# RAJALAKSHMI ENGINEERING COLLEGE

**RAJALAKSHMI NAGAR, THANDALAM-602 105**


## CS23333 OOPS Using Java

**Laboratory Record Note Book**

| | |
|---|---|
| **Name :** | GOKULAN V |
| **Year / Branch / Section :** | II / CSE |
| **University Register No. :** | 2116240701151 |
| **College Roll No. :** | 240701151 |
| **Semester :** | III |
| **Academic Year :** | 2024-2028 |

# RAJALAKSHMI ENGINEERING COLLEGE

**An Autonomous Institution**

## BONAFIDE CERTIFICATE

**Name:** GOKULAN V

**Academic Year:** 2024-2028 **Semester:** III **Branch:** CSE

**Register No.** 240701151

***Certified that this is the bonafide record of work done by the above student in the*** OOPS USING JAVA ***Laboratory during the academic year 2025- 2026***

**Signature of Faculty in-charge**

**Submitted for the Practical Examination held on……**

**Internal Examiner** **External Examiner**

# INDEX

| EX.NO | DATE | NAME OF THE EXPERIMENT | GITHUB QR |
|---|---|---|---|
| 1 | | I/O, Data Types, Operators | https://github.com/Gokulan00/240701151-CS23333-JAVA-Lab |
| 2 | | Control Structures | |
| 3 | | Arrays | |
| 4 | | Strings | |
| 5 | | Classes & Objects | |
| 6 | | Inheritance | |
| 7 | | Interface | |
| 8 | | Exceptions | |
| 9 | | Collections | |
| 10 | | Collections | |
| 11 | | Project | |
| 12 | | Lambda | |

# ABSTRACT

The Smart Canteen Management System is a Java-based application designed to automate and simplify the food ordering and billing process within an institution. Manual canteen operations often lead to confusion, delays, and errors; this system eliminates those issues by providing a fast, accurate, and reliable digital solution. The application enables administrators to manage menu items, orders, and user details, while students or staff can log in, view the menu, and place orders conveniently. It is developed using Java, JSP, and Servlets for the application layer, and MySQL as the back-end database to ensure efficient data handling and secure storage. The system follows a three-tier architecture—Presentation Layer, Application Layer, and Database Layer—providing modularity, scalability, and ease of maintenance. Overall, it offers an effective platform that reduces manual effort, improves accuracy, and enhances the efficiency of canteen operations.

# ACKNOWLEDGEMENT

We express our sincere thanks to our beloved and honorable Chairman **Mr. S. MEGANATHAN** and Chairperson **Dr. M. THANGAM MEGANATHAN** for their continuous encouragement and support.We are deeply grateful to our respected Principal **Dr. S. N. MURUGESAN** for his guidance and motivation throughout our project. Our heartfelt thanks to our Head of the Department **Dr. E. M. MALATHY** and Deputy Head **Dr. J. MANORANJINI** for their valuable suggestions and encouragement. We also extend our sincere gratitude to our internal guide **Mr. N. KARTHIKEYAN** for his valuable guidance, timely assistance, and constant motivation during the development of our project. Finally, we thank our family members, friends, and all the staff members of the Department of Computer Science and Engineering for their continuous support and cooperation in completing this project successfully.

**240701151 – GOKULAN V**

**240701155 – GOWTHAM RAJ M**

**240701156 – GOWTHAM S**

# TABLE OF CONTENTS

## LIST OF FIGURES

# CHAPTER 1 - INTRODUCTION

The Smart Canteen Management System is a Java-based application developed to automate and streamline the daily operations of an institutional canteen. Managing food orders, billing, and menu details manually often leads to confusion, delays, and errors. This project aims to overcome these challenges by providing a computerized system that efficiently handles user login, order placement, and billing operations in a structured and organized manner.

The system allows administrators to manage menu items, update availability, and monitor daily orders, while students or staff can log in, view the available menu, and place their orders conveniently. By integrating all these features into a single platform, the system reduces manual workload, saves time, and improves the accuracy and efficiency of canteen operations.

Developed using Java, JSP, and Servlets for the front-end and MySQL for data storage, the system follows a three-tier architecture consisting of the Presentation Layer, Application Layer, and Database Layer. This architectural design ensures modularity, scalability, and easy maintenance while applying object-oriented programming principles for code reusability and efficiency.

In summary, the Smart Canteen Management System provides a reliable, user-friendly, and efficient platform for managing canteen operations. It automates key processes such as ordering, billing, and data management, minimizes human errors, and lays the foundation for future enhancements such as online payment integration and real-time order tracking.

# CHAPTER 2 - SYSTEM SPECIFICATIONS

**Hardware Specifications:**

**Processor**: Intel i5
**RAM**: 8 GB minimum
**Hard Disk**: 500 GB minimum

**Software Specifications:**

**Operating System:** Windows 11
**Front-End**: Java
**Back-End**: File Handling / Database (if extended)
**Language:** Java

# CHAPTER 3 - MODULE DESCRIPTION

**1. Admin Module:**

The Admin Module provides the administrator with complete control over the canteen's operations. It is responsible for managing menu items, monitoring orders, handling user data, and maintaining overall system efficiency. The key functionalities include:

- Menu Management:
  The admin can add new food items, update existing menu details such as price and availability, and remove items that are no longer served.
- Order Management:
  Allows the admin to view and track all orders placed by users, including order ID, item details, quantity, total price, and order status. This helps in maintaining transparency and ensuring timely service.
- User Management:
  Enables the admin to view user accounts, check their order history, and manage login credentials securely.
- Billing and Report Generation:
  Displays daily sales reports, total revenue, and order summaries, helping the canteen monitor performance and inventory requirements efficiently.

This module ensures that the administrator has centralized access to manage all canteen-related data securely and effectively, improving operational efficiency and customer satisfaction.

**2. Customer Module:**

The Customer Module focuses on improving the ordering experience for students and staff using the Smart Canteen Management System. It allows users to register, log in, view available food items, place orders, and view their bills in a convenient and efficient manner. The major components include:

- User Registration and Login:
  Users can create an account by providing details such as username, password, and contact information. Existing users can securely log in using their credentials to access the system.
- Menu Browsing and Search:
  Users can view the available food items along with their prices and availability. The search option helps them find specific dishes quickly and efficiently.
- Order Placement:
  Customers can select the desired food items, specify the quantity, and place their orders. The system automatically calculates the total price and records the order details in the database.

- Billing and Order Summary:
  Once the order is placed, the system generates a detailed bill displaying item names, quantity, price, and total amount.
- Account Settings:
  Users can update their personal information and change their login credentials when required.

This module ensures a simple, fast, and user-friendly food ordering process, improving convenience for users while maintaining accuracy and reliability in order management.

**3. Database Management Module:**

The Database Management Module serves as the backbone of the Smart Canteen Management System. It stores and manages all the essential data related to users, menu items, orders, and billing using a MySQL database. The system performs operations such as data insertion, updating, retrieval, and deletion using SQL queries, ensuring efficient and accurate data handling.

Key tables include:

- students – stores user login details such as username and password.
- menu – contains information about food items, their prices, and availability.
- orders – records details of every order placed, including item ID, quantity, and total amount.
- bills – maintains billing information for each completed order.

This module ensures data integrity, consistency, and security across all operations, supporting smooth communication between the application and the database while maintaining reliable storage for all canteen-related data.

**4. Security and Validation Module:**

The Security and Validation Module ensures the safety, reliability, and accuracy of all user interactions within the Smart Canteen Management System. It manages user authentication and validates all input data to prevent errors and unauthorized access.

- Login Validation:
  Verifies login credentials for both admin and users to ensure only authorized individuals can access the system.

- Registration Validation:
  Checks input fields such as username, password, and contact details for correctness and uniqueness to avoid duplicate or invalid accounts.
- Order and Feedback Validation:
  Prevents invalid entries such as empty orders, incorrect quantities, or incomplete feedback submissions.
- Database Protection:
  Uses parameterized SQL queries to prevent SQL injection and ensure secure communication between the application and the database.

This module plays a vital role in maintaining data security, system stability, and user trust by ensuring all transactions and inputs are properly verified before processing.

-

**5. Reporting and Analytics Module:**

The Reporting and Analytics Module provides the administrator with summarized data and useful insights to improve decision-making and canteen management efficiency. It helps track performance and identify key trends within the system.

- Sales and Revenue Reports:
  Displays overall daily, weekly, or monthly sales records and calculates total revenue generated.
- Order and User Tracking:
  Analyzes customer order patterns, frequent users, and popular menu items to enhance service and planning.
- Menu Insights:
  Identifies the most-ordered food items, helping the admin manage stock, adjust prices, and update the menu accordingly.

Although currently limited to text-based reporting, this module can be enhanced in the future with graphical dashboards and data visualization tools to provide a more interactive and real-time analysis of canteen operations.

# CHAPTER – 4 ARCHITECTURE

## 4.1 ARCHITECTURE DIAGRAM

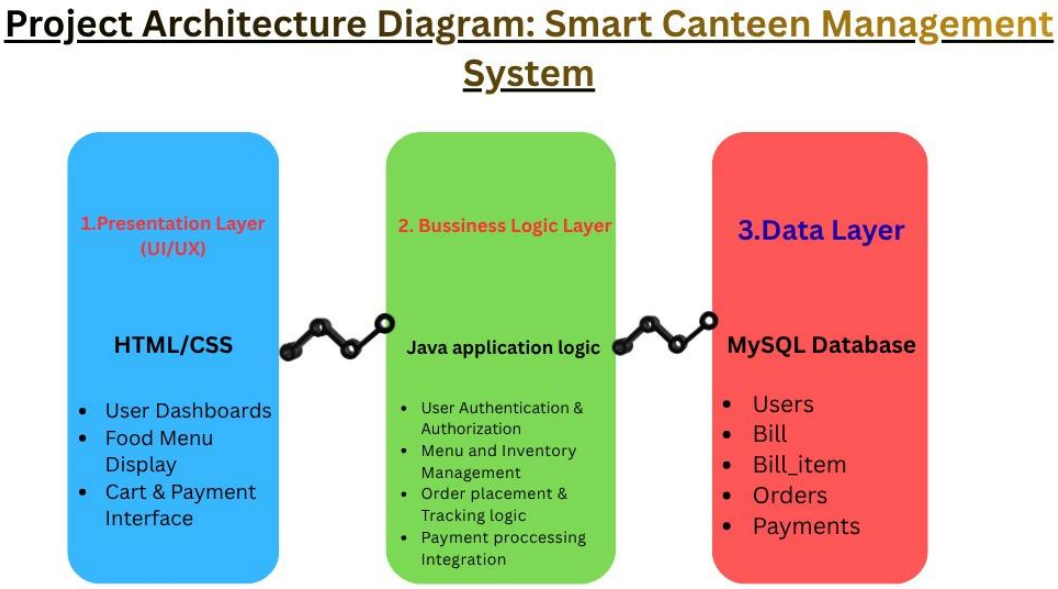


**FIG NO: 4.1 ARCHITECTURE DIAGRAM**

### 4.1 Architecture Overview

The Smart Canteen Management System follows a three-tier architecture that includes the Presentation Layer, Application (Business Logic) Layer, and Database Layer. This structure enhances maintainability, scalability, and performance by separating user interaction, business logic, and data management into distinct layers.

1. Presentation Layer
The Presentation Layer serves as the user interface of the system. It allows users (Admin and Students/Staff) to interact with the application through web pages such as login, registration, menu display, and order placement. Implemented using JSP and HTML, this layer captures user inputs, sends them to the business logic layer, and displays processed results like order confirmation and billing information.

2. Application Layer (Business Logic Layer)
The Application Layer handles the main processing and decision-making of the system. It validates user inputs, applies business rules, and coordinates communication between the user interface and the database. Implemented using Java Servlets, this layer manages functions like verifying login credentials, processing orders, generating bills, and updating menu details. It ensures accuracy, efficiency, and data consistency before interacting with the database.

3. Database Layer
The Database Layer is responsible for the storage, retrieval, and management of data. It uses a MySQL database to maintain tables for users, menu items, and orders. All operations such as adding, updating, or deleting data are performed using JDBC connectivity. This layer ensures data integrity, security, and consistency across all canteen operations.

This three-tier architecture ensures that each layer works independently yet cohesively, making the Smart Canteen Management System reliable, modular, and easy to maintain or upgrade in the future.

## 4.2 E - R DIAGRAM :

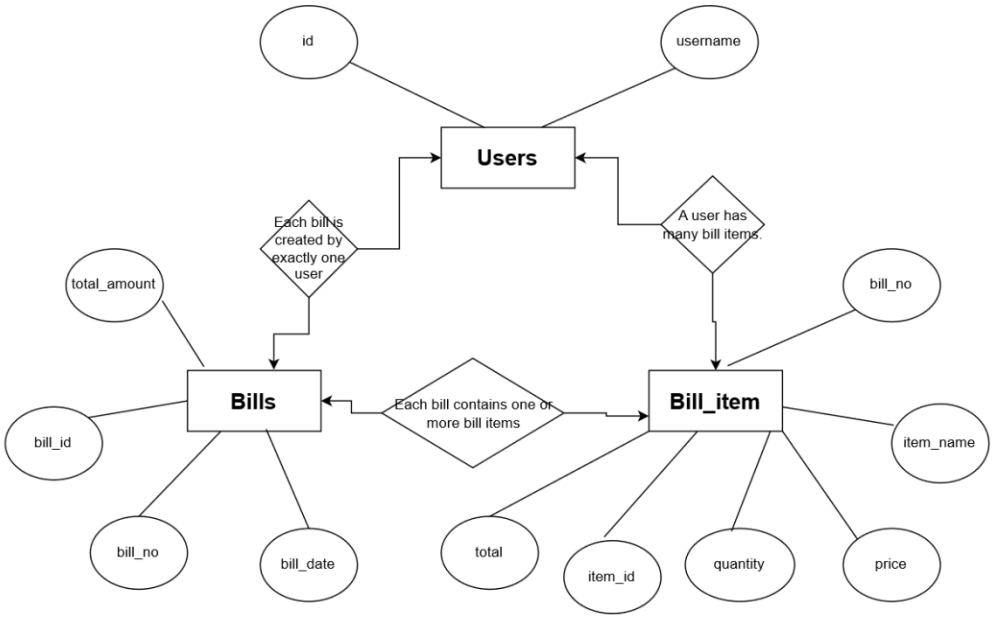


**FIG NO : 4.2 ENTITY RELATIONSHIP DIAGRAM.**

**Entities**

1. **Customer** – Represents users who register and make purchases in the system.
    - *Attributes:* Username, Password, Age, Mobile Number, Email.
2. **Admin** – Represents the administrator who manages the system's operations.
    - *Attributes:* Admin_ID, Name, Password.
3. **Product** – Contains details of sports items available in the store.
    - *Attributes:* Product_ID, Product_Name, Stock, Price.
4. **Employee** – Represents store staff managed by the admin.
    - *Attributes:* Name, Contact, Job, Salary.
5. **Sales** – Stores information about purchase transactions.
    - *Attributes:* Sale_ID, Username, Product_ID, Quantity, Total_Price, Date_Time.
6. **Feedback** – Records customer feedback and ratings for the store.

- *Attributes:* Feedback_ID, Username, Feedback_Text, Rating.

**Relationships**

1. **Customer – Sales** → *One-to-Many*
    - A single customer can make multiple purchases, but each sale is linked to only one customer.
2. **Product – Sales** → *One-to-Many*
    - A single product can appear in multiple sales transactions.
3. **Admin – Employee** → *One-to-Many*
    - The admin manages multiple employees.
4. **Customer – Feedback** → *One-to-Many*
    - Each customer can provide multiple feedback entries, but each feedback belongs to one customer.
5. **Admin – Product** → *One-to-Many*
    - The admin can add or update multiple products in the store database.

**Connections :**

- **Customer ↔ Sales ↔ Product:** Connects customers with the products they purchase.
- **Customer ↔ Feedback:** Shows the link between customers and their submitted reviews.
- **Admin ↔ Employee / Product:** Represents administrative control over staff and inventory.
- **Sales ↔ Product:** Establishes the product's involvement in specific sales transactions.

# CHAPTER 5 – IMPLEMENTATION

1) SQL -

```
-- create database (run once)
CREATE DATABASE IF NOT EXISTS smart_canteen;
USE smart_canteen;

-- students table
CREATE TABLE IF NOT EXISTS students (
  id INT AUTO_INCREMENT PRIMARY KEY,
  username VARCHAR(100) UNIQUE NOT NULL,
  password VARCHAR(100) NOT NULL,
  name VARCHAR(200)
);

-- menu table
CREATE TABLE IF NOT EXISTS menu (
  id INT AUTO_INCREMENT PRIMARY KEY,
  name VARCHAR(200) NOT NULL,
  price DECIMAL(8,2) NOT NULL,
  available BOOLEAN DEFAULT TRUE
);

-- orders table
```

```
CREATE TABLE IF NOT EXISTS orders (
  id INT AUTO_INCREMENT PRIMARY KEY,
  student_id INT,
  item_id INT,
  quantity INT,
  total_price DECIMAL(10,2),
  order_time TIMESTAMP DEFAULT CURRENT_TIMESTAMP,
  FOREIGN KEY (student_id) REFERENCES students(id) ON DELETE SET NULL,
  FOREIGN KEY (item_id) REFERENCES menu(id) ON DELETE SET NULL
);

-- Insert sample user and sample menu items
INSERT INTO students (username, password, name) VALUES ('Gokulan','240701151','Gokul');
INSERT INTO menu (name, price, available) VALUES
 ('Veg Biriyani',55.00,TRUE),
 ('Chole Puri',55.00,TRUE),
 ('Egg Fried Rice',65.00,TRUE),
 ('Omelette',25.00,TRUE);
```

---

2) Java source files (package db)

DBConnection.java

```
package db;

import java.sql.Connection;
import java.sql.DriverManager;

public class DBConnection {
   // Update DB name/user/password if yours are different
   private static final String URL =
"jdbc:mysql://localhost:3306/smart_canteen?useSSL=false&allowPublicKeyRetrieval=true&ser
verTimezone=UTC";
   private static final String USER = "root";
   private static final String PASS = "Gokul@2007";

   public static Connection getConnection() throws Exception {
      Class.forName("com.mysql.cj.jdbc.Driver");
      return DriverManager.getConnection(URL, USER, PASS);
   }
}
```

---

StudentDAO.java

```
package db;

import java.sql.Connection;
import java.sql.PreparedStatement;
import java.sql.ResultSet;

public class StudentDAO {
   public static boolean login(String username, String password) {
      String sql = "SELECT id FROM students WHERE username=? AND password=?";
      try (Connection con = DBConnection.getConnection();
           PreparedStatement ps = con.prepareStatement(sql)) {
         ps.setString(1, username);
         ps.setString(2, password);
         try (ResultSet rs = ps.executeQuery()) {
            return rs.next();
         }
      } catch (Exception e) {
         e.printStackTrace();
         return false;
      }
   }

   // fetch id by username
   public static int getStudentId(String username) {
      String sql = "SELECT id FROM students WHERE username=?";
      try (Connection con = DBConnection.getConnection();
           PreparedStatement ps = con.prepareStatement(sql)) {
         ps.setString(1, username);
```

```
            try (ResultSet rs = ps.executeQuery()) {
                if (rs.next()) return rs.getInt("id");
            }
        } catch (Exception e) {
            e.printStackTrace();
        }
        return 0;
    }
}
```

---

MenuDAO.java

```
package db;

import java.sql.Connection;
import java.sql.PreparedStatement;
import java.sql.ResultSet;
import java.util.ArrayList;
import java.util.List;
```

```
public class MenuDAO {
  public static class Item {
    public int id;
    public String name;
    public double price;
    public boolean available;
    public Item(int id, String name, double price, boolean available) {
      this.id = id; this.name = name; this.price = price; this.available = available;
    }
  }

  public static List<Item> getAllItems() {
    List<Item> list = new ArrayList<>();
    String sql = "SELECT id, name, price, available FROM menu";
    try (Connection con = DBConnection.getConnection();
       PreparedStatement ps = con.prepareStatement(sql);
       ResultSet rs = ps.executeQuery()) {
      while (rs.next()) {
        list.add(new Item(rs.getInt("id"),
                  rs.getString("name"),
                  rs.getDouble("price"),
                  rs.getBoolean("available")));
      }
    } catch (Exception e) {
      e.printStackTrace();
    }
    return list;
```

```
    }

    public static double getPrice(int itemId) {
        String sql = "SELECT price FROM menu WHERE id=?";
        try (Connection con = DBConnection.getConnection();
             PreparedStatement ps = con.prepareStatement(sql)) {
            ps.setInt(1, itemId);
            try (ResultSet rs = ps.executeQuery()) {
                if (rs.next()) return rs.getDouble("price");
            }
        } catch (Exception e) {
            e.printStackTrace();
        }
        return 0;
    }
}
```

---

OrderDAO.java

```
package db;

import java.sql.Connection;
import java.sql.PreparedStatement;

public class OrderDAO {
```

```
   public static boolean placeOrder(int studentId, int itemId, int quantity, double totalPrice) {
      String sql = "INSERT INTO orders (student_id, item_id, quantity, total_price) VALUES
(?,?,?,?)";
      try (Connection con = DBConnection.getConnection();
           PreparedStatement ps = con.prepareStatement(sql)) {
         ps.setInt(1, studentId);
         ps.setInt(2, itemId);
         ps.setInt(3, quantity);
         ps.setDouble(4, totalPrice);
         int rows = ps.executeUpdate();
         return rows > 0;
      } catch (Exception e) {
         e.printStackTrace();
         return false;
      }
   }
}
```

---

LoginServlet.java

```
package db;

import jakarta.servlet.ServletException;
import jakarta.servlet.http.*;
import java.io.IOException;
```

```
public class LoginServlet extends HttpServlet {
   protected void doPost(jakarta.servlet.http.HttpServletRequest request,
                         jakarta.servlet.http.HttpServletResponse response)
           throws ServletException, IOException {

       String username = request.getParameter("username");
       String password = request.getParameter("password");

       if (StudentDAO.login(username, password)) {
           // set username in session
           HttpSession session = request.getSession();
           session.setAttribute("username", username);
           response.sendRedirect("menu"); // mapped to MenuServlet
       } else {
           response.sendRedirect("login.jsp?error=1");
       }
   }
}
```

---

MenuServlet.java

```
package db;

import jakarta.servlet.ServletException;
```

```
import jakarta.servlet.RequestDispatcher;
import jakarta.servlet.http.*;
import java.io.IOException;
import java.util.List;

public class MenuServlet extends HttpServlet {
   protected void doGet(jakarta.servlet.http.HttpServletRequest request,
                        jakarta.servlet.http.HttpServletResponse response)
           throws ServletException, IOException {

       List<MenuDAO.Item> items = MenuDAO.getAllItems();
       request.setAttribute("items", items);
       // forward to JSP
       RequestDispatcher rd = request.getRequestDispatcher("menu.jsp");
       rd.forward(request, response);
   }
}
```

---

OrderServlet.java

```
package db;

import jakarta.servlet.ServletException;
import jakarta.servlet.RequestDispatcher;
import jakarta.servlet.http.*;
```

```
import java.io.IOException;

public class OrderServlet extends HttpServlet {
   protected void doPost(jakarta.servlet.http.HttpServletRequest request,
                         jakarta.servlet.http.HttpServletResponse response)
         throws ServletException, IOException {

      HttpSession session = request.getSession(false);
      if (session == null || session.getAttribute("username") == null) {
         // not logged in
         response.sendRedirect("login.jsp");
         return;
      }
      String username = (String)session.getAttribute("username");
      int studentId = StudentDAO.getStudentId(username);

      try {
         int itemId = Integer.parseInt(request.getParameter("item_id"));
         int quantity = Integer.parseInt(request.getParameter("quantity"));
         double price = MenuDAO.getPrice(itemId);
         double total = price * quantity;

         boolean ok = OrderDAO.placeOrder(studentId, itemId, quantity, total);
         if (ok) {
            request.setAttribute("total", total);
            RequestDispatcher rd = request.getRequestDispatcher("order_success.jsp");
            rd.forward(request, response);
         } else {
```

```
            response.sendRedirect("menu?error=1");
          }
        } catch (Exception e) {
          e.printStackTrace();
          response.sendRedirect("menu?error=1");
        }
    }
}
```

---

3) web.xml (WEB-INF/web.xml)

```
<?xml version="1.0" encoding="UTF-8"?>
<web-app xmlns="https://jakarta.ee/xml/ns/jakartaee"
        xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
        xsi:schemaLocation="https://jakarta.ee/xml/ns/jakartaee
https://jakarta.ee/xml/ns/jakartaee/web-app_6_0.xsd"
        version="6.0">

    <servlet>
        <servlet-name>LoginServlet</servlet-name>
        <servlet-class>db.LoginServlet</servlet-class>
    </servlet>
    <servlet-mapping>
```

```
    <servlet-name>LoginServlet</servlet-name>
    <url-pattern>/login</url-pattern>
  </servlet-mapping>

  <servlet>
    <servlet-name>MenuServlet</servlet-name>
    <servlet-class>db.MenuServlet</servlet-class>
  </servlet>
  <servlet-mapping>
    <servlet-name>MenuServlet</servlet-name>
    <url-pattern>/menu</url-pattern>
  </servlet-mapping>

  <servlet>
    <servlet-name>OrderServlet</servlet-name>
    <servlet-class>db.OrderServlet</servlet-class>
  </servlet>
  <servlet-mapping>
    <servlet-name>OrderServlet</servlet-name>
    <url-pattern>/order</url-pattern>
  </servlet-mapping>

  <session-config>
    <session-timeout>30</session-timeout>
  </session-config>
</web-app>
```

---

4) JSPs / HTML (put under web/ or project root depending how you deploy)

index.html

```
<!DOCTYPE html>
<html>
<head><meta charset="utf-8"><title>Smart Canteen</title></head>
<body>
  <h1>Welcome to Smart Canteen</h1>
  <a href="login.jsp">Login</a>
</body>
</html>
```

---

login.jsp

```
<%@ page contentType="text/html;charset=UTF-8" language="java" %>
<html>
<head><title>Login</title></head>
<body>
<h2>Login</h2>
<form method="post" action="login">
   Username: <input type="text" name="username" required><br>
   Password: <input type="password" name="password" required><br>
```

```
  <input type="submit" value="Login">
</form>
<% if (request.getParameter("error") != null) { %>
  <p style="color:red;">Invalid username or password</p>
<% } %>
</body>
</html>
```

---

menu.jsp

```
<%@ page import="java.util.List" %>
<%@ page import="db.MenuDAO" %>
<%@ page contentType="text/html;charset=UTF-8" language="java" %>
<html>
<head><title>Menu</title></head>
<body>
<h2>Menu</h2>
<a href="index.html">Home</a>
<br><br>
<form method="post" action="order">
  <table border="1" cellpadding="5">
    <tr><th>ID</th><th>Name</th><th>Price</th><th>Available</th></tr>
    <%
      List<MenuDAO.Item> items = (List<MenuDAO.Item>) request.getAttribute("items");
      if (items != null) {
```

```
            for (MenuDAO.Item it : items) {
        %>
                <tr>
                  <td><%=it.id %></td>
                  <td><%=it.name %></td>
                  <td><%=String.format("%.2f", it.price) %></td>
                  <td><%= it.available ? "Yes":"No" %></td>
                </tr>
        <%
            }
          } else {
        %>
          <tr><td colspan="4">No items found.</td></tr>
        <%
          }
        %>
      </table>
      <br>
      Enter Item ID to order: <input type="number" name="item_id" required>
      Quantity: <input type="number" name="quantity" value="1" min="1" required>
      <input type="submit" value="Place Order">
    </form>
    </body>
    </html>
```

---

order_success.jsp

```
<%@ page contentType="text/html;charset=UTF-8" language="java" %>
<html>
<head><title>Order Placed</title></head>
<body>
<h2>Order Successful</h2>
<p>Your order has been placed.</p>
<p>Total amount: Rs. <%= request.getAttribute("total") %></p>
<a href="menu">Back to Menu</a>
</body>
</html>
```

---

# CHAPTER 6 – SCREENSHOTS

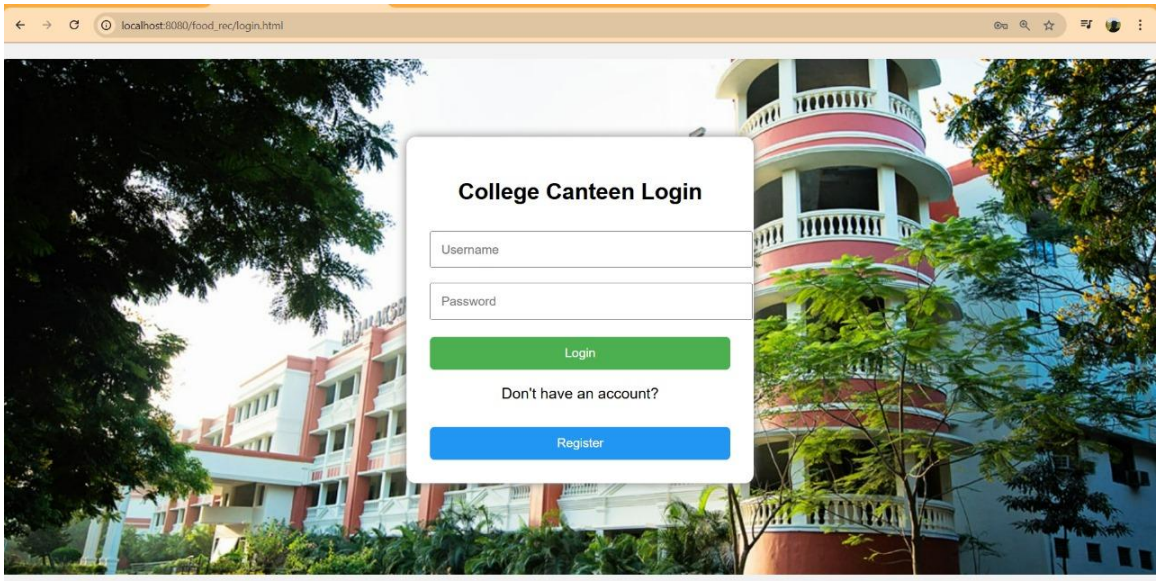


Fig 5.1 Login Page (Placeholder)

Fig 5.1 describes the login page which created with the help of CSS ans HTML . This page helps to login into the web if they have an existing account. If they did't have an account, no worries . They can able to create a new account.

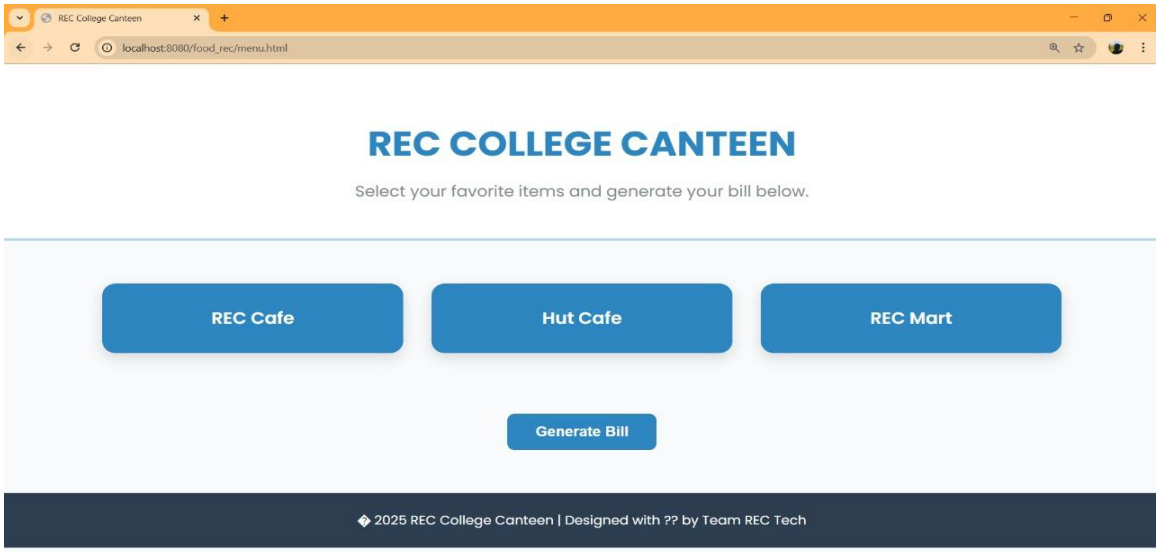


Fig 5.2 Product List Page (Placeholder)

Fig 5.2 shows the different canteen in our college like REC Cafe, Hut Cafe, and REC Mart. We can able to put order in this page

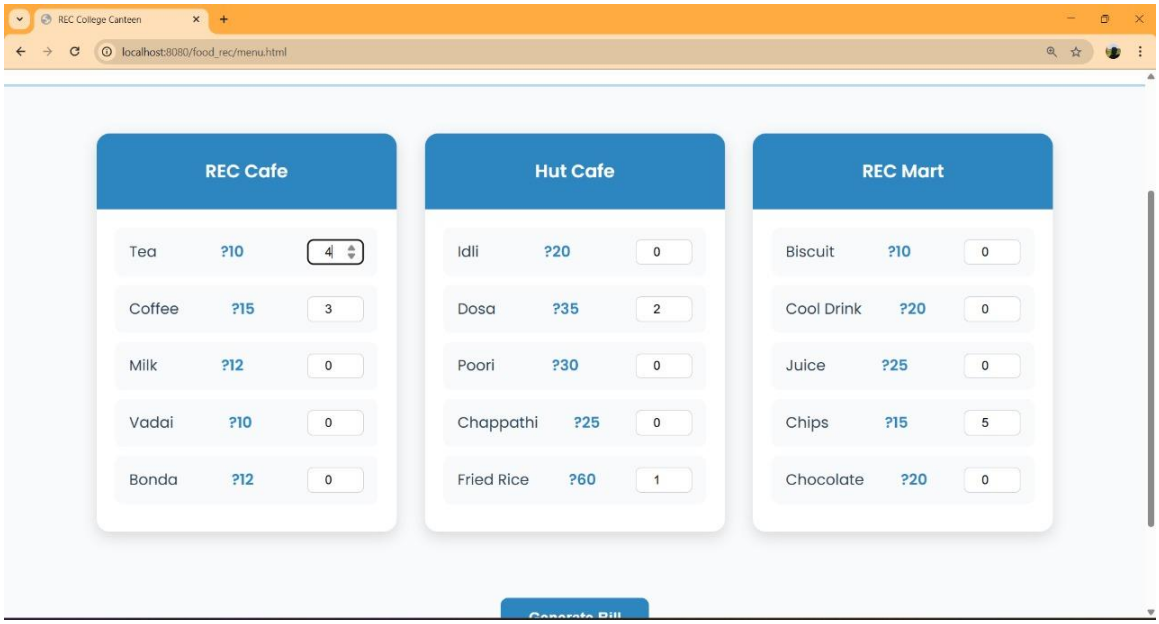


Fig 5.3 Order Placement Page (Placeholder)

Fig 5.3 shows the different types of foods available in the canteen which can able to select the foods and put the order

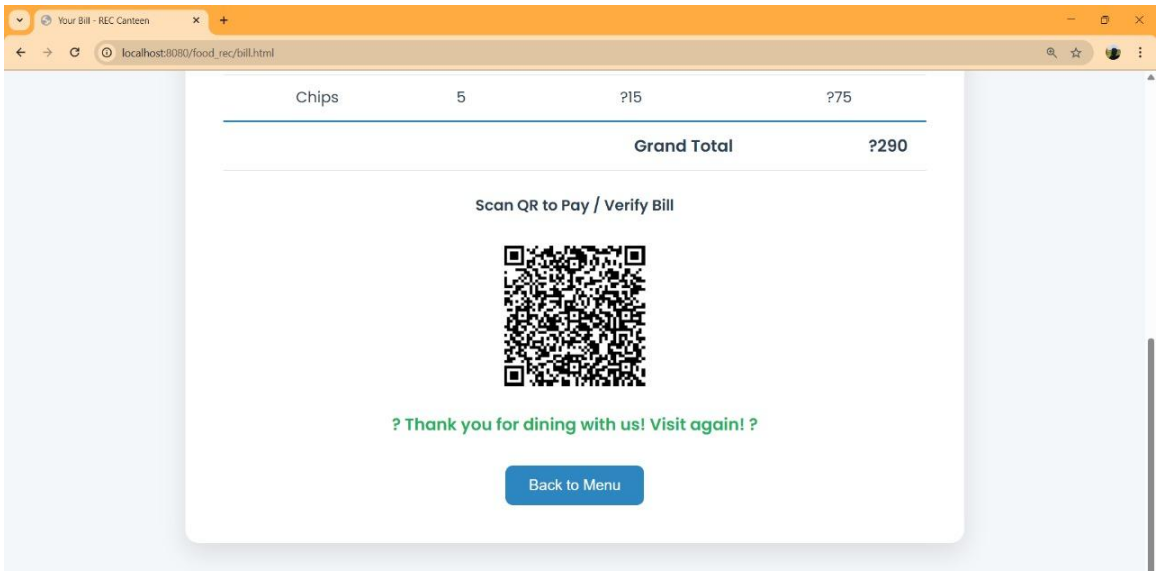


Fig 5.4 Bill payment  Page (Placeholder)

Fig 5.4 shows the Total price and QR Code to scan and proceed to the payment option which was not integreted in this project . But we can able to insert in the future plans.

# CHAPTER 7

# CONCLUSION AND FUTURE ENHANCEMENT

## 7.1 Conclusion:

The **Smart Canteen Management System** is an efficient and reliable solution designed to automate and simplify the operations of an institutional canteen. It integrates essential functions such as user authentication, menu management, order placement, and billing within a single platform. Developed using **Java, Servlets, JSP, and MySQL**, the system ensures accurate data processing, secure storage, and a user-friendly experience for both administrators and users.

The use of **three-tier architecture** enhances modularity, scalability, and maintainability, making the system adaptable for future growth. This project also provided valuable hands-on experience in **database connectivity (JDBC)**, **web development (JSP/Servlets)**, and **object-oriented programming principles**. Overall, the system minimizes manual effort, reduces errors, and improves the overall efficiency and transparency of canteen operations.

## 7.2 Future Enhancement:

The current system can be enhanced in the future with the following features:

1. **Online Ordering System** – Develop a web or mobile version to enable users to place orders remotely.
2. **Digital Payment Integration** – Incorporate secure online payment methods such as UPI, debit/credit cards, or wallets.
3. **Real-Time Order Tracking** – Allow users to track their order status from preparation to delivery.
4. **Automated Notifications** – Send email or SMS alerts for order confirmation and billing updates.
5. **Data Analytics Dashboard** – Include visual reports to analyze daily sales, popular items, and customer preferences.
6. **Cloud Database Support** – Migrate to cloud storage for improved scalability, data security, and remote access.

These enhancements will help transform the **Smart Canteen Management System** into a complete digital canteen solution capable of supporting real-time, secure, and user-friendly operations.

# REFERENCES


1. https://www.w3schools.com/java/
2. https://www.tutorialspoint.com/java/
3. https://www.geeksforgeeks.org/java/
4. https://www.oracle.com/java/technologies/
5. https://www.github.com/