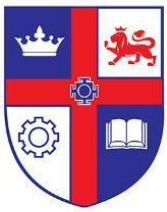
LONDON METROPOLITAN UNIVERSITY



Islington college
(इस्लिङ्टन कलेज)

CS4001NI Programming

60% Individual Coursework

2025 Spring

Student Name: Dilip Shrestha
London Met ID: 24046750
College ID: NP01CP4A240021
Group: L1C1

Assignment Due Date: Friday, May 16, 2025
Assignment Submission Date: Friday, May 16, 2025

I confirm that I understand my coursework needs to be submitted online via MySecondTeacher under the relevant module page before the deadline in order for my assignment to be accepted and marked. I am fully aware that late submissions will be treated as non-submission and a marks of zero will be awarded.

24046750 Dilip Shrestha doc (AutoRecovered) 1.docx

Islington College,Nepal

Document Details

Submission ID
trn:oid:::3618:96193547

Submission Date
May 16, 2025, 12:16 PM GMT+5:45

Download Date
May 16, 2025, 12:20 PM GMT+5:45

File Name
24046750 Dilip Shrestha doc (AutoRecovered) 1.docx

File Size
99.9 KB

138 Pages

14,524 Words

84,514 Characters

7% Overall Similarity

The combined total of all matches, including overlapping sources, for each database.

Match Groups

- **94** Not Cited or Quoted 6%
  Matches with neither in-text citation nor quotation marks
- **0** Missing Quotations 0%
  Matches that are still very similar to source material
- **13** Missing Citation 1%
  Matches that have quotation marks, but no in-text citation
- **0** Cited and Quoted 0%
  Matches with in-text citation present, but no quotation marks

Top Sources

1% Internet sources
0% Publications
7% Submitted works (Student Papers)

Integrity Flags

0 Integrity Flags for Review

Our system's algorithms look deeply at a document for any inconsistencies that would set it apart from a normal submission. If we notice something strange, we flag it for you to review.

A Flag is not necessarily an indicator of a problem. However, we'd recommend you focus your attention there for further review.

Match Groups

- 94 Not Cited or Quoted 6%
  Matches with neither in-text citation nor quotation marks
- 0 Missing Quotations 0%
  Matches that are still very similar to source material
- 13 Missing Citation 1%
  Matches that have quotation marks, but no in-text citation
- 0 Cited and Quoted 0%
  Matches with in-text citation present, but no quotation marks

Top Sources

1% Internet sources
0% Publications
7% Submitted works (Student Papers)

Top Sources

The sources with the highest number of matches within the submission. Overlapping sources will not be displayed.

1	Submitted works	iacademy on 2025-05-14 — 1%
2	Submitted works	islingtoncollege on 2025-05-16 — 1%
3	Submitted works	Informatics Education Limited on 2009-04-09 — <1%
4	Submitted works	University of Queensland on 2018-11-05 — <1%
5	Submitted works	University of Technology, Sydney on 2006-09-21 — <1%
6	Submitted works	Southern New Hampshire University - Continuing Education on 2020-04-26 — <1%
7	Submitted works	University of Auckland on 2017-08-10 — <1%
8	Submitted works	Singapore Institute of Technology on 2024-02-05 — <1%
9	Submitted works	Central Queensland University on 2023-04-25 — <1%
10	Submitted works	Domain Academy on 2024-12-04 — <1%

Table of Contents

Table of figure

Table of Table

Introduction

This is the module “Programming” coursework assigned by our module leader, Mr. Ujjwal Adhikari. The coding part of this coursework is accomplished by using “BlueJ”, the documentation part is accomplished by using “MS Word”, and the wireframe part of this coursework is achieved by using “Figma”.

The main aim of this course is to create a user-friendly gym membership management system by apply concept of OOP principles, designing program using “swing” and “awt”, storing data using ArrayList and usage of file handling to work with file, use of exception handling to pervert unexpected program termination.

The main objective, this coursework focuses on creating an abstract class named “GymMember” which contains its two subclasses i.e. RegularMember and PremiumMember. Here, GymMember is the parent class/super class, whereas RegularMember and PremiumMember are its two-child class/base class. Each classes contains their own attributes along with constructors, method and accessor method/getter method. Additionally, a class named “GymGUI” which plays the main role for UI contains a “Swing” is the package in java which contains various components like JFrame, JButton, JLabel, JTextField, JTextField, etc. The AWT package enhances the GUI by adding visual elements like colors and fonts, making the interface more appealing and user-friendly. Event handling, implemented through the java.awt.event package, ensures that user actions like button clicks or selections trigger the appropriate responses, such as adding a member or marking attendance. Exception handling is used to pervert unexpected termination of program. File handling is used to make sure that the data entered by users is not lost and can be retrieved whenever needed. This means that the system can save member information to files and read it back, which is important for real-world applications. The use of ArrayList makes it easy to store and manage multiple members at once, and Java’s event handling enables the program to respond to user actions like button clicks.

Java

Java is a high-level OOP language which has the capability of solving real-life scenarios by creating classes and objects. It was released in 1995 as a part of Sun Microsystems Java platform. The syntax is heavily inspired by C and C++, which makes it easier for a lot of programmers to understand it. While standing for an object-oriented language, it promotes modularity and code reuse through classes, inheritance, and polymorphism. Java is used in various fields such as mobile app development (especially Android), web applications, enterprise software, cloud computing, and IoT devices.

OOP

OOP which stands for Object Oriented Programming, is an approach to design and write programs using the concept of class and object. Here object means real life thing/scenario and class means blueprint for making object. There are 4 pillars of OOP: Inheritance, Polymorphism, Encapsulation and Abstraction.

AWT

In java, the first package used for GUI was AWT and it stands for Abstract Window Toolkit. It is platform-dependent which means it display according to graphical theme of OS (Operating System). It is heavyweight component (uses resources of OS). Its package is: java.awt. There are various awt components like: Button, Label, Font, Color, Cursor, TextArea, etc. It is native look and feel.

Swing

Swing is part of Java-Foundation Classes (JFC) that is used to create window-based applications. It is built on the top of AWT API (Application Programming Interface). It is platform-independent which means the components works and display exactly same in all OS (Operating System). It is lightweight component. Its package is: javax.swing.
There are various swing components like: JFrame, JPanel, JLabel, ButtonGroup, etc. It

is pluggable look and feel.

Tools and Technology used

BlueJ

BlueJ is a simple and easy-to-learn tool for learning and writing Java programming. It is widely used by students and beginners due to its simple layout, ensuring that nothing confuses them. In BlueJ, you can write and test Java programs in a friendly environment. It presents your classes and objects visually so that it is easier for you to understand the inner working of your program. One can very easily create new classes and edit the code and immediately see the result. BlueJ supports several other features like error highlighting and debugging in steps, making the process of finding and fixing errors less stressful. All in all, BlueJ is a powerful tool for anyone starting to learn Java programming.

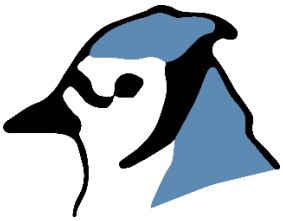


Figure 1: Logo of BlueJ

MS Word

MS Word is also called Microsoft Word, a very popular word processor. It is used for writing letters, reports, essays, and so on. It is easy to use and highly equipped with features like spell check, grammar suggestions, and templates for varied types of documents. You can format font style, size, color, insert pictures, tables, and charts. MS Word also makes saving, printing, and sharing your work simple. Being a common software, most people are well familiar with its basic functions that further facilitate collaborative work on documents. Be it for a student, a teacher, or a working professional, MS Word is a helpful tool for all your writing needs.

Figure 2: Logo of MS Word

Figma

Figma is an online design tool that many use to create UI, website, and app designs. It works right in the user's browser and need not be installed on the system. Figma got big because it supports multiple working on the same file simultaneously, which is highly beneficial for teamwork. We can draw shapes, type in texts, pick a color to apply to objects, or simply drag and drop anything on the canvas It is used by both beginner and professional designers because of its easy learning ability and power-packed features for advanced users.

Figure 3: Logo of Figma

Draw.io

draw.io is a totally free and simple application to create diagrams and flowcharts. We can use it online, and there is no need for signing up or downloading any software. With draw.io, it is very easy to draw diagrams for streamlining project plans, organization charts, or even computer networks. It offers different shapes and icons for drag-and-drop on page. Then, connect shapes through lines to demonstrate relationships or directions. draw.io makes it easy to save these diagrams on your computer or in the cloud and can share with other people whenever needed. It is great for students, teachers, or whoever needs to see an idea visualized.

Figure 4: Logo of Draw.io

GUI Wireframe

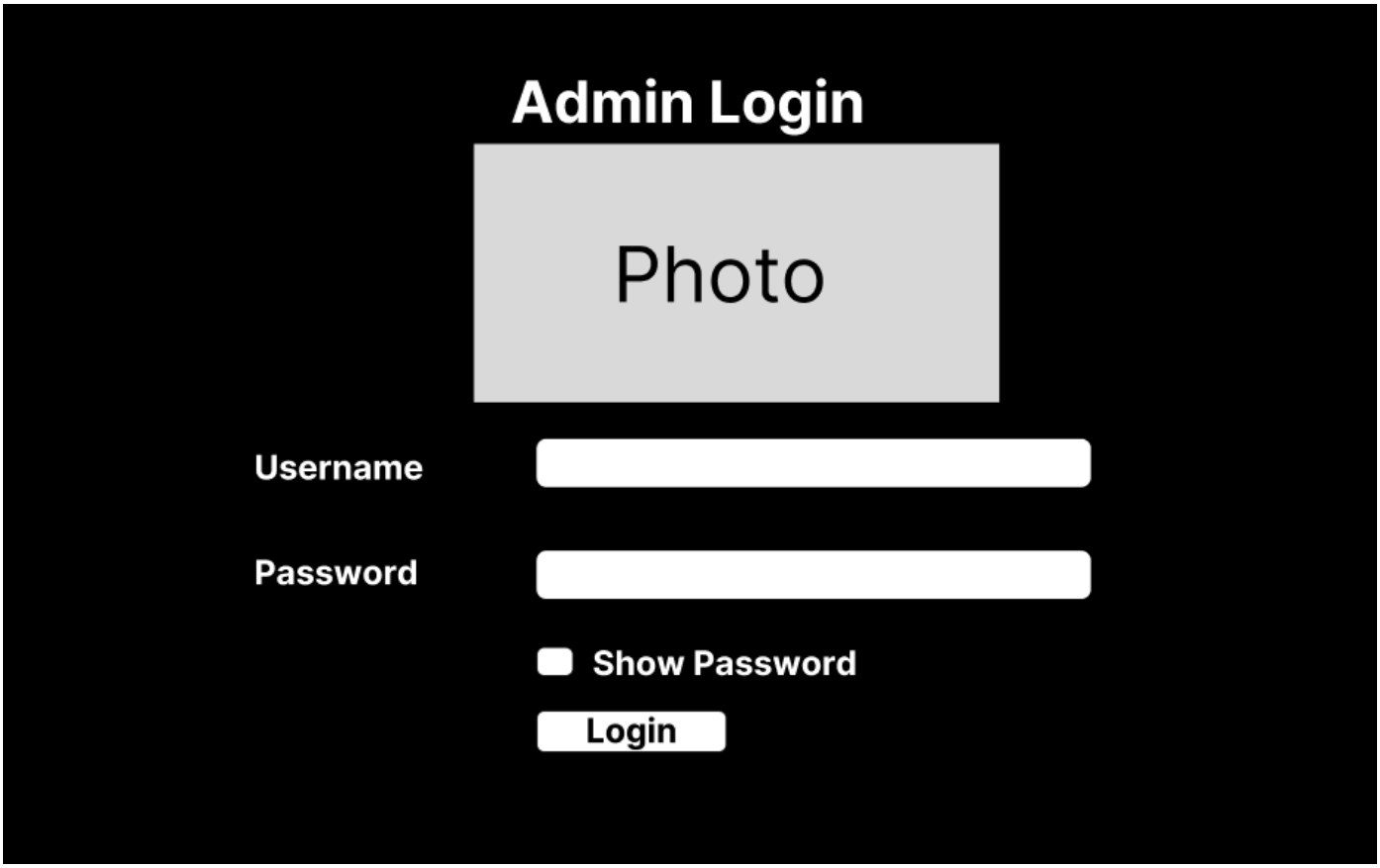


Figure 5: Wireframe login page

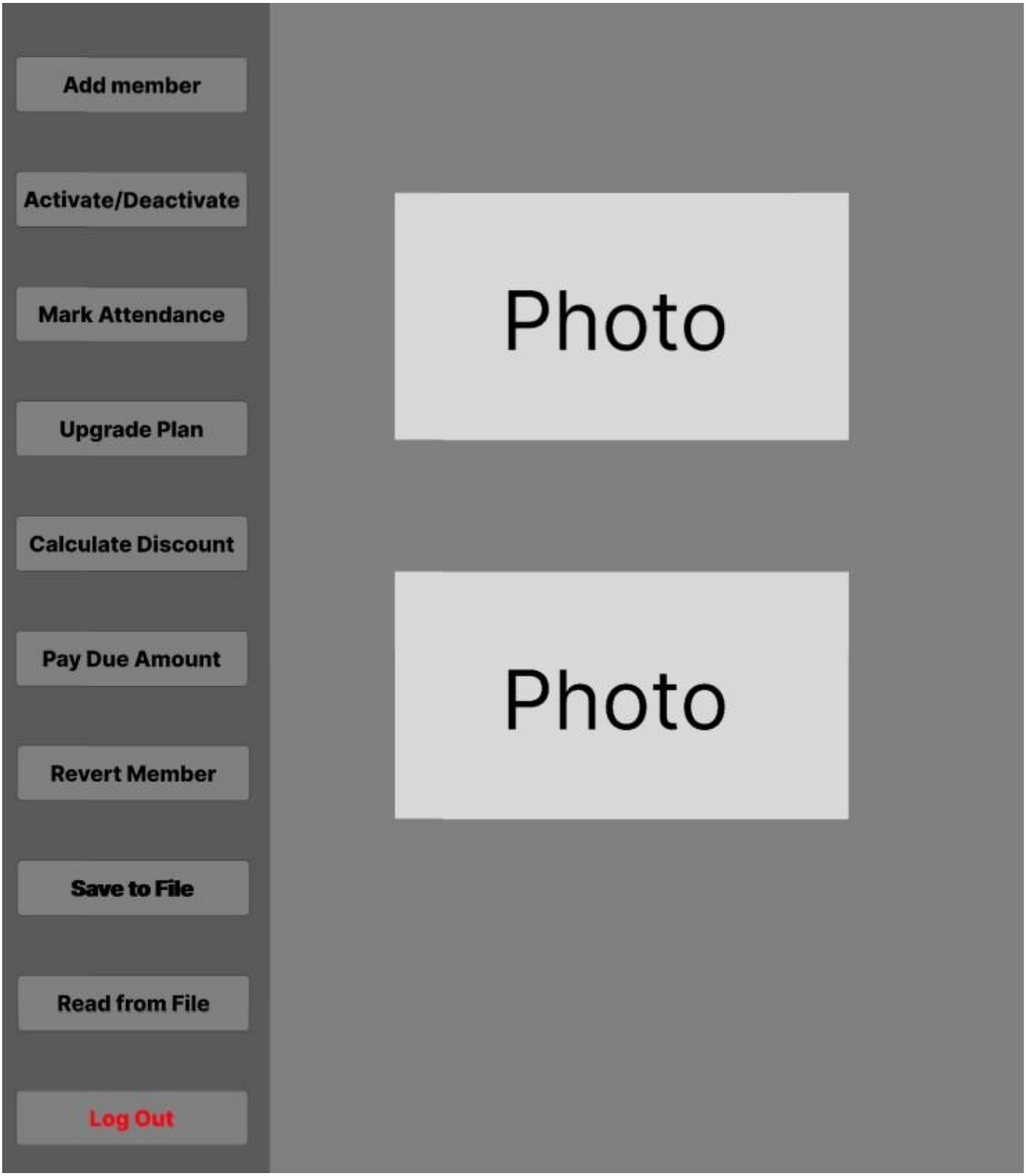


Figure 6: Wireframe of home page

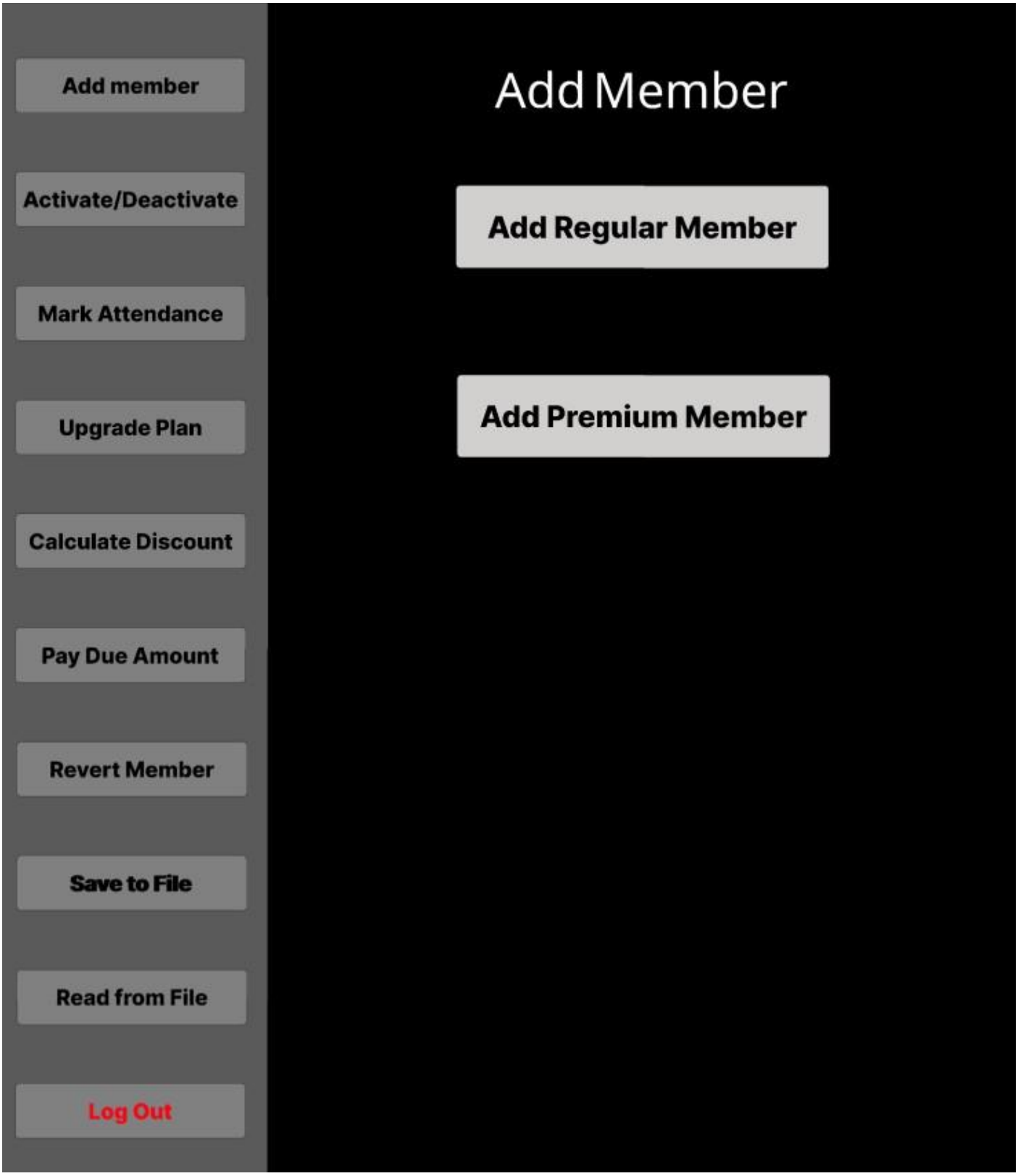


Figure 7: Wireframe of Add Member Panel

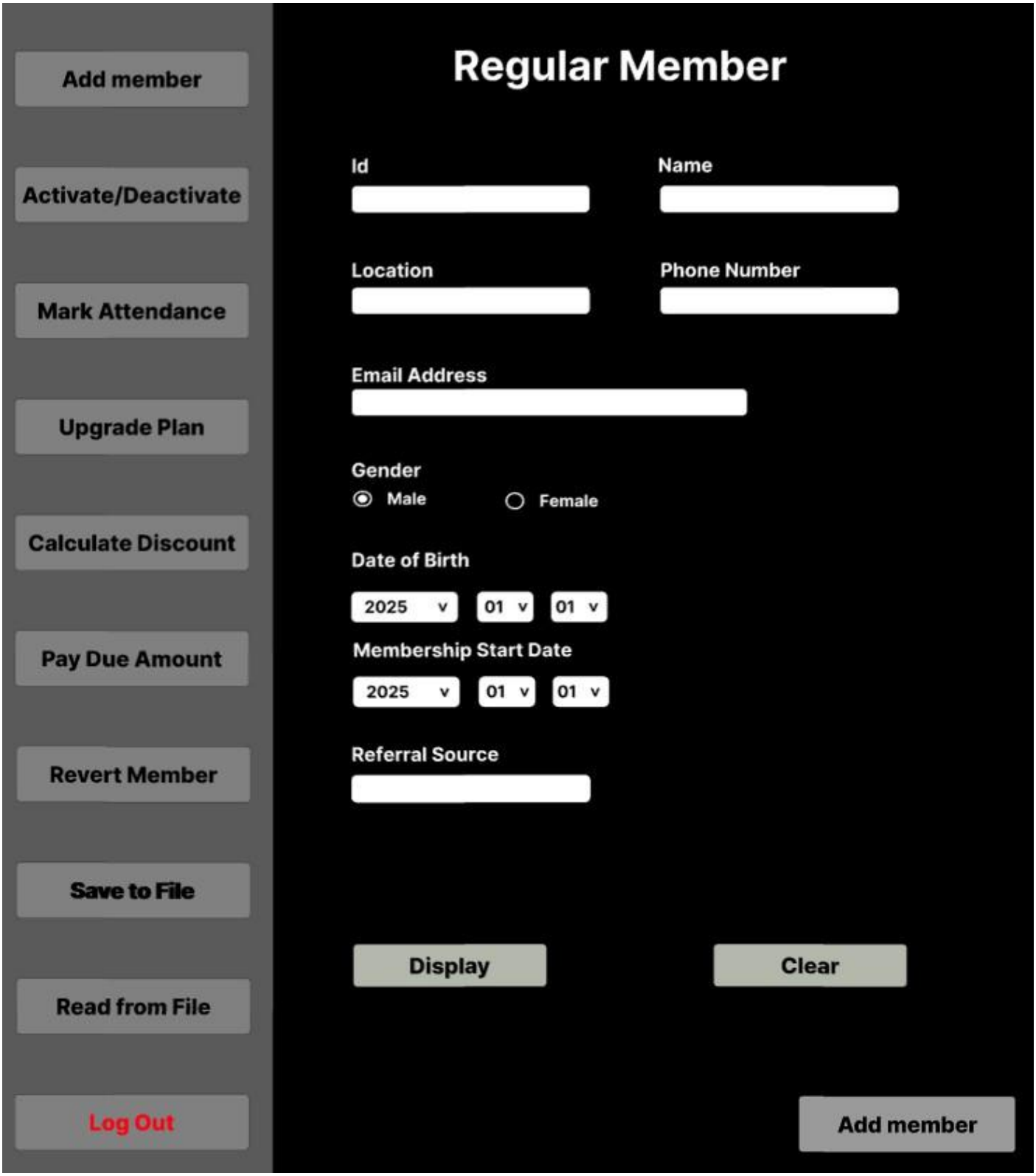


Figure 8: Wireframe of Regular Panel

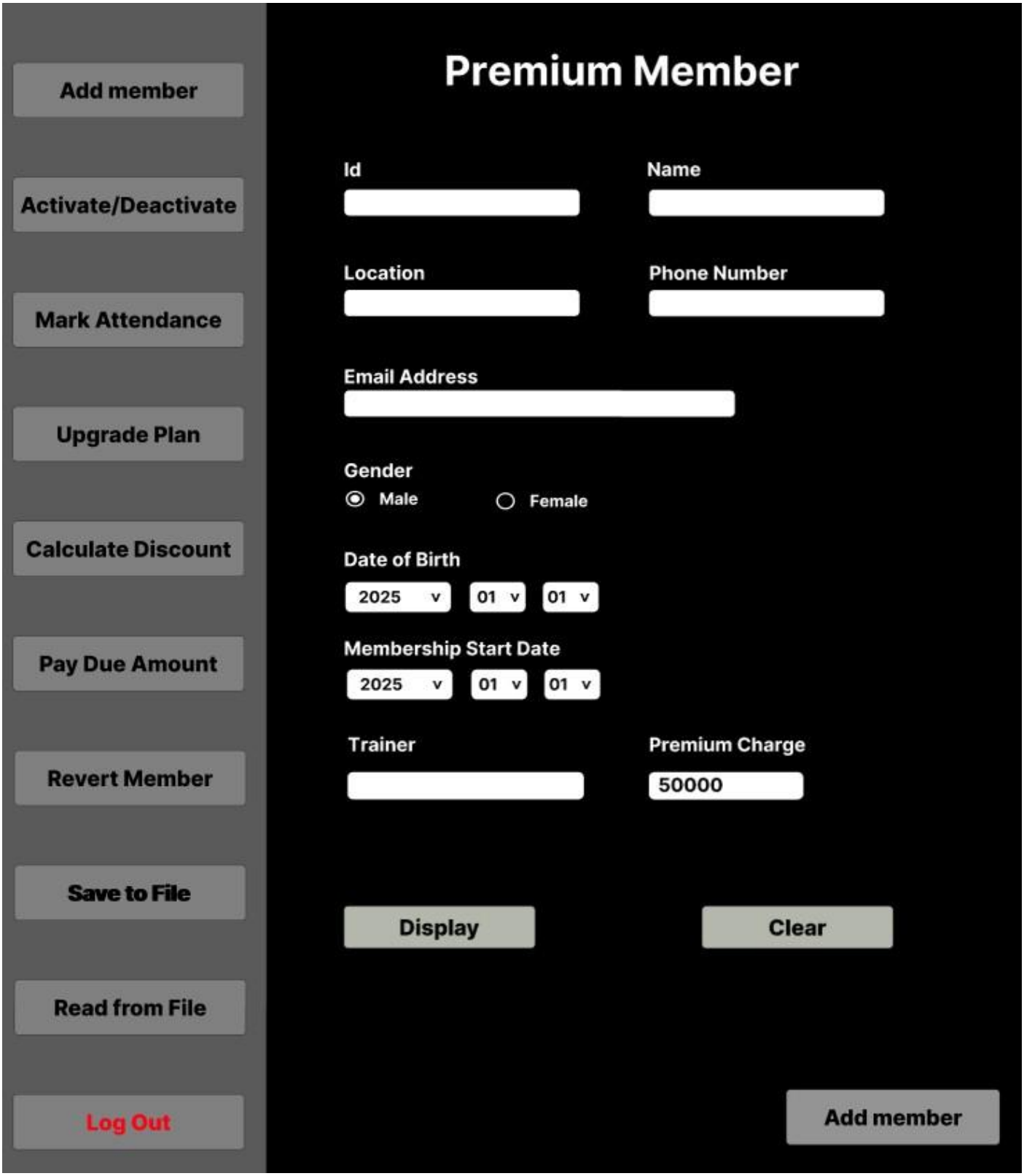


Figure 9: Wireframe of Premium Panel

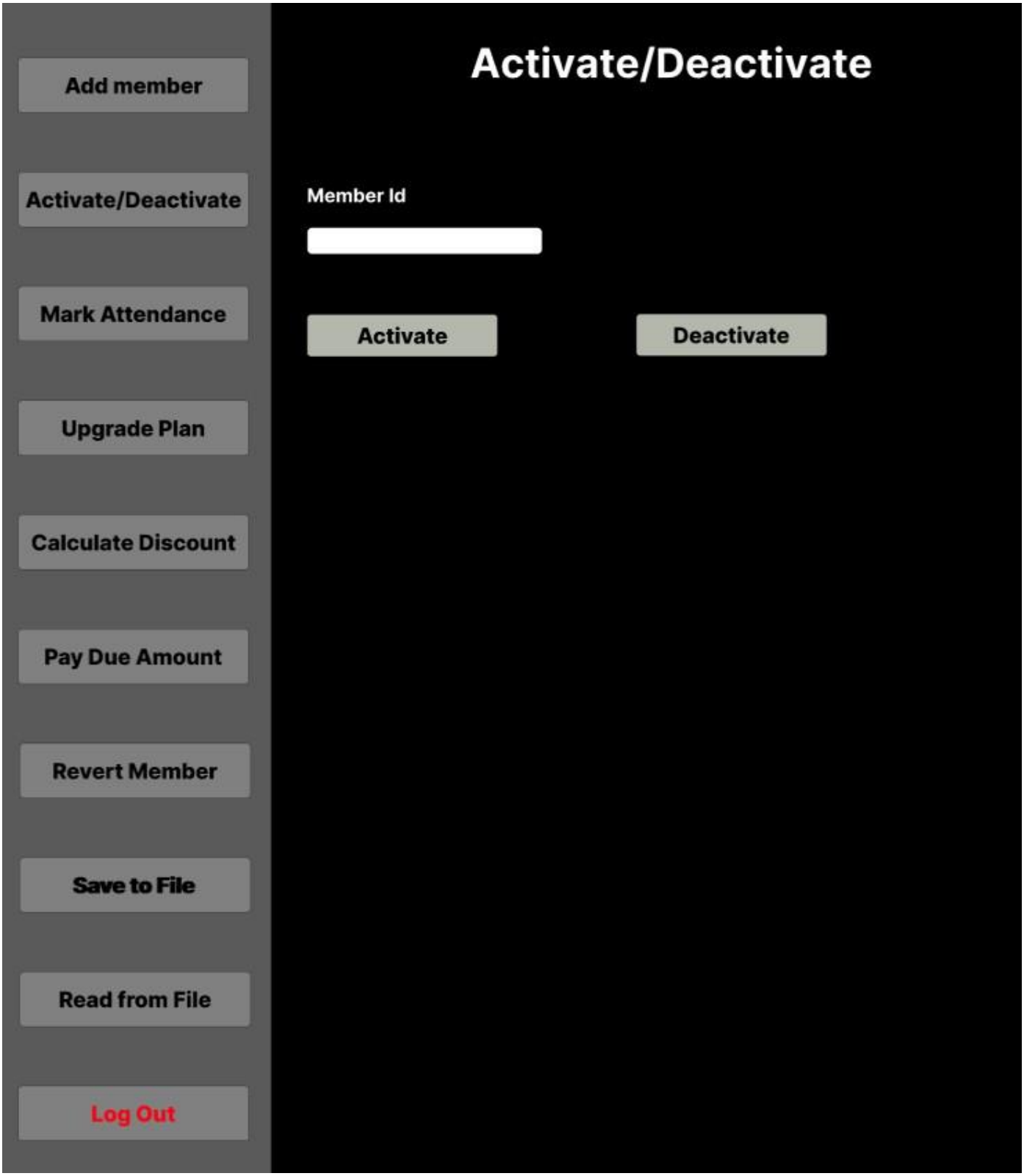


Figure 10: Wireframe of Activate/Deactivate Panel

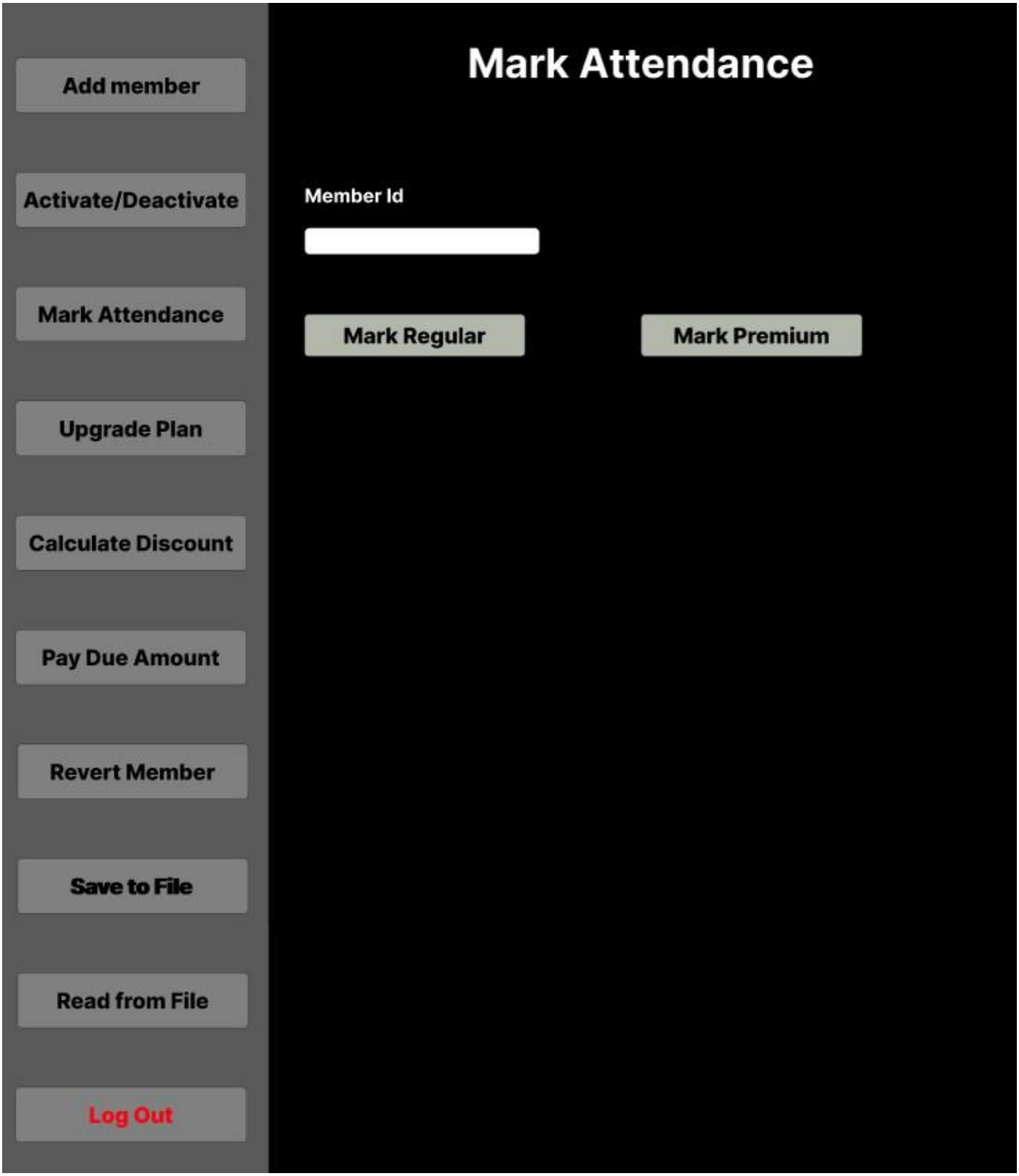


Figure 11: Wireframe of Mark Attendance Panel

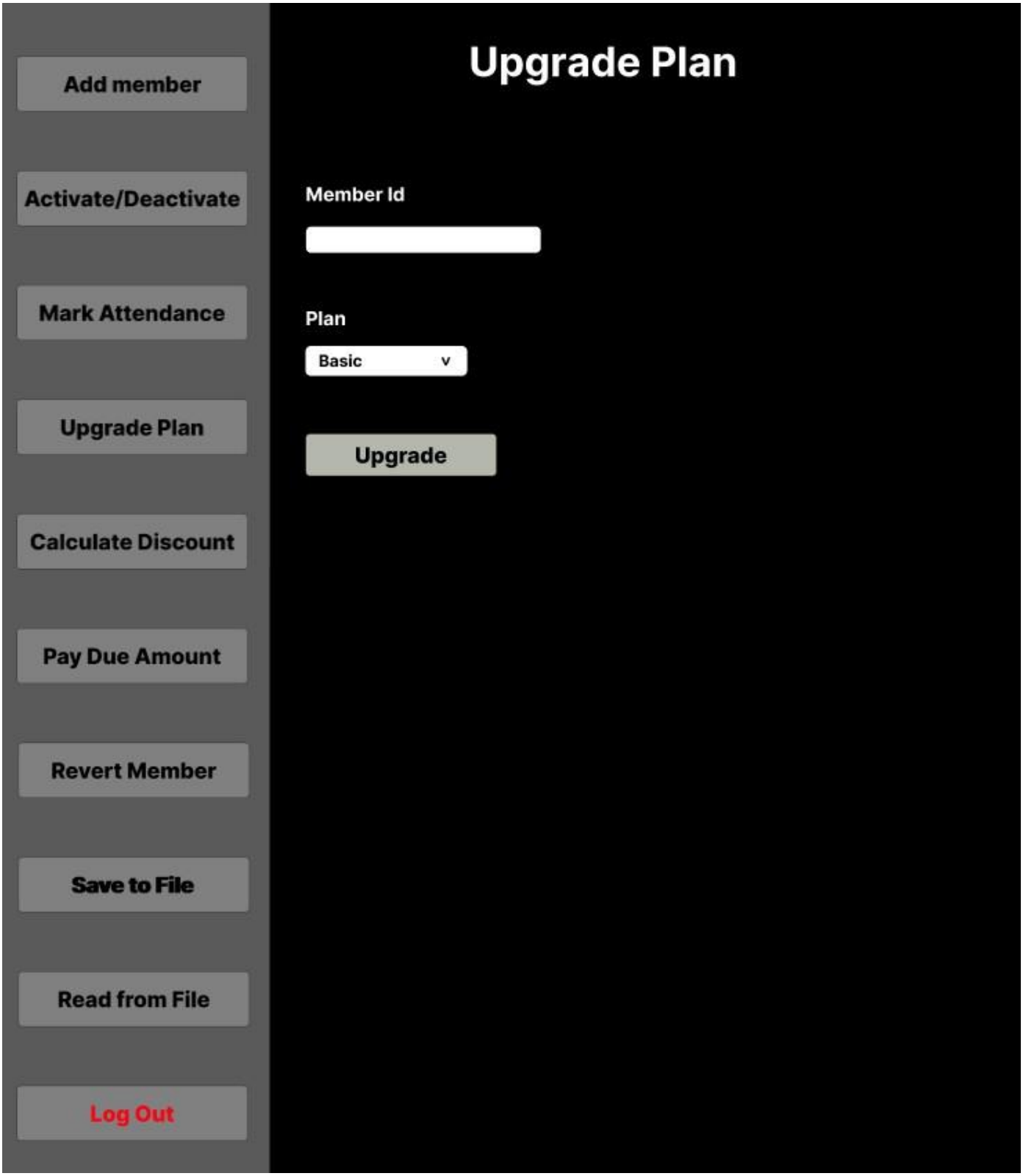


Figure 12: Wireframe of Upgrade Plan Panel

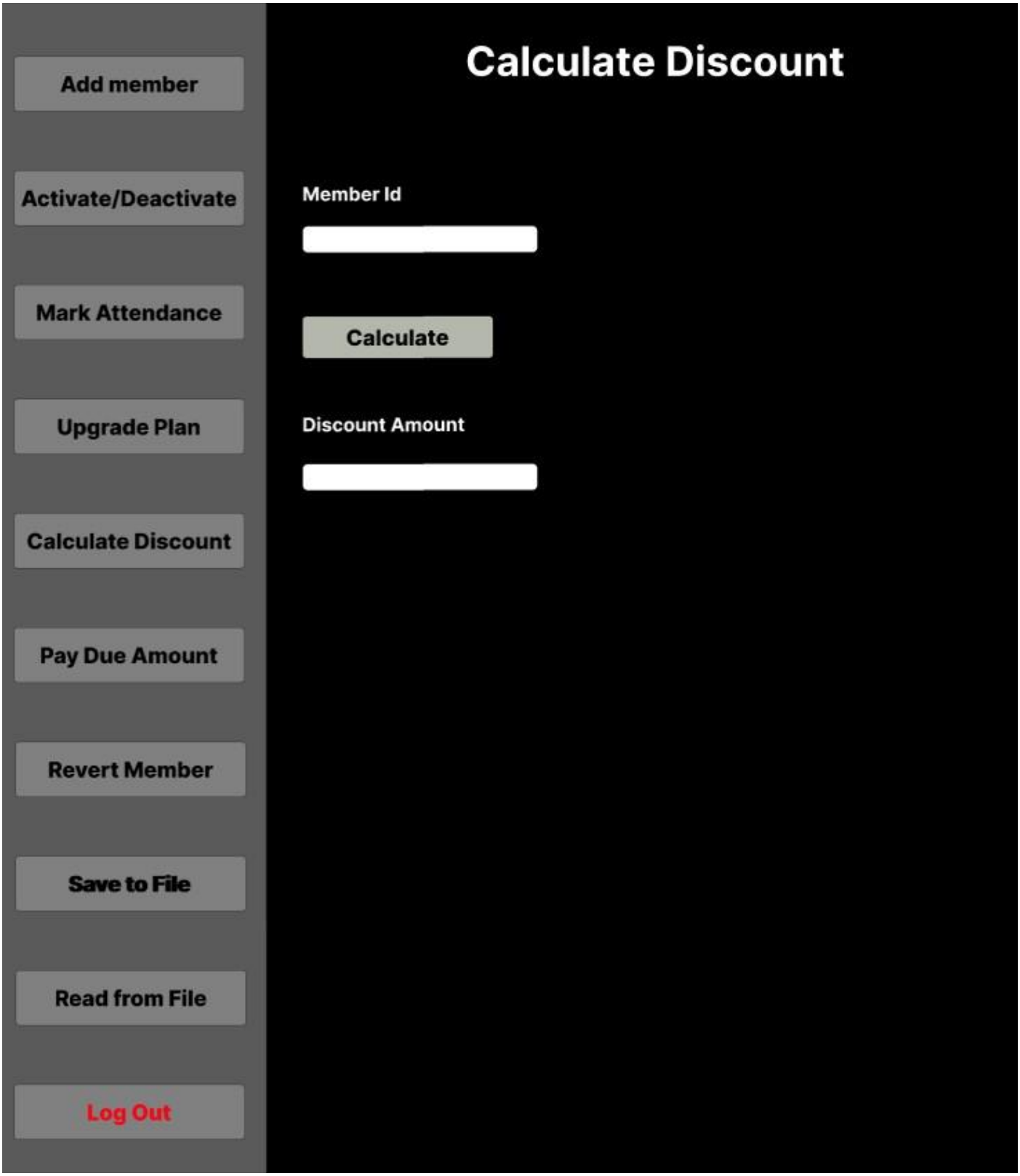


Figure 13: Wireframe of Calculate Discount Panel

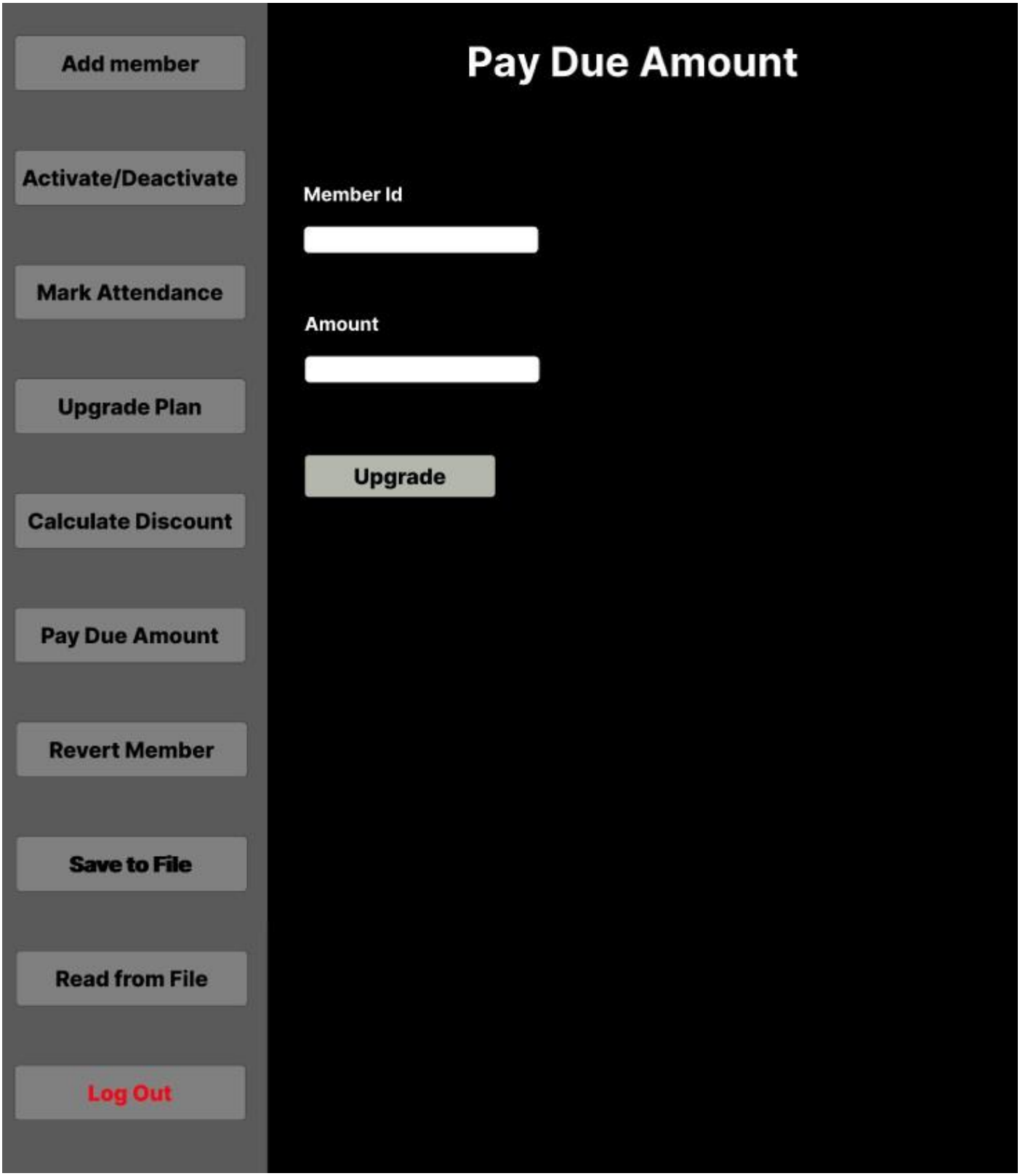


Figure 14: Wireframe of Pay Due Panel

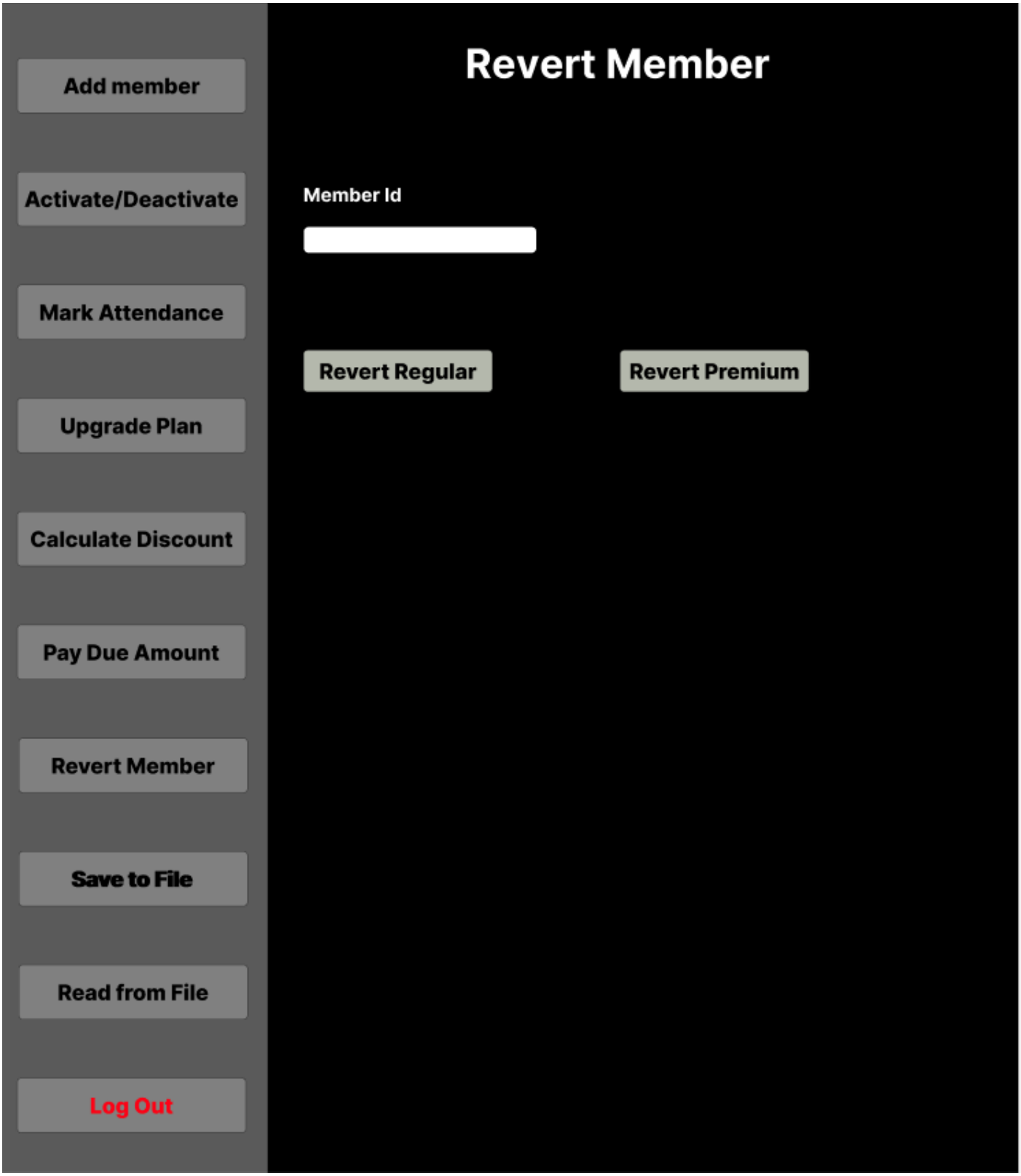


Figure 15: Wireframe of Revert Member Panel

Developed GUI

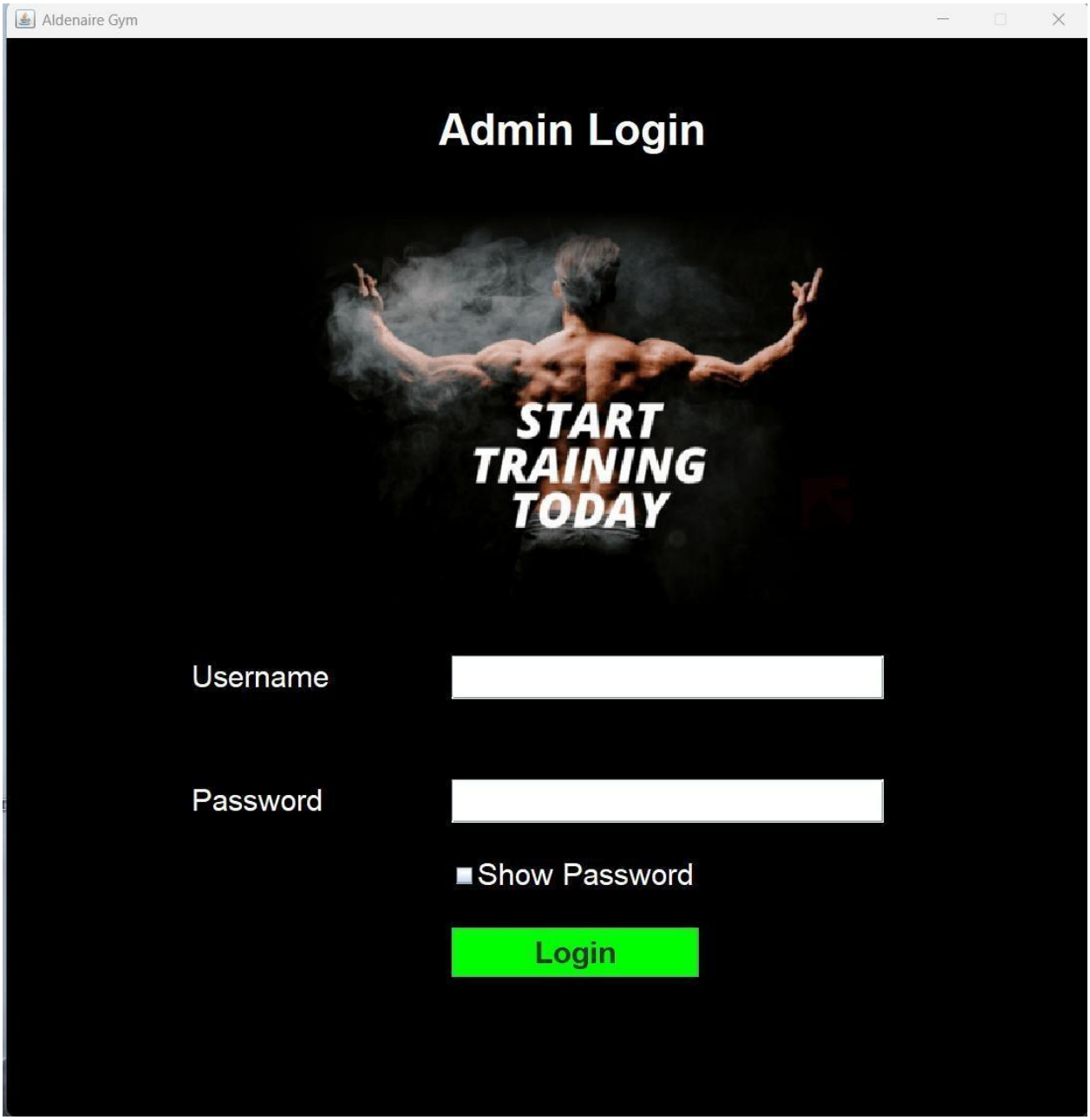


Figure 16: Screenshot of Home Page

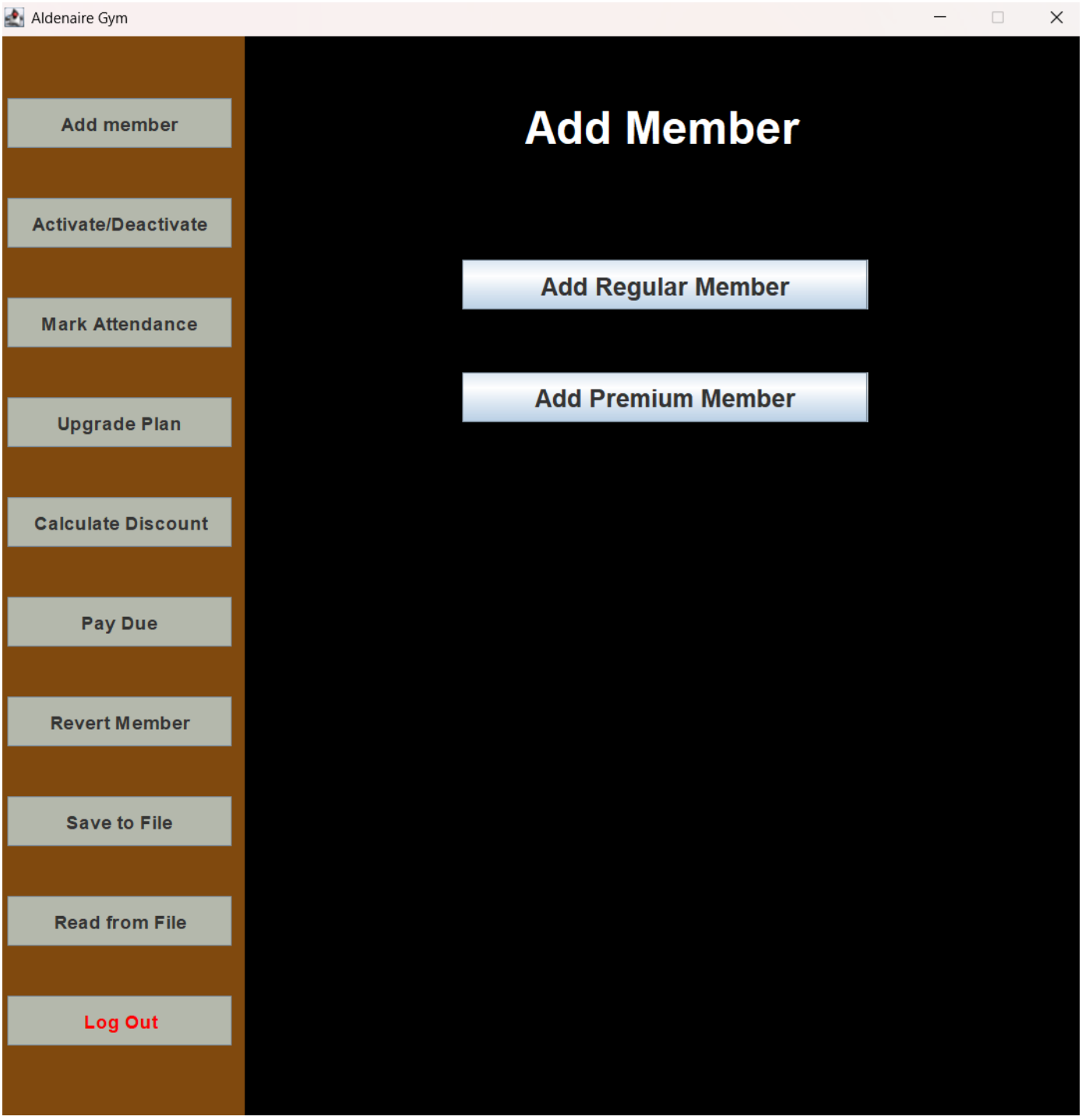


Figure 17: Screenshot of Add Member Page

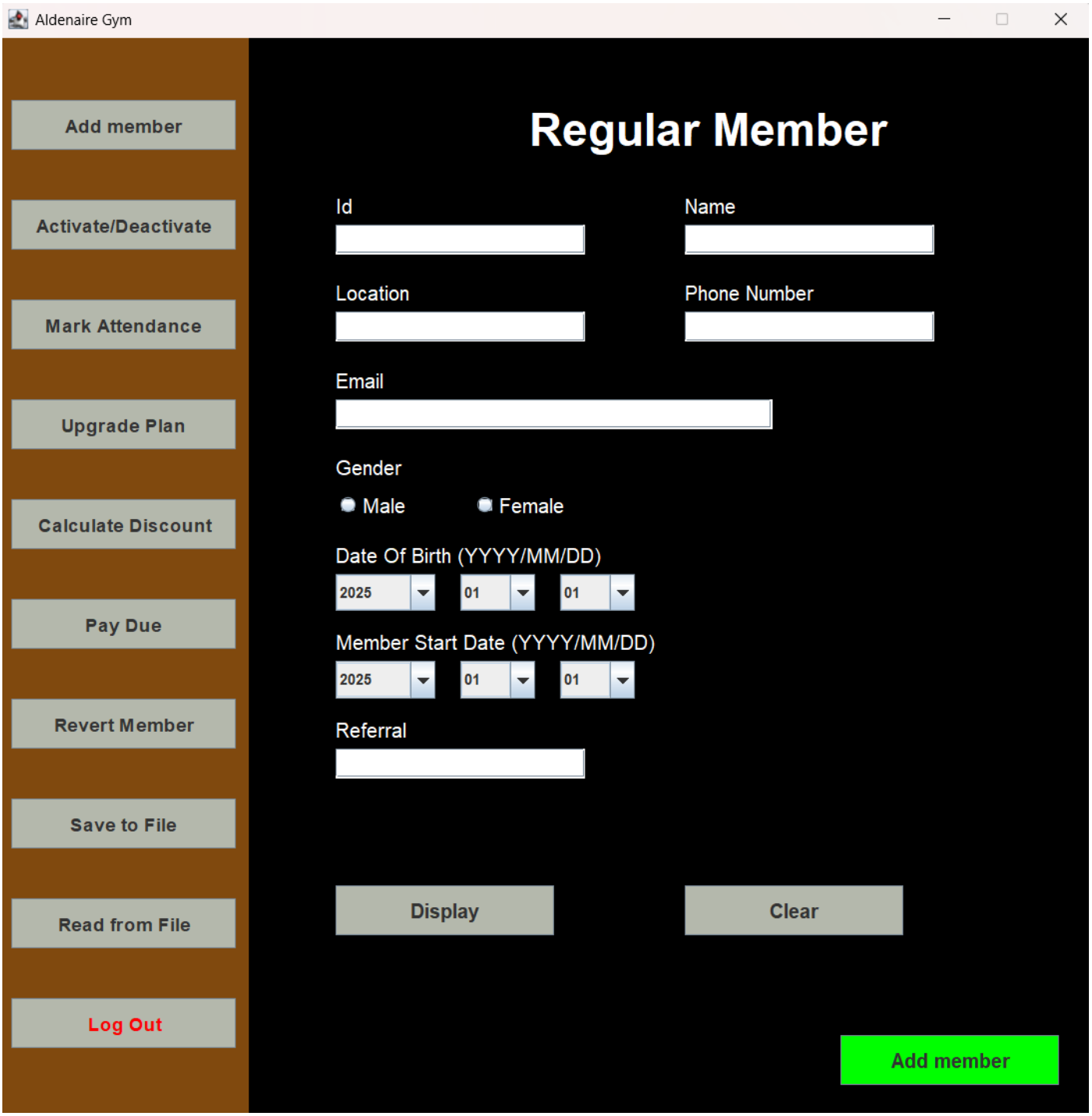


Figure 18: Screenshot of Regular Member Panel

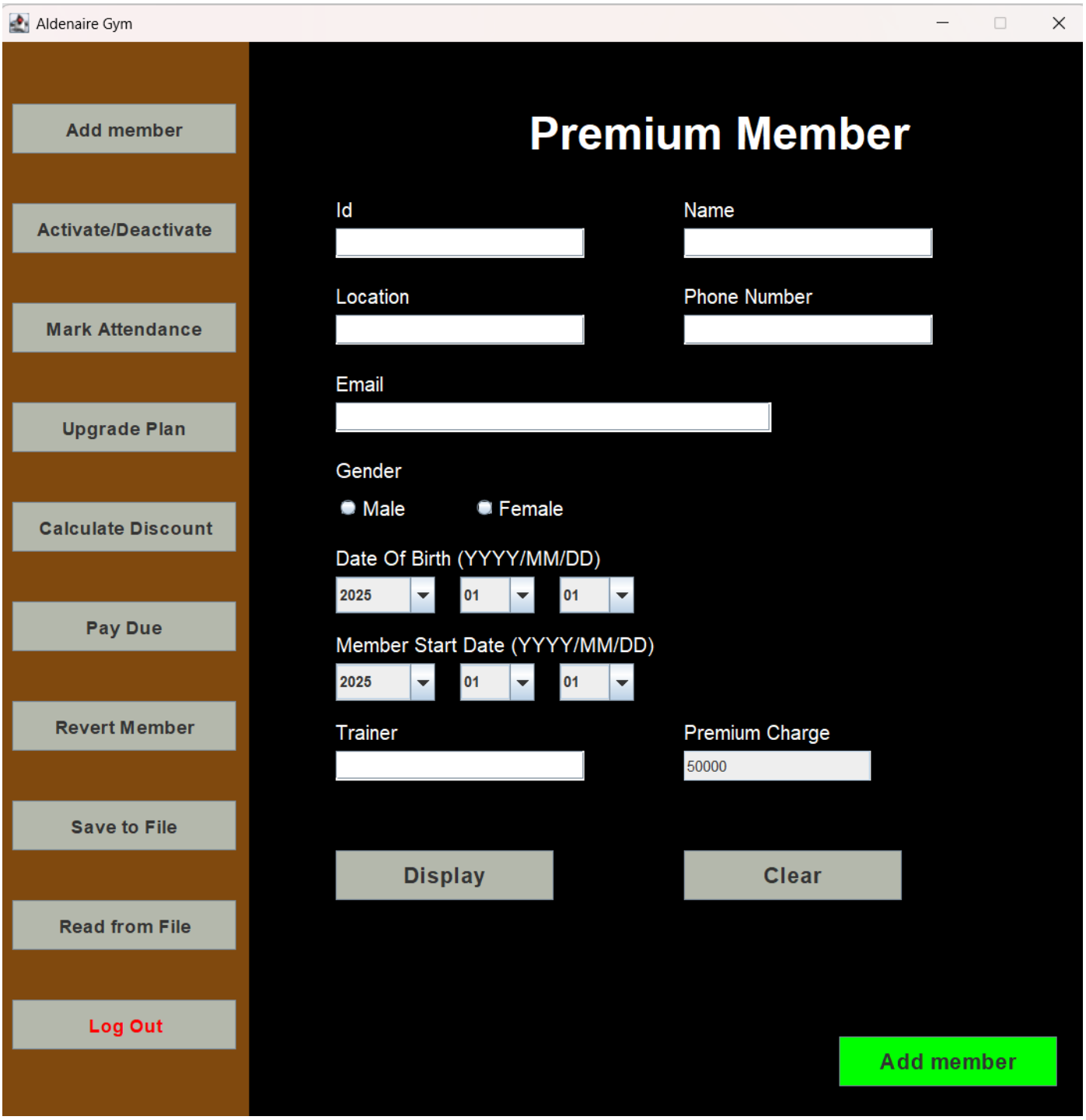


Figure 19: Screenshot of Premium Member Panel

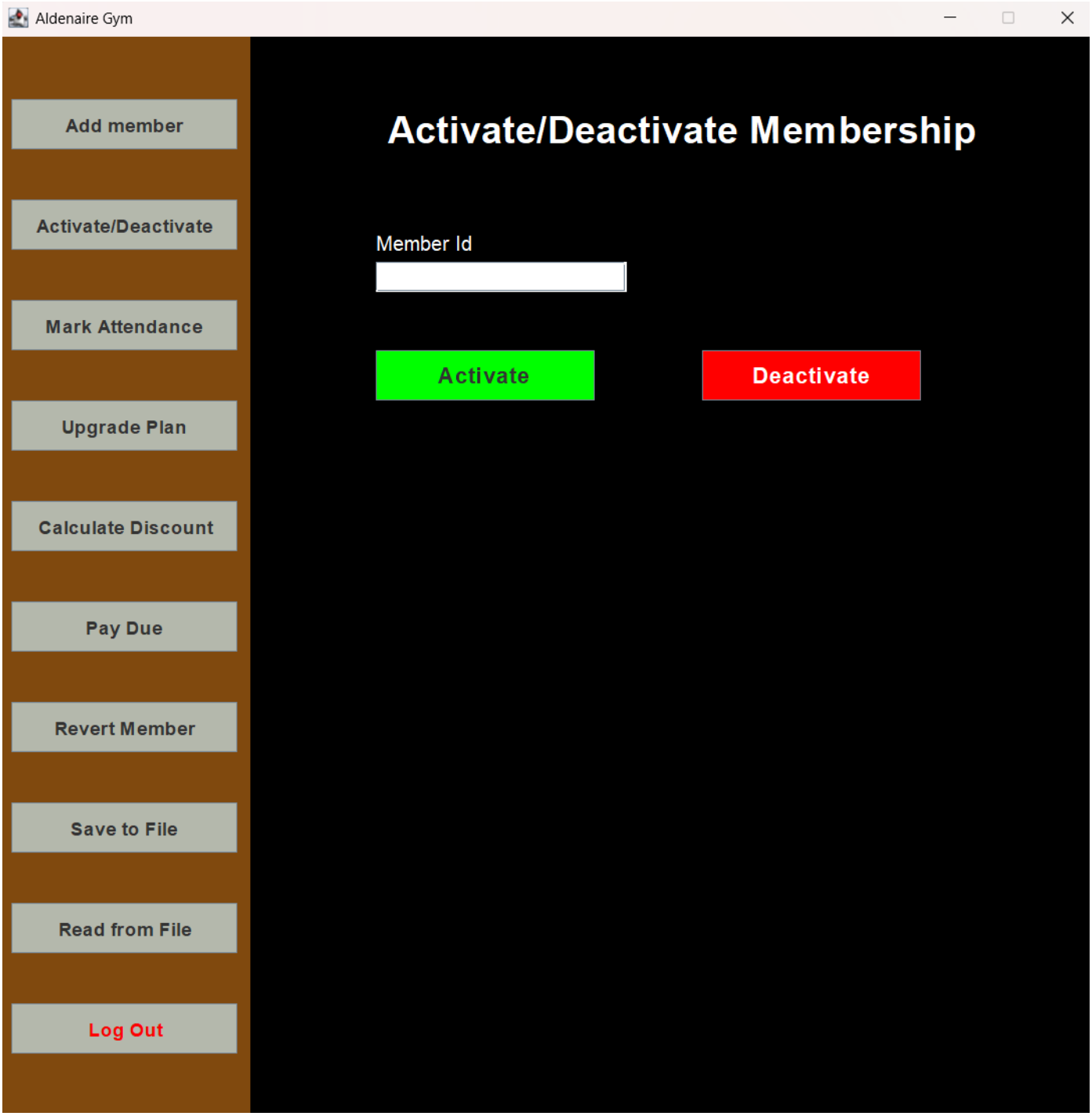


Figure 20: Screenshot of Activate/Deactivate Panel

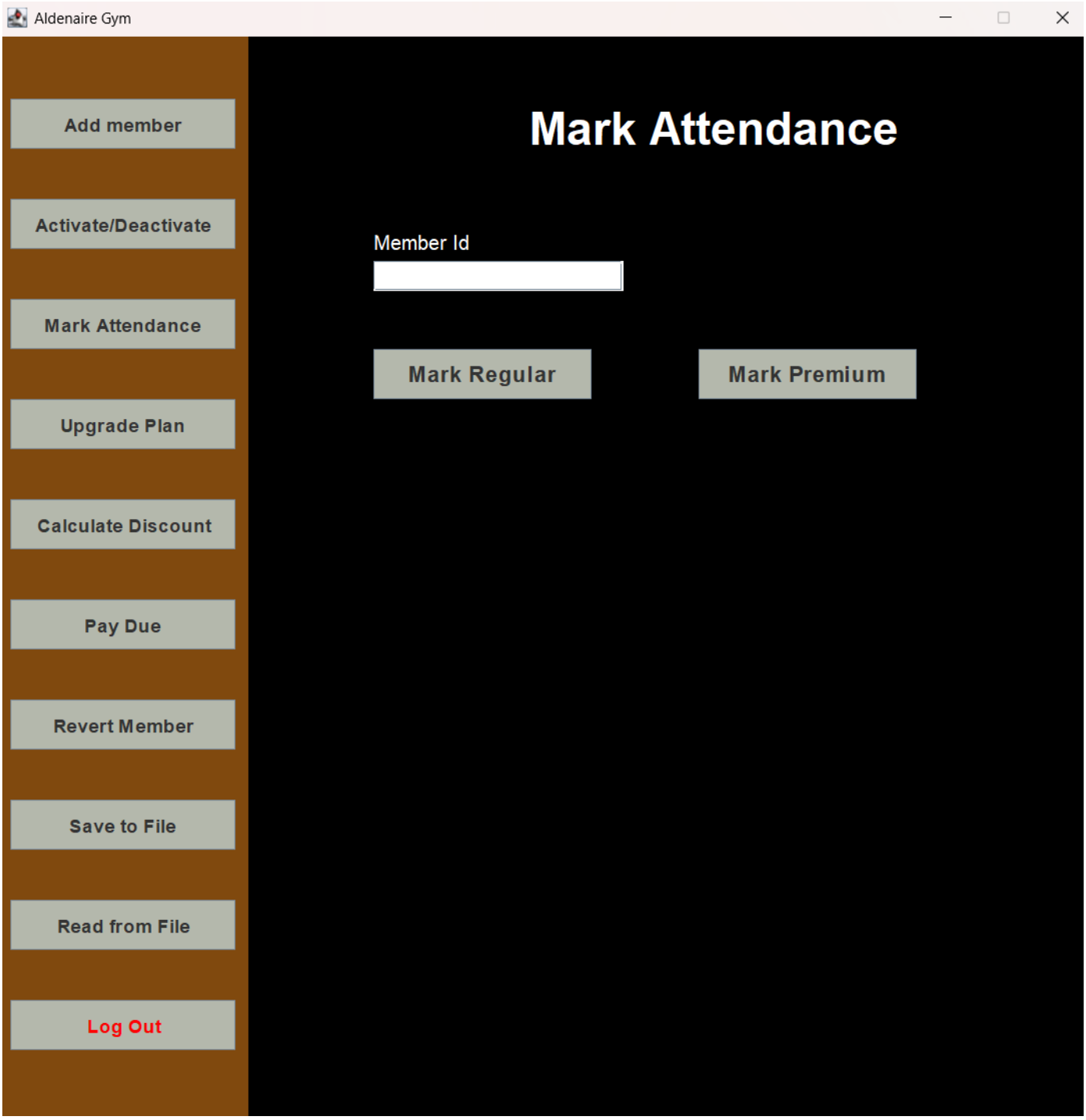


Figure 21: Screenshot of Mark Attendance Panel

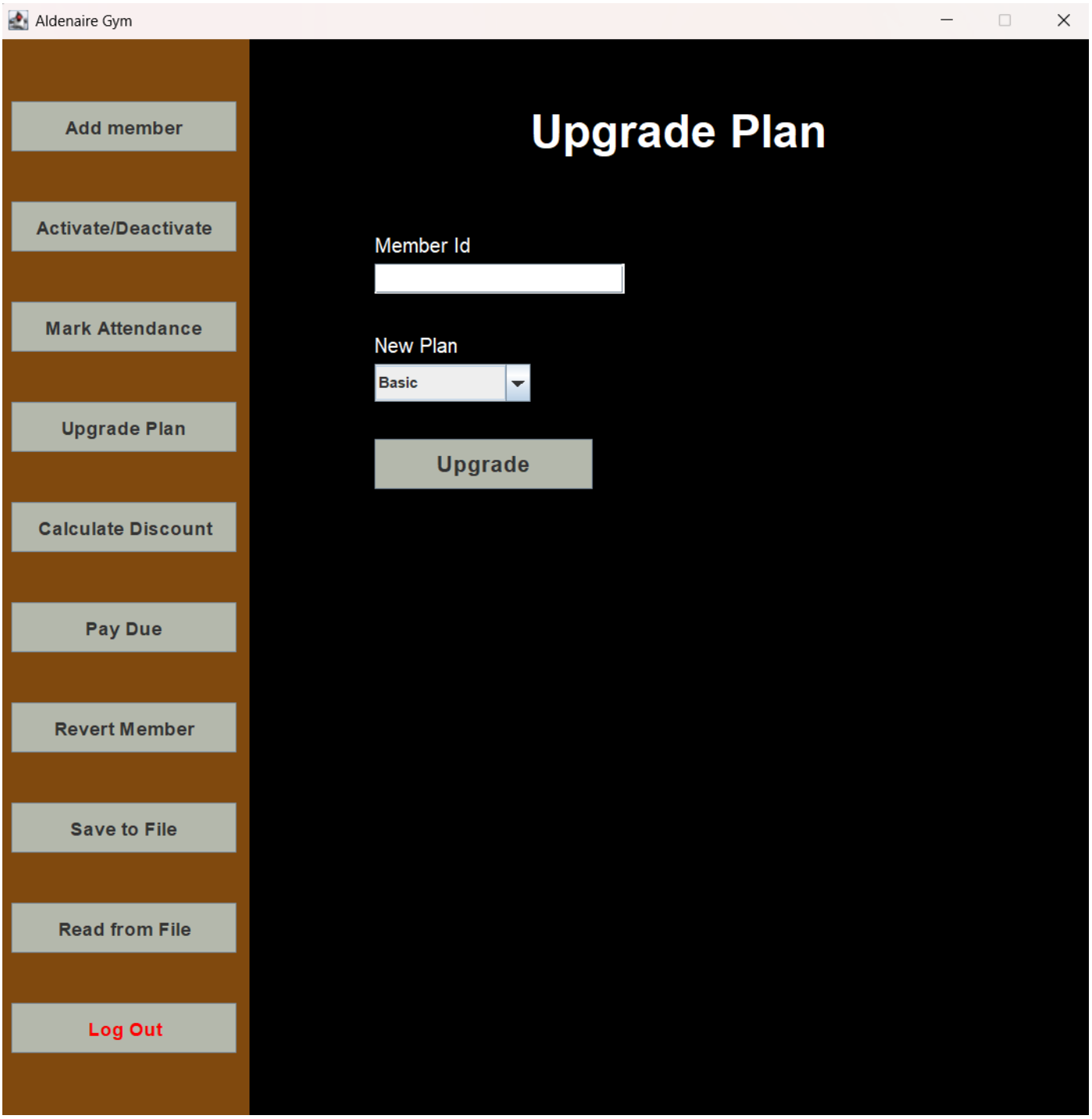


Figure 22: Screenshot of Upgrade Plan Panel

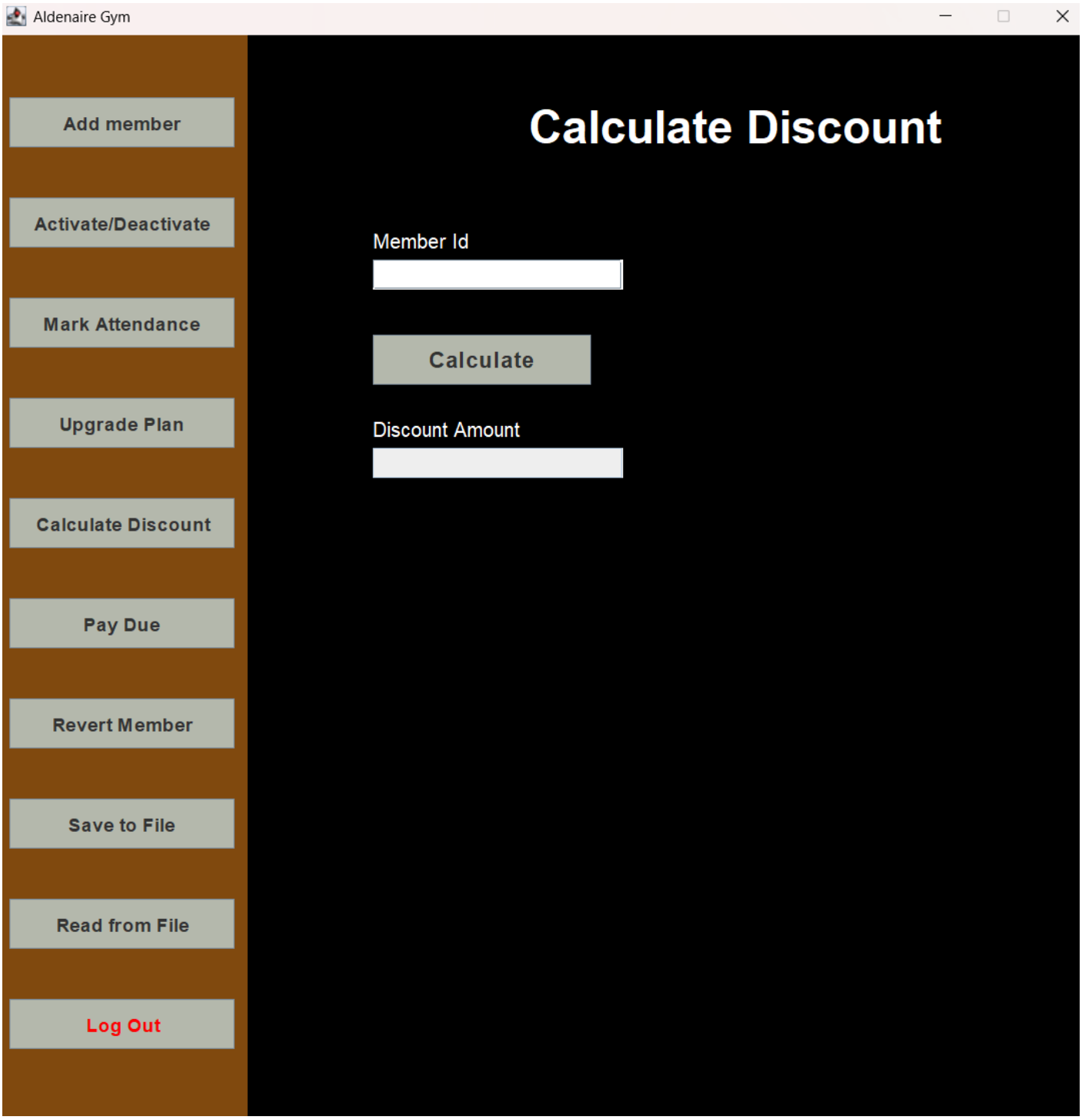


Figure 23: Screenshot of Calculate Discount Panel

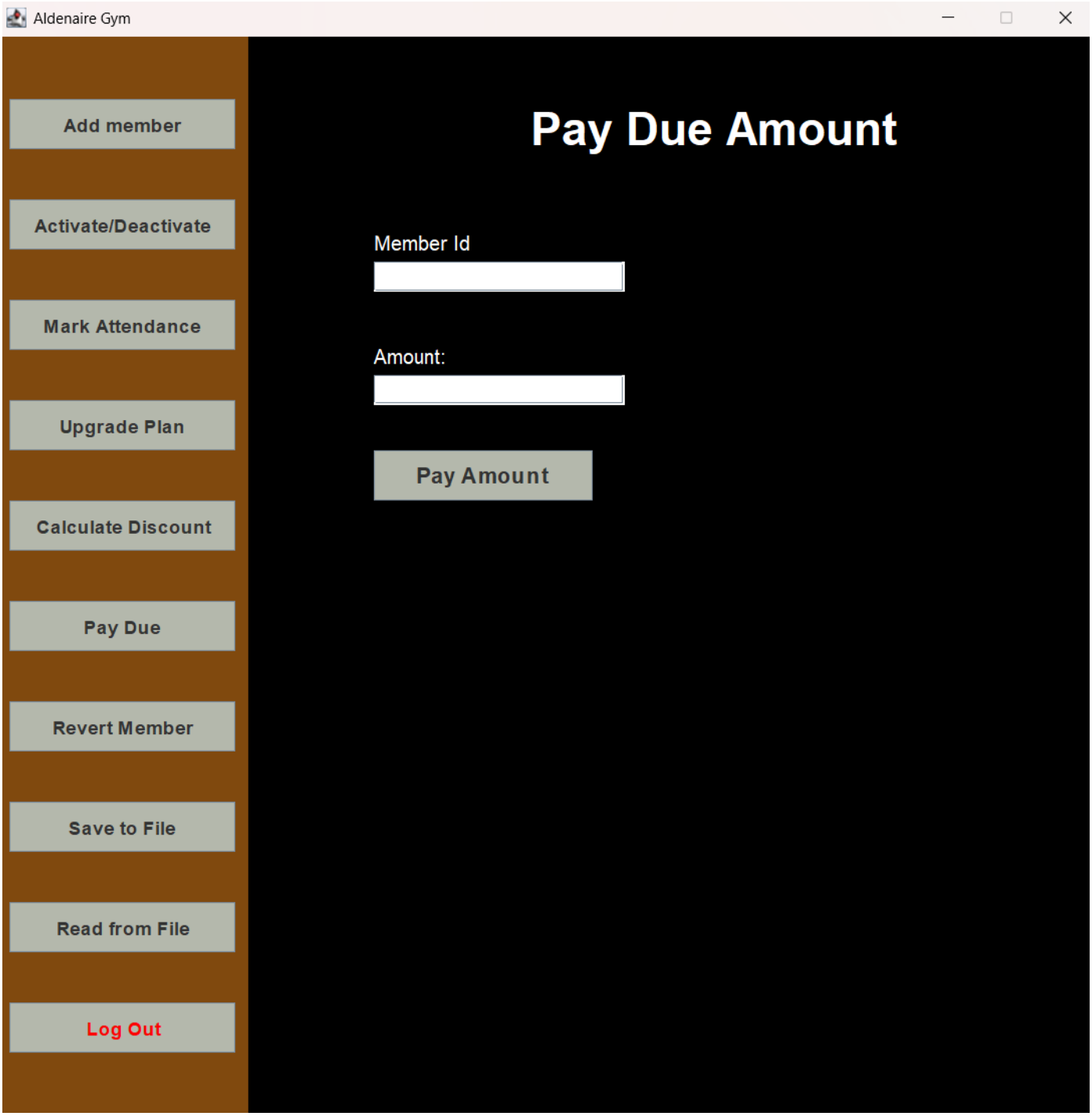


Figure 24: Screenshot of Pay Due Panel

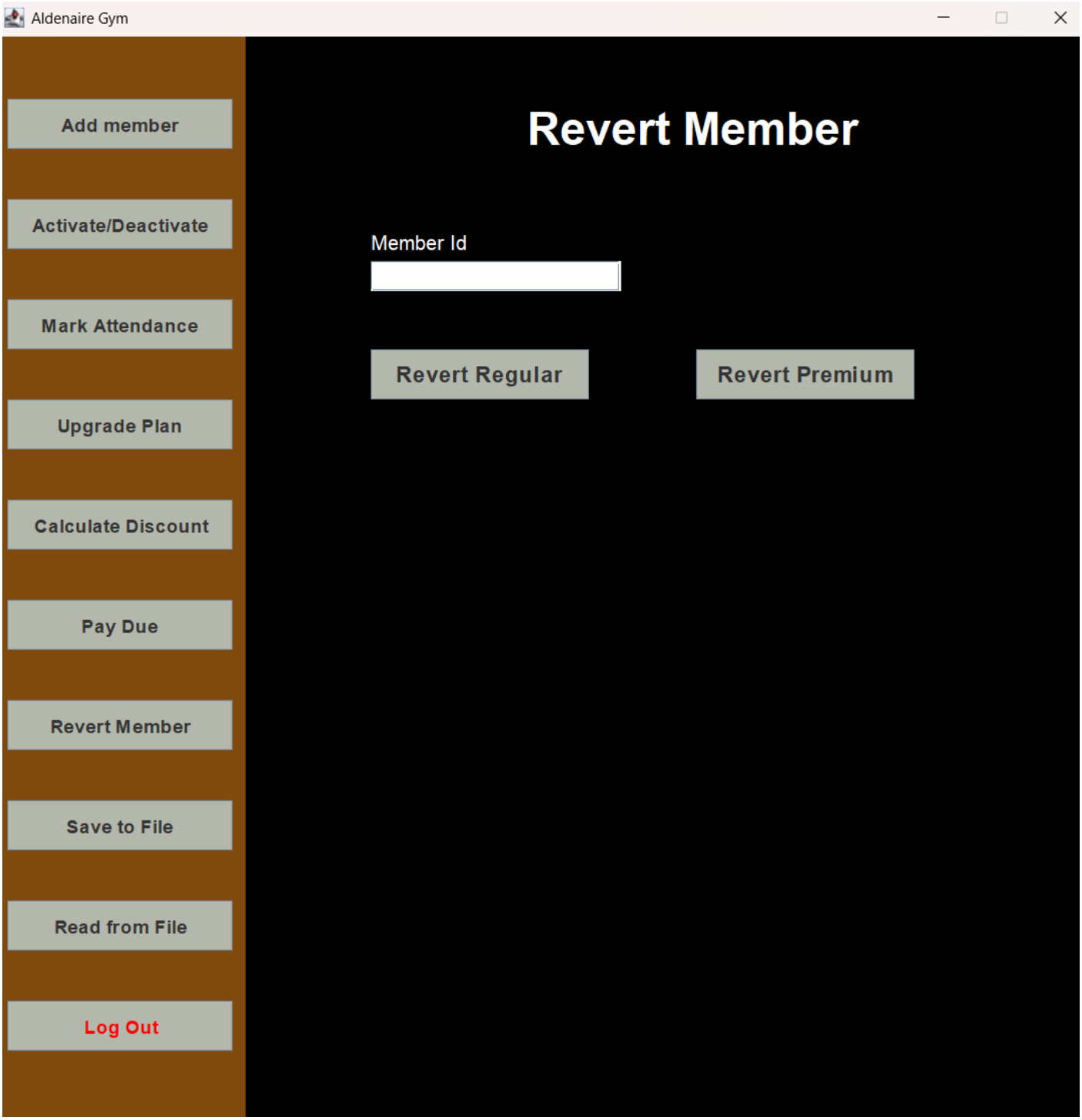


Figure 25: Screenshot of Revert Member Panel

Class Diagram

A class diagram often represented using the Unified Modeling Language (UML), is a graphical notation used to construct and visualize the static structure of an object-oriented system. It shows the system's classes, their attributes (variables), methods (operations), and the relationships between the classes. Each class divided as a box divided into three parts: the top part contains the class name, the middle lists its attributes, and the bottom shows its methods. Class diagrams help programmers and designers understand how different classes interact, how inheritance and associations are structured, and how the overall system is organized before writing the actual code. They are essential for planning, documenting, and communicating the design of Java programs, making complex systems easier to manage and develop. In my program there are 4 classes: GymMember is parent class, RegularMember and PremiumMember are its child class which extends GymMember and GymGUI is the user interface class.

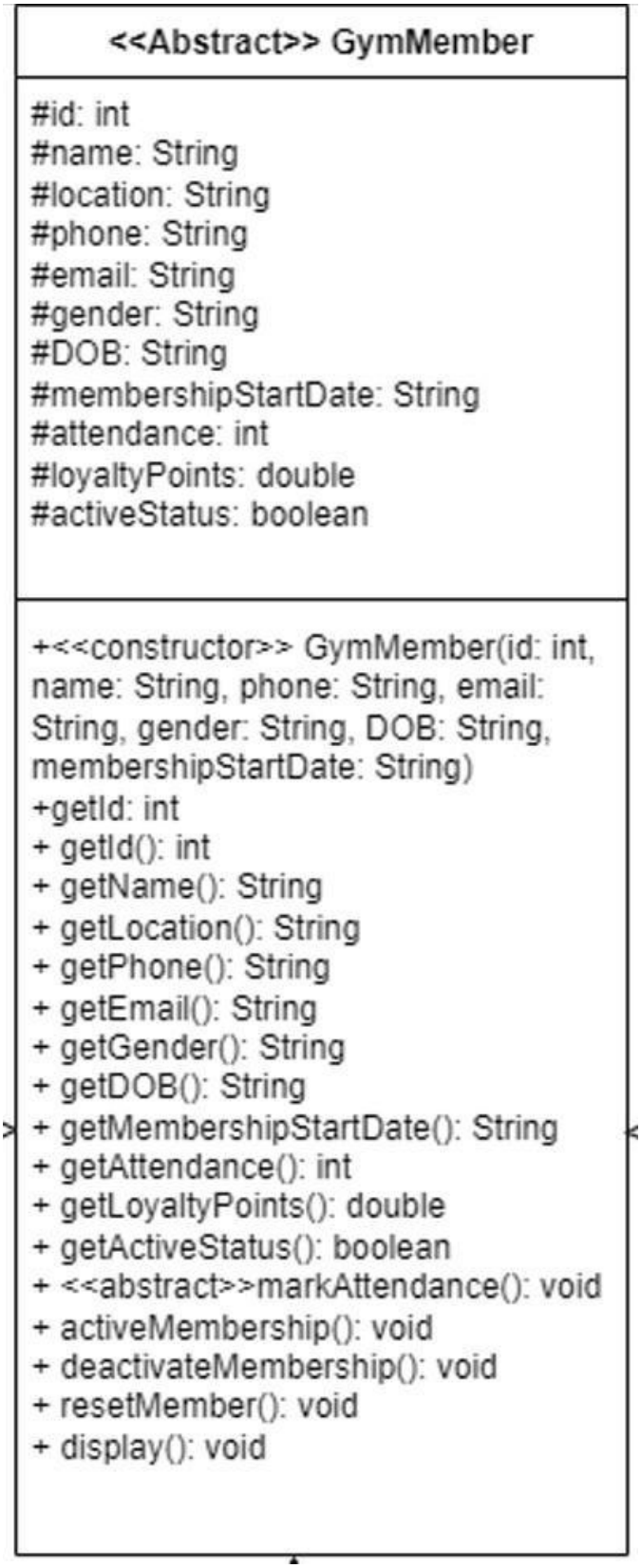

<<Abstract>> GymMember
#id: int
#name: String
#location: String
#phone: String
#email: String
#gender: String
#DOB: String
#membershipStartDate: String
#attendance: int
#loyaltyPoints: double
#activeStatus: boolean
+<<constructor>> GymMember(id: int, name: String, phone: String, email: String, gender: String, DOB: String, membershipStartDate: String)
+getId: int
+ getId(): int
+ getName(): String
+ getLocation(): String
+ getPhone(): String
+ getEmail(): String
+ getGender(): String
+ getDOB(): String
+ getMembershipStartDate(): String
+ getAttendance(): int
+ getLoyaltyPoints(): double
+ getActiveStatus(): boolean
+ <<abstract>>markAttendance(): void
+ activeMembership(): void
+ deactivateMembership(): void
+ resetMember(): void
+ display(): void

Figure 26: Class Diagram of GymMember

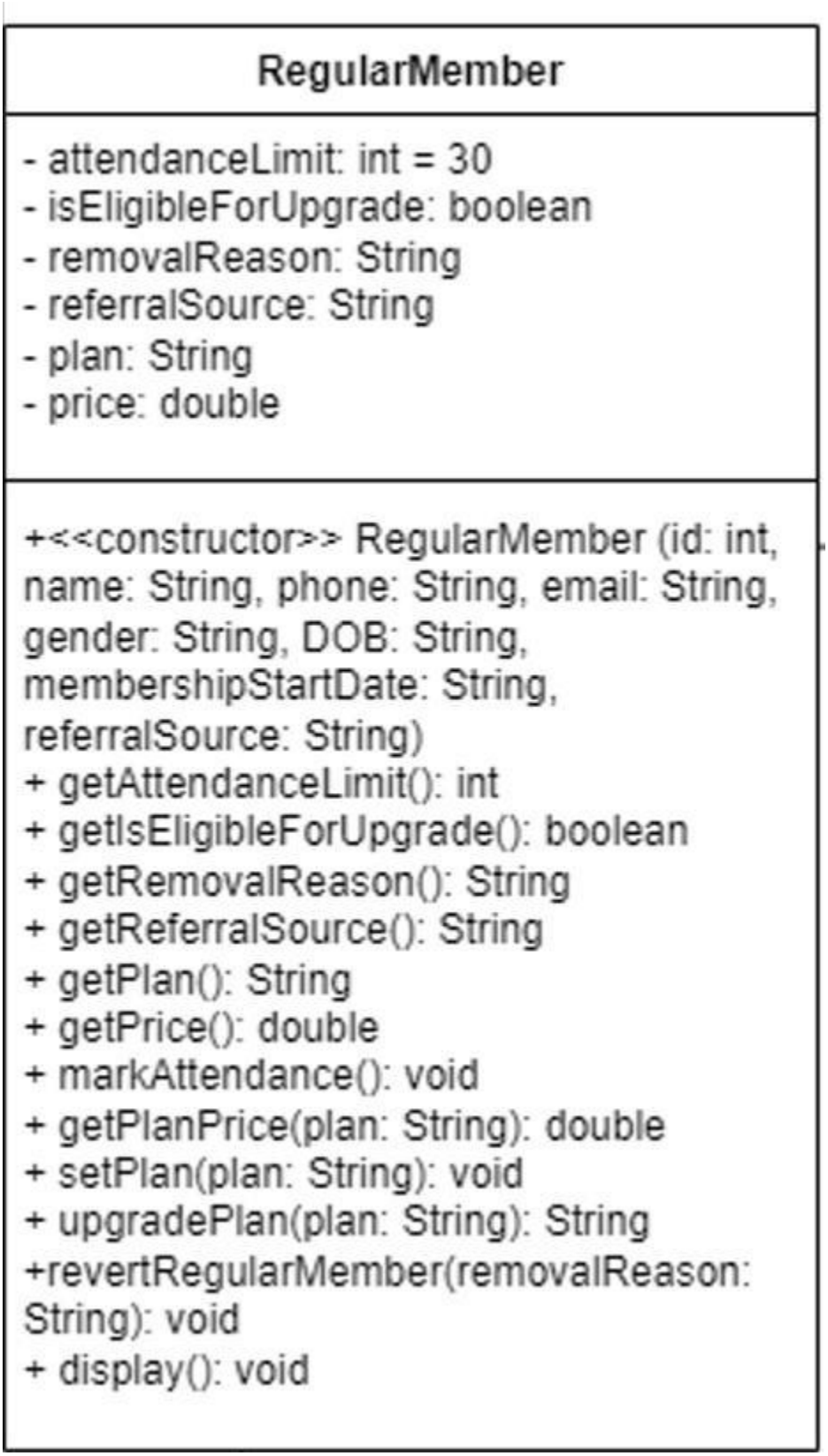


Figure 27: Class Diagram of RegularMember

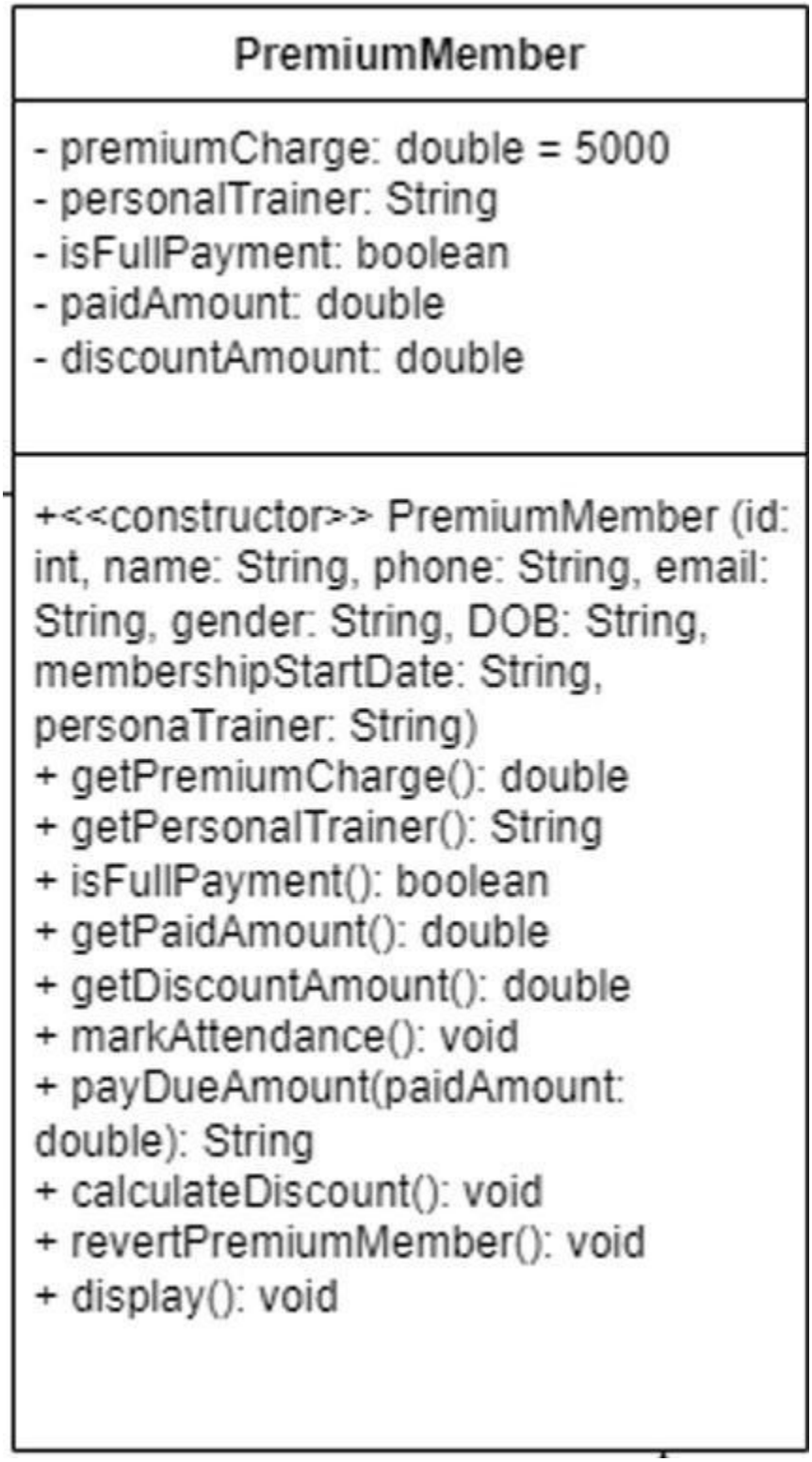


Figure 28: Class Diagram of PremiumMember

Figure 29: Class Diagram of GymGUI

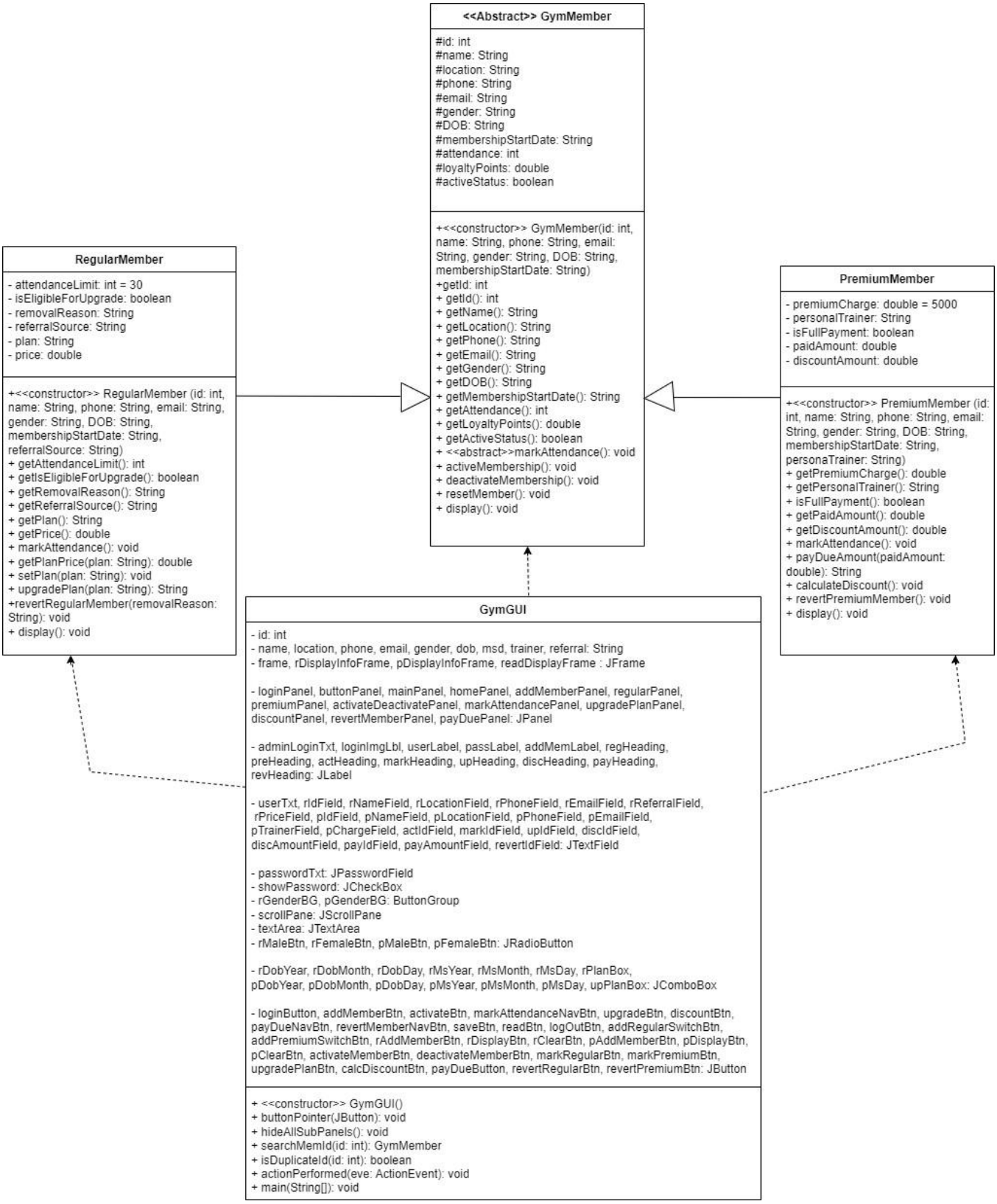

<<Abstract>> GymMember
#id: int
#name: String
#location: String
#phone: String
#email: String
#gender: String
#DOB: String
#membershipStartDate: String
#attendance: int
#loyaltyPoints: double
#activeStatus: boolean
+<<constructor>> GymMember(id: int, name: String, phone: String, email: String, gender: String, DOB: String, membershipStartDate: String)
+getId: int
+ getId(): int
+ getName(): String
+ getLocation(): String
+ getPhone(): String
+ getEmail(): String
+ getGender(): String
+ getDOB(): String
+ getMembershipStartDate(): String
+ getAttendance(): int
+ getLoyaltyPoints(): double
+ getActiveStatus(): boolean
+ <<abstract>>markAttendance(): void
+ activeMembership(): void
+ deactivateMembership(): void
+ resetMember(): void
+ display(): void
RegularMember
- attendanceLimit: int = 30
- isEligibleForUpgrade: boolean
- removalReason: String
- referralSource: String
- plan: String
- price: double
+<<constructor>> RegularMember (id: int, name: String, phone: String, email: String, gender: String, DOB: String, membershipStartDate: String, referralSource: String)
+ getAttendanceLimit(): int
+ getIsEligibleForUpgrade(): boolean
+ getRemovalReason(): String
+ getReferralSource(): String
+ getPlan(): String
+ getPrice(): double
+ markAttendance(): void
+ getPlanPrice(plan: String): double
+ setPlan(plan: String): void
+ upgradePlan(plan: String): String
+revertRegularMember(removalReason: String): void
+ display(): void
PremiumMember
- premiumCharge: double = 5000
- personalTrainer: String
- isFullPayment: boolean
- paidAmount: double
- discountAmount: double
+<<constructor>> PremiumMember (id: int, name: String, phone: String, email: String, gender: String, DOB: String, membershipStartDate: String, personaTrainer: String)
+ getPremiumCharge(): double
+ getPersonalTrainer(): String
+ isFullPayment(): boolean
+ getPaidAmount(): double
+ getDiscountAmount(): double
+ markAttendance(): void
+ payDueAmount(paidAmount: double): String
+ calculateDiscount(): void
+ revertPremiumMember(): void
+ display(): void
GymGUI
- id: int
- name, location, phone, email, gender, dob, msd, trainer, referral: String
- frame, rDisplayInfoFrame, pDisplayInfoFrame, readDisplayFrame : JFrame
- loginPanel, buttonPanel, mainPanel, homePanel, addMemberPanel, regularPanel, premiumPanel, activateDeactivatePanel, markAttendancePanel, upgradePlanPanel, discountPanel, revertMemberPanel, payDuePanel: JPanel
- adminLoginTxt, loginImgLbl, userLabel, passLabel, addMemLabel, regHeading, preHeading, actHeading, markHeading, upHeading, discHeading, payHeading, revHeading: JLabel
- userTxt, rIdField, rNameField, rLocationField, rPhoneField, rEmailField, rReferralField, rPriceField, pIdField, pNameField, pLocationField, pPhoneField, pEmailField, pTrainerField, pChargeField, actIdField, markIdField, upIdField, discIdField, discAmountField, payIdField, payAmountField, revertIdField: JTextField
- passwordTxt: JPasswordField
- showPassword: JCheckBox
- rGenderBG, pGenderBG: ButtonGroup
- scrollPane: JScrollPane
- textArea: JTextArea
- rMaleBtn, rFemaleBtn, pMaleBtn, pFemaleBtn: JRadioButton
- rDobYear, rDobMonth, rDobDay, rMsYear, rMsMonth, rMsDay, rPlanBox, pDobYear, pDobMonth, pDobDay, pMsYear, pMsMonth, pMsDay, upPlanBox: JComboBox
- loginButton, addMemberBtn, activateBtn, markAttendanceNavBtn, upgradeBtn, discountBtn, payDueNavBtn, revertMemberNavBtn, saveBtn, readBtn, logOutBtn, addRegularSwitchBtn, addPremiumSwitchBtn, rAddMemberBtn, rDisplayBtn, rClearBtn, pAddMemberBtn, pDisplayBtn, pClearBtn, activateMemberBtn, deactivateMemberBtn, markRegularBtn, markPremiumBtn, upgradePlanBtn, calcDiscountBtn, payDueButton, revertRegularBtn, revertPremiumBtn: JButton
+ <<constructor>> GymGUI()
+ buttonPointer(JButton): void
+ hideAllSubPanels(): void
+ searchMemId(id: int): GymMember
+ isDuplicateId(id: int): boolean
+ actionPerformed(eve: ActionEvent): void
+ main(String[]): void

Figure 30: Inheritance in Class Diagram

Pseudocode

Pseudocode is a simple way to describe the steps or logic of a computer program using plain English mixed with some programming ideas. It is not the actual code and cannot be run on a computer, but it helps people understand how a program should work before writing the real code. It is useful for programmers and others involved in development to communicate ideas, design algorithms, and organize thoughts without worrying about the exact syntax of a programming language. The psuedocode for GymMember, RegularMember, PremiumMember and GymGUI classes are given below:

Pseudocode of GymMember

CREATE a parent abstract class GymMember
DO
    DECLARE instance variable id as int using protected access modifier
    DECLARE instance variable name as string using protected access modifier
    DECLARE instance variable location as string using protected access modifier
    DECLARE instance variable phone as string using protected access modifier
    DECLARE instance variable email as string using protected access modifier
    DECLARE instance variable gender as string using protected access modifier
    DECLARE instance variable DOB as string using protected access modifier
    DECLARE instance variable membershipStartDate as string using protected access `
        modifier
    DECLARE instance variable attendance as int using protected access modifier
    DECLARE instance variable loyaltyPoints as double using protected access modifier
    DECLARE instance variable activeStatus as Boolean using protected access modifier

CREATE constructor GymMember (id, name, location, phone, email, gender, DOB, membershipStartDate
DO
    INITIALIZE instance variable id passed in parameter

INITIALIZE instance variable name passed in parameter
INITIALIZE instance variable location passed in parameter
INITIALIZE instance variable phone passed in parameter
INITIALIZE instance variable email passed in parameter
INITIALIZE instance variable gender passed in parameter
INITIALIZE instance variable DOB passed in parameter
INITIALIZE instance variable membershipStartDate passed in parameter
SET instance variable attendance to 0
SET instance variable loyalty point to 0.0
SET instance variable activeStatus to false
END DO

DECLARE abstract method markAttendance() with return type void
CREATE method activeMembership()
DO
SET instance variable activeStatus to true
END DO

CREATE method deactivateMemberhship()
DO
IF instance variable activeStatus is equal to true
Set instance variable activeStatus to false
ELSE
DISPLAY "Membership is not actived."
END IF
END DO

CREATE a method resetMember() with void return type
DO
SET instance variable activeStatus to false

SET instance variable attendance to 0

SET instance variable loyaltyPoints to 0.0

END DO

CREATE a method display() with void return type

DO

DISPLAY "ID of member is: " + id

DISPLAY "Name of member is: " + name

DISPLAY "Location of member is: " + location

DISPLAY "Phone Number of member is: " + phone

DISPLAY "Email of member is: " + email

DISPLAY "Gender of member is: " + gender

DISPLAY "DOB: " + DOB

DISPLAY "Membership Start Date: " + membershipStartDate

DISPLAY "Attendance: " + attendance

DISPLAY "Loyalty Points: " + loyaltyPoints

DISPLAY "Active Status: " + activeStatus

END DO

END CLASS

RegularMember Class

CREATE a child class RegularMember that extends GymMember

DO

DECLARE and INITIALIZE instance variable attendanceLimit to 30 as int using private access modifier and final keyword

DECLARE instance variable isEligibleForUpgrade as boolean using private access modifier

DECLARE instance variable removalReason as string using private access modifier

DECLARE instance variable referralSource as string using private access modifier

DECLARE instance variable plan as string using private access modifier

DECLARE instance variable price as double using private access modifier

END DO

CREATE RegularMember constructor (id, name, location, phone, email, gender, DOB, membershipStartDate, referralSource)
DO
CALL parent class constructor (id, name, location, phone, email, gender, DOB, membershipStartDate)
INITIALIZE instance variable isEligibleForUpgrade to false
INITIALIZE instance variable removalReason to “”
INITIALIZE instance variable referralSource passed in parameter
INITIALIZE instance variable plan to basic
INITIALIZE instance variable price to 6500
END DO
CREATE a method markAttendance() with return type void which override method of parent class
DO
Increment parent variable attendance by 1
Increment parent variable loyaltyPoints by 5
IF parent variable attendance is greater or equal to attendanceLimit
SET instance variable isEligibleForUpgrade to true
END IF
END DO
CREATE a method getPlanPrice(String plan) with return type double
DO
SWITCH (plan which is in lower case)
DO
case “basic”:
return 6500
END CASE
case “standard”:
return 12500

END CASE

case “deluxe”

return 18500

END CASE

default:

return -1

END DO

END SWITCH

END DO

CREATE a method setPlan(String plan) with return type void

DO

SET instance variable plan to plan

END DO

CREATE a method upgradePlan(String plan) with return type String

DO

IF instance variable plan which equalsIgnoreCase(plan)

DO

return "You are already subscribed to the " + plan + " plan."

END DO

END IF

IF parent variable attendance is greater or equal to attendanceLimit

DO

SET instance variable isEligibleForUpgrade to true

END DO

END IF

IF not isEligibleForUpgrade

DO

return "Not eligible for upgrade yet. Attendance must reach " + attendanceLimit + "."

END DO

END IF

DECLARE and **INITIALIZE** variable newPrice with return type double to getPlanPrice(plan)
IF newPrice is equal to -1
DO
return "Invalid plan selected."
END DO
END IF
SET instance variable plan to plan
SET instance variable price to newPrice
return "Plan upgraded to " + plan + ". New price: Rs. " + newPrice

CREATE a method revertRegularMember(String removalReason) with return type void
DO
CALL parent method resetMember()
SET instance variable isEligibleForUpgrade to false
SET instance variable plan to “basic’
SET instance variable price to 6500
SET instance variable removalReason to removalReason
END DO

CREATE a method display() with return type void which override the method of GymMember
DO
CALL parent method display()
DISPLAY "Plan: " + plan
DISPLAY "Price: " + price
IF not removalReason which is isEmpty()
DO
DISPLAY "Removal Reason: " + removalReason
END DO
END IF
DISPLAY “”
END DO
END CLASS

Pseudocode of PremiumMember

CREATE a child class PremiumMember that extends GymMember
DO

DECLARE and **INITIALIZE** instance variable premiumCharge to 50000.0 as double using private access modifier and final keyword
DECLARE instance variable personalTrainer as String using private access modifier
DECLARE instance variable isFullPayment as boolean using private access modifier
DECLARE instance variable paidAmount as double using private access modifier
DECLARE instance variable discountAmount as double using private access modifier

CREATE PremiumMember constructor (id, name, location, phone, email, gender, DOB, membershipStartDate, personalTrainer)
DO

CALL parent class constructor (id, name, location, phone, email, gender, DOB, membershipStartDate)
INITIALIZE instance variable personalTrainer to parameter personalTrainer
INITIALIZE instance variable isFullPayment to false
INITIALIZE instance variable paidAmount to 0.0
INITIALIZE instance variable discountAmount to 0.0

END DO
CREATE a method markAttendance() with return type void which overrides parent class
DO

INCREMENT parent variable attendance by 1
INCREMENT parent variable loyaltyPoints by 10

END DO

CREATE a method payDueAmount(double paidAmount) with return type String
DO

IF instance variable isFullPayment is true

DO

return "Payment is already fully completed."

END DO

IF sum of instance variable paidAmount and paidAmount is greater than premiumCharge

DO

DECLARE and **INITIALIZE** variable excess to (sum of instance variable paidAmount and paidAmount) – premiumCharge with return type double

RETURN "Excess amount detected of Rs. " + excess

END DO

INCREMENT instance variable paidAmount by paidAmount

INITIALIZE remainingAmount with return type double to difference of premiumCharge and instance variable paidAmount

IF remainingAmount is equal to 0

DO

SET instance variable isFullPayment to true

RETURN "Payment successful. No remaining amount."

ELSE

RETURN "Payment successful. Remaining amount: Rs. " + remainingAmount

END IF

END DO

CREATE a method calculateDiscount() with return type void

DO

IF isFullPayment is true

DO

SET instance variable discountAmount to 0.10 * premiumCharge

DISPLAY "Discounted amount is Rs.: " + discountAmount

END DO

ELSE

```
        DO
            DISPLAY "No discount is given as payment is not full."
        END DO
        END IF
    END DO

    CREATE a method revertPremiumMember() with return type void
    DO
        CALL parent method resetMember()
        SET instance variable personalTrainer to ""
        SET instance variable isFullPayment to false
        SET instance variable paidAmount to 0.0
        SET instance variable discountAmount to 0.0
    END DO

    CREATE a method display() with return type void which overrides parent class
    DO
        CALL parent method display()
        DISPLAY "Personal Trainer: " + personalTrainer
        DISPLAY "Paid Amount: " + paidAmount
        DISPLAY "Full Payment: " + isFullPayment
        double remainingAmount = premiumCharge - paidAmount
        DISPLAY "Remaining Amount: Rs. " + remainingAmount
        IF instance variable isFullPayment is true
        DO
            DISPLAY "Discount Amount: Rs. " + discountAmount
        END IF
        DISPLAY "discountAmount"
    END DO
END DO
```

END CLASS

Pseudocode of GymGUI

CREATE a class GymGUI that implements ActionListener
DO

DECLARE membersObj as ArrayList<GymMember> with access modifier private
DECLARE id as int
DECLARE name, location, phone, email, gender, DOb, msd, trainer, referral as String

DECLARE frame as JFrame with access modifier private
DECLARE loginPanel, buttonPanel, mainPanel as JPanel with access modifier private
DECLARE homePanel, **ADD**MemberPanel, regularPanel, premiumPanel, activateDeactivatePanel, markAttendancePanel, upgradePlanPanel, discountPanel, revertMemberPanel, payDuePanel as JPanel

DECLARE userTxt as JTextField with access modifier private
DECLARE passwordTxt as JPasswordField with access modifier private
DECLARE showPassword as JCheckBox with access modifier private
DECLARE loginButton as JButton with access modifier private

DECLARE **ADD**MemberBtn, activateBtn, markAttendanceNavBtn, upgradeBtn, discountBtn, payDueNavBtn, revertMemberNavBtn, logOutBtn as JButton with access modifier private

DECLARE ADD RegularSwitchBtn, **ADD**PremiumSwitchBtn as JButton

DECLARE rDisplayInfoFrame as JFrame
DECLARE rIdField, rNameField, rLocationField, rPhoneField, rEmailField, rReferralField, rPriceField as JTextField
DECLARE rMaleBtn, rFemaleBtn as JRadioButton

DECLARE rdobYear, rdobMonth, rdobDay, rMsYear, rMsMonth, rMsDay, rPlanBox as JComboBox<String>
DECLARE r**ADD**MemberBtn, rDisplayBtn, rClearBtn as JButton

DECLARE pDisplayInfoFrame as JFrame
DECLARE pIdField, pNameField, pLocationField, pPhoneField, pEmailField, pTrainerField, pChargeField as JTextField
DECLARE pMaleBtn, pFemaleBtn as JRadioButton
DECLARE pdobYear, pdobMonth, pdobDay, pMsYear, pMsMonth, pMsDay as JComboBox<String>
DECLARE p**ADD**MemberBtn, pDisplayBtn, pClearBtn as JButton

DECLARE actIdField as JTextField
DECLARE activateMemberBtn, deactivateMemberBtn as JButton

DECLARE markIdField as JTextField
DECLARE markRegularBtn, markPremiumBtn as JButton

DECLARE upIdField as JTextField
DECLARE upPlanBox as JComboBox<String>
DECLARE upgradePlanBtn as JButton

DECLARE discIdField, discAmountField as JTextField
DECLARE calcDiscountBtn as JButton

DECLARE revertIdField, revertReasonField as JTextField
DECLARE revertRegularBtn, revertPremiumBtn as JButton

DECLARE payIdField, payAmountField as JTextField
DECLARE payDueButton as JButton

DECLARE and **INITIALIZE** yearsArr as Array of String to {"2025", "2024", "2023", "2022", "2021", "2020", "2019", "2018", "2017", "2016", "2015", "2010", "2000", "1995", "1990", "1980", "1970", "1960", "1950"}

DECLARE and **INITIALIZE** monthsArr as Array of String to "01", "02", "03", "04", "05", "06", "07", "08", "09", "10", "11", "12"

DECLARE and **INITIALIZE** daysArr as Array of String to "01", "02", "03", "04", "05", "06", "07", "08", "09", "10", "11", "12", "13", "14", "15", "16", "17", "18", "19", "20", "21", "22", "23", "24", "25", "26", "27", "28", "29", "30", "31"

DECLARE and **INITIALIZE** planOptionsArr as Array of String to {"Basic","Standard","Deluxe"}

CREATE constructor GymGUI () with no parameter

DO

INITIALIZE frame with title "Aldenaire Gym"

SET frame size to (890,910)

SET frame layout to null

SET frame default close operation to EXIT_ON_CLOSE

SET frame resizable to false

INITIALIZE membersObj as new ArrayList of GymMember

INITIALIZE loginPanel to new JPanel

SET loginPanel layout to null

SET loginPanel bounds to (0, 0, 890, 910)

SET loginPanel background color to black from Color Package

DECLARE and **INITIALIZE** adminLoginTxt as new Jlabel with title "Admin Login"

SET adminLoginTxt foreground color to white from Color Package

SET adminLoginTxt bounds to (350, 50, 300, 50)

SET adminLoginTxt font properties to ("Arial", Font.PLAIN, 25)) from Font Package
SET loginPanel background color to black

CREATE and **ADD** loginImg as ImageIcon
DECLARE and **INITIALIZE** loginImgLbl as new JLabel(loginImg)
SET loginImgLbl bounds to (200, 100, 550, 359)

DECLARE and **INITIALIZE** userLabel as JLabel("Username")
SET userLabel foreground color to white from Font package
SET userLabel bounds to (350, 50, 300, 50)
SET userLabel font properties to ("Arial", Font.PLAIN, 25)) from Font package
ADD userLabel to loginPanel

DECLARE passLabel as JLabel with text "Password"
SET passLabel foreground color to white
SET passLabel bounds to (150, 600, 200, 35)
SET passLabel font to ("Arial", PLAIN, 25)
ADD passLabel to loginPanel

DECLARE passwordTxt as JPasswordField
SET passwordTxt bounds to (360, 600, 350, 35)
SET passwordTxt font to ("Arial", PLAIN, 25)
ADD passwordTxt to loginPanel

INTIALIZE showPassword as JCheckBox with text "Show Password"
SET showPassword bounds to (360, 660, 350, 35)
SET showPassword font to ("Arial", PLAIN, 25)
SET showPassword background color to black
SET showPassword foreground color to white
ADD ActionListener instance ro showPassword

ADD showPassword ro loginPanel

INITIALIZE loginButton as JButton with text"Login"
SET loginButton bounds to (360, 720, 200, 40)
SET loginButton font to ("Arial", BOLD, 25)
SET loginButton background to GREEN
ADD ActionListener instance to loginButton
CALL buttonPointer in loginButton
ADD loginButton to loginPanel

SET frame CONTENT PANE to loginPanel

DECLARE buttonPanel as JPanel
SET buttonPanel layout to null
SET buttonPanel bounds to (0, 0, 200, 910)
SET buttonPanel background to Color(0x80490E)

DECLARE ADD MemberBtn as JButton with text"**ADD** member"
SET ADD MemberBtn bounds to (10, 50, 180, 40)
SET ADD MemberBtn font to ("SansSerif", BOLD, 15)
SET ADD MemberBtn background to Color(0xB4B8AC)
ADD ActionListener instance to **ADD**MemberBtn
CALL buttonPointer to **ADD**MemberBtn
ADD ADD MemberBtn to buttonPanel

DECLARE activateBtn as JButton with text"Activate/Deactivate"
SET activateBtn bounds to (10, 130, 180, 40)
SET activateBtn font to ("SansSerif", BOLD, 15)
SET activateBtn background to Color(0xB4B8AC)
ADD ActionListener instance to activateBtn

CALL buttonPointer with activateBtn
ADD activateBtn to buttonPanel

DECLARE markAttendanceNavBtn as JButton with text"Mark Attendance"
SET markAttendanceNavBtn bounds to (10, 210, 180, 40)
SET markAttendanceNavBtn font to ("SansSerif", BOLD, 15)
SET markAttendanceNavBtn background to Color(0xB4B8AC)
ADD ActionListener instance to markAttendanceNavBtn
CALL buttonPointer with markAttendanceNavBtn
ADD markAttendanceNavBtn to buttonPanel

DECLARE upgradeBtn as JButton with text"Upgrade Plan"
SET upgradeBtn bounds to (10, 290, 180, 40)
SET upgradeBtn font to ("SansSerif", BOLD, 15)
SET upgradeBtn background to Color(0xB4B8AC)
ADD ActionListener instanceto upgradeBtn
CALL buttonPointer instance upgradeBtn
ADD upgradeBtn to buttonPanel

DECLARE discountBtn as JButton with text"Calculate Discount"
SET discountBtn bounds to (10, 370, 180, 40)
SET discountBtn font to ("SansSerif", BOLD, 15)
SET discountBtn background to Color(0xB4B8AC)
ADD ActionListener instance to discountBtn
CALL buttonPointer with discountBtn
ADD discountBtn to buttonPanel

DECLARE payDueNavBtn as JButton with text"Pay Due"
SET payDueNavBtn bounds to (10, 450, 180, 40)
SET payDueNavBtn font to ("SansSerif", BOLD, 15)

SET payDueNavBtn background to Color(0xB4B8AC)
ADD ActionListener instance to payDueNavBtn
CALL buttonPointer with payDueNavBtn
ADD payDueNavBtn to buttonPanel

DECLARE revertMemberNavBtn as JButton with text"Revert Member"
SET revertMemberNavBtn bounds to (10, 530, 180, 40)
SET revertMemberNavBtn font to ("SansSerif", BOLD, 15)
SET revertMemberNavBtn background to Color(0xB4B8AC)
ADD ActionListener instance to revertMemberNavBtn
CALL buttonPointer with revertMemberNavBtn
ADD revertMemberNavBtn to buttonPanel

DECLARE saveBtn as JButton with text"Save to File"
SET saveBtn bounds to (10, 610, 180, 40)
SET saveBtn font to ("SansSerif", BOLD, 15)
SET saveBtn background to Color(0xB4B8AC)
ADD ActionListener instance to saveBtn
CALL buttonPointer with saveBtn
ADD saveBtn to buttonPanel

DECLARE readBtn as JButton with text"Read from File"
SET readBtn bounds to (10, 690, 180, 40)
SET readBtn font to ("SansSerif", BOLD, 15)
SET readBtn background to Color(0xB4B8AC)
ADD ActionListener instanceto readBtn
CALL buttonPointer with readBtn
ADD readBtn to buttonPanel

DECLARE logOutBtn as JButton with text"Log Out"

SET logOutBtn bounds to (10, 770, 180, 40)
SET logOutBtn font to ("SansSerif", BOLD, 15)
SET logOutBtn background to Color(0xB4B8AC)
SET logOutBtn foreground to RED
ADD ActionListener instanceto logOutBtn
CALL buttonPointer instance logOutBtn
ADD logOutBtn to buttonPanel

INTIALIZE mainPanel as JPanel
SET mainPanel layout to null
SET mainPanel bounds to (0, 0, 690, 910)
SET mainPanel background to BLACK

INITIALZE homePanel as JPanel
SET homePanel layout to null
SET homePanel bounds to (0, 0, 690, 910)
SET homePanel background to Black

CREATE homeImg as ImageIcon("homeImg.png")
DECLARE and **INITIALIZE** homeImgLbl as JLabel(homeImg)
SET homeImgLbl bounds to (0, 370, 775, 500)

CREATE logoHomeImg as ImageIcon("logoImg.png")
DECLARE and **INITIALIZE** logoHomeImgLbl as JLabel(logoHomeImg)
SET logoHomeImgLbl bounds to (125, 50, 450, 450)
ADD homeImgLbl to homePanel
ADD logoHomeImgLbl to homePanel
ADD homePanel to mainPanel

INITIALIZE addMemberPanel as JPanel

SET addMemberPanel layout to null
SET addMemberPanel bounds to (0, 0, 690, 910)
SET addMemberPanel background to Black

INITIALIZE addMemLabel as JLabel("Add Member")
SET addMemLabel foreground to Color.WHITE
SET addMemLabel font to ("Arial", BOLD, 37)
SET addMemLabel bounds to (225, 50, 400, 50)
ADD addMemLabel to addMemberPanel

INITIALIZE addRegularSwitchBtn as JButton("ADD Regular Member")
SET addRegularSwitchBtn bounds to (175, 180, 325, 40)
SET addRegularSwitchBtn font to ("SansSerif", BOLD, 20)
ADD ActionListener to addRegularSwitchBtn
CALL buttonPointer(addRegularSwitchBtn)
ADD addRegularSwitchBtn to addMemberPanel

INITIALIZE addPremiumSwitchBtn as JButton("ADD Premium Member")
SET addPremiumSwitchBtn bounds to (175, 270, 325, 40)
SET addPremiumSwitchBtn font to ("SansSerif", BOLD, 20)
ADD ActionListener to addPremiumSwitchBtn
CALL buttonPointer(addPremiumSwitchBtn)
ADD addPremiumSwitchBtn to addMemberPanel
ADD addMemberPanel to mainPanel

INITIALIZE regularPanel as JPanel
SET regularPanel layout to null
SET regularPanel bounds to (0, 0, 690, 910)
SET regularPanel background to Black
DECLARE regHeading as JLabel("Regular Member")

SET regHeading foreground to Color.WHITE
SET regHeading font to ("Arial", BOLD, 37)
SET regHeading bounds to (225, 50, 400, 50)
ADD regHeading to regularPanel

DECLARE rIdLbl as JLabel("Id")
SET rIdLbl foreground to Color.WHITE
SET rIdLbl font to ("Arial", PLAIN, 16)
SET rIdLbl bounds to (70, 120, 100, 30)
ADD rIdLbl to regularPanel
DECLARE rIdField as JTextField
SET rIdField bounds to (70, 150, 200, 24)
ADD rIdField to regularPanel

INITIALIZE rNameLbl as JLabel("Name")
SET rNameLbl foreground to Color.WHITE
SET rNameLbl bounds to (350, 120, 100, 30)
SET rNameLbl font to ("Arial", PLAIN, 16)
ADD rNameLbl to regularPanel

INITIALIZE rNameField as JTextField
SET rNameField bounds to (350, 150, 200, 24)
ADD rNameField to regularPanel

INITIALIZE rLocLbl as JLabel("Location")
SET rLocLbl foreground to Color.WHITE
SET rLocLbl font to ("Arial", PLAIN, 16)
SET rLocLbl bounds to (70, 190, 100, 30)
ADD rLocLbl to regularPanel

INITIALIZE rLocationField as JTextField
SET rLocationField bounds to (70, 220, 200, 24)
ADD rLocationField to regularPanel

DECLARE rPhoneLbl as JLabel("Phone Number")
SET rPhoneLbl foreground to Color.WHITE
SET rPhoneLbl font to ("Arial", PLAIN, 16)
SET rPhoneLbl bounds to (350, 190, 150, 30)
ADD rPhoneLbl to regularPanel

DECLARE rPhoneField as JTextField
SET rPhoneField bounds to (350, 220, 200, 24)
ADD rPhoneField to regularPanel

DECLARE rEmailLbl as JLabel("Email")
SET rEmailLbl foreground to Color.WHITE
SET rEmailLbl font to ("Arial", PLAIN, 16)
SET rEmailLbl bounds to (70, 260, 100, 30)
ADD rEmailLbl to regularPanel

DECLARE rEmailField as JTextField
SET rEmailField bounds to (70, 290, 350, 24)
ADD rEmailField to regularPanel

DECLARE rGenderLbl as JLabel("Gender")
SET rGenderLbl foreground to Color.WHITE
SET rGenderLbl font to ("Arial", PLAIN, 16)
SET rGenderLbl bounds to (70, 330, 100, 30)
ADD rGenderLbl to regularPanel

DECLARE rMaleBtn as JRadioButton("Male")
SET rMaleBtn font to ("Arial", PLAIN, 16)
SET rMaleBtn background to Black
SET rMaleBtn foreground to Color.WHITE
SET rMaleBtn bounds to (70, 360, 100, 30)
ADD rMaleBtn to regularPanel

DECLARE rFemaleBtn as JRadioButton("Female")
SET rFemaleBtn font to ("Arial", PLAIN, 16)
SET rFemaleBtn background to Black
SET rFemaleBtn foreground to Color.WHITE
SET rFemaleBtn bounds to (180, 360, 100, 30)
ADD rFemaleBtn to regularPanel
DECLARE rGenderBG as ButtonGroup
ADD rMaleBtn to rGenderBG
ADD rFemaleBtn to rGenderBG

DECLARE rDObYear as JComboBox(yearsArr)
SET rDObYear bounds to (70, 430, 80, 30)
ADD rDObYear to regularPanel

DECLARE rDObMonth as JComboBox(monthsArr)
SET rDObMonth bounds to (170, 430, 60, 30)
ADD rDObMonth to regularPanel

DECLARE rDObDay as JComboBox(daysArr)
SET rDObDay bounds to (250, 430, 60, 30)
ADD rDObDay to regularPanel

DECLARE rMsYear as JComboBox(yearsArr)

SET rMsYear bounds to (70, 500, 80, 30)
ADD rMsYear to regularPanel

DECLARE rMsMonth as JComboBox(monthsArr)
SET rMsMonth bounds to (170, 500, 60, 30)
ADD rMsMonth to regularPanel

DECLARE rMsDay as JComboBox(daysArr)
SET rMsDay bounds to (250, 500, 60, 30)
ADD rMsDay to regularPanel

DECLARE rRefLbl as JLabel("Referral")
SET rRefLbl foreground to Color.WHITE
SET rRefLbl font to ("Arial", PLAIN, 16)
SET rRefLbl bounds to (70, 540, 100, 30)
ADD rRefLbl to regularPanel

DECLARE rReferralField as JTextField
SET rReferralField bounds to (70, 570, 200, 24)
ADD rReferralField to regularPanel

DECLARE rDisplayBtn as JButton("Display")
SET rDisplayBtn bounds to (70, 680, 175, 40)
SET rDisplayBtn font to ("Arial", BOLD, 16)
SET rDisplayBtn background to Color(0xB4B8AC)
ADD ActionListener to rDisplayBtn
CALL buttonPointer(rDisplayBtn)
ADD rDisplayBtn to regularPanel

DECLARE rClearBtn as JButton("Clear")

SET rClearBtn bounds to (350, 680, 175, 40)
SET rClearBtn font to ("Arial", BOLD, 16)
SET rClearBtn background to Color(0xB4B8AC)
ADD ActionListener to rClearBtn
CALL buttonPointer(rClearBtn)
ADD rClearBtn to regularPanel

DECLARE r**ADD**MemberBtn as JButton("**ADD** member")
SET r**ADD**MemberBtn bounds to (475, 800, 175, 40)
SET r**ADD**MemberBtn background to Color.GREEN
SET r**ADD**MemberBtn font to ("Arial", BOLD, 16)
ADD ActionListener to r**ADD**MemberBtn
CALL buttonPointer(r**ADD**MemberBtn)
ADD r**ADD**MemberBtn to regularPanel
ADD regularPanel to mainPanel

INITIALIZE premiumPanel as JPanel
SET premiumPanel layout to null
SET premiumPanel bounds to (0, 0, 690, 910)
SET premiumPanel background to Black
DECLARE preHeading as JLabel("Premium Member")
SET preHeading foreground to Color.WHITE
SET preHeading font to ("Arial", BOLD, 37)
SET preHeading bounds to (225, 50, 400, 50)
ADD preHeading to premiumPanel

DECLARE pIdLbl as JLabel("Id")
SET pIdLbl foreground to Color.WHITE
SET pIdLbl font to ("Arial", PLAIN, 16)
SET pIdLbl bounds to (70, 120, 100, 30)

ADD pIdLbl to premiumPanel

DECLARE pIdField as JTextField
SET pIdField bounds to (70, 150, 200, 24)
ADD pIdField to premiumPanel

DECLARE premiumPanel as JPanel
INITIALIZE premiumPanel
SET premiumPanel layout to null
SET premiumPanel bounds to (0, 0, 690, 910)
SET premiumPanel background to Color.BLACK
ADD premiumPanel to mainPanel

DECLARE preHeading as JLabel("Premium Member")
SET preHeading foreground to Color.WHITE
SET preHeading font to ("Arial", BOLD, 37)
SET preHeading bounds to (225, 50, 400, 50)
ADD preHeading to premiumPanel

DECLARE pIdLbl as JLabel("Id")
SET pIdLbl foreground to Color.WHITE
SET pIdLbl font to ("Arial", PLAIN, 16)
SET pIdLbl bounds to (70, 120, 100, 30)
ADD pIdLbl to premiumPanel

DECLARE pIdField as JTextField
SET pIdField bounds to (70, 150, 200, 24)
ADD pIdField to premiumPanel

DECLARE pNameLbl as JLabel("Name")

SET pNameLbl foreground to Color.WHITE
SET pNameLbl font to ("Arial", PLAIN, 16)
SET pNameLbl bounds to (350, 120, 100, 30)

ADD pNameLbl to premiumPanel
DECLARE pNameField as JTextField
SET pNameField bounds to (350, 150, 200, 24)
ADD pNameField to premiumPanel

DECLARE pLocLbl as JLabel("Location")
SET pLocLbl foreground to Color.WHITE
SET pLocLbl font to ("Arial", PLAIN, 16)
SET pLocLbl bounds to (70, 190, 100, 30)
ADD pLocLbl to premiumPanel

DECLARE pLocationField as JTextField
SET pLocationField bounds to (70, 220, 200, 24)
ADD pLocationField to premiumPanel

DECLARE pPhoneLbl as JLabel("Phone Number")
SET pPhoneLbl foreground to Color.WHITE
SET pPhoneLbl font to ("Arial", PLAIN, 16)
SET pPhoneLbl bounds to (350, 190, 150, 30)
ADD pPhoneLbl to premiumPanel

DECLARE pPhoneField as JTextField
SET pPhoneField bounds to (350, 220, 200, 24)
ADD pPhoneField to premiumPanel

DECLARE pEmailLbl as JLabel("Email")

SET pEmailLbl foreground to Color.WHITE
SET pEmailLbl font to ("Arial", PLAIN, 16)
SET pEmailLbl bounds to (70, 260, 100, 30)
ADD pEmailLbl to premiumPanel

DECLARE pEmailField as JTextField
SET pEmailField bounds to (70, 290, 350, 24)
ADD pEmailField to premiumPanel

DECLARE pGenderLbl as JLabel("Gender")
SET pGenderLbl foreground to Color.WHITE
SET pGenderLbl font to ("Arial", PLAIN, 16)
SET pGenderLbl bounds to (70, 330, 100, 30)
ADD pGenderLbl to premiumPanel

DECLARE pMaleBtn as JRadioButton("Male")
SET pMaleBtn font to ("Arial", PLAIN, 16)
SET pMaleBtn background to Color.BLACK
SET pMaleBtn foreground to Color.WHITE
SET pMaleBtn bounds to (70, 360, 100, 30)
ADD pMaleBtn to premiumPanel

DECLARE pFemaleBtn as JRadioButton("Female")
SET pFemaleBtn font to ("Arial", PLAIN, 16)
SET pFemaleBtn background to Color.BLACK
SET pFemaleBtn foreground to Color.WHITE
SET pFemaleBtn bounds to (180, 360, 100, 30)
ADD pFemaleBtn to premiumPanel

DECLARE pGenderBG as ButtonGroup

ADD pMaleBtn to pGenderBG
ADD pFemaleBtn to pGenderBG

DECLARE pDObLbl as JLabel("Date Of Birth (YYYY/MM/DD)")
SET pDObLbl foreground to WHITE
SET pDObLbl font to ("Arial", PLAIN, 16)
SET pDObLbl bounds to (70, 400, 250, 30)
ADD pDObLbl to premiumPanel

DECLARE pDObYear as JComboBox(yearsArr)
SET pDObYear bounds to (70, 430, 80, 30)
ADD pDObYear to premiumPanel
DECLARE pDObMonth as JComboBox(monthsArr)
SET pDObMonth bounds to (170, 430, 60, 30)

ADD pDObMonth to premiumPanel
DECLARE pDObDay as JComboBox(daysArr)
SET pDObDay bounds to (250, 430, 60, 30)
ADD pDObDay to premiumPanel

DECLARE pMsLbl as JLabel("Member Start Date (YYYY/MM/DD)")
SET pMsLbl foreground to Color.WHITE
SET pMsLbl font to ("Arial", PLAIN, 16)
SET pMsLbl bounds to (70, 470, 300, 30)
ADD pMsLbl to premiumPanel

DECLARE pMsYear as JComboBox(yearsArr)
SET pMsYear bounds to (70, 500, 80, 30)
ADD pMsYear to premiumPanel

DECLARE pMsMonth as JComboBox(monthsArr)
SET pMsMonth bounds to (170, 500, 60, 30)
ADD pMsMonth to premiumPanel

DECLARE pMsDay as JComboBox(daysArr)
SET pMsDay bounds to (250, 500, 60, 30)
ADD pMsDay to premiumPanel

DECLARE tLbl as JLabel("Trainer")
SET tLbl foreground to Color.WHITE
SET tLbl font to ("Arial", PLAIN, 16)
SET tLbl bounds to (70, 540, 100, 30)
ADD tLbl to premiumPanel
DECLARE pTrainerField as JTextField
SET pTrainerField bounds to (70, 570, 200, 24)
ADD pTrainerField to premiumPanel

DECLARE chargeLbl as JLabel("Premium Charge")
SET chargeLbl foreground to Color.WHITE
SET chargeLbl font to ("Arial", PLAIN, 16)
SET chargeLbl bounds to (350, 540, 200, 30)
ADD chargeLbl to premiumPanel

DECLARE pChargeField as JTextField("50000")
SET pChargeField bounds to (350, 570, 150, 24)
SET pChargeField EDITABLE to false
ADD pChargeField to premiumPanel

DECLARE pDisplayBtn as JButton("Display")
SET pDisplayBtn bounds to (70, 650, 175, 40)

SET pDisplayBtn font to ("Arial", BOLD, 18)
SET pDisplayBtn background to Color(0xB4B8AC)
ADD ActionListener to pDisplayBtn
CALL buttonPointer(pDisplayBtn)
ADD pDisplayBtn to premiumPanel

DECLARE pClearBtn as JButton("Clear")
SET pClearBtn bounds to (350, 650, 175, 40)
SET pClearBtn font to ("Arial", BOLD, 18)
SET pClearBtn background to Color(0xB4B8AC)
ADD ActionListener to pClearBtn
CALL buttonPointer(pClearBtn)
ADD pClearBtn to premiumPanel

DECLARE p**ADD**MemberBtn as JButton("**ADD** member")
SET p**ADD**MemberBtn bounds to (475, 800, 175, 40)
SET p**ADD**MemberBtn background to Color.GREEN
SET p**ADD**MemberBtn font to ("Arial", BOLD, 18)
ADD ActionListener to p**ADD**MemberBtn
CALL buttonPointer(p**ADD**MemberBtn)
ADD p**ADD**MemberBtn to premiumPanel

DECLARE activateDeactivatePanel as JPanel
INITIALIZE activateDeactivatePanel
SET activateDeactivatePanel layout to null
SET activateDeactivatePanel bounds to (0, 0, 690, 910)
SET activateDeactivatePanel background to Color.BLACK
ADD activateDeactivatePanel to mainPanel

DECLARE actHeading as JLabel("Activate/Deactivate Membership")

SET actHeading foreground to Color.WHITE
SET actHeading font to ("Arial", BOLD, 30)
SET actHeading bounds to (110, 50, 500, 50)
ADD actHeading to activateDeactivatePanel

DECLARE actIdLbl as JLabel("Member Id")
SET actIdLbl foreground to Color.WHITE
SET actIdLbl font to ("Arial", PLAIN, 16)
SET actIdLbl bounds to (100, 150, 100, 30)
ADD actIdLbl to activateDeactivatePanel

DECLARE actIdField as JTextField
SET actIdField bounds to (100, 180, 200, 24)
ADD actIdField to activateDeactivatePanel
DECLARE activateMemberBtn as JButton("Activate")
SET activateMemberBtn bounds to (100, 250, 175, 40)
SET activateMemberBtn font to ("Arial", BOLD, 18)
SET activateMemberBtn background to green from COLOR package
ADD ActionListener to activateMemberBtn
CALL buttonPointer(activateMemberBtn)
ADD activateMemberBtn to activateDeactivatePanel

DECLARE deactivateMemberBtn as JButton("Deactivate")
SET deactivateMemberBtn bounds to (360, 250, 175, 40)
SET deactivateMemberBtn font to ("Arial", BOLD, 18)
SET deactivateMemberBtn background to Color.RED
SET deactivateMemberBtn foreground to Color.WHITE
ADD ActionListener to deactivateMemberBtn
CALL buttonPointer(deactivateMemberBtn)
ADD deactivateMemberBtn to activateDeactivatePanel

DECLARE markAttendancePanel as JPanel
INITIALIZE markAttendancePanel
SET markAttendancePanel layout to null
SET markAttendancePanel bounds to (0, 0, 690, 910)
SET markAttendancePanel background to Color.BLACK
ADD markAttendancePanel to mainPanel

DECLARE markHeading as JLabel("Mark Attendance")
SET markHeading foreground to Color.WHITE
SET markHeading font to ("Arial", BOLD, 37)
SET markHeading bounds to (225, 50, 400, 50)
ADD markHeading to markAttendancePanel

DECLARE markIdLbl as JLabel("Member Id")
SET markIdLbl foreground to Color.WHITE
SET markIdLbl font to ("Arial", PLAIN, 16)
SET markIdLbl bounds to (100, 150, 100, 30)
ADD markIdLbl to markAttendancePanel

DECLARE markIdField as JTextField
SET markIdField bounds to (100, 180, 200, 24)
ADD markIdField to markAttendancePanel

INITIALIZE markRegularBtn as new JButton with text“Mark Regular"
SET markRegularBtn bounds to (100, 250, 175, 40)
SET markRegularBtn font to ("Arial", BOLD, 18)
SET markRegularBtn background color to Color(0xB4B8AC)
ADD ActionListener instance to markRegularBtn
CALL buttonPointer(markRegularBtn)

ADD markRegularBtn to markAttendancePanel

DECLARE markPremiumBtn as JButton("Mark Premium")
SET markPremiumBtn bounds to (360, 250, 175, 40)
SET markPremiumBtn font to ("Arial", BOLD, 18)
SET markPremiumBtn background to Color(0xB4B8AC)
ADD ActionListener to markPremiumBtn
CALL buttonPointer(markPremiumBtn)
ADD markPremiumBtn to markAttendancePanel

ADD markAttendancePanel to mainPanel

INITIALIZE upgradePlanPanel as new JPanel
SET upgradePlanPanel layout to null
SET upgradePlanPanel bounds to (0, 0, 690, 910)
SET upgradePlanPanel background to Color.BLACK

DECLARE upHeading as JLabel("Upgrade Plan")
SET upHeading foreground to Color.WHITE
SET upHeading font to ("Arial", BOLD, 37)
SET upHeading bounds to (225, 50, 400, 50)
ADD upHeading to upgradePlanPanel

DECLARE upIdLbl as JLabel("Member Id")
SET upIdLbl foreground to Color.WHITE
SET upIdLbl font to ("Arial", PLAIN, 16)
SET upIdLbl bounds to (100, 150, 100, 30)
ADD upIdLbl to upgradePlanPanel

INITIALIZE upIdField as JTextField

SET upIdField bounds to (100, 180, 200, 24)
ADD upIdField to upgradePlanPanel

DECLARE upPlanLbl as JLabel("New Plan")
SET upPlanLbl foreground to Color.WHITE
SET upPlanLbl font to ("Arial", PLAIN, 16)
SET upPlanLbl bounds to (100, 230, 100, 30)
ADD upPlanLbl to upgradePlanPanel

DECLARE upPlanBox as JComboBox(planOptionsArr)
SET upPlanBox bounds to (100, 260, 125, 30)
ADD upPlanBox to upgradePlanPanel

DECLARE upgradePlanBtn as JButton("Upgrade")
SET upgradePlanBtn bounds to (100, 320, 175, 40)
SET upgradePlanBtn font to ("Arial", BOLD, 18)
SET upgradePlanBtn background to Color(0xB4B8AC)
ADD ActionListener to upgradePlanBtn
CALL buttonPointer(upgradePlanBtn)
ADD upgradePlanBtn to upgradePlanPanel
ADD upgradePlanPanel to mainPanel

INITIALIZE discountPanel as JPanel
INITIALIZE discountPanel
SET discountPanel layout to null
SET discountPanel bounds to (0, 0, 690, 910)
SET discountPanel background to Color.BLACK

DECLARE and **INITIALIZE** discHeading as JLabel("Calculate Discount")
SET discHeading foreground to Color.WHITE

SET discHeading font to ("Arial", BOLD, 37)
SET discHeading bounds to (225, 50, 400, 50)
ADD discHeading to discountPanel

DECLARE and **INITIALIZE** discIdLbl as JLabel("Member Id")
SET discIdLbl foreground to Color.WHITE
SET discIdLbl font to ("Arial", PLAIN, 16)
SET discIdLbl bounds to (100, 150, 100, 30)
ADD discIdLbl to discountPanel

INITIALIZE discIdField as JTextField
SET discIdField bounds to (100, 180, 200, 24)
ADD discIdField to discountPanel

INITIALIZE calcDiscountBtn as JButton("Calculate")
SET calcDiscountBtn bounds to (100, 240, 175, 40)
SET calcDiscountBtn font to ("Arial", BOLD, 18)
SET calcDiscountBtn background to Color(0xB4B8AC)
ADD ActionListener to calcDiscountBtn
CALL buttonPointer(calcDiscountBtn)
ADD calcDiscountBtn to discountPanel

DECLARE dAmtLbl as JLabel("Discount Amount")
SET dAmtLbl foreground to Color.WHITE
SET dAmtLbl font to ("Arial", PLAIN, 16)
SET dAmtLbl bounds to (100, 300, 200, 30)
ADD dAmtLbl to discountPanel

DECLARE discAmountField as JTextField
SET discAmountField bounds to (100, 330, 200, 24)

SET discAmountField EDITABLE to false
ADD discAmountField to discountPanel
ADD discountPanel to mainPanel

INITIALIZE payDuePanel as JPanel
INITIALIZE payDuePanel
SET payDuePanel layout to null
SET payDuePanel bounds to (0, 0, 690, 910)
SET payDuePanel background to Color.BLACK

DECLARE and **INITIALIZE** payHeading as JLabel("Pay Due Amount")
SET payHeading foreground to Color.WHITE
SET payHeading font to ("Arial", BOLD, 37)
SET payHeading bounds to (225, 50, 400, 50)
ADD payHeading to payDuePanel

DECLARE payIdLbl as JLabel("Member Id")
SET payIdLbl foreground to Color.WHITE
SET payIdLbl font to ("Arial", PLAIN, 16)
SET payIdLbl bounds to (100, 150, 100, 30)
ADD payIdLbl to payDuePanel

DECLARE payIdField as JTextField
SET payIdField bounds to (100, 180, 200, 24)
ADD payIdField to payDuePanel

DECLARE payAmtLbl as JLabel("Amount:")
SET payAmtLbl foreground to Color.WHITE
SET payAmtLbl font to ("Arial", PLAIN, 16)
SET payAmtLbl bounds to (100, 240, 100, 30)

ADD payAmtLbl to payDuePanel

DECLARE payAmountField as JTextField
SET payAmountField bounds to (100, 270, 200, 24)
ADD payAmountField to payDuePanel

DECLARE payDueButton as JButton("Pay Amount")
SET payDueButton bounds to (100, 330, 175, 40)
SET payDueButton font to ("Arial", BOLD, 18)
SET payDueButton background to Color(0xB4B8AC)
ADD ActionListener to payDueButton
CALL buttonPointer(payDueButton)
ADD payDueButton to payDuePanel
ADD payDuePanel to mainPanel

DECLARE revertMemberPanel as JPanel
INITIALIZE revertMemberPanel
SET revertMemberPanel layout to null
SET revertMemberPanel bounds to (0, 0, 690, 910)
SET revertMemberPanel background to Color.BLACK

DECLARE revHeading as JLabel("Revert Member")
SET revHeading foreground to Color.WHITE
SET revHeading font to ("Arial", BOLD, 37)
SET revHeading bounds to (225, 50, 400, 50)
ADD revHeading to revertMemberPanel

DECLARE revIdLbl as JLabel("Member Id")
SET revIdLbl foreground to Color.WHITE
SET revIdLbl font to ("Arial", PLAIN, 16)

SET revIdLbl bounds to (100, 150, 100, 30)
ADD revIdLbl to revertMemberPanel

DECLARE revertIdField as JTextField
SET revertIdField bounds to (100, 180, 200, 24)
ADD revertIdField to revertMemberPanel

DECLARE revertRegularBtn as JButton("Revert Regular")
SET revertRegularBtn bounds to (100, 250, 175, 40)
SET revertRegularBtn font to ("Arial", BOLD, 18)
SET revertRegularBtn background to Color(0xB4B8AC)
ADD ActionListener to revertRegularBtn
CALL buttonPointer(revertRegularBtn)
ADD revertRegularBtn to revertMemberPanel

DECLARE revertPremiumBtn as JButton("Revert Premium")
SET revertPremiumBtn bounds to (360, 250, 175, 40)
SET revertPremiumBtn font to ("Arial", BOLD, 18)
SET revertPremiumBtn background to Color(0xB4B8AC)
ADD ActionListener to revertPremiumBtn

CALL buttonPointer(revertPremiumBtn)
ADD revertPremiumBtn TO revertMemberPanel
ADD revertMemberPanel TO mainPanel

ADD mainPanel to frame
SET frame visibility to true
CALL hideAllSubPanels()

END DO

CREATE method hideAllSubPanels with return type void and access modifier public

DO

SET homePanel visibility to false

SET ADD MemberPanel visibility to false

SET regularPanel visibility to false

SET premiumPanel visibility to false

SET activateDeactivatePanel visibility to false

SET markAttendancePanel visibility to false

SET upgradePlanPanel visibility to false

SET discountPanel visibility to false

SET revertMemberPanel visibility to false

SET payDuePanel visibility to false

END DO

CREATE method buttonPointer(JButton button) with return type void, access modifier private and parameter JButton button

DO

SET button focusable to false

SET button Cursor to new Cursor HAND_CURSOR from Cursor package

END DO

CREATE method searchMemId(int id) RETURNS GymMember

DO

FOR each GymMember gm in membersObj

DO

IF gm which get id equals id

DO

RETURN gm

END DO

END IF

END DO

END FOR

RETURN null

END DO

CREATE method isDuplicateId(int id) with return type boolean, parameter id and access modifier as private

DO

RETURN CALL searchMemId(id) is not equal null

END DO

CREATE method actionPerformed(ActionEvent eve) with return type void, parameter ActionEvent and overrides

DO

DECLARE and **INITIALIZE** btn as Object to eve which get source

IF btn equals loginButton

DO

DECLARE and INITIALIZE user as string to userTxt which getText

DECLARE and INITIALIZE pass as string to new String (passwordTxt which get Password)

IF user equals "admin" and pass equals "admin"

DO

DECLARE and INITIALIZE container as JPanel to new JPanel(null)

SET container bounds to (0, 0, 890, 910)

ADD buttonPanel to container

ADD mainPanel to container

SET mainPanel bounds to (200, 0, 690, 910)

SET frame ContentPane(container)

CALL repaint in frame

CALL revalidate in frame

CALL hideAllSubPanels

SET homePanel visibility to true

END DO

ELSE IF user equals "" or pass equals ""
DO
CALL JOptionPane which show MessageDialog(frame, "Please fill up both text areas!", "Login Message", INFORMATION_MESSAGE)
END DO
ELSE
DO
CALL JOptionPane which show MessageDialog(frame, "Invalid username or password!", "Login Message", ERROR_MESSAGE)
END DO
END IF
END DO
END DO

ELSE IF btn equals showPassword
DO
IF showPassword is Selected
DO
CALL passwordTxt which **SET** EchoChar((char)0)
END DO
ELSE
DO
CALL passwordTxt which **SET** EchoChar('*')
END DO
END IF
END DO

IF btn equals **ADD**MemberBtn
DO
CALL hideAllSubPanels

CALL ADD MemberPanel which **SET** visibility to true

END DO

IF btn equals activateBtn

DO

CALL hideAllSubPanels

CALL activateDeactivatePanel which **SET** visibility to true

END DO

IF btn equals markAttendanceNavBtn

DO

CALL hideAllSubPanels

CALL markAttendancePanel which **SET** visibility to true

END DO

IF btn equals upgradeBtn

DO

CALL hideAllSubPanels

CALL upgradePlanPanel which **SET** visibility to true

END DO

IF btn EQUALS discountBtn

DO

CALL hideAllSubPanels

CALL discountPanel which **SET** visibility to true

END DO

IF btn EQUALS payDueNavBtn

DO

CALL hideAllSubPanels

CALL payDuePanel which **SET** visibility to true

END DO

IF btn EQUALS revertMemberNavBtn

DO

CALL hideAllSubPanels

CALL revertMemberPanel which **SET** visibility to true

END DO

IF btn EQUALS logOutBtn

DO

CALL frame which get ContentPane and removeAll

CALL frame which **SET** ContentPane(loginPanel)

CALL frame which repaint

CALL frame which revalidate

END DO

IF btn EQUALS **ADD**RegularSwitchBtn

DO

CALL hideAllSubPanels()

CALL regularPanel which **SET** visibility to true

END DO

IF btn EQUALS **ADD**PremiumSwitchBtn

DO

CALL hideAllSubPanels()

CALL premiumPanel which **SET** visibility to true

END DO

IF btn equals r**ADD**MemberBtn

DO

IF rIdField which get text and is empty OR rNameField which get text and is empty OR rLocationField which get text and is empty OR rPhoneField which get text and is empty OR rEmailField which get text and is empty OR rReferralField which get text and is empty OR (NOT rMaleBtn isSelected AND NOT rFemaleBtn isSelected)

DO

CALL JOptionPane which showMessageDialog(frame, "Please fill in all required fields for Regular Member.")

RETURN

END DO

DECLARE and **INITIALZE** phoneStr as String to rPhoneField which getText and trim

TRY

DO

CALL Long which parseLong(phoneStr)

END DO

CATCH NumberFormatException

DO

CALL JOptionPane which showMessageDialog(frame, "Phone number must contain number only.", "Warning", WARNING_MESSAGE)

RETURN

END DO

IF phoneStr which length is not equals 10

DO

CALL JOptionPane which showMessageDialog(frame, "Phone number must be of 10 digits.", "Warning", WARNING_MESSAGE)

RETURN

END DO

TRY

DO

INITIALIZE id to Integer which parseInt(rIdField which getText and trim)

IF isDuplicateId(id)

DO

CALL JOptionPane which showMessageDialog(frame, "Member with this ID already exists!")

RETURN

END DO

INITIALIZE name to rNameField which getText and trim

INITIALIZE location to rLocationField which getText and trim

INITIALIZE phone to phoneStr

INITIALIZE email to rEmailField which getText and trim

INITIALIZE gender to (IF rMaleBtn isSelected **THEN** "Male" **ELSE** "Female")

INITIALIZE **DO**b to r**DO**bYear which get SelectedItem + "/" + r**DO**bMonth which get SelectedItem + "/" + r**DO**bDay which get SelectedItem

INITIALIZE msd to rMsYear which get SelectedItem + "/" + rMsMonth which get SelectedItem + "/" + rMsDay which get SelectedItem

INITIALIZE referral to rReferralField which get Text and trim

DELCARE and INITIALIZE rMemObj as RegularMember to new RegularMember(id, name, location, phone, email, gender, **DO**b, msd, referral)

CALL rMemObj which **SET** Plan to "Basic"

ADD rMemObj to membersObj

CALL JOptionPane which showMessageDialog(frame, "Regular Member **ADD**ed successfully!")

END DO

CATCH NumberFormatException ex

DO

CALL JOptionPane which showMessageDialog(frame, "Character value detected. Please input number only!")

END DO

END DO

IF btn equals rDisplayBtn

DO

IF membersObj is Empty

DO

CALL JOptionPane which showMessageDialog(frame, "No members available!")

RETURN

END DO

DECLARE and **INITIALIZE** rDisplayInfoFrame as JFrame to new JFrame("Regular Member Display")

SET rDisplayInfoFrame size to (320, 730)

SET rDisplayInfoFrame Layout to null

DECLARE and **INITIALIZE** textArea as JTextArea to new JTextArea()

SET textArea editable to false

SET textArea font to ("Arial", PLAIN, 14)

DECLARE and **INITIALIZE** displayInfo as String to ""

FOR each gm in membersObj

DO

IF gm is instanceof RegularMember

DO

INITIALIZE rm as RegularMember to (RegularMember) gm

APPEND displayInfo WITH "ID: " + rm which get Id concat "\n"

APPEND displayInfo WITH "Name: " + rm which getName + "\n"

APPEND displayInfo WITH "Location: " + rm which get Location() + "\n"

APPEND displayInfo WITH "Phone: " + rm which get Phone() + "\n"
APPEND displayInfo WITH "Email: " + rm which get Email() + "\n"
APPEND displayInfo WITH "Gender: " + rm which get Gender() + "\n"
APPEND displayInfo WITH "**DO**B: " + rm which get **DO**B() + "\n"
APPEND displayInfo WITH "Membership Start Date: " + rm which get MembershipStartDate() + "\n"
APPEND displayInfo WITH "Attendance: " + rm which get Attendance() + "\n"
APPEND displayInfo WITH "Loyalty Points: " + rm which get LoyaltyPoints() + "\n"
APPEND displayInfo WITH "Active Status: " + rm which get ActiveStatus() + "\n"
APPEND displayInfo WITH "Plan: " + rm which get Plan() + "\n"
APPEND display WITH "Price: " + rm which get Price() + "\n"
APPEND display WITH "Referral Source: " + rm which get ReferralSource() + "\n"
IF not rm which get RemovalReason and is Empty
DO
APPEND display WITH "Removal Reason: " + rm.getRemovalReason() + "\n"
END DO
END IF
END DO
END FOR
CALL textArea to **SET** Text(display)
DECLARE and **INITIALIZE** scrollPane as JScrollPane to new JScrollPane(textArea)
SET scrollPane to bounds(10, 10, 300, 690)

SET rDisplayInfoFrame resizable to false

CALL rDisplayInfoFrame to **ADD**(scrollPane)

SET rDisplayInfoFrame visibility to true

END DO

ELSE IF btn equals rClearBtn

DO

SET rIdField Text to ""

SET rNameField Text to ""

SET rLocationField Text to ""

SET rPhoneField Text to ""

SET rEmailField Text to ""

SET rReferralField Text to ""

SET rMaleBtn Selected to false

SET rFemaleBtn Selected to false

SET r**DO**bYear SelectedIndex to 0

SET r**DO**bMonth SelectedIndex to 0

SET r**DO**bDay SelectedIndex to 0

SET rMsYear SelectedIndex to 0

SET rMsMonth SelectedIndex to 0

SET rMsDay SelectedIndex to 0

END DO

ELSE IF btn equals p**ADD**MemberBtn

DO

IF pIdField which get text and is empty OR pNameField which get text and is empty OR pLocationField which get text and is empty OR pPhoneField which get text and is empty OR pEmailField which get text and is empty OR pTrainerField which get text and is empty OR (NOT rMaleBtn isSelected AND NOT rFemaleBtn isSelected)

DO

CALL JOptionPane which showMessageDialog(frame, "Please fill in all required fields for Premium Member.")

RETURN

END DO

DECLARE phoneStr as String to pPhoneField which get text and trim

TRY

DO

CALL Long which parseLong(phoneStr)

END DO

CATCH NumberFormatException e

DO

CALL JOptionPane which showMessageDialog(frame, "Phone number must contain number only.", "Warning", WARNING_MESSAGE)

RETURN

END DO

END TRY

IF phoneStr whose length not equals 10

DO

CALL JOptionPane which showMessageDialog(frame, "Phone number must be of 10 digits.", "Warning", WARNING_MESSAGE)

RETURN

END DO

TRY

DO

INITIALIZE id to Integer which parseInt(pIdField which getText and trim)

IF isDuplicateId(id)

DO

CALL JOptionPane which showMessageDialog(frame, "Member with this ID already exists!")

RETURN

END DO

INITIALIZE name to pNameField which getText and trim

INITIALIZE location to pLocationField which getText and trim

INITIALIZE phone to phoneStr

INITIALIZE email to pEmailField which getText and trim

INITIALIZE gender to (IF pMaleBtn isSelected **THEN** "Male" **ELSE** "Female")

INITIALIZE **DO**b to p**DO**bYear which get SelectedItem + "/" + p**DO**bMonth which get SelectedItem + "/" + p**DO**bDay which get SelectedItem

INITIALIZE msd to pMsYear which get SelectedItem + "/" + pMsMonth which get SelectedItem + "/" + pMsDay which get SelectedItem

INITIALIZE trainer to pTrainerField which get Text and trim

DELCARE and **INITIALIZE** pMemObj as PremiumMember to new PremiumMember(id, name, location, phone, email, gender, **DO**b, msd, trainer)

ADD pMemObj to membersObj

CALL JOptionPane which showMessageDialog(frame, "Premium Member **ADD**ed successfully!")

END DO

CATCH NumberFormatException ex

DO

CALL JOptionPane which showMessageDialog(frame, "Character value detected. Please input number only!")

END DO

END DO

ELSE IF btn equals pDisplayBtn

DO

IF membersObj is Empty

DO

CALL JOptionPane which showMessageDialog(frame, "No members **ADD**ed yet to display!")

RETURN

END DO

DECLARE and **INITIALIZE** pDisplayInfoFrame as JFrame to new JFrame("Premium Member Display")

SET pDisplayInfoFrame size to (320, 730)

SET pDisplayInfoFrame Layout to null

DECLARE and **INITIALIZE** textArea as JTextArea to new JTextArea()

SET textArea editable to false

SET textArea font to ("Arial", PLAIN, 14)

DECLARE and **INITIALIZE** displayInfo as String to ""

FOR each gm in membersObj

DO

IF gm is instanceof PremiumMember

DO

INITIALIZE pm as PremiumMember to (PremiumMember) gm

DECLARE and **SET** remaining as **DO**uble to difference of (pm which get PremiumCharge and pm which get PaidAmount)

DECLARE and **SET full as Boolean to pm which isFullPayment**

DECLARE and **SET discount as DOuble to pm which get DiscountAmount**

DECLARE and **SET paidAmt as DOuble to** (IF full **THEN** difference of (pm which get PremiumCharge and discount) **ELSE** pm which get PaidAmount))

APPEND displayInfo WITH "ID: " + pm which get Id concat "\n"
APPEND displayInfo WITH "Name: " + pm which getName + "\n"
APPEND displayInfo WITH "Location: " + pm which get Location() + "\n"
APPEND displayInfo WITH "Phone: " + pm which get Phone() + "\n"
APPEND displayInfo WITH "Email: " + pm which get Email() + "\n"
APPEND displayInfo WITH "Gender: " + pm which get Gender() + "\n"
APPEND displayInfo WITH "**DO**B: " + pm which get **DO**B() + "\n"
APPEND displayInfo WITH "Membership Start Date: " + pm which get MembershipStartDate() + "\n"
APPEND displayInfo WITH "Attendance: " + pm which get Attendance() + "\n"
APPEND displayInfo WITH "Loyalty Points: " + pm which get LoyaltyPoints() + "\n"
APPEND displayInfo WITH "Active Status: " + pm which get ActiveStatus() + "\n"
APPEND displayInfo WITH "Premium Charge: " + pm which get PremiumCharge() + "\n"
APPEND displayInfo WITH "Paid Amount: " + pm which get paidAmt() + "\n"
APPEND displayInfo WITH "Remaining Amount: " + (IF full then 0 else remaining)
IF not pm which get RemovalReason and is Empty
DO
APPEND display WITH "Discount Amount: " + discount + "\n"
APPEND display with " ---------- \n"
END DO

END IF

END DO

END FOR

CALL textArea to **SET** Text(display)

DECLARE and **INITIALIZE** scrollPane as JScrollPane to new JScrollPane(textArea)

SET scrollPane to bounds(10, 10, 300, 690)

SET rDisplayInfoFrame resizable to false

CALL rDisplayInfoFrame to **ADD** (scrollPane)

SET rDisplayInfoFrame visibility to true

END DO

ELSE IF btn equals pClearBtn

DO

SET pIdField Text to ""

SET pNameField Text to ""

SET pLocationField Text to ""

SET pPhoneField Text to ""

SET pEmailField Text to ""

SET pTrainerField Text to ""

SET pMaleBtn Selected to false

SET pFemaleBtn Selected to false

SET pdobbYear SelectedIndex to 0

SET pdobMonth SelectedIndex to 0

SET pdobDay SelectedIndex to 0

SET pMsYear SelectedIndex to 0

SET pMsMonth SelectedIndex to 0

SET pMsDay SelectedIndex to 0

SET pChargeField Text to 50000

END DO

```
IF btn equals activateMemberBtn
DO
    TRY
    DO
        DECLARE id as int = Integer which parseInt(actIdField which get Text and trim)
        DECLARE gm as GymMember to searchMemId(id)
        IF gm NOT EQUALS null
        DO
            CALL gm which activeMembership
            CALL JOptionPane which showMessageDialog(frame, "Membership activated!", "Status message", INFORMATION_MESSAGE)
        END DO
        ELSE
        DO
            CALL JOptionPane which show MessageDialog(frame, "Member not found!", "Status message", ERROR_MESSAGE)
        END DO
    END DO
    CATCH NumberFormatException    ex
    DO
        CALL JOptionPane which showMessageDialog(frame, "Member Id can be number only!", "Status message", WARNING_MESSAGE)
    END DO
    END TRY
END DO
END IF
```

IF btn equals deactivateMemberBtn
DO
TRY
DO
DECLARE id as int to Integer which parseInt(actIdField which get Text and trim)
DECLARE gm as GymMember to searchMemId(id)
IF gm NOT equals null
DO
IF gm which get ActiveStatus
DO
CALL gm which deactivateMemberhship
CALL JOptionPane which showMessageDialog(frame, "Membership deactivated!", "Status message", INFORMATION_MESSAGE)
END DO
ELSE
DO
CALL JOptionPane which showMessageDialog(frame, "Membership is not active!", "Status message", WARNING_MESSAGE)
END DO
END DO
END IF
ELSE
DO
CALL JOptionPane which showMessageDialog(frame, "Member not found!", "Status message", ERROR_MESSAGE)
END DO
END IF

```
        END DO
        CATCH NumberFormatException ex
        DO
            CALL JOptionPane which showMessageDialog(frame, "Member Id can
        be number only!", "Status message", WARNING_MESSAGE)
        END DO
        END TRY
END DO

IF btn equals markRegularBtn
DO
        TRY
        DO
            DECLARE id as int to Integer which parseInt(markIdField which get Text
            and trim)
            DECLARE gm as GymMember to searchMemId(id)
            IF gm NOT equals null AND gm is instanceof RegularMember
            DO
                IF gm which get ActiveStatus
                DO
                    CALL gm which markAttendance
                    CALL JOptionPane which showMessageDialog(frame,
                    "Attendance marked for Regular Member!", "Attendance
                    message", INFORMATION_MESSAGE)
                END DO
                ELSE
                DO
                    CALL JOptionPane which showMessageDialog(frame,
                    "Member is not active!", "Attendance message",
                    WARNING_MESSAGE)
```

```
                    END DO
                END DO
                END IF
                ELSE
                DO
                    CALL JOptionPane which showMessageDialog(frame, "Regular
                Member not found!", "Attendance message", ERROR_MESSAGE)
                END DO
                END IF
            END DO
            CATCH NumberFormatException ex
            DO
                CALL JOptionPane which showMessageDialog(frame, "Invalid ID.",
                "Attendance message", WARNING_MESSAGE)
            END DO
            END TRY
        END DO

        IF btn equals markPremiumBtn
        DO
            TRY
            DO
                DECLARE id as int to Integer which parseInt(markIdField which get Text
                and trim)
                DECLARE gm as GymMember to searchMemId(id)
                IF gm NOT equals null AND gm is instanceof PremiumMember
                DO
                    IF gm which get ActiveStatus
                    DO
                        CALL gm which markAttendance
```

CALL JOptionPane which showMessageDialog(frame, "Attendance marked for Premium Member!", "Attendance message", INFORMATION_MESSAGE)

END DO

ELSE

DO

CALL JOptionPane which showMessageDialog(frame, "Member is not active!", "Attendance message", WARNING_MESSAGE)

END DO

END IF

END DO

ELSE

DO

CALL JOptionPane which showMessageDialog(frame, "Premium Member not found!", "Attendance message", ERROR_MESSAGE)

END DO

END IF

END DO

CATCH NumberFormatException ex

DO

CALL JOptionPane which showMessageDialog(frame, "Invalid ID.", "Attendance message", WARNING_MESSAGE)

END DO

END TRY

END DO

IF btn equals upgradePlanBtn

DO

TRY

DO

DECLARE id as int to Integer which parseInt(upIdField which get Text and trim)

DECLARE gm as GymMember to searchMemId(id)

IF gm NOT equals null AND gm is instanceof RegularMember

DO

DECLARE rm as RegularMember to (RegularMember) gm

DECLARE selectedPlan as String to upPlanBox which get SelectedItem

IF rm which get Plan equalsIgnoreCase selectedPlan

DO

CALL JOptionPane which showMessageDialog(frame, "You are already subscribed to the " + selectedPlan + " plan.")

RETURN

END DO

IF rm which get Attendance less than 30

DO

CALL JOptionPane which showMessageDialog(frame, "Not enough attendance for upgrade! (Minimum 30 required)")

RETURN

END DO

DECLARE result as String to rm which upgradePlan(selectedPlan)

CALL JOptionPane which showMessageDialog(frame, result)

END DO

ELSE

DO

CALL JOptionPane which showMessageDialog(frame, "Regular Member not found!")

END DO

END IF

END DO

CATCH NumberFormatException ex

DO

CALL JOptionPane which showMessageDialog(frame, "Invalid ID.")

END DO

END TRY

END DO

ELSE IF btn equals calcDiscountBtn

DO

TRY

DO

DECLARE id as int to Integer which parseInt(discIdField which get Text and trim)

DECLARE gm as GymMember to searchMemId(id)

IF gm NOT equals null AND gm is instanceof PremiumMember

DO

DECLARE pm as PremiumMember to (PremiumMember) gm

CALL pm which calculateDiscount

CALL discAmountField which **SET**Text(String which valueOf(pm which get DiscountAmount))

END DO

ELSE

DO

CALL JOptionPane which showMessageDialog(frame, "Premium Member not found!")

END DO

END IF

END DO

CATCH NumberFormatException ex

DO

CALL JOptionPane which showMessageDialog(frame, "Invalid ID.")

END DO

END TRY

END DO

ELSE IF btn equals revertRegularBtn

DO

TRY

DO

DECLARE id as int to Integer which parseInt(revertIdField which get Text and trim)

DECLARE gm as GymMember to searchMemId(id)

IF gm NOT equals null AND gm is instanceof RegularMember

DO

DECLARE reason as String to JOptionPane which showInputDialog(frame, "Enter removal reason:")

IF reason equals "" OR reason equals null

DO

CALL JOptionPane which showMessageDialog(frame, "Removal reason is required.")

RETURN

END DO

DECLARE rm as RegularMember to (RegularMember) gm

CALL rm which revertRegularMember(reason)

CALL JOptionPane which showMessageDialog(frame, "Regular Member reverted!")

END DO

ELSE

DO

CALL JOptionPane which showMessageDialog(frame, "Regular Member not found!")

END DO

END IF

END DO

CATCH NumberFormatException ex

DO

CALL JOptionPane which showMessageDialog(frame, "Invalid ID.")

END DO

END TRY

END DO

ELSE IF btn equals revertPremiumBtn

DO

TRY

DO

DECLARE id as int to Integer which parseInt(revertIdField which get Text and trim)

DECLARE gm as GymMember to searchMemId(id)

IF gm NOT equals null AND gm is instanceof PremiumMember

DO

DECLARE pm as PremiumMember to (PremiumMember) gm

CALL pm which revertPremiumMember

CALL JOptionPane which showMessageDialog(frame, "Premium Member reverted!")

END DO

ELSE

DO

CALL JOptionPane which showMessageDialog(frame, "Premium Member not found!")

END DO

END IF

END DO

CATCH NumberFormatException ex

DO

CALL JOptionPane which showMessageDialog(frame, "Invalid ID.")

END DO

END TRY

END DO

ELSE IF btn equals payDueButton

DO

TRY

DO

DECLARE id as int to Integer which parseInt(payIdField which get Text and trim)

DECLARE amt as **DO**uble to **DO**uble which parse**DO**uble(payAmountField which get Text and trim)

DECLARE gm as GymMember to searchMemId(id)

IF gm NOT equals null

DO

IF gm is instanceof PremiumMember

DO

DECLARE pm as PremiumMember to (PremiumMember) gm

DECLARE result as String to pm which payDueAmount(amt)

CALL JOptionPane which showMessageDialog(frame, result)

ELSE

DO

CALL JOptionPane which showMessageDialog(frame, "Pay Due Amount is applicable for Premium Members only.")

END DO

END IF

END DO

ELSE

DO

CALL JOptionPane which showMessageDialog(frame, "Member not found!")

END DO

END IF

END DO

CATCH NumberFormatException ex

DO

CALL JOptionPane which showMessageDialog(frame, "Invalid ID or amount.")

END DO

END TRY

END DO

ELSE IF btn equals saveBtn

DO

```
IF membersObj is Empty
DO
    CALL JOptionPane which showMessageDialog(frame, "No members to save!", "Save to File", ERROR_MESSAGE)
    RETURN
END DO
TRY
DO
    DECLARE writer as FileWriter to new FileWriter("MemberDetails.txt")
    CALL writer which write(String.format(
        "%-5s %-15s %-15s %-15s %-25s %-20s %-10s %-10s %-10s %-15s %-12s %-20s %-12s %-15s %-15s\n",
        "ID", "Name", "Location", "Phone", "Email",
        "Membership Start", "Plan", "Price", "Attendance",
        "Loyalty Points", "ActiveStatus", "Personal Trainer", "FullPayment", "DiscountAmt", "PaidAmt"
    ))
    FOR each gm in membersObj
    DO
        DECLARE plan as String
        DECLARE price as DOuble
        DECLARE trainer as String
        DECLARE fullPay as boolean
        DECLARE discount as DOuble
        DECLARE totalPaid as DOuble
        IF gm is instanceof RegularMember
        DO
            DECLARE rm as RegularMember to (RegularMember) gm
            SET plan to rm which get Plan
```

SET price to rm which get Price

SET trainer to "N/A"

SET fullPay to true

SET discount to 0.0

SET totalPaid to price

END DO

ELSE

DO

DECLARE pm as PremiumMember to (PremiumMember) gm

SET plan to "Premium"

SET price to pm which get PremiumCharge

SET trainer to pm which get PersonalTrainer

SET fullPay to pm which isFullPayment

SET discount to pm which get DiscountAmount

SET totalPaid to IF fullPay THEN (price – discount) ELSE pm which get PaidAmount

END DO

END IF

CALL writer which write(String.format(

"%-5d %-15s %-15s %-15s %-25s %-20s %-10s %-10.2f %-10d %-15.2f %-12b %-20s %-12b %-15.2f %-15.2f\n",

gm which getId,

gm which getName,

gm which getLocation,

gm which getPhone,

gm which getEmail,

gm which getMembershipStartDate,

plan, price, gm which getAttendance, gm which getLoyaltyPoints, gm which getActiveStatus, trainer, fullPay, discount, totalPaid

```
                    ))
                END FOR
                CALL writer which close
                CALL JOptionPane which showMessageDialog(frame, "Member details
saved to file!", "Save to File", INFORMATION_MESSAGE)
            END DO
            CATCH IOException ex
            DO
                CALL JOptionPane which showMessageDialog(frame, "Error saving to
file!", "Save to File", ERROR_MESSAGE)
            END DO
            END TRY
        END DO

        ELSE IF btn equals readBtn
        DO
            TRY
            DO
                DECLARE reader as FileReader to new
FileReader("MemberDetails.txt")
                DECLARE content as String to ""
                WHILE reader which read NOT equals -1
                DO
                    SET content to content + (char) reader which read
                END WHILE
                CALL reader which close
                DECLARE readDisplayFrame as JFrame to new JFrame("Member
Details from File")
                CALL readDisplayFrame which SETSize(1300, 610)
                CALL readDisplayFrame which SETLayout(null)
```

SET readDisplayFrame Resizable to false
DECLARE textArea as JTextArea to new JTextArea()

CALL textArea which **SET**Editable(false)
CALL textArea which **SET**Font(new Font("Monospaced", PLAIN, 12))
CALL textArea which **SET**Text(content)
DECLARE scrollPane as JScrollPane to new JScrollPane(textArea)
CALL scrollPane which **SET**Bounds(10, 10, 1270, 550)
CALL re**ADD**isplayFrame which add(scrollPane)
CALL re**ADD**isplayFrame which **SET**Visible(true)
END DO
CATCH IOException ex
DO
CALL JOptionPane which showMessageDialog(frame, "Error reading file!", "Read from File", ERROR_MESSAGE)
END DO
END TRY
END DO

END DO

CREATE method main(String[] args) WITH return type void AND access modifier public static
DO
CALL new GymGUI()
END DO
END DO
END CLASS

Method Description

Method Description of GymMember

<<constructor>>GymMember()

- Return Type: N/A (constructor)
- Parameters: id (int), name (String), location (String), phone (String), email (String), gender (String), DOB (String), membershipStartDate (String)
- Description: This is the constructor of the GymMember class. It takes up member details like ID, name, location, etc. and uses them to set up a new gym member. It also sets default values likes attendance to 0, loyalty points to 0.0, and active status to false.

getId()

- Return Type: int
- Parameters: N/A
- Description: This method returns the member's ID as a number.

getName()

- Return Type: String
- Parameters: N/A
- Description: This method gives back the member's name as string.

getLocation()

- Return Type: String
- Parameters: N/A
- Description: This method returns the member's location as a string.

getPhone()

- Return Type: String
- Parameters: N/A
- Description: This method provides the member's phone number as string.

getEmail()

- Return Type: String
- Parameters: N/A

- Description: This method returns the member's email address as a string.

getGender()

- Return Type: String
- Parameters: N/A
- Description: This method gives the member's gender as text.

getDOB()

- Return Type: String
- Parameters: N/A
- Description: This method returns the member's date of birth as a string.

getMembershipStartDate()

- Return Type: String
- Parameters: N/A
- Description: This method provides the member's membership start date as string.

getAttendance()

- Return Type: int
- Parameters: N/A
- Description: This method returns the member's attendance count as a number. It’s a getter that shows how many times the member went to gym stored in attendance.

getLoyaltyPoints()

- Return Type: double
- Parameters: N/A
- Description: This method gives the member's loyalty points as a decimal number.

getActiveStatus()

- Return Type: boolean
- Parameters: N/A
- Description: This method returns whether the membership is active (true) or not (false).

<<abstract>>markAttendance()

- Return Type: void
- Parameters: N/A
- Description: This is an abstract method with no body in GymMember. It’s meant to be defined

in subclasses like RegularMember or PremiumMember to record attendance.

activeMembership()

- Return Type: void
- Parameters: N/A
- Description: This method turns on the member's membership by setting activeStatus to true. It's used when a member starts or renews their membership.

deactivateMemberhship()

- Return Type: void
- Parameters: N/A
- Description: This method turns off the member's membership by setting activeStatus to false, but only if it's already active. If it's not active, it prints a message saying the membership isn't active.

resetMember()

- Return Type: void
- Parameters: N/A
- Description: This method resets the member's details to default: activeStatus to false, attendance to 0, and loyaltyPoints to 0.0.

display()

- Return Type: void
- Parameters: N/A
- Description: This method prints all the member's details, like ID, name, phone, and attendance, to the terminal. It's a best way to see everything about a member at once.

Method Description of RegularMember

<<Constructor>> RegularMember(int id, String name, String location, String phone, String email, String gender, String DOB, String membershipStartDate, String referralSource)

- Return Type: N/A (constructor)
- Parameters: id (int), name (String), location (String), phone (String), email (String), gender (String), DOB (String), membershipStartDate (String), referralSource (String)
- Description: This constructor sets up a new RegularMember by passing basic details to the GymMember constructor and adding regular member specifics. It sets referralSource, starts the plan as "basic" with a price of 6500, and initializes isEligibleForUpgrade to false and removalReason to empty.

getAttendanceLimit()

- Return Type: int
- Parameters: N/A
- Description: This method returns the attendance limit, which is fixed at 30. It's a getter for the attendanceLimit constant that helps decide upgrade eligibility.

getIsEligibleForUpgrade()

- Return Type: boolean
- Parameters: N/A
- Description: This method tells if the member can upgrade their plan (true or false). It returns the isEligibleForUpgrade value based on attendance.

getRemovalReason()

- Return Type: String
- Parameters: N/A
- Description: This method returns the reason why a member was removed, if any, as text. It gives the removalReason value, which is empty if there's no reason.

getReferralSource()

- Return Type: String
- Parameters: N/A
- Description: This method provides the source of the member's referral as a string. It returns

the referralSource attribute set during construction.

getPlan()

- Return Type: String
- Parameters: N/A
- Description: This method returns the member’s current plan "basic" as text. It allows the plan attribute to show the subscription type.

getPrice()

- Return Type: double
- Parameters: N/A
- Description: This method gives the price of the member’s current plan as a decimal number. It returns the price attribute according to respective plan.

markAttendance()

- Return Type: void
- Parameters: N/A
- Description: This method adds 1 to the member’s attendance and 5 to their loyalty points each time it’s called. If attendance hits 30 or more, it sets isEligibleForUpgrade to true.

getPlanPrice(String plan)

- Return Type: double
- Parameters: plan (String)
- Description: This method returns the price for a given plan (like "basic" for 6500) as a number. It uses a switch case to match the plan name and returns -1 if the plan is invalid.

setPlan(String plan)

- Return Type: void
- Parameters: plan (String)
- Description: This method updates the member’s plan to the one provided. It simply changes the plan attribute to the new value.

upgradePlan(String plan)

- Return Type: String
- Parameters: plan (String)
- Description: This method tries to upgrade the member’s plan. It checks if the plan is the

same, if the member is eligible (attendance ≥ 30), and then updates plan and price if valid, returning a success or error message.

revertRegularMember(String removalReason)

- Return Type: void
- Parameters: removalReason (String)
- Description: This method resets the member by calling resetMember(), sets the plan back to "basic" with a price of 6500, marks them ineligible for upgrades, and stores the removal reason.

display()

- Return Type: void
- Parameters: N/A
- Description: This method prints all member details from GymMember plus RegularMember specifics like plan, price, and removal reason. It's an expanded version of the parent's display.

Method Description for PremiumMember

<<Constructor>> PremiumMember(int id, String name, String location, String phone, String email, String gender, String DOB, String membershipStartDate, String personalTrainer)

- Return Type: N/A (constructor)
- Parameters: id (int), name (String), location (String), phone (String), email (String), gender (String), DOB (String), membershipStartDate (String), personalTrainer (String)
- Description: This constructor creates a PremiumMember by passing basic info to GymMember and adding premium details. It sets personalTrainer, starts isFullPayment as false, and sets paidAmount and discountAmount to 0.0.

getPremiumCharge()

- Return Type: double
- Parameters: N/A
- Description: This method returns the premium membership cost, fixed at 50000, as a number.

getPersonalTrainer()

- Return Type: String
- Parameters: N/A
- Description: This method gives the name of the member's personal trainer as text. It returns the personalTrainer attribute's value.

isFullPayment()

- Return Type: boolean
- Parameters: N/A
- Description: This method returns if the member has paid the full premium charge payment or not (true or false).

getPaidAmount()

- Return Type: double
- Parameters: N/A
- Description: This method returns how much the member has paid so far as a number.

getDiscountAmount()

- Return Type: double
- Parameters: N/A
- Description: This method gives the discount amount, if given, as a number. It returns the discountAmount set by calculateDiscount().

markAttendance()

- Return Type: void
- Parameters: N/A
- Description: Each time this method is called, 1 is added to attendance and 10 to loyalty points, which is 5 more points than regular.

payDueAmount(double paidAmount)

- Return Type: String
- Parameters: paidAmount (double)
- Description: This method allows the member to make payments towards premium charges. It checks whether the paid amount is complete or in excess, updates paidAmount, and returns a message about the payment status.

calculateDiscount()

- Return Type: void
- Parameters: N/A
- Description: This method sets a 10% discount on the premium charge if it is fully paid; the discount is then stored in discountAmount. It then prints the discounts or nothing if it is not due.

revertPremiumMember()

- Return Type: void
- Parameters: N/A
- Description: This method resets the member using resetMember(), clears the trainer, sets payment to false and paid amount and discount amount back to zero. It's a full reset process for premium members.

display()

- Return Type: void
- Parameters: N/A
- Description: This method prints all GymMember details plus premium member details like trainer, paid amount, and discount (if full payment is made). It shows a complete detail of the member.

Method Description of GymGUI

<<Constructor>> GymGUI()

- Return Type: N/A (constructor)
- Parameters: N/A
- Description: This is the constructor of the GymGUI class. It sets up the main core program of the gym management system, creates an ArrayList to hold gym members, and builds all the frames, panels, labels, text area, buttons, and other components, needed for the interface. It starts with the login screen appearing when the system runs.

buttonPointer(JButton button)

- Return Type: void
- Parameters: button (JButton)
- Description: This method focuses on presenting a button cursor to a pointed hand when hovering over the button. It also prevents the rectangle highlighting upon clicking so that the neatness of the interface is maintained. It is used for all buttons to prevent a rectangle outline from appearing when clicking down on a button.

hideAllSubPanels()

- Return Type: void
- Parameters: N/A
- Description: This method hides all sub-panels (home, add member, attendance, etc.) in the main panel, so that only one panel can be seen at a time based on user selection. It's called whenever the user switches between different sections (panels).

findMemberById(int id)

- Return Type: GymMember
- Parameters: id (int)
- Description: This method searches through the ArrayList of the gym members to find one with a matching ID. It returns the member if found or null if no match is found. It is used when looking up a member by their unique id in the system.

isDuplicateId(int id)

- Return Type: boolean
- Parameters: id (int)
- Description: This method checks if a member with the given ID already exists in the ArrayList. It uses findMemberById and returns true if the ID is taken, false if not taken. It helps avoid adding two members with the same ID.

actionPerformed(ActionEvent eve)

- Return Type: void
- Parameters: eve (ActionEvent)
- Description: This method is responsible for button clicks and other actions in the system, and decides what to perform. Actions include login, panel switching, member addition, attendance marking, activation, deactivation, and more. It's basically the main way the system responds to anything that a user does.

Testing

Testing 1

Table 1: Testing 1: Compile and the program using command prompt command prompt

Objective	To compile and run the program using the command prompt.
Action	i. Opened the java file located folder. ii. Wrote cmd in the file location info. iii. Wrote “javac GymGUI.java” to compile the program. iv. Wrote “java GymGUI” to run the program.
Expected Output	The program should be compiled and run without any error.
Actual Output	The program was compiled and run successfully.
Result	The test was successful.

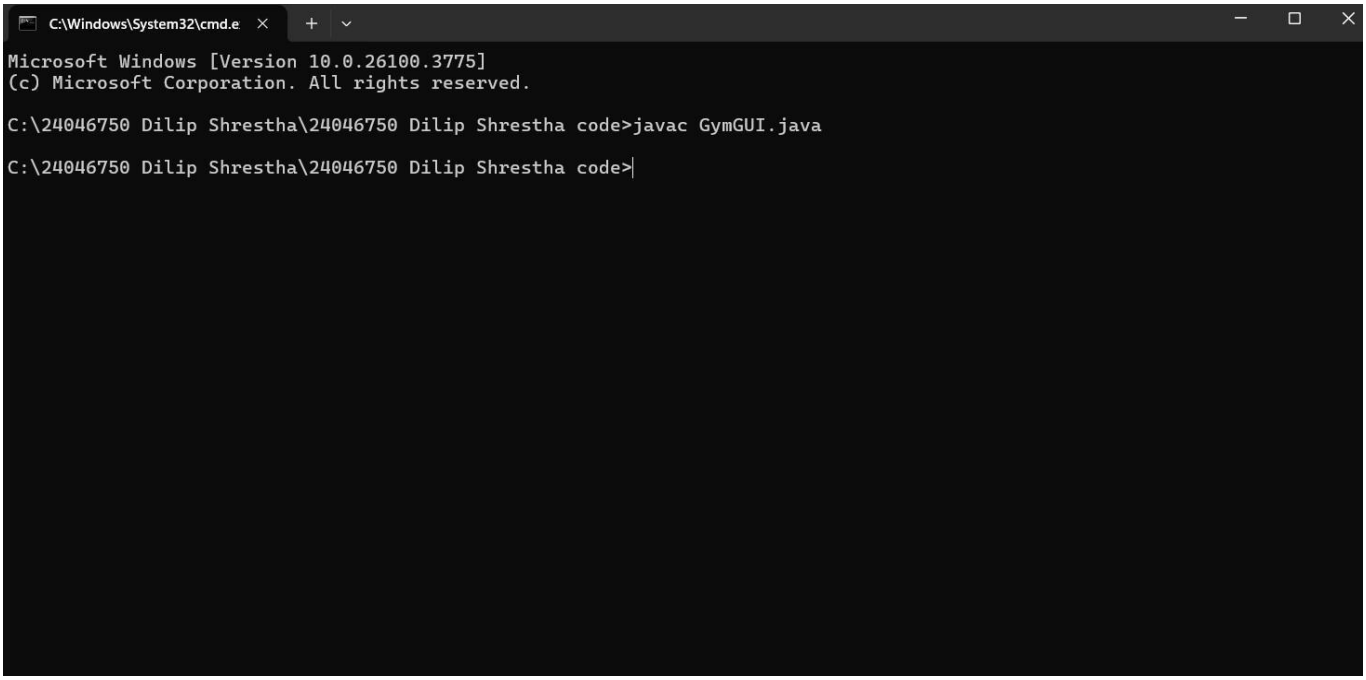


Figure 31: Prompt to Compile Java Program

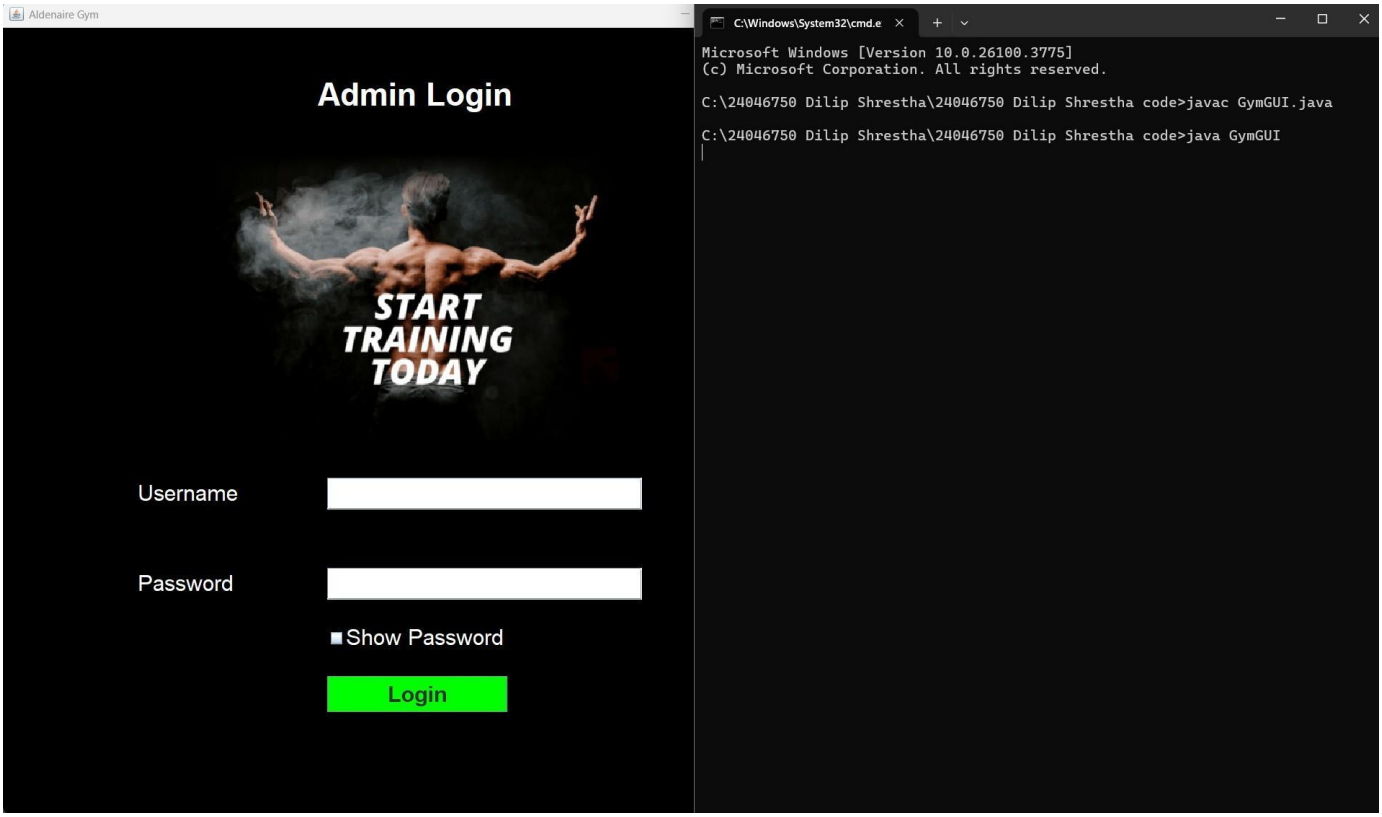


Figure 32: Prompt to Run Java Program

Testing 2

Table 2: Testing 2: Add regular and premium member

Objective	To add a regular member and premium member.
Action	i. All the required text fields for regular member were filled and "Add member" button was clicked. ii. All the required text fields for premium member were filled and "Add member" button was clicked.
Expected Output	The details of regular member and premium member should be added successfully with a message in an information dialog box.
Actual Output	The details of regular member and premium member were added successfully with a message in an information dialog box.
Result	The test was successful.

Add Regular Member

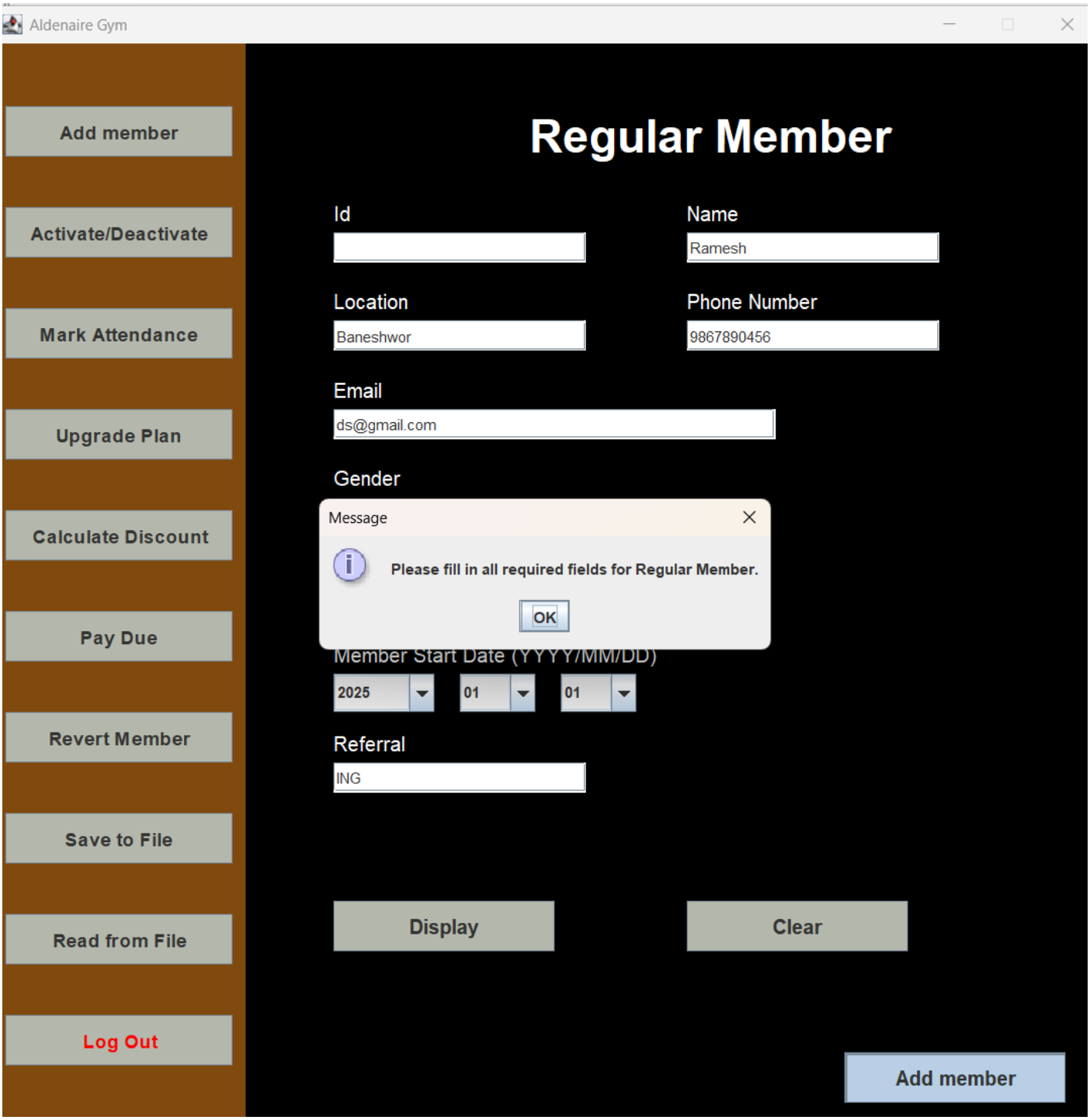


Figure 33: Keeping 1 field empty in Regular Member

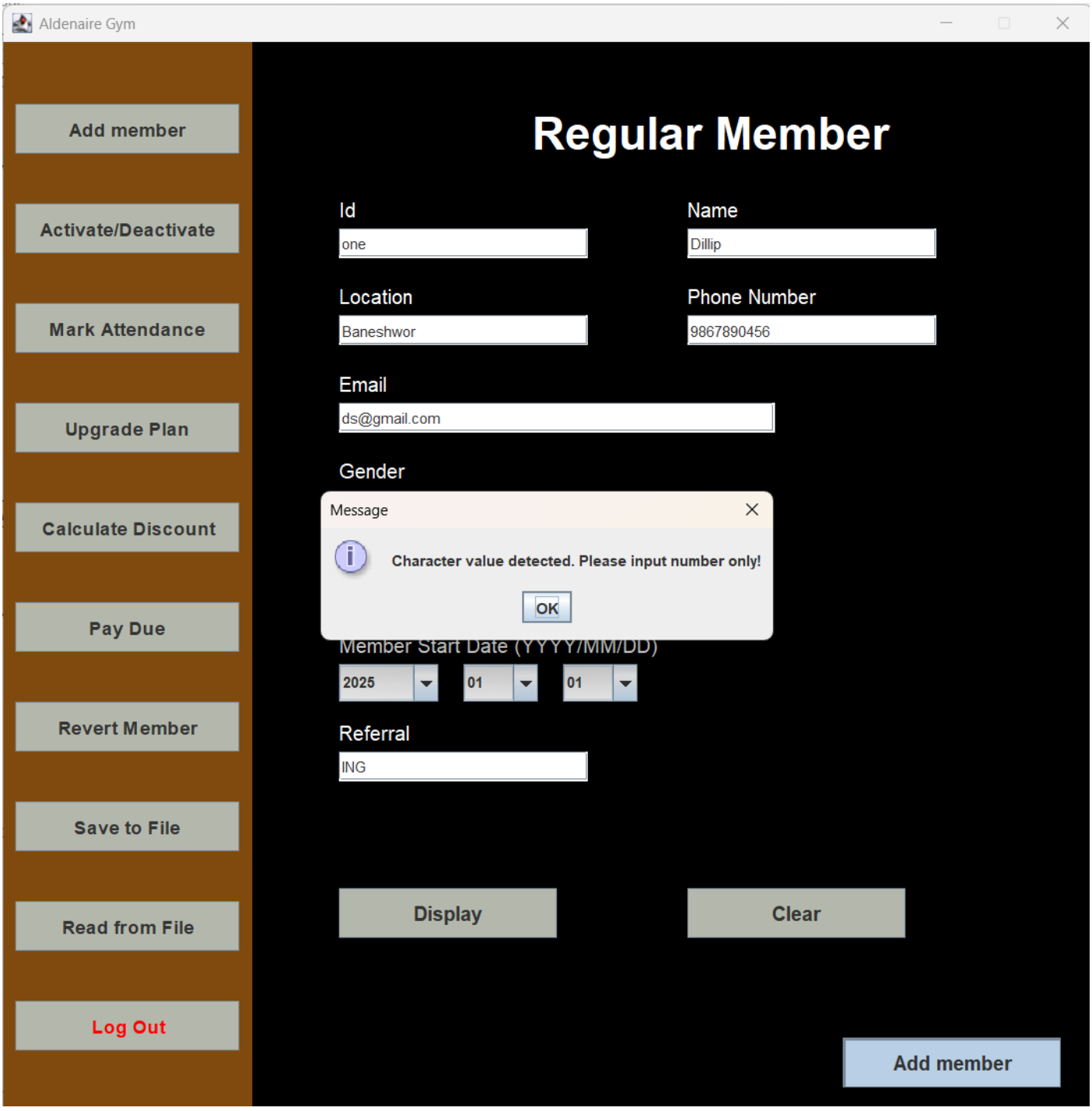


Figure 34: Passing character in id (Exception Handling)

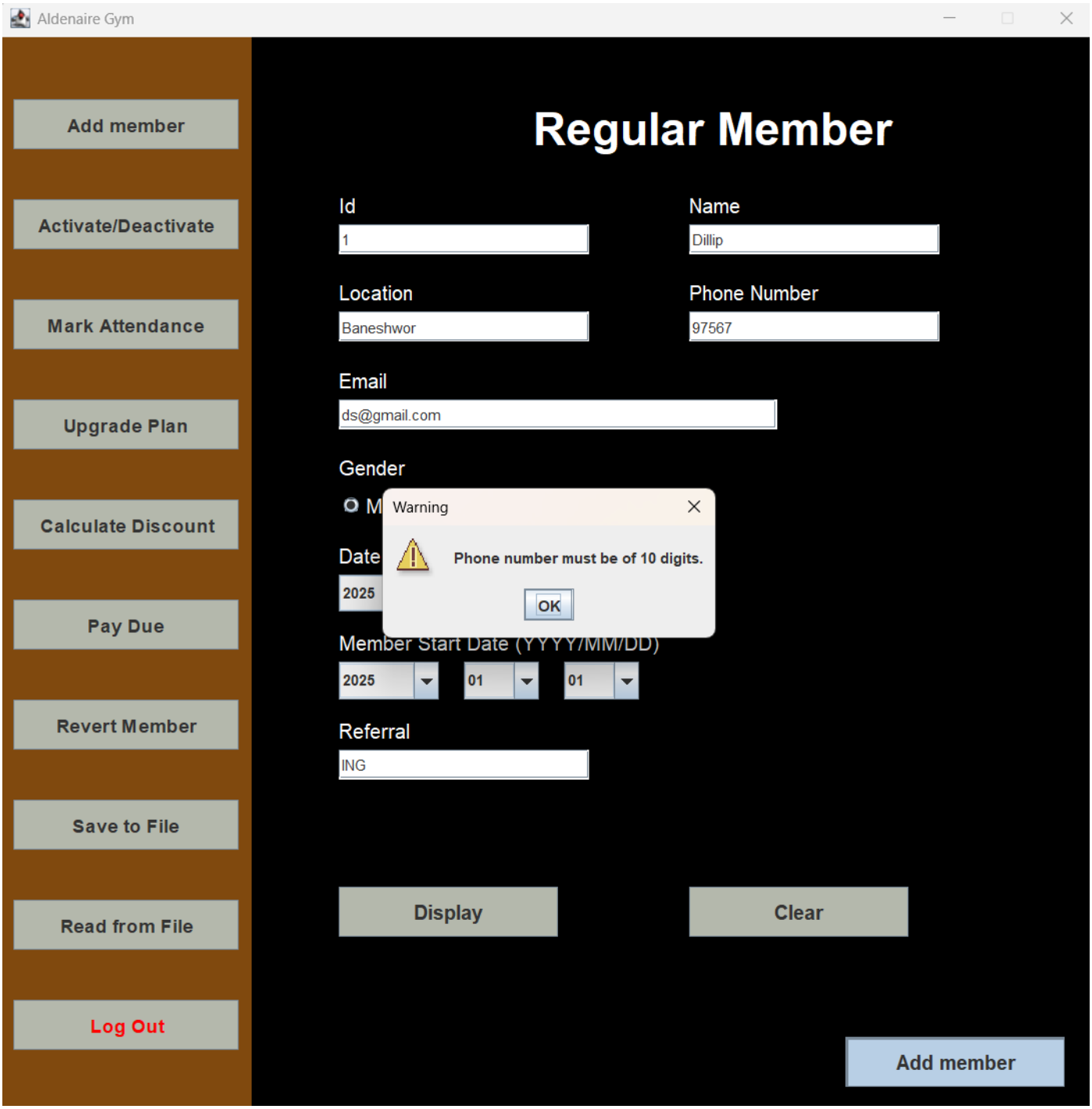


Figure 35: Phone Number must be of 10 digits

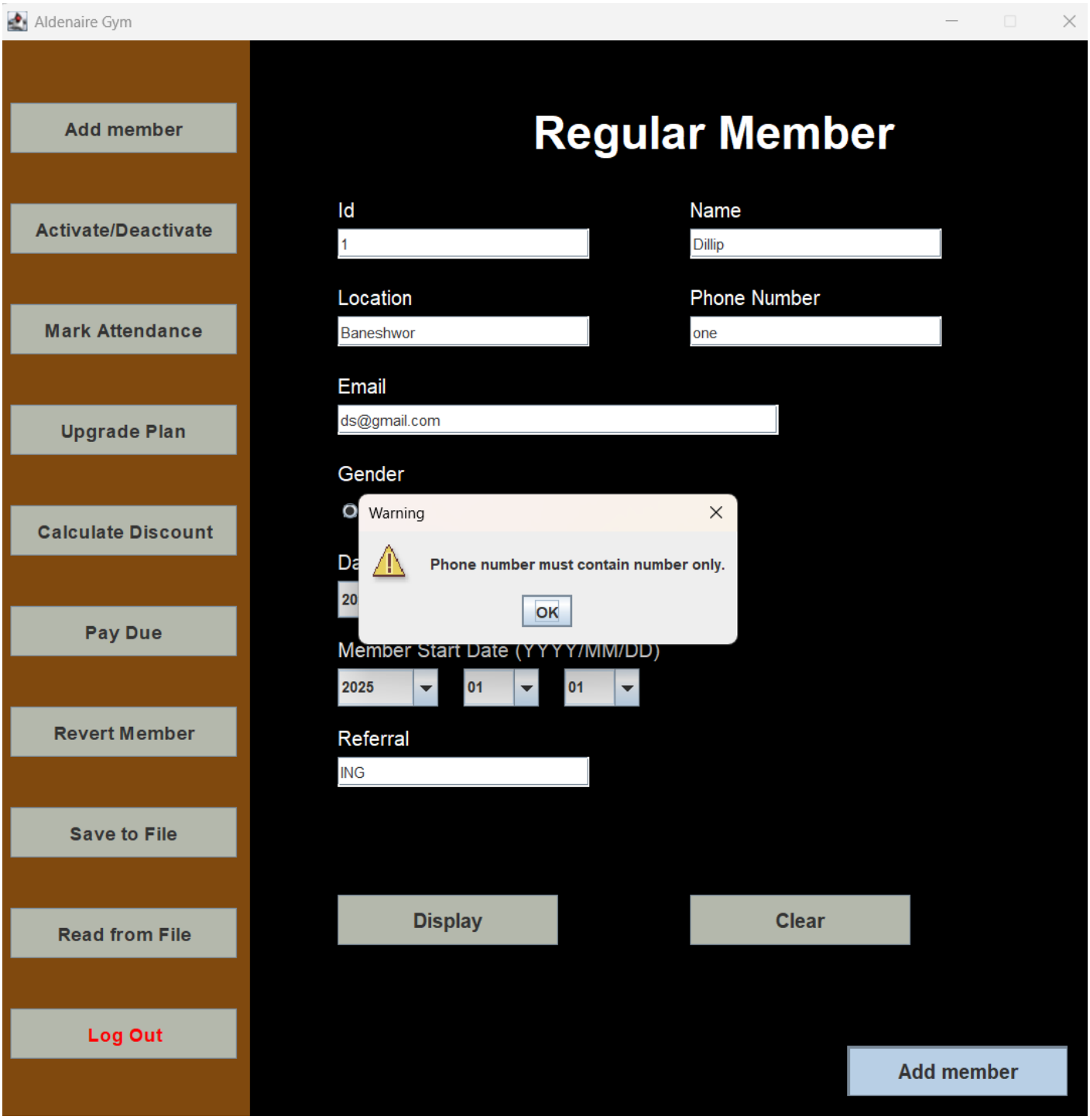


Figure 36: Phone number must contain number only

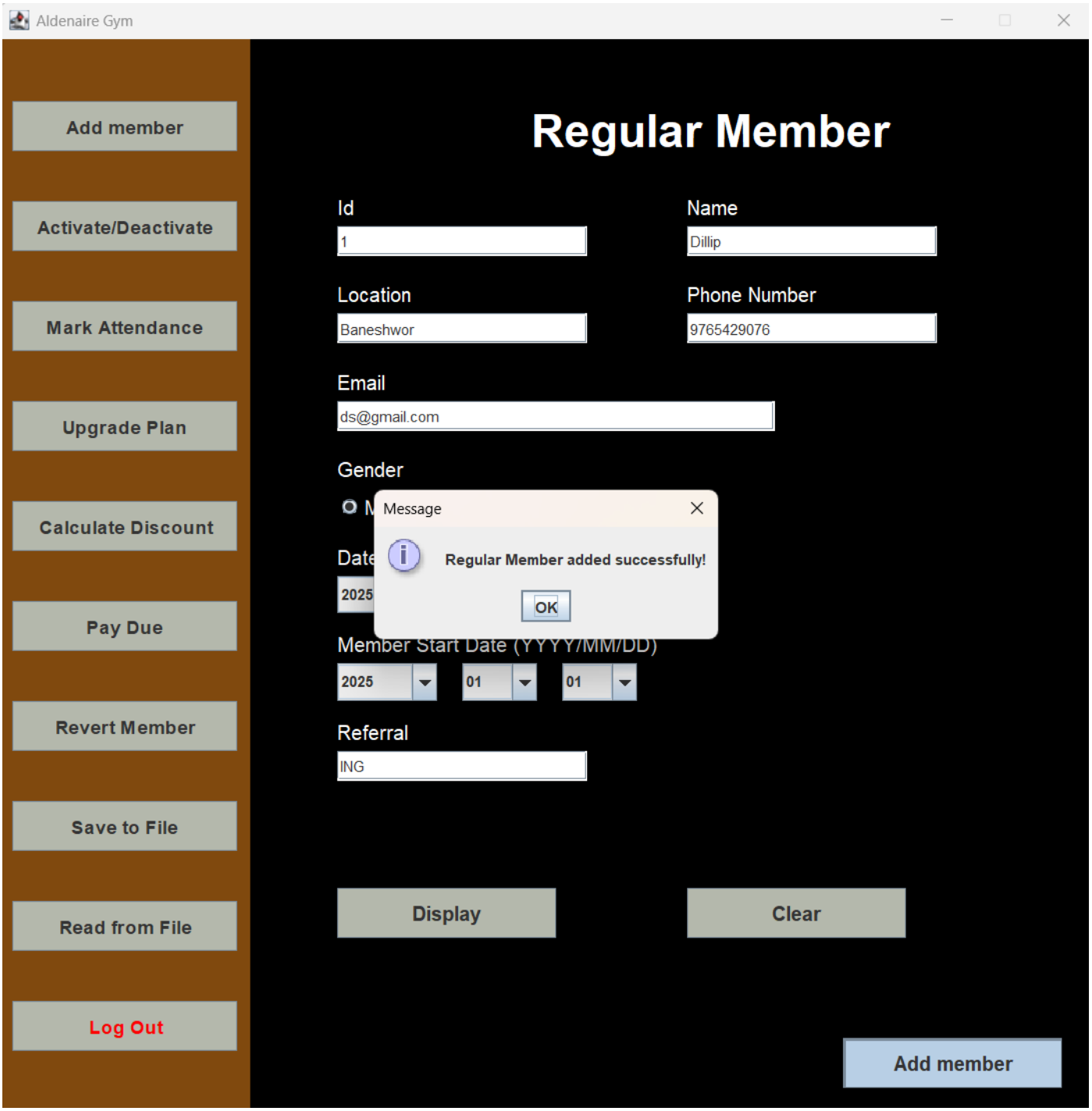


Figure 37: Successful addition of Regular Member

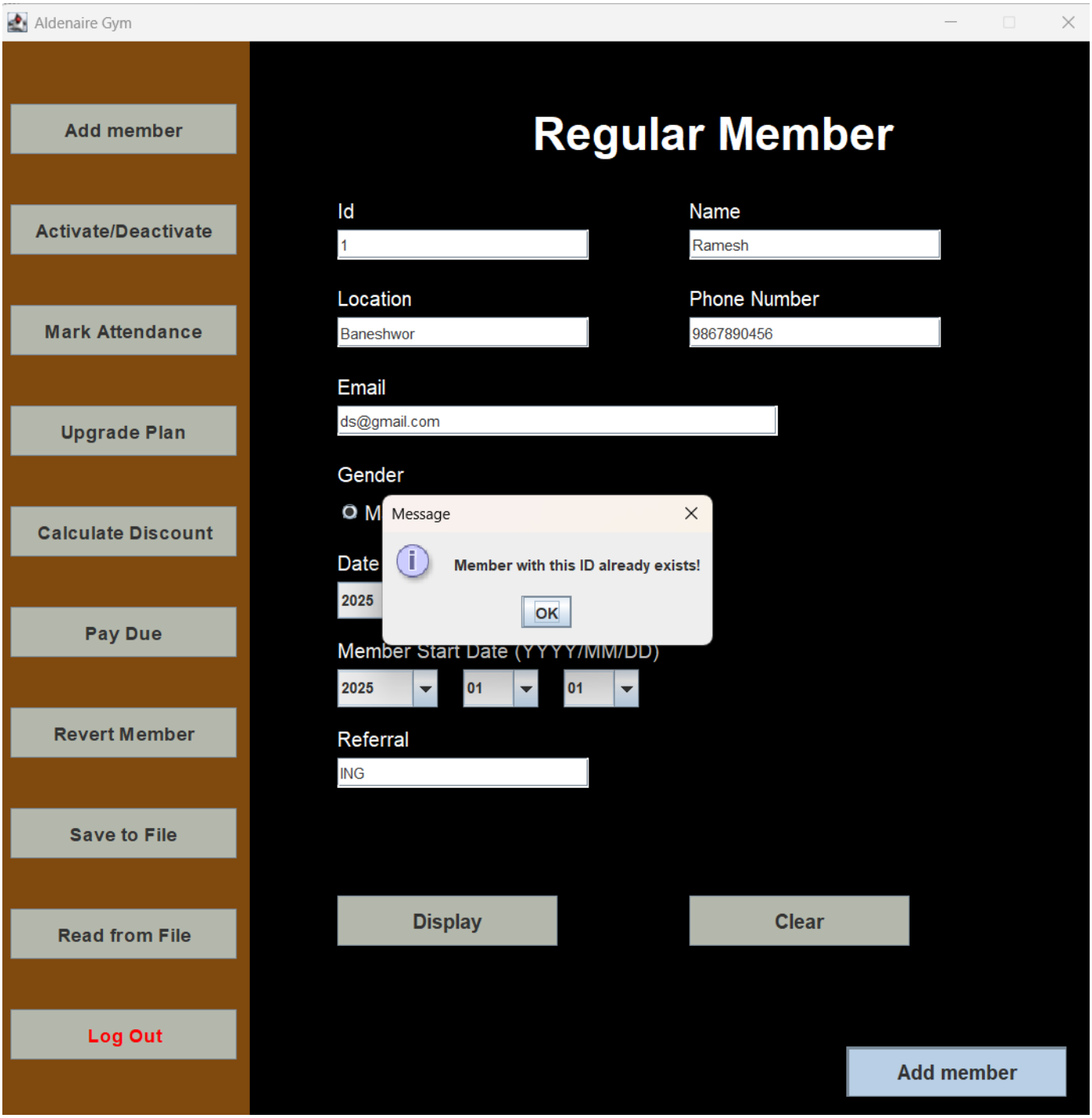


Figure 38: Duplicate id cannot be added

Add Premium Member

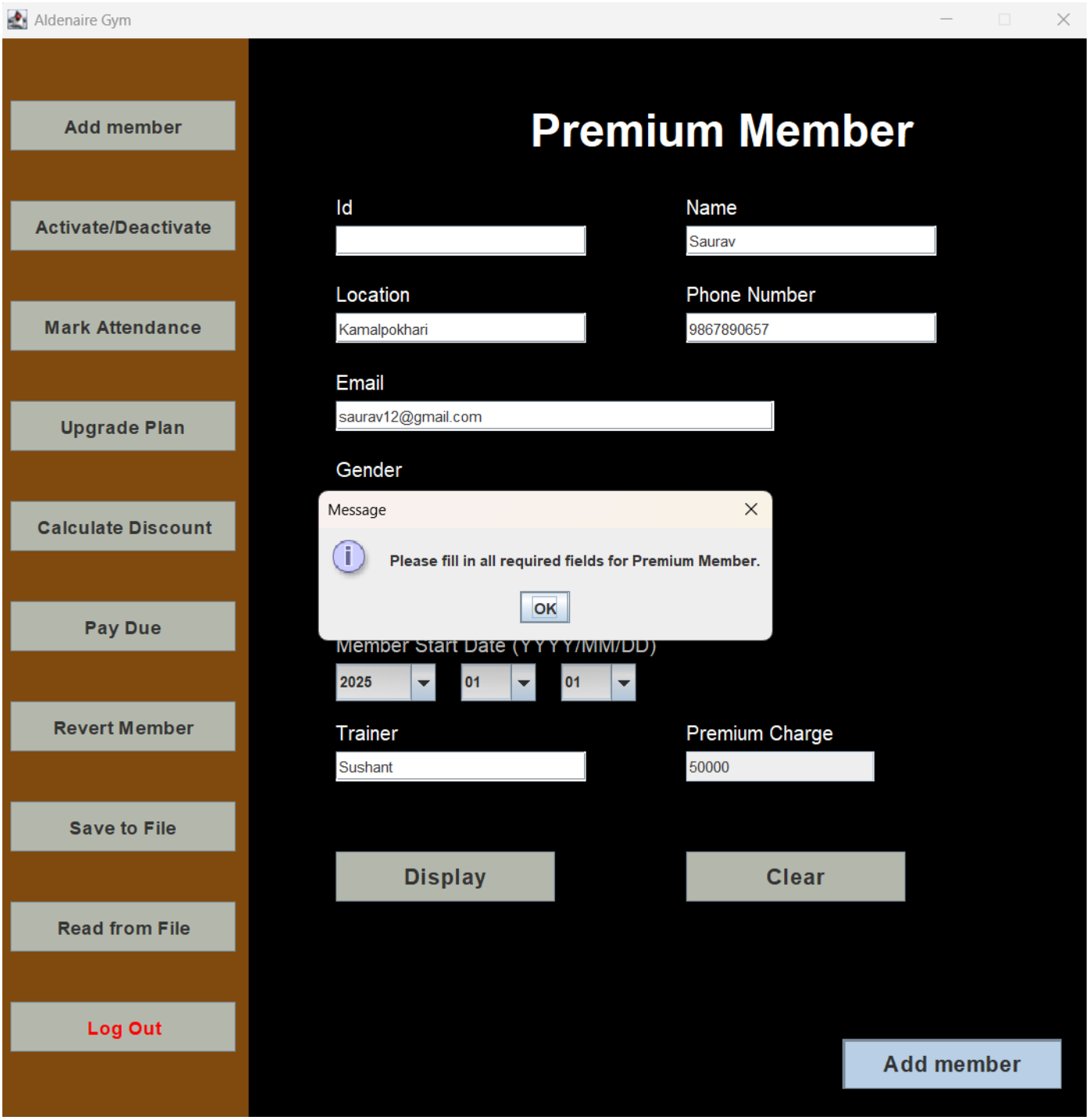


Figure 39: Keeping 1 field empty in Premium Member

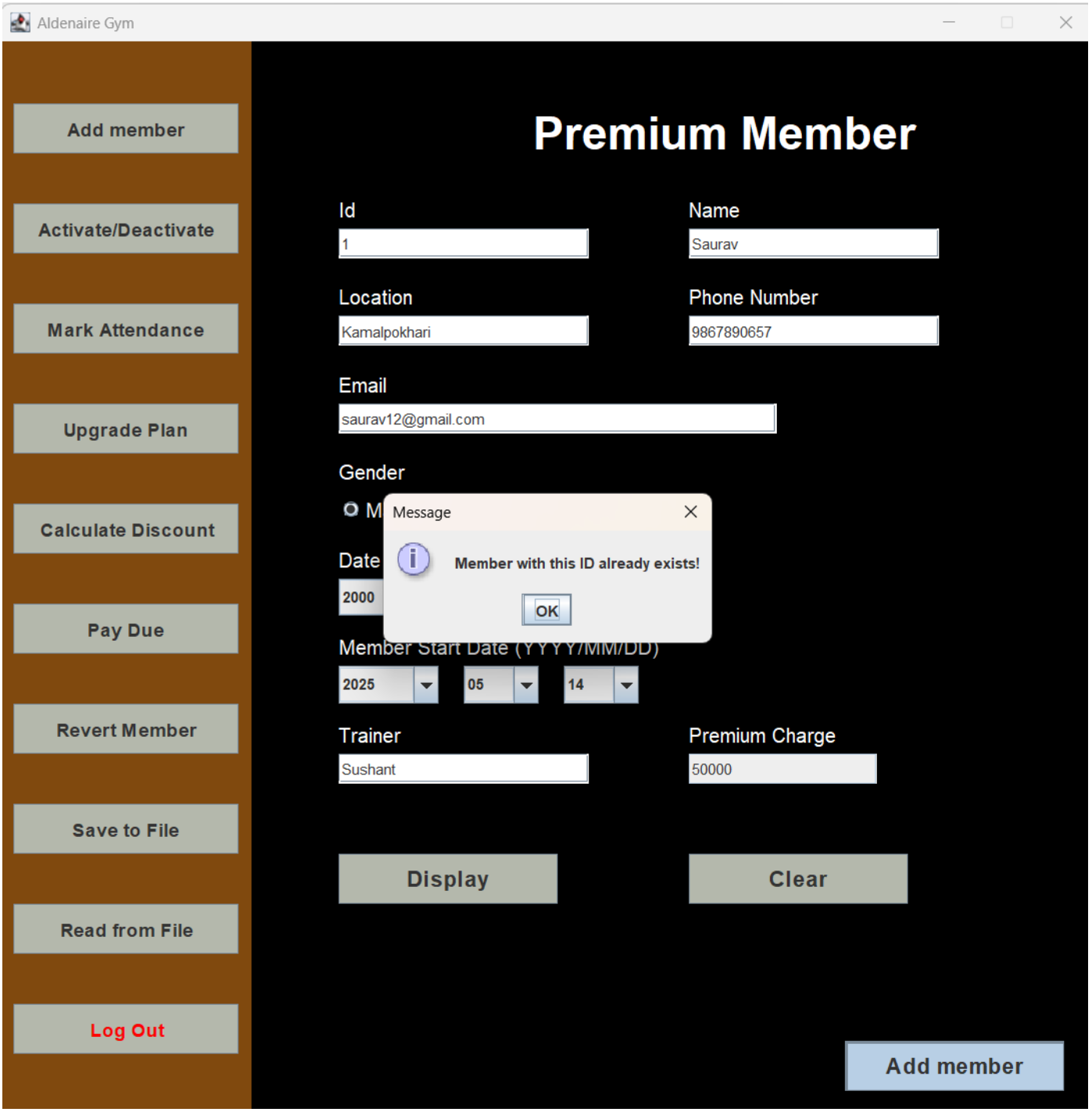


Figure 40: Duplicate id cannot be used

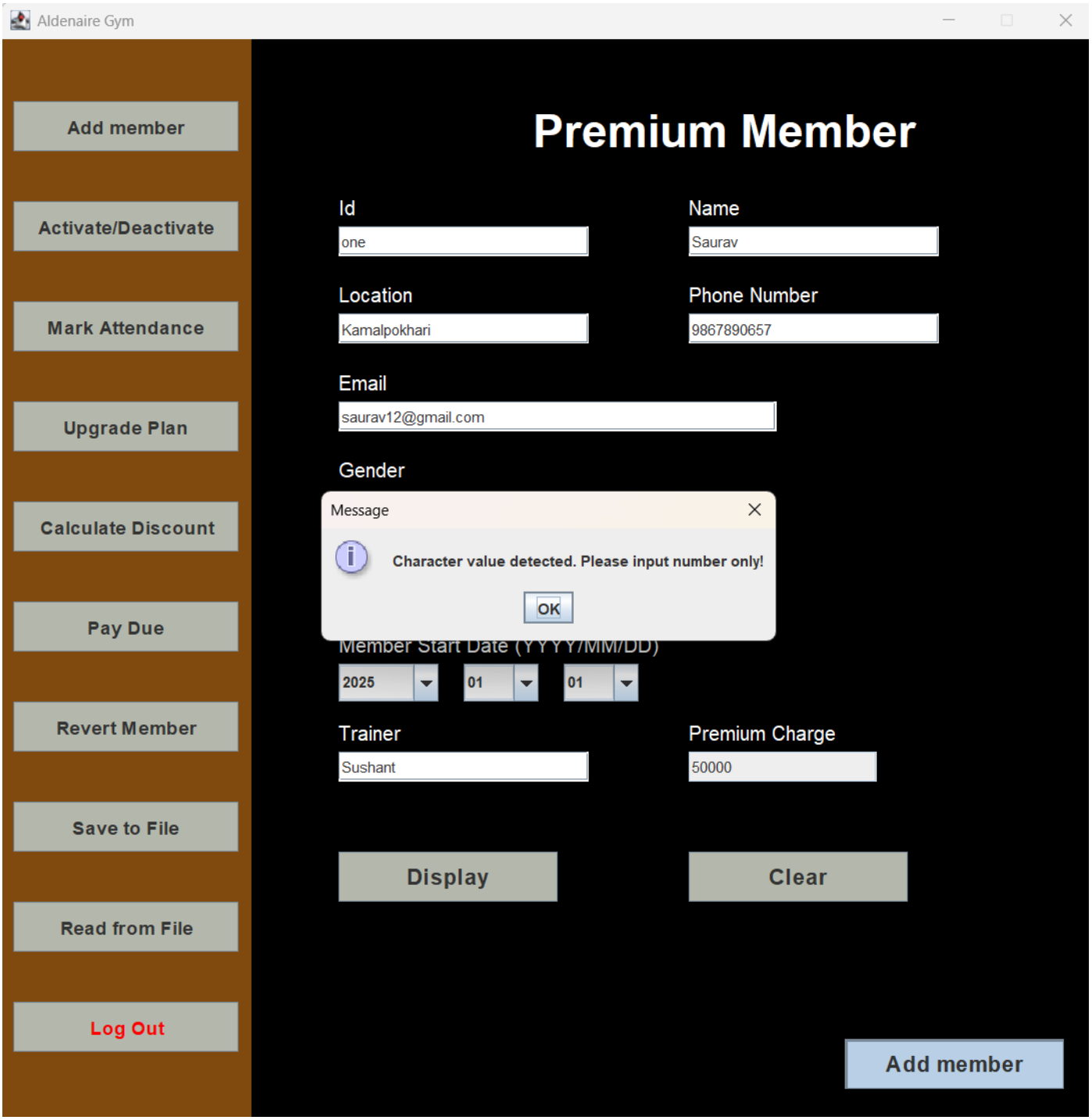


Figure 41: Passing character in id (Exception Handling)

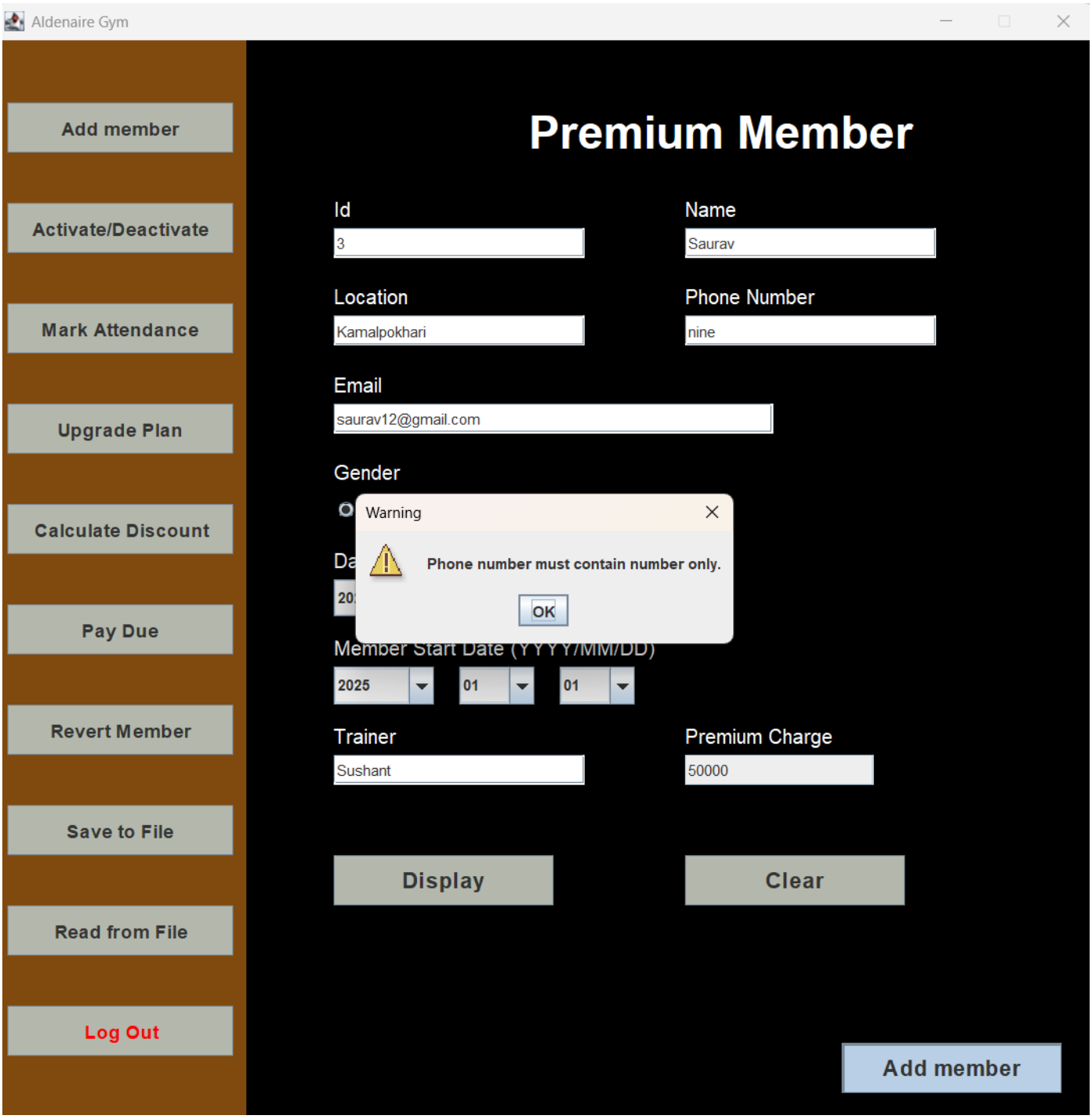


Figure 42: Passing character in Phone Number

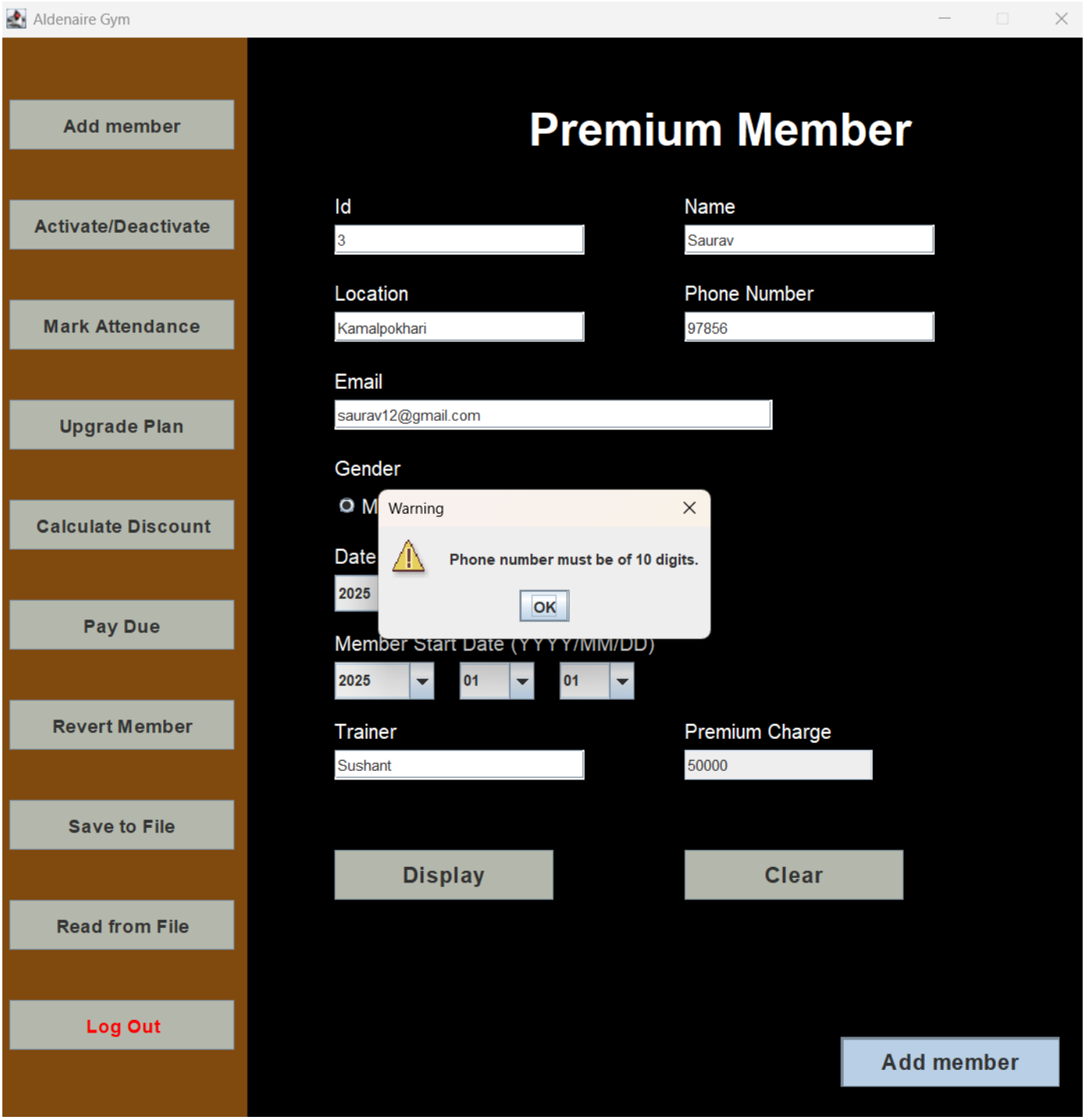


Figure 43: Passing digits less or more than 10 in Phone Number

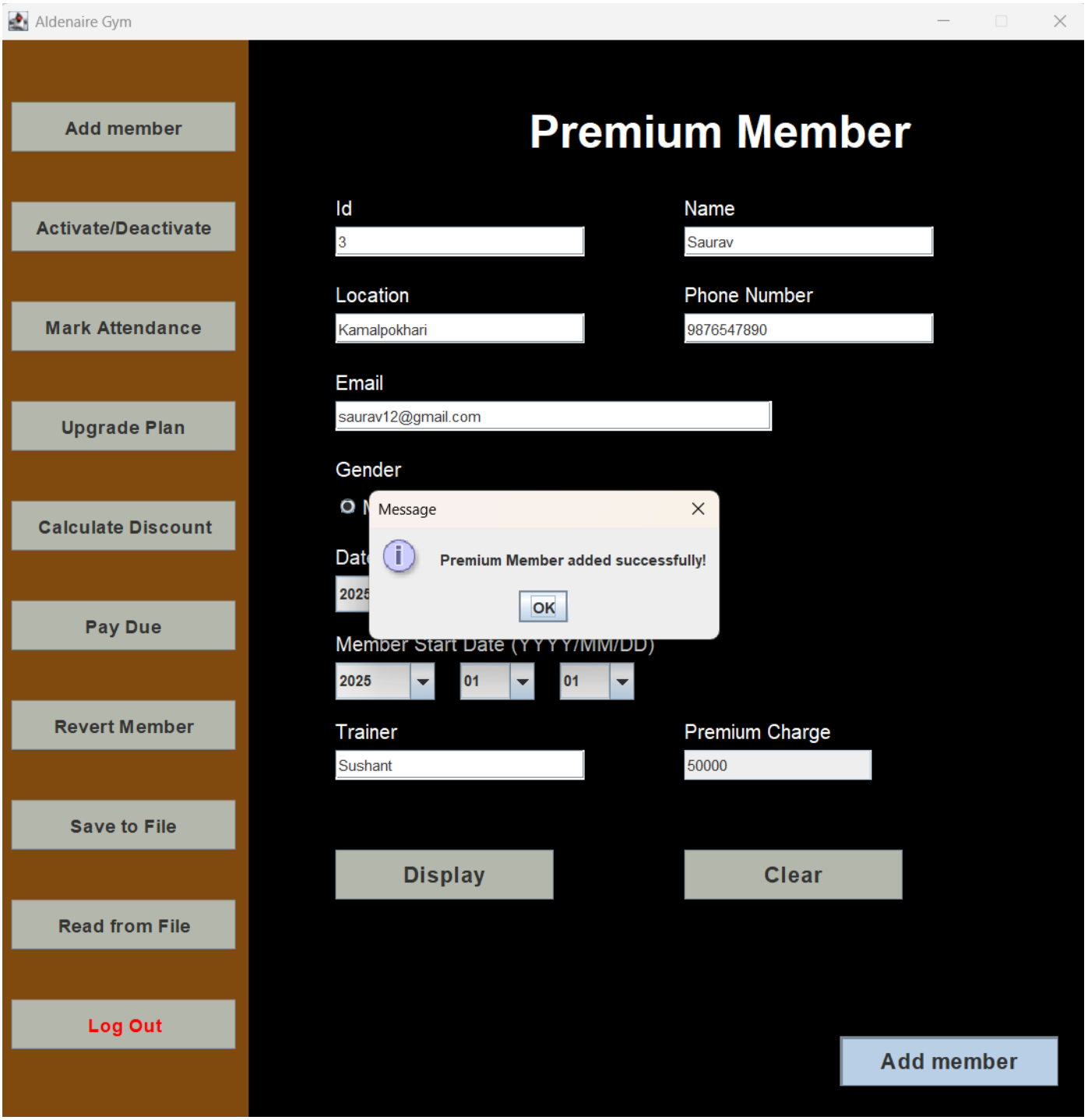


Figure 44: Successful addition of Premium Member

Testing 3

Testing 3.1

Table 3: Testing 3.1: Testing Case for Mark Attendance

Objective	To test case for mark attendance.
Action	i. Id of regular member entered in the “Member Id” text field. ii. Id of premium member entered in the “Member Id” text field.
Expected Output	The attendance of member should be marked and an appropriate message in a dialog box should appear.
Actual Output	The attendance of member was marked and an appropriate message in a dialog box was displayed.
Result	The test was successful.

Mark Regular Member

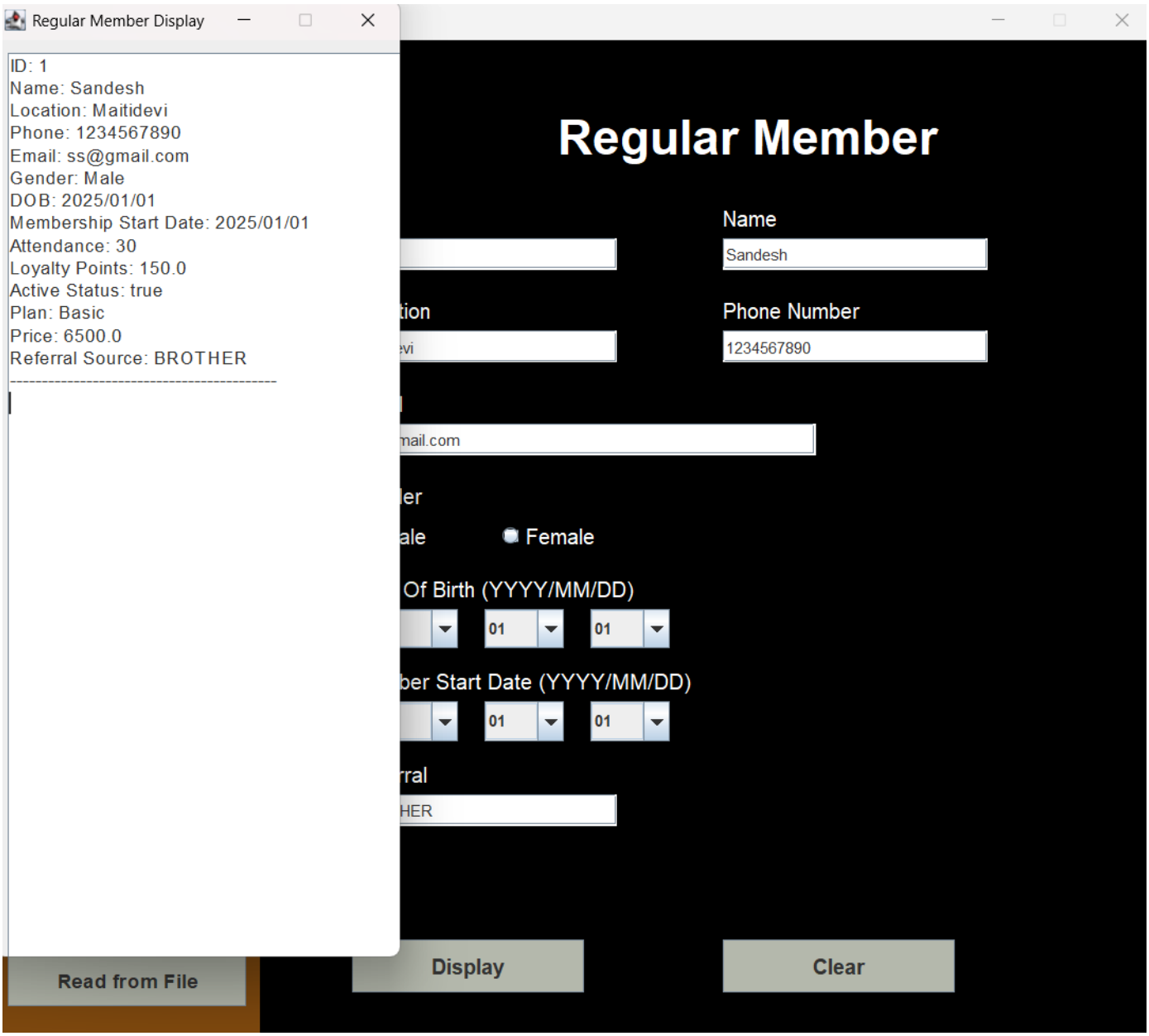


Figure 45: Display Info before Mark Regular Member

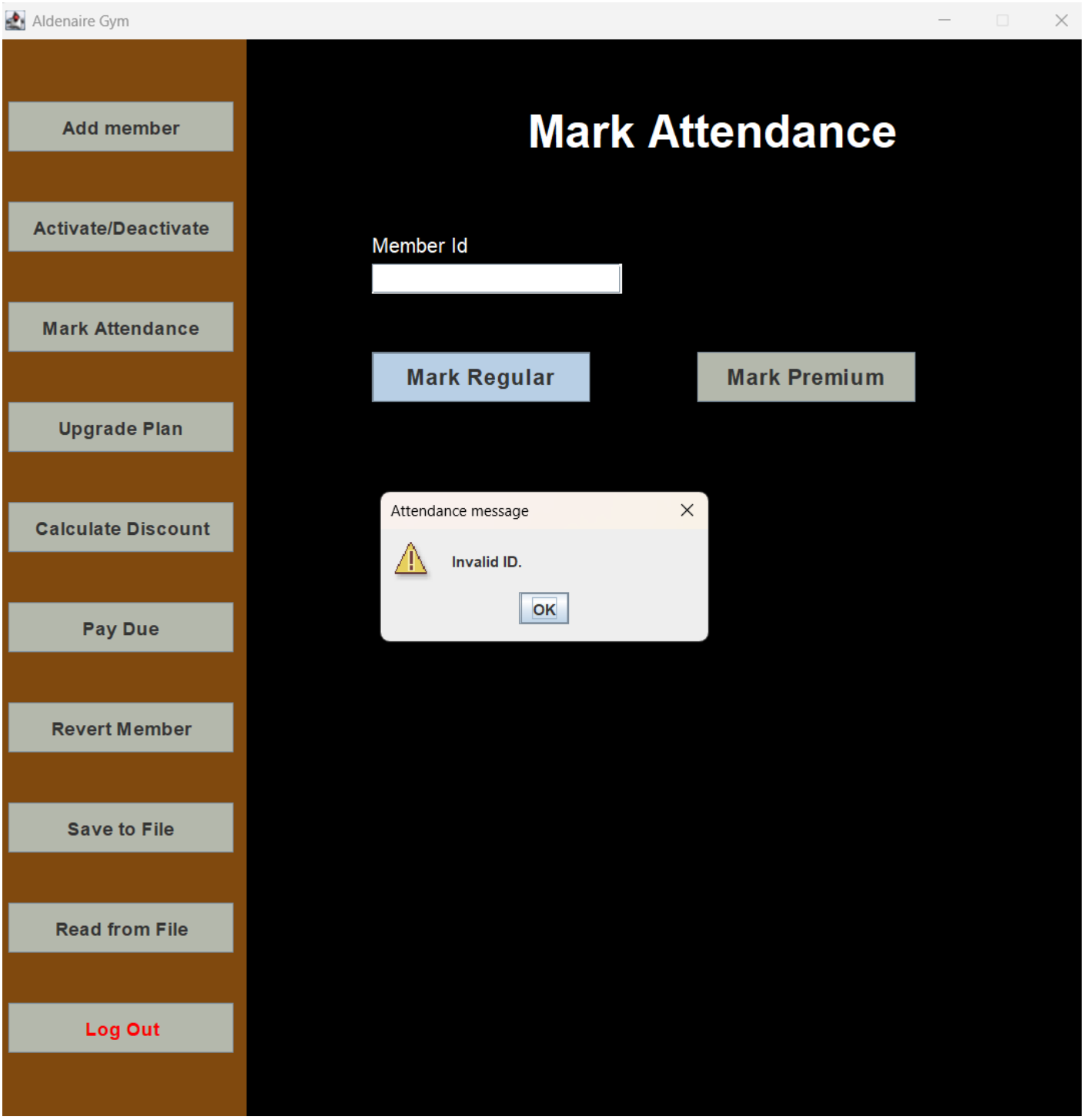


Figure 46: Passing Empty id in Mark Regular

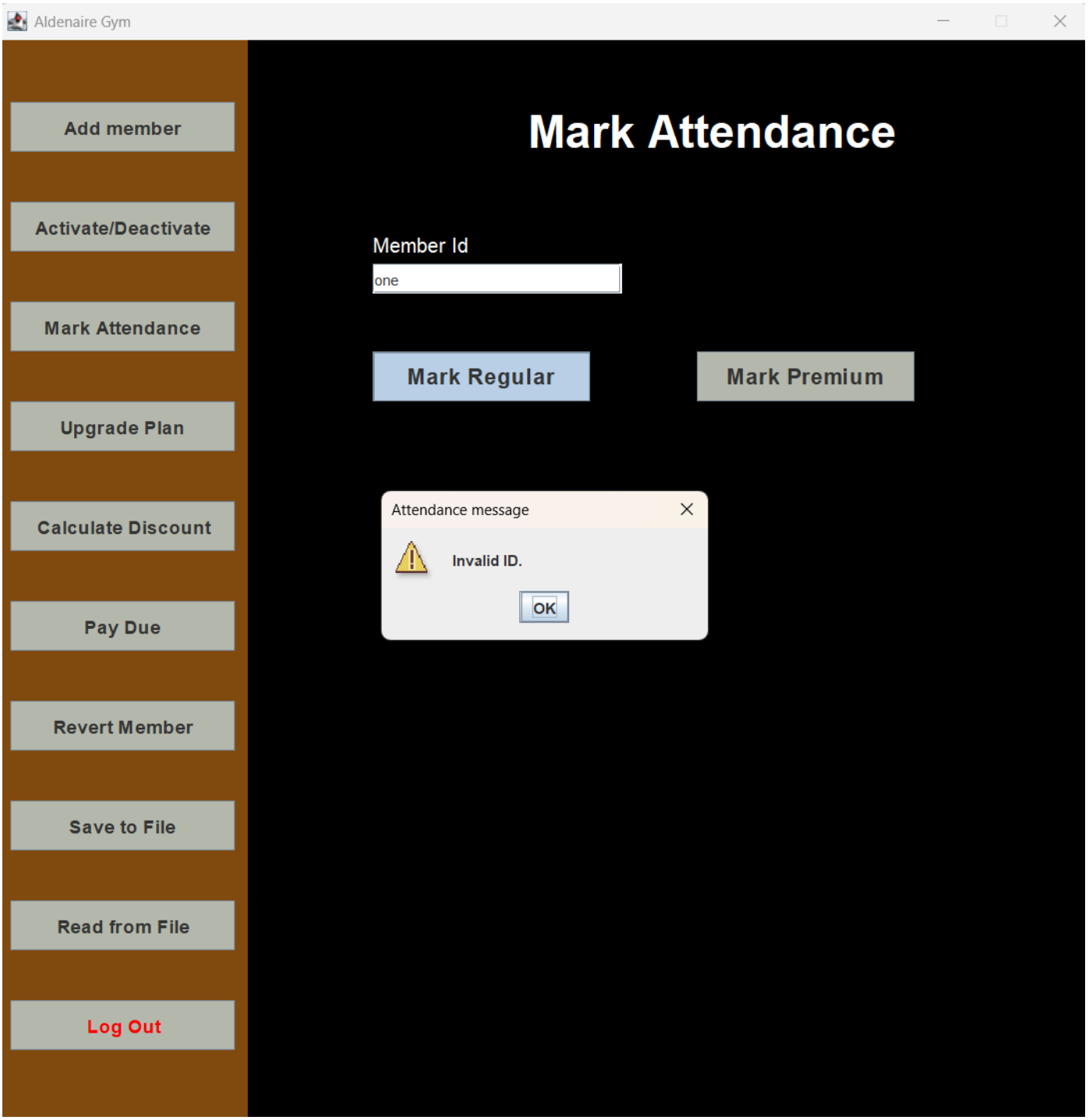


Figure 47: Passing Character in Mark Regular

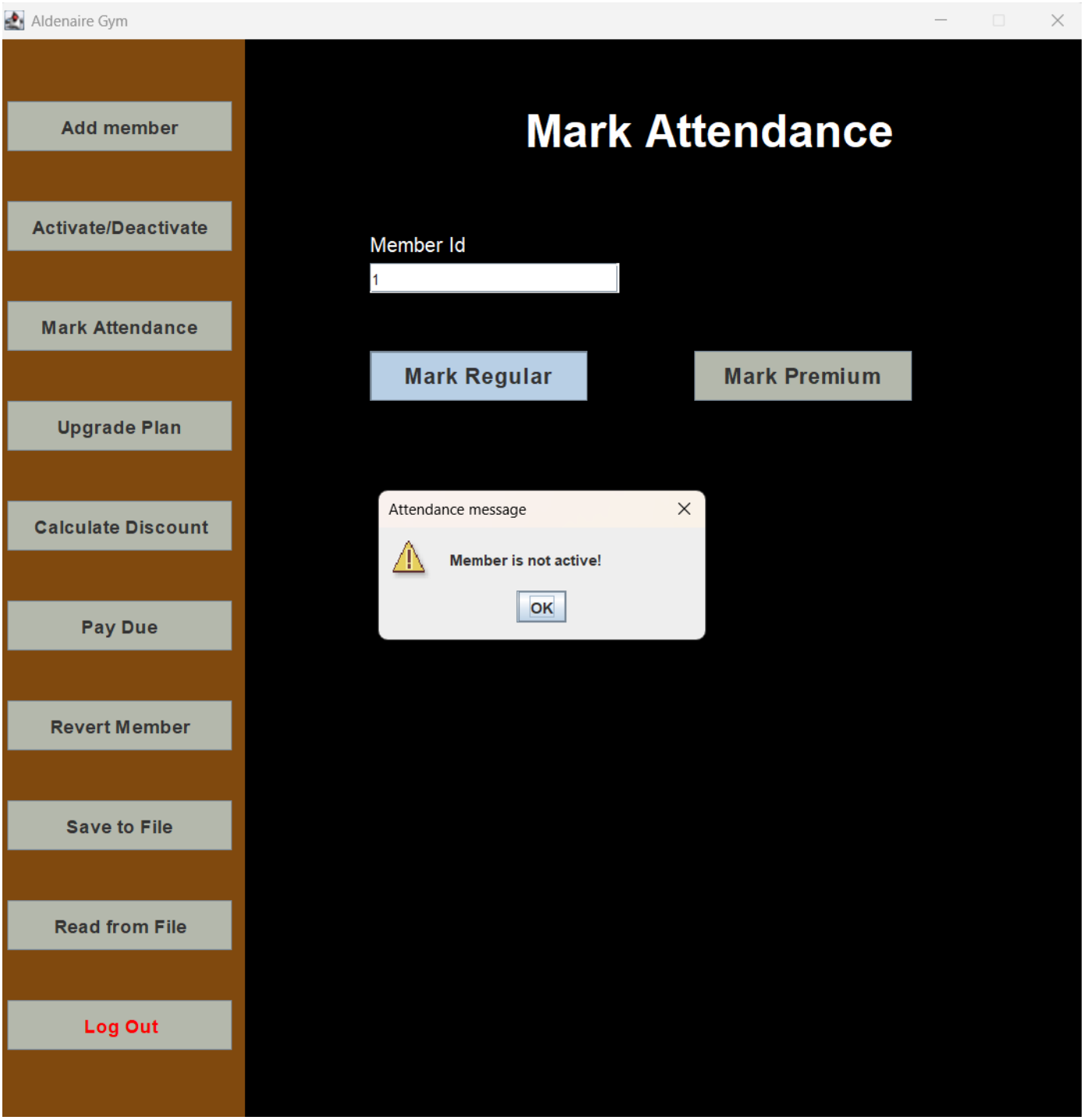


Figure 48: Passing correct id but member is not active yet in Regular

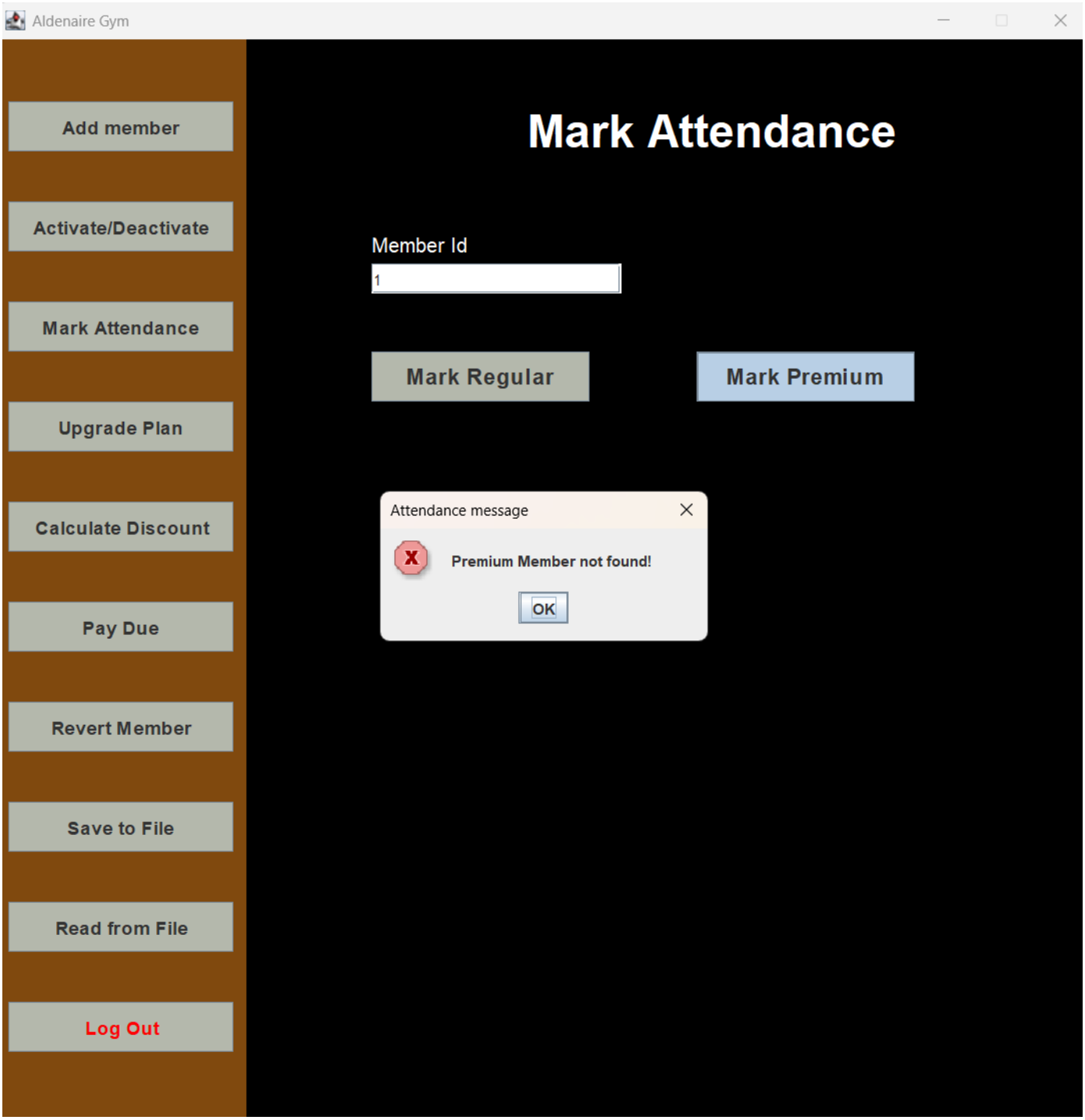


Figure 49: Passing correct Regular id but clicked Mark Premium

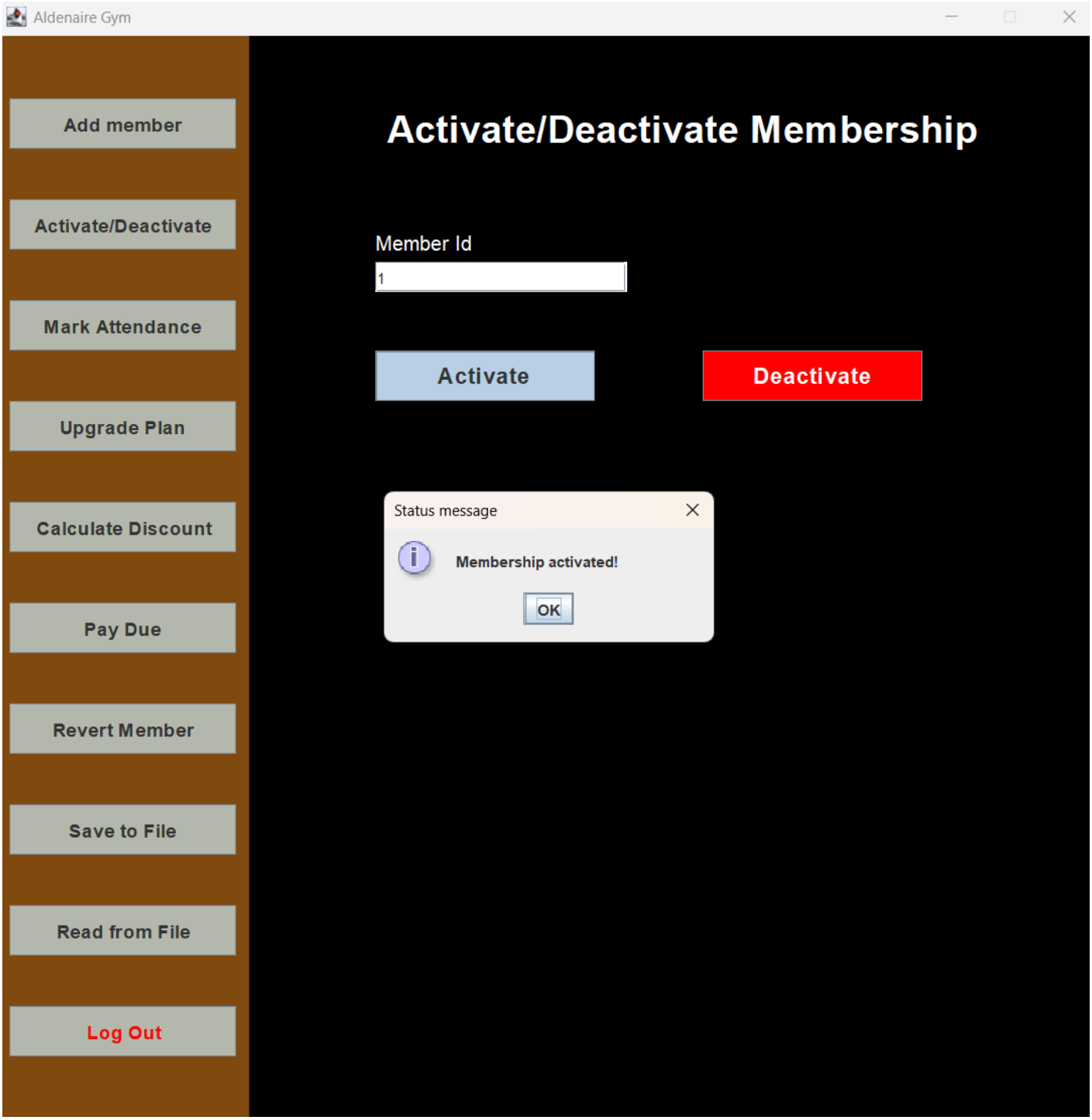


Figure 50: Activating member to Mark Regular Member

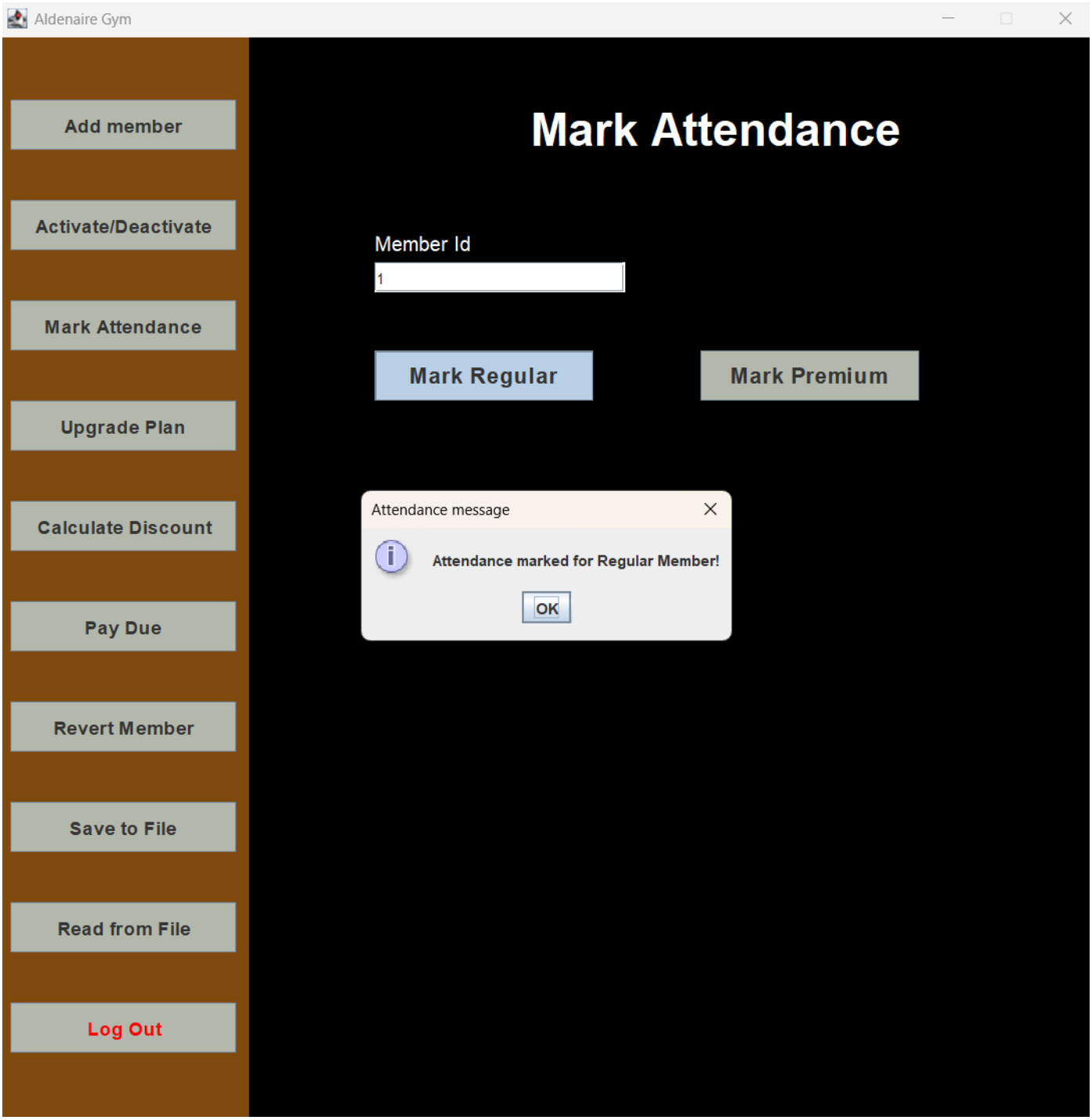


Figure 51: Attendance of Regular Member was Marked

Mark Premium Member

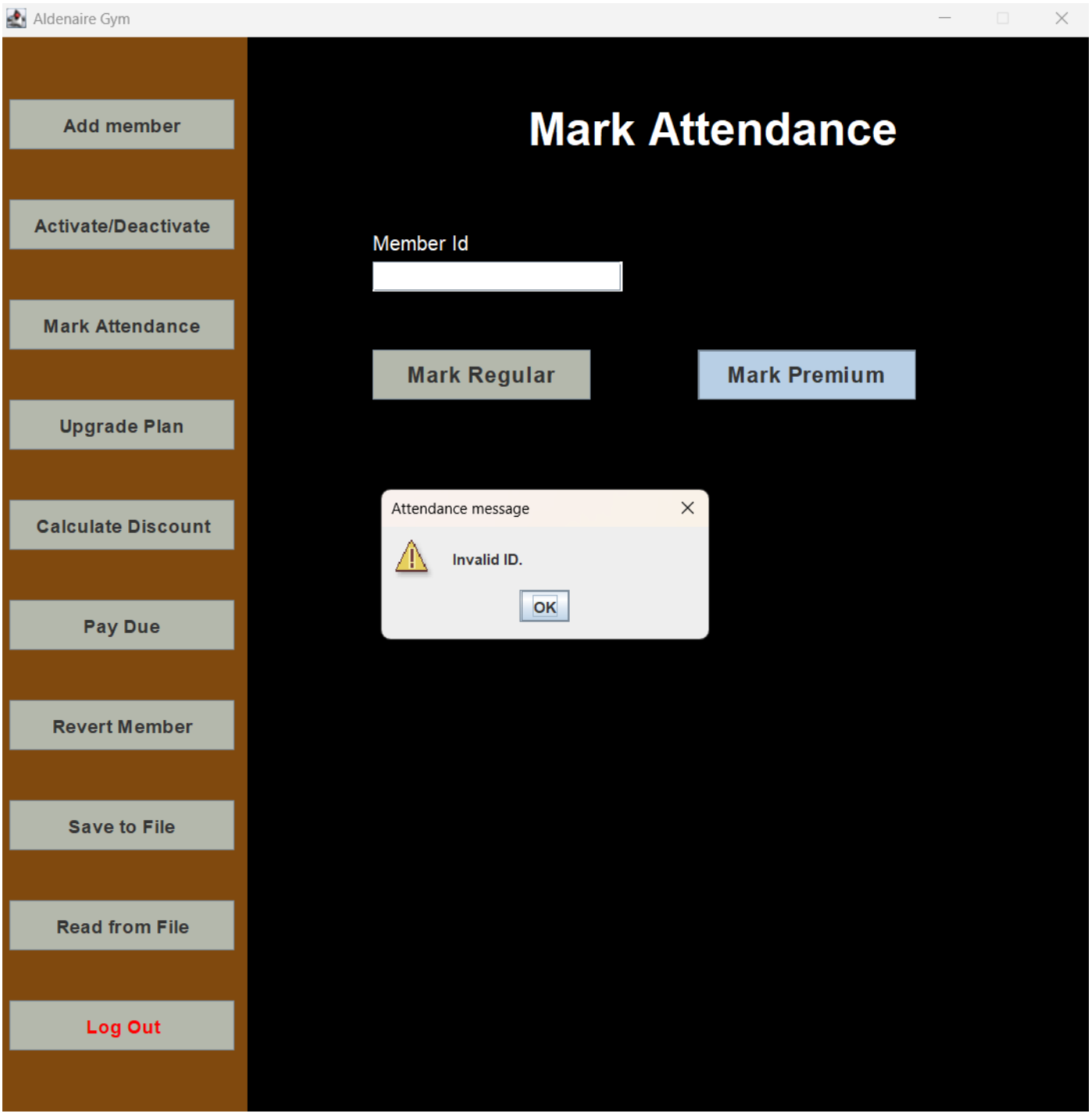


Figure 52: Passing Empty id in Mark Premium

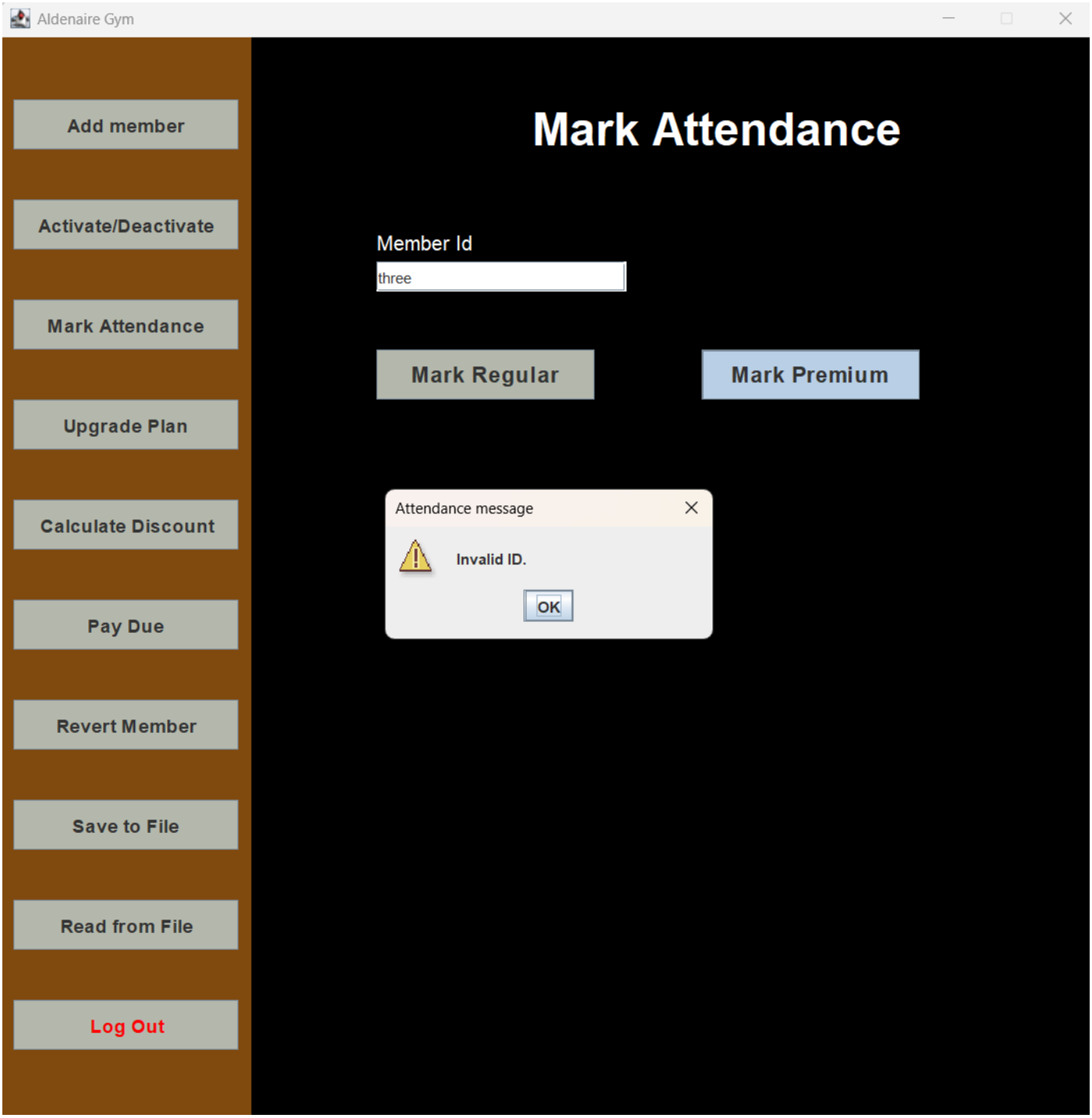


Figure 53: Passing Character in Mark Premium

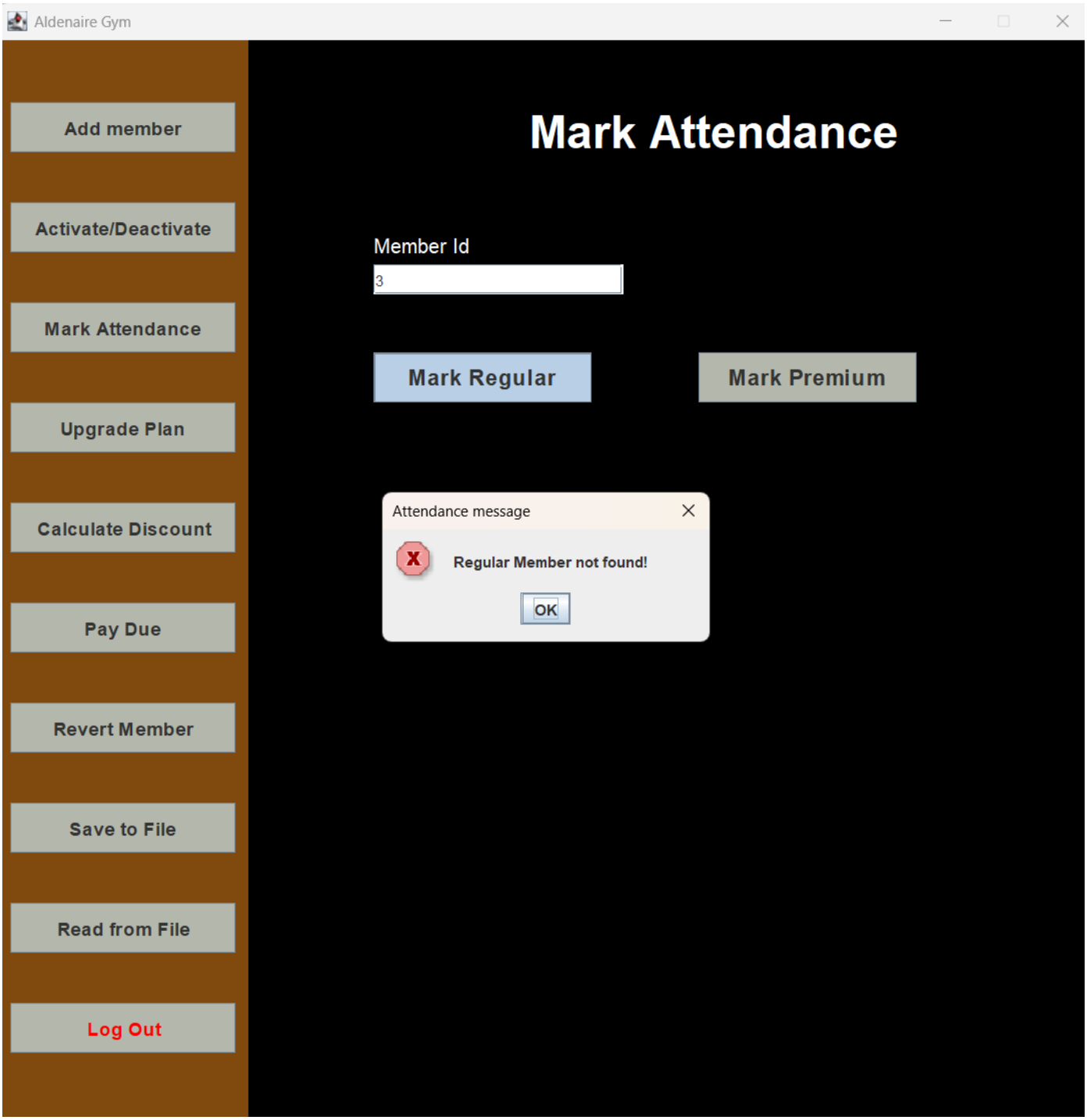


Figure 54: Passing correct member id but clicked Mark Regular

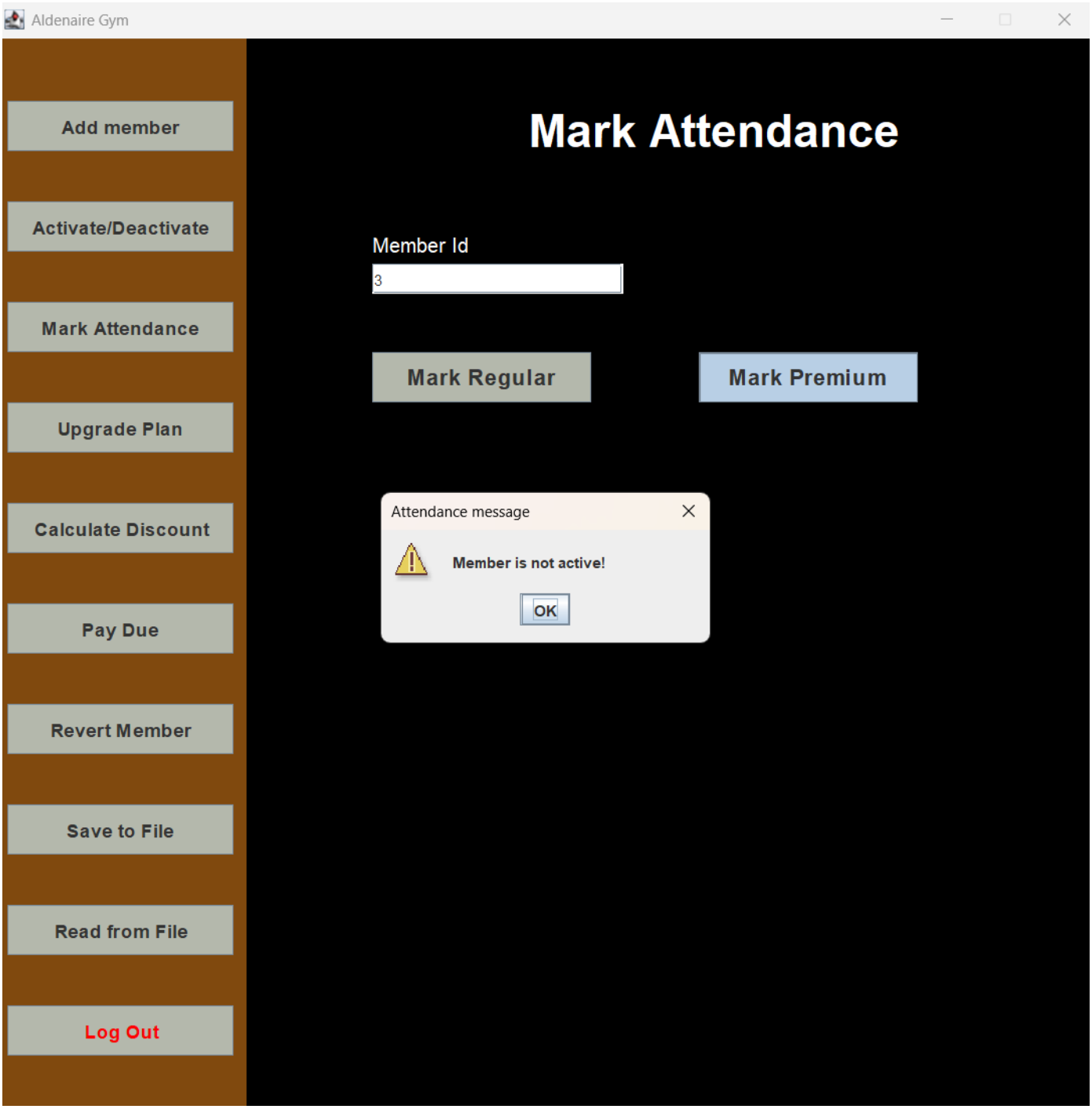


Figure 55: Passing correct member Id but Premium Member is not active

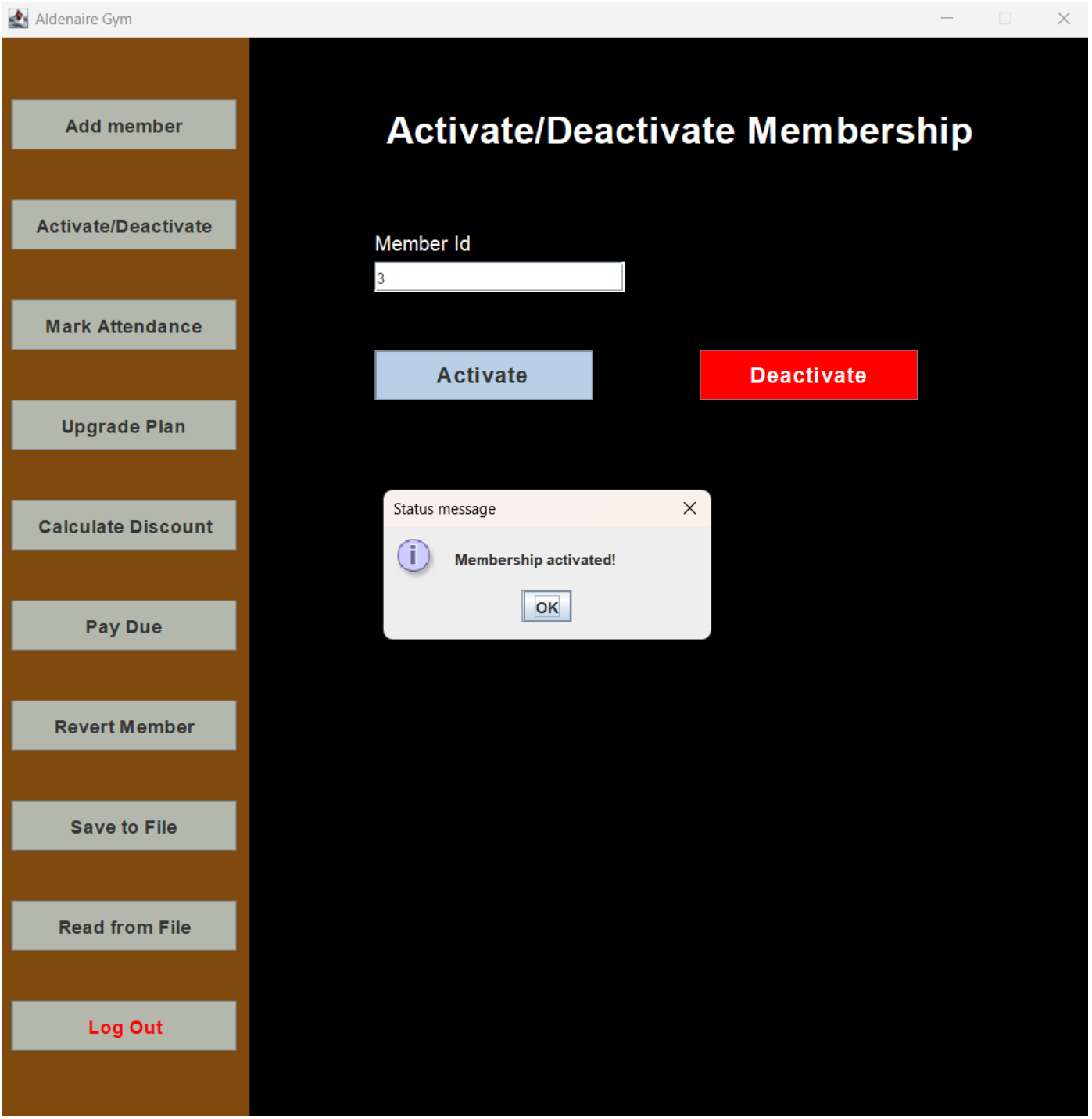


Figure 56: Premium Member was activated in order to Mark Premium Member

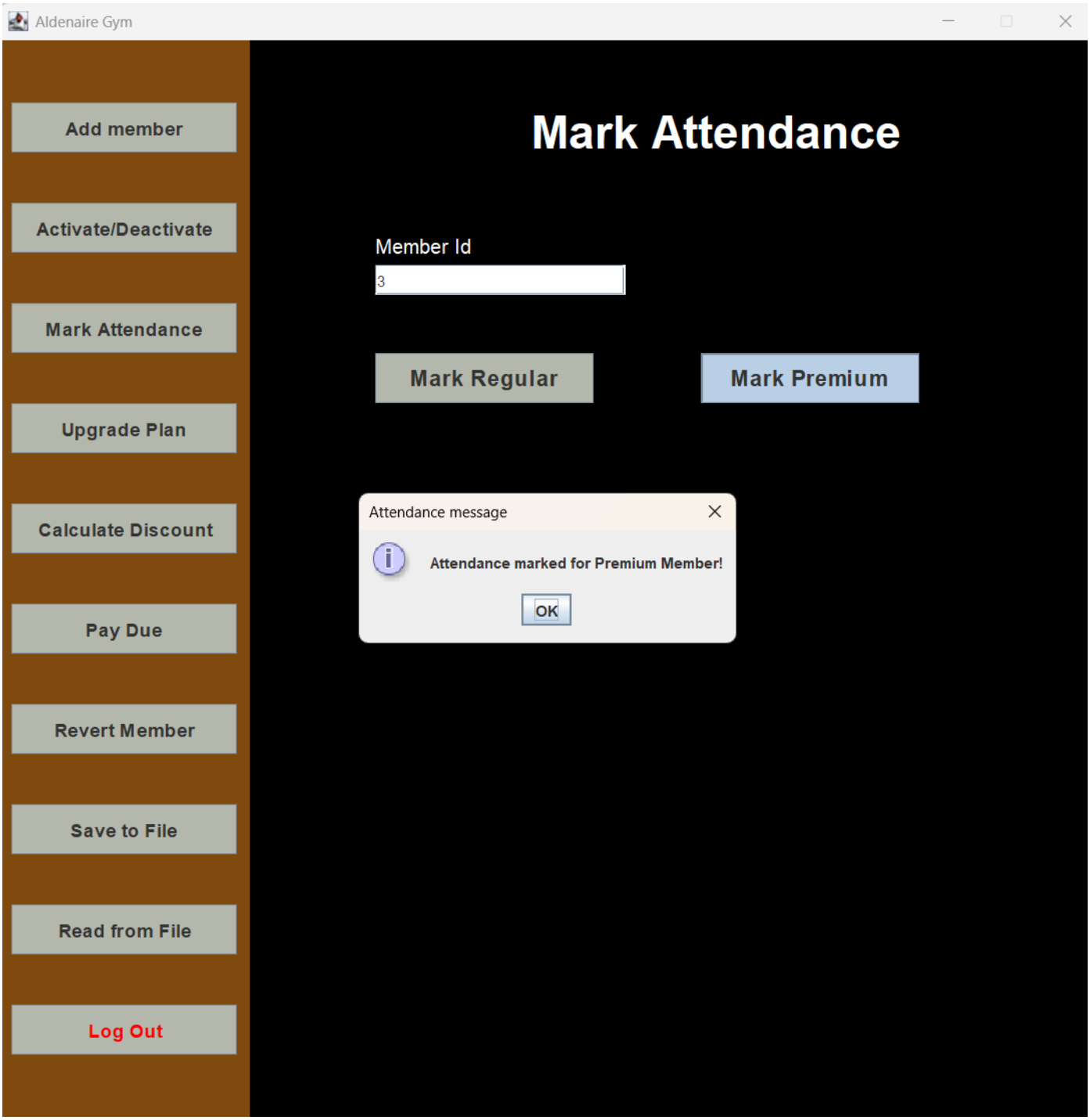


Figure 57: Attendance of Premium Member was Marked

Testing 3.2

Table 4: Testing 3.2: To test case for upgrade plan

Objective	To test case for upgrade plan
Action	i. Added a regular member detail in add member with id “1”. ii. Entered empty id in upgrade plan. iii. Entered character in text field. iv. Entered not existing id. v. Clicked Activate Member. vi. Clicked Mark attendance. vii. Selected standard plan from JComboBox. viii. Display to check
Expected Output	The plan of regular member should be upgraded to “Standard” and an appropriate message in a dialog box should appear.
Actual Output	The plan of regular member was upgraded to “Standard” and an appropriate message in a dialog box had appeared.
Result	The test was successful.

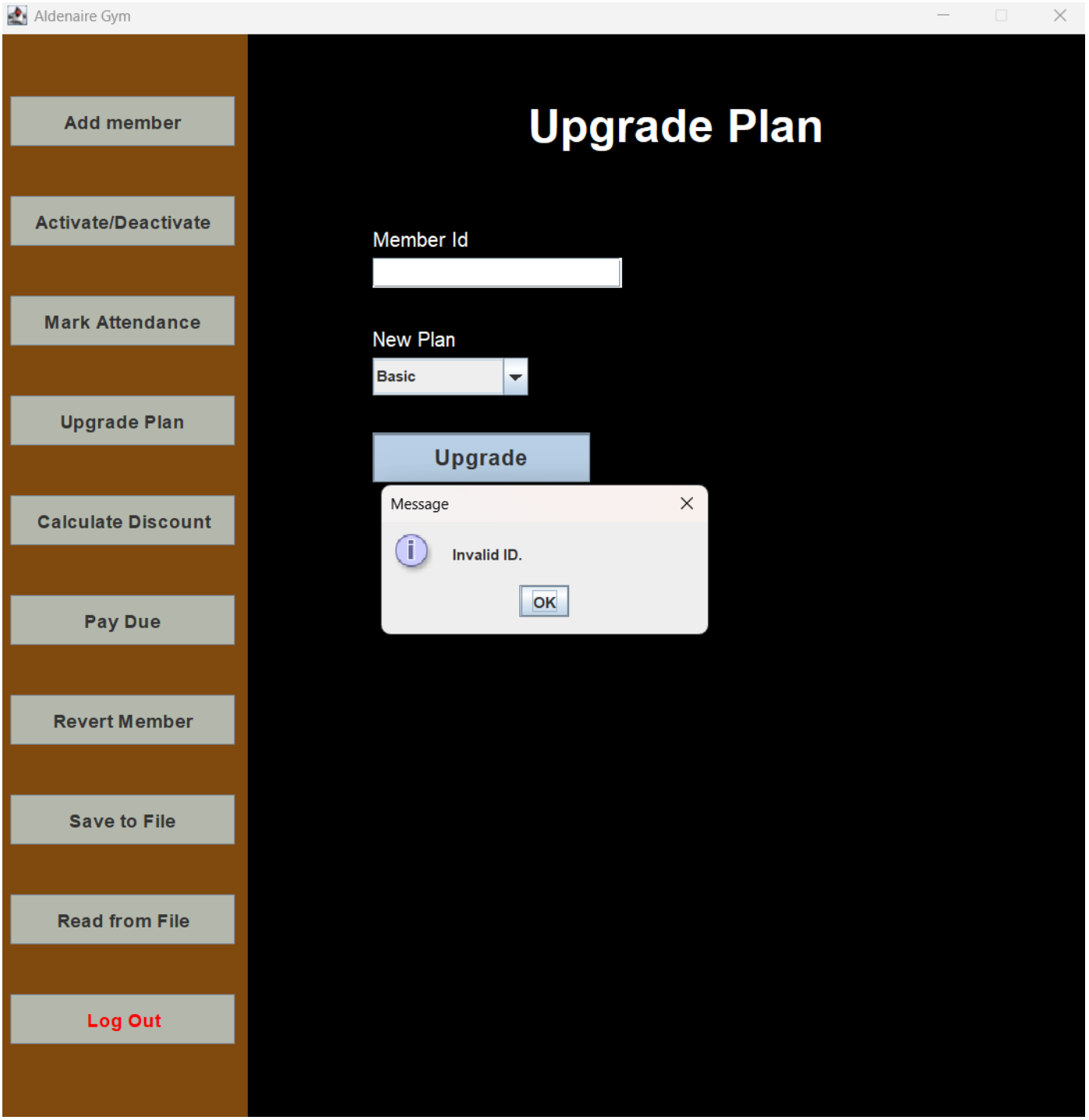


Figure 58: Passing Empty Id in Upgrade Plan

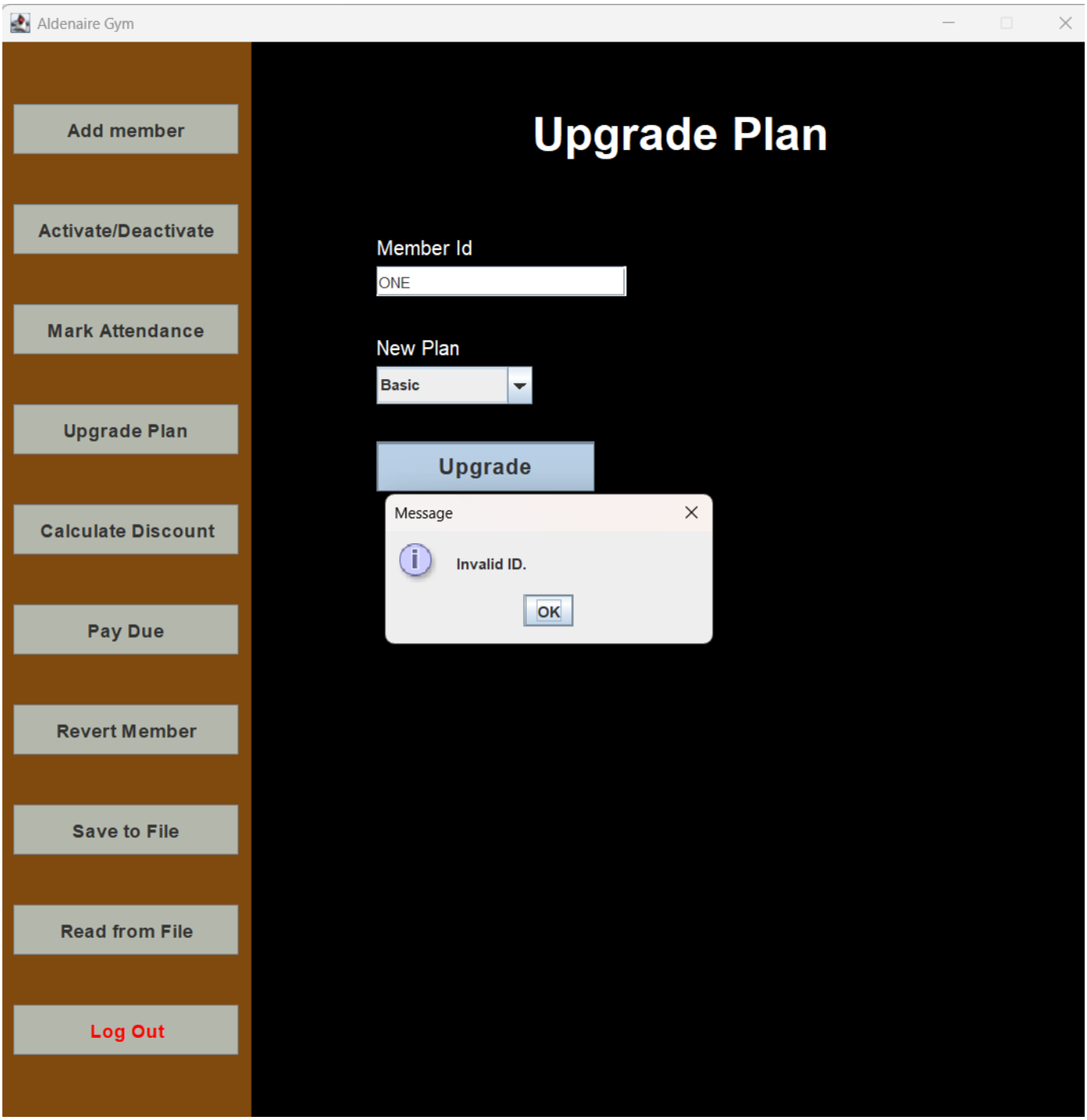


Figure 59: Passing Character in Upgrade Plan

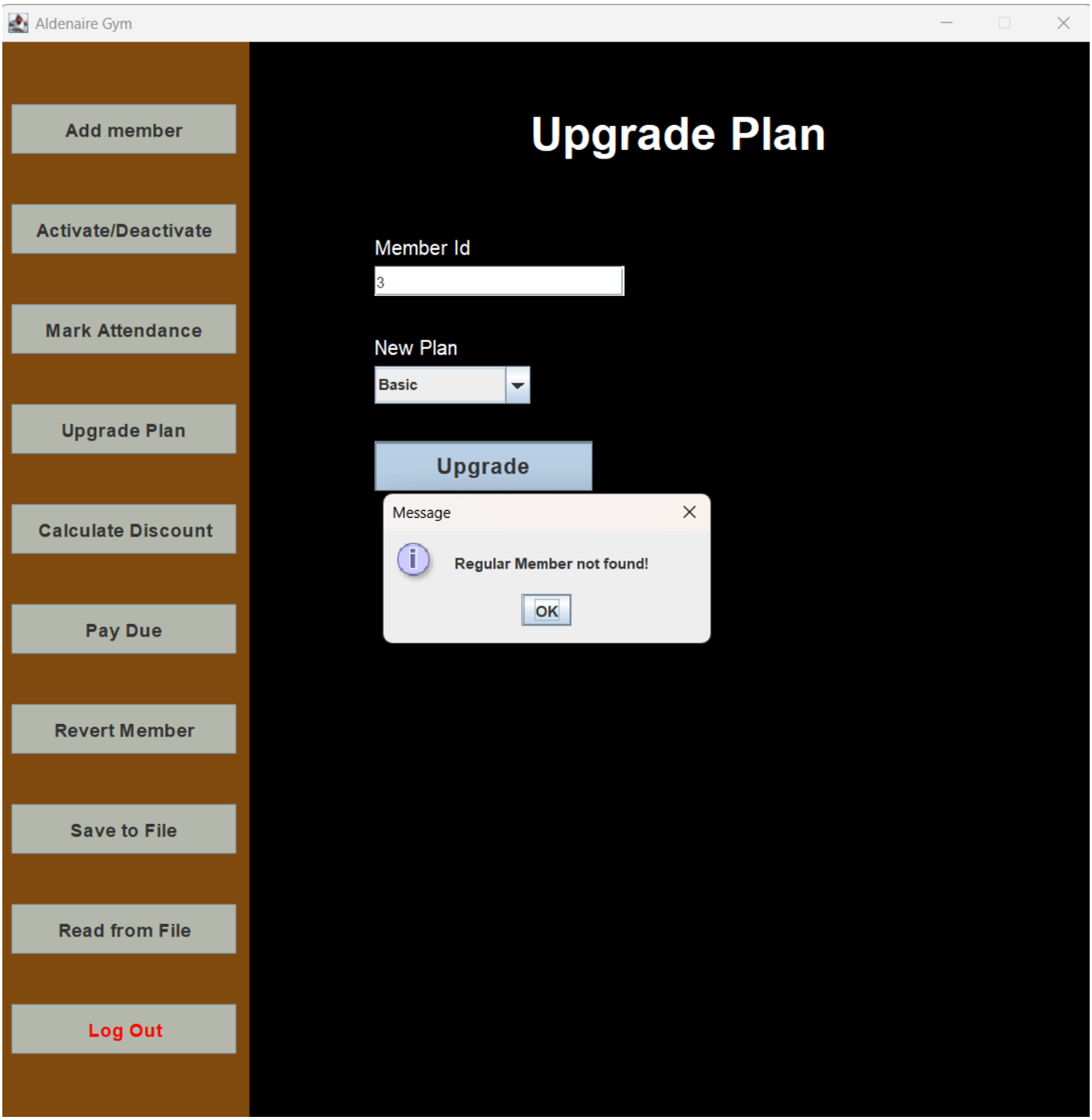


Figure 60: Passing Non existing Id in Upgrade Plan

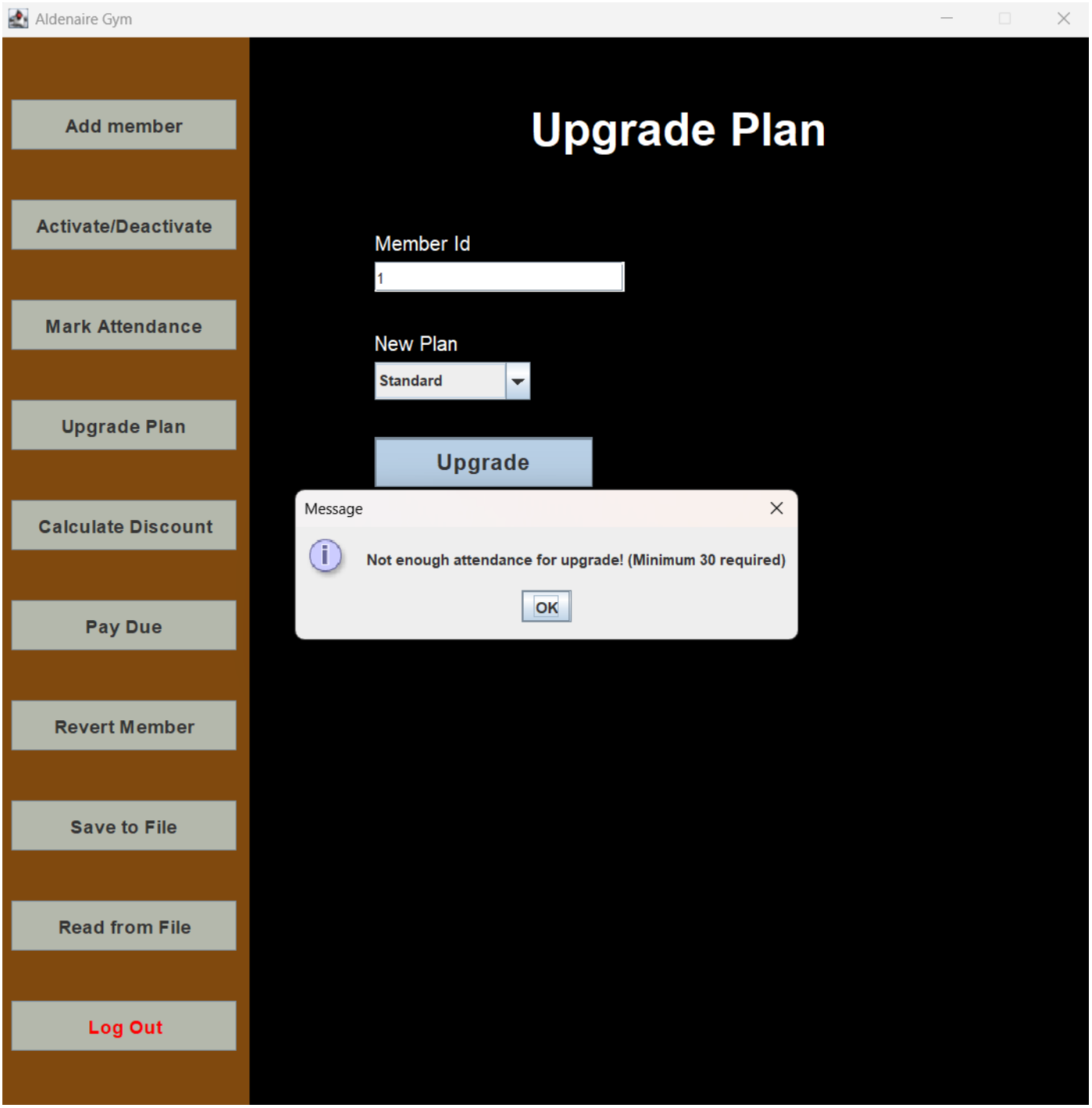


Figure 61: Passed correct id but member is not eligible for Plan Upgrade

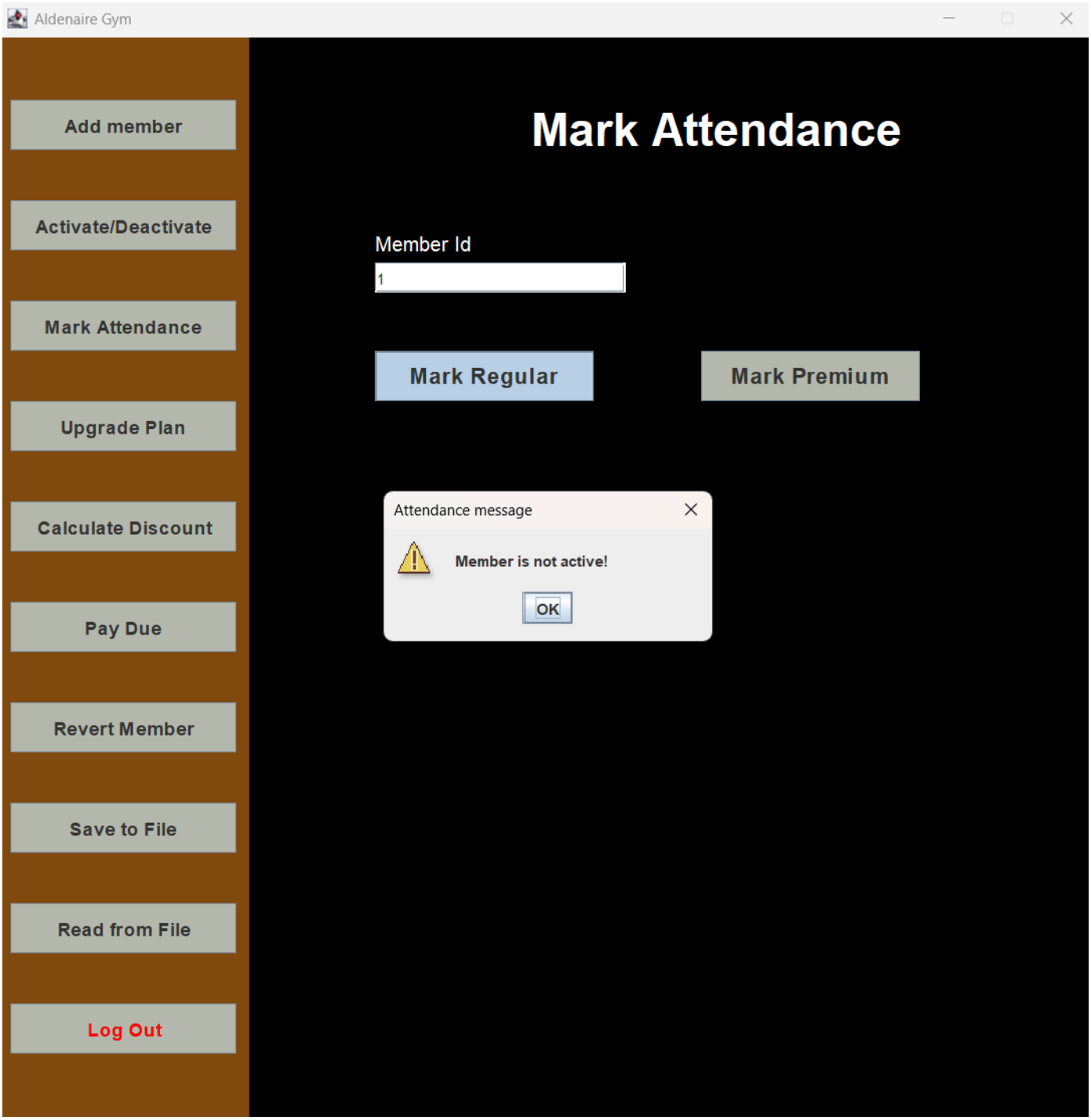


Figure 62: Regular Member Should be active to Mark Attendance

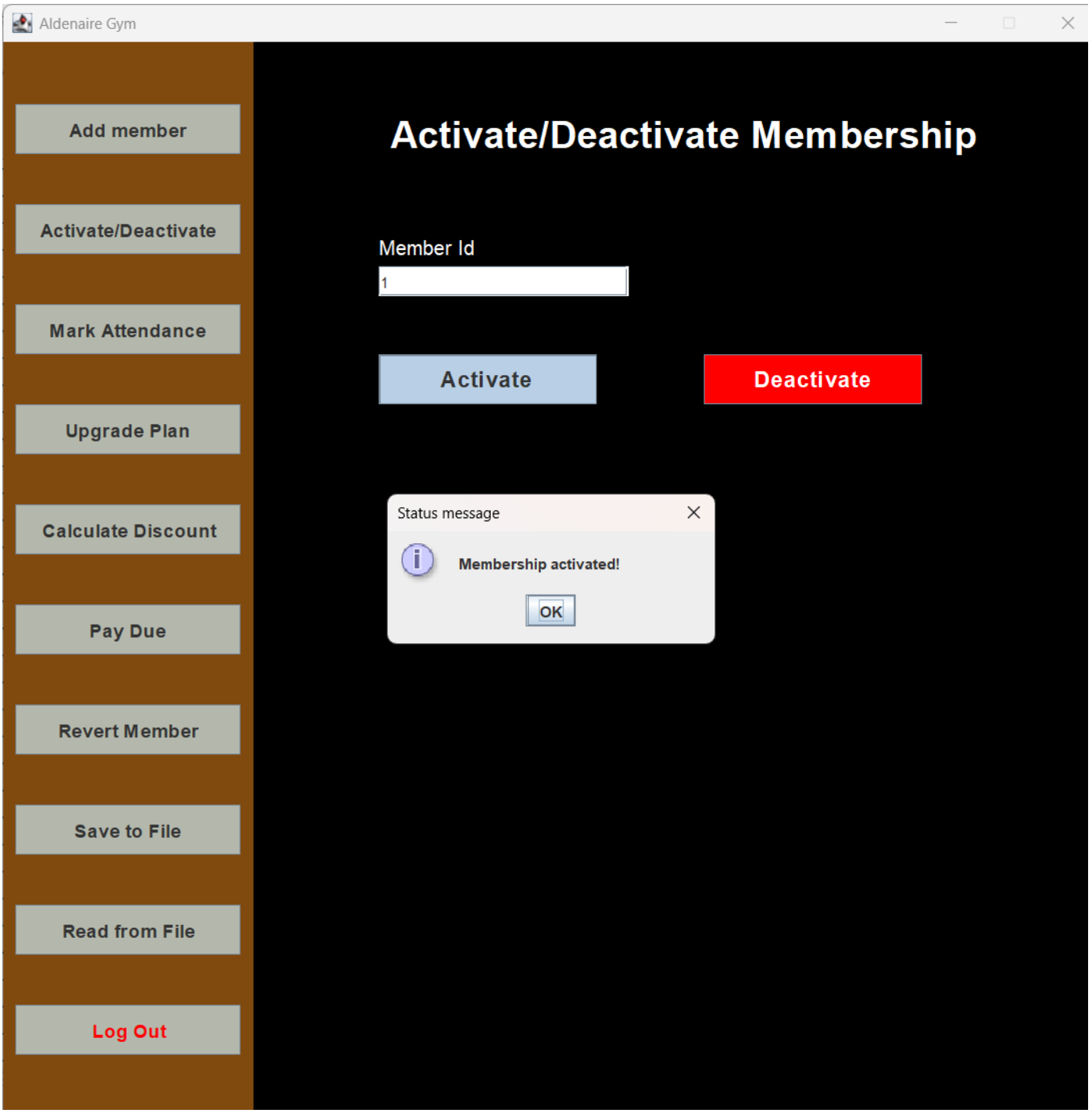


Figure 63: Regular Member successfully activated

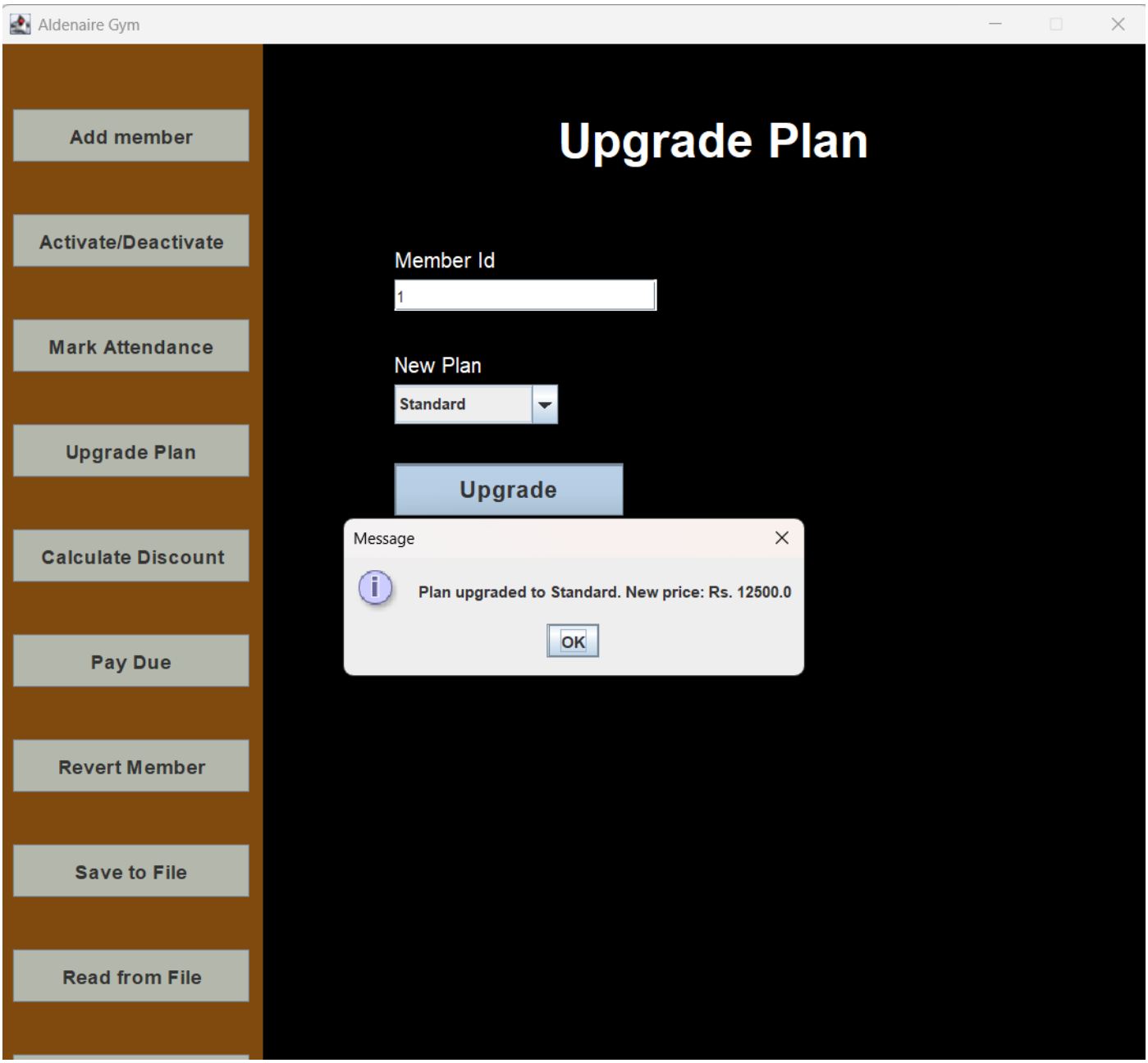


Figure 64: Upgrading Plan of Regular Member to Standard

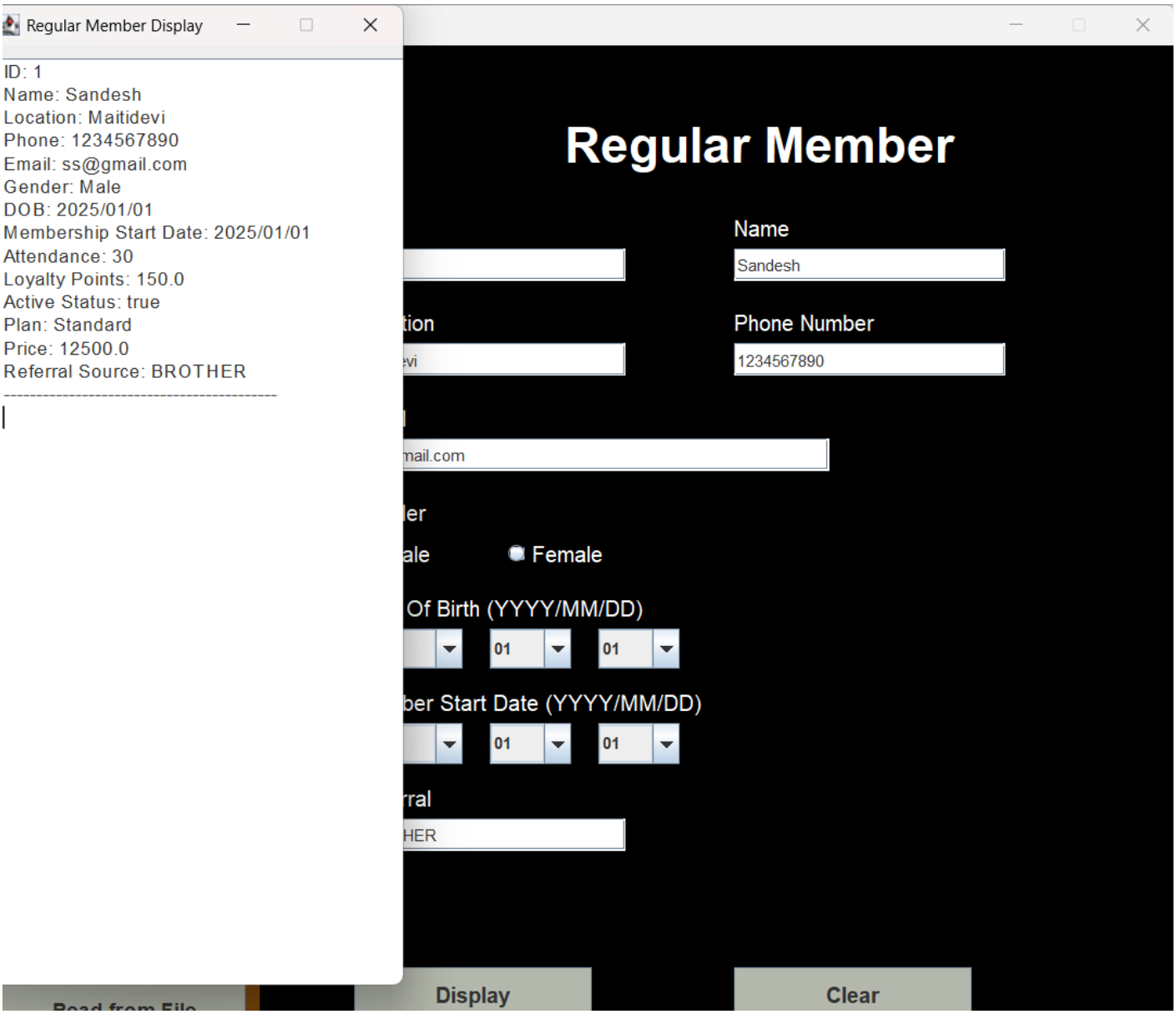


Figure 65: Display details to check Attendance and Loyalty Points

Testing 4:

Testing 4.1:

Table 5: Testing 4.1: To test case for calculate discount

Objective	To test case for calculate discount.
Action	i. Displayed the details of premium members. ii. Entered invalid member id in text field. iii. Entered correct member id in member id. iv. Entered Paid amount of 20,000. v. Entered Paid amount of 30,000. vi. Clicked Calculate discount.
Expected Output	Only Premium members should be able to get a discount if a member makes a full payment of 50,000 with an appropriate message in a dialog box.
Actual Output	Only Premium members were able to get a discount as a member makes full payment of 50,000 with an appropriate message in a dialog box.
Result	The test was successful.

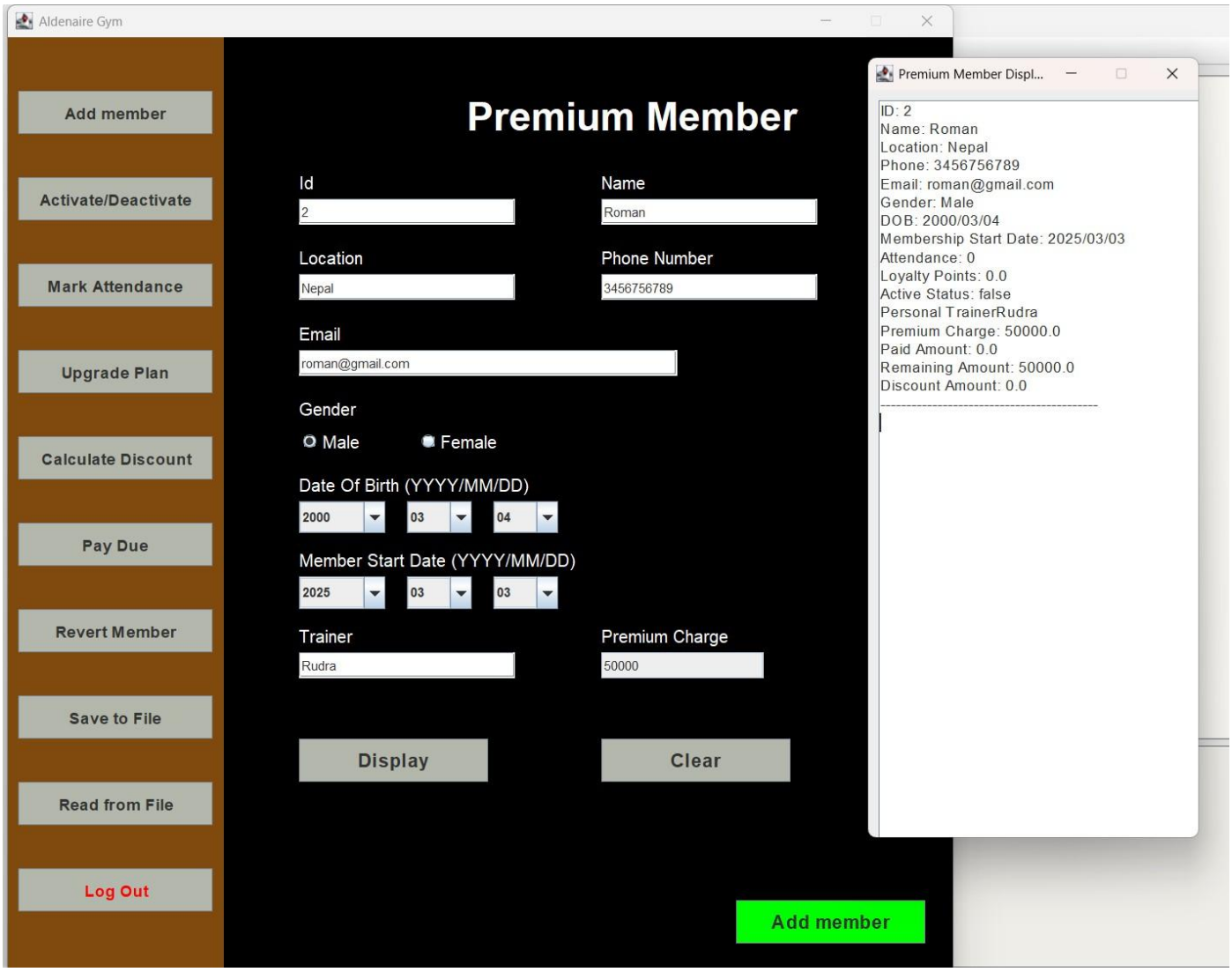


Figure 66: Display Discount amount before any Discount

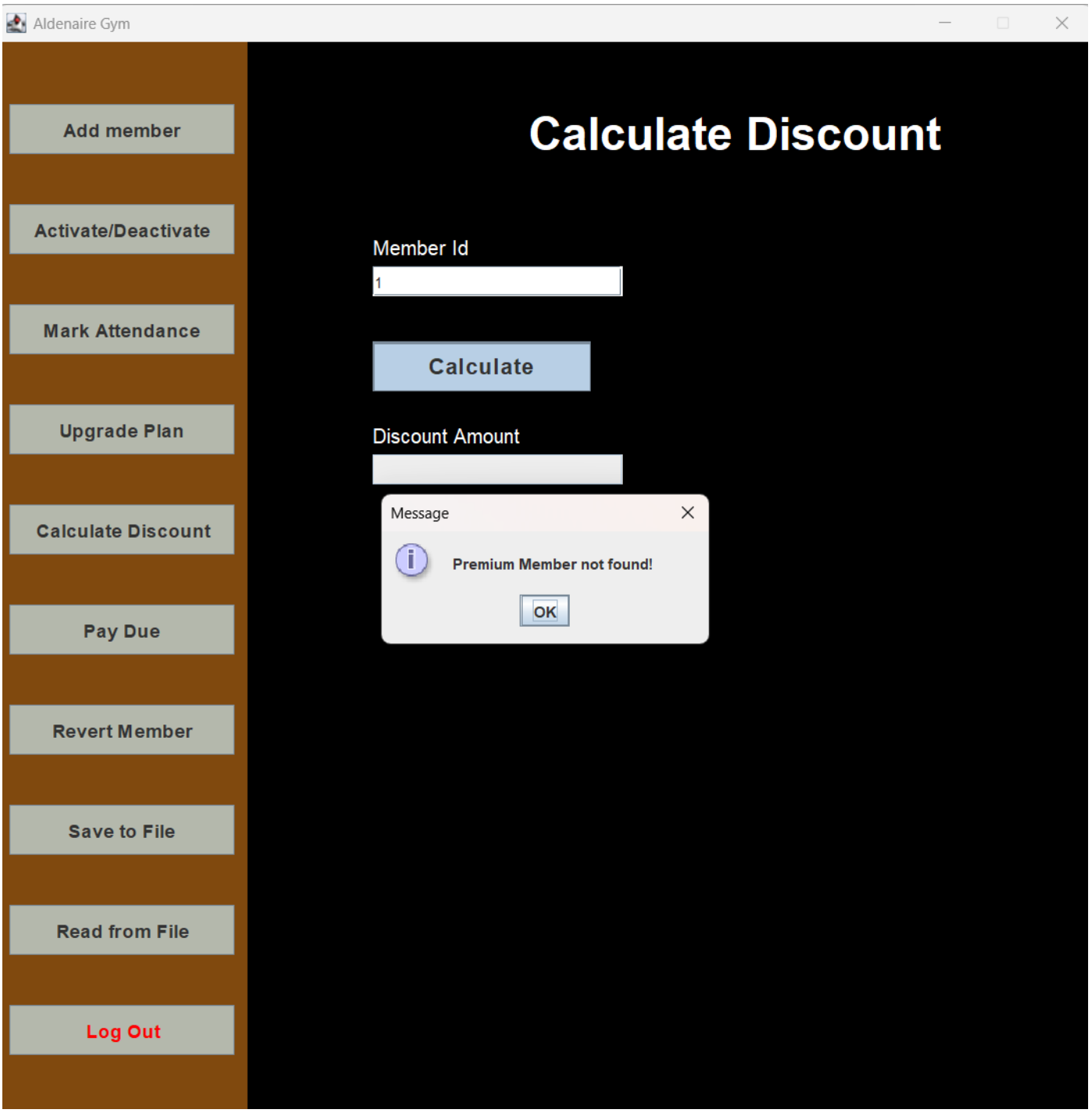


Figure 67: Discount is only available for Premium Member

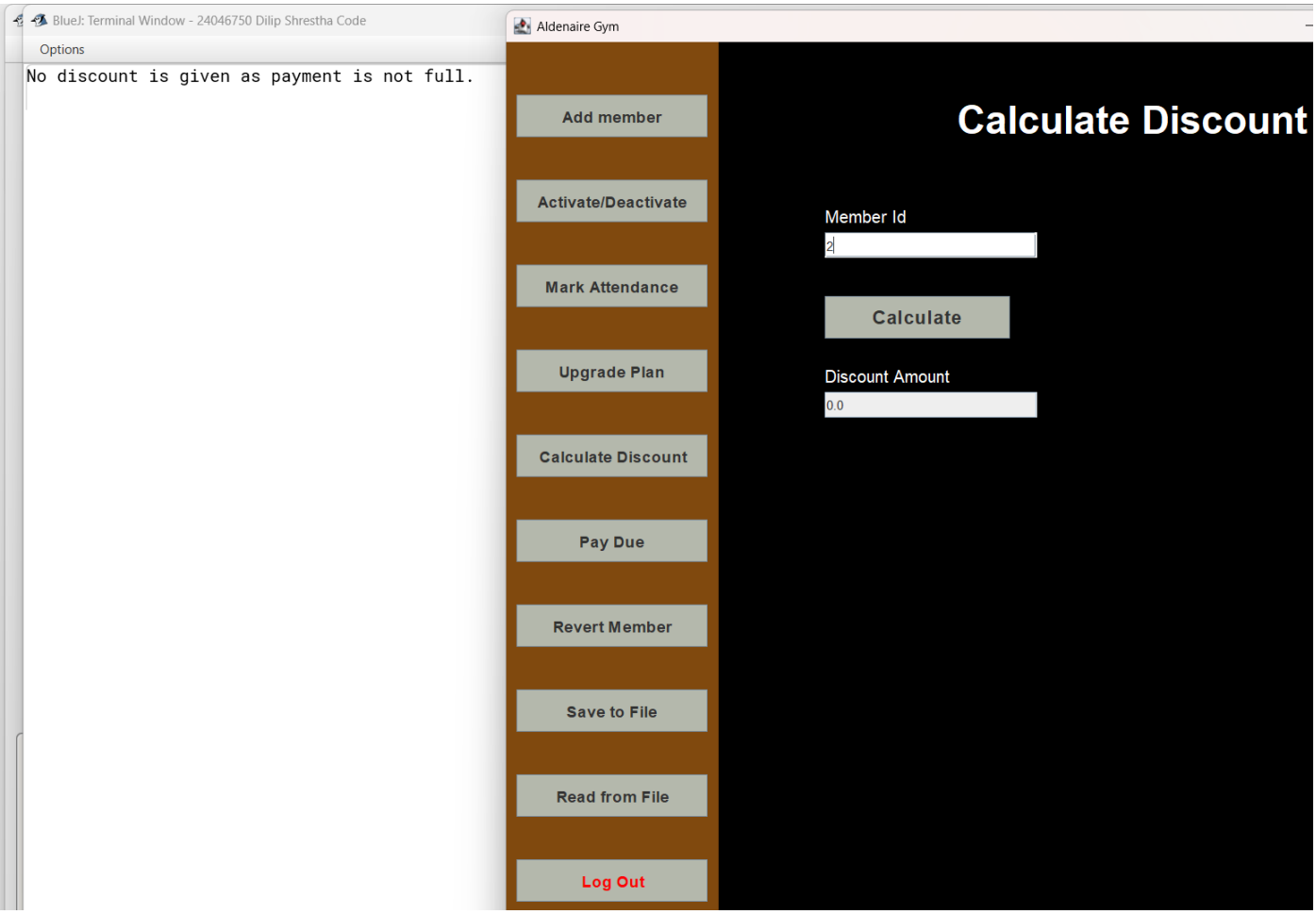


Figure 68: No discount given as members hasn't paid Full Payment

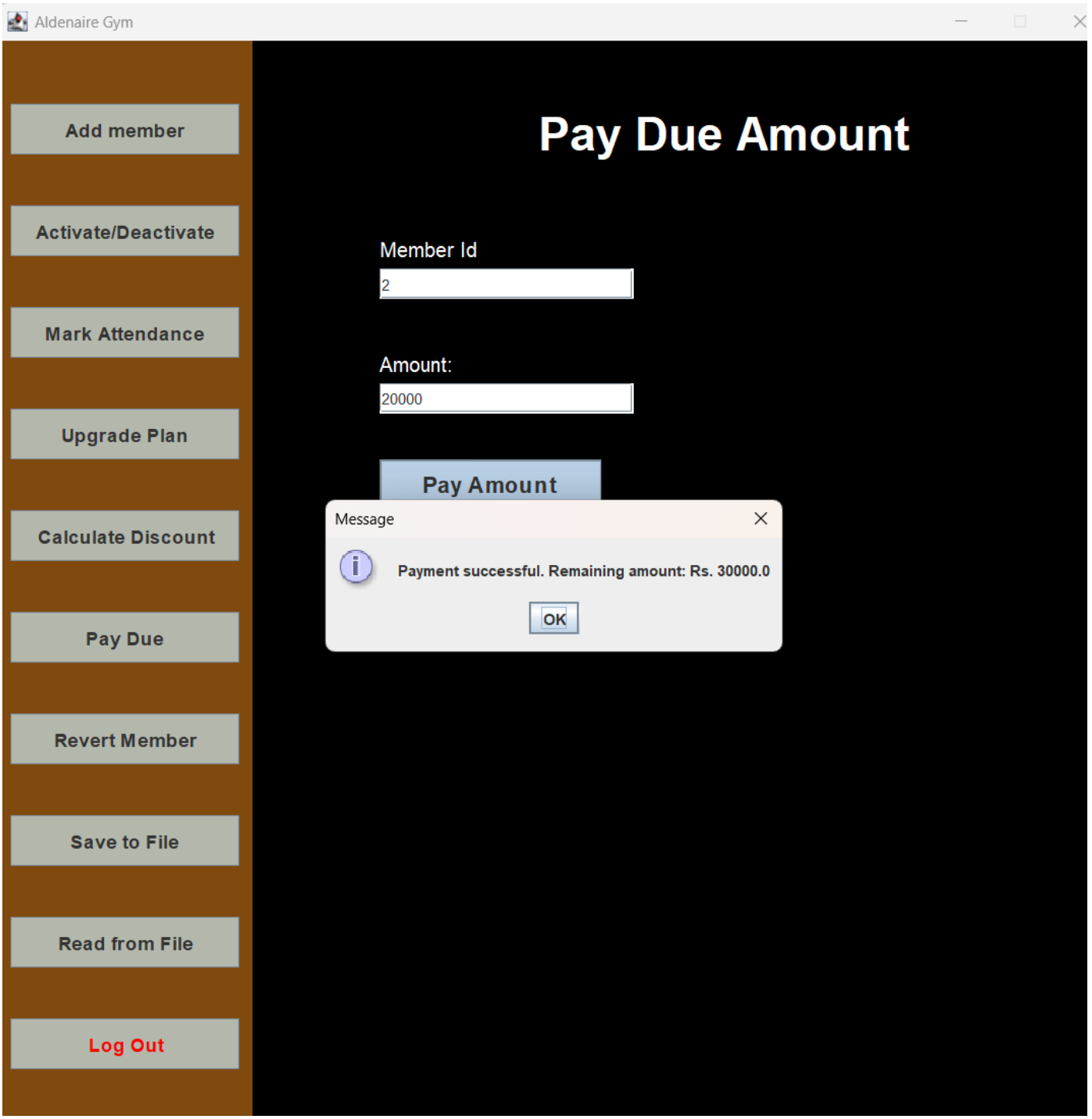


Figure 69: Paying 20,000 first and remaining amount is 30,000

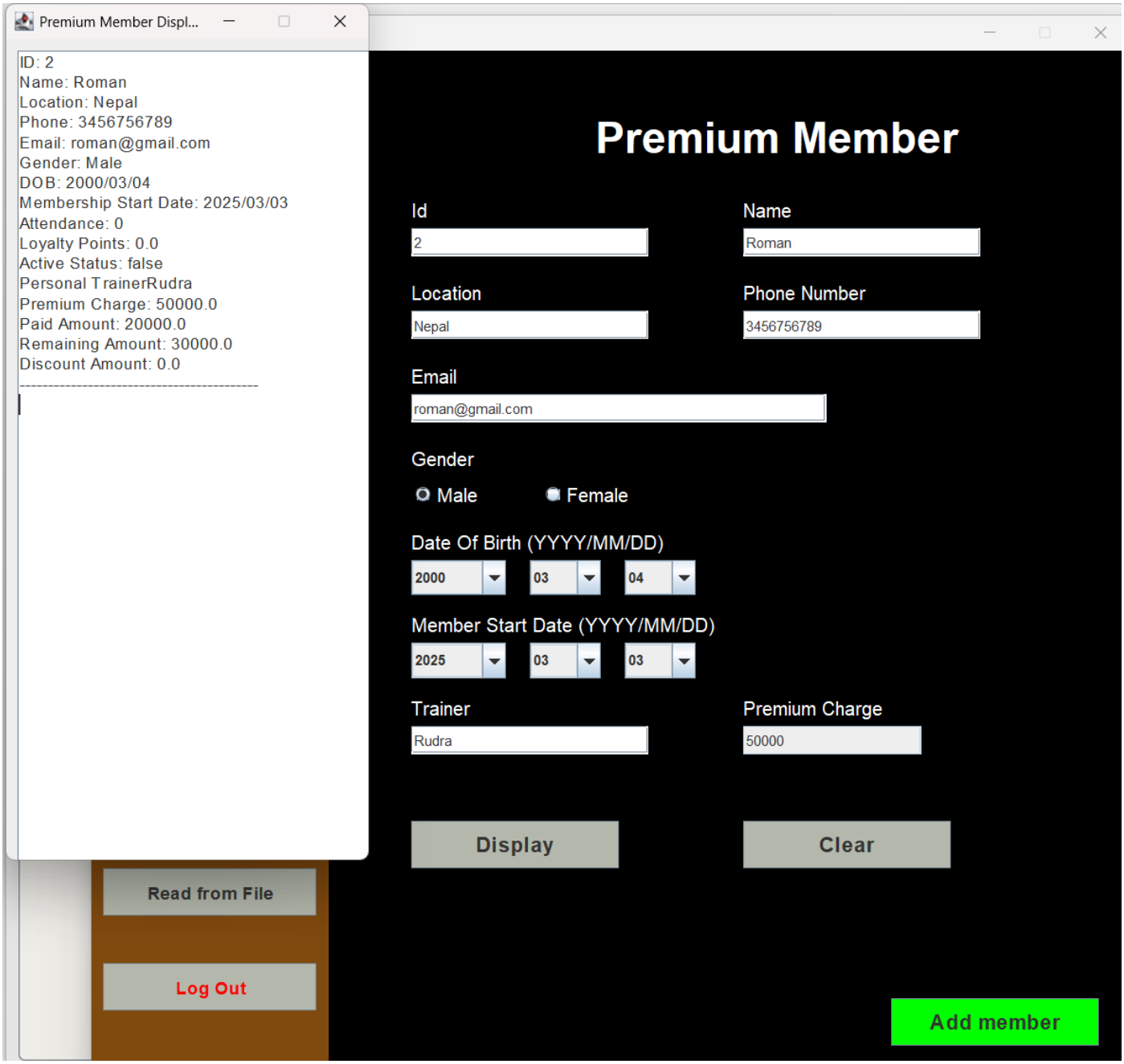


Figure 70: Display after paying 20,000

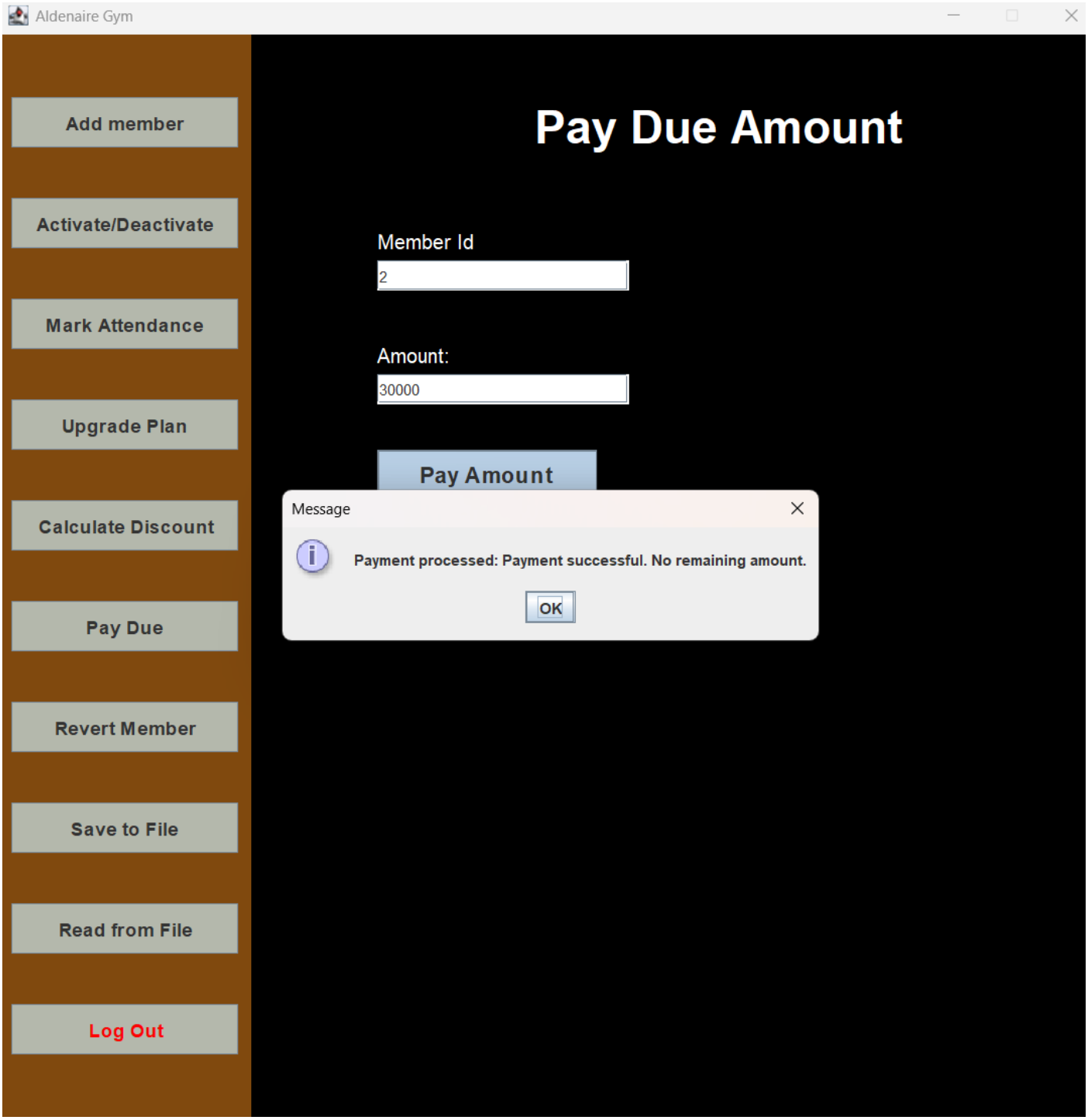


Figure 71: Paid 30,000 and Remaining amount is 0

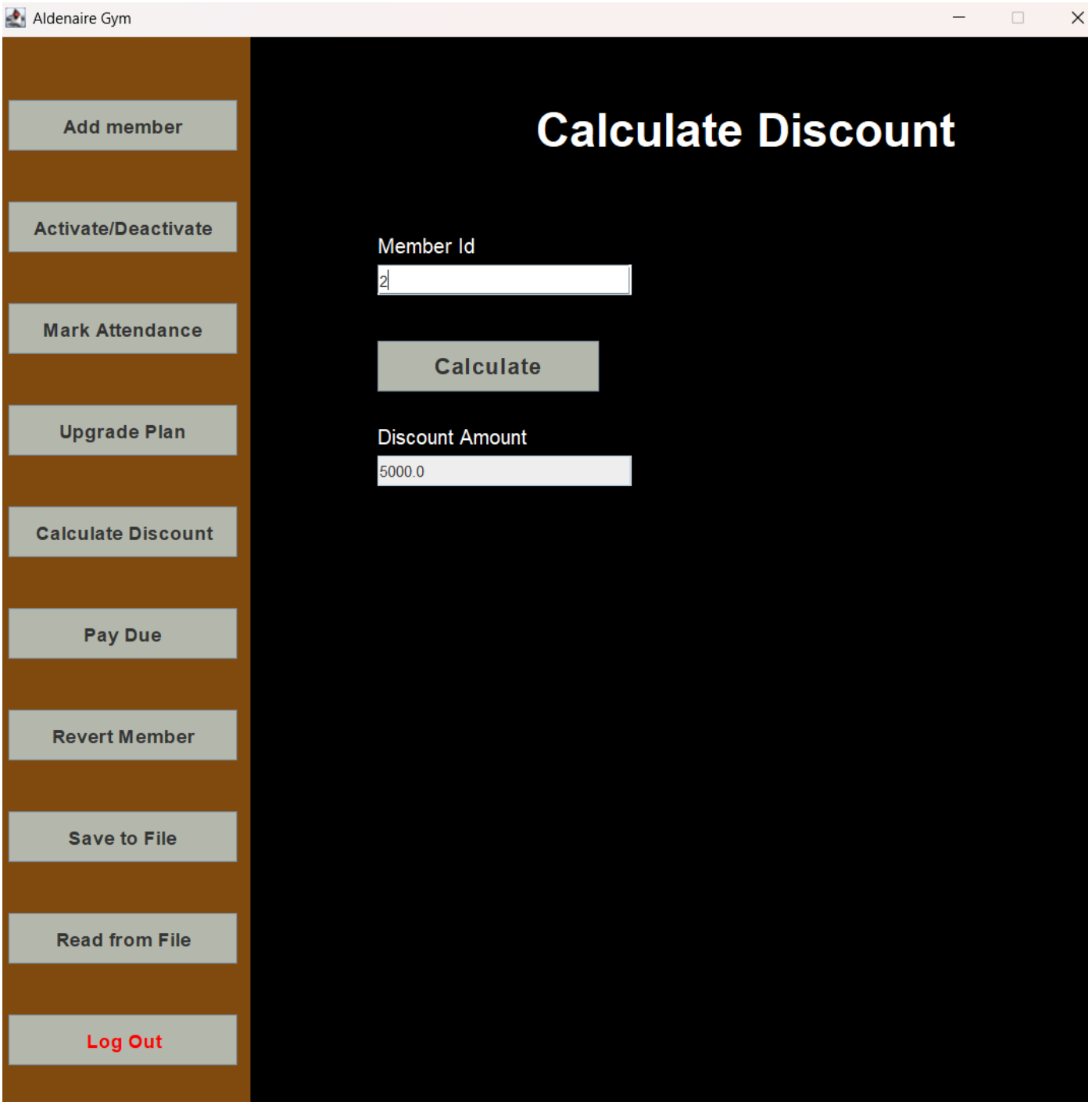


Figure 72: Calculate discount as Premium Member has done full payment

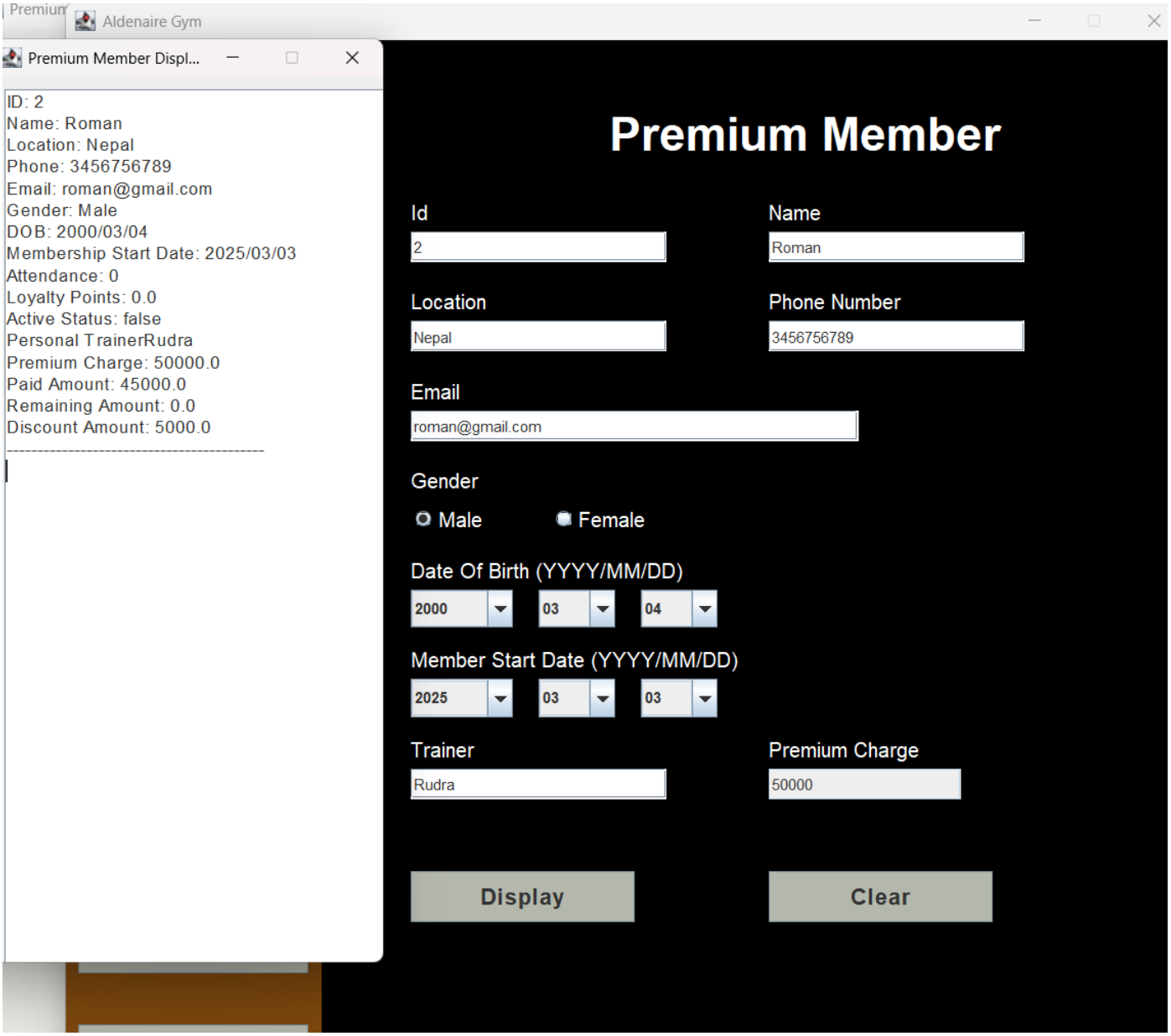


Figure 73: Displaying as 5,000 Discount is given then Paid Amount is 45,000

Testing 4.2:

Table 6: Testing 4.2: To test case for pay due amount

Objective	To test case for pay due amount.
Action	i. Entered empty id in member id. ii. Entered regular member id. iii. Entered 20,000 and paid. iv. Displayed the details v. Entered 50,000 and paid. vi. Entered 30,000 and paid. vii. Displayed the details of premium member.
Expected Output	Only Premium members should be able to get a discount if a member makes full payment of 50,000 with an appropriate message in dialog box and amount calculation.
Actual Output	Only Premium members were able to get a discount as a member makes full payment of 50,000 with an appropriate message in dialog box and amount calculation.
Result	The test was successful.

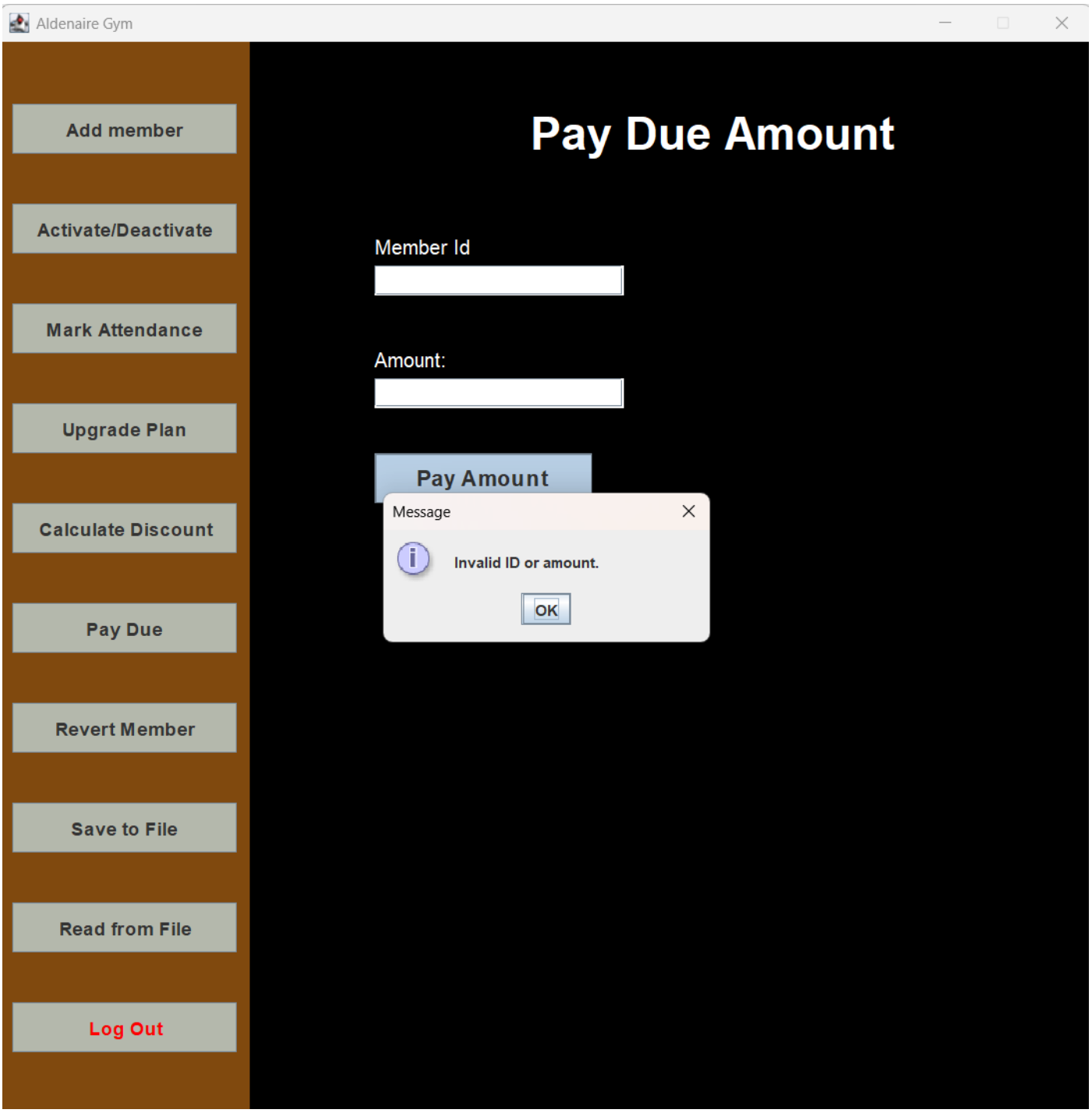


Figure 74: Passing Empty id for Pay Due Amount

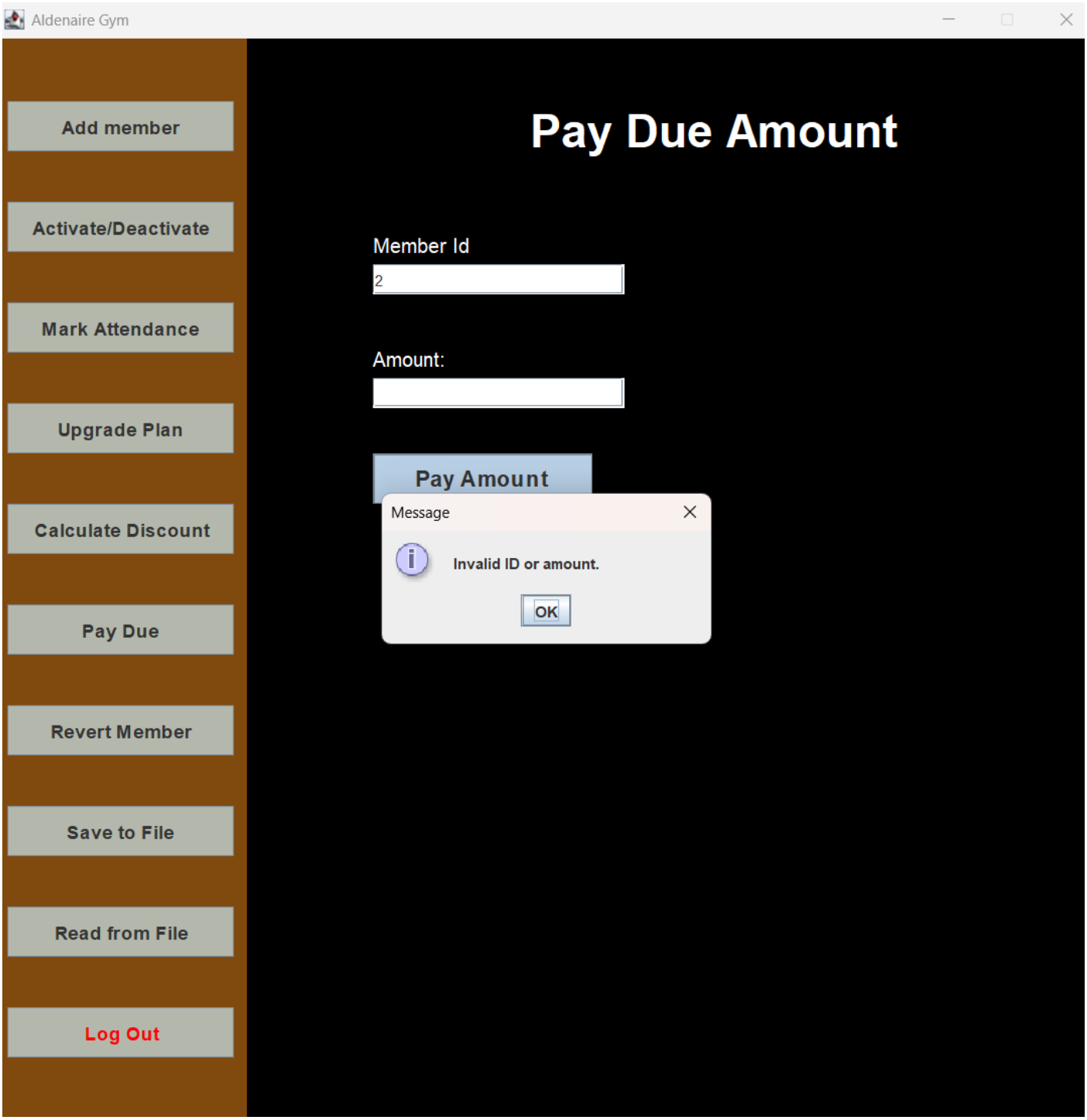


Figure 75: Passing Non-existed id in Pay Due Amount

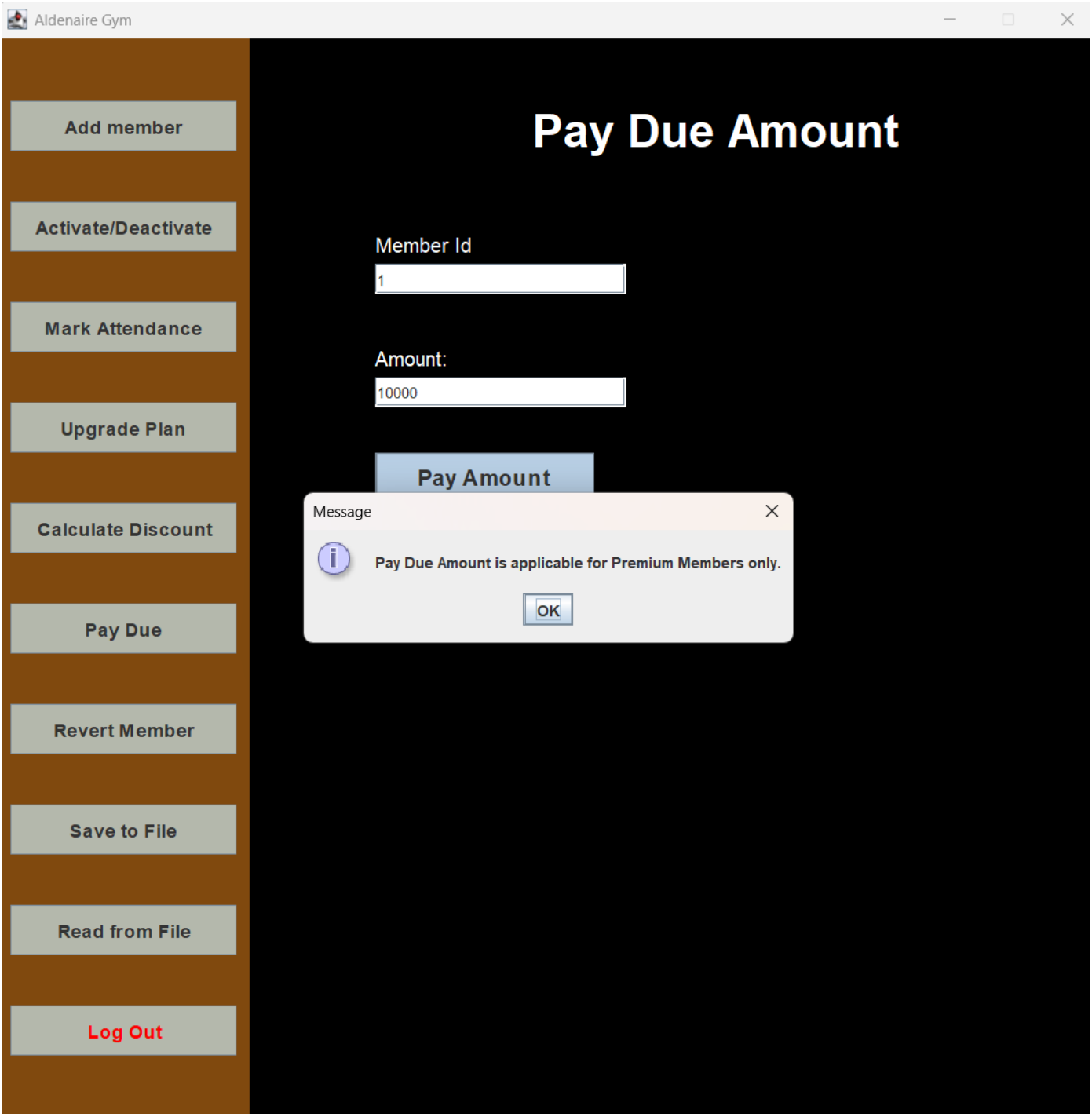


Figure 76: Pay Amount is not applicable for Regular Member

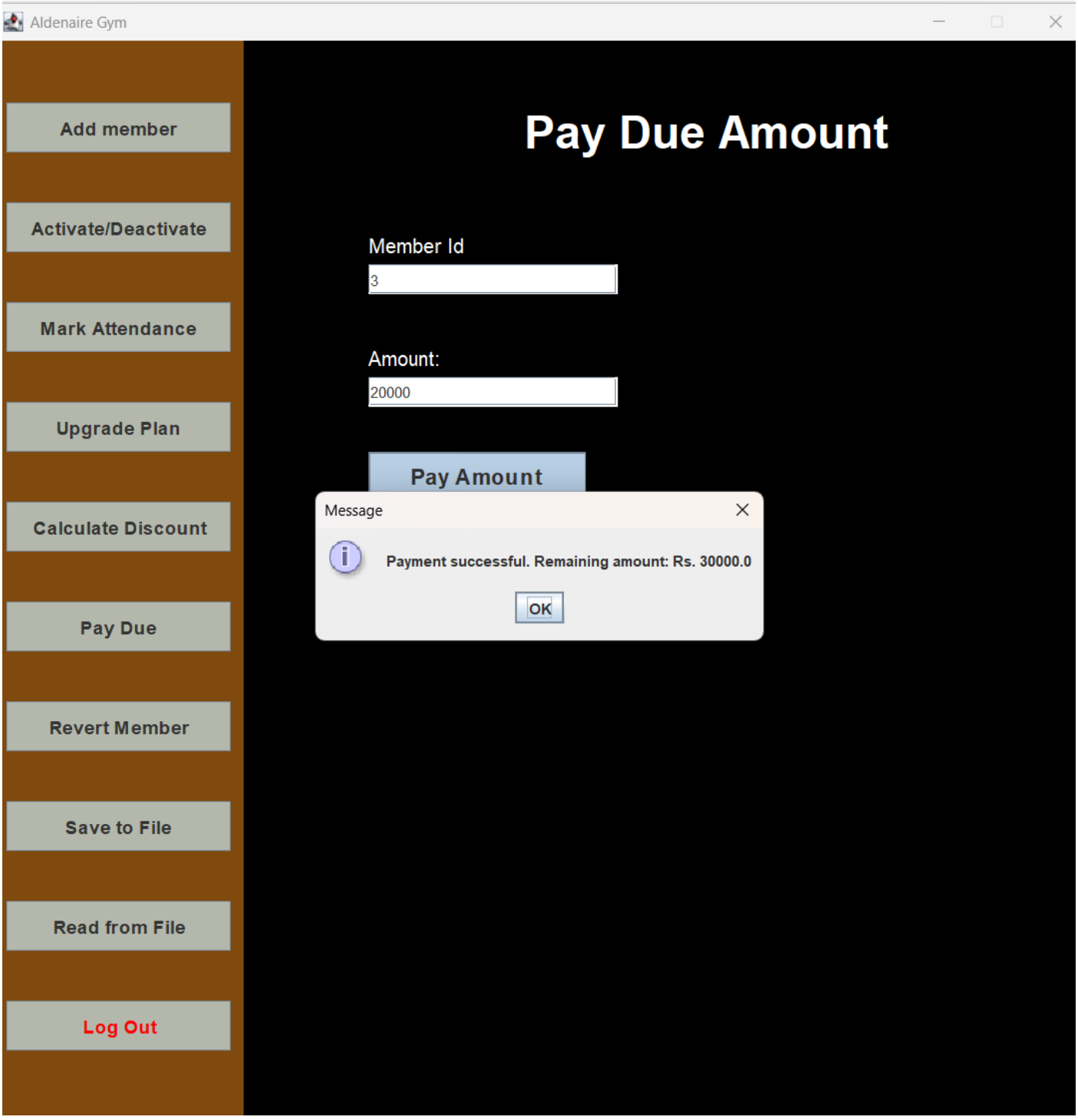


Figure 77: Paid 20,000 then Remaining Amount is 30,000

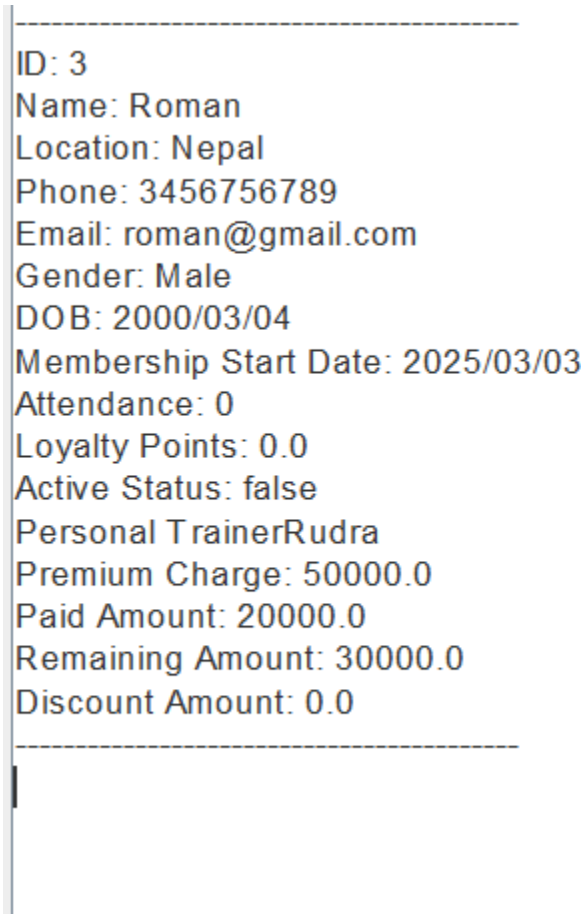


Figure 78: Display Pay Due Amount after paying 20,000

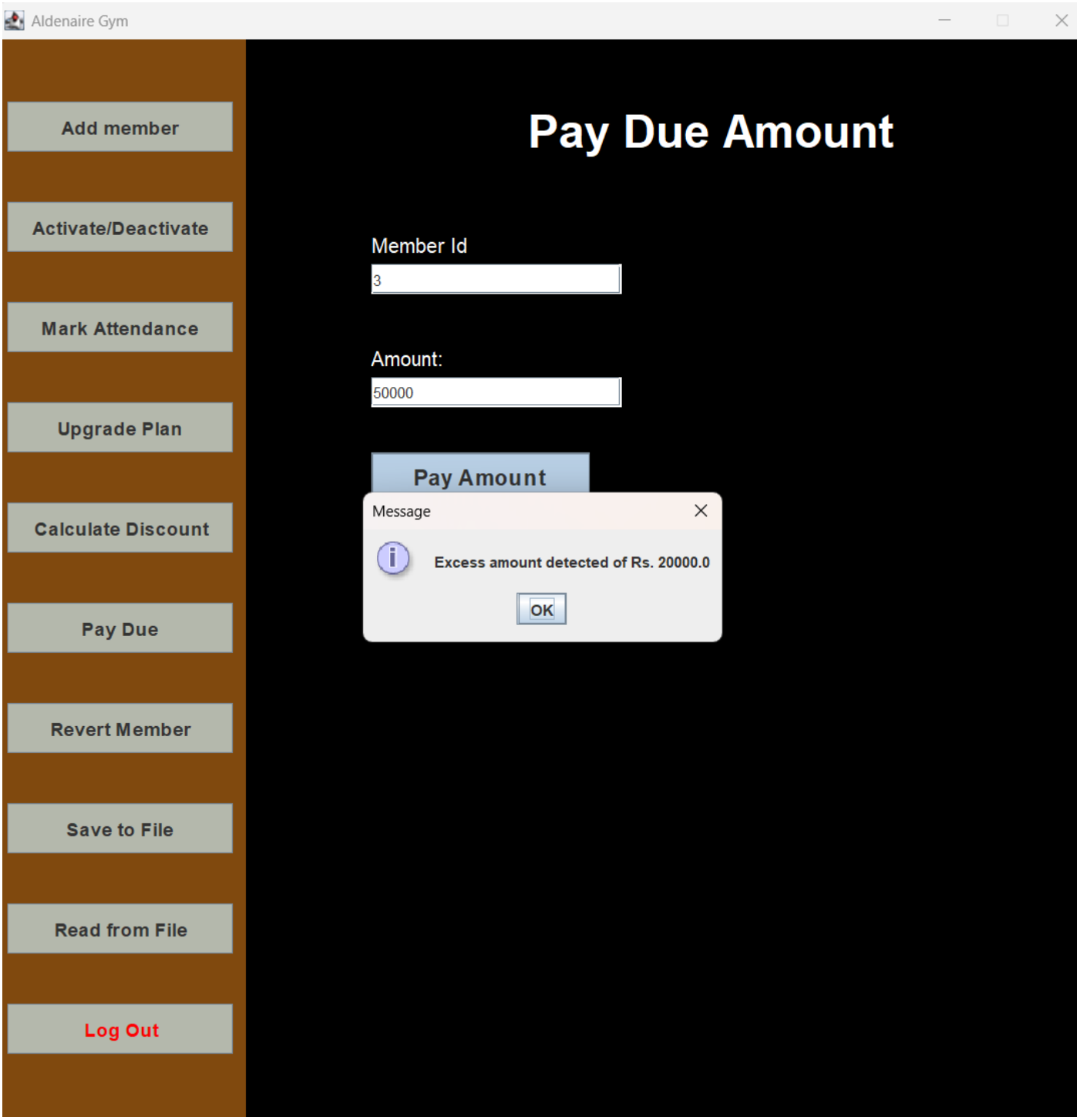


Figure 79: Paying excess amount

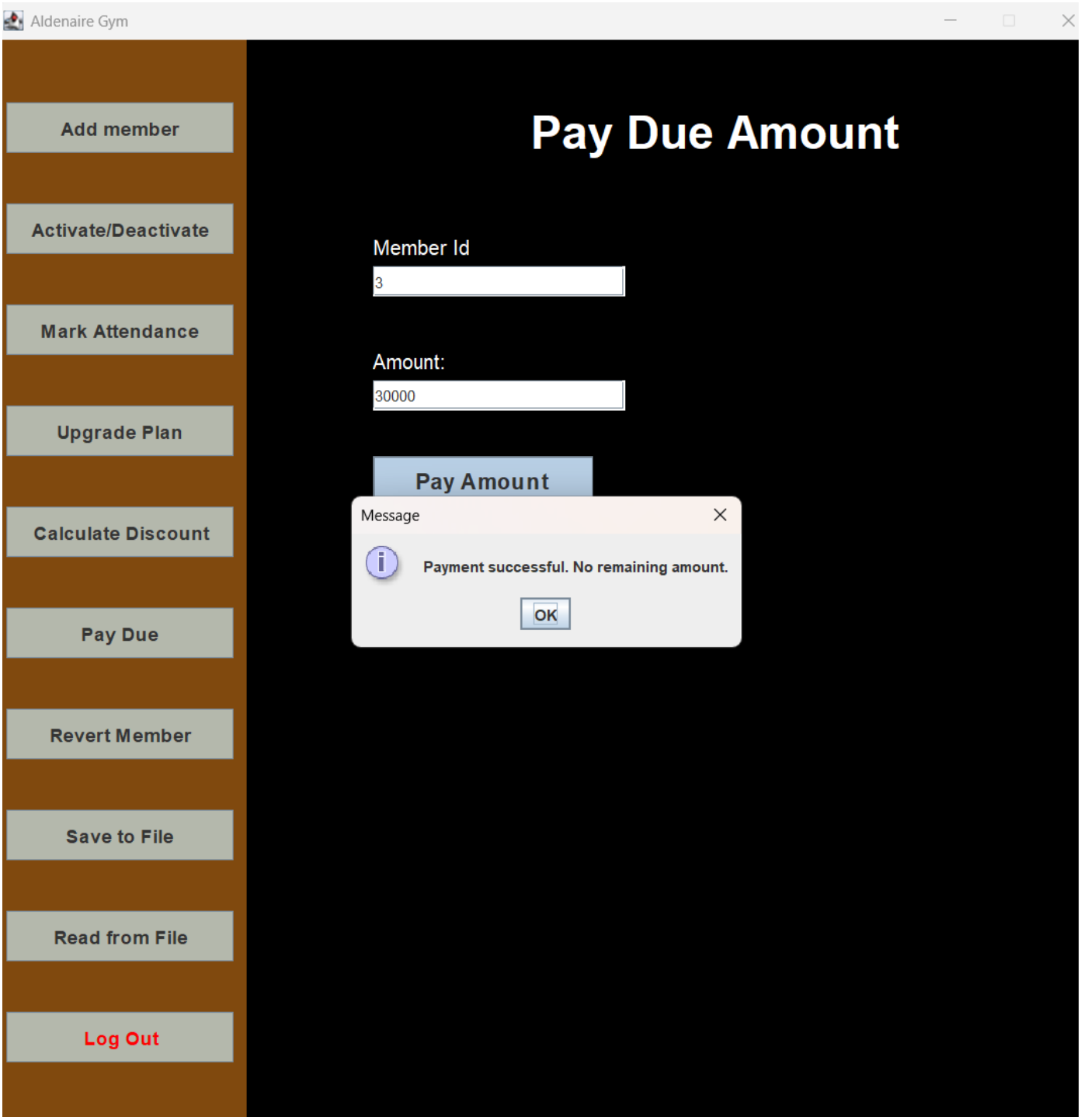


Figure 80: Paying 30,000 and Remaining Amount is 0

```
----------------------------------------
ID: 3
Name: Roman
Location: Nepal
Phone: 3456756789
Email: roman@gmail.com
Gender: Male
DOB: 2000/03/04
Membership Start Date: 2025/03/03
Attendance: 0
Loyalty Points: 0.0
Active Status: false
Personal TrainerRudra
Premium Charge: 50000.0
Paid Amount: 50000.0
Remaining Amount: 0.0
Discount Amount: 0.0
----------------------------------------
```

Figure 81: Display Pay Due Amount after Full Payment

Testing 4.3

Table 7: Testing 4.3: To test case for revert member

Objective	To test case for revert member.
Action	For Regular Members. i. Displayed the Regular member info before reverting ii. Entered empty id in the member id text field. iii. Entered wrong member id in the member id text field. iv. Entered character in the text field. v. Entered the correct member id and clicked Revert Regular. vi. Wrote removal reason. vii. Displayed the Regular member info after reverting. For Premium Members. i. Displayed the Premium member info before reverting ii. Entered empty id in the member id text field. iii. Entered wrong member id in the member id text field. iv. Entered the correct member id and clicked Revert Regular. v. Displayed the Regular member info after reverting.

Expected Output	For regular members while reverting removal reason should be asked, and member should be reverted. As per premium member no removal reason should be asked and should be reverted.
Actual Output	For regular members while reverting removal reason was asked, and member was reverted. As per premium member no removal reason was asked and was reverted.
Result	The test was successful.

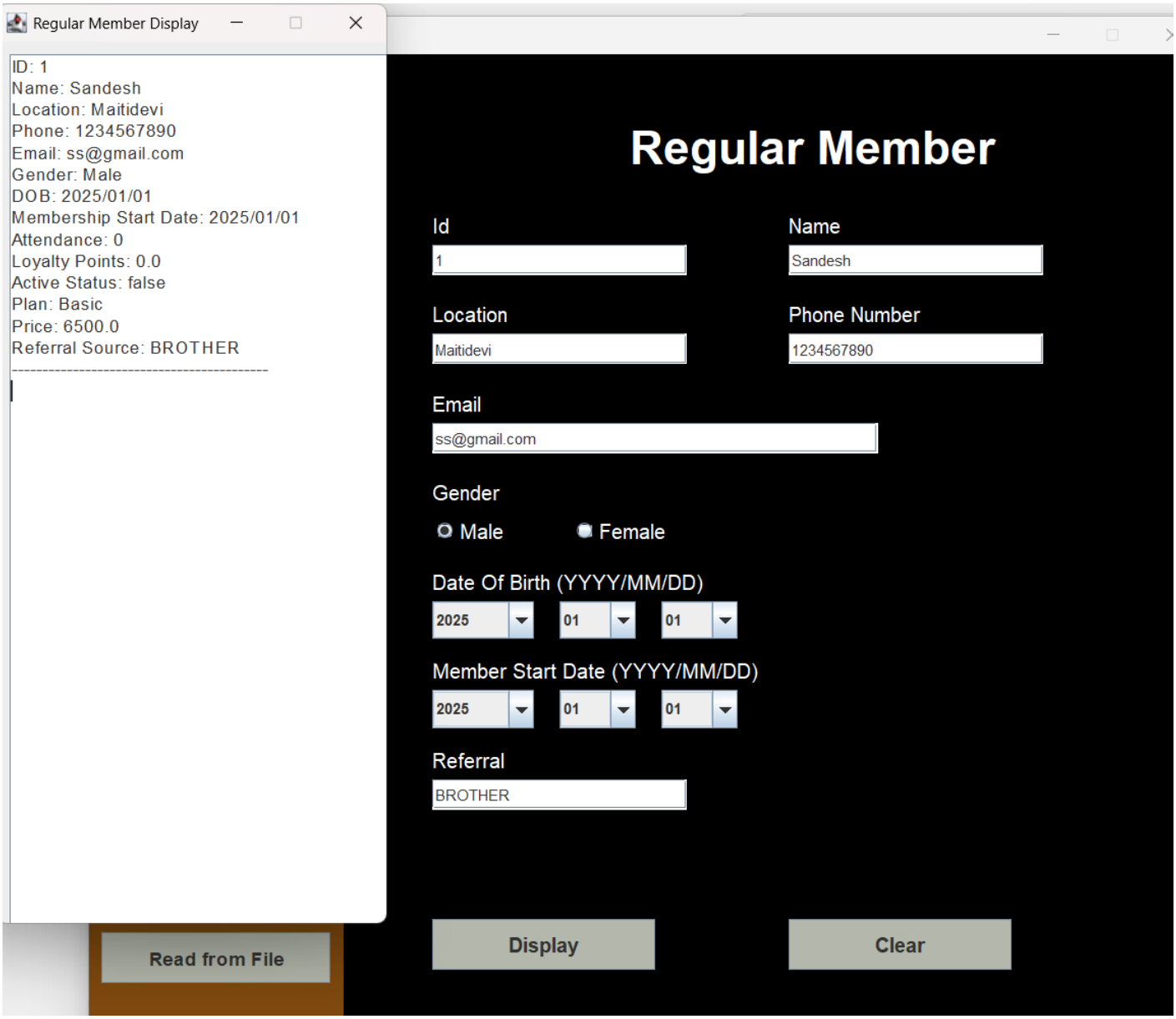


Figure 82: Display details before Revert Member

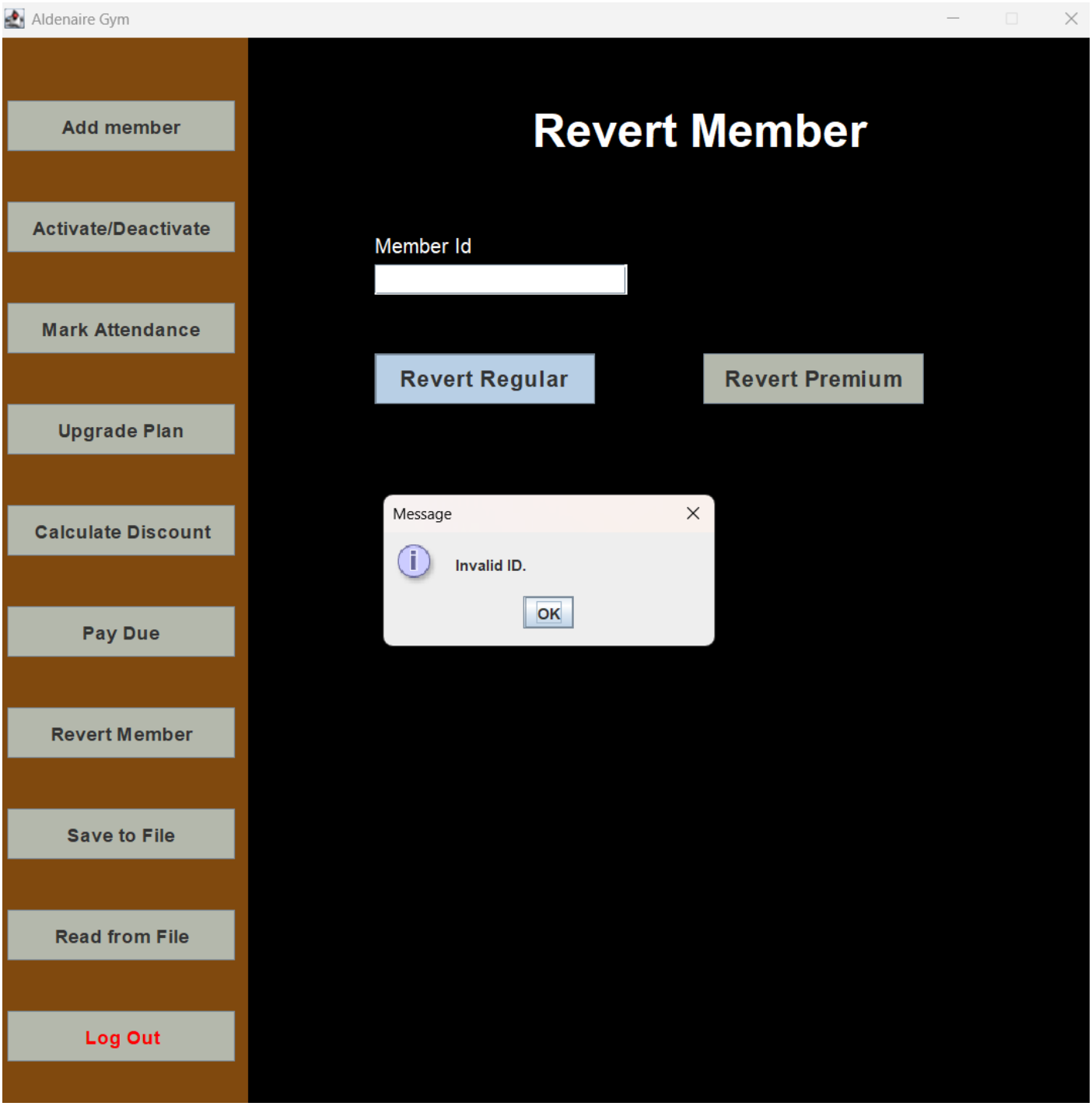


Figure 83: Passing Empty id in Revert Member

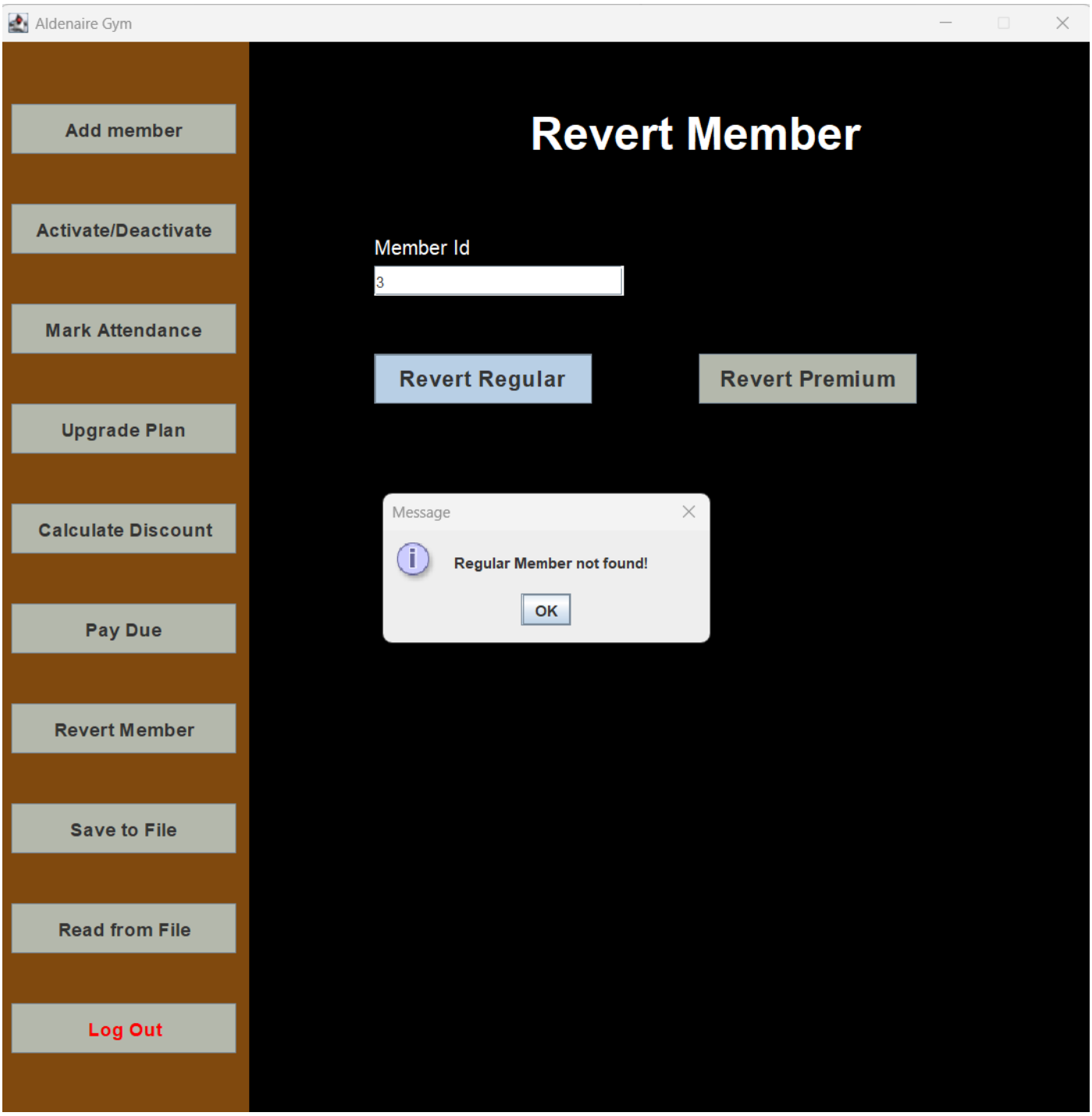


Figure 84: Passing Non-exited id in Revert Regular

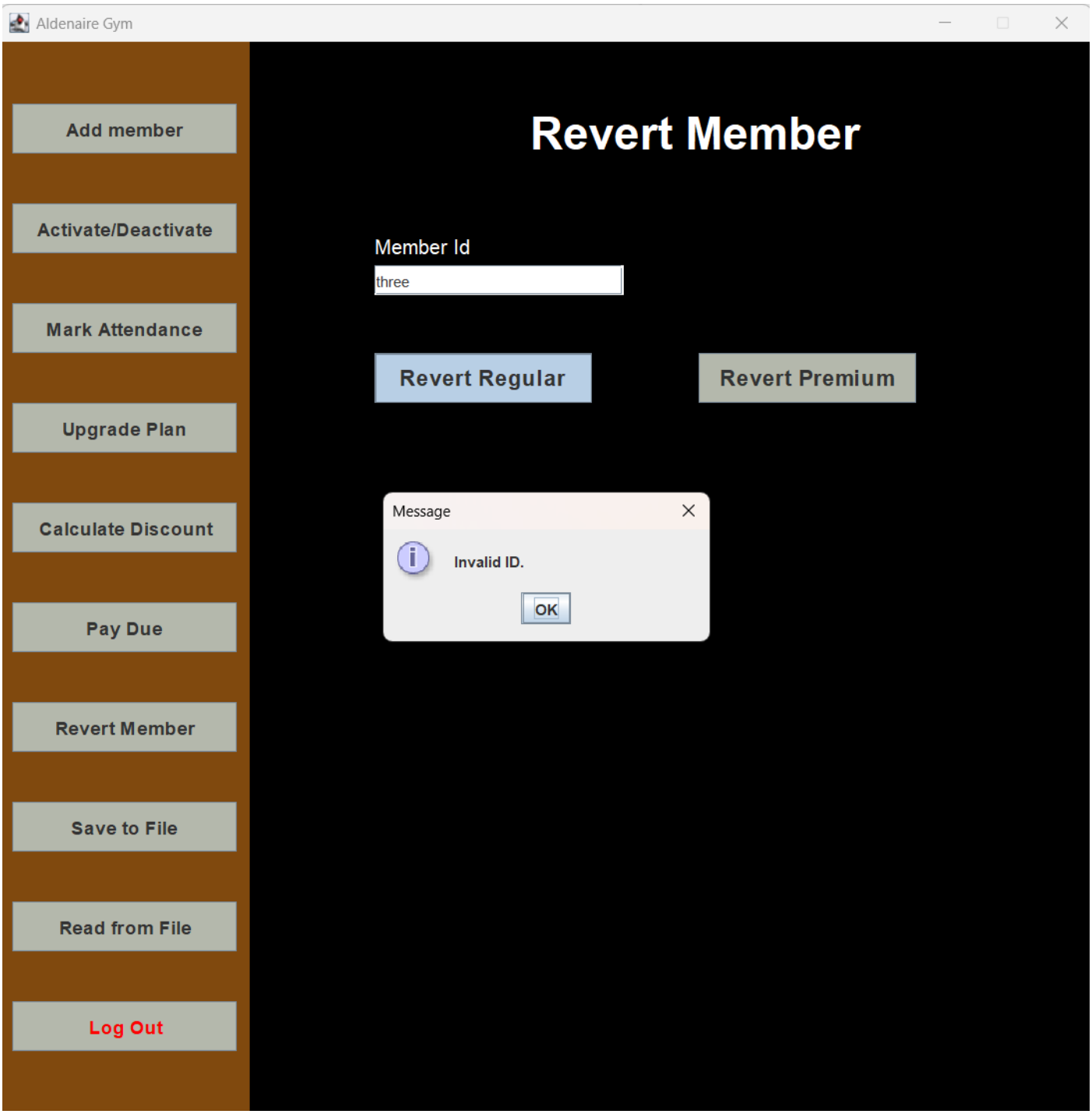


Figure 85: Passing Character in id of Revert Member

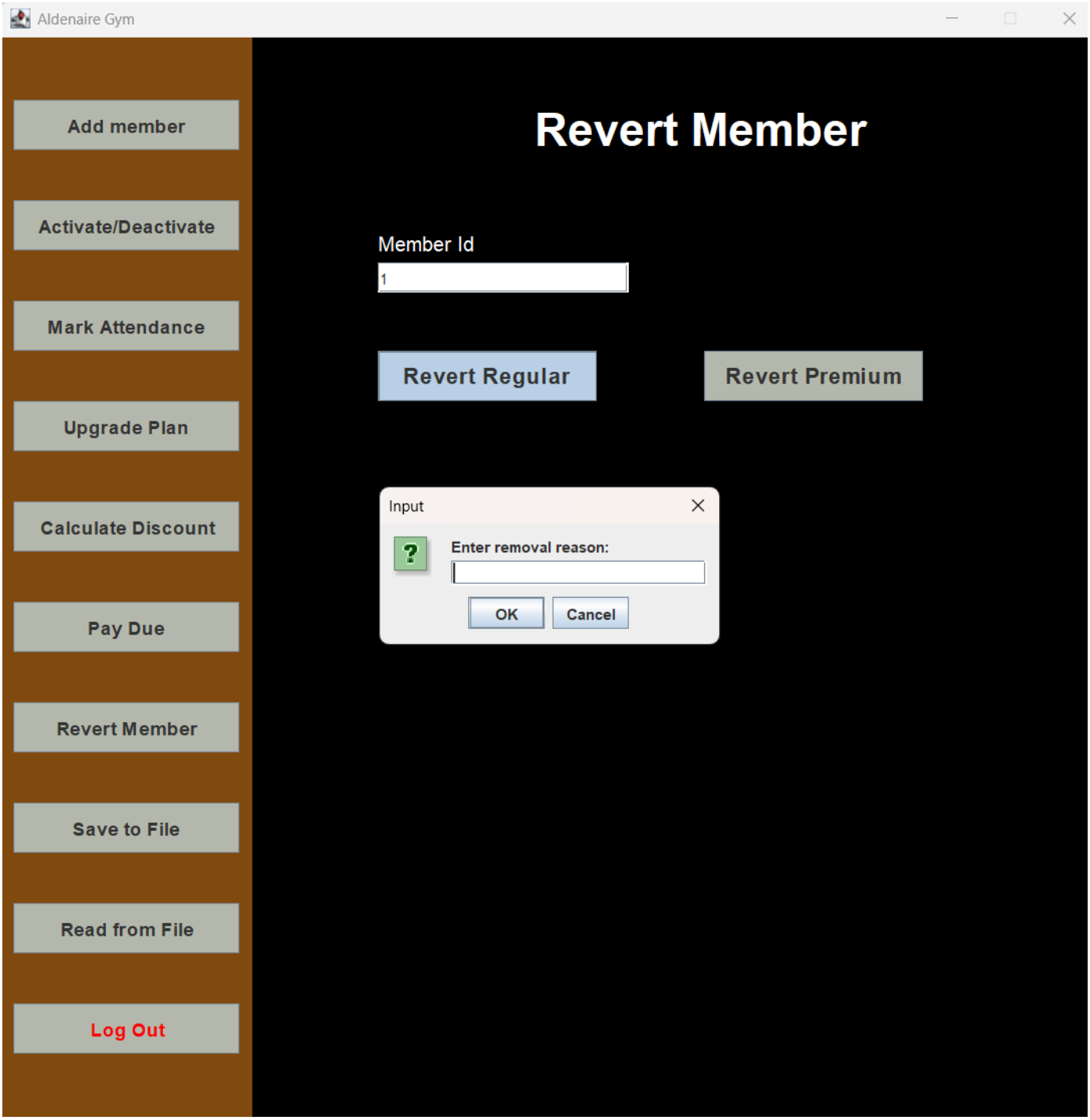


Figure 86: Successful id and ask for Removal Reason in Revert Regular

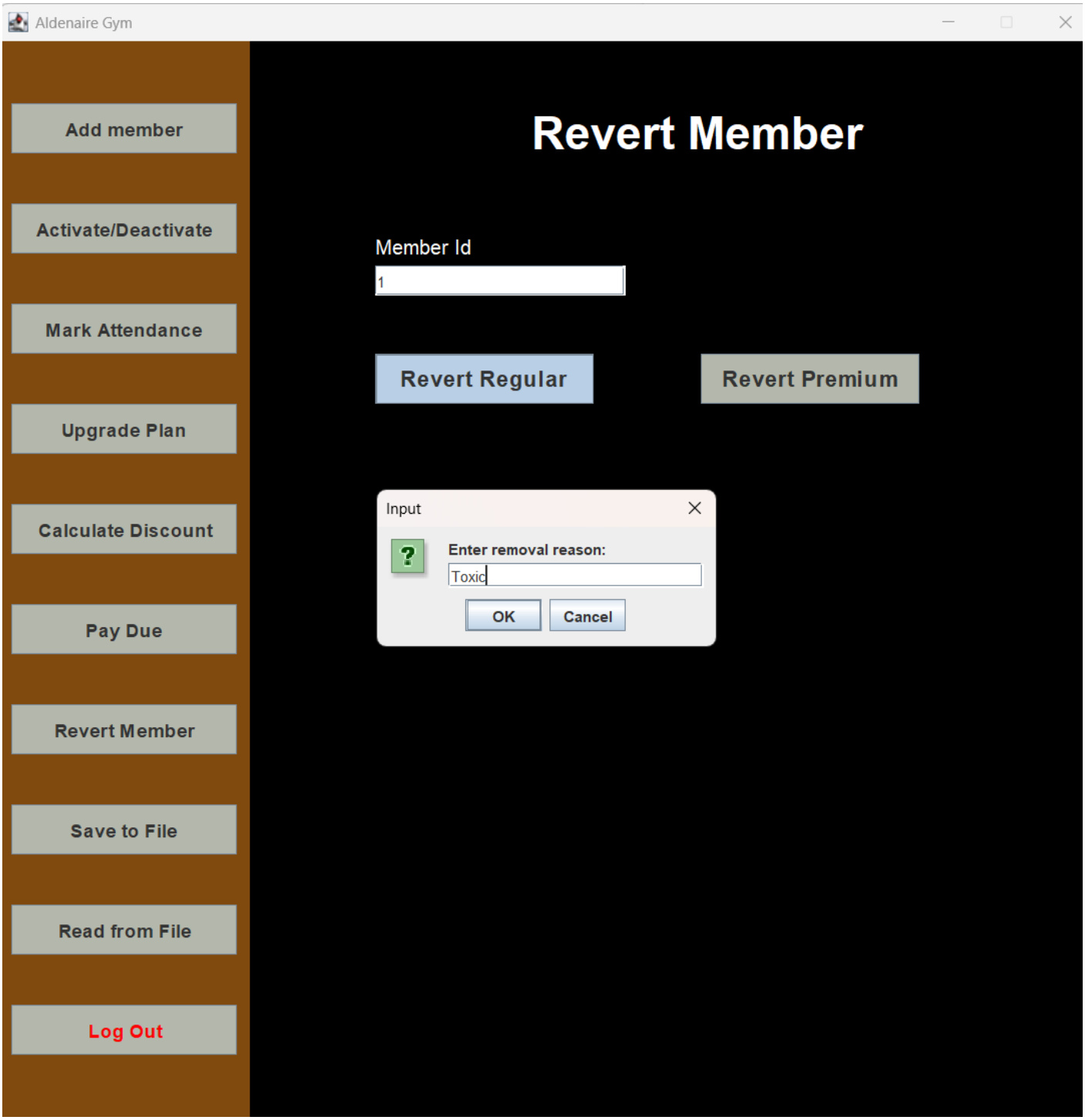


Figure 87: Writing the Removal Reason

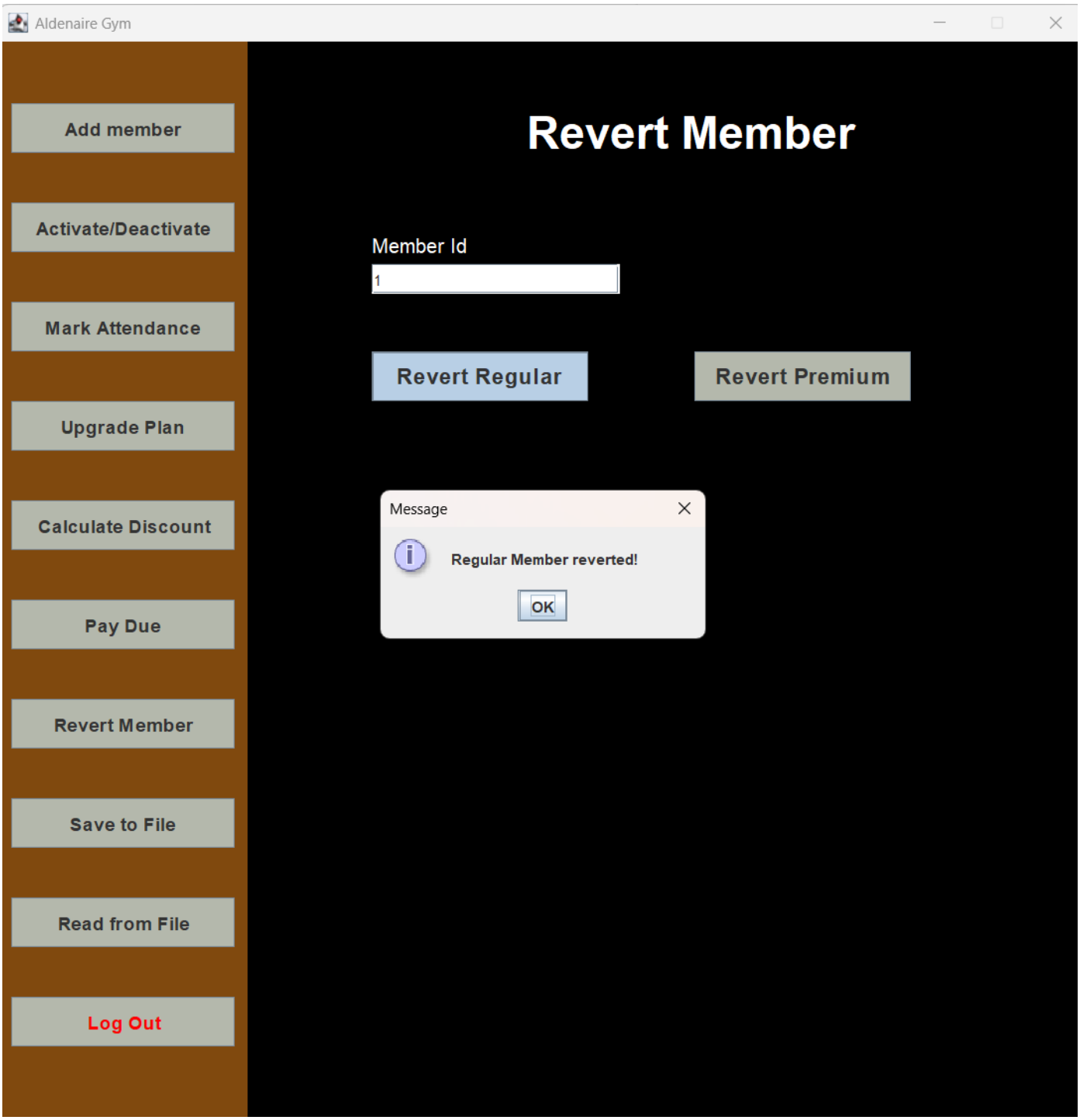


Figure 88: Regular Member Reverted Successfully

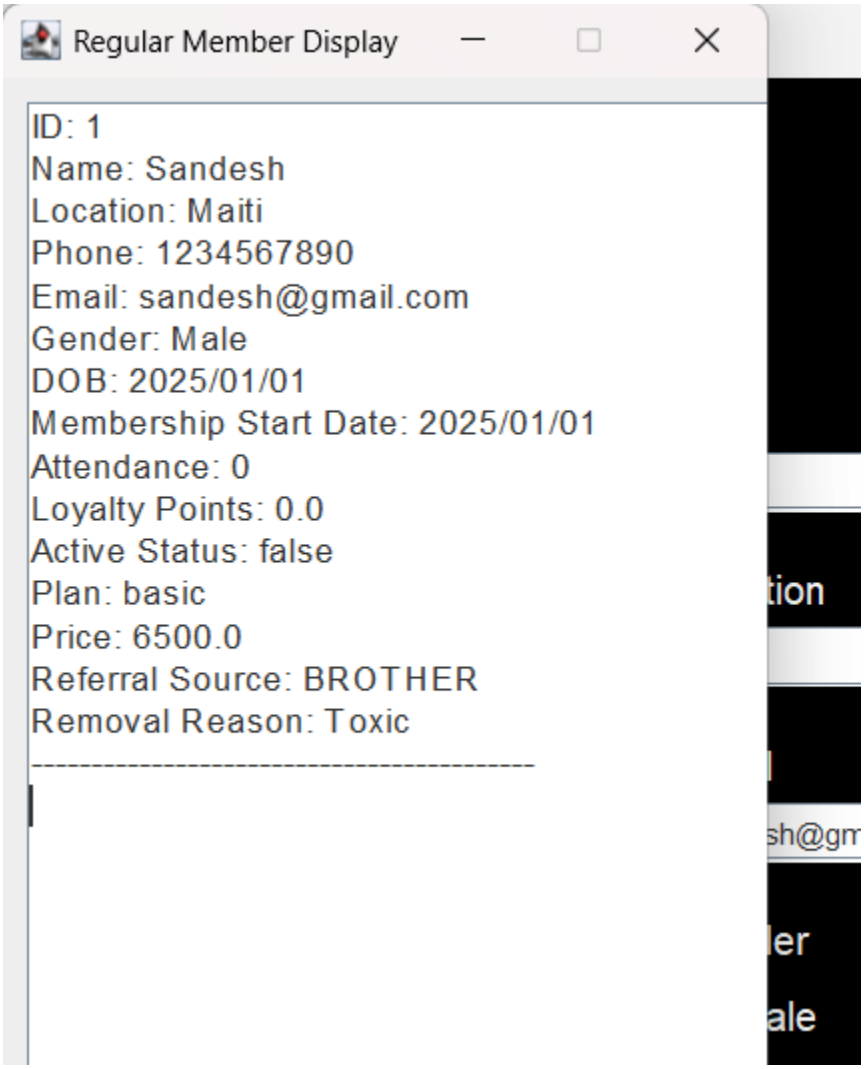


Figure 89: Regular DIsplay after Reverting Member

Revert for Premium Member

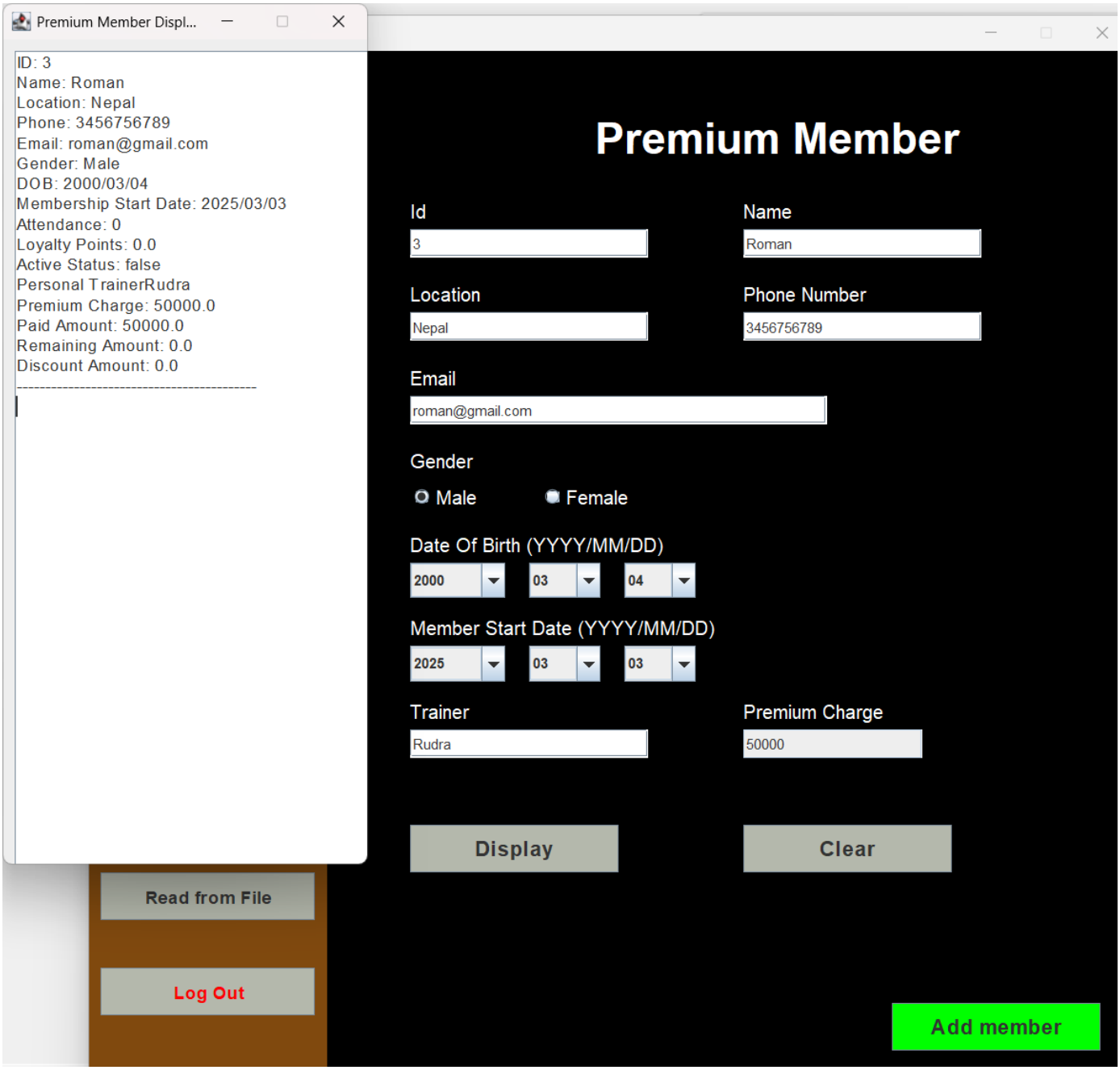


Figure 90: Display info before reverting Premium Member

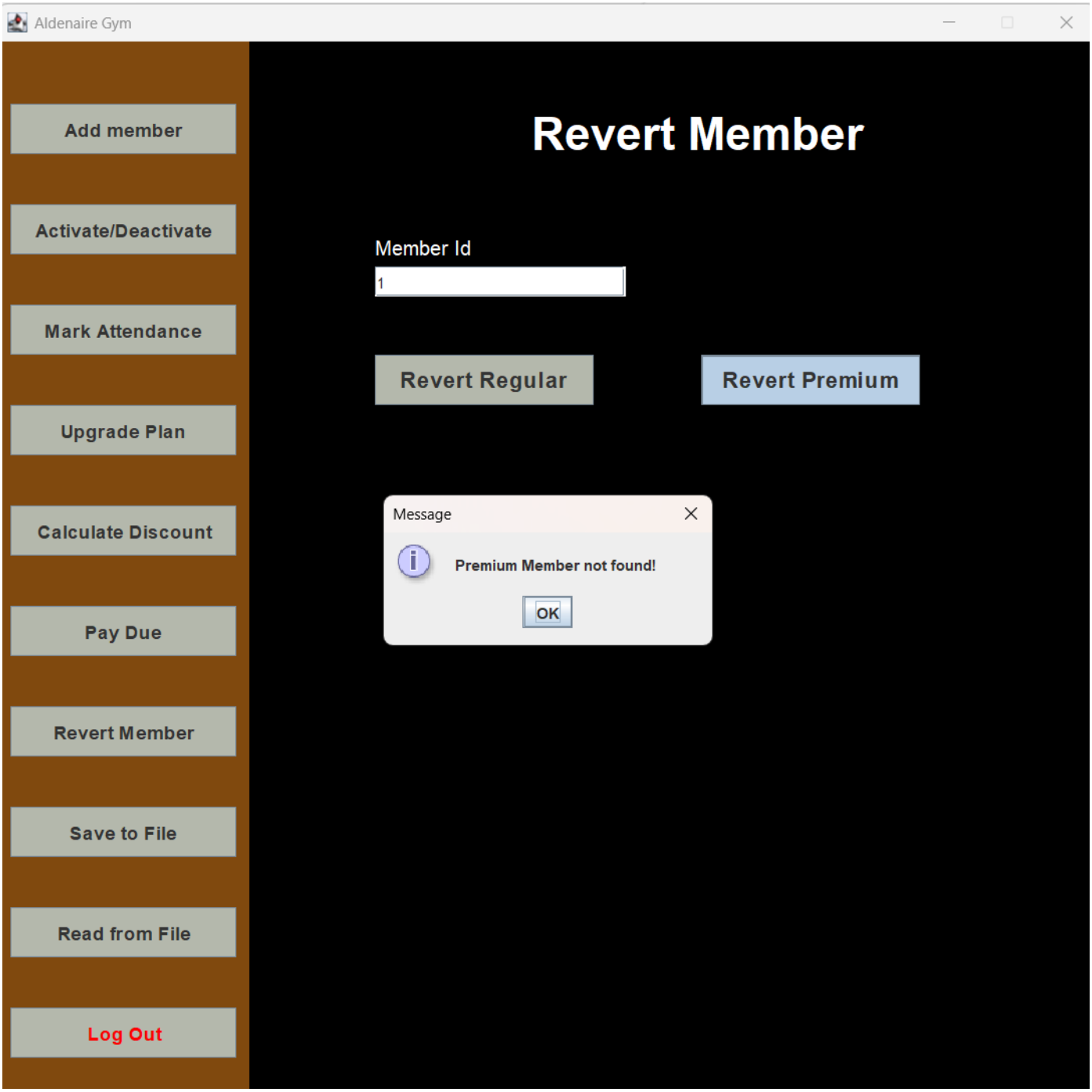


Figure 91: Passing Non-existed id in Premium Member

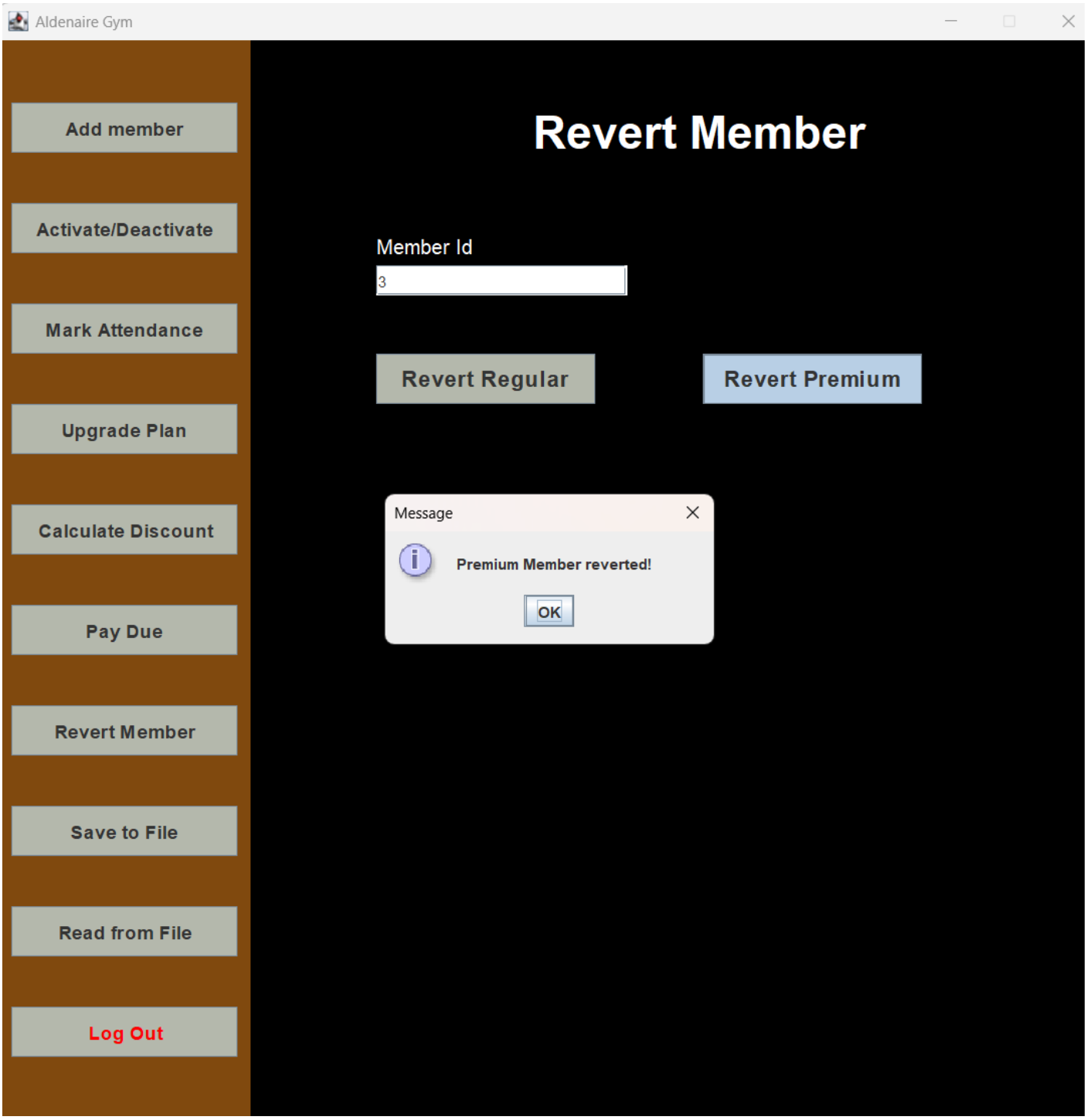


Figure 92: Premium Member Reverted Successfully

Testing 5

Testing 5.1

Table 8: Testing 5.1: To test case for saving to file

Objective	To test case for saving to file.
Action	i. Click Save to File button. ii. Fill out all the details of regular members and add members. iii. Click Save to File button. iv. Fill out all the details of premium members and add members. v. Click Save to File button.
Expected Output	The details of members stored in the ArrayList should be written in the file.
Actual Output	The details of members stored in the ArrayList was written in the file.
Result	The test was successful.

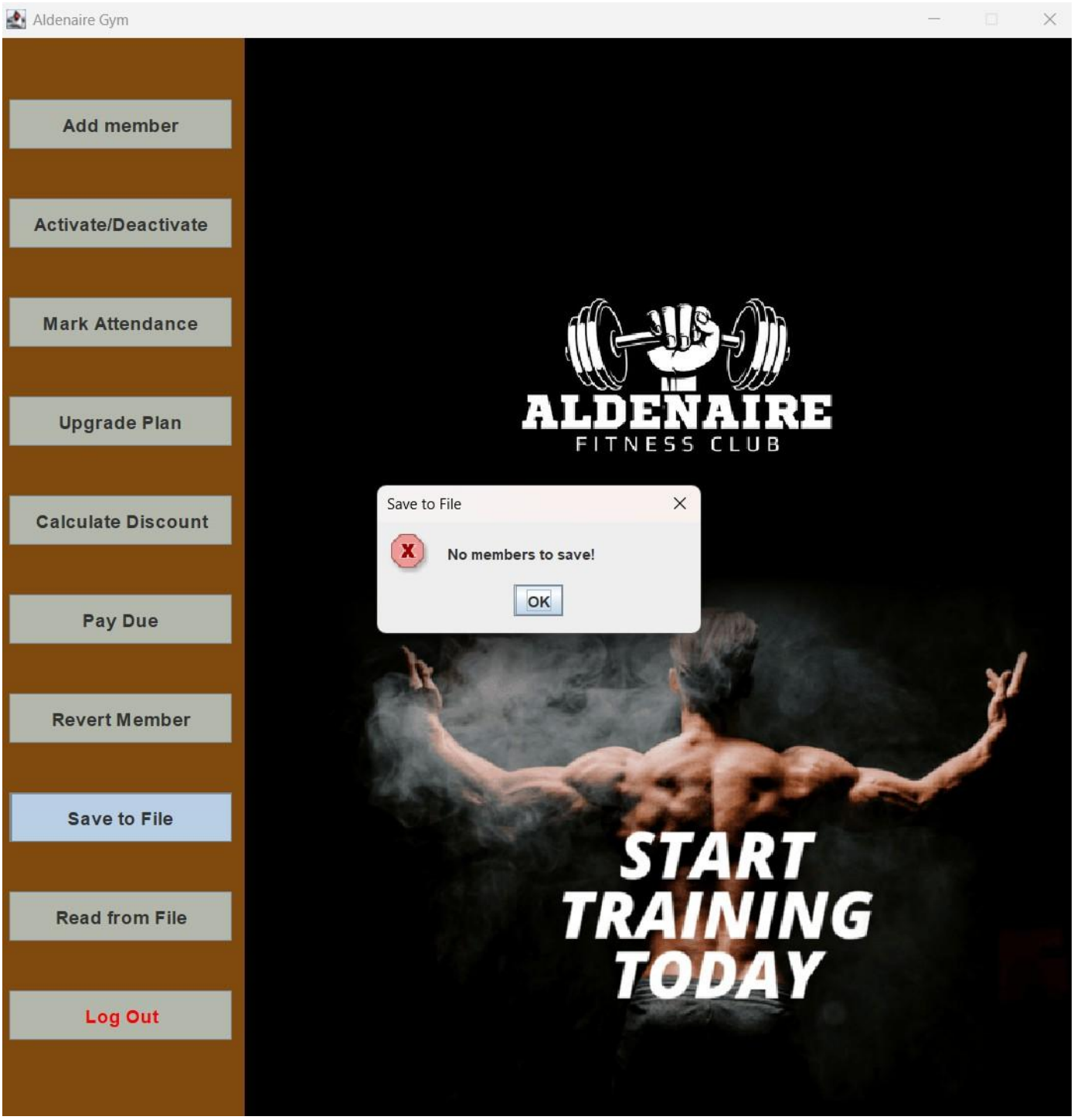


Figure 93: No member is available to Save to File

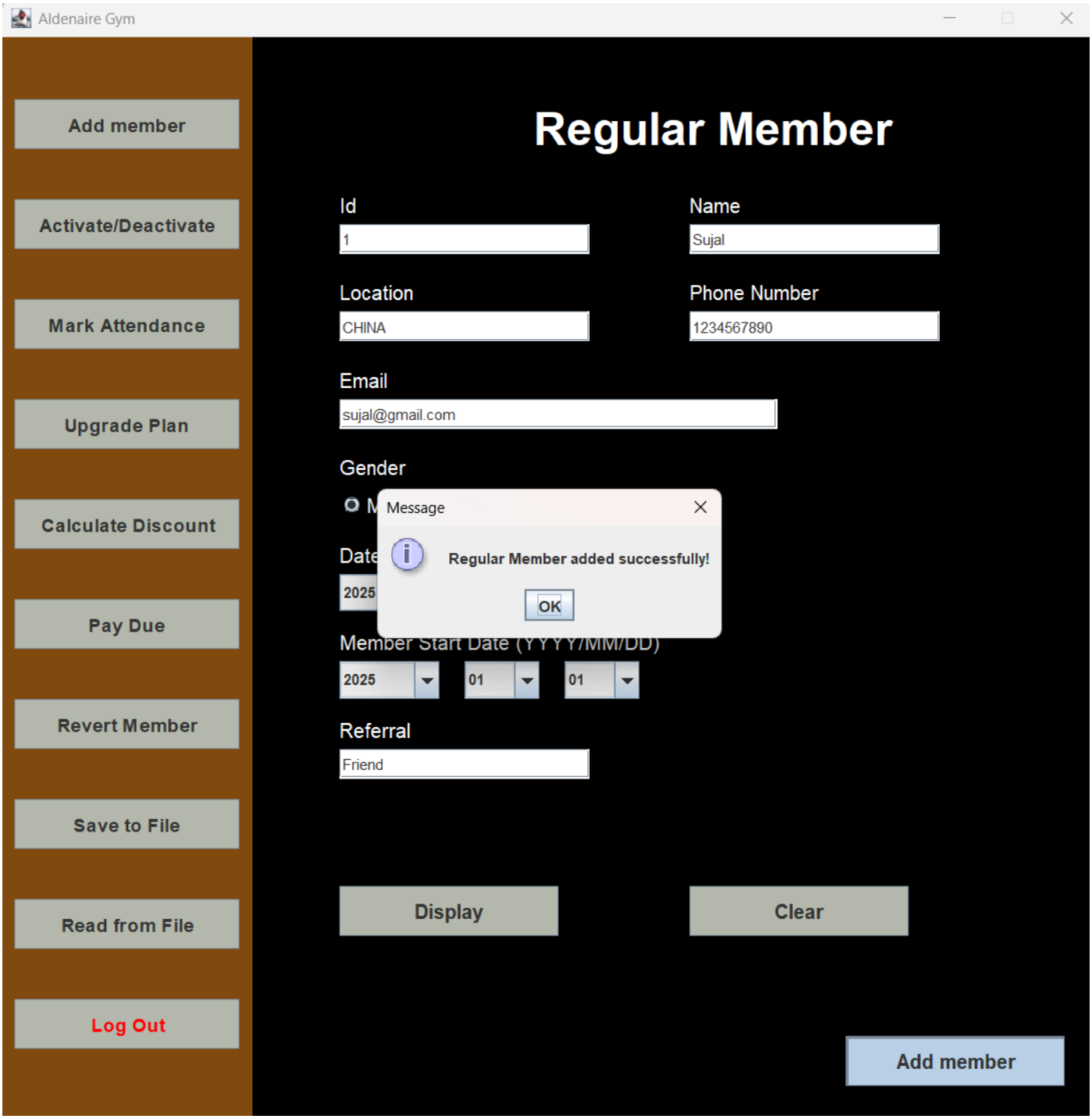


Figure 94: Added Regular Member to Save to File

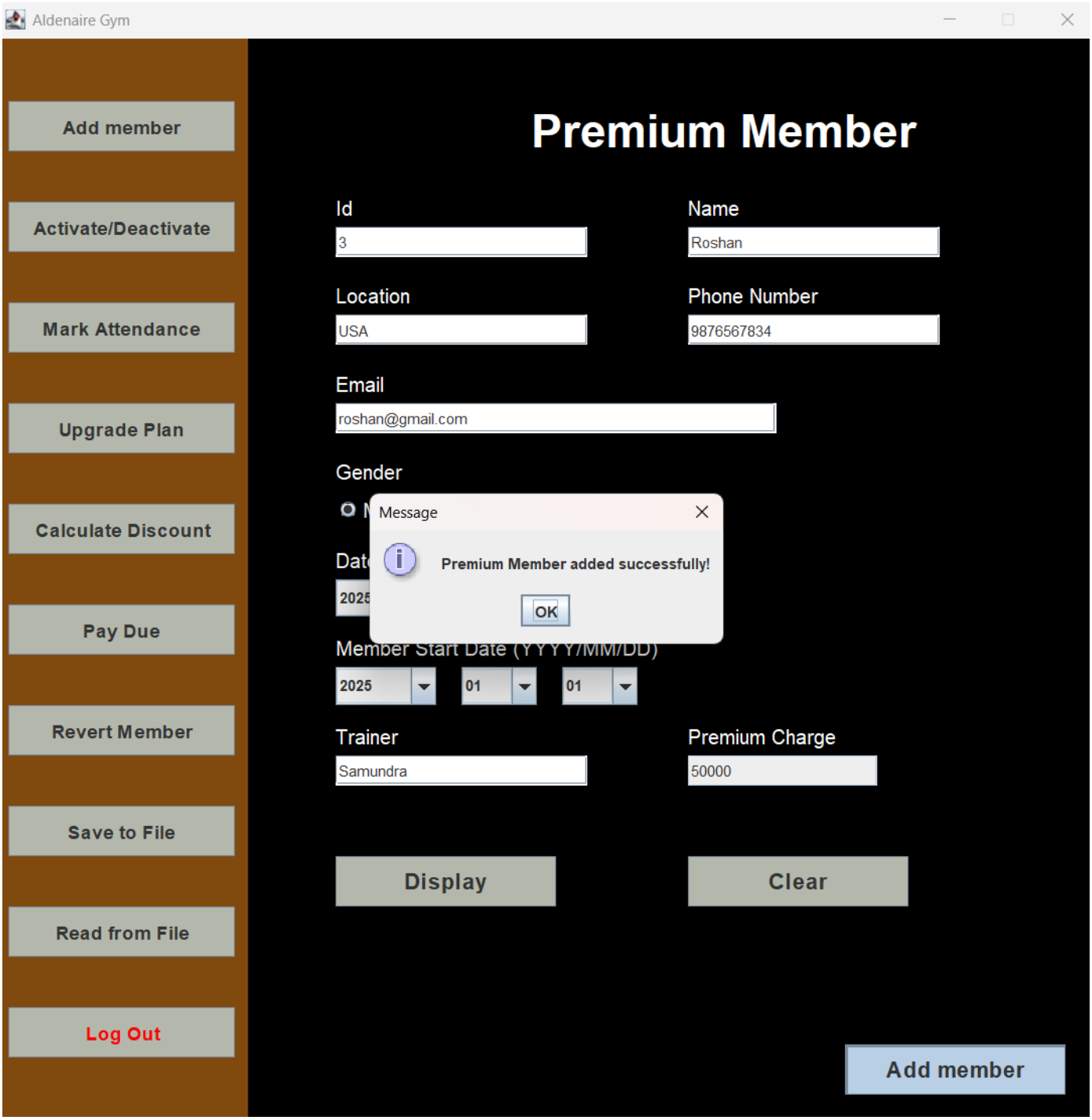


Figure 95: Added Premium Member to Save to File

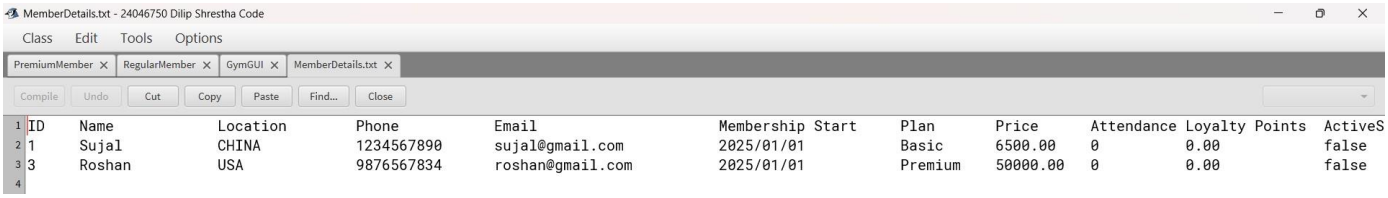


Figure 96: Opened MemberDetails.txt to check saved files

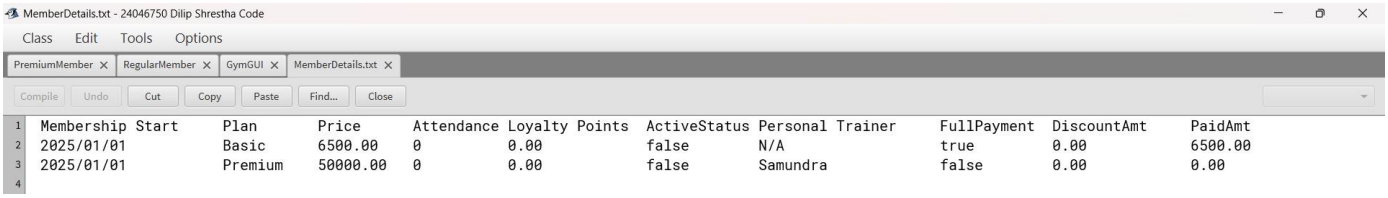


Figure 97: Opened MemberDetails.txt to check saved files 2

Testing 5.2

Table 9: Testing 5.2: To test case for reading from file

Objective	To test case for reading from file
Action	i. Clicked “Read from File” button. ii. Add a new member. iii. Clicked “Save to File” button. iv. Clicked “Read from File”.
Expected Output	The details of members stored in arraylist should be read in a new frame. And when a new member is added, then it should override the old details.
Actual Output	The details of members stored in arraylist were read in a new frame. And when new members are added, then it overrides the old details.
Result	The test was successful.

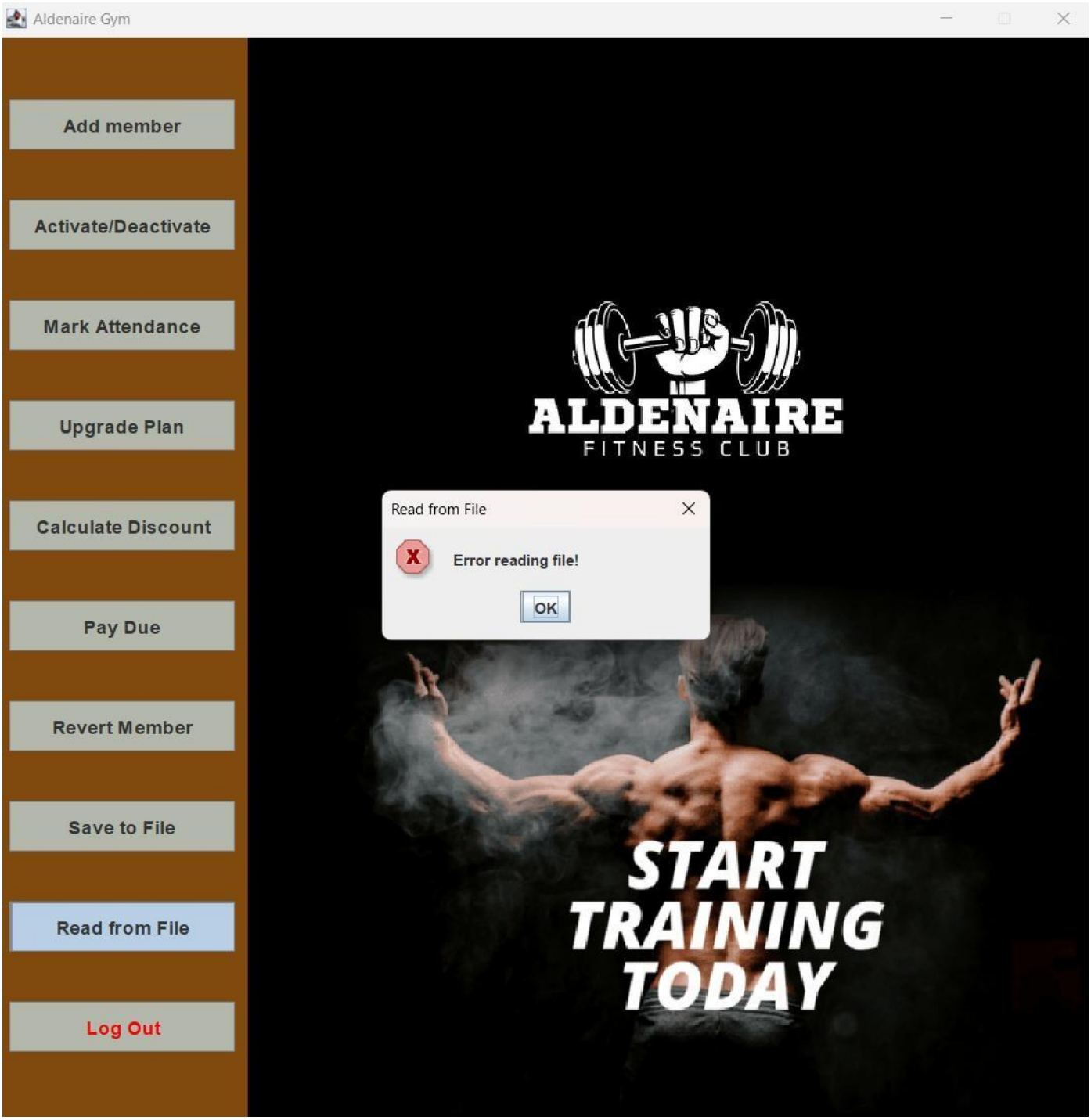


Figure 98: Message after clicking Read From File when none of the file is saved

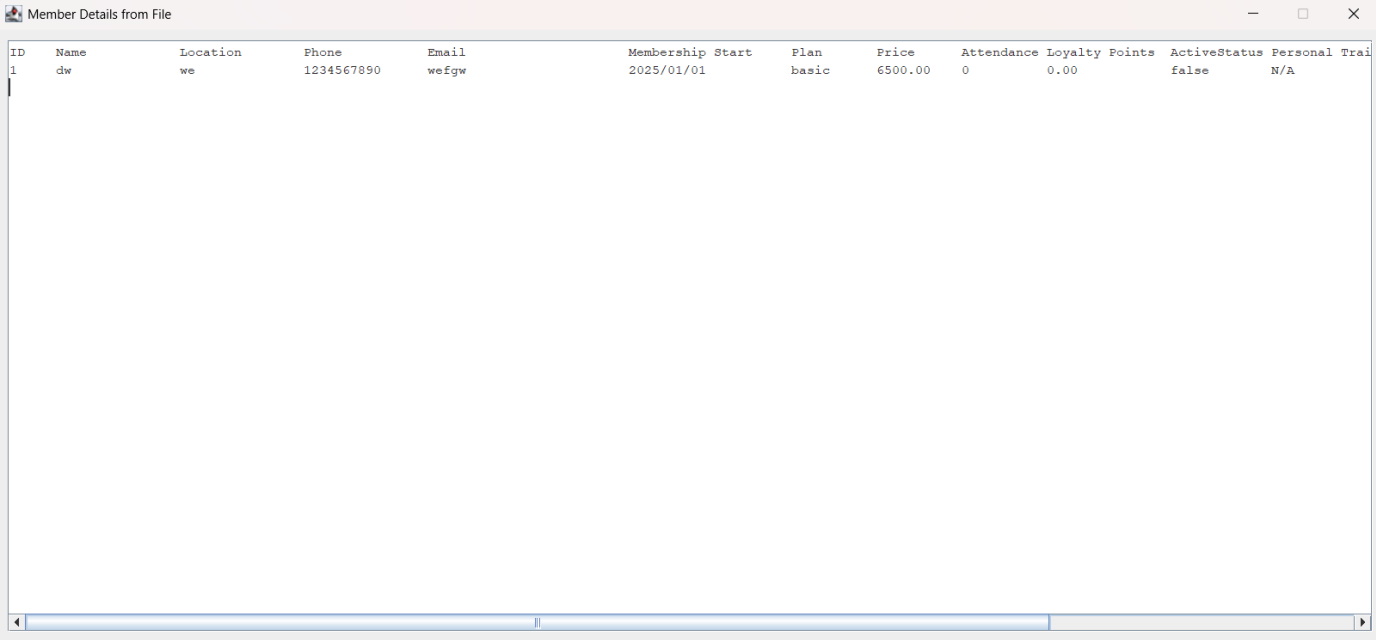


Figure 99: Read from File Panel

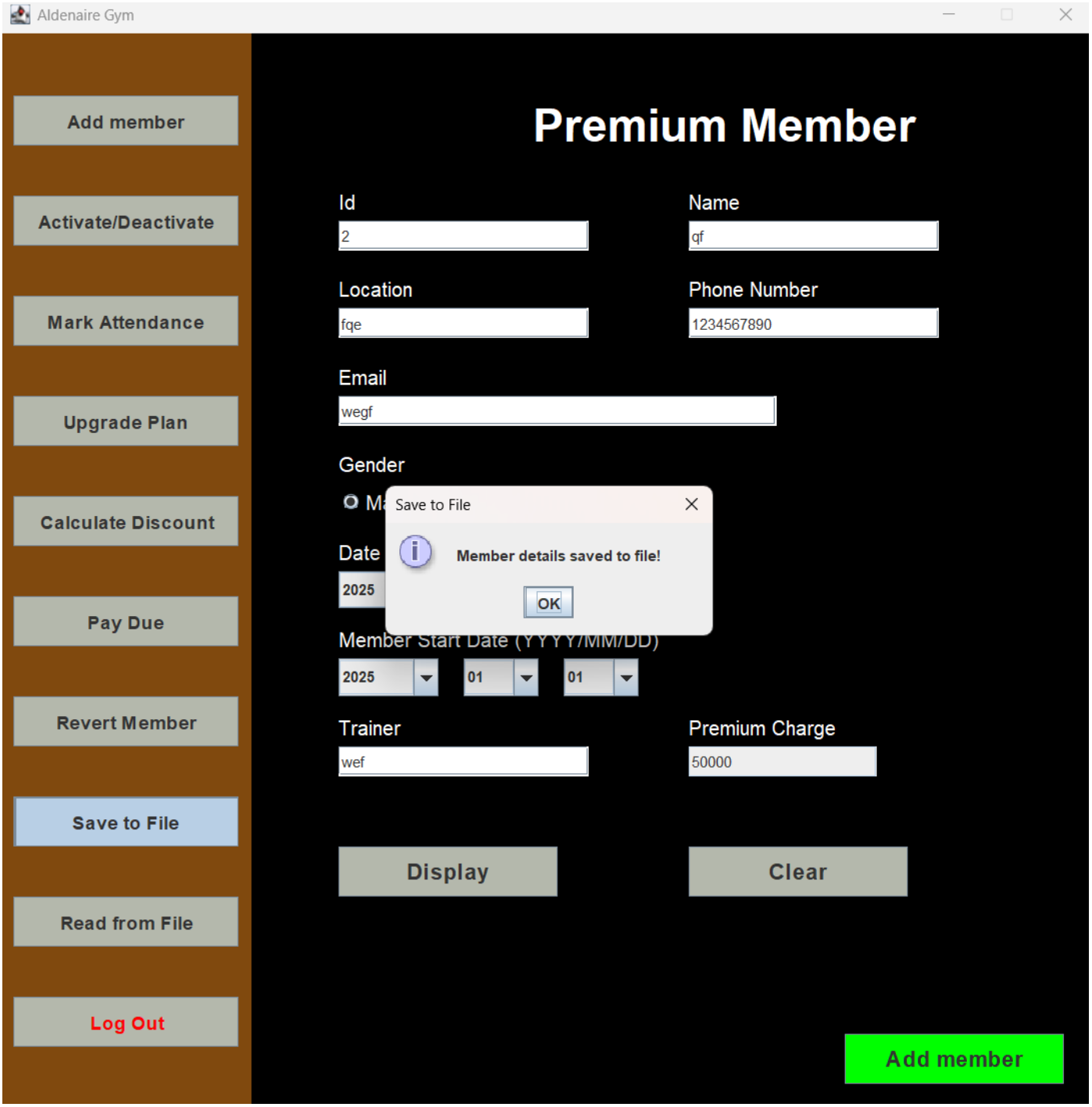


Figure 100: Added Premium Member and Save to File

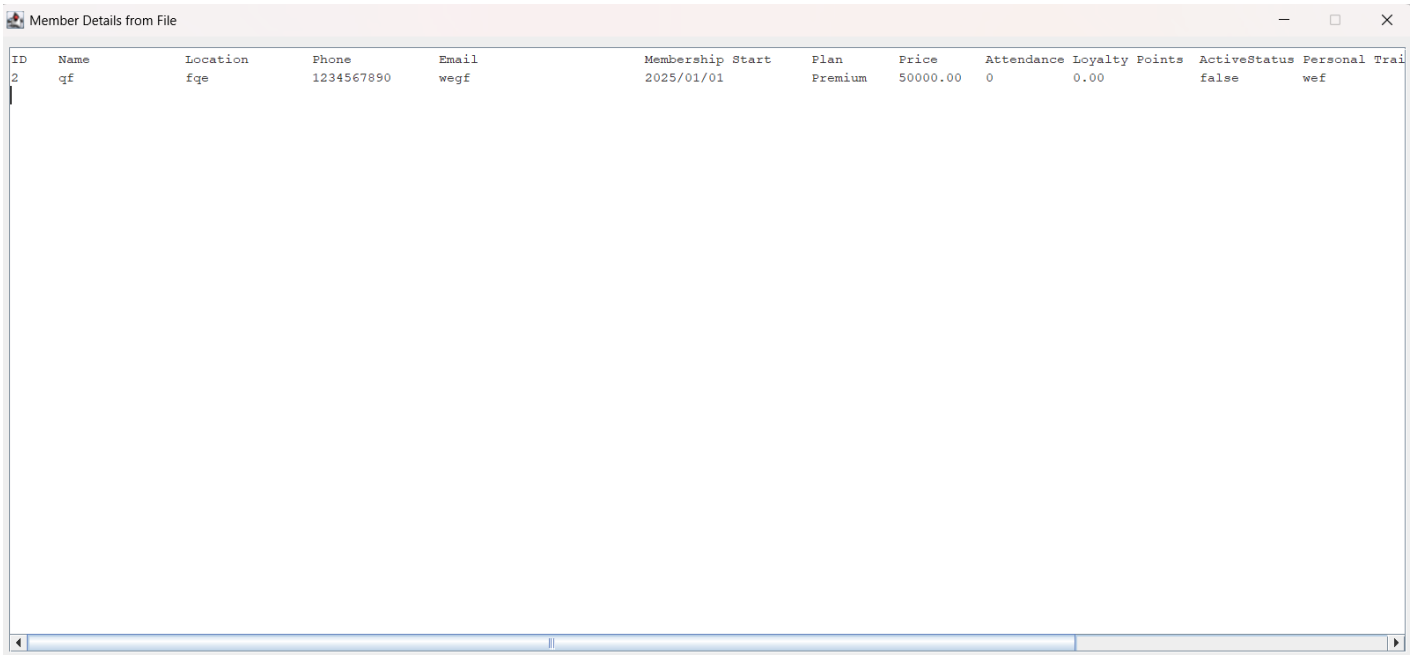


Figure 101: Clicked Read from File and Opened Panel

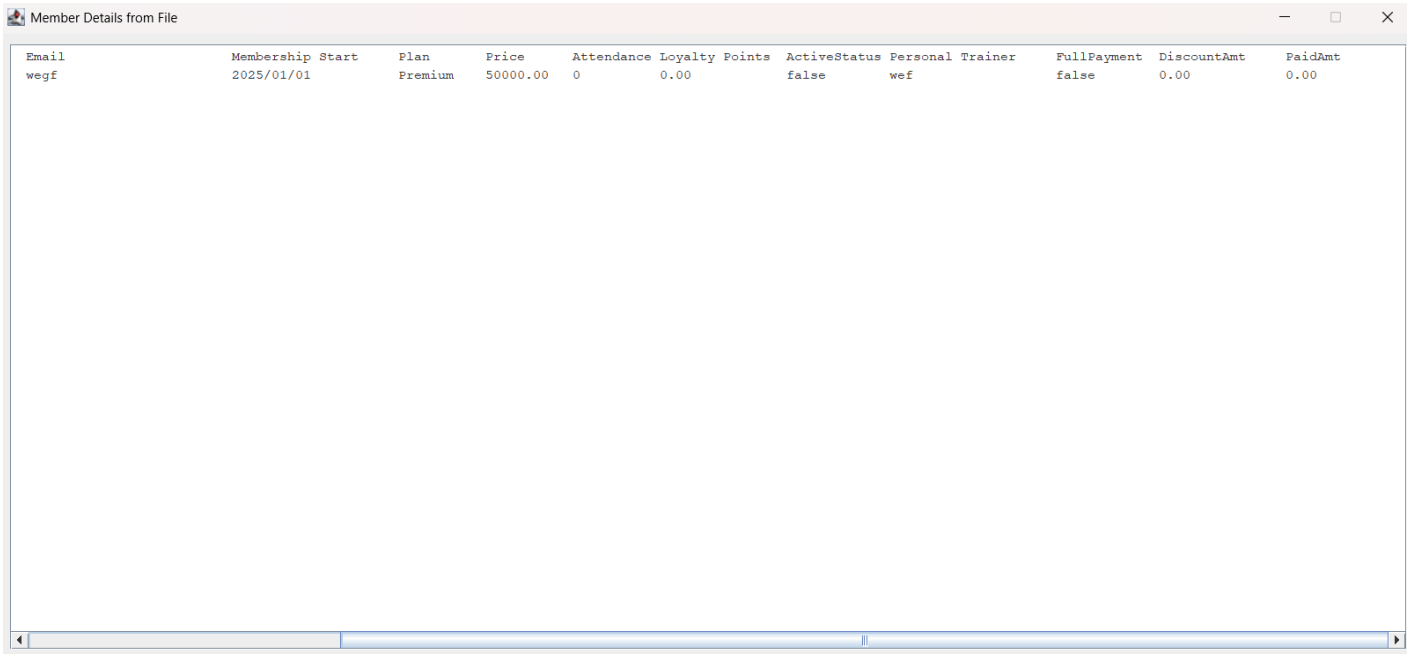


Figure 102: Clicked Read from File and opened Panel

Error Detection and Correction

Error detection and error correction within programming involves the identification and fixing of errors to make sure the source code will work as specified. Error detection means searching for the problem that is interfering in the program and can cause error in the future. Whether it is a syntax error: like missing a semicolon or bracket, which prevents the code from even running, runtime error arising during the program's execution, such as dividing by zero, or trying to retrieve an item from a list that does not exist, or logical error code runs fine but produces incorrect results, such as math errors.

In this coursework while developing the program, several errors were identified and resolved to ensure that program functioned as expected. Below are three specific examples of errors: syntax, semantic, and logical occurred during implementation, along with how they were detected and corrected.

Syntax Error

A syntax error occurred in the GymGUI class while setting up the method for code reusability. The error stemmed from a missing semicolon semi color at the end of a statement and ending bracket.

```
050
051     // Method to hide all sub-panels
052     public void hideAllSubPanels()
053     {
054         homePanel.setVisible(false); // Hiding the panel as setting visible to false
055         addMemberPanel.setVisible(false); // Hiding the panel as setting visible to false
056         regularPanel.setVisible(false); // Hiding the panel as setting visible to false
057         premiumPanel.setVisible(false); // Hiding the panel as setting visible to false
058         activateDeactivatePanel.setVisible(false); // Hiding the panel as setting visible to false
059         markAttendancePanel.setVisible(false); // Hiding the panel as setting visible to false
060         upgradePlanPanel.setVisible(false); // Hiding the panel as setting visible to false
061         discountPanel.setVisible(false); // Hiding the panel as setting visible to false
062         revertMemberPanel.setVisible(false)// Hiding the panel as setting visible to false
063         payDuePanel.setVisible(false)// Hiding the panel as setting visible to false
064
065
066     // Button pointer method explanation
067     // Enhances button usability by setting cursor and focus properties
068     private void buttonPointer(JButton button)
069     {
```

Figure 103: Syntax error in GymGUI

Correction

The error was then resolved by adding the missing semicolon to properly terminate the statement and adding closing bracket to close the method/behaviour.

```
050
051     // Method to hide all sub-panels
052     public void hideAllSubPanels()
053     {
054         homePanel.setVisible(false); // Hiding the panel as setting visible to false
055         addMemberPanel.setVisible(false); // Hiding the panel as setting visible to false
056         regularPanel.setVisible(false); // Hiding the panel as setting visible to false
057         premiumPanel.setVisible(false); // Hiding the panel as setting visible to false
058         activateDeactivatePanel.setVisible(false); // Hiding the panel as setting visible to false
059         markAttendancePanel.setVisible(false); // Hiding the panel as setting visible to false
060         upgradePlanPanel.setVisible(false); // Hiding the panel as setting visible to false
061         discountPanel.setVisible(false); // Hiding the panel as setting visible to false
062         revertMemberPanel.setVisible(false); // Hiding the panel as setting visible to false
063         payDuePanel.setVisible(false); // Hiding the panel as setting visible to false
064     }
065
066     // Button pointer method explanation
067     // Enhances button usability by setting cursor and focus properties
068     private void buttonPointer(JButton button)
069     {
```

Figure 104: Correction of Syntax Error

Syntax errors arise when the code violets grammatical rules of the programming language. The missing semicolon and closing bracket caused the compiler to fail because Java requires statements to end with a semicolon.

Semantic Error

A semantic error was detected in the RegularMember class within the constructor. Here in the referralSource I didn't use "this" keyword where this line accidentally re-assigned the local parameter back to itself, leaving the object's this.referralSource field uninitialized. As a result, every new member's referral source stayed empty, no matter what the user typed in. At runtime, when displaying newly added regular members, I saw that the "Referral Source" line was always blank which led me for semantic error detection.

```
//Constructor of class RegularMember accepting following attributes
public RegularMember(int id, String name, String location, String phone, String email, String gender, String DOB, String membershipStartDate,
String referralSource)
{
    super(id, name, location, phone, email, gender, DOB, membershipStartDate); //super keyword used to access the parent class attributes
    this.isEligibleForUpgrade = false;
    this.removalReason = "";
    referralSource = referralSource;
    this.plan = "basic";
    this.price = 6500;
}

```

Figure 105: Semantic Error in RegularMember

Correction

The semantic error was then resolved by this key word, which initializes the instance variable. Now the constructor writes into the object's field instead of the method's local variable.

```
//Constructor of class RegularMember accepting following attributes
public RegularMember(int id, String name, String location, String phone, String email, String gender, String DOB, String membershipStartDate,
String referralSource)
{
    super(id, name, location, phone, email, gender, DOB, membershipStartDate); //super keyword used to access the parent class attributes
    this.isEligibleForUpgrade = false;
    this.removalReason = "";
    this.referralSource = referralSource;
    this.plan = "basic";
    this.price = 6500;
}
```

Figure 106: Correction of Semantic Error

A semantic error occurs when the code is grammatically correct, it compiles and runs without any error but doesn't do what the program was intended to do.

Logical Error

A logical error was detected in RegularMember. he attendanceLimit is 30, and I wanted members to be eligible for an upgrade when their attendance reaches 30 or more. But using > (greater than) meant that someone with exactly 30 attendances wouldn't qualify, only 31 or higher would work. That wasn't my intention.

I detected this error when I kept attendance as 30 but wasn't eligible for plan upgrade.

```
91          {
92              return "You are already subscribed to the " + plan + " plan.";
93          }
94
95          if (super.attendance > attendanceLimit) // Check if the member is eligible for an upgrade based on attendance
96          {
97              this.isEligibleForUpgrade = true;
98          }
99
```

Figure 107: Logical Error in RegularMember

Correction

I changed the condition to include 30 by using greater or equal to sign. Now, if a member has 30 or more attendances, they're eligible.

```
    }

    if (super.attendance >= attendanceLimit) // Check if the member is eligible for an upgrade based on attendance
    {
        this.isEligibleForUpgrade = true;
    }

    if (!isEligibleForUpgrade)
```

Figure 108: Correction of Logical Error

Overall, these three errors: a missing semicolon and closing parenthesis (syntax), absence of this keyword (semantic), and a wrong comparison (logical), show how different mistakes can arise during coding. Fixing them made my program work better and taught me to double-check my syntax, understand my variables, and test my logic.

Conclusion

In conclusion, the coursework aimed to make a program to help gyms keep records of the members stored in its system: regular gym members and premium members. The system, therefore, should allow for new members to be added, attendance to be marked, memberships to be canceled, payments to be made, discounts to be calculated, and plans to be upgraded. Also, having made use of some of the object-oriented programming (OOP) features, such as abstraction, encapsulation, polymorphism, and inheritance, the program has an organized and easy-to-understand source code. A GUI has been made for the system to be user-friendly and to allow effortless interaction with the core functionalities.

While working on this project It has been a great learning opportunity for me in programming and solving problems. I discovered how important it is to plan the structure of a program even before writing a line of code. The breaking up of the whole project into smaller chunks, which were easier to handle and less intimidating, was a good move. I also realized the importance of reflective practices: after every process, I always reflected on what went right and could be improved; this further helped in decision making and early correction of mistakes. Testing and debugging another process taught me the value of patience and attention to detail. I must admit there is now a lot of confidence working with Java Swing for building GUIs, handling user inputs, and events. Most importantly, I have learned that sometimes the best education comes from real-life experiences and mistakes rather than textbooks or clearly outlined tutorials.

There were several challenges and difficulties while doing this coursework. One of the biggest was dealing with changing requirements. While I was working on the project, it suddenly dawned on me that some features had to be added, others had to be changed, and the entire system had to be made practically usable. This called for some going back to code some parts again. Another was time management, especially when imposed with short deadlines. Debugging errors and fixing bugs was one tough road, because some of the errors were so elusive, involving a lot of trial and error. Designing the GUI in a way that would be appealing and functional was also quite a challenge, as this took some planning to ensure easy accessibility to all application features. Lastly, it was continuously a challenge to keep the code organized and maintainable as the project began to grow. To overcome these difficulties, whenever requirements were changing, I always kept

communication open with anyone involved and maintained documentation changes in a clear manner. I split larger tasks into smaller manageable chunks and set realistic deadlines for completing each one, keeping me on track and preventing overwhelm. When troubleshooting technical problems or facing stubborn bugs, I would not hesitate to seek help from more experienced programmers or consult online resources. I used debugging tools and tested my programs regularly to catch errors as early as possible. In designing the GUI, I reviewed a few example applications and attempted to keep things straightforward and logical. Regular reflection and review helped me spot areas for improvement and make changes before problems became too big. Staying positive and treating every setback as a learning opportunity made it easier to keep going, even when things got tough.

In conclusion, this project was a valuable learning experience. It helped me improve my programming skills, taught me how to manage a project from start to finish, and showed me the importance of planning, reflection, and adaptability. The difficulties I faced were challenging but also rewarding, as overcoming them made me a better problem-solver

Appendix

```
public abstract class GymMember //Abstract Class GymMember
{
    //attributes of GymMember and access modifier as protected
    protected int id;
    protected String name;
    protected String location;
    protected String phone;
    protected String email;
    protected String gender;
    protected String DOB;
    protected String membershipStartDate;
    protected int attendance;
    protected double loyaltyPoints;
    protected boolean activeStatus;

    //Constructor of class GymMember accepting following attributes
    public GymMember(int id, String name, String location, String phone, String email,
String gender, String DOB, String membershipStartDate)
    {
        this.id = id;
        this.name = name;
        this.location = location;
        this.phone = phone;
        this.email = email;
        this.gender = gender;
        this.DOB = DOB;
        this.membershipStartDate = membershipStartDate;
        this.attendance = 0;
        this.loyaltyPoints = 0.0;
        this.activeStatus = false; //Membership is deactivated by default i.e activeStatus
set to false
    }

    //Each attributes with its corresponding accessor/getter method
    public int getId()
    {
        return this.id;
    }

    public String getName()
    {
        return this.name;
    }
```

```
    public String getLocation()
    {
        return this.location;
    }

    public String getPhone()
    {
        return this.phone;
    }

    public String getEmail()
    {
        return this.email;
    }

    public String getGender()
    {
        return this.gender;
    }

    public String getDOB()
    {
        return this.DOB;
    }

    public String getMembershipStartDate()
    {
        return this.membershipStartDate;
    }

    public int getAttendance()
    {
        return this.attendance;
    }

    public double getLoyaltyPoints()
    {
        return this.loyaltyPoints;
    }

    public boolean getActiveStatus()
    {
        return this.activeStatus;
    }

    /* Abstract method markAttendance() is created
```

```
     * and abstract method does not contain body*/
    public abstract void markAttendance();

    /* Concrete method activateMembership() is created and activeStatus is set to true,
       when the members want to activate or renew their membership*/
    public void activeMembership()
    {
        this.activeStatus = true;
    }

    /* Concrete method deactivateMembership() is created and activeStatus is set to
false by using if else condition
       when the members want to deactivate or pause their membership*/
    public void deactivateMemberhship()
    {
        if(this.activeStatus == true) //membership must be activated first in order to
deactivate membership
        {
            this.activeStatus = false;
        }
        else //else membership cannot be deactivated
        {
            System.out.println("Membership is not actived.");
        }
    }


    /* resetMember() method is created in order to set values to default,
       i.e. activeStatus to false, attendance to zero and loyaltyPoints to zero.*/
    public void resetMember()
    {
        this.activeStatus = false;
        this.attendance = 0;
        this.loyaltyPoints = 0.0;
    }

    //display() method is created for suitable output
    public void display()
    {
        System.out.println("ID of member is: "+ id);
        System.out.println("Name of member is: "+ name);
        System.out.println("Location of member is: "+ location);
        System.out.println("Phone Number of member is: "+ phone);
        System.out.println("Email of member is: "+ email);
        System.out.println("Gender of member is: "+ gender);
        System.out.println("DOB: "+ DOB);
```

```
        System.out.println("Membership Start Date: "+ membershipStartDate);
        System.out.println("Attendace: "+ attendance);
        System.out.println("Loyalty Points: "+ loyaltyPoints);
        System.out.println("Active Status: "+ activeStatus);
    }
}
```

Apendix of Regular Member

```
public class RegularMember extends GymMember //RegularMember extends
GymMember
{
    //attributes and access modifier as private
    private final int attendanceLimit = 30; //value 30 is stored as constant in variable
attendanceLimit
    private boolean isEligibleForUpgrade;
    private String removalReason;
    private String referralSource;
    private String plan;
    private double price;

    //Constructor of class RegularMember accepting following attributes
    public RegularMember(int id, String name, String location, String phone, String email,
String gender, String DOB, String membershipStartDate,
    String referralSource)
    {
        super(id, name, location, phone, email, gender, DOB, membershipStartDate);
//super keyword used to access the parent class attributes
        this.isEligibleForUpgrade = false;
        this.removalReason = "";
        this.referralSource = referralSource;
        this.plan = "basic";
        this.price = 6500;
    }

    // Getter/Accessor method
    public int getAttendanceLimit()
    {
        return this.attendanceLimit;
    }

    public boolean getIsEligibleForUpgrade()
    {
        return this.isEligibleForUpgrade;
    }

    public String getRemovalReason()
```

```
    {
        return this.removalReason;
    }

    public String getReferralSource()
    {
        return this.referralSource;
    }

    public String getPlan()
    {
        return this.plan;
    }

    public double getPrice()
    {
        return this.price;
    }

    //Overriding method markAttendace() of parent class
    @Override
    public void markAttendance()
    {
        super.attendance++; //attendance increase by 1 every time this method is called
        super.loyaltyPoints += 5; //loyaltyPoints incease by 5 every time this method is
called
        if (super.attendance >= attendanceLimit) // Check if the member is eligible for an
upgrade based on attendance
        {
            this.isEligibleForUpgrade = true;
        }
    }

    //getPlanPrice() method is created with return type double and using switch case for
the returning following  plan's price and returns -1 as default
    public double getPlanPrice(String plan)
    {
        switch (plan.toLowerCase())  //toLowerCase() method set all the gym plan to lower
case
        {
            case "basic":
                return 6500;
            case "standard":
                return 12500;
            case "deluxe":
                return 18500;
```

```
            default:
                return -1;
        }
    }

    public void setPlan(String plan) {
        this.plan = plan;
    }

    /* upgradePlan() method is created with return type String
       used to upgrade the plan subscribed by the member */
    public String upgradePlan(String plan)
    {
        if (this.plan.equalsIgnoreCase(plan)) // Validate if the same plan is being chosen.
And equalsIgnoreCase() ignore the case of letters
        {
            return "You are already subscribed to the " + plan + " plan.";
        }

        if (super.attendance >= attendanceLimit) // Check if the member is eligible for an
upgrade based on attendance
        {
            this.isEligibleForUpgrade = true;
        }

        if (!isEligibleForUpgrade)
        {
            return "Not eligible for upgrade yet. Attendance must reach " + attendanceLimit
+ ".";
        }

        // Retrieve the new plan's price
        double newPrice = getPlanPrice(plan);
        if (newPrice == -1)
        {
            return "Invalid plan selected.";
        }

        // Update the plan and its price if eligible for plan upgrade
        this.plan = plan;
        this.price = newPrice;
        return "Plan upgraded to " + plan + ". New price: Rs. " + newPrice;
    }

    //revertRegularMember() method is created which accepts removalReason as a
parameter
```

```
    public void revertRegularMember(String removalReason)
    {
        super.resetMember(); //A super class method resetMember() is called
        this.isEligibleForUpgrade = false;
        this.plan = "basic";
        this.price = 6500;
        this.removalReason = removalReason;
    }

    //Using same method as in GymMember class using "@Override" keyword
    @Override
    public void display()
    {
        super.display();
        System.out.println("Plan: " + plan);
        System.out.println("Price: " + price);
        if (!removalReason.isEmpty()) // if removalReason value is not empty then
removalReason is displayed
        {
            System.out.println("Removal Reason: " + removalReason);
        }
        System.out.println("");
    }
}
```

Appendix of Premium Member

```
public class PremiumMember extends GymMember //RegularMember extends
GymMember
{
   //attributes and access modifier as private
   private final double premiumCharge=50000;
   private String personalTrainer;
   private boolean isFullPayment;
   private double paidAmount;
   private double discountAmount;

   // Constructor of class PremiumMember accepts nine parameters
   public PremiumMember(int id, String name, String location, String phone, String
email, String gender, String DOB, String membershipStartDate,
   String personalTrainer)
   {
      super(id, name, location, phone, email, gender, DOB, membershipStartDate); // A
call is made to the superclass constructor with eight parameters
      this.personalTrainer = personalTrainer;
      this.isFullPayment = false;
```

```
    this.paidAmount = 0.0;
    this.discountAmount = 0.0;
  }

  public double getPremiumCharge()
  {
     return this.premiumCharge;
  }

  public String getPersonalTrainer()
  {
     return this.personalTrainer;

  }

  public boolean isFullPayment()
  {
     return this.isFullPayment;
  }

  public double getPaidAmount()
  {
     return this.paidAmount;
  }

  public double getDiscountAmount()
  {
     return this.discountAmount;
  }

  //a method markAttendance() method created to override abstract class
  @Override
  public void markAttendance()
  {
     super.attendance = 30; //Attendance increase by 1 every time markAttendace() is
called
     super.loyaltyPoints += 300; // Premium members get 10 loyaltyPoints each at a
time markAttendance() is called
  }

  //a method payDueAmount() of return type String is created to pay the due amount
with paidAmount as a parameter
  public String payDueAmount(double paidAmount)
  {
     if (this.isFullPayment == true) // Check if full payment has already been made
     {
```

```
            return "Payment is already fully completed.";
        }

        if ((this.paidAmount + paidAmount) > premiumCharge) // Check if adding this
payment would exceed the premium charge
        {
            double excess = (this.paidAmount + paidAmount) - premiumCharge;
            return "Excess amount detected of Rs. " + excess;
        }

        // Update the paid amount
        this.paidAmount += paidAmount;
        double remainingAmount = premiumCharge - this.paidAmount;

        // Check if the full payment has been achieved
        if (remainingAmount == 0)
        {
            this.isFullPayment = true;
            return "Payment successful. No remaining amount.";
        }
        else
        {
            return "Payment successful. Remaining amount: Rs. " + remainingAmount;
        }
    }

    /* calculateDiscount() method is created
       method is used to calculate discount amounts based on payment status */
    public void calculateDiscount()
    {
        if (this.isFullPayment)
        {
            this.discountAmount = 0.10 * premiumCharge;
            System.out.println("Discounted amount is Rs.: " + this.discountAmount);
        }
        else
        {
            System.out.println("No discount is given as payment is not full.");
        }
    }

    // a method name revertPremiumMember()
    public void revertPremiumMember()
    {
        super.resetMember();
        this.personalTrainer = "";
```

```
        this.isFullPayment = false;
        this.paidAmount = 0.0;
        this.discountAmount = 0.0;
    }

    // display method created to display the details of the PremiumMember
    @Override //Method of parent class used so override is used
    public void display()
    {
        super.display();
        System.out.println("Personal Trainer: " + personalTrainer);
        System.out.println("Paid Amount: " + paidAmount);
        System.out.println("Full Payment: " + isFullPayment);

        double remainingAmount = premiumCharge - this.paidAmount;
        System.out.println("Remaining Amount: Rs. " + remainingAmount);

        if (this.isFullPayment == true)
        {
            System.out.println("Discount Amount: Rs. " + discountAmount);
        }
        System.out.println("");
     }
}
```

Appendix of GymGUI

```
// Import necessary Swing components from swing package for building the GUI
import javax.swing.JFrame;
import javax.swing.JPanel;
import javax.swing.JTextField;
import javax.swing.JPasswordField;
import javax.swing.JCheckBox;
import javax.swing.JButton;
import javax.swing.JRadioButton;
import javax.swing.JComboBox;
import javax.swing.JLabel;
import javax.swing.ImageIcon;
import javax.swing.ButtonGroup;
import javax.swing.JOptionPane;
import javax.swing.JScrollPane;
import javax.swing.JTextArea;

// Import ArrayList from util package for dynamic storage of member objects
import java.util.ArrayList;
```

```
// Import AWT components from awt package for basic GUI features
import java.awt.Font;
import java.awt.Color;
import java.awt.Cursor;

// Import AWT event classes for handling user interactions
import java.awt.event.ActionListener; // Interface for responding to events like button clicks
import java.awt.event.ActionEvent;     // Represents an event triggered by a user action

// Import IO classes for file handling, enabling reading from and writing to files
import java.io.FileWriter;       // Writes data to a file
import java.io.FileReader;       // Reads data from a file
import java.io.IOException;      // Handles input/output exceptions


// This class, GymGUI, is designed to create a graphical user interface for managing gym memberships
// It includes functionalities such as adding members, marking attendance, and handling payments
// The following lines define the class and its implementation of the ActionListener interface

// GymGUI class implements ActionListener to respond to GUI events
public class GymGUI implements ActionListener
{
    /*This ArrayList is used to store all gym members, both regular and premium
      It provides a dynamic way to manage member data throughout the application
    ArrayList to store GymMember objects (RegularMember and PremiumMember)*/
    private ArrayList<GymMember> membersObj;

    // Instance member variables for input collection
    int id;
    String name, location, phone, email, gender, dob, msd, trainer, referral;

    /* Main application frame and panels
       The frame serves as the main window of the application
       Panels are used to organize different sections of the interface*/
    private JFrame frame;
    private JPanel loginPanel, buttonPanel, mainPanel;
    private JPanel homePanel, addMemberPanel, regularPanel, premiumPanel,
activateDeactivatePanel, markAttendancePanel, upgradePlanPanel,
    discountPanel, revertMemberPanel, payDuePanel;

    // Components for Login Panel
    private JTextField userTxt;
```

```
    private JPasswordField passwordTxt;
    private JCheckBox showPassword;
    private JButton loginButton;

    // Navigation Panel Buttons
    private JButton addMemberBtn, activateBtn, markAttendanceNavBtn, upgradeBtn,
discountBtn, payDueNavBtn, revertMemberNavBtn, saveBtn, readBtn, logOutBtn;

    /*Components for "Add Member" Panel
      Buttons to switch between adding regular and premium members
    These allow the user to choose the type of member to add*/
    private JButton addRegularSwitchBtn, addPremiumSwitchBtn;

    // Components for "Add Regular Member" Panel
    // Frame for displaying regular member information
    // This frame will show details of all regular members when requested
    private JFrame rDisplayInfoFrame;
    // Text fields for inputting regular member details
    // These fields collect data such as ID, name, and contact information
    private JTextField rIdField, rNameField, rLocationField, rPhoneField, rEmailField,
rReferralField, rPriceField;
    // Radio buttons for selecting gender
    // Allow the user to specify the gender of the regular member
    private JRadioButton rMaleBtn, rFemaleBtn;
    // Combo boxes for selecting date of birth and membership start date
    // Provide dropdown options for dates to ensure valid input
    private JComboBox<String> rDobYear, rDobMonth, rDobDay, rMsYear, rMsMonth,
rMsDay, rPlanBox;
    // Buttons for adding, displaying, and clearing regular member information
    // Facilitate the management of regular member data
    private JButton rAddMemberBtn, rDisplayBtn, rClearBtn;

    // Components for "Add Premium Member" Panel
    // Frame for displaying premium member information
    // Similar to the regular member display frame but for premium members
    private JFrame pDisplayInfoFrame;
    // Text fields for inputting premium member details
    // Include additional fields like trainer specific to premium members
    private JTextField pIdField, pNameField, pLocationField, pPhoneField, pEmailField,
pTrainerField, pChargeField;
    // Radio buttons for selecting gender
    // Same purpose as in the regular member panel
    private JRadioButton pMaleBtn, pFemaleBtn;
    // Combo boxes for selecting date of birth and membership start date
    // Ensure consistent date input for premium members
```

```
    private JComboBox<String> pDobYear, pDobMonth, pDobDay, pMsYear, pMsMonth,
pMsDay;
    // Buttons for adding, displaying, and clearing premium member information
    // Manage premium member data similarly to regular members
    private JButton pAddMemberBtn, pDisplayBtn, pClearBtn;

    // Components for "Activate/Deactivate Membership" Panel
    // Text field for entering member ID to activate or deactivate membership
    // Used to identify the member whose status will change
    private JTextField actIdField;
    // Buttons for activating and deactivating membership
    // Trigger the respective actions on the specified member
    private JButton activateMemberBtn, deactivateMemberBtn;

    // Components for "Mark Attendance" Panel
    // Text field for entering member ID to mark attendance
    // Identifies the member for attendance tracking
    private JTextField markIdField;
    // Buttons for marking attendance for regular and premium members
    // Differentiate between member types for attendance
    private JButton markRegularBtn, markPremiumBtn;

    // Components for "Upgrade Plan" Panel
    // Text field for entering member ID to upgrade plan
    // Specifies which member's plan will be upgraded
    private JTextField upIdField;
    // Combo box for selecting the new plan
    // Offers plan options for the upgrade
    private JComboBox<String> upPlanBox;
    // Button for upgrading the plan
    // Initiates the plan upgrade process
    private JButton upgradePlanBtn;

    // Components for "Calculate Discount" Panel
    // Text field for entering member ID to calculate discount
    // Identifies the premium member for discount calculation
    private JTextField discIdField, discAmountField;
    // Button for calculating the discount
    // Triggers the discount computation
    private JButton calcDiscountBtn;

    // Components for "Revert Member" Panel
    // Text field for entering member ID to revert member status
    // Targets the member to be removed or reverted
    private JTextField revertIdField;
    // Buttons for reverting regular and premium members
```

```
    // Handle reversion for each member type
    private JButton revertRegularBtn, revertPremiumBtn;

    // Components for "Pay Due" Panel
    // Text fields for entering member ID and amount to pay due
    // Collect payment details for premium members
    private JTextField payIdField, payAmountField;
    // Button for paying the due amount
    // Processes the payment
    private JButton payDueButton;

    // Arrays for combo boxes
    // These arrays provide predefined options for date and plan selections
    // They ensure consistency and validity in user inputs
    private String[] yearsArr = {"2025", "2024", "2023", "2022", "2021", "2020", "2019",
"2018", "2017", "2016", "2015", "2010", "2000", "1995", "1990",
    "1980", "1970", "1960", "1950"};
    private String[] monthsArr = {"01", "02", "03", "04", "05", "06", "07", "08", "09", "10",
"11", "12"};
    private String[] daysArr = { "01", "02", "03", "04", "05", "06", "07", "08", "09", "10",
"11", "12", "13", "14", "15", "16", "17", "18", "19", "20",
    "21", "22", "23", "24", "25", "26", "27", "28", "29", "30", "31"};
    private String[] planOptionsArr = {"Basic", "Standard", "Deluxe"};

    // Constructor explanation
    // The constructor initializes all GUI components and sets up the layout
    // It is the entry point for building the application's interface
    // Constructor of GymGUI class where all GUI is done
    public GymGUI()
    {
        // Frame setup comments
        // Setting up the main window with a title and specific dimensions
        frame = new JFrame("Aldenaire Gym");
        frame.setSize(890, 910);
        frame.setLayout(null);
        frame.setDefaultCloseOperation(JFrame.EXIT_ON_CLOSE);
        frame.setResizable(false);

        // Initialize the ArrayList for member storage
        // This ensures we can start adding members immediately
        membersObj = new ArrayList<GymMember>();

        // Login panel configuration
        // The login panel is the first screen users see
        // It includes fields and buttons for authentication
        // Login Panel
```

```
        loginPanel = new JPanel();
        loginPanel.setLayout(null);
        loginPanel.setBounds(0, 0, 890, 910);
        loginPanel.setBackground(Color.BLACK);

        // Adding the admin login title
        // This label informs the user of the login purpose
        JLabel adminLoginTxt = new JLabel("Admin Login");
        adminLoginTxt.setForeground(Color.WHITE);
        adminLoginTxt.setBounds(350, 50, 300, 50);
        adminLoginTxt.setFont(new Font("Arial", Font.BOLD, 36));
        loginPanel.add(adminLoginTxt);

        // Adding a login image
        // Enhances the visual appeal of the login screen
        ImageIcon loginImg = new ImageIcon("loginImg.png");
        JLabel loginImgLbl = new JLabel(loginImg);
        loginImgLbl.setBounds(200, 100, 550, 359);
        loginPanel.add(loginImgLbl);

        // Username label and field setup
        // Allows the user to input their username
        JLabel userLabel = new JLabel("Username");
        userLabel.setForeground(Color.WHITE);
        userLabel.setBounds(150, 500, 200, 35);
        userLabel.setFont(new Font("Arial", Font.PLAIN, 25));
        loginPanel.add(userLabel);

        userTxt = new JTextField();
        userTxt.setBounds(360, 500, 350, 35);
        userTxt.setFont(new Font("Arial", Font.PLAIN, 25));
        loginPanel.add(userTxt);

        // Password label and field setup
        // Securely collects the user's password
        JLabel passLabel = new JLabel("Password");
        passLabel.setForeground(Color.WHITE);
        passLabel.setBounds(150, 600, 200, 35);
        passLabel.setFont(new Font("Arial", Font.PLAIN, 25));
        loginPanel.add(passLabel);

        passwordTxt = new JPasswordField();
        passwordTxt.setBounds(360, 600, 350, 35);
        passwordTxt.setFont(new Font("Arial", Font.PLAIN, 25));
        loginPanel.add(passwordTxt);
```

```
        // Show password checkbox
        // Provides an option to toggle password visibility
        showPassword = new JCheckBox("Show Password");
        showPassword.setBounds(360, 660, 350, 35);
        showPassword.setFont(new Font("Arial", Font.PLAIN, 25));
        showPassword.setBackground(Color.BLACK);
        showPassword.setForeground(Color.WHITE);
        showPassword.addActionListener(this);
        loginPanel.add(showPassword);

        // Login button setup
        // Initiates the login process when clicked
        loginButton = new JButton("Login");
        loginButton.setBounds(360, 720, 200, 40);
        loginButton.setFont(new Font("Arial", Font.BOLD, 25));
        loginButton.setBackground(Color.GREEN);
        loginButton.addActionListener(this);
        buttonPointer(loginButton);
        loginPanel.add(loginButton);

        // Set login panel as the initial content
        // Displays the login screen when the app starts
        frame.setContentPane(loginPanel);

        // Button panel configuration
        // Contains navigation buttons for the main interface
        buttonPanel = new JPanel();
        buttonPanel.setLayout(null);
        buttonPanel.setBounds(0, 0, 200, 910);
        buttonPanel.setBackground(new Color(0x80490E));

        // Add member button
        // Navigates to the member addition panel
        addMemberBtn = new JButton("Add member");
        addMemberBtn.setBounds(10, 50, 180, 40);
        addMemberBtn.setFont(new Font("SansSerif", Font.BOLD, 15));
        addMemberBtn.setBackground(new Color(0xB4B8AC));
        addMemberBtn.addActionListener(this);
        buttonPointer(addMemberBtn);
        buttonPanel.add(addMemberBtn);

        // Activate/deactivate button
        // Accesses membership status controls
        activateBtn = new JButton("Activate/Deactivate");
        activateBtn.setBounds(10, 130, 180, 40);
        activateBtn.setFont(new Font("SansSerif", Font.BOLD, 15));
```

```
activateBtn.setBackground(new Color(0xB4B8AC));
activateBtn.addActionListener(this);
buttonPointer(activateBtn);
buttonPanel.add(activateBtn);

// Mark attendance button
// Opens the attendance marking panel
markAttendanceNavBtn = new JButton("Mark Attendance");
markAttendanceNavBtn.setBounds(10, 210, 180, 40);
markAttendanceNavBtn.setFont(new Font("SansSerif", Font.BOLD, 15));
markAttendanceNavBtn.setBackground(new Color(0xB4B8AC));
markAttendanceNavBtn.addActionListener(this);
buttonPointer(markAttendanceNavBtn);
buttonPanel.add(markAttendanceNavBtn);

// Upgrade plan button
// Allows plan upgrades for regular members
upgradeBtn = new JButton("Upgrade Plan");
upgradeBtn.setBounds(10, 290, 180, 40);
upgradeBtn.setFont(new Font("SansSerif", Font.BOLD, 15));
upgradeBtn.setBackground(new Color(0xB4B8AC));
upgradeBtn.addActionListener(this);
buttonPointer(upgradeBtn);
buttonPanel.add(upgradeBtn);

// Calculate discount button
// Computes discounts for premium members
discountBtn = new JButton("Calculate Discount");
discountBtn.setBounds(10, 370, 180, 40);
discountBtn.setFont(new Font("SansSerif", Font.BOLD, 15));
discountBtn.setBackground(new Color(0xB4B8AC));
discountBtn.addActionListener(this);
buttonPointer(discountBtn);
buttonPanel.add(discountBtn);

// Pay due button
// Manages payments for premium members
payDueNavBtn = new JButton("Pay Due");
payDueNavBtn.setBounds(10, 450, 180, 40);
payDueNavBtn.setFont(new Font("SansSerif", Font.BOLD, 15));
payDueNavBtn.setBackground(new Color(0xB4B8AC));
payDueNavBtn.addActionListener(this);
buttonPointer(payDueNavBtn);
buttonPanel.add(payDueNavBtn);

// Revert member button
```

```
// Handles member removal
revertMemberNavBtn = new JButton("Revert Member");
revertMemberNavBtn.setBounds(10, 530, 180, 40);
revertMemberNavBtn.setFont(new Font("SansSerif", Font.BOLD, 15));
revertMemberNavBtn.setBackground(new Color(0xB4B8AC));
revertMemberNavBtn.addActionListener(this);
buttonPointer(revertMemberNavBtn);
buttonPanel.add(revertMemberNavBtn);

// Save to file button
// Exports member data to a file
saveBtn = new JButton("Save to File");
saveBtn.setBounds(10, 610, 180, 40);
saveBtn.setFont(new Font("SansSerif", Font.BOLD, 15));
saveBtn.setBackground(new Color(0xB4B8AC));
saveBtn.addActionListener(this);
buttonPointer(saveBtn);
buttonPanel.add(saveBtn);

// Read from file button
// Imports member data from a file
readBtn = new JButton("Read from File");
readBtn.setBounds(10, 690, 180, 40);
readBtn.setFont(new Font("SansSerif", Font.BOLD, 15));
readBtn.setBackground(new Color(0xB4B8AC));
readBtn.addActionListener(this);
buttonPointer(readBtn);
buttonPanel.add(readBtn);

// Logout button
// Returns to the login screen
logOutBtn = new JButton("Log Out");
logOutBtn.setBounds(10, 770, 180, 40);
logOutBtn.setFont(new Font("SansSerif", Font.BOLD, 15));
logOutBtn.setBackground(new Color(0xB4B8AC));
logOutBtn.setForeground(Color.RED);
logOutBtn.addActionListener(this);
buttonPointer(logOutBtn);
buttonPanel.add(logOutBtn);

// Main panel setup
// Holds all sub-panels for different functionalities
mainPanel = new JPanel();
mainPanel.setLayout(null);
mainPanel.setBounds(0, 0, 690, 910);
mainPanel.setBackground(Color.BLACK);
```

```
// Home panel configuration
// The default panel shown after login
homePanel = new JPanel();
homePanel.setLayout(null);
homePanel.setBounds(0, 0, 690, 910);
homePanel.setBackground(Color.BLACK);

// Adding home images
// Decorative elements for the home screen
ImageIcon homeImg = new ImageIcon("homeImg.png");
JLabel homeImgLbl = new JLabel(homeImg);
homeImgLbl.setBounds(0, 370, 775, 500);
ImageIcon logoHomeImg = new ImageIcon("logoImg.png");

JLabel logoHomeImgLbl = new JLabel(logoHomeImg);
logoHomeImgLbl.setBounds(125, 50, 450, 450);
homePanel.add(homeImgLbl);
homePanel.add(logoHomeImgLbl);
mainPanel.add(homePanel);

// Add member panel setup
// Allows selection between regular and premium member addition
addMemberPanel = new JPanel();
addMemberPanel.setLayout(null);
addMemberPanel.setBounds(0, 0, 690, 910);
addMemberPanel.setBackground(Color.BLACK);

JLabel addMemLabel = new JLabel("Add Member");
addMemLabel.setForeground(Color.WHITE);
addMemLabel.setFont(new Font("Arial", Font.BOLD, 37));
addMemLabel.setBounds(225, 50, 400, 50);
addMemberPanel.add(addMemLabel);

addRegularSwitchBtn = new JButton("Add Regular Member");
addRegularSwitchBtn.setBounds(175, 180, 325, 40);
addRegularSwitchBtn.setFont(new Font("SansSerif", Font.BOLD, 20));
addRegularSwitchBtn.addActionListener(this);
buttonPointer(addRegularSwitchBtn);
addMemberPanel.add(addRegularSwitchBtn);

addPremiumSwitchBtn = new JButton("Add Premium Member");
addPremiumSwitchBtn.setBounds(175, 270, 325, 40);
addPremiumSwitchBtn.setFont(new Font("SansSerif", Font.BOLD, 20));
addPremiumSwitchBtn.addActionListener(this);
buttonPointer(addPremiumSwitchBtn);
```

```
addMemberPanel.add(addPremiumSwitchBtn);
mainPanel.add(addMemberPanel);

// Regular member panel configuration
// Form for adding regular members
regularPanel = new JPanel();
regularPanel.setLayout(null);
regularPanel.setBounds(0, 0, 690, 910);
regularPanel.setBackground(Color.BLACK);
JLabel regHeading = new JLabel("Regular Member");
regHeading.setForeground(Color.WHITE);
regHeading.setFont(new Font("Arial", Font.BOLD, 37));
regHeading.setBounds(225, 50, 400, 50);
regularPanel.add(regHeading);

// ID field setup
// Collects the unique identifier for the member
JLabel rIdLbl = new JLabel("Id");
rIdLbl.setForeground(Color.WHITE);
rIdLbl.setBounds(70, 120, 100, 30);
rIdLbl.setFont(new Font("Arial", Font.PLAIN, 16));
regularPanel.add(rIdLbl);
rIdField = new JTextField();
rIdField.setBounds(70, 150, 200, 24);
regularPanel.add(rIdField);

// Name field setup
// Stores the member's full name
JLabel rNameLbl = new JLabel("Name");
rNameLbl.setForeground(Color.WHITE);
rNameLbl.setBounds(350, 120, 100, 30);
rNameLbl.setFont(new Font("Arial", Font.PLAIN, 16));
regularPanel.add(rNameLbl);

rNameField = new JTextField();
rNameField.setBounds(350, 150, 200, 24);
regularPanel.add(rNameField);

// Location field setup
// Records the member's location
JLabel rLocLbl = new JLabel("Location");
rLocLbl.setForeground(Color.WHITE);
rLocLbl.setBounds(70, 190, 100, 30);
rLocLbl.setFont(new Font("Arial", Font.PLAIN, 16));
regularPanel.add(rLocLbl);
```

```
rLocationField = new JTextField();
rLocationField.setBounds(70, 220, 200, 24);
regularPanel.add(rLocationField);

// Phone field setup
// Captures the member's contact number
JLabel rPhoneLbl = new JLabel("Phone Number");
rPhoneLbl.setForeground(Color.WHITE);
rPhoneLbl.setBounds(350, 190, 150, 30);
rPhoneLbl.setFont(new Font("Arial", Font.PLAIN, 16));
regularPanel.add(rPhoneLbl);

rPhoneField = new JTextField();
rPhoneField.setBounds(350, 220, 200, 24);
regularPanel.add(rPhoneField);

// Email field setup
// Stores the member's email address
JLabel rEmailLbl = new JLabel("Email");
rEmailLbl.setForeground(Color.WHITE);
rEmailLbl.setBounds(70, 260, 100, 30);
rEmailLbl.setFont(new Font("Arial", Font.PLAIN, 16));
regularPanel.add(rEmailLbl);

rEmailField = new JTextField();
rEmailField.setBounds(70, 290, 350, 24);
regularPanel.add(rEmailField);

// Gender selection setup
// Allows specification of member gender
JLabel rGenderLbl = new JLabel("Gender");
rGenderLbl.setForeground(Color.WHITE);
rGenderLbl.setBounds(70, 330, 100, 30);
rGenderLbl.setFont(new Font("Arial", Font.PLAIN, 16));
regularPanel.add(rGenderLbl);

rMaleBtn = new JRadioButton("Male");
rFemaleBtn = new JRadioButton("Female");
rMaleBtn.setFont(new Font("Arial", Font.PLAIN, 16));
rFemaleBtn.setFont(new Font("Arial", Font.PLAIN, 16));
rMaleBtn.setBackground(Color.BLACK);
rFemaleBtn.setBackground(Color.BLACK);
rMaleBtn.setForeground(Color.WHITE);
rFemaleBtn.setForeground(Color.WHITE);
rMaleBtn.setBounds(70, 360, 100, 30);
rFemaleBtn.setBounds(180, 360, 100, 30);
```

```
ButtonGroup rGenderBG = new ButtonGroup();
rGenderBG.add(rMaleBtn);
rGenderBG.add(rFemaleBtn);

regularPanel.add(rMaleBtn);
regularPanel.add(rFemaleBtn);

// Date of birth setup
// Collects the member's birth date
JLabel rDobLbl = new JLabel("Date Of Birth (YYYY/MM/DD)");
rDobLbl.setForeground(Color.WHITE);
rDobLbl.setBounds(70, 400, 250, 30);
rDobLbl.setFont(new Font("Arial", Font.PLAIN, 16));
regularPanel.add(rDobLbl);

rDobYear = new JComboBox(yearsArr);
rDobMonth = new JComboBox(monthsArr);
rDobDay = new JComboBox(daysArr);
rDobYear.setBounds(70, 430, 80, 30);
rDobMonth.setBounds(170, 430, 60, 30);
rDobDay.setBounds(250, 430, 60, 30);

regularPanel.add(rDobYear);
regularPanel.add(rDobMonth);
regularPanel.add(rDobDay);

// Membership start date setup
// Records when the membership begins
JLabel rMsLbl = new JLabel("Member Start Date (YYYY/MM/DD)");
rMsLbl.setForeground(Color.WHITE);
rMsLbl.setBounds(70, 470, 300, 30);
rMsLbl.setFont(new Font("Arial", Font.PLAIN, 16));
regularPanel.add(rMsLbl);
rMsYear = new JComboBox(yearsArr);
rMsMonth = new JComboBox(monthsArr);
rMsDay = new JComboBox(daysArr);
rMsYear.setBounds(70, 500, 80, 30);
rMsMonth.setBounds(170, 500, 60, 30);
rMsDay.setBounds(250, 500, 60, 30);
regularPanel.add(rMsYear);
regularPanel.add(rMsMonth);
regularPanel.add(rMsDay);

// Referral field setup
// Captures referral information
JLabel rRefLbl = new JLabel("Referral");
```

```
rRefLbl.setForeground(Color.WHITE);
rRefLbl.setBounds(70, 540, 100, 30);
rRefLbl.setFont(new Font("Arial", Font.PLAIN, 16));
regularPanel.add(rRefLbl);
rReferralField = new JTextField();
rReferralField.setBounds(70, 570, 200, 24);
regularPanel.add(rReferralField);

// Display button
// Shows all regular member details
rDisplayBtn = new JButton("Display");
rDisplayBtn.setBounds(70, 680, 175, 40);
rDisplayBtn.setFont(new Font("Arial", Font.BOLD, 16));
rDisplayBtn.setBackground(new Color(0xB4B8AC));
rDisplayBtn.addActionListener(this);
buttonPointer(rDisplayBtn);
regularPanel.add(rDisplayBtn);

// Clear button
// Resets the form fields
rClearBtn = new JButton("Clear");
rClearBtn.setBounds(350, 680, 175, 40);
rClearBtn.setFont(new Font("Arial", Font.BOLD, 16));
rClearBtn.setBackground(new Color(0xB4B8AC));
rClearBtn.addActionListener(this);
buttonPointer(rClearBtn);
regularPanel.add(rClearBtn);

// Add member button
// Submits the regular member data
rAddMemberBtn = new JButton("Add member");
rAddMemberBtn.setBounds(475, 800, 175, 40);
rAddMemberBtn.setBackground(Color.GREEN);
rAddMemberBtn.setFont(new Font("Arial", Font.BOLD, 16));
rAddMemberBtn.addActionListener(this);
buttonPointer(rAddMemberBtn);
regularPanel.add(rAddMemberBtn);

mainPanel.add(regularPanel);

// Premium member panel configuration
// Form for adding premium members
premiumPanel = new JPanel();
premiumPanel.setLayout(null);
premiumPanel.setBounds(0, 0, 690, 910);
premiumPanel.setBackground(Color.BLACK);
```

```
JLabel preHeading = new JLabel("Premium Member");

preHeading.setForeground(Color.WHITE);
preHeading.setFont(new Font("Arial", Font.BOLD, 37));
preHeading.setBounds(225, 50, 400, 50);
premiumPanel.add(preHeading);

// ID field for premium member
JLabel pIdLbl = new JLabel("Id");
pIdLbl.setForeground(Color.WHITE);
pIdLbl.setBounds(70, 120, 100, 30);
pIdLbl.setFont(new Font("Arial", Font.PLAIN, 16));
premiumPanel.add(pIdLbl);

pIdField = new JTextField();
pIdField.setBounds(70, 150, 200, 24);
premiumPanel.add(pIdField);

// Name field for premium member
JLabel pNameLbl = new JLabel("Name");
pNameLbl.setForeground(Color.WHITE);
pNameLbl.setBounds(350, 120, 100, 30);
pNameLbl.setFont(new Font("Arial", Font.PLAIN, 16));
premiumPanel.add(pNameLbl);

pNameField = new JTextField();
pNameField.setBounds(350, 150, 200, 24);
premiumPanel.add(pNameField);

// Location field for premium member
JLabel pLocLbl = new JLabel("Location");
pLocLbl.setForeground(Color.WHITE);
pLocLbl.setBounds(70, 190, 100, 30);
pLocLbl.setFont(new Font("Arial", Font.PLAIN, 16));
premiumPanel.add(pLocLbl);

pLocationField = new JTextField();
pLocationField.setBounds(70, 220, 200, 24);
premiumPanel.add(pLocationField);

// Phone field for premium member
JLabel pPhoneLbl = new JLabel("Phone Number");
pPhoneLbl.setForeground(Color.WHITE);
pPhoneLbl.setBounds(350, 190, 150, 30);
pPhoneLbl.setFont(new Font("Arial", Font.PLAIN, 16));
premiumPanel.add(pPhoneLbl);
```

```
pPhoneField = new JTextField();
pPhoneField.setBounds(350, 220, 200, 24);
premiumPanel.add(pPhoneField);

// Email field for premium member
JLabel pEmailLbl = new JLabel("Email");
pEmailLbl.setForeground(Color.WHITE);
pEmailLbl.setBounds(70, 260, 100, 30);
pEmailLbl.setFont(new Font("Arial", Font.PLAIN, 16));
premiumPanel.add(pEmailLbl);

pEmailField = new JTextField();
pEmailField.setBounds(70, 290, 350, 24);
premiumPanel.add(pEmailField);

// Gender selection for premium member
JLabel pGenderLbl = new JLabel("Gender");
pGenderLbl.setForeground(Color.WHITE);
pGenderLbl.setBounds(70, 330, 100, 30);
pGenderLbl.setFont(new Font("Arial", Font.PLAIN, 16));
premiumPanel.add(pGenderLbl);

pMaleBtn = new JRadioButton("Male");
pFemaleBtn = new JRadioButton("Female");
pMaleBtn.setFont(new Font("Arial", Font.PLAIN, 16));
pFemaleBtn.setFont(new Font("Arial", Font.PLAIN, 16));
pMaleBtn.setBackground(Color.BLACK);
pFemaleBtn.setBackground(Color.BLACK);
pMaleBtn.setForeground(Color.WHITE);
pFemaleBtn.setForeground(Color.WHITE);
pMaleBtn.setBounds(70, 360, 100, 30);
pFemaleBtn.setBounds(180, 360, 100, 30);
ButtonGroup pGenderBG = new ButtonGroup();

pGenderBG.add(pMaleBtn);
pGenderBG.add(pFemaleBtn);
premiumPanel.add(pMaleBtn);
premiumPanel.add(pFemaleBtn);

// Date of birth for premium member
JLabel pDobLbl = new JLabel("Date Of Birth (YYYY/MM/DD)");
pDobLbl.setForeground(Color.WHITE);
pDobLbl.setBounds(70, 400, 250, 30);
pDobLbl.setFont(new Font("Arial", Font.PLAIN, 16));
premiumPanel.add(pDobLbl);
```

```
pDobYear = new JComboBox(yearsArr);
pDobMonth = new JComboBox(monthsArr);
pDobDay = new JComboBox(daysArr);
pDobYear.setBounds(70, 430, 80, 30);
pDobMonth.setBounds(170, 430, 60, 30);
pDobDay.setBounds(250, 430, 60, 30);
premiumPanel.add(pDobYear);
premiumPanel.add(pDobMonth);
premiumPanel.add(pDobDay);

// Membership start date for premium member
JLabel pMsLbl = new JLabel("Member Start Date (YYYY/MM/DD)");
pMsLbl.setForeground(Color.WHITE);
pMsLbl.setBounds(70, 470, 300, 30);
pMsLbl.setFont(new Font("Arial", Font.PLAIN, 16));
premiumPanel.add(pMsLbl);

pMsYear = new JComboBox(yearsArr);
pMsMonth = new JComboBox(monthsArr);
pMsDay = new JComboBox(daysArr);

pMsYear.setBounds(70, 500, 80, 30);
pMsMonth.setBounds(170, 500, 60, 30);
pMsDay.setBounds(250, 500, 60, 30);

premiumPanel.add(pMsYear);
premiumPanel.add(pMsMonth);
premiumPanel.add(pMsDay);

// Trainer field for premium member
// Specific to premium members for personal training
JLabel tLbl = new JLabel("Trainer");
tLbl.setForeground(Color.WHITE);
tLbl.setBounds(70, 540, 100, 30);
tLbl.setFont(new Font("Arial", Font.PLAIN, 16));
premiumPanel.add(tLbl);

pTrainerField = new JTextField();
pTrainerField.setBounds(70, 570, 200, 24);
premiumPanel.add(pTrainerField);

// Charge field for premium member
// Displays the fixed premium membership cost
JLabel chargeLbl = new JLabel("Premium Charge");
chargeLbl.setForeground(Color.WHITE);
```

```
chargeLbl.setBounds(350, 540, 200, 30);
chargeLbl.setFont(new Font("Arial", Font.PLAIN, 16));
premiumPanel.add(chargeLbl);

pChargeField = new JTextField("50000");
pChargeField.setBounds(350, 570, 150, 24);
pChargeField.setEditable(false);
premiumPanel.add(pChargeField);

// Display button for premium members
pDisplayBtn = new JButton("Display");
pDisplayBtn.setBounds(70, 650, 175, 40);
pDisplayBtn.setFont(new Font("Arial", Font.BOLD, 18));
pDisplayBtn.setBackground(new Color(0xB4B8AC));
pDisplayBtn.addActionListener(this);
buttonPointer(pDisplayBtn);
premiumPanel.add(pDisplayBtn);

// Clear button for premium members
pClearBtn = new JButton("Clear");
pClearBtn.setBounds(350, 650, 175, 40);
pClearBtn.setFont(new Font("Arial", Font.BOLD, 18));
pClearBtn.setBackground(new Color(0xB4B8AC));
pClearBtn.addActionListener(this);
buttonPointer(pClearBtn);
premiumPanel.add(pClearBtn);

// Add member button for premium members
pAddMemberBtn = new JButton("Add member");
pAddMemberBtn.setBounds(475, 800, 175, 40);
pAddMemberBtn.setBackground(Color.GREEN);
pAddMemberBtn.setFont(new Font("Arial", Font.BOLD, 18));
pAddMemberBtn.addActionListener(this);
buttonPointer(pAddMemberBtn);
premiumPanel.add(pAddMemberBtn);

mainPanel.add(premiumPanel);

// Activate/deactivate panel setup
// Manages membership status changes
activateDeactivatePanel = new JPanel();
activateDeactivatePanel.setLayout(null);
activateDeactivatePanel.setBounds(0, 0, 690, 910);
activateDeactivatePanel.setBackground(Color.BLACK);
JLabel actHeading = new JLabel("Activate/Deactivate Membership");
actHeading.setForeground(Color.WHITE);
```

```
actHeading.setBounds(110, 50, 500, 50);
actHeading.setFont(new Font("Arial", Font.BOLD, 30));
activateDeactivatePanel.add(actHeading);

// ID field for activation/deactivation
JLabel actIdLbl = new JLabel("Member Id");
actIdLbl.setForeground(Color.WHITE);
actIdLbl.setBounds(100, 150, 100, 30);
actIdLbl.setFont(new Font("Arial", Font.PLAIN, 16));
activateDeactivatePanel.add(actIdLbl);

actIdField = new JTextField();
actIdField.setBounds(100, 180, 200, 24);
activateDeactivatePanel.add(actIdField);

// Activate button
activateMemberBtn = new JButton("Activate");
activateMemberBtn.setBounds(100, 250, 175, 40);
activateMemberBtn.setFont(new Font("Arial", Font.BOLD, 18));
activateMemberBtn.setBackground(Color.GREEN);
activateMemberBtn.addActionListener(this);
buttonPointer(activateMemberBtn);
activateDeactivatePanel.add(activateMemberBtn);

// Deactivate button
deactivateMemberBtn = new JButton("Deactivate");
deactivateMemberBtn.setBounds(360, 250, 175, 40);
deactivateMemberBtn.setFont(new Font("Arial", Font.BOLD, 18));
deactivateMemberBtn.setBackground(Color.RED);
deactivateMemberBtn.setForeground(Color.WHITE);
deactivateMemberBtn.addActionListener(this);
buttonPointer(deactivateMemberBtn);
activateDeactivatePanel.add(deactivateMemberBtn);
mainPanel.add(activateDeactivatePanel);

// Mark attendance panel setup
// Tracks member attendance
markAttendancePanel = new JPanel();
markAttendancePanel.setLayout(null);
markAttendancePanel.setBounds(0, 0, 690, 910);
markAttendancePanel.setBackground(Color.BLACK);
JLabel markHeading = new JLabel("Mark Attendance");
markHeading.setForeground(Color.WHITE);
markHeading.setFont(new Font("Arial", Font.BOLD, 37));
markHeading.setBounds(225, 50, 400, 50);
markAttendancePanel.add(markHeading);
```

```
// ID field for attendance
JLabel markIdLbl = new JLabel("Member Id");
markIdLbl.setForeground(Color.WHITE);
markIdLbl.setBounds(100, 150, 100, 30);
markIdLbl.setFont(new Font("Arial", Font.PLAIN, 16));
markAttendancePanel.add(markIdLbl);

markIdField = new JTextField();
markIdField.setBounds(100, 180, 200, 24);
markAttendancePanel.add(markIdField);

// Mark regular attendance button
markRegularBtn = new JButton("Mark Regular");
markRegularBtn.setBounds(100, 250, 175, 40);
markRegularBtn.setFont(new Font("Arial", Font.BOLD, 18));
markRegularBtn.setBackground(new Color(0xB4B8AC));
markRegularBtn.addActionListener(this);
buttonPointer(markRegularBtn);
markAttendancePanel.add(markRegularBtn);

// Mark premium attendance button
markPremiumBtn = new JButton("Mark Premium");
markPremiumBtn.setBounds(360, 250, 175, 40);
markPremiumBtn.setFont(new Font("Arial", Font.BOLD, 18));
markPremiumBtn.setBackground(new Color(0xB4B8AC));
markPremiumBtn.addActionListener(this);
buttonPointer(markPremiumBtn);
markAttendancePanel.add(markPremiumBtn);

mainPanel.add(markAttendancePanel);

// Upgrade plan panel setup
// Facilitates plan upgrades
upgradePlanPanel = new JPanel();
upgradePlanPanel.setLayout(null);
upgradePlanPanel.setBounds(0, 0, 690, 910);
upgradePlanPanel.setBackground(Color.BLACK);

JLabel upHeading = new JLabel("Upgrade Plan");
upHeading.setForeground(Color.WHITE);
upHeading.setFont(new Font("Arial", Font.BOLD, 37));
upHeading.setBounds(225, 50, 400, 50);
upgradePlanPanel.add(upHeading);

// ID field for upgrade
```

```
JLabel upIdLbl = new JLabel("Member Id");
upIdLbl.setForeground(Color.WHITE);
upIdLbl.setBounds(100, 150, 100, 30);
upIdLbl.setFont(new Font("Arial", Font.PLAIN, 16));
upgradePlanPanel.add(upIdLbl);

upIdField = new JTextField();
upIdField.setBounds(100, 180, 200, 24);
upgradePlanPanel.add(upIdField);

// Plan selection for upgrade
JLabel upPlanLbl = new JLabel("New Plan");
upPlanLbl.setForeground(Color.WHITE);
upPlanLbl.setBounds(100, 230, 100, 30);
upPlanLbl.setFont(new Font("Arial", Font.PLAIN, 16));
upgradePlanPanel.add(upPlanLbl);

upPlanBox = new JComboBox(planOptionsArr);
upPlanBox.setBounds(100, 260, 125, 30);
upgradePlanPanel.add(upPlanBox);

// Upgrade button
upgradePlanBtn = new JButton("Upgrade");
upgradePlanBtn.setBounds(100, 320, 175, 40);
upgradePlanBtn.setFont(new Font("Arial", Font.BOLD, 18));
upgradePlanBtn.setBackground(new Color(0xB4B8AC));
upgradePlanBtn.addActionListener(this);
buttonPointer(upgradePlanBtn);
upgradePlanPanel.add(upgradePlanBtn);
mainPanel.add(upgradePlanPanel);

// Discount panel setup
// Calculates discounts for premium members
discountPanel = new JPanel();
discountPanel.setLayout(null);
discountPanel.setBounds(0, 0, 690, 910);
discountPanel.setBackground(Color.BLACK);

JLabel discHeading = new JLabel("Calculate Discount");
discHeading.setForeground(Color.WHITE);
discHeading.setFont(new Font("Arial", Font.BOLD, 37));
discHeading.setBounds(225, 50, 400, 50);
discountPanel.add(discHeading);

// ID field for discount
JLabel discIdLbl = new JLabel("Member Id");
```

```
discIdLbl.setForeground(Color.WHITE);
discIdLbl.setBounds(100, 150, 100, 30);
discIdLbl.setFont(new Font("Arial", Font.PLAIN, 16));
discountPanel.add(discIdLbl);

discIdField = new JTextField();
discIdField.setBounds(100, 180, 200, 24);
discountPanel.add(discIdField);

// Calculate discount button
calcDiscountBtn = new JButton("Calculate");
calcDiscountBtn.setBounds(100, 240, 175, 40);
calcDiscountBtn.setFont(new Font("Arial", Font.BOLD, 18));
calcDiscountBtn.setBackground(new Color(0xB4B8AC));
calcDiscountBtn.addActionListener(this);
buttonPointer(calcDiscountBtn);
discountPanel.add(calcDiscountBtn);

// Discount amount display
JLabel dAmtLbl = new JLabel("Discount Amount");
dAmtLbl.setForeground(Color.WHITE);
dAmtLbl.setBounds(100, 300, 200, 30);
dAmtLbl.setFont(new Font("Arial", Font.PLAIN, 16));
discountPanel.add(dAmtLbl);
discAmountField = new JTextField();
discAmountField.setBounds(100, 330, 200, 24);
discAmountField.setEditable(false);
discountPanel.add(discAmountField);
mainPanel.add(discountPanel);

// Pay due panel setup
// Manages due payments
payDuePanel = new JPanel();
payDuePanel.setLayout(null);
payDuePanel.setBounds(0, 0, 690, 910);
payDuePanel.setBackground(Color.BLACK);

JLabel payHeading = new JLabel("Pay Due Amount");
payHeading.setForeground(Color.WHITE);
payHeading.setFont(new Font("Arial", Font.BOLD, 37));
payHeading.setBounds(225, 50, 400, 50);
payDuePanel.add(payHeading);

// ID field for payment
JLabel payIdLbl = new JLabel("Member Id");
payIdLbl.setForeground(Color.WHITE);
```

```
payIdLbl.setBounds(100, 150, 100, 30);
payIdLbl.setFont(new Font("Arial", Font.PLAIN, 16));
payDuePanel.add(payIdLbl);

payIdField = new JTextField();
payIdField.setBounds(100, 180, 200, 24);
payDuePanel.add(payIdField);

// Amount field for payment
JLabel payAmtLbl = new JLabel("Amount:");
payAmtLbl.setForeground(Color.WHITE);
payAmtLbl.setBounds(100, 240, 100, 30);
payAmtLbl.setFont(new Font("Arial", Font.PLAIN, 16));
payDuePanel.add(payAmtLbl);

payAmountField = new JTextField();
payAmountField.setBounds(100, 270, 200, 24);
payDuePanel.add(payAmountField);

// Pay button
payDueButton = new JButton("Pay Amount");
payDueButton.setBounds(100, 330, 175, 40);
payDueButton.setFont(new Font("Arial", Font.BOLD, 18));
payDueButton.setBackground(new Color(0xB4B8AC));
payDueButton.addActionListener(this);
buttonPointer(payDueButton);
payDuePanel.add(payDueButton);
mainPanel.add(payDuePanel);

// Revert member panel setup
// Handles member reversion
revertMemberPanel = new JPanel();
revertMemberPanel.setLayout(null);
revertMemberPanel.setBounds(0, 0, 690, 910);
revertMemberPanel.setBackground(Color.BLACK);

JLabel revHeading = new JLabel("Revert Member");
revHeading.setForeground(Color.WHITE);
revHeading.setFont(new Font("Arial", Font.BOLD, 37));
revHeading.setBounds(225, 50, 400, 50);
revertMemberPanel.add(revHeading);

// ID field for reversion
JLabel revIdLbl = new JLabel("Member Id");
revIdLbl.setForeground(Color.WHITE);
revIdLbl.setBounds(100, 150, 100, 30);
```

```
        revIdLbl.setFont(new Font("Arial", Font.PLAIN, 16));
        revertMemberPanel.add(revIdLbl);

        revertIdField = new JTextField();
        revertIdField.setBounds(100, 180, 200, 24);
        revertMemberPanel.add(revertIdField);

        // Revert regular button
        revertRegularBtn = new JButton("Revert Regular");
        revertRegularBtn.setBounds(100, 250, 175, 40);
        revertRegularBtn.setFont(new Font("Arial", Font.BOLD, 18));
        revertRegularBtn.setBackground(new Color(0xB4B8AC));
        revertRegularBtn.addActionListener(this);
        buttonPointer(revertRegularBtn);
        revertMemberPanel.add(revertRegularBtn);

        // Revert premium button
        revertPremiumBtn = new JButton("Revert Premium");
        revertPremiumBtn.setBounds(360, 250, 175, 40);
        revertPremiumBtn.setFont(new Font("Arial", Font.BOLD, 18));
        revertPremiumBtn.setBackground(new Color(0xB4B8AC));
        revertPremiumBtn.addActionListener(this);
        buttonPointer(revertPremiumBtn);
        revertMemberPanel.add(revertPremiumBtn);
        mainPanel.add(revertMemberPanel);

        // Final frame setup
        // Adds the main panel and makes the frame visible
        frame.add(mainPanel);
        frame.setVisible(true);
        hideAllSubPanels();
    }

    // Method to hide all sub-panels
    public void hideAllSubPanels()
    {
        homePanel.setVisible(false); // Hiding the panel as setting visible to false
        addMemberPanel.setVisible(false); // Hiding the panel as setting visible to false
        regularPanel.setVisible(false); // Hiding the panel as setting visible to false
        premiumPanel.setVisible(false); // Hiding the panel as setting visible to false
        activateDeactivatePanel.setVisible(false); // Hiding the panel as setting visible to
false
        markAttendancePanel.setVisible(false); // Hiding the panel as setting visible to
false
        upgradePlanPanel.setVisible(false); // Hiding the panel as setting visible to false
        discountPanel.setVisible(false); // Hiding the panel as setting visible to false
```

```
        revertMemberPanel.setVisible(false); // Hiding the panel as setting visible to false
        payDuePanel.setVisible(false); // Hiding the panel as setting visible to false
    }

    // Button pointer method explanation
    // Enhances button usability by setting cursor and focus properties
    private void buttonPointer(JButton button)
    {
        button.setFocusable(false);
        button.setCursor(new Cursor(Cursor.HAND_CURSOR));
    }

    // Find member by ID method
    // Searches the ArrayList for a member with the given ID
    private GymMember searchMemId(int id)
    {
        // Using a for-each loop to iterate through the ArrayList of members
        for (GymMember gm : membersObj)
        {
            if (gm.getId() == id)
            {
                return gm;
            }
        }
        return null;
    }

    // Duplicate ID check method
    // Verifies if an ID is already in use
    private boolean isDuplicateId(int id)
    {
        return searchMemId(id) != null;
    }

    // Action listener implementation
    // Handles all button clicks and events in the GUI
    @Override
    public void actionPerformed(ActionEvent eve)
    {
        Object btn = eve.getSource();

        // Login button handler
        // Authenticates the admin user
        if (btn == loginButton)
        {
            String user = userTxt.getText();
```

```
            String pass = new String(passwordTxt.getPassword());
            if (user.equals("admin") && pass.equals("admin"))
            {
                JPanel container = new JPanel(null);
                container.setBounds(0, 0, 890, 910);
                container.add(buttonPanel);
                container.add(mainPanel);
                mainPanel.setBounds(200, 0, 690, 910);
                frame.setContentPane(container);
                frame.repaint();
                frame.revalidate();
                hideAllSubPanels();
                homePanel.setVisible(true);
            }
            else if (user.equals("") || pass.equals(""))
            {
                JOptionPane.showMessageDialog(frame, "Please fill up both text areas!",
"Login Message", JOptionPane.INFORMATION_MESSAGE);
            }
            else
            {
                JOptionPane.showMessageDialog(frame, "Invalid username or password!",
"Login Message", JOptionPane.ERROR_MESSAGE);
            }
        }
        // Show password checkbox handler
        // Toggles password visibility
        else if (btn == showPassword)
        {
            if (showPassword.isSelected())
            {
                passwordTxt.setEchoChar((char) 0);
            }
            else
            {
                passwordTxt.setEchoChar('*');
            }
        }
        // Navigation button handlers
        // Switch between different panels
        if (btn == addMemberBtn)
        {
            homePanel.setVisible(false); // Hiding the panel as setting visible to false
            regularPanel.setVisible(false); // Hiding the panel as setting visible to false
            premiumPanel.setVisible(false); // Hiding the panel as setting visible to false
```

```
            activateDeactivatePanel.setVisible(false); // Hiding the panel as setting visible to
false
            markAttendancePanel.setVisible(false); // Hiding the panel as setting visible to
false
            upgradePlanPanel.setVisible(false); // Hiding the panel as setting visible to false
            discountPanel.setVisible(false); // Hiding the panel as setting visible to false
            revertMemberPanel.setVisible(false); // Hiding the panel as setting visible to
false
            payDuePanel.setVisible(false); // Hiding the panel as setting visible to false
            addMemberPanel.setVisible(true);
        }
        if (btn == activateBtn)
        {
            hideAllSubPanels();
            activateDeactivatePanel.setVisible(true);
        }
        if (btn == markAttendanceNavBtn)
        {
            hideAllSubPanels();
            markAttendancePanel.setVisible(true);
        }
        if (btn == upgradeBtn)
        {
            hideAllSubPanels();
            upgradePlanPanel.setVisible(true);
        }
        if (btn == discountBtn)
        {
            hideAllSubPanels();
            discountPanel.setVisible(true);
        }
        if (btn == payDueNavBtn)
        {
            hideAllSubPanels();
            payDuePanel.setVisible(true);
        }
        if (btn == revertMemberNavBtn)
        {
            hideAllSubPanels();
            revertMemberPanel.setVisible(true);
        }
        if (btn == logOutBtn)
        {
            frame.getContentPane().removeAll();
            frame.setContentPane(loginPanel);
            frame.repaint();
```

```
            frame.revalidate();
        }
        if (btn == addRegularSwitchBtn)
        {
            hideAllSubPanels();
            regularPanel.setVisible(true);
        }
        if (btn == addPremiumSwitchBtn)
        {
            hideAllSubPanels();
            premiumPanel.setVisible(true);
        }

        // Add regular member handler
        // Validates and adds a regular member
        if (btn == rAddMemberBtn)
        {
            if (rIdField.getText().isEmpty() || rNameField.getText().isEmpty() ||
rLocationField.getText().isEmpty() || rPhoneField.getText().isEmpty() ||
            rEmailField.getText().isEmpty() || rReferralField.getText().isEmpty() ||
(!rMaleBtn.isSelected() && !rFemaleBtn.isSelected()))
            {
                JOptionPane.showMessageDialog(frame, "Please fill in all required fields for
Regular Member.");
                return;
            }
            String phoneStr = rPhoneField.getText().trim();
            try
            {
                Long.parseLong(phoneStr);
            }
            catch (NumberFormatException e)
            {
                JOptionPane.showMessageDialog(frame, "Phone number must contain
number only.", "Warning", JOptionPane.WARNING_MESSAGE);
                return;
            }
            if (phoneStr.length() != 10)
            {
                JOptionPane.showMessageDialog(frame, "Phone number must be of 10
digits.", "Warning", JOptionPane.WARNING_MESSAGE);
                return;
            }
            try
            {
                id = Integer.parseInt(rIdField.getText().trim());
```

```
            if (isDuplicateId(id))
            {
                JOptionPane.showMessageDialog(frame, "Member with this ID already
exists!");
                return;
            }
            name = rNameField.getText().trim();
            location = rLocationField.getText().trim();
            phone = phoneStr;
            email = rEmailField.getText().trim();
            gender = (rMaleBtn.isSelected()) ? "Male" : "Female";
            dob = rDobYear.getSelectedItem() + "/" + rDobMonth.getSelectedItem() + "/"
+ rDobDay.getSelectedItem();
            msd = rMsYear.getSelectedItem() + "/" + rMsMonth.getSelectedItem() + "/" +
rMsDay.getSelectedItem();
            referral = rReferralField.getText().trim();
            RegularMember rMemObj = new RegularMember(id, name, location, phone,
email, gender, dob, msd, referral);
            // Upcasting: Adding RegularMember to ArrayList<GymMember>
            rMemObj.setPlan("Basic");
            membersObj.add(rMemObj);
            JOptionPane.showMessageDialog(frame, "Regular Member added
successfully!");
        }
        catch (NumberFormatException ex)
        {
            JOptionPane.showMessageDialog(frame, "Character value detected. Please
input number only!");
        }
    }
    // Display regular members handler
    // Shows all regular member details
    if (btn == rDisplayBtn)
    {
        if (membersObj.isEmpty())
        {
            JOptionPane.showMessageDialog(frame, "No members available!");
            return;
        }
        JFrame rDisplayInfoFrame = new JFrame("Regular Member Display");
        rDisplayInfoFrame.setSize(320, 730);
        rDisplayInfoFrame.setLayout(null);
        JTextArea textArea = new JTextArea();
        textArea.setEditable(false);
        textArea.setFont(new Font("Arial", Font.PLAIN, 14)); // Increased font size
slightly
```

```
            String display = "";
            // Using a for-each loop to iterate through the ArrayList of members
            for (GymMember gm : membersObj)
            {
                // Downcasting: Converting GymMember to RegularMember to access
specific methods
                if (gm instanceof RegularMember)
                {
                    RegularMember rm = (RegularMember) gm;
                    display += "ID: " + rm.getId() + "\n";
                    display += "Name: " + rm.getName() + "\n";
                    display += "Location: " + rm.getLocation() + "\n";
                    display += "Phone: " + rm.getPhone() + "\n";
                    display += "Email: " + rm.getEmail() + "\n";
                    display += "Gender: " + rm.getGender() + "\n";
                    display += "DOB: " + rm.getDOB() + "\n";
                    display += "Membership Start Date: " + rm.getMembershipStartDate() +
"\n";
                    display += "Attendance: " + rm.getAttendance() + "\n";
                    display += "Loyalty Points: " + rm.getLoyaltyPoints() + "\n";
                    display += "Active Status: " + rm.getActiveStatus() + "\n";
                    display += "Plan: " + rm.getPlan() + "\n";
                    display += "Price: " + rm.getPrice() + "\n";
                    display += "Referral Source: " + rm.getReferralSource() + "\n";
                    if (!rm.getRemovalReason().isEmpty())
                    {
                        display += "Removal Reason: " + rm.getRemovalReason() + "\n";
                    }
                    display += "----------------------------\n";
                }
            }
            textArea.setText(display);
            JScrollPane scrollPane = new JScrollPane(textArea);
            scrollPane.setBounds(10, 10, 300, 690); // Scroll bar spans top to bottom
            rDisplayInfoFrame.setResizable(false);
            rDisplayInfoFrame.add(scrollPane);
            rDisplayInfoFrame.setVisible(true);
        }
        // Clear regular member fields handler
        // Resets the regular member form
        else if (btn == rClearBtn)
        {
            rIdField.setText("");
            rNameField.setText("");
            rLocationField.setText("");
            rPhoneField.setText("");
```

```
            rEmailField.setText("");
            rReferralField.setText("");
            rMaleBtn.setSelected(false);
            rFemaleBtn.setSelected(false);
            rDobYear.setSelectedIndex(0);
            rDobMonth.setSelectedIndex(0);
            rDobDay.setSelectedIndex(0);
            rMsYear.setSelectedIndex(0);
            rMsMonth.setSelectedIndex(0);
            rMsDay.setSelectedIndex(0);
        }
        // Add premium member handler
        // Validates and adds a premium member
        else if (btn == pAddMemberBtn)
        {
            if (pIdField.getText().isEmpty() || pNameField.getText().isEmpty() ||
pLocationField.getText().isEmpty() || pPhoneField.getText().isEmpty() ||
            pEmailField.getText().isEmpty() || pTrainerField.getText().isEmpty() ||
(!pMaleBtn.isSelected() && !pFemaleBtn.isSelected()))
            {
                JOptionPane.showMessageDialog(frame, "Please fill in all required fields for
Premium Member.");
                return;
            }
            String phoneStr = pPhoneField.getText().trim();
            try
            {
                Long.parseLong(phoneStr);
            }
            catch (NumberFormatException e)
            {
                JOptionPane.showMessageDialog(frame, "Phone number must contain
number only.", "Warning", JOptionPane.WARNING_MESSAGE);
                return;
            }
            if (phoneStr.length() != 10)
            {
                JOptionPane.showMessageDialog(frame, "Phone number must be of 10
digits.", "Warning", JOptionPane.WARNING_MESSAGE);
                return;
            }
            try
            {
                id = Integer.parseInt(pIdField.getText().trim());
                if (isDuplicateId(id))
                {
```

```
                JOptionPane.showMessageDialog(frame, "Member with this ID already
exists!");
                return;
            }
            name = pNameField.getText().trim();
            location = pLocationField.getText().trim();
            phone = phoneStr;
            email = pEmailField.getText().trim();
            gender = (pMaleBtn.isSelected()) ? "Male" : "Female";
            dob = pDobYear.getSelectedItem() + "/" + pDobMonth.getSelectedItem() + "/"
+ pDobDay.getSelectedItem();
            msd = pMsYear.getSelectedItem() + "/" + pMsMonth.getSelectedItem() + "/" +
pMsDay.getSelectedItem();
            trainer = pTrainerField.getText().trim();
            PremiumMember pMemObj = new PremiumMember(id, name, location,
phone, email, gender, dob, msd, trainer);
            // Upcasting: Adding PremiumMember to ArrayList<GymMember>
            membersObj.add(pMemObj);
            JOptionPane.showMessageDialog(frame, "Premium Member added
successfully!");
        }
        catch (NumberFormatException ex)
        {
            JOptionPane.showMessageDialog(frame, "Character value detected. Please
input number only!");
        }
    }
    // Display premium members handler
    // Shows all premium member details
    else if (btn == pDisplayBtn)
    {
        if (membersObj.isEmpty())
        {
            JOptionPane.showMessageDialog(frame, "No members available!");
            return;
        }
        JFrame pDisplayInfoFrame = new JFrame("Premium Member Display");
        pDisplayInfoFrame.setSize(320, 730);
        pDisplayInfoFrame.setLayout(null);
        JTextArea textArea = new JTextArea();
        textArea.setEditable(false);
        textArea.setFont(new Font("Arial", Font.PLAIN, 14)); // Increased font size
slightly
        String display = "";
        // Using a for-each loop to iterate through the ArrayList of members
        for (GymMember gm : membersObj)
```

```
        {
            // Downcasting: Converting GymMember to PremiumMember to access
specific methods
            if (gm instanceof PremiumMember)
            {
                PremiumMember pm = (PremiumMember) gm;
                double remaining = pm.getPremiumCharge() - pm.getPaidAmount();
                boolean full = pm.isFullPayment();
                double discount = pm.getDiscountAmount();
                double paidAmt = full ? (pm.getPremiumCharge() - discount) :
pm.getPaidAmount();
                display += "ID: " + pm.getId() + "\n";
                display += "Name: " + pm.getName() + "\n";
                display += "Location: " + pm.getLocation() + "\n";
                display += "Phone: " + pm.getPhone() + "\n";
                display += "Email: " + pm.getEmail() + "\n";
                display += "Gender: " + pm.getGender() + "\n";
                display += "DOB: " + pm.getDOB() + "\n";
                display += "Membership Start Date: " + pm.getMembershipStartDate() +
"\n";
                display += "Attendance: " + pm.getAttendance() + "\n";
                display += "Loyalty Points: " + pm.getLoyaltyPoints() + "\n";
                display += "Active Status: " + pm.getActiveStatus() + "\n";
                display += "Personal Trainer" + pm.getPersonalTrainer() + "\n";
                display += "Premium Charge: " + pm.getPremiumCharge() + "\n";
                display += "Paid Amount: " + paidAmt + "\n";
                display += "Remaining Amount: " + (full ? 0 : remaining) + "\n";
                display += "Discount Amount: " + discount + "\n";
                display += "----------------------------\n";
            }
        }
        textArea.setText(display);
        JScrollPane scrollPane = new JScrollPane(textArea);
        scrollPane.setBounds(10, 10, 300, 690); // Scroll bar spans top to bottom
        pDisplayInfoFrame.setResizable(false);
        pDisplayInfoFrame.add(scrollPane);
        pDisplayInfoFrame.setVisible(true);
    }
    // Clear premium member fields handler
    // Resets the premium member form
    else if (btn == pClearBtn)
    {
        pIdField.setText("");
        pNameField.setText("");
        pLocationField.setText("");
        pPhoneField.setText("");
```

```
            pEmailField.setText("");
            pTrainerField.setText("");
            pMaleBtn.setSelected(false);
            pFemaleBtn.setSelected(false);
            pDobYear.setSelectedIndex(0);
            pDobMonth.setSelectedIndex(0);
            pDobDay.setSelectedIndex(0);
            pMsYear.setSelectedIndex(0);
            pMsMonth.setSelectedIndex(0);
            pMsDay.setSelectedIndex(0);
            pChargeField.setText("50000");
        }

        // Activate membership handler
        // Activates a member's status
        if (btn == activateMemberBtn)
        {
            try
            {
                int id = Integer.parseInt(actIdField.getText().trim());
                GymMember gm = searchMemId(id);
                if (gm != null)
                {
                    gm.activeMembership();
                    JOptionPane.showMessageDialog(frame, "Membership activated!", "Status
message", JOptionPane.INFORMATION_MESSAGE);
                }
                else
                {
                    JOptionPane.showMessageDialog(frame, "Member not found!", "Status
message", JOptionPane.ERROR_MESSAGE);
                }
            }
            catch (NumberFormatException ex)
            {
                JOptionPane.showMessageDialog(frame, "Member Id can be number only!",
"Status message", JOptionPane.WARNING_MESSAGE);
            }
        }
        // Deactivate membership handler
        // Deactivates a member's status
        if (btn == deactivateMemberBtn)
        {
            try
            {
                int id = Integer.parseInt(actIdField.getText().trim());
```

```
            GymMember gm = searchMemId(id);
            if (gm != null)
            {
                if (gm.getActiveStatus())
                {
                    gm.deactivateMemberhship();
                    JOptionPane.showMessageDialog(frame, "Membership deactivated!",
"Status message", JOptionPane.INFORMATION_MESSAGE);
                }
                else
                {
                    JOptionPane.showMessageDialog(frame, "Membership is not active!",
"Status message", JOptionPane.WARNING_MESSAGE);
                }
            }
            else
            {
                JOptionPane.showMessageDialog(frame, "Member not found!", "Status
message", JOptionPane.ERROR_MESSAGE);
            }
        }
        catch (NumberFormatException ex)
        {
            JOptionPane.showMessageDialog(frame, "Member Id can be number only!",
"Status message", JOptionPane.WARNING_MESSAGE);
        }
    }
    // Mark regular attendance handler
    // Records attendance for regular members
    if (btn == markRegularBtn)
    {
        try
        {
            int id = Integer.parseInt(markIdField.getText().trim());
            GymMember gm = searchMemId(id);
            // Downcasting to check if the member is a RegularMember
            if (gm != null && gm instanceof RegularMember)
            {
                if (gm.getActiveStatus())
                {
                    gm.markAttendance();
                    JOptionPane.showMessageDialog(frame, "Attendance marked for
Regular Member!", "Attendance message", JOptionPane.INFORMATION_MESSAGE);
                }
                else
                {
```

```
                    JOptionPane.showMessageDialog(frame, "Member is not active!",
"Attendance message", JOptionPane.WARNING_MESSAGE);
                }
            }
            else
            {
                JOptionPane.showMessageDialog(frame, "Regular Member not found!",
"Attendance message", JOptionPane.ERROR_MESSAGE);
            }
        }
        catch (NumberFormatException ex)
        {
            JOptionPane.showMessageDialog(frame, "Invalid ID.", "Attendance
message", JOptionPane.WARNING_MESSAGE);
        }
    }
    // Mark premium attendance handler
    // Records attendance for premium members
    if (btn == markPremiumBtn)
    {
        try
        {
            int id = Integer.parseInt(markIdField.getText().trim());
            GymMember gm = searchMemId(id);
            // Downcasting to check if the member is a PremiumMember
            if (gm != null && gm instanceof PremiumMember)
            {
                if (gm.getActiveStatus())
                {
                    gm.markAttendance();
                    JOptionPane.showMessageDialog(frame, "Attendance marked for
Premium Member!", "Attendance message", JOptionPane.INFORMATION_MESSAGE);
                }
                else
                {
                    JOptionPane.showMessageDialog(frame, "Member is not active!",
"Attendance message", JOptionPane.WARNING_MESSAGE);
                }
            }
            else
            {
                JOptionPane.showMessageDialog(frame, "Premium Member not found!",
"Attendance message", JOptionPane.ERROR_MESSAGE);
            }
        }
        catch (NumberFormatException ex)
```

```
            {
                JOptionPane.showMessageDialog(frame, "Invalid ID.", "Attendance
message", JOptionPane.WARNING_MESSAGE);
            }
        }
        // Upgrade plan handler
        // Upgrades a regular member's plan
        if (btn == upgradePlanBtn)
        {
            try
            {
                int id = Integer.parseInt(upIdField.getText().trim());
                GymMember gm = searchMemId(id);
                // Downcasting to RegularMember for plan upgrade functionality
                if (gm != null && gm instanceof RegularMember)
                {
                    RegularMember rm = (RegularMember) gm;
                    String selectedPlan = (String) upPlanBox.getSelectedItem();
                    if (rm.getPlan().equalsIgnoreCase(selectedPlan))
                    {
                        JOptionPane.showMessageDialog(frame, "You are already subscribed
to the " + selectedPlan + " plan.");
                        return;
                    }
                    if (rm.getAttendance() < 30)
                    {
                        JOptionPane.showMessageDialog(frame, "Not enough attendance for
upgrade! (Minimum 30 required)");
                        return;
                    }
                    String result = rm.upgradePlan(selectedPlan);
                    JOptionPane.showMessageDialog(frame, result);
                }
                else
                {
                    JOptionPane.showMessageDialog(frame, "Regular Member not found!");
                }
            }
            catch (NumberFormatException ex)
            {
                JOptionPane.showMessageDialog(frame, "Invalid ID.");
            }
        }
        // Calculate discount handler
        // Computes discount for premium members
        if (btn == calcDiscountBtn)
```

```
        {
            try
            {
                int id = Integer.parseInt(discIdField.getText().trim());
                GymMember gm = searchMemId(id);
                // Downcasting to PremiumMember for discount calculation
                if (gm != null && gm instanceof PremiumMember)
                {
                    PremiumMember pm = (PremiumMember) gm;
                    pm.calculateDiscount();
                    discAmountField.setText(String.valueOf(pm.getDiscountAmount()));
                }
                else
                {
                    JOptionPane.showMessageDialog(frame, "Premium Member not found!");
                }
            }
            catch (NumberFormatException ex)
            {
                JOptionPane.showMessageDialog(frame, "Invalid ID.");
            }
        }
        // Revert regular member handler
        // Removes a regular member
        if (btn == revertRegularBtn)
        {
            try
            {
                int id = Integer.parseInt(revertIdField.getText().trim());
                GymMember gm = searchMemId(id);
                // Downcasting to RegularMember for reversion
                if (gm != null && gm instanceof RegularMember)
                {
                    String reason = JOptionPane.showInputDialog(frame, "Enter removal
reason:");
                    if (reason == null || reason.isEmpty())
                    {
                        JOptionPane.showMessageDialog(frame, "Removal reason is
required.");
                        return;
                    }
                    RegularMember rm = (RegularMember) gm;
                    rm.revertRegularMember(reason);
                    JOptionPane.showMessageDialog(frame, "Regular Member reverted!");

                }
```

```
            else
            {
                JOptionPane.showMessageDialog(frame, "Regular Member not found!");
            }
        }
        catch (NumberFormatException ex)
        {
            JOptionPane.showMessageDialog(frame, "Invalid ID.");
        }
    }
    // Revert premium member handler
    // Removes a premium member
    if (btn == revertPremiumBtn)
    {
        try
        {
            int id = Integer.parseInt(revertIdField.getText().trim());
            GymMember gm = searchMemId(id);
            // Downcasting to PremiumMember for reversion
            if (gm != null && gm instanceof PremiumMember)
            {
                PremiumMember pm = (PremiumMember) gm;
                pm.revertPremiumMember();
                JOptionPane.showMessageDialog(frame, "Premium Member reverted!");

            }
            else
            {
                JOptionPane.showMessageDialog(frame, "Premium Member not found!");
            }
        }
        catch (NumberFormatException ex)
        {
            JOptionPane.showMessageDialog(frame, "Invalid ID.");
        }
    }
    // Pay due handler
    // Processes payments for premium members
    if (btn == payDueButton)
    {
        try
        {
            int id = Integer.parseInt(payIdField.getText().trim());
            double amt = Double.parseDouble(payAmountField.getText().trim());
            GymMember gm = searchMemId(id);
            // Downcasting to PremiumMember for payment processing
```

```
            if (gm != null)
            {
                if (gm instanceof PremiumMember)
                {
                    PremiumMember pm = (PremiumMember) gm;
                    String result = pm.payDueAmount(amt);
                    JOptionPane.showMessageDialog(frame, "Payment processed: " +
result);
                }
                else
                {
                    JOptionPane.showMessageDialog(frame, "Pay Due Amount is
applicable for Premium Members only.");
                }
            }
            else
            {
                JOptionPane.showMessageDialog(frame, "Member not found!");
            }
        }
        catch (NumberFormatException ex)
        {
            JOptionPane.showMessageDialog(frame, "Invalid ID or amount.");
        }
    }
    // Save to file handler
    // Exports member data to a text file
    if (btn == saveBtn)
    {
        if (membersObj.isEmpty())
        {
            JOptionPane.showMessageDialog(frame, "No members to save!", "Save to
File", JOptionPane.ERROR_MESSAGE);
            return;
        }
        try
        {
            // File handling: Using FileWriter to write member data to a file
            FileWriter writer = new FileWriter("MemberDetails.txt");

            // Header line, with Personal Trainer column added
            writer.write(String.format(
                "%-5s %-15s %-15s %-15s %-25s %-20s %-10s %-10s %-10s %-15s %-
12s %-20s %-12s %-15s %-15s\n",
                "ID", "Name", "Location", "Phone", "Email",
                "Membership Start", "Plan", "Price", "Attendance",
```

```
                "Loyalty Points", "ActiveStatus", "Personal Trainer", "FullPayment",
"DiscountAmt", "PaidAmt"
            ));

            // Using a for-each loop to iterate through the ArrayList of members
            for (GymMember gm : membersObj)
            {
                String plan;
                double price;
                String trainer;      // now included for both types
                boolean fullPay;
                double discount;
                double totalPaid;

                // Downcasting to determine member type and access specific attributes
                if (gm instanceof RegularMember)
                {
                    RegularMember rm = (RegularMember) gm;
                    plan      = rm.getPlan();
                    price     = rm.getPrice();
                    trainer   = "N/A";              // no trainer for regular members
                    fullPay   = true;
                    discount  = 0.0;
                    totalPaid = price;
                }
                else  // PremiumMember
                {
                    PremiumMember pm = (PremiumMember) gm;
                    plan      = "Premium";
                    price     = pm.getPremiumCharge();
                    trainer   = pm.getPersonalTrainer();
                    fullPay   = pm.isFullPayment();
                    discount  = pm.getDiscountAmount();
                    totalPaid = fullPay
                                ? (price - discount)
                                : pm.getPaidAmount();
                }

                writer.write(String.format(
                    "%-5d %-15s %-15s %-15s %-25s %-20s %-10s %-10.2f %-10d %-15.2f
%-12b %-20s %-12b %-15.2f %-15.2f\n",
                    gm.getId(),
                    gm.getName(),
                    gm.getLocation(),
                    gm.getPhone(),
                    gm.getEmail(),
```

```
                    gm.getMembershipStartDate(),
                    plan,
                    price,
                    gm.getAttendance(),
                    gm.getLoyaltyPoints(),
                    gm.getActiveStatus(),
                    trainer,
                    fullPay,
                    discount,
                    totalPaid
                ));
            }

            writer.close();
            JOptionPane.showMessageDialog(frame, "Member details saved to file!",
"Save to File", JOptionPane.INFORMATION_MESSAGE);
        }
        catch (IOException ex)
        {
            JOptionPane.showMessageDialog(frame, "Error saving to file!", "Save to
File", JOptionPane.ERROR_MESSAGE);
        }
    }

    // Read from file handler
    // Imports member data from a text file
    if (btn == readBtn)
    {
        try
        {
            // File handling: Using FileReader to read member data from a file
            FileReader reader = new FileReader("MemberDetails.txt");
            String readData = "";
            int c;
            // Using a while loop to read the file character by character
            while ((c = reader.read()) != -1)
            {
                readData += (char) c;
            }
            reader.close();

            // Display in a new frame with monospaced font for alignment
            JFrame readDisplayFrame = new JFrame("Member Details from File");
            readDisplayFrame.setSize(1300, 610);
            readDisplayFrame.setLayout(null);
            readDisplayFrame.setResizable(false);
```

```
                JTextArea textArea = new JTextArea();
                textArea.setEditable(false);
                textArea.setFont(new Font("Monospaced", Font.PLAIN, 12));
                textArea.setText(readData);

                JScrollPane scrollPane = new JScrollPane(textArea);
                scrollPane.setBounds(10, 10, 1270, 550);

                readDisplayFrame.add(scrollPane);
                readDisplayFrame.setVisible(true);
            }
            catch (IOException ex)
            {
                JOptionPane.showMessageDialog(frame, "Error reading file!", "Read from
File", JOptionPane.ERROR_MESSAGE);
            }
        }
    }
    // Main method
    // Entry point of the application
    public static void main(String[] args)
    {
        new GymGUI();
    }
}
```