

FINTNESS TRACKING SYSTEM

A MINI PROJECT REPORT

Submitted by

HARESH R (230701097)

ELUMALAI B (230701084)

In partial fulfillment for the award of the degree of

BACHELOR OF ENGINEERING

IN

COMPUTER SCIENCE



RAJALAKSHMI ENGINEERING COLLEGE (AUTONOMOUS)

THANDALAM

CHENNAI-602105

2024- 25

BONAFIDE CERTIFICATE

Certified that this project report “**FITNESS TRACKING SYSTEM**” is the bonafide work of “**HARESH R (230701097) ELUMALAI B (230701084)**”

who carried out the project work under my supervision.

Submitted for the Practical Examination held on _____

SIGNATURE

Mr.RAGHU G
Assistance **Professor**,
Computer Science and Engineering,
Rajalakshmi Engineering College
Thandalam, Chennai - 602 105

INTERNAL EXAMINER

SIGNATURE

Mrs.DHIVYA M
Assistance Professor,
Computer Science and Engineering,
Rajalakshmi Engineering College,
Thandalam, Chennai - 602 105

EXTERNAL EXAMINER

ABSTRACT:

FitLife Dashboard is a fitness tracking software developed using Java Swing for the user interface and MySQL for backend data management. The application allows users to track their daily workout activities, log the number of sets, reps, and exercises performed, and calculate the calories burned. The system incorporates user authentication, workout logging, and data visualization features using the JFreeChart library to provide visual insights into users' fitness progress. It is designed to support fitness enthusiasts by helping them maintain a record of their workouts, monitor their daily calorie intake, and track overall progress over time. The backend MySQL database efficiently stores user information and workout data, while Java Swing delivers a modern, intuitive interface. The integration of JFreeChart adds interactive charts, allowing users to see their progress through pie charts and statistics, enhancing user engagement and motivation. With features like progress tracking, workout analytics, and customizable user settings, FitLife Dashboard aims to be a valuable tool for anyone serious about their fitness journey.

TABLE OF CONTENTS

Chapter 1

1 INTRODUCTION

1.1 INTRODUCTION -----	6
1.2 OBJECTIVES -----	7
1.3 MODULES -----	7

Chapter 2

2 SYSTEM SPECIFICATIONS

2.1 HARDWARE SPECIFICATIONS -----	9
2.2 SOFTWARE SPECIFICATIONS -----	9

Chapter 3

3 MODULO DESCRIPTION -----	X
-----------------------------------	----------

Chapter 4

4 PROGRAM CODE

4.1 PROGRAM CODE-----	18
-----------------------	----

Chapter 5

5 RESULTS (SCREENSHOTS)

5.1 RESULTS AND DISCUSSION -----	27
----------------------------------	----

Chapter 6

6 CONCLUSION AND FUTURE ENHANCEMENT

6.1 CONCLUSION-----	31
---------------------	----

Chapter 7

7 REFERENCES

7.1 REFERENCES-----	32
---------------------	----

INTRODUCTION

CHAPTER 1 [1.1] -

IN TODAY'S WORLD, MAINTAINING PERSONAL HEALTH AND FITNESS IS BECOMING INCREASINGLY IMPORTANT. FITLIFE DASHBOARD IS A FITNESS TRACKING SOFTWARE DESIGNED TO HELP USERS EFFICIENTLY LOG AND MONITOR THEIR DAILY WORKOUT ROUTINES AND PROGRESS TOWARDS THEIR FITNESS GOALS.

THE APPLICATION, BUILT USING JAVA SWING FOR THE USER INTERFACE AND MYSQL FOR DATA STORAGE, PROVIDES A SIMPLE PLATFORM WHERE USERS CAN LOG EXERCISES, TRACK CALORIES BURNED, AND VIEW DETAILED STATISTICS ON THEIR FITNESS ACTIVITIES.

WITH FEATURES LIKE USER AUTHENTICATION, WORKOUT LOGGING, AND DATA VISUALIZATION USING JFREECHART, FITLIFE DASHBOARD OFFERS USERS A MODERN AND INTERACTIVE WAY TO MANAGE THEIR FITNESS JOURNEY. THIS APPLICATION HELPS USERS UNDERSTAND THEIR PROGRESS WITH DETAILED ANALYTICS PRESENTED IN A USER-FRIENDLY FORMAT, ENABLING THEM TO MAKE BETTER FITNESS DECISIONS AND STAY MOTIVATED.

CHAPTER 1 [1.2]

OBJECTIVES

THE PRIMARY OBJECTIVES OF FITLIFE DASHBOARD ARE:

1. PROVIDE A PLATFORM FOR USERS TO LOG WORKOUTS INCLUDING EXERCISE DETAILS SUCH AS SETS, REPS, DURATION, AND CALORIES BURNED.
2. OFFER A USER-FRIENDLY INTERFACE DEVELOPED IN JAVA SWING THAT MAKES IT EASY TO NAVIGATE AND USE.
3. INCORPORATE DATA VISUALIZATION USING JFREECHART TO HELP USERS UNDERSTAND THEIR PROGRESS THROUGH CHARTS AND STATISTICS.
4. IMPLEMENT A USER AUTHENTICATION SYSTEM TO SECURELY MANAGE USER DATA AND ACCESS TO PERSONAL FITNESS INFORMATION.
5. STORE USER DATA AND WORKOUT LOGS IN A MYSQL DATABASE FOR EASY RETRIEVAL AND MANAGEMENT.
6. PROVIDE USERS WITH GOAL-ORIENTED STATISTICS AND MOTIVATIONAL FEEDBACK TO KEEP THEM ON TRACK WITH THEIR FITNESS JOURNEY.

CHAPTER 1 [1.3] –

MODULES

FITLIFE DASHBOARD CONSISTS OF SEVERAL MODULES THAT WORK TOGETHER TO PROVIDE A SEAMLESS FITNESS TRACKING EXPERIENCE:

1. USER AUTHENTICATION MODULE:

ENABLES USERS TO CREATE ACCOUNTS, LOG IN, AND ACCESS THEIR FITNESS DATA SECURELY. THIS MODULE ENSURES THAT ONLY AUTHORIZED USERS CAN ACCESS THEIR PERSONAL INFORMATION.

2. WORKOUT LOGGING MODULE:

ALLOWS USERS TO INPUT DETAILS ABOUT THEIR WORKOUTS INCLUDING EXERCISES, SETS, REPS, AND CALORIES BURNED. THE SYSTEM CALCULATES THE TOTAL CALORIES AND PROVIDES USERS WITH A LOG OF THEIR ACTIVITIES.

3. DATA VISUALIZATION MODULE:

UTILIZES JFREECHART TO DISPLAY USER WORKOUT DATA IN PIE CHARTS AND OTHER VISUAL FORMATS, HELPING USERS SEE THEIR PROGRESS IN A CLEAR AND INTERACTIVE WAY.

4. USER PROFILE AND SETTINGS MODULE:

PROVIDES USERS WITH THE ABILITY TO UPDATE THEIR PERSONAL DETAILS INCLUDING WEIGHT, HEIGHT, AND AGE, WHICH ARE USED TO CALCULATE FITNESS STATISTICS. THIS MODULE ALSO INCLUDES ACCOUNT MANAGEMENT FUNCTIONS.

5. WORKOUT HISTORY MODULE:

ALLOWS USERS TO VIEW A DETAILED HISTORY OF THEIR PAST WORKOUTS, INCLUDING THE EXERCISES THEY PERFORMED AND THE CALORIES BURNED, PROVIDING A COMPREHENSIVE VIEW OF THEIR FITNESS JOURNEY.

6. STATISTICS AND GOAL TRACKING MODULE:

TRACKS AND DISPLAYS OVERALL FITNESS STATISTICS SUCH AS TOTAL WORKOUTS, CALORIES BURNED, AND PROVIDES USERS WITH FEEDBACK ON THEIR PROGRESS TOWARD THEIR FITNESS GOALS.

SYSTEM SPECIFICATIONS

CHAPTER 2 [2.1]

HARDWARE SPECIFICATIONS

THE FOLLOWING HARDWARE SPECIFICATIONS ARE REQUIRED TO RUN FITLIFE DASHBOARD EFFICIENTLY:

1. **PROCESSOR:**

INTEL CORE I3 OR HIGHER (OR EQUIVALENT AMD PROCESSOR) FOR SMOOTH PERFORMANCE AND QUICK DATA PROCESSING.

2. **MEMORY (RAM):**

MINIMUM 4 GB OF RAM TO SUPPORT JAVA SWING APPLICATIONS AND MYSQL DATABASE OPERATIONS.

3. **STORAGE:**

AT LEAST 500 MB OF AVAILABLE DISK SPACE FOR INSTALLING THE APPLICATION AND STORING WORKOUT LOGS AND USER DATA.

4. **DISPLAY RESOLUTION:**

A MINIMUM RESOLUTION OF 1024 X 768 FOR A CLEAR DISPLAY OF THE USER INTERFACE.

5. **OPERATING SYSTEM:**

COMPATIBLE WITH WINDOWS 10/11, MACOS, OR LINUX OPERATING SYSTEMS TO SUPPORT JAVA APPLICATIONS.

SOFTWARE SPECIFICATIONS

TO RUN AND DEVELOP THE FITLIFE DASHBOARD APPLICATION, THE FOLLOWING SOFTWARE SPECIFICATIONS ARE REQUIRED:

1. **JAVA DEVELOPMENT KIT (JDK):**

JAVA JDK 8 OR HIGHER IS REQUIRED TO COMPILE AND RUN THE JAVA SWING APPLICATION.

2. **JAVA SWING:**

JAVA SWING IS USED FOR BUILDING THE GRAPHICAL USER INTERFACE (GUI) OF THE APPLICATION.

3. **MYSQL DATABASE:**

MYSQL IS USED FOR STORING AND MANAGING USER DATA AND WORKOUT RECORDS. MYSQL CONNECTOR/J IS REQUIRED TO INTEGRATE JAVA WITH MYSQL DATABASE.

4. **JFREECHART LIBRARY:**

JFREECHART LIBRARY IS USED FOR GENERATING PIE CHARTS AND OTHER VISUAL DATA REPRESENTATIONS OF THE USER'S FITNESS PROGRESS.

5. **INTEGRATED DEVELOPMENT ENVIRONMENT (IDE):**

NETBEANS IS RECOMMENDED FOR JAVA DEVELOPMENT AND DEBUGGING.

6. **APACHE MAVEN :**

MAVEN CAN BE USED FOR MANAGING PROJECT DEPENDENCIES, INCLUDING THE MYSQL CONNECTOR AND JFREECHART LIBRARIES.

7. **JDBC (JAVA DATABASE CONNECTIVITY):**

REQUIRED TO CONNECT JAVA APPLICATIONS TO THE MYSQL DATABASE FOR DATA RETRIEVAL AND STORAGE.

MODULO DESCRIPTION

CHAPTER 3.0

MODULE DESCRIPTION

FITLIFE DASHBOARD CONSISTS OF SEVERAL INTERDEPENDENT MODULES THAT PROVIDE USERS WITH A SEAMLESS FITNESS TRACKING EXPERIENCE. EACH MODULE PLAYS A VITAL ROLE IN ENSURING THE FUNCTIONALITY OF THE APPLICATION:

1. USER AUTHENTICATION MODULE:

- THIS MODULE IS RESPONSIBLE FOR HANDLING USER REGISTRATION, LOGIN, AND SECURITY. IT ALLOWS USERS TO CREATE ACCOUNTS AND LOG IN WITH SECURE CREDENTIALS. THIS MODULE VALIDATES USER INFORMATION AGAINST THE DATA STORED IN THE MYSQL DATABASE.

2. WORKOUT LOGGING MODULE:

- THIS MODULE PROVIDES USERS WITH THE FUNCTIONALITY TO LOG THEIR DAILY WORKOUTS. USERS CAN INPUT DETAILS SUCH AS EXERCISES, SETS, REPS, DURATION, AND CALORIES BURNED. THE DATA IS THEN STORED IN THE DATABASE FOR FUTURE TRACKING AND ANALYSIS.

3. DATA VISUALIZATION MODULE:

- USING THE JFREECHART LIBRARY, THIS MODULE CREATES VISUAL REPRESENTATIONS OF WORKOUT DATA, SUCH AS PIE CHARTS THAT DISPLAY CALORIE

DISTRIBUTION ACROSS DIFFERENT EXERCISES. THIS HELPS USERS VISUALIZE THEIR FITNESS PROGRESS.

4. USER PROFILE AND SETTINGS MODULE:

- THIS MODULE ALLOWS USERS TO MANAGE THEIR PERSONAL INFORMATION, INCLUDING THEIR NAME, AGE, WEIGHT, AND HEIGHT. THESE DETAILS ARE USED TO CALCULATE IMPORTANT FITNESS METRICS, SUCH AS DAILY CALORIE INTAKE, AND CAN BE UPDATED AT ANY TIME.

5. WORKOUT HISTORY MODULE:

- USERS CAN VIEW A COMPREHENSIVE HISTORY OF ALL THEIR WORKOUTS IN THIS MODULE. IT PROVIDES A RECORD OF EXERCISES, SETS, REPS, DURATION, AND CALORIES BURNED FOR EACH WORKOUT SESSION, ALLOWING USERS TO ANALYZE THEIR FITNESS PROGRESS OVER TIME.

6. STATISTICS AND GOAL TRACKING MODULE:

- THIS MODULE CALCULATES KEY FITNESS STATISTICS SUCH AS TOTAL WORKOUTS COMPLETED, TOTAL CALORIES BURNED, AND PROGRESS TOWARD SET FITNESS GOALS. IT PROVIDES MOTIVATIONAL INSIGHTS AND STATISTICS THAT HELP USERS STAY ON TRACK WITH THEIR FITNESS PLANS.

PROGRAM CODE

CHAPTER 4:

MYSQL CODE:

```
CREATE DATABASE gym_tracker;
```

```
USE gym_tracker;
```

```
CREATE TABLE users (  
    id INT AUTO_INCREMENT PRIMARY KEY,  
    username VARCHAR(50) NOT NULL,  
    email VARCHAR(100),  
    password VARCHAR(50) NOT NULL  
);
```

```
CREATE TABLE workouts (  
    id INT AUTO_INCREMENT PRIMARY KEY,  
    user_id INT NOT NULL,  
    workout_date DATE NOT NULL,  
    exercise_name VARCHAR(255) NOT NULL,  
    sets INT NOT NULL,  
    reps INT NOT NULL,
```

```
weight FLOAT NOT NULL,  
  
notes TEXT,  
  
FOREIGN KEY (user_id) REFERENCES users(id)  
  
);
```

JAVA CODE

```
import javax.swing.*;  
  
import org.jfree.chart.*;  
  
import org.jfree.chart.plot.*;  
  
import org.jfree.data.general.DefaultPieDataset;  
  
import java.awt.*;  
  
import java.awt.event.*;  
  
import java.sql.*;  
  
  
public class Main extends JFrame {  
  
    private JTextField usernameField, nameField, emailField, ageField, heightField,  
weightField;  
  
    private JPasswordField passwordField;  
  
    private JButton loginButton, createAccountButton;  
  
    private UserDao userDao;
```

```
private ImageIcon dumbbellIcon = new ImageIcon("resources/dumbbell.png");
```

```
private ImageIcon flameIcon = new ImageIcon("resources/flame.png");
```

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

```
public Main() {  
    userDao = new UserDao();  
  
    // Main frame setup  
    setTitle("FitLife Dashboard");  
    setSize(400, 400);  
    setLayout(new GridBagLayout());  
    setDefaultCloseOperation(JFrame.EXIT_ON_CLOSE);  
    setLocationRelativeTo(null);  
    setResizable(false);  
    getContentPane().setBackground(new Color(30, 30, 30));  
  
    GridBagConstraints gbc = new GridBagConstraints();
```

```
gbc.insets = new Insets(10, 10, 10, 10);

// Create a background panel

BackgroundPanel backgroundPanel = new BackgroundPanel();
backgroundPanel.setLayout(new GridBagLayout());

// Modern title setup

JLabel titleLabel = new JLabel("FitLife Dashboard");
titleLabel.setFont(new Font("Arial", Font.BOLD, 26));
titleLabel.setForeground(Color.WHITE);

gbc.gridx = 0;

gbc.gridy = 0;

gbc.gridwidth = 2;

gbc.anchor = GridBagConstraints.CENTER;

backgroundPanel.add(titleLabel, gbc);

// Add dumbbell icon

JLabel iconLabel = new JLabel(dumbbellIcon);

gbc.gridy = 1;

backgroundPanel.add(iconLabel, gbc);
```

```
// Username field

gbc.gridwidth = 1;

gbc.gridx = 0;

gbc.gridy = 2;

gbc.anchor = GridBagConstraints.EAST;

backgroundPanel.add(new JLabel("Username:"), gbc);
```

```
gbc.gridx = 1;

gbc.anchor = GridBagConstraints.WEST;

usernameField = new JTextField(15);

backgroundPanel.add(usernameField, gbc);
```

```
// Password field

gbc.gridx = 0;

gbc.gridy = 3;

gbc.anchor = GridBagConstraints.EAST;

backgroundPanel.add(new JLabel("Password:"), gbc);
```

```
gbc.gridx = 1;

gbc.anchor = GridBagConstraints.WEST;

passwordField = new JPasswordField(15);
```

```
backgroundPanel.add(passwordField, gbc);
```

```
// Login button
```

```
gbc.gridx = 0;
```

```
gbc.gridy = 4;
```

```
gbc.gridwidth = 2;
```

```
loginButton = new JButton("Login");
```

```
styleButton(loginButton);
```

```
backgroundPanel.add(loginButton, gbc);
```

```
// Create account button
```

```
gbc.gridy = 5;
```

```
createAccountButton = new JButton("Create Account");
```

```
styleButton(createAccountButton);
```

```
backgroundPanel.add(createAccountButton, gbc);
```

```
// Add flame icon
```

```
JLabel flameLabel = new JLabel(flameIcon);
```

```
gbc.gridy = 6;
```

```
backgroundPanel.add(flameLabel, gbc);
```

```
// Add background panel to the frame

add(backgroundPanel);


// Button actions

loginButton.addActionListener(e -> authenticateUser());

createAccountButton.addActionListener(e -> new
CreateAccountFrame(this).setVisible(true));


setVisible(true);
}


private void authenticateUser() {

    String username = usernameField.getText();

    String password = new String(passwordField.getPassword());

    int userId = userDAO.authenticate(username, password);

    if (userId != -1) {

        JOptionPane.showMessageDialog(null, "Login successful!");

        new DashboardFrame(userId, username, password).setVisible(true);

        dispose();

    } else {

        JOptionPane.showMessageDialog(null, "Invalid credentials.");

    }

}
```

```
}  
}
```

```
private void styleButton(JButton button) {  
    button.setBackground(new Color(0, 150, 136));  
    button.setForeground(Color.WHITE);  
    button.setFocusPainted(false);  
    button.setCursor(Cursor.getPredefinedCursor(Cursor.HAND_CURSOR));  
    button.setPreferredSize(new Dimension(200, 40));  
    button.setBorder(BorderFactory.createEmptyBorder(5, 5, 5, 5));  
    button.addMouseListener(new MouseAdapter() {  
        @Override  
        public void mouseEntered(MouseEvent e) {  
            button.setBackground(new Color(0, 120, 100));  
        }  
  
        @Override  
        public void mouseExited(MouseEvent e) {  
            button.setBackground(new Color(0, 150, 136));  
        }  
    });  
};
```



```
}
```

```
class UserDAO {
```

```
    private static final String URL = "jdbc:mysql://localhost:3306/gym_tracker";
```

```
    private static final String USER = "root";
```

```
    private static final String PASSWORD = "gym";
```

```
    public int authenticate(String username, String password) {
```

```
        try (Connection conn = DriverManager.getConnection(URL, USER,
PASSWORD)) {
```

```
            String query = "SELECT id FROM users WHERE username = ? AND
password = ?";
```

```
            PreparedStatement pstmt = conn.prepareStatement(query);
```

```
            pstmt.setString(1, username);
```

```
            pstmt.setString(2, password);
```

```
            ResultSet rs = pstmt.executeQuery();
```

```
            if (rs.next()) {
```

```
                return rs.getInt("id");
```

```
            }
```

```
        } catch (SQLException e) {
```

```
            e.printStackTrace();
```

```
        }
```

```
    return -1;
}
```

```
public int createAccount(String username, String password, String name,
String email, int age, int height, int weight) {
```

```
    int generatedId = -1;
```

```
    try (Connection conn = DriverManager.getConnection(URL, USER,
PASSWORD)) {
```

```
        String query = "INSERT INTO users (username, password, name, email,
age, height, weight) VALUES (?, ?, ?, ?, ?, ?, ?)";
```

```
        PreparedStatement pstmt = conn.prepareStatement(query,
Statement.RETURN_GENERATED_KEYS);
```

```
        pstmt.setString(1, username);
```

```
        pstmt.setString(2, password);
```

```
        pstmt.setString(3, name);
```

```
        pstmt.setString(4, email);
```

```
        pstmt.setInt(5, age);
```

```
        pstmt.setInt(6, height);
```

```
        pstmt.setInt(7, weight);
```

```
        int rowsInserted = pstmt.executeUpdate();
```

```
        if (rowsInserted > 0) {
```

```
            ResultSet rs = pstmt.getGeneratedKeys();
```

```
        if (rs.next()) {  
            generatedId = rs.getInt(1);  
        }  
    }  
    } catch (SQLException e) {  
        e.printStackTrace();  
    }  
    return generatedId;  
}
```

```
public void saveWorkout(int userId, String exercise, int sets, int reps, int  
duration, int calories) {
```

```
    try (Connection conn = DriverManager.getConnection(URL, USER,  
PASSWORD)) {
```

```
        String query = "INSERT INTO workouts (user_id, exercise, sets, reps,  
duration, calories_burned) VALUES (?, ?, ?, ?, ?, ?)";
```

```
        PreparedStatement pstmt = conn.prepareStatement(query);
```

```
        pstmt.setInt(1, userId);
```

```
        pstmt.setString(2, exercise);
```

```
        pstmt.setInt(3, sets);
```

```
        pstmt.setInt(4, reps);
```

```
        pstmt.setInt(5, duration);
```

```
        pstmt.setInt(6, calories);
```

```

        pstmt.executeUpdate();
    } catch (SQLException e) {
        e.printStackTrace();
    }
}

```

```

public String getWorkouts(int userId) {
    StringBuilder workouts = new StringBuilder();

    try (Connection conn = DriverManager.getConnection(URL, USER,
PASSWORD)) {
        String query = "SELECT * FROM workouts WHERE user_id = ?";
        PreparedStatement pstmt = conn.prepareStatement(query);
        pstmt.setInt(1, userId);
        ResultSet rs = pstmt.executeQuery();
        while (rs.next()) {
            workouts.append("Workout ID: ").append(rs.getInt("id"))
                .append(", Exercise: ").append(rs.getString("exercise"))
                .append(", Sets: ").append(rs.getInt("sets"))
                .append(", Reps: ").append(rs.getInt("reps"))
                .append(", Duration: ").append(rs.getInt("duration")).append("
mins")
                .append(", Calories: ").append(rs.getInt("calories_burned"))

```

```

        .append("\n");
    }
} catch (SQLException e) {
    e.printStackTrace();
}
return workouts.toString();
}

```

```

public int getTotalWorkouts(int userId) {
    int total = 0;

    try (Connection conn = DriverManager.getConnection(URL, USER,
PASSWORD)) {
        String query = "SELECT COUNT(*) FROM workouts WHERE user_id
= ?";

        PreparedStatement pstmt = conn.prepareStatement(query);

        pstmt.setInt(1, userId);

        ResultSet rs = pstmt.executeQuery();

        if (rs.next()) {
            total = rs.getInt(1);
        }
    } catch (SQLException e) {
        e.printStackTrace();
    }
}

```

```
    }  
  
    return total;  
  
}
```

```
public int getTotalCaloriesBurned(int userId) {  
  
    int total = 0;  
  
    try (Connection conn = DriverManager.getConnection(URL, USER,  
PASSWORD)) {  
  
        String query = "SELECT SUM(calories_burned) FROM workouts  
WHERE user_id = ?";  
  
        PreparedStatement pstmt = conn.prepareStatement(query);  
  
        pstmt.setInt(1, userId);  
  
        ResultSet rs = pstmt.executeQuery();  
  
        if (rs.next()) {  
  
            total = rs.getInt(1);  
  
        }  
  
    } catch (SQLException e) {  
  
        e.printStackTrace();  
  
    }  
  
    return total;  
  
}
```

```

public DefaultPieDataset getWorkoutTypeDistribution(int userId) {

    DefaultPieDataset dataset = new DefaultPieDataset();

    try (Connection conn = DriverManager.getConnection(URL, USER,
PASSWORD)) {

        String query = "SELECT exercise, SUM(calories_burned) as
total_calories FROM workouts WHERE user_id = ? GROUP BY exercise";

        PreparedStatement pstmt = conn.prepareStatement(query);

        pstmt.setInt(1, userId);

        ResultSet rs = pstmt.executeQuery();

        while (rs.next()) {

            dataset.setValue(rs.getString("exercise"),
rs.getDouble("total_calories"));

        }

    } catch (SQLException e) {

        e.printStackTrace();

    }

    return dataset;

}

```

```

public User getUserDetails(int userId) {

    User user = null;

    try (Connection conn = DriverManager.getConnection(URL, USER,
PASSWORD)) {

```

```
String query = "SELECT age, height, weight, gender FROM users  
WHERE id = ?";
```

```
PreparedStatement pstmt = conn.prepareStatement(query);  
  
pstmt.setInt(1, userId);  
  
ResultSet rs = pstmt.executeQuery();  
  
if (rs.next()) {  
  
    int age = rs.getInt("age");  
  
    int height = rs.getInt("height");  
  
    int weight = rs.getInt("weight");  
  
    String gender = rs.getString("gender");  
  
    user = new User(age, height, weight, gender);  
  
    }  
  
    } catch (SQLException e) {  
  
        e.printStackTrace();  
  
    }  
  
    return user;  
  
    }  
  
}
```

```
class User {  
  
    private int age;  
  
    private int height; // in cm
```



```
private int weight; // in kg
```

```
private String gender;
```

```
public User(int age, int height, int weight, String gender) {
```

```
    this.age = age;
```

```
    this.height = height;
```

```
    this.weight = weight;
```

```
    this.gender = gender;
```

```
}
```

```
public int calculateDailyCalories() {
```

```
    double bmr;
```

```
    if ("M".equalsIgnoreCase(gender)) {
```

```
        bmr = 10 * weight + 6.25 * height - 5 * age + 5;
```

```
    } else {
```

```
        bmr = 10 * weight + 6.25 * height - 5 * age - 161;
```

```
    }
```

```
    return (int)(bmr * 1.55); // Adjust multiplier as necessary for activity level
```

```
}
```

```
}
```

```

class BackgroundPanel extends JPanel {

    private Image backgroundImage;

    public BackgroundPanel() {

        try {

            backgroundImage =
Toolkit.getDefaultToolkit().getImage("D:\\gym_background.jpg"); // Update the
path

        } catch (Exception e) {

            e.printStackTrace();

        }

    }

    @Override

    protected void paintComponent(Graphics g) {

        super.paintComponent(g);

        g.drawImage(backgroundImage, 0, 0, getWidth(), getHeight(), this);

    }

}

```

```

class CreateAccountFrame extends JFrame {

    private Main mainFrame;

```

```
public CreateAccountFrame(Main mainFrame) {  
    this.mainFrame = mainFrame;  
  
    setTitle("Create Account");  
    setSize(400, 400);  
    setLayout(new GridBagLayout());  
    setDefaultCloseOperation(JFrame.DISPOSE_ON_CLOSE);  
    setLocationRelativeTo(null);  
    setResizable(false);  
    getContentPane().setBackground(new Color(55, 55, 55));  
  
    GridBagConstraints gbc = new GridBagConstraints();  
    gbc.insets = new Insets(10, 10, 10, 10);  
    gbc.fill = GridBagConstraints.HORIZONTAL;  
  
    int row = 0;  
  
    addField(gbc, row++, "Username:", usernameField = new JTextField(15));  
    addField(gbc, row++, "Password:", passwordField = new  
JPasswordField(15));  
    addField(gbc, row++, "Name:", nameField = new JTextField(15));
```

```
addField(gbc, row++, "Email:", emailField = new JTextField(15));  
addField(gbc, row++, "Age:", ageField = new JTextField(15));  
addField(gbc, row++, "Height (cm):", heightField = new JTextField(15));  
addField(gbc, row++, "Weight (kg):", weightField = new JTextField(15));
```

```
JButton createAccountButton = new JButton("Create Account");
```

```
styleButton(createAccountButton);
```

```
gbc.gridx = 0;
```

```
gbc.gridy = row;
```

```
gbc.gridwidth = 2;
```

```
add(createAccountButton, gbc);
```

```
createAccountButton.addActionListener(e -> createUser());
```

```
setVisible(true);
```

```
}
```

```
private void addField(GridBagConstraints gbc, int row, String label,  
JTextField field) {
```

```
    gbc.gridx = 0;
```

```
    gbc.gridy = row;
```

```
    gbc.anchor = GridBagConstraints.EAST;
```

```
JLabel jLabel = new JLabel(label);  
  
jLabel.setForeground(Color.WHITE);  
  
add(jLabel, gbc);  
  
gbc.gridx = 1;  
  
gbc.anchor = GridBagConstraints.WEST;  
  
field.setBackground(new Color(100, 100, 100));  
  
field.setForeground(Color.WHITE);  
  
field.setBorder(BorderFactory.createLineBorder(Color.WHITE));  
  
add(field, gbc);  
  
}
```

```
private void createUser() {  
  
    String username = usernameField.getText();  
  
    String password = new String(passwordField.getPassword());  
  
    String name = nameField.getText();  
  
    String email = emailField.getText();  
  
    try {  
  
        int age = Integer.parseInt(ageField.getText());  
  
        int height = Integer.parseInt(heightField.getText());  
  
        int weight = Integer.parseInt(weightField.getText());
```

```
        int userId = userDAO.createAccount(username, password, name, email,
age, height, weight);

        if (userId != -1) {

            JOptionPane.showMessageDialog(null, "Account created successfully!
User ID: " + userId);

            new DashboardFrame(userId, username, password).setVisible(true);

            dispose();

        } else {

            JOptionPane.showMessageDialog(null, "Account creation failed.");

        }

    } catch (NumberFormatException e) {

        JOptionPane.showMessageDialog(this, "Please enter valid numbers for
age, height, and weight.", "Input Error", JOptionPane.ERROR_MESSAGE);

    }

}

}
```

```
class DashboardFrame extends JFrame {

    private JPanel sidebarPanel;

    private JButton toggleSidebarButton;

    private boolean isSidebarVisible = false;
```

```
private JButton trackWorkoutButton, viewWorkoutsButton, accountButton,  
bmrButton, viewDistributionButton;
```

```
private JTextArea statsArea;
```

```
private UserDao userDao;
```

```
private int userId;
```

```
private String username;
```

```
private String password;
```

```
private JPanel animationPanel;
```

```
private JLabel caloriesBurnedLabel;
```

```
private JProgressBar caloriesProgressBar;
```

```
private Timer progressBarTimer;
```

```
private int calorieGoal = 700;
```

```
private String[] motivationalMessages = {
```

```
    "Keep pushing!",
```

```
    "You're doing great!",
```

```
    "Stay active!"
```

```
};
```

```
private JLabel motivationalMessageLabel;
```

```
private JLabel dailyCalorieIntakeLabel;
```

```
public DashboardFrame(int userId, String username, String password) {
```

```
this.userId = userId;

this.username = username;

this.password = password;

userDAO = new UserDAO();

setTitle("Dashboard");

setSize(800, 600);

setLayout(new BorderLayout());

setDefaultCloseOperation(JFrame.DO_NOTHING_ON_CLOSE);

setLocationRelativeTo(null);

setResizable(false);

getContentPane().setBackground(new Color(30, 30, 30));


addWindowListener(new WindowAdapter() {

    @Override

    public void windowClosing(WindowEvent e) {

        int confirm = JOptionPane.showConfirmDialog(DashboardFrame.this,

            "Are you sure you want to exit?", "Exit Confirmation",

            JOptionPane.YES_NO_OPTION);

        if (confirm == JOptionPane.YES_OPTION) {

            System.exit(0);

        }

    }

});
```



```
}  
});
```

```
sidebarPanel = new JPanel();  
  
sidebarPanel.setLayout(new BoxLayout(sidebarPanel,  
BoxLayout.Y_AXIS));  
  
sidebarPanel.setBackground(new Color(70, 70, 70));  
  
sidebarPanel.setPreferredSize(new Dimension(200, 500));  
  
  
trackWorkoutButton = new JButton("Track Workout");  
viewWorkoutsButton = new JButton("View Workouts");  
accountButton = new JButton("Account Details");  
bmrButton = new JButton("Calculate BMR");  
viewDistributionButton = new JButton("View Workout Distribution");  
  
  
styleButton(trackWorkoutButton);  
styleButton(viewWorkoutsButton);  
styleButton(accountButton);  
styleButton(bmrButton);  
styleButton(viewDistributionButton);  
  
sidebarPanel.add(trackWorkoutButton);
```

```
sidebarPanel.add(Box.createRigidArea(new Dimension(0, 10)));  
sidebarPanel.add(viewWorkoutsButton);  
sidebarPanel.add(Box.createRigidArea(new Dimension(0, 10)));  
sidebarPanel.add(accountButton);  
sidebarPanel.add(Box.createRigidArea(new Dimension(0, 10)));  
sidebarPanel.add(bmrButton);  
sidebarPanel.add(Box.createRigidArea(new Dimension(0, 10)));  
sidebarPanel.add(viewDistributionButton);  
sidebarPanel.add(Box.createVerticalGlue());
```

```
toggleSidebarButton = new JButton("≡");  
toggleSidebarButton.setPreferredSize(new Dimension(50, 50));  
toggleSidebarButton.setBackground(new Color(0, 150, 136));  
toggleSidebarButton.setForeground(Color.WHITE);  
toggleSidebarButton.setFocusPainted(false);
```

```
toggleSidebarButton.setCursor(Cursor.getPredefinedCursor(Cursor.HAND_CURSOR));
```

```
toggleSidebarButton.addActionListener(e -> toggleSidebar());
```

```
animationPanel = new JPanel();
```

```
animationPanel.setLayout(new BoxLayout(animationPanel,  
BoxLayout.Y_AXIS));
```

```
animationPanel.setBackground(new Color(30, 30, 30));  
animationPanel.setBorder(BorderFactory.createEmptyBorder(20, 20, 20,  
20));
```

```
JLabel statsTitle = new JLabel("Track Your Progress");  
statsTitle.setFont(new Font("Arial", Font.BOLD, 24));  
statsTitle.setForeground(Color.WHITE);  
statsTitle.setAlignmentX(Component.CENTER_ALIGNMENT);  
animationPanel.add(statsTitle);
```

```
JPanel caloriesCard = createStatsCard("Calories Burned", flameIcon, null);  
caloriesBurnedLabel = new JLabel();  
caloriesCard.add(caloriesBurnedLabel);  
caloriesProgressBar = new JProgressBar(0, calorieGoal);  
caloriesProgressBar.setStringPainted(true);  
caloriesCard.add(caloriesProgressBar);  
animationPanel.add(caloriesCard);
```

```
motivationalMessageLabel = new JLabel();  
motivationalMessageLabel.setFont(new Font("Arial", Font.ITALIC, 16));  
motivationalMessageLabel.setForeground(Color.WHITE);
```

```
motivationalMessageLabel.setAlignmentX(Component.CENTER_ALIGNMENT);

animationPanel.add(motivationalMessageLabel);

rotateMotivationalMessage();


// Daily Calorie Intake Display

dailyCalorieIntakeLabel = new JLabel();

dailyCalorieIntakeLabel.setFont(new Font("Arial", Font.BOLD, 18));

dailyCalorieIntakeLabel.setForeground(Color.WHITE);


dailyCalorieIntakeLabel.setAlignmentX(Component.CENTER_ALIGNMENT);

animationPanel.add(dailyCalorieIntakeLabel);


JPanel topPanel = new JPanel(new FlowLayout(FlowLayout.LEFT));

topPanel.setBackground(new Color(30, 30, 30));

topPanel.add(toggleSidebarButton);


add(topPanel, BorderLayout.NORTH);

add(sidebarPanel, BorderLayout.WEST);

add(animationPanel, BorderLayout.CENTER);


loadUserStats();
```

```
loadDailyCalorieIntake();
```

```
trackWorkoutButton.addActionListener(e -> {  
    WorkoutTrackerFrame workoutFrame = new  
    WorkoutTrackerFrame(userId);  
    workoutFrame.addWorkoutListener(new WorkoutListener() {  
        @Override  
        public void workoutAdded() {  
            loadUserStats();  
        }  
    });  
    workoutFrame.setVisible(true);  
});
```

```
viewWorkoutsButton.addActionListener(e -> viewWorkouts());  
accountButton.addActionListener(e -> showAccountDetails());  
bmrButton.addActionListener(e -> calculateBMR());  
viewDistributionButton.addActionListener(e ->  
showWorkoutTypeDistribution());
```

```
setVisible(true);  
}
```

```
private JPanel createStatsCard(String title, ImageIcon icon, String prefix) {  
  
    JPanel card = new JPanel();  
  
    card.setBackground(new Color(50, 50, 50));  
  
    card.setBorder(BorderFactory.createEmptyBorder(10, 10, 10, 10));  
  
    card.setLayout(new BoxLayout(card, BoxLayout.Y_AXIS));  
  
    card.setAlignmentX(Component.CENTER_ALIGNMENT);  
  
    card.setBorder(BorderFactory.createLineBorder(Color.WHITE, 1));  
  
    card.setPreferredSize(new Dimension(250, 100));  
  
    card.setMaximumSize(new Dimension(250, 100));  
  
  
    JLabel titleLabel = new JLabel(title);  
  
    titleLabel.setFont(new Font("Arial", Font.BOLD, 18));  
  
    titleLabel.setForeground(Color.WHITE);  
  
    card.add(titleLabel);  
  
  
    if (icon != null) {  
  
        JLabel iconLabel = new JLabel(icon);  
  
        card.add(iconLabel);  
  
    }  
}
```

```
if (prefix != null) {  
    JLabel prefixLabel = new JLabel(prefix);  
    prefixLabel.setFont(new Font("Arial", Font.PLAIN, 14));  
    prefixLabel.setForeground(Color.LIGHT_GRAY);  
    card.add(prefixLabel);  
}  
  
return card;  
}  
  
private void loadUserStats() {  
    int totalCaloriesBurned = userDao.getTotalCaloriesBurned(userId);  
    caloriesBurnedLabel.setText(String.valueOf(totalCaloriesBurned));  
  
    caloriesProgressBar.setValue(totalCaloriesBurned);  
    caloriesProgressBar.setString(totalCaloriesBurned + " / " + calorieGoal + "  
calories");  
    caloriesProgressBar.setForeground(totalCaloriesBurned >= calorieGoal ?  
Color.GREEN : Color.RED);  
  
    caloriesBurnedLabel.setForeground(totalCaloriesBurned > calorieGoal ?  
Color.GREEN : Color.RED);  
}
```

```
private void loadDailyCalorieIntake() {  
    User user = userDao.getUserDetails(userId);  
    if (user != null) {  
        int dailyCalories = user.calculateDailyCalories();  
        dailyCalorieIntakeLabel.setText("Daily Caloric Intake: " + dailyCalories  
+ " calories");  
    }  
}
```

```
private void rotateMotivationalMessage() {  
    Timer timer = new Timer(3000, e -> {  
        int index = (int) (Math.random() * motivationalMessages.length);  
        motivationalMessageLabel.setText(motivationalMessages[index]);  
    });  
    timer.start();  
}
```

```
private void toggleSidebar() {  
    isSidebarVisible = !isSidebarVisible;  
    sidebarPanel.setVisible(isSidebarVisible);  
    revalidate();  
}
```



```
repaint();  
}
```

```
private void viewWorkouts() {  
    String workouts = userDAO.getWorkouts(userId);  
    if (workouts.isEmpty()) {  
        JOptionPane.showMessageDialog(this, "No workouts found for this  
user.", "Workout Info", JOptionPane.INFORMATION_MESSAGE);  
        return;  
    }  
}
```

```
JTextArea workoutsArea = new JTextArea(workouts);  
workoutsArea.setEditable(false);  
workoutsArea.setMargin(new Insets(10, 10, 10, 10));  
workoutsArea.setBackground(new Color(100, 100, 100));  
workoutsArea.setForeground(Color.WHITE);  
workoutsArea.setBorder(BorderFactory.createEmptyBorder(10, 10, 10,  
10));
```

```
JScrollPane scrollPane = new JScrollPane(workoutsArea);  
scrollPane.setPreferredSize(new Dimension(600, 300));  
JOptionPane.showMessageDialog(this, scrollPane, "Saved Workouts",  
JOptionPane.INFORMATION_MESSAGE);
```

```
}
```

```
private void showAccountDetails() {
```

```
    String accountDetails = "Username: " + username + "\nPassword: " +  
password;
```

```
    JOptionPane.showMessageDialog(this, accountDetails, "Account Details",  
JOptionPane.INFORMATION_MESSAGE);
```

```
}
```

```
private void calculateBMR() {
```

```
    String gender = JOptionPane.showInputDialog(this, "Enter your gender  
(M/F):");
```

```
    if (gender == null) {
```

```
        return;
```

```
    }
```

```
    gender = gender.toUpperCase();
```

```
    String weightInput = JOptionPane.showInputDialog(this, "Enter your  
weight (kg):");
```

```
    String heightInput = JOptionPane.showInputDialog(this, "Enter your height  
(cm):");
```

```
    String ageInput = JOptionPane.showInputDialog(this, "Enter your age  
(years):");
```

```

    if (weightInput == null || heightInput == null || ageInput == null) {
        return;
    }

    try {
        int weight = Integer.parseInt(weightInput);
        int height = Integer.parseInt(heightInput);
        int age = Integer.parseInt(ageInput);

        double bmr;

        if ("M".equals(gender)) {
            bmr = 88.362 + (13.397 * weight) + (4.799 * height) - (5.677 * age);
        } else if ("F".equals(gender)) {
            bmr = 447.593 + (9.247 * weight) + (3.098 * height) - (4.330 * age);
        } else {
            JOptionPane.showMessageDialog(this, "Invalid gender input. Please
enter 'M' or 'F'.");
            return;
        }

        JOptionPane.showMessageDialog(this, "Your BMR is: " + bmr + "
calories/day");
    }

```

```
    } catch (NumberFormatException e) {  
        JOptionPane.showMessageDialog(this, "Please enter valid numbers for  
weight, height, and age.", "Input Error", JOptionPane.ERROR_MESSAGE);  
    }  
}
```

```
private void showWorkoutTypeDistribution() {  
    DefaultPieDataset dataset = userDAO.getWorkoutTypeDistribution(userId);  
    JFreeChart pieChart = ChartFactory.createPieChart(  
        "Workout Type Distribution", dataset, true, true, false);  
    ChartPanel chartPanel = new ChartPanel(pieChart);  
    chartPanel.setPreferredSize(new Dimension(500, 400));  
    JOptionPane.showMessageDialog(this, chartPanel, "Workout Type  
Distribution", JOptionPane.PLAIN_MESSAGE);  
}
```

```
private void styleButton(JButton button) {  
    button.setBackground(new Color(0, 150, 136));  
    button.setForeground(Color.WHITE);  
    button.setFocusPainted(false);  
    button.setCursor(Cursor.getPredefinedCursor(Cursor.HAND_CURSOR));  
    button.setPreferredSize(new Dimension(200, 40));  
}
```

```
button.setBorder(BorderFactory.createEmptyBorder(5, 5, 5, 5));

button.addMouseListener(new MouseAdapter() {

    @Override

    public void mouseEntered(MouseEvent e) {

        button.setBackground(new Color(0, 120, 100));

    }

    @Override

    public void mouseExited(MouseEvent e) {

        button.setBackground(new Color(0, 150, 136));

    }

});

}
```

```
class WorkoutTrackerFrame extends JFrame {

    private JTextField exerciseField, setsField, repsField, durationField,
caloriesField;

    private int userId;

    private WorkoutListener listener;

    public WorkoutTrackerFrame(int userId) {
```

```
this.userId = userId;
```

```
setTitle("Track Workout");
```

```
setSize(400, 400);
```

```
setLayout(new GridBagLayout());
```

```
setDefaultCloseOperation(JFrame.DISPOSE_ON_CLOSE);
```

```
setLocationRelativeTo(null);
```

```
setResizable(false);
```

```
getContentPane().setBackground(new Color(55, 55, 55));
```

```
GridBagConstraints gbc = new GridBagConstraints();
```

```
gbc.insets = new Insets(10, 10, 10, 10);
```

```
int row = 0;
```

```
exerciseField = new JTextField(15);
```

```
setsField = new JTextField(15);
```

```
repsField = new JTextField(15);
```

```
durationField = new JTextField(15);
```

```
caloriesField = new JTextField(15);
```

```
addField(gbc, row++, "Exercise:", exerciseField);  
addField(gbc, row++, "Sets:", setsField);  
addField(gbc, row++, "Reps:", repsField);  
addField(gbc, row++, "Duration (mins):", durationField);  
addField(gbc, row++, "Calories burned:", caloriesField);
```

```
JButton saveButton = new JButton("Save Workout");  
styleButton(saveButton);  
gbc.gridx = 0;  
gbc.gridy = row;  
gbc.gridwidth = 2;  
add(saveButton, gbc);
```

```
saveButton.addActionListener(e -> {  
    saveWorkout();  
    if (listener != null) {  
        listener.workoutAdded();  
    }  
});
```

```
setVisible(true);
```

```
}
```

```
public void addWorkoutListener(WorkoutListener listener) {  
    this.listener = listener;  
}
```

```
private void addField(GridBagConstraints gbc, int row, String label,  
JTextField field) {  
    gbc.gridx = 0;  
    gbc.gridy = row;  
    gbc.anchor = GridBagConstraints.EAST;  
    JLabel jLabel = new JLabel(label);  
    jLabel.setForeground(Color.WHITE);  
    add(jLabel, gbc);  
    gbc.gridx = 1;  
    gbc.anchor = GridBagConstraints.WEST;  
    field.setBackground(new Color(100, 100, 100));  
    field.setForeground(Color.WHITE);  
    field.setBorder(BorderFactory.createLineBorder(Color.WHITE));  
    add(field, gbc);  
}
```



```
private void saveWorkout() {  
    String exercise = exerciseField.getText();  
    int sets = Integer.parseInt(setsField.getText());  
    int reps = Integer.parseInt(repsField.getText());  
    int duration = Integer.parseInt(durationField.getText());  
    int calories = Integer.parseInt(caloriesField.getText());  
  
    UserDAO userDAO = new UserDAO();  
    userDAO.saveWorkout(userId, exercise, sets, reps, duration, calories);  
  
    JOptionPane.showMessageDialog(this, "Workout saved successfully!");  
    clearFields();  
}
```

```
private void clearFields() {  
    exerciseField.setText("");  
    setsField.setText("");  
    repsField.setText("");  
    durationField.setText("");  
    caloriesField.setText("");  
}
```

```
}
```

```
interface WorkoutListener {
```

```
    void workoutAdded();
```

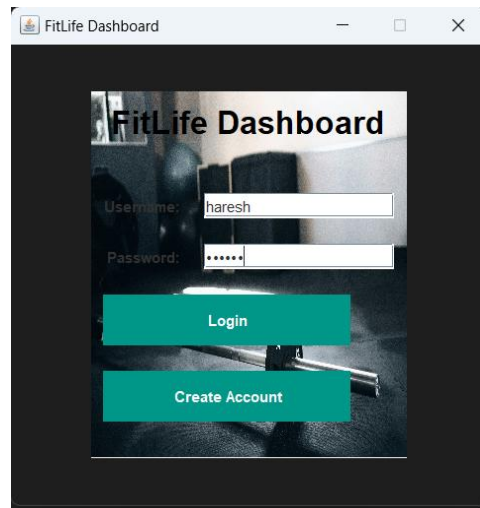
```
}
```

```
}
```

RESULTS & DISCUSSIONS

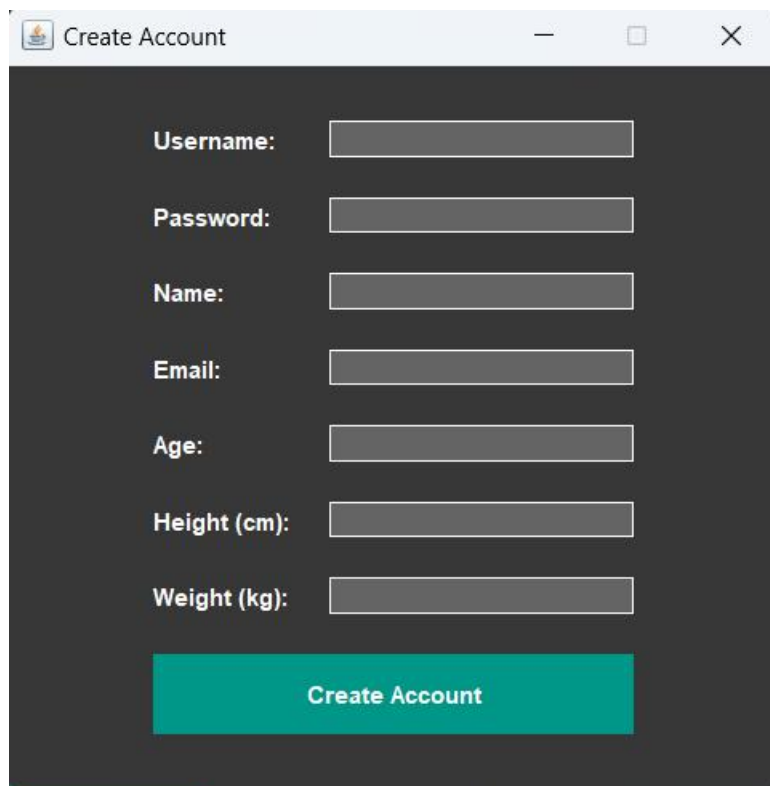
CHAPTER 5

LOGIN PAGE



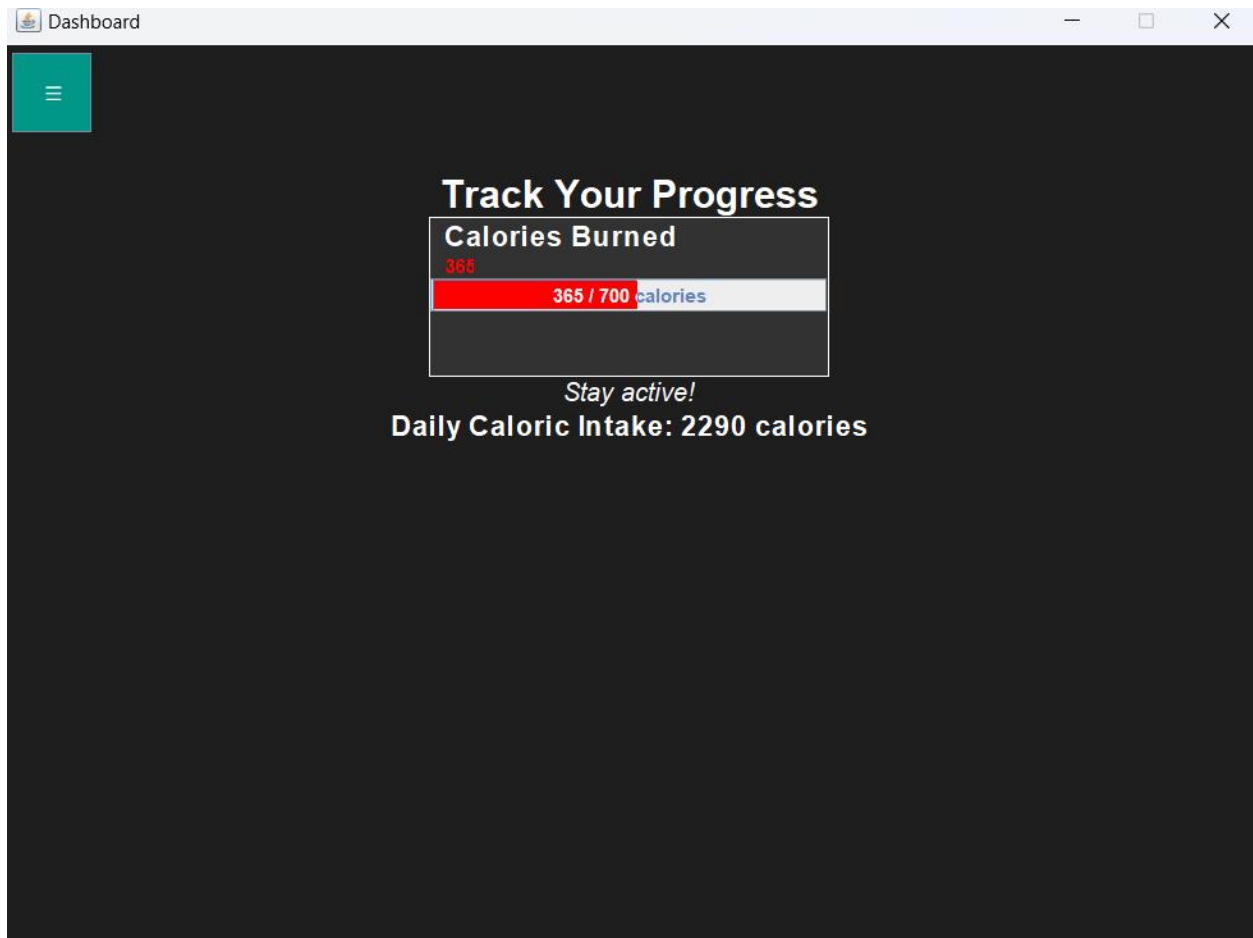
A screenshot of a web browser window titled "FitLife Dashboard". The page has a dark background with a blurred image of gym equipment. The title "FitLife Dashboard" is at the top. Below it, there are two input fields: "Username:" with the text "haresh" and "Password:" with masked characters "*****". Below the password field are two teal buttons: "Login" and "Create Account".

CREATE ACCOUNT PAGE



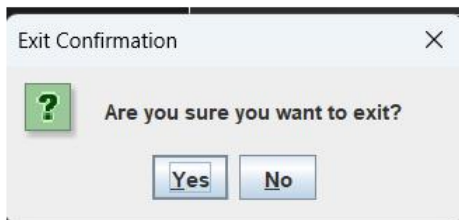
A screenshot of a web browser window titled "Create Account". The page has a dark background. It contains several input fields for user registration: "Username:", "Password:", "Name:", "Email:", "Age:", "Height (cm):", and "Weight (kg):". Each label is followed by a white input box. At the bottom, there is a large teal button labeled "Create Account".

MAIN PAGE

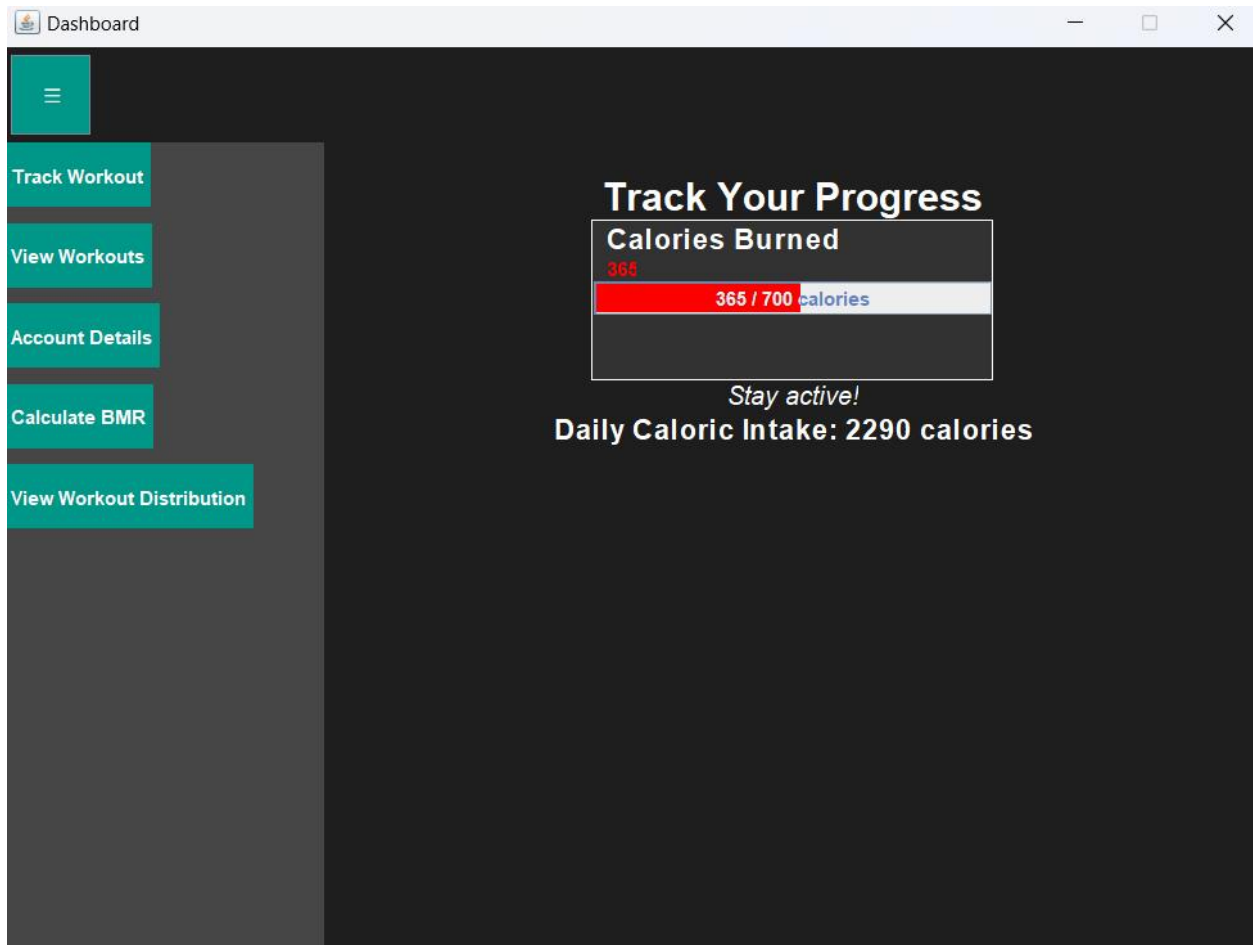


EXIT CONFIRMATION

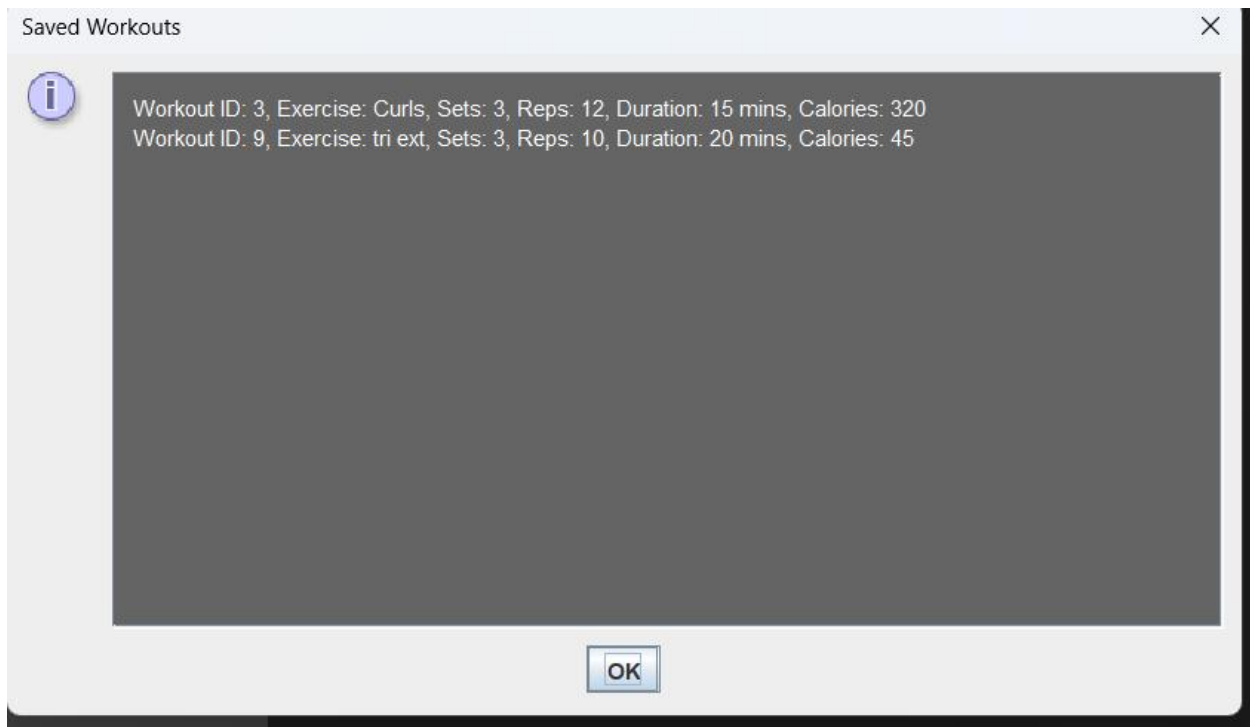
.



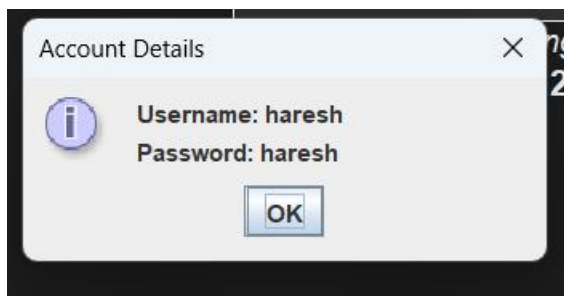
SIDEBAR FACILITY



SAVED WORKOUTS (USERS CAN SEE HISTORY)



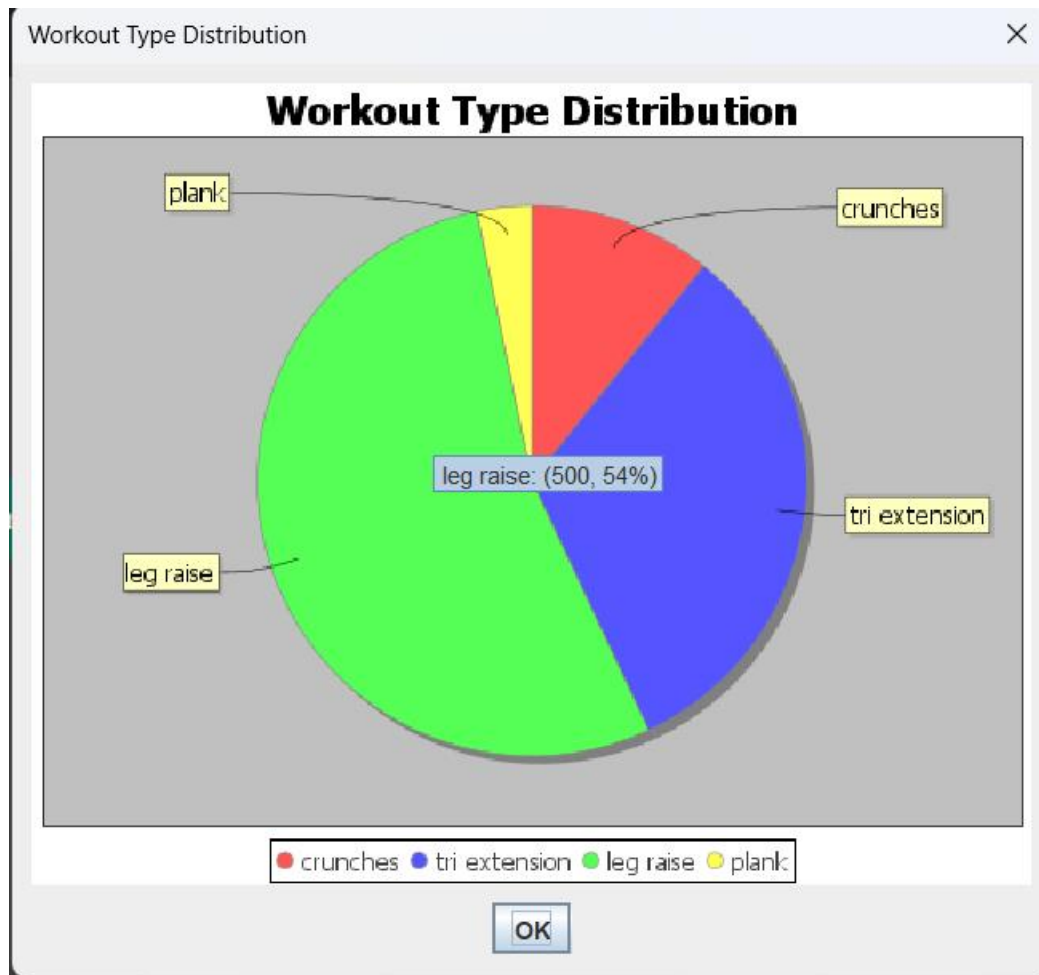
ACCOUNT DETAILS



BMR - (BODY METABOLIC RATE)



WORKOUT DISTRIBUTION (PIE CHART)



DATABASE FOR USER ACCOUNTS (USERS TABLE)

[illegible]

DATABASE FOR WORKOUTS(TABLE)

[illegible]

CONCLUSION & FUTURE ENHANCEMENTS

CHAPTER 6

CONCLUSION

THE DEVELOPMENT OF THE FITLIFE DASHBOARD PROVIDES A COMPREHENSIVE SOLUTION FOR USERS LOOKING TO TRACK AND IMPROVE THEIR FITNESS ROUTINES. BY COMBINING JAVA SWING FOR THE USER INTERFACE, MYSQL FOR DATA STORAGE, AND JFREECHART FOR DATA VISUALIZATION, THE APPLICATION OFFERS A SEAMLESS AND USER-FRIENDLY EXPERIENCE.

USERS CAN EFFICIENTLY LOG THEIR WORKOUTS, MONITOR PROGRESS, AND GAIN INSIGHTS INTO THEIR FITNESS THROUGH DETAILED STATISTICS AND CHARTS. THE SYSTEM'S USER AUTHENTICATION ENSURES THAT PERSONAL DATA IS SECURE, WHILE ITS MODULAR STRUCTURE MAKES IT EASY TO MAINTAIN AND EXPAND. FITLIFE DASHBOARD HAS SUCCESSFULLY MET ITS OBJECTIVE OF PROVIDING A PRACTICAL TOOL FOR FITNESS TRACKING, MAKING IT EASIER FOR USERS TO STAY MOTIVATED AND ACHIEVE THEIR HEALTH GOALS.

FUTURE ENHANCEMENTS

WHILE THE FITLIFE DASHBOARD OFFERS A STRONG FOUNDATION, THERE ARE SEVERAL POTENTIAL ENHANCEMENTS THAT COULD BE IMPLEMENTED IN FUTURE VERSIONS TO FURTHER IMPROVE THE USER EXPERIENCE:

- 1. MOBILE APP SUPPORT:**

- DEVELOPING A MOBILE VERSION OF THE FITLIFE DASHBOARD WOULD MAKE IT EVEN MORE ACCESSIBLE TO USERS, ALLOWING THEM TO LOG WORKOUTS AND TRACK PROGRESS ON-THE-GO USING THEIR SMARTPHONES OR TABLETS.

- 2. REAL-TIME WORKOUT TRACKING:**

- INTEGRATING WEARABLE TECHNOLOGY OR FITNESS TRACKERS COULD ALLOW USERS TO AUTOMATICALLY LOG THEIR WORKOUTS AND CALORIE BURN IN REAL TIME, MAKING THE APP EVEN MORE DYNAMIC.

3. SOCIAL SHARING FEATURES:

- USERS COULD BE GIVEN THE OPTION TO SHARE THEIR WORKOUT PROGRESS, ACHIEVEMENTS, OR CHALLENGES ON SOCIAL MEDIA PLATFORMS, CREATING A COMMUNITY AROUND FITNESS AND FOSTERING MOTIVATION THROUGH SOCIAL INTERACTION.

4. PERSONALIZED FITNESS PLANS:

- A FUTURE ENHANCEMENT COULD INCLUDE AI-BASED SUGGESTIONS FOR WORKOUT ROUTINES BASED ON THE USER'S FITNESS HISTORY, GOALS, AND PREFERENCES, PROVIDING A MORE CUSTOMIZED EXPERIENCE.

REFERENCES

CHAPTER 7

REFERENCE

Here is the **References** section with just the links:

1. <https://www.youtube.com/c/Codecademy>
2. <https://www.youtube.com/c/ProgrammingKnowledge>
3. <https://www.youtube.com/c/BroCodez>
4. <https://docs.oracle.com/javase/8/docs/>
5. <https://dev.mysql.com/doc/>
6. <https://www.jfree.org/jfreechart/>
7. <https://stackoverflow.com>
8. <https://mvnrepository.com/>
9. <https://docs.oracle.com/javase/8/docs/technotes/guides/jdbc/>

THANK YOU