A

Report

Submitted in partial fulfilment of the Requirements for the award of the Degree of

BACHELOR OF TECHNOLOGY

IN

INFORMATION TECHNOLOGY

By

PIYUSH BHUYAN<1602-21-737-014>

Under the Guidance of

B. Leelavathy

Department of Information Technology

Vasavi College of Engineering (Autonomous)

(Affiliated to Osmania University)

Ibrahimbagh, Hyderabad-31

2022-23

BONAFIDE CERTIFICATE

This is to certify that this project report titled

'**FOOD AND NUTRITION SUGGESTION SYSTEM**' is a project work of PIYUSH BHUYAN bearing roll no.

1602-21-737-014 who carried out this project under my supervision in the IV semester for the academic year 2022-23.

Signature Internal Examiner	Signature External Examiner

FOOD AND NUTRITION SUGGESTION SYSTEM

ABSTRACT

In modern times, people are focusing a lot on monitoring their diet and making sure that adequate nutrients are being taken. It is essential for everyone to keep track of their food and nutrients. The project "Food and Nutrition Suggestion System" can help them achieve this. The database includes tables for users, foods, nutrition content, and favourites. Users can enter his or her data like height, weight and also keep a track of their food and nutrition consumption. Users can add the foods they like into the favourites table. Users can view the data they've entered and find the number of calories they're consuming over a period of time.

Requirements Analysis:

List of Tables:

1. User
2. Foods
3. Nutrients
4. Favourites

Attributes list and domain types:

User:

- user_id number(20)
- name varchar2(20)
- age number(5)
- weight number(5,2)
- height number(5,2)

Foods:

- food_id number(20)
- user_id number(20)
- food_name varchar2(20)
- calories number(20)
- weight_in_grams number(10,2)

Nutrients:

- nutrient_id number(20)
- food_id number(20)
- protein number(20)
- sugar number(20)
- fiber number(20)

Favourites:

- user_id number(20)
- food_id number(20)

AIM AND PRIORITY OF THE PROJECT

To create a Java GUI-based desktop application that connects students looking for career choices with skills and Interest. It takes values like student name, username, Age, Skills, etc through forms which are then updated in the database using JDBC connectivity.

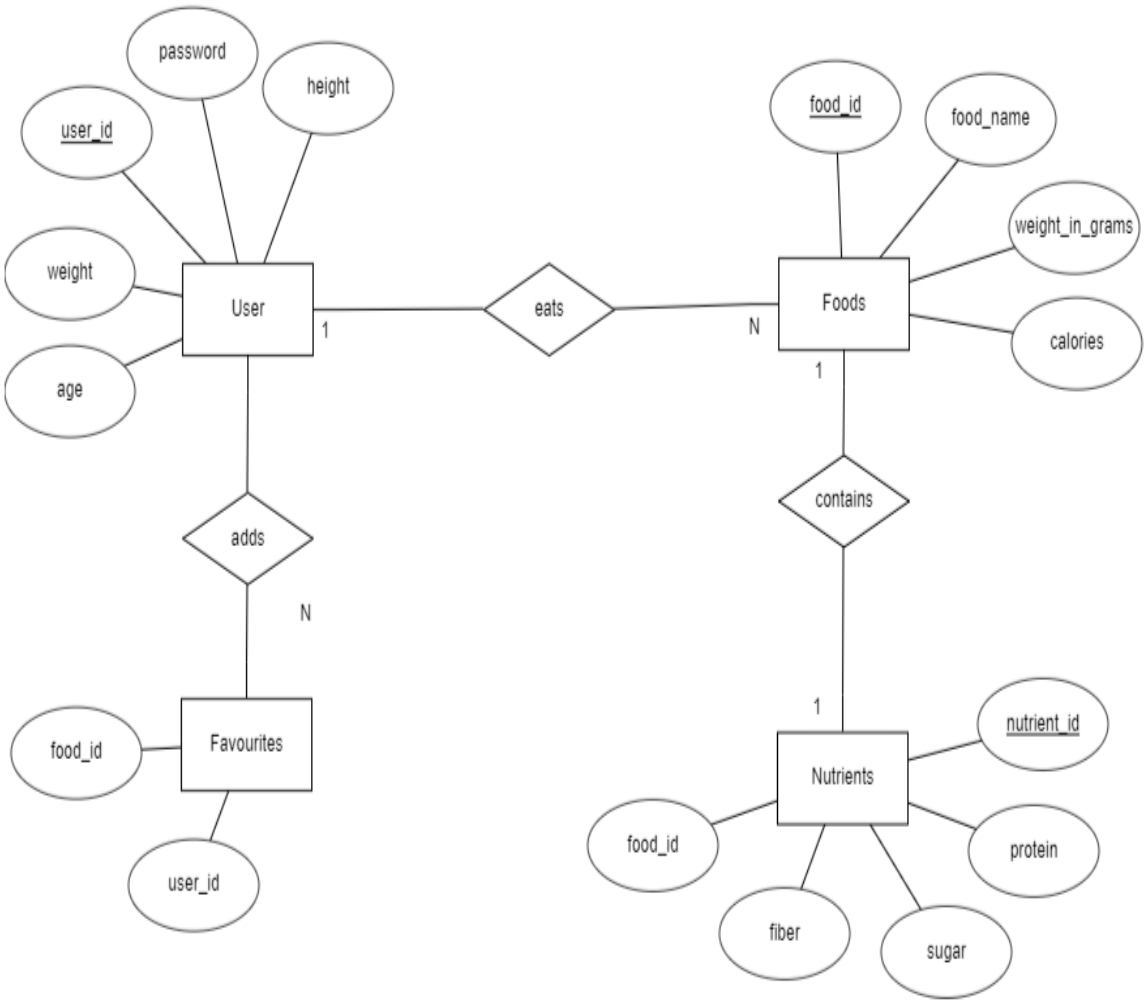
ARCHITECTURE AND TECHNOLOGY

Software used:
Java, Oracle 11g Database, Java SE version 14, Run SQL.

Java SWING:
Java SWING is a GUI widget toolkit for Java. It is part of Oracle's Java Foundation Classes (JFC) - an API for providing a graphical user interface (GUI) for Java programs.
Swing was developed to provide a more sophisticated set of GUI components than the earlier AWT. Swing provides a look and feel that emulates the look and feel of several platforms, and also supports a pluggable look and feel that allows applications to have a look and feel unrelated to the underlying platform. It has more powerful and flexible components than AWT. In addition to familiar components such as buttons, check boxes and labels, Swing provides several advanced components such as tabbed panel, scroll panes, trees, tables, and lists

SQL:
Structure Query Language(SQL) is a database query language used for storing and managing data in Relational DBMS. SQL was the first commercial language introduced for E.F Codd's Relational model of database. Today almost all RDBMS (MySql, Oracle, Infomix, Sybase, MS Access) use SQL as the standard database query language. SQL is used to perform all types of data operations in RDBMS

ER MODELLING:



Mapping Cardinalities and Participation Constraints:

User table has one-to-many (1-N) relationship with foods table as one user can enter any number of foods.

Foods and Nutrients table have one-one (1-1) relationship and Nutrients table has complete participation in Foods table.

Users and Favourites tables have one-many (1-N) relationship.

Key Constraints:

Users table:

- user_id – primary key
- name – not null

Foods table:

- food_id – primary key

Nutrients table:

- nutrient_id – primary key
- food_id – foreign key

Favourites table:

- (user_id , food_id) – primary key
- user_id – foreign key, not null
- food_id – foreign key, not null

DDL OPERATIONS:

Users Table:

SQL> create table users (

user_id number(20), name varchar2(20), weight number(5,2), height number(5,2));

Run SQL Command Line

```
SQL> create table users(
  2  user_id number(20),
  3  name varchar2(20),
  4  weight number(5,2),
  5  height number(5,2));

Table created.

SQL> alter table users
  2  add age number(5);

Table altered.

SQL> desc users;
 Name                                      Null?    Type
 ----------------------------------------- -------- ----------------------------
 USER_ID                                            NUMBER(20)
 NAME                                               VARCHAR2(20)
 WEIGHT                                             NUMBER(5,2)
 HEIGHT                                             NUMBER(5,2)
 AGE                                                NUMBER(5)

SQL> alter table users
  2  add primary key(user_id);

Table altered.

SQL> desc users;
 Name                                      Null?    Type
 ----------------------------------------- -------- ----------------------------
 USER_ID                                   NOT NULL NUMBER(20)
 NAME                                               VARCHAR2(20)
 WEIGHT                                             NUMBER(5,2)
 HEIGHT                                             NUMBER(5,2)
 AGE                                                NUMBER(5)
```

C:\oracleexe\app\oracle\product\11.2.0\server\bin\sqlplus.exe

```
SQL> desc users;
 Name                                      Null?    Type
 ----------------------------------------- -------- ----------------------------
 USER_ID                                   NOT NULL NUMBER(20)
 NAME                                               VARCHAR2(20)
 WEIGHT                                             NUMBER(5,2)
 HEIGHT                                             NUMBER(5,2)
 AGE                                                NUMBER(5)
 PASSWORD                                           VARCHAR2(20)

SQL>
```

Foods Table:

Run SQL Command Line

```
SQL> create table foods
  2  (
  3  food_id number(20),
  4  food_name varchar2(20),
  5  calories number(20),
  6  weight_in_grams number(10,2),
  7  PRIMARY KEY(food_id));

Table created.

SQL> desc foods;
 Name                                      Null?    Type
 ----------------------------------------- -------- ----------------------------
 FOOD_ID                                   NOT NULL NUMBER(20)
 FOOD_NAME                                          VARCHAR2(20)
 CALORIES                                           NUMBER(20)
 WEIGHT_IN_GRAMS                                    NUMBER(10,2)

SQL> _
```

Run SQL Command Line

```
SQL> alter table foods
  2  add user_id number(20);

Table altered.

SQL> alter table foods
  2  add foreign key(user_id) references users;

Table altered.

SQL> desc foods;
 Name                                      Null?    Type
 ----------------------------------------- -------- ----------------------------
 FOOD_ID                                   NOT NULL NUMBER(20)
 FOOD_NAME                                          VARCHAR2(20)
 CALORIES                                           NUMBER(20)
 WEIGHT_IN_GRAMS                                    NUMBER(10,2)
 USER_ID                                            NUMBER(20)

SQL>
```

Nutrients Table:

Run SQL Command Line

```
SQL> create table nutrients (
  2  user_id number(20),
  3  food_id number(20),
  4  protein number(10),
  5  sugar number(10),
  6  fiber number(10),
  7  nutrient_id number(5),
  8  PRIMARY KEY(nutrient_id),
  9  FOREIGN KEY(user_id) REFERENCES users,
 10  FOREIGN KEY(food_id) REFERENCES foods);

Table created.

SQL> desc nutrients;
 Name                                      Null?    Type
 ----------------------------------------- -------- ----------------------------
 USER_ID                                            NUMBER(20)
 FOOD_ID                                            NUMBER(20)
 PROTEIN                                            NUMBER(10)
 SUGAR                                              NUMBER(10)
 FIBER                                              NUMBER(10)
 NUTRIENT_ID                               NOT NULL NUMBER(5)

SQL>
```

Favourites Table:

Run SQL Command Line

```
  7  PRIMARY KEY(nutrient_id),
  8  FOREIGN KEY(food_id) REFERENCES foods);

Table created.

SQL> desc nutrients;
 Name                                      Null?    Type
 ----------------------------------------- -------- ----------------------------
 NUTRIENT_ID                               NOT NULL NUMBER(5)
 PROTEIN                                            NUMBER(10)
 SUGAR                                              NUMBER(10)
 FIBER                                              NUMBER(10)
 FOOD_ID                                            NUMBER(20)

SQL> create table favourites (
  2  user_id number(20) NOT NULL,
  3  food_id number(20) NOT NULL,
  4  PRIMARY KEY(user_id, food_id),
  5  FOREIGN KEY(user_id) references users,
  6  FOREIGN KEY(food_id) references foods);

Table created.

SQL> desc favourites;
 Name                                      Null?    Type
 ----------------------------------------- -------- ----------------------------
 USER_ID                                   NOT NULL NUMBER(20)
 FOOD_ID                                   NOT NULL NUMBER(20)

SQL> _
```

DML OPERATIONS:

Users Table:

SQL> insert into users values (&user_id, ‘&name’,&weight,&height,&age);

Run SQL Command Line

```
SQL> insert into users values (&user_id,'&name',&weight,&height,&age);
Enter value for user_id: 101
Enter value for name: Piyush
Enter value for weight: 82
Enter value for height: 183
Enter value for age: 20
old   1: insert into users values (&user_id,'&name',&weight,&height,&age)
new   1: insert into users values (101,'Piyush',82,183,20)

1 row created.

SQL> /
Enter value for user_id: 102
Enter value for name: Meghana
Enter value for weight: 48
Enter value for height: 155
Enter value for age: 18
old   1: insert into users values (&user_id,'&name',&weight,&height,&age)
new   1: insert into users values (102,'Meghana',48,155,18)

1 row created.

SQL> /
Enter value for user_id: 103
Enter value for name: Anish
Enter value for weight: 73
Enter value for height: 180
Enter value for age: 19
old   1: insert into users values (&user_id,'&name',&weight,&height,&age)
new   1: insert into users values (103,'Anish',73,180,19)

1 row created.

SQL> /
Enter value for user_id: 104
Enter value for name: Om Kadem
Enter value for weight: 75
Enter value for height: 183
Enter value for age: 19
old   1: insert into users values (&user_id,'&name',&weight,&height,&age)
```

Run SQL Command Line

```
SQL> /
Enter value for user_id: 105
Enter value for name: Anushka
Enter value for weight: 53
Enter value for height: 160
Enter value for age: 18
old   1: insert into users values (&user_id,'&name',&weight,&height,&age)
new   1: insert into users values (105,'Anushka ',53,160,18)

1 row created.

SQL> select * from users;

   USER_ID NAME                      WEIGHT     HEIGHT        AGE
---------- -------------------- ---------- ---------- ----------
       101 Piyush                       82        183         20
       102 Meghana                      48        155         18
       103 Anish                        73        180         19
       104 Om Kadem                     75        183         19
       105 Anushka                      53        160         18

SQL>
```

Updating password column in users table:

SQL> update users set password='&password' where user_id = &user_id;

C:\oraclexe\app\oracle\product\11.2.0\server\bin\sqlplus.exe

```
SQL> desc users;
 Name                                      Null?    Type
 ----------------------------------------- -------- ----------------------------
 USER_ID                                   NOT NULL NUMBER(20)
 NAME                                               VARCHAR2(20)
 WEIGHT                                             NUMBER(5,2)
 HEIGHT                                             NUMBER(5,2)
 AGE                                                NUMBER(5)
 PASSWORD                                           VARCHAR2(20)

SQL> update users set password = '&password' where user_id = &user_id;
Enter value for password: pi2023*
Enter value for user_id: 101
old   1: update users set password = '&password' where user_id = &user_id
new   1: update users set password = 'pi2023*' where user_id = 101

1 row updated.

SQL> /
Enter value for password: megh2004
Enter value for user_id: 102
old   1: update users set password = '&password' where user_id = &user_id
new   1: update users set password = 'megh2004' where user_id = 102

1 row updated.

SQL> /
Enter value for password: anish19
Enter value for user_id: 103
old   1: update users set password = '&password' where user_id = &user_id
new   1: update users set password = 'anish19' where user_id = 103

1 row updated.

SQL> /
Enter value for password: omk9999
Enter value for user_id: 104
old   1: update users set password = '&password' where user_id = &user_id
new   1: update users set password = 'omk9999' where user_id = 104
```

C:\oracle\app\oracle\product\11.2.0\server\bin\sqlplus.exe

```
SQL> /
Enter value for password: anu1234*
Enter value for user_id: 105
old   1: update users set password = '&password' where user_id = &user_id
new   1: update users set password = 'anu1234*' where user_id = 105

1 row updated.

SQL> select * from users;

   USER_ID NAME                     WEIGHT     HEIGHT        AGE
---------- -------------------- ---------- ---------- ----------
PASSWORD
--------------------
       101 Piyush                       82        183         20
pi2023*

       102 Meghana                      48        155         18
megh2004

       103 Anish                        73        180         19
anish19


   USER_ID NAME                     WEIGHT     HEIGHT        AGE
---------- -------------------- ---------- ---------- ----------
PASSWORD
--------------------
       104 Om Kadem                     75        183         19
omk9999

       105 Anushka                      53        160         18
anu1234*


SQL> _
```

Foods Table:

```
SQL> insert into foods values (&food_id,'&food_name',&calories,&weight_in_grams,&user_id);
Enter value for food_id: 11
Enter value for food_name: Brown Rice
Enter value for calories: 200
Enter value for weight_in_grams: 150
Enter value for user_id: 101
old   1: insert into foods values (&food_id,'&food_name',&calories,&weight_in_grams,&user_id)
new   1: insert into foods values (11,'Brown Rice',200,150,101)

1 row created.

SQL> /
Enter value for food_id: 12
Enter value for food_name: Dal Makhani
Enter value for calories: 300
Enter value for weight_in_grams: 200
Enter value for user_id: 102
old   1: insert into foods values (&food_id,'&food_name',&calories,&weight_in_grams,&user_id)
new   1: insert into foods values (12,'Dal Makhani',300,200,102)

1 row created.

SQL> /
Enter value for food_id: 13
Enter value for food_name: Vegetable Curry
Enter value for calories: 200
Enter value for weight_in_grams: 300
Enter value for user_id: 103
old   1: insert into foods values (&food_id,'&food_name',&calories,&weight_in_grams,&user_id)
new   1: insert into foods values (13,'Vegetable Curry',200,300,103)

1 row created.

SQL> /
Enter value for food_id: 14
Enter value for food_name: Baked Sweet Potato
Enter value for calories: 180
Enter value for weight_in_grams: 200
Enter value for user_id: 104
old   1: insert into foods values (&food_id,'&food_name',&calories,&weight_in_grams,&user_id)
```

Run SQL Command Line

```
old   1: insert into foods values (&food_id,'&food_name',&calories,&weight_in_grams,&user_id)
new   1: insert into foods values (14,'Baked Sweet Potato',180,200,104)

1 row created.

SQL> /
Enter value for food_id: 15
Enter value for food_name: Lentil Soup
Enter value for calories: 200
Enter value for weight_in_grams: 300
Enter value for user_id: 101
old   1: insert into foods values (&food_id,'&food_name',&calories,&weight_in_grams,&user_id)
new   1: insert into foods values (15,'Lentil Soup',200,300,101)

1 row created.

SQL> select * from foods;

   FOOD_ID FOOD_NAME              CALORIES WEIGHT_IN_GRAMS    USER_ID
---------- -------------------- ---------- --------------- ----------
        11 Brown Rice                  200             150        101
        12 Dal Makhani                 300             200        102
        13 Vegetable Curry             200             300        103
        14 Baked Sweet Potato          180             200        104
        15 Lentil Soup                 200             300        101

SQL>
```

Nutrients Table:

Run SQL Command Line

```
SQL> desc nutrients;
 Name                                      Null?    Type
 ----------------------------------------- -------- ----------------------------
 USER_ID                                            NUMBER(20)
 FOOD_ID                                            NUMBER(20)
 PROTEIN                                            NUMBER(10)
 SUGAR                                              NUMBER(10)
 FIBER                                              NUMBER(10)
 NUTRIENT_ID                               NOT NULL NUMBER(5)

SQL> insert into nutrients values (&user_id,&food_id,&protein,&sugar,&fiber,&nutrient_id);
Enter value for user_id: 101
Enter value for food_id: 11
Enter value for protein: 5
Enter value for sugar: 1
Enter value for fiber: 3
Enter value for nutrient_id: 201
old   1: insert into nutrients values (&user_id,&food_id,&protein,&sugar,&fiber,&nutrient_id)
new   1: insert into nutrients values (101,11,5,1,3,201)

1 row created.

SQL> /
Enter value for user_id: 101
Enter value for food_id: 15
Enter value for protein: 18
Enter value for sugar: 4
Enter value for fiber: 8
Enter value for nutrient_id: 202
old   1: insert into nutrients values (&user_id,&food_id,&protein,&sugar,&fiber,&nutrient_id)
new   1: insert into nutrients values (101,15,18,4,8,202)

1 row created.

SQL> /
Enter value for user_id: 102
Enter value for food_id: 12
Enter value for protein: 12
Enter value for sugar: 4
Enter value for fiber: 8
```

Run SQL Command Line

```
Enter value for nutrient_id: 203
old   1: insert into nutrients values (&user_id,&food_id,&protein,&sugar,&fiber,&nutrient_id)
new   1: insert into nutrients values (102,12,12,4,8,203)

1 row created.

SQL> /
Enter value for user_id: 103
Enter value for food_id: 13
Enter value for protein: 7
Enter value for sugar: 8
Enter value for fiber: 10
Enter value for nutrient_id: 204
old   1: insert into nutrients values (&user_id,&food_id,&protein,&sugar,&fiber,&nutrient_id)
new   1: insert into nutrients values (103,13,7,8,10,204)

1 row created.

SQL> /
Enter value for user_id: 104
Enter value for food_id: 14
Enter value for protein: 2
Enter value for sugar: 10
Enter value for fiber: 8
Enter value for nutrient_id: 205
old   1: insert into nutrients values (&user_id,&food_id,&protein,&sugar,&fiber,&nutrient_id)
new   1: insert into nutrients values (104,14,2,10,8,205)

1 row created.

SQL> select * from nutrients;

   USER_ID    FOOD_ID    PROTEIN      SUGAR      FIBER NUTRIENT_ID
---------- ---------- ---------- ---------- ---------- -----------
       101         11          5          1          3         201
       101         15         18          4          8         202
       102         12         12          4          8         203
       103         13          7          8         10         204
       104         14          2         10          8         205

SQL>
```

Favourites Table:

Run SQL Command Line

```
SQL> desc favourites;
 Name                                      Null?    Type
 ----------------------------------------- -------- ----------------------------
 USER_ID                                   NOT NULL NUMBER(20)
 FOOD_ID                                   NOT NULL NUMBER(20)

SQL> insert into favourites values (&user_id,&food_id);
Enter value for user_id: 101
Enter value for food_id: 15
old   1: insert into favourites values (&user_id,&food_id)
new   1: insert into favourites values (101,15)

1 row created.

SQL> /
Enter value for user_id: 102
Enter value for food_id: 12
old   1: insert into favourites values (&user_id,&food_id)
new   1: insert into favourites values (102,12)

1 row created.

SQL> /
Enter value for user_id: 103
Enter value for food_id: 13
old   1: insert into favourites values (&user_id,&food_id)
new   1: insert into favourites values (103,13)

1 row created.

SQL> /
Enter value for user_id: 104
Enter value for food_id: 14
old   1: insert into favourites values (&user_id,&food_id)
new   1: insert into favourites values (104,14)

1 row created.

SQL>
```

Select Run SQL Command Line

```
1 row created.

SQL> /
Enter value for user_id: 104
Enter value for food_id: 14
old   1: insert into favourites values (&user_id,&food_id)
new   1: insert into favourites values (104,14)

1 row created.

SQL> /
Enter value for user_id: 105
Enter value for food_id: 15
old   1: insert into favourites values (&user_id,&food_id)
new   1: insert into favourites values (105,15)

1 row created.

SQL> select * from favourites;

   USER_ID    FOOD_ID
---------- ----------
       101         15
       102         12
       103         13
       104         14
       105         15

SQL>
```

IMPLEMENTATION

JAVA-SQL Connectivity using JDBC:

Java Database Connectivity (JDBC) is an application programming interface (API) for the programming language Java, which defines how a client may access a database. It is a Java-based data access technology used for Java database connectivity. It is part of the Java Standard Edition platform, from Oracle Corporation. It provides methods to query and update data in a database and is oriented towards relational databases.

The connection to the database can be performed using Java programming (JDBC API) as:

```
{
    try {
        Class.forName("oracle.jdbc.OracleDriver");
        connection = DriverManager.getConnection("jdbc:oracle:thin:@localhost:1521:xe",
"piyush", "piyush");
    } catch (ClassNotFoundException e) {
        System.out.println("Oracle JDBC driver not found.");
    } catch (SQLException e) {
        System.out.println("Error connecting to the database: " + e.getMessage()); }
```

FRONT END PROGRAMS – HOME PAGE:

```
import javax.swing.*;
import java.sql.*;
import java.awt.*;
import java.awt.event.*;
import java.awt.event.ActionListener;
import java.awt.image.BufferedImage;
import java.io.File;
import java.io.IOException;
import javax.imageio.ImageIO;
import java.util.List;
import java.util.*;

public class HomePage extends JFrame {
    private JLabel titleLabel;
    private JLabel welcomeLabel;
            private JPanel foodsPanel;
    private BufferedImage backgroundImage;
    private JTabbedPane tabbedPane;
    private JButton proceedButton;



    public HomePage() {
        setTitle("Food and Nutrition Suggestion System");
        setDefaultCloseOperation(JFrame.EXIT_ON_CLOSE);
        setSize(800, 600);
        setLocationRelativeTo(null);
        setLayout(new BorderLayout());
                    //createMenu();
                    Users users = new Users();
                    Food food = new Food();
```

```
        // Load the background image
        try {
            backgroundImage = ImageIO.read(new File("E:/food and nutrition suggestion
system/canvas.jpg"));
        } catch (IOException e) {
            e.printStackTrace();
        }



        // Add the content panel
        setContentPane(new JPanel() {
            @Override
            protected void paintComponent(Graphics g) {
                super.paintComponent(g);
                if (backgroundImage != null) {
                    g.drawImage(backgroundImage, 0, 0, getWidth(), getHeight(), null);
                }
            }
        });

        // Set layout for the content panel
        setLayout(new GridBagLayout());
                JComboBox<Integer> userIDComboBox = new JComboBox<>();

                // Populate the userIDComboBox with existing user IDs
                List<Integer> userIDs = users.getUserIDs();
                for (Integer userID : userIDs) {
                        userIDComboBox.addItem(userID);
                }

        // Add title label
        titleLabel = new JLabel("Food and Nutrition Suggestion System");
        titleLabel.setFont(new Font("Helvetica", Font.BOLD, 30));
```

```
titleLabel.setForeground(Color.BLACK);
GridBagConstraints titleConstraints = new GridBagConstraints();
titleConstraints.gridx = 0;
titleConstraints.gridy = 0;
titleConstraints.insets = new Insets(20, 0, 20, 0);
add(titleLabel, titleConstraints);

// Add welcome label
welcomeLabel = new JLabel("Welcome!");
welcomeLabel.setFont(new Font("Arial", Font.BOLD, 24));
welcomeLabel.setForeground(Color.WHITE);
GridBagConstraints welcomeConstraints = new GridBagConstraints();
welcomeConstraints.gridx = 0;
welcomeConstraints.gridy = 1;
add(welcomeLabel, welcomeConstraints);

// Add proceed button
proceedButton = new JButton("Click here to proceed");
proceedButton.setFont(new Font("Arial", Font.BOLD, 16));
GridBagConstraints buttonConstraints = new GridBagConstraints();
buttonConstraints.gridx = 0;
buttonConstraints.gridy = 2;
buttonConstraints.insets = new Insets(20, 0, 0, 0);
add(proceedButton, buttonConstraints);

// Create the tabbed pane
/*tabbedPane = new JTabbedPane();
        JPanel foodsPanel = new JPanel();
        frame.add(foodsPanel);
        JPanel foodSectionPanel = new JPanel();
```

```
            JPanel nutrientsSectionPanel = new JPanel();
            JTabbedPane foodsTabbedPane = new JTabbedPane();
            foodsTabbedPane.addTab("Food", foodSectionPanel);
            foodsTabbedPane.addTab("Nutrients", nutrientsSectionPanel);
            foodsPanel.add(foodsTabbedPane);*/
// Add action listener to the proceed button
proceedButton.addActionListener(new ActionListener() {
    @Override
    public void actionPerformed(ActionEvent e) {
        // Show the tabbed pane
        tabbedPane.setVisible(true);
                                proceedButton.setVisible(false);
    }
});
            tabbedPane = new JTabbedPane();

// Add the tabbed pane to the main content pane
GridBagConstraints tabbedPaneConstraints = new GridBagConstraints();
tabbedPaneConstraints.gridx = 0;
tabbedPaneConstraints.gridy = 3;
tabbedPaneConstraints.fill = GridBagConstraints.BOTH;
tabbedPaneConstraints.weightx = 1.0;
tabbedPaneConstraints.weighty = 1.0;
tabbedPaneConstraints.insets = new Insets(10, 10, 10, 10);
tabbedPane.setVisible(false); // Initially hide the tabbed pane
add(tabbedPane, tabbedPaneConstraints);
            setVisible(true);
// USERS PANEL
            JPanel usersPanel = new JPanel();
            usersPanel.setLayout(new GridBagLayout());
            GridBagConstraints gbc = new GridBagConstraints();
            gbc.insets = new Insets(10, 10, 10, 10);
```

```
        // Add User ID label and text field
        JLabel userIdLabel = new JLabel("User ID:");
        gbc.gridx = 0;
        gbc.gridy = 0;
        usersPanel.add(userIdLabel, gbc);

        JTextField userIdField = new JTextField(10); // Adjust the size as needed
        gbc.gridx = 1;
        gbc.gridy = 0;
        usersPanel.add(userIdField, gbc);

        // Add Name label and text field
        JLabel nameLabel = new JLabel("Name:");
        gbc.gridx = 0;
        gbc.gridy = 1;
        usersPanel.add(nameLabel, gbc);

        JTextField nameField = new JTextField(10); // Adjust the size as needed
        gbc.gridx = 1;
        gbc.gridy = 1;
        usersPanel.add(nameField, gbc);

// Add Age label and text field
JLabel ageLabel = new JLabel("Age:");
gbc.gridx = 0;
gbc.gridy = 2;
usersPanel.add(ageLabel, gbc);

JTextField ageField = new JTextField(10); // Adjust the size as needed
gbc.gridx = 1;
gbc.gridy = 2;
```

```
usersPanel.add(ageField, gbc);

// Add Weight label and text field
JLabel weightLabel = new JLabel("Weight:");
gbc.gridx = 0;
gbc.gridy = 3;
usersPanel.add(weightLabel, gbc);

JTextField weightField = new JTextField(10); // Adjust the size as needed
gbc.gridx = 1;
gbc.gridy = 3;
usersPanel.add(weightField, gbc);

// Add Height label and text field
JLabel heightLabel = new JLabel("Height:");
gbc.gridx = 0;
gbc.gridy = 4;
usersPanel.add(heightLabel, gbc);

JTextField heightField = new JTextField(10); // Adjust the size as needed
gbc.gridx = 1;
gbc.gridy = 4;
usersPanel.add(heightField, gbc);

// Add Submit button
JButton submitButton = new JButton("Submit");
gbc.gridx = 0;
gbc.gridy = 5;
gbc.gridwidth = 2;
usersPanel.add(submitButton, gbc);

// Add Delete button
```

```
JButton deleteButton = new JButton("Delete");
gbc.gridx = 0;
gbc.gridy = 6;
gbc.gridwidth = 2;
usersPanel.add(deleteButton, gbc);

// Add Update button
JButton updateButton = new JButton("Update");
gbc.gridx = 0;
gbc.gridy = 7;
gbc.gridwidth = 2;

usersPanel.add(updateButton, gbc);

JButton viewUserButton = new JButton("View Users");
gbc.gridx = 0;
gbc.gridy = 8;
gbc.gridwidth = 2;
usersPanel.add(viewUserButton, gbc);

                    // Create the tabbed pane
tabbedPane = new JTabbedPane();

// Create the FOODS panel
JPanel foodsPanel = new JPanel();
foodsPanel.setLayout(new GridBagLayout());
GridBagConstraints gbcFoods = new GridBagConstraints();
gbcFoods.insets = new Insets(10, 10, 10, 10);

// Add User ID label and combobox
JLabel userLabel = new JLabel("Select your user id:");
gbcFoods.gridx = 0;
```

```
gbcFoods.gridy = 0;
foodsPanel.add(userLabel, gbcFoods);

JComboBox<String> userIdComboBox = new JComboBox<>();
gbc.gridx = 1;
gbc.gridy = 0;
foodsPanel.add(userIdComboBox, gbc);

List<Integer> userIds = users.getUserIDs();
List<String> userIdStrings = new ArrayList<>();

// Convert List<Integer> to List<String>
for (Integer userId : userIds) {
    userIdStrings.add(String.valueOf(userId));
}

// Populate the userIdComboBox with existing user IDs
for (String userId : userIdStrings) {
    userIdComboBox.addItem(userId);
}

// Add Food ID label and text field
JLabel foodIdLabel = new JLabel("Food ID:");
gbcFoods.gridx = 0;
gbcFoods.gridy = 1;
foodsPanel.add(foodIdLabel, gbcFoods);

JTextField foodIdField = new JTextField(10);
gbcFoods.gridx = 1;
gbcFoods.gridy = 1;
foodsPanel.add(foodIdField, gbcFoods);
```

```
// Add Food Name label and text field
JLabel foodNameLabel = new JLabel("Food Name:");
gbcFoods.gridx = 0;
gbcFoods.gridy = 2;
foodsPanel.add(foodNameLabel, gbcFoods);

JTextField foodNameField = new JTextField(10);
gbcFoods.gridx = 1;
gbcFoods.gridy = 2;
foodsPanel.add(foodNameField, gbcFoods);

// Add Calories label and text field
JLabel caloriesLabel = new JLabel("Calories:");
gbcFoods.gridx = 0;
gbcFoods.gridy = 3;
foodsPanel.add(caloriesLabel, gbcFoods);

JTextField caloriesField = new JTextField(10);
gbcFoods.gridx = 1;
gbcFoods.gridy = 3;
foodsPanel.add(caloriesField, gbcFoods);

// Add Weight label and text field
JLabel foodweightLabel = new JLabel("Weight in grams:");
gbcFoods.gridx = 0;
gbcFoods.gridy = 4;
foodsPanel.add(foodweightLabel, gbcFoods);

JTextField foodweightField = new JTextField(10);
gbcFoods.gridx = 1;
gbcFoods.gridy = 4;
foodsPanel.add(foodweightField, gbcFoods);
```

```
// Add Submit button
JButton submitfoodButton = new JButton("Submit");
gbcFoods.gridx = 0;
gbcFoods.gridy = 5;
gbcFoods.gridwidth = 2;
foodsPanel.add(submitfoodButton, gbcFoods);

// Add Nutrient ID label and text field
JLabel nutrientIdLabel = new JLabel("Nutrient ID:");
gbcFoods.gridx = 0;
gbcFoods.gridy = 6;
gbcFoods.gridwidth = 1;
foodsPanel.add(nutrientIdLabel, gbcFoods);

JTextField nutrientIdField = new JTextField(10);
gbcFoods.gridx = 1;
gbcFoods.gridy = 6;
foodsPanel.add(nutrientIdField, gbcFoods);

// Add Protein label and text field
JLabel proteinLabel = new JLabel("Protein:");
gbcFoods.gridx = 0;
gbcFoods.gridy = 7;
foodsPanel.add(proteinLabel, gbcFoods);

JTextField proteinField = new JTextField(10);
gbcFoods.gridx = 1;
gbcFoods.gridy = 7;
foodsPanel.add(proteinField, gbcFoods);

// Add Sugar label and text field
```

```
JLabel sugarLabel = new JLabel("Sugar:");
gbcFoods.gridx = 0;
gbcFoods.gridy = 8;
foodsPanel.add(sugarLabel, gbcFoods);

JTextField sugarField = new JTextField(10);
gbcFoods.gridx = 1;
gbcFoods.gridy = 8;
foodsPanel.add(sugarField, gbcFoods);

// Add Fiber label and text field
JLabel fiberLabel = new JLabel("Fiber:");
gbcFoods.gridx = 0;
gbcFoods.gridy = 9;
foodsPanel.add(fiberLabel, gbcFoods);

JTextField fiberField = new JTextField(10);
gbcFoods.gridx = 1;
gbcFoods.gridy = 9;
foodsPanel.add(fiberField, gbcFoods);

JButton foodUpdateButton = new JButton("Update");
JButton foodDeleteButton = new JButton("Delete");
// Add Update button
gbcFoods.gridx = 0;
gbcFoods.gridy = 10;
foodsPanel.add(foodUpdateButton, gbcFoods);

// Add Delete button
gbcFoods.gridx = 1;
foodsPanel.add(foodDeleteButton, gbcFoods);
```

```
// Hide the input fields initially
foodIdLabel.setVisible(false);
foodIdField.setVisible(false);
foodNameLabel.setVisible(false);
foodNameField.setVisible(false);
caloriesLabel.setVisible(false);
caloriesField.setVisible(false);
foodweightLabel.setVisible(false);
foodweightField.setVisible(false);
submitfoodButton.setVisible(false);
nutrientIdLabel.setVisible(false);
nutrientIdField.setVisible(false);
proteinLabel.setVisible(false);
proteinField.setVisible(false);
foodUpdateButton.setVisible(false);
foodDeleteButton.setVisible(false);
sugarLabel.setVisible(false);
sugarField.setVisible(false);
fiberLabel.setVisible(false);
fiberField.setVisible(false);

userIdComboBox.addActionListener(new ActionListener() {
    @Override
    public void actionPerformed(ActionEvent e) {
        // Show or hide the input fields based on user ID selection
        boolean showInputs = userIdComboBox.getSelectedItem() != null;
                        userIdComboBox.setVisible(false);
                        userLabel.setVisible(false);
        foodIdLabel.setVisible(showInputs);
        foodIdField.setVisible(showInputs);
        foodNameLabel.setVisible(showInputs);
        foodNameField.setVisible(showInputs);
```

```
        caloriesLabel.setVisible(showInputs);
        caloriesField.setVisible(showInputs);
        foodweightLabel.setVisible(showInputs);
        foodweightField.setVisible(showInputs);
        submitfoodButton.setVisible(showInputs);
        nutrientIdLabel.setVisible(showInputs);
        nutrientIdField.setVisible(showInputs);
        proteinLabel.setVisible(showInputs);
        proteinField.setVisible(showInputs);
        sugarLabel.setVisible(showInputs);
        sugarField.setVisible(showInputs);
        fiberLabel.setVisible(showInputs);
        fiberField.setVisible(showInputs);
                foodUpdateButton.setVisible(showInputs);
        foodDeleteButton.setVisible(showInputs);
    }
});

// Add the tabbed pane to the main content pane
//GridBagConstraints tabbedPaneConstraints = new GridBagConstraints();
tabbedPaneConstraints.gridx = 0;
tabbedPaneConstraints.gridy = 3;
tabbedPaneConstraints.fill = GridBagConstraints.BOTH;
tabbedPaneConstraints.weightx = 1.0;
tabbedPaneConstraints.weighty = 1.0;
tabbedPaneConstraints.insets = new Insets(10, 10, 10, 10);


tabbedPane.setVisible(false); // Initially hide the tabbed pane
add(tabbedPane, tabbedPaneConstraints);
tabbedPane.addTab("Users", usersPanel);
tabbedPane.addTab("Foods", foodsPanel);
```

```
//ACTION LISTENERS
   //SUBMIT USER DATA
                    submitButton.addActionListener(new ActionListener() {
                        @Override
                        public void actionPerformed(ActionEvent e) {
                            Users users = new Users();
                            int userId = Integer.parseInt(userIdField.getText());
                            String name = nameField.getText();
                            int age = Integer.parseInt(ageField.getText());
                            double weight = Double.parseDouble(weightField.getText());
                            double height = Double.parseDouble(heightField.getText());



                            if (name.isEmpty() || userIdField.getText().isEmpty() ||
ageField.getText().isEmpty() || weightField.getText().isEmpty() || heightField.getText().isEmpty()) {
                        JOptionPane.showMessageDialog(HomePage.this, "Fill all details",
"Warning", JOptionPane.WARNING_MESSAGE);
                            } else {



                        // Call the insertIntoUsers method from Users class
                            users.insertUser(userId, name, age, weight, height);


                        JOptionPane.showMessageDialog(HomePage.this, "Data inserted
successfully", "Success", JOptionPane.INFORMATION_MESSAGE);
                            }


                            }
                        });

        //DELETE USERS BUTTON
        deleteButton.addActionListener(new ActionListener() {
```

```
    @Override
    public void actionPerformed(ActionEvent e) {
        // Fetch the available user IDs from the database
                    Users users = new Users();
        List<Integer> userIDs = users.getUserIDs();

        if (userIDs.isEmpty()) {
            JOptionPane.showMessageDialog(HomePage.this, "No users found", "Error",
JOptionPane.ERROR_MESSAGE);
        } else {
            // Create a JComboBox with the available user IDs
            JComboBox<Integer> idComboBox = new JComboBox<>(userIDs.toArray(new Integer[0]));

            // Show an input dialog with the ID dropdown
            int option = JOptionPane.showOptionDialog(HomePage.this, idComboBox, "Select User ID to
delete", JOptionPane.OK_CANCEL_OPTION, JOptionPane.PLAIN_MESSAGE, null, null, null);

            if (option == JOptionPane.OK_OPTION) {
                // Get the selected user ID
                int selectedUserID = (int) idComboBox.getSelectedItem();

                // Call the deleteFromUsers method from Users class
                users.deleteUser(selectedUserID);

                JOptionPane.showMessageDialog(HomePage.this, "User deleted successfully", "Success",
JOptionPane.INFORMATION_MESSAGE);
            }
        }
    }
});
         foodDeleteButton.addActionListener(new ActionListener() {
    @Override
    public void actionPerformed(ActionEvent e) {
```

```
List<String> foodIds = food.getFoodIdsFromDatabase();

// Create a dropdown menu for food IDs
JComboBox<String> foodIdDropdown = new JComboBox<>();
for (String foodId : foodIds) {
    foodIdDropdown.addItem(foodId);
}

// Create a panel to hold the dropdown
JPanel deletePanel = new JPanel();
deletePanel.add(new JLabel("Select Food ID:"));
deletePanel.add(foodIdDropdown);

// Show the confirmation dialog with the dropdown
int result = JOptionPane.showConfirmDialog(null, deletePanel, "Delete Food",
        JOptionPane.OK_CANCEL_OPTION, JOptionPane.PLAIN_MESSAGE);

if (result == JOptionPane.OK_OPTION) {
    String selectedFoodId = (String) foodIdDropdown.getSelectedItem();

    // Call the deleteFood method in Food.java with the selected food ID
    food.deleteFood(selectedFoodId);

    // Clear the input fields
    foodIdField.setText("");
    foodNameField.setText("");
    caloriesField.setText("");
    foodweightField.setText("");
    nutrientIdField.setText("");
    proteinField.setText("");
    sugarField.setText("");
    fiberField.setText("");
```

```
        // Show a confirmation message to the user
        JOptionPane.showMessageDialog(null, "Food deleted successfully!", "Delete",
JOptionPane.INFORMATION_MESSAGE);
      }
    }
});


         submitfoodButton.addActionListener(new ActionListener() {
        @Override
        public void actionPerformed(ActionEvent e) {
            // Check if all fields are filled
            if (foodIdField.getText().isEmpty() || foodNameField.getText().isEmpty() ||
                caloriesField.getText().isEmpty() || foodweightField.getText().isEmpty() ||
                nutrientIdField.getText().isEmpty() || proteinField.getText().isEmpty() ||
                sugarField.getText().isEmpty() || fiberField.getText().isEmpty()) {
                // Display a warning message if any field is empty
                JOptionPane.showMessageDialog(null, "Please fill in all fields", "Warning",
JOptionPane.WARNING_MESSAGE);
            } else {
                // Get the selected user ID from the combobox
                int selectedUserId = Integer.parseInt(userIdComboBox.getSelectedItem().toString());


                try {
                    // Insert data into foods table
                    int foodId = Integer.parseInt(foodIdField.getText());
                    String foodName = foodNameField.getText();
                    int calories = Integer.parseInt(caloriesField.getText());
                    int weightInGrams = Integer.parseInt(foodweightField.getText());
                    food.insertFood(foodId, foodName, calories, weightInGrams, selectedUserId);


                    // Insert data into nutrients table
                    int nutrientId = Integer.parseInt(nutrientIdField.getText());
```

```
                int protein = Integer.parseInt(proteinField.getText());
                int sugar = Integer.parseInt(sugarField.getText());
                int fiber = Integer.parseInt(fiberField.getText());
                food.insertNutrient(protein, sugar, fiber, nutrientId, selectedUserId, foodId);

                // Clear the input fields
                foodIdField.setText("");
                foodNameField.setText("");
                caloriesField.setText("");
                foodweightField.setText("");
                nutrientIdField.setText("");
                proteinField.setText("");
                sugarField.setText("");
                fiberField.setText("");

                // Display a success message
                JOptionPane.showMessageDialog(null, "Data inserted successfully", "Success",
JOptionPane.INFORMATION_MESSAGE);
            } catch (SQLException ex) {
                // Handle any database errors
                ex.printStackTrace();
                JOptionPane.showMessageDialog(null, "Error inserting data", "Error",
JOptionPane.ERROR_MESSAGE);
            }
        }
                        }
        });

        //UPDATE USERS
        updateButton.addActionListener(new ActionListener() {
    @Override
    public void actionPerformed(ActionEvent e) {
        // Get the available user IDs
```

```
List<Integer> userIDs = users.getUserIDs();

// Show a dialog to choose the user ID
Integer selectedUserID = (Integer) JOptionPane.showInputDialog(HomePage.this,
        "Select a User ID:", "Update User", JOptionPane.PLAIN_MESSAGE,
        null, userIDs.toArray(), userIDs.get(0));

if (selectedUserID != null) {
    // Show an input dialog to choose the field to update
    String[] fields = { "Name", "Age", "Weight", "Height" };
    String selectedField = (String) JOptionPane.showInputDialog(HomePage.this,
            "Select a field to update:", "Update User", JOptionPane.PLAIN_MESSAGE,
            null, fields, fields[0]);

    if (selectedField != null) {
        // Show an input dialog to enter the new value
        String newValue = JOptionPane.showInputDialog(HomePage.this,
                "Enter the new value for " + selectedField + ":", "Update User",
                JOptionPane.PLAIN_MESSAGE);

        if (newValue != null) {
            // Perform the update operation
            users.updateUser(selectedUserID, selectedField, newValue);

            // Show appropriate message
            JOptionPane.showMessageDialog(HomePage.this,
                    "User details updated successfully", "Update Successful",
                    JOptionPane.INFORMATION_MESSAGE);
        } else {
            // User canceled entering the new value
            JOptionPane.showMessageDialog(HomePage.this,
                    "Update canceled", "Update Failed",
```

```
                    JOptionPane.WARNING_MESSAGE);
            }
        } else {
            // User canceled selecting the field to update
            JOptionPane.showMessageDialog(HomePage.this,
                    "Update canceled", "Update Failed",
                    JOptionPane.WARNING_MESSAGE);
        }
    } else {
        // User canceled selecting the User ID
        JOptionPane.showMessageDialog(HomePage.this,
                "Update canceled", "Update Failed",
                JOptionPane.WARNING_MESSAGE);
    }
  }
});

//VIEW USERS
         // Action listener for View Users button
viewUserButton.addActionListener(new ActionListener() {
  public void actionPerformed(ActionEvent e) {
    // Call the getUsersFromDatabase() method to retrieve all rows from the users table
    List<Map<String, Object>> userList = users.getUsersFromDatabase();
    // Create a two-dimensional array to hold the table data
    Object[][] tableData = new Object[userList.size()][5];

    // Populate the table data array with user information
    for (int i = 0; i < userList.size(); i++) {
        Map<String, Object> user = userList.get(i);
        tableData[i][0] = user.get("user_id");
        tableData[i][1] = user.get("name");
        tableData[i][2] = user.get("age");
```

```
            tableData[i][3] = user.get("weight");
            tableData[i][4] = user.get("height");
        }
        // Create an array of column names
        String[] columnNames = { "User ID", "Name", "Age", "Weight", "Height" };
        // Create a new JTable with the table data and column names
        JTable table = new JTable(tableData, columnNames);
        // Customize the table appearance if needed
        table.getTableHeader().setFont(new Font("SansSerif", Font.BOLD, 12));
        table.setFont(new Font("SansSerif", Font.PLAIN, 12));
        // Create a JScrollPane to hold the table
        JScrollPane scrollPane = new JScrollPane(table);
        // Show the table in a dialog box
        JOptionPane.showMessageDialog(null, scrollPane, "Users", JOptionPane.PLAIN_MESSAGE);
    }
});


            foodUpdateButton.addActionListener(new ActionListener() {
    public void actionPerformed(ActionEvent e) {
        // Display a message dialog to ask the user to select table (foods or nutrients)
        String[] options = {"Foods", "Nutrients"};
                    List<String> foodIds = food.getFoodIdsFromDatabase();
                    List<String> nutrientIds = food.getNutrientIdsFromDatabase();
        int tableChoice = JOptionPane.showOptionDialog(
                null,
                "Select a table to update:",
                "Table Selection",
                JOptionPane.DEFAULT_OPTION,
                JOptionPane.QUESTION_MESSAGE,
                null,
                options,
```

```
        options[0]
    );
    if (tableChoice == 0) {
        // User selected Foods table
        String selectedFoodId = (String) JOptionPane.showInputDialog(
                null,
                "Select a food ID:",
                "Food ID Selection",
                JOptionPane.PLAIN_MESSAGE,
                null,
                foodIds.toArray(),
                null
        );

        if (selectedFoodId != null) {
            String[] fieldOptions = {"food_name", "calories", "weight_in_grams"};
            String selectedField = (String) JOptionPane.showInputDialog(
                    null,
                    "Select a field to update:",
                    "Field Selection",
                    JOptionPane.PLAIN_MESSAGE,
                    null,
                    fieldOptions,
                    null
            );

            if (selectedField != null) {
                String newValue = JOptionPane.showInputDialog(
                        null,
                        "Enter the new value:",
                        "New Value",
                        JOptionPane.PLAIN_MESSAGE
```

```
            );

            if (newValue != null) {
                // Call the updateFood method with the selected food ID, field, and new value
                food.updateFood(selectedFoodId, selectedField, newValue);
            }
        }
    }
} else if (tableChoice == 1) {
    // User selected Nutrients table
    String selectedNutrientId = (String) JOptionPane.showInputDialog(
            null,
            "Select a nutrient ID:",
            "Nutrient ID Selection",
            JOptionPane.PLAIN_MESSAGE,
            null,
            nutrientIds.toArray(),
            null
    );

    if (selectedNutrientId != null) {
        String[] fieldOptions = {"protein", "sugar", "fiber"};
        String selectedField = (String) JOptionPane.showInputDialog(
                null,
                "Select a field to update:",
                "Field Selection",
                JOptionPane.PLAIN_MESSAGE,
                null,
                fieldOptions,
                null
        );
```

```
            if (selectedField != null) {
                String newValue = JOptionPane.showInputDialog(
                        null,
                        "Enter the new value:",
                        "New Value",
                        JOptionPane.PLAIN_MESSAGE
                );

                if (newValue != null) {
                    // Call the updateNutrient method with the selected nutrient ID, field, and new value
                    food.updateNutrient(selectedNutrientId, selectedField, newValue);
                }
            }
        }
    }
}});

        }
    public static void main(String[] args) {
        SwingUtilities.invokeLater(HomePage::new);
    }
}
```

USERS TABLE:

```
import java.sql.Connection;
import java.sql.DriverManager;
import java.sql.PreparedStatement;
import java.sql.ResultSet;
import java.sql.SQLException;
import java.sql.*;
import java.util.ArrayList;
```

```
import java.util.List;
import java.util.*;

public class Users {
    private Connection connection;

    public Users() {
        try {
            Class.forName("oracle.jdbc.OracleDriver");
            connection = DriverManager.getConnection("jdbc:oracle:thin:@localhost:1521:xe",
"piyush", "piyush");
        } catch (ClassNotFoundException e) {
            System.out.println("Oracle JDBC driver not found.");
        } catch (SQLException e) {
            System.out.println("Error connecting to the database: " + e.getMessage());
        }
    }

    public void insertUser(int userId, String name, int age, double weight, double height) {
        try {
            String query = "INSERT INTO users (user_id, name, age, weight, height) VALUES
(?, ?, ?, ?, ?)";
            PreparedStatement pstmt = connection.prepareStatement(query);
            pstmt.setInt(1, userId);
            pstmt.setString(2, name);
            pstmt.setInt(3, age);
            pstmt.setDouble(4, weight);
            pstmt.setDouble(5, height);

            int rowsAffected = pstmt.executeUpdate();
            pstmt.close();
```

```
        if (rowsAffected > 0) {
            System.out.println("User inserted successfully!");
        } else {
            System.out.println("User insertion failed.");
        }
    } catch (SQLException e) {
        System.out.println("Error inserting user: " + e.getMessage());
    }
}

public void deleteUser(int userId) {
    try {
        String query = "DELETE FROM users WHERE user_id = ?";
        PreparedStatement pstmt = connection.prepareStatement(query);
        pstmt.setInt(1, userId);

        int rowsAffected = pstmt.executeUpdate();
        pstmt.close();

        if (rowsAffected > 0) {
            System.out.println("User deleted successfully!");
        } else {
            System.out.println("User deletion failed. User not found.");
        }
    } catch (SQLException e) {
        System.out.println("Error deleting user: " + e.getMessage());
    }
}
```

```
    public void updateUser(int userId, String field, Object newValue) {
    try {
        String query = "UPDATE users SET " + field.toLowerCase() + " = ? WHERE user_id =
?";
        PreparedStatement pstmt = connection.prepareStatement(query);

        if (newValue instanceof String) {
            pstmt.setString(1, (String) newValue);
        } else if (newValue instanceof Integer) {
            pstmt.setInt(1, (Integer) newValue);
        } else if (newValue instanceof Double) {
            pstmt.setDouble(1, (Double) newValue);
        }

        pstmt.setInt(2, userId);

        int rowsAffected = pstmt.executeUpdate();
        pstmt.close();

        if (rowsAffected > 0) {
            System.out.println("User updated successfully!");
        } else {
            throw new SQLException("User update failed. User not found.");
        }
    } catch (SQLException e) {
        System.out.println("Error updating user: " + e.getMessage());

    }
}

            public  List<Integer> getUserIDs() {
```

```
        List<Integer> userIDs = new ArrayList<>();

        // Establish a connection to the database
        try {
            //Class.forName("oracle.jdbc.OracleDriver");
            //Connection con =
DriverManager.getConnection("jdbc:oracle:thin:@localhost:1521:xe", "phani", "phani");

            // Execute a SELECT query to retrieve user IDs
            String query = "SELECT user_id FROM users";
            Statement stmt = connection.createStatement();
            ResultSet rs = stmt.executeQuery(query);

            // Iterate over the result set and add user IDs to the list
            while (rs.next()) {
                int userID = rs.getInt("user_id");
                userIDs.add(userID);
            }

            // Close the database resources
            rs.close();
            stmt.close();
            //con.close();
        } catch (Exception e) {
            e.printStackTrace();
        }

        return userIDs;
    }
            public List<Map<String, Object>> getUsersFromDatabase() {
    List<Map<String, Object>> users = new ArrayList<>();
```

```
    try {
        Statement statement = connection.createStatement();

        // Execute the query to retrieve users from the database
        String query = "SELECT * FROM users";
        ResultSet resultSet = statement.executeQuery(query);

        // Iterate over the result set and populate the users list
        while (resultSet.next()) {
            Map<String, Object> user = new HashMap<>();
            user.put("user_id", resultSet.getInt("user_id"));
            user.put("name", resultSet.getString("name"));
            user.put("age", resultSet.getInt("age"));
            user.put("weight", resultSet.getDouble("weight"));
            user.put("height", resultSet.getDouble("height"));
            users.add(user);
        }

        // Close the resources
        resultSet.close();
        statement.close();
        //connection.close();
    } catch (SQLException e) {
        e.printStackTrace();
    }
    return users;
}
}
```

FOODS AND NUTRIENTS TABLE:

```
import java.sql.Connection;

import java.sql.PreparedStatement;

import java.sql.SQLException;

import java.sql.*;

import java.util.ArrayList;

import java.util.*;

public class Food {

    private Connection connection;

   public Food() {


        try {

            Class.forName("oracle.jdbc.OracleDriver");

            connection = DriverManager.getConnection("jdbc:oracle:thin:@localhost:1521:xe",
"piyush", "piyush");

        } catch (ClassNotFoundException e) {

            System.out.println("Oracle JDBC driver not found.");

        } catch (SQLException e) {

            System.out.println("Error connecting to the database: " + e.getMessage());

        }

    }


    public void insertFood(int foodId, String foodName, int calories, int weightInGrams, int
userId) throws SQLException {

        String query = "INSERT INTO foods (food_id, food_name, calories, weight_in_grams,
user_id) VALUES (?, ?, ?, ?, ?)";

        try (PreparedStatement statement = connection.prepareStatement(query)) {

            statement.setInt(1, foodId);

            statement.setString(2, foodName);

            statement.setInt(3, calories);

            statement.setInt(4, weightInGrams);
```

```
            statement.setInt(5, userId);
            statement.executeUpdate();
        }
    }

    public void insertNutrient(int protein, int sugar, int fiber, int nutrientId, int userId, int
foodId) throws SQLException {
        String query = "INSERT INTO nutrients (protein, sugar, fiber, nutrient_id, user_id,
food_id) VALUES (?, ?, ?, ?, ?, ?)";
        try (PreparedStatement statement = connection.prepareStatement(query)) {
            statement.setInt(1, protein);
            statement.setInt(2, sugar);
            statement.setInt(3, fiber);
            statement.setInt(4, nutrientId);
            statement.setInt(5, userId);
            statement.setInt(6, foodId);
            statement.executeUpdate();
        }
    }
             public List<Map<String, Object>> getFoodsFromDatabase() {
        List<Map<String, Object>> foods = new ArrayList<>();

        try {
            // Establish the database connection
            //Connection connection = DriverManager.getConnection(DB_URL,
DB_USERNAME, DB_PASSWORD);
            Statement statement = connection.createStatement();

            // Execute the query to retrieve foods from the database
            String query = "SELECT * FROM foods";
            ResultSet resultSet = statement.executeQuery(query);
```

```
            // Iterate over the result set and populate the foods list
            while (resultSet.next()) {
                Map<String, Object> food = new HashMap<>();
                food.put("food_id", resultSet.getInt("food_id"));
                food.put("food_name", resultSet.getString("food_name"));
                food.put("calories", resultSet.getDouble("calories"));
                food.put("weight_in_grams", resultSet.getDouble("weight_in_grams"));
                food.put("user_id", resultSet.getInt("user_id"));
                foods.add(food);
            }

            // Close the resources
            resultSet.close();
            statement.close();
            connection.close();
        } catch (SQLException e) {
            e.printStackTrace();
        }

        return foods;
    }

    public List<Map<String, Object>> getNutrientsFromDatabase() {
        List<Map<String, Object>> nutrients = new ArrayList<>();

        try {

            //Connection connection = DriverManager.getConnection(DB_URL,
DB_USERNAME, DB_PASSWORD);
            Statement statement = connection.createStatement();
```

```
        // Execute the query to retrieve nutrients from the database
        String query = "SELECT * FROM nutrients";
        ResultSet resultSet = statement.executeQuery(query);

        // Iterate over the result set and populate the nutrients list
        while (resultSet.next()) {
            Map<String, Object> nutrient = new HashMap<>();
            nutrient.put("nutrient_id", resultSet.getInt("nutrient_id"));
            nutrient.put("protein", resultSet.getDouble("protein"));
            nutrient.put("sugar", resultSet.getDouble("sugar"));
            nutrient.put("fiber", resultSet.getDouble("fiber"));
            nutrient.put("user_id", resultSet.getInt("user_id"));
            nutrient.put("food_id", resultSet.getInt("food_id"));
            nutrients.add(nutrient);
        }

        // Close the resources
        resultSet.close();
        statement.close();
        connection.close();
    } catch (SQLException e) {
        e.printStackTrace();
    }

    return nutrients;
}


    public List<String> getFoodIdsFromDatabase() {
```

```
List<String> foodIDs = new ArrayList<>();
ResultSet resultSet = null;
Statement statement = null;

try {
                connection =
DriverManager.getConnection("jdbc:oracle:thin:@localhost:1521:xe", "piyush", "piyush");

    statement = connection.createStatement();

    // Select all food IDs from the foods table
    String selectFoodIDsQuery = "SELECT food_id FROM foods";
     resultSet = statement.executeQuery(selectFoodIDsQuery);

    // Iterate over the result set and add each food ID to the list
    while (resultSet.next()) {
        String foodID = resultSet.getString("food_id");
        foodIDs.add(foodID);
    }

} catch (SQLException e) {
    e.printStackTrace();
    // Handle any database errors or exceptions
} finally {
    // Close the result set, statement, and connection
    if (resultSet != null) {
        try {
            resultSet.close();
        } catch (SQLException e) {
            e.printStackTrace();
        }
```

```
        }
        if (statement != null) {
            try {
                statement.close();
            } catch (SQLException e) {
                e.printStackTrace();
            }
        }
        if (connection != null) {
            try {
                connection.close();
            } catch (SQLException e) {
                e.printStackTrace();
            }
        }
    }

    return foodIDs;
}
            public List<String> getNutrientIdsFromDatabase() {
    List<String> nutrientIds = new ArrayList<>();
            ResultSet resultSet = null;
    Statement statement = null;



    try {
                        connection =
DriverManager.getConnection("jdbc:oracle:thin:@localhost:1521:xe", "piyush", "piyush");
        statement = connection.createStatement();

        // Select all nutrient IDs from the nutrients table
```

```
    String selectNutrientIDsQuery = "SELECT nutrient_id FROM nutrients";
    resultSet = statement.executeQuery(selectNutrientIDsQuery);

    // Iterate over the result set and add each nutrient ID to the list
    while (resultSet.next()) {
        String nutrientID = resultSet.getString("nutrient_id");
        nutrientIds.add(nutrientID);
    }

} catch (SQLException e) {
    e.printStackTrace();
    // Handle any database errors or exceptions
} finally {
    // Close the result set, statement, and connection
    if (resultSet != null) {
        try {
            resultSet.close();
        } catch (SQLException e) {
            e.printStackTrace();
        }
    }
    if (statement != null) {
        try {
            statement.close();
        } catch (SQLException e) {
            e.printStackTrace();
        }
    }

}
```

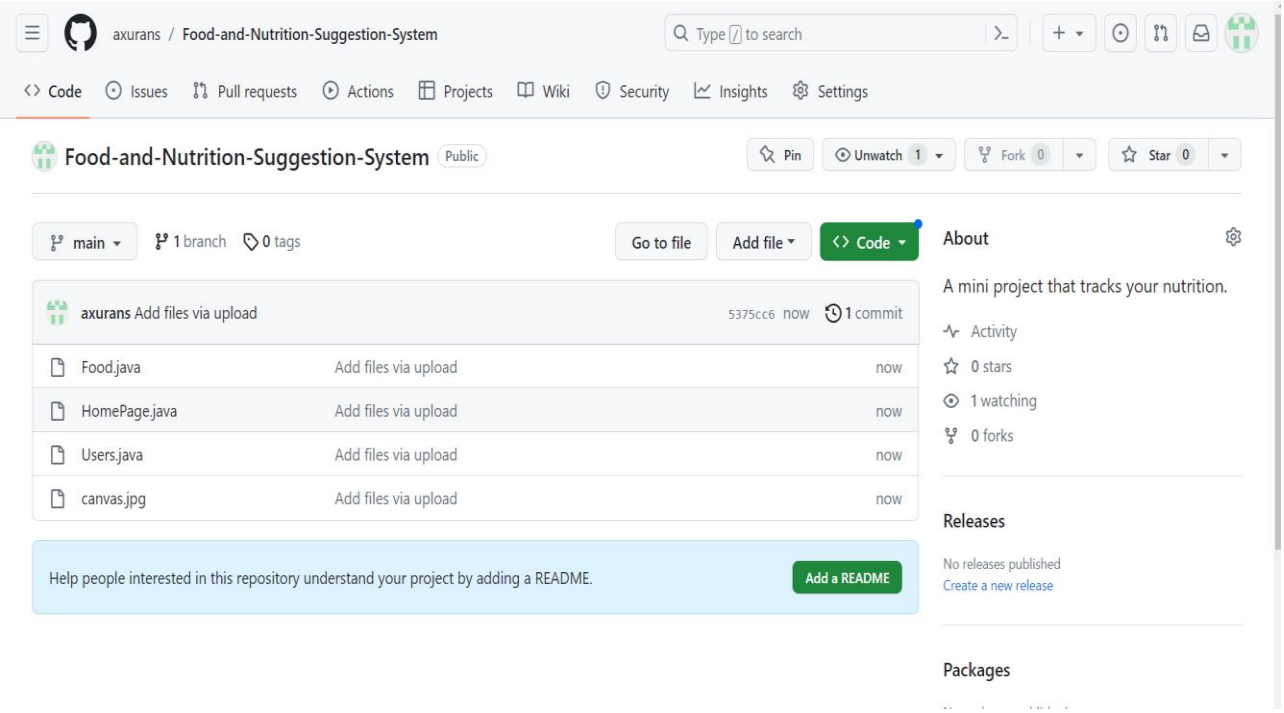
```
    return nutrientIds;
}




        public void deleteFood(String foodId) {
    PreparedStatement statement = null;

    try {
                    connection =
DriverManager.getConnection("jdbc:oracle:thin:@localhost:1521:xe", "piyush", "piyush");
        // Delete the food record from the foods table
        String deleteFoodQuery = "DELETE FROM foods WHERE food_id = ?";
        statement = connection.prepareStatement(deleteFoodQuery);
        statement.setString(1, foodId);
        int foodRowsAffected = statement.executeUpdate();

        // Delete the corresponding nutrient record from the nutrients table
        String deleteNutrientQuery = "DELETE FROM nutrients WHERE food_id = ?";
        statement = connection.prepareStatement(deleteNutrientQuery);
        statement.setString(1, foodId);
        int nutrientRowsAffected = statement.executeUpdate();

        if (foodRowsAffected > 0 || nutrientRowsAffected > 0) {
            System.out.println("Food with ID " + foodId + " deleted successfully!");
        } else {
            System.out.println("No food found with the specified ID.");
        }

    } catch (SQLException e) {
```

```
        e.printStackTrace();
        // Handle any database errors or exceptions
    } finally {
        // Close the statement
        if (statement != null) {
            try {
                statement.close();
            } catch (SQLException e) {
                e.printStackTrace();
            }
        }
    }
}
                    public void updateFood(String selectedFoodId, String selectedField, String
newValue) {
    try {
        Statement statement = connection.createStatement();


        // Update the selected field of the specified food ID in the foods table
        String updateFoodQuery = "UPDATE foods SET " + selectedField + " = '" + newValue
+ "' WHERE food_id = '" + selectedFoodId + "'";
        statement.executeUpdate(updateFoodQuery);
        System.out.println("Food with ID " + selectedFoodId + " updated successfully.");
    } catch (SQLException e) {
        e.printStackTrace();
    }
}


            public void updateNutrient(String selectedFoodId, String selectedField, String
newValue) {
    try {
```

```
        Statement statement = connection.createStatement();

        String updateNutrientQuery = "UPDATE nutrients SET " + selectedField + " = '" +
newValue + "' WHERE food_id = '" + selectedFoodId + "'";

        statement.executeUpdate(updateNutrientQuery);

        System.out.println("Nutrient for food with ID " + selectedFoodId + " updated
successfully.");

    } catch (SQLException e) {

        e.printStackTrace();

    }

}

}
```

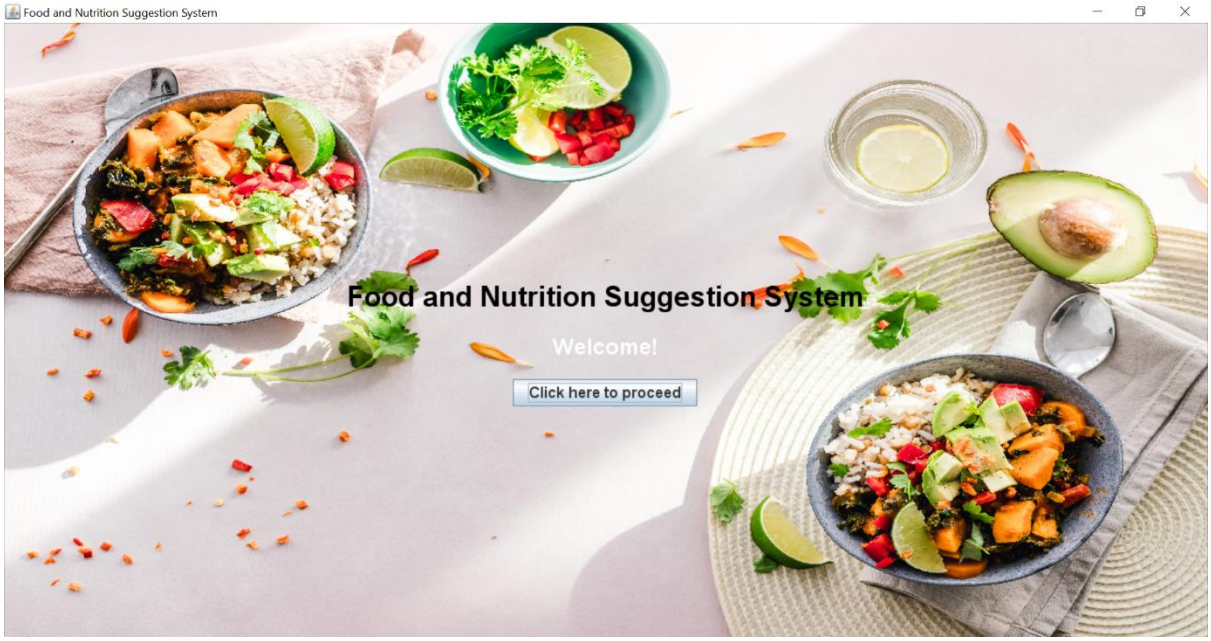
GITHUB LINK AND FOLDER STRUCTURE

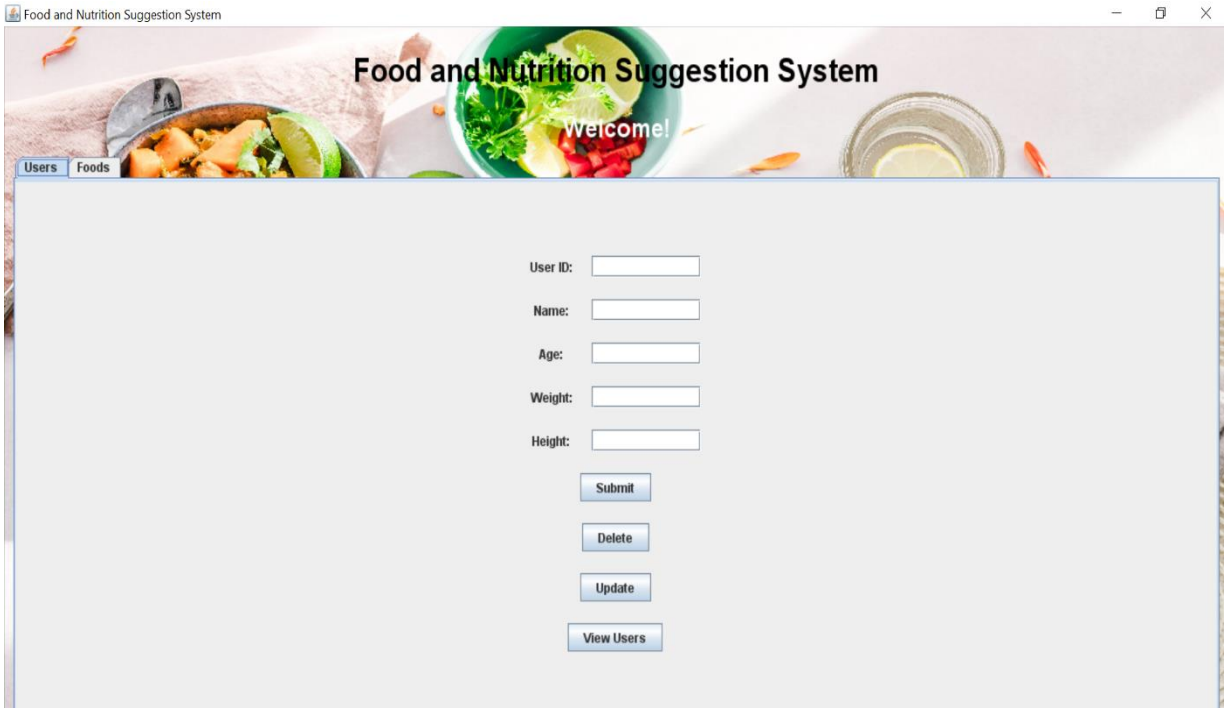
Link: https://github.com/axurans/Food-and-Nutrition-Suggestion-System



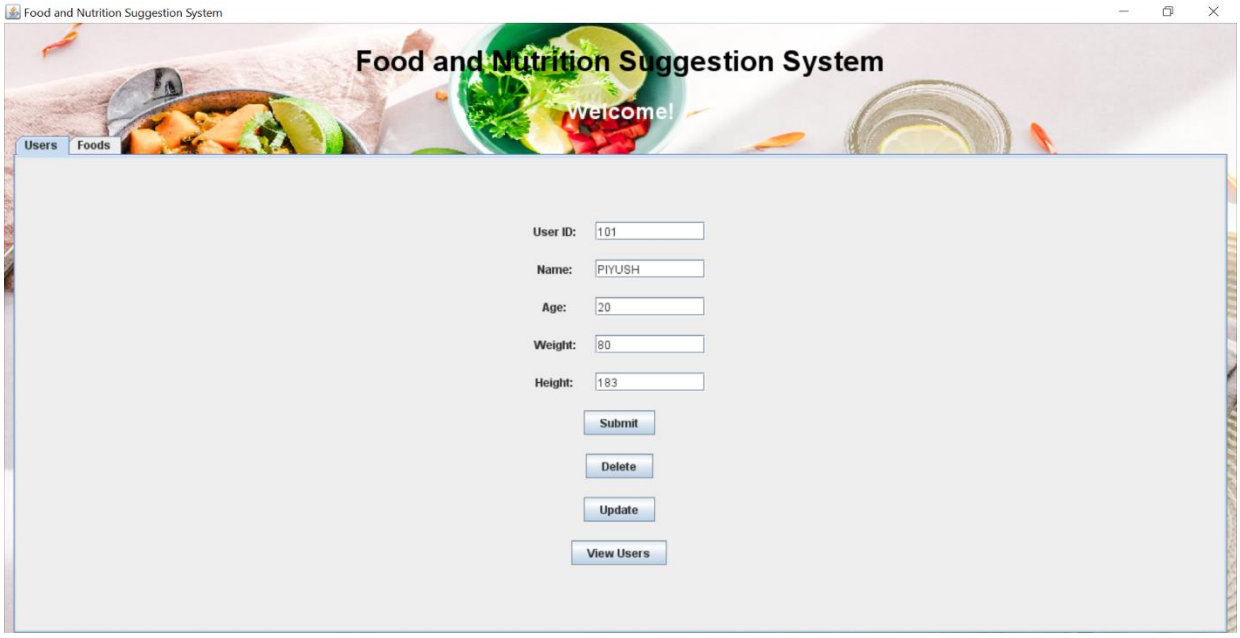
TESTING

MAIN PAGE:





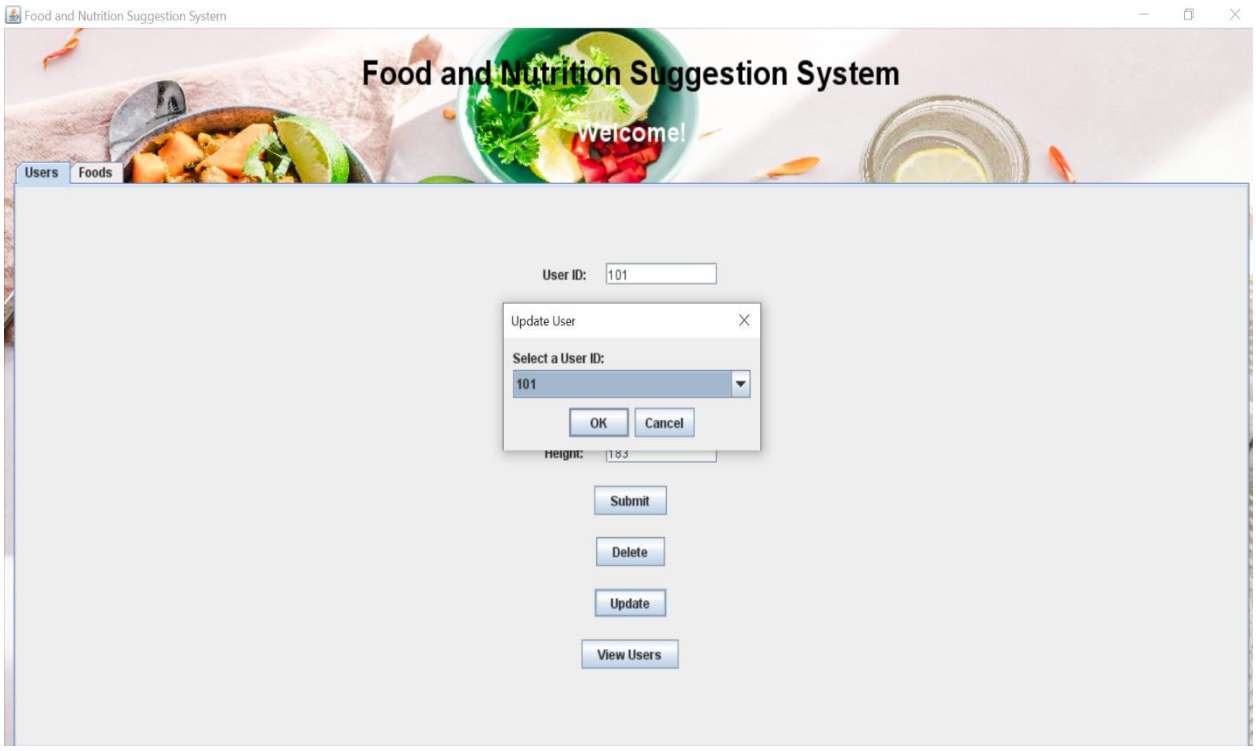
USER DATA INPUT:



DELETING USER DATA:

VIEWING USERS LIST:

UPDATING USER DATA:




Food and Nutrition Suggestion System
Food and Nutrition Suggestion System
Welcome!
Users
Foods
User ID:
101
Update User
Select a field to update:
Name
OK
Cancel
Submit
Delete
Update
View Users

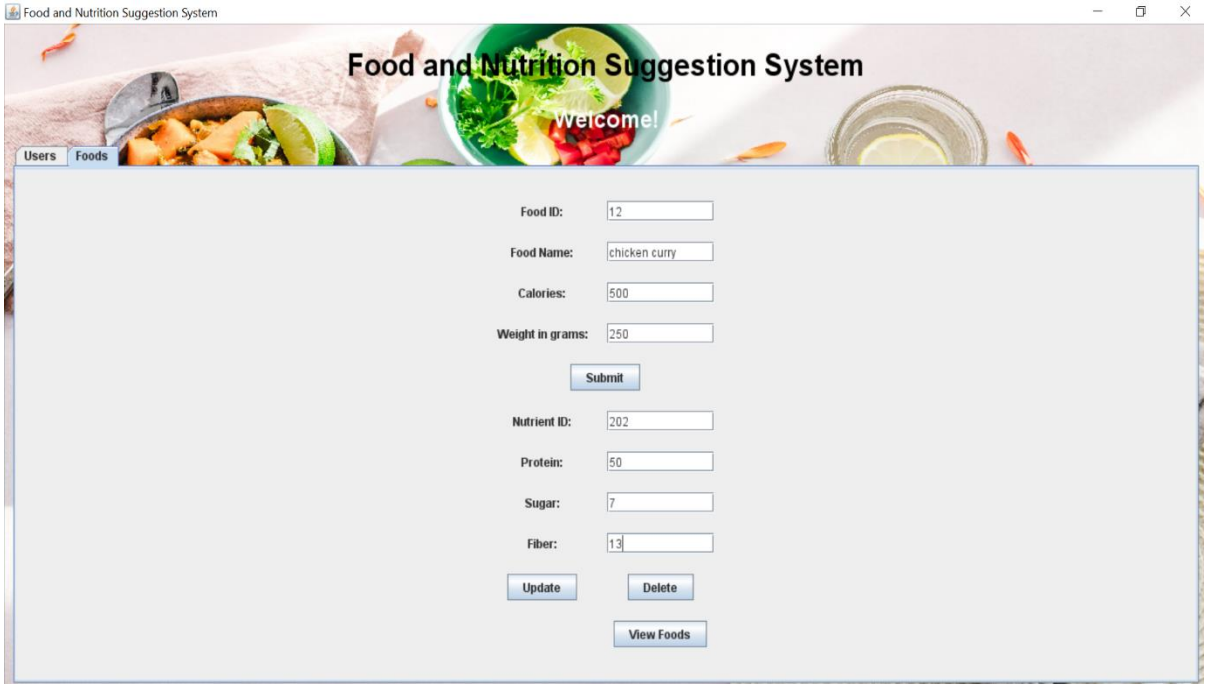

Food and Nutrition Suggestion System
Food and Nutrition Suggestion System
Welcome!
Users
Foods
User ID:
101
Update User
Select a field to update:
Weight
OK
Cancel
Submit
Delete
Update
View Users

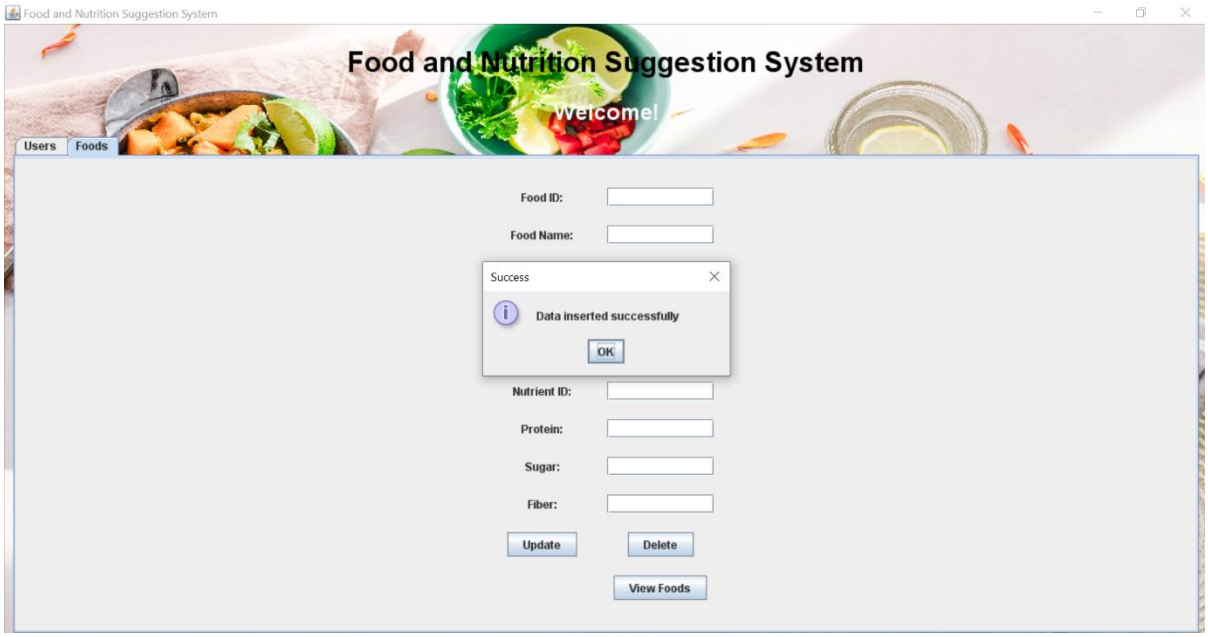

Food and Nutrition Suggestion System
Food and Nutrition Suggestion System
Welcome!
Users
Foods
User ID:
101
Update User
Enter the new value for Weight:
79
OK
Cancel
Height:
183
Submit
Delete
Update
View Users


Food and Nutrition Suggestion System
Food and Nutrition Suggestion System
Welcome!
Users
Foods
User ID:
101
Update Successful
User details updated successfully
OK
Height:
183
Submit
Delete
Update
View Users

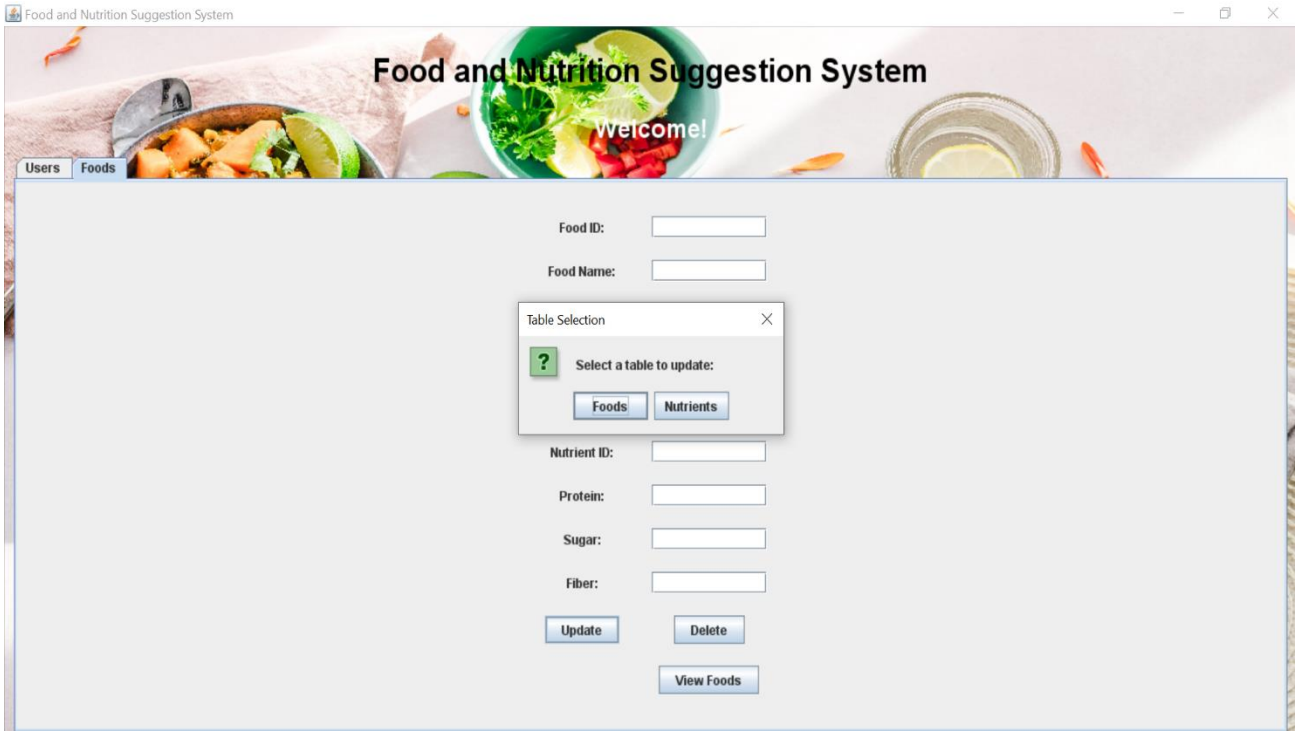
FOODS PAGE:

INSERTING DATA:



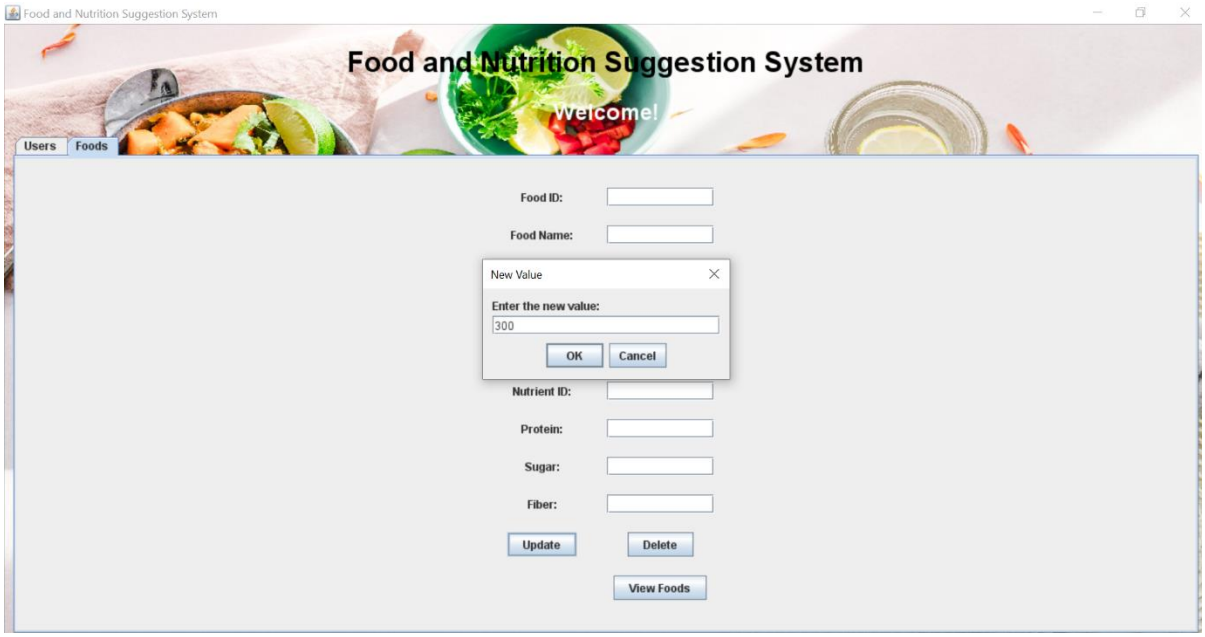


UPDATING TABLES:

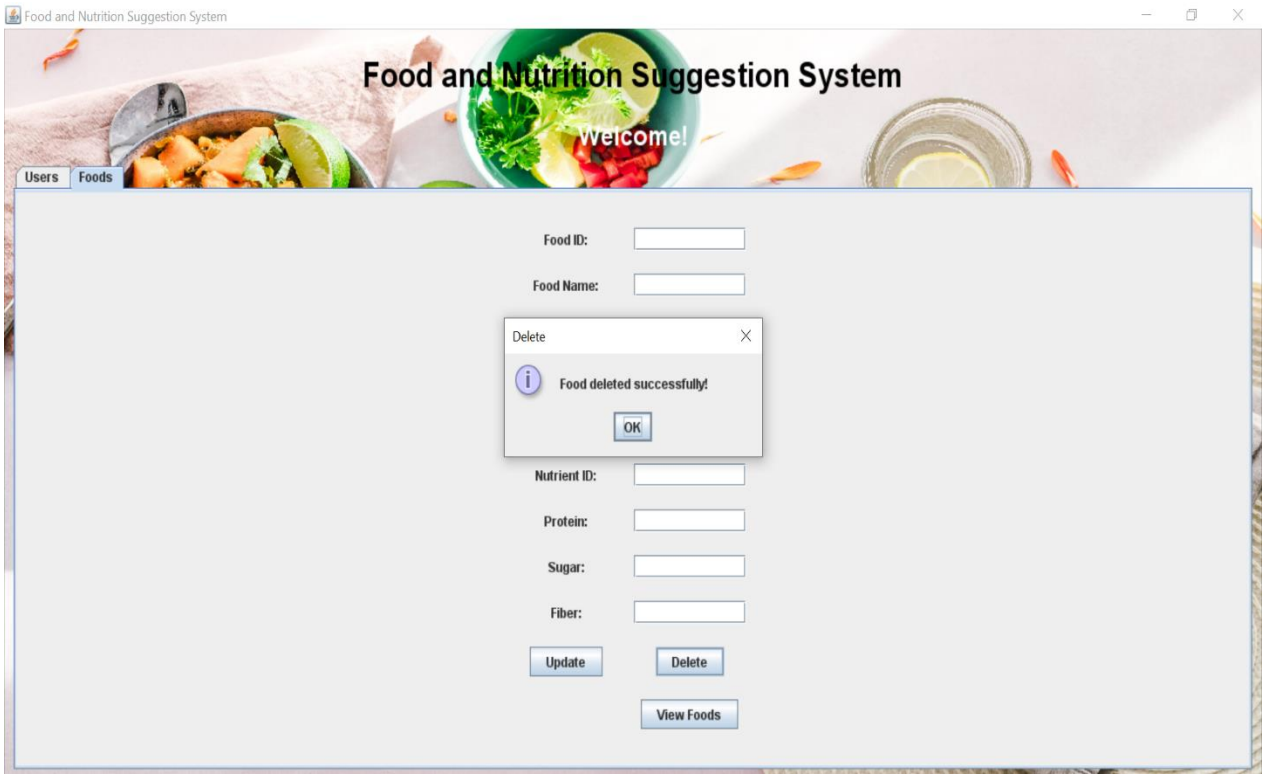



Food and Nutrition Suggestion System
Food and Nutrition Suggestion System
Welcome!
Users
Foods
Food ID:
Food Name:
Food ID Selection
Select a food ID:
12
OK
Cancel
Nutrient ID:
Protein:
Sugar:
Fiber:
Update
Delete
View Foods


Food and Nutrition Suggestion System
Food and Nutrition Suggestion System
Welcome!
Users
Foods
Food ID:
Food Name:
Field Selection
Select a field to update:
weight_in_grams
OK
Cancel
Nutrient ID:
Protein:
Sugar:
Fiber:
Update
Delete
View Foods



DELETING VALUES FROM TABLE:


Food and Nutrition Suggestion System
Food and Nutrition Suggestion System
Welcome!
Users
Foods
Food ID:
Food Name:
Delete
Food deleted successfully!
OK
Nutrient ID:
Protein:
Sugar:
Fiber:
Update
Delete
View Foods

RESULTS

I have successfully completed the mini project "**FOOD AND NUTRITION SUGGESTION SYSTEM**"

DISCUSSION AND FUTURE WORK

In the future, there is significant potential for further development and expansion of this project. Some key areas for improvement include user authentication to ensure data security, enhancing the user interface with features like data validation and search options, and integrating with external APIs for expanded food and nutrition information. Additionally, introducing meal planning and tracking functionalities, along with advanced reporting and analytics capabilities, would provide users with valuable insights and personalized recommendations. The project could also benefit from a mobile application version to increase accessibility, multi-language support for a global user base, and social sharing features to foster community engagement. Consideration for allergen and dietary restriction management, as well as collaboration among users, would further enhance the system's usefulness and user experience. With these future enhancements, the project can become a comprehensive, user-friendly, and dynamic platform for individuals seeking to manage their nutrition and make informed dietary choices.

REFERENCES

- https://docs.oracle.com/javase/8/docs/
- https://stackoverflow.com/
- https://docs.oracle.com/cd/E12151_01/index.htm
- https://docs.oracle.com/javase%2F7%2Fdocs%2Fapi%2F%2F/javax/swing/package-summary.html