

gym\_buddy

Object Oriented Programming



JuNE 29, 2024

IQRA UNIVERSITY

MAIN CAMPUS

**Muhammad Faheem (64821)**

**Abdul Sarim Khan (62527)**

**Shees (62609)**

**Mohib Sidique (62464)**

|  |  |
| --- | --- |
| NO | INDEX |
|  |  |
| 1. | User Account Login and Registration |
| 2. | BMI Calculator |
| 3. | Health Condition Optimizer |
| 4. | Caloric Burn Estimator |
| 5. | Workout Plan Suggester |
| 6. | Progress Tracker |
| 7. | Fitness Challenges |
| 8. | Points System |
| 9. | Leaderboard |
| 10. | User-Friendly Interface |
|  |  |
|  |  |
| ***GYM BUDDY REPORT******1.User Account Login and Registration*** The core functionality of GYM\_BUDDY is the user account login and registration feature, which guarantees that users have individualized access to the system. Users must enter their username and password in the login form, but their email address, age, and fitness objectives are collected in the registration form. | |

## ***2.BMI Calculator***

Users can evaluate their body mass index, a crucial health indicator, with the aid of the BMI calculator. When users enter their height and weight, the calculator calculates their BMI using the formula BMI = weight (kg) / (height (m)^2. Users are given a clear knowledge of their weight status with the resultant categories of underweight, normal weight, overweight, and obese. Users are encouraged to take proper action and are educated about their health by this feature.

### ***3.Health Condition Optimizer***

Based on the user's age, gender, BMI, and other health information, the Health Condition Optimizer provides tailored recommendations to enhance their well-being. It recommends appropriate exercise regimens.

## ***4.Caloric Burn Estimator***

## The Caloric Burn Estimator calculates the number of calories burned during various physical activities. Users can select activities such as running, cycling, or weightlifting, and input the duration and intensity of the exercise. The estimator uses metabolic equivalent (MET) values to determine caloric expenditure, helping users understand the impact of their workouts on their overall calorie balance. This feature aids in weight management and fitness planning.

## ***5.Workout Plan Suggester***

The Workout Plan Suggester helps choose workout schedules for beginners, intermediate & advance. It provides a range of workouts arranged into ever more intense weekly schedules. The suggester modifies workout routines in response to consumer input and advancement, guaranteeing that the exercises stay demanding but doable. Users who use this function are more likely to stick with their workout regimens.

## ***6.Progress Tracker***

The Progress Tracker tracks and shows users' long-term fitness goals. Metrics like weight fluctuations, frequency of workouts, and performance enhancements are monitored. The tracker makes it simple for users to see their improvement by visually representing progress through charts and graphs. Frequent tracking encourages users to maintain consistency and points up areas that could require modification.

## ***7.Fitness Challenges***

Fitness challenges, like a 30-day plank challenge or a weekly step goal, compel users to participate by giving them precise objectives to meet within a predetermined amount of time. The purpose of these challenges is to inspire users to stretch their boundaries while being enjoyable and motivating. Users can participate in challenges, monitor their progress, and receive prizes when they finish them. This feature encourages friendly rivalry and a sense of community.

## ***8.Points System***

Users are rewarded for their engagement and advancement inside the application through the Points System. Points are awarded to users for finishing exercises, hitting fitness goals, taking part in challenges, and being consistent. This aspect of gamification increases user motivation and engagement.

## ***9.Leaderboard***

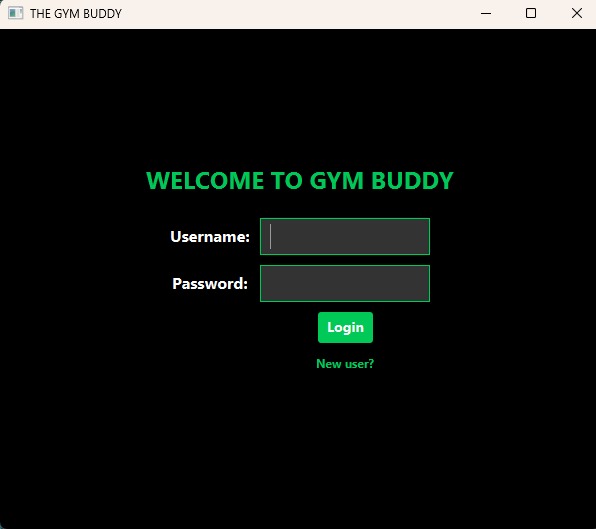
Users are ranked by the Leaderboard according on their total points, frequency of workouts, and accomplishment of challenges. It offers a competitive setting where users can assess their own performance in relation to others. This feature capitalises on users' competitive tendency to increase motivation. Giving consumers the option to show their full name or an alias on the Leaderboard protects their privacy.

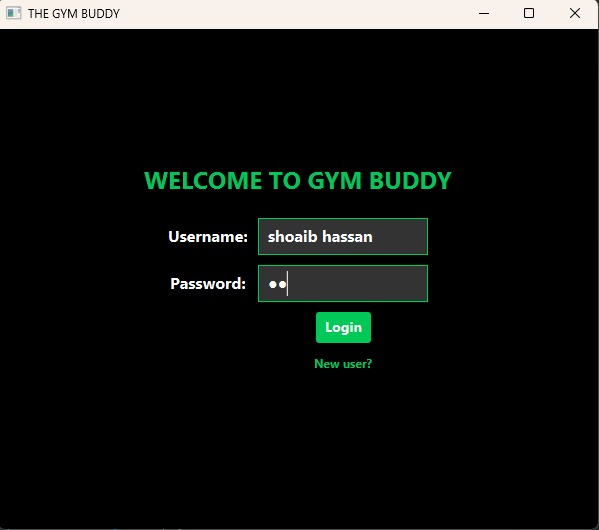
## ***10.User-Friendly Interface***

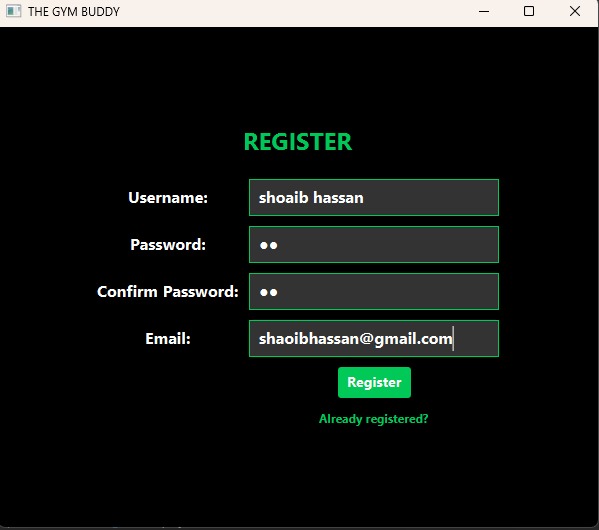
The usability and accessibility of the User-Friendly Interface are priorities in its design. It has a simple, straightforward design that is simple to navigate. Essential design tenets like responsiveness, consistency, and simplicity are adhered to guarantee a smooth user experience across platforms. The interface has been improved through user feedback and usability testing, making it usable for users of all ages and technical skill levels.

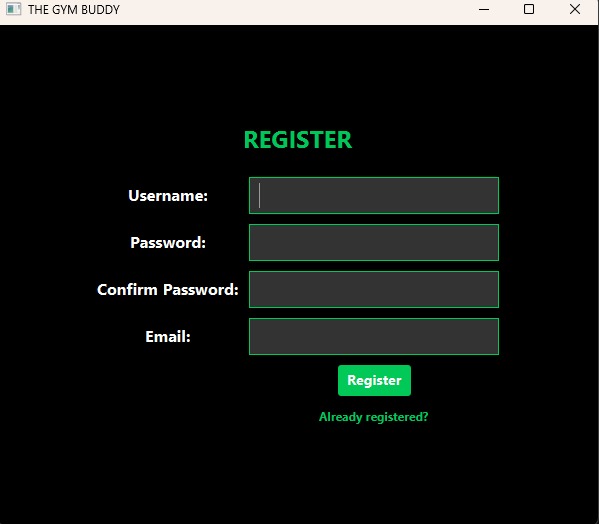
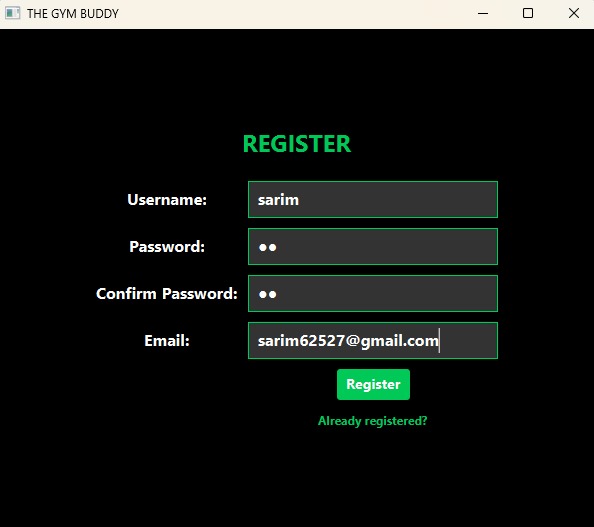
***PROJECTS SCREENSHOTS:***

***Login Page***

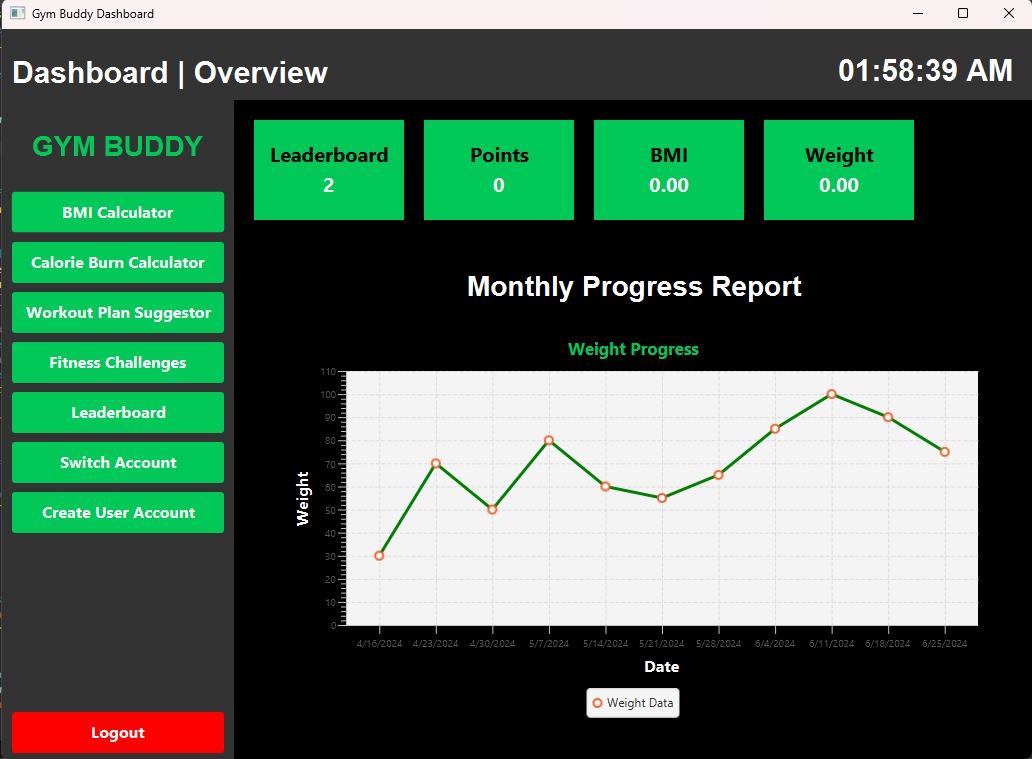
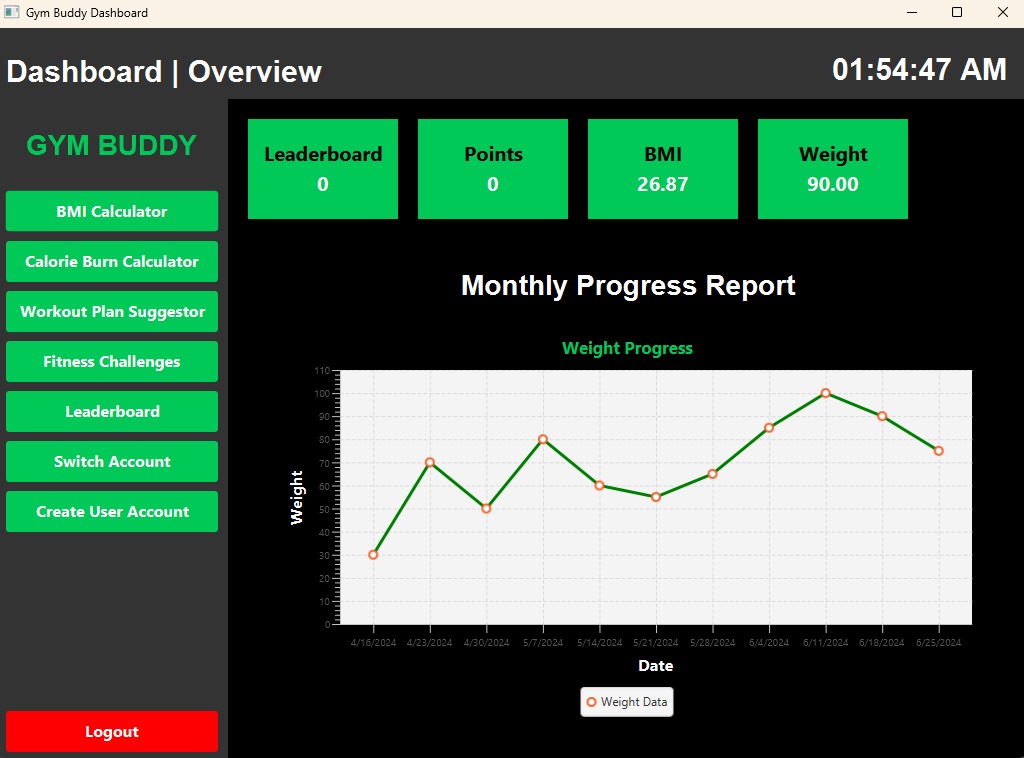
******

******

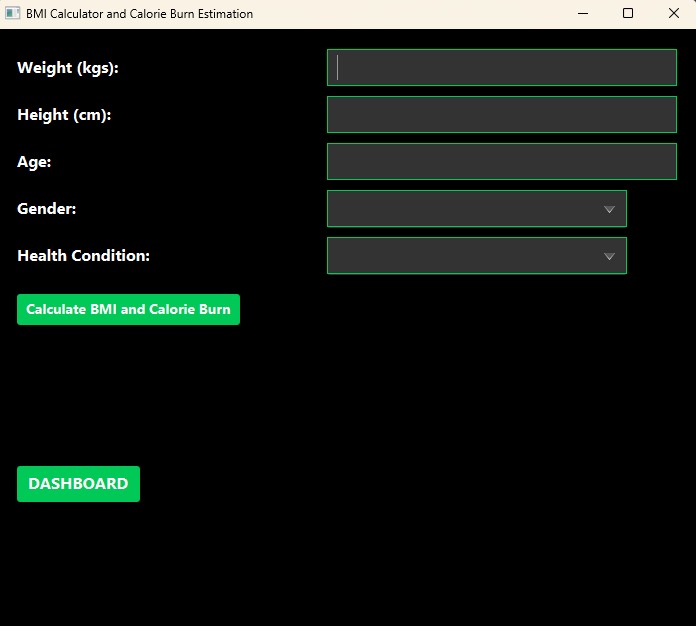
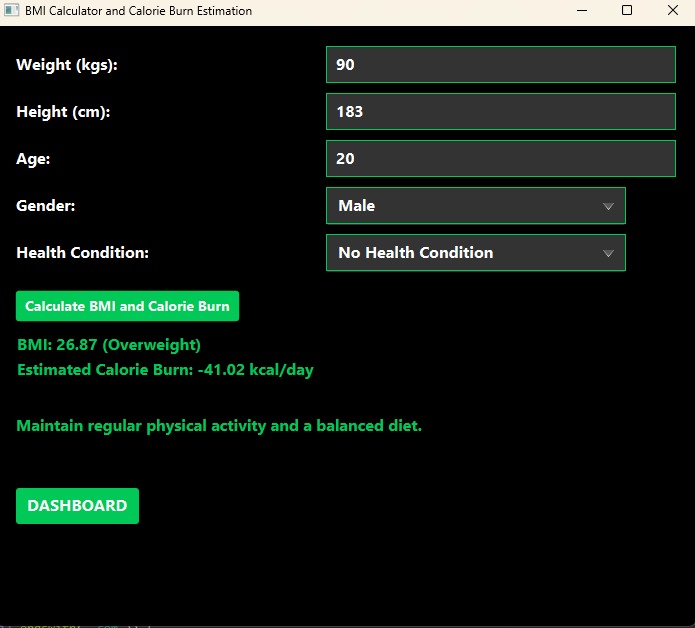
***Registration page***

******

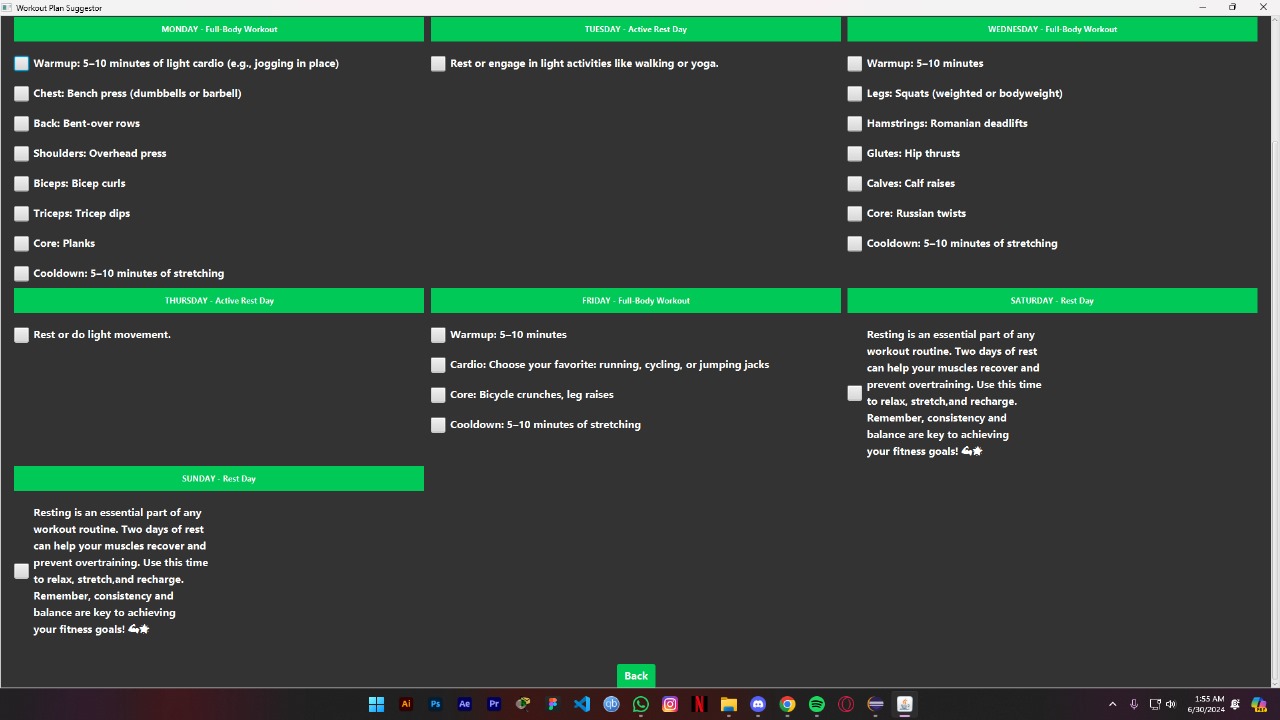
**Dashboard**

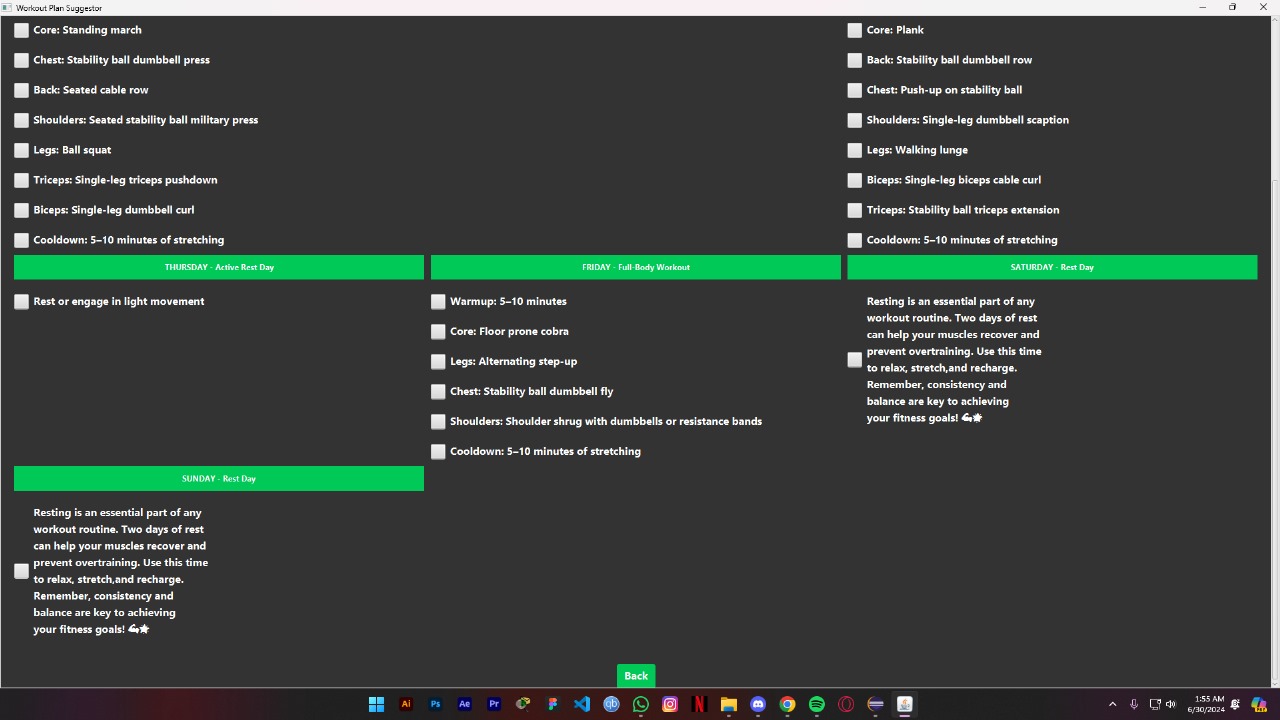
****

**BMI calculator**

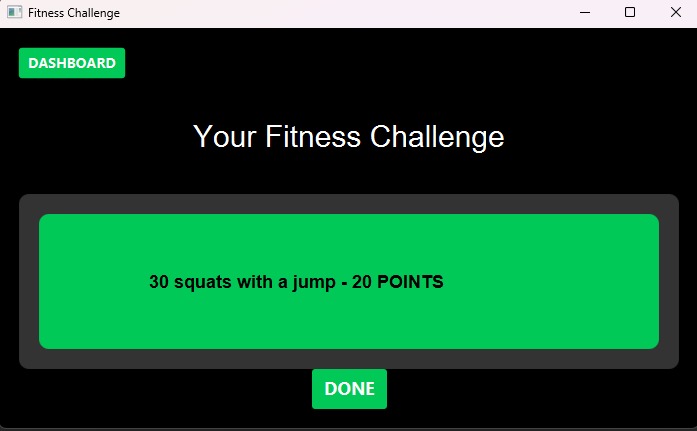
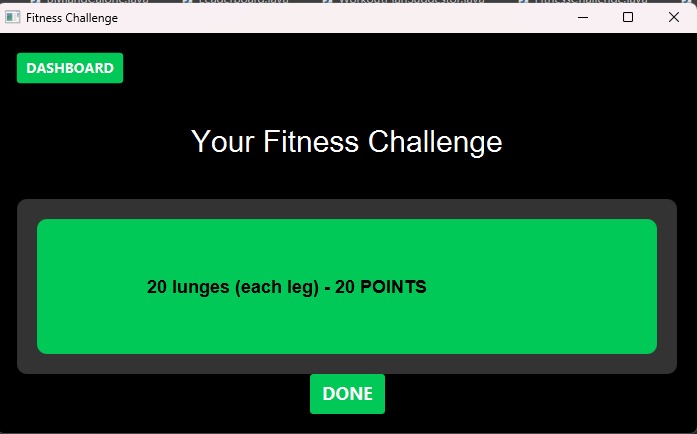
****

**Workout Plans**

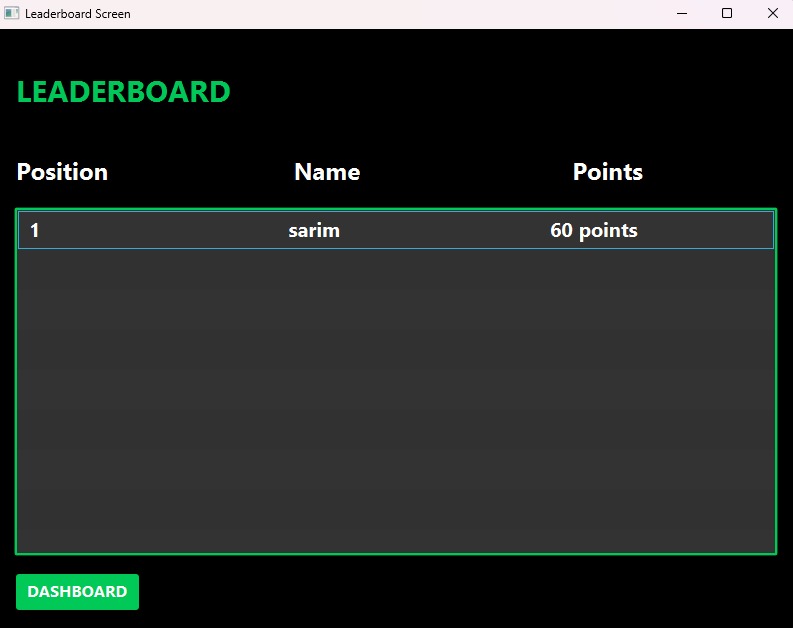
****

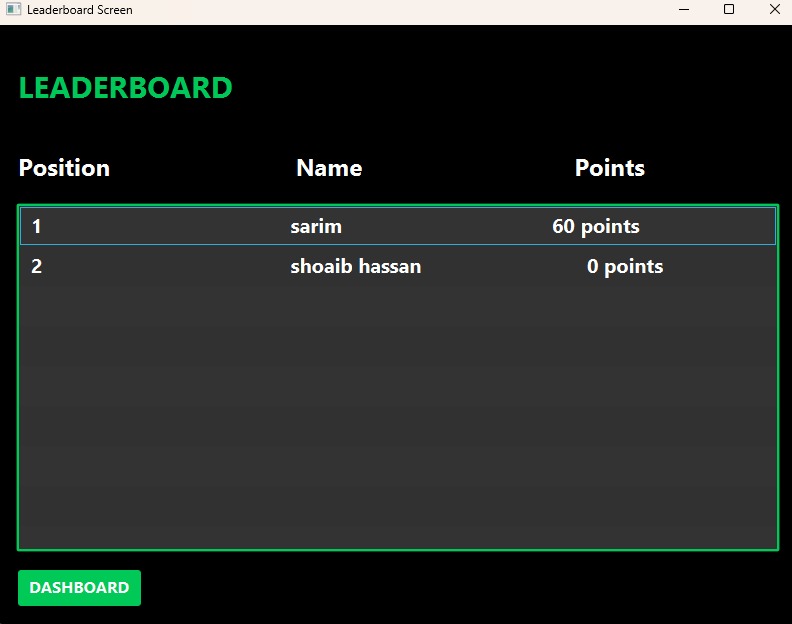
****

**Random Challenges**

****

**Leaderboard**

****

****

***SOURCE CODE:***

***PROJECT CODE***

# GYM\_BUDDY.java

package application;

import javafx.application.Application;

import javafx.geometry.HPos;

import javafx.geometry.Pos;

import javafx.scene.Scene;

import javafx.scene.control.Button;

import javafx.scene.control.Hyperlink;

import javafx.scene.control.Label;

import javafx.scene.control.PasswordField;

import javafx.scene.control.TextField;

import javafx.scene.layout.GridPane;

import javafx.scene.layout.VBox;

import javafx.scene.paint.Color;

import javafx.scene.text.Font;

import javafx.stage.Stage;

import java.io.BufferedReader;

import java.io.BufferedWriter;

import java.io.FileNotFoundException;

import java.io.FileReader;

import java.io.FileWriter;

import java.io.IOException;

import java.util.ArrayList;

import java.util.List;

import static application.ScreenShiftUtils.\*;

public class GYM\_BUDDY extends Application {

private Stage primaryStage;

public static String filePath = "Data.txt";

public static void main(String[] args) {

launch(args);

}

@Override

public void start(Stage primaryStage) {

this.primaryStage = primaryStage;

primaryStage.setTitle("THE GYM BUDDY");

showLoginForm();

}

public void showLoginForm() {

showLoginForm(this.primaryStage);

}

public String regerror;

public void showLoginForm(Stage primaryStage) {

this.primaryStage = primaryStage;

primaryStage.setTitle("THE GYM BUDDY");

// Create the heading

Label heading = new Label("WELCOME TO GYM BUDDY");

heading.setFont(new Font("Arial", 24));

heading.setStyle("-fx-text-fill: #00C958; -fx-font-size: 24px; -fx-font-weight: bold;");

heading.setAlignment(Pos.CENTER);

// Create the grid pane for login

GridPane loginGrid = new GridPane();

loginGrid.setHgap(10);

loginGrid.setVgap(10);

loginGrid.setAlignment(Pos.CENTER); // Center the grid pane

// Create the login components

Label userLabel = new Label("Username:");

userLabel.setTextFill(Color.WHITE);

userLabel.setStyle("-fx-text-fill: white; -fx-font-size: 16px; -fx-font-weight: bold;");

TextField userTextField = new TextField();

userTextField.setStyle("-fx-text-fill: white; -fx-font-size: 16px; -fx-font-weight: bold; -fx-background-color: #333333; -fx-pref-height: 20px; -fx-border-color: #00C958; -fx-pref-width: 170px;");

Label pwLabel = new Label("Password:");

pwLabel.setStyle("-fx-text-fill: white; -fx-font-size: 16px; -fx-font-weight: bold;");

PasswordField pwField = new PasswordField();

pwField.setStyle("-fx-text-fill: white; -fx-font-size: 16px; -fx-font-weight: bold; -fx-background-color: #333333; -fx-pref-height: 20px; -fx-border-color: #00C958; -fx-pref-width: 170px;");

Button loginButton = new Button("Login");

// Set initial style

loginButton.setStyle("-fx-background-color: #00C958; -fx-text-fill: white; -fx-font-size: 14px; -fx-font-weight: bold;");

// Mouse entered style

loginButton.setOnMouseEntered(e -> {

loginButton.setStyle("-fx-background-color: white; -fx-text-fill: #00C958; -fx-font-size: 14px; -fx-font-weight: bold;");

});

// Mouse exited style

loginButton.setOnMouseExited(e -> {

loginButton.setStyle("-fx-background-color: #00C958; -fx-text-fill: white; -fx-font-size: 14px; -fx-font-weight: bold;");

});

Hyperlink signUpLink = new Hyperlink("New user?");

signUpLink.setStyle("-fx-text-fill: #00C958; -fx-font-weight: bold;");

signUpLink.setOnMouseEntered(e -> {

signUpLink.setStyle("-fx-text-fill: white; -fx-font-size: 12px; -fx-font-weight: bold;");

signUpLink.getScene().setCursor(javafx.scene.Cursor.HAND);

});

signUpLink.setOnMouseExited(e -> {

signUpLink.setStyle("-fx-text-fill: #00C958; -fx-font-size: 12px; -fx-font-weight: bold;");

signUpLink.getScene().setCursor(javafx.scene.Cursor.DEFAULT);

});

// Add components to the login grid

loginGrid.add(userLabel, 0, 0);

loginGrid.add(userTextField, 1, 0);

loginGrid.add(pwLabel, 0, 1);

loginGrid.add(pwField, 1, 1);

loginGrid.add(loginButton, 1, 2);

loginGrid.add(signUpLink, 1, 3);

// Center the components within their cells

GridPane.setHalignment(userLabel, HPos.CENTER);

GridPane.setHalignment(userTextField, HPos.CENTER);

GridPane.setHalignment(pwLabel, HPos.CENTER);

GridPane.setHalignment(pwField, HPos.CENTER);

GridPane.setHalignment(loginButton, HPos.CENTER);

GridPane.setHalignment(signUpLink, HPos.CENTER);

// Create the grid pane for Error

GridPane ErrorGrid = new GridPane();

ErrorGrid.setHgap(10);

ErrorGrid.setVgap(10);

ErrorGrid.setAlignment(Pos.CENTER);

// Create a VBox to center the grid and add the heading at the top

VBox vbox = new VBox(20); // 20 is the spacing between elements

vbox.setAlignment(Pos.CENTER);

vbox.getChildren().addAll(heading, loginGrid, ErrorGrid);

vbox.setStyle("-fx-background-color: black;");

// Set up the scene and stage

Scene loginScene = new Scene(vbox, 600, 500);

primaryStage.setScene(loginScene);

primaryStage.show();

// Handle the sign-up link action

signUpLink.setOnAction(e -> showRegistrationForm());

// Handle the login action

loginButton.setOnAction(e -> {

String username = userTextField.getText();

String password = pwField.getText();

Label errorMessage = new Label("User not registered.");

if (verifyUserLogin(username, password, filePath, ",")) {

showDashboard(primaryStage);

} else {

System.out.println("Login failed. User not registered.");

// Display an alert or error message

errorMessage.setStyle("-fx-text-fill: #FF0000; -fx-font-size: 16px; -fx-font-weight: bold");

if (!vbox.getChildren().contains(errorMessage)) {

ErrorGrid.add(errorMessage, 0, 4);

}

}

});

}

public static String user = null;

public void showRegistrationForm() {

showRegistrationForm(this.primaryStage);

}

public void showRegistrationForm(Stage primaryStage) {

this.primaryStage = primaryStage;

primaryStage.setTitle("THE GYM BUDDY");

// Create the heading

Label heading = new Label("REGISTER");

heading.setFont(new Font("Arial", 24));

heading.setStyle("-fx-text-fill: #00C958; -fx-font-size: 24px; -fx-font-weight: bold;");

heading.setAlignment(Pos.CENTER);

// Create the grid pane for registration

GridPane regGrid = new GridPane();

regGrid.setHgap(10);

regGrid.setVgap(10);

regGrid.setAlignment(Pos.CENTER); // Center the grid pane

// Create the registration components

Label usernameLabel = new Label("Username:");

usernameLabel.setStyle("-fx-text-fill: white; -fx-font-size: 16px; -fx-font-weight: bold;");

TextField usernameField = new TextField();

usernameField.setStyle("-fx-text-fill: white; -fx-font-size: 16px; -fx-font-weight: bold; -fx-background-color: #333333; -fx-pref-height: 20px; -fx-border-color: #00C958; -fx-pref-width: 170px;");

Label pwLabel = new Label("Password:");

pwLabel.setStyle("-fx-text-fill: white; -fx-font-size: 16px; -fx-font-weight: bold;");

PasswordField pwField = new PasswordField();

pwField.setStyle("-fx-text-fill: white; -fx-font-size: 16px; -fx-font-weight: bold; -fx-background-color: #333333; -fx-pref-height: 20px; -fx-border-color: #00C958; -fx-pref-width: 170px;");

Label confirmPwLabel = new Label("Confirm Password:");

confirmPwLabel.setStyle("-fx-text-fill: white; -fx-font-size: 16px; -fx-font-weight: bold;");

PasswordField confirmPwField = new PasswordField();

confirmPwField.setStyle("-fx-text-fill: white; -fx-font-size: 16px; -fx-font-weight: bold; -fx-background-color: #333333; -fx-pref-height: 20px; -fx-border-color: #00C958; -fx-pref-width: 170px;");

Label emailLabel = new Label("Email:");

emailLabel.setStyle("-fx-text-fill: white; -fx-font-size: 16px; -fx-font-weight: bold;");

TextField emailField = new TextField();

emailField.setStyle("-fx-text-fill: white; -fx-font-size: 16px; -fx-font-weight: bold; -fx-background-color: #333333; -fx-pref-height: 20px; -fx-border-color: #00C958; -fx-pref-width: 250px;");

Button registerButton = new Button("Register");

// Set initial style

registerButton.setStyle("-fx-background-color: #00C958; -fx-text-fill: white; -fx-font-size: 14px; -fx-font-weight: bold;");

// Mouse entered style

registerButton.setOnMouseEntered(e -> {

registerButton.setStyle("-fx-background-color: white; -fx-text-fill: #00C958; -fx-font-size: 14px; -fx-font-weight: bold;");

});

// Mouse exited style

registerButton.setOnMouseExited(e -> {

registerButton.setStyle("-fx-background-color: #00C958; -fx-text-fill: white; -fx-font-size: 14px; -fx-font-weight: bold;");

});

Hyperlink oldUserLink = new Hyperlink("Already registered?");

oldUserLink.setStyle("-fx-text-fill: #00C958; -fx-font-weight: bold;");

oldUserLink.setOnMouseEntered(e -> {

oldUserLink.setStyle("-fx-text-fill: white; -fx-font-size: 12px; -fx-font-weight: bold;");

oldUserLink.getScene().setCursor(javafx.scene.Cursor.HAND);

});

oldUserLink.setOnMouseExited(e -> {

oldUserLink.setStyle("-fx-text-fill: #00C958; -fx-font-size: 12px; -fx-font-weight: bold;");

oldUserLink.getScene().setCursor(javafx.scene.Cursor.DEFAULT);

});

// Add components to the registration grid

regGrid.add(usernameLabel, 0, 0);

regGrid.add(usernameField, 1, 0);

regGrid.add(pwLabel, 0, 1);

regGrid.add(pwField, 1, 1);

regGrid.add(confirmPwLabel, 0, 2);

regGrid.add(confirmPwField, 1, 2);

regGrid.add(emailLabel, 0, 3);

regGrid.add(emailField, 1, 3);

regGrid.add(registerButton, 1, 4);

regGrid.add(oldUserLink, 1, 5);

// Center the components within their cells

GridPane.setHalignment(usernameLabel, HPos.CENTER);

GridPane.setHalignment(usernameField, HPos.CENTER);

GridPane.setHalignment(pwLabel, HPos.CENTER);

GridPane.setHalignment(pwField, HPos.CENTER);

GridPane.setHalignment(confirmPwLabel, HPos.CENTER);

GridPane.setHalignment(confirmPwField, HPos.CENTER);

GridPane.setHalignment(emailLabel, HPos.CENTER);

GridPane.setHalignment(emailField, HPos.CENTER);

GridPane.setHalignment(registerButton, HPos.CENTER);

GridPane.setHalignment(oldUserLink, HPos.CENTER);

// Create a VBox to center the grid and add the heading at the top

VBox vbox = new VBox(20); // 20 is the spacing between elements

vbox.setAlignment(Pos.CENTER);

vbox.getChildren().addAll(heading, regGrid);

vbox.setStyle("-fx-background-color: black;");

// Set up the scene and stage

Scene regScene = new Scene(vbox, 600, 500);

primaryStage.setScene(regScene);

primaryStage.show();

// Handle the old user link action

oldUserLink.setOnAction(e -> showLoginForm());

// Handle the register button action

registerButton.setOnAction(e -> {

String username = usernameField.getText();

String password = pwField.getText();

String confirmPassword = confirmPwField.getText();

String email = emailField.getText();

if (password.equals(confirmPassword)) {

if (registerUser(username, password, email)) {

System.out.println("Account Registered successfully");

showLoginForm();

} else {

System.out.println("Registration failed.");

}

} else {

System.out.println("Passwords do not match.");

}

});

}

private boolean verifyUserLogin(String username, String password, String filePath, String delimiter) {

String currentLine;

String[] data;

try (BufferedReader br = new BufferedReader(new FileReader(filePath))) {

while ((currentLine = br.readLine()) != null) {

data = currentLine.split(delimiter);

if (data[0].equals(username) && data[1].equals(password)) {

user = username;

// Load the points

FitnessChallenge.POINTS = Integer.parseInt(data[2]);

//Load the Position

Leaderboard.pos = Integer.parseInt(data[3]);

updateUserPosition(username, Leaderboard.pos);

return true;

}

}

} catch (FileNotFoundException e) {

return false;

} catch (Exception e) {

e.printStackTrace();

}

return false;

}

private boolean registerUser(String username, String password, String email) {

if (username == null || username.isEmpty() || password == null || password.isEmpty() || email == null || email.isEmpty()) {

System.out.println("All fields are required.");

return false;

}

else if (!email.contains("@") || !email.endsWith(".com")) {

System.out.println("Invalid syntax of Email");

return false;

}

try (BufferedWriter writer = new BufferedWriter(new FileWriter(filePath, true))) {

writer.write(username + "," + password + "," + "0" + "," + "0"); // Initialize points and Position to 0

writer.newLine();

return true;

} catch (Exception e) {

e.printStackTrace();

return false;

}

}

public static void updateUserPosition(String username, int newPosition) {

List<String> fileContent = new ArrayList<>();

String currentLine;

String delimiter = ",";

try (BufferedReader br = new BufferedReader(new FileReader(filePath))) {

while ((currentLine = br.readLine()) != null) {

String[] data = currentLine.split(delimiter);

if (data[0].equals(username)) {

data[3] = String.valueOf(newPosition); // Update the position

currentLine = String.join(delimiter, data);

}

fileContent.add(currentLine);

}

} catch (IOException e) {

e.printStackTrace();

}

try (BufferedWriter writer = new BufferedWriter(new FileWriter(filePath))) {

for (String line : fileContent) {

writer.write(line);

writer.newLine();

}

} catch (IOException e) {

e.printStackTrace();

}

}

}

# Dashboard.java

package application;

import javafx.animation.KeyFrame;

import javafx.animation.Timeline;

import javafx.geometry.Insets;

import javafx.geometry.Pos;

import javafx.scene.Scene;

import javafx.scene.chart.CategoryAxis;

import javafx.scene.chart.LineChart;

import javafx.scene.chart.NumberAxis;

import javafx.scene.chart.XYChart;

import javafx.scene.control.Button;

import javafx.scene.control.Label;

import javafx.scene.layout.BorderPane;

import javafx.scene.layout.HBox;

import javafx.scene.layout.Priority;

import javafx.scene.layout.VBox;

import javafx.scene.text.Font;

import javafx.scene.text.Text;

import javafx.stage.Stage;

import javafx.util.Duration;

import java.time.LocalTime;

import java.time.format.DateTimeFormatter;

import static application.ScreenShiftUtils.\*;

public class Dashboard {

// Define static fields for the scene and stage

public static Scene dashboardscene;

public static Stage DashboardStage;

public void start(Stage stage) {

DashboardStage = stage;

// Top bar with the dashboard label and clock

Label dashboardLabel = new Label("Dashboard | Overview");

dashboardLabel.setFont(new Font("Arial", 30));

dashboardLabel.setStyle("-fx-text-fill: white; -fx-font-weight: bold; -fx-padding: 15px 40px 0 0");

Label clock = new Label();

clock.setStyle("-fx-text-fill: white; -fx-font-weight: bold; -fx-padding: 10px 0 0 450px;");

clock.setFont(new Font("Arial", 30));

HBox topBar = new HBox(10);

topBar.setPadding(new Insets(10));

topBar.setAlignment(Pos.CENTER\_LEFT); // Aligning dashboardLabel to the left

HBox.setHgrow(clock, Priority.ALWAYS); // Allowing clock to grow to fill space

topBar.getChildren().addAll(dashboardLabel, new Label(""), clock); // Using a placeholder to push clock to right

topBar.setStyle("-fx-background-color: #333333;");

Timeline timeline = new Timeline(new KeyFrame(Duration.seconds(1), event -> {

LocalTime currentTime = LocalTime.now();

DateTimeFormatter formatter = DateTimeFormatter.ofPattern("hh:mm:ss a");

clock.setText(currentTime.format(formatter));

}));

timeline.setCycleCount(Timeline.INDEFINITE);

timeline.play();

VBox sidebar = new VBox(10);

sidebar.setPadding(new Insets(10));

Label gymBuddyLabel = new Label("GYM BUDDY");

gymBuddyLabel.setFont(new Font("Arial", 28));

gymBuddyLabel.setStyle("-fx-text-fill: #00C958; -fx-font-weight: bold; -fx-padding: 20px ;");

Button BMIButton = createSidebarButton("BMI Calculator");

Button CalorieButton = createSidebarButton("Calorie Burn Calculator");

Button createAccountButton = createSidebarButton("Create User Account");

Button workoutPlanButton = createSidebarButton("Workout Plan Suggestor");

Button challengesButton = createSidebarButton("Fitness Challenges");

Button leaderboardButton = createSidebarButton("Leaderboard");

Button logoutButton = createSidebarButton("Logout");

Button SwitchAccountButton = createSidebarButton("Switch Account");

// Button Actions

BMIButton.setOnAction(event -> {

showBMIandCalorie(stage); // Pass the current stage to close it

});

CalorieButton.setOnAction(event -> {

showBMIandCalorie(stage); // Pass the current stage to close it

});

createAccountButton.setOnAction(event -> {

BMIandCalorie.w = 0;

BMIandCalorie.b = 0;

showRegistration(stage); // Pass the current stage to close it

});

workoutPlanButton.setOnAction(event -> {

showWorkoutPlanSuggestor(stage); // Pass the current stage to close it

});

challengesButton.setOnAction(event -> {

showFitnessChallenge(stage); // Implement logic for challenges button action

});

leaderboardButton.setOnAction(event -> {

showLeaderboard(stage); // Pass the current stage to close it

});

SwitchAccountButton.setOnAction(event -> {

BMIandCalorie.w = 0;

BMIandCalorie.b = 0;

showLogin(stage); // Pass the current stage to close it

});

logoutButton.setOnAction(event -> {

BMIandCalorie.w = 0;

BMIandCalorie.b = 0;

stage.close(); // Close the current stage (assuming logout clears session)

});

// Button hover effects

applyButtonHoverEffect(BMIButton);

applyButtonHoverEffect(CalorieButton);

applyButtonHoverEffect(createAccountButton);

applyButtonHoverEffect(workoutPlanButton);

applyButtonHoverEffect(challengesButton);

applyButtonHoverEffect(leaderboardButton);

applyButtonHoverEffect(logoutButton);

applyButtonHoverEffect(SwitchAccountButton);

logoutButton.setStyle("-fx-background-color: red; -fx-text-fill: white; -fx-font-size: 16px; -fx-font-weight: bold;");

applyButtonHoverEffect(logoutButton, "red", "#FF6666");

VBox logoutContainer = new VBox();

logoutContainer.getChildren().add(logoutButton);

logoutContainer.setPadding(new Insets(170, 0, 0, 0));

VBox.setVgrow(logoutContainer, Priority.ALWAYS);

sidebar.getChildren().addAll(gymBuddyLabel, BMIButton, CalorieButton, workoutPlanButton, challengesButton, leaderboardButton, SwitchAccountButton, createAccountButton);

sidebar.getChildren().add(logoutContainer);

sidebar.setStyle("-fx-background-color: #333333; -fx-padding: 10;");

VBox mainContent = new VBox(20);

mainContent.setPadding(new Insets(20));

HBox buttonRow = new HBox(20);

VBox leaderBoardContainer = createMainContainer("Leaderboard", "black", 30);

VBox pointsContainer = createMainContainer("Points", "black", 30);

VBox bmiContainer = createMainContainer("BMI", "black", 30);

VBox weightContainer = createMainContainer("Weight", "black", 30);

int pos = Leaderboard.pos; // Your logic to calculate new position

GYM\_BUDDY.updateUserPosition(GYM\_BUDDY.user, Leaderboard.pos);

Label Position = new Label(Integer.toString(pos));

Position.setStyle("-fx-text-fill: white; -fx-font-size: 20px; -fx-font-weight: bold;");

leaderBoardContainer.getChildren().addAll(Position);

int points = FitnessChallenge.POINTS;

String point = String.valueOf(points); // Convert int to String

Label pointsLabel = new Label(point);

pointsLabel.setStyle("-fx-text-fill: white; -fx-font-size: 20px; -fx-font-weight: bold;");

pointsContainer.getChildren().addAll(pointsLabel);

double Bami = BMIandCalorie.b ;

String bmiAsString = String.format("%.2f", Bami);

Label bmiLabel = new Label(bmiAsString);

bmiLabel.setStyle("-fx-text-fill: white; -fx-font-size: 20px; -fx-font-weight: bold;");

bmiContainer.getChildren().addAll(bmiLabel);

double wei = BMIandCalorie.w;

String weightAsString = String.format("%.2f", wei);

Label weightLabel = new Label(weightAsString);

weightLabel.setStyle("-fx-text-fill: white; -fx-font-size: 20px; -fx-font-weight: bold;");

weightContainer.getChildren().addAll(weightLabel);

buttonRow.getChildren().addAll(leaderBoardContainer, pointsContainer, bmiContainer, weightContainer);

Label weeklyProgressReport = new Label("Monthly Progress Report");

weeklyProgressReport.setFont(new Font("Arial", 28));

weeklyProgressReport.setStyle("-fx-text-fill: white; -fx-font-weight: bold; -fx-padding: 0 180px;");

HBox reportBox = new HBox();

reportBox.getChildren().addAll(weeklyProgressReport);

reportBox.setPadding(new Insets(10));

reportBox.setStyle("-fx-alignment: center-right;");

LineChart<String, Number> weightChart = createWeightChart();

VBox chartContainer = new VBox(20, reportBox, weightChart);

chartContainer.setPadding(new Insets(20));

chartContainer.setStyle("-fx-background-color: black;");

mainContent.getChildren().addAll(buttonRow, chartContainer);

mainContent.setStyle("-fx-background-color: black; -fx-padding: 20;");

BorderPane root = new BorderPane();

root.setTop(topBar);

root.setLeft(sidebar);

root.setCenter(mainContent);

dashboardscene = new Scene(root, 1030, 730);

DashboardStage.setTitle("Gym Buddy Dashboard");

DashboardStage.setScene(dashboardscene);

DashboardStage.show();

}

private Button createSidebarButton(String text) {

Button button = new Button(text);

button.setStyle("-fx-background-color: #666666; -fx-text-fill: white; -fx-font-size: 16px; -fx-font-weight: bold;");

button.setMaxWidth(Double.MAX\_VALUE);

button.setMinHeight(40);

return button;

}

private VBox createMainContainer(String text, String textColor, int fontSize) {

VBox container = new VBox();

container.setStyle("-fx-background-color: #00C958; -fx-text-fill: " + textColor + "; -fx-font-size: " + fontSize + "px; -fx-font-weight: bold;");

container.setMinSize(150, 100);

Label label = new Label(text);

label.setStyle("-fx-text-fill: " + textColor + "; -fx-font-size: 20px; -fx-font-weight: bold;");

container.getChildren().add(label);

container.setAlignment(Pos.CENTER);

return container;

}

private void applyButtonHoverEffect(Button button) {

button.setStyle("-fx-background-color: #00C958; -fx-text-fill: white; -fx-font-size: 16px; -fx-font-weight: bold;");

button.setOnMouseEntered(e -> button.setStyle("-fx-background-color: white; -fx-text-fill: #00C958; -fx-font-size: 16px; -fx-font-weight: bold;"));

button.setOnMouseExited(e -> button.setStyle("-fx-background-color: #00C958; -fx-text-fill: white; -fx-font-size: 16px; -fx-font-weight: bold;"));

}

private void applyButtonHoverEffect(Button button, String baseColor, String hoverColor) {

button.setOnMouseEntered(e -> button.setStyle("-fx-background-color: " + hoverColor + "; -fx-text-fill: white; -fx-font-size: 16px; -fx-font-weight: bold;"));

button.setOnMouseExited(e -> button.setStyle("-fx-background-color: " + baseColor + "; -fx-text-fill: white; -fx-font-size: 16px; -fx-font-weight: bold;"));

}

private LineChart<String, Number> createWeightChart() {

CategoryAxis xAxis = new CategoryAxis();

xAxis.setLabel("Date");

NumberAxis yAxis = new NumberAxis();

yAxis.setLabel("Weight");

LineChart<String, Number> lineChart = new LineChart<>(xAxis, yAxis);

lineChart.setTitle("Weight Progress");

XYChart.Series<String, Number> series = new XYChart.Series<>();

series.setName("Weight Data");

// Add data points

series.getData().add(new XYChart.Data<>("4/16/2024", 30));

series.getData().add(new XYChart.Data<>("4/23/2024", 70));

series.getData().add(new XYChart.Data<>("4/30/2024", 50));

series.getData().add(new XYChart.Data<>("5/7/2024", 80));

series.getData().add(new XYChart.Data<>("5/14/2024", 60));

series.getData().add(new XYChart.Data<>("5/21/2024", 55));

series.getData().add(new XYChart.Data<>("5/28/2024", 65));

series.getData().add(new XYChart.Data<>("6/4/2024", 85));

series.getData().add(new XYChart.Data<>("6/11/2024", 100));

series.getData().add(new XYChart.Data<>("6/18/2024", 90));

series.getData().add(new XYChart.Data<>("6/25/2024", 75));

lineChart.getData().add(series);

// Apply CSS styling

lineChart.setStyle("-fx-text-fill: #00C958;");

xAxis.setStyle("-fx-text-fill: #00C958;");

yAxis.setStyle("-fx-text-fill: #00C958;");

// Set line color to green

series.getNode().setStyle("-fx-stroke: green;");

// Set chart title text color and font weight

lineChart.lookup(".chart-title").setStyle("-fx-text-fill: #00C958; -fx-font-weight: bold;");

// Set axis labels to white with increased font size and bold weight

xAxis.lookup(".axis-label").setStyle("-fx-text-fill: white; -fx-font-size: 16px; -fx-font-weight: bold;");

yAxis.lookup(".axis-label").setStyle("-fx-text-fill: white; -fx-font-size: 16px; -fx-font-weight: bold;");

// Set tick label color to white using inline CSS

lineChart.applyCss(); // Ensure CSS is applied

xAxis.lookupAll(".axis-tick-mark-label").forEach(label -> ((Text)label).setStyle("-fx-fill: white;"));

yAxis.lookupAll(".axis-tick-mark-label").forEach(label -> ((Text)label).setStyle("-fx-fill: white;"));

return lineChart;

}

}

# Leaderboard.java

package application;

import javafx.collections.FXCollections;

import javafx.collections.ObservableList;

import javafx.geometry.Insets;

import javafx.scene.Scene;

import javafx.scene.control.Button;

import javafx.scene.control.Label;

import javafx.scene.control.ListView;

import javafx.scene.layout.Background;

import javafx.scene.layout.BackgroundFill;

import javafx.scene.layout.CornerRadii;

import javafx.scene.layout.VBox;

import javafx.scene.paint.Color;

import javafx.stage.Stage;

import static application.ScreenShiftUtils.\*;

import java.io.BufferedReader;

import java.io.BufferedWriter;

import java.io.File;

import java.io.FileReader;

import java.io.FileWriter;

import java.io.IOException;

import java.util.ArrayList;

import java.util.Comparator;

import java.util.List;

import java.util.stream.Collectors;

// Leaderboard Class

public class Leaderboard {

public static int pos;

public static String user;

public void start(Stage LeaderBoardStage, String loggedInUser) {

user = loggedInUser;

// Sample data for leaderboard

ObservableList<Player> leaderboardData = FXCollections.observableArrayList();

// Read all users and their points from the file

try (BufferedReader br = new BufferedReader(new FileReader(GYM\_BUDDY.filePath))) {

String currentLine;

String[] data;

while ((currentLine = br.readLine()) != null) {

data = currentLine.split(",");

String playerName = data[0];

int playerPoints = Integer.parseInt(data[2]);

leaderboardData.add(new Player(playerName, playerPoints));

}

} catch (Exception e) {

e.printStackTrace();

}

// Sort leaderboard data in descending order with respect to points

List<Player> sortedPlayers = leaderboardData.stream()

.sorted(Comparator.comparingInt(Player::getPoints).reversed())

.collect(Collectors.toList());

// Format sorted leaderboard data and find the position of the logged-in user

ObservableList<String> formattedLeaderboardData = FXCollections.observableArrayList();

int position = 1;

//

for (Player player : sortedPlayers) {

updatePositionInFile(player.getName(), position); // Update position in the file

formattedLeaderboardData.add(formatLeaderboardEntry(position++, player.getName(), player.getPoints()));

}

// Creating UI elements

Label titleLabel = new Label("LEADERBOARD");

titleLabel.setStyle("-fx-font-size: 30px; -fx-text-fill: #00C958; -fx-font-weight: bold");

titleLabel.setPadding(new Insets(20, 0, 20, 0));

Label headingLabel = new Label(String.format("%-35s %-35s %s", "Position", "Name", "Points"));

headingLabel.setStyle("-fx-font-size: 24px; -fx-text-fill: white; -fx-font-weight: bold");

ListView<String> leaderboardListView = new ListView<>(formattedLeaderboardData);

leaderboardListView.setStyle("-fx-control-inner-background: #333333; -fx-background-color: #00C958; -fx-font-size: 20px; -fx-font-weight: bold");

leaderboardListView.setPrefHeight(400);

// Create a button to go back to the Dashboard

Button HomeButton = new Button("DASHBOARD");

HomeButton.setStyle("-fx-background-color: #00C958; -fx-text-fill: white; -fx-font-size: 16px; -fx-font-weight: bold;");

HomeButton.setOnMouseEntered(e -> HomeButton.setStyle("-fx-background-color: white; -fx-text-fill: #00C958; -fx-font-size: 16px; -fx-font-weight: bold;"));

HomeButton.setOnMouseExited(e -> HomeButton.setStyle("-fx-background-color: #00C958; -fx-text-fill: white; -fx-font-size: 16px; -fx-font-weight: bold;"));

HomeButton.setOnAction(event -> {

showDashboard(LeaderBoardStage); // Pass the current stage to close it

});

// VBox layout for the screen

VBox vbox = new VBox(20);

vbox.setPadding(new Insets(20));

vbox.setBackground(new Background(new BackgroundFill(Color.rgb(0, 0, 0), CornerRadii.EMPTY, Insets.EMPTY))); // Set overall background color

vbox.getChildren().addAll(titleLabel, headingLabel, leaderboardListView, HomeButton);

// Scene creation

Scene scene = new Scene(vbox, 800, 600);

LeaderBoardStage.setTitle("Leaderboard Screen");

LeaderBoardStage.setScene(scene);

LeaderBoardStage.show();

}

private String formatLeaderboardEntry(int position, String playerName, int points) {

return String.format("%-45s %-42s %d points", position, playerName, points);

}

public static void updatePositionInFile(String playerName, int position) {

List<String> fileContent = new ArrayList<>();

// Read all lines from the file into the list

try (BufferedReader reader = new BufferedReader(new FileReader(GYM\_BUDDY.filePath))) {

String currentLine;

while ((currentLine = reader.readLine()) != null) {

fileContent.add(currentLine);

}

} catch (IOException e) {

e.printStackTrace();

}

// Update the specific player's position in the list

boolean playerFound = false;

for (int i = 0; i < fileContent.size(); i++) {

String line = fileContent.get(i);

String[] data = line.split(",");

if (data[0].equals(playerName)) {

fileContent.set(i, data[0] + "," + data[1] + "," + data[2] + "," + position);

pos = position;

playerFound = true;

break;

}

}

if (!playerFound) {

System.out.println("Player not found: " + playerName);

}

// Write the updated list back to the file

try (BufferedWriter writer = new BufferedWriter(new FileWriter(GYM\_BUDDY.filePath))) {

for (String line : fileContent) {

writer.write(line);

writer.newLine();

}

writer.flush(); // Flush the output to ensure it's written immediately

} catch (IOException e) {

e.printStackTrace();

}

}

}

// Helper class to represent a player

class Player {

private final String name;

private final int points;

public Player(String name, int points) {

this.name = name;

this.points = points;

}

public String getName() {

return name;

}

public int getPoints() {

return points;

}

}

# BMIandCalorie.java

package application;

import javafx.geometry.Insets;

import javafx.scene.Scene;

import javafx.scene.control.\*;

import javafx.scene.layout.ColumnConstraints;

import javafx.scene.layout.GridPane;

import javafx.scene.paint.Color;

import javafx.stage.Stage;

import static application.ScreenShiftUtils.\*;

public class BMIandCalorie {

private Stage primaryStage;

private static final String CSS\_FILE = "styles.css";

public void start(Stage CalculationStage) {

primaryStage = CalculationStage;

GridPane gridPane = createGridPane();

Scene scene = new Scene(gridPane, 700, 600);

scene.setFill(Color.web("#000000")); // Set scene background color

scene.getStylesheets().add(getClass().getResource(CSS\_FILE).toExternalForm());

primaryStage.setTitle("BMI Calculator and Calorie Burn Estimation");

primaryStage.setScene(scene);

primaryStage.show();

}

private GridPane createGridPane() {

GridPane gridPane = new GridPane();

gridPane.setPadding(new Insets(20));

gridPane.setHgap(10);

gridPane.setVgap(10);

ColumnConstraints labelColumn = new ColumnConstraints();

labelColumn.setPrefWidth(300);

gridPane.getColumnConstraints().addAll(labelColumn);

addUIControls(gridPane);

return gridPane;

}

public static double b = 0.0;

public static double w = 0.0;

private void addUIControls(GridPane gridPane) {

gridPane.add(createLabel("Weight (kgs):"), 0, 0);

gridPane.add(createTextField(), 1, 0);

gridPane.add(createLabel("Height (cm):"), 0, 1);

gridPane.add(createTextField(), 1, 1);

gridPane.add(createLabel("Age:"), 0, 2);

gridPane.add(createTextField(), 1, 2);

gridPane.add(createLabel("Gender:"), 0, 3);

gridPane.add(createGenderChoiceBox(), 1, 3);

gridPane.add(createLabel("Health Condition:"), 0, 4);

gridPane.add(createHealthConditionChoiceBox(), 1, 4);

Button calculateButton = new Button("Calculate BMI and Calorie Burn");

calculateButton.getStyleClass().add("calculate-button");

gridPane.add(calculateButton, 0, 6, 2, 1);

Label resultLabel = new Label();

resultLabel.getStyleClass().add("result-label");

gridPane.add(resultLabel, 0, 7, 2, 1);

Label ConditionLabel = new Label();

ConditionLabel.setPrefWidth(800); // Set preferred width to ensure full text is visible

ConditionLabel.setWrapText(true); // Enable text wrapping for long advice

gridPane.add(ConditionLabel, 0, 10, 2, 1);

// Create a button to go back to the Dashboard

Button HomeButton = new Button("DASHBOARD");

HomeButton.setStyle("-fx-background-color: #00C958; -fx-text-fill: white; -fx-font-size: 16px; -fx-font-weight: bold;");

HomeButton.setOnMouseEntered(e -> HomeButton.setStyle("-fx-background-color: white; -fx-text-fill: #00C958; -fx-font-size: 16px; -fx-font-weight: bold;"));

HomeButton.setOnMouseExited(e -> HomeButton.setStyle("-fx-background-color: #00C958; -fx-text-fill: white; -fx-font-size: 16px; -fx-font-weight: bold;"));

HomeButton.setOnAction(event -> {

showDashboard(primaryStage); // Pass the current stage to close it

});

gridPane.add(HomeButton, 0, 15, 2, 1);

ConditionLabel.setStyle("-fx-text-fill: #00C958; -fx-font-size: 16px; -fx-font-weight: bold");

calculateButton.setOnAction(e -> calculateBMIAndCalories(gridPane, resultLabel, ConditionLabel));

}

private Label createLabel(String labelText) {

Label label = new Label(labelText);

label.getStyleClass().add("label");

return label;

}

private TextField createTextField() {

TextField textField = new TextField();

textField.getStyleClass().add("input-field");

return textField;

}

private ChoiceBox<String> createGenderChoiceBox() {

ChoiceBox<String> genderChoiceBox = new ChoiceBox<>();

genderChoiceBox.getStyleClass().add("choice-box");

genderChoiceBox.getItems().addAll("Male", "Female");

return genderChoiceBox;

}

private ChoiceBox<String> createHealthConditionChoiceBox() {

ChoiceBox<String> healthConditionChoiceBox = new ChoiceBox<>();

healthConditionChoiceBox.getStyleClass().add("choice-box");

healthConditionChoiceBox.getItems().addAll("No Health Condition", "Heart Disease", "High Blood Pressure", "Chronic Pain", "Back Or Joint Pain");

return healthConditionChoiceBox;

}

private void calculateBMIAndCalories(GridPane gridPane, Label resultLabel, Label ConditionLabel) {

TextField weightField = (TextField) gridPane.getChildren().get(1);

TextField heightField = (TextField) gridPane.getChildren().get(3);

TextField ageField = (TextField) gridPane.getChildren().get(5);

ChoiceBox<String> genderChoiceBox = (ChoiceBox<String>) gridPane.getChildren().get(7);

ChoiceBox<String> healthConditionChoiceBox = (ChoiceBox<String>) gridPane.getChildren().get(9);

try {

double weightKgs = Double.parseDouble(weightField.getText());

w = weightKgs;

double heightCm = Double.parseDouble(heightField.getText());

int age = Integer.parseInt(ageField.getText());

String gender = genderChoiceBox.getValue();

String healthCondition = healthConditionChoiceBox.getValue();

if (weightKgs <= 0 || heightCm <= 0 || age <= 0 || gender == null || healthCondition == null) {

throw new NumberFormatException();

}

double heightMeters = heightCm / 100.0;

double bmi = weightKgs / (heightMeters \* heightMeters);

b = bmi;

String bmiText = String.format("%.2f", bmi);

String bmiClassification = classifyBMI(bmi);

double estimatedCaloriesBurned = calculateCaloriesBurned(weightKgs, heightCm, String.valueOf(age), gender);

String caloriesText = String.format("%.2f", estimatedCaloriesBurned);

resultLabel.setText("BMI: " + bmiText + " (" + bmiClassification + ")\nEstimated Calorie Burn: " + caloriesText + " kcal/day");

resultLabel.setStyle("-fx-text-fill: #00C958; -fx-font-size: 16px; -fx-font-weight: bold");

String healthAdvice = getHealthAdvice(healthCondition);

ConditionLabel.setText(healthAdvice);

ConditionLabel.setStyle("-fx-text-fill: #00C958; -fx-font-size: 16px; -fx-font-weight: bold");

} catch (NumberFormatException ex) {

resultLabel.setText("Invalid input. Please enter valid numbers and select all fields.");

resultLabel.setStyle("-fx-text-fill: #FF0000; -fx-font-size: 16px; -fx-font-weight: bold");

}

}

private String classifyBMI(double bmi) {

if (bmi < 16) {

return "Severe Thinness";

} else if (bmi >= 16 && bmi < 17) {

return "Moderate Thinness";

} else if (bmi >= 17 && bmi < 18.5) {

return "Mild Thinness";

} else if (bmi >= 18.5 && bmi < 25) {

return "Normal";

} else if (bmi >= 25 && bmi < 30) {

return "Overweight";

} else if (bmi >= 30 && bmi < 35) {

return "Obese Class I";

} else if (bmi >= 35 && bmi < 40) {

return "Obese Class II";

} else {

return "Obese Class III";

}

}

private double calculateCaloriesBurned(double weightKgs, double heightCm, String age, String gender) {

final double BASE\_CALORIES\_PER\_MINUTE = 0.035;

final double WEIGHT\_FACTOR = 0.029;

final double HEIGHT\_FACTOR = 0.026;

final double AGE\_FACTOR = 0.203;

double heightMeters = heightCm / 100.0;

double caloriesPerMinute = BASE\_CALORIES\_PER\_MINUTE +

(WEIGHT\_FACTOR \* weightKgs) +

(HEIGHT\_FACTOR \* heightMeters) -

(AGE\_FACTOR \* Integer.parseInt(age));

double totalCaloriesBurned = caloriesPerMinute \* 30;

if (gender.equalsIgnoreCase("female")) {

totalCaloriesBurned \*= 0.9;

}

return totalCaloriesBurned;

}

private String getHealthAdvice(String healthCondition) {

switch (healthCondition) {

case "No Health Condition":

return "Maintain regular physical activity and a balanced diet.";

case "Heart Disease":

return "Focus on a heart-healthy diet, regular exercise, and stress management. Avoid smoking and excessive alcohol consumption.";

case "High Blood Pressure":

return "Limit salt intake, maintain a healthy weight, exercise regularly, and manage stress. Avoid excessive caffeine and alcohol.";

case "Chronic Pain":

return "Consult a healthcare professional for pain management strategies. Avoid prolonged inactivity and overuse of pain medications.";

case "Back Or Joint Pain":

return "Strengthen core muscles, maintain good posture, and avoid heavy lifting. Consult a physical therapist for personalized guidance. Avoid prolonged sitting and poor ergonomics.";

default:

return "Consult a healthcare professional for personalized advice.";

}

}

}

# WorkoutPlanSuggestor.java

package application;

import javafx.geometry.Pos;

import javafx.scene.Scene;

import javafx.scene.control.Button;

import javafx.scene.control.CheckBox;

import javafx.scene.control.Label;

import javafx.scene.control.ScrollPane;

import javafx.scene.layout.ColumnConstraints;

import javafx.scene.layout.GridPane;

import javafx.scene.layout.HBox;

import javafx.scene.layout.VBox;

import javafx.scene.text.Font;

import javafx.stage.Stage;

import static application.ScreenShiftUtils.\*;

public class WorkoutPlanSuggestor {

private Stage WorkoutStage;

private Scene mainScene;

public void start(Stage WorkoutStage) {

this.WorkoutStage = WorkoutStage;

// Create an HBox layout for the top bar

HBox topBar = new HBox();

topBar.setAlignment(Pos.TOP\_LEFT);

// Create home button

Button homeButton = new Button("DASHBOARD");

homeButton.setStyle("-fx-background-color: #00C958; -fx-text-fill: white; -fx-font-size: 14px; -fx-font-weight: bold;");

// Mouse entered style

homeButton.setOnMouseEntered(e -> {

homeButton.setStyle("-fx-background-color: white; -fx-text-fill: #00C958; -fx-font-size: 14px; -fx-font-weight: bold;");

});

// Mouse exited style

homeButton.setOnMouseExited(e -> {

homeButton.setStyle("-fx-background-color: #00C958; -fx-text-fill: white; -fx-font-size: 14px; -fx-font-weight: bold;");

});

homeButton.setOnAction(event -> {

showDashboard(WorkoutStage); // Pass the current stage to close it

});

topBar.getChildren().add(homeButton);

topBar.setStyle("-fx-padding: 20px;");

// Create heading label

Label headingLabel = new Label("Choose Your Customized Workout Plan");

headingLabel.setFont(new Font("Arial", 30));

headingLabel.setStyle("-fx-text-fill: white;");

// Create three plans with descriptions

VBox plan1 = createPlan("Ideal for newbies or those returning to exercise. Focuses on building a foundation and improving overall fitness.", "BEGINNER PLAN");

VBox plan2 = createPlan("For consistent exercisers who want to level up. Offers variety, intensity, and targeted muscle development.", "INTERMEDIATE PLAN");

VBox plan3 = createPlan("Designed for seasoned fitness enthusiasts. Aims to build muscle, strength, and enhance athletic performance.", "ADVANCED PLAN");

// Create HBox to hold the three plans

HBox hbox = new HBox(20, plan1, plan2, plan3);

hbox.setAlignment(Pos.CENTER);

// Create a gray box to contain the plans

VBox grayBox = new VBox(hbox);

grayBox.setStyle("-fx-background-color: black; -fx-padding: 20px; -fx-border-radius: 10px; -fx-background-radius: 10px; -fx-pref-height: 500px; -fx-max-width: 900px");

grayBox.setAlignment(Pos.CENTER);

// Create a VBox to hold the top bar, heading, and the gray box

VBox outerVBox = new VBox(20, topBar, headingLabel, grayBox);

outerVBox.setAlignment(Pos.TOP\_CENTER);

// Create the main scene with the outer VBox layout

mainScene = new Scene(outerVBox, 1020, 600);

outerVBox.setStyle("-fx-background-color: black;");

// Set the title of the stage (window) and add the main scene to it

WorkoutStage.setTitle("Workout Plan Suggestor");

WorkoutStage.setScene(mainScene);

WorkoutStage.show();

}

private VBox createPlan(String description, String buttonText) {

// Create description label

Label descriptionLabel = new Label(description);

descriptionLabel.setWrapText(true);

descriptionLabel.setMaxWidth(200);

descriptionLabel.setFont(new Font("Arial", 18));

descriptionLabel.setStyle("-fx-text-fill: white; -fx-padding: 10px; -fx-font-weight: bold;");

// Create a green box for description

VBox descriptionBox = new VBox(descriptionLabel);

descriptionBox.setStyle("-fx-background-color: #00C958; -fx-border-radius: 10px; -fx-background-radius: 10px; -fx-padding: 20px;");

// Create button

Button planButton = new Button(buttonText);

planButton.setStyle("-fx-background-color: #00C958; -fx-text-fill: white; -fx-font-size: 18px; -fx-font-weight: bold;");

// Mouse entered style

planButton.setOnMouseEntered(e -> {

planButton.setStyle("-fx-background-color: white; -fx-text-fill: #00C958; -fx-font-size: 18px; -fx-font-weight: bold;");

});

// Mouse exited style

planButton.setOnMouseExited(e -> {

planButton.setStyle("-fx-background-color: #00C958; -fx-text-fill: white; -fx-font-size: 18px; -fx-font-weight: bold;");

});

// Set action for each plan button

switch (buttonText) {

case "BEGINNER PLAN":

planButton.setOnAction(event -> openBeginnerPlanPage((Stage) planButton.getScene().getWindow()));

break;

case "INTERMEDIATE PLAN":

planButton.setOnAction(event -> openIntermediatePlanPage((Stage) planButton.getScene().getWindow()));

break;

case "ADVANCED PLAN":

planButton.setOnAction(event -> openAdvancedPlanPage((Stage) planButton.getScene().getWindow()));

break;

}

// Create VBox to hold description box and button

VBox vbox = new VBox(20, descriptionBox, planButton);

vbox.setStyle("-fx-background-color: #333333; -fx-padding: 20px; -fx-border-radius: 10px; -fx-background-radius: 10px; -fx-pref-height: 400px; -fx-min-width: 250px");

vbox.setAlignment(Pos.CENTER);

return vbox;

}

private void openBeginnerPlanPage(Stage primaryStage) {

// Create the main container for the new page

VBox mainContainer = new VBox(20);

mainContainer.setAlignment(Pos.TOP\_CENTER);

mainContainer.setStyle("-fx-background-color: #333333;");

// Create a ScrollPane

ScrollPane scrollPane = new ScrollPane();

scrollPane.setFitToWidth(true);

scrollPane.setContent(mainContainer);

// Create the header section

VBox header = new VBox(10);

header.setAlignment(Pos.CENTER);

header.setStyle("-fx-background-color: #000000; -fx-padding: 20px;");

Label headerTitle = new Label("BEGINNER WORKOUT PLAN");

headerTitle.setFont(new Font("Arial", 36));

headerTitle.setStyle("-fx-text-fill: #00C958; -fx-font-weight: bold;");

Label subTitle = new Label("FOR 2024");

subTitle.setFont(new Font("Arial", 18));

subTitle.setStyle("-fx-text-fill: #00C958;");

header.getChildren().addAll(headerTitle, subTitle);

// Create the weekday schedule header

Label weekdayScheduleLabel = new Label("WEEKLY SCHEDULE");

weekdayScheduleLabel.setFont(new Font("Arial", 24));

weekdayScheduleLabel.setStyle("-fx-background-color: #000000; -fx-text-fill: #00C958; -fx-padding: 10px; -fx-alignment: center;");

weekdayScheduleLabel.setMaxWidth(Double.MAX\_VALUE);

weekdayScheduleLabel.setAlignment(Pos.CENTER);

// Create the schedule grid

GridPane scheduleGrid = new GridPane();

scheduleGrid.setAlignment(Pos.CENTER);

scheduleGrid.setHgap(10);

scheduleGrid.setVgap(10);

scheduleGrid.setStyle("-fx-padding: 20px;");

// Set column constraints for three columns

ColumnConstraints col1 = new ColumnConstraints();

col1.setPercentWidth(33.33); // 33.33% width for each column

ColumnConstraints col2 = new ColumnConstraints();

col2.setPercentWidth(33.33);

ColumnConstraints col3 = new ColumnConstraints();

col3.setPercentWidth(33.33);

scheduleGrid.getColumnConstraints().addAll(col1, col2, col3);

// All day Tasks.....

String[] days = {"MONDAY - Full-Body Workout", "TUESDAY - Active Rest Day", "WEDNESDAY - Full-Body Workout", "THURSDAY - Active Rest Day", "FRIDAY - Full-Body Workout", "SATURDAY - Rest Day", "SUNDAY - Rest Day"};

String[] Montasks = {"Warmup: 5–10 minutes of light cardio (e.g., marching in place)", "Core: Standing march", "Chest: Stability ball dumbbell press", "Back: Seated cable row", "Shoulders: Seated stability ball military press", "Legs: Ball squat", "Triceps: Single-leg triceps pushdown", "Biceps: Single-leg dumbbell curl", "Cooldown: 5–10 minutes of stretching"};

String[] Tuetasks = {"Take it easy! Rest or do light activities like walking or yoga"};

String[] Wedtasks = {"Warmup: 5–10 minutes", "Core: Plank", "Back: Stability ball dumbbell row", "Chest: Push-up on stability ball", "Shoulders: Single-leg dumbbell scaption", "Legs: Walking lunge", "Biceps: Single-leg biceps cable curl", "Triceps: Stability ball triceps extension", "Cooldown: 5–10 minutes of stretching"};

String[] Thurtasks = {"Rest or engage in light movement"};

String[] Fritasks = {"Warmup: 5–10 minutes", "Core: Floor prone cobra","Legs: Alternating step-up","Chest: Stability ball dumbbell fly","Shoulders: Shoulder shrug with dumbbells or resistance bands", "Cooldown: 5–10 minutes of stretching"};

String[] Sattasks = {"Resting is an essential part of any\nworkout routine. Two days of rest\ncan help your muscles recover and\nprevent overtraining. Use this time\nto relax, stretch,and recharge.\nRemember, consistency and\nbalance are key to achieving\nyour fitness goals! 💪🌟 "};

String[] Suntasks = {"Resting is an essential part of any\nworkout routine. Two days of rest\ncan help your muscles recover and\nprevent overtraining. Use this time\nto relax, stretch,and recharge.\nRemember, consistency and\nbalance are key to achieving\nyour fitness goals! 💪🌟 "};

// Add day labels and checklists to the grid

for (int i = 0; i < days.length; i++) {

Label dayLabel = new Label(days[i]);

dayLabel.setStyle("-fx-background-color: #00C958; -fx-text-fill: white; -fx-font-weight: bold; -fx-padding: 10px;");

dayLabel.setMaxWidth(Double.MAX\_VALUE);

dayLabel.setAlignment(Pos.CENTER);

VBox dayColumn = new VBox(20, dayLabel);

String[] tasks;

switch(i) {

case 0: tasks = Montasks; break;

case 1: tasks = Tuetasks; break;

case 2: tasks = Wedtasks; break;

case 3: tasks = Thurtasks; break;

case 4: tasks = Fritasks; break;

case 5: tasks = Sattasks; break;

case 6: tasks = Suntasks; break;

default: tasks = new String[0]; break;

}

for (String task : tasks) {

CheckBox taskCheckBox = new CheckBox(task);

// taskCheckBox.setStyle("-fx-text-fill: #FFFFFF; -fx-font-size: 14px;");

taskCheckBox.setStyle("-fx-text-fill: white; -fx-font-size: 16px; -fx-font-weight: bold;");

taskCheckBox.setOnMouseEntered(e -> taskCheckBox.setStyle("-fx-text-fill: #00C958; -fx-font-size: 16px; -fx-font-weight: bold;"));

taskCheckBox.setOnMouseExited(e -> taskCheckBox.setStyle("-fx-text-fill: white; -fx-font-size: 16px; -fx-font-weight: bold;"));

dayColumn.getChildren().add(taskCheckBox);

}

scheduleGrid.add(dayColumn, i % 3, i / 3); // Adjust to have 3 columns

}

// Create a button to go back to the main scene

Button backButton = new Button("Back");

backButton.setStyle("-fx-background-color: #00C958; -fx-text-fill: white; -fx-font-size: 16px; -fx-font-weight: bold;");

backButton.setOnMouseEntered(e -> backButton.setStyle("-fx-background-color: white; -fx-text-fill: #00C958; -fx-font-size: 16px; -fx-font-weight: bold;"));

backButton.setOnMouseExited(e -> backButton.setStyle("-fx-background-color: #00C958; -fx-text-fill: white; -fx-font-size: 16px; -fx-font-weight: bold;"));

backButton.setOnAction(event -> primaryStage.setScene(mainScene));

// Assemble all parts in the main container

mainContainer.getChildren().addAll(header, weekdayScheduleLabel, scheduleGrid, backButton);

// Create a new scene with the scroll pane containing the main container

Scene newScene = new Scene(scrollPane, 1200, 1020);

// Set the new scene on the primary stage

primaryStage.setScene(newScene);

}

private void openIntermediatePlanPage(Stage primaryStage) {

// Create the main container for the new page

VBox mainContainer = new VBox(20);

mainContainer.setAlignment(Pos.TOP\_CENTER);

mainContainer.setStyle("-fx-background-color: #333333;");

// Create a ScrollPane

ScrollPane scrollPane = new ScrollPane();

scrollPane.setFitToWidth(true);

scrollPane.setContent(mainContainer);

// Create the header section

VBox header = new VBox(10);

header.setAlignment(Pos.CENTER);

header.setStyle("-fx-background-color: #000000; -fx-padding: 20px;");

Label headerTitle = new Label("INTERMEDIATE WORKOUT PLAN");

headerTitle.setFont(new Font("Arial", 36));

headerTitle.setStyle("-fx-text-fill: #00C958; -fx-font-weight: bold;");

Label subTitle = new Label("FOR 2024");

subTitle.setFont(new Font("Arial", 18));

subTitle.setStyle("-fx-text-fill: #00C958;");

header.getChildren().addAll(headerTitle, subTitle);

// Create the weekday schedule header

Label weekdayScheduleLabel = new Label("WEEKLY SCHEDULE");

weekdayScheduleLabel.setFont(new Font("Arial", 24));

weekdayScheduleLabel.setStyle("-fx-background-color: #000000; -fx-text-fill: #00C958; -fx-padding: 10px; -fx-alignment: center;");

weekdayScheduleLabel.setMaxWidth(Double.MAX\_VALUE);

weekdayScheduleLabel.setAlignment(Pos.CENTER);

// Create the schedule grid

GridPane scheduleGrid = new GridPane();

scheduleGrid.setAlignment(Pos.CENTER);

scheduleGrid.setHgap(10);

scheduleGrid.setVgap(10);

scheduleGrid.setStyle("-fx-padding: 20px;");

// Set column constraints for three columns

ColumnConstraints col1 = new ColumnConstraints();

col1.setPercentWidth(33.33); // 33.33% width for each column

ColumnConstraints col2 = new ColumnConstraints();

col2.setPercentWidth(33.33);

ColumnConstraints col3 = new ColumnConstraints();

col3.setPercentWidth(33.33);

scheduleGrid.getColumnConstraints().addAll(col1, col2, col3);

// All day Tasks.....

String[] days = {"MONDAY - Full-Body Workout", "TUESDAY - Active Rest Day", "WEDNESDAY - Full-Body Workout", "THURSDAY - Active Rest Day", "FRIDAY - Full-Body Workout", "SATURDAY - Rest Day", "SUNDAY - Rest Day"};

String[] Montasks = {"Warmup: 5–10 minutes of light cardio (e.g., jogging in place)", "Chest: Bench press (dumbbells or barbell)", "Back: Bent-over rows", "Shoulders: Overhead press", "Biceps: Bicep curls", "Triceps: Tricep dips", "Core: Planks", "Cooldown: 5–10 minutes of stretching"};

String[] Tuetasks = {"Rest or engage in light activities like walking or yoga."};

String[] Wedtasks = {"Warmup: 5–10 minutes", "Legs: Squats (weighted or bodyweight)", "Hamstrings: Romanian deadlifts", "Glutes: Hip thrusts", "Calves: Calf raises", "Core: Russian twists", "Cooldown: 5–10 minutes of stretching"};

String[] Thurtasks = {"Rest or do light movement."};

String[] Fritasks = {"Warmup: 5–10 minutes", "Cardio: Choose your favorite: running, cycling, or jumping jacks","Core: Bicycle crunches, leg raises","Cooldown: 5–10 minutes of stretching"};

String[] Sattasks = {"Resting is an essential part of any\nworkout routine. Two days of rest\ncan help your muscles recover and\nprevent overtraining. Use this time\nto relax, stretch,and recharge.\nRemember, consistency and\nbalance are key to achieving\nyour fitness goals! 💪🌟 "};

String[] Suntasks = {"Resting is an essential part of any\nworkout routine. Two days of rest\ncan help your muscles recover and\nprevent overtraining. Use this time\nto relax, stretch,and recharge.\nRemember, consistency and\nbalance are key to achieving\nyour fitness goals! 💪🌟 "};

// Add day labels and checklists to the grid

for (int i = 0; i < days.length; i++) {

Label dayLabel = new Label(days[i]);

dayLabel.setStyle("-fx-background-color: #00C958; -fx-text-fill: white; -fx-font-weight: bold; -fx-padding: 10px;");

dayLabel.setMaxWidth(Double.MAX\_VALUE);

dayLabel.setAlignment(Pos.CENTER);

VBox dayColumn = new VBox(20, dayLabel);

String[] tasks;

switch(i) {

case 0: tasks = Montasks; break;

case 1: tasks = Tuetasks; break;

case 2: tasks = Wedtasks; break;

case 3: tasks = Thurtasks; break;

case 4: tasks = Fritasks; break;

case 5: tasks = Sattasks; break;

case 6: tasks = Suntasks; break;

default: tasks = new String[0]; break;

}

for (String task : tasks) {

CheckBox taskCheckBox = new CheckBox(task);

// taskCheckBox.setStyle("-fx-text-fill: #FFFFFF; -fx-font-size: 14px;");

taskCheckBox.setStyle("-fx-text-fill: white; -fx-font-size: 16px; -fx-font-weight: bold;");

taskCheckBox.setOnMouseEntered(e -> taskCheckBox.setStyle("-fx-text-fill: #00C958; -fx-font-size: 16px; -fx-font-weight: bold;"));

taskCheckBox.setOnMouseExited(e -> taskCheckBox.setStyle("-fx-text-fill: white; -fx-font-size: 16px; -fx-font-weight: bold;"));

dayColumn.getChildren().add(taskCheckBox);

}

scheduleGrid.add(dayColumn, i % 3, i / 3); // Adjust to have 3 columns

}

// Create a button to go back to the main scene

Button backButton = new Button("Back");

backButton.setStyle("-fx-background-color: #00C958; -fx-text-fill: white; -fx-font-size: 16px; -fx-font-weight: bold;");

backButton.setOnMouseEntered(e -> backButton.setStyle("-fx-background-color: white; -fx-text-fill: #00C958; -fx-font-size: 16px; -fx-font-weight: bold;"));

backButton.setOnMouseExited(e -> backButton.setStyle("-fx-background-color: #00C958; -fx-text-fill: white; -fx-font-size: 16px; -fx-font-weight: bold;"));

backButton.setOnAction(event -> primaryStage.setScene(mainScene));

// Assemble all parts in the main container

mainContainer.getChildren().addAll(header, weekdayScheduleLabel, scheduleGrid, backButton);

// Create a new scene with the scroll pane containing the main container

Scene newScene = new Scene(scrollPane, 1200, 1020);

// Set the new scene on the primary stage

primaryStage.setScene(newScene);

}

private void openAdvancedPlanPage(Stage primaryStage) {

// Create the main container for the new page

VBox mainContainer = new VBox(20);

mainContainer.setAlignment(Pos.TOP\_CENTER);

mainContainer.setStyle("-fx-background-color: #333333;");

// Create a ScrollPane

ScrollPane scrollPane = new ScrollPane();

scrollPane.setFitToWidth(true);

scrollPane.setContent(mainContainer);

// Create the header section

VBox header = new VBox(10);

header.setAlignment(Pos.CENTER);

header.setStyle("-fx-background-color: #000000; -fx-padding: 20px;");

Label headerTitle = new Label("ADVANCED WORKOUT PLAN");

headerTitle.setFont(new Font("Arial", 36));

headerTitle.setStyle("-fx-text-fill: #00C958; -fx-font-weight: bold;");

Label subTitle = new Label("FOR 2024");

subTitle.setFont(new Font("Arial", 18));

subTitle.setStyle("-fx-text-fill: #00C958;");

header.getChildren().addAll(headerTitle, subTitle);

// Create the weekday schedule header

Label weekdayScheduleLabel = new Label("WEEKLY SCHEDULE");

weekdayScheduleLabel.setFont(new Font("Arial", 24));

weekdayScheduleLabel.setStyle("-fx-background-color: #000000; -fx-text-fill: #00C958; -fx-padding: 10px; -fx-alignment: center;");

weekdayScheduleLabel.setMaxWidth(Double.MAX\_VALUE);

weekdayScheduleLabel.setAlignment(Pos.CENTER);

// Create the schedule grid

GridPane scheduleGrid = new GridPane();

scheduleGrid.setAlignment(Pos.CENTER);

scheduleGrid.setHgap(10);

scheduleGrid.setVgap(10);

scheduleGrid.setStyle("-fx-padding: 20px;");

// Set column constraints for three columns

ColumnConstraints col1 = new ColumnConstraints();

col1.setPercentWidth(33.33); // 33.33% width for each column

ColumnConstraints col2 = new ColumnConstraints();

col2.setPercentWidth(33.33);

ColumnConstraints col3 = new ColumnConstraints();

col3.setPercentWidth(33.33);

scheduleGrid.getColumnConstraints().addAll(col1, col2, col3);

// All day Tasks.....

String[] days = {"MONDAY - Full-Body Workout", "TUESDAY - Active Rest Day", "WEDNESDAY - Full-Body Workout", "THURSDAY - Active Rest Day", "FRIDAY - Full-Body Workout", "SATURDAY - Rest Day", "SUNDAY - Rest Day"};

String[] Montasks = {"Warmup: 5–10 minutes of light cardio (e.g., brisk walking or jogging)", "Core: Standing march (10 repetitions)","Chest: Stability ball dumbbell press (10 repetitions)", "Back: Seated cable row (10 repetitions)", "Shoulders: Seated stability ball military press (10 repetitions)", "Legs: Ball squat (10 repetitions)", "Triceps: Single-leg triceps pushdown (10 repetitions)", "Biceps: Single-leg dumbbell curl (10 repetitions)", "Cooldown: 5–10 minutes of stretching"};

String[] Tuetasks = {"Take it easy today! Engage in light activities like walking or gentle stretching."};

String[] Wedtasks = {"Warmup: 5–10 minutes of light cardio (e.g., brisk walking or jogging)", "Core: Plank (hold for 30 seconds)", "Back: Stability ball dumbbell row (10 repetitions)", "Chest: Push-up on stability ball (10 repetitions)", "Shoulders: Single-leg dumbbell scaption (10 repetitions)", "Legs: Walking lunge (10 repetitions per leg)", "Biceps: Single-leg biceps cable curl (10 repetitions)", "Triceps: Stability ball triceps extension (10 repetitions)", "Cooldown: 5–10 minutes of stretching"};

String[] Thurtasks = {"Another day of light activity. Listen to your body and rest as needed."};

String[] Fritasks = {"Warmup: 5–10 minutes of light cardio (e.g., brisk walking or jogging)", "Core: Floor prone cobra (10 repetitions)", "Legs: Alternating step-up (10 repetitions per leg)", "Chest: Stability ball dumbbell fly (10 repetitions)", "Shoulders: Shoulder shrug with dumbbells or resistance bands (10 repetitions)", "Cooldown: 5–10 minutes of stretching"};

String[] Sattasks = {"Resting is an essential part of any\nworkout routine. Two days of rest\ncan help your muscles recover and\nprevent overtraining. Use this time\nto relax, stretch,and recharge.\nRemember, consistency and\nbalance are key to achieving\nyour fitness goals! 💪🌟 "};

String[] Suntasks = {"Resting is an essential part of any\nworkout routine. Two days of rest\ncan help your muscles recover and\nprevent overtraining. Use this time\nto relax, stretch,and recharge.\nRemember, consistency and\nbalance are key to achieving\nyour fitness goals! 💪🌟 "};

// Add day labels and checklists to the grid

for (int i = 0; i < days.length; i++) {

Label dayLabel = new Label(days[i]);

dayLabel.setStyle("-fx-background-color: #00C958; -fx-text-fill: white; -fx-font-weight: bold; -fx-padding: 10px;");

dayLabel.setMaxWidth(Double.MAX\_VALUE);

dayLabel.setAlignment(Pos.CENTER);

VBox dayColumn = new VBox(20, dayLabel);

String[] tasks;

switch(i) {

case 0: tasks = Montasks; break;

case 1: tasks = Tuetasks; break;

case 2: tasks = Wedtasks; break;

case 3: tasks = Thurtasks; break;

case 4: tasks = Fritasks; break;

case 5: tasks = Sattasks; break;

case 6: tasks = Suntasks; break;

default: tasks = new String[0]; break;

}

for (String task : tasks) {

CheckBox taskCheckBox = new CheckBox(task);

taskCheckBox.setStyle("-fx-text-fill: white; -fx-font-size: 16px; -fx-font-weight: bold;");

taskCheckBox.setOnMouseEntered(e -> taskCheckBox.setStyle("-fx-text-fill: #00C958; -fx-font-size: 16px; -fx-font-weight: bold;"));

taskCheckBox.setOnMouseExited(e -> taskCheckBox.setStyle("-fx-text-fill: white; -fx-font-size: 16px; -fx-font-weight: bold;"));

dayColumn.getChildren().add(taskCheckBox);

}

scheduleGrid.add(dayColumn, i % 3, i / 3); // Adjust to have 3 columns

}

// Create a button to go back to the main scene

Button backButton = new Button("Back");

backButton.setStyle("-fx-background-color: #00C958; -fx-text-fill: white; -fx-font-size: 16px; -fx-font-weight: bold;");

backButton.setOnMouseEntered(e -> backButton.setStyle("-fx-background-color: white; -fx-text-fill: #00C958; -fx-font-size: 16px; -fx-font-weight: bold;"));

backButton.setOnMouseExited(e -> backButton.setStyle("-fx-background-color: #00C958; -fx-text-fill: white; -fx-font-size: 16px; -fx-font-weight: bold;"));

backButton.setOnAction(event -> primaryStage.setScene(mainScene));

// Assemble all parts in the main container

mainContainer.getChildren().addAll(header, weekdayScheduleLabel, scheduleGrid, backButton);

// Create a new scene with the scroll pane containing the main container

Scene newScene = new Scene(scrollPane, 1200, 1020);

// Set the new scene on the primary stage

primaryStage.setScene(newScene);

}

}

# FitnessChallenge.java

package application;

import javafx.geometry.Pos;

import javafx.scene.Scene;

import javafx.scene.control.Button;

import javafx.scene.control.Label;

import javafx.scene.layout.HBox;

import javafx.scene.layout.VBox;

import javafx.scene.text.Font;

import javafx.stage.Stage;

import java.util.Random;

import java.io.\*;

import static application.ScreenShiftUtils.\*;

import static application.GYM\_BUDDY.\*;

public class FitnessChallenge {

private Stage ChallengeStage;

private Scene mainScene;

public static int POINTS = 0;

public static int x = 0;

public void start(Stage ChallengeStage) {

this.ChallengeStage = ChallengeStage;

// Create an HBox layout for the top bar

HBox topBar = new HBox();

topBar.setAlignment(Pos.TOP\_LEFT);

// Create home button

Button homeButton = new Button("DASHBOARD");

homeButton.setStyle("-fx-background-color: #00C958; -fx-text-fill: white; -fx-font-size: 14px; -fx-font-weight: bold;");

// Mouse entered style

homeButton.setOnMouseEntered(e -> {

homeButton.setStyle("-fx-background-color: white; -fx-text-fill: #00C958; -fx-font-size: 14px; -fx-font-weight: bold;");

});

// Mouse exited style

homeButton.setOnMouseExited(e -> {

homeButton.setStyle("-fx-background-color: #00C958; -fx-text-fill: white; -fx-font-size: 14px; -fx-font-weight: bold;");

});

homeButton.setOnAction(event -> {

showDashboard(ChallengeStage); // Pass the current stage to close it

});

topBar.getChildren().add(homeButton);

topBar.setStyle("-fx-padding: 20px;");

// Create heading label

Label headingLabel = new Label("Your Fitness Challenge");

headingLabel.setFont(new Font("Arial", 30));

headingLabel.setStyle("-fx-text-fill: white;");

// Create an array of challenges

VBox[] challenges = new VBox[] {

createChallenge("10-minute brisk walk - 20 POINTS", 5),

createChallenge("20-minute yoga session - 20 POINTS", 10),

createChallenge("20-minute bodyweight circuit (push-ups, squats, lunges) - 20 POINTS", 20),

createChallenge("5-minute meditation - 20 POINTS", 5),

createChallenge("30-minute bike ride - 20 POINTS", 20),

createChallenge("20-minute jog - 20 POINTS", 20),

createChallenge("10 push-ups - 20 POINTS", 5),

createChallenge("20 sit-ups - 20 POINTS", 5),

createChallenge("20 squats - 20 POINTS", 5),

createChallenge("1-minute plank - 20 POINTS", 5),

createChallenge("20-minute stretching session - 20 POINTS", 10),

createChallenge("20 burpees - 20 POINTS", 10),

createChallenge("30 jumping jacks - 20 POINTS", 5),

createChallenge("10-minute dance session - 20 POINTS", 5),

createChallenge("1-hour hike - 20 POINTS", 20),

createChallenge("10 pull-ups - 20 POINTS", 10),

createChallenge("10-minute jump rope - 20 POINTS", 10),

createChallenge("20 lunges (each leg) - 20 POINTS", 10),

createChallenge("50-meter sprint - 20 POINTS", 10),

createChallenge("5 flights of stairs - 20 POINTS", 5),

createChallenge("10 tricep dips - 20 POINTS", 5),

createChallenge("20-minute swim - 20 POINTS", 20),

createChallenge("10-minute mindfulness walk - 20 POINTS", 5),

createChallenge("20-minute HIIT workout - 20 POINTS", 20),

createChallenge("20-minute Pilates session - 20 POINTS", 20),

createChallenge("10-minute core workout (crunches, leg raises, bicycles) - 20 POINTS", 10),

createChallenge("20 calf raises - 20 POINTS", 5),

createChallenge("30-second side planks (each side) - 20 POINTS", 5),

createChallenge("10-minute stair climbing - 20 POINTS", 10),

createChallenge("30-second mountain climbers - 20 POINTS", 5),

createChallenge("10 minutes of foam rolling - 20 POINTS", 5),

createChallenge("20-minute resistance band workout - 20 POINTS", 20),

createChallenge("10 kettlebell swings - 20 POINTS", 5),

createChallenge("5-minute breathing exercises - 20 POINTS", 5),

createChallenge("20-minute shadow boxing - 20 POINTS", 20),

createChallenge("1-mile walk/run - 20 POINTS", 20),

createChallenge("30 squats with a jump - 20 POINTS", 10),

createChallenge("10-minute tai chi session - 20 POINTS", 10),

createChallenge("20-minute rowing session - 20 POINTS", 20),

createChallenge("50 jumping jacks - 20 POINTS", 10),

createChallenge("20-minute stair workout - 20 POINTS", 20),

createChallenge("20-minute elliptical session - 20 POINTS", 20),

createChallenge("5-minute balance exercises - 20 POINTS", 5),

createChallenge("20-minute strength training (weights) - 20 POINTS", 20),

createChallenge("10 minutes of boxing/kickboxing - 20 POINTS", 10),

createChallenge("1-minute wall sit - 20 POINTS", 5),

createChallenge("10 burpees with a push-up - 20 POINTS", 10),

createChallenge("20 toe touches - 20 POINTS", 5),

createChallenge("20-minute light jogging in place - 20 POINTS", 10),

createChallenge("30-minute mixed cardio (bike, elliptical, rower) - 20 POINTS", 20)

};

// Randomly select a challenge

Random random = new Random();

VBox selectedChallenge = challenges[random.nextInt(challenges.length)];

// Create a gray box to contain the selected challenge

VBox grayBox = new VBox(selectedChallenge);

grayBox.setStyle("-fx-background-color: black; -fx-padding: 20px; -fx-border-radius: 10px; -fx-background-radius: 10px; -fx-max-width: 900px");

grayBox.setAlignment(Pos.CENTER);

// Create "Done" button

Button doneButton = new Button("DONE");

doneButton.setStyle("-fx-background-color: #00C958; -fx-text-fill: white; -fx-font-size: 18px; -fx-font-weight: bold;");

doneButton.setOnMouseEntered(e -> {

doneButton.setStyle("-fx-background-color: white; -fx-text-fill: #00C958; -fx-font-size: 18px; -fx-font-weight: bold;");

});

doneButton.setOnMouseExited(e -> {

doneButton.setStyle("-fx-background-color: #00C958; -fx-text-fill: white; -fx-font-size: 18px; -fx-font-weight: bold;");

});

doneButton.setOnAction(event -> {

updatePoints(user, x);

showDashboard(ChallengeStage);

});

grayBox.getChildren().add(doneButton);

// Create a VBox to hold the top bar, heading, and the gray box

VBox outerVBox = new VBox(20, topBar, headingLabel, grayBox);

outerVBox.setAlignment(Pos.CENTER);

outerVBox.setStyle("-fx-background-color: black;");

// Create the main scene with the outer VBox layout

mainScene = new Scene(outerVBox, 700, 400);

// Set the title of the stage (window) and add the main scene to it

ChallengeStage.setTitle("Fitness Challenge");

ChallengeStage.setScene(mainScene);

ChallengeStage.show();

}

private VBox createChallenge(String description, int points) {

// Create description label

x = points;

Label descriptionLabel = new Label(description);

descriptionLabel.setWrapText(true);

descriptionLabel.setMaxWidth(400); // Adjust the width as needed

descriptionLabel.setPrefHeight(600); // Set the preferred height to 600 pixels

descriptionLabel.setFont(new Font("Arial", 18));

descriptionLabel.setStyle("-fx-text-fill: black; -fx-font-weight: bold;");

// Create a green box for description

VBox descriptionBox = new VBox(descriptionLabel);

descriptionBox.setStyle("-fx-background-color: #00C958; -fx-border-radius: 10px; -fx-background-radius: 10px; -fx-padding: 20px;");

descriptionBox.setAlignment(Pos.CENTER); // Center the content inside the green box

// Create VBox to hold description box and button

VBox vbox = new VBox(descriptionBox);

vbox.setStyle("-fx-background-color: #333333; -fx-padding: 20px; -fx-border-radius: 10px; -fx-background-radius: 10px;");

vbox.setAlignment(Pos.CENTER);

return vbox;

}

private void updatePoints(String username, int pointsToAdd) {

File file = new File(filePath);

StringBuilder newData = new StringBuilder();

boolean userFound = false;

try (BufferedReader reader = new BufferedReader(new FileReader(file))) {

String line;

while ((line = reader.readLine()) != null) {

String[] parts = line.split(",");

if (parts[0].equals(username)) {

int currentPoints = Integer.parseInt(parts[2]); // Assuming points are stored at index 2

int newPoints = currentPoints + pointsToAdd;

POINTS = newPoints;

// Corrected index usage and appending format

newData.append(parts[0]).append(",").append(parts[1]).append(",").append(newPoints).append(",").append(parts[3]).append(System.lineSeparator());

userFound = true;

} else {

newData.append(line).append(System.lineSeparator());

}

}

} catch (IOException e) {

e.printStackTrace();

}

if (!userFound) {

newData.append(username).append(",").append("password").append(",").append(pointsToAdd).append(",").append("default position").append(System.lineSeparator());

}

try (BufferedWriter writer = new BufferedWriter(new FileWriter(file))) {

writer.write(newData.toString());

} catch (IOException e) {

e.printStackTrace();

}

}

}

# ScreenShiftUtils.java

package application;

import javafx.stage.Stage;

public class ScreenShiftUtils {

public static void showDashboard(Stage currentStage) {

try {

Dashboard dashboard = new Dashboard();

Stage newStage = new Stage();

dashboard.start(newStage);

currentStage.close(); // Close the current stage

} catch (Exception e) {

e.printStackTrace();

}

}

public static void showLeaderboard(Stage currentStage) {

try {

Leaderboard leaderboard = new Leaderboard();

Stage newStage = new Stage();

leaderboard.start(newStage, Leaderboard.user);

currentStage.close(); // Close the current stage

} catch (Exception e) {

e.printStackTrace();

}

}

public static void showWorkoutPlanSuggestor(Stage currentStage) {

try {

WorkoutPlanSuggestor workoutPlanSuggestor = new WorkoutPlanSuggestor();

workoutPlanSuggestor.start(currentStage);

} catch (Exception e) {

e.printStackTrace();

}

}

public static void showFitnessChallenge(Stage currentStage) {

try {

FitnessChallenge fitnessChallenge = new FitnessChallenge();

fitnessChallenge.start(currentStage);

} catch (Exception e) {

e.printStackTrace();

}

}

public static void showBMIandCalorie(Stage currentStage) {

try {

BMIandCalorie bMIandCalorie = new BMIandCalorie();

Stage newStage = new Stage();

bMIandCalorie.start(newStage);

currentStage.close(); // Close the current stage

} catch (Exception e) {

e.printStackTrace();

}

}

public static void showRegistration(Stage currentStage) {

try {

GYM\_BUDDY gymBuddy = new GYM\_BUDDY();

Stage newStage = new Stage();

gymBuddy.showRegistrationForm(newStage);

currentStage.close();

} catch (Exception e) {

e.printStackTrace();

}

}

public static void showLogin(Stage currentStage) {

try {

GYM\_BUDDY gymBuddy = new GYM\_BUDDY();

Stage newStage = new Stage();

gymBuddy.showLoginForm(newStage);

currentStage.close();

} catch (Exception e) {

e.printStackTrace();

}

}

}

# Styles.css

.root {

-fx-background-color: black;

}

.input-field, .label, .choice-box {

-fx-text-fill: white;

-fx-font-size: 16px;

-fx-font-weight: bold;

}

.choice-box, .input-field {

-fx-background-color: #333333;

-fx-pref-height: 20px;

-fx-border-color: #00C958;

-fx-pref-width: 300px;

}

.choice-box .menu-item {

-fx-background-color: #333333; /\* Set background color for dropdown items \*/

-fx-text-fill: #FFFFFF; /\* Set text color for dropdown items \*/

}

.choice-box .menu-item:hover {

-fx-background-color: #00C958; /\* Set background color on hover \*/

-fx-text-fill: #000000; /\* Set text color on hover \*/

}

.choice-box:hover {

-fx-background-color: #444444; /\* Set background color on hover \*/

}

.calculate-button {

-fx-background-color: #00C958;

-fx-text-fill: white;

-fx-font-size: 14px;

-fx-font-weight: bold;

}

.calculate-button:hover {

-fx-text-fill: #00C958;

-fx-background-color: white;

}

.result-label {

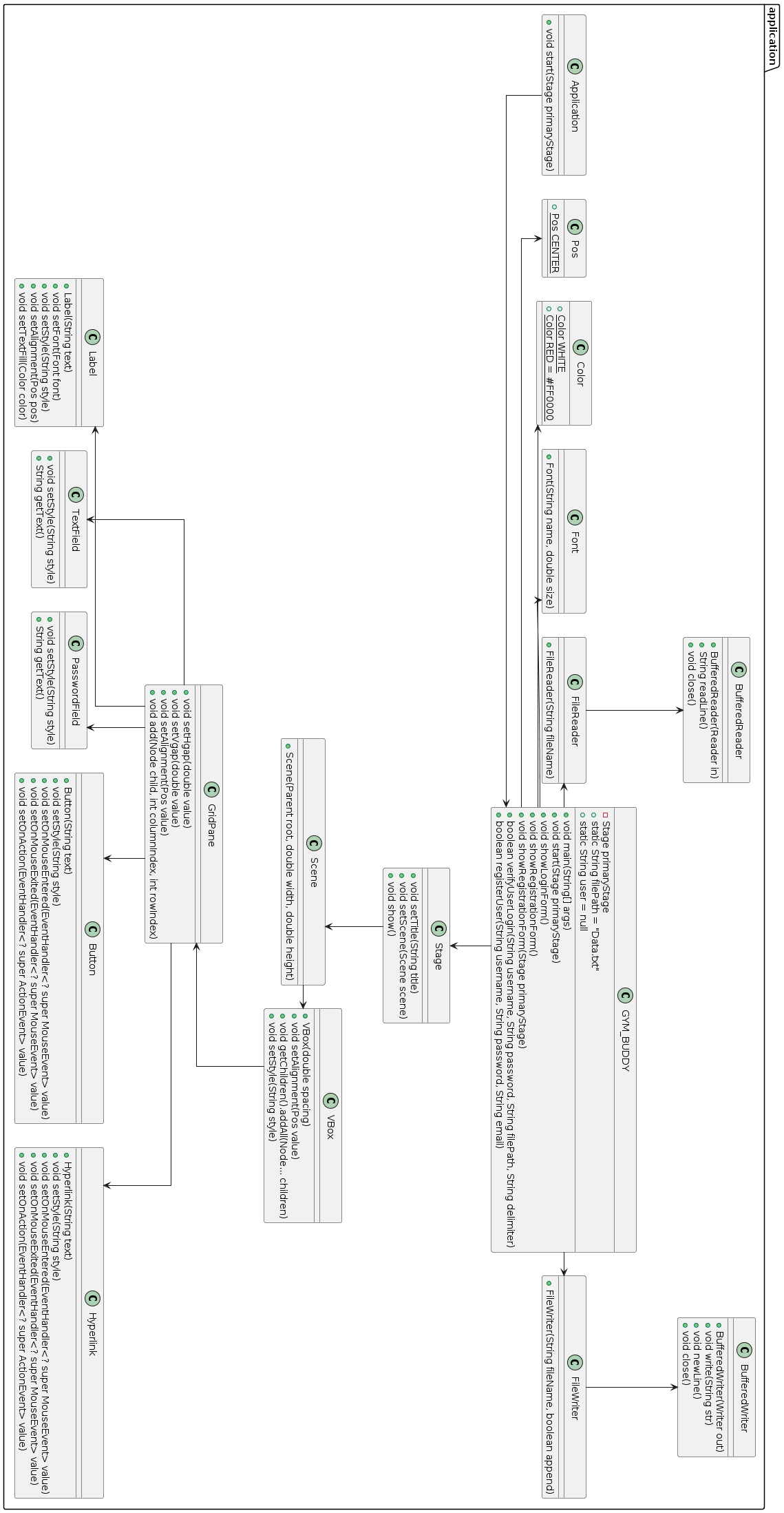
-fx-text-fill: #00C958;

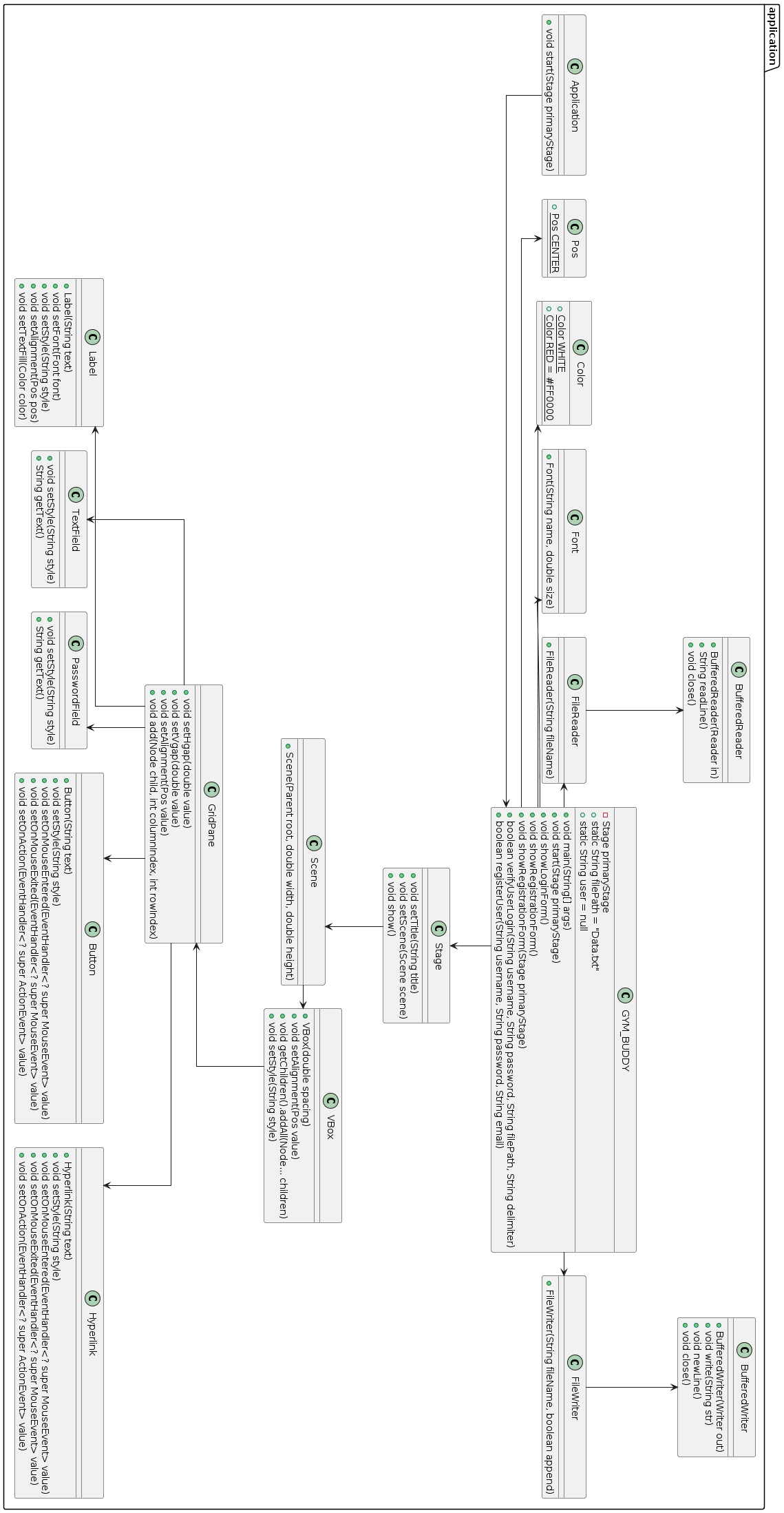
-fx-font-size: 16px;

-fx-font-weight: bold;

-fx-border-color: transparent; /\* Remove the border \*/

}

**UML Diagram**

****