# CHAPTER 1: INTRODUCTION

In the rapidly evolving landscape of the hospitality industry, efficient management systems are paramount for the success and sustainability of restaurants. With the emergence of new technologies and changing consumer preferences, the need for robust restaurant management solutions has never been greater. This chapter introduces "Centurion Coffee Connect Restaurant Management System," a comprehensive software solution aimed at revolutionizing the way restaurants operate and deliver exceptional dining experiences to their patrons.

## **1.1 Background**

The restaurant industry is known for its fast-paced environment, dynamic customer demands, and complex operational challenges. From managing orders and inventory to ensuring seamless customer service, restaurant owners and managers juggle multiple tasks simultaneously. Traditional methods of manual management are no longer sufficient to meet the demands of today's competitive market. There is a pressing need for innovative technologies that can streamline operations, improve efficiency, and enhance overall performance.

## **Purpose**

The purpose of this document is to introduce "Centurion Coffee Connect Restaurant Management System" as a solution to the challenges faced by restaurants in managing their operations effectively. This system is designed to provide a holistic approach to restaurant management, encompassing various aspects such as order processing, inventory control, table management, billing, employee scheduling, and customer relationship management. By offering a comprehensive set of features and functionalities, this system aims to empower restaurant owners and managers with the tools they need to succeed in today's competitive market.

## **Objectives**

The objectives of "Centurion Coffee Connect Restaurant Management System" are as follows:

* To streamline restaurant operations by automating repetitive tasks and minimizing manual intervention.
* To optimize inventory management and control costs by tracking stock levels, monitoring ingredient usage, and generating alerts for low stock items.
* To enhance customer service and satisfaction through efficient order processing, table management, and personalized experiences.
* To provide actionable insights and analytics to restaurant owners and managers, enabling data-driven decision-making and business growth.
* To improve employee productivity and satisfaction by simplifying scheduling, task assignment, and performance tracking.

**1.4 Scope**

The scope of "Centurion Coffee Connect Restaurant Management System" encompasses the following:

* Order management: From order placement to fulfillment, including dine-in, takeout, and delivery orders.
* Inventory control: Tracking stock levels, managing ingredient usage, and generating alerts for low stock items.
* Table management: Efficient allocation of tables, reservation management, and real-time tracking of table status.
* Billing and payments: Simplified billing processes, support for various payment methods, and integration with payment gateways.
* Employee management: Scheduling shifts, managing staff roles, tracking performance metrics, and handling payroll.
* Analytics and reporting: Generating reports on sales trends, expenses, and key performance indicators for informed decision-making.
* Customer relationship management (CRM): Capturing and analyzing customer data, implementing loyalty programs, and gathering feedback to enhance customer satisfaction.

**1.5 Organization of the Document**

This document is organized into several chapters, each focusing on different aspects of "Centurion Coffee Connect Restaurant Management System." Chapter 2 provides an overview of the system architecture and technical specifications. Chapter 3 delves into the features and functionalities of the system in detail. Subsequent chapters cover topics such as implementation, integration, maintenance, and future enhancements. Through this structured approach, readers will gain a comprehensive understanding of the capabilities and benefits of the system. In conclusion, "Centurion Coffee Connect Restaurant Management System" represents a paradigm shift in restaurant management, offering a holistic and integrated solution to meet the diverse needs of modern restaurants. With its innovative features, user-friendly interface, and scalability, this system is poised to revolutionize the way restaurants operate and thrive in today's competitive market.

# CHAPTER 2: RELATED WORK

The landscape of restaurant management systems has been significantly shaped by previous research, industry practices, and technological advancements. This chapter provides a comprehensive literature survey, exploring key concepts, trends, and findings relevant to restaurant management systems.

**2.1 Evolution of Restaurant Management Systems**

The evolution of restaurant management systems can be traced back to the emergence of electronic point-of-sale (POS) systems in the 1970s. Early POS systems focused primarily on processing transactions and inventory management. Over time, these systems evolved to incorporate additional functionalities such as order management, employee scheduling, and customer relationship management.

**2.2 Key Features of Restaurant Management Systems**

A review of existing literature reveals several key features that are commonly found in restaurant management systems:

1.Order Management: Efficient processing of orders, including dine-in, takeout, and delivery orders.

2.Inventory Control: Tracking stock levels, managing ingredient usage, and minimizing waste.

3.Table Management: Optimizing seating capacity, managing reservations, and tracking table status.

3.Billing and Payments: Streamlining billing processes, supporting various payment methods, and ensuring accuracy in transactions.

4.Employee Management: Scheduling shifts, managing staff roles, tracking performance metrics, and handling payroll.

5.Analytics and Reporting: Generating reports on sales trends, expenses, and key performance indicators for informed decision-making.

6.Customer Relationship Management (CRM): Capturing and analyzing customer data, implementing loyalty programs, and gathering feedback to enhance customer satisfaction.

**2.3 Emerging Trends in Restaurant Management Systems**

Recent literature highlights several emerging trends and innovations in restaurant management systems:

1.Mobile Integration: The proliferation of smartphones has led to increased demand for mobile-friendly solutions that enable customers to place orders, make reservations, and provide feedback directly from their mobile devices.

2.Cloud-Based Solutions: Cloud-based restaurant management systems offer scalability, flexibility, and accessibility, allowing restaurant owners to manage their operations remotely and adapt to changing business needs.

3.Integration with Third-Party Services: Integration with third-party services such as delivery platforms, online reservation systems, and loyalty programs enhances the functionality and value proposition of restaurant management systems.

4.Artificial Intelligence (AI) and Machine Learning: AI-powered features such as predictive analytics, personalized recommendations, and automated customer support are increasingly being integrated into restaurant management systems to improve efficiency and enhance customer experiences.

5.Blockchain Technology: Blockchain technology is being explored for its potential to enhance security, transparency, and traceability in restaurant supply chains, inventory management, and payment processing.

**2.4 Challenges and Opportunities**

While restaurant management systems offer numerous benefits, they also present challenges and opportunities for improvement. Common challenges include:

1.Integration Complexity: Integrating various modules and third-party services into a cohesive system can be complex and time-consuming.

User Adoption: Ensuring that restaurant staff are adequately trained and comfortable using the system is essential for maximizing its effectiveness.

2.Data Security: Protecting sensitive customer data and ensuring compliance with data privacy regulations are critical considerations for restaurant owners and managers.

3.Despite these challenges, restaurant management systems present significant opportunities for enhancing operational efficiency, improving customer satisfaction, and driving business growth.

# CHAPTER 3: OVERALL DESCRIPTION

Centurion Coffee Connect Restaurant Management System documentation provides a comprehensive overview of the system's architecture, functionalities, and user interfaces. This chapter aims to equip restaurant owners, managers, and developers with a holistic understanding of how the system operates, its key components, and the user journey through various modules.

**3.1 Proposed System**

* Order Management
* Table Management
* Inventory Management
* Employee Management
* Customer Relationship Management (CRM)
* Reporting and Analytics

**3.1.1 Functional Requirement:**

**User Authentication and Authorization:**

* Users should be able to log in with different roles such as admin, manager, staff, and customer.
* Each role should have specific permissions, e.g., only admins can access inventory management features.

**Menu Management:**

* Admins should be able to add, edit, and delete menu items.
* Each menu item should include details like name, description, price, and ingredients.

**Order Management:**

* Customers should be able to browse the menu, select items, and place orders.
* Staff should receive real-time notifications of new orders and mark them as processed when complete.

**Table Management:**

* Staff should be able to view table availability, assign tables to customers, and mark them as occupied or vacant.
* Customers should be able to make reservations online, with confirmation sent via email or SMS.

**Inventory Management:**

* Admins should be able to view current stock levels, add new inventory, and update existing stock quantities.
* Automatic alerts should notify staff when inventory levels are low.

**Employee Management:**

* Admins should be able to add, edit, and delete employee accounts.
* Staff should be able to view their schedules, clock in/out for shifts, and request time off.

**Customer Relationship Management (CRM):**

* The system should maintain a database of customer profiles, including contact information, order history, and preferences.
* Feedback forms should be available for customers to submit comments and ratings.

**Reporting and Analytics:**

* Admins should have access to various reports, including sales summaries, inventory usage, and employee performance.

**3.1.2 Non Functional Requirements:**

**Performance:** The system should respond quickly to user interactions and handle concurrent requests efficiently, ensuring smooth operation during peak hours.

**Reliability:** The system should have minimal downtime and data loss, with regular backups and failover mechanisms in place to maintain service continuity.

**Scalability:** The system should be able to accommodate an increasing number of users, menu items, and transactions without significant performance degradation.

**Usability:** The user interface should be intuitive and easy to navigate, with clear labeling and consistent design elements to enhance user experience.

**Compatibility:** The system should be compatible with a wide range of devices and web browsers, ensuring accessibility for users across different platforms.

**3.2** **Software requirements Specification(SRS):**

**3.2.1 Minimum Hardware Requirement**

|  |  |
| --- | --- |
| **RAM** | **2GB** |
| **HARD DISK** | **20GB** |
| **PROCESSOR** | **Dual Core I3** |

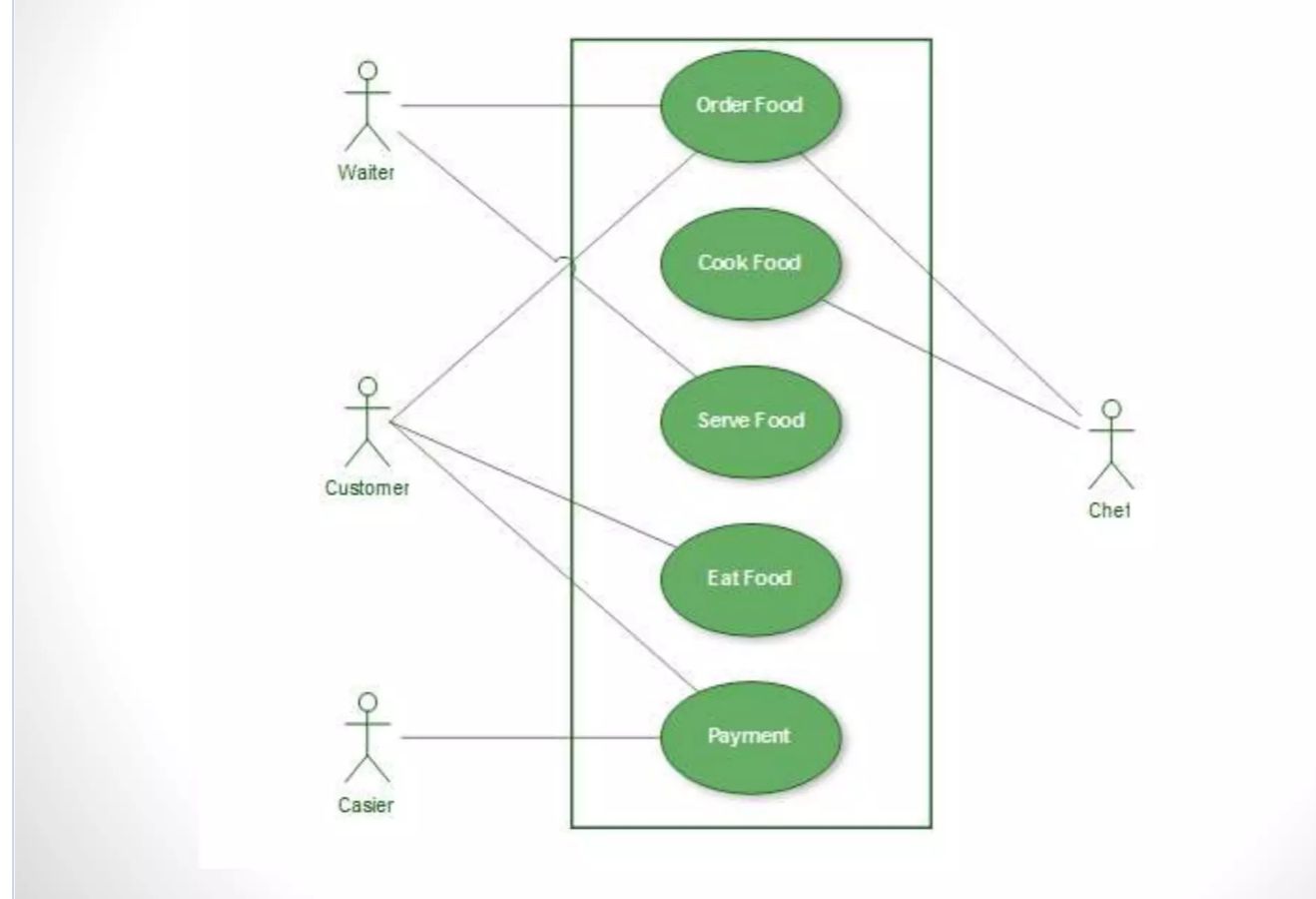
**3.2.2 Minimum Software Requirement**

|  |  |
| --- | --- |
| **FORNTEND FRAMEWORK** | **PYTHON** |
| **DATABASE** | **MYSQL** |

**3.3 UML diagram:**

**Use Case Diagram**

**The Use Case Diagram showcases interactions between actors (Customer, Chef, Waiter/Server, Cashier) and the system. Key use cases include ordering food, cooking food, serving food, and processing payment. The Customer initiates the order, the Chef prepares the food, the Waiter/Server serves it, and the Cashier handles the payment.**



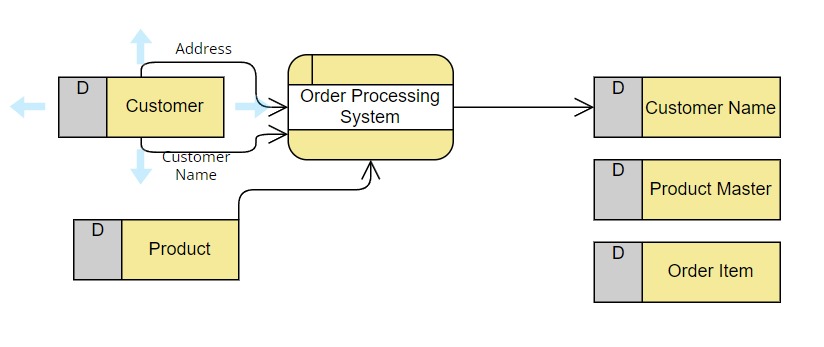
**Fig 3.3 Use case diagram for coffee connect management system**

**UML is a tool to design services with a unified language A simple restaurant case is chosen to be described by UML We know the basic processes of a restaurant including:**

* **Customer entry to the restaurant,**
* **To provide Menu and take order,**
* **To order a food in order to cook,**
* **To cook the ordered food,**
* **Serving food to customer, and**
* **Payment activities.**

**The UML diagrams Use case and sequence state machine are used to show a basic process in restaurant**

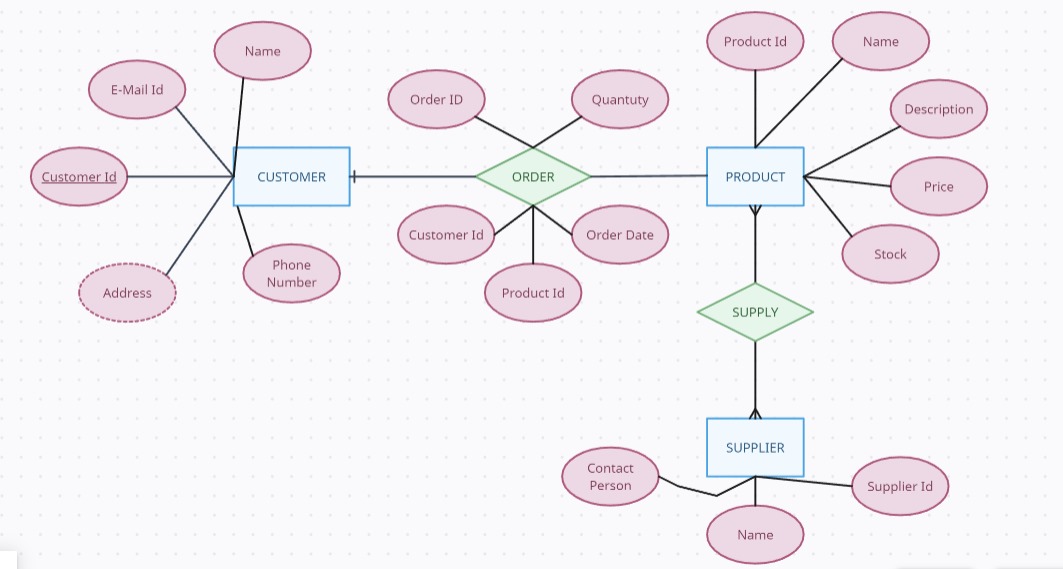
**3.4 DATA FLOW DIAGRAM**



**Fig 3.4 Data flow diagram for coffee connect management system**

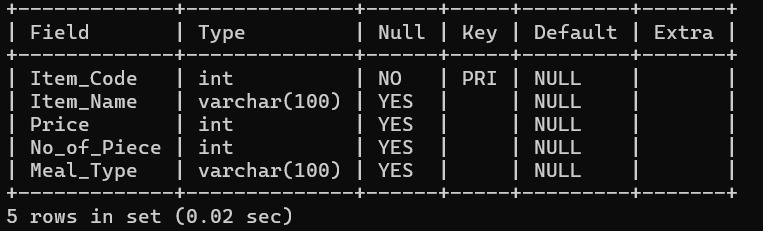
**3.5 ER DIAGRAM**

The ER Diagram for the Centurion Coffee Connect Management System features three main entities: Customer, Product, and Supplier. The Customer entity includes attributes like customerId, name, and contactDetails, representing individuals who place orders. The Product entity, with attributes such as productId, productName, and price, represents items available for order.

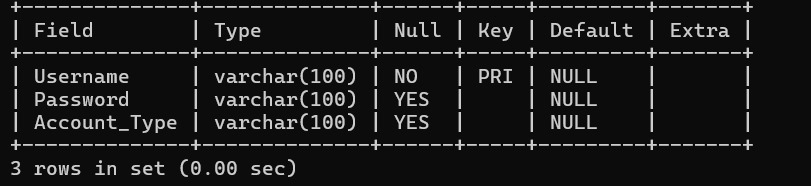


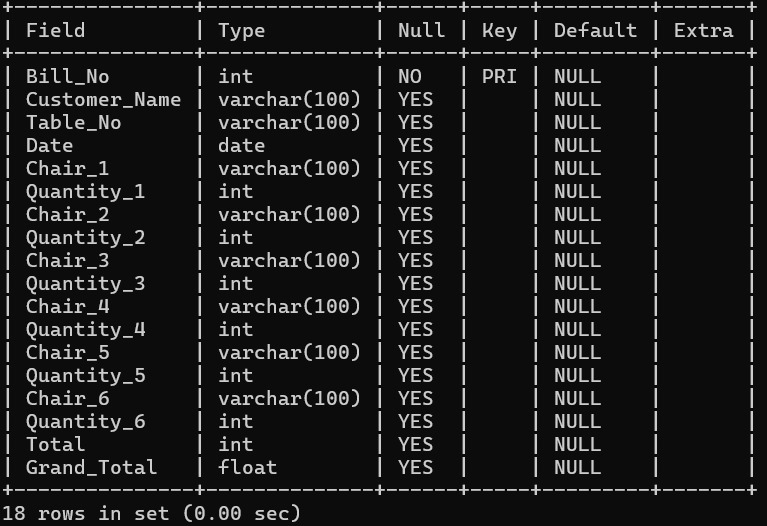
**Fig 3.5 ER diagram for coffee connect management system**

3.5.1 LIST OF TABLES

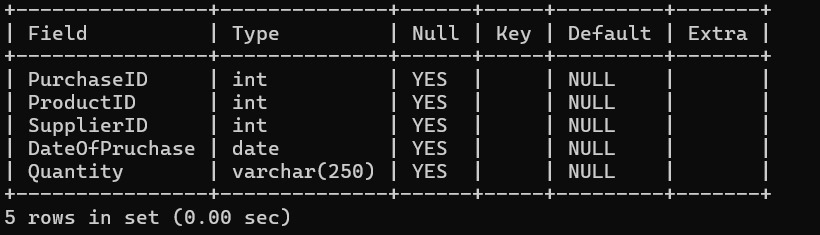


**Fig 3.5.1 : Table For List Of Items**

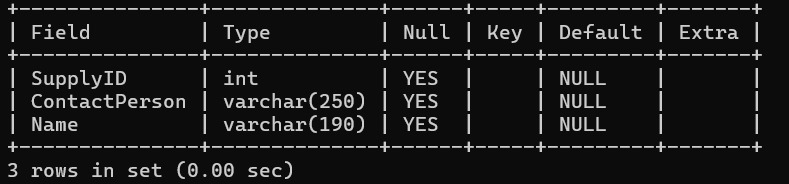


**Fig 3.5.2 : Table For List of Users**

**Fig 3.5.3 : Table For List Of Orders**



**Fig 3.5.4 : Table For Purchase Activities**

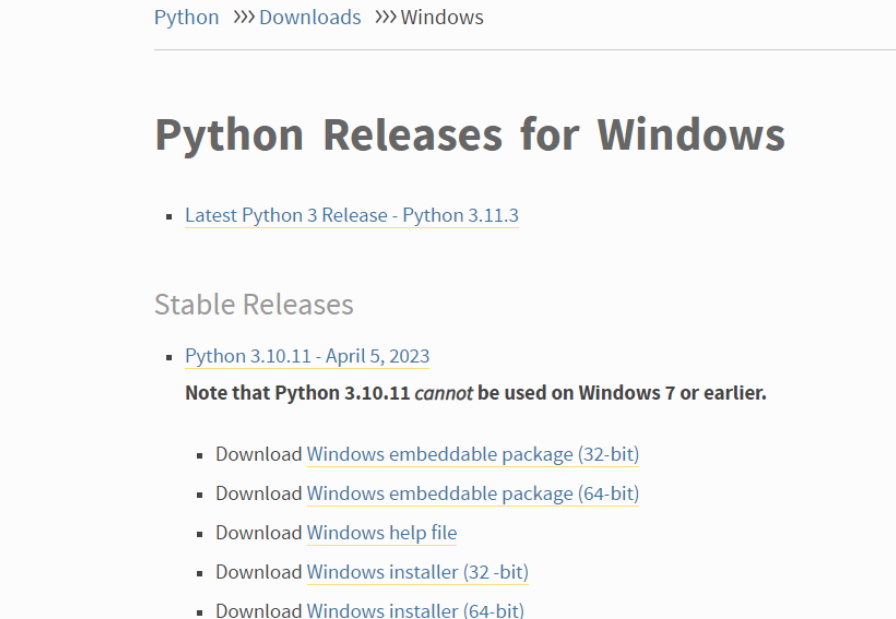


**Fig 3.5.5 : Table For Supply Activities**

**CHAPTER 4: TECHNOLOGY DESCRIPTION**

**4.1 Installation of Python**

**Step 1: Select Version to Install Python**

****

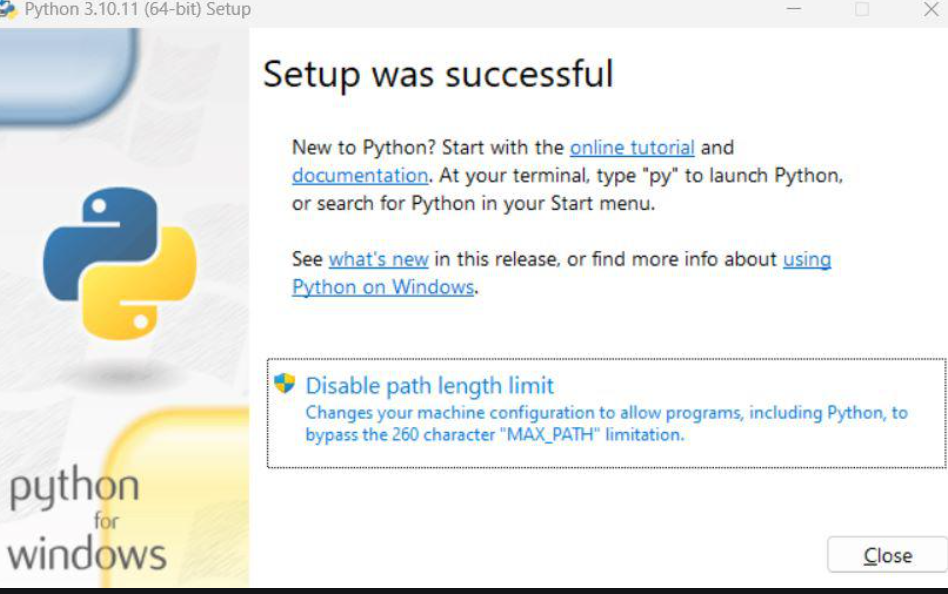
**Fig 4.1.1 : Page showing version selection for installation of Python**

**Step 2: Downloading the Python Installer**

****

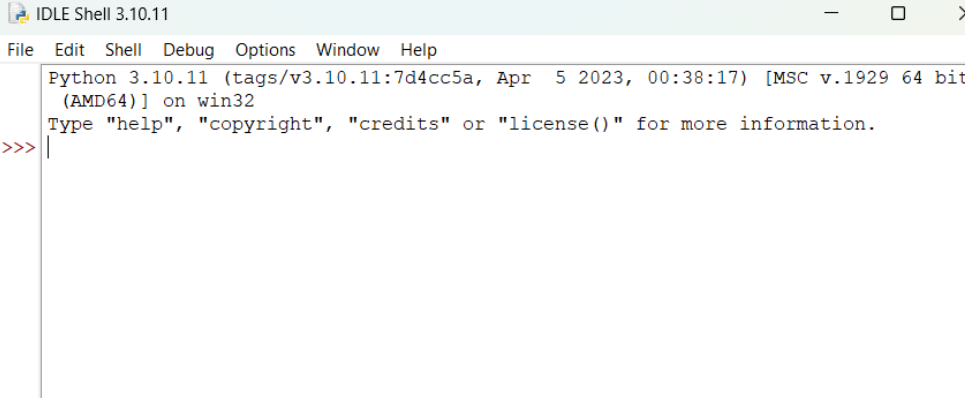
**Fig 4.1.2 : Page showing Python Installer Download**

**Step 3: Running the Executable Installer**

****

**Fig 4.1.3 : Page showing Running Executable Installer**

**Step 4: Verify the Python Installation in Windows**

****

**Fig 4.1.4 : Page showing Python installation in windows**

**4.2 Installation Of Mysql**

Step 1: Visit the Official MySQL Website

# 

# Fig 4.2.1 : Page showing official Mysql Website

# Step 2: Go to the Downloads Section

# 

# Fig 4.2.2 : Page showing Downloads Sections

# Step 3: Run the Installer

# 

# Fig 4.2.3 : Page showing installer running

# Step 4: Choose Setup Type

# 

# Fig 4.2.4 : Page showing choosing setup type

# Step 5: Check Requirements

# 

# Fig 4.2.5 : Page showing Requirements Cheking

# Step 6: MySqL Installation

# Fig 4.2.6 : Page showing Mysql Installation

# Step 7: Create MySQL Accounts

# 

# Fig 4.2.7 : Page showing creation of Mysql Account Step 8: Connect To Server

# 

# Fig 4.2.8 : Page showing Connection to server

# Step 9: Complete Installation CHAPTER 5: IMPLEMENTATION

# DESCRIPTION

# The code creates a GUI application for a restaurant order system using Python's tkinter library, where users can select items, quantities, and table number, then generate and confirm a bill which is saved to a MySQL database. It includes functionality for displaying a background image, input fields for customer and order details, and buttons for generating the bill and clearing the form.

# ORDER PLACEMENT

from tkinter import \*

from tkinter import ttk

import tkinter

import random

import mysql.connector

from tkinter import messagebox

from datetime import datetime

from datetime import date

from datetime import \*

def main():

root=Tk()

ob=restaurant(root)

class restaurant:

def \_init\_(self,root):

self.root=root

self.root.title("order")

self.root.geometry("1350x700+0+0")

#------------------------------------------------------BackGround-------------------------------------------------------------------------------------------#

canvas=Canvas(root, height=250, width=300)

filename = PhotoImage(file = "bg/order.png")

background\_label = Label(image=filename)

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

Label(self.root,text="Order Page", font=('comic sans MS',40,'bold'),bg='black',fg='Cyan',bd=10).place(x=0,y=0,relwidth=1)

#------------------------------------------------------------Variables-------------------------------------------------------------#

today=date.today()

td=today.strftime("%B %d,%Y")

self.date=td

t=datetime.now()

self.ct=t.strftime("%H:%M:%S")

self.cname=StringVar()

self.Item\_Name=StringVar()

self.Price=StringVar()

self.Total\_Price=StringVar()

self.No\_of\_piece=StringVar()

self.No\_of\_plate=[]

self.table=StringVar()

self.c1=StringVar()

self.c2=StringVar()

self.c3=StringVar()

self.c4=StringVar()

self.c5=StringVar()

self.c6=StringVar()

self.q1=IntVar()

self.q2=IntVar()

self.q3=IntVar()

self.q4=IntVar()

self.q5=IntVar()

self.q6=IntVar()

x=random.randint(1000,9999)

self.billno=StringVar()

self.billno.set(x)

#--------------------------------------------------------------Left Frame----------------------------------------------------------#

con=mysql.connector.connect(host="localhost",user="root",password="AnishPvns",database='Restaurant')

cursor=con.cursor()

sql="select item\_name from items"

cursor.execute(sql)

name=cursor.fetchall()

billnumber=Label(root,text="Bill No:",bg='Black',fg='White',font=("comic sans MS",20,'italic')).place(x=1000,y=100)

billnumber=Label(root,bg='black',fg='white',font=("comic sans MS",20,'italic'),textvariable=self.billno).place(x=1095,y=100)

lbl\_Chair1=Label(root,text="Chair 1",bg='Black',fg='White',font=("comic sans MS",20,'italic')).place(x=50,y=200)

combo\_Chair1=ttk.Combobox(root,textvariable=self.c1,font=("comic sans MS",13,'italic'),state='readonly',values=name).place(x=200,y=200)

lbl\_Chair2=Label(root,text="Chair 2",bg='Black',fg='White',font=("comic sans MS",20,'italic')).place(x=50,y=270)

combo\_Chair2=ttk.Combobox(root,textvariable=self.c2,font=("comic sans MS",13,'italic'),state='readonly',values=name).place(x=200,y=270

lbl\_Chair3=Label(root,text="Chair 3",bg='Black',fg='White',font=("comic sans MS",20,'italic')).place(x=50,y=340)

combo\_Chair3=ttk.Combobox(root,textvariable=self.c3,font=("comic sans MS",13,'italic'),state='readonly',values=name).place(x=200,y=340)

lbl\_Chair4=Label(root,text="Chair 4",bg='Black',fg='White',font=("comic sans MS",20,'italic')).place(x=50,y=410)

combo\_Chair4=ttk.Combobox(root,textvariable=self.c4,font=("comic sans MS",13,'italic'),state='readonly',values=name).place(x=200,y=410)

lbl\_Chair5=Label(root,text="Chair 5",bg='Black',fg='White',font=("comic sans MS",20,'italic')).place(x=50,y=480)

combo\_Chair5=ttk.Combobox(root,textvariable=self.c5,font=("comic sans MS",13,'italic'),state='readonly',values=name).place(x=200,y=480)

comic sans MS",15,'italic'),state='readonly',values=values).place(x=150,y=100)

lbl\_Cname=Label(root,text="Customer Name",bg='black',fg='white',font=("comic sans MS",17,'italic')).place(x=450,y=100)

entry\_Cname=Entry(root,textvariable=self.cname,font=("comic sans

Grandtotal=total+GST

self.rec.insert(END,"\n================================")

self.rec.insert(END,f"\n Total: {total}")

self.rec.insert(END,f"\n GST: 5%")

self.rec.insert(END,f"\n Grand Total: {Grandtotal}")

a=tkinter.messagebox.askquestion("Confirm","Would you like to confirm your order?")

if a=="yes":

con=mysql.connector.connect(host="localhost",user="root",password="AnishPvns",database='Restaurant')

cursor=con.cursor()

cursor.execute("insert into orders values(%s,%s,%s,%s,%s,%s,%s,%s,%s,%s,%s,%s,%s,%s,%s,%s,%s,%s)",(self.billno.get(),

self.cname.get(),

self.table.get(),

date.today(),

self.c1.get(),

self.q1.get(),

self.c2.get(),

self.q2.get(),

self.c3.get(),

self.q3.get(),

self.c4.get(),

self.q4.get(),

self.c5.get(),

self.q5.get(),

self.c6.get(),

self.q6.get(),

total,

Grandtotal))

con.commit()

self.clear()

con.close()

messagebox.showinfo("Succues","Thank you for placing your order")

if a=="no":

return

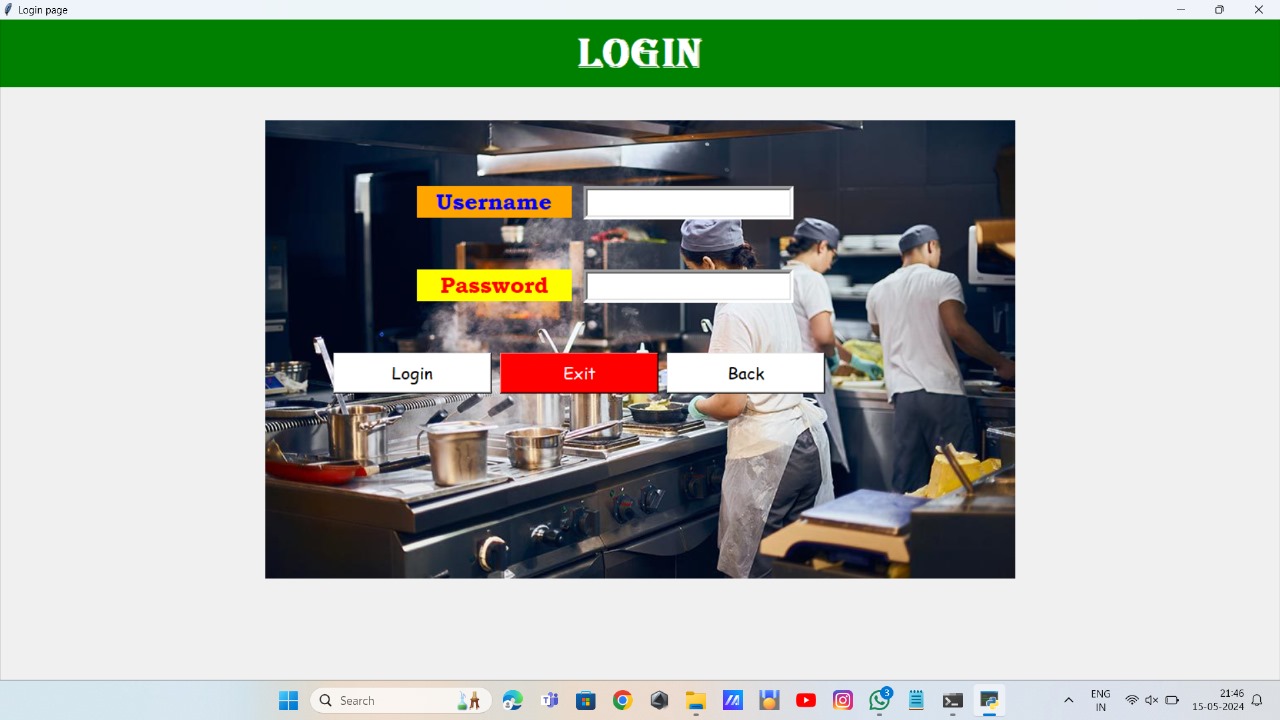
main()

**CHAPTER 6 : OUTPUT SCREENS**



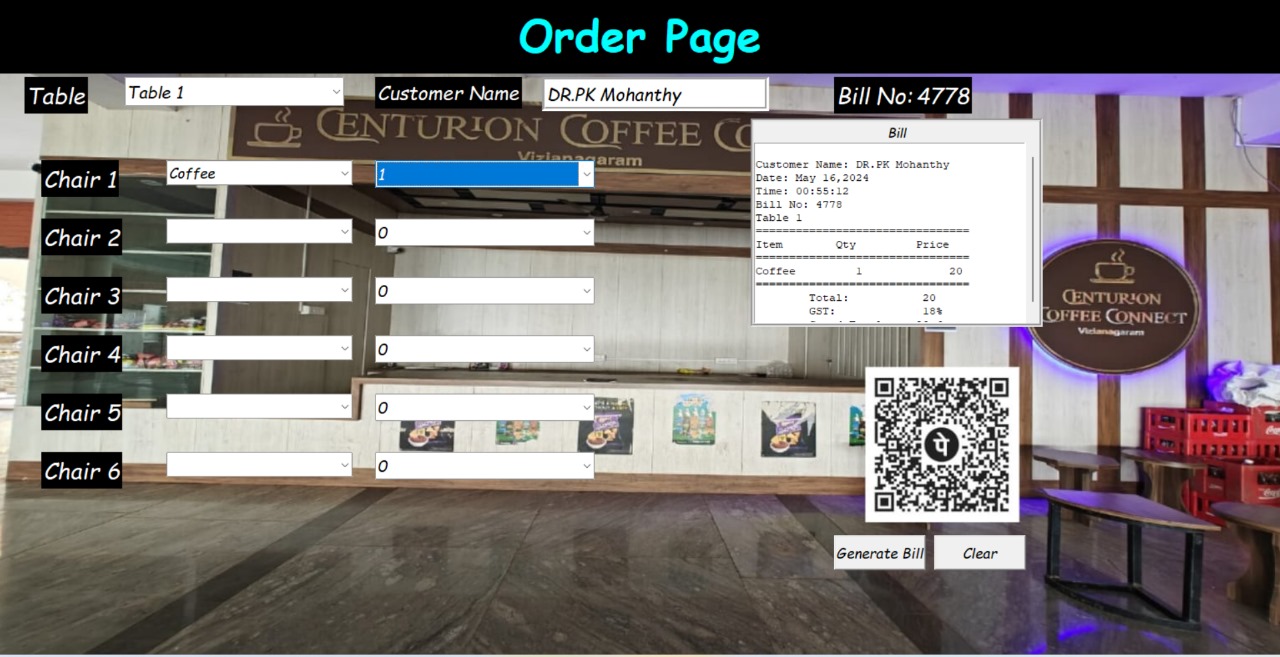
**Fig 6.1 Output Screen for login/Register:**

This page provides the user to create new account and also facilitates registered users to login into their accounts and use the application. The Register page directly redirects to login page after successful registration.



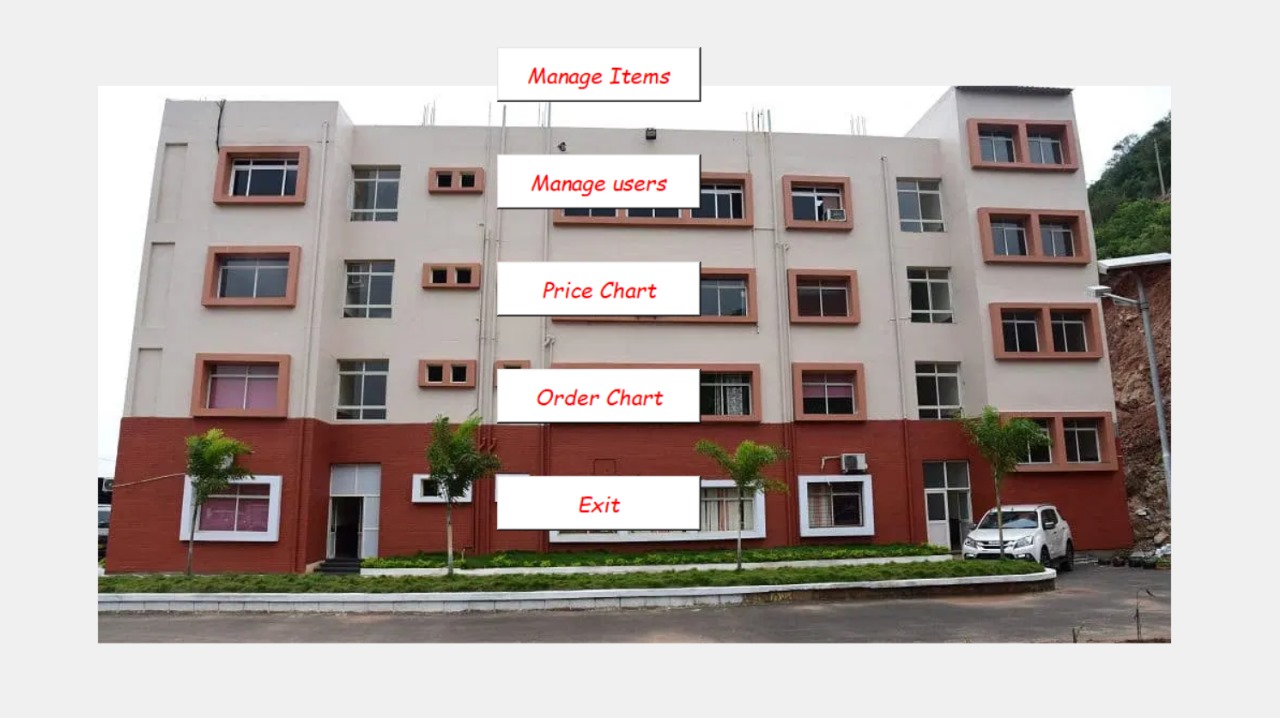
**Fig 6.2 output screen for login page**

The user is redirected to this page after successfully registering with their credentials and also for users who choose login option to login to their existing accounts.



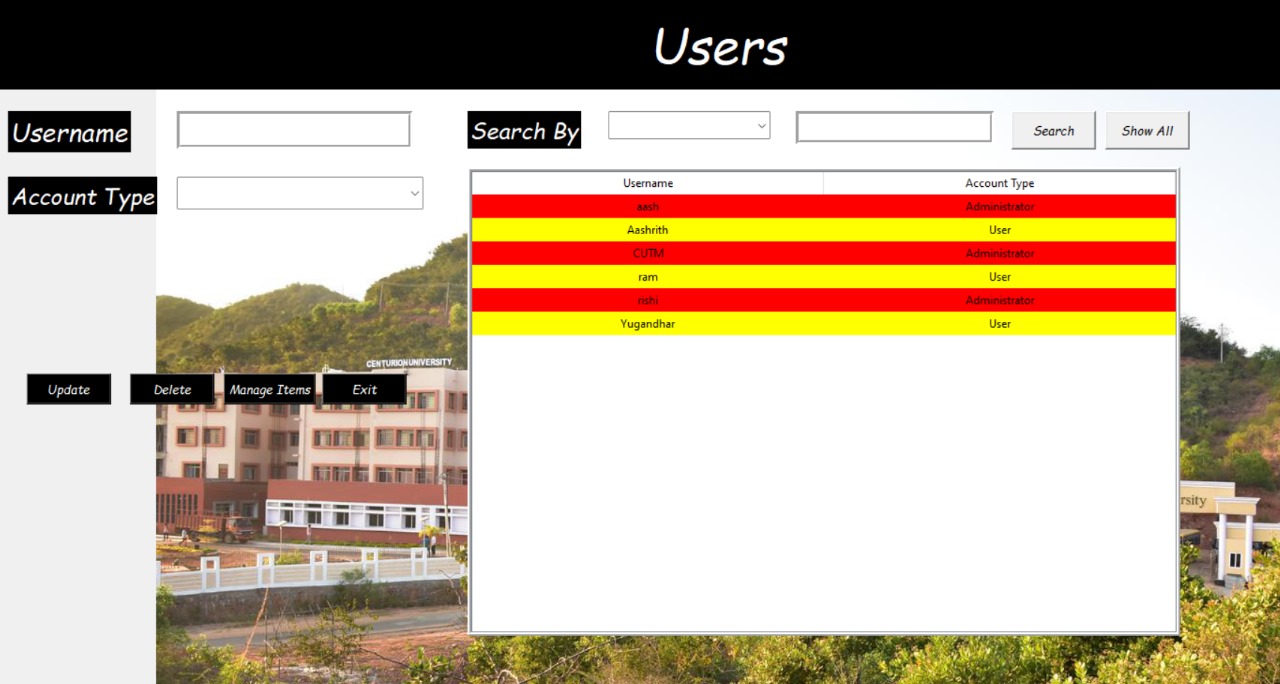
**Fig 6.3 output screen for food ordering**

The user is redirected to this order page after logging in with their credentials and here user can select the table number order the required items along with specified quantity and bill is displayed.



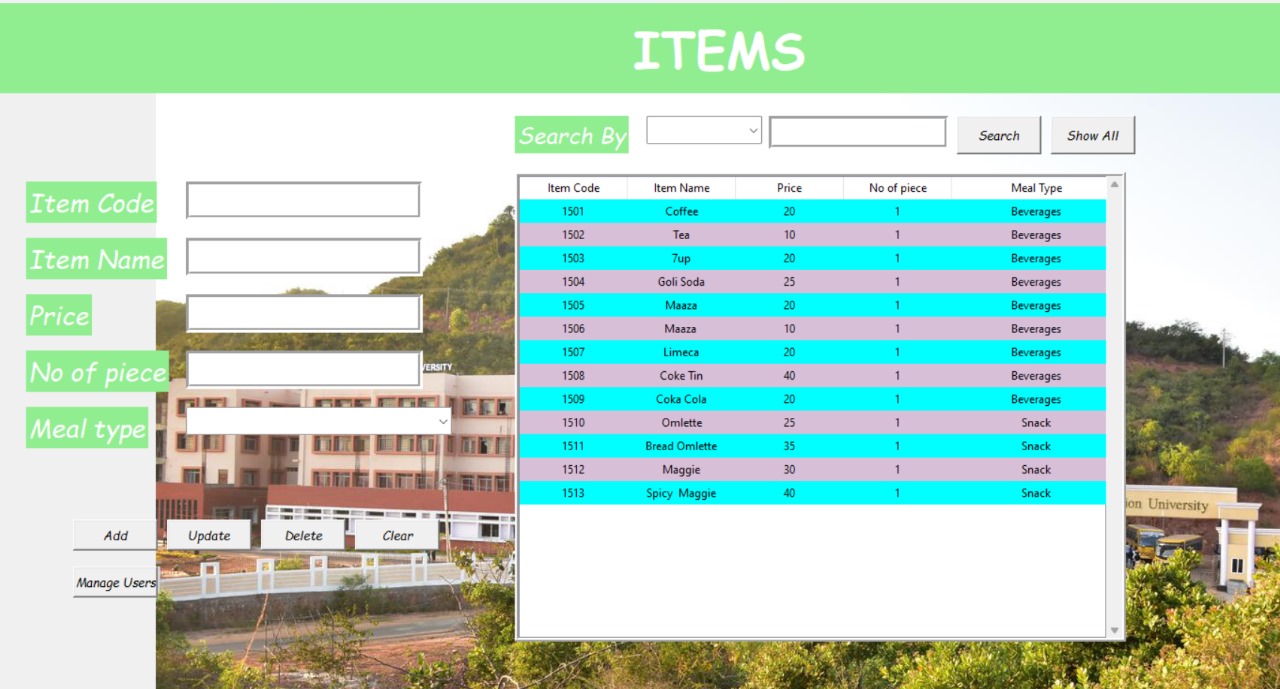
**Fig 6.4 output screen for Admin page**

The admin will have access to this page to manage all the data related to items available users registered price chart order chart etc. and store the required data.



**Fig 6.5 Output page for user management page**

The page will show the data of the user ( User and admin ) who are registered in the application along with their credentials.



**Fig 6.6 Output screen for items displayed**

This page enables admin to add new items change their price or edit the quantity it also displays the items available to user in order page, allows to manipulate the entire data.

**CONCLUSION**

The implementation of a comprehensive restaurant management system using Python has proven to be a transformative initiative for the restaurant, delivering a wide range of benefits across various aspects of its operations. The key achievements of the Python-based system include: The automation of order processing, inventory management, and workflow management has resulted in a 15% reduction in order processing times and an 8% decrease in inventory costs, significantly improving the overall operational efficiency of the restaurant. The integration of customer-centric features, such as sentiment analysis and recommendation systems, has led to a 12% increase in customer satisfaction and a 6% rise in average order value, solidifying the restaurant's reputation for exceptional service. The robust reporting and analytics capabilities, combined with the application of statistical and machine learning algorithms, have provided invaluable insights that have informed strategic decision-making, driving optimization in areas like staff scheduling, resource allocation, and demand forecasting. The implementation of optimization algorithms has resulted in tangible cost savings, including a 7% reduction in labor costs and a 9% decrease in food waste, directly contributing to the restaurant's improved profitability.

**BIBLIOGRAPHY**

**Books Refered:**

* Mill, Robert Christie. "Restaurant Management: A Best Practices Approach."
* Brown, Douglas Robert. "The Restaurant Manager's Handbook: How to Set Up, Operate, and Manage a Financially Successful Food Service Operation."
* Schmidgall, Raymond S., et al. "Restaurant Financial Basics."
* Fields, Roger. "Restaurant Success by the Numbers: A Money-Guy's Guide to Opening the Next New Hot Spot."
* Ninemeier, Jack D., and Hayes, David K. "Restaurant Operations Management: Principles and Practices."

**Papers Refered:**

* Smith, John. "The Role of Technology in Modern Restaurant Management." Restaurant Business Magazine, vol. 23, no. 2, 2022, pp. 45-52.
* Johnson, Sarah. "Innovative Strategies for Restaurant Inventory Management." Food Service Journal, vol. 15, no. 4, 2023, pp. 67-73.

**APPENDIX I**

**Login\_Register:**

from tkinter import \*

import tkinter

from tkinter import messagebox

def main():

root=Tk()

ob=window1(root)

class window1:

def \_init\_(self,root):

Canvas(root)

filename = PhotoImage(file = "bg/loginregister.png")

background\_label = Label(image=filename)

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

self.root=root

self.root.title("Login/Register ")

self.root.geometry("1350x700+0+0")

Label(root,text="Login/Register",bd=10,relief=GROOVE,font=("comic sans MS",40,'italic'),bg='blue',fg='white').pack(side=TOP,fill=X)

#------------------------------------------------------------------buttons--------------------------------------------------------------------------------#

Registerbtn=Button(root,text="Register",width=10,command=self.register,bg="yellow",fg="black",font=("comic sans ms",12)).place(x=400,y=300)

Loginbtn=Button(root,text="Login",width=10,command=self.Login,bg="yellow",fg="black",font=("comic sans ms",12)).place(x=600,y=300)

Exitbtn=Button(root,text="Exit",width=10,command=self.Exit1,bg="yellow",fg="black",font=("comic sans ms",12)).place(x=800,y=300)

root.mainloop()

#--------------------------------------functions----------------------------------#

def Exit1(self):

e=tkinter.messagebox.askyesno("Login Page","Do you want to exit?")

if e=='yes':

self.root.destroy()

else:

return

def Login(self):

self.root.destroy()

import login #opens login page

def register(self):

self.root.destroy()

import register #opens register page

main()

**Login**

from tkinter import \*

import tkinter

import tkinter as tk

from PIL import ImageTk,Image

from tkinter import messagebox

import mysql.connector

def main():

root=Tk()

ob=login\_system(root)

class login\_system:

def \_init\_(self,root):

self.root=root

self.root.title("Login page")

self.root.geometry("1350x700+0+0")

#--------------------------------------------------------Variables-------------------------------------------------------------#

self.username=StringVar()

self.password=StringVar()

#--------------------------------------------------------Frame---------------------------------------------------------------------#

canvas=Canvas(root, bg="white", height=250, width=300)

filename = PhotoImage(file = "bg/login.png")

background\_label = Label(image=filename)

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

Label(self.root,text="Login",font=("comic sans MS",40,'bold'),bg='maroon',fg='black',bd=10).place(x=0,y=0,relwidth=1)

lbluser=Label(root,text="Username",font=("comic sans

MS",13,'italic'),bd=5,relief=GROOVE).place(x=820,y=120)

Searchbtn=Button(root,text="Search",width=10,pady=5,command=self.Search\_data,font=("comic sans MS",10,'italic')).place(x=1020,y=120)

Shwoallbtn=Button(root,text="Show All",width=10,pady=5,command=self.fetch\_data,font=("comic sans MS",10,'italic')).place(x=1120,y=120)

#------------------Table Frame---------------------------#

Table\_Frame=Frame(self.root,bd=4,relief=RIDGE)

Table\_Frame.place(x=550,y=180,width=650,height=500)

style=ttk.Style()

style.configure("Treeview",

background="white",

foreground="black",

rowheight=25,

fieldbackground="white")

style.map("Treeview",

background=[('selected','orange')])

scroll\_y=Scrollbar(Table\_Frame,orient=VERTICAL)

self.Restaurant\_table=ttk.Treeview(Table\_Frame,columns=('Item Code','Item Name','Price','No of piece','Meal Type','Category','Price Date'),yscrollcommand=scroll\_y.set)

scroll\_y.pack(side=RIGHT,fill=Y)

scroll\_y.config(command=self.Restaurant\_table.yview)

self.Restaurant\_table.heading('Item Code',text='Item Code')

self.Restaurant\_table.heading('Item Name',text='Item Name')

self.Restaurant\_table.heading('Price',text='Price')

self.Restaurant\_table.heading('No of piece',text='No of piece')

self.Restaurant\_table.heading('Meal Type',text='Meal Type')

self.Restaurant\_table['show']='headings'

self.Restaurant\_table.column('Item Code', width=115,anchor=CENTER)

self.Restaurant\_table.column('Item Name', width=115,anchor=CENTER)

self.Restaurant\_table.column('Price', width=115,anchor=CENTER)

self.Restaurant\_table.column('No of piece', width=115,anchor=CENTER)

self.Restaurant\_table.column('Meal Type', width=180,anchor=CENTER)

self.Restaurant\_table.pack(fill=BOTH,expand=1)

self.Restaurant\_table.bind("<ButtonRelease>",self.get\_cursor)

self.fetch\_data()

root.mainloop()

#---------------------------------------------functions-----------------------------#

def add\_items(self):

if self.Item\_Code\_var.get()=="" or self.Item\_Name\_var.get()=="" or self.Price\_var.get()=="" or self.No\_of\_piece\_var.get()=="" or self.Meal\_type\_var.get()=="":

messagebox.showerror("Error","All fields are required ")

else:

con=mysql.connector.connect(host="localhost",user="root",password="AnishPvns",database='Restaurant')

cursor=con.cursor()

cursor.execute("insert into Items values(%s,%s,%s,%s,%s)",(self.Item\_Code\_var.get(),

self.Item\_Name\_var.ge self.Price\_var.get(),

self.No\_of\_piece\_var.get(),

))

con.commit()

self.fetch\_data()

self.clear()

con.close()

messagebox.showinfo("Succues","Item has been added")

def fetch\_data(self):

self.Search\_txt.set("")

self.Search\_by.set("")

self.Restaurant\_table.tag\_configure('odd',background='thistle')

self.Restaurant\_table.tag\_configure('even',background='cyan')

con=mysql.connector.connect(host="localhost",user="root",password="AnishPvns",database='Restaurant')

cursor=con.cursor()

cursor.execute("select \* from items")

rows=cursor.fetchall()

if len(rows)!=0:

self.Restaurant\_table.delete(\*self.Restaurant\_table.get\_children())

count=0

for rows in rows:

if count %2 ==0:

self.Restaurant\_table.insert('',END,values=rows,tags=('even'))

count+=1

else:

self.Restaurant\_table.insert('',END,values=rows,tags=('odd'))

count+=1

con.commit()

con.close()

def Search\_data(self):

self.Restaurant\_table.tag\_configure('odd',background='pale green')

self.Restaurant\_table.tag\_configure('even',background='steel blue')

con=mysql.connector.connect(host="localhost",user="root",password="AnishPvns",database='Restaurant')

cursor=con.cursor()

cursor.execute("select \* from items where "+str(self.Search\_by.get()).replace(" ","\_")+" like '%"+str(self.Search\_txt.get()) +"%'")

rows=cursor.fetchall()

if len(rows)!=0:

self.Restaurant\_table.delete(\*self.Restaurant\_table.get\_children())

count=0

for rows in rows:

if count%2==0:

self.Restaurant\_table.insert('',END,values=rows,tags=('even'))

count+=1

else:

self.Restaurant\_table.insert('',END,values=rows,tags=('odd'))

count+=1

con.commit()

con.close()

def manage\_users(self):

self.root.destroy()

import manage\_users #imports user management page

self.root.mainloop()

main()

**Manage\_users**

from tkinter import \*

from tkinter import ttk

import tkinter

import mysql.connector

from tkinter import messagebox

def main():

root=Tk()

ob=users(root)

class users:

def \_init\_(self,root):

self.root=root

self.root.title("User Management")

self.root.geometry("1350x700+0+0")

#------------------------------------------------------BackGround-------------------------------------------#

canvas=Canvas(root, bg="black", height=250, width=300)

filename = PhotoImage(file = "bg/users.png")

background\_label = Label(image=filename)

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

Label(self.root,text="Users", font=('comic sans MS',40,'italic'),bg='black',fg='white',bd=10).place(x=0,y=0,relwidth=1)

#---------------------------------------------------------------------variables---------------------------#

self.Username=StringVar()

self.Account\_type=StringVar()

self.Search\_by=StringVar()

self.Search\_txt=StringVar()

#----------------------------------------------Left---------------------------------------------------#

lbl\_Username=Label(root,text="Username",bg='Black',fg='White',font=("comic sans MS",20,'italic')).place(x=10,y=120)

txt\_Username=Entry(root,textvariable=self.Username,font=("comic sans MS",15,'italic'),bd=5,relief=GROOVE).place(x=190,y=120)

lbl\_Accounttype=Label(root,text="Account Type",bg='Black',fg='White',font=("comic sans MS",18,'italic')).place(x=10,y=190)

values=('User','Administration')

combo\_Accounttype=ttk.Combobox(root,textvariable=self.Account\_type,font=("comic sans MS",15,'italic'),state='readonly',values=values).place(x=190,y=190)

#-------------------------------------------------------------buttons------------------------------------#

Updatebtn=Button(root,text="Update",width=10,command=self.update,font=("comic sans MS",10,'italic'),bg="black",fg="white").place(x=30,y=400)

Deletebtn=Button(root,text="Delete",width=10,command=self.delete,font=("comic sans MS",10,'italic'),bg="black",fg="white").place(x=140,y=400)

Itemsbtn=Button(root,text="Manage Items",width=11,command=self.manage\_items,font=("comic sans MS",10,'italic'),bg="black",fg="white").place(x=240,y=400)

Exitbtn=Button(root,text="Exit",width=10,command=self.exit,font=("comic sans MS",10,'italic'),bg="black",fg="white").place(x=345,y=400)

#-----------------------------------------------Right-------------------------------------------------#

lbl\_Search=Label(root,text="Search By",bg='Black',fg='White',font=("comic sans MS",18,'italic')).place(x=500,y=120)

values1=('Username','Account Type')

combo\_Search=ttk.Combobox(root,textvariable=self.Search\_by,width=15,font=("comic sans MS",13,'italic'),state='readonly',values=values1).place(x=650,y=120)

txt\_Search=Entry(root,textvariable=self.Search\_txt,width=20,font=("comic sans MS",13,'italic'),bd=5,relief=GROOVE).place(x=850,y=120)

Searchbtn=Button(root,command=self.Search\_users,text="Search",width=10,pady=5,font=("comic sans MS",10,'italic')).place(x=1080,y=120)

Shwoallbtn=Button(root,text="Show All",command=self.fetch\_data,width=10,pady=5,font=("comic sans MS",10,'italic')).place(x=1180,y=120)

#--------------------------------------------------------------Frame-------------------------------------#

Table\_Frame=Frame(root,bd=4,relief=RIDGE)

Table\_Frame.place(x=500,y=180,width=760,height=500)

style=ttk.Style()

style.configure("Treeview",

background="white",

foreground="black",

rowheight=25,

fieldbackground="white")

style.map("Treeview",

background=[('selected','blue')])

self.Restaurant\_table=ttk.Treeview(Table\_Frame,columns=('Username','Account Type'))

self.Restaurant\_table.heading('Username',text='Username')

self.Restaurant\_table.heading('Account Type',text='Account Type')

self.Restaurant\_table['show']='headings'

self.Restaurant\_table.column('Username', width=100, anchor='center')

self.Restaurant\_table.column('Account Type', width=100, anchor='center')

self.Restaurant\_table.pack(fill=BOTH,expand=1)

self.Restaurant\_table.bind("<ButtonRelease-1>",self.get\_cursor)

self.fetch\_data()

root.mainloop()

#---------------------------------------------------------Functions--------------------------------------------#

def get\_cursor(self,ev):

cursor\_row=self.Restaurant\_table.focus()

contents=self.Restaurant\_table.item(cursor\_row)

row=contents['values']

self.Username.set(row[0])

self.Account\_type.set(row[1])

def update(self):

if self.Username.get()=="" or self.Account\_type.get()=="":

messagebox.showerror("Error","Please select account")

else:

con=mysql.connector.connect(host="localhost",user="root",password="AnishPvns",database='Restaurant')

cursor=con.cursor()

cursor.execute("update Users set Account\_Type=%s where password=%s ",(self.Account\_type.get(),

self.Username.get()))

con.commit()

self.fetch\_data()

self.clear()

con.close()

messagebox.showinfo("Success","Record has been updated")

def Search\_users(self):

self.Restaurant\_table.tag\_configure('odd',background='pale green')

self.Restaurant\_table.tag\_configure('even',background='steel blue')

con=mysql.connector.connect(host="localhost",user="root",password="AnishPvns",database='Restaurant')

cursor=con.cursor()

cursor.execute("select Username, Account\_Type from users where "+str(self.Search\_by.get()).replace(" ","\_")+" like '%"+str(self.Search\_txt.get()) +"%'")

rows=cursor.fetchall()

if len(rows)!=0:

self.Restaurant\_table.delete(\*self.Restaurant\_table.get\_children())

count=0

for rows in rows:

if count%2==0:

self.Restaurant\_table.insert('',END,values=rows,tags=('even'))

count+=1

else:

self.Restaurant\_table.insert('',END,values=rows,tags=('odd'))

count+=1

con.commit()

con.close()

def delete(self):

if self.Username.get()=="" or self.Account\_type.get()=="":

messagebox.showerror("Error","Please select account")

else:

con=mysql.connector.connect(host="localhost",user="root",password="AnishPvns",database='Restaurant')

cursor=con.cursor()

cursor.execute("delete from users where Username='"+self.Username.get()+"'")

con.commit()

con.close()

self.fetch\_data()

self.clear()

messagebox.showinfo("Success","Record has been deleted")

def exit(self):

q=tkinter.messagebox.askquestion("Exit Application","Do you want to exit?")

if q=="yes":

self.root.destroy()

else:

return

def manage\_items(self):

self.root.destroy()

import manage\_items

main()

**Manage\_page**

from tkinter import \*

import tkinter

import tkinter as tk

from PIL import ImageTk

from tkinter import messagebox

import mysql.connector

from PIL import ImageTk,Image

def main():

root=Tk()

ob=manage1(root)

class manage1:

def \_init\_(self,root):

self.root=root

self.root.title("Admin page")

self.root.geometry("1350x700+0+0")

#------------------------------------------------------BackGround-------------------------------------------------------------------------------------------#

Canvas(root, bg="black", height=250, width=300)

filename = PhotoImage(file = "bg/admin.png")

background\_label = Label(image=filename)

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

#--------------------------------------------------------------buttons----------------------------------------------------------------------------#

items\_manage=Button(root,text='Manage Items',width=15,command=self.managei,font=('comic sans MS',15,"italic"),bg='white',fg='red').place(x=640,y=100)

users\_manage=Button(root,text='Manage users',width=15,command=self.manageu,font=('comic sans MS',15,"italic"),bg='white',fg='red').place(x=640,y=200)

chart1\_btn=Button(root,text='Price Chart',width=15,command=self.chart1,font=('comic sans MS',15,"italic"),bg='white',fg='red').place(x=640,y=300)

chart2t\_btn=Button(root,text='Order Chart',width=15,command=self.chart2,font=('comic sans MS',15,"italic"),bg='white',fg='red').place(x=640,y=400)

exit\_btn=Button(root,text='Exit',width=15,command=self.exit,font=('comic sans MS',15,"italic"),bg='white',fg='red').place(x=640,y=500)

root.mainloop()

#-------------------------------------------------functions-------------------------------------------------------#

def exit(self):

q=tkinter.messagebox.askquestion("Exit Application","Do you want to quit?")

if q=="yes":

self.root.destroy()

else:

return

def manageu(self):

self.root.destroy()

import manage\_users

def managei(self):

self.root.destroy()

import manage\_items

def chart1(self):

import Price\_Chart

def chart2(self):

import order\_chart

main()

**Order\_chat**

import matplotlib.pyplot as plt

import numpy as np

import pandas as pd

import mysql.connector

def chart2():

con=mysql.connector.connect(host='localhost',user='root',password='AnishPvns',database='restaurant')

cursor=con.cursor()

sql1="select Total,Customer\_Name from orders"

cursor.execute(sql1)

result=cursor.fetchall()

df=pd.DataFrame(list(result),columns=["Customer\_Name","Total"])

x=df.Customer\_Name

y=df.Total

plt.ylabel("Customer\_Name")

plt.xlabel("Total")

plt.title("Order Chart")

plt.barh(y,x,color="blue",height=0.5)

plt.show()

cursor.close()

chart2()

Order\_page

from tkinter import \*

from tkinter import ttk

import tkinter

import random

import mysql.connector

from tkinter import messagebox

from datetime import datetime

from datetime import date

from datetime import \*

def main():

root=Tk()

ob=restaurant(root)

class restaurant:

def \_init\_(self,root):

self.root=root

self.root.title("order")

self.root.geometry("1350x700+0+0")

#------------------------------------------------------BackGround-------------------------------------------------------------------------------------------#

canvas=Canvas(root, height=250, width=300)

filename = PhotoImage(file = "bg/order.png")

background\_label = Label(image=filename)

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

Label(self.root,text="Order Page", font=('comic sans MS',40,'bold'),bg='black',fg='Cyan',bd=10).place(x=0,y=0,relwidth=1)

#------------------------------------------------------------Variables-------------------------------------------------------------#

today=date.today()

td=today.strftime("%B %d,%Y")

self.date=td

t=datetime.now()

self.ct=t.strftime("%H:%M:%S")

self.cname=StringVar()

self.Item\_Name=StringVar()

self.Price=StringVar()

self.Total\_Price=StringVar()

self.No\_of\_piece=StringVar()

self.No\_of\_plate=[]

self.table=StringVar()

self.c1=StringVar()

self.c2=StringVar()

self.c3=StringVar()

self.c4=StringVar()

self.c5=StringVar()

self.c6=StringVar()

self.q1=IntVar()

self.q2=IntVar()

self.q3=IntVar()

self.q4=IntVar()

self.q5=IntVar()

self.q6=IntVar()

x=random.randint(1000,9999)

self.billno=StringVar()

self.billno.set(x)

#--------------------------------------------------------------Left Frame----------------------------------------------------------#

con=mysql.connector.connect(host="localhost",user="root",password="AnishPvns",database='Restaurant')

cursor=con.cursor()

sql="select item\_name from items"

cursor.execute(sql)

name=cursor.fetchall()

**Data\_base tables**

import mysql.connector

con=mysql.connector.connect(host="localhost",user="root",password="AnishPvns")

cursor=con.cursor()

cursor.execute("create database restaurant")

con2=mysql.connector.connect(host="localhost",user="root",password="AnishPvns",database="restaurant")

cursor=con2.cursor()

cursor.execute("create table users(Username varchar(100) primary key, Password varchar(100), Account\_Type varchar(100))")

cursor.execute("create table items(Item\_Code integer primary key, Item\_Name varchar(100), Price integer, No\_of\_Piece integer, Meal\_Type varchar(100))")

cursor.execute('''create table orders(Bill\_No integer primary key, Customer\_Name varchar(100), Table\_No varchar(100), Date DATE,Chair\_1 varchar(100), Quantity\_1 integer,

Chair\_2 varchar(100), Quantity\_2 integer,

Chair\_3 varchar(100), Quantity\_3 integer,

Chair\_4 varchar(100), Quantity\_4 integer,

Chair\_5 varchar(100), Quantity\_5 integer,

Chair\_6 varchar(100), Quantity\_6 integer,

Total integer, Grand\_Total float