**Restaurant Management System**

A PROJECT REPORT

#### Submitted by

ANISHA PANDEY [ RA2311056010019 ]

SAKET RUIA [ RA2311056010045 ]

#### Under the Guidance of

## Dr. Yasotha B

Assistant Professor, Department of Data Science and Business Systems

### *in partial fulfillment of the requirements* *for the degree of*

## BACHELOR OF TECHNOLOGY

## in

## COMPUTER SCIENCE AND ENGINEERING

## DATA SCIENCE



## DEPARTMENT OF DATA SCIENCE AND BUSINESS SYSTEMS

## COLLEGE OF ENGINEERING AND TECHNOLOGY

SRM INSTITUTE OF SCIENCE AND TECHNOLOGY

## KATTANKULATHUR- 603 203

### NOVEMBER 2024



# SRM INSTITUTE OF SCIENCE AND TECHNOLOGY KATTANKULATHUR – 603 203

## BONAFIDE CERTIFICATE

Certified that this minor project report for the course **21CSC206P ADVANCED OBJECT ORIENTED AND PROGRAMMING** entitled in **" Restaurant Management System** “is the Bonafide work ofAnisha Pandey [RA2311056010019] and Saket Ruia [RA2311056010045] who carried out the work under my supervision.

**SIGNATURE SIGNATURE**

**Dr. Yasotha B Dr. Kavitha V**

**SUPERVISOR HEAD OF DEPARTMENT**

**ASSISTANT PROFESSOR PROFESSOR**

**DEPARTMENT OF DATA DEPARTMENT OF DATA**

**SCIENCE AND BUSINESS SCIENCE AND BUSINESS**

**SYSTEMS SYSTEMS**

**ACKNOWLEDGEMENTS**

We express our heartfelt thanks to our honourable Vice Chancellor Dr. **C.Muthamizhchelvan**, for being the beacon in all our endeavours.

﻿﻿We would like to express my warmth of gratitude to our Registrar Dr. **S.Ponnusamy,** for his encouragement.

﻿﻿We express our profound gratitude to our Dean (College of Engineering and Technology) **Dr. T. V. Gopal,** for bringing out novelty in all executions.

﻿﻿We extend my gratitude to our HOD **Dr. Kavitha V**, Professor, Department of Data Science and Business Systems and my Departmental colleagues for their Support.

﻿﻿We would like to express my heartfelt thanks to Chairperson, School of Computing Dr. Revathi Venkataraman, for imparting confidence to complete my course project.

﻿﻿We wish to express my sincere thanks to Course Audit Professors **Dr. Lakshmi M**, Professor & Head, Department of Networking and Communication for her constant encouragement and support.

﻿﻿We are highly thankful to my Course project Faculty **Dr. Yasotha B**, Assistant Professor, Department of Data Science and Business Systems for her assistance, timely suggestion and guidance throughout the duration of this course project.

Finally, we thank our parents and friends near and dear ones who directly and indirectly contributed to the successful completion of our project.

Above all, I thank the almighty for showering his blessings on me to complete my Course project.

Anisha Pandey

Saket Ruia

Authors

**ABSTRACT**

The Restaurant Management System is a comprehensive software solution developed to automate essential operations in restaurant management, enhancing accuracy, efficiency, and customer experience. Built with Java Swing for an intuitive graphical user interface and MySQL as the backend database, this system provides restaurant managers and staff with a user-friendly, streamlined platform. Core functionalities of the system include reservation management, menu configuration, and billing processing, each designed to reduce manual workloads and mitigate errors associated with traditional methods. The reservation module allows for seamless booking and tracking of tables, enabling managers to maximize seating arrangements based on availability. Additionally, the billing module organizes transaction processes, ensuring accurate and quick billing for a better customer experience. By centralizing these operations within a cohesive platform, the system not only simplifies daily tasks but also ensures consistent service standards across the restaurant.

Beyond these foundational capabilities, the Restaurant Management System is designed to support scalability for future functionality enhancements, positioning it as an adaptable tool for dynamic restaurant environments. Planned improvements include the integration of a secure payment gateway, allowing managers to facilitate seamless digital transactions directly through the system, which would reduce cash handling and speed up payment processing. Additionally, advanced analytics and reporting features are envisioned, utilizing visualization libraries such as JFreeChart or JavaFX, to provide actionable insights into key metrics. This analytical feature aims to transform raw data into visual insights, allowing managers to identify trends and make informed adjustments to menus, staffing, and other operations. Ultimately, the Restaurant Management System is a robust solution that improves operational efficiency, enhances customer satisfaction, and contributes to long-term business growth. As restaurants increasingly seek technology-driven solutions to remain competitive, this system offers an innovative approach to managing resources effectively, enabling smoother day-to-day operations and helping restaurants stay adaptable to market demands.

**TABLE OF CONTENTS**

|  |  |  |
| --- | --- | --- |
| **CHAPTER NO** | **CONTENTS** | **PAGE NO** |
|  | **ABSTRACT** |  |
| **1** | **INTRODUCTION** | **1** |
| **2** | **LITERATURE SURVEY** | **6** |
| **3** | **REQUIREMENT ANALYSIS** | **9** |
| **4** | **ARCHITECTURE & DESIGN** | **12** |
| **5** | **IMPLEMENTATION** | **15** |
| **6** | **RESULTS AND DISCUSSION** | **20** |
| **7** | **CONCLUSION** | **25** |
| **8** | **REFERENCES** | **26** |

**CHAPTER 1**

**INTRODUCTION**

The Restaurant Management System is a comprehensive software solution developed to transform and optimize restaurant operations by automating core tasks, including reservations, menu management, and billing processes. With Java Swing used for the graphical user interface and MySQL as the backend database, this system offers a robust and user-friendly platform designed to meet the unique needs of restaurant managers and staff. In a fast-paced restaurant environment, managing reservations, updating menus, and processing customer orders manually often results in delays, errors, and a subpar customer experience. This project aims to modernize these processes, providing a streamlined, efficient solution that ensures accuracy and consistency in daily operations. By automating these critical functions, the Restaurant Management System enhances productivity, reduces manual workload, and fosters a more satisfying dining experience for customers.

**1.1 Motivation:**

The motivation behind the Restaurant Management System lies in the growing demand for operational efficiency, precision, and enhanced customer experiences in the restaurant industry. In today’s competitive landscape, customers expect quick, reliable, and seamless service, which traditional, manual processes often fail to deliver. Managing reservations, updating menus, and processing orders and payments manually is not only time-consuming but also prone to significant human error, particularly during peak hours when staff are under pressure. Issues such as double-booking tables, offering unavailable menu items, and miscalculating bills are common, and they can detract from the overall dining experience, leading to customer dissatisfaction and potential loss of business. Additionally, without real-time tracking and centralized management, restaurants face challenges in maintaining an accurate view of inventory, which often results in over-ordering or stockouts, directly impacting service efficiency and profitability.

Beyond customer expectations, there is an industry-wide push towards digital transformation, where businesses adopt technology to optimize workflows, reduce costs, and improve data management. The restaurant industry is no exception, as technology has become essential for streamlining day-to-day operations and meeting the needs of a modern, digital-savvy customer base. By implementing an automated system, restaurants can achieve faster, more reliable service while reducing the burden on staff, who can then focus on customer engagement and quality service rather than repetitive administrative tasks. Moreover, an effective management system provides valuable data insights that help managers make informed decisions on inventory, staffing, and customer preferences, contributing to better business planning and optimized resource allocation.

The motivation for this project is to create a solution that addresses these industry challenges, providing restaurant managers with a comprehensive, easy-to-use platform that not only automates essential tasks but also enhances operational visibility and control. This project aims to bridge the gap between traditional restaurant management methods and the demands of a fast-evolving market, offering a scalable, adaptable system that aligns with the technological needs and growth aspirations of modern restaurants.

**1.2 Problem Statement:**

The restaurant industry faces numerous operational challenges due to its reliance on manual management processes, which often result in inefficiencies, errors, and poor customer experiences. Restaurants that still use traditional methods to manage reservations, update menus, and process customer orders often encounter issues such as overbooked tables, inconsistent menu information, and lengthy billing processes.

For example, without a centralized, real-time system, it is difficult for managers to track table availability accurately, leading to overbookings or delays in seating customers. These issues are exacerbated during busy hours, where manual processes struggle to keep up with demand, creating stress for staff and potentially leading to customer dissatisfaction.

Moreover, managing inventory and updating menus manually are labour-intensive and error-prone tasks, particularly in high-turnover environments. When menu items are not accurately updated in real-time, it can result in customer disappointment when they attempt to order unavailable dishes. This affects the overall perception of service quality and reduces customer satisfaction.

The billing process also poses a challenge, as manual calculations increase the risk of errors, delay the payment process, and detract from the dining experience. Without an automated system to handle these functions, restaurants are unable to meet the high standards of efficiency and accuracy that customers expect in a modern dining environment.

Therefore, the *Restaurant Management System* seeks to resolve these operational issues by centralizing and automating essential tasks, enhancing the accuracy of reservations, menu management, and billing. By reducing the reliance on manual processes, this system offers a comprehensive solution that can improve both the efficiency of restaurant operations and the satisfaction of customers.

**1.3 Objectives:**

The primary objectives of the Restaurant Management System project are to automate and streamline critical restaurant operations to enhance efficiency, accuracy, and customer satisfaction.

These objectives are aligned with the goal of creating a cohesive and effective management solution for restaurants.

Enhance Reservation Management: The system will enable real-time tracking of table availability, preventing double bookings and improving customer flow management. With a structured reservation system, restaurants can maximize seating efficiency and minimize conflicts in table assignments, especially during peak hours.

* + 1. Efficient Menu Management: By providing a platform for easy menu updates, the system aims to improve the accuracy and availability of menu information. Staff will be able to update descriptions, prices, and availability instantly, ensuring that customers always see the latest offerings and avoiding the disappointment that comes with unavailable items.
    2. Automated Billing and Order Processing: Automation of billing processes reduces human error, expedites payment, and improves the overall customer experience. By providing an organized approach to order processing, the system ensures accurate and timely transactions, allowing customers to enjoy a seamless payment experience and reducing workload on staff.
    3. Enhanced User Experience: The system’s intuitive interface is designed to streamline interactions for restaurant staff and managers, simplifying navigation and minimizing the need for extensive training. This design goal enhances usability and supports a smoother operational workflow.
    4. Scalability and Future Expansion: Built to adapt to changing industry needs, the system is designed with scalability in mind, allowing for future integrations, such as digital payments, customer analytics, and personalized services. This forward-looking objective ensures that the system remains relevant as the restaurant grows and evolves.

These objectives collectively aim to provide a reliable, user-centered management solution that addresses the needs of restaurants looking to improve their service quality, operational efficiency, and long-term adaptability.

**1.5 Scope of Work:**

The scope of the *Restaurant Management System* encompasses the development of a fully functional, user-friendly software application that supports essential restaurant operations. This includes real-time reservation management, dynamic menu updates, and automated billing, all integrated within a single platform. The system will be built using Java Swing for the interface and MySQL for data management, ensuring a stable and scalable foundation that can handle the demands of various restaurant sizes.

* + 1. **Reservation Management Module**: This module provides a structured, centralized approach to booking and managing table reservations, minimizing the chances of double bookings and improving customer satisfaction. It allows managers to view and allocate available tables efficiently, ensuring a smoother customer experience and maximizing seating capacity.

**1.5.2 Menu Management Module**: Designed to offer flexibility and convenience, this module allows staff to easily update menu items, including their descriptions, prices, and availability. With real-time menu updates, restaurants can ensure customers have access to accurate information, reducing ordering conflicts and enhancing service quality.

**1.5.3 Billing and Order Processing Module**: The automated billing system organizes and simplifies order processing, ensuring accurate and prompt billing for customers. This module is designed to support a seamless payment experience, reducing the risk of errors and making it easier for staff to handle transactions quickly and efficiently.

**1.5.4 User-Friendly Interface Design**: The system’s interface, created using Java Swing, prioritizes ease of use and accessibility, allowing even non-technical staff to navigate the platform with minimal training. The interface features organized layouts and intuitive controls to ensure smooth, efficient workflows for users.

**Future Expansion Capabilities**: The system is developed with scalability in mind, supporting future enhancements such as online payment integration, analytics for customer and sales data, and preference tracking for personalized services. This adaptable design ensures that the system can grow alongside the restaurant, providing an evolving solution that meets the demands of an ever-changing industry.

This scope outlines a project aimed at delivering a robust, adaptable, and user-centered solution to address common challenges in restaurant management, with the flexibility to integrate future innovations as the industry continues to evolve.

**CHAPTER 2**

**LITERATURE SURVEY**

Restaurant management systems are integral to the efficient functioning of modern food and beverage establishments. These systems automate essential operations like reservation management, order processing, billing, and inventory control. The primary objectives of these systems are to improve accuracy, reduce manual intervention, and enhance customer service. In recent years, advancements in software development tools, database management, and graphical user interfaces have made RMS solutions more accessible, intuitive, and adaptable to various restaurant sizes and types.

**2.1 Historical Development of Restaurant Management Systems**

The evolution of RMS began in the 1970s with basic point-of-sale (POS) systems designed to facilitate payment processing. Early systems were limited in functionality, primarily focusing on transaction logging and financial tracking. By the 1980s and 1990s, advances in computing allowed for the integration of inventory management and order tracking capabilities. Modern RMS solutions are comprehensive platforms that include POS, customer relationship management (CRM), and real-time analytics, providing restaurant owners with detailed insights into operational performance.

**2.2 Core Components of Contemporary RMS**

A typical RMS today comprises several modules, each catering to a specific aspect of restaurant operations. In the case of the RMS described in the document, the project uses Java Swing for its graphical user interface and MySQL for backend database management, underscoring a trend toward using modular software tools in RMS development.

Key modules in contemporary RMS include:

1. **Reservation Management**: Allows customers or staff to reserve tables, view available seating, and track upcoming reservations. Automating this process reduces errors associated with manual booking and ensures optimized seating utilization.
2. **Menu Management**: This module enables restaurant managers to update menu items, adjust prices, and track item availability. It is essential for reducing human errors and ensuring that servers have accurate information on available items.
3. **Order and Billing Processing**: Order processing is vital for streamlining food preparation and service workflows. Integrating billing with order processing provides a seamless experience for customers and reduces wait times. Systems using real-time data processing frameworks, such as Java and MySQL, improve accuracy and speed.
4. **Inventory Management**: Advanced RMS solutions include inventory tracking to monitor stock levels, track usage patterns, and notify management of low stock. While this component is not explicitly covered in the presented RMS, it remains critical for restaurant efficiency, as it ensures ingredients are always available without overstocking.

**2.3. Technologies and Tools Used in RMS Development**

The chosen tools for RMS development impact its functionality, scalability, and usability. In this project, Java Swing is used for the graphical user interface, and MySQL serves as the relational database. Such a combination of technologies is common in small-to-medium RMS due to the following advantages:

1. **Java Swing**: Swing is widely used for developing desktop applications because of its robust GUI components, flexibility, and ability to create a consistent user experience across platforms. Its use in the RMS project enables developers to implement essential interface components, such as menus, tables, and input forms, essential for smooth restaurant operations.
2. **MySQL Database**: MySQL provides reliable data management, which is essential for storing and retrieving large volumes of structured data. Through Java Database Connectivity (JDBC), Java applications can interact with MySQL, allowing real-time data updates in response to system inputs. This approach ensures that all operational data, including reservations, menu details, and billing, are accurately managed.
3. **Other Libraries and Frameworks**: The project utilizes a mix of additional Java libraries, including java.awt for graphical layout and java.sql for database interactions. This modular approach allows the system to handle complex functions, like date and time formatting, and facilitates user interactions through button clicks.

**2.4. Impact on Operational Efficiency and Customer Experience**

The integration of RMS has a profound impact on both operational efficiency and customer satisfaction. Automating reservations, orders, and billing improves the speed and accuracy of service. This system reduces the likelihood of errors in order-taking, minimizes waiting times, and enables restaurants to offer prompt, consistent service, which directly influences customer experience.

Studies show that RMS can improve profitability by reducing labour costs and optimizing inventory usage. According to industry research, restaurants with fully implemented RMS platforms can expect up to a 15% reduction in operational costs due to streamlined processes and better resource management. The system described in the project presentation aligns with these findings by offering a scalable solution that could reduce manual errors and provide long-term cost savings.

**2.5. Future Enhancements and Potential Developments**

The future of RMS development points toward more integrated, data-driven systems. As suggested in the document, integrating payment gateways (like Stripe or PayPal) would allow for direct customer billing, improving cash flow and reducing manual processing errors. Additionally, analytics and reporting tools, potentially built using libraries like JavaFX, could provide managers with insights into customer preferences, peak hours, and revenue trends. These features enable more informed decision-making, helping restaurants stay competitive.

**CHAPTER 3**

**REQUIREMENT ANALYSIS**

* 1. **Operating System Compatibility**

The system is designed to be compatible across multiple operating systems, including Windows, MacOS, and Linux. This cross-platform capability provides flexibility for the end-users, allowing the application to run smoothly on different devices regardless of the operating system they use.

**Rationale**: Cross-platform compatibility is crucial for expanding the user base and ensuring that the system can be deployed in a variety of environments without modification, accommodating the technical diversity of different users and devices.

**Development Tools**

The following development tools are required to effectively build, test, and maintain the Restaurant Management System.

**3.1.1 Java Development Kit (JDK 8)**

*Purpose*: The JDK provides the core environment for Java development. It includes essential tools like the compiler, libraries, and utilities needed to write, compile, and run Java code.

*Requirement*: JDK 8 is chosen because it offers stability and broad compatibility with the libraries and frameworks used in the project. It is a widely supported version that balances performance with reliability.

*Rationale*: Selecting a stable JDK version like JDK 8 ensures that the application can utilize Java features while maintaining compatibility with various libraries that might not yet be optimized for newer versions.

**3.1.2 Integrated Development Environment (IDE): IntelliJ IDEA**

*Purpose*: IntelliJ IDEA is recommended for developing the application due to its advanced features specifically tailored for Java programming, such as debugging tools, code refactoring, and built-in support for GUI development.

*Requirement*: An IDE is necessary to streamline development, improve productivity, and reduce errors through features like syntax highlighting, autocompletion, and integrated version control. While other IDEs (e.g., Eclipse, NetBeans) can be used, IntelliJ IDEA is preferred for its user-friendly interface and extensive Java support.

*Rationale*: A professional IDE enhances code quality and simplifies development, allowing for efficient testing, debugging, and project management.

**3.1.3 MySQL Database**

*Purpose*: MySQL serves as the backend database for the system, responsible for storing structured data such as customer details, reservations, order histories, and billing records.

*Requirement*: MySQL is a reliable and widely-used database management system, well-suited for handling large amounts of relational data. Its compatibility with SQL standards makes it ideal for managing, querying, and updating data in a structured format.

*Rationale*: MySQL’s stability and scalability make it an excellent choice for the system’s database requirements. It ensures data integrity, supports concurrent transactions, and has a strong community backing, which aids in long-term maintenance and troubleshooting.

**3.1.4 JDBC (Java Database Connectivity)**

*Purpose*: JDBC enables the connection between the Java application and the MySQL database, allowing data to flow seamlessly between the user interface and backend.

*Requirement*: JDBC provides a bridge between Java and SQL, enabling the application to execute SQL queries, retrieve results, and update records in the database. It is an integral part of Java’s data access capabilities.

*Rationale*: JDBC standardizes database access, making it easier to integrate SQL databases into Java applications. Its use allows for efficient data retrieval and manipulation, which is vital for managing customer orders, inventory, and billing in real-time.

* 1. **Libraries and Frameworks**

Specific libraries and frameworks are necessary to manage the graphical user interface and database connectivity effectively.

**3.2.1 Java Swing**

*Purpose*: Java Swing is used to build the application’s graphical user interface, creating interactive elements such as windows, buttons, tables, and forms that users can interact with directly.

*Requirement*: Java Swing is essential for developing a responsive and visually appealing user interface that allows easy navigation and efficient data input.

*Rationale*: Java Swing provides a comprehensive and customizable GUI framework that enables developers to design user-friendly interfaces, making it ideal for a desktop application like this restaurant management system.

**3.2.2 MySQL Connector/J**

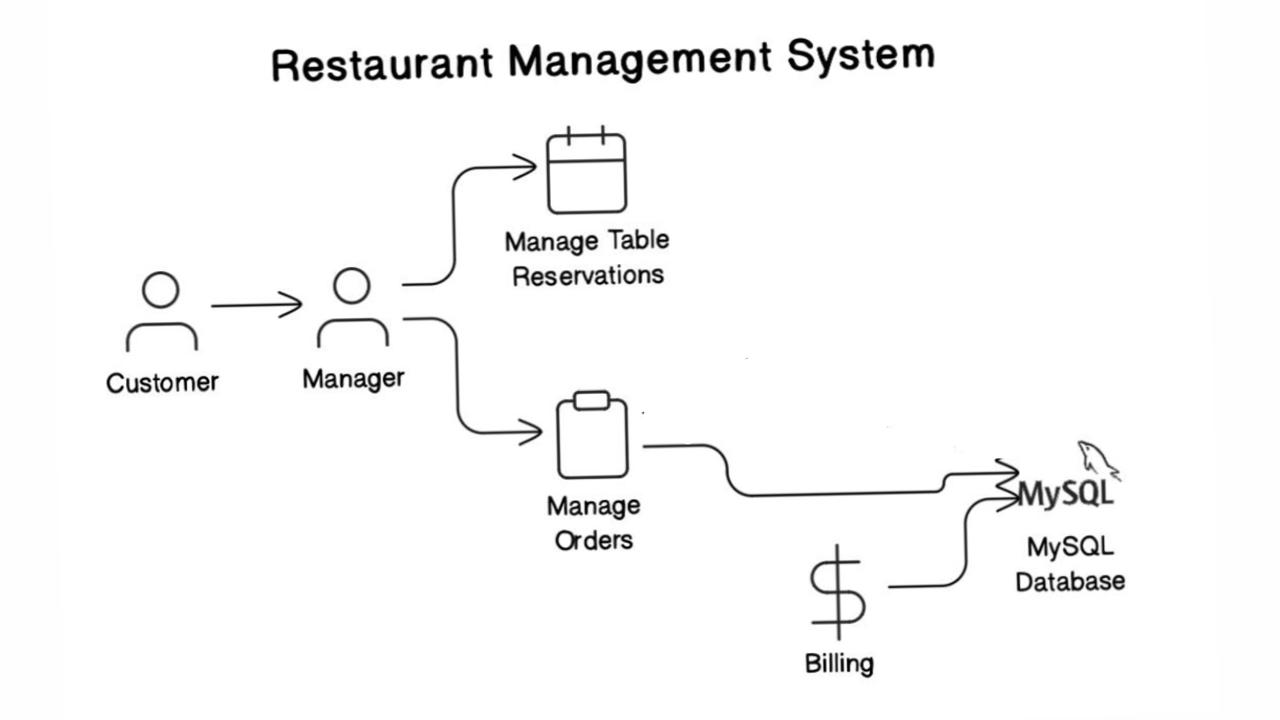
*Purpose*: MySQL Connector/J is a JDBC driver that allows Java applications to connect to and interact with MySQL databases, making it possible to execute queries and retrieve data from the database within the Java environment.

*Requirement*: This connector is essential for facilitating smooth communication between the Java application and MySQL, ensuring that data can be accessed and updated as needed.

*Rationale*: MySQL Connector/J enables seamless integration of MySQL with Java, ensuring reliable data exchange and simplifying database interactions for the application.

**CHAPTER 4**

A**CHITECTURE & DESIGN**

**Figure 4.1 Architecture Diagram**

**4.1. Customer**

* The customer represents the end-user who visits the restaurant to dine. Although they do not directly interact with the system itself, the services they receive (such as table reservations, ordering, and billing) are facilitated through the system.
* Their experience is indirectly managed by the system, which enables the manager to efficiently handle customer requests like reserving a table or placing orders.

**4.2. Manager**

* The **manager** is the primary user and operator of the system. Their responsibilities involve using the system to meet the needs of the customers, thereby creating a streamlined experience. The manager’s interactions with the system include two main operations: **Managing Table Reservations** and **Managing Orders**.

**4.2.1 Manage Table Reservations**

* The manager handles customer table reservations through the system. This process involves checking the availability of tables and reserving tables for specific times, dates, and customer requests.
* When a reservation is made, the system:
  + **Updates the Database**: Information like the table number, reservation time, and customer details (if required) is recorded in the MySQL database.
  + **Tracks Availability**: The system keeps track of reserved tables to prevent double bookings and allows the manager to view or modify reservations.
  + **Example Scenario**: If a customer calls in to reserve a table, the manager checks available tables through the system and enters the reservation details, which are saved to the database for reference.

**4.2.2 Manage Orders**

* The manager also uses the system to take and manage orders from customers, especially if multiple orders are placed at different times by various tables.
* For each order:
  + **Order Entry**: The manager enters each order into the system, specifying details such as table number, ordered items, and quantities.
  + **Inventory Management**: The system could be linked with an inventory to automatically update quantities of ingredients or stock items based on each order.
  + **Real-Time Data Storage**: Each order is stored in the MySQL database in real-time, making it accessible for billing and tracking.
  + **Example Scenario**: When customers at a table order multiple items, the manager inputs each item into the system, and it’s saved in the database under the respective table. The manager can also modify or add to orders as requested by the customers.

**4.3. Billing**

* Once orders are completed, the manager uses the system to generate a **bill** for each table.
* The billing process involves:
  + **Calculating Total Cost**: The system sums up the prices of each ordered item, applying any taxes or discounts if applicable.
  + **Generating and Storing Bills**: Each bill is saved in the MySQL database with essential details like the table number, timestamp, list of ordered items, and the total amount due.
  + **Display and Print Option**: The system might allow the manager to display the bill to the customer or print it for them. This ensures transparency and convenience.
  + **Example Scenario**: After the meal, the manager generates the bill for the table, which is automatically calculated and formatted. The bill is stored in the database with a timestamp, ensuring a record of all transactions for future reference.

**4.4. MySQL Database**

* The **MySQL Database** is the backend storage component that supports all data processing and retrieval needs of the system. It acts as the central repository for all information related to reservations, orders, and billing.
* Key tables and their functions might include:
  + **Reservations Table**: Stores data on all table reservations, such as table number, reservation time, and customer details.
  + **Orders Table**: Tracks every order placed by each table, including item details, quantities, and timestamps.
  + **Billing Table**: Records each generated bill, including the table number, items ordered, total cost, and the billing timestamp.
* The database ensures that all operations performed by the manager are stored securely and can be accessed for reporting, inventory management, customer history, and auditing purposes.

**4.5. System Workflow Overview**

* The workflow of the system can be summarized as follows:

**4.5.1 Customer Interacts with Manager**: The customer requests services (reservations, orders).

* + 1. **Manager Uses the System**:
    - Manages reservations to ensure tables are efficiently allocated.
    - Handles orders by inputting customer choices, updating inventory, and tracking each item ordered.

**4.5.3 Billing Process**: Once the service is complete, the system generates a bill, which the manager presents to the customer. The bill details are stored in the database.

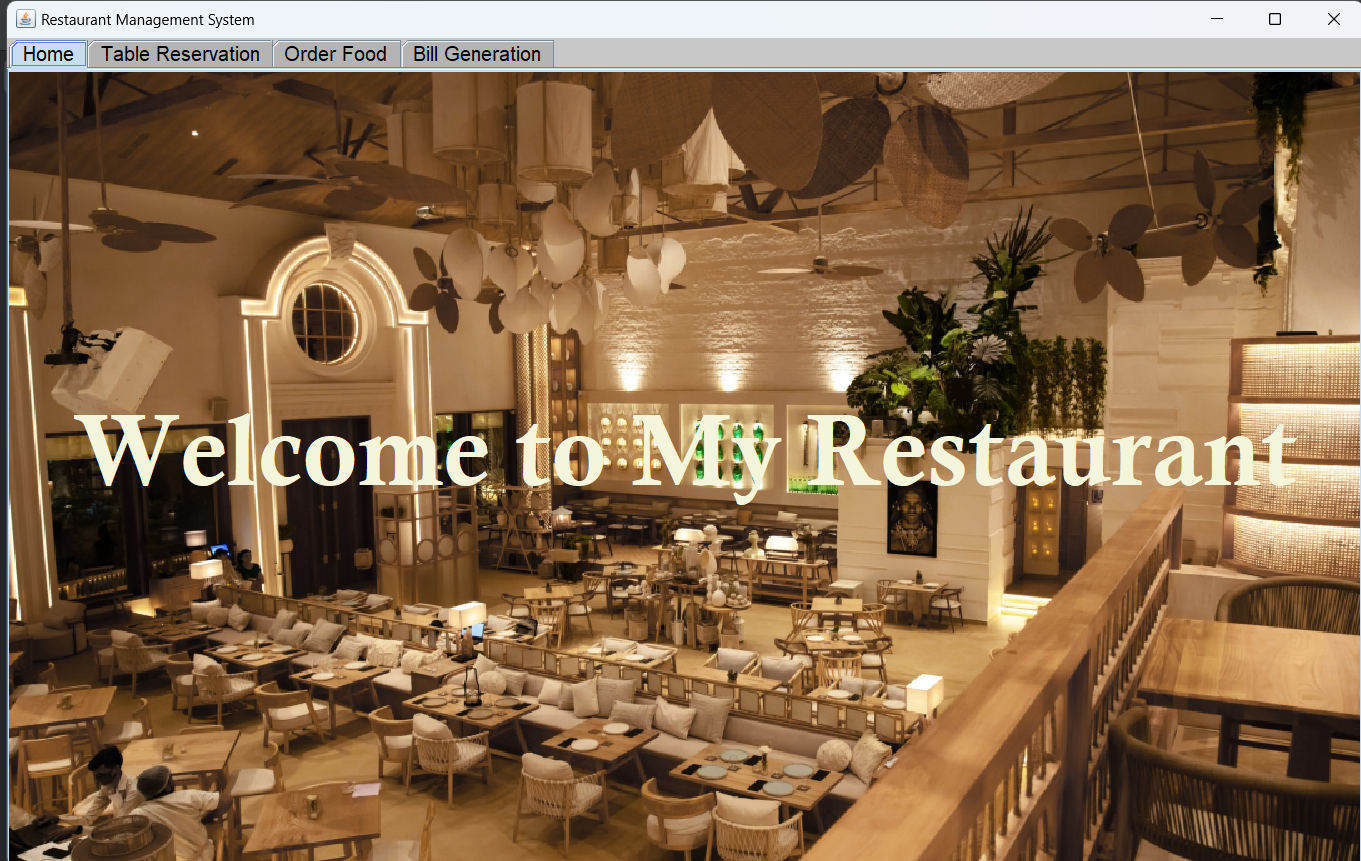
**CHAPTER 5**

**IMPLEMENTATION**

import javax.swing.\*;  
import java.awt.\*;  
import java.awt.event.ActionEvent;  
import java.awt.event.ActionListener;  
import java.sql.\*;  
import java.time.LocalDateTime;  
import java.time.format.DateTimeFormatter;  
import java.util.Enumeration;  
class RestaurantManagementSystem {  
 // To store ordered items quantities for each table  
 static int[][] *tableOrders* = new int[7][12]; // 7 tables and 12 food items  
 // Database credentials  
 private static final String *DB\_URL* = "jdbc:mysql://localhost:3306/restaurant\_management";  
 private static final String *USER* = "root";  
 private static final String *PASSWORD* = "saket\_18";  
 public static void main(String[] args) {  
 // Create the main frame (window)  
 JFrame frame = new JFrame("Restaurant Management System");  
 frame.setDefaultCloseOperation(JFrame.*EXIT\_ON\_CLOSE*);  
 frame.setSize(1100, 700);  
 frame.getContentPane().setBackground(new Color(200, 200, 200));  
 // Create a JTabbedPane to hold the three tabs  
 JTabbedPane tabbedPane = new JTabbedPane();  
 tabbedPane.setBackground(new Color(180, 180, 180));  
 tabbedPane.setForeground(Color.*BLACK*);  
 tabbedPane.setFont(new Font("Arial", Font.*PLAIN*, 16));  
 JPanel homePanel = *createhomePanel*();  
 tabbedPane.addTab("Home", homePanel);  
 //Table Reservation Tab  
 JPanel reservationPanel = *createReservationPanel*();  
 tabbedPane.addTab("Table Reservation", reservationPanel);  
 //Order Food Tab  
 JPanel orderPanel = *createOrderPanel*(tabbedPane);  
 tabbedPane.addTab("Order Food", orderPanel);  
 //Billing Management Tab  
 JPanel billingPanel = *createbillingPanel*();  
 tabbedPane.addTab("Bill Generation", billingPanel);  
 //Add Tabbed Pane to Frame  
 frame.add(tabbedPane);  
 frame.setVisible(true);  
 }  
 private static JPanel createhomePanel() {  
 JPanel homePanel = new JPanel();  
 homePanel.setLayout(new BorderLayout()); // Use BorderLayout for full-screen layout  
 String imagePath = "src/images/home.jpeg"; // Change to the correct path of your image  
 ImageIcon backgroundImage = new ImageIcon(imagePath);  
 // Scale the image to fit the entire frame  
 Image scaledImage = backgroundImage.getImage().getScaledInstance(1100, 700, Image.*SCALE\_SMOOTH*);  
 ImageIcon scaledIcon = new ImageIcon(scaledImage);  
 // Create a JLabel for the scaled image  
 JLabel backgroundLabel = new JLabel(scaledIcon);  
 backgroundLabel.setLayout(new GridBagLayout()); //to center text  
 // Create a JLabel for the welcome text  
 JLabel welcomeText = new JLabel("Welcome to My Restaurant");  
 welcomeText.setForeground(new Color(245, 245, 220)); // Set text color to black  
 welcomeText.setFont(new Font("Calisto MT", Font.*BOLD*, 80)); // Set the font and size of the text  
 // Add the welcome text to the background label (centered)  
 backgroundLabel.add(welcomeText, new GridBagConstraints());  
 // Add the background label (with the text) to the panel  
 homePanel.add(backgroundLabel, BorderLayout.*CENTER*);  
 return homePanel;  
 }  
 //Table Reservation panel  
 private static JPanel createReservationPanel() {  
 JPanel reservationPanel = new JPanel();  
 reservationPanel.setLayout(null);  
 reservationPanel.setBackground(new Color(210, 210, 210));  
 // Array to hold table buttons  
 JButton[] tableButtons = new JButton[7];  
 Color[] tableStates = {new Color(34, 139, 34), Color.*RED*, Color.*YELLOW*}; // States: Empty, Occupied, Reserved  
 String[] statuses = {": Empty", ": Occupied", ": Reserved"};  
 //7 table buttons  
 for (int i = 0; i < 7; i++) {  
 JButton tableButton = new JButton("Table " + (i + 1) + statuses[0]);  
 tableButton.setBackground(tableStates[0]);  
 tableButton.setOpaque(true);  
 tableButton.setBorderPainted(false);  
 tableButton.setForeground(Color.*BLACK*);  
 tableButton.setFont(new Font("Arial", Font.*BOLD*, 16));  
 // Position each table button based on its index  
 switch (i) {  
 case 0 -> tableButton.setBounds(50, 50, 300, 150);  
 case 1 -> tableButton.setBounds(400, 50, 300, 150);  
 case 2 -> tableButton.setBounds(750, 50, 300, 150);  
 case 3 -> tableButton.setBounds(100, 250, 200, 350);  
 case 4 -> tableButton.setBounds(380, 250, 320, 150);  
 case 5 -> tableButton.setBounds(380, 450, 320, 150);  
 case 6 -> tableButton.setBounds(780, 250, 200, 350);  
 }  
 int finalI = i;  
 tableButton.addActionListener(new ActionListener() {  
 int currentState = 0;  
 @Override  
 public void actionPerformed(ActionEvent e) {  
 currentState = (currentState + 1) % 3;  
 tableButton.setBackground(tableStates[currentState]);  
 tableButton.setText("Table " + (finalI + 1) + statuses[currentState]);  
 }  
 });  
 tableButtons[i] = tableButton;  
 reservationPanel.add(tableButton);  
 }  
 return reservationPanel;  
 }  
 // Order Food panel  
 private static JPanel createOrderPanel(JTabbedPane tabbedPane) {  
 JPanel orderPanel = new JPanel(new BorderLayout());  
 orderPanel.setBackground(new Color(210, 210, 210));  
 // panel on the right for order summary  
 JPanel orderSummaryPanel = new JPanel();  
 orderSummaryPanel.setLayout(new BorderLayout());  
 orderSummaryPanel.setPreferredSize(new Dimension(200, 700));  
 orderSummaryPanel.setBackground(new Color(230, 230, 230));  
 JLabel summaryLabel = new JLabel("Order Summary:");  
 summaryLabel.setFont(new Font("Arial", Font.*BOLD*, 16));  
 summaryLabel.setHorizontalAlignment(SwingConstants.*CENTER*);  
 orderSummaryPanel.add(summaryLabel, BorderLayout.*NORTH*);  
 // Text area to show order summary  
 JTextArea orderSummaryArea = new JTextArea();  
 orderSummaryArea.setEditable(false);  
 orderSummaryPanel.add(new JScrollPane(orderSummaryArea), BorderLayout.*CENTER*);  
 // Center panel to display food items  
 JPanel foodItemsPanel = new JPanel(new GridLayout(4, 3, 10, 10));  
 foodItemsPanel.setBackground(new Color(210, 210, 210));  
 // Create a combo box to select table number  
 JPanel selectTablePanel = new JPanel();  
 String[] tableNumbers = {"Table 1", "Table 2", "Table 3", "Table 4", "Table 5", "Table 6", "Table 7"};  
 JComboBox<String> tableComboBox = new JComboBox<>(tableNumbers);  
 selectTablePanel.add(new JLabel("Select Table: "));  
 selectTablePanel.add(tableComboBox);  
 orderPanel.add(selectTablePanel, BorderLayout.*NORTH*);  
 // Food items data (name, price, images)  
 String[] foodNames = {"Pizza", "Pasta", "Spaghetti", "Salad", "Coffee", "Nachos", "Chicken Wings", "Tres Cake", "Ice Tea", "Chocolate Cake", "Water", "French Fries"};  
 int[] foodPrices = {200, 150, 180, 150, 80, 160, 250, 60, 160, 60, 30, 140};  
 String[] foodImages = {"pizza.jpeg", "pasta.jpeg", "spaghetti.jpeg", "salad.jpeg", "coffee.jpeg", "nachos.jpeg", "chickenwings.jpeg", "trescake.jpeg", "icetea.jpeg", "chocolatecake.jpeg", "water.jpeg", "frenchfries.jpeg"};  
 // Create individual food items with images, labels, and quantity control  
 for (int i = 0; i < foodNames.length; i++) {  
 JPanel foodItemPanel = new JPanel();  
 foodItemPanel.setLayout(new BoxLayout(foodItemPanel, BoxLayout.*Y\_AXIS*));  
 foodItemPanel.setBackground(new Color(230, 230, 230));  
 // Add image placeholder  
 String imagePath = "src/images/" + foodImages[i]; // Adjust path as necessary  
 JLabel foodImage = new JLabel(new ImageIcon(imagePath));  
 foodImage.setPreferredSize(new Dimension(200, 200));  
 foodItemPanel.add(foodImage);  
 // Add food name and price  
 JLabel foodLabel = new JLabel(foodNames[i] + " - ₹" + foodPrices[i]);  
 foodLabel.setAlignmentX(Component.*LEFT\_ALIGNMENT*);  
 foodLabel.setFont(new Font("Arial", Font.*BOLD*, 14));  
 foodItemPanel.add(foodLabel);  
 // Quantity controls  
 JPanel quantityPanel = new JPanel(new FlowLayout());  
 JButton reduceButton = new JButton("-");  
 JButton addButton = new JButton("+");  
 JLabel quantityLabel = new JLabel("0");  
 quantityLabel.setFont(new Font("Arial", Font.*BOLD*, 14));  
 quantityPanel.add(reduceButton);  
 quantityPanel.add(quantityLabel);  
 quantityPanel.add(addButton);  
 foodItemPanel.add(quantityPanel);  
 final int foodIndex = i;  
 addButton.addActionListener(e -> {  
 int tableIndex = tableComboBox.getSelectedIndex();  
 *tableOrders*[tableIndex][foodIndex]++;  
 quantityLabel.setText(String.*valueOf*(*tableOrders*[tableIndex][foodIndex]));  
 *updateOrder*(tableComboBox, foodNames, foodPrices, orderSummaryArea);  
 });  
 reduceButton.addActionListener(e -> {  
 int tableIndex = tableComboBox.getSelectedIndex();  
 if (*tableOrders*[tableIndex][foodIndex] > 0) {  
 *tableOrders*[tableIndex][foodIndex]--;  
 quantityLabel.setText(String.*valueOf*(*tableOrders*[tableIndex][foodIndex]));  
 *updateOrder*(tableComboBox, foodNames, foodPrices, orderSummaryArea);  
 }  
 });  
 foodItemsPanel.add(foodItemPanel);  
 }  
 // Add the food items panel to the center of the order panel  
 orderPanel.add(new JScrollPane(foodItemsPanel), BorderLayout.*CENTER*);  
 // Add order summary panel to the right of the order panel  
 orderPanel.add(orderSummaryPanel, BorderLayout.*EAST*);  
 // Add action to place order  
 JButton placeOrderButton = new JButton("Place Order");  
 placeOrderButton.addActionListener(e -> {  
 String selectedTable = (String) tableComboBox.getSelectedItem();  
 int tableNumber = Integer.*parseInt*(selectedTable.split(" ")[1]) - 1; // to get table number  
 *placeOrderToDatabase*(tableNumber); // Save the order to the database  
 JOptionPane.*showMessageDialog*(null, "Order placed successfully for " + selectedTable + "!");  
 });  
 orderPanel.add(placeOrderButton, BorderLayout.*SOUTH*);  
 return orderPanel;  
 }  
 // Method to update the order summary  
 private static void updateOrder(JComboBox<String> tableComboBox, String[] foodNames, int[] foodPrices, JTextArea orderSummaryArea) {  
 int tableIndex = tableComboBox.getSelectedIndex();  
 StringBuilder summary = new StringBuilder("Order Summary for " + tableComboBox.getSelectedItem() + ":\n");  
 int total = 0;  
 for (int i = 0; i < foodNames.length; i++) {  
 if (*tableOrders*[tableIndex][i] > 0) {  
 summary.append(foodNames[i]).append(": ").append(*tableOrders*[tableIndex][i]).append(" x ₹").append(foodPrices[i]).append("\n");  
 total += *tableOrders*[tableIndex][i] \* foodPrices[i];  
 }  
 }  
 summary.append("Total: ₹").append(total);  
 orderSummaryArea.setText(summary.toString());  
 }  
 // Method to place the order in the database  
 private static void placeOrderToDatabase(int tableNumber) {  
 try (Connection conn = DriverManager.*getConnection*(*DB\_URL*, *USER*, *PASSWORD*);  
 PreparedStatement pstmt = conn.prepareStatement("INSERT INTO orders (table\_number, food\_item, quantity) VALUES (?, ?, ?)")) {  
 // Loop through the table orders and save each non-zero order  
 for (int i = 0; i < *tableOrders*[tableNumber].length; i++) {  
 if (*tableOrders*[tableNumber][i] > 0) {  
 pstmt.setInt(1, tableNumber + 1); // Table number (1-indexed)  
 pstmt.setString(2, *getFoodNameById*(i)); // Food item  
 pstmt.setInt(3, *tableOrders*[tableNumber][i]); // Quantity  
 pstmt.addBatch();

**CHAPTER 6**

**RESULTS AND DISCUSSION**

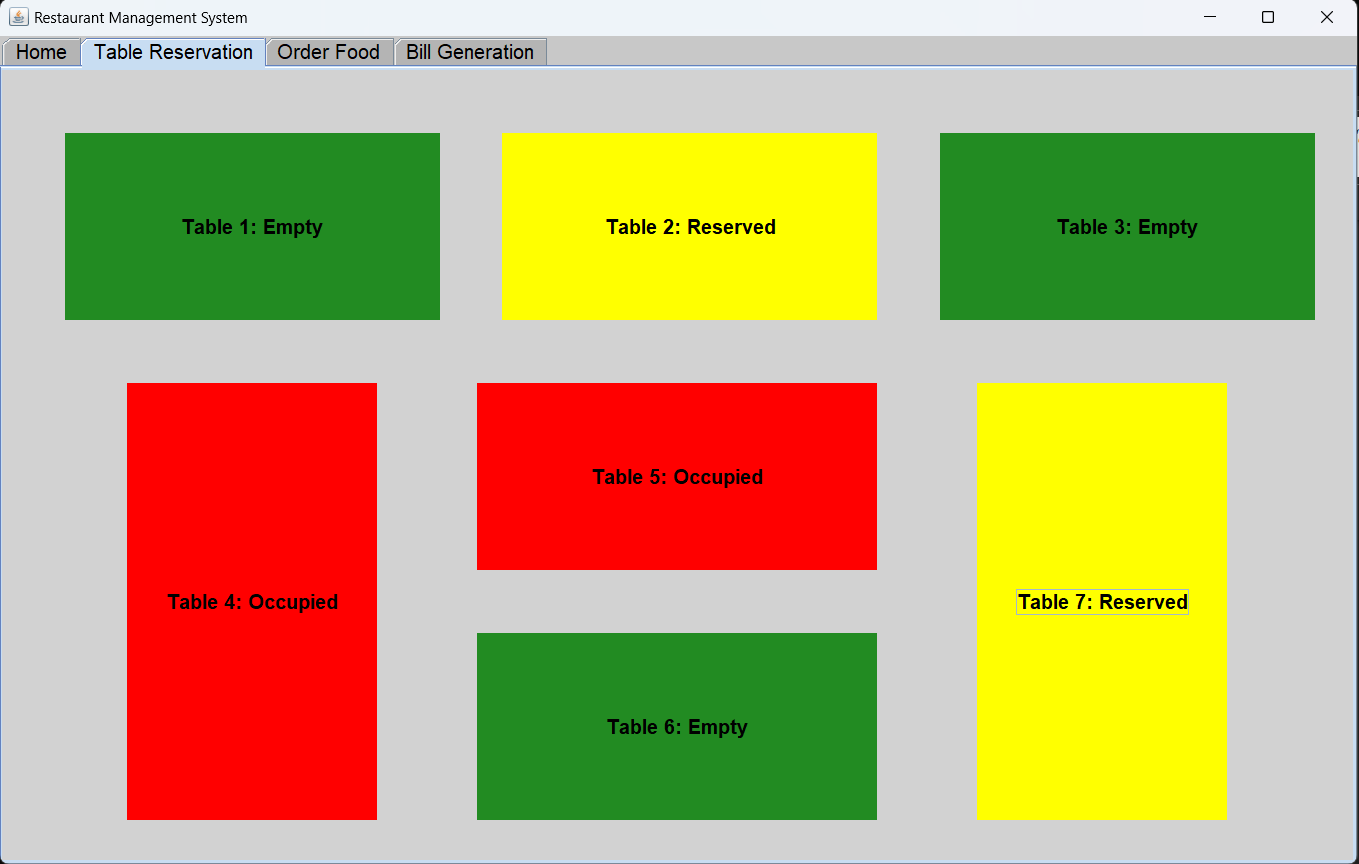


**Figure 6.1 Home Page**

This is the welcome screen of a Java-based Restaurant Management System, designed to provide an intuitive and visually appealing interface for restaurant staff and management. The screen greets users with a large, welcoming message, "Welcome to My Restaurant," displayed over a high-quality background image of a modern restaurant interior, setting a warm and inviting atmosphere.

At the top of the screen, a navigation bar offers essential functionalities: Home, Table Reservation, Order Food, and Bill Generation. Each tab represents a core feature of the system, allowing users to seamlessly switch between tasks. The Table Reservation tab enables staff to manage reservations, the Order Food tab allows quick access to order management, and the Bill Generation tab provides an option to generate and view customer bills.

This interface exemplifies simplicity and functionality, making it easy for staff to perform their tasks efficiently while maintaining a visually appealing design that enhances the user experience. The welcoming screen reflects the user-friendly and organized nature of the entire system, designed to streamline restaurant operations.

****

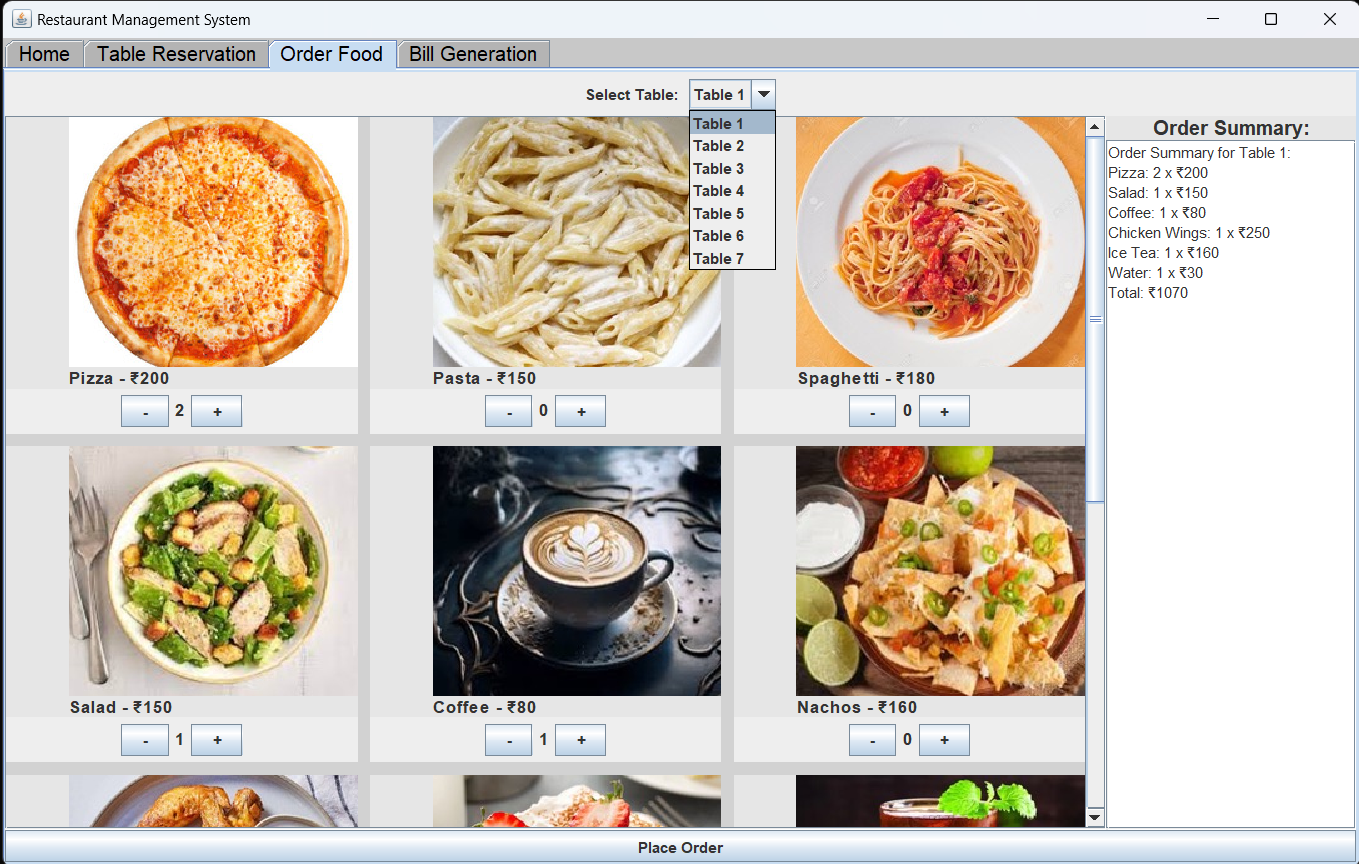
**Figure 6.2 Table Reservation**

This screen is an essential component of the **Table Reservation** feature in the Restaurant Management System, offering a visually clear and organized way for restaurant staff to manage table availability. The screen displays a **color-coded layout**, where each table is represented by a block of color with its status clearly labeled in text:

* **Green**: Indicates that tables are currently **empty** and available for new customers.
* **Red**: Denotes tables that are **occupied** by customers.
* **Yellow**: Signifies **reserved** tables, indicating they are booked but not yet occupied.

This color-coding system provides an at-a-glance overview of table statuses, allowing staff to assess and update seating arrangements in real-time. With this intuitive layout, staff can efficiently assign tables to incoming customers, manage reservations, and plan seating for optimal usage.

The simplicity of this interface, combined with its functional color-coded indicators, helps staff maintain smooth table turnover and ensures a well-organized seating process. This approach enhances the overall dining experience by minimizing wait times and creating a seamless flow in a busy restaurant environment. The screen reflects a user-friendly design that improves operational efficiency and customer satisfaction, making it easier for staff to navigate high-traffic times and provide attentive service to all guests.

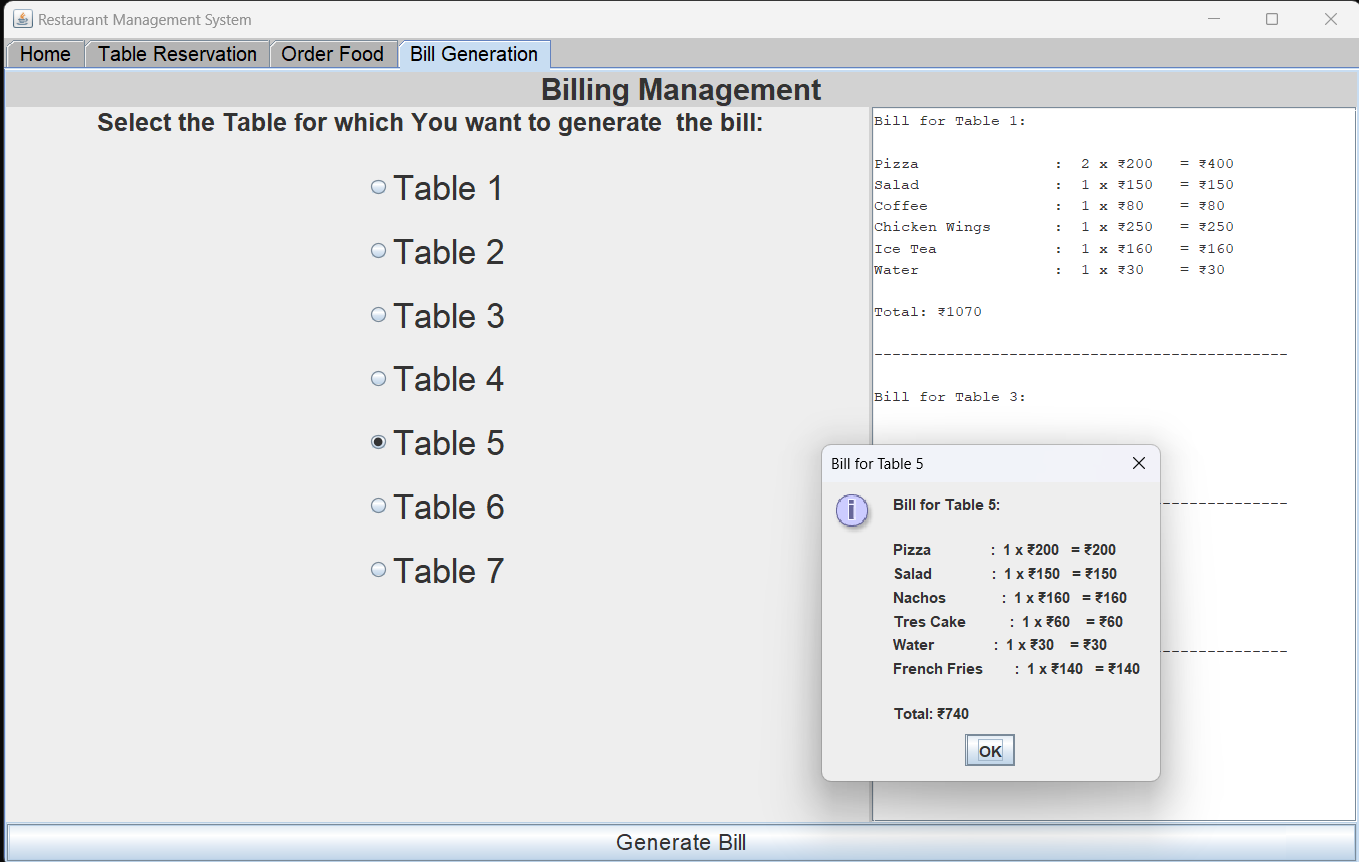


**Figure 6.3 Order Food**

This screen is a core element of the **Order Food** feature in the Restaurant Management System, designed to streamline and simplify the food ordering process for both staff and customers. The interface showcases a **menu** with images, descriptions, and prices of each food item, allowing users to browse the options visually. Each item has a quantity selector with "+" and "-" buttons, enabling easy customization of orders by adjusting the quantity as needed.

At the top of the screen, a **table selection dropdown** allows staff to assign the order to a specific table, ensuring that orders are accurately tracked. On the right side, an **Order Summary** panel provides a real-time update of the order, listing each selected item, its quantity, and the associated price. The summary also displays a **total bill amount** at the bottom, giving staff a clear view of the table’s order details.

This organized, visually engaging layout enables staff to place orders quickly and accurately, minimizing errors and reducing service time. The intuitive design enhances the restaurant’s workflow, allowing for a seamless order-taking process that improves service efficiency and ensures customer satisfaction. The interface contributes to a more enjoyable dining experience by simplifying the order placement process, which in turn helps maintain a steady and well-coordinated service flow in the restaurant.

****

**Figure 6.4 Bill Generation**

This screen is part of the **Bill Generation** feature in the Restaurant Management System, designed to streamline billing and provide a comprehensive view of past transactions. The interface includes a **table selection dropdown** at the top, allowing staff to select a specific table to generate or view the bill for its recent orders. This makes it easy to handle bills for each table independently and accurately.

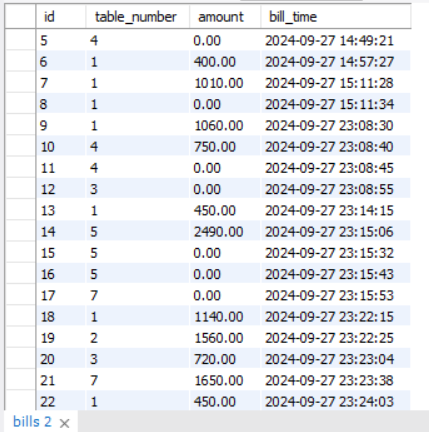
On the right side of the screen, there is a **Billing Panel** that displays all previous bills. Each bill entry shows essential details, such as the items ordered, quantities, prices, and the total amount for that table. This organized record of past bills provides staff with a convenient reference to review previous transactions, manage billing discrepancies, or answer customer inquiries about their charges.

This user-friendly layout simplifies the billing process, ensuring accuracy in transaction records and enabling quick access to a table’s billing history. The clear separation of current and past bills allows staff to manage both active and completed transactions efficiently, enhancing the restaurant’s operational flow and contributing to a seamless checkout experience for customers. This feature promotes accuracy, accountability, and transparency in billing, which improves customer satisfaction and supports effective management of the restaurant's finances.

MYSQL DATABASE:

****

**Figure 6.5 and 6.6 Database Tables**

****

**Figure 6.7 Database Table**

**CHAPTER 7**

**CONCLUSION**

The *Restaurant Management System* is a transformative solution designed to address critical operational challenges faced by restaurants. By automating reservations, menu management, and billing processes, this system effectively enhances both efficiency and accuracy, enabling restaurants to meet the high expectations of today’s customers. Built with Java Swing for a user-friendly interface and MySQL for a reliable backend, this system is easy for staff to use while providing the stability and scalability required to support a growing business.

Throughout the project, we aimed to create a system that not only simplifies day-to-day restaurant operations but also offers a foundation for future growth. Key features, such as real-time reservation tracking, dynamic menu updates, and streamlined billing, allow restaurants to deliver a seamless dining experience and significantly reduce manual errors. Furthermore, the system’s scalable architecture enables easy integration of advanced features, such as digital payments and analytics, supporting data-driven decision-making that can optimize business performance over time.

This system positions restaurants to operate more efficiently, delivering faster, more accurate service while fostering an improved customer experience. Future improvements, like customer preference tracking and sales analytics, will provide deeper insights into customer behaviour, allowing restaurants to make informed decisions that drive loyalty and profitability.

In conclusion, the *Restaurant Management System* is a modern, adaptable solution that not only meets current industry needs but also has the potential to evolve with the demands of an increasingly digital and customer-focused market, making it a valuable asset for any restaurant aiming for long-term success and competitiveness.

CHAPTER 8

**REFERENCES**

**1. Oracle Corporation**. (2015). *Java Platform, Standard Edition 8 Documentation*. Oracle Corporation. Retrieved from <https://docs.oracle.com/javase/8/docs/>

**2. MySQL AB**. (2023). *MySQL 8.0 Reference Manual*. Oracle Corporation. Retrieved from <https://dev.mysql.com/doc/refman/8.0/en/>

**3. Morrison, J. D., & Raju, R.** (2018). *Designing and Developing Restaurant Management Systems Using Relational Databases*. *International Journal of Computer Applications*, 181(12), 15-20.

**4. Java Database Connectivity (JDBC) Guide**. (2023). Oracle Corporation. Retrieved from <https://docs.oracle.com/javase/8/docs/technotes/guides/jdbc/>

**5. Java Swing Tutorial**. (2023). Oracle Corporation. Retrieved from <https://docs.oracle.com/javase/tutorial/uiswing/>

6. **Eck, D. J.** (2014). *Introduction to Programming Using Java, Sixth Edition*. Hobart and William Smith Colleges. Retrieved from http://math.hws.edu/javanotes/.

7. **Java Database Connectivity (JDBC) Guide**. (2023). Oracle Corporation. Retrieved from <https://docs.oracle.com/javase/8/docs/technotes/guides/jdbc/>