

RECIPE GENERATOR

A MINI-PROJECT BY:

AISHWARYA A 230701015

AKSHITHAA H 230701025

in partial fulfilment of the award of the degree

OF

BACHELOR OF ENGINEERING

IN

COMPUTER SCIENCE AND ENGINEERING



RAJALAKSHMI ENGINEERING COLLEGE, CHENNAI

An Autonomous Institute

CHENNAI

NOVEMBER 2024

BONAFIDE CERTIFICATE

It is hereby certified that the project titled “**Recipe Generator**” is a genuine work carried out by **Aishwarya A (230701015)** and **Akshithaa H (230701025)**, 2nd-year students of the **Department of Computer Science and Engineering** for **CS23333-Object Oriented Programming using JAVA** in the year **2024-2025**.

This project was undertaken as part of the academic requirements under the guidance and supervision of the undersigned.

Academic Year : 2024-2025 Semester: IIIrd sem Branch : CSE

FACULTY SIGNATURE
MRS B.DEEPA

ABSTRACT

A **Recipe Generator** is an application designed to simplify meal planning and cooking by offering recipe suggestions based on user preferences and available ingredients. This project serves as a practical implementation of database and programming concepts, combining a user-friendly interface with a robust backend system.

Our project aims to assist users in selecting and generating customized recipes effortlessly. Users can register on the website, specify their meal preferences (such as type of meal, course, dietary requirements, and available ingredients), and submit their choices. The system processes the input, queries the database, and provides a relevant recipe for the user.

The application also allows users to store and manage their data, enabling them to review and reuse recipes when needed. This project demonstrates the ability to create a seamless and dynamic connection between a Java-based front-end and a MySQL-powered backend, offering practical benefits in everyday life while showcasing technical proficiency.

TABLE OF CONTENTS

1. INTRODUCTION

- 1.1 INTRODUCTION
- 1.2 IMPLEMENTATION
- 1.3 SCOPE OF THE PROJECT
- 1.4 WEBSIT FEATURES
- 1.5 SYSTEM DESGIN AND ARCHITECTURE
- 1.6 IMPLEMENTATION DETAILS

2. SYSTEM SPECIFICATION

- 2.1 HARDWARE SPECIFICATION
- 2.2 SOFTWARE SPECIFICATION

3. SAMPLE CODE

- 3.1 HOME PAGE DESIGN
- 3.2 REGISTRATION PAGE DESIGN
- 3.3 LOGIN PAGE DESIGN
- 3.4 DASHBOARD DESIGN
- 3.5 REGISTRATION BACKEND
- 3.6 LOGIN PAGE BACKEND
- 3.7 REMOVE STUDENT OPERATION IN
DASHBOARD 3.8 UPDATE DATA
OPERATION IN DASHBOARD

4. SNAPSHOTS

- 4.1 HOME PAGE
- 4.2 STUDENT REGISTRATION PAGE
- 4.3 GUIDE REGISTRATION PAGE
- 4.4 LOGIN PAGE
- 4.5 DASHBOARD
- 4.6 REMOVE STUDENT OPERATION
- 4.7 UPDATE DATA OPERATION

5. CONCLUSION

6. REFERENCES

INTRODUCTION

1.1 INTRODUCTION

This project report details the development of a recipe generator application, designed to assist users in discovering new recipes based on their dietary needs and available ingredients. In today's fast-paced world, finding suitable recipes that match one's available resources and preferences can be challenging. The recipe generator leverages both database management (MySQL) and object-oriented programming (Java) to streamline the recipe suggestion process, thereby making cooking more accessible and enjoyable.

The primary goal is to allow users to filter recipes by meal type, course type, dietary preference, and available ingredients, and then retrieve a recipe from a database designed specifically for this purpose. This application showcases how technology can be employed to make everyday tasks, such as cooking, more efficient and engaging.

1.2 IMPLEMENTATION

The **RECIPE GENERATOR** project discussed here is implemented using the concepts of **JAVA** and **MYSQL**.

1.3 SCOPE OF THE PROJECT

The website is designed to simplify and enhance the process of generating personalized recipes. Users will be required to register by creating an account, enabling them to securely store their preferences, ingredient lists, and generated recipes in one centralized location. This ensures an organized and efficient experience, saving users valuable time.

By offering a user-friendly and professional interface, the project aims to provide seamless access to tailored recipe suggestions while demonstrating practical applications of database and programming concepts. This platform is scalable, with potential future integrations such as meal planning, nutritional analysis, and shopping list generation, further enhancing user convenience.

1.4 WEBSITE FEATURES

- Registration page.
- Custom profile for each user.
- Meal option page.
- Ingredient Selection page.
- Recipe Generation Feature.

1.5 System Design and Architecture

➤ Database Design

- The database schema includes tables such as Recipes, Ingredients, MealTypes, Courses, and DietaryPreferences.
- The Recipes table contains essential fields such as recipe name, instructions, meal type, course, dietary preference, and associated ingredients.
- A many-to-many relationship exists between recipes and ingredients, implemented through a join table Recipe_Ingredients.

➤ User Interface Design

- A simple and user-friendly interface is designed using Java Swing/JavaFX.
- Users can select from drop-down menus for meal type, course, and dietary preference, and input available ingredients.
- The interface fetches data from the database based on the user's selection criteria and displays the most suitable recipe.

1.6 Implementation Details

➤ Front-end (Java)

- Java is used for developing the GUI, making it easy for users to interact with the application.
- Classes for handling user input, database queries, and data display are created using object-oriented principles.

➤ Back-end (MySQL)

- MySQL stores the recipe data and enables efficient querying based on user preferences.

- SQL queries are dynamically generated to fetch recipes that match the selected meal type, course, dietary preference, and ingredients.
- Testing and Evaluation
The application underwent rigorous testing to ensure smooth user interaction and accurate recipe recommendations. Tests were conducted for:
 - Data retrieval based on different combinations of meal type, course, and ingredients.
 - Error handling, especially in cases where no matching recipe is available.
 - GUI functionality and responsiveness.
 - Results showed that the system effectively retrieves recipes matching user specifications and provides alternate suggestions if needed.

SYSTEM SPECIFICATIONS

2.1 HARDWARE SPECIFICATIONS:

PROCESSOR : Intel i7

MEMORY SIZE : 4GB(Minimum)

STORAGE: 10GB minimum

OPERATING SYSTEM: Windows

2.2 SOFTWARE SPECIFICATIONS:

PROGRAMMING LANGUAGE : Java, MySQL

FRONT-END : Java

BACK-END : MySQL

OPERATING SYSTEM : Windows 10

SAMPLE CODE

3.1 LOGIN/REGISTRATION PAGE DESIGN

```
import java.awt.*;
import java.awt.event.ActionEvent;
import java.awt.event.ActionListener;
import java.sql.Connection;
import java.sql.DriverManager;
import java.sql.PreparedStatement;
import java.sql.SQLException;
import javax.swing.*;

public class LoginFrame extends JFrame implements ActionListener {

    // Declare components
    Container container = getContentPane();
    JLabel userLabel = new JLabel("USERNAME");
    JLabel passwordLabel = new JLabel("PASSWORD");
    JTextField userTextField = new JTextField();
    JPasswordField passwordField = new JPasswordField();
    JButton loginButton = new JButton("SAVE");
    JButton resetButton = new JButton("RESET");
    JCheckBox showPassword = new JCheckBox("Show Password");
    JButton nextButton = new JButton("NEXT"); // New "Next" button
    GridBagConstraints gbc = new GridBagConstraints();

    LoginFrame() {
        setLayoutManager();
        setLocationAndSize();
        addActionEvent();
        gbc.gridx = 1;
        gbc.gridy = 4; // Place it below the other buttons
        nextButton.setVisible(false); // Hide initially
        container.add(nextButton, gbc);
    }

    public void setLayoutManager() {
        // Use GridBagLayout for more control over positioning
        container.setLayout(new GridBagLayout());
    }

    public void setLocationAndSize() {
        // Set GridBag constraints for layout
        gbc.insets = new Insets(10, 10, 10, 10);

        // Set up user label and text field
```



```

        gbc.gridx = 0;
        gbc.gridy = 0;
        container.add(userLabel, gbc); // Add userLabel at position (0,0)

        gbc.gridx = 1;
        gbc.gridy = 0;
        gbc.gridwidth = 2; // Make the user text field span two columns for
more space
        userTextField.setPreferredSize(new Dimension(200, 30)); // Set a
larger width for the username field
        container.add(userTextField, gbc); // Add the username text field at
position (1,0)

        // Set up password label and password field
        gbc.gridx = 0;
        gbc.gridy = 1;
        gbc.gridwidth = 1; // Reset to 1 column width for password label
        container.add(passwordLabel, gbc); // Add passwordLabel at position
(0,1)

        gbc.gridx = 1;
        gbc.gridy = 1;
        gbc.gridwidth = 2; // Make password field span two columns
        passwordField.setPreferredSize(new Dimension(200, 30)); // Set a
larger width for the password field
        container.add(passwordField, gbc); // Add the password field at
position (1,1)

        // Add show password checkbox
        gbc.gridx = 1;
        gbc.gridy = 2;
        container.add(showPassword, gbc); // Add showPassword checkbox at
position (1,2)

        // Set up buttons
        gbc.gridx = 0;
        gbc.gridy = 3;
        gbc.gridwidth = 1; // Reset to 1 column width for login button
        container.add(loginButton, gbc); // Add loginButton at position (0,3)

        gbc.gridx = 1;
        gbc.gridy = 3;
        container.add(resetButton, gbc); // Add resetButton at position (1,3)

        gbc.gridx = 1;
        gbc.gridy = 4;
        container.add(nextButton, gbc);
    }

```

```

public void addActionEvent() {
    loginButton.addActionListener(this);
    resetButton.addActionListener(this);
    showPassword.addActionListener(this);
}

@Override
public void actionPerformed(ActionEvent e) {
    if (e.getSource() == loginButton) {
        String userText;
        String pwdText;
        userText = userTextField.getText();
        pwdText = passwordField.getText();
        JOptionPane.showMessageDialog(this, "Login Successful");
        BlankFrame blankFrame = new BlankFrame();
        blankFrame.setVisible(true);
        this.setVisible(false);
    }
    if (e.getSource() == resetButton) {
        userTextField.setText("");
        passwordField.setText("");
    }
    if (e.getSource() == showPassword) {
        if (showPassword.isSelected()) {
            passwordField.setEchoChar((char) 0);
        } else {
            passwordField.setEchoChar('*');
        }
    }
}

}

private boolean registerUser(String username, String password) {
    String url = "jdbc:mysql://localhost:3306/project_java_dbms";
    // Replace with your actual database name
    String dbUser = "root"; // Replace with your MySQL username
    String dbPassword = "1234"; // Replace with your MySQL password

    try {
        // Load MySQL JDBC driver
        Class.forName("com.mysql.cj.jdbc.Driver");
        System.out.println("JDBC Driver loaded.");

        // Try to establish the connection
    }
}

```

```

        Connection conn = DriverManager.getConnection(url, dbUser,
dbPassword);
        System.out.println("Connection successful.");

        // Generate salt and hash the password
        String salt = PasswordUtils.generateSalt();
        String hashedPassword = PasswordUtils.hashPassword(password,
salt);

        // SQL query to insert user details
        String sql = "INSERT INTO users (username, password_hash, salt)
VALUES (?, ?, ?)";
        try (PreparedStatement stmt = conn.prepareStatement(sql)) {
            stmt.setString(1, username);
            stmt.setString(2, hashedPassword);
            stmt.setString(3, salt);
            stmt.executeUpdate();
            System.out.println("User saved successfully.");
            return true;
        }
        } catch (ClassNotFoundException e) {
            JOptionPane.showMessageDialog(this, "Error: JDBC Driver not
found.\n" + e.getMessage(), "Error", JOptionPane.ERROR_MESSAGE);
        } catch (SQLException e) {
            JOptionPane.showMessageDialog(this, "SQL Error: " + e.getMessage()
+ "\nSQLState: " + e.getSQLState() + "\nErrorCode: " + e.getErrorCode(),
"Database Error", JOptionPane.ERROR_MESSAGE);
        } catch (Exception e) {
            JOptionPane.showMessageDialog(this, "Unexpected error: " +
e.getMessage(), "Error", JOptionPane.ERROR_MESSAGE);
        }

        return false;
    }

    public static void main(String[] args) {
        LoginFrame loginFrame = new LoginFrame();
        loginFrame.setTitle("User Registration");
        loginFrame.setSize(400, 400);
        loginFrame.setDefaultCloseOperation(JFrame.EXIT_ON_CLOSE);
        loginFrame.setVisible(true);
    }
}

```

3.2 REGISTRATION PAGE DESIGN

```
import javax.swing.*;

public class Rgistration {
    public static void main(String[] a) {
        LoginFrame frame = new LoginFrame();
        frame.setTitle("Login Form");
        frame.setVisible(true);
        frame.setBounds(10, 10, 370, 600);
        frame.setDefaultCloseOperation(JFrame.EXIT_ON_CLOSE);
        frame.setResizable(false);
    }
}
```

3.3 HOME PAGE DESIGN:

```
import javax.swing.*;

public class HomeScreen extends JFrame {
    public HomeScreen() {
        setTitle("Home Screen");
        setSize(400, 400);
        setDefaultCloseOperation(JFrame.EXIT_ON_CLOSE);

        JLabel welcomeLabel = new JLabel("Welcome to the Home Screen!");
        add(welcomeLabel);

        setLocationRelativeTo(null); // Center the window
        setVisible(true);
    }
}
```

3.4 LOGIN PAGE BACKEND

```
import java.security.MessageDigest;
import java.security.NoSuchAlgorithmException;
import java.util.Base64;

public class PasswordUtils {
    public static String generateSalt() {
        return "some_random_salt_value"; // You should ideally generate a
        random salt
    }

    public static String hashPassword(String password, String salt) {
        try {
            MessageDigest digest = MessageDigest.getInstance("SHA-256");
            String saltedPassword = password + salt;
        }
    }
}
```

```

        byte[] hash = digest.digest(saltedPassword.getBytes());
        return Base64.getEncoder().encodeToString(hash);
    } catch (NoSuchAlgorithmException e) {
        e.printStackTrace();
        return null;
    }
}
}

```

3.7 DATABASE BACKEND

```

import java.sql.Connection;

import java.sql.DriverManager;
import java.sql.SQLException;

public class DBConnection {

    // Method to get the connection
    public static Connection getConnection() {
        // Database URL, username, and password
        String url = "jdbc:mysql://localhost:3306/your_database_name"; //
        Replace with your database name
        String user = "root"; // Replace with your MySQL username
        String password = "your_password"; // Replace with your MySQL password

        try {
            // Load and register the JDBC driver (if not using newer versions)
            Class.forName("com.mysql.cj.jdbc.Driver");

            // Return the connection object
            return DriverManager.getConnection(url, user, password);

        } catch (ClassNotFoundException e) {
            System.out.println("MySQL JDBC Driver not found.");
        } catch (SQLException e) {
            System.out.println("Connection failed. Check your database URL,
            username, and password.");
        }

        return null; // Return null if connection fails
    }
}

```

3.5 MEAL CHOOSER PAGE DESGIN WITH TOGGLE BAR

```
import java.awt.*;
import java.awt.event.*;
import javax.swing.*;

public class BlankFrame extends JFrame implements ActionListener {
    // Buttons for the recipes
    JButton simpleButton1, simpleButton2, simpleButton3;
    // Panel for the drawer
    JPanel drawerPanel;
    // Buttons inside the drawer
    JButton savedRecipesButton, favoriteRecipesButton, recipesThisWeekButton,
    newRecipeButton;
    // For the slide in/out animation
    boolean drawerOpen = false;

    BlankFrame() {
        // Initialize frame settings
        setTitle("Recipe manager");
        setLayout(null); // Using absolute positioning for components
        setSize(650, 600);
        setDefaultCloseOperation(JFrame.EXIT_ON_CLOSE);

        // Load images for the buttons
        ImageIcon buttonIcon1 = new
ImageIcon("C:\\Users\\aaish\\OneDrive\\Desktop\\Project_Java_dbms\\src\\hand-
drawn-caldo-verde-illustration\\8840235.jpg");
        ImageIcon buttonIcon2 = new
ImageIcon("C:\\Users\\aaish\\OneDrive\\Desktop\\Project_Java_dbms\\src\\hand-
drawn-salchipapa-illustration\\4783666.jpg");
        ImageIcon buttonIcon3 = new
ImageIcon("C:\\Users\\aaish\\OneDrive\\Desktop\\Project_Java_dbms\\src\\hand-
drawn-tequenos-illustration\\4831379.jpg");

        // Scale the button icons to fit the button size
        buttonIcon1 = new
ImageIcon(buttonIcon1.getImage().getScaledInstance(250, 100,
Image.SCALE_SMOOTH));
        buttonIcon2 = new
ImageIcon(buttonIcon2.getImage().getScaledInstance(250, 100,
Image.SCALE_SMOOTH));
        buttonIcon3 = new
ImageIcon(buttonIcon3.getImage().getScaledInstance(250, 100,
Image.SCALE_SMOOTH));

        // Initialize the buttons (3 recipe buttons)
        simpleButton1 = new JButton("BREAKFAST");
```

```
simpleButton1.setBounds(350, 100, 250, 100);
simpleButton1.addActionListener(this);
simpleButton1.setFocusable(false);
simpleButton1.setIcon(buttonIcon1);
simpleButton1.setHorizontalTextPosition(JButton.CENTER);
simpleButton1.setVerticalTextPosition(JButton.CENTER);
simpleButton1.setFont(new Font("Comic Sans", Font.BOLD, 25));
simpleButton1.setIconTextGap(0);
simpleButton1.setForeground(Color.cyan);
simpleButton1.setBackground(Color.lightGray);
simpleButton1.setBorder(BorderFactory.createEtchedBorder());

simpleButton2 = new JButton("LUNCH");
simpleButton2.setBounds(350, 220, 250, 100);
simpleButton2.addActionListener(this);
simpleButton2.setFocusable(false);
simpleButton2.setIcon(buttonIcon2);
simpleButton2.setHorizontalTextPosition(JButton.CENTER);
simpleButton2.setVerticalTextPosition(JButton.CENTER);
simpleButton2.setFont(new Font("Comic Sans", Font.BOLD, 25));
simpleButton2.setIconTextGap(0);
simpleButton2.setForeground(Color.cyan);
simpleButton2.setBackground(Color.lightGray);
simpleButton2.setBorder(BorderFactory.createEtchedBorder());

simpleButton3 = new JButton("DINNER");
simpleButton3.setBounds(350, 340, 250, 100);
simpleButton3.addActionListener(this);
simpleButton3.setFocusable(false);
simpleButton3.setIcon(buttonIcon3);
simpleButton3.setHorizontalTextPosition(JButton.CENTER);
simpleButton3.setVerticalTextPosition(JButton.CENTER);
simpleButton3.setFont(new Font("Comic Sans", Font.BOLD, 25));
simpleButton3.setIconTextGap(0);
simpleButton3.setForeground(Color.cyan);
simpleButton3.setBackground(Color.lightGray);
simpleButton3.setBorder(BorderFactory.createEtchedBorder());

// Add buttons to the frame
add(simpleButton1);
add(simpleButton2);
add(simpleButton3);

// Initialize the drawer panel
drawerPanel = new JPanel();
drawerPanel.setBackground(new Color(90, 90, 90));
drawerPanel.setLayout(new BoxLayout(drawerPanel, BoxLayout.Y_AXIS));
```

```

        drawerPanel.setBounds(-250, 0, 250, getHeight()); // Hidden initially
(off-screen)

        // Add menu items to the drawer
        savedRecipesButton = new JButton("Saved Recipes");
        favoriteRecipesButton = new JButton("Favorite Recipes");
        recipesThisWeekButton = new JButton("Recipes Used This Week");
        newRecipeButton = new JButton("New Recipe");

        // Customize buttons in the drawer
        savedRecipesButton.setBackground(new Color(100, 100, 100));
        favoriteRecipesButton.setBackground(new Color(100, 100, 100));
        recipesThisWeekButton.setBackground(new Color(100, 100, 100));
        newRecipeButton.setBackground(new Color(100, 100, 100));

        // Add buttons to the drawer panel
        drawerPanel.add(savedRecipesButton);
        drawerPanel.add(favoriteRecipesButton);
        drawerPanel.add(recipesThisWeekButton);
        drawerPanel.add(newRecipeButton); // New recipe button added
        // Set the layout to BoxLayout with Y_AXIS to arrange buttons
vertically
        drawerPanel.setLayout(new BoxLayout(drawerPanel, BoxLayout.Y_AXIS));

        // Make each button fill the entire width of the drawer and adjust the
height accordingly
        savedRecipesButton.setMaximumSize(new Dimension(250, 50)); // Width =
250px (same as drawer width), Height = auto
        favoriteRecipesButton.setMaximumSize(new Dimension(250, 50));
        recipesThisWeekButton.setMaximumSize(new Dimension(250, 50));
        newRecipeButton.setMaximumSize(new Dimension(250, 50));

        // Add the drawer to the frame
        add(drawerPanel);

        // Drawer toggle button (placed on the left side)
        JButton toggleDrawerButton = new JButton("Toggle Drawer");
        toggleDrawerButton.setBounds(10, 20, 130, 30); // Positioned to the
left side
        toggleDrawerButton.addActionListener(e ->
toggleDrawer(toggleDrawerButton)); // Pass button to toggle visibility
        add(toggleDrawerButton);

        // Add a MouseListener to close the drawer when clicking outside
        this.addMouseListener(new java.awt.event.MouseAdapter() {
            @Override
            public void mousePressed(java.awt.event.MouseEvent evt) {

```



```

        // Check if the drawer is open and the click is outside the
drawer
        if (drawerOpen &&
!drawerPanel.getBounds().contains(evt.getPoint())) {
            // Close the drawer if the click is outside
            toggleDrawer(toggleDrawerButton); // This will also update
the toggle button visibility
        }
    }
});

// Make sure the frame is visible
setVisible(true);
}

// Action for the 3 buttons (recipe buttons)
@Override
public void actionPerformed(ActionEvent e) {
    if (e.getSource() == simpleButton1) {
        System.out.println("Button 1 clicked");
        simpleButton1.setEnabled(false);
        new blank2nd();
    } else if (e.getSource() == simpleButton2) {
        System.out.println("Button 2 clicked");
        simpleButton2.setEnabled(false);
        new blank2nd();
    } else if (e.getSource() == simpleButton3) {
        System.out.println("Button 3 clicked");
        simpleButton3.setEnabled(false);
        new blank2nd();
    }
    // Add action for new recipe button
    if (e.getSource() == newRecipeButton) {
        System.out.println("New Recipe button clicked");
        // Implement your logic for adding a new recipe here
    }
}

// Toggle the drawer panel
private void toggleDrawer(JButton toggleDrawerButton) {
    if (drawerOpen) {
        // Slide the drawer out
        drawerPanel.setBounds(-250, 0, 250, getHeight());
        toggleDrawerButton.setVisible(true); // Make the toggle button
visible again when the drawer closes
    } else {
        // Slide the drawer in
        drawerPanel.setBounds(0, 0, 250, getHeight());
    }
}

```

```

        toggleDrawerButton.setVisible(false); // Hide the button when the
drawer is open
    }
    drawerOpen = !drawerOpen;
}

public static void main(String[] args) {
    new BlankFrame(); // Create and show the frame
}
}

```

3.5 MEAL CHOOSER PAGE 2 DESGIN

```

import java.awt.*;
import java.awt.event.ActionEvent;
import java.awt.event.ActionListener;
import javax.swing.*;

public class blank2nd extends JFrame {

    private JButton simpleButton1;
    private JButton simpleButton2;
    private JButton simpleButton3;
    private Icon buttonIcon1, buttonIcon2, buttonIcon3; // Placeholder for
icons

    public blank2nd() {
        // Set up the frame for the blank page
        setTitle("Blank Page");
        setSize(650, 600);
        setDefaultCloseOperation(JFrame.EXIT_ON_CLOSE);
        setLayout(null); // Use null layout for absolute positioning

        // Create a panel for two buttons at the top
        JPanel buttonPanel = new JPanel();
        buttonPanel.setLayout(new FlowLayout(FlowLayout.CENTER, 30, 20)); //
Centered alignment with gaps
        buttonPanel.setBounds(0, 0, 650, 80); // Position the panel at the top

        // Create the two top buttons
        JButton button1 = new JButton("MODE: Non-Vegeterian");
        JButton button2 = new JButton("MODE: Vegeterian");

        button1.setForeground(Color.RED);
        button2.setForeground(new Color(0, 128, 0));
        button1.setBackground(Color.LIGHT_GRAY);
        button2.setBackground(Color.LIGHT_GRAY);
    }
}

```

```

// Add buttons to the panel
buttonPanel.add(button1);
buttonPanel.add(button2);

// Add the panel to the frame
add(buttonPanel);

// Initialize the "APPETIZER" button
simpleButton1 = new JButton("APPETIZER");
simpleButton1.setBounds(200, 120, 250, 100);
simpleButton1.setIcon(buttonIcon1);
simpleButton1.setHorizontalTextPosition(JButton.CENTER);
simpleButton1.setVerticalTextPosition(JButton.CENTER);
simpleButton1.setFont(new Font("Comic Sans", Font.BOLD, 25));
simpleButton1.setIconTextGap(0);
simpleButton1.setForeground(Color.cyan);
simpleButton1.setBackground(Color.lightGray);
simpleButton1.setBorder(BorderFactory.createEtchedBorder());

// Add an ActionListener to open Blank3 on click
simpleButton1.addActionListener(new ActionListener() {
    @Override
    public void actionPerformed(ActionEvent e) {
        new blank3(); // Create and display a new Blank3 frame
    }
});

// Initialize the "ENTREE" button
simpleButton2 = new JButton("ENTREE");
simpleButton2.setBounds(200, 240, 250, 100);
simpleButton2.setIcon(buttonIcon2);
simpleButton2.setHorizontalTextPosition(JButton.CENTER);
simpleButton2.setVerticalTextPosition(JButton.CENTER);
simpleButton2.setFont(new Font("Comic Sans", Font.BOLD, 25));
simpleButton2.setIconTextGap(0);
simpleButton2.setForeground(Color.cyan);
simpleButton2.setBackground(Color.lightGray);
simpleButton2.setBorder(BorderFactory.createEtchedBorder());

simpleButton2.addActionListener(new ActionListener() {
    @Override
    public void actionPerformed(ActionEvent e) {
        new blank3(); // Create and display a new Blank3 frame
    }
});

// Initialize the "DESSERT" button
simpleButton3 = new JButton("DESSERT");

```

```

        simpleButton3.setBounds(200, 360, 250, 100);
        simpleButton3.setIcon(buttonIcon3);
        simpleButton3.setHorizontalTextPosition(JButton.CENTER);
        simpleButton3.setVerticalTextPosition(JButton.CENTER);
        simpleButton3.setFont(new Font("Comic Sans", Font.BOLD, 25));
        simpleButton3.setIconTextGap(0);
        simpleButton3.setForeground(Color.cyan);
        simpleButton3.setBackground(Color.lightGray);
        simpleButton3.setBorder(BorderFactory.createEtchedBorder());

        simpleButton3.addActionListener(new ActionListener() {
            @Override
            public void actionPerformed(ActionEvent e) {
                new blank3(); // Create and display a new Blank3 frame
            }
        });

        // Add the recipe buttons to the frame
        add(simpleButton1);
        add(simpleButton2);
        add(simpleButton3);

        // Set frame visibility
        setVisible(true);
    }

    public static void main(String[] args) {
        new blank2nd(); // Create and display the frame
    }
}

```

3.5 INGREDIENTS PAGE DESGIN

```

import java.awt.*;
import java.awt.event.KeyAdapter;
import java.awt.event.KeyEvent;
import java.io.File;
import java.io.IOException;
import java.sql.Connection;
import java.sql.DriverManager;
import java.sql.PreparedStatement;
import java.sql.ResultSet;
import java.sql.SQLException;
import javax.imageio.ImageIO;
import javax.swing.*;

public class blank3 extends JFrame {

```

```

private JComboBox<String> comboBox;
private Image backgroundImage;
private JTextArea recipeTextArea;

public blank3() {
    // Load the background image
    try {
        backgroundImage = ImageIO.read(new
File("C:\\Users\\aaish\\OneDrive\\Desktop\\Project_Java_dbms\\src\\519.jpg"));
    } catch (IOException e) {
        e.printStackTrace();
    }

    // Set up the frame
    setTitle("Blank Page 3");
    setSize(650, 600);
    setDefaultCloseOperation(JFrame.DISPOSE_ON_CLOSE);
    setLayout(new BorderLayout());

    // Custom panel with background image
    BackgroundPanel mainPanel = new BackgroundPanel();
    mainPanel.setLayout(new GridBagLayout()); // Center components in the
panel

    // Create a panel for the title and combo box to align them vertically
    JPanel centerPanel = new JPanel();
    centerPanel.setLayout(new BoxLayout(centerPanel, BoxLayout.Y_AXIS));
    centerPanel.setOpaque(false); // Transparent panel to show background

    // Title label at the top
    JLabel titleLabel = new JLabel("What u got in der?",
SwingConstants.CENTER);
    titleLabel.setFont(new Font("Comic Sans MS", Font.BOLD, 50));
    titleLabel.setForeground(Color.WHITE); // Set font color to white
    titleLabel.setAlignmentX(Component.CENTER_ALIGNMENT); // Center-align
in box layout

    // Initialize JComboBox with an empty list and add KeyListener
    comboBox = new JComboBox<>();
    comboBox.setEditable(true);
    comboBox.setPreferredSize(new Dimension(250, 30));
    comboBox.setMaximumSize(new Dimension(250, 30)); // Set maximum size
to keep it centered

    // Recipe text area to display search results
    recipeTextArea = new JTextArea(10, 30);
    recipeTextArea.setEditable(false);
    recipeTextArea.setLineWrap(true);

```

```

        recipeTextArea.setWrapStyleWord(true);
        recipeTextArea.setFont(new Font("Arial", Font.PLAIN, 16));
        JScrollPane scrollPane = new JScrollPane(recipeTextArea);
        scrollPane.setPreferredSize(new Dimension(600, 200)); // Adjust size
to avoid plain box appearance

        // Add KeyListener to JComboBox for Enter key to trigger recipe search
        comboBox.getEditor().getEditorComponent().addKeyListener(new
KeyAdapter() {
            @Override
            public void keyPressed(KeyEvent e) {
                if (e.getKeyCode() == KeyEvent.VK_ENTER) {
                    String input = (String) comboBox.getEditor().getItem();
                    if (input != null && !input.trim().isEmpty()) {
                        searchRecipes(input); // Search recipes based on the
input
                        comboBox.setSelectedItem(""); // Clear the input after
search
                    }
                }
            }
        });

        // Add components to the center panel
        centerPanel.add(titleLabel);
        centerPanel.add(Box.createRigidArea(new Dimension(0, 10))); // Spacer
between label and combo box
        centerPanel.add(comboBox);
        centerPanel.add(Box.createRigidArea(new Dimension(0, 10))); // Spacer
between combo box and text area
        centerPanel.add(scrollPane);

        // Add center panel to the main panel and center it
        mainPanel.add(centerPanel, new GridBagConstraints());

        // Set main panel as content pane to display background
        setContentPane(mainPanel);

        // Display the frame
        setVisible(true);
    }

    private void searchRecipes(String ingredient) {
        String url = "jdbc:mysql://localhost:3306/project_java_dbms"; //
Update with your database name
        String user = "root"; // Update with your MySQL username
        String password = "1234"; // Update with your MySQL password
    }

```

```

        // Database connection and query
        try (Connection connection = DriverManager.getConnection(url, user,
password)) {
            // SQL query to find recipes based on an ingredient across the
tables
            String query = "SELECT r.recipe_name, i.instruction " +
                "FROM recipes r " +
                "JOIN recipeingredients ri ON r.recipe_id = ri.recipe_id " +
                "JOIN ingredients ing ON ri.ingredient_id = ing.ingredient_id "
+
                "JOIN instructions i ON r.recipe_id = i.recipe_id " +
                "WHERE ing.name = ?";

            PreparedStatement statement = connection.prepareStatement(query);
statement.setString(1, ingredient); // Set the ingredient
parameter

            ResultSet resultSet = statement.executeQuery();
recipeTextArea.setText(""); // Clear previous results

            // Process the query results
            boolean found = false;
            while (resultSet.next()) {
                String recipeName = resultSet.getString("recipe_name");
                String instructions = resultSet.getString("instruction");

                recipeTextArea.append("Recipe: " + recipeName + "\n");
                recipeTextArea.append("Instructions: " + instructions + "\n");

                found = true;
            }

            if (!found) {
                recipeTextArea.setText("No recipes found for the entered
ingredient.");
            }

        } catch (SQLException e) {
            recipeTextArea.setText("Error retrieving recipes: " +
e.getMessage());
            e.printStackTrace();
        }
    }

    // Custom panel to paint the background image
    private class BackgroundPanel extends JPanel {

```

```

        @Override
        protected void paintComponent(Graphics g) {
            super.paintComponent(g);
            if (backgroundImage != null) {
                g.drawImage(backgroundImage, 0, 0, getWidth(), getHeight(),
this);
            }
        }
    }

    public static void main(String[] args) {
        new blank3();
    }
}

```

3.5 RECIPE GENERATOR PAGE DESGIN

```

import java.sql.*;
import java.util.*;

public class RecipeFinder {
    private Connection conn;

    public RecipeFinder(Connection conn) {
        this.conn = conn;
    }

    public List<String> findRecipes(List<String> ingredients) throws
SQLException {
        // Build SQL query
        String query = "SELECT r.recipe_id, r.recipe_name, i.step_number,
i.instruction "
            + "FROM Recipes r "
            + "JOIN RecipeIngredients ri ON r.recipe_id =
ri.recipe_id "
            + "JOIN Ingredients ing ON ri.ingredient_id =
ing.ingredient_id "
            + "JOIN Instructions i ON r.recipe_id = i.recipe_id "
            + "WHERE ing.name IN (" + String.join(",",
Collections.nCopies(ingredients.size(), "?")) + ") "
            + "GROUP BY r.recipe_id, r.recipe_name, i.step_number,
i.instruction "
            + "HAVING COUNT(DISTINCT ing.ingredient_id) = ?";

        try (PreparedStatement stmt = conn.prepareStatement(query)) {
            // Set ingredient names dynamically in query
            for (int i = 0; i < ingredients.size(); i++) {
                stmt.setString(i + 1, ingredients.get(i));
            }
        }
    }
}

```



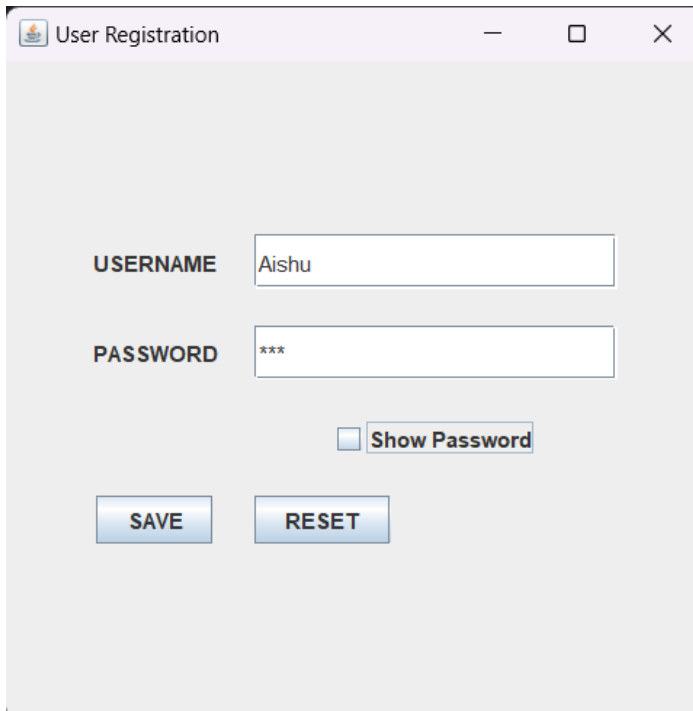
```
// Set the number of ingredients for the HAVING clause
stmt.setInt(ingredients.size() + 1, ingredients.size());

ResultSet rs = stmt.executeQuery();
List<String> recipes = new ArrayList<>();

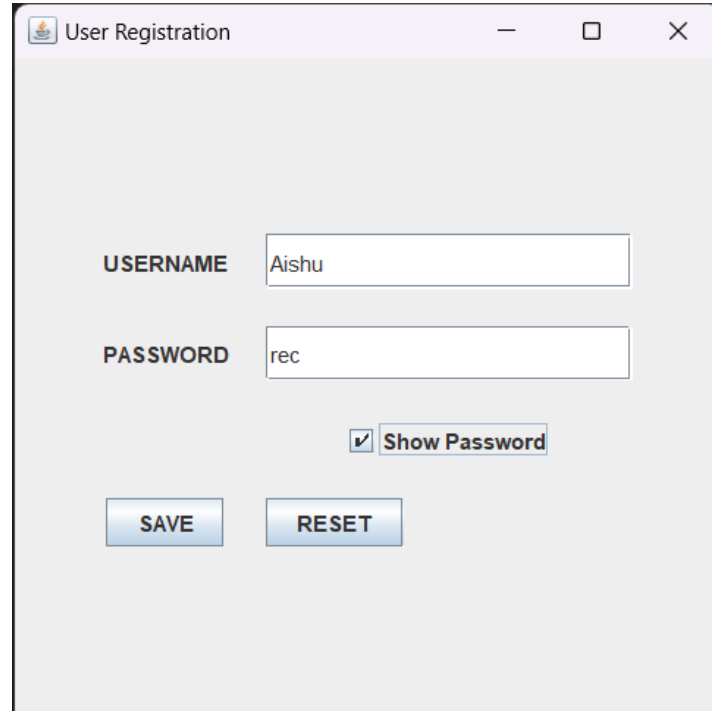
while (rs.next()) {
    int recipeId = rs.getInt("recipe_id");
    String recipeName = rs.getString("recipe_name");
    int stepNumber = rs.getInt("step_number");
    String instruction = rs.getString("instruction");
    recipes.add("Recipe ID: " + recipeId + ", Name: " + recipeName
        + ", Step " + stepNumber + ": " + instruction);
}
return recipes;
}
}
```

SNAPSHOTS

4.1 REGISTRATION PAGE

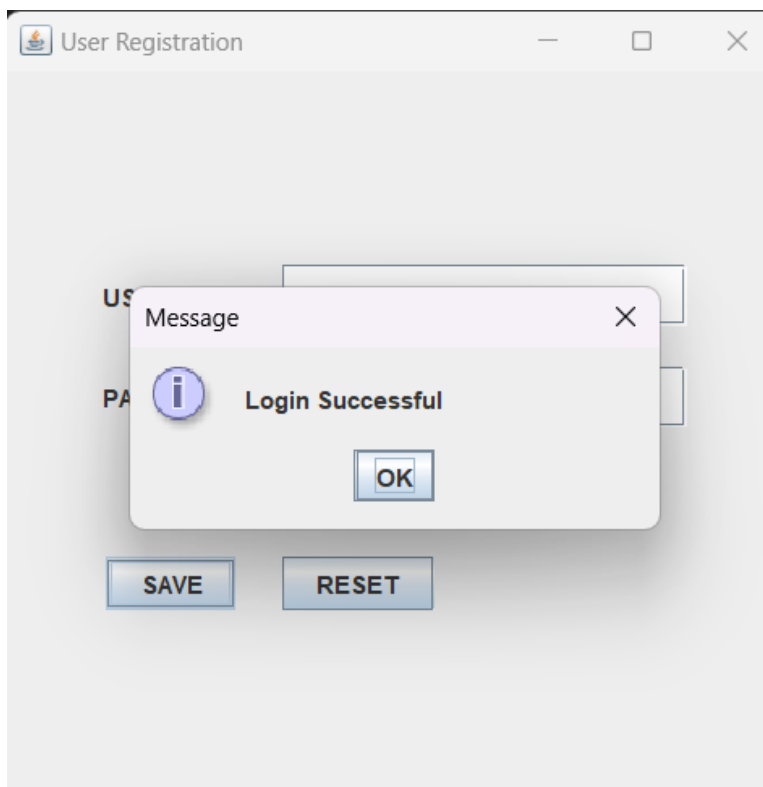


A screenshot of a web browser window titled "User Registration". The window has a light gray background. It contains two input fields: "USERNAME" with the text "Aishu" and "PASSWORD" with three asterisks "***". Below the password field is a checkbox labeled "Show Password" which is currently unchecked. At the bottom of the form are two buttons: "SAVE" and "RESET".



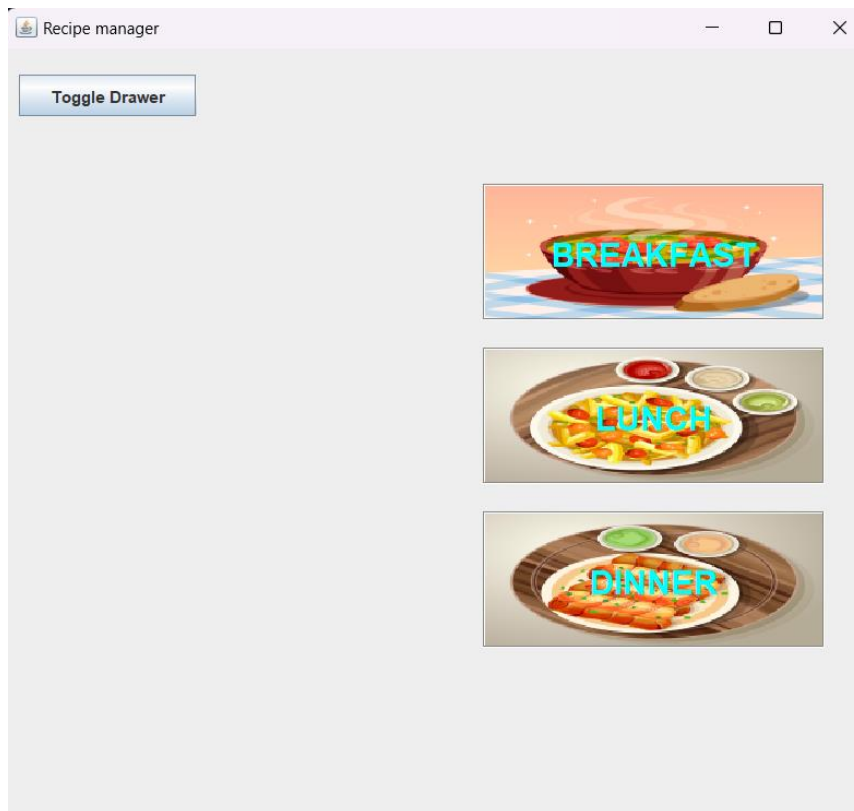
A second screenshot of the "User Registration" window. In this state, the "PASSWORD" field contains the text "rec". The "Show Password" checkbox is now checked, and the password is visible in the input field. The "USERNAME" field still contains "Aishu", and the "SAVE" and "RESET" buttons remain at the bottom.

4.2 LOGIN SUCCESSFUL

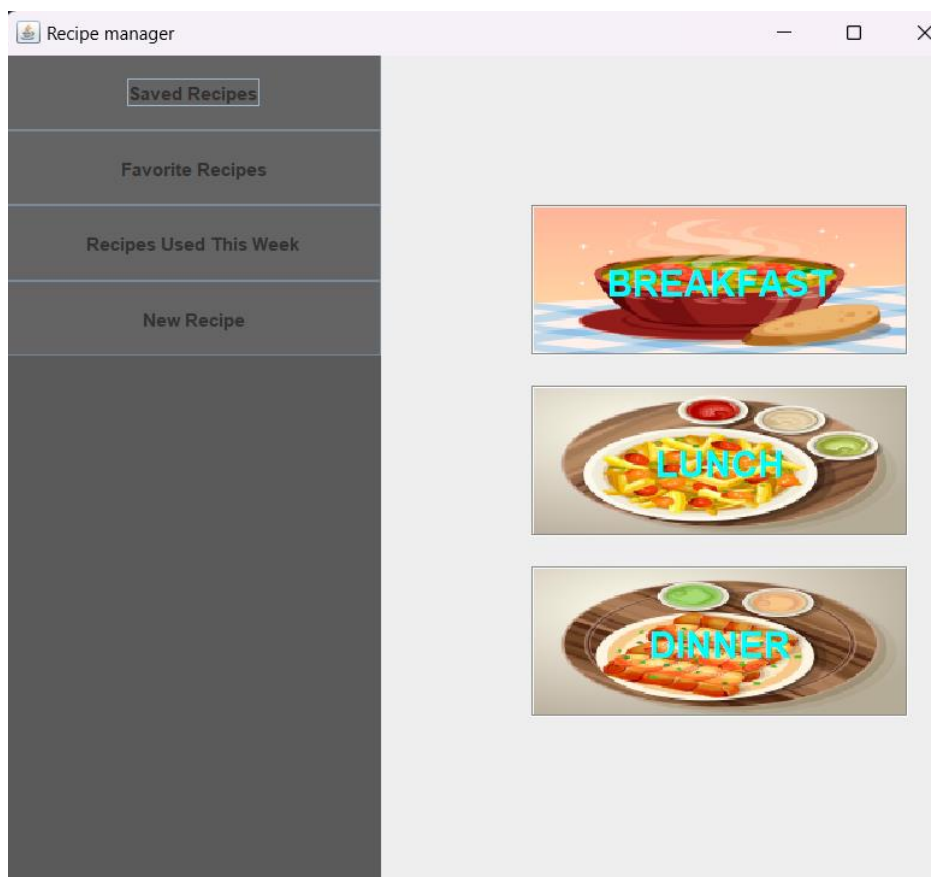


A screenshot of the "User Registration" window with a modal message box overlaid. The message box is titled "Message" and contains an information icon (a lowercase 'i' inside a circle) followed by the text "Login Successful". At the bottom of the message box is an "OK" button. The background window is partially obscured by the message box, but the "SAVE" and "RESET" buttons are still visible at the bottom.

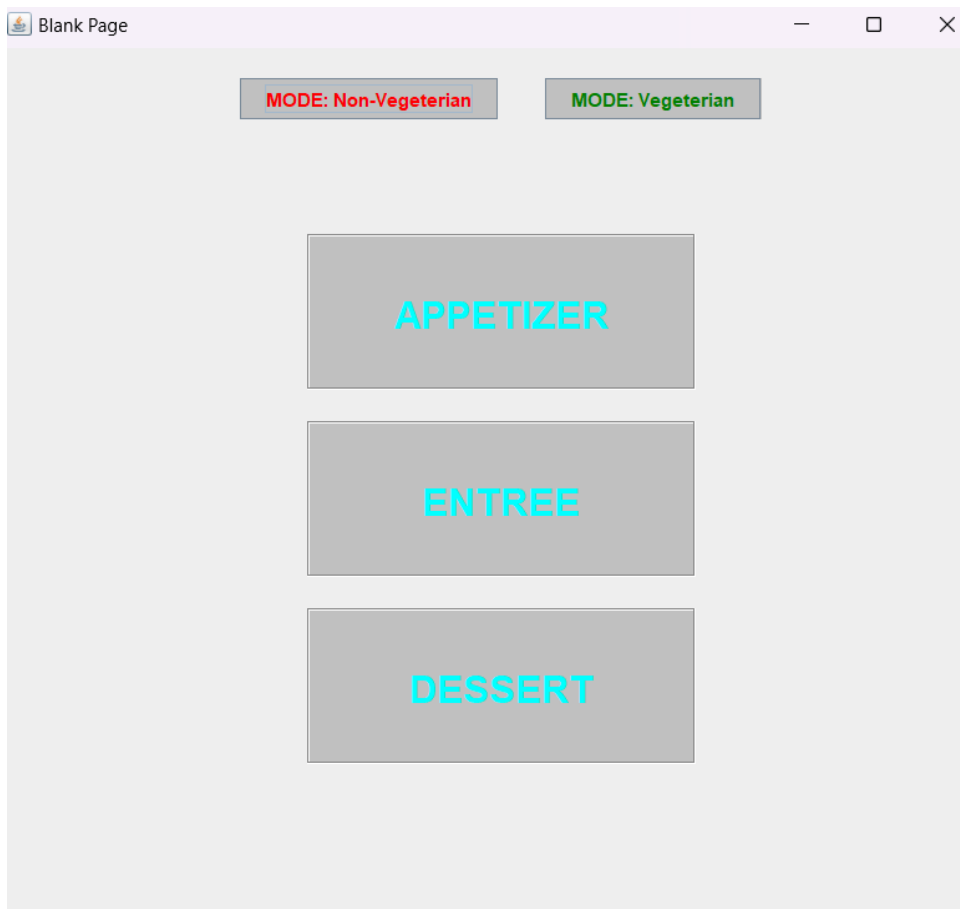
4.3 HOME PAGE



4.4 TOGGLE DRAWER

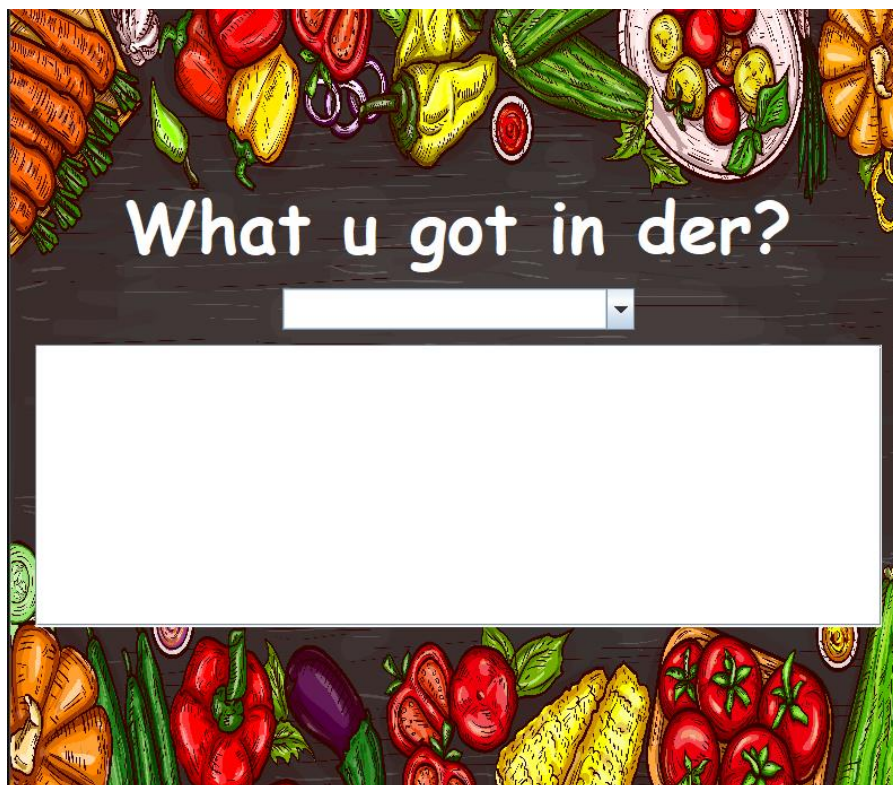


4.5 MEAL CHOOSE



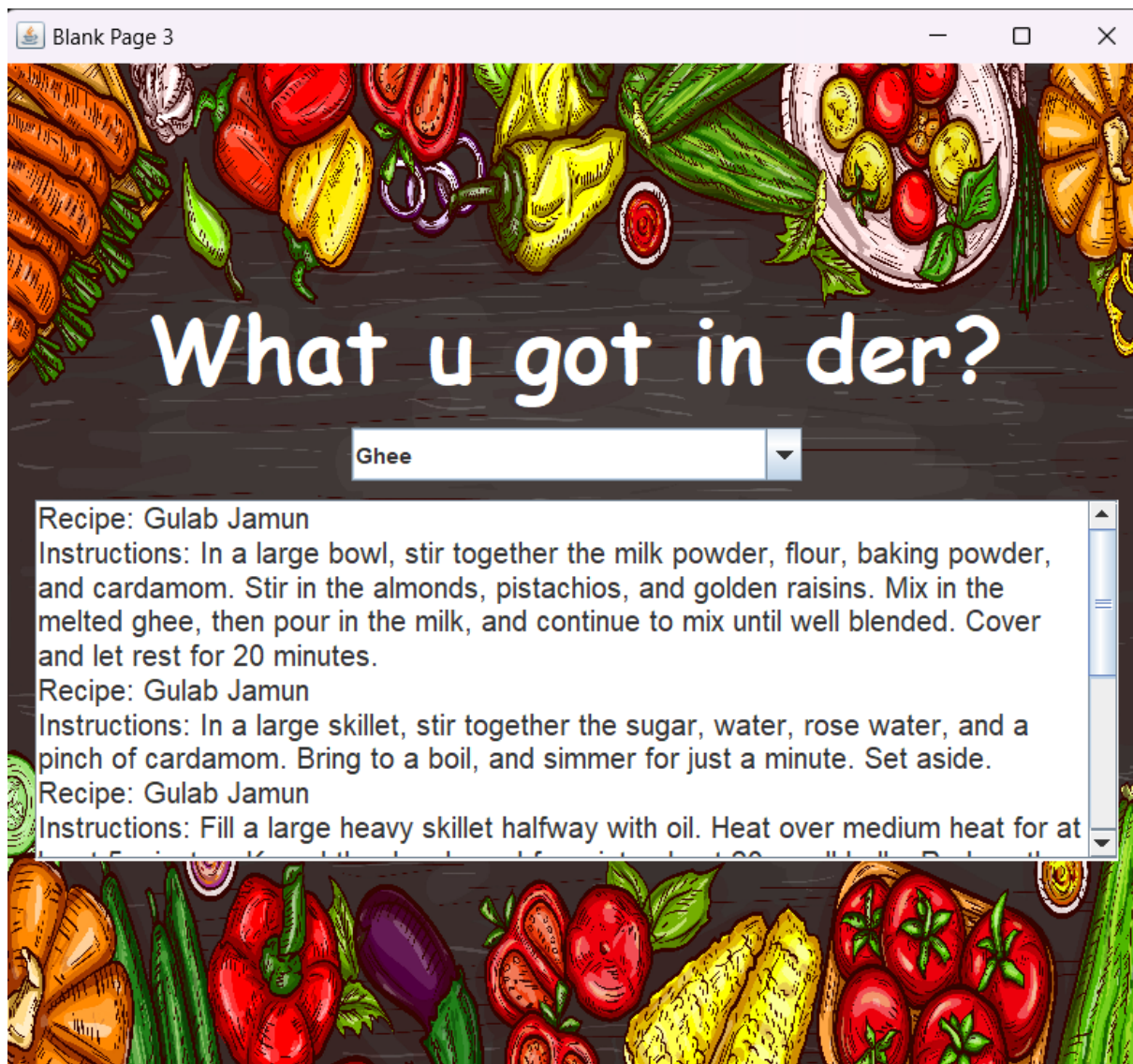
The screenshot shows a web browser window titled "Blank Page". At the top, there are two buttons: "MODE: Non-Vegetarian" in red text and "MODE: Vegetarian" in green text. Below these, there are three large, light gray rectangular buttons stacked vertically, each containing cyan text: "APPETIZER", "ENTREE", and "DESSERT".

4.6 INGREDIENTS SPECIFIER PAGE



The screenshot shows a web application interface with a dark background and a colorful border of various vegetables like carrots, bell peppers, and tomatoes. The text "What u got in der?" is displayed in a white, casual font. Below the text is a white input field with a small downward arrow on the right. Underneath the input field is a large, empty white rectangular box.

4.7 RECIPE GENERATING PAGE



CONCLUSION

The **Recipe Generator** project illustrates the integration of **Java** for the front-end and **MySQL** for the back-end to develop a robust and interactive recipe suggestion system. This tool allows users to streamline their meal preparation process by providing customized recipe recommendations based on their dietary preferences, available ingredients, and meal type.

The application delivers several features, such as user registration, personalized profiles, a guided ingredient selection process, and real-time recipe generation. By leveraging **object-oriented programming**, the project ensures maintainability and scalability, while **database management** with MySQL guarantees efficient data storage and retrieval.

This project not only simplifies meal planning for users but also serves as a practical demonstration of software development concepts, including:

- **Dynamic User Interaction:** A clean and user-friendly interface enabling effortless navigation and interaction.
- **Efficient Query Processing:** Rapid retrieval of recipes from the database using optimized queries.
- **Data Management and Security:** Safe storage of user preferences and activity logs within the database.
- **Scalability Potential:** The project is designed to incorporate advanced features like nutritional analysis, meal planning, and integration with grocery delivery platforms.

The **Recipe Generator** is a practical, real-world application that demonstrates proficiency in programming, database integration, and UI/UX design while solving a common challenge in daily life.

REFERENCES

1. <https://www.javatpoint.com/java-tutorial>
2. <https://docs.oracle.com/javase/>
3. <https://www.w3schools.com/sql/>
4. [SQL | Codecademy](#)
5. <https://dev.mysql.com/doc/>