**\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\* Input CODE \*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\***

from datetime import datetime

class Thala07Fitness:

def \_\_init\_\_(self):

self.customers = {}

self.memberships = {}

self.next\_customer\_id = 1

self.next\_membership\_id = 1

def register\_customer(self):

name = input("Enter customer name: ")

phone\_no = int(input("Enter customer phone number: "))

join\_date = input("Enter customer join date (DD/MM/YYYY): ")

new\_customer = Customer(name, phone\_no, join\_date)

self.customers[self.next\_customer\_id] = new\_customer

self.next\_customer\_id += 1

print(f"\nCustomer {new\_customer.name} registered successfully with ID {self.next\_customer\_id - 1}.")

def add\_membership(self):

print("\nSelect membership type:")

print("1. Premium (Gym + Cardio + Zumba + Crossfit + Ice Bath + Swimming + Steam)\n")

print("2. Diamond (Gym + Cardio + Crossfit + Zumba)\n")

print("3. Gold (Gym + Cardio + Crossfit)\n")

print("4. Silver (Gym + Cardio + Ice Bath)\n")

print("5. Bronze (Gym + Cardio)\n")

choice = input("Enter your choice: ")

if choice in ["1", "2", "3", "4", "5"]:

type, facilities, cost = self.get\_membership\_details(choice)

self.memberships[self.next\_membership\_id] = Membership(type, facilities, cost)

print(f"\nMembership {type} added successfully with ID {self.next\_membership\_id}.")

self.next\_membership\_id += 1

else:

print("\nInvalid choice. Please enter a valid option.")

def get\_membership\_details(self, choice):

membership\_data = {

"1": ("Premium", "Gym, Cardio, Zumba, Crossfit, Ice Bath, Swimming, Steam", {"1month": 20000.00, "3month": 57000.00, "6month": 108000.00, "12month": 204000.00}),

"2": ("Diamond", "Gym, Cardio, Crossfit, Zumba", {"1month": 15000.00, "3month": 42000.00, "6month": 78000.00, "12month": 144000.00}),

"3": ("Gold", "Gym, Cardio, Crossfit", {"1month": 10000.00, "3month": 27000.00, "6month": 48000.00, "12month": 90000.00}),

"4": ("Silver", "Gym, Cardio, Ice Bath", {"1month": 8000.00, "3month": 21000.00, "6month": 36000.00, "12month": 66000.00}),

"5": ("Bronze", "Gym, Cardio", {"1month": 5000.00, "3month": 12000.00, "6month": 21000.00, "12month": 39000.00}),

}

type, facilities, cost = membership\_data[choice]

return type, facilities, cost

def choose\_membership(self):

customer\_id = int(input("\nEnter customer ID: "))

membership\_id = int(input("Enter membership ID: "))

if customer\_id not in self.customers:

print("\nCustomer not found.")

return

if membership\_id not in self.memberships:

print("\nMembership not found.")

return

membership\_cost = self.memberships[membership\_id].cost

print(f"\nMembership {self.memberships[membership\_id].type} costs:")

for tenure, price in membership\_cost.items():

print(f"{tenure.capitalize()}: ₹{price:.2f}")

print("\nChoose the tenure:")

print("1. 1 month")

print("2. 3 months")

print("3. 6 months")

print("4. 12 months")

tenure\_choice = input("Enter your choice: ")

tenure\_map = {"1": "1month", "2": "3month", "3": "6month", "4": "12month"}

tenure = tenure\_map.get(tenure\_choice)

if not tenure:

print("\nInvalid tenure. Please choose from available options.")

return

confirm = input("\nDo you want to proceed with the payment? (yes/no): ")

if confirm.lower() == "yes":

amount\_paid = membership\_cost[tenure]

payment\_methods = {

"1": "UPI-GPay",

"2": "Paytm",

"3": "PhonePe",

"4": "Cash",

"5": "Bank Transfer"

}

print(f"\nYou are paying ₹{amount\_paid:.2f} for {tenure} {self.memberships[membership\_id].type} membership.")

print("Select payment method:")

for key, value in payment\_methods.items():

print(f"{key}. {value}")

payment\_method\_choice = input("Enter your choice: ")

if payment\_method\_choice in payment\_methods:

payment\_method = payment\_methods[payment\_method\_choice]

print(f"\nPayment successful via {payment\_method} on {datetime.now().strftime('%Y-%m-%d')}.")

new\_payment = Payment(amount\_paid, datetime.now().strftime("%Y-%m-%d"))

self.customers[customer\_id].payments.append(new\_payment)

print(f"Payment of ₹{amount\_paid:.2f} made by customer {self.customers[customer\_id].name} on {new\_payment.date}.")

self.customers[customer\_id].memberships.append(self.memberships[membership\_id])

print(f"Customer {self.customers[customer\_id].name} has chosen membership {self.memberships[membership\_id].type}.")

else:

print("\nInvalid payment method. Payment cancelled.")

else:

print("\nPayment cancelled.")

def track\_attendance(self):

customer\_id = int(input("\nEnter customer ID: "))

check\_in = input("Press enter to check in: ")

check\_out = input("Press enter to check out: ")

if customer\_id not in self.customers:

print("\nCustomer not found.")

return

current\_month = datetime.now().strftime("%B")

attendance\_record = {"Check-in": check\_in, "Check-out": check\_out}

self.customers[customer\_id].attendance[current\_month].append(attendance\_record)

print("\nAttendance recorded successfully.")

def display\_customer\_info(self):

customer\_id = int(input("\nEnter customer ID: "))

if customer\_id not in self.customers:

print("\nCustomer not found.")

return

print(self.customers[customer\_id])

def display\_all\_customers(self):

for customer\_id, customer in self.customers.items():

print(customer)

class Customer:

def \_\_init\_\_(self, name, phone\_no, join\_date):

self.name = name

self.phone\_no = phone\_no

self.join\_date = join\_date

self.memberships = []

self.payments = []

self.attendance = {"January": [], "February": [], "March": [], "April": [], "May": [], "June": [], "July": [], "August": [], "September": [], "October": [], "November": [], "December": []}

def \_\_str\_\_(self):

memberships\_info = "\nMemberships: " + ", ".join([membership.type for membership in self.memberships])

payments\_info = "\nPayments: " + ", ".join([f"₹{payment.amount:.2f} on {payment.date}" for payment in self.payments])

attendance\_info = "\nAttendance: "

for month, records in self.attendance.items():

attendance\_info += f"\n{month}: {len(records)} check-ins/check-outs"

return f"\nName: {self.name}, Phone: {self.phone\_no}, Joining Date: {self.join\_date}" + memberships\_info + payments\_info + attendance\_info

class Membership:

def \_\_init\_\_(self, type, facilities, cost):

self.type = type

self.facilities = facilities

self.cost = cost

def \_\_str\_\_(self):

return f"Type: {self.type}, Facilities: {self.facilities}, Cost: {self.cost}"

class Payment:

def \_\_init\_\_(self, amount, date):

self.amount = amount

self.date = date

def \_\_str\_\_(self):

return f"Amount: ₹{self.amount:.2f}, Date: {self.date}"

# Example usage:

thalaFitness = Thala07Fitness()

while True:

print("\n---------------------------------------------")

print(" Welcome to Thala 07 Fitness Club")

print("---------------------------------------------")

print("\n1. Register Customer")

print("2. Add Membership")

print("3. Choose Membership")

print("4. Track Attendance")

print("5. Display Customer Information")

print("6. Display All Customers")

print("7. Quit")

choice = input("\nEnter your choice: ")

if choice == "1":

thalaFitness.register\_customer()

elif choice == "2":

thalaFitness.add\_membership()

elif choice == "3":

thalaFitness.choose\_membership()

elif choice == "4":

thalaFitness.track\_attendance()

elif choice == "5":

thalaFitness.display\_customer\_info()

elif choice == "6":

thalaFitness.display\_all\_customers()

elif choice == "7":

print("\nTHANK YOU FOR USING OUR GYM \n 'NO PAIN NO GAIN'\n")

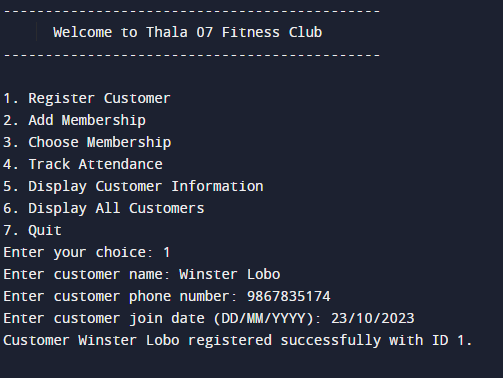
print("\n\*\*\*\*\*THALA FOR A REASON\*\*\*\*\*")

break

else:

print("\nInvalid choice. Please enter a valid option.")

**\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\* OUTPUT CODE \*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\***



A screenshot of a computer program

Description automatically generated

A screenshot of a computer program

Description automatically generated

A screenshot of a computer program

Description automatically generated

A screenshot of a computer

Description automatically generated

A screenshot of a computer

Description automatically generated

A screenshot of a computer program

Description automatically generated