RECIPE FINDER

A MINI-PROJECT REPORT

Submitted by

MITHULESH S 240701314

KRISHNA KUMAR M S 240701622

in partial fulfillment of the award of the degree of

BACHELOR OF ENGINEERING

IN

COMPUTER SCIENCE AND ENGINEERING

RAJALAKSHMI ENGINEERING COLLEGE, CHENNAI

An Autonomous Institute

CHENNAI

NOVEMBER 2025

BONAFIDE CERTIFICATE

Certified that this project **" RECIPE FINDER "** is the bonafide work of **"MITHULESH S, KRISHNA KUMAR M S"** who carried out the project work under my supervision.

SIGNATURE
Ms. B. DEEPA
ASSISTANT PROFESSOR (SG)

Dept. of Computer Science and Engg,
Rajalakshmi Engineering College
Chennai

This mini project report is submitted for the viva voce examination to be held on

_____

INTERNAL EXAMINER **EXTERNAL EXAMINER**

ABSTRACT

The **Recipe Finder** project is a desktop-based application designed to help users efficiently manage and explore various recipes. The system is developed using **Java Swing** for the front-end interface and **MySQL** as the back-end database, connected through the **JDBC (Java Database Connectivity)** driver. The application allows users to **add, search, and view recipes** along with their detailed ingredients and preparation instructions.

The main objective of this project is to create a simple and user-friendly platform for storing and retrieving recipes in a structured database. Users can easily add new recipes with their ingredients and step-by-step instructions, and search for them by name using the graphical interface. The GUI, developed in **IntelliJ IDEA**, ensures an intuitive and visually clear experience by displaying recipe details in a tabular format.

This project demonstrates key concepts of **Java programming**, **GUI design**, and **database connectivity**. It serves as a practical example of integrating front-end and back-end technologies to build a functional and interactive desktop application.

ACKNOWLEDGEMENT

We express our sincere thanks to our beloved and honorable Chairman **Mr. S. MEGANATHAN** and the chairperson **Dr.M.THANGAM MEGANATHAN** for their timely support and encouragement.

We are greatly indebted to our respected and honorable principal **Dr. S.N. MURUGESAN** for his able support and guidance.

No words of gratitude will suffice for the unquestioning support extended to us by our Head Of The Department **Dr. E.M. MALATHY** and our Deputy Head Of The Department **Dr. J. MANORANJINI** for being ever supporting force during our project work

We also extend our sincere and hearty thanks to our internal guide **Mrs. B. DEEPA,** for her valuable guidance and motivation during the completion of this project.

Our sincere thanks to our family members, friends and other staff members of computer science engineering.

1. MITHULESH S

2. KRISHNA KUMAR M S

TABLE OF CONTENTS

LIST OF FIGURES

CHAPTER 1

INTRODUCTION

1.1 INTRODUCTION

The **Recipe Finder** is a desktop application developed using **Java Swing** and **MySQL**. It allows users to add, search, and view recipes along with their ingredients and preparation steps. The system provides a simple, offline platform for managing recipes efficiently with an easy-to-use graphical interface.

1.2 SCOPE OF WORK

This project focuses on creating a basic offline recipe management system. Users can add new recipes, search them by name, and view complete details. It can later be enhanced with edit, delete, or cloud storage features.

1.3 PROBLEM STATEMENT

People often struggle to organize or recall recipes. Online apps may require internet access and be complex. This project offers an **offline, user-friendly** tool to store and find recipes easily.

1.4 AIM AND OBJECTIVES OF THE PROJECT

The aim of this project is to develop an offline application that helps users store and retrieve recipes efficiently.

Objectives:

- Use **Java Swing** for GUI and **MySQL** for storage.
- Enable adding, searching, and viewing recipes.
- Display ingredients and preparation steps clearly.

CHAPTER 2

SYSTEM SPECIFICATIONS

2.1 HARDWARE SPECIFICATIONS

Processor	:	AMD Ryzen 7
Memory Size	:	16 GB
HDD	:	512 GB

2.2 SOFTWARE SPECIFICATIONS

Operating System	:	WINDOWS 11
Front – End	:	Java Swing
Back - End	:	MySql
Language	:	Java,MySQL

CHAPTER 3

MODULE DESCRIPTION

Recipe Management Module

This module allows users to **add new recipes** by entering details such as the recipe name, ingredients, and preparation steps. It stores all recipe information in the database for easy retrieval and ensures proper data organization.

Search Module

The search module enables users to **find recipes by name** quickly. It retrieves matching records from the MySQL database and displays them on the interface, helping users access their desired recipes instantly.

Database Module

This module handles all **database operations** using **MySQL**. It stores and manages recipe details and ensures smooth data access through **JDBC connectivity**, maintaining accuracy and consistency of the stored information.

User Interface Module

The user interface is built using **Java Swing**. It provides an easy-to-use and interactive layout that allows users to navigate, add, and view recipes conveniently. The interface displays recipes in a clear and organized tabular format.

CHAPTER 4 - SAMPLE CODING

```
import javax.swing.*;
import javax.swing.border.*;
import javax.swing.table.*;
import java.awt.*;
import java.awt.event.*;
import java.sql.*;

public class RecipeFinderApp extends JFrame {
    private JTextField nameField, searchField;
    private JTextArea newIngredientsArea, newInstructionsArea;
    private JTable recipeTable;
    private DefaultTableModel model;
    private JTextArea detailIngredientsArea, detailInstructionsArea;
    private JLabel detailRecipeNameLabel;

    public RecipeFinderApp() {
        System.setProperty("awt.useSystemAAFontSettings", "on");
        System.setProperty("swing.aatext", "true");

        setTitle("Recipe Finder");
        setDefaultCloseOperation(JFrame.EXIT_ON_CLOSE);
        setSize(1366, 768);
        setLocationRelativeTo(null);

        Color bg = new Color(245, 247, 250);
        Color accent = new Color(46, 204, 113);
        Color darkAccent = new Color(39, 174, 96);
        Color panelColor = Color.WHITE;

        JPanel header = createHeaderPanel(panelColor, accent);
        add(header, BorderLayout.NORTH);

        JPanel viewAndDetailPanel = createViewAndDetailPanel(panelColor,
darkAccent);
        JPanel addRecipePanel = createAddRecipePanel(panelColor, accent);
        addRecipePanel.setPreferredSize(new Dimension(360, 0));

        JSplitPane mainSplitPane = new
JSplitPane(JSplitPane.HORIZONTAL_SPLIT, viewAndDetailPanel,
addRecipePanel);
        mainSplitPane.setResizeWeight(0.78);
        mainSplitPane.setDividerSize(5);
        mainSplitPane.setBorder(null);
```

```
    add(mainSplitPane, BorderLayout.CENTER);
    getContentPane().setBackground(bg);

    loadRecipes();
    setVisible(true);
}

private JPanel createHeaderPanel(Color panelColor, Color accent) {
    JPanel header = new JPanel(new BorderLayout(20, 10));
    header.setBackground(panelColor);
    header.setBorder(new EmptyBorder(15, 30, 15, 30));

    JLabel title = new JLabel("Recipe Finder");
    title.setFont(new Font("Segoe UI Semibold", Font.BOLD, 30));
    title.setForeground(accent);
    header.add(title, BorderLayout.WEST);

    JPanel searchPanel = new JPanel(new BorderLayout(10, 0));
    searchPanel.setBackground(panelColor);

    searchField = new JTextField();
    searchField.setFont(new Font("Segoe UI", Font.PLAIN, 18));
    searchField.setPreferredSize(new Dimension(400, 40));
    searchField.setBorder(BorderFactory.createCompoundBorder(
            BorderFactory.createLineBorder(new Color(160, 160, 160)),
            new EmptyBorder(5, 10, 5, 10)
    ));

    JButton searchButton = new JButton("Search by Name");
    searchButton.setFont(new Font("Segoe UI", Font.BOLD, 16));
    searchButton.setBackground(Color.WHITE);
    searchButton.setForeground(accent);
    searchButton.setFocusPainted(false);
    searchButton.setBorder(BorderFactory.createLineBorder(accent, 2));
    searchButton.setPreferredSize(new Dimension(200, 40));
    searchButton.addActionListener(e -> searchRecipes());

    searchPanel.add(searchField, BorderLayout.CENTER);
    searchPanel.add(searchButton, BorderLayout.EAST);
    header.add(searchPanel, BorderLayout.EAST);

    return header;
}

private JPanel createViewAndDetailPanel(Color panelColor, Color accent) {
    JPanel viewPanel = new JPanel(new BorderLayout(10, 10));
    viewPanel.setBorder(BorderFactory.createEmptyBorder(10, 30, 10, 10));
```

```
        model = new DefaultTableModel(new String[]{"ID", "Recipe Name"}, 0)
{
            @Override
            public boolean isCellEditable(int row, int column) { return false; }
        };
        recipeTable = new JTable(model);
        recipeTable.setRowHeight(30);
        recipeTable.setFont(new Font("Segoe UI", Font.PLAIN, 16));

        JTableHeader header = recipeTable.getTableHeader();
        header.setFont(new Font("Segoe UI Semibold", Font.BOLD, 17));
        header.setBackground(Color.WHITE);
        header.setForeground(accent);
        header.setBorder(BorderFactory.createLineBorder(accent, 1));

        recipeTable.setGridColor(new Color(220, 220, 220));
        recipeTable.setSelectionBackground(accent);
        recipeTable.setSelectionForeground(Color.WHITE);

recipeTable.setAutoResizeMode(JTable.AUTO_RESIZE_LAST_COLUMN);
        recipeTable.getColumnModel().getColumn(0).setPreferredWidth(80);
        recipeTable.getColumnModel().getColumn(1).setPreferredWidth(700);

        JScrollPane tableScroll = new JScrollPane(recipeTable);
        tableScroll.setPreferredSize(new Dimension(0, 300));

        recipeTable.getSelectionModel().addListSelectionListener(e -> {
            if (!e.getValueIsAdjusting() && recipeTable.getSelectedRow() != -1) {
                int                    id                    =                    (Integer)
recipeTable.getValueAt(recipeTable.getSelectedRow(), 0);
                showRecipeDetails(id);
            }
        });

        JPanel detailPanel = createDetailPanel(panelColor);
        JSplitPane splitPane = new JSplitPane(JSplitPane.VERTICAL_SPLIT,
tableScroll, detailPanel);
        splitPane.setResizeWeight(0.35);
        splitPane.setDividerSize(8);
        splitPane.setBorder(null);

        viewPanel.add(splitPane, BorderLayout.CENTER);
        return viewPanel;
    }

    private JPanel createDetailPanel(Color panelColor) {
```

```
        JPanel detailPanel = new JPanel(new BorderLayout(10, 10));


        detailPanel.setBackground(panelColor);
        detailPanel.setBorder(BorderFactory.createCompoundBorder(
                new  TitledBorder(BorderFactory.createLineBorder(new  Color(200,
200, 200), 2),
                        "Selected Recipe Details",
                        TitledBorder.LEFT, TitledBorder.TOP,
                        new Font("Segoe UI", Font.BOLD, 18),
                        new Color(44, 62, 80)),
                new EmptyBorder(15, 15, 15, 15)
        ));

        detailRecipeNameLabel = new JLabel("Click a recipe to view details");
        detailRecipeNameLabel.setFont(new Font("Segoe UI", Font.BOLD, 22));
        detailRecipeNameLabel.setForeground(new Color(44, 62, 80));

        detailIngredientsArea = createReadableTextArea();
        JScrollPane ingScroll = new JScrollPane(detailIngredientsArea);
        ingScroll.setBorder(BorderFactory.createTitledBorder("Ingredients
List"));

        detailInstructionsArea = createReadableTextArea();
        JScrollPane instScroll = new JScrollPane(detailInstructionsArea);
        instScroll.setBorder(BorderFactory.createTitledBorder("Preparation
Instructions"));

        JSplitPane split = new JSplitPane(JSplitPane.HORIZONTAL_SPLIT,
ingScroll, instScroll);
        split.setResizeWeight(0.5);
        split.setDividerSize(5);

        detailPanel.add(detailRecipeNameLabel, BorderLayout.NORTH);
        detailPanel.add(split, BorderLayout.CENTER);
        return detailPanel;
    }

    private JTextArea createReadableTextArea() {
        JTextArea area = new JTextArea();
        area.setFont(new Font("Segoe UI", Font.PLAIN, 16));
        area.setLineWrap(true);
        area.setWrapStyleWord(true);
        area.setEditable(false);
        area.setMargin(new Insets(10, 10, 10, 10));
        return area;
    }
```

```
    private JPanel createAddRecipePanel(Color panelColor, Color accent) {
        JPanel panel = new JPanel();


        panel.setLayout(new BoxLayout(panel, BoxLayout.Y_AXIS));
        panel.setBackground(panelColor);
        panel.setBorder(new  TitledBorder(BorderFactory.createLineBorder(new
Color(200, 200, 200), 2),
                "Add New Recipe", TitledBorder.LEFT, TitledBorder.TOP,
                new Font("Segoe UI", Font.BOLD, 18), new Color(44, 62, 80)));

        JLabel nameLabel = new JLabel("Recipe Name");
        nameLabel.setFont(new Font("Segoe UI", Font.BOLD, 14));
        nameLabel.setForeground(accent);
        nameField = new JTextField();
        nameField.setFont(new Font("Segoe UI", Font.PLAIN, 15));
        nameField.setMaximumSize(new Dimension(Integer.MAX_VALUE, 35));

        JLabel ingLabel = new JLabel("Ingredients (Separate with new lines)");
        ingLabel.setFont(new Font("Segoe UI", Font.BOLD, 14));
        newIngredientsArea = new JTextArea(5, 20);
        newIngredientsArea.setFont(new Font("Segoe UI", Font.PLAIN, 15));
        JScrollPane ingScroll = new JScrollPane(newIngredientsArea);

        JLabel instLabel = new JLabel("Instructions (Step-by-step)");
        instLabel.setFont(new Font("Segoe UI", Font.BOLD, 14));
        newInstructionsArea = new JTextArea(5, 20);
        newInstructionsArea.setFont(new Font("Segoe UI", Font.PLAIN, 15));
        JScrollPane instScroll = new JScrollPane(newInstructionsArea);

        JButton addBtn = new JButton("Save Recipe");
        addBtn.setFont(new Font("Segoe UI", Font.BOLD, 16));
        addBtn.setBackground(Color.WHITE);
        addBtn.setForeground(accent);
        addBtn.setBorder(BorderFactory.createLineBorder(accent, 2));
        addBtn.setFocusPainted(false);
        addBtn.addActionListener(e -> addRecipe());

        panel.add(nameLabel);
        panel.add(Box.createVerticalStrut(5));
        panel.add(nameField);
        panel.add(Box.createVerticalStrut(15));
        panel.add(ingLabel);
        panel.add(Box.createVerticalStrut(5));
        panel.add(ingScroll);
        panel.add(Box.createVerticalStrut(15));
        panel.add(instLabel);
        panel.add(Box.createVerticalStrut(5));
```

```
        panel.add(instScroll);
        panel.add(Box.createVerticalStrut(20));
        panel.add(addBtn);


        return panel;
    }

    private void showRecipeDetails(int id) {
        try (Connection conn = DatabaseConnection.getConnection()) {
            String q = "SELECT name, ingredients, instructions FROM recipes
WHERE id=?";
            PreparedStatement ps = conn.prepareStatement(q);
            ps.setInt(1, id);
            ResultSet rs = ps.executeQuery();
            if (rs.next()) {
                detailRecipeNameLabel.setText("Recipe: " + rs.getString("name"));
                detailIngredientsArea.setText(rs.getString("ingredients"));
                detailInstructionsArea.setText(rs.getString("instructions"));
            }
        } catch (SQLException e) {
            JOptionPane.showMessageDialog(this, "Error: " + e.getMessage());
        }
    }

    private void addRecipe() {
        String name = nameField.getText().trim();
        String ingredients = newIngredientsArea.getText().trim();
        String instructions = newInstructionsArea.getText().trim();
        if (name.isEmpty() || ingredients.isEmpty() || instructions.isEmpty()) {
            JOptionPane.showMessageDialog(this, "Please fill all fields!");
            return;
        }
        try (Connection conn = DatabaseConnection.getConnection()) {
            String q = "INSERT INTO recipes(name, ingredients, instructions)
VALUES(?,?,?)";
            PreparedStatement ps = conn.prepareStatement(q);
            ps.setString(1, name);
            ps.setString(2, ingredients);
            ps.setString(3, instructions);
            ps.executeUpdate();
            JOptionPane.showMessageDialog(this, "Recipe added successfully!");
            loadRecipes();
        } catch (SQLException e) {
            JOptionPane.showMessageDialog(this, "Database error: " +
e.getMessage());
        }
    }
```

```
    private void loadRecipes() {
        model.setRowCount(0);
        try (Connection conn = DatabaseConnection.getConnection()) {


            Statement stmt = conn.createStatement();
            // ✅ Ascending order by ID
            ResultSet rs = stmt.executeQuery("SELECT id, name FROM recipes
ORDER BY id ASC");
            while (rs.next()) {
                model.addRow(new Object[]{rs.getInt("id"), rs.getString("name")});
            }
        } catch (SQLException e) {
            JOptionPane.showMessageDialog(this, "Error loading recipes: " +
e.getMessage());
        }
    }

    private void searchRecipes() {
        String keyword = searchField.getText().trim();
        model.setRowCount(0);
        try (Connection conn = DatabaseConnection.getConnection()) {
            PreparedStatement stmt = conn.prepareStatement("SELECT id, name
FROM recipes WHERE name LIKE ? ORDER BY id ASC");
            stmt.setString(1, "%" + keyword + "%");
            ResultSet rs = stmt.executeQuery();
            while (rs.next()) {
                model.addRow(new Object[]{rs.getInt("id"), rs.getString("name")});
            }
        } catch (SQLException e) {
            JOptionPane.showMessageDialog(this, "Search error: " +
e.getMessage());
        }
    }

    public static void main(String[] args) {
        try                                                                 {
UIManager.setLookAndFeel(UIManager.getSystemLookAndFeelClassName()
); } catch (Exception ignored) {}
        SwingUtilities.invokeLater(RecipeFinderApp::new);
    }
}
```

TABLES USED :

RECIPES TABLE:

Field	Type	Null	Key	Default	Extra
id	INT	NO	PRI	NULL	auto_increment
name	VARCHAR(100)	NO	UNI	NULL	
instructions	TEXT	YES		NULL	

INGREDIENTS TABLE:

Field	Type	Null	Key	Default	Extra
ingredient_id	INT	NO	PRI	NULL	auto_increment
ingredient_name	VARCHAR(100)	NO		NULL	

RECIPE_INGREDIENTS TABLE:

Field	Type	Null	Key	Default	Extra
recipe_id	INT	NO	MUL	NULL	
ingredient_id	INT	NO	MUL	NULL	

CHAPTER 5 - SCREEN SHOTS

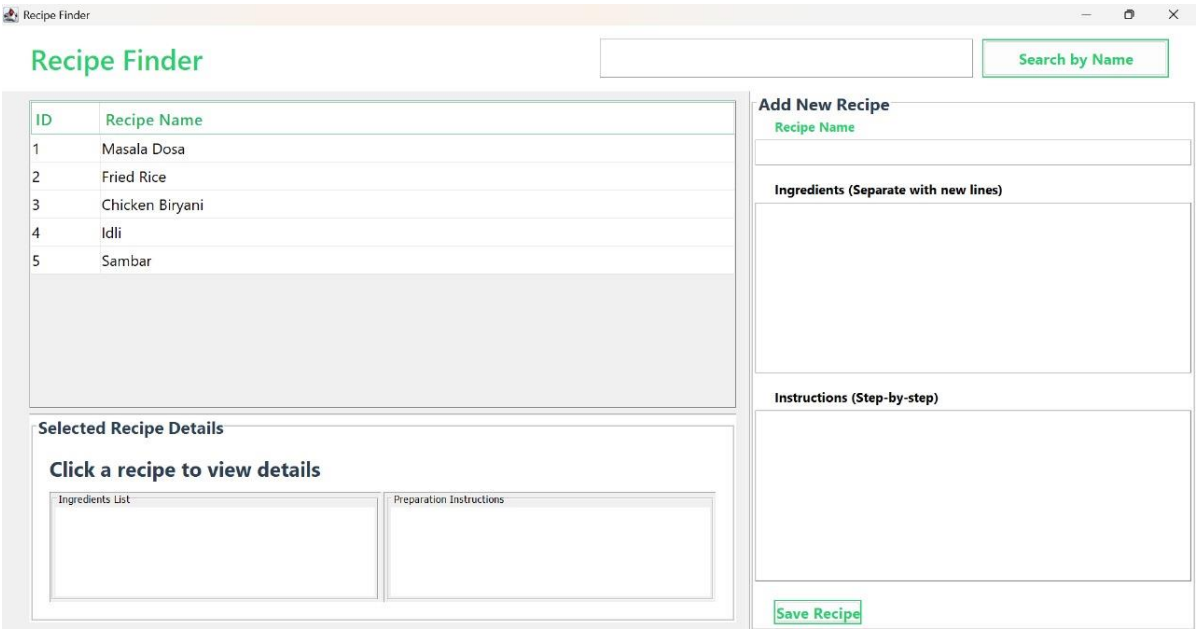

Fig 5.1 INTRO PAGE

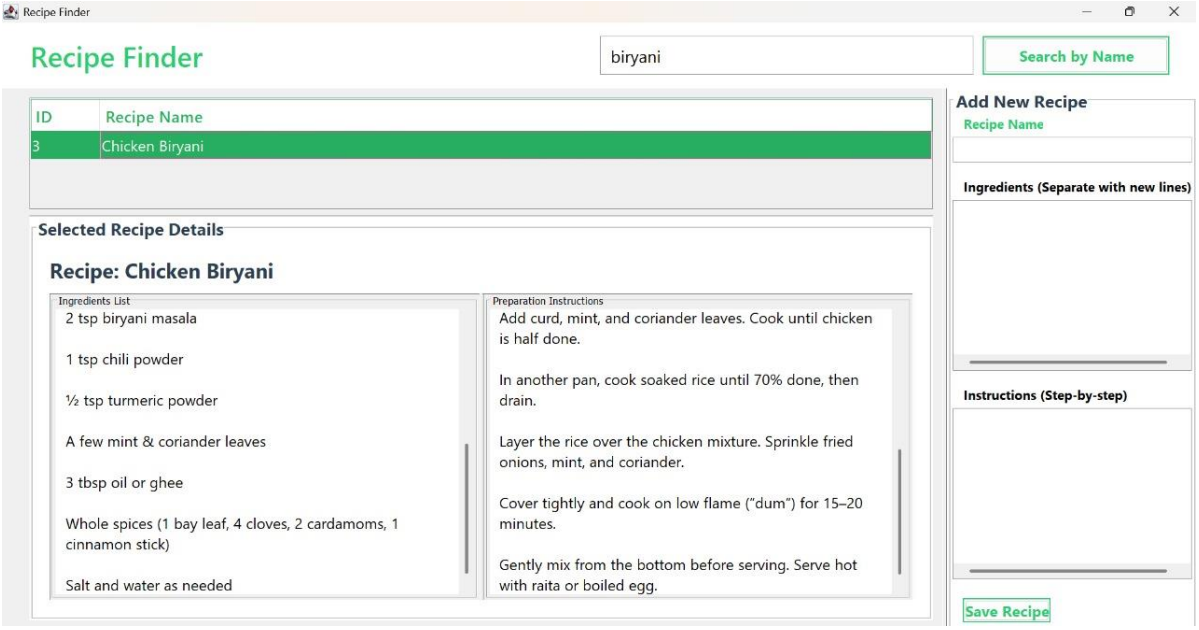

Fig 5.2 RECIPE SEARCH INTERFACE

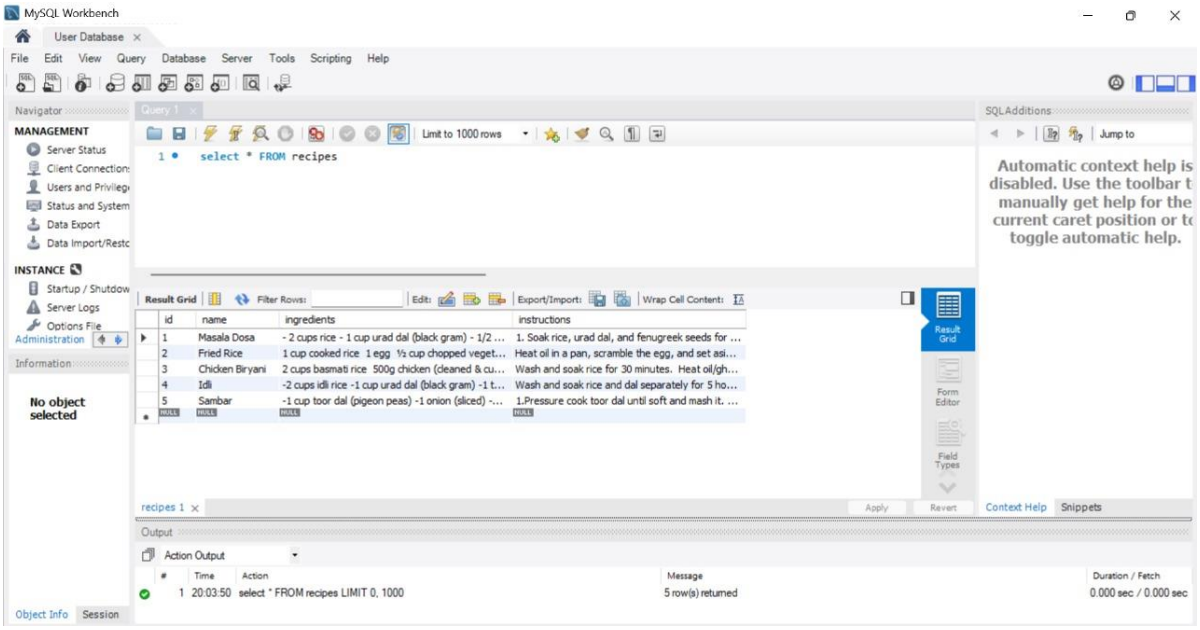

Fig 5.3 RECIPE DETAILS VIEW

Fig 5.4 ADD RECIPE PAGE

Fig 5.5 RECIPE TABLE DISPLAY (GUI TABLE)

ID	Recipe Name
1	Masala Dosa
2	Fried Rice
3	Chicken Biryani
4	Idli
5	Sambar

CHAPTER 6

CONCLUSION AND FUTURE ENHANCEMENT

The **Recipe Finder** application provides a simple and efficient way for users to **add, search, and view recipes** using a clean, user-friendly interface. Built with **Java Swing** and **MySQL**, it ensures reliable data storage and smooth database connectivity through **JDBC**. The system achieves its goal of offering an offline platform for managing recipes easily and effectively.

In the future, this project can be enhanced by adding features such as **editing and deleting recipes**, **categorization** (e.g., vegetarian or non-vegetarian), and **user authentication** for personalized access. Integration with **cloud storage**, **mobile applications**, or **voice-based search** could further improve accessibility and make the system more dynamic and interactive.

REFERENCES

1. https://www.w3schools.com/MySQL/
2. https://www.geeksforgeeks.org/java/java-database-connectivity-with-mysql/
3. https://www.geeksforgeeks.org/java/introduction-to-java-swing/
4. https://docs.oracle.com/en/java/
5. https://www.tutorialspoint.com/swing/index.htm