RECIPE FINDER AND NUTRITIONAL INFO

A MINI-PROJECT REPORT

Submitted by

NAVEED SHERIFF J 240701348

NEIL DANIEL A 240701356

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 AND NUTRITIONAL INFO "** is the bonafide work of **"NAVEED SHERIFF J, NEIL DANIEL A"** 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 and Nutritional Info* system is a desktop application developed using **Java Swing** and **MySQL** that allows users to efficiently search and manage recipes based on available ingredients and dietary preferences. The application provides detailed nutritional information, including calories, protein, carbohydrates, and fat content for each recipe. Users can add, edit, delete, and save recipes, as well as mark their favorite ones for easy access. Recipes are categorized into **vegetarian**, **non-vegetarian**, and **vegan** types for better organization. The system emphasizes data accuracy, user-friendly design, and efficient database handling through JDBC connectivity. Overall, this project aims to promote healthy eating habits while demonstrating key software engineering concepts such as **database normalization, GUI design, and CRUD operations**.

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. NAVEED SHERIFF J

2. NEIL DANIEL A

TABLE OF CONTENTS

CHAPTER	TITLE	PAGE NO
-	**– ABSTRACT –**	**3**
1	**– INTRODUCTION –**	**7**
1.1	INTRODUCTION	7
1.2	SCOPE OF THE WORK	7
1.3	PROBLEM STATEMENT	7
1.4	AIM AND OBJECTIVES OF THE PROJECT	7
2	**– SYSTEM SPECIFICATIONS –**	**8**
2.1	HARDWARE SPECIFICATIONS	8
2.2	SOFTWARE SPECIFICATIONS	8
3	**– MODULE DESCRIPTION –**	**9**
4	**– CODING –**	**10**
5	**– SCREENSHOTS –**	**17**
6	**– CONCLUSION AND FUTURE ENHANCEMENT –**	**20**
7	**– REFERENCES –**	**21**

LIST OF FIGURES

FIGURE NO	TITLE	PAGE NO
5.1	INTRODUCTION PAGE	17
5.2	GENERAL SEARCH	17
5.3	FAVORITES PAGE	18
5.4	ADD RECIPE PAGE	18
5.5	EDIT RECIPE PAGE	19

CHAPTER 1

INTRODUCTION

1.1 INTRODUCTION

The Recipe Finder and Nutritional Info application helps users find recipes based on available ingredients and diet preferences. Built using Java Swing and MySQL, it allows users to view, add, edit, and save recipes along with their nutritional values. The system supports vegetarian, non-vegetarian, and vegan diets.

1.2 SCOPE OF WORK

The project focuses on providing a simple, offline recipe management system. It allows ingredient-based searching, nutritional tracking, and favorite recipe storage. The application can be extended for cloud sync and multi-user access in the future.

1.3 PROBLEM STATEMENT

People often struggle to decide what to cook with the ingredients they have. Existing apps may require internet access and lack personalized nutrition tracking. This project offers an offline, user-friendly solution to search recipes by ingredients and diet type.

1.4 AIM AND OBJECTIVES OF THE PROJECT

The aim of this project is to create an application that helps users find, manage, and view recipes with nutritional details. Its objectives are to allow adding and editing recipes, searching by ingredients or diet type, and saving favorite dishes for quick access in a simple, user-friendly way.

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, update existing ones, and delete unwanted recipes. Each recipe includes details like ingredients, preparation steps, and nutritional values (calories, protein, carbohydrates, and fat). It ensures data consistency and easy modification of recipe information.

Search Module

The search module enables users to find recipes based on either recipe names or available ingredients. It also supports filtering by diet preferences such as Vegetarian, Non-Vegetarian, or Vegan. This helps users quickly discover suitable recipes according to their lifestyle or available ingredients.

Favorites Module

This module allows users to mark certain recipes as favorites for quick access later. Users can view, manage, and remove their favorite recipes conveniently, enhancing personalization and user engagement within the system.

Database Module

This module manages the backend storage of all recipe-related data using MySQL. It handles recipe information, ingredient lists, and favorite mappings efficiently, ensuring reliable data retrieval, integrity, and scalability for future expansion.

User Interface Module

The interface is designed using Java Swing, offering a user-friendly and visually appealing layout. It provides intuitive navigation through menus, recipe cards with images, and well-organized forms for adding or editing recipes, making the system simple and interactive to use.

CHAPTER 4 - SAMPLE CODING

```
package dao;
import database.DBConnection;
import model.Recipe;
import util.ImageUtil;
import javax.swing.*;
import java.awt.Image;
import java.io.*;
import java.sql.*;
import java.util.*;
public class RecipeDAO {
    public static List<Recipe>
searchRecipesByIngredientOrName(String query, String
dietType) {
        List<Recipe> list = new ArrayList<>();
        query = query == null ? "" :
query.trim().toLowerCase();
        dietType = dietType == null ? "All" :
dietType.trim();
        try (Connection conn = DBConnection.getConnection())
{
            String[] terms = query.split(",");
            List<String> keywords = new ArrayList<>();
            for (String t : terms) {
                t = t.trim().toLowerCase();
                if (!t.isEmpty()) keywords.add(t);
            }
            if (!keywords.isEmpty()) {
                sql.append(" AND (");
                for (int i = 0; i < keywords.size(); i++) {
                    if (i > 0) sql.append(" OR ");
                    sql.append("(LOWER(r.name) LIKE ? OR
LOWER(i.name) LIKE ?)");
                }
                sql.append(")");
            }
            if (!dietType.equalsIgnoreCase("All")) {
                if (dietType.equalsIgnoreCase("Vegetarian"))
{
```

```
                    sql.append(" AND (LOWER(r.diet_type) =
'vegetarian' OR LOWER(r.diet_type) = 'vegan')");
                } else if
(dietType.equalsIgnoreCase("Vegan")) {
                    sql.append(" AND LOWER(r.diet_type) =
'vegan'");
                } else if
(dietType.equalsIgnoreCase("Non-Veg") ||
dietType.equalsIgnoreCase("Non-Vegetarian")) {
                    sql.append(" AND (LOWER(r.diet_type) =
'non-veg' OR LOWER(r.diet_type) = 'non-vegetarian')");
                }
            }
            sql.append(" ORDER BY r.name ASC");
            PreparedStatement ps =
conn.prepareStatement(sql.toString());
            int idx = 1;
            for (String kw : keywords) {
                ps.setString(idx++, "%" + kw + "%");
                ps.setString(idx++, "%" + kw + "%");
            }
            ResultSet rs = ps.executeQuery();
            while (rs.next()) {
                Image img =
ImageUtil.readImageFromBlob(rs.getBinaryStream("image"));
                list.add(new Recipe(
                        rs.getInt("id"),
                        rs.getString("name"),
                        rs.getString("diet_type"),
                        rs.getString("description"),
                        rs.getString("instructions"),
                        rs.getInt("calories"),
                        rs.getInt("protein"),
                        rs.getInt("carbs"),
                        rs.getInt("fat"),
                        img
                ));
            }
        } catch (Exception e) {
            e.printStackTrace();
```

```
        }
        return list;
    }
    public static void addRecipe(String name, String diet,
String desc, String instr,
                                 int cal, int pro, int carb,
int fat, File imageFile,
                                 String ingredients) {
        try (Connection conn = DBConnection.getConnection())
{
            String recipeSQL = """
            INSERT INTO recipes
            (name, diet_type, description, instructions,
calories, protein, carbs, fat, image)
            VALUES (?, ?, ?, ?, ?, ?, ?, ?, ?)
        """;
            int recipeId = 0;
            try (PreparedStatement ps =
conn.prepareStatement(recipeSQL,
Statement.RETURN_GENERATED_KEYS)) {
                ps.setString(1, name);
                ps.setString(2, diet);
                ps.setString(3, desc);
                ps.setString(4, instr);
                ps.setInt(5, cal);
                ps.setInt(6, pro);
                ps.setInt(7, carb);
                ps.setInt(8, fat);
                if (imageFile != null)
                    ps.setBinaryStream(9, new
FileInputStream(imageFile));
                else
                    ps.setNull(9, Types.BLOB);
                ps.executeUpdate();
                ResultSet rs = ps.getGeneratedKeys();
                if (rs.next()) recipeId = rs.getInt(1);
            }
            if (ingredients != null &&
!ingredients.trim().isEmpty()) {
                String[] ingArr = ingredients.split(",");
```

```
                for (String ing : ingArr) {
                    String ingName =
ing.trim().toLowerCase();
                    if (ingName.isEmpty()) continue;
                    try (PreparedStatement pi =
conn.prepareStatement(
                            "INSERT IGNORE INTO ingredients
(name) VALUES (?)")) {
                        pi.setString(1, ingName);
                        pi.executeUpdate();
                    }
                    int ingId = 0;
                    try (PreparedStatement getId =
conn.prepareStatement(
                            "SELECT id FROM ingredients
WHERE name=?")) {
                        getId.setString(1, ingName);
                        ResultSet rs = getId.executeQuery();
                        if (rs.next()) ingId = rs.getInt(1);
                    }
                    if (ingId > 0) {
                        try (PreparedStatement link =
conn.prepareStatement(
                                "INSERT INTO
recipe_ingredients (recipe_id, ingredient_id, quantity)
VALUES (?, ?, '1 unit')")) {
                            link.setInt(1, recipeId);
                            link.setInt(2, ingId);
                            link.executeUpdate();
                        }
                    }
                }
            }
            System.out.println("✅ Recipe added
successfully (ID: " + recipeId + ")");
        } catch (Exception e) {
            e.printStackTrace();
            JOptionPane.showMessageDialog(null, "Error
adding recipe: " + e.getMessage());
        }
```

```
    }
    public static void updateRecipe(Recipe recipe,
List<String> ingredients) {
        try (Connection conn = DBConnection.getConnection())
{
            String sql = """
            UPDATE recipes
            SET name=?, diet_type=?, description=?,
instructions=?, calories=?, protein=?, carbs=?, fat=?
            WHERE id=?
        """;
            try (PreparedStatement ps =
conn.prepareStatement(sql)) {
                ps.setString(1, recipe.getName());
                ps.setString(2, recipe.getDietType());
                ps.setString(3, recipe.getDescription());
                ps.setString(4, recipe.getInstructions());
                ps.setInt(5, recipe.getCalories());
                ps.setInt(6, recipe.getProtein());
                ps.setInt(7, recipe.getCarbs());
                ps.setInt(8, recipe.getFat());
                ps.setInt(9, recipe.getId());
                ps.executeUpdate();
            }
            try (PreparedStatement del =
conn.prepareStatement(
                    "DELETE FROM recipe_ingredients WHERE
recipe_id=?")) {
                del.setInt(1, recipe.getId());
                del.executeUpdate();
            }
            for (String ing : ingredients) {
                String ingName = ing.trim().toLowerCase();
                if (ingName.isEmpty()) continue;
                try (PreparedStatement pi =
conn.prepareStatement("INSERT IGNORE INTO ingredients (name)
VALUES (?)")) {
                    pi.setString(1, ingName);
                    pi.executeUpdate();
                }
```

```
                int ingId = 0;
                try (PreparedStatement ps2 =
conn.prepareStatement("SELECT id FROM ingredients WHERE
name=?")) {
                    ps2.setString(1, ingName);
                    ResultSet rs = ps2.executeQuery();
                    if (rs.next()) ingId = rs.getInt(1);
                }
                if (ingId > 0) {
                    try (PreparedStatement link =
conn.prepareStatement() {
                        link.setInt(1, recipe.getId());
                        link.setInt(2, ingId);
                        link.executeUpdate();
                    }
                }
            }
        } catch (Exception e) {
            e.printStackTrace();
        }
    }
    public static List<String> getIngredientsForRecipe(int
recipeId) {
        List<String> ingredients = new ArrayList<>();
        try (Connection conn = DBConnection.getConnection();
        {
            ps.setInt(1, recipeId);
            ResultSet rs = ps.executeQuery();
            while (rs.next())
ingredients.add(rs.getString("name"));
        } catch (Exception e) {
            e.printStackTrace();
        }
        return ingredients;
    }
}
```

TABLES USED :

INGREDIENTS TABLE:

Field	Type	Null	Key	Default	Extra
id	int	NO	PRI	NULL	auto_increment
name	varchar(100)	NO	UNI	NULL	

RECIPE_INGREDIENTS TABLE:

Field	Type	Null	Key	Default	Extra
id	int	NO	PRI	NULL	auto_increment
recipe_id	int	YES	MUL	NULL	
ingredient_id	int	YES	MUL	NULL	
quantity	varchar(100)	YES		NULL	

RECIPES TABLE:

Field	Type	Null	Key	Default	Extra
id	int	NO	PRI	NULL	auto_increment
name	varchar(255)	NO	MUL	NULL	
diet_type	enum('Vegetarian','Non-Vegetarian','Vegan')	NO	MUL	NULL	
description	text	YES		NULL	
instructions	text	YES		NULL	
calories	int	YES		0	
protein	int	YES		0	
carbs	int	YES		0	
fat	int	YES		0	
image	longblob	YES		NULL	

USER_FAVORITES:

Field	Type	Null	Key	Default	Extra
id	int	NO	PRI	NULL	auto_increment
user_id	int	YES	MUL	1	
recipe_id	int	YES	MUL	NULL	

CHAPTER 5 - SCREEN SHOTS

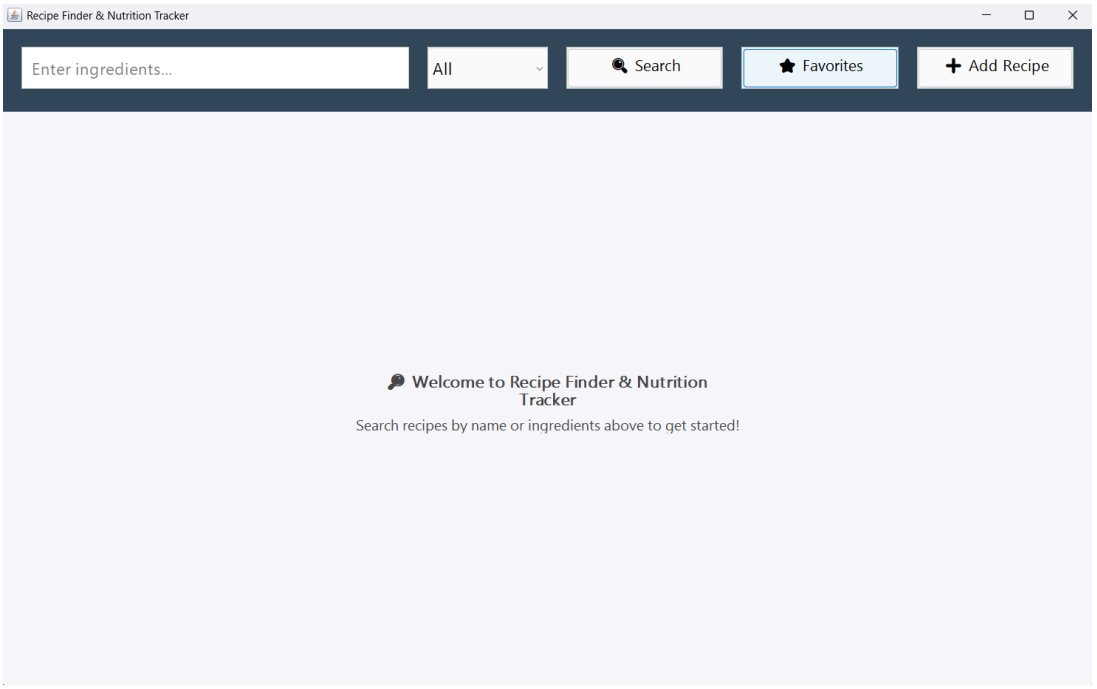

Fig 5.1 INTRO PAGE

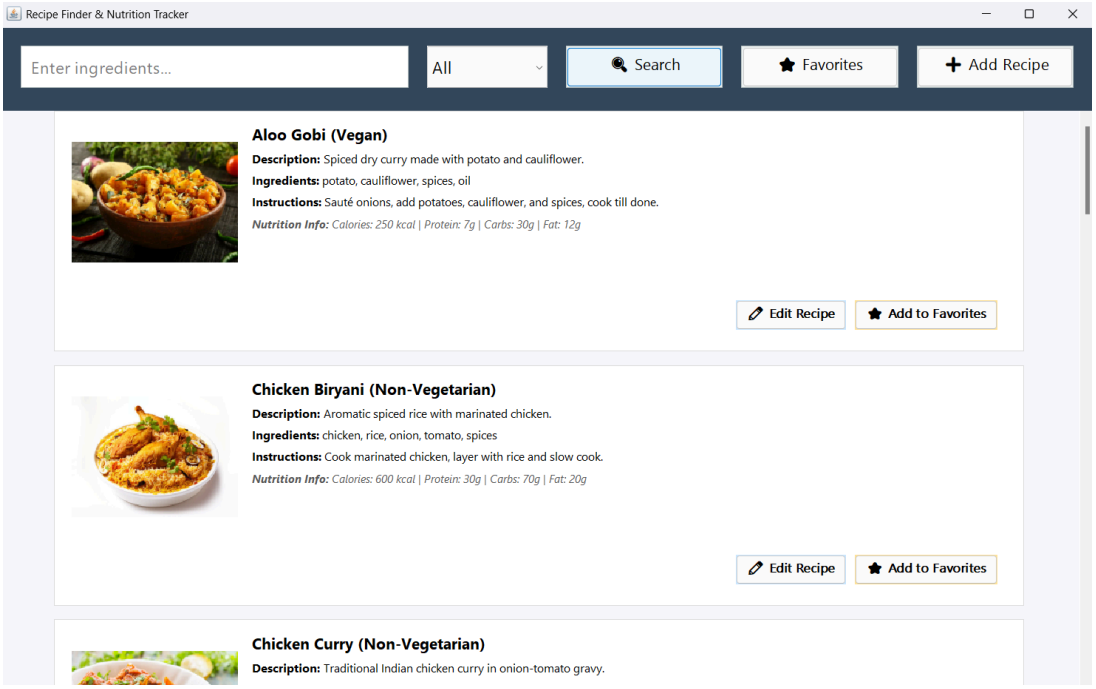

Fig 5.2 GENERAL SEARCH

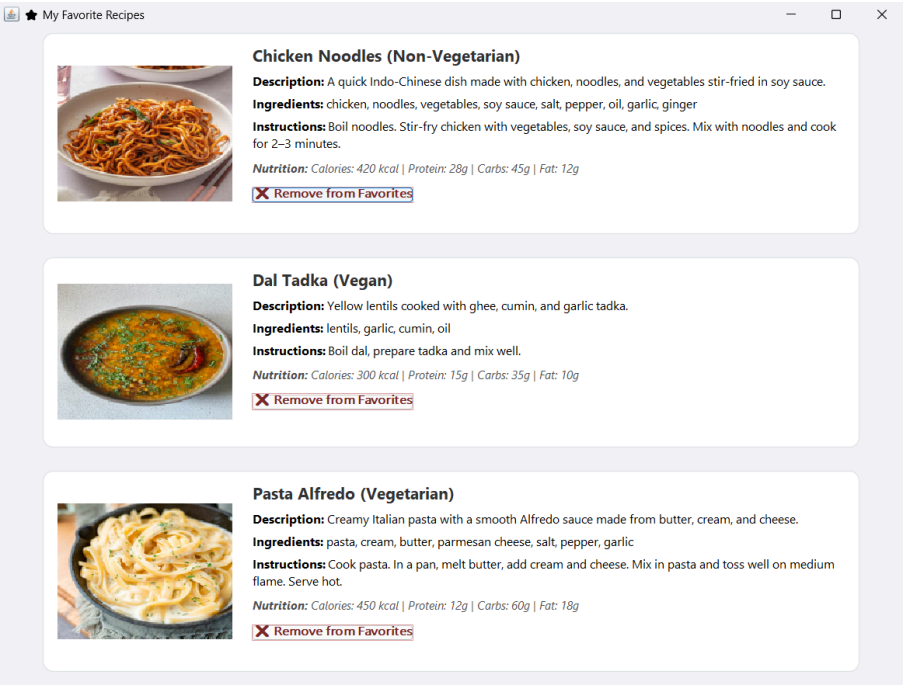

Fig 5.3 FAVORITES PAGE

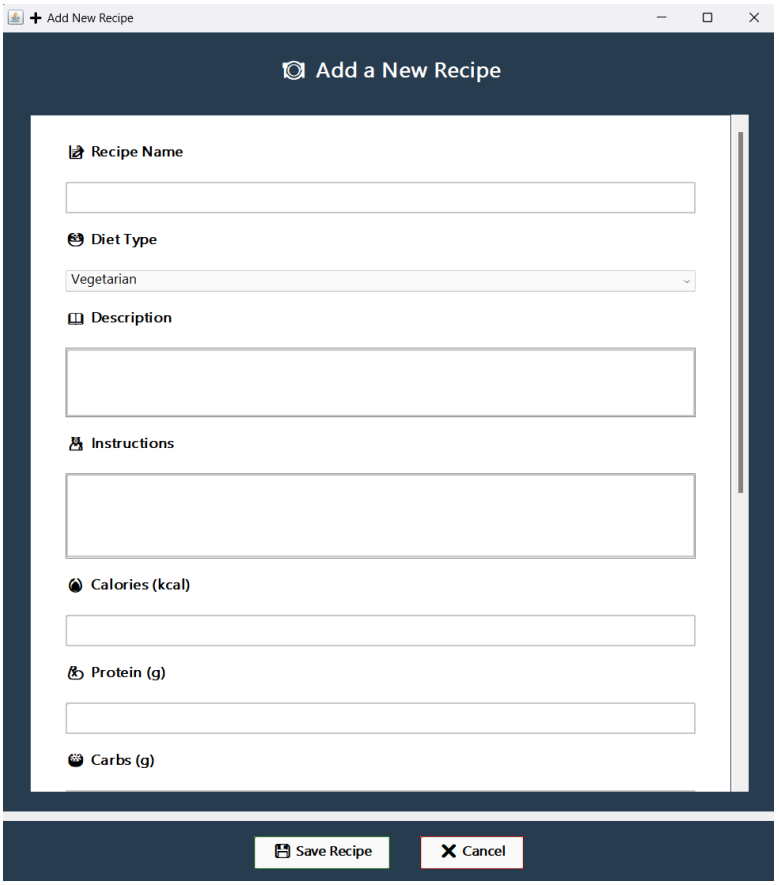

Fig 5.4 ADD RECIPE PAGE

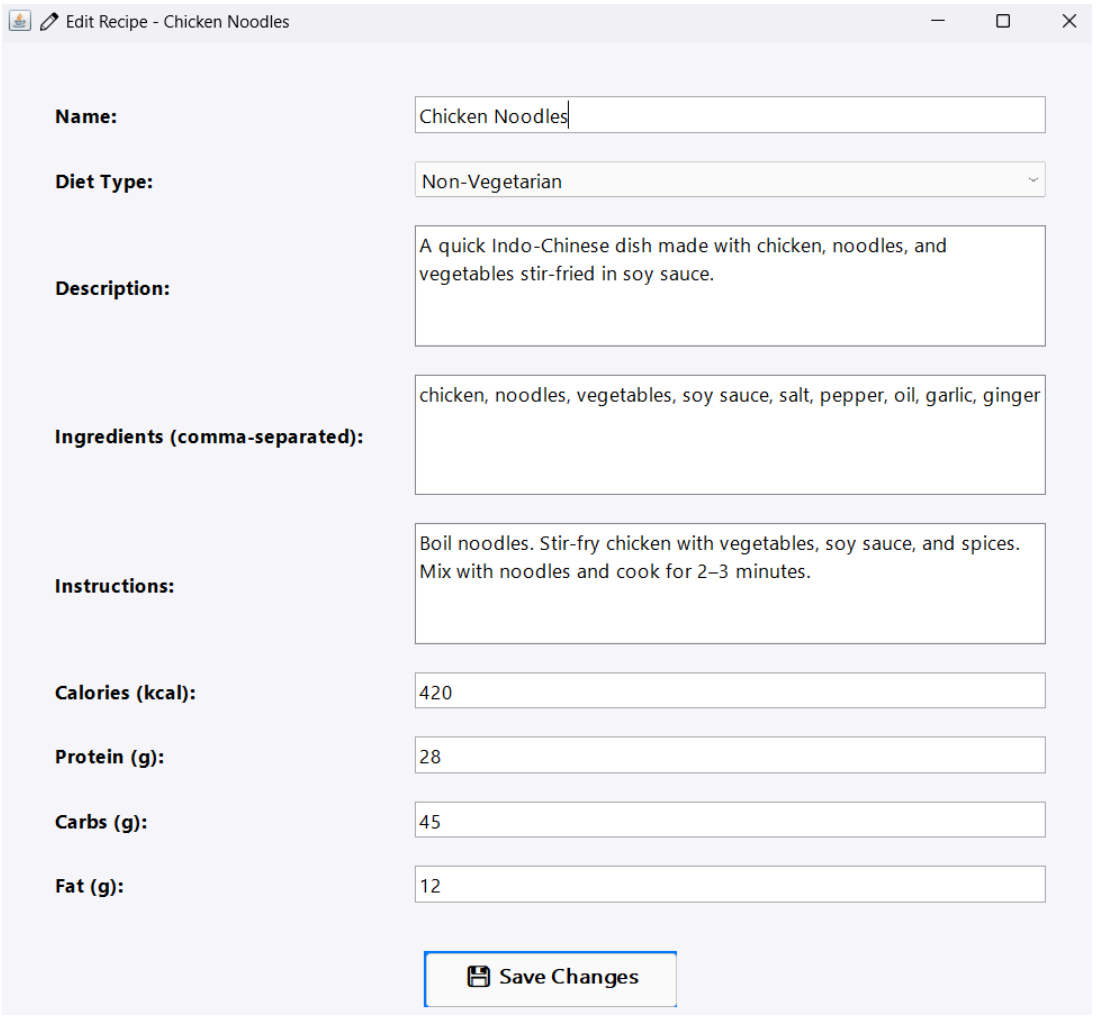

Fig 5.5 EDIT RECIPE PAGE

CHAPTER 6

CONCLUSION AND FUTURE ENHANCEMENT

The Recipe Finder and Nutritional Information System successfully provides users with an efficient way to search, manage, and store recipes based on ingredients and dietary preferences. It simplifies meal planning by offering detailed nutritional information for each recipe and ensures easy access through a clean, user-friendly interface. The integration with a MySQL database ensures data reliability and smooth functionality across all modules.

In the future, the system can be enhanced by adding user authentication to support multiple users, enabling personalized dashboards, and allowing cloud-based data synchronization. Additional features like calorie tracking, meal suggestions based on health goals, voice-based search, and mobile app integration could further improve usability and make the system more interactive and intelligent.

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