", font=("Times",24))

cost\_display = Label(window, text="", font=("Times",24))

tax\_display = Label(window, text="", font=("Times",24))

total\_cost\_display = Label(window, text="", font=("Times",24))

def final\_bill():

global table, final\_total, total

# ------------- labels to show the total cost -----------------

# label to show the Cost

cost['text'] = "Cost : "

cost.place(x=900,y=400)

cost\_display['text'] = total

cost\_display.place(x=1000,y=400)

# label to show the tax %

tax["text"] = "Tax % : "

tax.place(x=900,y=450)

tax\_display['text'] = '10%'

tax\_display.place(x=1000,y=450)

# label to show the total cost

total\_cost['text']="Total Cost : "

total\_cost.place(x=900,y=500)

final\_total = total+(total/10)

total\_cost\_display['text'] = final\_total

total\_cost\_display.place(x=1050,y=500)

getbill = Button(window, text="Get Bill", font=("",20), command=final\_bill)

getbill.place(x=300,y=700)

# -------------------- Reset --------------------

# function of reset

def resetbill():

global total, final\_total

messagebox.showinfo("Total Cost",f"Total Cost : {final\_total}\nSee you again\n Have a nice Day :)")

total = 0

final\_total = 0

for item in table.get\_children():

table.delete(item)

cost['text'] = ''

tax['text'] = ''

total\_cost['text'] = ''

cost\_display['text'] = ''

tax\_display['text'] = ''

total\_cost\_display['text'] = ''

reset\_bill = Button(window, text="Reset Bill", font=("",20), command=resetbill)

reset\_bill.place(x=500, y=700)

# -------------------- Exit --------------------

# funtion to close the window

def close\_window():

window.after(100,window.destroy)

exit\_window = Button(window, text="Close", font=("",20), command=close\_window)

exit\_window.place(x=1000,y=700)

# -------------------- Remove ------------------

# function to remove item

def delete\_items():

global total

for i in table.selection():

total -= table.item(i)['values'][3]

table.delete(i)

cost\_display['text'] = total

final\_total = total+(total/10)

total\_cost\_display['text'] = final\_total

table.bind('<Delete>',delete\_items)

remove\_item = Button(window, text="Remove Item" , font=("",20), command=delete\_items)

remove\_item.place(x=980,y=340)

window.mainloop()

# 12. BIBLIOGRAPHY