CANTEEN MANAGEMENT SYSTEM

A MINI PROJECT REPORT

Submitted by

MADHAN SHANKAR G	**220701149**
MADHAVA GANESH A	**220701150**
MANOJ R L	**220701161**

In partial fulfillment for the award of the degree of

BACHELOR OF ENGINEERING

IN

COMPUTER SCIENCE

RAJALAKSHMI ENGINEERING COLLEGE (AUTONOMOUS)

THANDALAM

CHENNAI-602105

2024-2025

BONAFIDE CERTIFICATE

Certified that this project report "**CANTEEN MANAGEMENT SYSTEM**" is the bonafide work of "**MADHAN SHANKAR G (220701149)"** who carried out the project under my supervision.

Submitted for the Practical Examination held on _____

SIGNATURE   SIGNATURE

Dr.R.SABITHA   **Mrs.D.KALPANA**

Professor and II Year Academic Head   **Assistant Professor ,**

Computer Science and Engineering,   **Computer Science and Engineering,**

Rajalakshmi Engineering College   **Rajalakshmi Engineering College**

(Autonomous),   **(Autonomous),**

Thandalam, Chennai - 602 105   **Thandalam, Chennai - 602 105**

INTERNAL EXAMINER   **EXTERNAL EXAMINER**

ABSTRACT

The Canteen Food Order System is an innovative solution designed to streamline and modernize the food ordering process within a canteen environment. This Python-based GUI application leverages PySimpleGUI for a user-friendly interface and MongoDB for a robust backend, providing an efficient and reliable platform for managing canteen operations. The system is composed of three primary modules: client, scanner, and admin.

The client module allows customers to conveniently place orders through a straightforward application interface, generating a unique QR code for each order. This QR code is then scanned by the scanner module, which is positioned within the canteen, to validate the order and ensure its authenticity before generating the final receipt. The admin module offers comprehensive management capabilities, empowering the canteen manager to efficiently handle recipes, update prices and quantities, and oversee orders with ease.

Key features of the system include token-based authentication to prevent fraudulent QR codes, tagging of special items for promotional offers, and options for take-away orders, all designed to enhance customer convenience and satisfaction. Additional functionalities such as user feedback mechanisms and payment integrations further enrich the user experience. The system is engineered to handle high order volumes effectively, ensuring optimal performance during peak times.

Security is a paramount concern, with robust measures in place to safeguard user data and transaction details. The system's scalability allows it to support multiple canteens and larger operations seamlessly. Furthermore, the intuitive user interface is designed with user-friendliness in mind, incorporating accessibility features to cater to a diverse user base.

Deployment of the Canteen Food Order System is straightforward, with support for integration with existing systems and services, making it a versatile addition to any canteen's technological infrastructure. This comprehensive solution not only enhances customer convenience but also optimizes overall canteen management, ensuring a smooth, secure, and efficient food ordering experience.

TABLE OF CONTENTS

CHAPTER 1

1.1 INTRODUCTION

The Canteen Management System is an innovative solution designed to enhance and streamline the food ordering process within canteens. Utilizing a Python-based GUI developed with PySimpleGUI for the interface and MongoDB for the backend, this system ensures efficient and reliable canteen operations.

It includes three main components: a client application, a scanner component, and an admin portal. The client application allows customers to place orders and generate QR codes effortlessly. The scanner component validates these QR codes and generates receipts, ensuring a smooth transaction process. The admin portal empowers canteen managers to manage recipes, update prices and quantities, and oversee orders effectively.

Key features of the system include token-based authentication to prevent fake QR codes, special item tagging for promotional offers, and take-away options. These features significantly improve customer convenience and optimize canteen management. This system aims to provide a smooth, secure, and modernized food ordering experience, benefiting both customers and canteen administrators.

1.2 OBJECTIVES

The objective of the Canteen Management System is to revolutionize and enhance the operational efficiency of canteen services through a comprehensive, user-friendly platform. This innovative system aims to simplify the food ordering process for customers, offering a seamless and enjoyable experience from order placement to receipt generation. Utilizing a Python-based GUI developed with PySimpleGUI for the interface, alongside MongoDB for backend support, the system ensures robust performance, secure data handling, and high reliability.

Key objectives include enabling quick and easy order placement via an intuitive client application, which allows customers to generate QR codes for their orders effortlessly. The system ensures order authenticity through a sophisticated QR code validation process, minimizing the risk of fraudulent activities. Additionally, the scanner component efficiently validates these QR codes and generates accurate receipts, streamlining the transaction process within the canteen.

For canteen administrators, the system provides powerful tools to manage recipes, update prices, adjust quantities, and oversee orders with ease. This comprehensive admin portal empowers managers to maintain optimal inventory levels and make informed decisions that enhance the overall efficiency of canteen operations.

The system also seeks to implement several advanced features to further enhance customer satisfaction and operational efficiency. These include token-based authentication to prevent the use of fake QR codes, promotional item tagging for easy identification and marketing of special offers, and take-away order options to cater to the diverse needs of customers. By integrating these features, the system aims to provide a flexible and versatile solution that addresses various customer preferences and operational requirements.

Ultimately, the Canteen Management System aspires to deliver a modernized, secure, and efficient canteen management experience for both customers and administrators. By leveraging cutting-edge technology and thoughtful design, this system aims to set a new standard for canteen services, ensuring a smooth, convenient, and enjoyable experience for all users.

1.3 MODULES

- Admin Management Module.

- Client Management Module.

- Scanner Module.

CHAPTER-2

2.1 SOFTWARE DESCRIPTION

Visual studio Code:

Visual Studio Code combines the simplicity of a source code editor withpowerful developer tooling, like IntelliSense code completion and debugging.

2.2 LANGUAGES

1. Python:

- It is used for scripting the application's logic, managing database operations, and integrating different modules.

2. Streamlit:

- A powerful Python library for creating interactive web applications with simple Python scripts, enabling rapid prototyping and deployment of data-driven applications with ease..

3. MongoDB:

- A scalable NoSQL database solution, offering high performance, flexibility, and seamless integration with modern applications .

CHAPTER-3
REQUIREMENT AND ANALYSIS

3.1 REQUIREMENT SPECIFICATION:

The canteen management system is designed to streamline the operations of a canteen by automating various processes and enhancing user experience. The primary functional requirements of the system include user management, which enables the creation, modification, and deletion of accounts for students, staff, and administrators. Menu management allows canteen staff to efficiently handle daily menus, including adding, updating, and removing items. The order management feature ensures that users can place, modify, and cancel orders seamlessly. The system also supports multiple payment options, including cash, credit/debit cards, and online payments, to facilitate easy and secure transactions. Inventory management is another critical feature, providing real-time tracking of stock levels and notifying staff when items are low. Additionally, the system generates comprehensive reports on sales, inventory, and user activity, aiding in data-driven decision-making. Notifications are integrated to inform users about order confirmations, order readiness, and stock alerts. Non-functional requirements emphasize performance, ensuring the system can handle high volumes of concurrent users efficiently. Usability is prioritized with an intuitive and user-friendly interface. Security measures are in place to protect data, including secure payment processing and safeguarding user information. Reliability is ensured with high system availability and minimal downtime, while scalability allows the system to grow and accommodate increasing numbers of users and expanding functionalities.

## 3.2 HARDWARE AND SOFTWARE REQUIREMENTS:

Hardware Requirements

- Processor: Intel Xeon Processor or equivalent.

- RAM: 16 GB or more

- Storage: At least 500 MB of available disk space

- Display: Minimum resolution of 1024x768

- Input Devices: Keyboard and mouse

Software Requirements :

- Operating System:
- **Server:** Linux (Ubuntu, CentOS) or Windows   Server.
- **Client:** Windows, macOS, Android, iOS

- Python: Version 3.6 or higher

- SQLite: Version 3 or higher

- Python Libraries:

  - 'pymongo' connecting to and interacting with MongoDB (use command pip install pymongo)

  - 'Streamlit' building the web application interface (use command pip install streamlit)

3.3 ARCHITECTURE DIAGRAM:

Canteen Management System Architecture
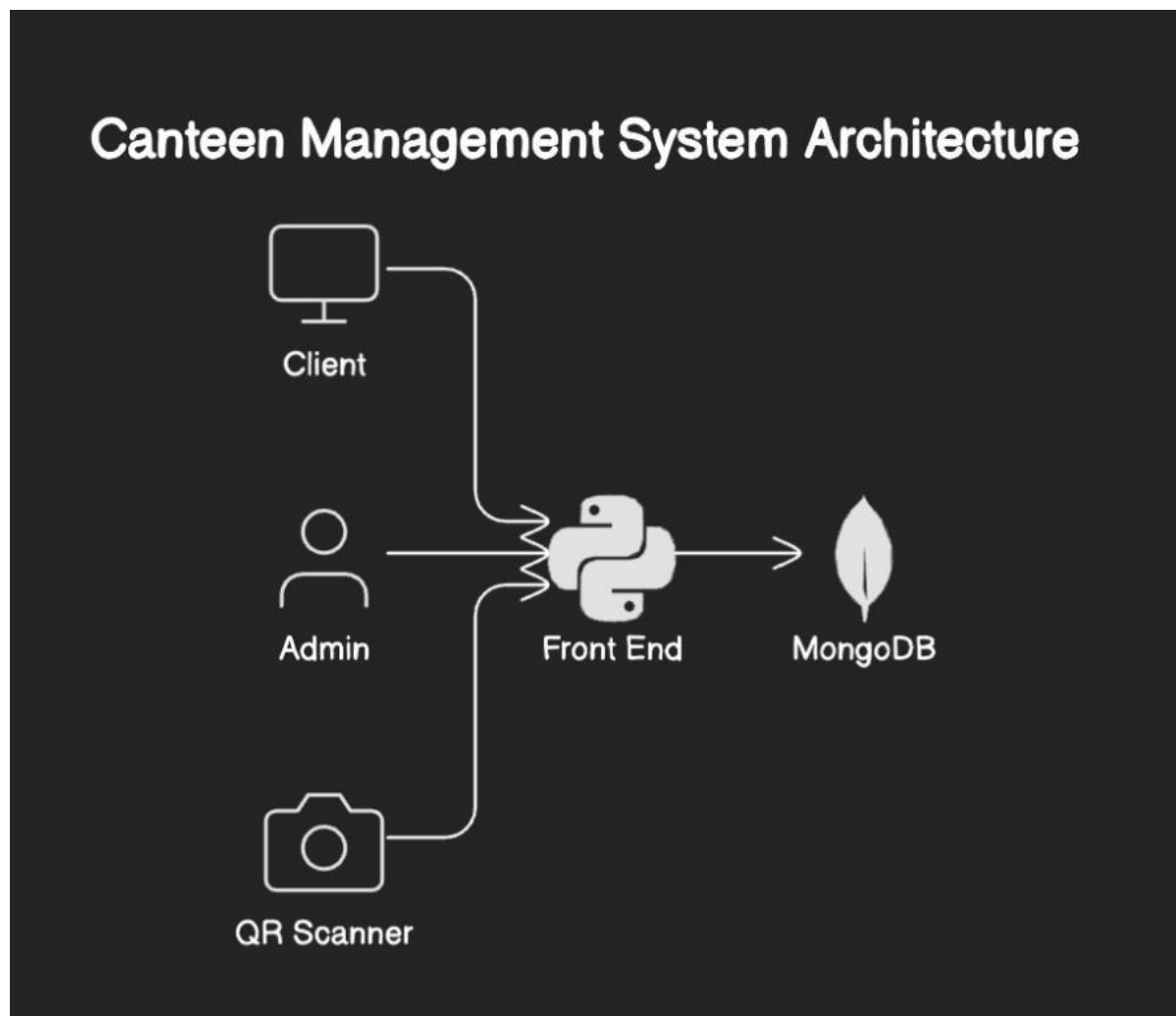
3.4 ER DIAGRAM:
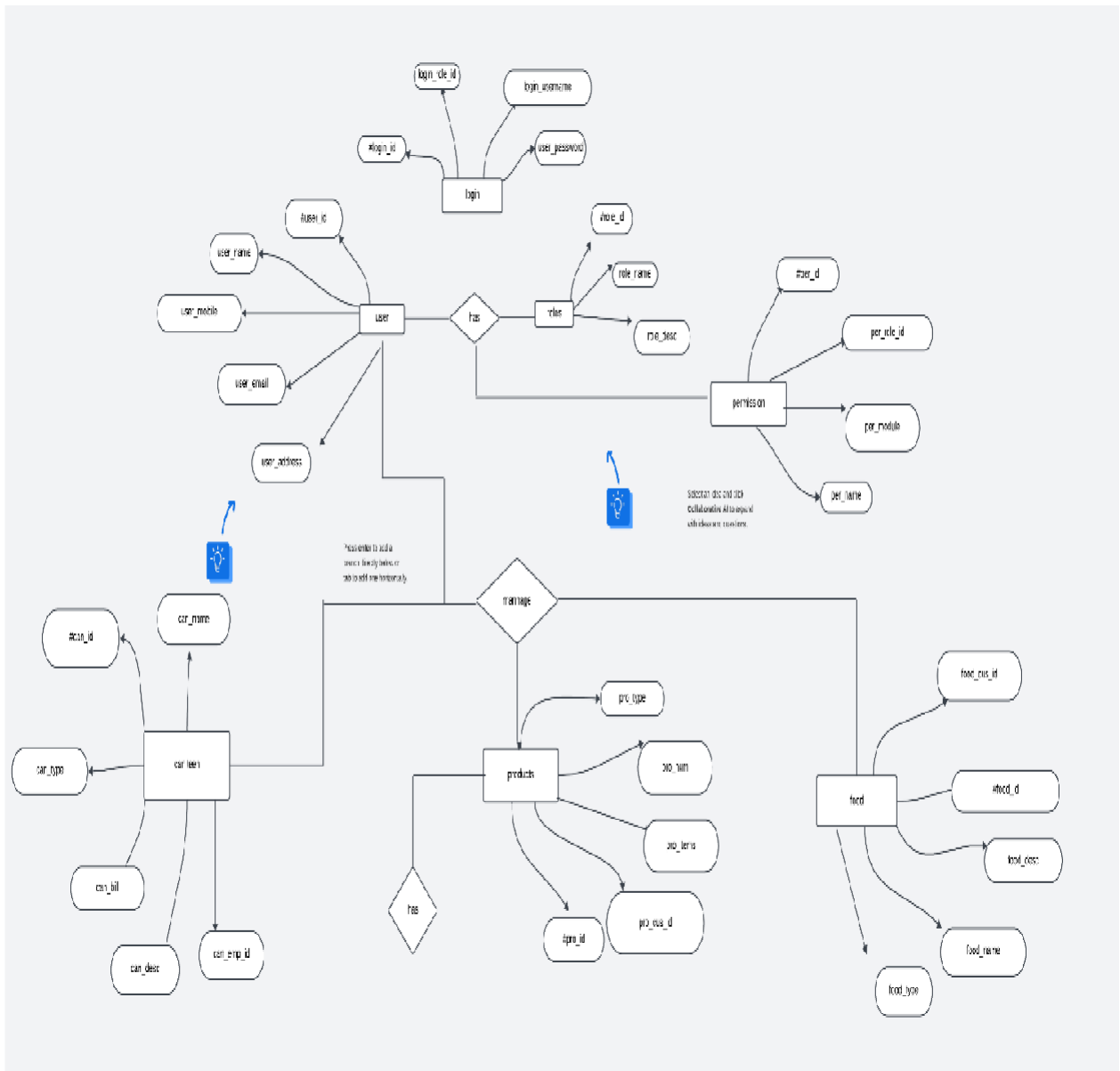
CHAPTER-4

PROGRAM CODE

Admin main program:

```
import secrets
import string
from database.func import
add_recipe,list_recipes,update_recipe_stock,update_recipe_price,remove_recipe,is_recipe,get_toke
ns
import PySimpleGUI as sg
sg.theme("DarkTeal10")
font = ("Helvetica", 13)
sg.set_options(font=font)
recipes={}
all = list_recipes()
for x in all:
    recipes[x.get("recipe")]={"stock":x.get("stock"),"price":x.get("price")}
tokens=get_tokens()
plain_receipes=[]
lst=[]
for x in recipes:
    plain_receipes.append(x)
    nme=x.ljust(24," ")
    nme1=recipes.get(x).get("price").strip().ljust(22,"  ")
    nme2=recipes.get(x).get("stock").strip().ljust(24," ")
    nme3="".ljust(17," ")
    lst.append([sg.Text(nme),sg.Text(nme2,key=f"-{x}_stock-"),sg.InputText("",size=(10),
key=f"{x}_stock"),sg.Text(nme3),sg.Text(nme1,key=f"-{x}_price-"),sg.InputText(size=(10),
key=f"{x}_price")])

layout = [
    [sg.Text('CANTEEN ADMIN PANEL', text_color="Cyan",justification="5")],
```

```python
    [sg.Text("Product\t\tQuantity\t\tNew Quantity\t\tPrice\t\tNew Price")]
]
for x in lst:
    layout.append(x)
layout.append([sg.Button('UPDATE STOCK')])
layout.append([sg.Text(key='-update-', text_color="Lime")])
layout.append([sg.Text(key='-error-', text_color="Red")])
layout.append([sg.Text('Add/Remove Products', text_color="Cyan")])
layout.append([sg.Text("Product\t\t\tQuantity\t\t\tPrice")])
layout.append([sg.InputText(size=(10),key="prod_name"),sg.Text("\t"),sg.InputText(size=(10),key
="prod_quan"),sg.Text("\t"),sg.InputText(size=(10),key="prod_price"),sg.Checkbox('Is it special?',
default=False,key="is_special")])
layout.append([sg.Button('Add Product'),sg.Text("   "),sg.Button('Remove Product')])
layout.append([sg.Text(key='-msggg-', text_color="Lime")])
layout.append([sg.Button('View Orders')])
layout.append([sg.Cancel()])
window = sg.Window('CANTEEN ORDER SYSTEM',
layout,icon=r'C:\Users\chsai\Desktop\folder_locker\enc.ico', size=(1000, 700))
while True:
    event, values = window.read()
    if event is None or event == 'Cancel':
        break
    if event=="View Orders":
        tokens=get_tokens()
        l=0
        for z in tokens:
            l+=1
        my_order=f"Pending Orders Count: {l}\n"
        my_order+="Users Order List is: \n"
        num=1
        tokens=get_tokens()
        for x in tokens:
            my_order+=f"\n\nOrder Number: {num}\n"
            my_order+=f"key: {x.get('key')}\n"
```

```python
        for y in x.get("dict"):
            my_order+=f"{y}     {x.get('dict').get(y)}\n"
        num+=1
    sg.popup_scrolled(my_order)
if event=="Add Product":
    print(values)
    if values['prod_quan'].isdigit() ==False or values['prod_price'].isdigit() ==False:
        window[f"-error-"].update("Quantity and price should be a number")
    else:
        if values['prod_name'].capitalize() in plain_receipes:
            window[f"-error-"].update("Product with same name already exists")
        else:
            add_recipe(values['prod_name'].capitalize(), values['prod_quan'],
values['prod_price'],values['is_special'])
            plain_receipes.append(values['prod_name'].capitalize())
            window[f"-msggg-"].update("Product added successfully")
if event=="Remove Product":
    if is_recipe(values['prod_name'].capitalize()):
        remove_recipe(values['prod_name'].capitalize())
        plain_receipes.remove(values['prod_name'].capitalize())
        window[f"-msggg-"].update("Product Removed successfully")
    else:
        window[f"-msggg-"].update("No such product exists")
if event=="UPDATE STOCK":
    print(values)
    for x in plain_receipes:
        if values[f"{x}_stock"]!=" or values[f"{x}_price"]!=":
            if values[f"{x}_stock"]!=":
                if values[f"{x}_stock"].isdigit() ==False:
                    window[f"-error-"].update("Quantity and price should be a number")
                else:

                    nme2=values.get(f"{x}_stock").ljust(24," ")
                    update_recipe_stock({x:values.get(f"{x}_stock")})
```

```
            window[f"-{x}_stock-"].update(nme2)
            window["-update-"].update("Server updated successfully.")
         if values[f"{x}_price"]!=":
            if values[f"{x}_price"].isdigit() ==False:
               window[f"-error-"].update("Quantity and price should be a number")
            else:
               nme1=values.get(f"{x}_price").ljust(22," ")
               update_recipe_price({x:values.get(f"{x}_price")})
               window[f"-{x}_price-"].update(nme1)
               window["-update-"].update("Server updated successfully.")
```

Client main program:

```
import secrets
import string
from database.func import add,list_recipes,key_info,update_recipe_stock
import PySimpleGUI as sg
import qrcode
from PIL import Image
from tinydb import TinyDB, Query
db = TinyDB('db.json')
def passgen():
    alphabet = string.ascii_letters + string.digits + string.punctuation
    password = ".join(secrets.choice(alphabet) for i in range(60))
    password = password.replace("'", "'")
    return password.replace("'","*")
sg.theme("DarkTeal10")
font = ("Helvetica", 13)
sg.set_options(font=font)
recipes={}
all = list_recipes()
for x in all:
```

```python
        recipes[x.get("recipe")]={"stock":x.get("stock"),"price":x.get("price"),
"is_special":x.get("is_special")}
# print(recipes)
lst=[]
lst_spcl=[]
def_space=15
plain_receipes=[]
for x in recipes:
    plain_receipes.append(x)
    space=len(x)-def_space
    space=abs(space)
    if recipes.get(x).get("is_special")==True:
        nme=x.ljust(24," ")
        nme1=recipes.get(x).get("price").strip().ljust(28,"  ")
        nme2=recipes.get(x).get("stock").strip().ljust(35," ")
        lol = nme+nme1+nme2
        lst_spcl.append([sg.Text(lol,text_color="white"),sg.InputText(size=(10), key=x,)])
    else:
        nme=x.ljust(24," ")
        nme1=recipes.get(x).get("price").strip().ljust(28,"  ")
        nme2=recipes.get(x).get("stock").strip().ljust(35," ")
        lol = nme+nme1+nme2
        lst.append([sg.Text(lol,text_color="white"),sg.InputText(size=(10),   key=x,)])


layout = [
    [sg.Text('FOOD COURT', text_color="cyan",justification="5")],
    [sg.Text('Regular Items',text_color="pink")],
    [sg.Text("Product\t\tPrice\t\tQuantity Avaiable\t\tQuantity Ordering", text_color="yellow")]
]


for x in lst:
    layout.append(x)
layout.append([sg.Text('Special Items', text_color="pink")])
```

```python
layout.append([sg.Text("Product\t\tPrice\t\tQuantity Avaiable\t\tQuantity Ordering",
text_color="yellow")])
for x in lst_spcl:
    layout.append(x)
layout.append([sg.Button('BILL IT',button_color="green")])
layout.append([sg.Button('ANALYSE BILL', button_color="green")])
layout.append([sg.Text(key='-bill-', text_color="yellow")])
layout.append([sg.Text('TOTAL: ', text_color="yellow"), sg.Input("", key='-ORDER-')])
layout.append([sg.Checkbox('take away?', default=False,key="is_parcel")])
layout.append([sg.Button('ORDER',button_color="green")])
layout.append([sg.Text(key='-ttt-', text_color="white")])
layout.append([sg.Button('My Orders')])
layout.append([sg.Cancel(button_color="green")])
window = sg.Window('CANTEEN ORDER SYSTEM',
layout,icon=r'C:\Users\chsai\Desktop\folder_locker\enc.ico', size=(1000, 900))
while True:
    event, values = window.read()
    if event is None or event == 'Cancel':
        break
    if event=="My Orders":
        my_order="Your Order List is: \n"
        num=1
        for item in db:
            p = key_info(item.get("key"))
            if p:
                if num<1:
                    my_order+=f"Order Number: {num}\n"
                else:
                    my_order+=f"\n\nOrder Number: {num}\n"
                num+=1
                for x in p.get("dict"):
                    my_order+=f"{x}     {p.get('dict').get(x)}\n"
                my_order+=f"Key: {p.get('key')}"
        sg.popup_scrolled(my_order)
```

```python
    if event=="BILL IT":
        vle=0
        for x in plain_receipes:
            if (values[x])!='':
                vle+=int(values[x])*int(recipes.get(x).get("price"))
        window['-ORDER-'].update(vle)
    if event=="ANALYSE BILL":
        vle=0
        for x in plain_receipes:
            if (values[x])!='':
                vle+=int(values[x])
        if vle==0:
            window["-bill-"].update("NOTHING TO ANALYSE, CART IS EMPTY.")
        else:
            txet="BILL ANALYSIS\nPRODUCT NAME\t\tRATE\t\tQUANTITY\t\tPRICE"
            for x in plain_receipes:
                if (values[x])!='':

txet+=f'\n{x.upper()}\t\t\t{int(recipes.get(x).get("price"))}\t\t{int(values[x])}\t\t\t{int(values[x])*int(recipes.get(x).get("price"))}'

            window["-bill-"].update(txet)
    if event=="ORDER":
        overflow=False
        lst={}
        stock_update={}
        for x in plain_receipes:
            if (values[x])!='':
                if int(values[x])>int(recipes.get(x).get("stock")):
                    window["-ttt-"].update(f"{x}'s entered stock is more than avaiable")
                    overflow=True
                    break
                else:
                    lst[x]=int(values[x])
```

```python
                stock_update[x]=int(recipes.get(x).get("stock"))-int(values[x])
                overflow=False
    if not overflow:
        key = passgen()
        if len(lst) == 0:
            window["-ttt-"].update("Cart is empty")
        else:
            window["-ttt-"].update("Cart is empty")
            add(lst,key,values['is_parcel'])
            order = {"key":key, "dict":lst, "is_parcel":str(values['is_parcel'])}
            db.insert({"key":key})
            # print(order)
            print(stock_update)
            update_recipe_stock(stock_update)
            qr_img = qrcode.make(str(order))
            qr_img.save("qr-img.jpg")
            im = Image.open(r"qr-img.jpg")
            im.show()
```

Scanner main program:

```python
import secrets
import string
from database.func import add,checker,update_valid,is_valid
import PySimpleGUI as sg
import cv2
import numpy as np
from pyzbar.pyzbar import decode
import json


def decoder(image):
    gray_img = cv2.cvtColor(image,0)
    barcode = decode(gray_img)


    for obj in barcode:
        points = obj.polygon
        (x,y,w,h) = obj.rect
        pts = np.array(points, np.int32)
        pts = pts.reshape((-1, 1, 2))
        cv2.polylines(image, [pts], True, (0, 255, 0), 3)


        barcodeData = obj.data.decode("utf-8")
        barcodeType = obj.type
        return barcodeData

sg.theme("DarkGreen")
layout = [
    [sg.Text('CANTTEN SCANNER', text_color="Red")],
    [sg.Button('SCAN QR')],
    [sg.Text(key='-TXT-', text_color="Red")],
    [sg.Cancel()],
]
```

```python
window = sg.Window('CANTEEN ORDER SYSTEM',
layout,icon=r'C:\Users\chsai\Desktop\folder_locker\enc.ico', size=(500, 500))
while True:
    event, values = window.read()
    if event is None or event == 'Cancel':
        break
    if event == "SCAN QR":
        cap = cv2.VideoCapture(0)
        while True:
            ret, frame = cap.read()
            lol = decoder(frame)
            cv2.imshow('Image', frame)
            code = cv2.waitKey(10)
            if lol:
                cv2.destroyWindow('Image')
                break
        lol=lol.replace("'", '"')
        print(lol)
        lol = json.loads(lol)
        key=lol.get("key")
        res=lol["dict"]
        is_parcel=lol.get("is_parcel")
        if checker(key) and is_valid(key).get("isvalid")==True:
            txt="RECEIPES ORDER LIST:\n\nRECEIPE\t\tQUANTITY\n"
            for x in res:
                txt+=f"{x}\t\t{res.get(x)}\n"
            if is_parcel=="True":
                txt+="\nOrder Type: Take Away"
            else:
                txt+=f"\nOrder Type: Dine-IN"
            print(txt)
            update_valid(key)
        else:
            print("OR is used or invalid")
```

CHAPTER-5

RESULTS AND DISCUSSION

5.1 USER DOCUMENTATION:

ADMIN MANAGEMENT MODULE:

CLIENT MANAGEMENT MODULE:

📄 CANTEEN ORDER SYSTEM

FOOD COURT

Regular Items

Product	Price	Quantity Avaiable	Quantity Ordering
Pizza	100	12	

Special Items

Product	Price	Quantity Avaiable	Quantity Ordering
Burger	400	19	
Mojito	200	10	

BILL IT

ANALYSE BILL

TOTAL:

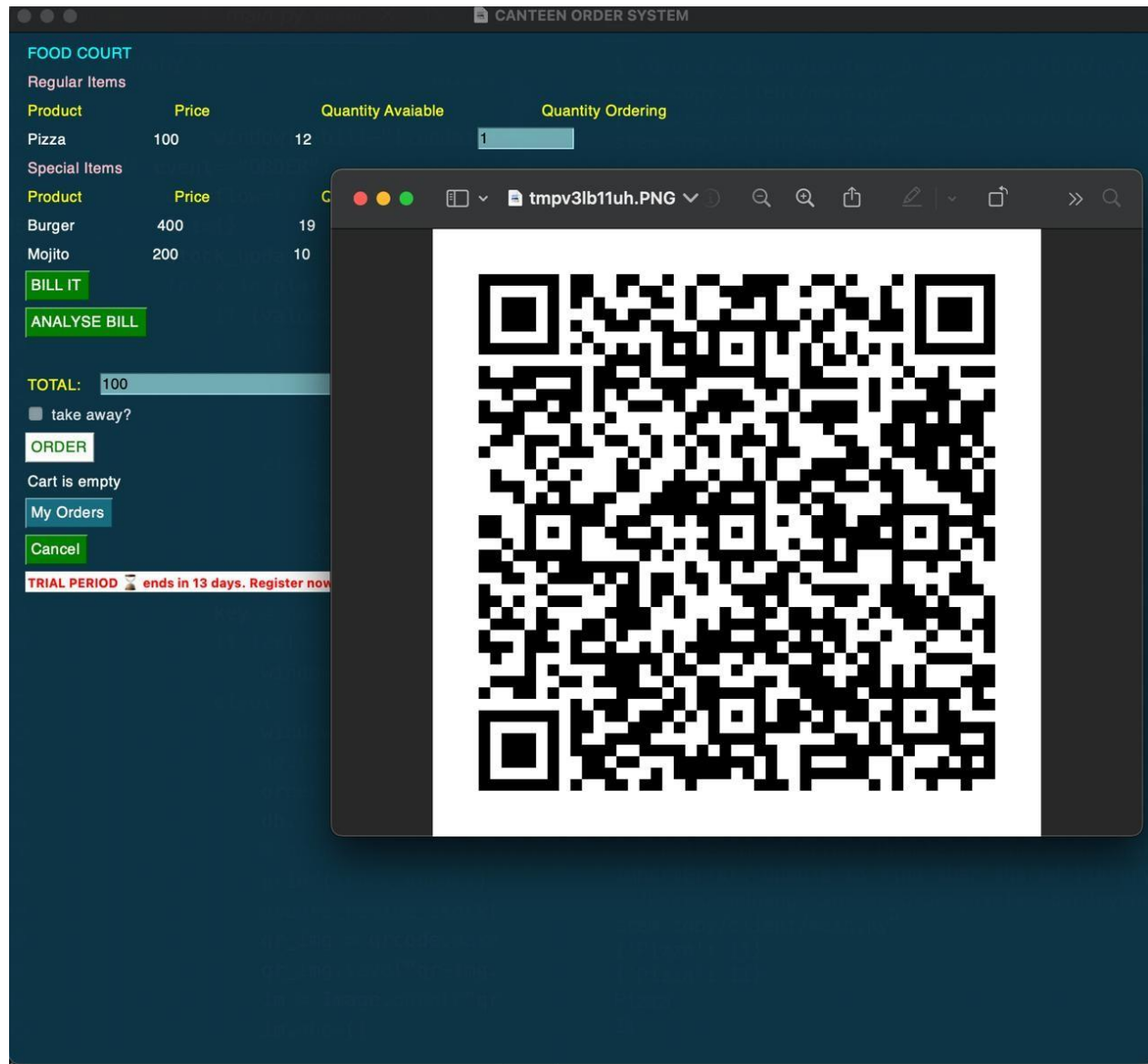☐ take away?

ORDER

My Orders

Cancel

TRIAL PERIOD ⏳ **ends in 13 days. Register now.**

SCANNER MODULE:

CHAPTER-6

6.1 CONCLUSION:

In conclusion, the Canteen Management System exemplifies the transformative impact of automation and modern technology on traditional canteen operations. This system successfully offers a modernized, secure, and efficient management experience that significantly benefits both customers and administrators. By reducing human errors and automating routine tasks, the system enhances operational efficiency and accuracy. Customers benefit from an easy-to-use client application that generates QR codes for order retrieval, while the scanner component ensures the authenticity of orders and generates final receipts, streamlining the entire transaction process.

Administrators gain powerful tools to manage recipes, update prices, and oversee orders, allowing them to focus on more strategic activities rather than routine tasks. The integration of token-based authentication prevents fraudulent activities, ensuring a secure transaction environment. Promotional item tagging and take-away order options further enhance customer satisfaction by catering to diverse needs and preferences.

Moreover, the system's robust performance, ensured by the use of PySimpleGUI for the interface and MongoDB for backend support, demonstrates the practical application of database management systems (DBMS) in solving real-world challenges. The secure data handling and consistent high performance of the system highlight the effectiveness of these modern technologies in creating a reliable and efficient solution.

This project underscores the value of leveraging advanced software solutions to address practical problems and improve service delivery in canteen settings. By modernizing the food ordering and management processes, the Canteen Management System sets a new standard for canteen services, ensuring a smooth, convenient, and enjoyable experience for all users.

CHAPTER-7

7.1 REFERENCES:

1. Python Documentation: Python Software Foundation. (n.d.). Python Documentation. Retrieved from https://docs.python.org/3/

2. Tkinter Documentation: TkDocs. (n.d.). Tkinter Tutorial. Retrieved from https://tkdocs.com/tutorial/

3. SQLite Documentation: SQLite. (n.d.). SQLite Documentation. Retrieved from https://www.sqlite.org/docs.html

4. Automate the Boring Stuff with Python: Sweigart, A. (2015). Automate the Boring Stuff with Python: Practical Programming forTotal Beginners. No Starch Press.

5. Python GUI Programming with Tkinter: Grayson, J. E. (2000). Python and Tkinter Programming. Manning Publications.

6. SQLite Database Programming: Owens, M. (2006). The Definitive Guide to SQLite. Apress.

7. Database System Concepts: Silberschatz, A., Korth, H. F., &Sudarshan, S. (2010). Database System Concepts (6th ed.). McGraw-Hill.

8. GeeksforGeeks: Various authors. (n.d.). GeeksforGeeks. Retrieved from https://www.geeksforgeeks.org/

9. W3Schools: W3Schools. (n.d.). SQL Tutorial. Retrieved from https://www.w3schools.com/sql/