RESTAURANT MANAGEMENT SYSTEM

A MINI PROJECT REPORT

Submitted by

MAIGYANA SIVA 220701157

MOHAMAD FAIZ 220701167

In partial fulfillment for the award of the degree of

BACHELOR OF ENGINEERING

IN

COMPUTER SCIENCE

RAJALAKSHMI ENGINEERING COLLEGE (AUTONOMOUS)

THANDALAM

CHENNAI-602105

**BONAFIDE CERTIFICATE**

Certified that this project report “LIBRARY MANAGEMENT SYSTEM” is the

bonafide work of “BHARATH E (200701042),ADITHYA NARAYANAN C

(200701010), GOKKUL BR (200701047),CLIFTON DANISH J (200701052) ”

who carried out the project work under my supervision.

Submitted for the Practical Examination held on

SIGNATURE SIGNATURE

Dr.R.SABITHA Ms.V.JANANEE

Professor and II Year Academic Head Assistant Professor (SG),

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 project is a restaurant management system named "Shetty's Cafe," designed to streamline and enhance restaurant operations using Streamlit for the front-end and MongoDB for the back-end. It features a secure authentication system with hashed passwords and role-based access control. The intuitive dashboard provides real-time metrics on menu items, reservations, orders, and staff, with customizable backgrounds for enhanced user experience. Menu management allows easy addition, viewing, updating, and deletion of menu items. Reservation management enables booking, viewing, updating, and deleting reservations through a user-friendly interface. Order management handles customer orders efficiently, including dine-in, takeout, and delivery options. Staff management facilitates adding, viewing, updating, and deleting staff information. The "About Us" section details the restaurant’s story, vision, mission, and values. Technically, the project employs Streamlit for a responsive user interface, MongoDB for robust data management, werkzeug.security for secure password handling, and base64 encoding for dynamic background images. "Shetty's Cafe" management system simplifies restaurant operations, allowing staff to focus on delivering exceptional service and culinary experiences.

**Table of Contents**

1**. Introduction**

- Project Overview

- Objectives

**2.Survey of Technologies**

- SOFTWARE DESCRIPTION

- LANGUAGES

PYTHON

-STREMLIT

MONGODB

3. **REQUIREMENTS AND ANALYSIS**

REQUIREMENT SPECIFICATION

HARDWARE AND SOFTWARE REQUIREMENTS

ARCHITECTURE DIAGRAM

ER DIAGRAM

**4. PROGRAM CODE**

**5. RESULTS AND DISCUSSION**

**6.CONCLUSION**

**7.REFERENCES**

CHAPTER 1

**INTRODUCTION**

The Restaurant Management System (RMS) is a comprehensive software solution designed to streamline and enhance the operations of restaurants. This project leverages the power of Streamlit for the user interface and MongoDB for data storage and management. By integrating key functionalities into a single platform, the RMS aims to improve efficiency, reduce operational complexity, and deliver an exceptional customer experience.

**Project Overview**

The RMS provides restaurant managers and staff with an intuitive and easy-to-use interface to handle various aspects of restaurant management, including menu items, reservations, orders, and staff. It features secure user authentication, allowing only authorized personnel to access the system. The system's design emphasizes simplicity and functionality, ensuring that users can quickly navigate through different modules and perform their tasks efficiently.

**Objectives**

The primary objectives of this project are to:

1. Simplify Daily Operations: Automate and streamline routine tasks such as managing menu items, reservations, orders, and staff details.

2. Enhance Customer Experience: Provide a seamless experience for customers through efficient reservation and order management.

3. Improve Operational Efficiency: Optimize resources and staff scheduling to ensure peak performance during busy hours.

4. Provide Data-Driven Insights: Utilize data analytics to offer valuable insights into restaurant performance, helping managers make informed decisions.

By achieving these objectives, the RMS aims to help restaurant owners and managers focus more on delivering excellent food and service, ultimately leading to higher customer satisfaction and business growth.

Chapter-2

**SURVEY OF TECHNOLOGIES**

**SOFTWARE DESCRIPTION**

**The Restaurant Management System (RMS) is a web-based application designed to streamline and manage the various operations of a restaurant. Built using modern web technologies, the RMS provides an intuitive interface for restaurant staff to manage menu items, reservations, orders, and staff details effectively. The system ensures secure authentication and efficient data handling, enhancing both operational efficiency and customer satisfaction.**

**Key Features and Modules**

**1. User Authentication:**

**- Login: Secure login functionality for authorized access.**

**- Sign Up: Allows new users to create an account with username and password.**

**- Password Hashing: Utilizes Werkzeug for secure password hashing and verification.**

**2. Home:**

**- Welcome Page: Displays a welcoming message and an overview of the restaurant's values and services.**

**- Dashboard: Presents key metrics such as total menu items, reservations, orders, and staff members.**

**3. About Us:**

**- Restaurant Information: Provides details about the restaurant's history, vision, mission, and values.**

**4. Manage Menu:**

**- Add Menu Item: Form to add new items to the restaurant's menu.**

**- View Menu: Displays a list of all menu items.**

**- Update/Delete Menu Item: Options to update or delete existing menu items.**

**5. Manage Reservations:**

**- Make a Reservation: Allows customers to book a table by providing necessary details.**

**- View Reservations: Displays a list of all reservations.**

**- Update/Delete Reservation: Options to update or delete existing reservations.**

**6. Manage Orders:**

**- Add Order: Form to add new customer orders.**

**- View Orders: Displays a list of all orders with details.**

**- Update/Delete Order: Options to update or delete existing orders.**

**7. Manage Staff:**

**- Add Staff Member: Form to add new staff members.**

**- View Staff: Displays a list of all staff members.**

**- Update/Delete Staff Member: Options to update or delete existing staff details.**

**LANGUAGE**

**1. Python:**

**- The primary programming language used for developing the RMS.**

**2. Streamlit:**

**- Framework for creating the web interface of the RMS. It enables rapid development of interactive web applications with real-time updates.**

**3. MongoDB:**

**- NoSQL database used for storing and managing all application data. Collections are used to store data related to users, menu items, reservations, orders, and staff.**

**- Stores staff details such as name, position, and contact information.**

Chapter3

**REQUIREMENTS AND ANALYSIS**

**The Restaurant Management System (RMS) aims to simplify and enhance the management of restaurant operations, providing tools for menu management, reservations, orders, and staff management. This section outlines the requirements and analysis for the RMS project.**

**Requirement Specification**

**1. User Authentication:**

**- Users should be able to securely log in and sign up.**

**- Passwords should be hashed and stored securely.**

**2. Menu Management:**

**- Users should be able to add, view, update, and delete menu items.**

**- Each menu item should have a name and price.**

**3. Reservation Management:**

**- Users should be able to make, view, update, and delete reservations.**

**- Each reservation should include customer name, date, time, and party size.**

**4. Order Management:**

**- Users should be able to add, view, update, and delete orders.**

**- Each order should include customer name, date, and items ordered.**

**5. Staff Management:**

**- Users should be able to add, view, update, and delete staff details.**

**- Each staff member should have a name, position, and contact information.**

**6. Dashboard:**

**- The system should display key metrics such as the total number of menu items, reservations, orders, and staff members.**

**7. User Interface:**

**- The UI should be intuitive and easy to navigate.**

**- The sidebar should provide quick access to all sections of the application.**

**8. Data Management:**

**- All data should be stored securely in a database.**

**- The system should provide efficient data retrieval and manipulation capabilities.**

**HARDWARE AND SOFTWARE REQUIREMENTS**

**Hardware Requirements:**

**1. Server:**

**- Processor: Quad-core CPU (e.g., Intel i5 or higher)**

**- Memory: 8 GB RAM or more**

**- Storage: 256 GB SSD or more**

**- Network: High-speed internet connection**

**2. Client:**

**- Processor: Dual-core CPU (e.g., Intel i3 or higher)**

**- Memory: 4 GB RAM or more**

**- Storage: 128 GB SSD or more**

**- Display: Full HD (1920x1080) monitor**

**- Network: High-speed internet connection**

**Software Requirements:**

**1. Operating System:**

**- Server: Linux (e.g., Ubuntu), Windows Server**

**- Client: Windows, macOS, Linux**

**2. Programming Language:**

**- Python 3.7 or higher**

**3. Frameworks and Libraries:**

**- Streamlit for the web interface**

**- Pandas for data manipulation**

**- Werkzeug for password hashing**

**4. Database:**

**- MongoDB for data storage and management**

**5. Development Tools:**

**- Code Editor/IDE: VS Code, PyCharm**

**- Version Control: Git**

**- Package Manager: pip**

**6. Dependencies:**

**- streamlit**

**- pymongo**

**- werkzeug**

**- pandas**

**DETAILED REQUIREMENT SPECIFICATION**

**1. User Authentication:**

**- Login:**

**- Users enter their username and password.**

**- System verifies credentials and grants access.**

**- Sign Up:**

**- Users create an account by providing a username and password.**

**- System hashes the password before storing it.**

**2. Menu Management:**

**- Add Menu Item:**

**- Form with fields for item name and price.**

**- Validation to ensure all fields are filled.**

**- View Menu:**

**- List of all menu items with names and prices.**

**- Update/Delete Menu Item:**

**- Option to update name and price or delete an item.**

**3. Reservation Management:**

**- Make a Reservation:**

**- Form with fields for customer name, date, time, and party size.**

**- Validation to ensure all fields are filled.**

**- View Reservations:**

**- List of all reservations with customer details, date, time, and party size.**

**- Update/Delete Reservation:**

**- Option to update details or delete a reservation.**

**4. Order Management:**

**- Add Order:**

**- Form with fields for customer name, date, and items ordered.**

**- Validation to ensure all fields are filled.**

**- View Orders:**

**- List of all orders with customer details, date, and items.**

**- Update/Delete Order:**

**- Option to update details or delete an order.**

**5. Staff Management:**

**- Add Staff Member:**

**- Form with fields for name, position, and contact information.**

**- Validation to ensure all fields are filled.**

**- View Staff:**

**- List of all staff members with names, positions, and contact information.**

**- Update/Delete Staff Member:**

**- Option to update details or delete a staff member.**

**6. Dashboard:**

**- Metrics Display:**

**- Display total counts of menu items, reservations, orders, and staff.**

**- Real-time updates to reflect current data.**

**7. User Interface:**

**- Navigation:**

**- Sidebar with links to Home, About Us, Manage Menu, Manage Reservations, Manage Orders, Manage Staff, and Logout.**

**- Responsive Design:**

**- Ensure the UI is accessible and functional on various devices and screen sizes.**

**By adhering to these requirements and specifications, the RMS will provide a robust and efficient system to manage restaurant operations effectively.**

**DATA FLOW DIAGRAM**

DFD is an important tool used by system analysis. A data flow diagram model, a system using external entities from which data flows through a process which transforms the data and creates output data transforms which go to other processes external entities such as files. The main merit of DFD is that it can provide an overview of what data a system would process.

• A data-flow diagram is a way of representing a flow of data through a process or a system.

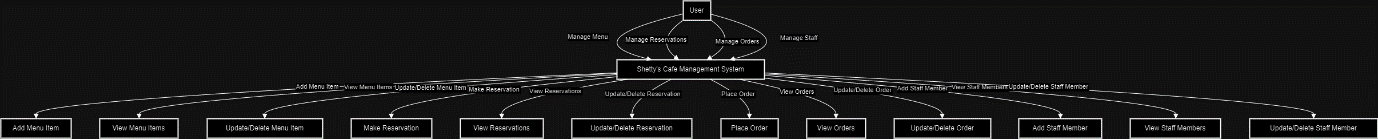
• The DFD also provides information about the outputs and inputs of each entity and the process itself.

**DFD LEVEL 0:**

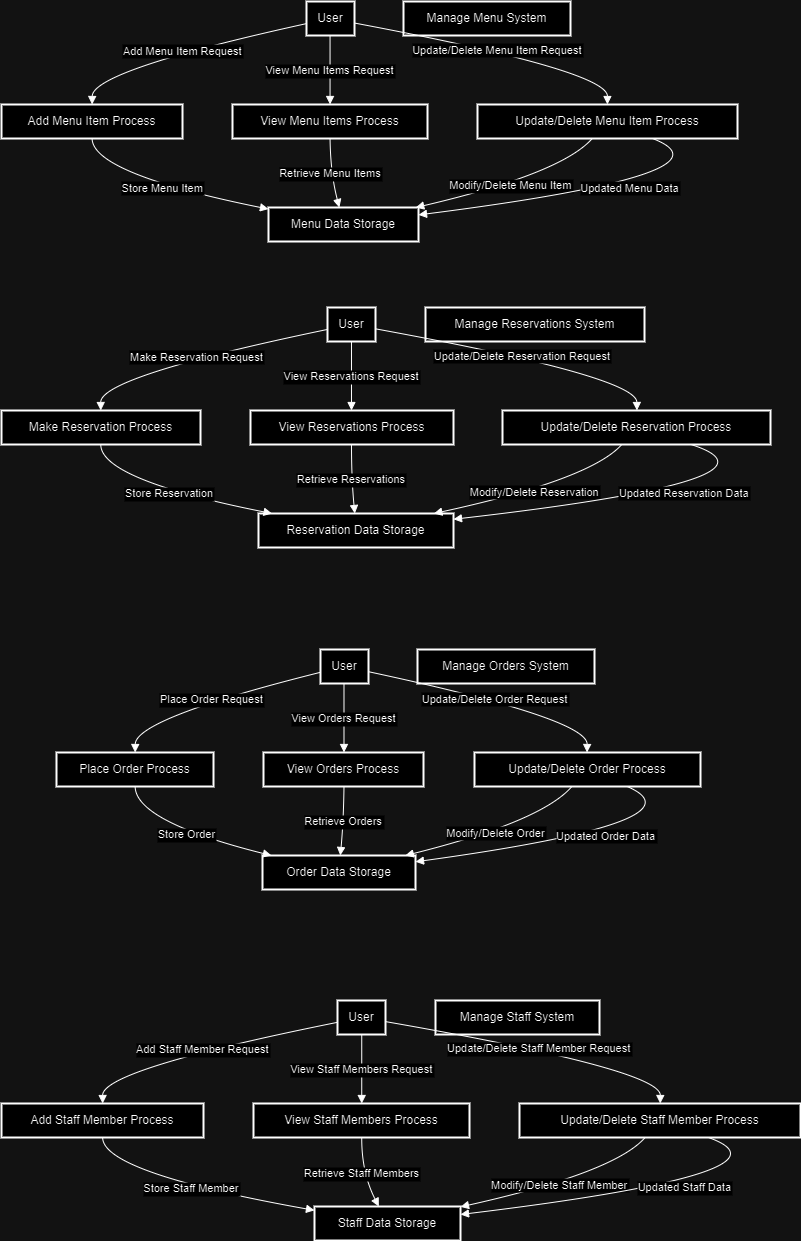
A diagram of a restaurant

Description automatically generated

**DFD LEVEL 1:**

****

**DFD LEVEL 2:**



**ER DIAGRAM:**

**A diagram of a server

Description automatically generated**

CHAPTER 4

**PROGRAM CODE:**

**THIS FILE CALLED DATABASE.PY TO MANAGE YOUR MONGODB CONNECTION :**

**from pymongo import MongoClient**

**from bson.objectid import ObjectId**

**from werkzeug.security import generate\_password\_hash, check\_password\_hash**

**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**

**THIS FILE CALLED APP.PY TO CREATE STRUCTURE OF STREAMLIT:**

**import base64**

**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**

**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('data:image/png;base64,%s');**

**background-size: cover;**

**}**

**</style>**

**''' % bin\_str**

**st.markdown(page\_bg\_img, unsafe\_allow\_html=True)**

**set\_background('coffee-bg.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.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.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", "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!!**

**At Shetty's Cafe, we understand that managing a restaurant is more than just serving delicious food—it's about creating unforgettable experiences for your customers while streamlining operations to maximize efficiency and profitability. Our state-of-the-art restaurant management system is designed to simplify your day-to-day tasks, allowing you to focus on what truly matters: delighting your guests.**

**With features tailored to meet the unique needs of your establishment, our comprehensive platform offers:**

**- \*Intuitive Order Management:\* Seamlessly handle orders from multiple channels—dine-in, takeout, and delivery—with real-time updates and efficient processing.**

**- \*Inventory Control:\* Keep track of stock levels, manage suppliers, and reduce waste.**

**- \*Staff Scheduling:\* Optimize your workforce with smart scheduling, shift management, and time tracking to ensure peak performance during busy hours.**

**- \*Customer Relationship Management:\* Build lasting relationships with your customers through personalized promotions, loyalty programs, and feedback collection.**

**- \*Sales Analytics:\* Gain valuable insights into your restaurant's performance with detailed reports and analytics to make data-driven decisions.**

**Join countless satisfied restaurant owners who have transformed their operations with [Your Restaurant Name] Management System. Explore our features, and discover how we can help you elevate your restaurant to new heights of success.**

**Ready to streamline your restaurant operations and enhance your customer experience? Sign up for a free demo today!**

**""")**

**# 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.**

**Our Values**

**- Quality: We never compromise on the quality of our ingredients.**

**- Customer Satisfaction: Our customers' happiness is our top priority.**

**- Integrity: We operate with honesty and transparency in everything we do.**

**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\_reservation = st.selectbox("Select Reservation", reservation\_list, format\_func=lambda x: x[0])**

**if selected\_reservation:**

**res = db.reservations.find\_one({"\_id": ObjectId(selected\_reservation[1])})**

**new\_name = st.text\_input("Your Name", value=res['name'])**

**new\_date = st.date\_input("Reservation Date", value=pd.to\_datetime(res['date']))**

**new\_time = st.time\_input("Reservation Time", value=pd.to\_datetime(res['time']).time())**

**new\_party\_size = st.number\_input("Party Size", min\_value=1, value=res['party\_size'])**

**if st.button("Update Reservation"):**

**update\_reservation(db, selected\_reservation[1], new\_name, new\_date, new\_time, new\_party\_size)**

**st.success("Reservation updated!")**

**if st.button("Delete Reservation"):**

**delete\_reservation(db, selected\_reservation[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\_area("Items (format: item1: quantity, item2: quantity)")**

**if st.button("Add Order"):**

**if customer\_name and date and items:**

**add\_order(db, customer\_name, date, items)**

**st.success(f"Added order for {customer\_name}!")**

**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: {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("Customer Name", value=order['customer\_name'])**

**new\_date = st.date\_input("Order Date", value=pd.to\_datetime(order['date']))**

**new\_items = st.text\_area("Items (format: item1: quantity, item2: quantity)", 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", "View Staff", "Update/Delete Staff"])**

**with tab1:**

**st.subheader("Add New Staff Member")**

**name = st.text\_input("Name")**

**positions = ["Manager", "Chef", "Waiter", "Cashier", "Cleaner", "Host", "Bartender"]**

**position = st.selectbox("Position",positions)**

**contact = st.text\_input("Contact")**

**if st.button("Add Staff Member"):**

**if name and position and contact:**

**add\_staff(db, name, position, contact)**

**st.success(f"Added staff member {name}!")**

**else:**

**st.error("Please fill in all fields.")**

**with tab2:**

**st.subheader("View Staff")**

**staff = get\_all\_staff(db)**

**for member in staff:**

**st.write(f"Name: {member['name']}")**

**st.write(f"Position: {member['position']}")**

**st.write(f"Contact: {member['contact']}")**

**st.write("---")**

**with tab3:**

**st.subheader("Update/Delete Staff Member")**

**staff = get\_all\_staff(db)**

**staff\_list = [(member['name'], member['\_id']) for member in staff]**

**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'])**

**positions = ["Manager", "Chef", "Waiter", "Cashier", "Cleaner", "Host", "Bartender"]**

**new\_position = st.selectbox("Position", positions, index=positions.index(member['position']))**

**new\_contact = st.text\_input("Contact", 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!")**

chapter 5

**RESULTS AND DISCUSSION:**

**FUNCTIONALITY OF THE RESTAURANT MANAGEMENT SYSTEM PROJECT:**

The Restaurant Management System project aims to streamline restaurant operations by providing a comprehensive platform for managing menu items, reservations, orders, and staff. The system offers an intuitive interface for users, including restaurant managers and staff, to efficiently handle various administrative tasks. Here is an overview of the key functionalities provided by the project:

1. User Authentication

Login: Allows users to securely log in to the system using their credentials. Utilizes hashed passwords for security.

Sign Up: Enables new users to create an account by providing a username and password.

2. Dashboard

Summary Metrics: Displays key metrics such as total menu items, reservations, orders, and staff members. Provides a quick overview of the restaurant’s operational status.

3. Manage Menu

Add Menu Item: Allows users to add new items to the restaurant's menu. Users can specify the name and price of each item.

View Menu: Provides a view of all menu items currently available, displaying their names and prices.

Update/Delete Menu Item: Enables users to update the details of existing menu items or delete them from the menu.

4. Manage Reservations

Make Reservation: Allows users to create new reservations by providing details such as the customer's name, reservation date and time, and party size.

View Reservations: Displays a list of all reservations, including customer names, dates, times, and party sizes.

Update/Delete Reservation: Enables users to modify existing reservations or delete them.

5. Manage Orders

Add Order: Allows users to place new orders, specifying the customer's name, order date, and a list of items ordered with their quantities.

View Orders: Provides a detailed view of all orders, including customer names, dates, and ordered items with quantities.

Update/Delete Order: Enables users to update the details of existing orders or delete them.

6. Manage Staff

Add Staff Member: Allows users to add new staff members by providing their name, position, and contact information.

View Staff Members: Displays a list of all staff members, including their names, positions, and contact details.

Update/Delete Staff Member: Enables users to update the details of existing staff members or delete them from the system.

7. User Interface and Experience

Background Setup: Provides a visually appealing background for the application using an image.

Sidebar Menu: Offers a navigation menu for users to switch between different sections of the application.

Hide Streamlit Default Components: Hides default Streamlit components (e.g., menu, header, and footer) to provide a cleaner interface.

Detailed Functionalities

User Authentication

Function: login()

Collects username and password from the user.

Authenticates the user against the database.

Sets session state to logged in upon successful authentication.

Function: signup()

Collects username, password, and password confirmation.

Creates a new user in the database if the passwords match.

Manage Menu

Function: add\_menu\_item(db, name, price)

Adds a new menu item to the database.

Function: get\_all\_menu\_items(db)

Retrieves all menu items from the database.

Function: update\_menu\_item(db, item\_id, name, price)

Updates the details of an existing menu item.

Function: delete\_menu\_item(db, item\_id)

Deletes a menu item from the database.

Manage Reservations

Function: add\_reservation(db, name, date, time, party\_size)

Creates a new reservation.

Function: get\_all\_reservations(db)

Retrieves all reservations from the database.

Function: update\_reservation(db, res\_id, name, date, time, party\_size)

Updates an existing reservation.

Function: delete\_reservation(db, res\_id)

Deletes a reservation from the database.

Manage Orders

Function: add\_order(db, customer\_name, date, items)

Places a new order.

Function: get\_all\_orders(db)

Retrieves all orders from the database.

Function: update\_order(db, order\_id, customer\_name, date, items)

Updates an existing order.

Function: delete\_order(db, order\_id)

Deletes an order from the database.

Manage Staff

Function: add\_staff(db, name, position, contact)

Adds a new staff member.

Function: get\_all\_staff(db)

Retrieves all staff members from the database.

Function: update\_staff(db, staff\_id, name, position, contact)

Updates an existing staff member's details.

Function: delete\_staff(db, staff\_id)

Deletes a staff member from the database.

Display All Tables

Function: display\_all\_tables(db)

Lists all collections (tables) in the database.

Displays the documents (records) from each collection in a table format.

**CHALLENGES FACED DURING DEVELOPMENT**

• Integration of Components:

i) Integrating different components such as user authentication, recipe

management, and image uploading presented challenges in ensuring

seamless communication between modules.

ii) Solution: Careful planning and use of consistent coding standards

helped mitigate integration issues.

• Database Design:

i) Designing a normalized database to handle complex relationships

between users, recipes, ingredients, and ratings was challenging.

ii) Solution: Iterative testing and refinement of the database schema

ensured a robust and efficient database design.

• Responsive Design:

i) Ensuring that the web application was fully responsive across

different devices required extensive testing and adjustments to CSS

and layout.

ii) Solution: Utilizing Bootstrap and thorough testing on various devices

helped achieve a responsive design

**SECURITY CONCERNS:**

i) Implementing secure user authentication and data protection

measures was crucial to protect user information.

ii) Solution: Use of prepared statements to prevent SQL injection, HTTPS

for secure data transmission, and thorough validation of user inputs

were implemented to enhance security.

• User Experience (UX) Design:

i) Designing an intuitive and user-friendly interface that caters to all

user needs was a continuous challenge.

ii) Solution: Regular feedback from test users and iterative design

adjustments helped improve the overall UX

**CONCLUSION :**

**The Restaurant Management System project effectively addresses the multifaceted needs of modern restaurant operations by integrating functionalities for menu management, reservations, orders, and staff management into a single platform. This system enhances operational efficiency and improves the overall dining experience for customers. It streamlines daily tasks such as adding menu items, handling reservations, processing orders, and managing staff, thereby reducing administrative burden and allowing staff to focus more on customer service. Secure login and signup mechanisms ensure authorized access, protecting sensitive information. The comprehensive dashboard offers quick access to key metrics, providing an overview of the establishment’s performance and aiding in informed decision-making.**

**Efficient handling of CRUD operations ensures data accuracy across different modules, and the user-friendly interface, enhanced with custom backgrounds, ensures seamless navigation. The system saves time and reduces errors, leading to more efficient restaurant management and a more satisfying customer experience. Future enhancements could include advanced analytics, mobile app integration, and a customer-facing interface for online reservations and orders. Overall, the Restaurant Management System is a comprehensive solution that simplifies administrative tasks, enhances efficiency, and contributes to a better dining experience, with potential for further evolution to meet the growing demands of the restaurant industry.**

**REFERENCE:**

MongoDB Documentation: Comprehensive guide on using MongoDB for database management, including CRUD operations and security measures.

URL: MongoDB Documentation

Streamlit Documentation: Official documentation for Streamlit, covering installation, basic usage, and advanced features for building web applications.

URL: Streamlit Documentation

Python Programming Language: General reference for Python programming, including standard libraries and common practices used in the project.

URL: Python Official Documentation

Werkzeug Security: Documentation on password hashing and security best practices using Werkzeug, a comprehensive WSGI web application library.

URL: Werkzeug Security

Pandas Documentation: Reference for using the Pandas library for data manipulation and analysis in Python.

URL: Pandas Documentation

BSON Library: Guide on using BSON for encoding and decoding data, particularly with MongoDB.

URL: BSON Documentation

Streamlit Forum and Community: Resource for troubleshooting, community support, and examples of various use cases and implementations.

URL: Streamlit Community

Base64 Encoding: Reference for encoding and decoding binary data using Base64, used in the project for background image setup.

URL: Base64 Encoding

HTML and CSS for Styling: General reference for using HTML and CSS to enhance the visual aspects of the web application.

URL: W3Schools HTML and CSS