VISVESVARAYA TECHNOLOGICAL UNIVERSITY

"JnanaSangama", Machhe, Belagavi, Karnataka -590018

A Mini Project Report
On

"RESTAURANT MANAGEMENT SYSTEM USING B+ TREE"

Submitted in partial fulfillment towards File Structures Laboratory with Mini Project(18ISL67) of VI Semester

Bachelor of Engineering
In
Information Science and Engineering

Submitted by

Ananya B Dalapathi	**[4GW20IS002]**
Megha K Prasad	**[4GW20IS029]**
Nagaveni S	**[4GW20IS032]**

Under the Guidance of

Mrs. Chaya P
Assistant Professor
Department of Information Science and Engineering
GSSSIETW, Mysuru

DEPARTMENT OF INFORMATION SCIENCE AND ENGINEERING
(Accredited by NBA, New Delhi, Validity 01.07.2017 to 30.06.2020 & 01.07.2020 to 30.06.2023)

GSSS INSTITUTE OF ENGINEERING & TECHNOLOGY FOR WOMEN
(Affiliated to VTU, Belagavi, Approved by AICTE, New Delhi & Govt. of Karnataka)
K.R.S ROAD, METAGALLI, MYSURU-570016, KARNATAKA
Accredited with Grade 'A' by NAAC

2022-2023

Geetha Shishu Shikshana Sangha (R)

GSSS INSTITUTE OF ENGINEERING & TECHNOLOGY FOR WOMEN

(Affiliated to VTU, Belagavi, Approved by AICTE -New Delhi & Govt. of Karnataka)

K.R.S Road, Metagalli, Mysuru, Karnataka-570016

Accredited with Grade 'A' by NAAC

DEPARTMENT OF INFORMATION SCIENCE AND ENGINEERING

(Accredited by NBA, New Delhi, Validity 01.07.2020 to 30.06.2023)

CERTIFICATE

Certified that the Mini project work entitled **"RESTAURANT MANAGEMENT SYSTEM USING B+ TREE"** is a bonafide work carried out by **Ananya B Dalapathi [4GW20IS002], Megha K Prasad [4GW20IS029], Nagaveni S [4GW20IS032]** in partial fulfillment for the award of degree of **Bachelor of Engineering in Information Science and Engineering** of the Visvesvaraya Technological University, Belagavi during the year 2022-23. The File Structure with Mini project report has been approved as it satisfies the academic requirements with respect to the Mini Project Work prescribed for Bachelor of Engineering Degree.

Signature of the Guide	**Signature of the HOD**
Mrs. Chaya P	**Dr. Gururaj K S**
Assistant Professor	**Professor and Head**

External Viva

Name of the Examiners	**Signature with Date**
1.	
2.	

ACKNOWLEDGEMENT

The joy and satisfaction that accompany the successful completion of any task would be incomplete without the mentioning the people who made it possible

First and foremost, we offer our sincere phrases of thanks to **Smt. Vanaja B Pandit, Honorary Secretary, GSSS(R) and the Management of GSSSIETW, Mysuru** for permitting us to utilize all the facilities of the Institution.

We would like to express our gratitude to our Principal, **Dr. Shivakumar M** for providing us a congenial environment for engineering studies and also for having showed us the way to carry out the project.

We consider it a privilege and honor to express our sincere thanks to, **Dr. Gururaj K S,** Professor and Head, Department of Information Science and Engineering for his support and valuable guidance throughout the tenure of this project.

We would like to thank our guide **Mrs. Chaya P**, Assistant Professor, Department of Information Science and Engineering for her constant monitoring, guidance & motivation throughout the tenure of this Mini project work.

We would also like to thank all our ISE staff members and non-teaching staff who have always been with us extending their precious suggestions, guidance and encouragement throughout the Mini project.

We intend to thank all the teaching and non-teaching staff's of our Department of Information Science and Engineering for their immense help and co-operation.

Finally, we would like to express our gratitude to our parents and friends who always stood with us to complete this work successfully.

Ananya B Dalapathi
Megha K Prasad
Nagaveni S

ABSTRACT

Restaurant Management System that utilizes B+ trees as a fundamental data structure for efficient storage and retrieval of restaurant-related information. This aims to streamline and automate various aspects of restaurant management, order processing, table management, and inventory control.

The table management module tracks table availability and reservations. B+ trees enable efficient handling of table status updates, ensuring quick identification and allocation of available tables based on customer preferences or reservation status.

The inventory control module assists in monitoring and managing restaurant inventory, including tracking stock levels, placing orders, and generating reports. B+ trees effectively organize inventory data, facilitating rapid inventory lookups and efficient management of stock.

Overall, the utilization of B+ trees as the underlying data structure in the Restaurant Management System enhances performance, scalability, and data management capabilities. It enables efficient handling of menu management, order processing, table management, and inventory control, ultimately optimizing restaurant operations and enhancing the overall dining experience.

CONTENTS

LIST OF FIGURES

LIST OF TABLES

LIST OF SNAPSHOTS

CHAPTER 1

INTRODUCTION

1.1 Overview

A Restaurant management system is a software application designed to help restaurant owners and managers streamline their operations and improve the overall efficiency of their business. The system typically includes features such as table management, order tracking, inventory management and employee management. The benefits of this system are numerous. By automating many of the tasks that are traditionally done manually, such as taking orders, tracking inventory, and managing employee schedules, managers can free up more time to focus on other important aspects of the business like, improving customer service, developing new menu items.

Additionally, restaurant management system can help reduce errors and waste as well as provide valuable insights into customer behavior and preferences. By analyzing data such as sales trends, and customer feedback managers can make more informed decisions about how to improve the business.

Overall, a restaurant management system can be a valuable tool for any restaurant looking to improve its operations, reduce costs, and increase profitability. With the right system in place, restaurant owner can better manage their resources and provide a better dining experience for their customers.

1.2 Existing System

It can be difficult for many restaurants to manage tasks like customer table reservations, and inventory control. If manual booking is utilized, the client may cancel a reservation for a table after making a reservation, and it would be difficult if staff members might lose the customer information since it is difficult to preserve accurate customer information. The existing system doesn’t contain proper methods for inventory management and menu planning.

1.3 Proposed System

The restaurant management using B+ trees is a software application designed to help restaurant owners and managers streamline their operations. The system utilizes B+ trees, a specialized data structure that can handle large amounts of data while maintaining fast access times, to organize and manage data. The system provides restaurant’s operations like menu planning,

table management and inventory tracking. By providing a centralized platform for managing all these tasks, the system can help restaurant owners and managers save time and reduce errors.

Overall, the restaurant management system using B+ trees is a robust and scalable platform for managing restaurant operations. By leveraging the power of B+ trees, restaurant owners and managers can make better decisions, improve the efficiency of their operations and ultimately increase their profitability.

1.4 Objective

The goal of the project is to develop web-based application on the restaurant management system which helps in managing day to day operations of a restaurant.

This system helps in achieving the following objectives:

- To keep the software user friendly.
- To ensure effective restaurant operations.
- Provide operations like menu planning, table management and inventory tracking.
- Provide easy billing process.

1.5 Organization of the Report

The project is organized as follows:

Chapter 1: Includes the introduction of the project, that is Overview, objectives, existing system and proposed system.

Chapter 2: Focus on the literature survey of the project. In order to understand the project in a better way, a survey was done to know about existing systems.

Chapter 3: Includes the requirement specification required for this project and software and hardware requirements along with the design diagrams of the project.

Chapter 4: Discuss the implementation of the project along with code snippets.

Chapter 5: Gives the brief description of software testing by test cases.

Chapter 6: Discuss the result of the project followed by conclusion and future enhancement.

CHAPTER 2

LITERATURE SURVEY

2.1 Related Work

[1] "Implementation of Smart Restaurant with e-menu card" by Mayur D. Jakhete and Piyush C. Mankar (June 2015).

The authors proposed the implementation of smart restaurant using Model View Controller Architecture which is used to support rapid web application development and dynamic interactivity with the database. Following iterative RUP development cycle it was easy to test the iterative increments of the software. The behavior of the system was successfully tested using black box testing. But the only drawback is that there is no efficient ordering system in the project**.**

[2] "Implementing a Web-Based Computerized Restaurant System" by Chin Loong Tan (2013).

The author proposed the implementation of computerized web-based restaurant system which is associated with point-of-sales system, a terminal that is used to process sales transaction. This system utilized Web Performance Testing and load Testing tools provided by VS to execute performance testing. However, this system does not provide inventory management and pricing methods for multiple recipes.

[3] "Online Restaurant Management System" by Nur Yasmin Binti Mohd Nasir (2022).

The author proposed an online restaurant management system which used agile software development methodology. This project was associated with Scrum and Kanban from which the tasks were tracked and issues were detected. The system functionalities were tested successfully against the requirements. The limitation in this was this system was not secure to use.

Table 2.1: Related work

Sl. No	Title	Author	Reference	Description	Limitations	Year
1	A Review of Restaurant Management System and their Implementation	Karim Allaham and Mahmoud Al-Qutayri	International Journal of Computer Applications, vol.88, no. 11, pp. 12-18, 2014	This paper presents a review of existing restaurant management system and their implementation. The review includes an overview of system architecture, features and limitations. The paper highlights the need for more scalable and customizable systems that can be tailored to meet the specific needs of a restaurant.	The paper does not provide a detailed comparison of the system reviewed and their respective strength and weaknesses.	2014
2	A Survey of Restaurant Management System	Kiran Kumar A, Jyothi B, and Basavaraj S. Anami	International Journal of Innovative Research in Computer and Communication Engineering, vol. 3, no. 10, pp. 10050-10056, 2015	This paper presents a survey of restaurant management system, including their features, advantages and limitations. The system reviewed include NCR Aloha, Micros, and POSist. The paper highlights the need for more integrated and customizable systems that can be tailored to meet the specific needs of a restaurant.	The paper does not provide a detailed evaluation of the system's security or data privacy.	2015

3	An Intelligent Restaurant Management System using Rule based Reasoning	Ahmed Zaki, Ahmed AL Zahrani, and Ahmed Alqahtani	International Journal of Engineering and Technology, vol. 5, no. 1, pp. 46-50, 2013	This paper presents an intelligent restaurant management system that uses rule-based reasoning. The system includes features such as customer ordering and payment, inventory management, and employee scheduling. The system also includes a recommendation engine that suggests menu items based on customer preferences.	The paper does not provide a detailed evaluation of the system's scalability customization.	2013
4	A Cloud-Based Restaurant Management System	Ahmed Zaki, Ahmed Alzahrani, and Ahmed Alqahtani	International Journal of Computer Science and Information Security, vol. 15, no. 3, pp. 12-16, 2017	This paper presents a cloud-based restaurant management system. The system includes features such as customer ordering and payment, inventory management, and employee scheduling. The system is designed to be a scalable and customizable and can be accessed from any device with an internet connection.	The paper does not provide a detailed evaluation of system's performance or security.	2017

CHAPTER 3

SYSTEM REQUIREMENT AND DESIGN

Software Requirements specification is the starting point of the software development activity. It includes an introduction that gives the purpose, scope and an overview of the system.

3.1 REQUIREMENTS

3.1.1 Functional Requirements

A functional requirement contains the description of the staff and the customer who are involved in restaurant management system operations. This project has three modules Manager, Staff and Customer. Functional requirements capture the intended behavior of the system. Functional requirements describe the inertia between the system and its implementation.

- **Manager**

The functional requirements of this module are:

- It includes authenticating the restaurant employees.
- Inputs include username and password.
- After successful login he can manage employee details, inventory, reservations and view bills.

- **Staff**

The functional requirements needed under this module are:

- Inputs include username and password.
- Username and passwords are checked to ensure valid employee.
- The employee here can take orders and manage the payment of the customers.

- **Customer**

The functional requirements needed under this module are:

- The system allows the customer to make a table reservation.
- The system also allows the customer to order the food and make an easy payment.

3.1.2 Non-Functional Requirements

Non-functional requirements are the constrains on the services or functions offered by the system such as timing constraints, constraints on development process or standards, reliability, response time and storage requirements.

USABILITY: The system must be able to make reservations as soon as the customer books it.

RELIABILITY: The system should be able to maintain the details of all the customer reservations and menu items etc.…

SECURITY: The system must be user friendly and trustable.

AVAILABILITY: The system is available for all the users.

3.2 SYSTEM REQUIREMENT

3.2.1 Software Requirements

Software requirements deals with defining software resources requirements and prerequisites that needs to be installed on a computer to provide optimal functioning of an application.

- Operating System : Windows 7 and above
- Coding Language : Python, Flask
- Back-end : Files
- Tools : Visual Studio Code

3.2.2 Hardware Requirements

Hardware requirements are the most common set of requirements defined by any operating system or software application.

- Processor : i3 or above
- Speed : 1.1GHz
- RAM : 4GB or above
- Hard Disk : 80GB

3.3 SYSTEM DESIGN

The purpose of the design phase is to plan a solution of the problem specified by the requirements document. This phase is the first step in moving from the problem domain to the solution domain. In other words, starting with what is needed; design takes us toward how to satisfy the needs. The design of a system is perhaps the most critical factor affecting the quality of the software; it has major impact on the later phases particularly testing and maintenance.

3.3.1 Use Case Diagram

The representation of characters, arrangement of the use cases are enclosed in framework limit, also the correspondence relationship between use cases and performing artist in a pictorial representation is the use case diagram. The connection of framework with external entities is depicted in the use case diagram. Each use cases talks about the usefulness that framework provides its clients. To distinguish and partition the framework amid the examination process the use cases are utilized very much. The frame work here is isolated into individual and according to their roles.

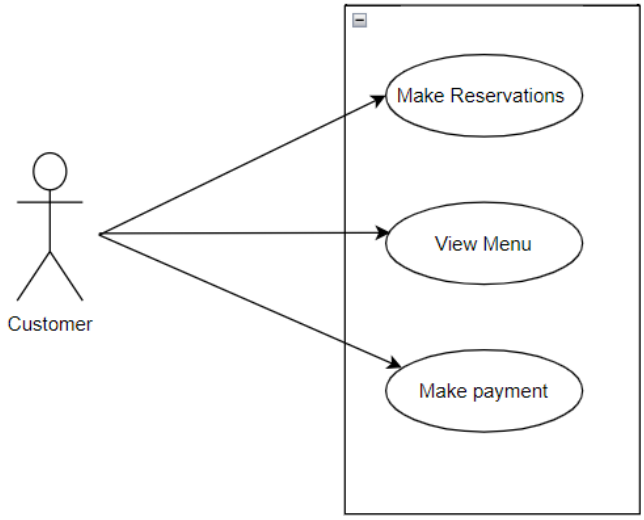


Figure 3.1: Use case diagram of Customer

Figure 3.1, shows the functions performed by Customer that is the customers can make reservations, view menu and they can make the payment for their orders.

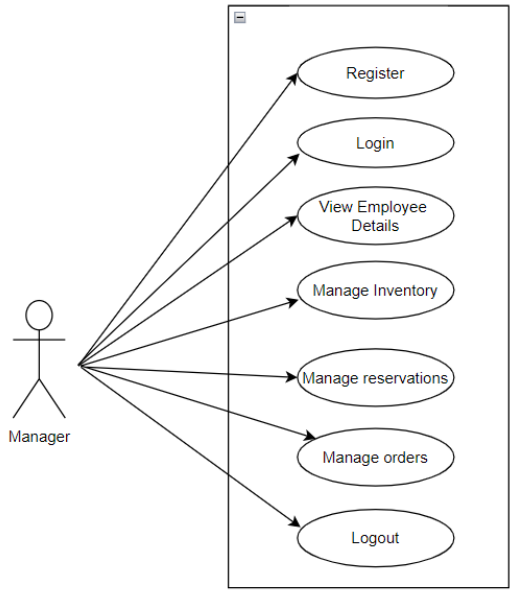


Figure 3.2: Use Case Diagram of Manager

Figure 3.2, shows the Manager module. Here we can see the functions of Manager. The manager has to create an account and register himself and later can login using the required credentials. After the successful login, he can view employee details, manage inventory, he can also see the orders of customers and also manage the reservations and Logout.

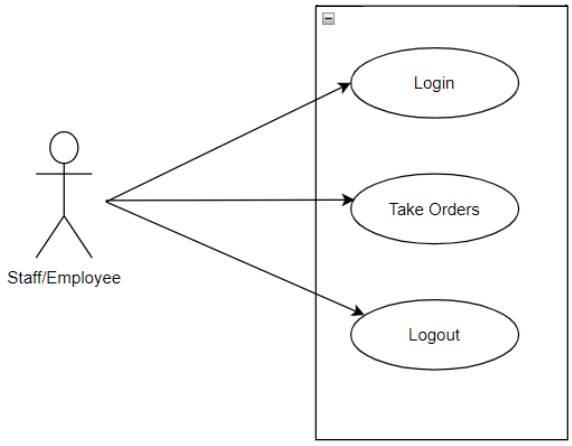


Fig 3.3: Use Case Diagram of Employee

Figure 3.3, shows employee module. Here we can see the functions of Employee. The employee has to create an account and register himself and later can login using the required credentials. After the successful login, the employee can take orders of customer and Logout.

3.3.2 Sequence Diagram

Figure 3.4 shows the interaction of the module in the application are represented in sequence diagram They are also organized as instances in diagram. Sequence diagram is also known as event situation or even graphs. The exchange of data is shown in sequence diagram. In the view of the framework UML (Unified Modelling Language) grouping graphs are very much valuable because they give an active view point.

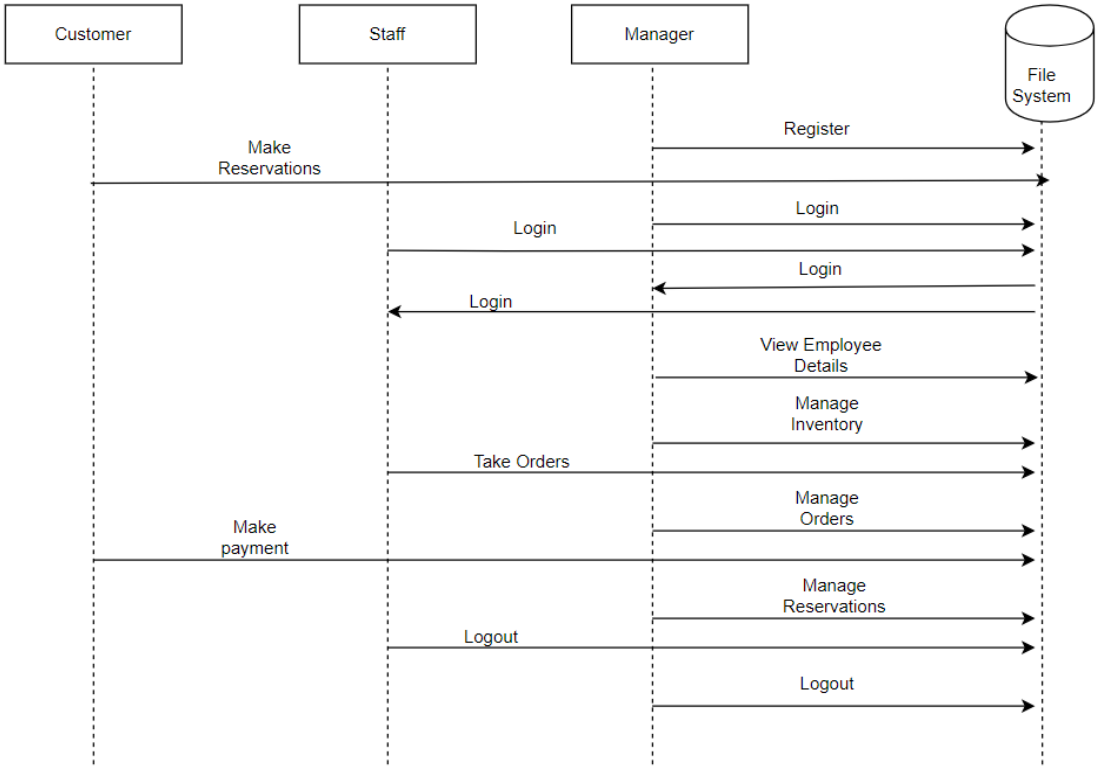


Figure 3.4: Sequence diagram of Restaurant Management System

3.3.3 Flow Chart

A flowchart is a picture of the separate steps of a process in sequential order. It is a generic tool that can be adapted for a wide variety of purposes, and can be used to describe various processes such as a manufacturing process, an administrative or service process or a project plan. It is a graphical aid, designed to visualize the sequence of steps to be followed throughout the project management process. This diagrammatic representation illustrates a solution model to a given problem.

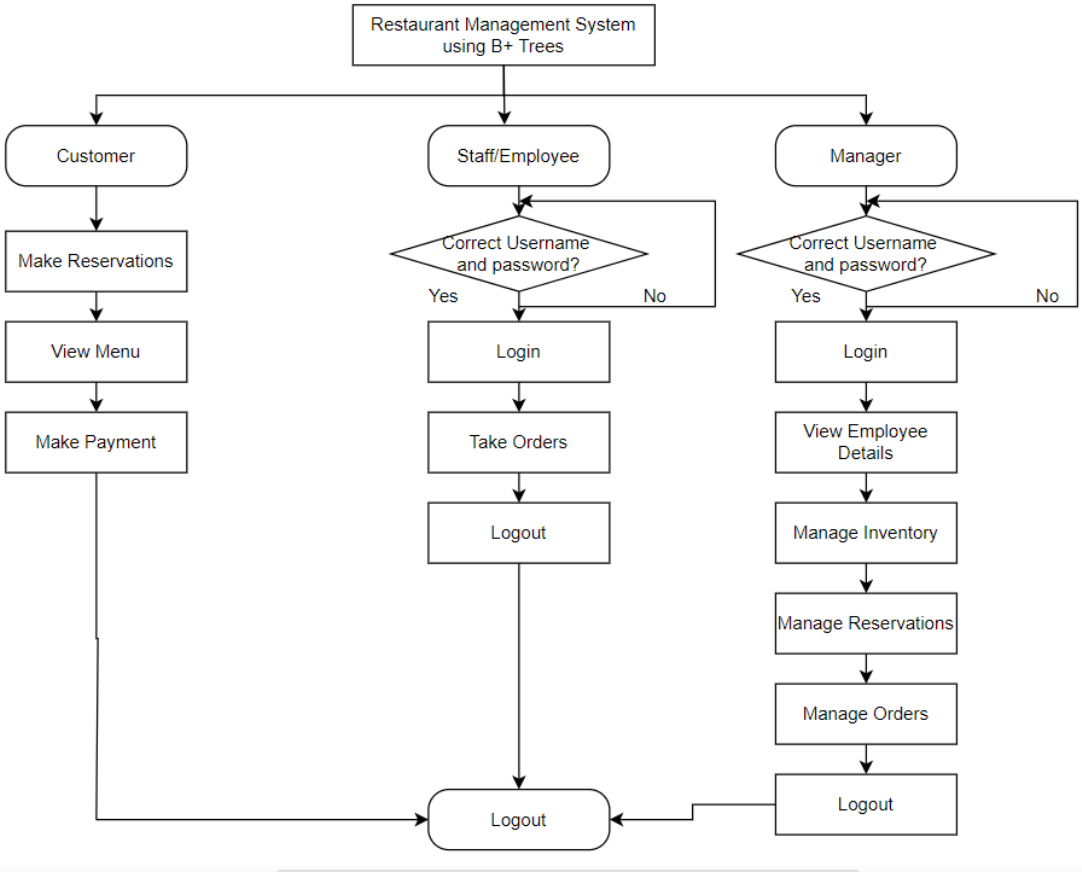


Figure 3.5: Flow chart of Restaurant Management System

Figure 3.5 shows how the restaurant management system works. The Customer book a table, view menu, order food and make payment. Whereas, the staff member can login, take orders and logout. The manager can login and after successful login he can view employee details, manage inventory, manage reservations and orders and logout.

3.3.4 System Architecture

The architecture of a system reflects how the system is used and how it interacts with other systems and the outside world. It describes the interconnection of all the system's components and the data link between them. The architecture of a system reflects the way it is thought about in terms of its structure, functions and relationships. In architecture, the term "System" usually refers to the architecture of the software itself, rather than the physical structure of the buildings or machinery.

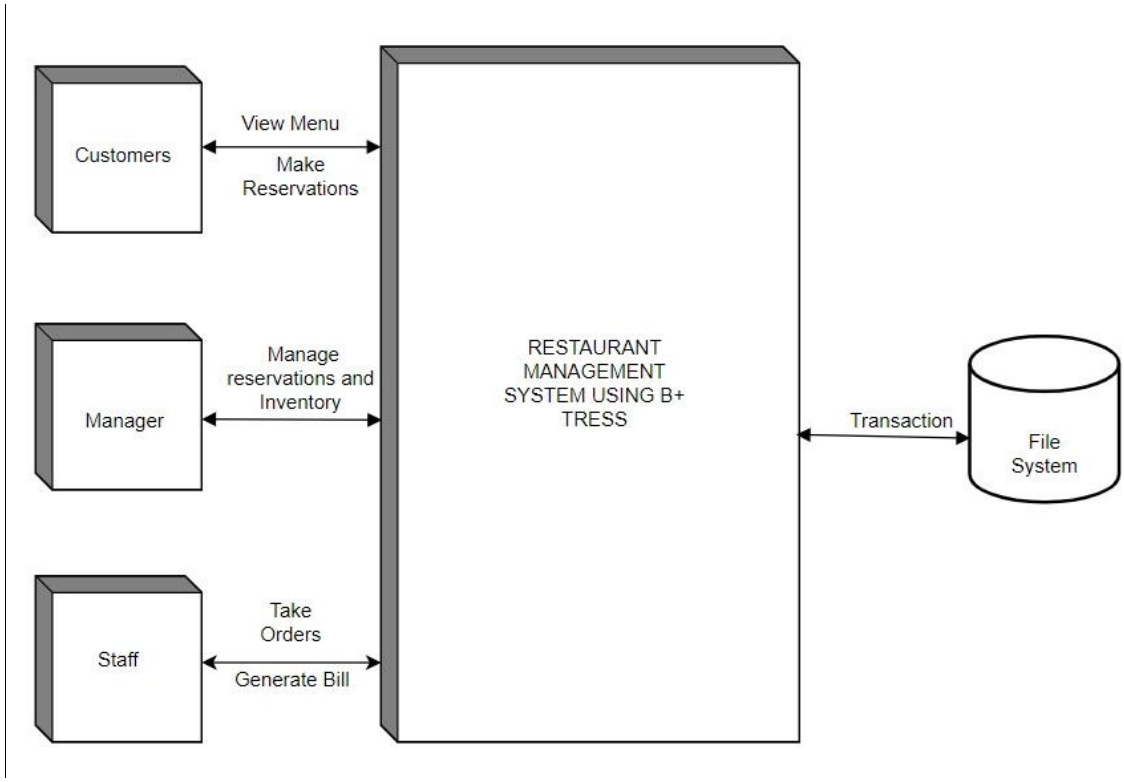


Figure 3.6: System Architecture of Restaurant Management System using B+ Trees

Figure 3.6 shows the system architecture of Restaurant Management System using B+ Trees. It shows how the Customers and Staff Modules are interconnected. In Customer module, the customers can make table reservations, view menu, order food and make payments. In the Manager Module, the manager login by providing required credentials where after a successful login he can manage reservations, inventory, orders and also can view employee details. The Employee logins providing the credentials where he can take orders

CHAPTER 4

IMPLEMENTATION

4.1 TOOLS AND TECHNOLOGY

4.1.1 Python

Python is a computer programming language often used to build websites and software, automate tasks, and conduct data analysis. Python is a general-purpose language, meaning it can be used to create a variety of different programs and isn't specialized for any specific problems. This versatility, along with its beginner-friendliness, has made it one of the most-used programming languages today.

Python is often used to develop the back end of a website or application—the parts that a user doesn't see. Python's role in web development can include sending data to and from servers, processing data and communicating with databases, URL routing, and ensuring security. Python offers several frameworks for web development. Commonly used ones include Django and Flask.

In software development, Python can aid in tasks like build control, bug tracking, and testing. With Python, software developers can automate testing for new products or features. Some Python tools used for software testing include Green.

4.1.2 Flask

Flask is a web application framework written in Python. It was developed by Armin Ronacher, who led a team of international Python enthusiasts called Pocco. Flask is based on the Werkzeg WSGI toolkit and the Jinja2 template engine. Both are Pocco projects.

Flask is often referred to as a micro framework. It is designed to keep the core of the application simple and scalable. Instead of an abstraction layer for database support, Flask supports extensions to add such capabilities to the application.

Flask's status as a micro framework means that you can use it to grow a tech project such as a web app incredibly quickly. If you want to make an app that starts small, but has the potential to grow quickly and in directions you haven't completely worked out yet, then it's an ideal choice. Its simplicity of use and few dependencies enable it to run smoothly even as it scales up and up.

4.2 MODULES WISE CODE

4.2.1 Customers Module

- **Book Table**

```
@app.route('/reservation', methods=['POST'])
def create_reservation():
    name = request.form.get('name')
    email = request.form.get('email')
    datetime_str = request.form.get('Datetime')
    select1 = request.form.get('select1')
    message = request.form.get('message')
    # Convert the selected datetime string to a datetime object
    selected_datetime = datetime.fromisoformat(datetime_str.replace('T', ' '))
    # Calculate the end time for the selected reservation
    end_datetime = selected_datetime + timedelta(hours=2)
    # Check if the maximum limit of reservations within the 2-hour time frame has been
reached
    available_tables = list(range(1, 7))  # List of available table numbers
    for dt in reservation_counter:
        if dt >= selected_datetime and dt < end_datetime:
            if reservation_counter[dt] >= 6:
                # Remove already reserved table numbers from the available_tables list
                reserved_table_numbers = set()
                for reservation in reservation_data.values():
                    if datetime.fromisoformat(reservation['datetime'].replace('T', ' ')) == dt:
                        reserved_table_numbers.add(reservation['table_number'])
                # If all tables are reserved, return a response indicating unavailability
                if len(reserved_table_numbers) == 6
    return '''
                    <script>
                        alert("Reservation is not available for the selected date and time.");
                        window.history.back();
                    </script>
```

```
                '''
                    # Remove the reserved table numbers from the available_tables list
                    available_tables = list(set(available_tables) - reserved_table_numbers)
            # Assign a random table number from the available_tables list
            table_number = random.choice(available_tables)
            # Increment the counter for each reservation slot within the 2-hour time frame
            current_datetime = selected_datetime
            while current_datetime < end_datetime:
                reservation_counter[current_datetime] =
reservation_counter.get(current_datetime, 0) + 1
                current_datetime += timedelta(minutes=30)
            # Create a reservation object
            reservation = {
                'name': name,
                'email': email,
                'datetime': datetime_str,
                'select1': select1,
                'message': message,
                'table_number': table_number
            }
            # Generate a unique reservation ID
            reservation_id = generate_reservation_id()
            # Store the reservation in the reservation data dictionary using the ID as the key
            reservation_data[reservation_id] = reservation

            # Save the reservation data and limits to the JSON files
            save_reservation_data()
            # Redirect to the reservation success page
            return redirect('/reservation_success')
        def load_reservation_data():
            global reservation_data
            if os.path.exists(reservation_file):
                with open(reservation_file, 'r') as f:
                    reservation_data = json.load(f)
            else:
```

```
        reservation_data = {}  # Initialize as an empty dictionary if the file doesn't exist
def save_reservation_data():
    with open(reservation_file, 'w') as f:
        json.dump(reservation_data, f,indent=4)
@app.route('/reservation_success')
def reservation_success():
    return render_template('payment_form.html', reservations=reservation_data)
```

- **Food Order**

```
var selectedItems = [];
var qrCodeInstance;1
var qrCodeGenerated = false;
// Calculate the total amount of selected items
function calculateTotalAmount() {
  var totalAmount = 0;
for (var i = 0; i < selectedItems.length; i++) {

totalAmount += selectedItems[i].price * selectedItems[i].quantity;
  }
  return totalAmount;
}
// Update the items list and total amount on the page
function updateItemList() {
  var itemsList = document.getElementById("itemsList");
  var totalAmount = calculateTotalAmount();
  itemsList.innerHTML = "";
  for (var i = 0; i < selectedItems.length; i++) {
    var item = selectedItems[i];
    var listItem = document.createElement("li");
    listItem.textContent = item.name + " (Quantity: " + item.quantity + ")";
    itemsList.appendChild(listItem);
  }
  document.getElementById("totalAmount").textContent = "Total Amount: ₹" +
totalAmount.toFixed(2);
  if (totalAmount < 0 && qrCodeGenerated) {
```

```
    document.getElementById("generateBillBtn").disabled = true;
  }
}
// Generate QR code based on the selected items
// Generate QR code based on the total amount
function generateQRCode() {
  var totalAmount = calculateTotalAmount();
// Create a QR code instance
var qrCodeInstance = new QRCode(document.getElementById("qrCodeContainer"), {
    width: 128,
    height: 128
  });
  // Generate the QR code for the total amount
  qrCodeInstance.makeCode("Total Amount: ₹" + totalAmount.toFixed(2));
  qrCodeGenerated = true;
}
// Clear the selected items and reset the form
function clearData() {
  selectedItems = [];
  qrCodeGenerated = false;
  updateItemList();
  document.getElementById("tableSelect").value = "";
  document.getElementById("itemSelect").value = "";
  document.getElementById("quantitySelect").value = 1;
  document.getElementById("qrCodeContainer").innerHTML = "";
  //document.getElementById("generateBillBtn").disabled = true;
  document.getElementById("selectionRequiredMsg").style.display = "none";
}
// Handle the "Add Item" button click event
document.getElementById("addItemBtn").addEventListener("click", function() {
  var table = document.getElementById("tableSelect").value;
  var item = document.getElementById("itemSelect").value;
  var quantity = parseInt(document.getElementById("quantitySelect").value);
```

```
if (table !== "" && item !== "" && quantity > 0) {
                    selectedItems.push({ name: item, quantity: quantity, price:
parseFloat(item.match(/₹(\d+)/)[1]) });
                updateItemList();
                generateQRCode();
                document.getElementById("selectionRequiredMsg").style.display = "none";
              } else {
                document.getElementById("selectionRequiredMsg").style.display = "block";
              }
            });
            // Handle the "Generate Bill" button click event
            document.getElementById("generateBillBtn").addEventListener("click",
function() {
              if (totalAmount < 0 ) {
                document.getElementById("generateBillBtn").disabled = true;
              }
              alert("Bill generated!\n Bill Amount: "+ totalAmount.toFixed(2));
            });
            // Handle the "Clear Data" button click event
            document.getElementById("clearDataBtn").addEventListener("click",
clearData);
            // Initialize the QR code generator
            var qrCodeInstance = new QRCode(document.getElementById("qrcode"), {
              width: 128,
              height: 128
            });
          </script>
```

4.2.2 Manager Module

- **Login**

```
<div class="login1">
  <form action="/login" method="POST">
    <label for="employee_id">Employee ID:</label>
    <input type="text" id="employee_id" name="employee_id" required>
```

```
            <br><br>

  <label for="password">Password:</label>
        <input type="password" id="password" name="password" required>
        <br><br>
        <input type="submit" value="Login">
        <p class="message">Not registered? <a href="signup">Create an account</a></p>
    </form>
    </div>
@app.route('/login', methods=['POST'])
def login():
    try:
        employee_id = request.form['employee_id']
        password = request.form['password']
        if os.path.exists(db_file):
            with open(db_file, 'r') as f:
                tree = json.load(f)
            if employee_id in tree and tree[employee_id]['password'] == password:
                session['employee_id'] = employee_id  # Store employee ID in session
                return redirect(url_for('dashboard'))
            else:
                return 'Invalid employee ID or password'
        else:
            return 'No registered employees found'
    except Exception as e:
        return f'An error occurred: {str(e)}'
```

- **Manage Inventory**

```
@app.route('/inventory', methods=['GET', 'POST'])
def inventory():
    if 'employee_id' in session:
        if request.method == 'POST':
            product_name = request.json['name']
            product_quantity = request.json['quantity']
            product_id = len(inventory_data) + 1
```

```
            product = {

     'name': product_name,
                'quantity': product_quantity
            }
            inventory_data[str(product_id)] = product
            save_inventory_data()
            return 'Product added successfully!'
        else:
            return render_template('inventory.html', inventory=inventory_data)
    else:
        return redirect(url_for('dashboard'))
@app.route('/inventory_data', methods=['GET'])
def get_inventory_data():
    if 'employee_id' in session:
        return json.dumps(inventory_data)
    else:
        return redirect(url_for('dashboard'))
def load_inventory_data():
    global inventory_data
  if os.path.exists(inventory_file):
                with open(inventory_file, 'r') as f:
            inventory_data = json.load(f)
    else:
        inventory_data = {}  # Initialize as an empty dictionary if the file doesn't exist
def save_inventory_data():
    with open(inventory_file, 'w') as f:
        json.dump(inventory_data, f,indent=4)
@app.route('/decrease_quantity', methods=['POST'])
def decrease_quantity():
    if 'employee_id' in session:
        product_id = request.form['product_id']
        if product_id in inventory_data:
            product = inventory_data[product_id]
            quantity = int(product['quantity'])
```

```
            if quantity > 0:

 product['quantity'] = quantity - 1
 save_inventory_data()
                flash('Quantity decreased successfully!', 'success')
            else:
                flash('Cannot decrease quantity. It is already zero.', 'warning')
        else:
            flash('Product not found', 'danger')
    else:
        return redirect(url_for('dashboard'))
    return redirect(url_for('inventory'))
@app.route('/increase_quantity', methods=['POST'])
def increase_quantity():
    if 'employee_id' in session:
  product_id = request.form['product_id']
        if product_id in inventory_data:
            product = inventory_data[product_id]
            product['quantity'] = int(product['quantity']) + 1
            save_inventory_data()
            flash('Quantity increased successfully!', 'success')
        else:
            flash('Product not found', 'danger')
    else:
        return redirect(url_for('dashboard'))
    return redirect(url_for('inventory'))
```

- **Manage Reservations**

```
@app.route('/view_reservation')
def view_reservations():
    if 'employee_id' in session:
        return render_template('reservation.html', reservations=reservation_data)
    else:
        return redirect(url_for('dashboard'))
```

```
@app.route('/delete_reservation', methods=['POST'])
def delete_reservation():
    if 'employee_id' in session:
 reservation_id = request.form['reservation_id']
        if reservation_id in reservation_data:
            del reservation_data[reservation_id]
            save_reservation_data()
            flash('Reservation deleted successfully!', 'success')
        else:
            flash('Reservation not found', 'danger')
    else:
return redirect(url_for('dashboard'))
    return redirect(url_for('view_reservations'))
```

- **Manage Employee Details**

```
@app.route('/employee')
def employee():
    return render_template('view.html')
@app.route('/delete_employee', methods=['POST'])
def delete_employee():
    if 'employee_id' in session:
        employee_id = request.form['employee_id']
        if os.path.exists(db_file):
            with open(db_file, 'r') as f:
                tree = json.load(f)
        else:
            tree = {}
        if employee_id in tree:
            del tree[employee_id]
            save_employee_data(tree)
            flash('Employee deleted successfully!', 'success')
        else:
            flash('Employee not found', 'danger')
    else:        return redirect(url_for('dashboard'))
```

CHAPTER 5

TESTING

The Software testing consists of execution of a software component or a system component or a systemcomponent to confirm one or more itemized properties. It helps to point out the extent the system orsoftware gratifies the necessities specified by user, responds in an indicated way, it have to be installed and executed in all the envisioned environments.

5.1 Purpose of Testing

- Finding defects which may get created by the programmer while developing the software.
- To make sure that the end result meets the business and user requirements.
- To ensure that it satisfies Software Requirement Specification
- To gain the confidence of the customers by providing them a reliable and secure product.

5.2 Levels of Testing

- **Unit Testing**

  Unit testing is a level of testing process where individual units of a software or a system is tested. The purpose is to validate that each unit of the software perform as designed. Here we have done unit testing for each unit like login, logout, reservation management, inventory management.

- **Integration Testing**

  Integration testing is a level of software testing process where individual units are combined and tested as a group. The purpose of this level of testing is to expose faults in the interaction between integrated units. Here we have tested for integration of staff and manager module such that all the activities done by staff is reflected to the manager.

- **System Testing**

  System testing is a level of software testing process where a complete integrated software is tested. The purpose of this test is to evaluate the system's compliance with the specified requirements. After integration of all modules the system is finally checked against requirement specified.

5. 3 TEST CASES

Table 5.1: Test Cases for staff

Test Case No.	Test Case Name	Description	Expected Result	Actual Result	Status
1	Staff login	The staff login should be validated	Upon entering the required credentials the staff is directed to the employee dashboard.	After successful login, the staff is directed to employee dashboard.	Pass
2	Staff login	The staff login should be validated	System should alert the employee to fill the valid details.	System will ask the employee to enter valid details only.	Pass
3	Inventory Management	The items can be added and deleted accordingly.	System should alert the staff after adding or deleting an item in inventory.	System will alert the staff after each successful deletion and addition of items.	Pass
4	Manage Employee	The employee details can be managed here.	On clicking delete employee the system should alert the staff on successful deletion.	On the deletion of an employee, the system will throw an alert “Employee Deleted Successfully”.	Pass

5	Daily Orders	Manager should be able to track the order on daily basis.	System should track and display the orders of the day with date and time.	System will not track and display the orders of the day with date and time.	Fail
6	Employee Shift Management	The employee should be able to see his shifts.	System will display the shifts of the employee.	System will not display the shifts of the employee.	Fail

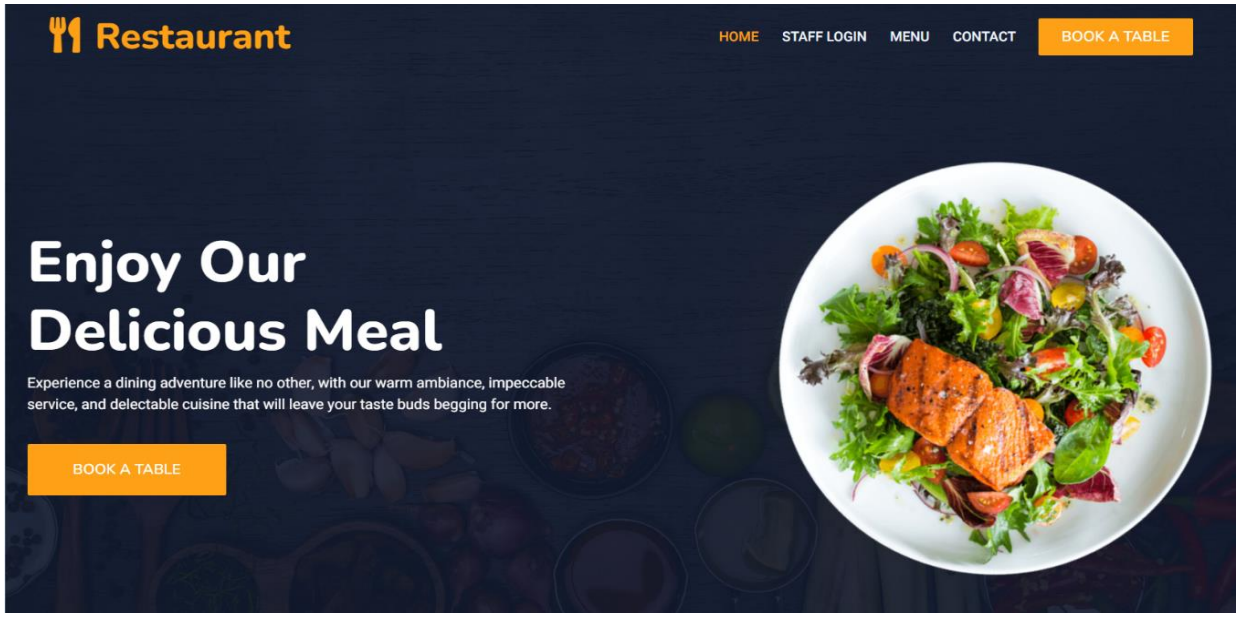
Table 5.2: Test Cases for Customer

Test Case No.	Test Case Name	Description	Expected Result	Actual Result	Status
1	Customer Booking	The customer can book a table according to his requirements.	System should throw an alert message when the customer makes a reservation.	System will throw an alert as soon as the customer makes the table reservation.	Pass
2	Customer Booking	The customer can book a table according to his requirements.	Error message as enter the valid credentials should be displayed.	Enter the valid credentials will be displayed as soon as the customer tries to make the reservation with invalid credentials.	Pass

CHAPTER 6

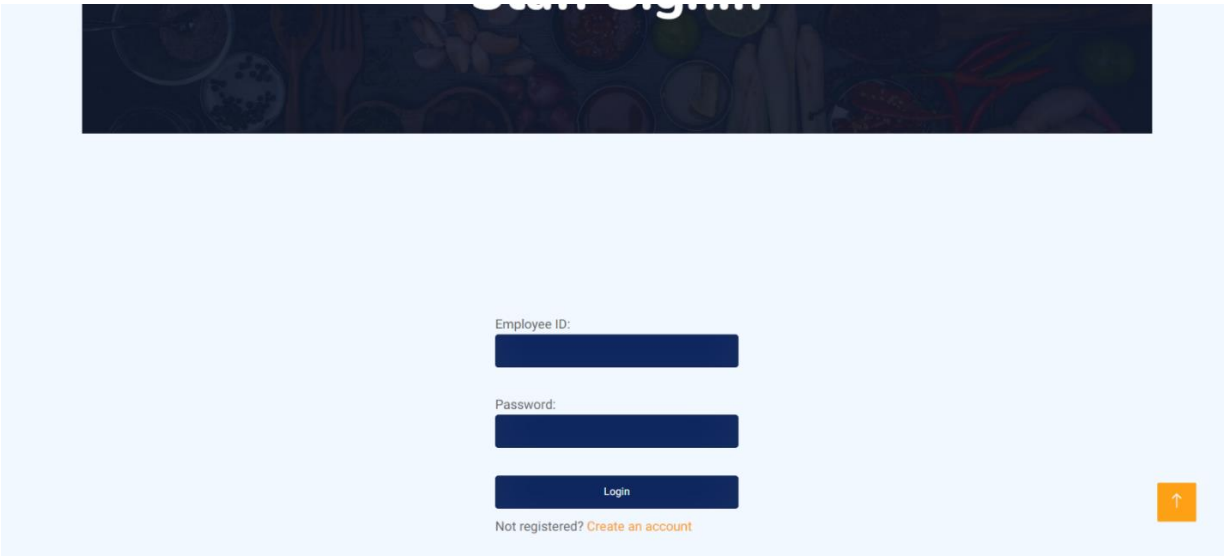
RESULTS AND DISCUSSION

6.1 Snapshots



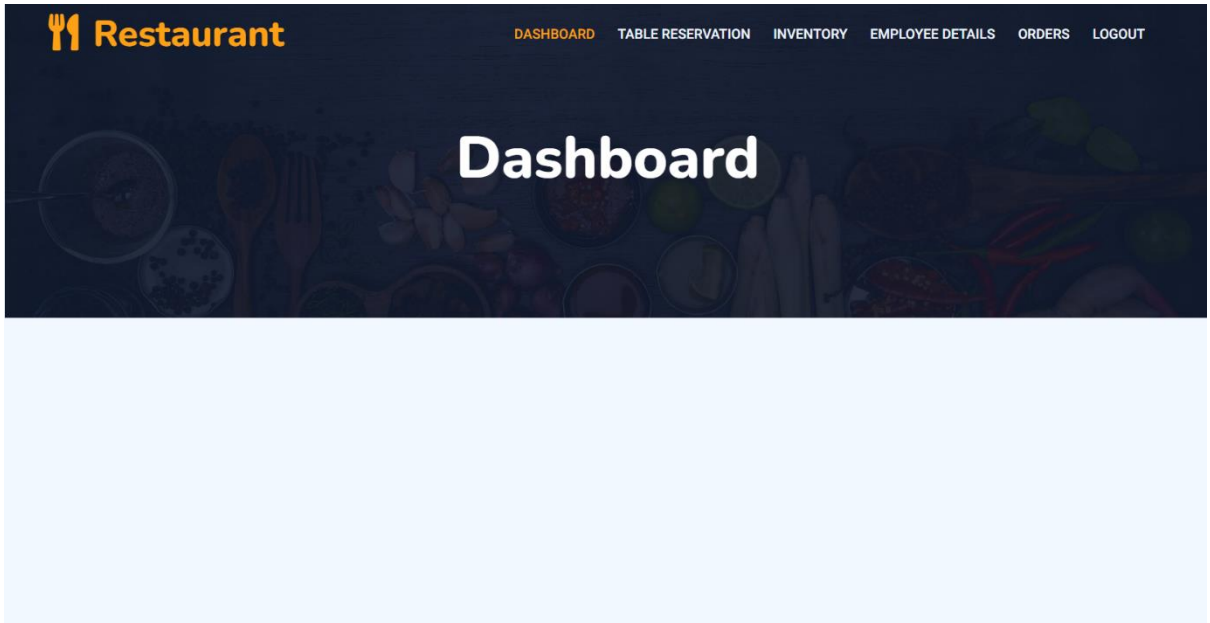
Snapshot 6.1: Home page of Restaurant Management System

Snapshot 6.1 is all about the home page of the web application from where desire activities can be carried out accordingly.



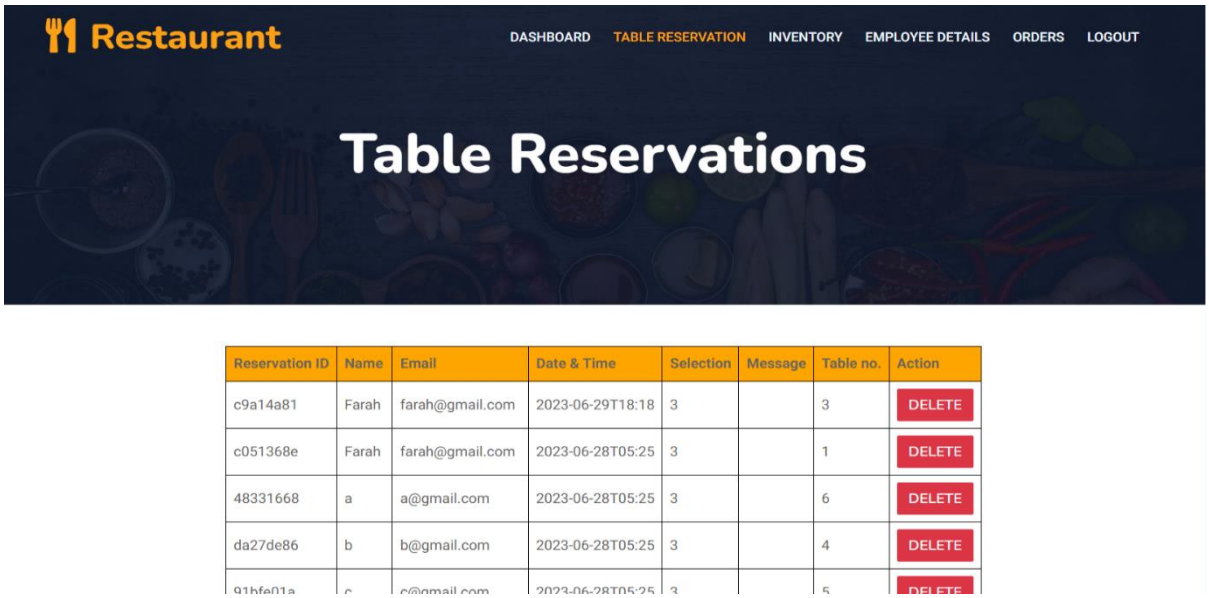
Snapshot 6.2: Login page for staff

Snapshot 6.2, shows the login page of the Restaurant Staff where upon the successful login he can manage menu, reservations, orders, and inventory.



Snapshot 6.3: Home page for Staff

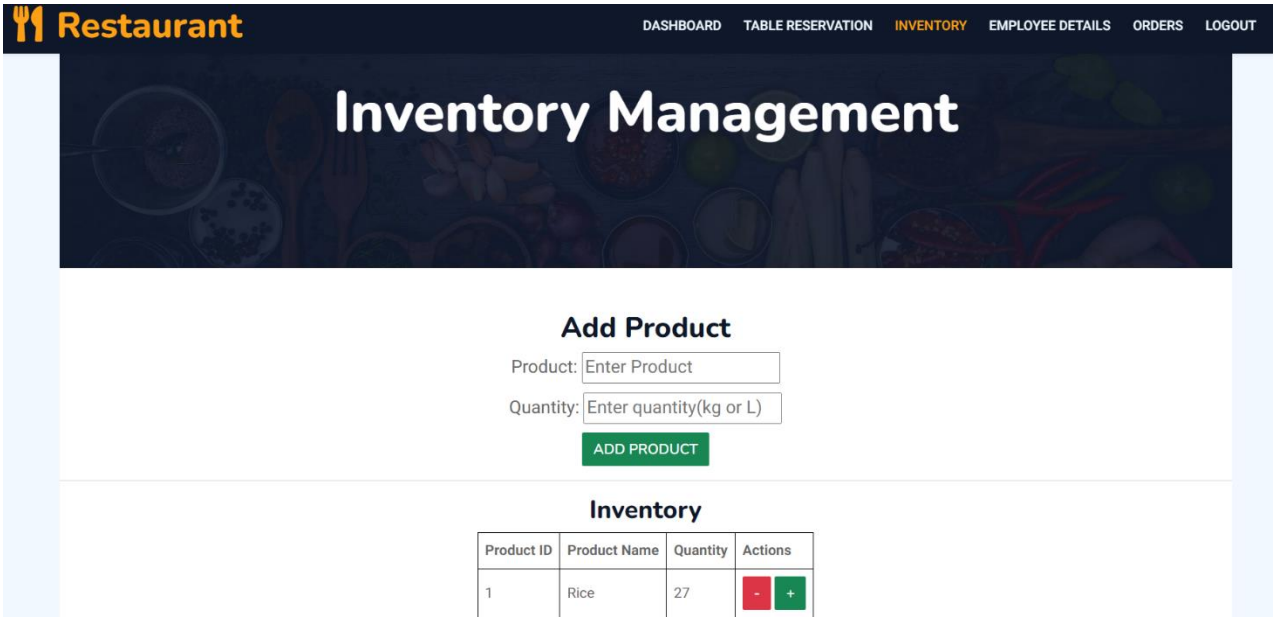
Snapshot 6.3 shows the home page of staff after the successful login. Here the Manager can perform various activities such as Manage Inventory, Manage orders and reservations, view employee details and at last can logout.



Reservation ID	Name	Email	Date & Time	Selection	Message	Table no.	Action
c9a14a81	Farah	farah@gmail.com	2023-06-29T18:18	3		3	DELETE
c051368e	Farah	farah@gmail.com	2023-06-28T05:25	3		1	DELETE
48331668	a	a@gmail.com	2023-06-28T05:25	3		6	DELETE
da27de86	b	b@gmail.com	2023-06-28T05:25	3		4	DELETE
91bfe01a	c	c@gmail.com	2023-06-28T05:25	3		5	DELETE

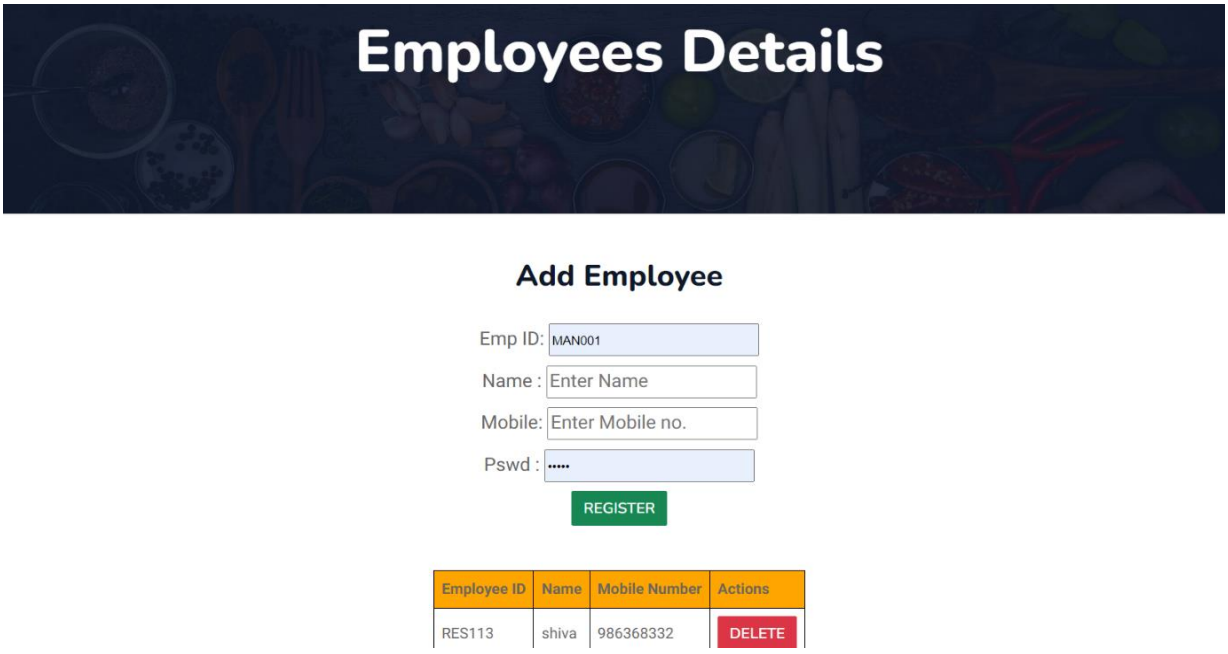
Snapshot 6.4: Table Reservations view for Staff

Snapshot 6.4 shows the reservation page for the staff after the successful login. Here the manager can see the reservations that are done by the customer and take necessary action if the tables are not available. Here the details of the customers provided during the reservation will be stored and managed.



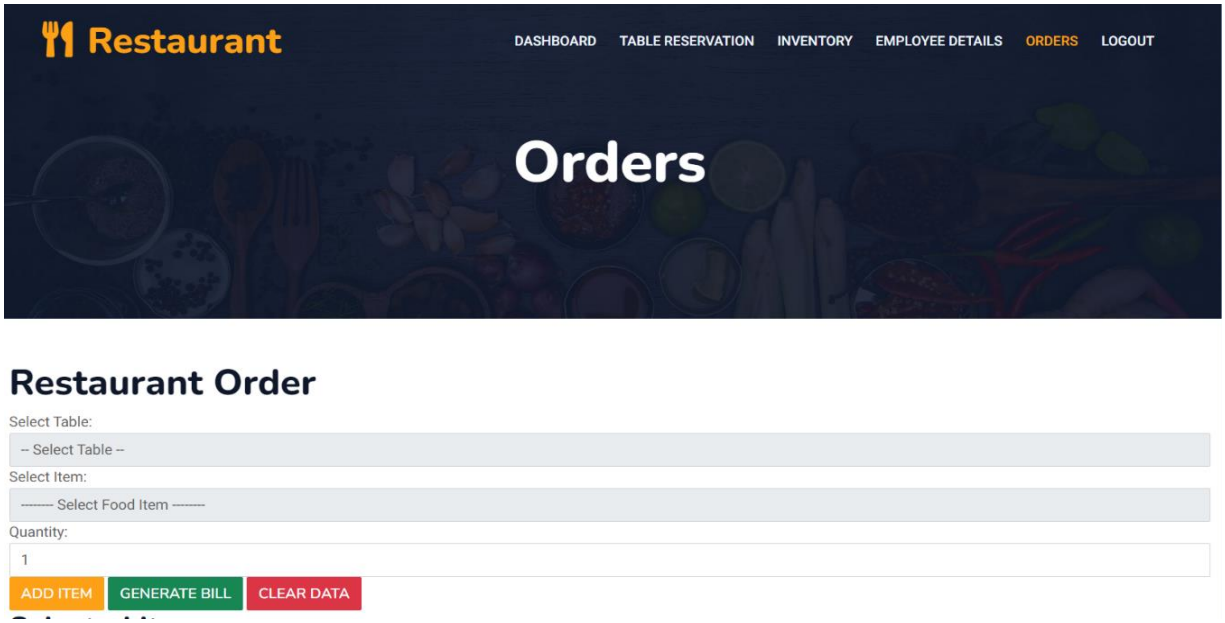
Snapshot 6.5: Inventory Management Page

Snapshot 6.5 shows the inventory management of the restaurant where the staff upon the successful registration will handle it. Here the manager gets to add items and delete items from the inventory accordingly.



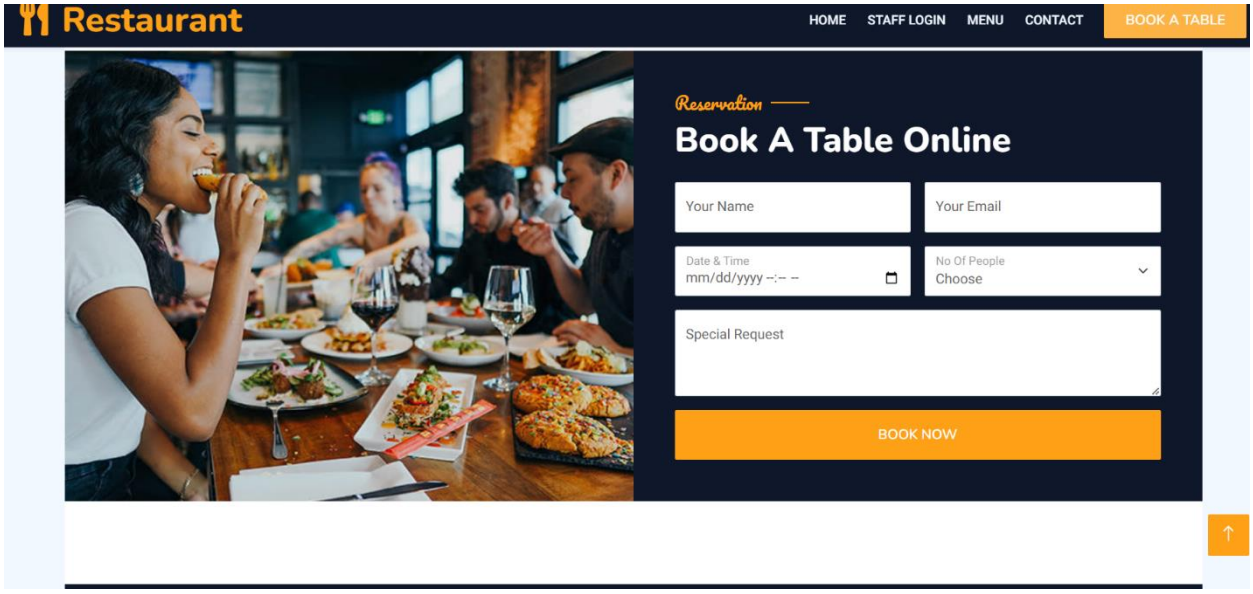
Snapshot 6.6: Employee details page

Snapshot 6.6 shows the employee details page. Here the employee details can be viewed and managed. Upon the successful login, manager can view and manage the employee details.



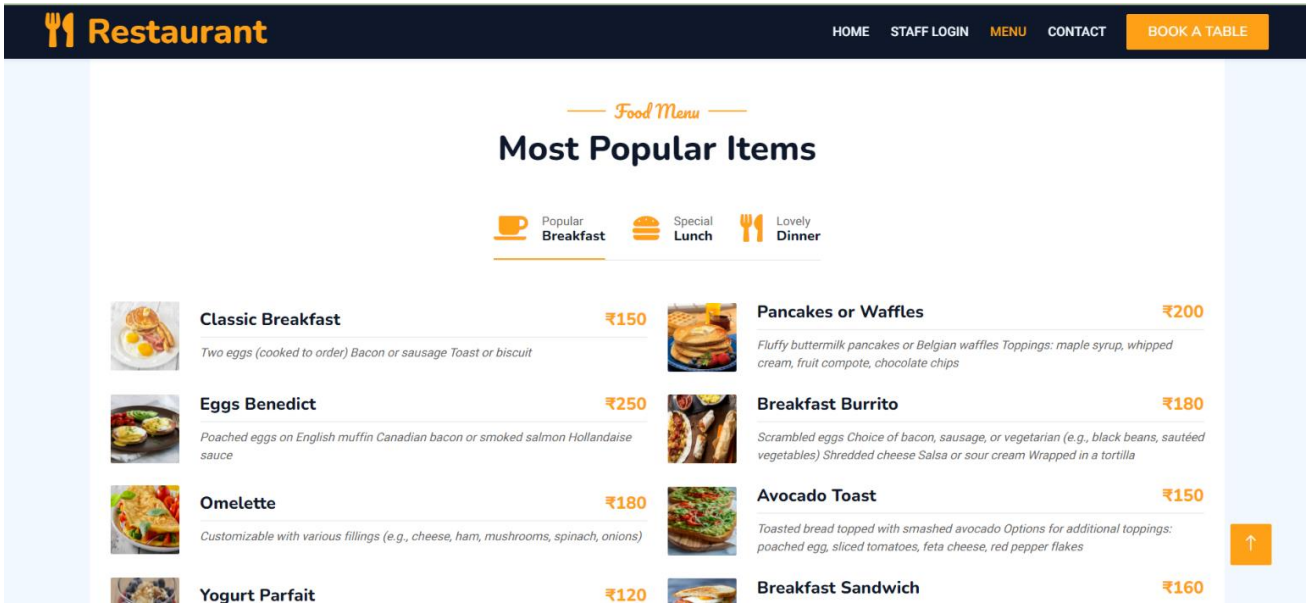
Snapshot 6.7: Orders page

Snapshot 6.7 shows the orders page of the Restaurant management system. Here the employee gets to manage the orders of the customers. Upon selecting the table and the food ordered from the menu the bill can be generated.



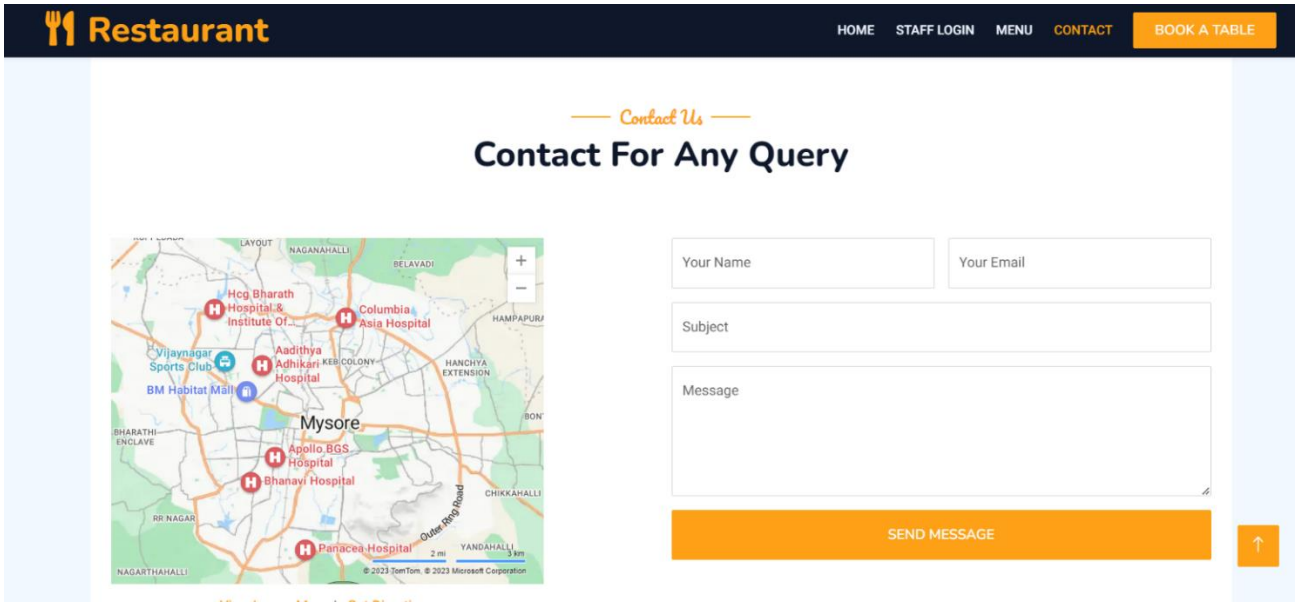
Snapshot 6.8: Table booking for customers

Snapshot 6.8 shows the table booking for customers. Here the customer can make a reservation accordingly by providing necessary required credentials.



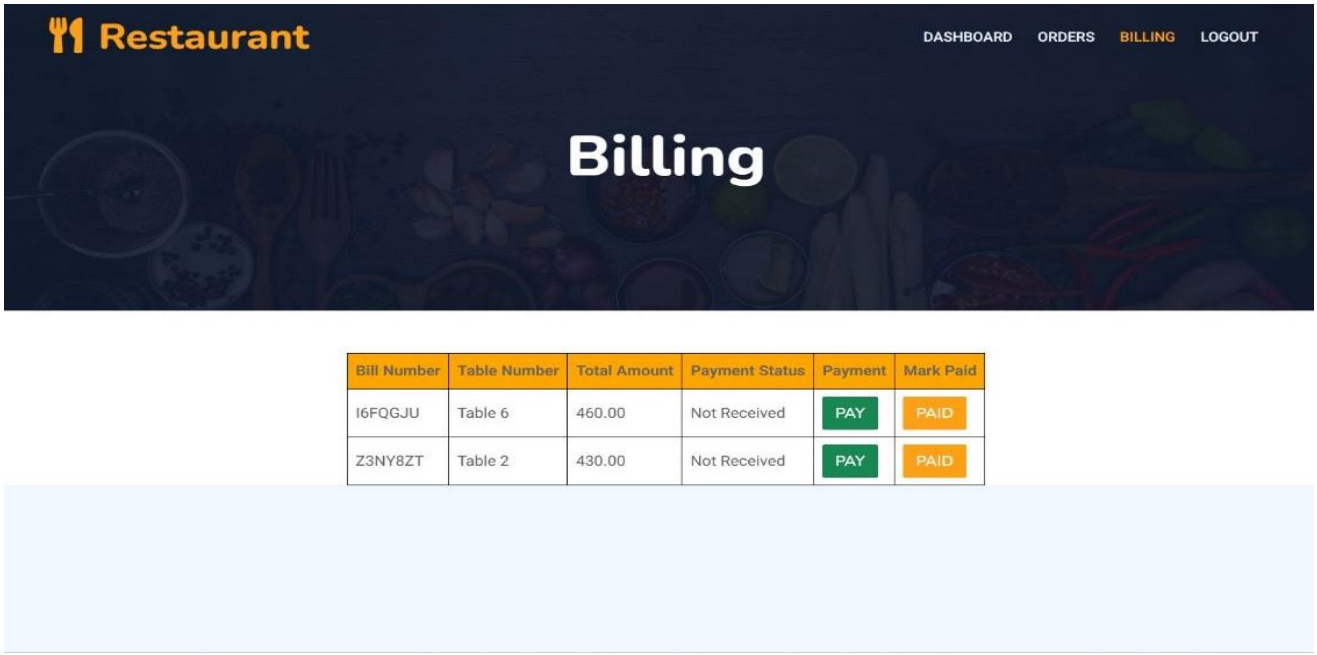
Snapshot 6.9: Menu provided in Restaurant

Snapshot 6.9 shows the menu of the restaurant. This is visible to customers to order the food as per their wish. It contains several food items along with their price tag. And this is managed by the Restaurant's staff.



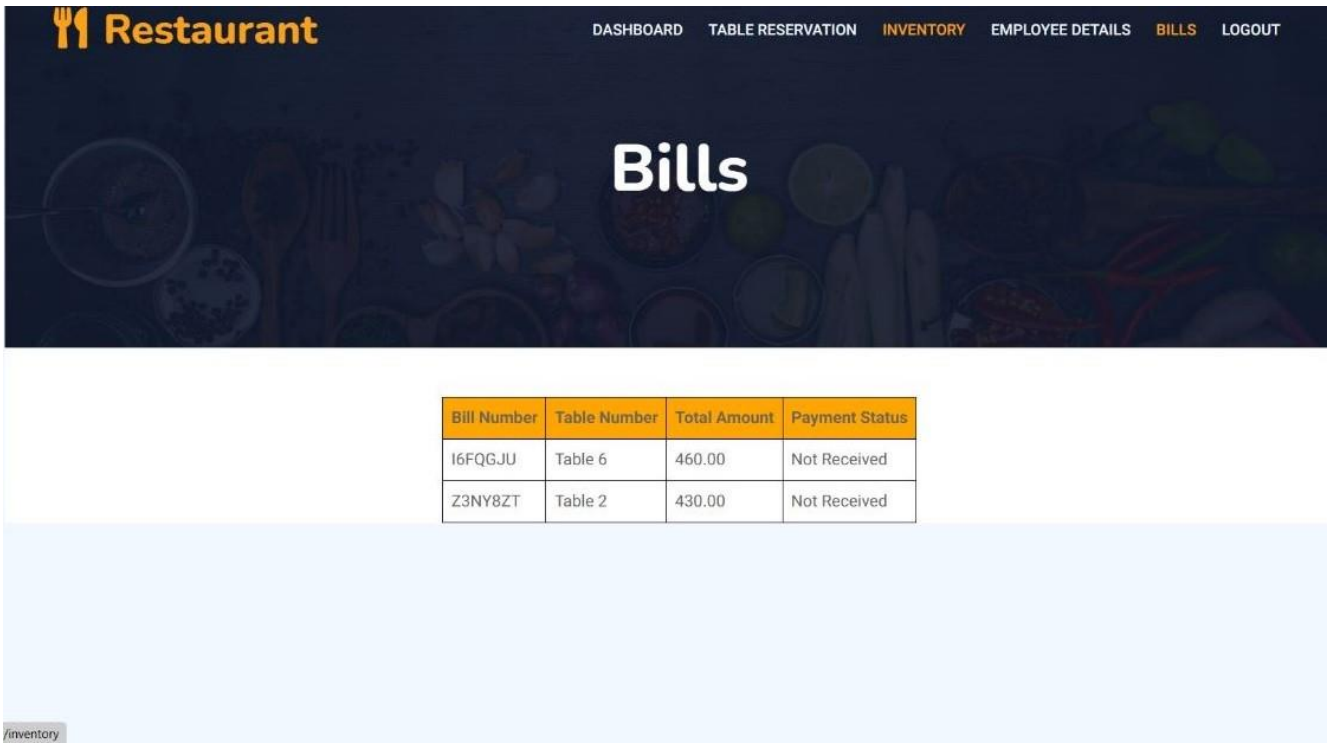
Snapshot 6.10: Contact

Snapshot 6.10 deals with the contact for any query, where the customers can contact the restaurant's staff members regarding any query by providing the following necessary credentials. The can drop their message in the text box provided for the message.



Snapshot 6.11: Billing

Snapshot 6.11 shows the billing details of customers where they can make the payment. By clicking on pay they can make the payment. After successfully making the payment, the staff will mark it as paid.



Snapshot 6.12: Bills Details

Snapshot 6.12 gives the overall view of bills generated to the manager where he can see if the payment has been made or not.

CONCLUSION AND FUTURE ENHANCEMENT

B+ tree is a data structure that can be used to optimize the performance of a Restaurant Management System. The use of B+ tree in a Restaurant Management System enables efficient table management, fast search and retrieval of data, accurate inventory management, enhanced customer experience, and streamlined billing processes. By providing real-time information about table availability, menu items, and wait times, a Restaurant Management System using B+ tree can improve customer satisfaction and increase the efficiency of the restaurant's operations. Overall, the use of B+ tree in an Restaurant Management System is a valuable technique that can help restaurant owners and managers manage their business more effectively.

We can enhance the project by adding following functions:

- We intend to advance analytics and reporting capabilities which provide deeper insights into key performance metrics, such as sales trends, customer behavior, and inventory management. These insights can help restaurant owners make data-driven decisions to optimize operations and profitability.
- We also intend to have enhanced staff management features which include employee scheduling, shift management, and performance tracking. These features can help optimize labor costs, ensure adequate staffing levels during peak hours, and improve communication among the staff.

REFERENCES

Literature Survey:

[1] Mayur D. Jakhete, Piyush C. Mankar," Implementation of Smart Restaurant with e-menu Card," International Journal of Computer Applications 2015.

[2] "Implementing a Web-Based Computerized Restaurant System" by Chin Loong Tan (2013).

[3] "Online Restaurant Management System" by Nur Yasmin Binti Mohd Nasir (2022).

[4] "A Review of Restaurant Management System and their Implementation" by Karim Allaham and Mahmoud Al-Qutayri (2014).

[5] "A Survey of Restaurant Management System" by Kiran Kumar A, Jyothi B, and Basavaraj S. Anami (2015).

[6] Hanisah Binti Md Taha,(2008), "Online Restaurant Management System (ORMS)," Faculty of information and communication technology, Universitite knikal Malaysia melaka. 34.

[7] Soon Nyean Cheong, Wei Wing Chiew,Wen JiunYap(2010), "Design and Development of MultiTouchable E-Restaurant Management System" ,in 2010 International Conference on Science and Social Research (CSSR 2010), December 5 - 7, 2010, Kuala Lumpur, Malaysia.

Websites:

[1] https://restaurant.eatapp.co/blog/ultimate-guide-to-restaurant-management-software

[2] https://www.business.com/articles/restaurant-management-system-guide/

[3] https://www.researchgate.net/publication/260652458_Developing_an_Intelligent_e-Restaurant_With_a_Menu_Recommender_for_Customer-Centric_Service