FITNESS TRACKING APPLICATION

A MINI PROJECT REPORT

Submitted by

FAREED AHAMED KM **230701087**

FARHEEN TABASSUM H **230701088**

In partial fulfillment for the award of the degree of

BACHELOR OF ENGINEERING

IN

COMPUTER SCIENCE ENGINEERING

RAJALAKSHMI ENGINEERING COLLEGE (AUTONOMOUS)

THANDALAM

CHENNAI-602105

2024-25

BONAFIDE CERTIFICATE

Certified that this project report "**FITNESS TRACKING APPLICATION**" is the bonafide work of **"FAREED AHAMED KM(230701087), FARHEEN TABASSUM H(230701088)"** who carried out the project work under my supervision.

Submitted for the Practical Examination held on ________________________

SIGNATURE	**SIGNATURE**
Mrs,Divya.M	**Mr.Ragu**
Assistant Professor,	**Assistant Professor,**
Computer Science and Engineering,	**Computer Science and Engineering,**
Rajalakshmi Engineering College	**Rajalakshmi Engineering College**
Thandalam, Chennai - 602 105	**Thandalam, Chennai - 602 105**
INTERNAL EXAMINER	**EXTERNAL EXAMIER**

ABSTRACT:

The Fitness Tracking System is a comprehensive tool designed to assist users in monitoring and achieving their fitness goals. This user-friendly application enables individuals to create detailed fitness profiles, set personalized goals, and track their health progress over time. It includes features that allow users to input essential information such as height, weight, activity level, and target weight, providing a customized fitness experience for each user.

Through the system's intuitive dashboard, users can track their Body Mass Index (BMI), monitor changes in their weight, and receive tailored workout plans and nutrition suggestions based on their personal fitness data. This tracking capability helps users stay motivated and make informed decisions to reach their fitness objectives. The application also encourages consistent engagement by allowing users to set both short-term and long-term weight goals, fostering a continuous focus on health and well-being.

Administrators of the Fitness Tracking System benefit from a centralized platform to manage user profiles and progress. They can oversee all user data, including fitness metrics, activity levels, and weight goals. This administrative functionality ensures that data is securely stored and easily accessible, contributing to a streamlined user experience.

In summary, the Fitness Tracking System offers a well-organized, interactive environment for users aiming to improve their physical health. By centralizing fitness data and providing meaningful insights, the system empowers users to achieve their fitness goals effectively, fostering a healthier lifestyle and improved overall well-being.
.

TABLE OF CONTENTS

Chapter 1 INTRODUCTION

1.1 INTRODUCTION

Physical fitness and health have become increasingly important in today's world, with many people seeking tools and resources to help them maintain a balanced lifestyle. The Fitness Tracking Application is designed to provide users with an efficient, personalized platform to monitor and achieve their fitness goals. This application simplifies the process of setting health objectives, tracking progress, and managing essential fitness data, empowering users to make informed choices about their well-being.

The Fitness Tracking Application offers a comprehensive suite of features to support a wide range of health and fitness needs. At its core, the system enables users to create detailed profiles with information such as height, weight, activity level, and fitness goals. Based on this data, the application calculates key metrics like Body Mass Index (BMI) and offers personalized recommendations for workouts and nutrition. By storing and organizing user information, the system allows individuals to track changes in their physical health over time, fostering accountability and motivation.

Developed using Java, Swing, and PostgreSQL, the Fitness Tracking Application leverages robust backend and frontend technologies to deliver a smooth user experience. PostgreSQL serves as the database backbone, ensuring reliable storage and retrieval of user profiles and fitness data, while Java and Swing power the application's interface, offering a visually appealing, responsive design. Additionally, Java's object-oriented structure enables seamless integration of complex features, including real-time BMI calculations, goal tracking, and progress analytics.

This report will delve into the development process, system architecture, and technologies used to build the Fitness Tracking Application. The objective is to showcase how the integration of these technologies results in a comprehensive, user-centric solution for fitness tracking, ultimately contributing to enhanced user engagement, health awareness, and positive lifestyle changes.

2. OBJECTIVES

- To create a centralized database for managing user profiles, health metrics, and fitness goals .
- To provide personalized workout and nutrition recommendations based on individual user profiles and fitness levels.
- To enable users to track their progress over time, including metrics such as weight, BMI, and activity levels.
- To ensure accurate data validation and secure storage of user information, maintaining privacy and compliance with data protection standards.
- To facilitate an engaging and user-friendly experience that motivates users to achieve their health and fitness goals.
- To offer real-time feedback and updates on goal achievement, helping users stay accountable and monitor their progress.
- To improve coordination between users and fitness resources, fostering a comprehensive approach to health and wellness.

3. MODULES

1. User Registration Module

The User Registration Module is responsible for capturing essential user data during the sign-up process. It collects details such as username, password, email, phone number, gender, date of birth, height, weight, target weight, activity level, and goal type. This module ensures accurate and secure data entry, with validation mechanisms for email format, password strength, and other fields. By guiding users through an easy-to-follow form, it ensures that all required information is entered, creating a seamless registration experience.

2. Profile Management Module

The Profile Management Module allows users to view, edit, and update their personal information, including height, weight, activity level, and target weight. It ensures that any updates are accurately stored and reflected in the user's account, including recalculating the BMI based on updated details. This module provides an intuitive interface for users to manage their profiles, helping them stay on top of their fitness data and ensure that their goals remain aligned with their current health status.

3. Dashboard Module

The Dashboard Module serves as the central hub for users to track their fitness progress. It presents an overview of key metrics such as BMI, current weight, target weight, and activity level. Users can also view personalized workout plans and nutrition suggestions that are tailored to their profile details. The dashboard is designed to be user-friendly and motivational, offering insights and recommendations based on the user's current data, helping them stay motivated and on track to achieve their goals.

4 BMI Calculation Module

This module calculates the user's Body Mass Index (BMI) based on their height and weight. It categorizes the results into different ranges, such as underweight, normal weight, overweight, and obese. Additionally, the BMI Calculation and Health Insights Module provides personalized health recommendations and tips based on the user's BMI category. This module ensures users are informed about their health status and provides guidance on how to achieve or maintain a healthy weight.

6. Goal Tracking Module

The Goal Tracking and Progress Analysis Module helps users set and track their fitness goals, such as target weight or activity milestones. It provides real-time feedback on the user's progress toward these goals and displays trends over time using charts and graphs. This module allows users to visualize their achievements, making it easier for them to stay motivated and adjust their plans accordingly to achieve their desired outcomes.

7. Exercise and Nutrition Recommendation Module

The Exercise and Nutrition Recommendations Module provides users with personalized workout plans and diet suggestions based on their profile information, including activity level, weight goals, and BMI. It ensures that users receive tailored recommendations to help them achieve their fitness objectives. Whether users aim to lose weight, gain muscle, or maintain their current weight, this module ensures that all fitness advice is suitable for their personal goals and preferences..

Chapter 2 SURVEY OF TECHNOLOGIES

1. SOFTWARE DESCRIPTION

The Fitness Tracking Application uses a combination of technologies to ensure an efficient and reliable system. The backend is powered by a relational database management system (RDBMS) to store and manage user data, while the frontend is designed using Java Swing for a smooth and interactive user experience. Middleware technologies enable seamless communication between the frontend and backend for real-time data synchronization.

2. LANGUAGES

The system is primarily developed using PHP as the backend programming language for server-side scripting, providing dynamic content generation and database connectivity.

1. JAVA

Role: Java is the primary language for developing the Fitness Tracking Application, utilizing its cross-platform compatibility and object-oriented features.

Usage: Java is employed for creating the business logic, user interface with AWT and Swing, and handling database operations. The application is built using Java for the server-side interactions.

Advantages:

- **Cross-Platform Compatibility:** Java applications are platform-independent, making the fitness app accessible on multiple operating systems.
- **Robustness:** With built-in exception handling, Java ensures a reliable and stable application that can handle complex tasks, such as database interactions and calculations, efficiently.

2. SQL

Role: SQL (Structured Query Language) is used for managing and manipulating the relational database that stores all data**.**

Usage: All information related to donors, blood inventory, and transactions in the system are stored and implemented as a relational schema using SQL.

Advantages:

- **Efficient Data Management**: SQL allows for efficient querying, updating, and management of large datasets.

3. AWT

Role: AWT is used for building the graphical user interface (GUI) of the Fitness Tracking Application.

Usage: AWT components such as buttons, labels, text fields, and panels are used to create interactive elements for the profile creation page, workout log inputs, and other essential features.

Advantages:

Platform Independence: AWT ensures that the app's GUI behaves consistently across different platforms.

Integration: AWT seamlessly integrates with Java Swing components for a richer user experience

4. SWING

Role: Swing is an extension of AWT that provides additional components and improved graphical user interface elements.

Usage: Swing is used for enhancing the user interface with advanced components such as combo boxes, tables, and custom dialogs to allow users to interact with the app easily.

Advantages:

Advanced UI Components: Swing enables the creation of a polished and professional-looking GUI, enhancing the overall user experience

Customizability: Swing offers extensive customization options for the app's interface, allowing it to match the desired aesthetics.

Chapter 3 REQUIREMENTS AND ANALYSIS

1. REQUIREMENT SPECIFICATION

1. Functional Requirements

User Authentication and Authorization

- **User Registration and Login**: Allow users to register, create accounts, and login securely.
- **Role-Based Access Control**: Assign specific permissions based on user roles (user, admin, etc.).

Profile Creation and Management

- **Profile Setup**: Enable users to create and manage their profiles with personal details like height, weight, activity level, and fitness goals.
- **Profile Updates**: Allow users to edit their profiles as needed, including personal and fitness-related information.

Goal Setting and Tracking

- **Goal Creation**: Allow users to set fitness goals such as weight loss, muscle gain, or activity level improvement.
- **Progress Tracking**: Provide tools for users to track progress toward their goals, including weight and activity level changes.

Workout and Nutrition Plan Management

- **Customized Plans**: Generate personalized workout and nutrition plans based on user profiles and goals.
- **Plan Updates**: Allow users to update their workout and nutrition plans based on progress or changes in goals.

Admin Dashboard

- **Comprehensive Overview**: Provide admins with an overview of user registrations, activity tracking, and system performance.
- **Management Tools**: Include tools for generating reports, managing user profiles, and monitoring fitness plan engagement

2. Non-Functional Requirements

Security

- **Data Encryption**: Ensure encryption of sensitive data both in transit and at rest to protect user privacy.
- **Compliance**: Adhere to relevant data privacy regulations and fitness industry standards to protect personal and health data.

Performance

- **Scalability**: Design the system to handle growing numbers of users and fitness data efficiently.
- **Response Time**: Ensure quick responses to user actions, including logging workouts, updating goals, and retrieving progress reports

Reliability

- **Availability**: Ensure the system is available with minimal downtime to support continuous user interaction.
- **Data Backup**: Regularly back up user data, including workout logs and progress records, to prevent data loss

Maintainability

- **Modular Design**: Implement a modular design to allow for easier updates, bug fixes, and feature enhancements.
- **Comprehensive Documentation**: Provide clear and comprehensive documentation for developers and end-users to ensure smooth system maintenance and usage.

Interoperability

- **System Integration**: Enable integration with third-party health management systems, wearables, and fitness tracking devices for a seamless experience.

2. HARDWARE AND SOFTWARE REQUIREMENTS

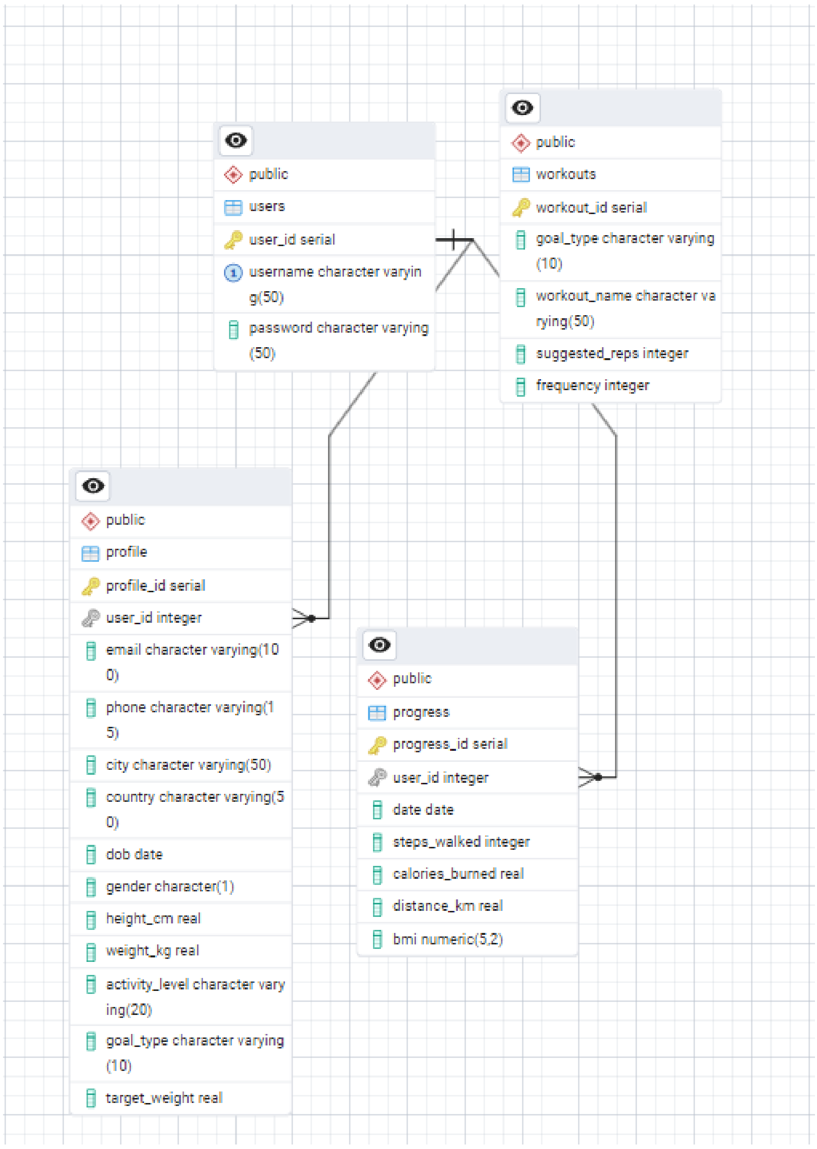
Hardware Requirements:

•**Desktop PC or Laptop**: A reliable desktop PC or laptop to host and run the Fitness Tracking Application.
•**Processor**: Intel® Core™ i3-6006U CPU @ 2.00GHz or equivalent for efficient processing.
•**RAM**: 4.00 GB RAM to handle concurrent user activities, fitness data storage, and app operations.
•**System Architecture**: 64-bit operating system with an x64-based processor for optimal performance.
•**Monitor Resolution**: 1024 x 768 monitor resolution for clear display of the system interface.
•**Input Devices**: Keyboard and Mouse for user interaction.
•**Server**: A reliable server with sufficient processing power and storage for managing large volumes of fitness data.
•**Network**: Stable internet connection for syncing data and app updates

Software Requirements:

•**Operating System**: Windows 10 or higher.
•**Code Editor**: IntelliJ IDEA for Java development.
•**Front End**: HTML, CSS, JavaScript for building a responsive and interactive user interface.
•**Back End**: Java (for backend logic and processing).
•**Database**: PostgreSQL for storing user profiles, workout logs, and progress data.
•**Version Control**: Git for managing code versions and collaboration.
•**API Integration**: RESTful APIs for connecting the app with external fitness tracking devices and wearables.

3.4 ER DIAGRAM



Chapter 4 PROGRAM CODE

1. USER LOGIN PAGE

```
import javax.swing.*;
import java.awt.*;
import java.awt.event.ActionEvent;
import java.awt.event.ActionListener;
import java. sql. Connection;
public class StylishLoginPage extends JFrame {
    private JTextField usernameField;
    private JPasswordField passwordField;
    private JButton loginButton;
    private JButton createAccountButton;

    public StylishLoginPage() {
        setTitle("Login");
        setSize(400, 500);
        setDefaultCloseOperation(JFrame.EXIT_ON_CLOSE);
        setLocationRelativeTo(null);
        setLayout(new BorderLayout());


        JLabel headerLabel = new JLabel("Welcome to
FitTracker", SwingConstants.CENTER);
        headerLabel.setFont(new Font("Arial", Font.BOLD,
24));
        headerLabel.setForeground(new Color(58, 134, 255));
        add(headerLabel, BorderLayout.NORTH);
JPanel formPanel = new JPanel();
        formPanel.setLayout(new GridBagLayout());
        formPanel.setBackground(Color.WHITE);
        GridBagConstraints gbc = new GridBagConstraints();
        gbc.insets = new Insets(10, 10, 10, 10);
        gbc.fill = GridBagConstraints.HORIZONTAL;
       JLabel usernameLabel = new JLabel("Username:");
        usernameLabel.setFont(new Font("Arial", Font.PLAIN,
16));
        usernameLabel.setForeground(new Color(58, 134,
255));
        gbc.gridx = 0;
        gbc.gridy = 0;
        formPanel.add(usernameLabel, gbc);

        usernameField = new JTextField(15);
        usernameField.setFont(new Font("Arial", Font.PLAIN,
14));

usernameField.setBorder(BorderFactory.createMatteBorder(0,
0, 2, 0, new Color(58, 134, 255)));
        gbc.gridx = 1;
        formPanel.add(usernameField, gbc);
JLabel passwordLabel = new JLabel("Password:");
 passwordLabel.setFont(new Font("Arial", Font.PLAIN, 16));
        passwordLabel.setForeground(new Color(58, 134,
255));
gbc.gridx = 0;
        gbc.gridy = 1;
        formPanel.add(passwordLabel, gbc);
        passwordField = new JPasswordField(15);
        passwordField.setFont(new Font("Arial",
Font.PLAIN, 14));

passwordField.setBorder(BorderFactory.createMatteBorder(
0, 0, 2, 0, new Color(58, 134, 255)));
        gbc.gridx = 1;
        formPanel.add(passwordField, gbc);


loginButton = new JButton("Login");
        loginButton.setFont(new Font("Arial", Font.BOLD,
16));
        loginButton.setBackground(new Color(58, 134,
255));
        loginButton.setForeground(Color.WHITE);
        loginButton.setFocusPainted(false);
        gbc.gridx = 0;
        gbc.gridy = 2;
        gbc.gridwidth = 2;
        formPanel.add(loginButton, gbc);


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

```
createAccountButton.setFont(new Font("Arial", Font.PLAIN,
14));
        createAccountButton.setForeground(new Color(58, 134,
255));
        createAccountButton.setContentAreaFilled(false);
        createAccountButton.setBorderPainted(false);
        gbc.gridy = 3;
        formPanel.add(createAccountButton, gbc);

        add(formPanel, BorderLayout.CENTER);

loginButton.addActionListener(new ActionListener() {
            @Override
            public void actionPerformed(ActionEvent e) {
                handleLogin();
            }
        });

        createAccountButton.addActionListener(new
ActionListener() {
            @Override
            public void actionPerformed(ActionEvent e) {
                openSignUpForm();
            }
        });
    }
private void handleLogin() {
        String username = usernameField.getText();
        String password = new
String(passwordField.getPassword());
```

```
 Connection connection = DatabaseUtil.connect();  // Use DatabaseUtil to get
connection
        if (connection != null) {
            int userId =
DatabaseUtil.validateLoginGetUserId(connection,username, password);
            if (userId != -1) {
                JOptionPane.showMessageDialog(this, "Login successful for "
+ username);
                openDashboard(userId, connection);
            } else {
                JOptionPane.showMessageDialog(this, "Invalid username or
password.", "Login Failed", JOptionPane.ERROR_MESSAGE);
            }
        } else {
            JOptionPane.showMessageDialog(this, "Database connection
failed.", "Error", JOptionPane.ERROR_MESSAGE);
        }
    }

    private void openDashboard(int userId, Connection connection) {
        new DashboardPage(userId, connection).setVisible(true);
        dispose(); // Close the login page
    }



    private void openSignUpForm() {
        new SignUpForm().setVisible(true);
        dispose(); // Close login form
    }


    public static void main(String[] args) {
        SwingUtilities.invokeLater(() -> {
            new StylishLoginPage().setVisible(true);
        });
    }
}
```

2. SIGNUP FORM PAGE

```
import javax.swing.*;
import java.awt.*;
import java.awt.event.ActionEvent;
import java.awt.event.ActionListener;

public class SignUpForm extends JFrame {
    private JTextField usernameField;
    private JPasswordField passwordField;
    private JPasswordField confirmPasswordField;
    private JButton signUpButton;
    private JLabel titleLabel;

    public SignUpForm() {
        setTitle("Create New Account");
        setSize(400, 500);
        setDefaultCloseOperation(JFrame.DISPOSE_ON_CLOSE);
        setLocationRelativeTo(null);
        setLayout(new BorderLayout());

        // Title Label
        titleLabel = new JLabel("Create Your Account",
SwingConstants.CENTER);
        titleLabel.setFont(new Font("Arial", Font.BOLD, 24));
        titleLabel.setForeground(new Color(58, 134, 255));
        titleLabel.setPreferredSize(new Dimension(400, 60));
        add(titleLabel, BorderLayout.NORTH);

        // Center Panel for the form
        JPanel formPanel = new JPanel();
        formPanel.setLayout(new GridBagLayout());
        formPanel.setBackground(new Color(245, 245, 245));

        GridBagConstraints gbc = new GridBagConstraints();
        gbc.insets = new Insets(10, 10, 10, 10);
        gbc.fill = GridBagConstraints.HORIZONTAL;

        // Username Label and Field
        JLabel usernameLabel = new JLabel("Username");
        usernameLabel.setFont(new Font("Arial", Font.PLAIN, 16));
        usernameLabel.setForeground(new Color(58, 134, 255));
        gbc.gridx = 0;
        gbc.gridy = 0;
        formPanel.add(usernameLabel, gbc);

        usernameField = createStyledTextField();
        gbc.gridx = 1;
        formPanel.add(usernameField, gbc);

ProfileCreationPage profilePage = new ProfileCreationPage(userId);
            profilePage.setVisible(true); // Open ProfileCreationPage
            dispose(); // Close sign-up form
// Password Label and Field
        JLabel passwordLabel = new JLabel("Password");
        passwordLabel.setFont(new Font("Arial", Font.PLAIN, 16));
        passwordLabel.setForeground(new Color(58, 134, 255));
        gbc.gridx = 0;
        gbc.gridy = 1;
        formPanel.add(passwordLabel, gbc);

        passwordField = createStyledPasswordField();
        gbc.gridx = 1;
        formPanel.add(passwordField, gbc);


        JLabel confirmPasswordLabel = new JLabel("Confirm Password");
        confirmPasswordLabel.setFont(new Font("Arial", Font.PLAIN, 16));
        confirmPasswordLabel.setForeground(new Color(58, 134, 255));
        gbc.gridx = 0;
        gbc.gridy = 2;
        formPanel.add(confirmPasswordLabel, gbc);

        confirmPasswordField = createStyledPasswordField();
        gbc.gridx = 1;
        formPanel.add(confirmPasswordField, gbc);


        signUpButton = new JButton("Sign Up");
        signUpButton.setFont(new Font("Arial", Font.BOLD, 18));
        signUpButton.setBackground(new Color(58, 134, 255));
signUpButton.setForeground(Color.WHITE);
        signUpButton.setFocusPainted(false);
        signUpButton.setPreferredSize(new Dimension(150, 40));

signUpButton.setCursor(Cursor.getPredefinedCursor(Cursor.HAND_CURSOR));
        signUpButton.addActionListener(new ActionListener() {
            @Override
            public void actionPerformed(ActionEvent e) {
                handleSignUp();
            }
        });

        gbc.gridx = 0;
        gbc.gridy = 3;
        gbc.gridwidth = 2;
        formPanel.add(signUpButton, gbc);

        add(formPanel, BorderLayout.CENTER);
    }
```

```
 private JTextField createStyledTextField() {
        JTextField textField = new JTextField(15);
        textField.setFont(new Font("Arial", Font.PLAIN, 16));
        textField.setForeground(new Color(58, 134, 255));
        textField.setBackground(Color.WHITE);
        textField.setBorder(BorderFactory.createLineBorder(new Color(58, 
134, 255), 2));
        textField.setPreferredSize(new Dimension(250, 40));
        return textField;
    }

    private JPasswordField createStyledPasswordField() {
        JPasswordField passwordField = new JPasswordField(15);
        passwordField.setFont(new Font("Arial", Font.PLAIN, 16));
        passwordField.setForeground(new Color(58, 134, 255));
        passwordField.setBackground(Color.WHITE);
        passwordField.setBorder(BorderFactory.createLineBorder(new 
Color(58, 134, 255), 2));
        passwordField.setPreferredSize(new Dimension(250, 40));
        return passwordField;
    }
private void handleSignUp() {
        // Extract input from fields
        String username = usernameField.getText();
        String password = new String(passwordField.getPassword());
        String confirmPassword = new 
String(confirmPasswordField.getPassword());

        // Basic validation
        if (username.isEmpty() || password.isEmpty() || 
confirmPassword.isEmpty()) {
            JOptionPane.showMessageDialog(this, "All fields are required.", 
"Error", JOptionPane.ERROR_MESSAGE);
            return;
        }
        if (!password.equals(confirmPassword)) {
            JOptionPane.showMessageDialog(this, "Passwords do not match.", 
"Error", JOptionPane.ERROR_MESSAGE);
            return;
        }




        Integer userId = DatabaseUtil.insertUser(username, password);
        if (userId != null) {
            JOptionPane.showMessageDialog(this, "Account created 
successfully!");
```

```
 // Pass the user ID (or other necessary details) to the
ProfileCreationPage, if needed.
 } else {
            JOptionPane.showMessageDialog(this, "Username already exists or
database error.", "Sign Up Failed", JOptionPane.ERROR_MESSAGE);
        }

    }

    public static void main(String[] args) {
        SwingUtilities.invokeLater(() -> {
            new SignUpForm().setVisible(true);
        });
    }
}
```

3. PROFILE CREATION PAGE

```
import javax.swing.*;
import java.awt.*;
import java.awt.event.ActionEvent;
import java.awt.event.ActionListener;
import java.sql.Date;

public class ProfileCreationPage extends JFrame {
    private JTextField emailField, phoneField,
cityField, countryField, dobField, heightField,
weightField, targetWeightField;
    private JComboBox<String> genderBox,
activityLevelBox, goalTypeBox;
    private JButton saveProfileButton;
    private int userId = 1; // Replace with actual
user ID as needed

    public ProfileCreationPage(int userId) {
        this.userId = userId;
        setTitle("Complete Your Profile");
        setSize(500, 700);

setDefaultCloseOperation(JFrame.EXIT_ON_CLOSE);
        setLocationRelativeTo(null);
        setLayout(new BorderLayout());

        // Header
        JLabel headerLabel = new JLabel("Profile
Setup", SwingConstants.CENTER);
        headerLabel.setFont(new Font("Arial",
Font.BOLD, 26));
        headerLabel.setForeground(new Color(34,
167, 240));

headerLabel.setBorder(BorderFactory.createEmptyBor
der(20, 10, 20, 10));
        add(headerLabel, BorderLayout.NORTH);

        // Center Panel for form
        JPanel formPanel = new JPanel();
        formPanel.setLayout(new GridBagLayout());
        formPanel.setBackground(Color.WHITE);
        GridBagConstraints gbc = new
GridBagConstraints();
        gbc.insets = new Insets(10, 10, 10, 10);
        gbc.fill = GridBagConstraints.HORIZONTAL;
```

```
 // Helper method to create styled labels
        Font labelFont = new Font("Arial",
Font.BOLD, 16);
        Color labelColor = new Color(34, 167,
240);

        // Create fields with labels
        emailField =
createStyledTextField("Email", labelFont,
labelColor, gbc, formPanel);
        phoneField =
createStyledTextField("Phone", labelFont,
labelColor, gbc, formPanel);
        cityField = createStyledTextField("City",
labelFont, labelColor, gbc, formPanel);
        countryField =
createStyledTextField("Country", labelFont,
labelColor, gbc, formPanel);
        dobField = createStyledTextField("Date of
Birth (YYYY-MM-DD)", labelFont, labelColor, gbc,
formPanel);
        heightField =
createStyledTextField("Height (cm)", labelFont,
labelColor, gbc, formPanel);
        weightField =
createStyledTextField("Weight (kg)", labelFont,
labelColor, gbc, formPanel);
        targetWeightField =
createStyledTextField("Target Weight (kg)",
labelFont, labelColor, gbc, formPanel);

        // Gender dropdown
        JLabel genderLabel = new JLabel("Gender");
        genderLabel.setFont(labelFont);
        genderLabel.setForeground(labelColor);
        gbc.gridx = 0;
        gbc.gridy++;
        formPanel.add(genderLabel, gbc);

        genderBox = new JComboBox<>(new
String[]{"M", "F", "Other"});
        genderBox.setFont(new Font("Arial",
Font.PLAIN, 14));
        genderBox.setPreferredSize(new
Dimension(200, 30));
        gbc.gridx = 1;
        formPanel.add(genderBox, gbc);
```

```
activityLevelBox.setFont(new Font("Arial", Font.PLAIN,
14));
        activityLevelBox.setPreferredSize(new
Dimension(200, 30));
        gbc.gridx = 1;
        formPanel.add(activityLevelBox, gbc);

        // Goal Type dropdown
        JLabel goalLabel = new JLabel("Goal Type");
        goalLabel.setFont(labelFont);
        goalLabel.setForeground(labelColor);
        gbc.gridx = 0;
        gbc.gridy++;
        formPanel.add(goalLabel, gbc);

        goalTypeBox = new JComboBox<>(new
String[]{"Gain", "Lose","Maintain"});
        goalTypeBox.setFont(new Font("Arial", Font.PLAIN,
14));
        goalTypeBox.setPreferredSize(new Dimension(200,
30));
        gbc.gridx = 1;
        formPanel.add(goalTypeBox, gbc);

        // Save Profile button
        saveProfileButton = new JButton("Save Profile");
        saveProfileButton.setFont(new Font("Arial",
Font.BOLD, 18));
        saveProfileButton.setBackground(new Color(34,
167, 240));
        saveProfileButton.setForeground(Color.WHITE);
        saveProfileButton.setFocusPainted(false);
        saveProfileButton.setPreferredSize(new
Dimension(200, 50));
        saveProfileButton.addActionListener(new
ActionListener() {
            @Override
            public void actionPerformed(ActionEvent e) {
                saveProfile();
            }
        });
         gbc.gridx = 0;
        gbc.gridy++;
        gbc.gridwidth = 2;
        gbc.anchor = GridBagConstraints.CENTER;
        formPanel.add(saveProfileButton, gbc);

        add(formPanel, BorderLayout.CENTER);
    }
```

```
private JTextField createStyledTextField(String
placeholder, Font labelFont, Color labelColor,
GridBagConstraints gbc, JPanel panel) {
JLabel label = new JLabel(placeholder);
        label.setFont(labelFont);
        label.setForeground(labelColor);
        gbc.gridx = 0;
        gbc.gridy++;
        panel.add(label, gbc);

        JTextField textField = new JTextField(20);
        textField.setFont(new Font("Arial", Font.PLAIN,
14));

textField.setBorder(BorderFactory.createLineBorder(new
Color(34, 167, 240), 2));
        textField.setPreferredSize(new Dimension(250,
30));
        gbc.gridx = 1;
        panel.add(textField, gbc);

        return textField;
    }

    private void saveProfile() {
        // Validate if fields are filled
        if (emailField.getText().isEmpty() ||
phoneField.getText().isEmpty() ||
cityField.getText().isEmpty() ||
                countryField.getText().isEmpty() ||
dobField.getText().isEmpty() ||
heightField.getText().isEmpty() ||
                weightField.getText().isEmpty() ||
targetWeightField.getText().isEmpty() ||
                genderBox.getSelectedItem() == null ||
activityLevelBox.getSelectedItem() == null ||
goalTypeBox.getSelectedItem() == null) {
            JOptionPane.showMessageDialog(this, "Please
fill in all the details.", "Incomplete Information",
JOptionPane.WARNING_MESSAGE);
            return;
        }

        try {
            String email = emailField.getText().trim();
            String phone = phoneField.getText().trim();
            String city = cityField.getText().trim();
            String country =
countryField.getText().trim();
            Date dob =
Date.valueOf(dobField.getText().trim()); // Parse
date
            String gender = (String)
genderBox.getSelectedItem();
            float height =
Float.parseFloat(heightField.getText().trim());
            float weight =
Float.parseFloat(weightField.getText().trim());
            String activityLevel = (String)
activityLevelBox.getSelectedItem();
            String goalType = (String)
goalTypeBox.getSelectedItem();

            // Call DatabaseUtil to save the profile
            boolean isSuccess =
DatabaseUtil.insertProfile(userId, email, phone,
city, country, dob, gender, height, weight,
activityLevel, goalType, targetWeight);

            if (isSuccess) {
                JOptionPane.showMessageDialog(this,
"Profile saved successfully!");

                dispose();
                new
StylishLoginPage().setVisible(true);
                // Proceed to dashboard or next step
            } else {
                JOptionPane.showMessageDialog(this,
"Error saving profile. Please try again.", "Database
Error", JOptionPane.ERROR_MESSAGE);
            }

        } catch (Exception e) {
            JOptionPane.showMessageDialog(this,
"Error: " + e.getMessage(), "Invalid Input",
JOptionPane.ERROR_MESSAGE);
        }
    }

    //public static void main(String[] args) {
    //     SwingUtilities.invokeLater(() -> new
ProfileCreationPage().setVisible(true));
  //   }
}
```

4.DASHBOARD PAGE

```
import javax.swing.*;
import javax.swing.border.EmptyBorder;
import javax.swing.border.LineBorder;
import java.awt.event.ActionListener;
import java.awt.*;
import java.sql.*;
import java.text.DecimalFormat;
import java.text.SimpleDateFormat;
import java.util.Calendar;
import java.util.Date;
import java.text.ParseException;

public class DashboardPage extends JFrame {
    private int userId;
    private Connection connection;

    // Declare JTextArea for each panel's content
    private JTextArea profileContentArea, bmiContentArea,
nutritionContentArea, workoutContentArea;

    public DashboardPage(int userId, Connection connection) {
        this.userId = userId;
        this.connection = connection;
        setTitle("Dashboard - MyFitnessApp");
        setSize(1100, 800);
        setDefaultCloseOperation(JFrame.EXIT_ON_CLOSE);
        setLocationRelativeTo(null);

        // Main layout
        JPanel mainPanel = new JPanel(new BorderLayout());
        mainPanel.setBackground(new Color(240, 240, 240));

        // Sidebar panel
        JPanel sidebarPanel = createSidebarPanel();
        sidebarPanel.setPreferredSize(new Dimension(250,
getHeight())); // Dashboard content panel
        JPanel contentPanel = createContentPanel();

        // Add panels to main panel
        mainPanel.add(sidebarPanel, BorderLayout.WEST);
        mainPanel.add(contentPanel, BorderLayout.CENTER);

        add(mainPanel);
        setVisible(true);
```

```
        // Load user data into panels
```

```
loadUserProfile();
        calculateAndDisplayBMI();
        displayNutritionSuggestions();
        displayWorkoutPlan();
    }
    private JPanel createSidebarPanel() {
        JPanel sidebar = new JPanel();
        sidebar.setBackground(new Color(60, 63, 65));
        sidebar.setLayout(new BoxLayout(sidebar, BoxLayout.Y_AXIS));
        sidebar.setBorder(new EmptyBorder(20, 20, 20, 20));

        JLabel logoLabel = new JLabel("Welcome to MyFitnessApp");
        logoLabel.setFont(new Font("Arial", Font.BOLD, 22));
        logoLabel.setForeground(new Color(200, 200, 200));
        logoLabel.setAlignmentX(Component.CENTER_ALIGNMENT);

        sidebar.add(logoLabel);
        sidebar.add(Box.createRigidArea(new Dimension(0, 30))); // Space
below logo

        // Add all sidebar buttons
        sidebar.add(createSidebarButton("Profile", "user_icon.png"));
        sidebar.add(Box.createRigidArea(new Dimension(0, 10)));
        sidebar.add(createSidebarButton("BMI", "bmi_icon.png"));
        sidebar.add(Box.createRigidArea(new Dimension(0, 10)));
        sidebar.add(createSidebarButton("Nutrition",
"nutrition_icon.png"));
        sidebar.add(Box.createRigidArea(new Dimension(0, 10)));
        sidebar.add(createSidebarButton("Workout", "workout_icon.png"));
        sidebar.add(Box.createRigidArea(new Dimension(0, 20))); //
Spacer before
 // Profile Update Button
        JButton updateProfileButton = createSidebarActionButton("Update
Profile", e -> openProfileUpdatePage());
        sidebar.add(updateProfileButton);
        sidebar.add(Box.createRigidArea(new Dimension(0, 10))); //
Spacer

        // Logout Button
        JButton logoutButton = createSidebarActionButton("Logout", e ->
{
            new StylishLoginPage().setVisible(true); // Open login page
            JFrame topFrame = (JFrame)
SwingUtilities.getWindowAncestor(sidebar);
            if (topFrame != null) {
                topFrame.dispose(); // Close the dashboard window
            }
 });
        sidebar.add(logoutButton);

        return sidebar;
    }

    // Helper function to style and create action buttons
    private JButton createSidebarActionButton(String text,
ActionListener action) {
        JButton button = new JButton(text);
        button.setMaximumSize(new Dimension(200, 50));
        button.setBackground(new Color(80, 80, 80));
        button.setForeground(Color.WHITE);
        button.setFont(new Font("Arial", Font.PLAIN, 16));
        button.setHorizontalAlignment(SwingConstants.LEFT);
        button.setIconTextGap(15);
        button.setBorder(new EmptyBorder(10, 20, 10, 10));
        button.setFocusPainted(false);
        button.addActionListener(action);
        return button;
    }


    private JButton createSidebarButton(String text, String iconPath) {
        JButton button = new JButton(text, new ImageIcon(iconPath));
        button.setMaximumSize(new Dimension(200, 50));
        button.setBackground(new Color(80, 80, 80));
        button.setForeground(Color.WHITE);
        button.setFont(new Font("Arial", Font.PLAIN, 16));
        button.setHorizontalAlignment(SwingConstants.LEFT);
        button.setIconTextGap(15);
        button.setBorder(new EmptyBorder(10, 20, 10, 10));
        button.setFocusPainted(false);
        return button;
    }

    private JPanel createContentPanel() {
        JPanel contentPanel = new JPanel(new GridLayout(2, 2, 15, 15));
        contentPanel.setBorder(new EmptyBorder(20, 20, 20, 20));
        contentPanel.setBackground(new Color(240, 240, 240));

        JPanel profilePanel = createCardPanel("User Profile", new
Color(60, 179, 113));
```

```
profileContentArea = (JTextArea) profilePanel.getComponent(2);  //
Assigning the text area for updating
        JPanel bmiPanel = createCardPanel("BMI Calculation", new
Color(30, 144, 255));
        bmiConatentArea = (JTextArea) bmiPanel.getComponent(2);
        JPanel nutritionPanel = createCardPanel("Nutrition
Suggestions", new Color(255, 140, 0));
        nutritionContentArea = (JTextArea)
nutritionPanel.getComponent(2);
        JPanel workoutPanel = createCardPanel("Workout Plan", new
Color(128, 0, 128));
        workoutContentArea = (JTextArea) workoutPanel.getComponent(2);

        contentPanel.add(profilePanel);
        contentPanel.add(bmiPanel);
        contentPanel.add(nutritionPanel);
        contentPanel.add(workoutPanel);

        return contentPanel;
    }

    private JPanel createCardPanel(String title, Color color) {
        JPanel cardPanel = new JPanel();
        cardPanel.setLayout(new BoxLayout(cardPanel,
BoxLayout.Y_AXIS));
        cardPanel.setBackground(Color.WHITE);

cardPanel.setBorder(BorderFactory.createCompoundBorder(BorderFactory.cr
eateLineBorder(color, 2, true),
                new EmptyBorder(20, 20, 20, 20)
        ));

        JLabel titleLabel = new JLabel(title);
        titleLabel.setFont(new Font("Arial", Font.BOLD, 18));
        titleLabel.setForeground(color);
        titleLabel.setAlignmentX(Component.CENTER_ALIGNMENT);

        JTextArea contentArea = new JTextArea("Loading " +
title.toLowerCase() + "...");
        contentArea.setFont(new Font("Arial", Font.PLAIN, 14));
        contentArea.setLineWrap(true);
        contentArea.setWrapStyleWord(true);
        contentArea.setEditable(false);
        contentArea.setBorder(BorderFactory.createEmptyBorder(10, 10,
10, 10));
        contentArea.setBackground(new Color(245, 245, 245));
contentArea.setBackground(new Color(245, 245, 245));

        cardPanel.add(titleLabel);
        cardPanel.add(Box.createRigidArea(new Dimension(0, 15)));
        cardPanel.add(contentArea);

        return cardPanel;
    }


private void loadUserProfile() {
    String userQuery = "SELECT username FROM Users WHERE user_id = ?";
    String profileQuery = "SELECT gender, dob FROM Profile WHERE
user_id = ?";

    try (PreparedStatement userStmt =
connection.prepareStatement(userQuery);
         PreparedStatement profileStmt =
connection.prepareStatement(profileQuery)) {

        // Set the user_id for both queries
        userStmt.setInt(1, userId);
        profileStmt.setInt(1, userId);

        // Execute the user query to get the username
        ResultSet userRs = userStmt.executeQuery();
        String username = "";
        if (userRs.next()) {
            username = userRs.getString("username");
        }

        // Execute the profile query to get the gender and dob
        ResultSet profileRs = profileStmt.executeQuery();
        String gender = "";
        String dob = "";
if (profileRs.next()) {
            gender = profileRs.getString("gender");
            dob = profileRs.getString("dob");
        }

        // Calculate age from dob
        int age = calculateAge(dob);

        // Build the profile info string with the required fields:
Name, Gender, Age
        String profileInfo = "Name: " + username + "\n"
                + "Gender: " + gender + "\n"
                + "Age: " + age + " years
        profileContentArea.setText(profileInfo);
```

```
} catch (SQLException e) {
        e.printStackTrace();
    }
}
private int calculateAge(String dob) {
        try {
            SimpleDateFormat sdf = new SimpleDateFormat("yyyy-MM-
dd");
            Date birthDate = sdf.parse(dob);
            Calendar birthCalendar = Calendar.getInstance();
            birthCalendar.setTime(birthDate);

            int birthYear = birthCalendar.get(Calendar.YEAR);
            int currentYear =
Calendar.getInstance().get(Calendar.YEAR);
            int age = currentYear - birthYear;

            // Check if the birthday has passed this year, if not
subtract 1 from age
            int currentMonth =
Calendar.getInstance().get(Calendar.MONTH);
            int birthMonth = birthCalendar.get(Calendar.MONTH);
            if (currentMonth < birthMonth || (currentMonth ==
birthMonth && Calendar.getInstance().get(Calendar.DAY_OF_MONTH) <
birthCalendar.get(Calendar.DAY_OF_MONTH))) {
                age--;
            }
            return age;
        } catch (ParseException e) {
            e.printStackTrace();
            return 0;
        }
    }
 private void openProfileUpdatePage() {
        // Implement the logic to open the profile update page
        Connection connection = DatabaseUtil.connect();
        // You can use a new frame or dialog to open the update form
        ProfileUpdatePage updatePage = new
ProfileUpdatePage(userId,connection);  // Assuming you have a
ProfileUpdatePage class
        updatePage.setVisible(true);
    }


    private void calculateAndDisplayBMI() {
        String query = "SELECT height_cm, weight_kg FROM Profile
WHERE user_id = ?";
        try (PreparedStatement stmt =
connection.prepareStatement(query)) {
 stmt.setInt(1, userId);
           ResultSet rs = stmt.executeQuery();
if (rs.next()) {
               double height = rs.getDouble("height_cm");
               double weight = rs.getDouble("weight_kg");
               height =height/100;
               double bmi = weight / (height * height);
               String bmiCategory = getBMICategory(bmi);

               bmiContentArea.setText("BMI: " + new
DecimalFormat("#.##").format(bmi) + "\nCategory: " + bmiCategory);
           }
       } catch (SQLException e) {
           e.printStackTrace();
       }
    }

    private String getBMICategory(double bmi) {
        if (bmi < 18.5) return "Underweight";
        else if (bmi < 24.9) return "Normal weight";
        else if (bmi < 29.9) return "Overweight";
        else return "Obesity";
    }

    private void displayNutritionSuggestions() {
        String query = "SELECT goal_type FROM Profile WHERE user_id =
?";
        try (PreparedStatement stmt =
connection.prepareStatement(query)) {
            stmt.setInt(1, userId);
            ResultSet rs = stmt.executeQuery();

            if (rs.next()) {
                String goalType = rs.getString("goal_type");

nutritionContentArea.setText(getNutritionSuggestions(goalType));
            }
        } catch (SQLException e) {
            e.printStackTrace();
        }
    }

    private String getNutritionSuggestions(String goalType) {
        if (goalType.equalsIgnoreCase("gain")) {
            return "Weight Gain: Focus on calorie-dense foods like
whole grains (brown rice, oats), lean proteins (chicken, fish, paneer),
and healthy fats (nuts, seeds, avocado). Include dairy (milk, yogurt)
and snacks like dry fruits and smoothies to boost calorie intake. Eat
5-6 smaller meals daily for steady energy.";
        }
```

```
else if (goalType.equalsIgnoreCase("lose")) {
            return "Weight Loss: Opt for low-calorie, nutrient-rich
foods like vegetables (spinach, cucumbers) and fruits (berries,
apples). Include whole grains (quinoa, bajra) and lean proteins (fish,
eggs, lentils). Drink plenty of water and herbal teas, and avoid
refined carbs and fried foods to control hunger and support fat loss.";
        }

else {
            return "Maintain Weight: Maintain a balanced diet with
whole grains (brown rice, oats), lean proteins (fish, tofu, paneer),
and healthy fats (nuts, olive oil). Include fiber-rich vegetables
(broccoli, carrots) and fruits (oranges, pomegranates). Drink water
regularly, practice portion control, and stay active to maintain
weight.";
        }
    }

    private void displayWorkoutPlan() {
        String query = "SELECT activity_level, goal_type FROM Profile
WHERE user_id = ?";
        try (PreparedStatement stmt =
connection.prepareStatement(query)) {
            stmt.setInt(1, userId);
            ResultSet rs = stmt.executeQuery();

            if (rs.next()) {
                String activityLevel = rs.getString("activity_level");
                String goalType = rs.getString("goal_type");

workoutContentArea.setText(getWorkoutPlan(activityLevel, goalType));
            }
catch (SQLException e) {
            e.printStackTrace();
        }
    }

    private String getWorkoutPlan(String activityLevel, String
goalType) {
        if (goalType.equalsIgnoreCase("gain")) {
            return activityLevel.equalsIgnoreCase("high") ? "Plan:
Intense strength training, high-calorie intake." : "Plan: Moderate
strength training, increase protein.";
        } else if (goalType.equalsIgnoreCase("lose")) {
            return activityLevel.equalsIgnoreCase("high") ? "Plan:
Cardio-intensive, calorie deficit, balanced meals." : "Plan: Moderate
cardio, healthy caloric intake.";
        } else {
            return "Plan: Regular moderate workouts, balanced diet,
stay hydrated.";}}}
```

5. DATABASE CONNECTION

```
import java.sql.*;
import java.util.*;
import java.sql.Date;
public class DatabaseUtil {
    private static final String DB_URL =
"jdbc:postgresql://localhost:5432/FitnessTracker"; // Replace with your
DB name
    private static final String USER = "FAREED"; // PostgreSQL username
    private static final String PASSWORD = "Bismillah"; // PostgreSQL
password

    // Method to connect to the database
    public static Connection connect() {
        try {
            return DriverManager.getConnection(DB_URL, USER, PASSWORD);
        } catch (SQLException e) {
            System.out.println("Connection failed: " + e.getMessage());
            return null;
        }
    }

    public static Integer insertUser(String username, String password) {
        String query = "INSERT INTO Users (username, password) VALUES
(?, ?) RETURNING user_id";
        try (Connection conn = connect();
             PreparedStatement stmt = conn.prepareStatement(query)) {

            // Set username and password parameters
            stmt.setString(1, username);
            stmt.setString(2, password);

            // Execute the query and retrieve the generated user_id
            ResultSet rs = stmt.executeQuery();
            if (rs.next()) {
                return rs.getInt("user_id"); // Return the user_id if
insertion was successful
            }
        } catch (SQLException e) {
            // Check for unique constraint violation (username already
exists)
            if ("23505".equals(e.getSQLState())) { // PostgreSQL unique
constraint violation code
                System.out.println("Error: Username already exists.");
            } else {
                System.out.println("Error inserting user: " +
e.getMessage());
            }
```

```
 }
        return null; // Return null if insertion fails
    } public static int validateLoginGetUserId(Connection conn, 
String username, String password) {
    int userId = -1;
    String query = "SELECT user_id FROM Users WHERE username = ? AND 
password = ?";

    try (PreparedStatement stmt = conn.prepareStatement(query)) {
        stmt.setString(1, username);
        stmt.setString(2, password);

        ResultSet rs = stmt.executeQuery();
        if (rs.next()) {
            userId = rs.getInt("user_id");  // Adjusted to match 
column name in the database
        }
    } catch (SQLException e) {
        e.printStackTrace();
    }

    return userId;
}




    // Method to insert a new profile for a user
    public static boolean insertProfile(int userId, String email, 
String phone, String city, String country,
                                        Date dob, String gender, 
float height, float weight,
                                        String activityLevel, String 
goalType, float targetWeight) {
        String query = "INSERT INTO Profile (user_id, email, phone, 
city, country, dob, gender, height_cm, weight_kg, activity_level, 
goal_type, target_weight) " +
                "VALUES (?, ?, ?, ?, ?, ?, ?, ?, ?, ?, ?, ?)";
        try (Connection conn = connect();
             PreparedStatement stmt = conn.prepareStatement(query)) {
            stmt.setInt(1, userId);
            stmt.setString(2, email);
            stmt.setString(3, phone);
            stmt.setString(4, city);
            stmt.setString(5, country);
            stmt.setDate(6, dob);  // java.sql.Date format
            stmt.setString(7, gender);
            stmt.setFloat(8, height);
stmt.setFloat(9, weight);
            stmt.setString(10, activityLevel);
            stmt.setString(11, goalType);
            stmt.setFloat(12, targetWeight);

            int rowsAffected = stmt.executeUpdate();
            return rowsAffected > 0; // Returns true if insertion was
successful
        } catch (SQLException e) {
            System.out.println("Error inserting profile: " +
e.getMessage());
            return false;
        }
    } // Method to fetch user profile by userId
    public static UserProfile getUserProfile(int userId) {
        String query = "SELECT * FROM Profile WHERE user_id = ?";
        try (Connection conn = connect();
             PreparedStatement stmt = conn.prepareStatement(query)) {
            stmt.setInt(1, userId);
            ResultSet rs = stmt.executeQuery();

            if (rs.next()) {
                String email = rs.getString("email");
                String phone = rs.getString("phone");
                String city = rs.getString("city");
                String country = rs.getString("country");
                Date dob = rs.getDate("dob");
                String gender = rs.getString("gender");
                float height = rs.getFloat("height_cm");
                float weight = rs.getFloat("weight_kg");
                String activityLevel =
rs.getString("activity_level");
                String goalType = rs.getString("goal_type");
                float targetWeight = rs.getFloat("target_weight");

                return new UserProfile(userId, email, phone, city,
country, dob, gender, height, weight, activityLevel, goalType,
targetWeight);
            }
        } catch (SQLException e) {
            System.out.println("Error fetching user profile: " +
e.getMessage());
        }
        return null;
    }
```

```
// Method to update user profile
    public static boolean updateUserProfile(int userId, String
email, String phone, String city, String country,
                                            Date dob, String gender,
float height, float weight,
                                            String activityLevel,
String goalType, float targetWeight) {
        String query = "UPDATE Profile SET email = ?, phone = ?,
city = ?, country = ?, dob = ?, gender = ?, " +
                "height_cm = ?, weight_kg = ?, activity_level = ?,
goal_type = ?, target_weight = ? WHERE user_id = ?";
        try (Connection conn = connect();
             PreparedStatement stmt = conn.prepareStatement(query))
{
            stmt.setString(1, email);
            stmt.setString(2, phone);
            stmt.setString(3, city);
            stmt.setString(4, country);
            stmt.setDate(5, dob);
            stmt.setString(6, gender);
            stmt.setFloat(7, height);
            stmt.setFloat(8, weight);
            stmt.setString(9, activityLevel);
            stmt.setString(10, goalType);
            stmt.setFloat(11, targetWeight);
            stmt.setInt(12, userId);

            int rowsAffected = stmt.executeUpdate();
            return rowsAffected > 0; // Returns true if update was
successful
        } catch (SQLException e) {
            System.out.println("Error updating user profile: " +
e.getMessage());
        }
        return false;
    }

    // Method to calculate BMI
    public static float calculateBMI(float height, float weight) {
        float heightInMeters = height / 100; // Convert height from
cm to meters
        return weight / (heightInMeters * heightInMeters);
    }

    // Method to provide nutrition suggestion based on goal type
(gain/lose/maintain)
    public static String getNutritionSuggestion(String goalType) {
        if (goalType.equalsIgnoreCase("gain")) {
            return "Suggested Calorie Intake: 2500-3000 kcal/day.
Focus on high-protein and calorie-dense foods.";
        } else if (goalType.equalsIgnoreCase("lose")) {
            return "Suggested Calorie Intake: 1500-2000 kcal/day. Focus
on low-calorie and nutrient-dense foods.";
        } else if (goalType.equalsIgnoreCase("maintain")) {
            return "Suggested Calorie Intake: 2000-2500 kcal/day. A
balanced diet of carbs, proteins, and fats.";
        }
        return "No goal specified.";
    }
 // Method to provide workout plan based on goal type
    public static String getWorkoutPlan(String goalType) {
        if (goalType.equalsIgnoreCase("gain")) {
            return "Workout Plan: Focus on strength training, weight
lifting, and compound exercises.";
        } else if (goalType.equalsIgnoreCase("lose")) {
            return "Workout Plan: Focus on cardio, HIIT (High-Intensity
Interval Training), and fat-burning exercises.";
        } else if (goalType.equalsIgnoreCase("maintain")) {
            return "Workout Plan: A balanced mix of cardio and strength
training.";
        }
        return "No goal specified.";
    }
}
```

6. PROFILE UPDATE PAGE

```
import javax.swing.*;
import java.awt.*;
import java.awt.event.ActionEvent;
import java.awt.event.ActionListener;
import java.sql.Connection;
import java.sql.PreparedStatement;
import java.sql.ResultSet;
import java.sql.SQLException;

public class ProfileUpdatePage extends JFrame {
    private JTextField emailField, phoneField, cityField,
countryField, heightField, weightField, targetWeightField;
    private JComboBox<String> activityLevelBox, goalTypeBox,
genderBox;
    private JButton updateButton, deleteProfileButton;
    private int userId;
    private Connection connection;

    public ProfileUpdatePage(int userId, Connection connection)
{
        this.userId = userId;
        this.connection = connection;

        setTitle("Update Profile");
        setSize(400, 600);
        setDefaultCloseOperation(JFrame.DISPOSE_ON_CLOSE);
        setLocationRelativeTo(null);
        setLayout(new BorderLayout());

        // Header Label
        JLabel headerLabel = new JLabel("Update Profile",
SwingConstants.CENTER);
        headerLabel.setFont(new Font("Arial", Font.BOLD, 24));
        headerLabel.setForeground(new Color(58, 134, 255));
        add(headerLabel, BorderLayout.NORTH);

        // Center Panel for form
        JPanel formPanel = new JPanel();
        formPanel.setLayout(new GridBagLayout());
        formPanel.setBackground(Color.WHITE);
```

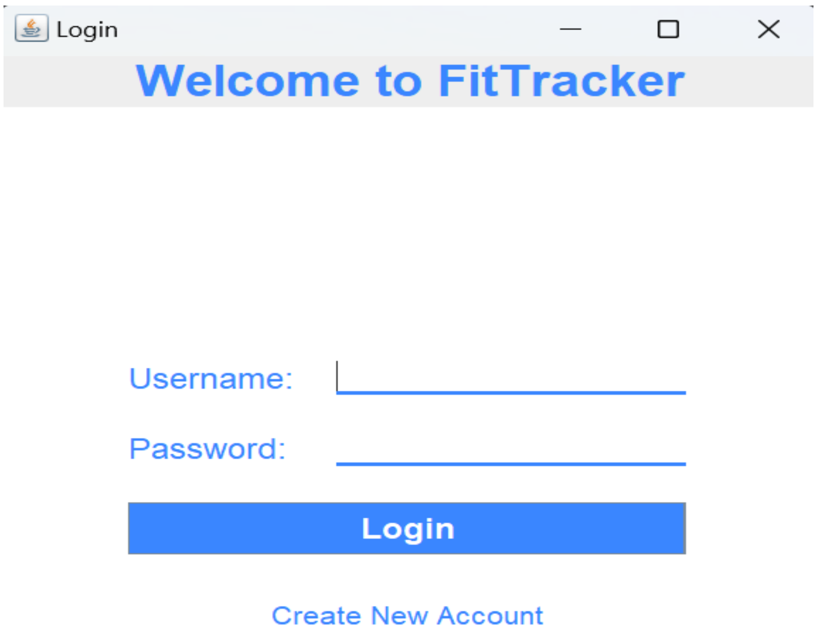
```
 GridBagConstraints gbc = new GridBagConstraints();
        gbc.insets = new Insets(10, 10, 10, 10);
        gbc.fill = GridBagConstraints.HORIZONTAL;

        // Load current user profile data
        loadUserProfileData(formPanel, gbc);

        // Update Button
        updateButton = new JButton("Update Profile");
        updateButton.setFont(new Font("Arial", Font.BOLD, 16));
        updateButton.setBackground(new Color(58, 134, 255));
        updateButton.setForeground(Color.WHITE);
        updateButton.setFocusPainted(false);
        gbc.gridx = 0;
        gbc.gridy = 10;
        gbc.gridwidth = 2;
        formPanel.add(updateButton, gbc);

        // Add Delete Button
        deleteProfileButton = new JButton("Delete Profile");
        deleteProfileButton.setFont(new Font("Arial",
Font.BOLD, 16));
        deleteProfileButton.setBackground(Color.RED);
        deleteProfileButton.setForeground(Color.WHITE);
        deleteProfileButton.setFocusPainted(false);
        gbc.gridy = 11;
        formPanel.add(deleteProfileButton, gbc);

        add(formPanel, BorderLayout.CENTER);

        // Action Listeners
        updateButton.addActionListener(new ActionListener() {
            @Override
            public void actionPerformed(ActionEvent e) {
                System.out.println("Update button clicked"); //
Debug statement
                updateProfile();
            }
        });

        deleteProfileButton.addActionListener(new
ActionListener() {
            @Override
            public void actionPerformed(ActionEvent e) {
                int confirm =
JOptionPane.showConfirmDialog(null,
                        "Are you sure you want to delete your
profile? All your details will be lost.",
 "Delete Confirmation",
                        JOptionPane.YES_NO_OPTION);
                if (confirm == JOptionPane.YES_OPTION) {
                    deleteProfile();
                }
            }
        });
    }

    private void loadUserProfileData(JPanel formPanel,
GridBagConstraints gbc) {
        String query = "SELECT email, phone, city, country,
gender, height_cm, weight_kg, activity_level, goal_type,
target_weight FROM Profile WHERE user_id = ?";
        try (PreparedStatement stmt =
connection.prepareStatement(query)) {
            stmt.setInt(1, userId);
            ResultSet rs = stmt.executeQuery();

            if (rs.next()) {
                // Email Field
                addLabelAndField(formPanel, gbc, "Email:",
emailField = new JTextField(rs.getString("email")), 0, 0);
                // Phone Field
                addLabelAndField(formPanel, gbc, "Phone:",
phoneField = new JTextField(rs.getString("phone")), 0, 1);
                // City Field
                addLabelAndField(formPanel, gbc, "City:",
cityField = new JTextField(rs.getString("city")), 0, 2);
                // Country Field
                addLabelAndField(formPanel, gbc, "Country:",
countryField = new JTextField(rs.getString("country")), 0, 3);
                // Gender Combo Box
                addLabelAndComboBox(formPanel, gbc, "Gender:",
genderBox = new JComboBox<>(new String[]{"M", "F"}), 0, 4,
rs.getString("gender"));
                // Height Field
                addLabelAndField(formPanel, gbc, "Height (cm):",
heightField = new
JTextField(String.valueOf(rs.getDouble("height_cm"))), 0, 5);
                // Weight Field
                addLabelAndField(formPanel, gbc, "Weight (kg):",
weightField = new
JTextField(String.valueOf(rs.getDouble("weight_kg"))), 0, 6);
```

```
 // Activity Level Combo Box
                addLabelAndComboBox(formPanel, gbc, "Activity
Level:", activityLevelBox = new JComboBox<>(new
String[]{"Low", "Medium", "High"}), 0, 7,
rs.getString("activity_level"));
                // Goal Type Combo Box
                addLabelAndComboBox(formPanel, gbc, "Goal
Type:", goalTypeBox = new JComboBox<>(new String[]{"Gain",
"Lose", "Maintain"}), 0, 8, rs.getString("goal_type"));
                // Target Weight Field
                addLabelAndField(formPanel, gbc, "Target
Weight (kg):", targetWeightField = new
JTextField(String.valueOf(rs.getDouble("target_weight"))), 0,
9);
            } } catch (SQLException e) {
            e.printStackTrace();
        }
    }

    private void updateProfile() {
        try {
            System.out.println("Attempting to update
profile..."); // Debug statement
            String updateQuery = "UPDATE Profile SET email =
?, phone = ?, city = ?, country = ?, gender = ?, height_cm =
?, weight_kg = ?, activity_level = ?, goal_type = ?,
target_weight = ? WHERE user_id = ?";
            try (PreparedStatement stmt =
connection.prepareStatement(updateQuery)) {
                stmt.setString(1, emailField.getText());
                stmt.setString(2, phoneField.getText());
                stmt.setString(3, cityField.getText());
                stmt.setString(4, countryField.getText());
                stmt.setString(5, (String)
genderBox.getSelectedItem());
                stmt.setDouble(6,
Double.parseDouble(heightField.getText()));
                stmt.setDouble(7,
Double.parseDouble(weightField.getText()));
                stmt.setString(8, (String)
activityLevelBox.getSelectedItem());
                stmt.setString(9, (String)
goalTypeBox.getSelectedItem());
                stmt.setDouble(10,
Double.parseDouble(targetWeightField.getText()));
                stmt.setInt(11, userId);
                int rowsUpdated = stmt.executeUpdate();
                if (rowsUpdated > 0) {
                    JOptionPane.showMessageDialog(this,
"Profile updated successfully.");
                    dispose();
                } else {
                    JOptionPane.showMessageDialog(this,
"Error updating profile.");
                }
            }
        } catch (SQLException e) {
            e.printStackTrace();
        }
    } private void deleteProfile() {
        String deleteProfileQuery = "DELETE FROM Profile
WHERE user_id = ?";
        String deleteUserQuery = "DELETE FROM Users WHERE
user_id = ?";

        try (PreparedStatement profileStmt =
connection.prepareStatement(deleteProfileQuery);
             PreparedStatement userStmt =
connection.prepareStatement(deleteUserQuery)) {

            // Set the user ID for deletion
            profileStmt.setInt(1, userId);
            userStmt.setInt(1, userId);

            // Delete from Profile table
            int profileRowsDeleted =
profileStmt.executeUpdate();

            // Delete from Users table
            int userRowsDeleted =
userStmt.executeUpdate();

            // Check if both deletions were successful
            if (profileRowsDeleted > 0 && userRowsDeleted
> 0) {
                JOptionPane.showMessageDialog(this,
"Account deleted successfully.");
```

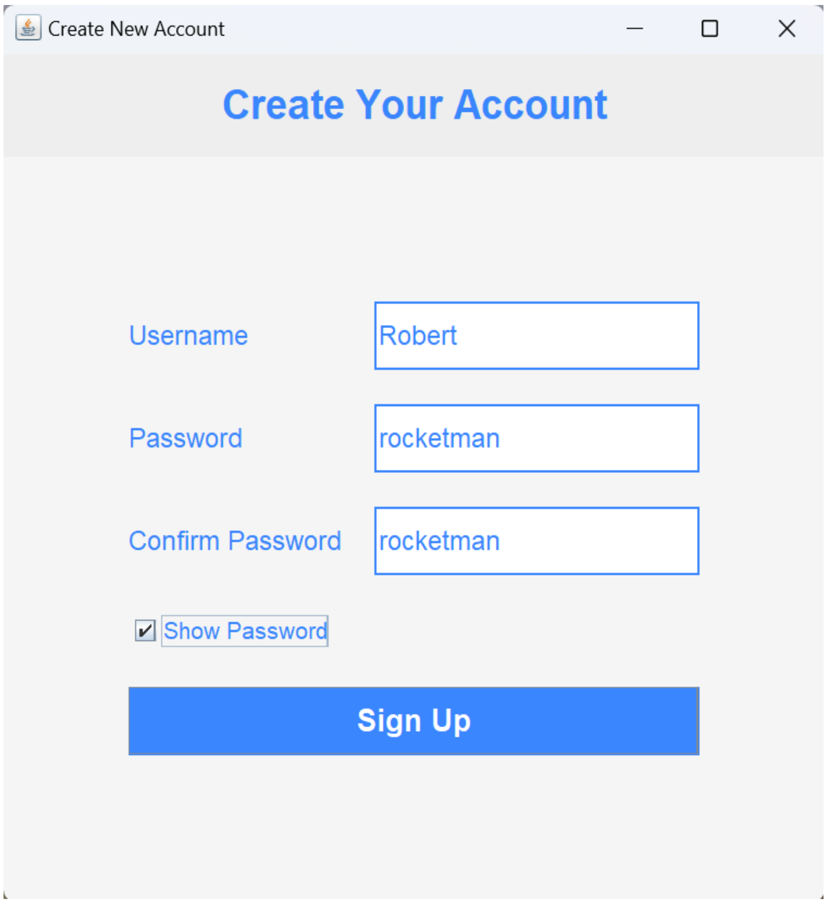
```
                // Open the login page and close the dashboard
                new StylishLoginPage().setVisible(true); //
Assuming StylishLoginPage constructor accepts the connection
                dispose();
            } else {
                JOptionPane.showMessageDialog(this, "Error
deleting account.");
            }
 } catch (SQLException e) {
            e.printStackTrace();
            JOptionPane.showMessageDialog(this, "An error
occurred while deleting the account.");
        }
    }

    private void addLabelAndField(JPanel panel,
GridBagConstraints gbc, String labelText, JTextField textField,
int x, int y) {
        gbc.gridx = x;
        gbc.gridy = y;
        panel.add(new JLabel(labelText), gbc);

        gbc.gridx = x + 1;
        gbc.gridy = y;
        panel.add(textField, gbc);
    }
    private void addLabelAndComboBox(JPanel panel,
GridBagConstraints gbc, String labelText, JComboBox<String>
comboBox, int x, int y, String selectedValue) {
        gbc.gridx = x;
        gbc.gridy = y;
        panel.add(new JLabel(labelText), gbc);

        gbc.gridx = x + 1;
        gbc.gridy = y;
        panel.add(comboBox, gbc);

        if (selectedValue != null) {
            comboBox.setSelectedItem(selectedValue);
        }
    }


}
```

Chapter 5 RESULTS AND DISCUSSION

LOGIN PAGE



SIGN UP PAGE



Create New Account

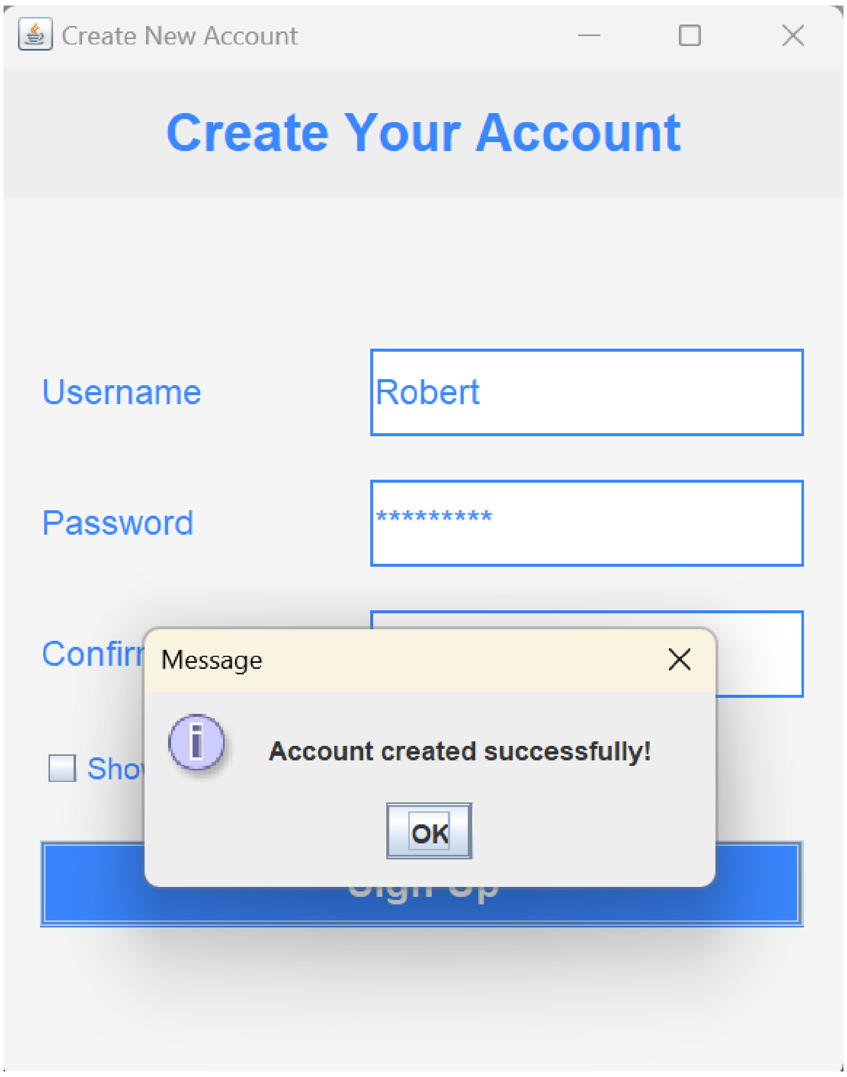
Create Your Account

Username: Robert

Password: rocketman

Confirm Password: rocketman

☑ Show Password



Create New Account

Create Your Account

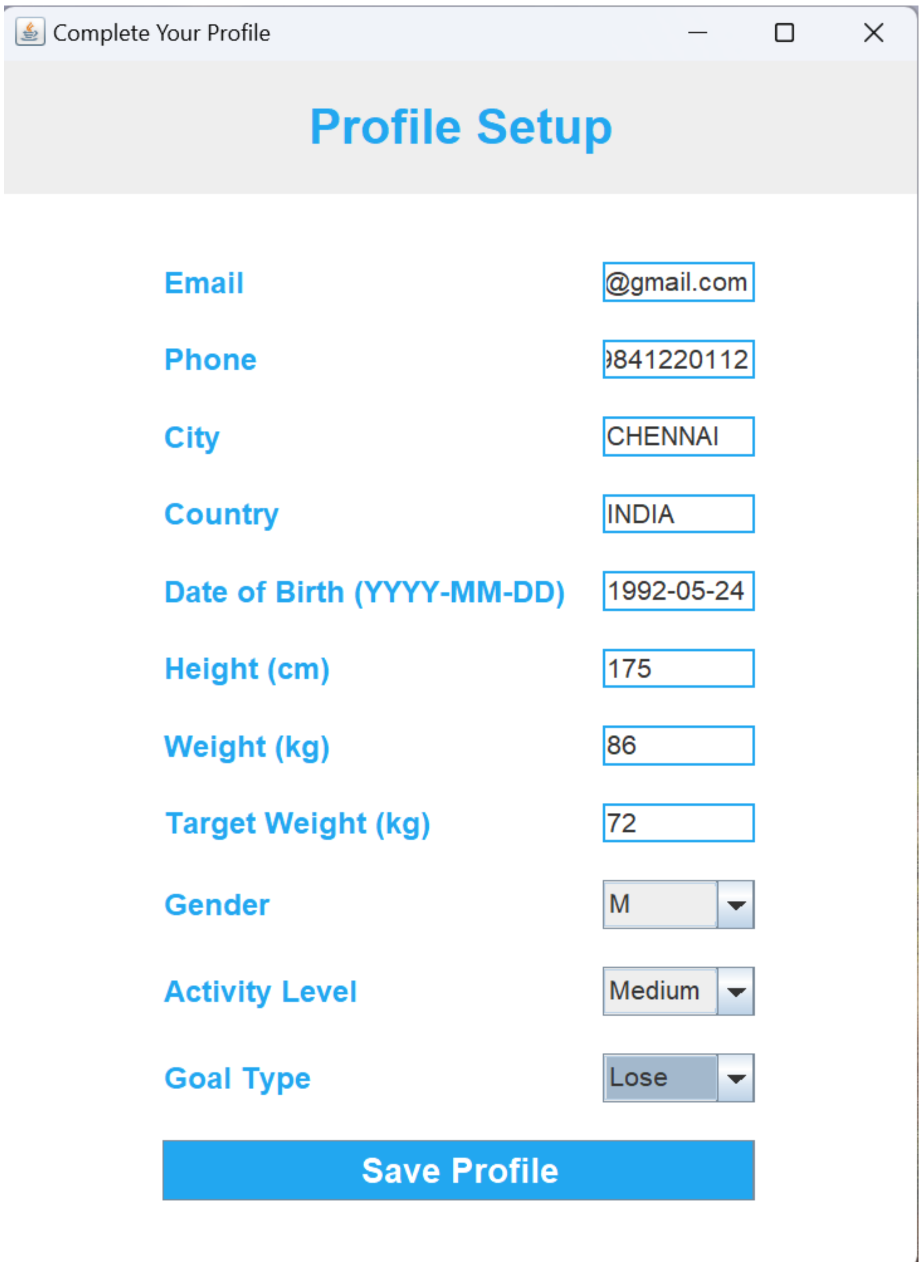
Username: Robert

Password: *********

Message

Account created successfully!

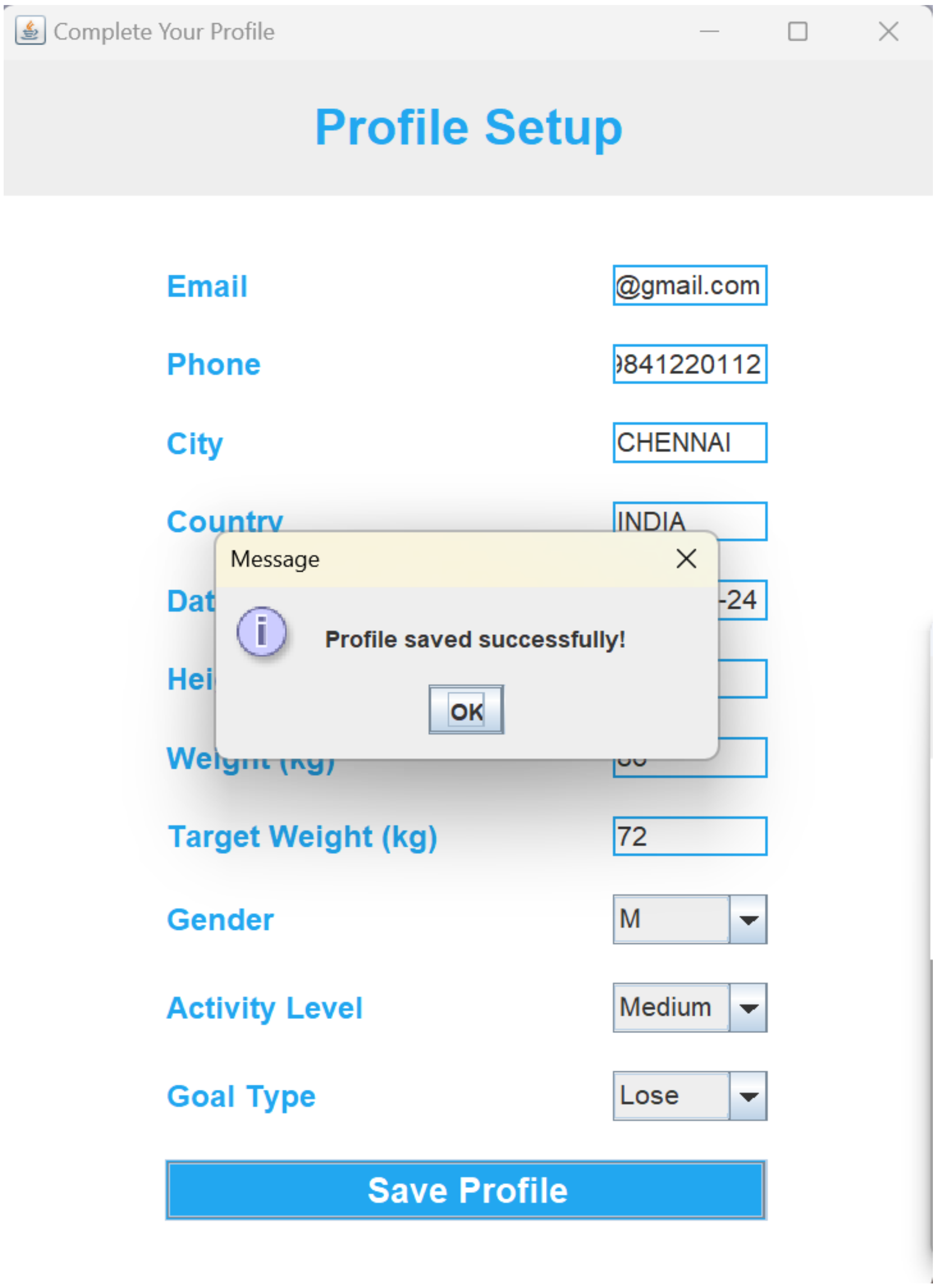
PROFILE CREATION PAGE



Complete Your Profile

Profile Setup

Email	@gmail.com
Phone	9841220112
City	CHENNAI
Country	INDIA
Date of Birth (YYYY-MM-DD)	1992-05-24
Height (cm)	175
Weight (kg)	86
Target Weight (kg)	72
Gender	M
Activity Level	Medium
Goal Type	Lose



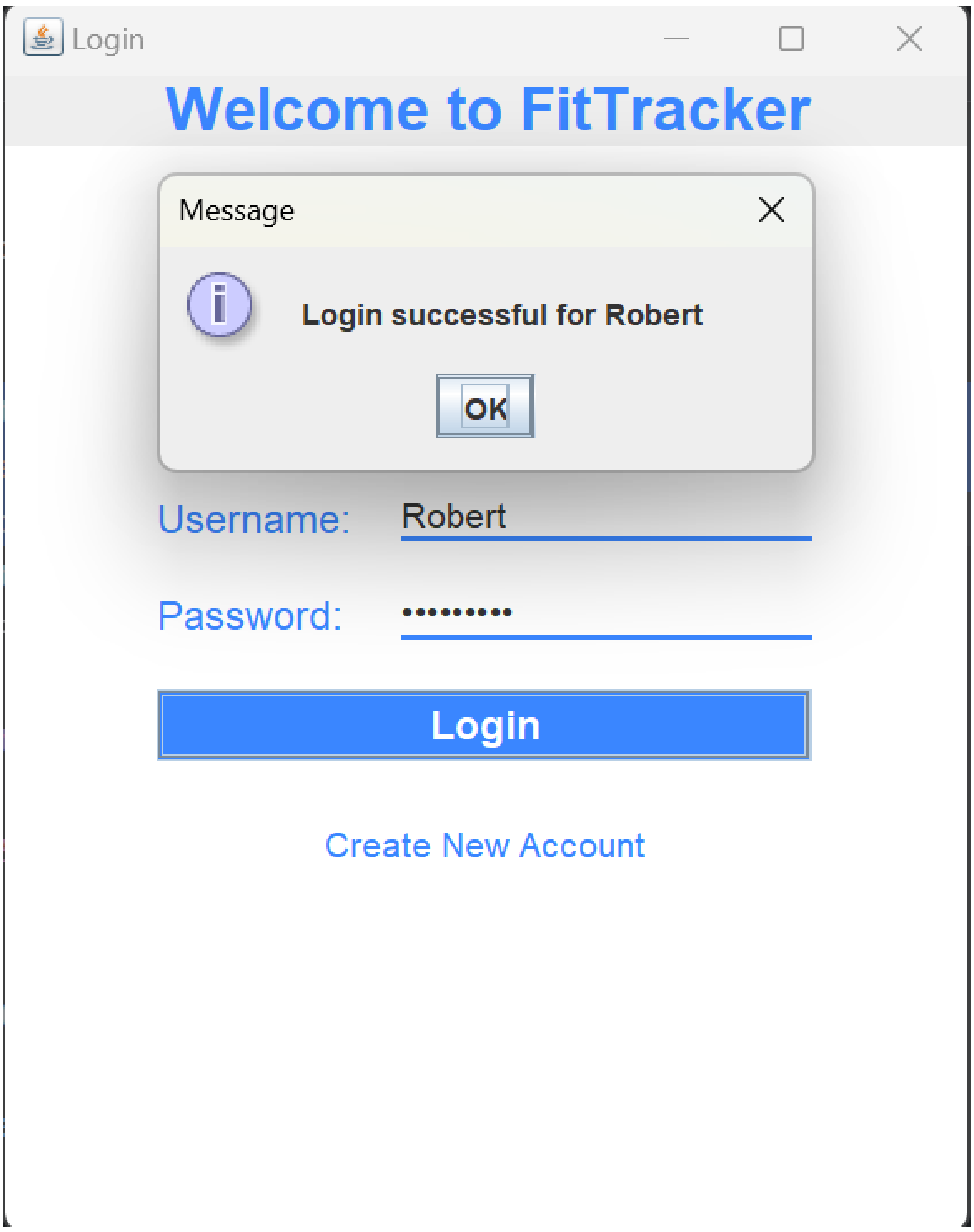
Complete Your Profile

Profile Setup

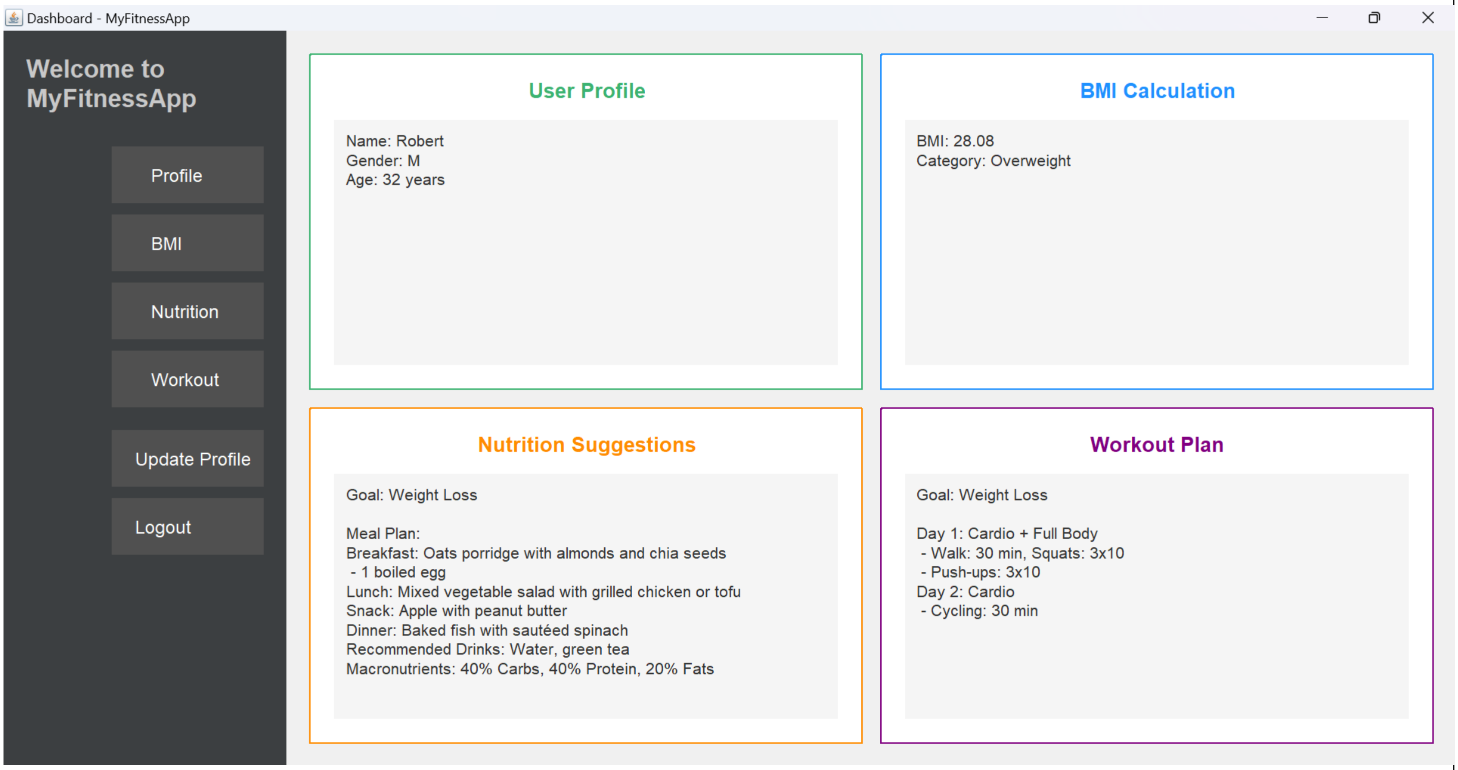
Email	@gmail.com
Phone	9841220112
City	CHENNAI
Country	INDIA
Dat	-24
Hei	
Weight (kg)	80
Target Weight (kg)	72
Gender	M
Activity Level	Medium
Goal Type	Lose

Message

Profile saved successfully!


Login
Welcome to FitTracker
Message
Login successful for Robert
OK
Username:
Robert
Password:
••••••••••
Login
Create New Account

USER DASHBOARD



User Profile

Name: Robert
Gender: M
Age: 32 years

BMI Calculation

BMI: 28.08
Category: Overweight

Nutrition Suggestions

Goal: Weight Loss

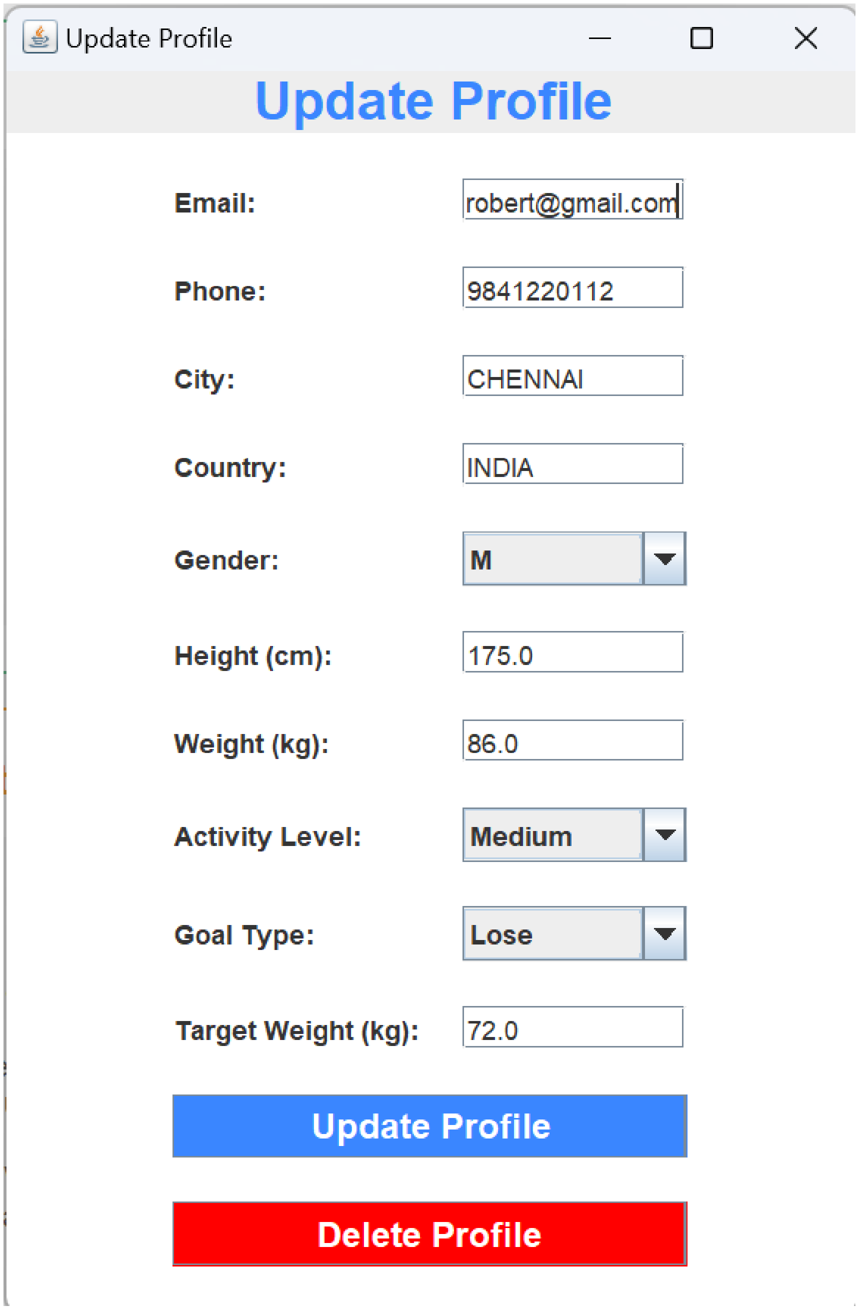
Meal Plan:
Breakfast: Oats porridge with almonds and chia seeds
 - 1 boiled egg
Lunch: Mixed vegetable salad with grilled chicken or tofu
Snack: Apple with peanut butter
Dinner: Baked fish with sautéed spinach
Recommended Drinks: Water, green tea
Macronutrients: 40% Carbs, 40% Protein, 20% Fats

Workout Plan

Goal: Weight Loss

Day 1: Cardio + Full Body
 - Walk: 30 min, Squats: 3x10
 - Push-ups: 3x10
Day 2: Cardio
 - Cycling: 30 min

USER PROFILE UPDATE/DELETE



Email:	robert@gmail.com
Phone:	9841220112
City:	CHENNAI
Country:	INDIA
Gender:	M
Height (cm):	175.0
Weight (kg):	86.0
Activity Level:	Medium
Goal Type:	Lose
Target Weight (kg):	72.0

Chapter 6 CONCLUSION

6.1 Conclusion

The development of the Fitness Tracking Application signifies a pivotal advancement in empowering individuals to monitor and improve their health and fitness journeys. By leveraging a combination of modern programming frameworks and database technologies, the application achieves its primary goal of providing users with a seamless, personalized, and interactive fitness management experience.

The integration of **Java AWT/Swing** for the user interface ensures an intuitive and visually engaging experience, allowing users to navigate features like login, user creation, and dashboard functionality with ease. The use of **PostgreSQL** as the backend database offers secure, efficient, and scalable management of user data, including BMI calculations, nutrition suggestions, and workout plans
.
Through its comprehensive functional capabilities, the application supports essential tasks such as user registration, BMI tracking, personalized nutrition and workout recommendations, and profile management with update/delete options. The modular design ensures flexibility, allowing users to seamlessly interact with features while maintaining data accuracy and security.

From a non-functional perspective, the application is designed with performance, usability, and scalability in mind. It offers a reliable solution capable of handling user interactions efficiently while being adaptable to future enhancements, such as integration with wearables or additional fitness insights.

In conclusion, the Fitness Tracking Application addresses the challenges of personalized fitness management by providing a holistic platform that caters to both beginner and advanced users. It not only meets immediate user needs but also serves as a scalable and adaptable solution for evolving health and fitness demands, fostering healthier lifestyles and greater user satisfaction.

Chapter 7 REFERENCES

7.1 REFERENCES

1. Smith, J., & Brown, T. (2021). *Designing User Interfaces with Swing.* Addison-Wesley.

2. Silberschatz, A., Korth, H. F., & Sudarshan, S. (2020). *Database System Concepts.* McGraw Hill. FitProTech. (2022).

3. Guidelines for Building Fitness Applications. Retrieved from https://www.fitprotech.com/resources

4. Dieter, R., & Matthew, K. (2019). *Developing Java Applications: Swing and AWT Integration.* O'Reilly Media.