**EX.NO: 11 Restaurant Management System**

**DATE:**

**Program:**

package restaurantmanagementapp;

import javax.swing.\*;

import java.awt.\*;

import java.awt.event.\*;

import java.sql.\*;

import javax.swing.table.DefaultTableModel;

public class RestaurantManagementApp extends JFrame implements ActionListener {

JTextField dishNameField, dishPriceField, dishAvailabilityField;

JTextArea dishDescriptionArea;

JButton addButton, updateButton, clearButton;

JTable dishTable;

DefaultTableModel tableModel;

public RestaurantManagementApp() {

// Creating the GUI components

JLabel nameLabel = new JLabel("Dish Name:");

JLabel descriptionLabel = new JLabel("Description:");

JLabel priceLabel = new JLabel("Price:");

JLabel availabilityLabel = new JLabel("Availability:");

dishNameField = new JTextField(15);

dishPriceField = new JTextField(15);

dishAvailabilityField = new JTextField(15);

dishDescriptionArea = new JTextArea(3, 15);

dishDescriptionArea.setLineWrap(true);

dishDescriptionArea.setWrapStyleWord(true);

addButton = new JButton("Add Dish");

updateButton = new JButton("Update Dish");

clearButton = new JButton("Clear");

addButton.addActionListener(this);

updateButton.addActionListener(this);

clearButton.addActionListener(this);

// Creating a Table

tableModel = new DefaultTableModel(new String[]{"Dish Name", "Description", "Price", "Availability"}, 0);

dishTable = new JTable(tableModel);

JScrollPane tableScrollPane = new JScrollPane(dishTable);

// Set Layout for the frame

setLayout(new GridLayout(9, 2));

add(nameLabel);

add(dishNameField);

add(descriptionLabel);

add(new JScrollPane(dishDescriptionArea));

add(priceLabel);

add(dishPriceField);

add(availabilityLabel);

add(dishAvailabilityField);

add(addButton);

add(updateButton);

add(clearButton);

add(tableScrollPane);

setTitle("Restaurant Management System");

setSize(800, 600);

setDefaultCloseOperation(JFrame.EXIT\_ON\_CLOSE);

setVisible(true);

// Load existing dishes

displayDishes();

}

// Database Connection

private Connection connect() {

try {

Class.forName("com.mysql.cj.jdbc.Driver");

return DriverManager.getConnection("jdbc:mysql://localhost:3306/restaurant\_management?useSSL=false&serverTimezone=UTC", "root", "dbms123");

} catch (Exception e) {

JOptionPane.showMessageDialog(this, "Database Connection Error");

e.printStackTrace();

return null;

}

}

// Action handling for buttons

public void actionPerformed(ActionEvent e) {

if (e.getSource() == addButton) {

addDish();

} else if (e.getSource() == updateButton) {

updateDish();

} else if (e.getSource() == clearButton) {

clearFields();

}

}

// Add dish data to the database

private void addDish() {

String dishName = dishNameField.getText();

String description = dishDescriptionArea.getText();

String price = dishPriceField.getText();

String availability = dishAvailabilityField.getText();

if (dishName.isEmpty() || price.isEmpty() || availability.isEmpty()) {

JOptionPane.showMessageDialog(this, "Please fill in all fields");

return;

}

try {

Double.parseDouble(price);

} catch (NumberFormatException ex) {

JOptionPane.showMessageDialog(this, "Price must be a number");

return;

}

Connection conn = connect();

if (conn != null) {

try {

String query = "INSERT INTO Dishes (dish\_name, description, price, availability) VALUES (?, ?, ?, ?)";

PreparedStatement stmt = conn.prepareStatement(query);

stmt.setString(1, dishName);

stmt.setString(2, description);

stmt.setString(3, price);

stmt.setString(4, availability);

stmt.executeUpdate();

JOptionPane.showMessageDialog(this, "Dish added successfully");

stmt.close();

conn.close();

clearFields();

displayDishes(); // Reload the table

} catch (Exception ex) {

ex.printStackTrace();

JOptionPane.showMessageDialog(this, "Error adding dish");

}

}

}

// Update dish data in the database

private void updateDish() {

int selectedRow = dishTable.getSelectedRow();

if (selectedRow == -1) {

JOptionPane.showMessageDialog(this, "Please select a dish to update");

return;

}

String dishName = dishNameField.getText();

String description = dishDescriptionArea.getText();

String price = dishPriceField.getText();

String availability = dishAvailabilityField.getText();

if (dishName.isEmpty() || price.isEmpty() || availability.isEmpty()) {

JOptionPane.showMessageDialog(this, "Please fill in all fields");

return;

}

Connection conn = connect();

if (conn != null) {

try {

String query = "UPDATE Dishes SET description=?, price=?, availability=? WHERE dish\_name=?";

PreparedStatement stmt = conn.prepareStatement(query);

stmt.setString(1, description);

stmt.setString(2, price);

stmt.setString(3, availability);

stmt.setString(4, dishName);

stmt.executeUpdate();

JOptionPane.showMessageDialog(this, "Dish updated successfully");

stmt.close();

conn.close();

displayDishes(); // Reload the table

} catch (Exception ex) {

ex.printStackTrace();

JOptionPane.showMessageDialog(this, "Error updating dish");

}

}

}

// Clear all input fields

private void clearFields() {

dishNameField.setText("");

dishDescriptionArea.setText("");

dishPriceField.setText("");

dishAvailabilityField.setText("");

}

// Display dish data from the database

private void displayDishes() {

Connection conn = connect();

if (conn != null) {

try {

String query = "SELECT \* FROM Dishes";

Statement stmt = conn.createStatement();

ResultSet rs = stmt.executeQuery(query);

// Clear existing rows

tableModel.setRowCount(0);

while (rs.next()) {

String name = rs.getString("dish\_name");

String description = rs.getString("description");

String price = rs.getString("price");

String availability = rs.getString("availability");

tableModel.addRow(new Object[]{name, description, price, availability});

}

rs.close();

stmt.close();

conn.close();

} catch (Exception ex) {

ex.printStackTrace();

JOptionPane.showMessageDialog(this, "Error displaying dishes");

}

}

}

// Main method to run the application

public static void main(String[] args) {

new RestaurantManagementApp();

}

}

**Output:**



