SYNOPSIS

SYNOPSIS:

The Restaurant Billing System is a Python-based desktop application built using Tkinter for the graphical user interface and MySQL for database management. It allows restaurant staff to manage customer orders, generate bills, and maintain order records efficiently. The application consists of multiple sections, including customer details, menu selection, billing summary, and additional utility functions. The menu is categorized into starters, main courses, and snacks, with each item linked to input fields for quantity selection. The system dynamically calculates the total bill, including applicable taxes, and displays a detailed bill summary in a text area. It integrates a MySQL database to store order details, ensuring data persistence. Customers receive a unique bill number, automatically generated at the time of order placement. A QR code feature is included to encode bill details for digital access. Additional functionalities like a built-in calculator and a computer vision module utilizing OpenCV enhance usability. The system is designed with a modern graphical layout, featuring intuitive widgets for ease of use. The bill generation process updates the interface in real-time, ensuring a seamless user experience. The order details are stored in a structured format in the database for future reference. The bill area dynamically lists purchased items along with their respective quantities and prices. A clear fields button resets all input fields, allowing new order entries without restarting the application. The application ensures smooth workflow management for restaurants by streamlining the billing process. It provides error handling mechanisms, including customer detail validation before bill generation. A simple exit function is implemented to close the application and safely disconnect from the database. The graphical interface is color-coded with distinct sections for better readability and interaction. The use of Tkinter widgets, including labels, entry fields, and buttons, ensures an organized layout. The program employs object-oriented programming principles, encapsulating functionalities within a structured class-based design. This project demonstrates a practical implementation of Python GUI programming integrated with a database-driven approach.

CONTENTS

INTRODUCTION

1 INTRODUCTION

The Restaurant Billing System is a Python-based desktop application designed to streamline the billing process for restaurants. Built using Tkinter for the graphical user interface (GUI), MySQL for database management, and OpenCV for computer vision functionalities, this system ensures an efficient, user-friendly experience for restaurant staff and customers alike. The application allows users to input customer details, select menu items, calculate totals, generate invoices, and store order details in a secure database.

One of the key features of this system is its well-organized menu section, divided into categories such as Starters, Main Course, and Snacks, making it easy for users to input order quantities. Each item has a predefined price, and the system automatically calculates the total amount, including applicable taxes. To enhance usability, the program also includes a QR code generator, allowing customers to scan and quickly access their bill details.

For ease of navigation, the system includes additional features such as a built-in calculator, a clear fields button to reset order entries, and a real-time computer vision module that uses OpenCV to capture webcam footage. Furthermore, all transactions are stored in a MySQL database, ensuring data persistence and easy retrieval of past orders.

The system is designed with aesthetic and functional UI elements, featuring an intuitive layout, colorful labels, and organized widgets. The billing area displays itemized details, helping restaurant employees verify order details before generating an invoice. The order-saving functionality prevents data loss, and the system prompts users with alerts and notifications to ensure smooth operations.

This project demonstrates an integration of Python programming, GUI design, database connectivity, and real-time image processing, making it a comprehensive solution for restaurant billing. Whether for small eateries or large dining establishments, this system optimizes the billing workflow, reduces manual errors, and improves overall efficiency.

2.0 SYSTEM SPECIFICATION

2.1 Hardware Requirements:

- Processor: Intel Core i3 or higher
- RAM: Minimum 4GB (8GB recommended)
- Storage: Minimum 500GB HDD or 128GB SSD
- Display: 1366x768 resolution or higher
- Peripherals: Keyboard, Mouse
- Camera: Required for Computer Vision feature

2.2 Software Requirements:

- Operating System: Windows 7/8.1/10/11 (64-bit)
- Database: MySQL (XAMPP recommended for local setup)
- Python Version: Python 3.x
- Required Libraries:
  - `mysql-connector-python` (For database connection)
  - `tkinter` (For GUI)
  - `random` (For bill number generation)
  - `qrcode` and `PIL` (For QR Code generation)
  - `OpenCV (cv2)` (For Computer Vision feature)

SYSTEM STUDY

SYSTEM STUDY

3.0 EXISTING SYSTEM

The existing system is a desktop-based Restaurant Billing System developed using Python, Tkinter, MySQL, and OpenCV. It provides an efficient way to manage restaurant orders, generate customer bills, and store billing records in a MySQL database. The system features a user-friendly graphical interface that allows restaurant staff to input customer details, select ordered items, calculate the bill including applicable taxes, generate a detailed bill receipt, and store the transaction in the database. Additionally, it incorporates features such as QR code generation, a calculator, and basic computer vision functionality using OpenCV for real-time camera access.

3.1 Key Features:

- Graphical User Interface (GUI): Built using Tkinter for ease of use.
- Billing and Order Management: Calculates itemized costs, tax, and total amounts.
- Database Integration: MySQL is used to store order details.
- Bill Printing & Display: Displays the bill in a text area within the interface.
- QR Code Generation: Generates QR codes for billing information.
- Calculator Integration: A built-in calculator for manual calculations.
- Computer Vision Module: Opens the webcam for real-time video capture.
- Data Persistence: Stores billing information for future reference.

3.2 Drawbacks:

Despite its features, the current system has several limitations that may affect its efficiency and usability:

- Limited Scalability: The system is designed as a standalone desktop application, making it difficult to scale for multiple users or integrate with cloud-based solutions.
- Basic UI Design: While functional, the Tkinter interface lacks modern design aesthetics and advanced UX features.

- Manual Data Entry: Customers and orders must be entered manually, increasing the chance of human errors.
- No Inventory Management: The system does not track stock levels, leading to potential inconsistencies in item availability.
- Lack of Payment Integration: The system does not support digital payments, requiring manual payment processing.
- Basic Computer Vision Functionality: The OpenCV module is underutilized, with no significant role in billing operations.
- Limited Reporting & Analytics: No built-in reporting tools for business insights, sales trends, or inventory tracking.
- No Multi-User Access: It cannot handle multiple simultaneous users, limiting its use in larger restaurants.
- No Cloud Backup: The system relies on local database storage without cloud backup options, risking data loss.

3.3 Proposed System

The **Restaurant Billing System** is an advanced, user-friendly application designed to streamline the restaurant billing process. Built using **Python, Tkinter, MySQL, and OpenCV**, the system allows restaurant owners to efficiently manage customer orders, generate bills, and store transaction records in a structured database.

3.3.1 Features of the Proposed System

- Graphical User Interface (GUI) – Developed using Tkinter, ensuring an intuitive and user-friendly experience.
- Customer Details Management – Allows entry and validation of customer name and contact information.
- Menu-Based Ordering – Categorized into Starter, Main Course, and Snacks for easy order selection.
- Automatic Bill Calculation – Computes the total bill, including applicable taxes, for quick and accurate billing.
- Database Integration – Orders and customer details are stored securely in a MySQL database for future reference.
- Bill Generation Area – Displays the bill summary dynamically, including itemized details.

- QR Code Generation – Generates QR codes for each transaction, ensuring secure and contactless bill verification.
- Clear and Reset Functionality – Provides an option to reset order details with a single click.
- Built-in Calculator – Facilitates quick calculations for additional charges or discounts.
- Computer Vision Module – Integrates OpenCV for real-time video capture, which can be extended for security or facial recognition.
- Error Handling and Validation – Ensures customer details are entered before processing the bill.
- Modern UI Design – Uses color-coded frames and structured placement of elements for an enhanced visual appeal.
- Optimized Performance – Designed for smooth execution with minimal lag, even on lower-end systems.
- Easy Exit and Safe Closure – Proper database disconnection and resource management when the application is closed.
- Scalability and Future Enhancements – Can be further improved by adding online payment integration and advanced analytics.

SYSTEM DESIGN AND DEVELOPMENT

4.0 SYSTEM DESIGN AND DEVELOPMENT

4.0. File Design

The system comprises multiple files that handle different aspects of the billing system:

- Main Python Script (`billing_system.py`): Contains the core logic, GUI design, and database connectivity.
- Database Schema (`restaurant.sql`): Defines table structures for orders, menu items, and customer details.
- Image Files (QR Codes, Logos): Stores generated QR codes and restaurant branding assets.
- Configuration Files: Can store user settings, themes, and preferences for future customization.

4.1. Input Design

The system accepts input via the Tkinter GUI, where users can enter:

- Customer Information: Name and phone number for order records.
- Item Quantities: Users can specify the number of ordered items via entry fields.
- Bill Number: Auto-generated for unique order tracking.
- Payment Details: The system can be expanded to include different payment methods.

4.2 Output Design

The system provides output in the following forms:

- Graphical User Interface (GUI): Displays order details, total bill, and billing options.
- Billing Receipt (Text Area): Shows itemized order details with quantities and prices.
- QR Code Generation: Displays a QR code with order details for quick scanning.
- Database Storage: Saves order details for future reference.
- Printed Bill (Future Scope): Can be extended to print physical receipts

4.3 Code Design

The program follows an Object-Oriented Programming (OOP) approach:

- Class-Based Structure: The `Bill_App` class encapsulates all functionalities.
- Modular Functions: Methods like `total_bill()`, `clear_fields()`, and `save_to_database()` ensure maintainability.
- GUI Components: Tkinter is used for a structured, user-friendly interface.
- Error Handling: Try-except blocks handle database errors and invalid inputs.
- External Libraries: Uses OpenCV for computer vision, QR code generation, and MySQL for data management.

4.4 Database Design

The database (`restaurant` in MySQL) consists of:

- Orders Table (`orders`): Stores bill number, customer details, and total price.
  - Fields: `order_id (PK)`, `bill_no`, `customer_name`, `phone`, `total_amount`
- Menu Table (`menu`): Stores item names, prices, and categories.
  - Fields: `item_id (PK)`, `name`, `price`, `category`
- Future Enhancements:
  - User authentication for multi-user access.
  - Integration with an inventory management system.

System Overview

The system allows restaurant staff to generate bills, store customer details, calculate totals with tax, and maintain an order history in a MySQL database. The graphical interface is built using Tkinter, ensuring an interactive and seamless experience.

System Modules

The system is divided into several functional modules:

1. User Interface Module

- Designed using Tkinter for a structured and interactive layout.
- Provides entry fields, buttons, and labels for billing operations.
- Allows users to input customer details, select items, and view bills.

2. Product and Order Management Module

- Defines various menu items categorized into Starter, Main Course, and Snacks.
- Uses Tkinter Entry fields linked to integer variables for tracking quantities.
- Dynamically updates the billing area based on item selection.

3. Billing and Calculation Module

- Computes total price based on selected items and quantities.
- Implements **tax calculations** for different categories.
- Displays the final amount in the bill area.

4. Database Management Module

- MySQL database stores customer details, order history, and total amounts.
- SQL queries handle order insertion into the database.
- Ensures data integrity and easy retrieval of records.

5. Bill Generation and Display Module

- Creates a detailed bill format using Tkinter Text Widget.
- Lists purchased items, quantities, and final price.
- Allows staff to print or save bill details.

6. QR Code Generation Module

- Generates a QR code with billing information using the qrcode library.
- Displays the QR code in a separate window for scanning.

7. Computer Vision Module

- Integrates OpenCV to open and process live webcam feed.
- Provides an additional layer of functionality for future expansions.

8. Utility Functions

- Calculator Access: Opens the system calculator for quick computations.
- Clear Fields: Resets all input fields and bill details.
- Exit Application: Closes the database connection and exits the application.

TESTING AND IMPLEMENTATION

5.0 TESTING AND IMPLEMENTATION

Testing and implementation are crucial phases in the software development lifecycle, ensuring that the application meets functional and non-functional requirements. The Restaurant Billing System underwent rigorous testing to identify and resolve defects before deployment.

5.1 Testing Methodologies

To ensure the reliability and accuracy of the system, the following testing methodologies were applied:

a. Unit Testing

- Each function and module were tested separately to verify their correctness.
- Functions like `total_bill()`, `save_to_database()`, and `update_bill_area()` were tested independently.

b. Integration Testing

- Ensured that different modules interact correctly.
- Verified database connectivity, UI components, and button functionalities.

c. System Testing

- Conducted end-to-end testing to validate the overall performance.
- Ensured that the system handled various inputs correctly.

d. User Acceptance Testing (UAT)

- Real users tested the system in a simulated restaurant environment.
- Feedback was collected for improvements before final deployment.

5.2 Implementation Details

a. System Requirements

- Hardware: Windows 8.1 or higher, 4GB RAM, and 500MB free disk space.
- Software: Python, Tkinter, MySQL, OpenCV, PIL, and QR Code library.

b. Database Implementation

- The system was connected to a MySQL database.
- SQL queries ensured seamless data retrieval and storage.

c. GUI Implementation

- Tkinter was used to create an interactive user interface.
- Components such as labels, entry fields, and buttons were properly aligned for usability.

d. Security Measures

- User input validation was implemented to prevent SQL injection.
- Only valid customer details were stored in the database.

5.3 Test Cases

Test ID	Test Scenario	Expected Output	Actual Output	Status
TC_01	Enter customer details and generate bill	Bill generated successfully	Bill generated successfully	Pass
TC_02	Leave customer details empty and try generating bill	Error message displayed	Error message displayed	Pass
TC_03	Check database entry after billing	Order saved successfully	Order saved successfully	Pass
TC_04	Generate QR code	QR code displayed	QR code displayed	Pass
TC_05	Open calculator	Calculator opens	Calculator opens	Pass
TC_06	Start webcam and display video feed	Video feed displayed	Video feed displayed	Pass

5.4 Bug Fixes and Improvements

- Fixed UI alignment issues.
- Optimized database queries for better performance.
- Improved error handling and message prompts.
- Enhanced user experience by adding additional validation checks.

5.5 Performance Testing

- The application was tested with multiple orders and high data loads.
- Performance remained stable with large numbers of transactions.

CONCLUSION

6.0 CONCLUSION

The development of the Restaurant Billing System has been a significant step in automating and streamlining the billing process for restaurants. The system, designed using Python and Tkinter for the user interface, along with MySQL for database management, successfully meets the requirements of an efficient and user-friendly billing system. Through this project, we have been able to create a functional solution that reduces manual errors, enhances customer experience, and improves the overall workflow of a restaurant's billing process.

One of the key highlights of this project is the seamless integration of database connectivity, ensuring that each transaction is recorded securely for future reference. This feature not only improves data management but also helps in generating reports and maintaining customer records. The use of OpenCV for implementing a real-time computer vision module adds an innovative touch, potentially expanding the system's future applications.

Additionally, the QR code generation feature enhances the system's usability by offering a digital means to access and store bill details. The inclusion of tax calculations, total bill computation, and categorized menu sections ensures that the system meets the practical needs of restaurant owners and customers alike.

During the development process, we encountered several challenges, including database handling, UI optimization, and ensuring smooth integration of different components. However, through continuous testing and refinement, we were able to overcome these issues, leading to a more robust and efficient system.

This project has provided valuable insights into the practical implementation of software development techniques, particularly in GUI design, database management, and integration of external libraries.

In conclusion, the Restaurant Billing System is a practical and efficient solution that can significantly contribute to the smooth functioning of restaurant operations. With potential enhancements such as mobile app integration, cloud storage, and AI-powered recommendation systems, this project can evolve into an even more powerful tool in the hospitality industry.

BIBLIOGRAPHY

7.0 BIBLIOGRAPHY

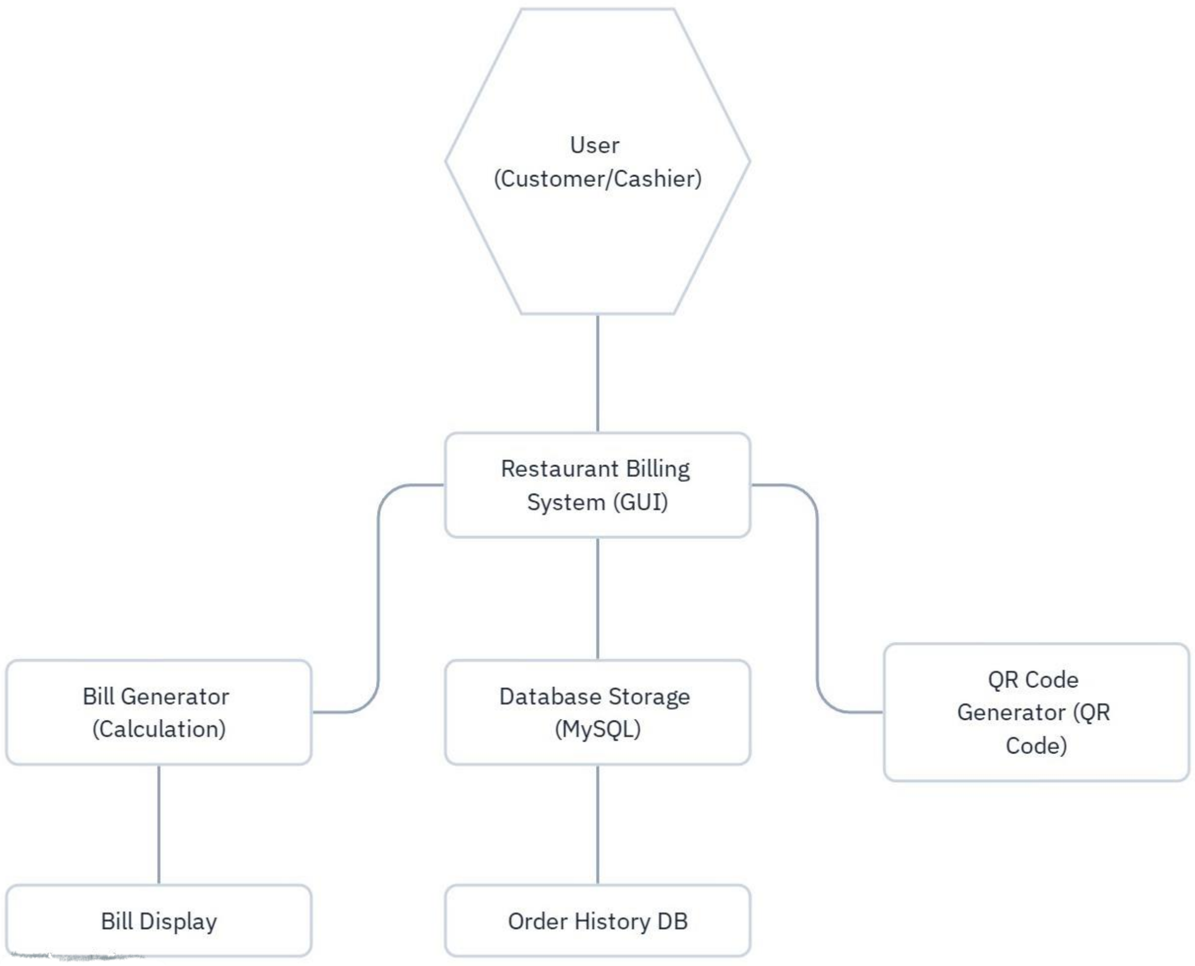
- Tanenbaum, A. S. (2014). Modern Operating Systems. Pearson Education.
- Guttag, J. (2013). Introduction to Computation and Programming Using Python. MIT Press.
- Grayson, J. (2000). Python and Tkinter Programming. Manning Publications.
- Lutz, M. (2013). Learning Python. O'Reilly Media.
- Beazley, D. M. (2009). Python Essential Reference. Addison-Wesley.
- Pilgrim, M. (2004). Dive into Python. Apress.
- Martelli, A. (2006). Python in a Nutshell. O'Reilly Media.
- Downey, A. (2012). Think Python: How to Think Like a Computer Scientist. O'Reilly Media.
- Hussain, A. (2018). Python Programming: A Practical Approach. Mercury Learning and Information.
- Sprankle, M. (2012). Problem Solving and Programming Concepts. Pearson.
- Tkinter Documentation - https://docs.python.org/3/library/tkinter.html
- MySQL Documentation - https://dev.mysql.com/doc/
- OpenCV Documentation - https://docs.opencv.org/
- QR Code Generation with Python - https://pypi.org/project/qrcode/
- Stack Overflow - https://stackoverflow.com/
- GeeksforGeeks: Python GUI with Tkinter - https://www.geeksforgeeks.org/python-gui-tkinter/
- W3Schools: Python MySQL Tutorial - https://www.w3schools.com/python/python_mysql_getstarted.asp
- Real Python: Working with Databases in Python - https://realpython.com/python-sql-libraries/
- Python.org: Official Python Documentation - https://www.python.org/doc/
- Coursera: Python for Everybody - https://www.coursera.org/specializations/python
- MDN Web Docs: Web Development Basics - https://developer.mozilla.org/
- O'Reilly Media: Python for Data Analysis - https://www.oreilly.com/library/view/python-for-data/
- Microsoft Docs: Windows GUI Development - https://learn.microsoft.com/en-us/
- GitHub: Python Project Repositories - https://github.com/topics/python
- Kumar, R. (2018). GUI Programming with Tkinter. Packt Publishing.
- MySQL Performance Tuning Guide - https://www.mysql.com/why-mysql/performance/
- LinkedIn Learning: Python and GUI Development

APPENDICES

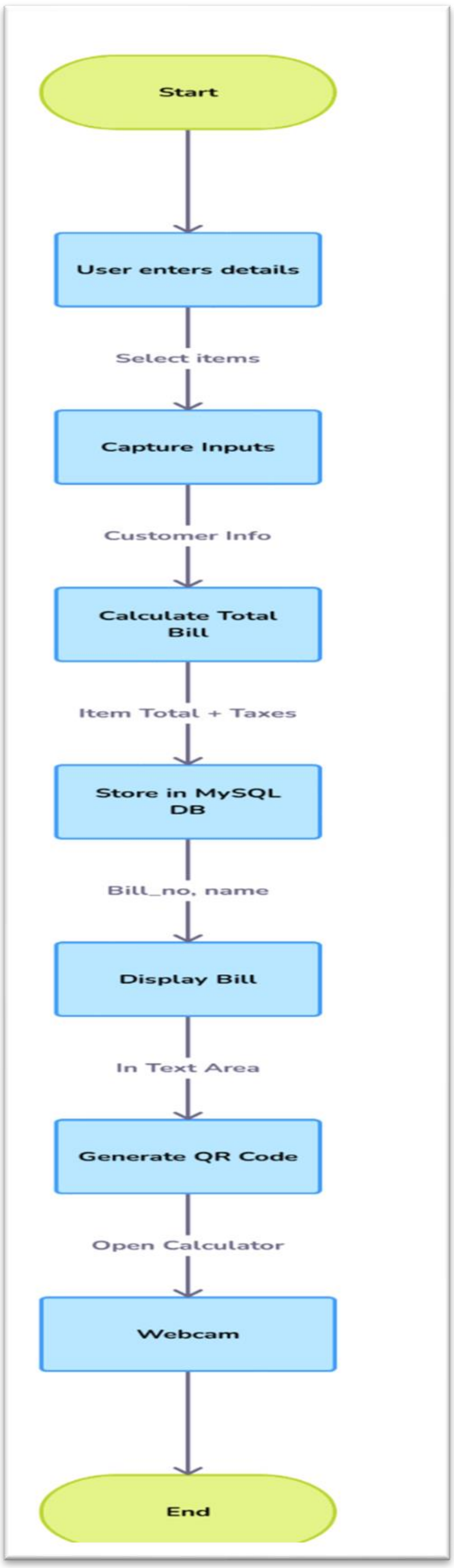
8.0 APPENDICS

8.1. Data Flow Diagram (DFD)

Zero level data flow diagram:


User (Customer/Cashier)
Restaurant Billing System (GUI)
Bill Generator (Calculation)
Database Storage (MySQL)
QR Code Generator (QR Code)
Bill Display
Order History DB

Level one data flow diagram:



8.2. Table Structure

Orders Table

Column Name	Data Type	Description
bill_no	INT (Primary Key)	Unique bill number
customer_name	VARCHAR(255)	Name of the customer
phone	VARCHAR(15)	Customer's contact number
starter_total	DECIMAL(10,2)	Total cost of starter items
main_total	DECIMAL(10,2)	Total cost of main course items
snacks_total	DECIMAL(10,2)	Total cost of snacks
total_amount	DECIMAL(10,2)	Final bill amount

8.3. Sample Coding Content

Database Connection Code

```
import mysql.connector

def connect_db():
    try:
        mydb = mysql.connector.connect(
            host="localhost",
            user="root",
            password="root",
            database="restaurant"
        )
        return mydb
    except Exception as e:
        print("Database connection error:", str(e))
        return None
```

Order Insertion Query

```
def save_order(bill_no, customer_name, phone, starter_total, main_total,
snacks_total, total_amount):
    db = connect_db()
    if db:
        cursor = db.cursor()
        query = """
        INSERT INTO orders (bill_no, customer_name, phone, starter_total,
main_total, snacks_total, total_amount)
        VALUES (%s, %s, %s, %s, %s, %s, %s)
        """
        values = (bill_no, customer_name, phone, starter_total, main_total,
snacks_total, total_amount)
        cursor.execute(query, values)
        db.commit()
        db.close()
```

Billing Calculation Code

```
def calculate_total(prices, quantities):
    total = sum(prices[i] * quantities[i].get() for i in range(len(prices)))
    tax = total * 0.05  # Example tax rate of 5%
    return round(total + tax, 2)
```

QR Code Generation

```
import qrcode

def generate_qr(data):
    qr = qrcode.QRCode(version=1, box_size=5, border=2)
    qr.add_data(data)
    qr.make(fit=True)
    img = qr.make_image(fill_color="black", back_color="white")
    img.show()
```

OpenCV Integration for Computer Vision

```
import cv2

def open_camera():
    cap = cv2.VideoCapture(0)  # Open webcam
    while True:
        ret, frame = cap.read()
        if not ret:
            break
        cv2.imshow("Live Feed", frame)
        if cv2.waitKey(1) & 0xFF == ord('q'):  # Press 'q' to exit
            break
    cap.release()
    cv2.destroyAllWindows()
```