

FITNESS TRACKER APP

-

A MINI PROJECT REPORT

Submitted by

Lohit S	230701165
Manu S D	230701179
Mithun A R	230701186

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 TRACKER APP**” is the bonafide work of

“LOHIT S (230701165) , MANU S D (230701179) & MITHUN A R (230701186)”

who carried out the project work under my supervision.

Submitted for the Practical Examination held on _____

SIGNATURE

Mrs. K. MAHESMEENA,

Assistant Professor,

Computer Science and Engineering, Rajalakshmi Engineering College, (Autonomous),

Thandalam, Chennai - 602 105

INTERNAL EXAMINER

EXTERNAL EXAMINER

ABSTRACT

The Fitness Tracking Application is a comprehensive platform designed to empower users to monitor and manage their health and wellness effectively. It provides features for tracking food intake, physical activity, and medical conditions, helping users achieve a balanced lifestyle while maintaining health records.

This application integrates a user-friendly JavaFX interface, allowing seamless navigation through modules like food tracking, activity tracking, and user profile management. The backend utilizes a MySQL database to securely store user data, ensuring efficient and reliable data handling. Key functionalities include profile creation, database connectivity for persistent storage, and interactive features for user engagement.

The Fitness Tracking Application aims to enhance the user's ability to visualize their fitness progress through an intuitive design and robust backend framework. By facilitating accurate tracking and insights, this project provides an efficient tool for personal health management, suitable for users at all fitness levels.

TABLE OF CONTENTS

I. INTRODUCTION

- I.1 Introduction
- I.2 Objectives
- I.3 Modules

II. SURVEY OF TECHNOLOGIES

- II.1 Software Description
- II.2 Languages
- II.3 Java
- II.4 SQL

III. REQUIREMENTS AND ANALYSIS

- III.1 Requirement Specification
 - III.1.1 Functional Requirements
- III.2 Hardware and Software Requirements
- III.3 ER Diagram

IV. PROGRAM CODE

- IV.1 Backend details and Program Code

V. OUTPUTS

- V.1 Outputs

VI. CONCLUSION

- VI.1 Conclusion

VII. REFERENCES

- VII.1 References

INTRODUCTION

1.1 INTRODUCTION

Effective management of health and fitness data is essential for individuals aiming to achieve their wellness goals and maintain a balanced lifestyle. The **Fitness Tracking Application** provides a streamlined and user-friendly platform designed to securely and efficiently manage various aspects of personal health. With a comprehensive set of tools, users can log food intake, monitor physical activity, and track medical conditions effortlessly.

The system offers core functionalities, including user registration, secure login, and detailed modules for tracking daily fitness activities. It maintains essential information such as personal profiles, activity records, and health data, ensuring that all fitness-related information is accessible in one place. These features empower users to make informed decisions about their health and wellness journey, promoting better habits and overall well-being.

Developed using **JavaFX** for an interactive graphical user interface and **MySQL** for secure data management, the Fitness Tracking Application leverages the power of Java for backend logic and efficient processing of user inputs. The intuitive JavaFX interface enhances usability, delivering a modern and engaging experience for users.

This report details the system's design, architecture, and technology integration, demonstrating how Java, MySQL, and JavaFX combine to create a secure and efficient fitness tracking solution. The system aims to provide a seamless and reliable user experience, addressing the need for accessible and effective health management tools.

1.2 OBJECTIVES

- To develop a centralized database for securely managing user profiles, fitness logs, and health-related data.
- To enable efficient handling of fitness tracking activities such as food logging, activity monitoring, and medical condition management.
- To provide a secure login system for user authentication, ensuring the confidentiality and integrity of personal data.

- To allow users to easily access and view essential health details, including progress summaries, dietary habits, and activity history, through a streamlined interface.
- To ensure compliance with data security standards for storing and managing sensitive user information.
- To provide a scalable platform that can accommodate future enhancements, such as wearable integration and personalized fitness recommendations.

1.3 MODULES

1. User Registration Module

This module captures essential user information such as name, email, password, and fitness preferences. Data is securely stored in the database, ensuring only authorized users can access their fitness records through encryption and validation mechanisms.

2. Profile Management Module

This module provides a centralized interface for users to view and manage their personal profiles. Users can update their details, such as name, email, and password, while the system ensures data accuracy and security through robust authentication.

3. Food Tracking Module

This module allows users to log their daily food intake and monitor nutritional information. Users can track dietary habits, set goals, and view progress summaries, enabling better management of their health and nutrition.

4. Activity Tracking Module

This module facilitates the logging of physical activities such as workouts and exercises. Users can monitor their activity levels, set fitness goals, and track progress over time, fostering an active lifestyle.

5. Medical Condition Management Module

This module helps users manage and log medical conditions or health concerns. It enables users to record details about their health status, providing insights for better health monitoring.

6. Admin Dashboard Module

The dashboard provides tools for overseeing user activity logs and managing system operations. Role-based access ensures data security and prevents unauthorized access to sensitive information.

7. Database Management Module

This module is responsible for securely storing and retrieving user profiles, fitness logs, and other health-related data. MySQL serves as the backend database, ensuring efficient, reliable, and secure data management.

SURVEY OF TECHNOLOGIES

2.1 SOFTWARE DESCRIPTION

The Fitness Tracking Application leverages a combination of technologies to deliver a robust and efficient solution for health and fitness management. The backend is supported by a relational database management system (RDBMS), while the frontend features an interactive and user-friendly interface developed using JavaFX. Middleware technologies enable seamless communication between the backend and frontend.

2.1.1 Java

- **Role:** Java serves as the primary programming language for both backend logic and GUI development.
- **Usage:**
 - 1.**Backend:** Manages operations like user registration, profile management, and activity tracking.
 - 2.**Frontend:** A JavaFX-based GUI provides an intuitive and visually engaging user experience.
 - 3.**Middleware:** Java Database Connectivity (JDBC) facilitates seamless interaction with the MySQL database.
- **Advantages:**
 - Platform independence for cross-platform compatibility.
 - Built-in security features to protect sensitive user data.
 - Object-oriented programming principles ensure maintainability and scalability.

2.1.2 MySQL

- **Role:** MySQL is used as the relational database for securely storing and managing all user-related data, such as profiles, fitness logs, and medical conditions.
- **Usage:**
 - Stores user information, including personal details, activity logs, and dietary records.
 - Efficient SQL queries ensure quick data retrieval and management, supporting the application's dynamic features.

- Advantages:
 - Reliable, open-source database management system.
 - Ensures data integrity and security while handling complex queries.
 - Scalable to accommodate future features, such as wearable device integration and personalized insights.

REQUIREMENTS AND ANALYSIS

3.1 REQUIREMENT SPECIFICATION

3.1.1 Functional Requirements

- User Authentication and Authorization
- Enable secure user registration and login.
- Maintain session details for logged-in users to ensure data privacy.
- **Profile Management**
 - Allow creation and updating of user profiles with personal and fitness-related information.
 - Provide a dashboard displaying user details, such as name, email, fitness goals, and recent activities.
- **Fitness Tracking Modules**
 - Enable logging and tracking of food intake with nutritional details.
 - Allow tracking of physical activities, including exercise and daily steps.
 - Record and manage medical conditions for comprehensive health tracking.
- **Database Records Management**
 - Use unique identifiers (e.g., user ID) for managing user data.
 - Separate tables for user profiles, food logs, activity records, and medical conditions for efficient data handling and retrieval.

3.1.2 Non-Functional Requirements

- **Performance**
 - Ensure quick response times for data retrieval and display, even with a growing database.
- **Scalability**
 - Design the system to accommodate future features like wearable integration or personalized insights.
- **Security**
 - Implement encryption for sensitive data and ensure secure database connections.

3.2 HARDWARE AND SOFTWARE REQUIREMENTS

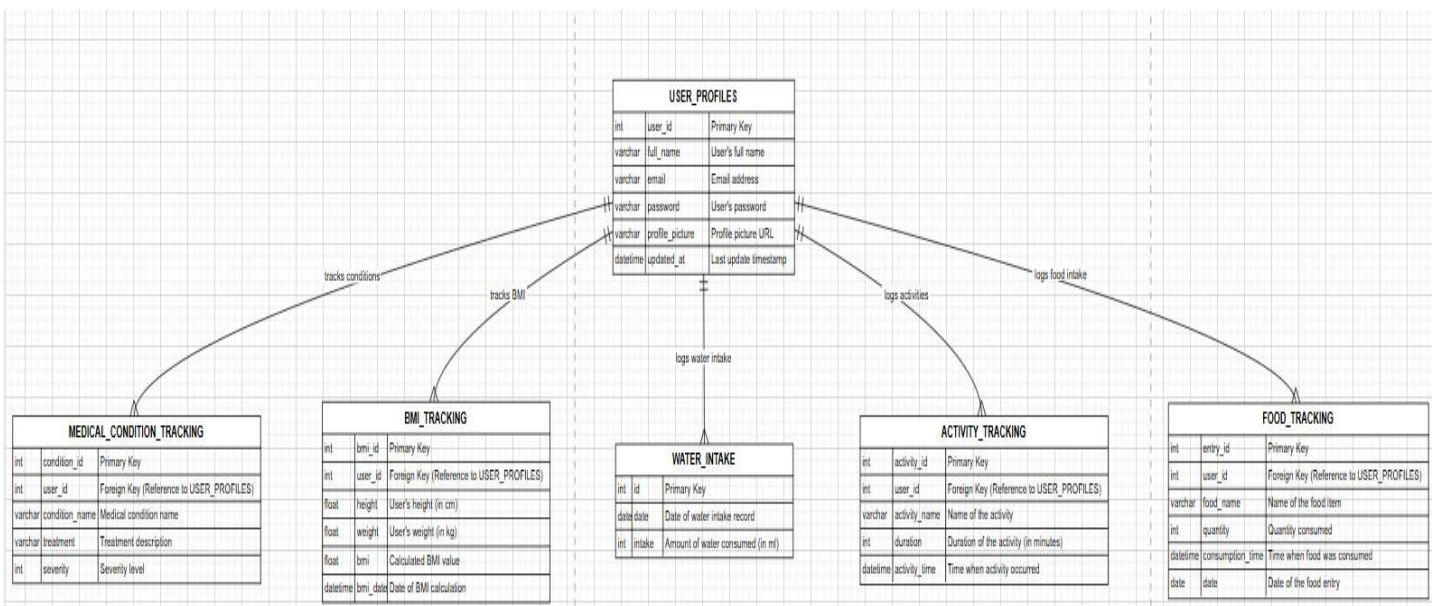
Hardware Requirements

- **Processor:** Intel Core i3 or equivalent for smooth application performance.
- **RAM:** 4 GB or higher to handle concurrent database operations and GUI rendering.
- **Storage:** At least 500 MB for application files and database storage.
- **Monitor Resolution:** 1024 x 768 or higher for optimal UI rendering.

Software Requirements

- **Operating System:** Windows 10, macOS, or higher.
- **Frontend:** JavaFX for an interactive graphical user interface.
- **Backend:** MySQL for secure and reliable database management.
- **IDE:** IntelliJ IDEA or Eclipse for development.
- **Version Control:** Git for code versioning and collaboration.

3.1 ER DIAGRAM



Chapter 4 :PROGRAM CODE

Backend Details

Tables

1. User Login

- **Columns:**

- user_id (Primary Key, Auto-increment)
- username (VARCHAR)
- password (VARCHAR, encrypted)
- email (VARCHAR)
- last_login (DATETIME)

- **Description:** Manages user authentication and tracks login details.

2. User Profile

- **Columns:**

- user_id (Foreign Key, linked to User Login)
- name (VARCHAR)
- age (INT)
- gender (VARCHAR)
- weight (FLOAT)
- height (FLOAT)
- fitness_goal (VARCHAR)

- **Description:** Stores personal details and fitness preferences for each user.

3. Food Intake Logs

- **Columns:**

- log_id (Primary Key, Auto-increment)
- user_id (Foreign Key, linked to User Login)

- date (DATE)
- food_item (VARCHAR)
- calories (FLOAT)
- protein (FLOAT)
- carbs (FLOAT)
- fats (FLOAT)
- **Description:** Logs daily food consumption with nutritional details.

4. Activity Logs

- **Columns:**
 - log_id (Primary Key, Auto-increment)
 - user_id (Foreign Key, linked to User Login)
 - activity_type (VARCHAR)
 - duration (FLOAT, in minutes)
 - calories_burned (FLOAT)
 - date (DATE)
- **Description:** Records physical activities and calculates calories burned.

5. Medical Condition Logs

- **Columns:**
 - log_id (Primary Key, Auto-increment)
 - user_id (Foreign Key, linked to User Login)
 - condition_name (VARCHAR)
 - severity (VARCHAR)
 - notes (TEXT)
 - date_recorded (DATE)
- **Description:** Tracks medical conditions and related details for users.

Admin Dashboard

1. Admin Login

- **Columns:**
 - admin_id (Primary Key, Auto-increment)
 - username (VARCHAR)
 - password (VARCHAR, encrypted)
- **Description:** Provides secure access for system administrators to oversee application data and operations.

2. System Monitoring

- **Details:**
 - View logs of user activity.
 - Monitor database usage and optimize queries for better performance.

Data Relationships

- **User Login ↔ User Profile:** One-to-One relationship (Each user has one profile).
- **User Login ↔ Food Intake Logs, Activity Logs, Medical Condition Logs:** One-to-Many relationship (A user can have multiple entries in these logs).

This backend structure ensures efficient and secure handling of user data while enabling seamless interactions between different application modules.

3.1 MAIN APPLICATION : [MainApp.java]

```
import javafx.application.Application;
import javafx.scene.Scene;
import javafx.scene.control.Button;
import javafx.scene.layout.Background;
import javafx.scene.layout.StackPane;
import javafx.scene.layout.VBox;
import javafx.geometry.Pos;
import javafx.stage.Stage;
import javafx.scene.image.Image;
import javafx.scene.layout.BackgroundImage;

public class MainApp extends Application {

    @Override
    public void start(Stage primaryStage) {
        showMainWindow(primaryStage);
    }

    private void showMainWindow(Stage primaryStage) {
        Button backButton = new Button("Back");
        backButton.setStyle("-fx-font-family: 'Arial'; -fx-font-size: 14px; -fx-font-weight: bold; -fx-text-fill: white; -fx-background-color: #333;");
        backButton.setOnAction(e -> showLoginPage(primaryStage));

        Button foodTrackingButton = new Button("Food Tracking");
        Button activityTrackingButton = new Button("Activity Tracking");
        Button medicalConditionTrackingButton = new Button("Medical Condition Tracking");
        Button waterTrackingButton = new Button("Water Tracking");
        Button bmiScaleButton = new Button("BMI Scale");
        Button userProfileButton = new Button("User Profile");

        foodTrackingButton.setStyle("-fx-font-family: 'Arial'; -fx-font-size: 14px;");
        activityTrackingButton.setStyle("-fx-font-family: 'Arial'; -fx-font-size: 14px;");
        medicalConditionTrackingButton.setStyle("-fx-font-family: 'Arial'; -fx-font-size: 14px;");
        waterTrackingButton.setStyle("-fx-font-family: 'Arial'; -fx-font-size: 14px;");
        bmiScaleButton.setStyle("-fx-font-family: 'Arial'; -fx-font-size: 14px;");
        userProfileButton.setStyle("-fx-font-family: 'Arial'; -fx-font-size: 14px;");

        // Set up button actions to navigate to corresponding pages
        foodTrackingButton.setOnAction(e -> openFoodTrackingPage());
        activityTrackingButton.setOnAction(e -> openActivityTrackingPage());
        medicalConditionTrackingButton.setOnAction(e -> openMedicalConditionPage());
    }
}
```

```

waterTrackingButton.setOnAction(e -> openWaterTrackingPage());
bmiScaleButton.setOnAction(e -> openBmiScalePage());
userProfileButton.setOnAction(e -> openUserProfilePage());

VBox layout = new VBox(20);
layout.getChildren().addAll(
    backButton,
    foodTrackingButton,
    activityTrackingButton,
    medicalConditionTrackingButton,
    waterTrackingButton,
    bmiScaleButton,
    userProfileButton
);
layout.setAlignment(Pos.CENTER);

// Set background and scene without using additional properties
StackPane root = new StackPane();
Image image = new Image("file:/C:/XboxGames/image.jpg"); // Update the file
path as needed
BackgroundImage backgroundImage = new BackgroundImage(image, null, null,
null, null);
Background background = new Background(backgroundImage);
root.setBackground(background);
root.getChildren().add(layout);

Scene scene = new Scene(root, 600, 400);
primaryStage.setTitle("Personal Fitness Tracker");
primaryStage.setScene(scene);
primaryStage.show();
}

private void openFoodTrackingPage() {
    // Open Food Tracking page logic here
    FoodTrackingPage foodTrackingPage = new FoodTrackingPage();
    Stage foodTrackingStage = new Stage();
    foodTrackingPage.start(foodTrackingStage);
}

private void openActivityTrackingPage() {
    // Open Activity Tracking page logic here
    ActivityTrackingPage activityTrackingPage = new ActivityTrackingPage();
    Stage activityTrackingStage = new Stage();
    activityTrackingPage.start(activityTrackingStage);
}

private void openMedicalConditionPage() {
    // Open Medical Condition page logic here
    MedicalConditionTrackingPage medicalConditionPage = new
MedicalConditionTrackingPage();
    Stage medicalConditionStage = new Stage();
    medicalConditionPage.start(medicalConditionStage);
}

```



```
}

private void openWaterTrackingPage() {
    // Open Water Tracking page logic here
    WaterTrackingPage waterTrackingPage = new WaterTrackingPage();
    Stage waterTrackingStage = new Stage();
    waterTrackingPage.start(waterTrackingStage);
}

private void openBmiScalePage() {
    // Open BMI Scale page logic here
    BmiScalePage bmiScalePage = new BmiScalePage();
    Stage bmiScaleStage = new Stage();
    bmiScalePage.start(bmiScaleStage);
}

private void openUserProfilePage() {
    // Open User Profile page logic here
    UserProfilePage userProfilePage = new UserProfilePage();
    Stage userProfileStage = new Stage();
    userProfilePage.start(userProfileStage);
}

private void showLoginPage(Stage primaryStage) {
    // Logic to show login page
}

public static void main(String[] args) {
    launch(args);
}
}
```

3.2 USER PROFILE DESIGN :

```
import javafx.application.Application;
import javafx.geometry.Pos;
import javafx.scene.Scene;
import javafx.scene.control.*;
import javafx.scene.layout.*;
import javafx.scene.text.Font;
import javafx.stage.Stage;
import java.sql.*;

public class UserProfilePage extends Application {

    @Override
    public void start(Stage stage) {
        // Title for the page
        Label titleLabel = new Label("User Profile");
        titleLabel.setFont(new Font("Arial", 24));

        // User information fields
        Label nameLabel = new Label("Full Name:");
        TextField nameField = new TextField();

        Label emailLabel = new Label("Email Address:");
        TextField emailField = new TextField();

        Label passwordLabel = new Label("Password:");
        PasswordField passwordField = new PasswordField();

        // Profile picture (for simplicity, use a placeholder)
        Label pictureLabel = new Label("Profile Picture:");
        Button changePicButton = new Button("Change Picture");

        // Button to save updated profile
        Button saveButton = new Button("Save Profile");
        saveButton.setStyle("-fx-font-family: 'Arial'; -fx-font-size: 14px;");

        // Label to show save status
        Label statusLabel = new Label();
        statusLabel.setStyle("-fx-font-size: 16px; -fx-font-weight: bold;");

        // Action for the "Save Profile" button
        saveButton.setOnAction(e -> {
            String name = nameField.getText();
            String email = emailField.getText();
            String password = passwordField.getText();

            // Insert or update user profile in the database
            if (!name.isEmpty() && !email.isEmpty() && !password.isEmpty()) {
                saveUserProfile(name, email, password);
                statusLabel.setText("Profile updated successfully!");
            } else {

```

```

        statusLabel.setText("Please fill in all fields.");
    }
});

// Back button to go to the previous page
Button backButton = new Button("Back");
backButton.setStyle("-fx-font-family: 'Arial'; -fx-font-size: 14px;");
backButton.setOnAction(e -> stage.close()); // Close the current page

// Layout for the page
VBox layout = new VBox(15);
layout.setAlignment(Pos.CENTER);
layout.getChildren().addAll(
    titleLabel,
    nameLabel,
    nameField,
    emailLabel,
    emailField,
    passwordLabel,
    passwordField,
    pictureLabel,
    changePicButton,
    saveButton,
    statusLabel,
    backButton
);

// Set up the scene
Scene scene = new Scene(layout, 400, 350);
stage.setTitle("User Profile");
stage.setScene(scene);
stage.show();
}

private void saveUserProfile(String name, String email, String password) {
    String url = "jdbc:mysql://localhost:3306/fitness_tracker";
    String user = "root";
    String pass = "manu";

    try (Connection conn = DriverManager.getConnection(url, user, pass)) {
        String query = "INSERT INTO user_profiles (full_name, email, password)
VALUES (?, ?, ?)";
        try (PreparedStatement stmt = conn.prepareStatement(query)) {
            stmt.setString(1, name);
            stmt.setString(2, email);
            stmt.setString(3, password); // You should hash the password in a
real app

            stmt.executeUpdate();
        }
    } catch (SQLException ex) {
        ex.printStackTrace();
    }
}
}

```

```

    public static void main(String[] args) {
        launch(args);
    }
}

```

3.3 ACTIVITY TRACKING PAGE DESIGN :

```

import javafx.application.Application;
import javafx.geometry.Pos;
import javafx.scene.Scene;
import javafx.scene.control.Button;
import javafx.scene.layout.VBox;
import javafx.stage.Stage;
import javafx.scene.control.Alert;
import javafx.scene.control.Alert.AlertType;
import javafx.scene.control.TextField;
import javafx.scene.control.Label;
import java.sql.*;

public class ActivityTrackingPage extends Application {

    @Override
    public void start(Stage stage) {
        // Title or text for the page title
        Button backButton = new Button("Back to Main");
        backButton.setStyle("-fx-font-family: 'Arial'; -fx-font-size: 14px; -fx-font-weight: bold; -fx-text-fill: white; -fx-background-color: #333;");
        backButton.setOnAction(e -> stage.close()); // This closes the current page, you may want to return to the main window

        // Buttons for activity tracking features
        Button trackActivityButton = new Button("Track New Activity");
        trackActivityButton.setStyle("-fx-font-family: 'Arial'; -fx-font-size: 14px;");
        trackActivityButton.setOnAction(e -> openTrackActivityWindow());

        Button activityHistoryButton = new Button("View Activity History");
        activityHistoryButton.setStyle("-fx-font-family: 'Arial'; -fx-font-size: 14px;");

        // Layout for the page
        VBox layout = new VBox(20);
        layout.setAlignment(Pos.CENTER);
        layout.getChildren().addAll(backButton, trackActivityButton, activityHistoryButton);

        // Set the scene for the page
        Scene scene = new Scene(layout, 400, 300);
    }
}

```

```

        stage.setTitle("Activity Tracking");
        stage.setScene(scene);
        stage.show();
    }

    private void openTrackActivityWindow() {
        // Open a new window to track a new activity
        Stage stage = new Stage();
        VBox layout = new VBox(15);

        // Inputs for activity type, duration, distance
        Label activityLabel = new Label("Enter activity type (e.g., Running):");
        TextField activityField = new TextField();

        Label durationLabel = new Label("Enter duration (minutes):");
        TextField durationField = new TextField();

        Label distanceLabel = new Label("Enter distance (km):");
        TextField distanceField = new TextField();

        Button saveButton = new Button("Save Activity");
        saveButton.setOnAction(e -> {
            String activityType = activityField.getText();
            String duration = durationField.getText();
            String distance = distanceField.getText();

            if (!activityType.isEmpty() && !duration.isEmpty() &&
!distance.isEmpty()) {
                saveActivity(activityType, Integer.parseInt(duration),
Double.parseDouble(distance));
                Alert alert = new Alert(AlertType.INFORMATION);
                alert.setTitle("Success");
                alert.setHeaderText(null);
                alert.setContentText("Activity saved successfully!");
                alert.showAndWait();
            } else {
                Alert alert = new Alert(AlertType.ERROR);
                alert.setTitle("Error");
                alert.setHeaderText(null);
                alert.setContentText("Please fill in all fields.");
                alert.showAndWait();
            }
        });

        layout.getChildren().addAll(activityLabel, activityField, durationLabel,
durationField, distanceLabel, distanceField, saveButton);
        Scene scene = new Scene(layout, 300, 250);
        stage.setTitle("Track New Activity");
        stage.setScene(scene);
        stage.show();
    }

    private void saveActivity(String activityType, int duration, double distance) {

```

```

        // Database connection details
        String url = "jdbc:mysql://localhost:3306/fitness_tracker"; // Replace with
your actual database name
        String user = "root"; // Replace with your MySQL username
        String password = "manu"; // Replace with your MySQL password

        String query = "INSERT INTO activity_tracking (user_id, activity_type,
duration, distance, activity_date) VALUES (?, ?, ?, ?, NOW())";

        try (Connection conn = DriverManager.getConnection(url, user, password);
            PreparedStatement stmt = conn.prepareStatement(query)) {

            // Assuming a hardcoded user_id for now (you'll likely get this from the
logged-in user)
            int userId = 1; // You can replace this with the actual user ID
dynamically

            stmt.setInt(1, userId);
            stmt.setString(2, activityType);
            stmt.setInt(3, duration);
            stmt.setDouble(4, distance);

            stmt.executeUpdate();
        } catch (SQLException ex) {
            ex.printStackTrace();
        }
    }

    public static void main(String[] args) {
        launch(args);
    }
}

```

3.4 BMI SCALE PAGE DESIGN

```

import javafx.application.Application;
import javafx.geometry.Pos;
import javafx.scene.Scene;
import javafx.scene.control.*;
import javafx.scene.layout.*;
import javafx.scene.text.Font;
import javafx.stage.Stage;

import java.sql.Connection;
import java.sql.DriverManager;
import java.sql.PreparedStatement;
import java.sql.SQLException;

```

```

public class BmiScalePage extends Application {

    @Override
    public void start(Stage stage) {
        // Title for the page
        Label titleLabel = new Label("BMI Scale");
        titleLabel.setFont(new Font("Arial", 24));

        // Input fields for height and weight
        Label heightLabel = new Label("Enter your height (in meters):");
        TextField heightField = new TextField();

        Label weightLabel = new Label("Enter your weight (in kilograms):");
        TextField weightField = new TextField();

        // Button to calculate BMI
        Button calculateButton = new Button("Calculate BMI");
        calculateButton.setStyle("-fx-font-family: 'Arial'; -fx-font-size:
14px;");

        // Label to display the result
        Label resultLabel = new Label("Your BMI will be displayed here.");
        resultLabel.setStyle("-fx-font-size: 16px; -fx-font-weight: bold;");

        // Calculate BMI when the button is clicked
        calculateButton.setOnAction(e -> {
            try {
                double height = Double.parseDouble(heightField.getText());
                double weight = Double.parseDouble(weightField.getText());
                if (height > 0 && weight > 0) {
                    double bmi = weight / (height * height);
                    resultLabel.setText(String.format("Your BMI is: %.2f", bmi));
                    saveBmiToDatabase(height, weight, bmi); // Save the result to
the database
                } else {
                    resultLabel.setText("Please enter valid positive values.");
                }
            } catch (NumberFormatException ex) {
                resultLabel.setText("Please enter valid numbers.");
            }
        });

        // Back button to go to the previous page
        Button backButton = new Button("Back");
        backButton.setStyle("-fx-font-family: 'Arial'; -fx-font-size: 14px;");
        backButton.setOnAction(e -> stage.close()); // Close the current page,
modify this as per your navigation logic

        // Layout for the page
        VBox layout = new VBox(15);
        layout.setAlignment(Pos.CENTER);
        layout.getChildren().addAll(
            titleLabel,

```

```

        heightLabel,
        heightField,
        weightLabel,
        weightField,
        calculateButton,
        resultLabel,
        backButton
    );

    // Set up the scene
    Scene scene = new Scene(layout, 400, 300);
    stage.setTitle("BMI Scale");
    stage.setScene(scene);
    stage.show();
}

private void saveBmiToDatabase(double height, double weight, double bmi) {
    String url = "jdbc:mysql://localhost:3306/fitness_tracker"; // Replace
with your actual database name
    String user = "root"; // Replace with your MySQL username
    String password = "manu"; // Replace with your MySQL password

    String query = "INSERT INTO bmi_tracking (user_id, height, weight, bmi,
bmi_date) VALUES (?, ?, ?, ?, NOW())";

    try (Connection conn = DriverManager.getConnection(url, user, password);
        PreparedStatement stmt = conn.prepareStatement(query)) {

        // Assuming a hardcoded user_id for now (you'll likely get this from
the logged-in user)
        int userId = 1; // Replace with actual user ID dynamically

        stmt.setInt(1, userId);
        stmt.setDouble(2, height);
        stmt.setDouble(3, weight);
        stmt.setDouble(4, bmi);

        stmt.executeUpdate();
    } catch (SQLException ex) {
        ex.printStackTrace();
    }
}

public static void main(String[] args) {
    launch(args);
}
}

```


3.5 FOOD TRACKING PAGE DESIGN

```
// Source code is decompiled from a .class file using FernFlower decompiler.
import java.sql.Connection;
import java.sql.DriverManager;
import java.sql.PreparedStatement;
import java.sql.SQLException;
import javafx.application.Application;
import javafx.geometry.Pos;
import javafx.scene.Node;
import javafx.scene.Scene;
import javafx.scene.control.Button;
import javafx.scene.control.Label;
import javafx.scene.control.TextField;
import javafx.scene.layout.VBox;
import javafx.scene.text.Font;
import javafx.stage.Stage;

public class FoodTrackingPage extends Application {
    public FoodTrackingPage() {

    }

    public void start(Stage stage) {
        Label titleLabel = new Label("Food Tracking");
        titleLabel.setFont(new Font("Arial", 24.0));
        Label foodLabel = new Label("Enter food item:");
        TextField foodField = new TextField();
        Label quantityLabel = new Label("Enter quantity (grams):");
        TextField quantityField = new TextField();
        Label timeLabel = new Label("Enter time of consumption:");
        TextField timeField = new TextField();
        Button logFoodButton = new Button("Log Food");
        logFoodButton.setStyle("-fx-font-family: 'Arial'; -fx-font-size: 14px;");
        Label resultLabel = new Label("Food entry will be displayed here.");
        resultLabel.setStyle("-fx-font-size: 16px; -fx-font-weight: bold;");
        logFoodButton.setOnAction((e) -> {
            String foodItem = foodField.getText();
            String quantity = quantityField.getText();
            String time = timeField.getText();
            if (!foodItem.isEmpty() && !quantity.isEmpty() && !time.isEmpty()) {
                this.saveFoodEntry(foodItem, Integer.parseInt(quantity), time);
                resultLabel.setText("Logged: " + foodItem + " (" + quantity + "g) at " + time);
            } else {
                resultLabel.setText("Please fill in all fields.");
            }
        });

        Button backButton = new Button("Back");
        backButton.setStyle("-fx-font-family: 'Arial'; -fx-font-size: 14px;");
        backButton.setOnAction((e) -> {
            stage.close();
        });
    }
}
```

```

VBox layout = new VBox(15.0);
layout.setAlignment(Pos.CENTER);
layout.getChildren().addAll(new Node[]{titleLabel, foodLabel, foodField, quantityLabel,
quantityField, timeLabel, timeField, logFoodButton, resultLabel, backButton});
Scene scene = new Scene(layout, 400.0, 300.0);
stage.setTitle("Food Tracking");
stage.setScene(scene);
stage.show();
}

private void saveFoodEntry(String foodItem, int quantity, String time) {
    String url = "jdbc:mysql://localhost:3306/fitness_tracker";
    String user = "root";
    String password = "manu";
    String query = "INSERT INTO food_tracking (user_id, food_name, quantity,
consumption_time, date) VALUES (?, ?, ?, ?, CURDATE())";

    try {
        Throwable var8 = null;
        Object var9 = null;

        try {
            Connection conn = DriverManager.getConnection(url, user, password);

            try {
                PreparedStatement stmt = conn.prepareStatement(query);

                try {
                    int userId = 1;
                    stmt.setInt(1, userId);
                    stmt.setString(2, foodItem);
                    stmt.setInt(3, quantity);
                    stmt.setString(4, time);
                    stmt.executeUpdate();
                } finally {
                    if (stmt != null) {
                        stmt.close();
                    }
                }
            } catch (Throwable var26) {
                if (var8 == null) {
                    var8 = var26;
                } else if (var8 != var26) {
                    var8.addSuppressed(var26);
                }

                if (conn != null) {
                    conn.close();
                }

                throw var8;
            }
        }
    }
}

```

```

        if (conn != null) {
            conn.close();
        }
    } catch (Throwable var27) {
        if (var8 == null) {
            var8 = var27;
        } else if (var8 != var27) {
            var8.addSuppressed(var27);
        }

        throw var8;
    }
} catch (SQLException var28) {
    var28.printStackTrace();
}

}

public static void main(String[] args) {
    launch(args);
}
}

```

3.6 MEDICAL TRACKING PAGE DESIGN

```

import javafx.application.Application;
import javafx.geometry.Pos;
import javafx.scene.Scene;
import javafx.scene.control.*;
import javafx.scene.layout.*;
import javafx.scene.text.Font;
import javafx.stage.Stage;
import java.sql.*;

public class MedicalConditionTrackingPage extends Application {

    @Override
    public void start(Stage stage) {
        // Title for the page
        Label titleLabel = new Label("Medical Condition Tracking");
        titleLabel.setFont(new Font("Arial", 24));

        // Input fields for condition name, severity, and medications
        Label conditionLabel = new Label("Enter medical condition:");
        TextField conditionField = new TextField();

        Label severityLabel = new Label("Enter severity (1-10):");
        TextField severityField = new TextField();
    }
}

```

```

Label medicationLabel = new Label("Enter medication (optional):");
TextField medicationField = new TextField();

// Button to log condition
Button logConditionButton = new Button("Log Condition");
logConditionButton.setStyle("-fx-font-family: 'Arial'; -fx-font-size: 14px;");

// Label to display the result
Label resultLabel = new Label("Condition entry will be displayed here.");
resultLabel.setStyle("-fx-font-size: 16px; -fx-font-weight: bold;");

// Action for the "Log Condition" button
logConditionButton.setOnAction(e -> {
    String condition = conditionField.getText();
    String severity = severityField.getText();
    String medication = medicationField.getText();

    if (!condition.isEmpty() && !severity.isEmpty()) {
        // Save the condition to the database
        logConditionToDatabase(condition, severity, medication);
        resultLabel.setText("Logged: " + condition + " (Severity: " + severity +
        ")");

        if (!medication.isEmpty()) {
            resultLabel.setText(resultLabel.getText() + " | Medication: " +
medication);
        }
    } else {
        resultLabel.setText("Please fill in all fields.");
    }
});

// Back button to go to the previous page
Button backButton = new Button("Back");
backButton.setStyle("-fx-font-family: 'Arial'; -fx-font-size: 14px;");
backButton.setOnAction(e -> stage.close()); // Close the current page

// Layout for the page
VBox layout = new VBox(15);
layout.setAlignment(Pos.CENTER);
layout.getChildren().addAll(
    titleLabel,
    conditionLabel,
    conditionField,
    severityLabel,
    severityField,
    medicationLabel,
    medicationField,
    logConditionButton,
    resultLabel,
    backButton
);

```

```

        // Set up the scene
        Scene scene = new Scene(layout, 400, 300);
        stage.setTitle("Medical Condition Tracking");
        stage.setScene(scene);
        stage.show();
    }

    // Method to log the medical condition into the MySQL database
    private void logConditionToDatabase(String condition, String severity, String medication)
    {
        String url = "jdbc:mysql://localhost:3306/fitness_tracker";
        String user = "root";
        String pass = "manu";

        try (Connection conn = DriverManager.getConnection(url, user, pass)) {
            String query = "INSERT INTO medical_condition_tracking (condition_name,user_id,severity, treatment) VALUES (?,?, ?, ?)";
            try (PreparedStatement stmt = conn.prepareStatement(query)) {
                stmt.setString(1, condition);
                stmt.setInt(2,1);
                stmt.setInt(3, Integer.parseInt(severity));
                stmt.setString(4, medication.isEmpty() ? null : medication);
                stmt.executeUpdate();
            }
        } catch (SQLException ex) {
            ex.printStackTrace();
        }
    }

    public static void main(String[] args) {
        launch(args);
    }
}

```

3.7 WATER TRACKING PAGE DESIGN

```

import javafx.application.Application;
import javafx.collections.FXCollections;
import javafx.collections.ObservableList;
import javafx.geometry.Pos;
import javafx.scene.Scene;
import javafx.scene.control.*;
import javafx.scene.layout.*;
import javafx.scene.text.Font;
import javafx.stage.Stage;
import java.sql.*;

public class WaterTrackingPage extends Application {

    private ObservableList<WaterIntakeItem> data = FXCollections.observableArrayList();

```

```

@Override
public void start(Stage stage) {
    // Title for the page
    Label titleLabel = new Label("Water Tracking");
    titleLabel.setFont(new Font("Arial", 24));

    // Water intake entry field
    Label intakeLabel = new Label("Enter Water Intake (ml):");
    TextField intakeField = new TextField();

    // Button to log water intake
    Button logButton = new Button("Log Water");
    logButton.setStyle("-fx-font-family: 'Arial'; -fx-font-size: 14px;");

    // Label to show intake status
    Label statusLabel = new Label();
    statusLabel.setStyle("-fx-font-size: 16px; -fx-font-weight: bold;");

    // Table for displaying water tracking history
    TableView<WaterIntakeItem> table = new TableView<>();
    TableColumn<WaterIntakeItem, String> dateColumn = new TableColumn<>("Date");
    TableColumn<WaterIntakeItem, String> intakeColumn = new TableColumn<>("Intake (ml)");

    // Adding columns to the table
    table.getColumns().add(dateColumn);
    table.getColumns().add(intakeColumn);

    // Load data from MySQL
    loadWaterData();

    // Setting items to the table
    table.setItems(data);

    // Action for the "Log Water" button
    logButton.setOnAction(e -> {
        String intake = intakeField.getText();
        if (!intake.isEmpty() && isNumeric(intake)) {
            // Save the water intake to the database
            logWaterIntake(intake);
            statusLabel.setText("Water logged: " + intake + " ml");
            loadWaterData(); // Reload data from the database
        } else {
            statusLabel.setText("Please enter a valid number.");
        }
    });

    // Back button to go to the previous page
    Button backButton = new Button("Back");
    backButton.setStyle("-fx-font-family: 'Arial'; -fx-font-size: 14px;");
    backButton.setOnAction(e -> stage.close()); // Close the current page

    // Layout for the page

```

```

VBox layout = new VBox(15);
layout.setAlignment(Pos.CENTER);
layout.getChildren().addAll(
    titleLabel,
    intakeLabel,
    intakeField,
    logButton,
    statusLabel,
    table,
    backButton
);

// Set up the scene
Scene scene = new Scene(layout, 500, 400);
stage.setTitle("Water Tracking");
stage.setScene(scene);
stage.show();
}

// Helper method to check if the input is a number
private boolean isNumeric(String str) {
    try {
        Integer.parseInt(str);
        return true;
    } catch (NumberFormatException e) {
        return false;
    }
}

// Method to load water intake data from the database
private void loadWaterData() {
    data.clear(); // Clear the existing data
    String url = "jdbc:mysql://localhost:3306/your_database";
    String user = "root";
    String pass = "your_password";

    try (Connection conn = DriverManager.getConnection(url, user, pass)) {
        String query = "SELECT date, intake FROM water_intake ORDER BY date DESC";
        try (Statement stmt = conn.createStatement(); ResultSet rs =
stmt.executeQuery(query)) {
            while (rs.next()) {
                String date = rs.getString("date");
                String intake = rs.getString("intake");
                data.add(new WaterIntakeItem(date, intake));
            }
        }
    } catch (SQLException ex) {
        ex.printStackTrace();
    }
}

// Method to log water intake into the database
private void logWaterIntake(String intake) {

```

```

String url = "jdbc:mysql://localhost:3306/fitness_tracker";
String user = "root";
String pass = "manu";

try (Connection conn = DriverManager.getConnection(url, user, pass)) {
    String query = "INSERT INTO water_intake (date, intake) VALUES (CURDATE(), ?)";
    try (PreparedStatement stmt = conn.prepareStatement(query)) {
        stmt.setInt(1, Integer.parseInt(intake));
        stmt.executeUpdate();
    }
} catch (SQLException ex) {
    ex.printStackTrace();
}

public static void main(String[] args) {
    launch(args);
}

// Class for representing each water intake item
public static class WaterIntakeItem {
    private String date;
    private String intake;

    public WaterIntakeItem(String date, String intake) {
        this.date = date;
        this.intake = intake;
    }

    public String getDate() {
        return date;
    }

    public String getIntake() {
        return intake;
    }
}
}

```

3.8 DATABASE CONNECTIVITY

```

import java.sql.Connection;
import java.sql.DriverManager;
import java.sql.SQLException;

public class DatabaseConnection {
    private static final String URL = "jdbc:mysql://localhost:3306/fitness_tracker"; //
    Database URL

```



```

private static final String USER = "root"; // MySQL username
private static final String PASSWORD = "manu"; // MySQL password

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

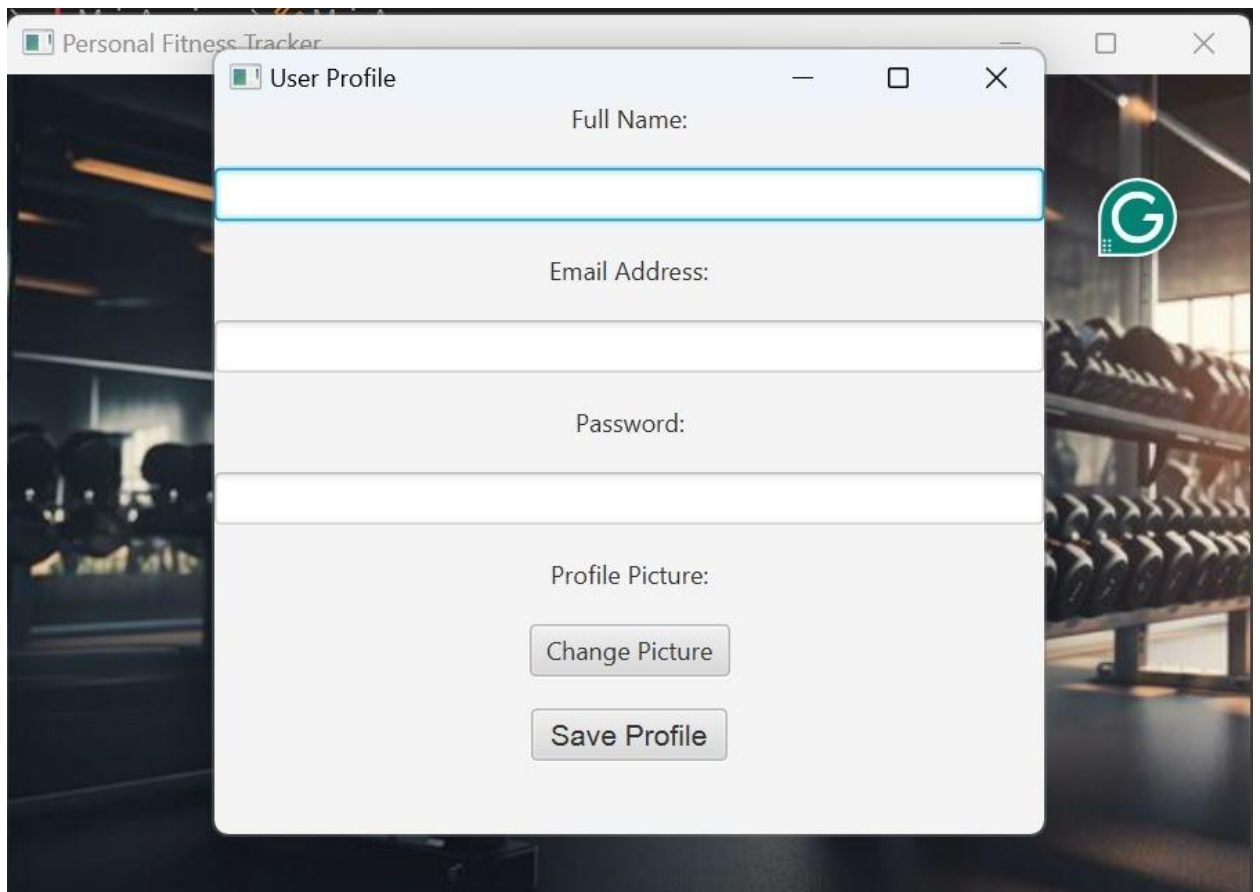
// Main method - Entry point of the application
public static void main(String[] args) {
    // Attempt to establish a connection
    Connection connection = connect();

    // Check if the connection was successful
    if (connection != null) {
        System.out.println("Connection to the database established successfully!");
        // Close the connection after use (optional for this demo, but always good
practice)
        try {
            connection.close();
        } catch (SQLException e) {
            System.out.println("Error closing the connection: " + e.getMessage());
        }
    } else {
        System.out.println("Failed to connect to the database.");
    }
}
}

```

OUTPUTS

4.1 LOGIN PAGE:

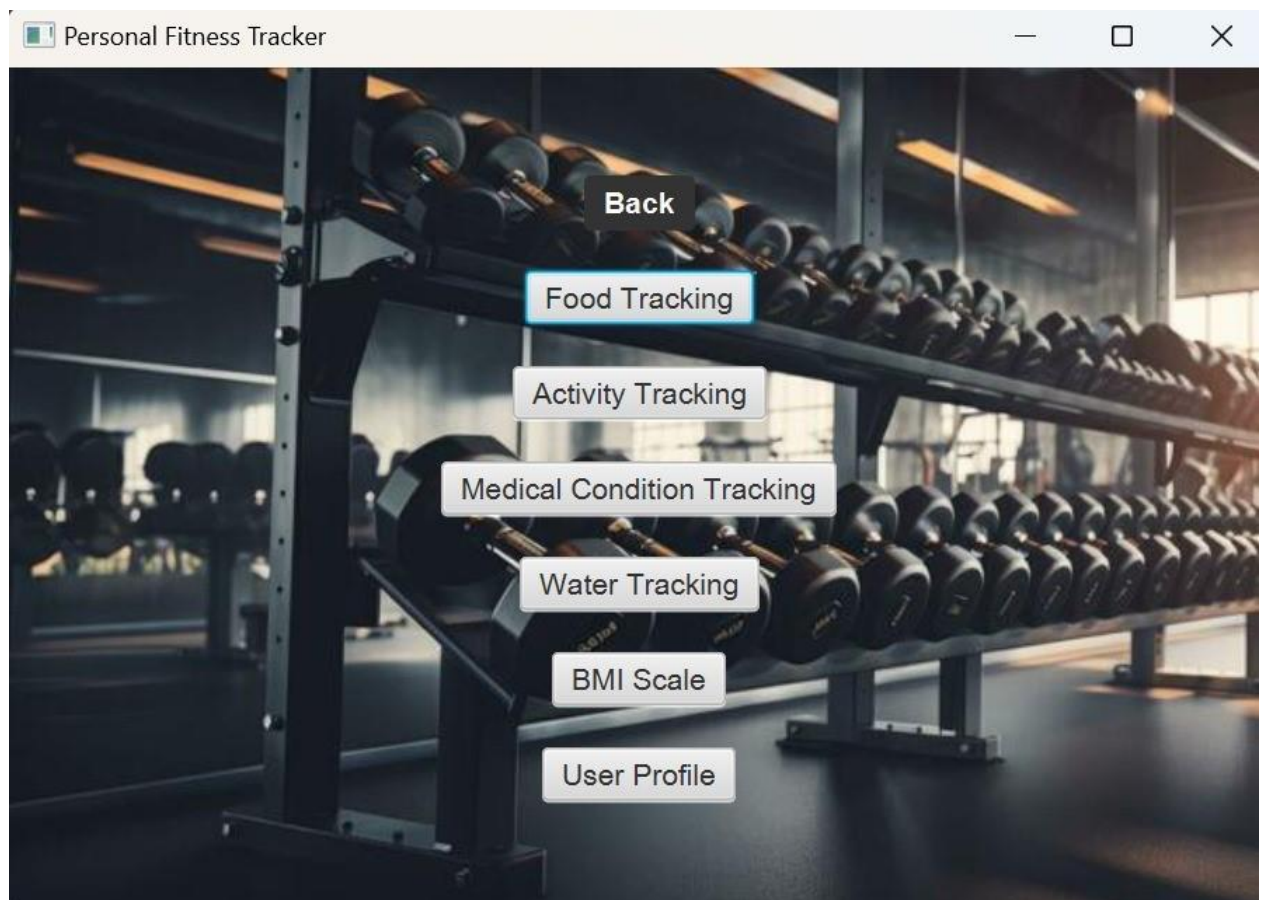


The screenshot displays a desktop application window titled "Personal Fitness Tracker". Overlaid on this is a "User Profile" dialog box. The dialog box contains the following fields and controls:

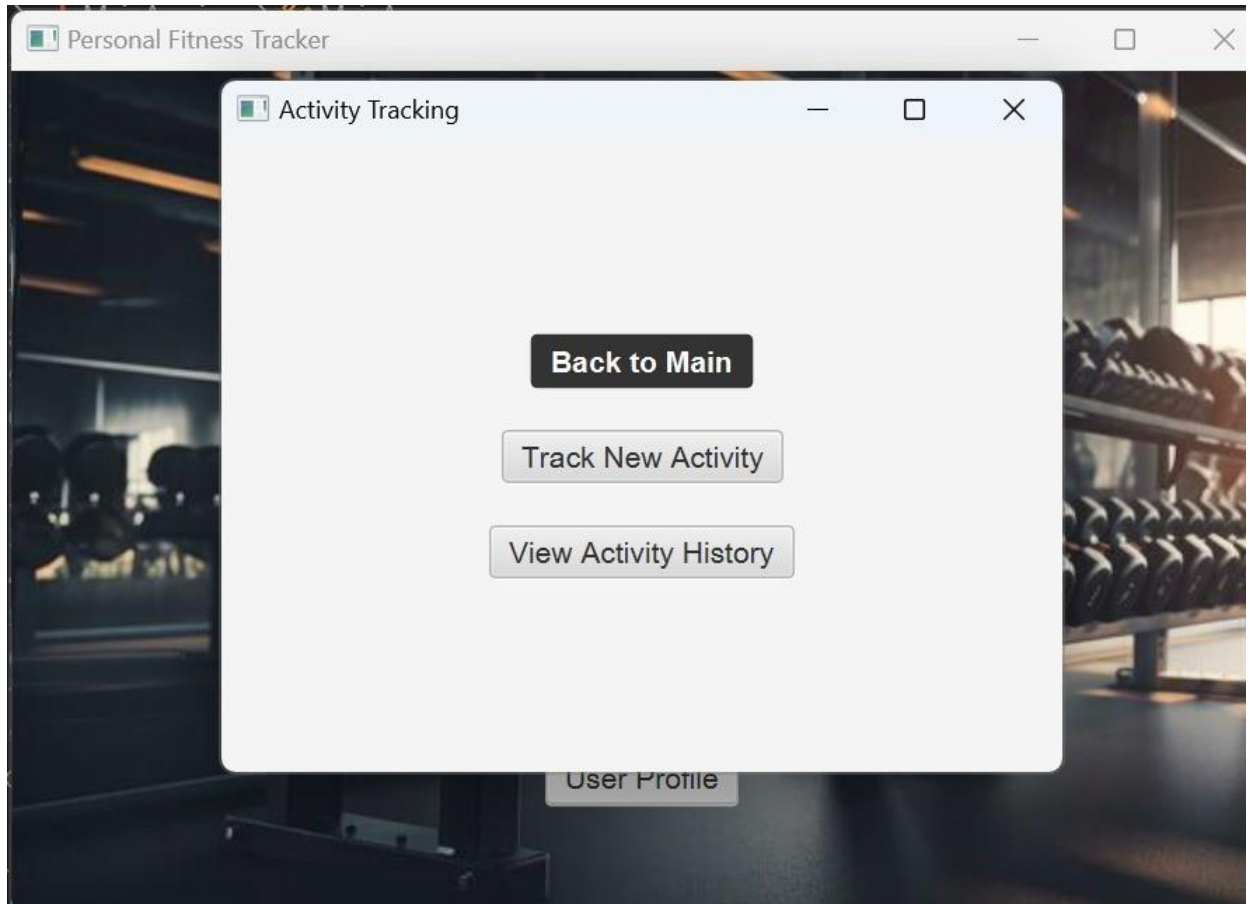
- Full Name:** A text input field with a blue border.
- Email Address:** A text input field.
- Password:** A text input field.
- Profile Picture:** A section containing a "Change Picture" button and a "Save Profile" button.

A green circular logo with a white 'G' is visible in the top right corner of the application window. The background of the application window shows a gym interior with dumbbells on a rack.

4.2 HOME PAGE:



4.3 ACTIVITY TRACKING PAGE:



4.4 WATER TRACKING PAGE:

Pers

Water Tracking

—

□

×

Water Tracking

Enter Water Intake (ml):

Log Water

Date	Intake (ml)
No content in table	

Back

4.5 FOOD TRACKING PAGE:

Personal Fitness Tracker

Food Tracking

Enter food item:

Enter quantity (grams):

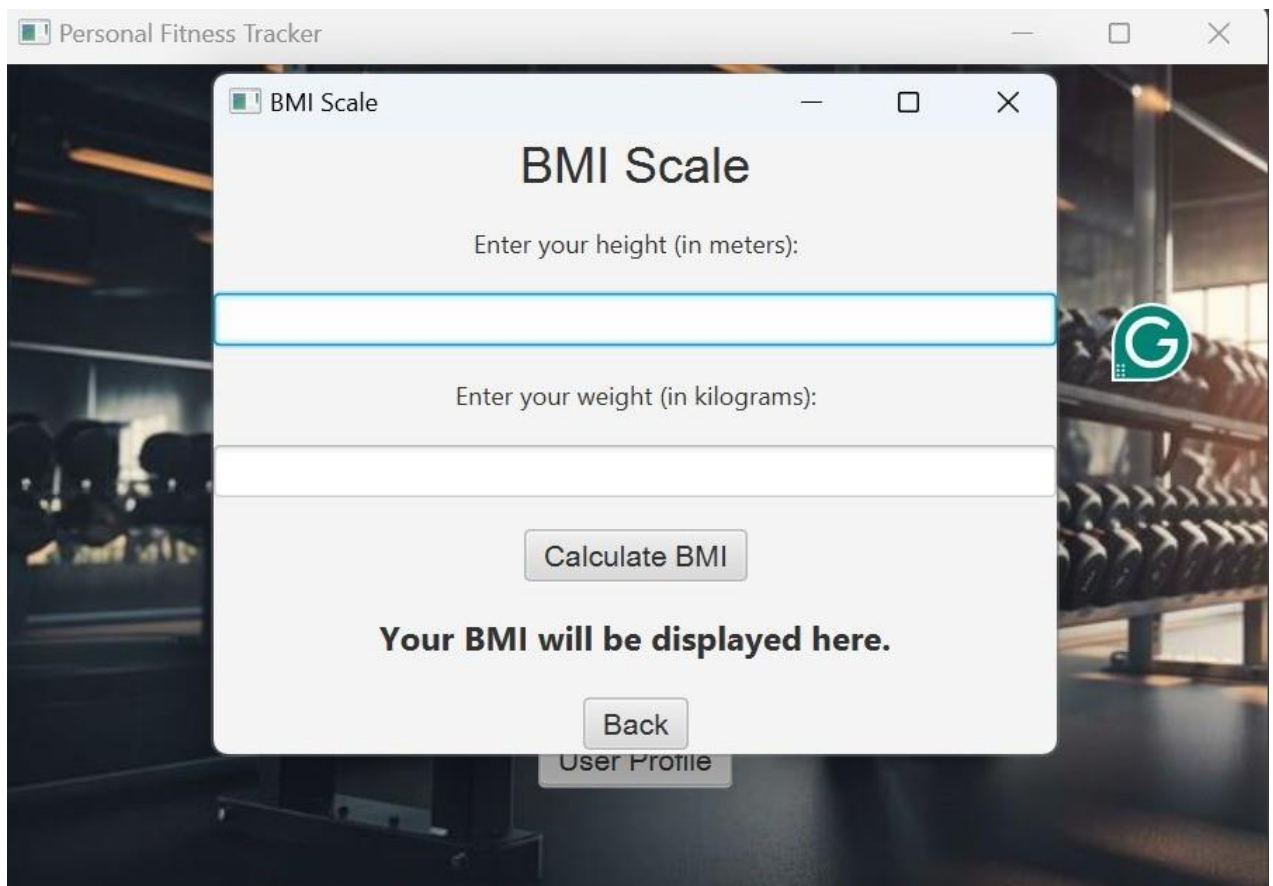
Enter time of consumption:

Log Food

Food entry will be displayed here.

User Profile

4.6 BMI SCALE PAGE:



The screenshot shows a web application window titled "Personal Fitness Tracker". Inside, there is a modal window titled "BMI Scale". The modal has a title bar with a green icon, a minus sign, a square icon, and a close button. The main content of the modal is as follows:

BMI Scale

Enter your height (in meters):

Enter your weight (in kilograms):

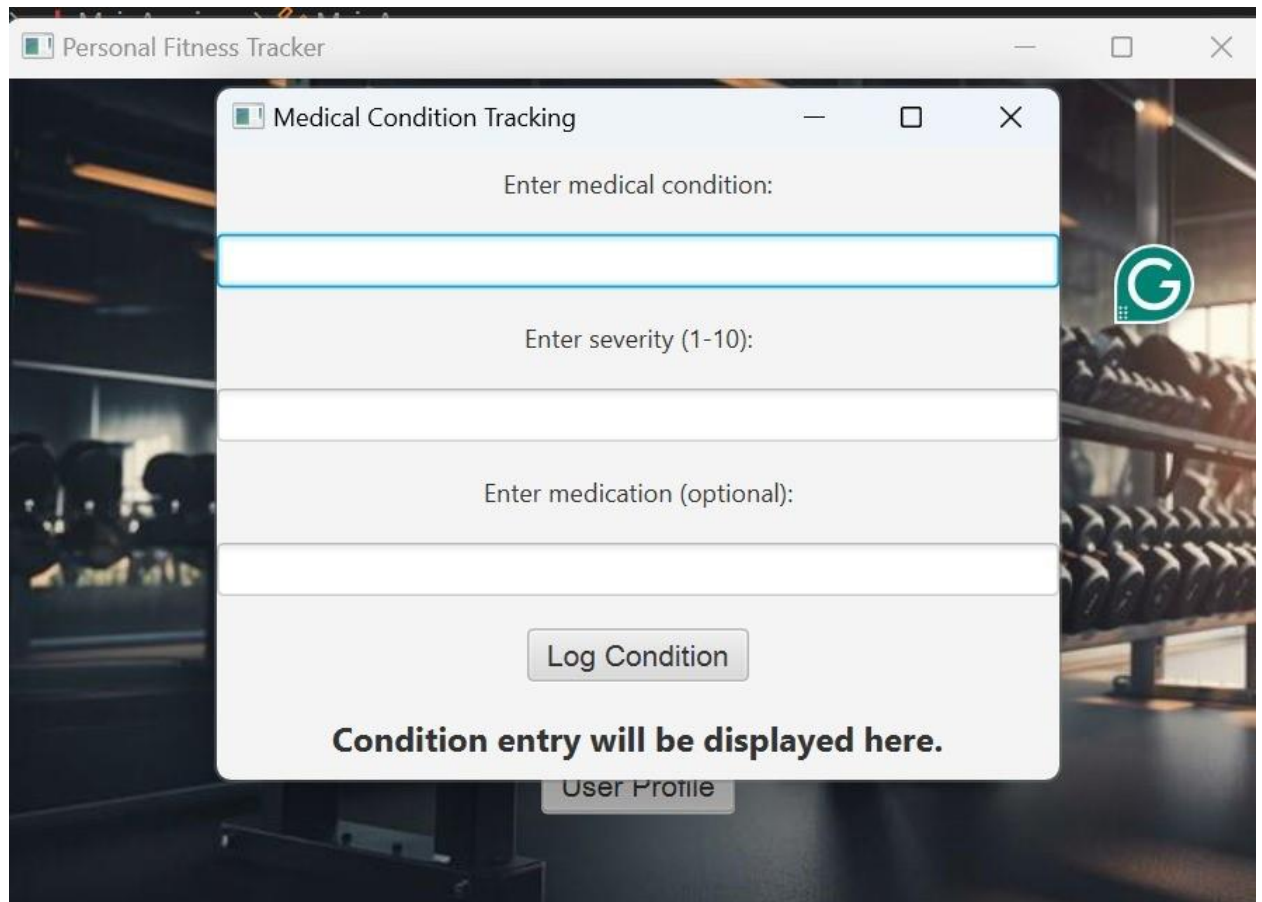
Calculate BMI

Your BMI will be displayed here.

Back

Below the modal, a "User Profile" button is partially visible. The background of the application is a blurred image of a gym with dumbbells on a rack. A green circular logo with a white 'G' is visible on the right side of the background image.

4.7 MEDICAL CONDITION TRACKING PAGE:



The screenshot shows a web application window titled "Personal Fitness Tracker". Inside, a modal window titled "Medical Condition Tracking" is open. The modal contains three input fields for "Enter medical condition:", "Enter severity (1-10):", and "Enter medication (optional):". Below these fields is a "Log Condition" button. At the bottom of the modal, a message states "Condition entry will be displayed here." The background of the application shows a gym setting with a rack of dumbbells. A green circular logo with a white 'G' is visible on the right side of the background image.

Personal Fitness Tracker

Medical Condition Tracking

Enter medical condition:

Enter severity (1-10):

Enter medication (optional):

Log Condition

Condition entry will be displayed here.

User Profile

Conclusion

Conclusion :

The development of the Fitness Tracking Application marks a significant step forward in health and fitness management, offering an integrated platform for users to monitor and improve their well-being. The project successfully fulfills its objectives by streamlining the process of tracking fitness goals, managing activity logs, and storing critical health-related data.

Built using Java with a JDBC connection to a MySQL database, the system demonstrates security, reliability, and efficiency in managing large volumes of user data. JavaFX enhances the user experience with an intuitive and visually appealing interface, allowing users to seamlessly log activities, track nutritional intake, and view progress reports. MySQL plays a crucial role in robust database management, ensuring the integrity and scalability of user records, activity logs, and health data.

The system's core functionalities, including user authentication, profile management, activity tracking, and medical condition logging, meet the essential requirements for a modern fitness management application. Non-functional aspects such as performance optimization, data security, and scalability ensure that the application remains robust, adaptable, and secure for users over time.

In conclusion, the Fitness Tracking Application effectively addresses the challenges of health and fitness monitoring through a comprehensive and integrated solution. By employing a strong backend architecture and a user-centric design, the system provides a reliable, efficient, and secure platform for users to achieve their fitness goals. Its scalability further ensures readiness for future enhancements, such as wearable integration or advanced health analytics, solidifying its role as a dependable tool for personal fitness management.

REFERENCE

7.1 REFERENCES

1. <https://www.javatpoint.com/java-tutorial>
2. <https://www.wikipedia.org/>
3. <https://www.w3schools.com/sql/>