RECEIPE EXPLORER – COOKING ASSISTANT

ABSTRACT :

This Java application facilitates exploring and selecting dishes categorized into vegetarian and non-vegetarian options. It integrates a graphical user interface (GUI) to display images of dishes, their descriptions loaded from text files, and a table view of materials and quantities retrieved from a MySQL database. Users can select their preferred dish category and view detailed information about specific dishes.

EXPLANATION:

DATABASE CONNECTION:

The program uses Java's JDBC (Java Database Connectivity) to establish a connection to a MySQL database. It uses the DriverManager.getConnection() method, passing in the database URL, username, and password.

GUI COMPONENTS & FUNCTIONALITY:

Intro Screen:

- Upon execution, an introduction screen appears, showcasing a title and an image.
- The "Explore now" button initiates the exploration process and leads to the Welcome screen.

Welcome Screen:

- Displays options for "Veg" and "Non-Veg" dishes through buttons.
- ActionListener linked to these buttons directs users to the respective dish categories.

Veg & NonVeg Screens:

- Each category (vegetarian/non-vegetarian) presents dish options through radio buttons.
- Selection of a dish triggers an ActionListener.
- ActionListener loads textual descriptions from text files and displays them in the TextArea component.
- It also queries the MySQL database for the selected dish's materials and quantities, displaying them in a table format within a panel.

INTERNAL LOGIC:

RadioListener Class:

- Listens for actions triggered by dish selection.
- Loads image, textual description, and database information for the selected dish.
- Handles the action event by calling methods to load content and display tables/images.

Displaying Text Content:

- Reads text files containing detailed procedures and descriptions for each dish.
- Utilizes a switch-case construct to determine the file path based on the selected dish's image name.
- Reads the content from the text files and displays it in the TextArea component on the GUI.

Displaying Database Information:

- Establishes a connection to the MySQL database within the displayTableAndImage() method.
- Executes SQL queries (SELECT * FROM tableName) to fetch data related to the selected dish.
- Constructs a JTable based on the ResultSet obtained from the database query.
- Removes the existing table (if any) from the panel and displays the newly constructed table within a scrollable pane.
-

USER INTERACTION FLOW:

Introduction:

- The introductory screen sets the context and initiates exploration.

Selection:

- Users navigate through categories (vegetarian/non-vegetarian) to choose dishes.
- Upon selecting a dish, detailed information is displayed - image, textual description, and database-derived material/quantity information.

PROCEDURE:

Database Connection: The program establishes a connection to a MySQL database to retrieve dish details.

Intro Screen: Upon execution, an introductory screen is displayed with an option to explore further.

Welcome Screen: Users choose between vegetarian and non-vegetarian options.

Dish Selection: After selecting a category, individual dishes within the chosen category can be selected.

Display Information: Details of the selected dish, including an image, description from text files, and database information, are displayed.

ALGORITHM:

1. **Connect to Database**:
    - Establish a connection using JDBC to the MySQL database.
2. **Intro Screen**:
    - Display a welcoming GUI screen.
    - On pressing "Explore now," move to the Welcome screen.
3. **Welcome Screen**:
    - Show options for vegetarian and non-vegetarian dishes.
    - On selection, move to the respective dish screen.
4. **Dish Screen**:
    - Display dish options within the selected category.
    - On selecting a dish, load its details, including an image, description from text files, and database information.
    - Show a table view of materials and quantities for the dish.

PROGRAM:

```
package connectmysql;
import java.awt.*;
import java.awt.event.*;
import java.io.*;
import java.sql.*;
import java.util.Vector;
import javax.swing.*;
import javax.swing.table.DefaultTableModel;
class veg extends JFrame {
    JLabel imageLabel;
    JPanel tablePanel;
    JTextArea descriptionArea;
    private Connection conn;
    public void setConnection(Connection conn) {
        this.conn = conn;
    }
    veg() {
        setSize(700, 600);
        setTitle("Veg");
        ButtonGroup buttonGroup = new ButtonGroup();
        JRadioButton dosaRadio = new JRadioButton("Dosa");
        dosaRadio.setBounds(100, 100, 150, 30);
        dosaRadio.addActionListener(new
RadioListener("Images\\Dosa.jpg", "Dosa"));
        JRadioButton vegBiriyaniRadio = new JRadioButton("Veg Biriyani");
        vegBiriyaniRadio.setBounds(100, 150, 150, 30);
        vegBiriyaniRadio.addActionListener(new
RadioListener("Images\\Veg Biriyani.jpg", "VegBiriyani"));
        JRadioButton chappathiRadio = new JRadioButton("Chappathi");
        chappathiRadio.setBounds(100, 200, 150, 30);
        chappathiRadio.addActionListener(new
```

```
RadioListener("Images\\Chappathi.jpg", "Chappathi"));
        buttonGroup.add(dosaRadio);
        buttonGroup.add(vegBiriyaniRadio);
        buttonGroup.add(chappathiRadio);
        JLabel label = new JLabel("Select your Favorite Dish:");
        label.setBounds(100, 50, 200, 30);
        imageLabel = new JLabel();
        imageLabel.setBounds(100, 280, 200, 200);
        tablePanel = new JPanel();
        tablePanel.setLayout(null);
        tablePanel.setBounds(350, 50, 300, 200);
        tablePanel.setBorder(BorderFactory.createLineBorder(Color.BLACK));
        JLabel descriptionLabel = new JLabel("Description:");
        descriptionLabel.setBounds(460, 250, 100, 30);
        descriptionArea = new JTextArea();
        descriptionArea.setBounds(350, 280, 300, 250);
        descriptionArea.setEditable(false);
        descriptionArea.setLineWrap(true);
        descriptionArea.setWrapStyleWord(true);
        JScrollPane descriptionScrollPane = new JScrollPane(descriptionArea);
        descriptionScrollPane.setBounds(350, 280, 300, 250);
        add(descriptionLabel);
        add(descriptionScrollPane);
        add(dosaRadio);
        add(vegBiriyaniRadio);
        add(chappathiRadio);
        add(label);
        add(imageLabel);
        add(tablePanel);
        setLayout(null);
        setVisible(true);
        setDefaultCloseOperation(JFrame.DISPOSE_ON_CLOSE);
```

```
    }
    class RadioListener implements ActionListener {
        String imageName;
        String tableName;
        RadioListener(String name, String table) {
            this.imageName = name;
            this.tableName = table;
        }
        @Override
        public void actionPerformed(ActionEvent e) {
            displayTableAndImage(tableName);
            loadTextContent(imageName);
            ImageIcon imageIcon = new ImageIcon(imageName);
            Image image = imageIcon.getImage().getScaledInstance(200, 200,
Image.SCALE_SMOOTH);
            imageIcon = new ImageIcon(image);
            imageLabel.setIcon(imageIcon);
        }
    }
    private void loadTextContent(String imageName) {
        try {
            String textFilePath = "";
            switch (imageName) {
                case "Images\\Dosa.jpg":
                    textFilePath = "texts\\Dosa.txt";
                    break;
                case "Images\\Chappathi.jpg":
                    textFilePath = "texts\\Chappathi.txt";
                    break;
                case "Images\\Veg Biriyani.jpg":
                    textFilePath = "texts\\Vegbiriyani.txt";
                    break;
```

```
            default:
                break;
        }
        if (!textFilePath.isEmpty()) {
            File file = new File(textFilePath);
            BufferedReader br = new BufferedReader(new FileReader(file));
            String content = "";
            String line;
            while ((line = br.readLine()) != null) {
                content += line + "\n";
            }
            br.close();
            descriptionArea.setText(content);
            descriptionArea.setCaretPosition(0);

        }
    } catch (IOException e) {
        e.printStackTrace();
    }
}
private void displayTableAndImage(String tableName) {
    try {
        Connection conn =
DriverManager.getConnection("jdbc:mysql://localhost:3306/mydb", "root", "root");
        Statement stmt = conn.createStatement();
        ResultSet rs = stmt.executeQuery("SELECT * FROM " + tableName);
        JTable table = new JTable(buildTableModel(rs));
        JScrollPane scrollPane = new JScrollPane(table);
        scrollPane.setBounds(0, 0, tablePanel.getWidth(), tablePanel.getHeight());
        tablePanel.removeAll();
        tablePanel.add(scrollPane);
        tablePanel.revalidate();
```

```
            tablePanel.repaint();
            rs.close();
            stmt.close();
            conn.close();
        } catch (SQLException e) {
            e.printStackTrace();
        }
    }
    private DefaultTableModel buildTableModel(ResultSet rs) throws SQLException {
        ResultSetMetaData metaData = rs.getMetaData();
        int columnCount = metaData.getColumnCount();

        Vector<String> columnNames = new Vector<>();
        for (int column = 1; column <= columnCount; column++) {
            columnNames.add(metaData.getColumnName(column));
        }

        Vector<Vector<Object>> data = new Vector<>();
        while (rs.next()) {
            Vector<Object> row = new Vector<>();
            for (int columnIndex = 1; columnIndex <= columnCount; columnIndex++) {
                row.add(rs.getObject(columnIndex));
            }
            data.add(row);
        }
        return new DefaultTableModel(data, columnNames);
    }
    public static void main(String[] args) {
        veg vegInstance = new veg();
        Connection conn = null;
        vegInstance.setConnection(conn);
    }
```

```
}
class nonveg extends JFrame {
    JLabel imageLabel;
    JPanel tablePanel;
    JTextArea descriptionArea;
    private Connection conn;
    public void setConnection(Connection conn) {
        this.conn = conn;
    }
    nonveg()  {
        setSize(700, 600);
        setTitle("Non-Veg");
        ButtonGroup buttonGroup = new ButtonGroup();
        JRadioButton chickenBiriyaniRadio = new JRadioButton("Chicken Biriyani");
        chickenBiriyaniRadio.setBounds(100, 100, 150, 30);
        chickenBiriyaniRadio.addActionListener(new
RadioListener("Images\\ChickenBiriyani.jpg",  "ChickenBiriyani"));
        JRadioButton muttonBiriyaniRadio = new JRadioButton("Mutton Biriyani");
        muttonBiriyaniRadio.setBounds(100, 150, 150, 30);
        muttonBiriyaniRadio.addActionListener(new
RadioListener("Images\\MuttonBiriyani.jpg", "MuttonBiriyani"));
        JRadioButton chicken65Radio = new JRadioButton("Chicken 65");
        chicken65Radio.setBounds(100, 200, 150, 30);
        chicken65Radio.addActionListener(new
RadioListener("Images\\Chicken65.jpg", "Chicken65"));
        buttonGroup.add(chickenBiriyaniRadio);
        buttonGroup.add(muttonBiriyaniRadio);
        buttonGroup.add(chicken65Radio);
        JLabel label = new JLabel("Select your Favorite Dish:");
        label.setBounds(100, 50, 200, 30);
        imageLabel = new JLabel();
        imageLabel.setBounds(100, 280, 200, 200);
```

```
        tablePanel = new JPanel();
        tablePanel.setLayout(null);
        tablePanel.setBounds(350, 50, 300, 200);
        tablePanel.setBorder(BorderFactory.createLineBorder(Color.BLACK));
        JLabel descriptionLabel = new JLabel("Description:");
        descriptionLabel.setBounds(460, 250, 100, 30);
        descriptionArea = new JTextArea();
        descriptionArea.setBounds(350, 280, 300, 250);
        descriptionArea.setEditable(false);
        descriptionArea.setLineWrap(true);
        descriptionArea.setWrapStyleWord(true);
        JScrollPane descriptionScrollPane = new JScrollPane(descriptionArea);
        descriptionScrollPane.setBounds(350, 280, 300, 250);
        add(descriptionLabel);
        add(descriptionScrollPane);
        add(chickenBiriyaniRadio);
        add(muttonBiriyaniRadio);
        add(chicken65Radio);
        add(label);
        add(imageLabel);
        add(tablePanel);
        setLayout(null);
        setVisible(true);
        setDefaultCloseOperation(JFrame.DISPOSE_ON_CLOSE);
    }
    class RadioListener implements ActionListener {
        String imageName;
        String tableName;
        RadioListener(String name, String table) {
            this.imageName = name;
            this.tableName = table;
        }
```

```
        @Override
        public void actionPerformed(ActionEvent e) {
            displayTableAndImage(tableName);
            loadTextContent(imageName);
            ImageIcon imageIcon = new ImageIcon(imageName);
            Image image = imageIcon.getImage().getScaledInstance(200, 200,
Image.SCALE_SMOOTH);
            imageIcon = new ImageIcon(image);
            imageLabel.setIcon(imageIcon);
        }
    }
    private void loadTextContent(String imageName) {
        try {
            String textFilePath = "";
            switch (imageName) {
                case "Images\\ChickenBiriyani.jpg":
                    textFilePath = "texts\\ChickenBiriyani.txt";
                    break;
                case " Images\\MuttonBiriyani.jpg":
                    textFilePath = "texts\\ MuttonBiriyani.txt";
                    break;
                case "Images\\Chicken65.jpg":
                    textFilePath = "texts\\Chicken65.txt";
                    break;
                default:
                    break;
            }
            if (!textFilePath.isEmpty()) {
                File file = new File(textFilePath);
                BufferedReader br = new BufferedReader(new FileReader(file));
                String content = "";
                String line;
```

```
            while ((line = br.readLine()) != null) {
                content += line + "\n";
            }
            br.close();
            descriptionArea.setText(content);
            descriptionArea.setCaretPosition(0);
        }
    } catch (IOException e) {
        e.printStackTrace();
    }
}
private void displayTableAndImage(String tableName) {
    try {
        Connection conn =
DriverManager.getConnection("jdbc:mysql://localhost:3306/mydb", "root", "root");
        Statement stmt = conn.createStatement();
        ResultSet rs = stmt.executeQuery("SELECT * FROM " + tableName);
        JTable table = new JTable(buildTableModel(rs));
        JScrollPane scrollPane = new JScrollPane(table);
        scrollPane.setBounds(0, 0, tablePanel.getWidth(), tablePanel.getHeight());
        tablePanel.removeAll();
        tablePanel.add(scrollPane);
        tablePanel.revalidate();
        tablePanel.repaint();
        rs.close();
        stmt.close();
        conn.close();
    } catch (SQLException e) {
        e.printStackTrace();
    }
}
private DefaultTableModel buildTableModel(ResultSet rs) throws SQLException {
```

```
        ResultSetMetaData metaData = rs.getMetaData();
        int columnCount = metaData.getColumnCount();
        Vector<String> columnNames = new Vector<>();
        for (int column = 1; column <= columnCount; column++) {
            columnNames.add(metaData.getColumnName(column));
        }
        Vector<Vector<Object>> data = new Vector<>();
        while (rs.next()) {
            Vector<Object> row = new Vector<>();
            for (int columnIndex = 1; columnIndex <= columnCount; columnIndex++) {
                row.add(rs.getObject(columnIndex));
            }
            data.add(row);
        }
        return new DefaultTableModel(data, columnNames);
    }
    public static void main(String[] args) {
        nonveg nonvegInstance = new nonveg();
        Connection conn = null;
        nonvegInstance.setConnection(conn);
    }
}
class welcome extends JFrame implements ActionListener{
    JLabel l1,vegLabel,nonvegLabel;
    JButton veg,nonveg;
    private Connection conn;
    public void setConnection(Connection conn) {
        this.conn = conn;
    }
    welcome(){
        Font f = new Font("TimesRoman",Font.BOLD,40);
        l1 = new JLabel("What You Prefer");
```

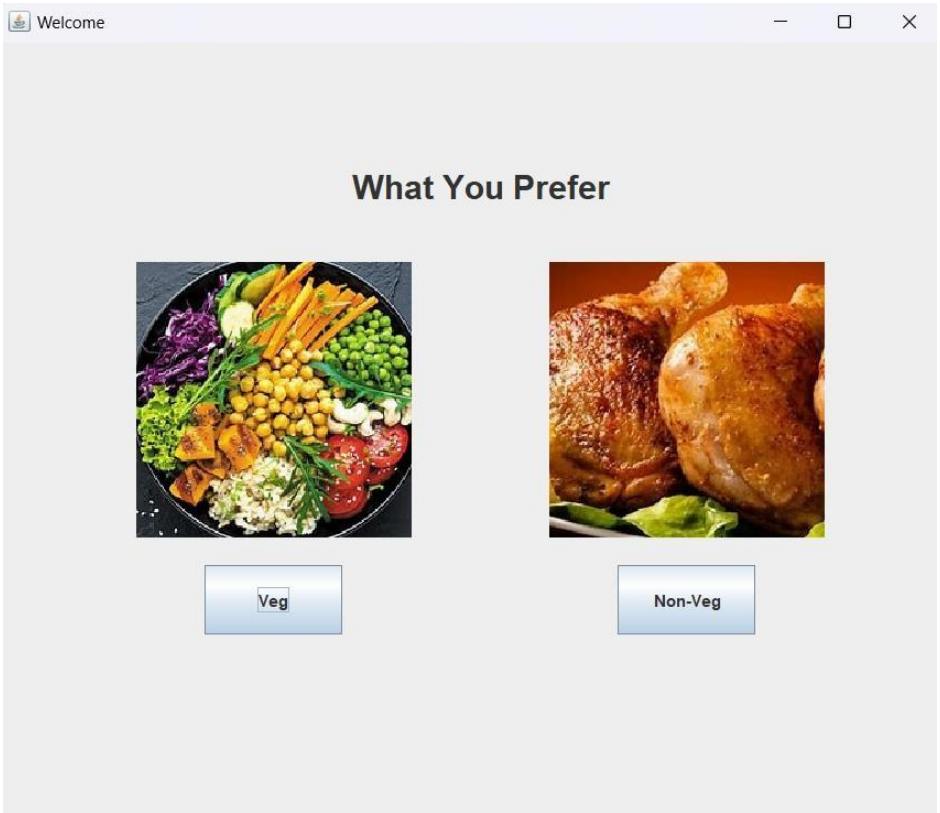
```
    l1.setBounds(100, 80, 500, 50);
    veg = new JButton("Veg");
    veg.setBounds(150, 380, 100, 50);
    ImageIcon vegIcon = new ImageIcon("Images\\veg.jpeg");
    vegLabel = new JLabel(vegIcon);
    vegLabel.setBounds(100, 160, 200, 200);
    nonveg = new JButton("Non-Veg");
    nonveg.setBounds(450, 380, 100, 50);
    ImageIcon nonVegIcon = new ImageIcon("Images\\nonveg.jpeg");
    nonvegLabel = new JLabel(nonVegIcon);
    nonvegLabel.setBounds(400, 160, 200, 200);
    add(vegLabel);
    add(nonvegLabel);
    l1.setFont(new Font(f.getName(),Font.BOLD,24));
    l1.setHorizontalAlignment(SwingConstants.CENTER);
    setLayout(null);
    add(l1);
    add(veg);
    add(nonveg);
    veg.addActionListener(this);
    nonveg.addActionListener(this);
    setSize(700, 600);
    setTitle("Welcome");
    setVisible(true);
    setDefaultCloseOperation(JFrame.DISPOSE_ON_CLOSE);
}
@Override
public void actionPerformed(ActionEvent ae) {
    if (ae.getSource() == veg) {
        veg vegScreen = new veg();
        vegScreen.setConnection(conn);
    }else if (ae.getSource() == nonveg) {
```

```
            nonveg nonvegScreen = new nonveg();
            nonvegScreen.setConnection(conn);
        }
        }
    }
    class Intro extends JFrame implements ActionListener {
        JLabel title;
        JButton enter;
        private Connection conn;
        public void setConnection(Connection conn) {
        this.conn = conn;
        }
        Intro() {
            Font f = new Font("TimesRoman", Font.BOLD, 40);
            title = new JLabel("Recipe Explorer - Cooking Assistant");
            title.setBounds(100, 100, 500, 50);
            title.setFont(new Font(f.getName(), Font.BOLD, 24));
            title.setHorizontalAlignment(SwingConstants.CENTER);
            add(title);
            ImageIcon imageIcon = new ImageIcon("Images\\main.png");
            JLabel imageLabel = new JLabel(imageIcon);
            imageLabel.setBounds(250, 170, 200, 200);
            add(imageLabel);
            enter = new JButton("Explore now");
            enter.setBounds(250, 400, 200, 50);
            add(enter);
            enter.addActionListener(this);
            setLayout(null);
            setSize(700, 600);
            setTitle("Cooking Assistant");
            setDefaultCloseOperation(JFrame.DISPOSE_ON_CLOSE);
            setVisible(true);
```

```
    }
    @Override
    public void actionPerformed(ActionEvent ae) {
        if (ae.getSource() == enter) {
            welcome welcomeScreen = new welcome();
            welcomeScreen.setConnection(conn);
        }
    }
}
public class ConnectMYSQL {
    public static void main(String[] args) {
        Connection conn = null;
        try {
            conn = DriverManager.getConnection("jdbc:mysql://localhost:3306/mydb", "root",
"root");
            System.out.println(conn);
            Intro intro = new Intro();
            intro.setConnection(conn);
        } catch (SQLException e) {
            e.printStackTrace();
        }
    }
}
```

OUTPUT:


Cooking Assistant
Recipe Explorer - Cooking Assistant
Explore now


Welcome
What You Prefer
Veg
Non-Veg

Veg

Select your Favorite Dish:

- (•) **Dosa**
- ( ) **Veg Biriyani**
- ( ) **Chappathi**

Materials	Quantity
Rice Flour	2 Cups (Avg)
Ghee or Oil	Required Level
Water	Required Level
Salt	Required Level

Description:

1. Soak 1 cup of rice and 1/4 cup urad dal with 1/2 teaspoon fenugreek seeds for 6 hours.
2. Grind soaked ingredients into a smooth batter, adding 2 tablespoons of poha.
3. Add 1 teaspoon of salt to the batter and let it ferment for 6-8 hours or overnight.
4. Heat a dosa tawa, spread a ladleful of batter, and thinly coat the surface.
5. Drizzle oil around the edges, cook until the edges lift, and the bottom is golden brown.
6. Optionally, flip and briefly cook the other side for a crispier texture.
7. Repeat for more dosas, adjusting batter thickness based on preference.
8. Serve hot dosas with coconut chutney or sambar.

Non-Veg

Select your Favorite Dish:

- ( ) **Chicken Biriyani**
- ( ) **Mutton Biriyani**
- (•) **Chicken 65**

Materials	Quantity
Chicken	1 Kg
Chicken Masala	2 Pkt
Oil	Required Level
Salt	Required Level

Description:

1. Marinate 500g boneless chicken pieces with 2 tbsp yogurt, 1 tbsp ginger-garlic paste, 1 tsp red chili powder, 1/2 tsp turmeric, and salt; let it marinate for 30 minutes.
2. In a separate bowl, mix 2 tbsp cornflour and 3 tbsp rice flour.
3. Heat oil for frying.
4. Coat the marinated chicken pieces in the flour mixture and deep fry until golden brown and crispy.
5. In a pan, heat 2 tbsp oil; sauté curry leaves, chopped garlic, and green chilies.
6. Add a dash of red chili sauce, soy sauce, and a pinch of pepper.
7. Toss the fried chicken in the sauce until well coated.

CONCLUSION:

- This Java application provides a user-friendly interface for exploring and selectingdishes based on user preferences. It integrates visual representations (images), textual descriptions, and database information to offer comprehensive details about each dish.