RESTAURANT MANAGEMENT SYSTEM

PROJECT REPORT

Submitted by:

GOPINATH R **220701076**

HARI BALAJI J.C. **220701080**

In partial fulfilment for the award of the degree of

BACHELOR OF ENGINEERING

IN

COMPUTER SCIENCE

RAJALAKSHMI ENGINEERING COLLEGE (AUTONOMOUS),

THANDALAM,

CHENNAI - 602105

2023 – 2024

BONAFIDE CERTIFICATE

Certified that this project '**RESTAURANT MANAGEMENT SYSTEM'** Is the bonafide work of '**GOPINATH R (220701076), HARI BALAJI J.C. (220701080)**' who carried out this project under my supervision.

SIGNATURE
Dr. R. SABITHA
ACADEMIC HEAD
PROFESSOR
Dept. of Computer of Science Engg.,
Rajalakshmi Engineering College,
Chennai.

SIGNATURE
Dr. G. DHARANI DEVI
ASSOCIATE PROFESSOR,
Dept. of Computer of Science Engg.,
Rajalakshmi Engineering College,
Chennai.

Submitted of the Practical Examination held on _____

INTERNAL EXAMINER **EXTERNAL EXAMINER**

ABSTRACT

The Restaurant Management System (RMS) is a web-based application built using Streamlit and MongoDB. It provides functionalities for managing menu items, reservations, orders, staff, and billing. The system includes features for user authentication, dashboard visualization, and CRUD operations for each management aspect. To enhance usability and reliability, additional error handling, input validation, and user feedback could be implemented. Overall, with further refinement and testing, the RMS offers a comprehensive solution for restaurant owners to streamline operations and improve customer satisfaction.

Key functionalities of Restaurant management system include:

- Menu Management: Allows adding, updating, and deleting menu items with their respective prices.

- Reservation Handling: Facilitates making, viewing, updating, and deleting reservations based on customer preferences.

- Order Processing: Enables the addition, viewing, updating, and deletion of customer orders, including items and quantities.

- Staff Management: Supports the management of staff members, including their names, positions, and contact information.

TABLE OF CONTENTS

1. INTRODUCTION

1.1 RESTAURANT MANAGEMENT SYSTEM (RMS) App :

The Restaurant Management System (RMS) is a helpful tool for restaurants. It helps manage different aspects like menus, reservations, orders, and staff. Using simple interfaces, restaurant owners can easily add, update, or delete menu items, reservations, and staff information. The RMS aims to make restaurant operations smoother and improve customer experiences.

1.2 IMPLEMENTATION

This digital solution serves as a comprehensive management system tailored specifically for restaurants. Built using Python and Streamlit for the user interface, it seamlessly integrates with MongoDB for efficient data storage.

With its user-friendly interface, restaurant owners and staff can effortlessly manage various aspects of restaurant operations, including menus, reservations, orders, staff management, and billing. It provides a centralized platform to add new menu items, schedule reservations, process orders, monitor staff activities, and generate bills for customers.

Designed to streamline day-to-day tasks, enhance operational efficiency, and improve overall customer experience, this system acts as a digital assistant for restaurant management. Its adaptability and scalability make it suitable for a wide range of restaurant types and sizes, offering flexibility to meet unique business needs.

1.3 FUNCTIONALITIES

Menu Management:

Easily add, view, update, and delete menu items, including their names and prices, to keep the menu up-to-date and appealing to customers.

Reservation Handling:

Efficiently manage reservations by allowing customers to book tables, view existing reservations, and modify or cancel their bookings as needed.

Order Processing:

Seamlessly process customer orders, whether for dine-in, takeout, or delivery, by recording order details, tracking their status, and updating the system in real-time.

Staff Management:

Simplify staff management tasks by adding, viewing, updating, and deleting staff members, including their names, positions, and contact information, to ensure smooth operations and effective communication.

Billing and Invoicing:

Generate accurate bills for customers based on their orders, calculate the total amount, and provide detailed invoices for easy payment processing and record-keeping.

2.SURVEY OF TECHNOLOGY

2.1 SOFTWARE DESCRIPTION

StreamLit (Python):

Streamlit is an open-source Python library used to create interactive web applications for data science and machine learning projects. It allows developers to quickly build user interfaces using simple Python scripts, without the need for knowledge of web development languages like HTML, CSS, or JavaScript.

MongoDB:

MongoDB is a widely used NoSQL database management system known for its flexibility, scalability, and performance. It stores data in a flexible, JSON-like format called BSON (Binary JSON), making it suitable for handling unstructured or semi-structured data

2.2 LANGUAGES

- Python,
- HTML, CSS
- JavaScript

3.SYSTEM_REQUIREMENTS AND ANALYSIS

3.1 HARDWARE SPECIFICATION:

PROCESSOR	- Intel® core™ i5-12400F@ 2.50 GHz
RAM	- 8 GB
OPERATING SYSTEM	- Microsoft Windows 10/11
HARD DISK	- 1 GB of free space

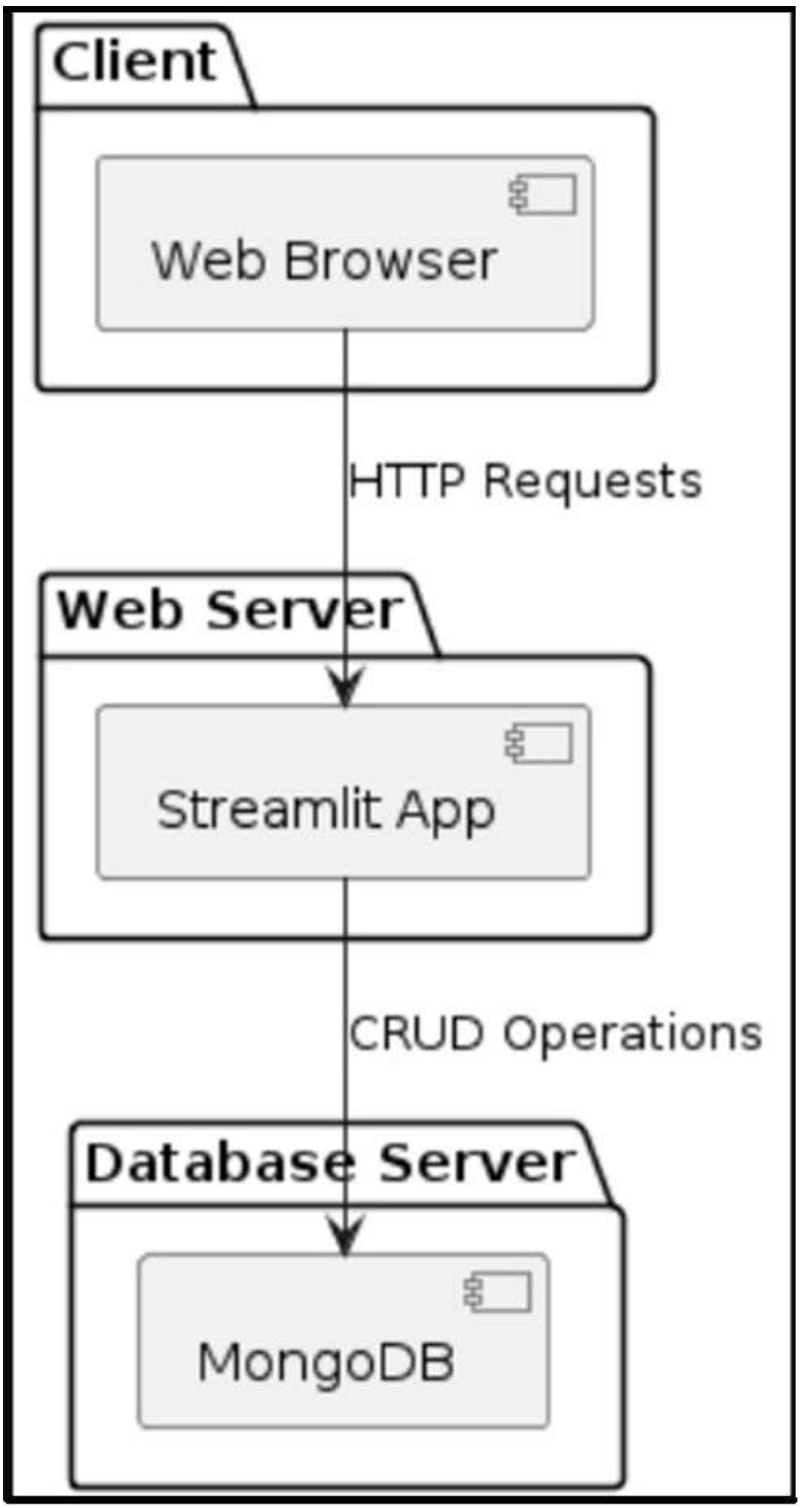
PRE- REQUISITES:

- PYTHON
- STREAMLIT
- PYMONGO
- BSON
- WERKZEUG
- DATETIME
- BASE64
- MONGODB
- MONGO SHELL/ ATLAS

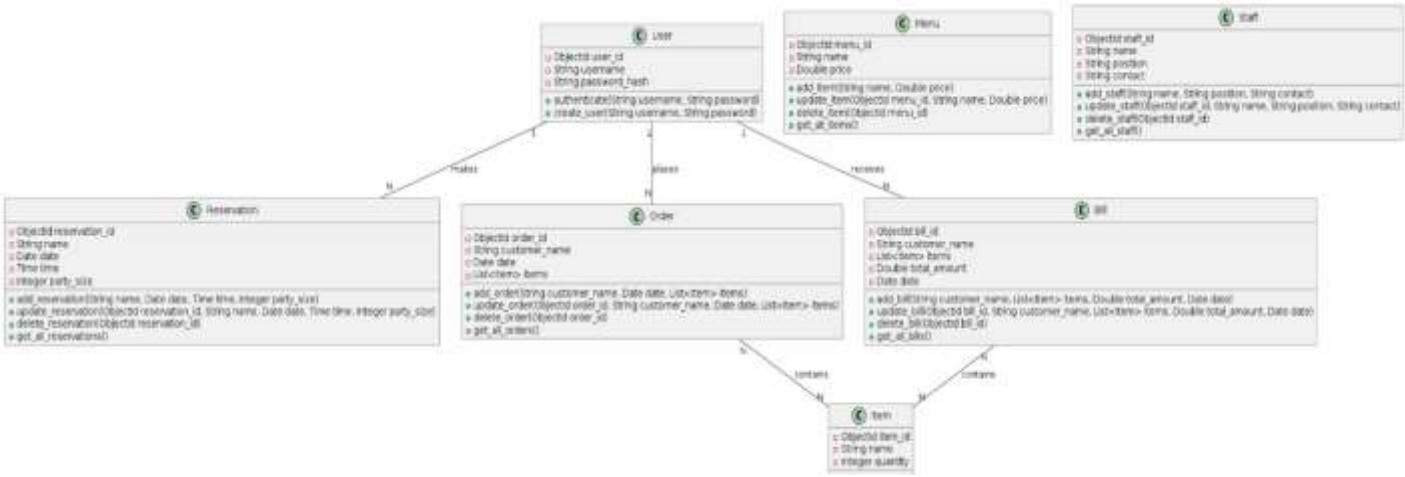
3.2 SOFTWARE SPECIFICATION

OPERATING SYSTEM	: Microsoft Windows 10
SOFTWARE REQUIRED	: Python, JavaScript , MongoDB

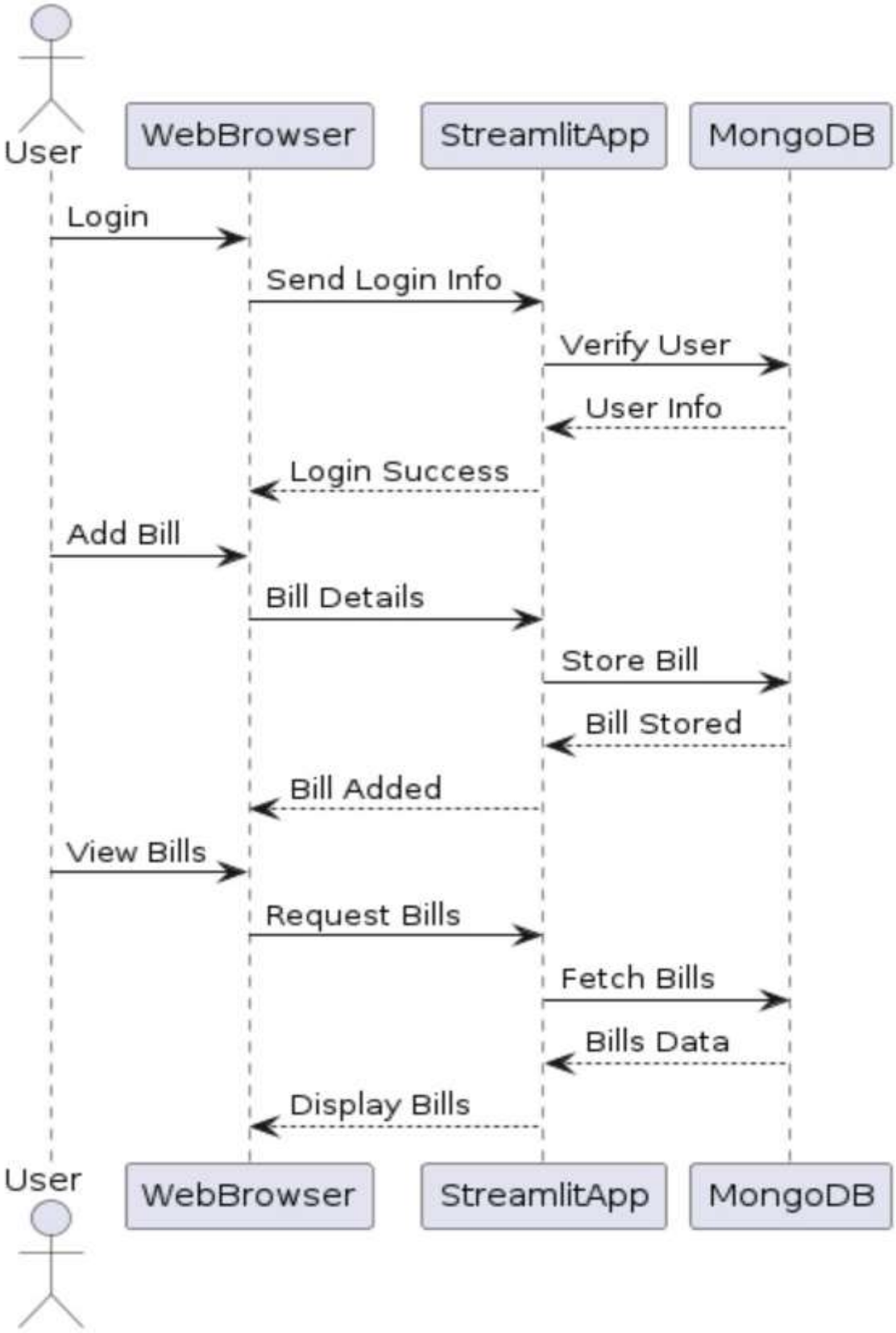
3.3 ARCHITECTURE DIAGRAM



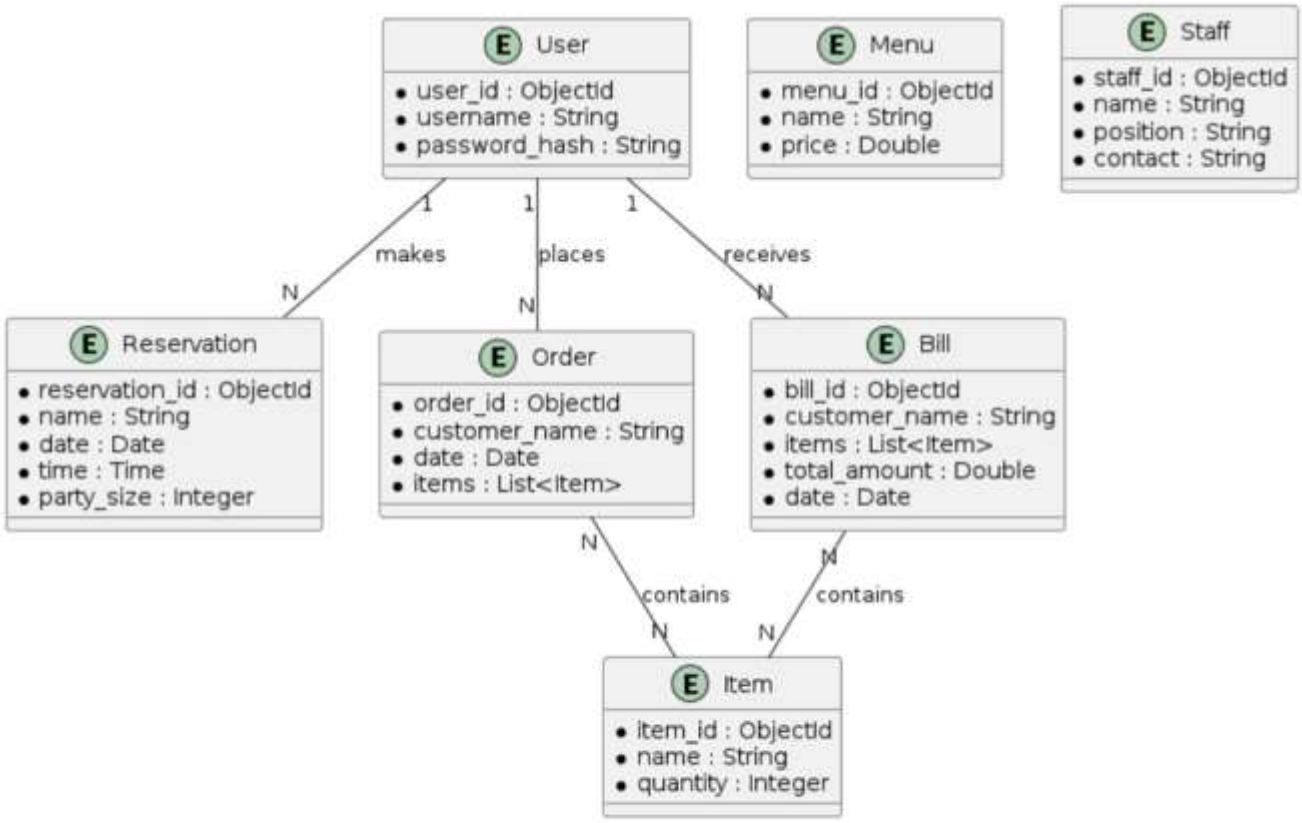
3.4 CLASS DIAGRAM



3.5 SEQUENCE DIAGRAM



3.6 ER DIAGRAM



4. PROGRAM

App.py

```
import base64
import datetime
from datetime import datetime as dt
import streamlit as st
from pymongo import MongoClient
from bson.objectid import ObjectId
import pandas as pd
from werkzeug.security import generate_password_hash, check_password_hash
from db_utils import get_database, add_menu_item, get_all_menu_items,
                     update_menu_item, delete_menu_item, \
                     add_reservation, get_all_reservations, update_reservation, delete_reservation, \
                     add_order, get_all_orders, update_order, delete_order, \
                     add_staff, get_all_staff, update_staff, delete_staff, \
                     add_user, authenticate_user, add_bill, get_all_bills,
                    update_bill, delete_bill
st.set_page_config(layout="wide")
hide_st_style = """ <style>
        #MainMenu {visibility: hidden;}
        footer {visibility: hidden;}
        header {visibility: hidden;}
        </style>  """
st.markdown(hide_st_style, unsafe_allow_html=True)

# MongoDB connection
db = get_database()

# Background setup
def get_base64(bin_file):
   with open(bin_file, 'rb') as f:
      data = f.read()
   return base64.b64encode(data).decode()
def set_background(png_file):
   bin_str = get_base64(png_file)
   page_bg_img = ''' <style>
   .stApp {
       background-image: linear-gradient(rgba(0, 0, 0, 0.5),  rgba(0,0,0,0.5)),
                                                   url('%s');
   background-size: cover;
   }
   </style> '''
   % bin_str
   st.markdown(page_bg_img, unsafe_allow_html=True)
   set_background('background.jpg')
```

```
# Authentication
    def login():
        st.subheader("Login")
        username = st.text_input("Username")
        password = st.text_input("Password", type="password")
        if st.button("Login"):
            if authenticate_user(db, username, password):
                st.session_state['logged_in'] = True
                st.session_state['username'] = username
                st.success("Logged in successfully!")
                st.experimental_rerun()
            else:
                st.error("Invalid username or password")
    def signup():
        st.subheader("Sign Up")
        username = st.text_input("Choose a Username")
        password = st.text_input("Choose a Password", type="password")
        confirm_password = st.text_input("Confirm Password", type="password")
        if st.button("Sign Up"):
            if password != confirm_password:
                st.error("Passwords do not match")
            else:
                add_user(db, username, password)
                st.success("Account created successfully! Please log in.")
                st.experimental_rerun()

# Main app
    if 'logged_in' not in st.session_state:
        st.session_state['logged_in'] = False
    if not st.session_state['logged_in']:
        auth_mode = st.sidebar.selectbox("Select Mode", ["Login", "Sign Up"])
        if auth_mode == "Login":
            login()
        else:
            signup()
    else:
        st.title("SHETTY's CAFE")

# Sidebar menu
     menu = ["Home", "About Us", "Manage Menu", "Manage Reservations", "Manage Orders",
    "Manage Staff", "Billing", "Logout"]
        selected_option = st.sidebar.radio("Menu", menu)

        if selected_option = = "Logout":
            st.session_state['logged_in'] = False
            st.experimental_rerun()
```

```
    elif selected_option = = "Home":
        st.subheader("Home")
        st.write(""Welcome to Shetty's Cafe!! "")

# Dashboard
        st.subheader("Dashboard")
        menu_count = db.menu.count_documents({})
        reservations_count = db.reservations.count_documents({})
        orders_count = db.orders.count_documents({})
        staff_count = db.staff.count_documents({})
        col1, col2, col3, col4 = st.columns(4)
        with col1:
            st.metric("Total Menu Items", menu_count)
        with col2:
            st.metric("Total Reservations", reservations_count)
        with col3:
            st.metric("Total Orders", orders_count)
        with col4:
            st.metric("Total Staff", staff_count)
    elif selected_option == "About Us":
        st.subheader("About Us")
        st.write("""
            *Our Story*
            Established in 2020, our restaurant has been dedicated to providing exceptional culinary experiences. Our chefs use the finest ingredients to create mouth-watering dishes that satisfy all taste buds.
            *Our Vision*
            To be a leading restaurant known for quality, service, and innovation.
            *Our Mission*
            To provide a memorable dining experience through exceptional service, delicious food, and a welcoming atmosphere.
            Thank you for choosing us. We look forward to serving you!
        """)
    elif selected_option == "Manage Menu":
        st.subheader("Manage Menu")
        tab1, tab2, tab3 = st.tabs(["Add Menu Item", "View Menu", "Update/Delete Menu Item"])
        with tab1:
            st.subheader("Add New Menu Item")
            name = st.text_input("Menu Item Name")
            price = st.number_input("Price", min_value=0.0, format="%.2f")
            if st.button("Add Item"):
                if name and price:
                    add_menu_item(db, name, price)
                    st.success(f"Added {name} to menu!")
                else:
                    st.error("Please fill in all fields.")
```

```
    with tab2:
        st.subheader("View Menu")
        menu_items = get_all_menu_items(db)
        for item in menu_items:
            st.write(f"*{item['name']}*")
            st.write(f"Price: ${item['price']:.2f}")
            st.write("---")
    with tab3:
        st.subheader("Update/Delete Menu Item")
        menu_items = get_all_menu_items(db)
        menu_item_list = [(item['name'], item['_id']) for item in menu_items]
        selected_item = st.selectbox("Select Menu Item", menu_item_list, format_func=lambda x: x[0])
        if selected_item:
            item = db.menu.find_one({"_id": ObjectId(selected_item[1])})
            new_name = st.text_input("Menu Item Name", value=item['name'])
            new_price = st.number_input("Price", min_value=0.0, format="%.2f", value=item['price'])
            if st.button("Update Item"):
                update_menu_item(db, selected_item[1], new_name, new_price)
                st.success("Menu item updated!")
            if st.button("Delete Item"):
                delete_menu_item(db, selected_item[1])
                st.success("Menu item deleted!")
elif selected_option == "Manage Reservations":
    st.subheader("Manage Reservations")
    tab1,tab2,tab3=st.tabs(["Make a Reservation","View Reservations","Update/Delete Reservation"])
    with tab1:
        st.subheader("Make a Reservation")
        name = st.text_input("Your Name")
        date = st.date_input("Reservation Date")
        time = st.time_input("Reservation Time")
        party_size = st.number_input("Party Size", min_value=1)
        if st.button("Book Reservation"):
            if name and date and time and party_size:
                add_reservation(db, name, date, time, party_size)
                st.success("Reservation made successfully!")
            else:
                st.error("Please fill in all fields.")
    with tab2:
        st.subheader("View Reservations")
        reservations = get_all_reservations(db)
        for res in reservations:
            st.write(f"*Name:* {res['name']}")
            st.write(f"*Date:* {res['date']}")
            st.write(f"*Time:* {res['time']}")
            st.write(f"*Party Size:* {res['party_size']}")
            st.write("---")
```

```
    with tab3:
        st.subheader("Update/Delete Reservation")
        reservations = get_all_reservations(db)
        reservation_list = [(res['name'], res['_id']) for res in reservations]
        selected_res = st.selectbox("Select Reservation", reservation_list,format_func=lambda x: x[0])
        if selected_res:
            res = db.reservations.find_one({"_id": ObjectId(selected_res[1])})
            new_name = st.text_input("Your Name", value=res['name'])
            new_date = st.date_input("Reservation Date",value=dt.strptime(res['date'], "%Y-%m-%d"))
            new_time = st.time_input("Reservation Time", value=dt.strptime(res['time'], "%H:%M"))
            new_party_size = st.number_input("Party Size",in_value=1, value=res['party_size'])

            if st.button("Update Reservation"):
                update_reservation(db, selected_res[1], new_name, new_date, new_time, new_party_size)
                st.success("Reservation updated!")
            if st.button("Delete Reservation"):
                delete_reservation(db, selected_res[1])
                st.success("Reservation deleted!")

elif selected_option == "Manage Orders":
    st.subheader("Manage Orders")
    tab1, tab2, tab3 = st.tabs(["Add Order", "View Orders", "Update/Delete Order"])

    with tab1:
        st.subheader("Add New Order")
        customer_name = st.text_input("Customer Name")
        date = st.date_input("Order Date")
        items = st.text_input("Items (format: Item1: Quantity1, Item2: Quantity2, ...)")

        if st.button("Add Order"):
            if customer_name and date and items:
                add_order(db, customer_name, date, items)
                st.success("Order added successfully!")
            else:
                st.error("Please fill in all fields.")

    with tab2:
        st.subheader("View Orders")
        orders = get_all_orders(db)
        for order in orders:
            st.write(f"*Customer Name:* {order['customer_name']}")
            st.write(f"*Date:* {order['date']}")
            st.write("*Items:*")
            for item in order['items']:
                st.write(f"  - {item['name']}: {item['quantity']}")
            st.write("---")
```

```
    with tab3:
        st.subheader("Update/Delete Order")
        orders = get_all_orders(db)
        order_list = [(order['customer_name'], order['_id']) for order in orders]
        selected_order = st.selectbox("Select Order", order_list, format_func=lambda x: x[0])

        if selected_order:
            order = db.orders.find_one({"_id": ObjectId(selected_order[1])})
            new_customer_name = st.text_input("  Name",value=order['customer_name'])
            new_date = st.date_input("Order Date", value=dt.strptime(order['date'], "%Y-%m-%d"))
            new_items = st.text_input("Items (format: Item1: Quantity1, Item2: Quantity2, ...)",
            value=",".join([f"item['name']}:{item['quantity']}" for item in order['items']]))

            if st.button("Update Order"):
                update_order(db, selected_order[1], new_customer_name, new_date, new_items)
                st.success("Order updated!")

            if st.button("Delete Order"):
                delete_order(db, selected_order[1])
                st.success("Order deleted!")

elif selected_option == "Manage Staff":
    st.subheader("Manage Staff")
    tab1, tab2, tab3 = st.tabs(["Add Staff Member", "View Staff", "Update/Delete Staff Member"])

    with tab1:
        st.subheader("Add New Staff Member")
        name = st.text_input("Name")
        position = st.text_input("Position")
        contact = st.text_input("Contact Information")
        if st.button("Add Staff Member"):
            if name and position and contact:
                add_staff(db, name, position, contact)
                st.success("Staff member added successfully!")
            else:
                st.error("Please fill in all fields.")

    with tab2:
        st.subheader("View Staff")
        staff_members = get_all_staff(db)
        for member in staff_members:
            st.write(f"*Name:* {member['name']}")
            st.write(f"*Position:* {member['position']}")
            st.write(f"*Contact Information:* {member['contact']}")
            st.write("---")
```

```
        with tab3:
            st.subheader("Update/Delete Staff Member")
            staff_members = get_all_staff(db)
            staff_list = [(member['name'], member['_id']) for member in staff_members]
            selected_staff = st.selectbox("Select Staff Member", staff_list,format_func=lambda x: x[0])
            if selected_staff:
                member = db.staff.find_one({"_id": ObjectId(selected_staff[1])})
                new_name = st.text_input("Name", value=member['name'])
                new_position = st.text_input("Position", value=member['position'])
                new_contact = st.text_input("Contact Information", value=member['contact'])

                if st.button("Update Staff Member"):
                    update_staff(db, selected_staff[1], new_name, new_position,new_contact)
                    st.success("Staff member updated!")
                if st.button("Delete Staff Member"):
                    delete_staff(db, selected_staff[1])
                    st.success("Staff member deleted!")

    elif selected_option == "Billing":
        st.subheader("Billing")
        tab1, tab2, tab3 = st.tabs(["Add Bill", "View Bills", "Update/Delete Bill"])
        def calculate_total(items, menu_items):
            total = 0.0
            item_dict = {item['name']: item['price'] for item in menu_items}
            for item in items:
                if item['name'] in item_dict:
                    total += item['quantity'] * item_dict[item['name']]
            return total

        with tab1:
            st.subheader("Add New Bill")
            customer_name = st.text_input("Customer Name")
            items = st.text_input("Items (format: Item1: Quantity1, Item2: Quantity2, ...)")
            menu_items = get_all_menu_items(db)
            if items:
                items_list = [{"name": item.split(":")[0].strip(),"quantity": int(item.split(":")[1].strip())}
                                                                        for item in items.split(",")]
                total_amount = calculate_total(items_list, menu_items)
                st.write(f"Total Amount: ${total_amount:.2f}")
            if st.button("Add Bill"):
                if customer_name and items:
                    add_bill(db, customer_name, items_list, total_amount)
                    st.success("Bill added successfully!")
                else:
                    st.error("Please fill in all fields.")
```

```
with tab2:
    st.subheader("View Bills")
    bills = get_all_bills(db)
    for bill in bills:
        st.write(f"*Customer Name:* {bill['customer_name']}")
        st.write("*Items:*")
        for item in bill['items']:
            st.write(f"  - {item['name']}: {item['quantity']}")
        st.write(f"*Total Amount:* ${bill['total_amount']:.2f}")
        st.write(f"*Date:* {bill['date']}")
        st.write("---")

with tab3:
    st.subheader("Update/Delete Bill")
    bills = get_all_bills(db)
    bill_list = [(bill['customer_name'], bill['_id']) for bill in bills]
    selected_bill = st.selectbox("Select Bill", bill_list, format_func=lambda x: x[0])

    if selected_bill:
        bill = db.bills.find_one({"_id": ObjectId(selected_bill[1])})
        new_customer_name = st.text_input("Customer Name", value=bill['customer_name'])
        new_items = st.text_input("Items (format: Item1: Quantity1, Item2: Quantity2, ...)",
        value=",".join([f"{item['name']}: {item['quantity']}"for item in bill['items']]))
        menu_items = get_all_menu_items(db)

        if new_items:
            new_items_list = [{"name": item.split(":")[0].strip(), "quantity":
                        int(item.split(":")[1].strip())} for item in new_items.split(",")]
            new_total_amount = calculate_total(new_items_list, menu_items)
            st.write(f"Total Amount: ${new_total_amount:.2f}")

        if st.button("Update Bill"):
            update_bill(db, selected_bill[1], new_customer_name, new_items_list,
                                                                    new_total_amount)
            st.success("Bill updated!")

        if st.button("Delete Bill"):
            delete_bill(db, selected_bill[1])
            st.success("Bill deleted!")
```

Database :

```
from pymongo import MongoClient
from bson.objectid import ObjectId
from werkzeug.security import generate_password_hash, check_password_hash
from datetime import datetime

def get_database():
   CONNECTION_STRING = "mongodb://localhost:27017/"
                                                client = MongoClient(CONNECTION_STRING)
   return client['restaurant_db']
```

Menu operations

```
def add_menu_item(db, name, price):
   menu_item = {"name": name, "price": price}
   db.menu.insert_one(menu_item)

def get_all_menu_items(db):
   return list(db.menu.find())

def update_menu_item(db, item_id, new_name, new_price):
   db.menu.update_one(
      {"_id": ObjectId(item_id)},
      {"$set": {"name": new_name, "price": new_price}})

def delete_menu_item(db, item_id):
   db.menu.delete_one({"_id": ObjectId(item_id)})
```

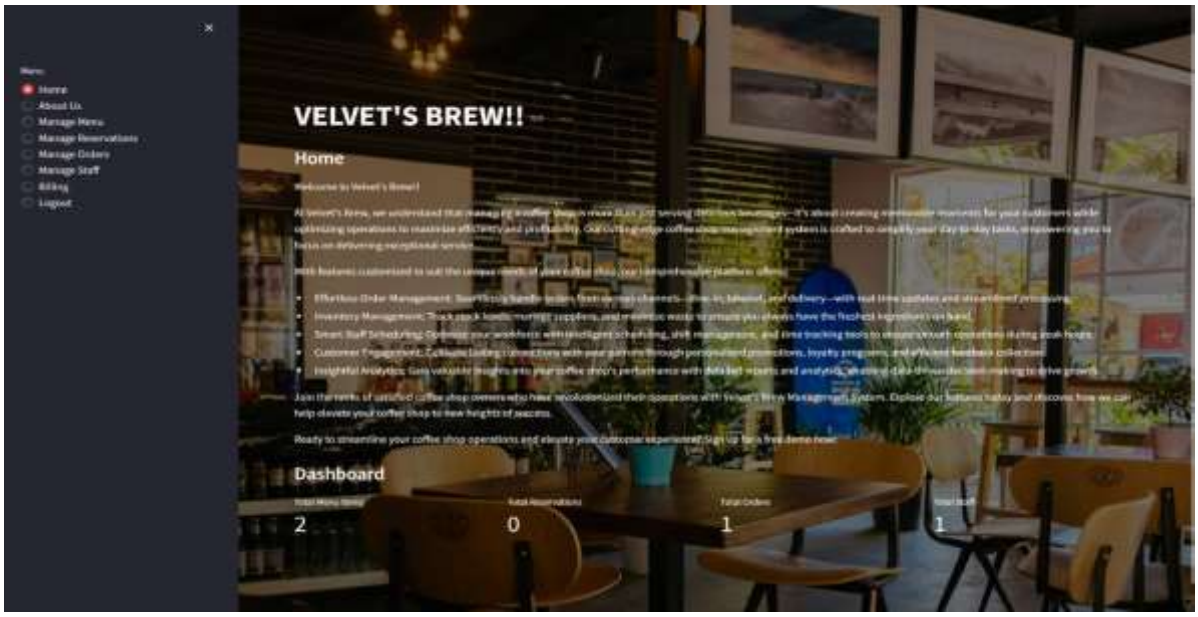
Reservation operations

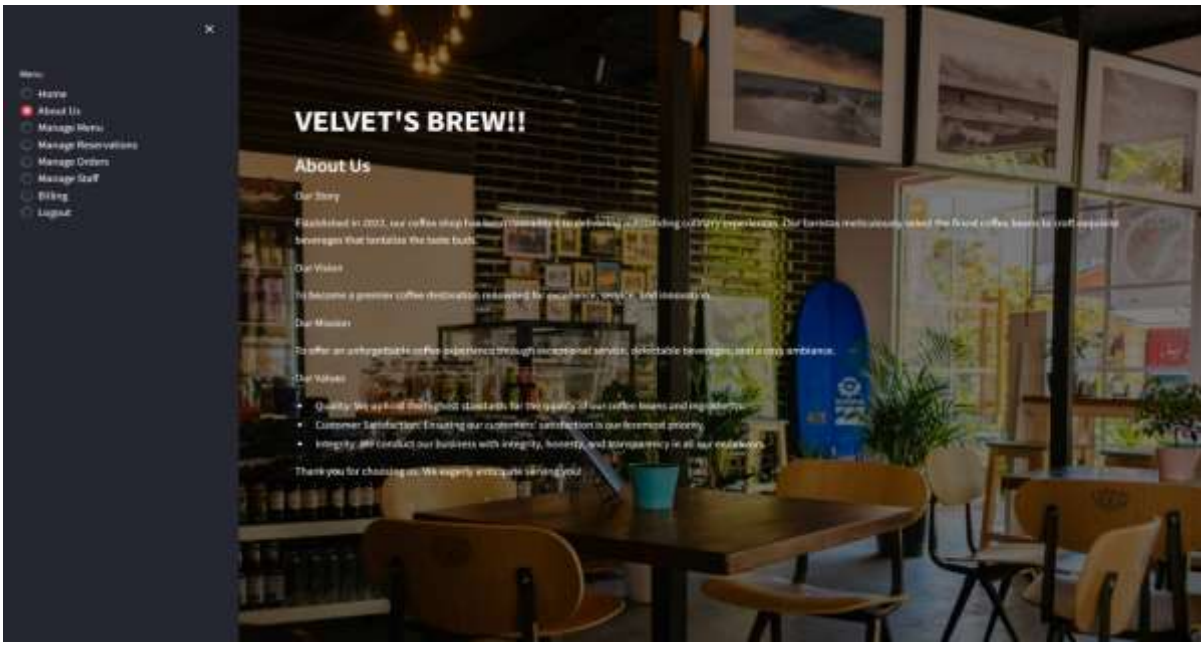
```
def add_reservation(db, name, date, time, party_size):
   reservation = {
      "name": name,
      "date": date.strftime("%Y-%m-%d"),
      "time": time.strftime("%H:%M"),
      "party_size": party_size
   }
   db.reservations.insert_one(reservation)

def get_all_reservations(db):
   return list(db.reservations.find())
def update_reservation(db, reservation_id, new_name, new_date, new_time, new_party_size):
   db.reservations.update_one(
      {"_id": ObjectId(reservation_id)},
      {"$set": {"name": new_name, "date": new_date.strftime("%Y-%m-%d"),
        "time": new_time.strftime("%H:%M"), "party_size": new_party_size}})
def delete_reservation(db, reservation_id):
   db.reservations.delete_one({"_id": ObjectId(reservation_id)})
```

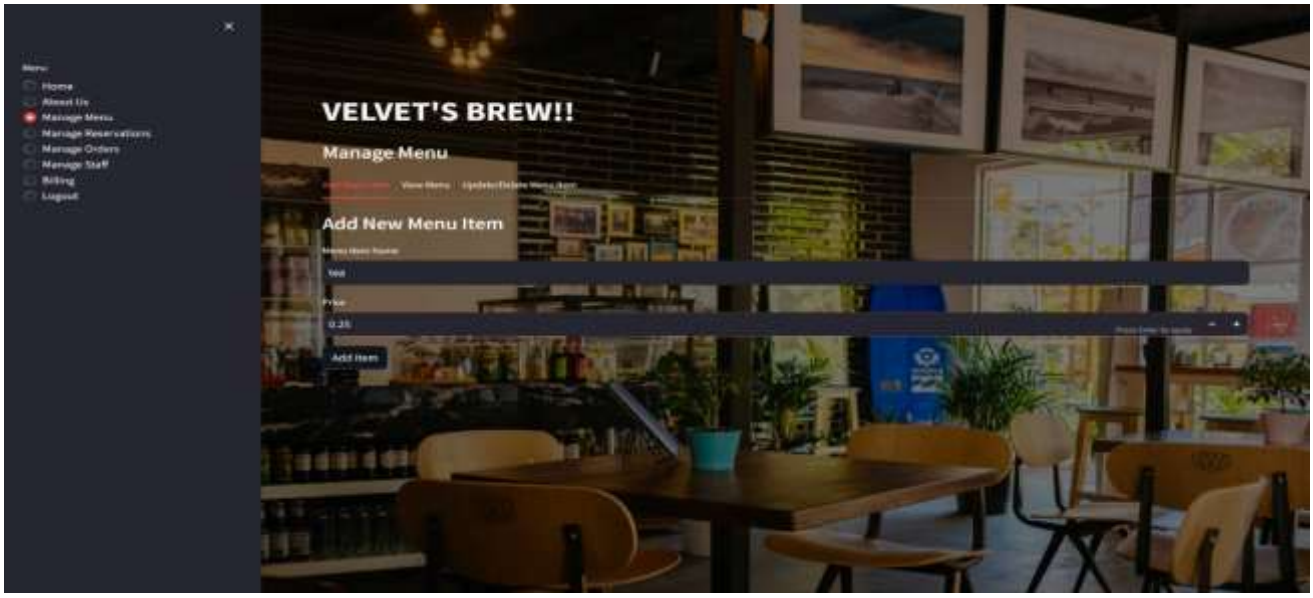
```
# Order operations
    def add_order(db, customer_name, date, items):
        items_list = [{"name": item.split(":")[0].strip(), "quantity": int(item.split(":")[1].strip())}
                                                                        for item in items.split(",")]
        order = {
            "customer_name": customer_name,
            "date": date.strftime("%Y-%m-%d"),
            "items": items_list
        }
        db.orders.insert_one(order)

    def get_all_orders(db):
        return list(db.orders.find())

    def update_order(db, order_id, new_customer_name, new_date, new_items):
        new_items_list = [{"name": item.split(":")[0].strip(), "quantity":int(item.split(":")[1].strip())}
                                                                    for item in new_items.split(",")]
        db.orders.update_one(
            {"_id": ObjectId(order_id)},
            {"$set": {"customer_name": new_customer_name,
              "date": new date.strftime("%Y-%m-%d"),
              "items": new_items_list}} )

    def delete_order(db, order_id):
        db.orders.delete_one({"_id": ObjectId(order_id)})

# Staff operations
    def add_staff(db, name, position, contact):
        staff_member = {"name": name, "position": position, "contact": contact}
        db.staff.insert_one(staff_member)

    def get_all_staff(db):
        return list(db.staff.find())

    def update_staff(db, staff_id, new_name, new_position, new_contact):
        db.staff.update_one(
            {"_id": ObjectId(staff_id)},
            {"$set": {"name": new_name,
              "position": new_position,
              "contact": new_contact}}
            )

    def delete_staff(db, staff_id):
        db.staff.delete_one({"_id": ObjectId(staff_id)})
```

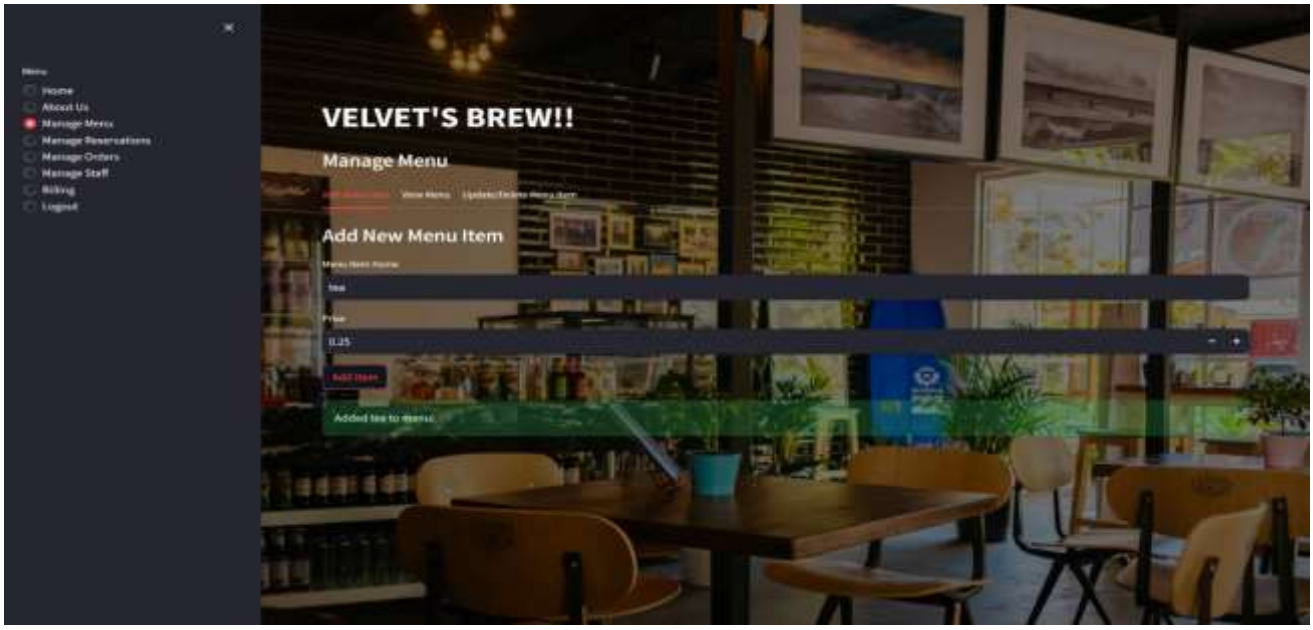
```
# User authentication operations
    def add_user(db, username, password):
        hashed_password = generate_password_hash(password)
        db.users.insert_one({"username": username, "password": hashed_password})

    def authenticate_user(db, username, password):
        user = db.users.find_one({"username": username})
        if user and check_password_hash(user['password'], password):
            return True
        return False

# Billing operations
    def add_bill(db, customer_name, items, total_amount):
        bill = {
            "customer_name": customer_name,
            "items": items,
            "total_amount": total_amount,
            "date": datetime.now().strftime("%Y-%m-%d %H:%M:%S")
        }
        db.bills.insert_one(bill)

    def get_all_bills(db):
        return list(db.bills.find())

    def update_bill(db, bill_id, new_customer_name, new_items, new_total_amount):
        db.bills.update_one(
            {"_id": ObjectId(bill_id)},
            {"$set": {
                "customer_name": new_customer_name,
                "items": new_items,
                "total_amount": new_total_amount,
                "date": datetime.now().strftime("%Y-%m-%d %H:%M:%S")
            }}
        )

    def delete_bill(db, bill_id):
        db.bills.delete_one(
                {"_id": ObjectId(bill_id)})
```

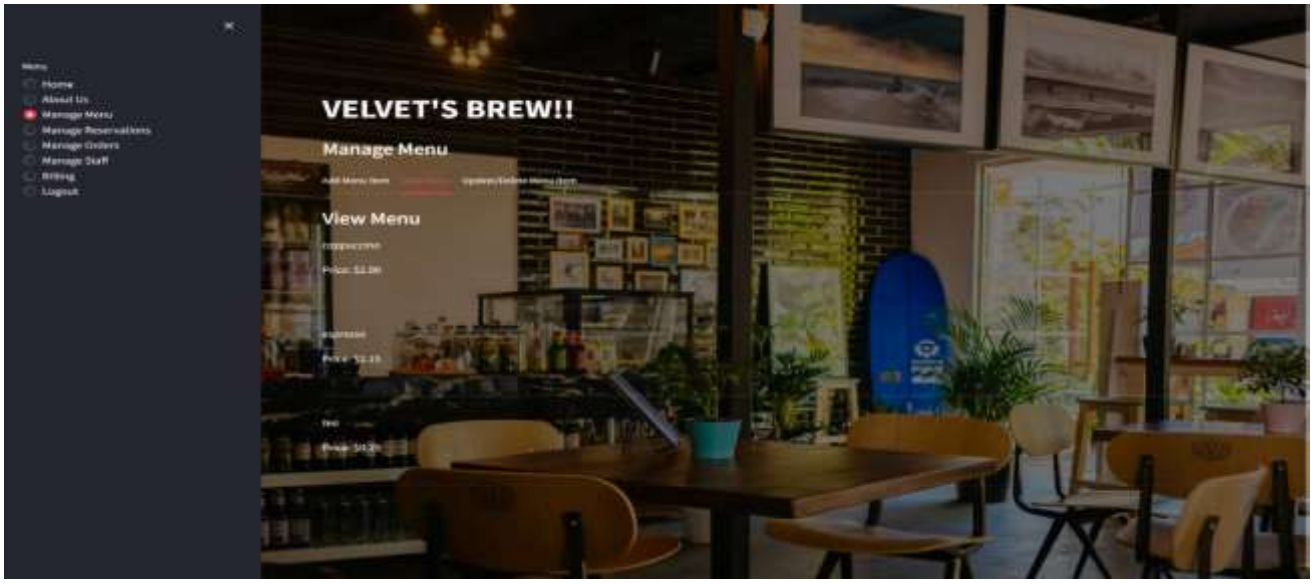
5. RESULTS AND SNAPSHOTS

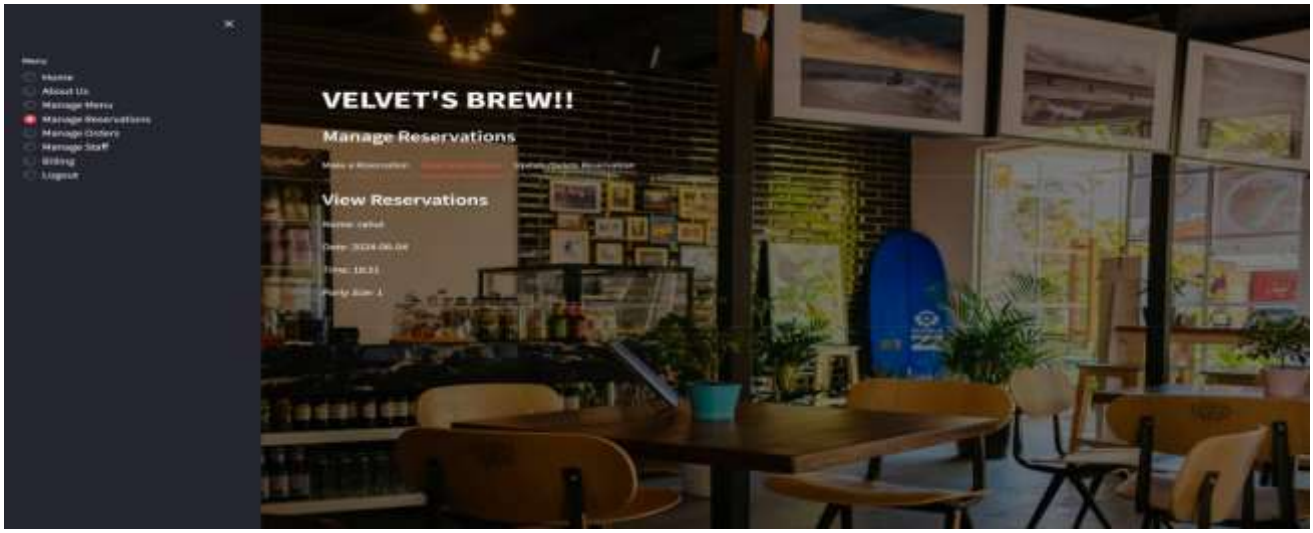


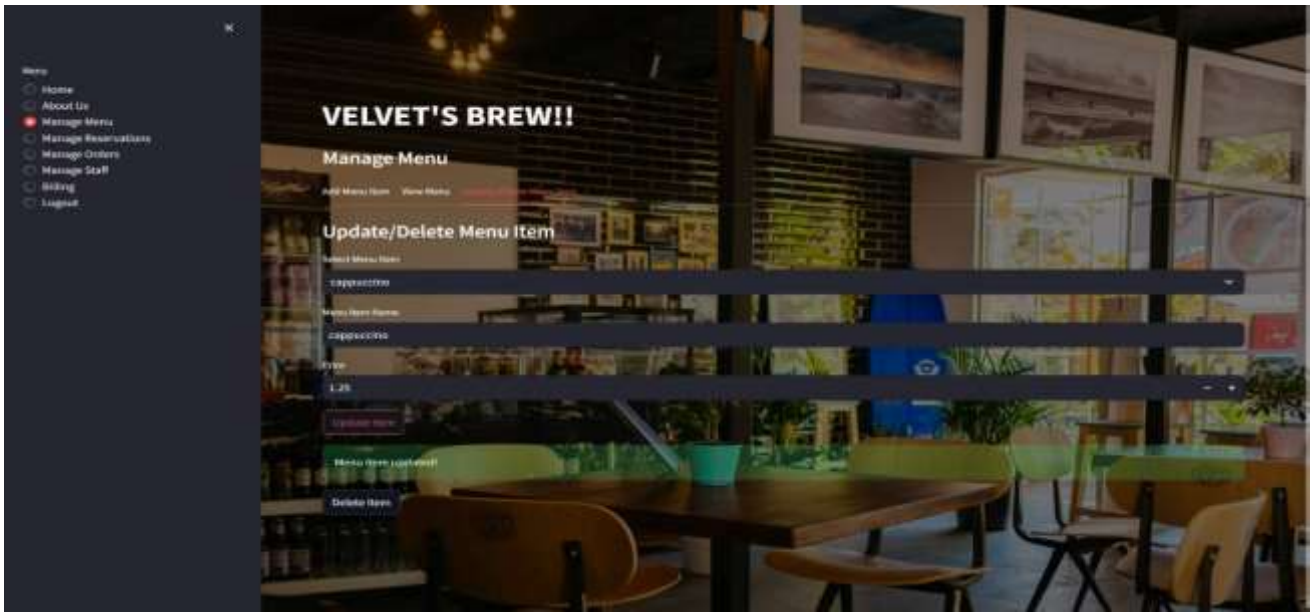


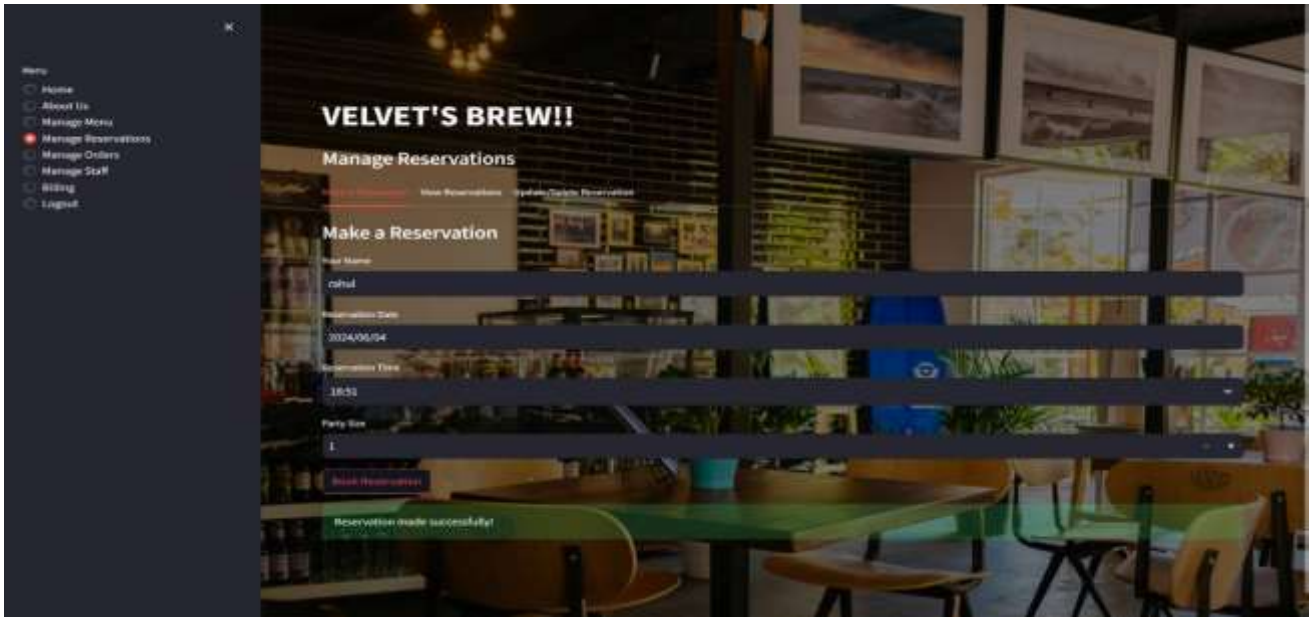

Menu
Home
About Us
Manage Menu
Manage Reservations
Manage Orders
Manage Staff
Billing
Logout
VELVET'S BREW!!
Manage Menu
Add New Menu Item
Price
0.25
Add Item

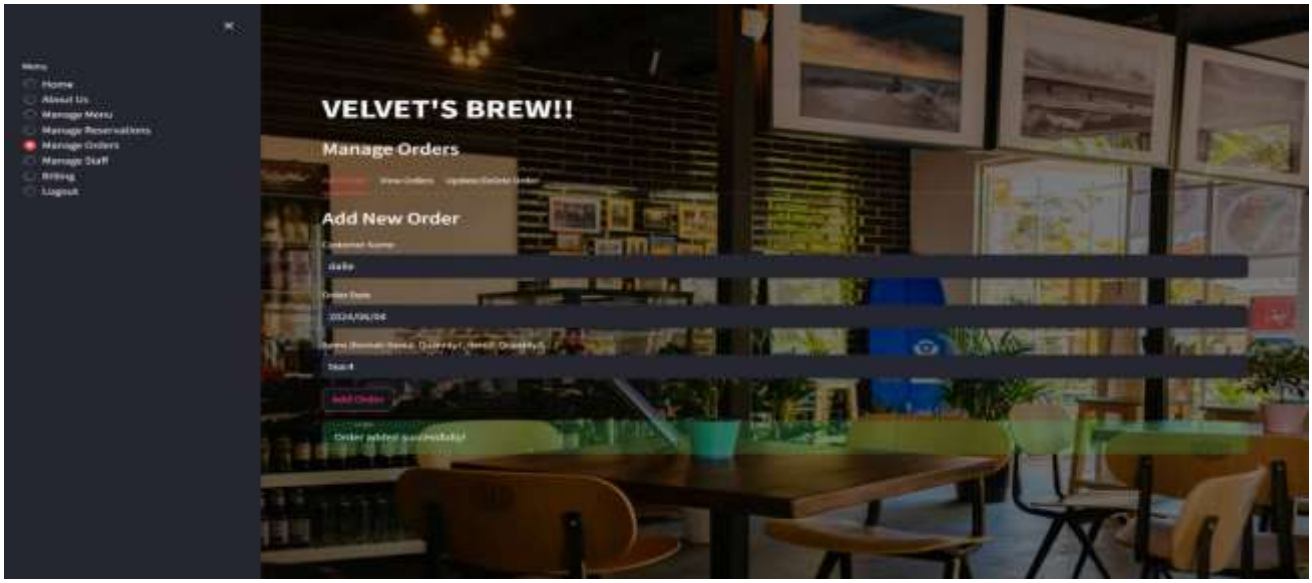

Menu
Home
About Us
Manage Menu
Manage Reservations
Manage Orders
Manage Staff
Billing
Logout
VELVET'S BREW!!
Manage Menu
Add New Menu Item
Price
0.25
Add Item
Added tea to menu!

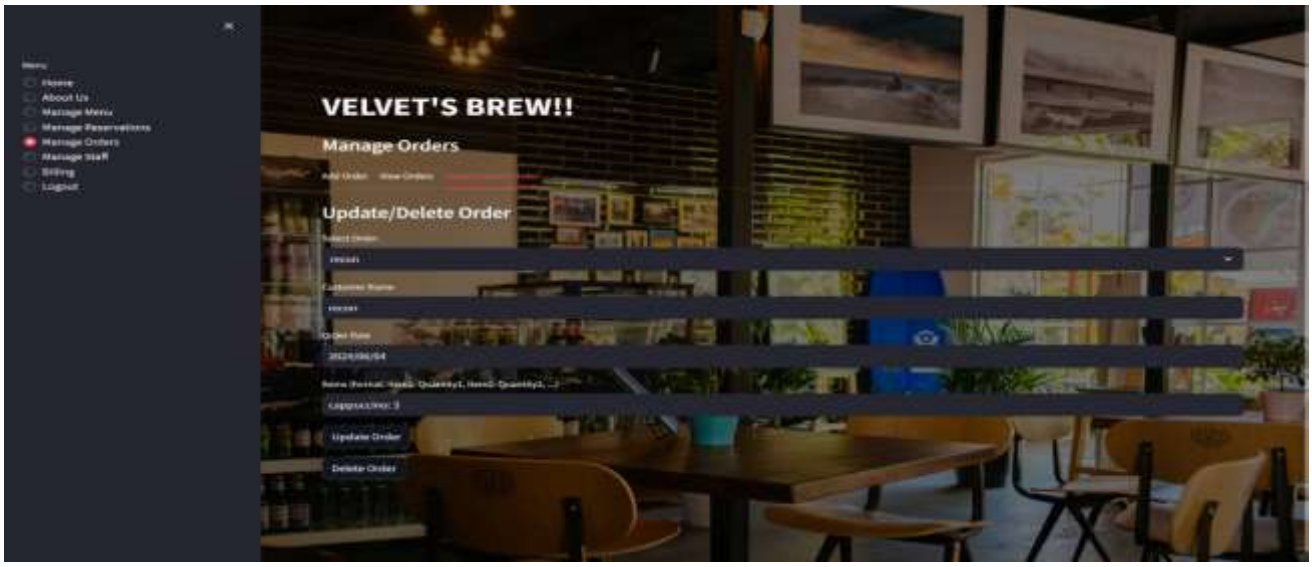

Menu
Home
About Us
Manage Menu
Manage Reservations
Manage Orders
Manage Staff
Billing
Logout
VELVET'S BREW!!
Manage Menu
View Menu

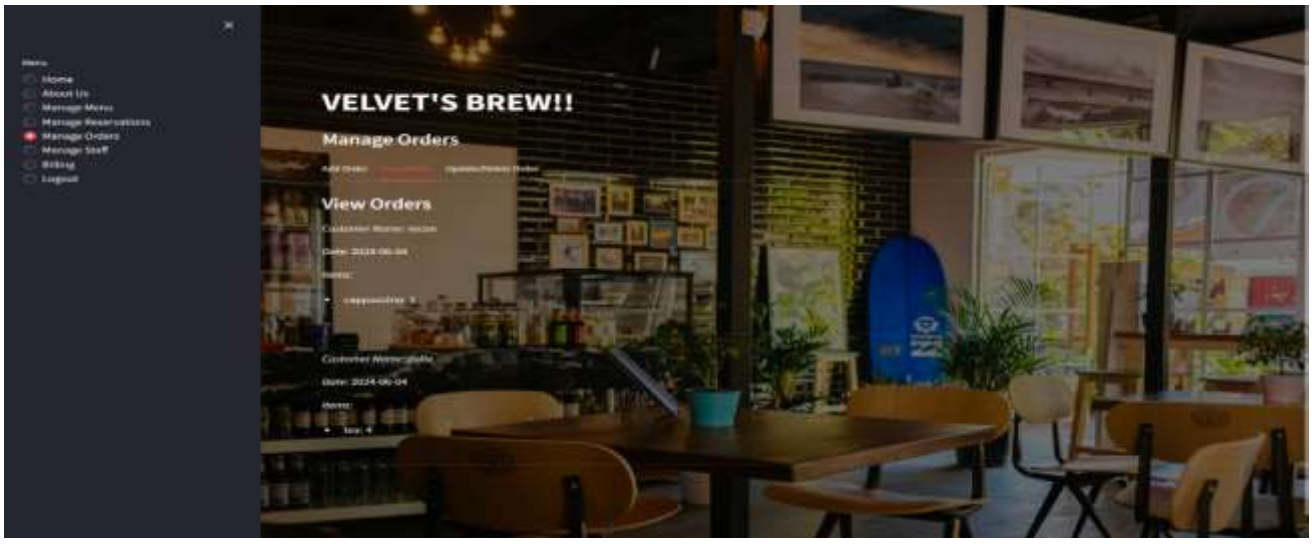

Menu
Home
About Us
Manage Menu
Manage Reservations
Manage Orders
Manage Staff
Billing
Logout
VELVET'S BREW!!
Manage Reservations
View Reservations

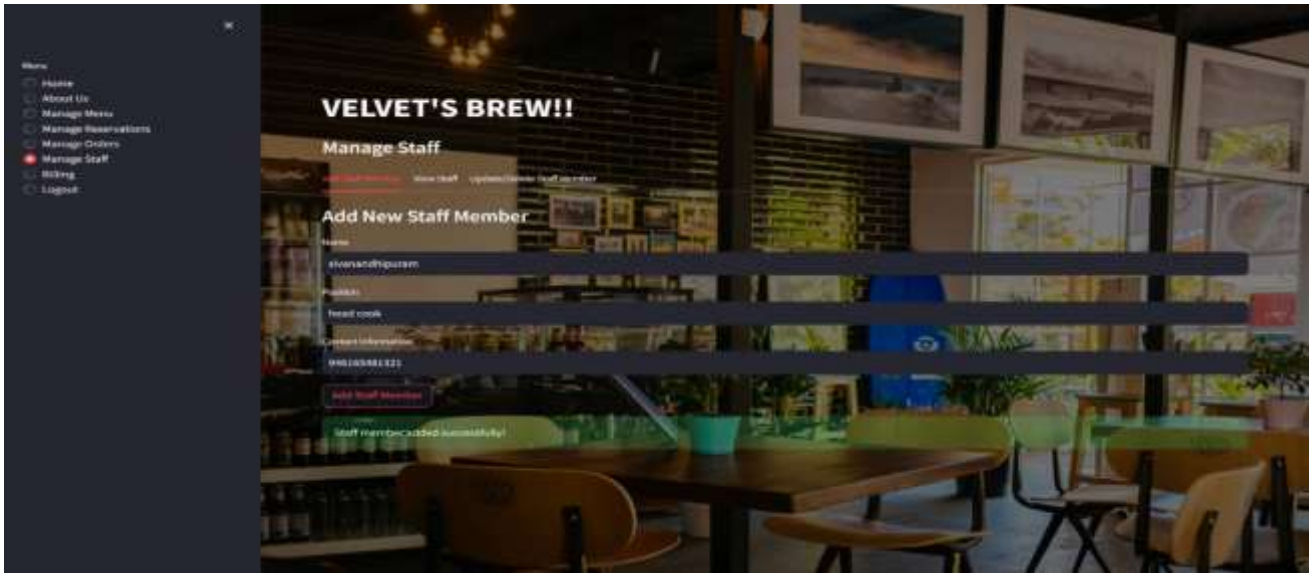

Menu
Home
About Us
Manage Menu
Manage Reservations
Manage Orders
Manage Staff
Billing
Logout
VELVET'S BREW!!
Manage Menu
Update/Delete Menu Item
cappuccino
cappuccino
1.25
Menu item updated!
Delete Item

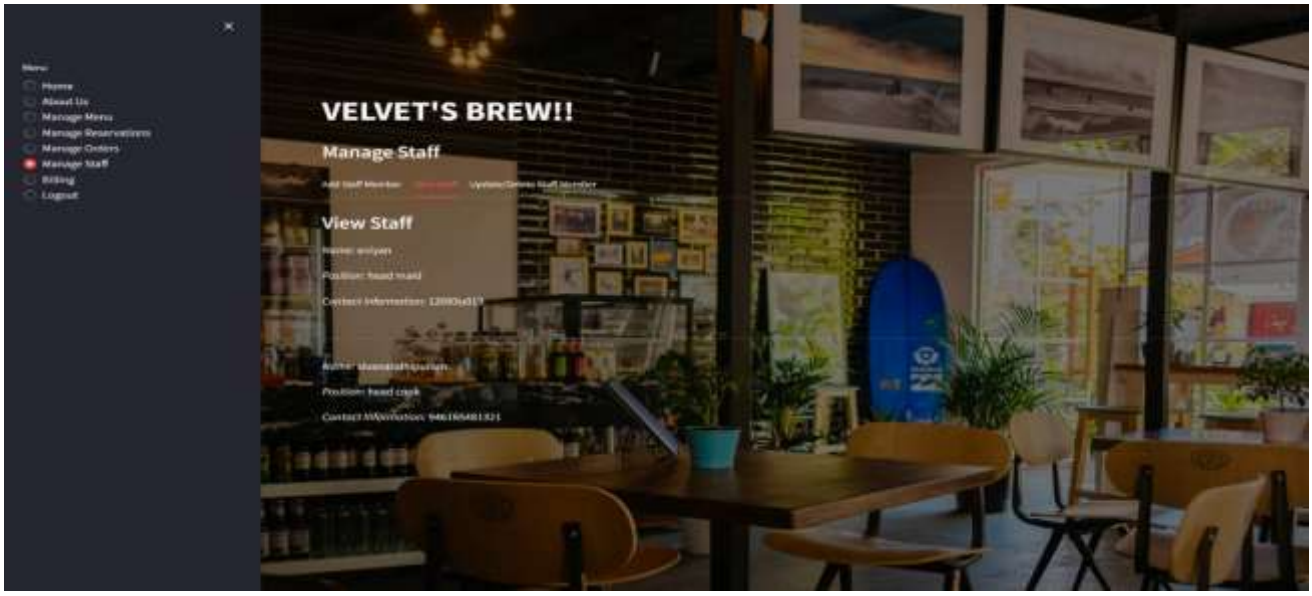

Menu
Home
About Us
Manage Menu
Manage Reservations
Manage Orders
Manage Staff
Billing
Logout
VELVET'S BREW!!
Manage Reservations
Make a Reservation
rahul
2024/05/04
18:51
1
Reservation made successfully!

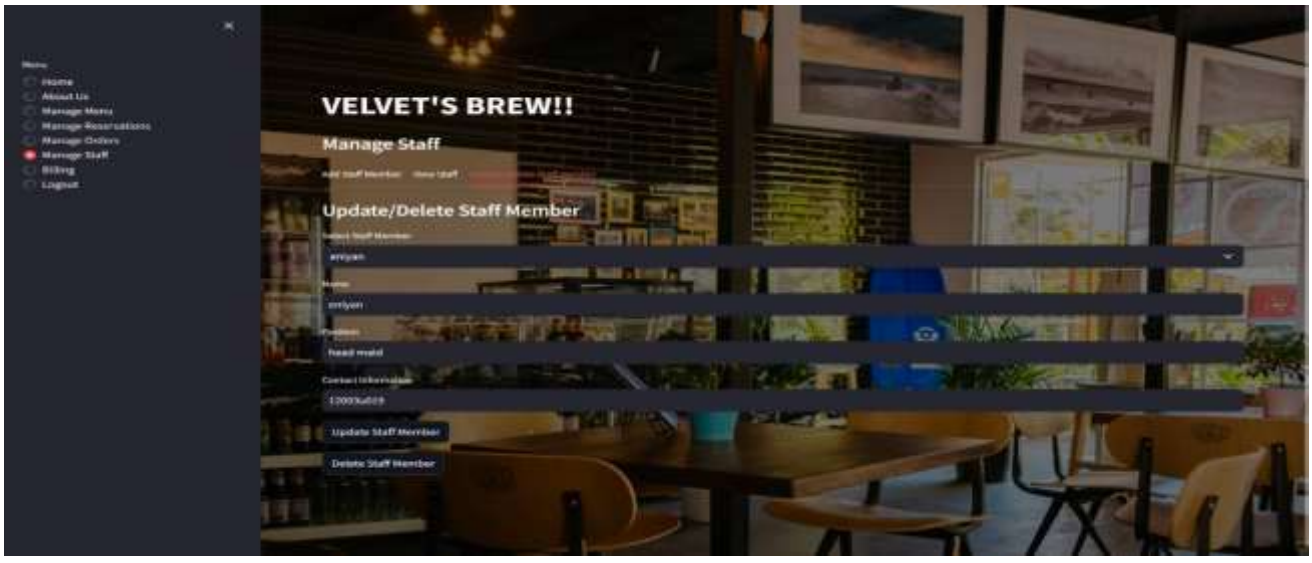

Menu
Home
About Us
Manage Menu
Manage Reservations
Manage Orders
Manage Staff
Billing
Logout
VELVET'S BREW!!
Manage Orders
Add New Order
Order added successfully!

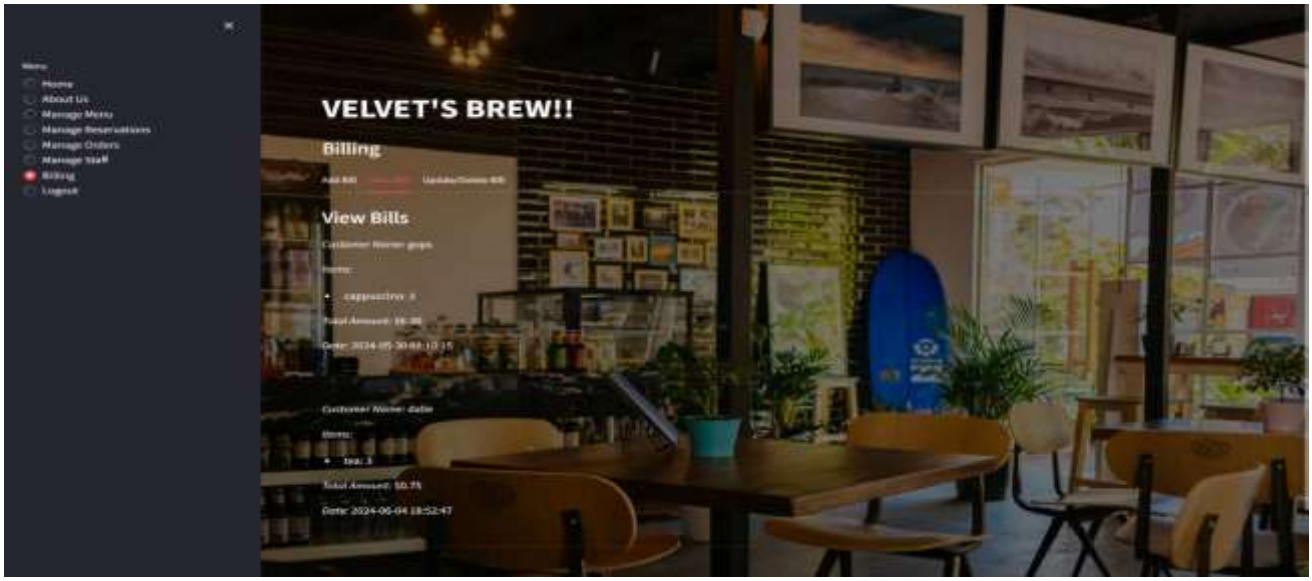

Menu
Home
About Us
Manage Menu
Manage Reservations
Manage Orders
Manage Staff
Billing
Logout
VELVET'S BREW!!
Manage Orders
Update/Delete Order
Update Order
Delete Order

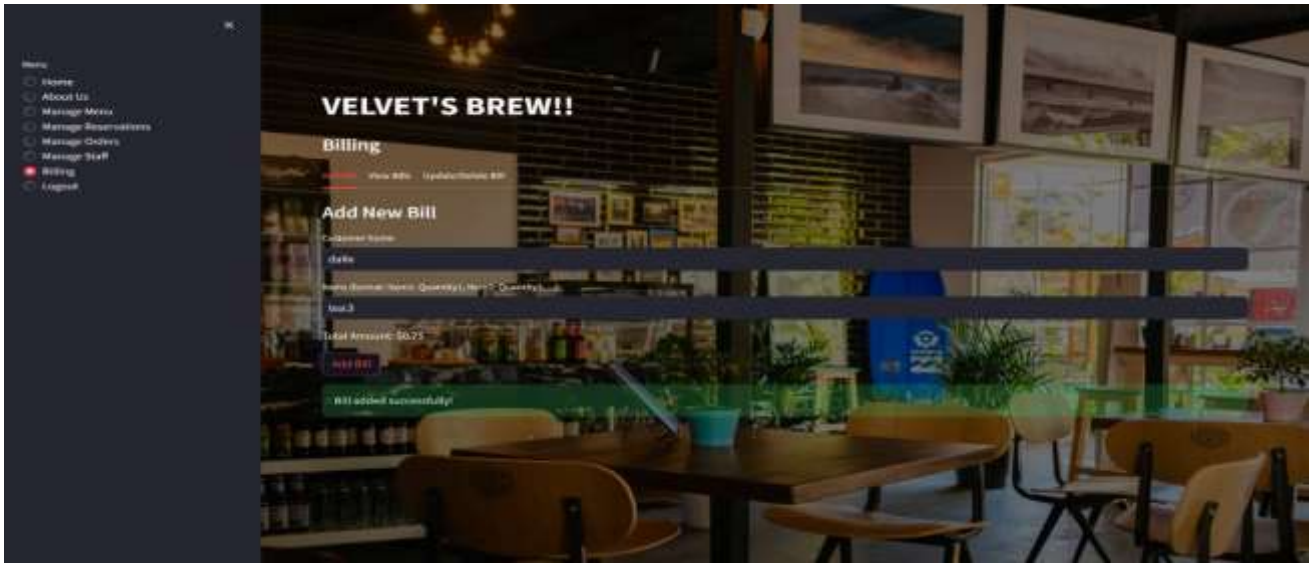

Menu
Home
About Us
Manage Menu
Manage Reservations
Manage Orders
Manage Staff
Billing
Logout
VELVET'S BREW!!
Manage Orders
View Orders

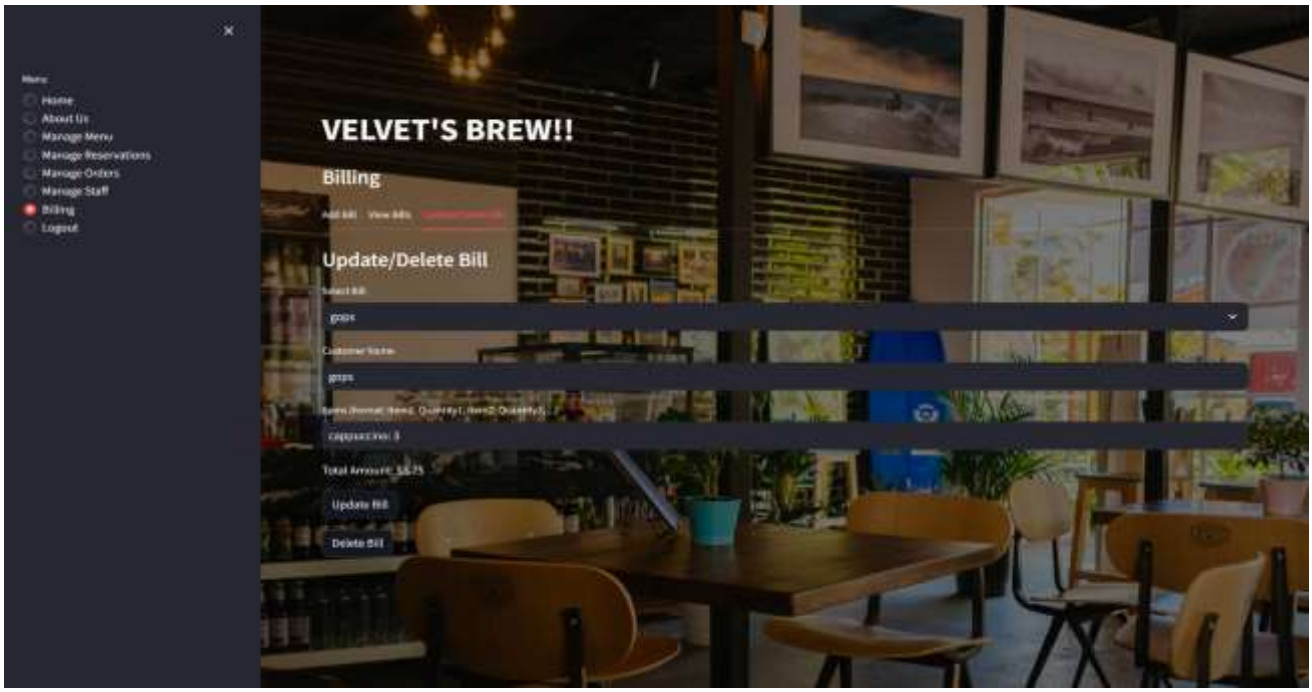

VELVET'S BREW!!
Manage Staff
Add New Staff Member
Home
About Us
Manage Menu
Manage Reservations
Manage Orders
Manage Staff
Billing
Logout


VELVET'S BREW!!
Manage Staff
View Staff
Home
About Us
Manage Menu
Manage Reservations
Manage Orders
Manage Staff
Billing
Logout


VELVET'S BREW!!
Manage Staff
Update/Delete Staff Member
Update Staff Member
Delete Staff Member
Home
About Us
Manage Menu
Manage Reservations
Manage Orders
Manage Staff
Billing
Logout


Menu
Home
About Us
Manage Menu
Manage Reservations
Manage Orders
Manage Staff
Billing
Logout
VELVET'S BREW!!
Billing
View Bills
Update/Delete Bill
View Bills
Customer Name: gops
Items:
cappuccino: 3
Total Amount: $6.75
Date: 2024-05-30 03:12:15
Customer Name: dalle
Items:
tea: 3
Total Amount: $0.75
Date: 2024-06-04 18:52:47


Menu
Home
About Us
Manage Menu
Manage Reservations
Manage Orders
Manage Staff
Billing
Logout
VELVET'S BREW!!
Billing
Add Bill
View Bills
Update/Delete Bill
Add New Bill
Customer Name
clarke
Items (format: Item1, Quantity1, Item2, Quantity2, ...)
tea:3
Total Amount: $0.75
Add Bill
Bill added successfully!


Menu
Home
About Us
Manage Menu
Manage Reservations
Manage Orders
Manage Staff
Billing
Logout
VELVET'S BREW!!
Billing
Add Bill
View Bills
Update/Delete Bill
Update/Delete Bill
Select Bill
gops
Customer Name
gops
Items (format: Item1, Quantity1, Item2, Quantity2, ...)
cappuccino: 3
Total Amount: $6.75
Update Bill
Delete Bill

6. CONCLUSION

This program has been created successfully a system/ application called **RESTAURANT MANAGEMENT SYSTEM**. The results and snapshots of the program for **RESTAURANT MANAGEMENT SYSTEM** are attached in this document.

7. REFERENCES

The below websites helped us in gaining more knowledge on the subject and in completing the project

- https://stackoverflow.com
- https://tutorialspoint.com
- https://youtube.com