ABOUT THE GYM

A gym, traditionally known as a "gymnasium," is a facility designed to promote physical fitness and well-being through various forms of exercise. Modern gyms are equipped with a wide range of machines, free weights, and spaces dedicated to different types of workouts, catering to people of varying fitness levels and goals.

Typically, a gym offers areas for strength training, where individuals can use free weights like dumbbells and barbells, along with resistance machines to build muscle. Cardio sections feature equipment like treadmills, stationary bikes, ellipticals, and rowing machines to improve cardiovascular endurance. Many gyms also provide functional training areas with equipment like kettlebells, battle ropes, and medicine balls for versatile, full-body workouts. In addition to solo exercise, gyms often offer group fitness classes, which can include activities such as yoga, Pilates, aerobics, spinning (indoor cycling), and high-intensity interval training (HIIT). These classes are usually led by certified instructors and provide a social, motivational environment for members to stay active and engaged. Many modern gyms also offer personal training services, where certified fitness professionals provide one-on-one coaching tailored to an individual's specific needs and goals. Personal trainers can create customized workout plans, monitor progress, and ensure proper exercise techniques, reducing the risk of injury while maximizing results. Beyond physical exercise, gyms are often equipped with additional amenities like saunas, swimming pools, steam rooms, and relaxation areas. Some even provide nutritional guidance, wellness coaching, and recovery options like massages or physiotherapy to support overall health and fitness. Overall, a gym is more than just a place to work out; it serves as a hub for people seeking to improve their physical and mental well-being. Regular exercise in a gym can lead to numerous health benefits, including enhanced cardiovascular health, increased muscle strength and endurance, better flexibility, stress reduction, weight management, and improved mental clarity and mood.

PROJECT ON GYM MANAGEMENT SYSTEM

INTRODUCTION

This program helps the user to manage a gym using the Python and MySQL.

After adding information this program automatically generates the details of the members, staff and many more on the basis of the information provided by the user.

OBJECTIVES OF THE PROJECT

The objective of this project is to let students apply the programming knowledge into real world situations/problems and exposed the students how programming skills help in developing a good software.

- Write programs utilizing modern software tools.
- Apply object-oriented programming principles effectively when developing small to medium sized projects.
- Write effective procedural code to solve small to medium sized problems.
- Students will demonstrate a breadth of knowledge in computer science, as exemplified in the areas of systems, theory and software development.
- Students will demonstrate ability to conduct a research or applied Computer Science project, requiring writing and presentation skills which exemplify scholarly style in computer science.

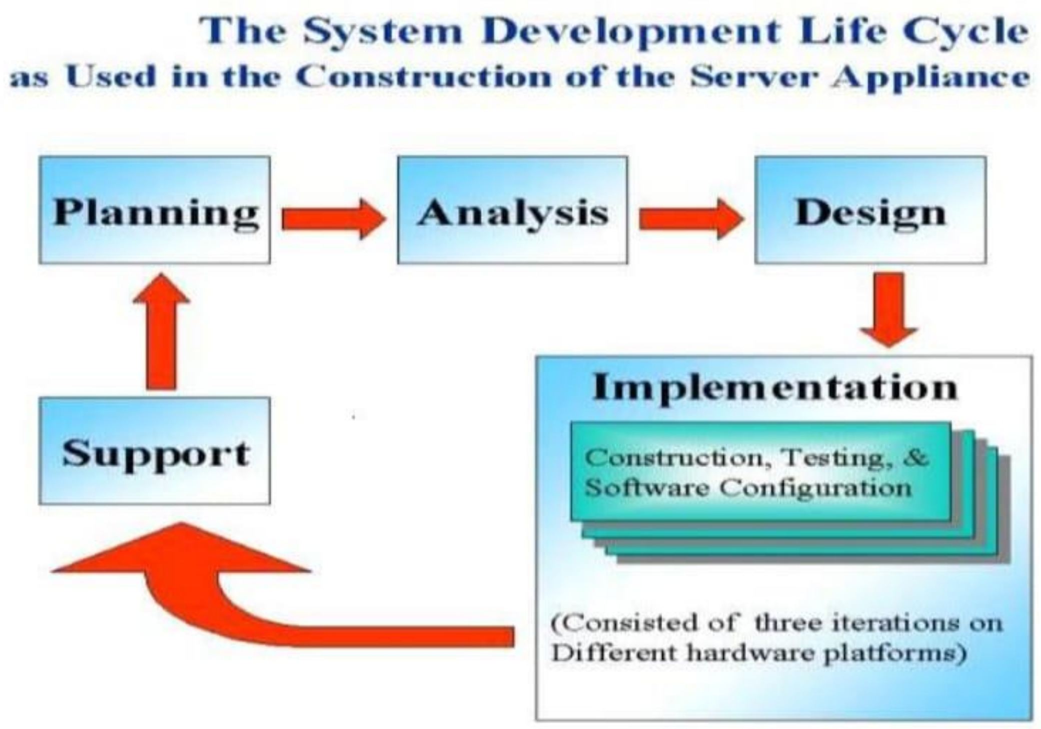
PROPOSED SYSTEM

Today one cannot afford to rely on the fallible human beings of be really wants to stand against today's merciless competition where not to wise saying "**to err is human**" no longer valid, it's out-dated to rationalize your mistake. So, to keep pace with time, to bring about the best result without malfunctioning and greater efficiency so to replace the unending heaps of flies with a much sophisticated hard disk of the computer.

One has to use the data management software. Software has been an ascent in atomization various organisations. Many software products working are now in markets, which have helped in making the organizations work easier and efficiently. Data management initially had to maintain a lot of ledgers and a lot of paperwork has to be done but now software production this organization has made their work faster and easier. Now only this software has to be loaded on the computer and work can be done.

This prevents a lot of time and money. The work becomes fully automated and any information regarding the organization can be obtained by clicking the button. Moreover, now it's an age of computers of and automating such an organization gives the better look.

SYSTEM DEVELOPMENT LIFE CYCLE(SDLC)


The System Development Life Cycle
as Used in the Construction of the Server Appliance
Planning
Analysis
Design
Implementation
Construction, Testing, & Software Configuration
(Consisted of three iterations on Different hardware platforms)
Support

The systems development life cycle is a project management technique that divides complex projects into smaller, more easily managed segments or phases. Segmenting projects allows managers to verify the successful completion of project phases before allocating resources to subsequent phases.

Software development projects typically include initiation, planning, design, development, testing, implementation, and maintenance phases. However, the phases may be divided differently depending on the organization involved.

For example, initial project activities might be designated as request, requirements-definition, and planning phases, or initiation, concept-development, and planning phases. End users of the system under development should be involved in reviewing the output of each phase to ensure the system is being built to deliver the needed functionality.

PHASES OF SYSTEM DEVELOPMENT LIFE CYCLE

INITIATION PHASE

The initiation phase begins when a business sponsor identifies a need or an opportunity

The purpose of the initiation phase is to:

- Identify and validate an opportunity to improve business accomplishments of the organization or a deficiency related to a business need.
- Identify significant assumptions and constraints on solutions to that need.
- Recommend the exploration of alternative concepts and methods to satisfy the need including questioning the need for technology, i.e., will a change in the business process offer a solution?
- Assure executive business and executive technical sponsorship. The Sponsor designates a Project Manager and the business need is documented in a Concept Proposal. The Concept Proposal includes information about the business process and the relationship to the Agency/Organization.
- Infrastructure and the Strategic Plan. A successful Concept Proposal results in a Project Management Charter which outlines the authority of the project manager to begin the project.

Careful oversight is required to ensure projects support strategic business objectives and resources are effectively implemented into an organization's enterprise architecture. The initiation phase begins when an opportunity to add, improve, or correct a system is identified and formally requested through the presentation of a business case. The business case should, at a minimum, describe a proposal's purpose, identify expected benefits, and explain how the proposed system supports one of the organization's business strategies. The business case should also identify alternative solutions and detail as many informational, functional, and network requirements as possible.

SYSTEM CONCEPT DEVELOPMENT PHASE

The System Concept Development Phase begins after a business need or opportunity is validated by the Agency/Organization Program Leadership and the Agency/Organization CIO.

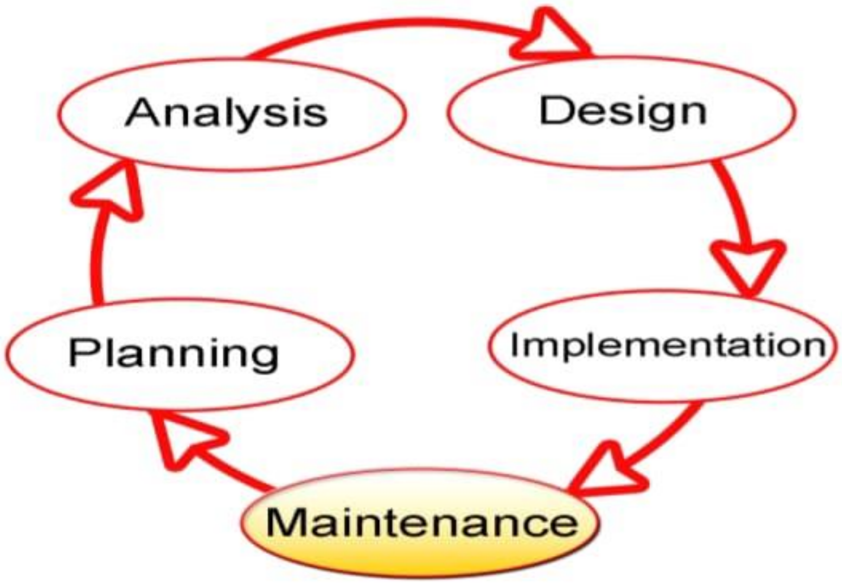
The purpose of the System Concept Development Phase is to:

- Determine the feasibility and appropriateness of the alternatives.
- Identify system interfaces.
- Identify basic functional and data requirements to satisfy the business need.
- Establish system boundaries; identify goals, objectives, critical success factors, and performance measures.
- Evaluate costs and benefits of alternative approaches to satisfy the basic functional requirements
- Assess project risks
- Identify and initiate risk mitigation actions, and develop high-level technical architecture, process models, data models, and a concept of operations. This phase explores potential technical solutions within the context of the business need.
- It may include several trade-off decisions such as the decision to use COTS software products as opposed to develop custom software or reusing software components,

or the decision to use an incremental delivery versus a complete, onetime deployment.

- Construction of executable prototypes is encouraged to evaluate technology to support the business process. The System Boundary Document serves as an important reference document to support the Information Technology Project Request (ITPR) process.
- The ITPR must be approved by the State CIO before the project can move forward.

PICTORIAL REPRESENTATION OF SDLC



PLANNING PHASE

The planning phase is the most critical step in completing development, acquisition, and maintenance projects. Careful planning, particularly in the early stages of a project, is necessary to coordinate activities and manage project risks effectively. The depth and formality of project plans should be commensurate with the characteristics and risks of a given project. Project plans refine the information gathered during the initiation phase by further identifying the specific activities and resources required to complete a project.

A critical part of a project manager’s job is to coordinate discussions between user, audit, security, design, development, and network personnel to identify and document as many functional, security, and network requirements as possible. During this phase, a plan is developed that documents the approach to be used and includes a discussion of methods, tools, tasks, resources, project schedules, and user input. Personnel assignments, costs, project schedule, and target dates are established.

A Project Management Plan is created with components related to acquisition planning, configuration management planning, quality assurance planning, concept of operations, system security, verification and validation, and systems engineering management planning.

REQUIREMENT ANALYSIS PHASE

This phase formally defines the detailed functional user requirements using high-level requirements identified in the Initiation, System Concept, and Planning phases. It also delineates the requirements in terms of data, system performance, security, and maintainability requirements for the system. The requirements are defined in this phase to a level of detail sufficient for systems design to proceed. They need to be measurable, testable, and relate to the business need or opportunity identified in the Initiation Phase. The requirements that will be used to determine acceptance of the system are captured in the Test and Evaluation Masterplan.

The purposes of this phase are to:

- Further define and refine the functional and data requirements and document them in the Requirements Document.
- Complete business process reengineering of the functions to be supported (i.e., verify what information drives the business process, what information is generated, who generates it, where does the information go, and who processes it),
- Develop detailed data and process models (system inputs, outputs, and the process.
- Develop the test and evaluation requirements that will be used to determine acceptable system performance.

DESIGN PHASE

The design phase involves converting the informational, functional, and network requirements identified during the initiation and planning phases into unified design specifications that developers use to script programs during the development phase. Program designs are constructed in various ways. Using a top-down approach, designers first identify and link major program components and interfaces, then expand design layouts as they identify and link smaller subsystems and connections. Using a bottom-up approach, designers first identify and link minor program components and interfaces, then expand design layouts as they identify and link larger systems and connections. Contemporary design techniques often use prototyping tools that build mock-up designs of items such as application screens, database layouts, and system architectures. End users, designers, developers, database managers, and network administrators should review and refine the prototyped designs in an iterative process until they agree on an acceptable design. Audit, security, and quality assurance personnel should be involved in the review and approval process. During this phase, the system is designed to satisfy the functional requirements identified in the previous phase. Since problems in the design phase could be very expensive to solve in the later stage of the software development, a variety of elements are considered in the design to mitigate risk. These include:

- Identifying potential risks and defining mitigating design features.
- Performing a security risk assessment.
- Developing a conversion plan to migrate current data to the new system.
- Determining the operating environment.
- Defining major subsystems and their inputs and outputs.
- Allocating processes to resources.
- Preparing detailed logic specifications for each software module. The result is a draft System Design Document which captures the preliminary design for the system.
- Everything requiring user input or approval is documented and reviewed by the user. Once these documents have been approved by the Agency CIO and Business Sponsor, the final System Design Document is created to serve as the Critical/Detailed Design for the system.
- This document receives a rigorous review by Agency technical and functional representatives to ensure that it satisfies the business requirements. Concurrent with the development of the system design, the Agency Project Manager begins development of the Implementation Plan, Operations and Maintenance Manual, and the Training Plan.

DEVELOPMENT PHASE

The development phase involves converting design specifications into executable programs. Effective development standards include requirements that programmers and other project participants discuss design specifications before programming begins. The procedures help ensure programmers clearly understand program designs and functional requirements. Programmers use various techniques to develop computer programs. The large transaction oriented programs associated with financial institutions have traditionally been developed using procedural programming techniques. Procedural programming involves the line-by-line scripting of logical instructions that are combined to form a program. Effective completion of the previous stages is a key factor in the success of the Development phase. The Development phase consists of:

- Translating the detailed requirements and design into system components.
- Testing individual elements (units) for usability.
- Preparing for integration and testing of the IT system.

INTEGRATION AND TEST PHASE

Subsystem integration, system, security, and user acceptance testing is conducted during the integration and test phase. The user, with those responsible for quality assurance, validates that the functional requirements, as defined in the functional requirements document, are satisfied by the developed or modified system. OIT Security staff assesses the system security and issue a security certification and accreditation prior to installation/implementation.

Multiple levels of testing are performed including

- Testing at the development facility by the contractor and possibly supported by end users
- Testing as a deployed system with end users working together with contract personnel
- Operational testing by the end user alone performing all functions. Requirements are traced throughout testing, a final Independent Verification & Validation evaluation is performed and all documentation is reviewed and accepted prior to acceptance of the system.

IMPLEMENTATION PHASE

This phase is initiated after the system has been tested and accepted by the user. In this phase, the system is installed to support the intended business functions. System performance is compared to performance objectives established during the planning phase. Implementation includes user notification, user training, installation of hardware, installation of software onto production computers, and integration of the system into daily work processes. This phase continues until the system is operating in production in accordance with the defined user requirements.

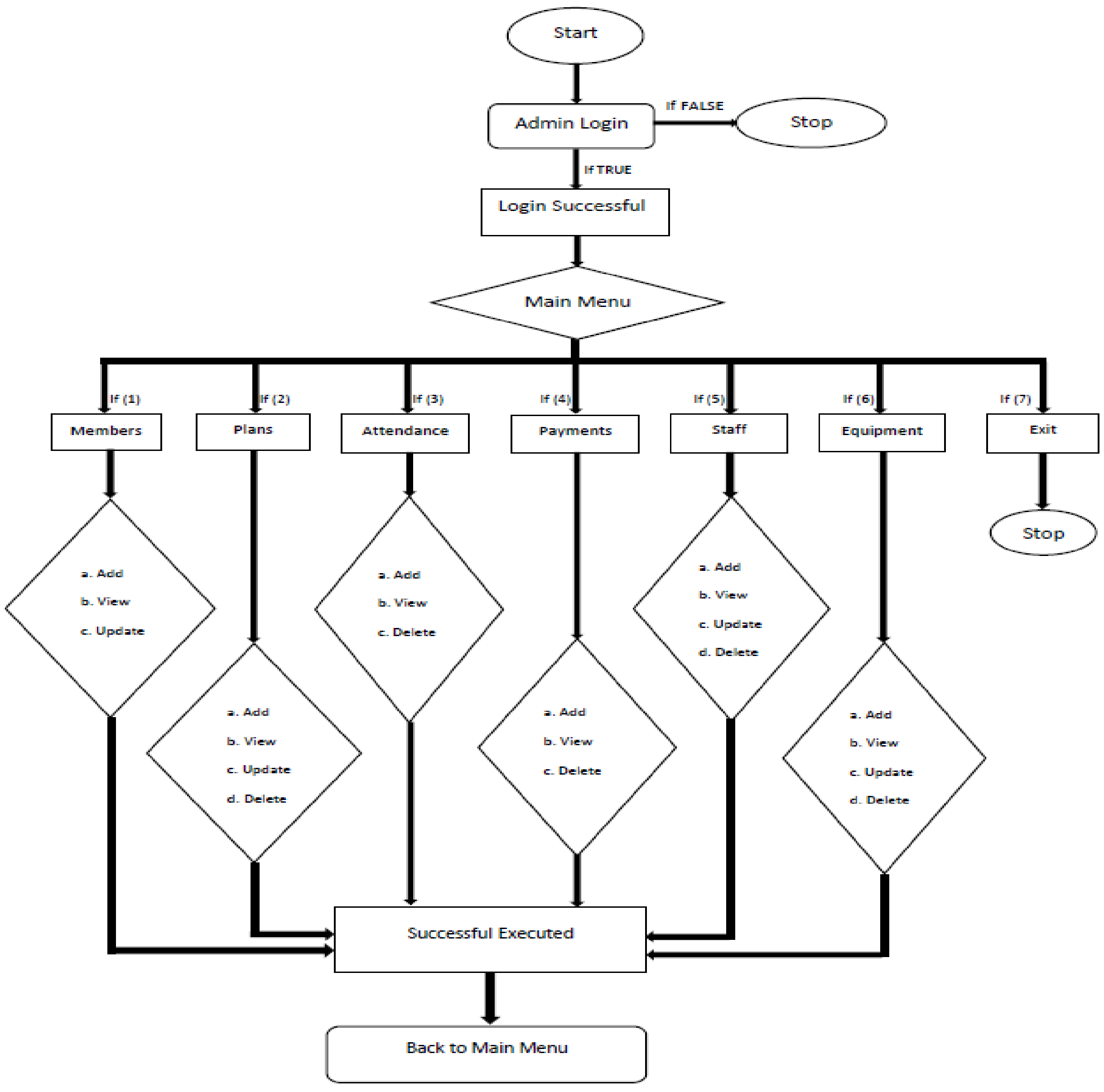
OPERATIONS AND MAINTENANCE PHASE

The system operation is on-going. The system is monitored for continued performance in accordance with user requirements and needed system modifications are incorporated. Operations continue as long as the system can be effectively adapted to respond to the organization's needs. When modifications or changes are identified, the system may re-enter the planning phase.

The purpose of this phase is to:

- Operate, maintain, and enhance the system.
- Certify that the system can process sensitive information.
- Conduct periodic assessments of the system to ensure the functional requirements continue to be satisfied.
- Determine when the system needs to be modernized, replaced, or retired.

FLOW CHART FOR SOURCE CODE



SOURCE CODE

```
import mysql.connector
print("Admin Login")
us = input("Enter your SQL User ID: ")
ps = input("Enter your SQL password: ")
try:
    cnx = mysql.connector.connect(
        user=str(us),
        password=str(ps),
        host='localhost',
        port=3306
    )
    database = 'gym_management'

    cursor = cnx.cursor()
    query = "SHOW DATABASES LIKE '%s'" % database
    cursor.execute(query)

    results = cursor.fetchall()

    if results:
        print("Gym_management Database was previously created
and may contain saved values")
```

```
    else:
        cursor.execute("CREATE DATABASE 
gym_management")
        cursor.execute("USE gym_management")

        cursor.execute("""
            CREATE TABLE members (
            member_id INT AUTO_INCREMENT PRIMARY 
KEY,
            first_name VARCHAR(50),
            last_name VARCHAR(50),
            age INT,
            gender VARCHAR(10),
            phone VARCHAR(15),
            email VARCHAR(50),
            join_date DATE,
            membership_plan VARCHAR(50)
            )
        """)

        cursor.execute("""
            CREATE TABLE plans (
            plan_id INT AUTO_INCREMENT PRIMARY KEY,
            plan_name VARCHAR(50),
            duration_months INT,
            cost DECIMAL(10, 2)
            )
        """)

        cursor.execute("""
            CREATE TABLE attendance (
            attendance_id INT AUTO_INCREMENT PRIMARY
KEY,
            member_id INT,
            date DATE,
            FOREIGN KEY (member_id) REFERENCES
members(member_id)
            )
        """)

        cursor.execute("""
            CREATE TABLE payments (
            payment_id INT AUTO_INCREMENT PRIMARY
KEY,
            member_id INT,
            amount DECIMAL(10, 2),
            payment_date DATE,
            FOREIGN KEY (member_id) REFERENCES
members(member_id)
            )
        """)

        cursor.execute("""
            CREATE TABLE staff (
            staff_id INT AUTO_INCREMENT PRIMARY KEY,
            name VARCHAR(100),
            position VARCHAR(50),
            contact_number VARCHAR(15),
            hire_date DATE
            )
        """)

        cursor.execute("""
            CREATE TABLE equipment (
            equipment_id INT AUTO_INCREMENT PRIMARY
KEY,
            name VARCHAR(100),
            conditions VARCHAR(50),
            seller_number VARCHAR(15),
            purchase_date DATE,
            amount DECIMAL(10, 2)
        )
      """)

      cnx.commit()

    cursor.close()
    cnx.close()

except mysql.connector.Error as err:
    print(f"Incorrect password entered: {err}")

# Main project(code for doing all the operation in database
gym_management)
try:
    db = mysql.connector.connect(
    host="localhost",
    user=str(us),
    password=str(ps),
    database="gym_management"
    )

    cursor = db.cursor()
```

```
# Members


    def add_member(first_name, last_name, age, gender, phone, email, join_date, membership_plan):
        try:
            sql = "INSERT INTO members (first_name, last_name, age, gender, phone, email, join_date, membership_plan) VALUES (%s, %s, %s, %s, %s, %s, %s, %s)"
            values = (first_name, last_name, age, gender, phone, email, join_date, membership_plan)
            cursor.execute(sql, values)
            db.commit()
            print("Member added successfully!")
        except mysql.connector.Error as err:
            print(f"Error adding member: {err}")


    def view_members():
        try:
            cursor.execute("SELECT * FROM members")
            members = cursor.fetchall()
            for member in members:
                print(member)
        except mysql.connector.Error as err:
            print(f"Error viewing members: {err}")
    def update_member(member_id, first_name=None,
last_name=None, age=None, phone=None, email=None):
        try:
            sql = "UPDATE members SET "
            fields = []
            values = []

            if first_name:
                fields.append("first_name = %s")
                values.append(first_name)
            if last_name:
                fields.append("last_name = %s")
                values.append(last_name)
            if age:
                fields.append("age = %s")
                values.append(age)
            if phone:
                fields.append("phone = %s")
                values.append(phone)
            if email:
                fields.append("email = %s")
                values.append(email)
            sql += ", ".join(fields)
            sql += " WHERE member_id = %s"
            values.append(member_id)

            cursor.execute(sql, tuple(values))
            db.commit()
            print("Member updated successfully!")
        except mysql.connector.Error as err:
            print(f"Error updating member: {err}")

# Plans

    def add_plan(plan_name, duration_months, cost):
        try:
            sql = "INSERT INTO plans(plan_name,
duration_months, cost) VALUES (%s, %s, %s)"
            values = (plan_name, duration_months, cost)
            cursor.execute(sql, values)
            db.commit()
            print("Plan added successfully!")
        except mysql.connector.Error as err:
            print(f"Error adding plan: {err}")
```

```
def view_plans():
    try:
        cursor.execute("SELECT * FROM plans")
        plans = cursor.fetchall()
        for plan in plans:
            print(plan)
    except mysql.connector.Error as err:
        print(f"Error viewing plans: {err}")

def update_plan(plan_id, plan_name=None,
duration_months=None, cost=None):
    try:
        sql = "UPDATE plans SET "
         fields = []
        values = []

        if plan_name:
            fields.append("plan_name = %s")
            values.append(plan_name)
        if duration_months:
            fields.append("duration_months = %s")
            values.append(duration_months)
        if cost:
            fields.append("cost = %s")
            values.append(cost)

        sql += ", ".join(fields)
        sql += " WHERE plan_id = %s"
        values.append(plan_id)

        cursor.execute(sql, tuple(values))
        db.commit()
        print("Plan updated successfully!")
    except mysql.connector.Error as err:
        print(f"Error updating plan: {err}")

def delete_plan(plan_id):
    try:
        query = "DELETE FROM plans WHERE plan_id =
%s"
        value = (plan_id,)
        cursor.execute(query, value)
        db.commit()
        print("Plan deleted successfully!")
    except mysql.connector.Error as err:
        print(f"Error deleting plan: {err}")

# Attendance

def add_attendance(member_id, date):
    try:
        sql = "INSERT INTO attendance(member_id, date) VALUES (%s, %s)"
        values = (member_id, date)
        cursor.execute(sql, values)
        db.commit()
        print("Attendance added successfully!")
    except mysql.connector.Error as err:
        print(f"Error adding attendance: {err}")

def view_attendance():
    try:
        cursor.execute("SELECT attendance_id, date, first_name, last_name FROM attendance, members WHERE attendance.member_id = members.member_id")
        result = cursor.fetchall()
        for row in result:
            print(row)
    except mysql.connector.Error as err:
```

```
        print(f"Error viewing attendance: {err}")

def delete_attendance(attendance_id):
    try:
        query = "DELETE FROM attendance WHERE
attendance_id = %s"
        value = (attendance_id,)
        cursor.execute(query, value)
        db.commit()
        print("Attendance deleted successfully!")
    except mysql.connector.Error as err:
        print(f"Error deleting attendance: {err}")

# Payment

def add_payment(member_id, amount):
    try:
        import datetime
        query = "INSERT INTO payments (member_id,
amount, payment_date) VALUES (%s, %s, %s)"
        values = (member_id, amount, datetime.date.today())
        cursor.execute(query, values)
        db.commit()
        print("Payment added successfully!")
    except mysql.connector.Error as err:
        print(f"Error adding payment: {err}")

def view_payments():
    try:
        cursor.execute("SELECT * FROM payments")
        result = cursor.fetchall()
        for row in result:
            print(row)
    except mysql.connector.Error as err:
        print(f"Error viewing payments: {err}")

def delete_payment(payment_id):
    try:
        query = "DELETE FROM payments WHERE
payment_id = %s"
        value = (payment_id,)
        cursor.execute(query, value)
        db.commit()
        print("Payment deleted successfully!")
    except mysql.connector.Error as err:
        print(f"Error deleting payment: {err}")
```

```
# Staff

    def add_staff(name, position, contact_number, hire_date):
        try:
            query = "INSERT INTO staff (name, position,
contact_number, hire_date) VALUES (%s, %s, %s, %s)"
            values = (name, position, contact_number, hire_date)
            cursor.execute(query, values)
            db.commit()
            print("Staff added successfully!")
        except mysql.connector.Error as err:
            print(f"Error adding staff: {err}")

    def view_staff():
        try:
            cursor.execute("SELECT * FROM staff")
            result = cursor.fetchall()
            for row in result:
                print(row)
        except mysql.connector.Error as err:
            print(f"Error viewing staff: {err}")

    def update_staff(staff_id, name=None, position=None,
contact_number=None):
```

```
try:
    sql = "UPDATE staff SET "
    fields = []
    values = []

    if name:
        fields.append("name = %s")
        values.append(name)
    if position:
        fields.append("position = %s")
        values.append(position)
    if contact_number:
        fields.append("contact_number = %s")
        values.append(contact_number)

    sql += ", ".join(fields)
    sql += " WHERE staff_id = %s"
    values.append(staff_id)

    cursor.execute(sql, tuple(values))
    db.commit()
    print("Staff updated successfully!")
except mysql.connector.Error as err:
```

```
            print(f"Error updating staff: {err}")

    def delete_staff(staff_id):
        try:
            query = "DELETE FROM staff WHERE staff_id = 
%s"
            value = (staff_id,)
            cursor.execute(query, value)
            db.commit()
            print("Staff deleted successfully!")
        except mysql.connector.Error as err:
            print(f"Error deleting staff: {err}")

#Equipmemt

    def add_equipment(name, conditions, seller_number, 
purchase_date, amount):
        try:
            query = "INSERT INTO equipment (name, conditions, 
seller_number, purchase_date, amount) VALUES (%s, %s, %s, 
%s, %s)"
            values = (name, conditions, seller_number, 
purchase_date, amount)

            cursor.execute(query, values)
            db.commit()

            print("Equipment added successfully!")
        except mysql.connector.Error as err:
            print(f"Error adding equipment: {err}")

    def view_equipment():
        try:
            cursor.execute("SELECT * FROM equipment")
            result = cursor.fetchall()

            for row in result:
                print(row)
        except mysql.connector.Error as err:
            print(f"Error viewing equipment: {err}")

    def update_equipment(equipment_id, name=None,
conditions=None, seller_number=None, purchase_date=None,
amount=None):
        try:
            sql = "UPDATE equipment SET "
            fields = []
            values = []
```

```
if name:
    fields.append("name = %s")
    values.append(name)
if conditions:
    fields.append("conditions = %s")
    values.append(conditions)
if seller_number:
    fields.append("seller_number = %s")
    values.append(seller_number)
if purchase_date:
    fields.append("purchase_date = %s")
    values.append(purchase_date)
if amount:
    fields.append("amount = %s")
    values.append(amount)

sql += ", ".join(fields)
sql += " WHERE equipment_id = %s"
values.append(equipment_id)

cursor.execute(sql, tuple(values))
db.commit()
print("Equipment updated successfully!")
```

```
        except mysql.connector.Error as err:
            print(f"Error updating equipment: {err}")


    def delete_equipment(equipment_id):
        try:
            query = "DELETE FROM equipment WHERE
equipment_id = %s"
            value = (equipment_id,)


            cursor.execute(query, value)


            print("Equipment deleted successfully!")
        except mysql.connector.Error as err:
            print(f"Error deleting equipment: {err}")


    if __name__ == "__main__":
        while True:
            print("\nGym Management System")
            print("Type the associate number for any of the related
issues listed below")
            print("1.Member")
            print("2.Plans")
            print("3.Attendance")
            print("4.Payments")
print("5.Staff")
print("6.Equipments")
print("7. Exit")

choice = input("Enter your choice: ")



if choice=="1":


    print("Type the letter associated with objective")
    print("a. Add Member")
    print("b. View Members")
    print("c. Update Member")



    choice1 = input("Enter your choice: ")

    if choice1 == "a":
        try:
            first_name = input("First Name: ")
            last_name = input("Last Name: ")
            age = int(input("Age: "))
            gender = input("Gender: ")
```

```
                    phone= input("Phone No: ")
                    email = input("Email Id: ")
                    join_date = input("Join Date (YYYY-MM-DD):
")
                    membership_plan = input("Membership Plan
(Silver/Gold/Diamond): ")
                    add_member(first_name, last_name, age, gender,
phone, email, join_date, membership_plan)
                except ValueError as err:
                    print(f"Error due to wrong type value in input:
{err}")

            elif choice1 == "b":
                view_members()

            elif choice1 == "c":
                try:
                    member_id = int(input("Enter Member ID: "))
                    print("Leave fields blank if you don't want to
update them.")
                    first_name = input("First Name: ")
                    last_name = input("Last Name: ")
                    age = input("Age: ")
                    age = int(age) if age else None
                    phone = input("Phone No: ")
                    email = input("Email ID: ")
                    update_member(member_id, first_name,
last_name, age, phone, email)
                except ValueError as err:
                    print(f"Error due to wrong type value in input:
{err}")

            else:
                print("Invalid choice! Please try again.")

        elif choice=="2":
            print("Type the letter associated with objective")
            print("a. Add Plan")
            print("b. View Plans")
            print("c. Update Plan")
            print("d. Delete Plan")

            choice1 = input("Enter your choice: ")

            if choice1 == "a":
                try:
                    plan_name=input("Plan Name :")
                    duration_months=int(input("Duration in
months :"))
```

```
                cost=float(input("Enter cost :"))
                add_plan(plan_name,duration_months,cost)
            except ValueError as err:
                print(f"Error due to wrong type value in input:
{err}")

        elif choice1=="b":
            view_plans()

        elif choice1=="c":
            try:
                plan_id=int(input("Enter Plan id:"))
                print("Leave fields blank if you don't want to
update them.")
                plan_name=input("Enter Plan Name :")
                duration_months=input("Enter duration in
months :")
                duration_months=int(duration_months) if
duration_months else None
                cost=input("Enter cost :")
                cost=float(cost) if cost else None
                update_plan(plan_id,plan_name,duration_month
s,cost)
            except ValueError as err:
                print(f"Error due to wrong type value in input:
{err}")

        elif choice1 =='d':
            try:
                plan_id = int(input("Enter plan ID to delete: "))
                delete_plan(plan_id)
            except ValueError as err:
                print(f"Error due to wrong type value in input:
{err}")

        else:
            print("Invalid choice! Please try again.")

    elif choice =="3":
        print("Type the letter associated with objective")
        print("a. Add Attendance")
        print("b. View Attendance")
        print("c. Delete Attendance")



        choice1 = input("Enter your choice: ")

        if choice1 == "a":
            try:
                member_id = int(input("Enter member ID: "))
                date =input(" Present on (YYYY-MM-DD):")
                add_attendance(member_id, date)
            except ValueError as err:
                print(f"Error due to wrong type value in input:
{err}")

        elif choice1 == "b":
            view_attendance()

        elif choice1 =="c":
            try:
                attendance_id = int(input("Enter Attendance ID
to delete: "))
                delete_attendance(attendance_id)
            except ValueError as err:
                print(f"Error due to wrong type value in input:
{err}")

        else:
            print("Invalid choice! Please try again.")

    elif choice =="4":
```

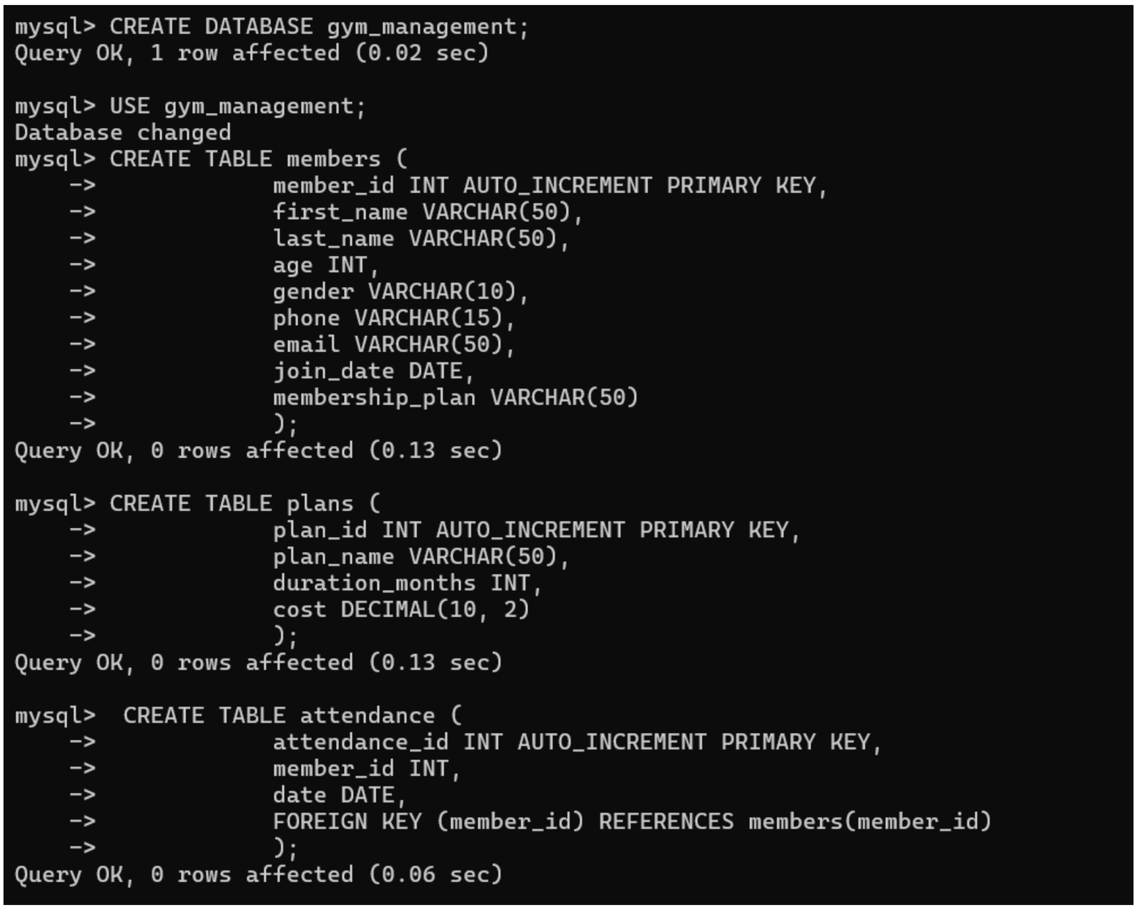
```
        print("\nManage Payments")
        print("a. Add Payment")
        print("b. View Payments")
        print("c. Delete Payment")

        choice1 = input("Enter your choice: ")

        if choice1 == 'a':
            try:
                member_id = int(input("Enter member ID: "))
                amount = float(input("Enter amount: "))
                add_payment(member_id, amount)
            except ValueError as err:
                print(f"Error due to wrong type value in input:
{err}")

        elif choice1 == 'b':
            view_payments()

        elif choice1 == 'c':
            try:
                payment_id = int(input("Enter payment ID to
delete: "))
                delete_payment(payment_id)
            except ValueError as err:
                print(f"Error due to wrong type value in input:
{err}")

        else:
            print("Invalid choice! Please try again.")

    elif choice=="5":

        print("Type the letter associated with objective")
        print("a. Add Staff")
        print("b. View Staff")
        print("c. Update Staff")
        print("d. Delete Staff")



        choice1 = input("Enter your choice: ")

        if choice1 == "a":
            try:
                name = input("Enter Name: ")
                position = input("Enter position: ")
                contact_number = input("Enter contact number:
")
```

```
                hire_date = input("Enter hire date (YYYY-MM-
DD)")
                amount = float(input("Enter amount: "))
                add_staff(name, position, contact_number,
hire_date, amount)
            except ValueError as err:
                print(f"Error due to wrong type value in input:
{err}")

        elif choice1 == "b":
            view_staff()

        elif choice1 == "c":
            try:
                staff_id = int(input("Enter Staff ID: "))
                print("Leave fields blank if you don't want to
update them.")
                name = input("Enter Name: ")
                position = input("Enter position: ")
                contact_number = input("Enter contact number:
")
                update_staff(staff_id, name, position,
contact_number)
            except ValueError as err:
                print(f"Error due to wrong type value in input: 
{err}")

        elif choice1 == "d":
            try:
                staff_id = int(input("Enter Staff ID to delete: "))
                delete_staff(staff_id)
            except ValueError as err:
                print(f"Error due to wrong type value in input: 
{err}")

        else:
            print("Invalid choice! Please try again.")

    elif choice == "6":
        print("\nManage Equipment")
        print("a. Add Equipment")
        print("b. View Equipment")
        print("c. Update Equipment")
        print("d. Delete Equipment")

        choice = input("Enter your choice: ")

        if choice == 'a':
```

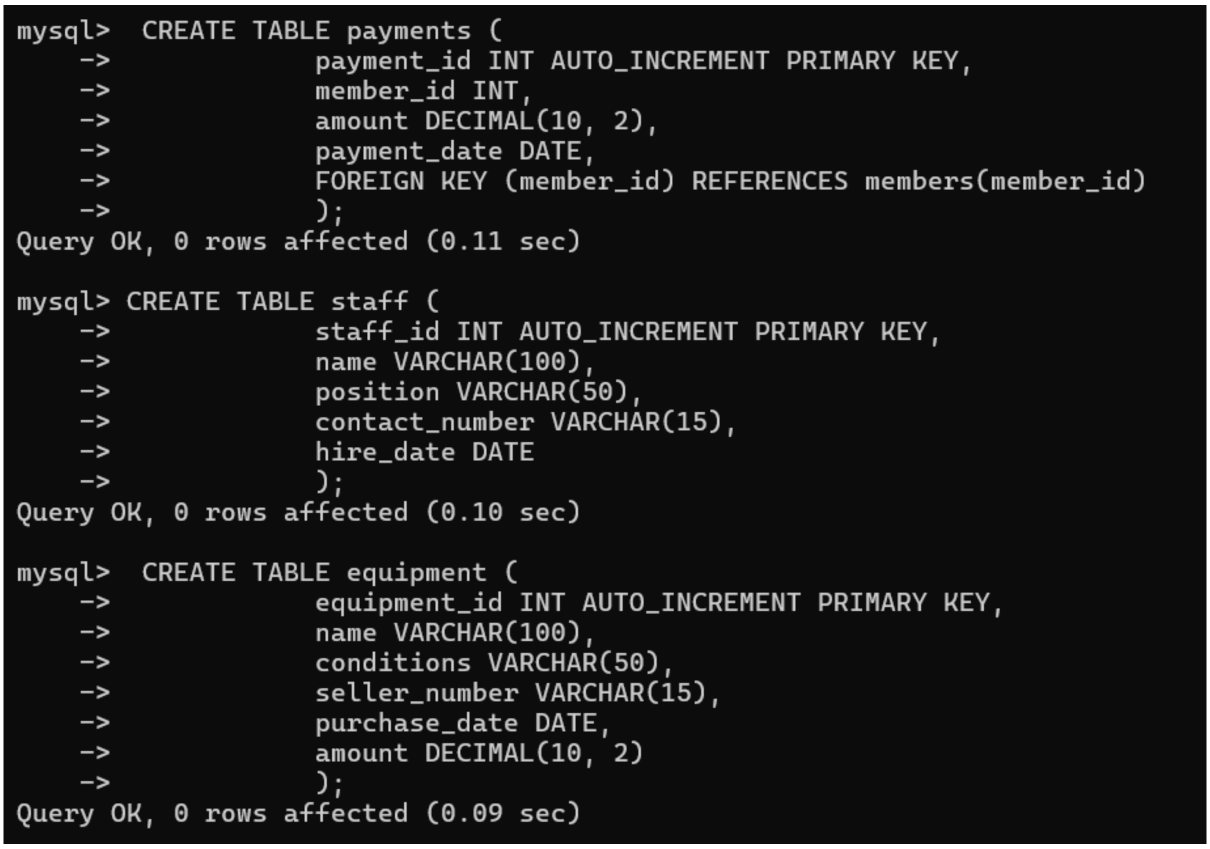
```
        try:
            name = input("Enter equipment name: ")
            purchase_date = input("Enter purchase date
(YYYY-MM-DD): ")
            conditions = input("Enter condition: ")
            seller_number = int(input("Enter the seller
code:"))
            amount = float(input("Enter amount of
equipment: "))
            add_equipment(name, conditions,
seller_number, purchase_date, amount)
        except ValueError as err:
            print(f"Error due to wrong type value in input:
{err}")

    elif choice == 'b':
        view_equipment()

    elif choice == 'c':
        try:
            equipment_id = int(input("Enter equipment ID
to update: "))
            name = input("Enter new name: ")
            purchase_date = input("Enter new purchase date
(YYYY-MM-DD): ")
```

```
                conditions = input("Enter new condition: ")
                seller_number = input("Enter the new seller
code:")
                amount = float(input("Enter amount of
equipment: "))
                update_equipment(equipment_id, name,
conditions, seller_number, purchase_date, amount)
            except ValueError as err:
                print(f"Error due to wrong type value in input:
{err}")

        elif choice == 'd':
            try:
                equipment_id = int(input("Enter equipment ID
to delete: "))
                delete_equipment(equipment_id)
            except ValueError as err:
                print(f"Error due to wrong type value in input:
{err}")

        else:
            print("Invalid choice! Please try again.")

    elif choice == "7":
        print("Thank You for using Gym Management
System")
        break


    else:
        print("Invalid choice! Please try again.")


except mysql.connector.Error as err:
    print(f"Incorrect password entered: {err}")
```

CREATING DATABASE AND TABLES IN MYSQL



```
mysql> CREATE DATABASE gym_management;
Query OK, 1 row affected (0.02 sec)

mysql> USE gym_management;
Database changed
mysql> CREATE TABLE members (
    ->             member_id INT AUTO_INCREMENT PRIMARY KEY,
    ->             first_name VARCHAR(50),
    ->             last_name VARCHAR(50),
    ->             age INT,
    ->             gender VARCHAR(10),
    ->             phone VARCHAR(15),
    ->             email VARCHAR(50),
    ->             join_date DATE,
    ->             membership_plan VARCHAR(50)
    ->             );
Query OK, 0 rows affected (0.13 sec)

mysql> CREATE TABLE plans (
    ->             plan_id INT AUTO_INCREMENT PRIMARY KEY,
    ->             plan_name VARCHAR(50),
    ->             duration_months INT,
    ->             cost DECIMAL(10, 2)
    ->             );
Query OK, 0 rows affected (0.13 sec)

mysql>  CREATE TABLE attendance (
    ->             attendance_id INT AUTO_INCREMENT PRIMARY KEY,
    ->             member_id INT,
    ->             date DATE,
    ->             FOREIGN KEY (member_id) REFERENCES members(member_id)
    ->             );
Query OK, 0 rows affected (0.06 sec)
```



```
mysql>  CREATE TABLE payments (
    ->             payment_id INT AUTO_INCREMENT PRIMARY KEY,
    ->             member_id INT,
    ->             amount DECIMAL(10, 2),
    ->             payment_date DATE,
    ->             FOREIGN KEY (member_id) REFERENCES members(member_id)
    ->             );
Query OK, 0 rows affected (0.11 sec)

mysql> CREATE TABLE staff (
    ->             staff_id INT AUTO_INCREMENT PRIMARY KEY,
    ->             name VARCHAR(100),
    ->             position VARCHAR(50),
    ->             contact_number VARCHAR(15),
    ->             hire_date DATE
    ->             );
Query OK, 0 rows affected (0.10 sec)

mysql>  CREATE TABLE equipment (
    ->             equipment_id INT AUTO_INCREMENT PRIMARY KEY,
    ->             name VARCHAR(100),
    ->             conditions VARCHAR(50),
    ->             seller_number VARCHAR(15),
    ->             purchase_date DATE,
    ->             amount DECIMAL(10, 2)
    ->             );
Query OK, 0 rows affected (0.09 sec)
```

OUTPUT

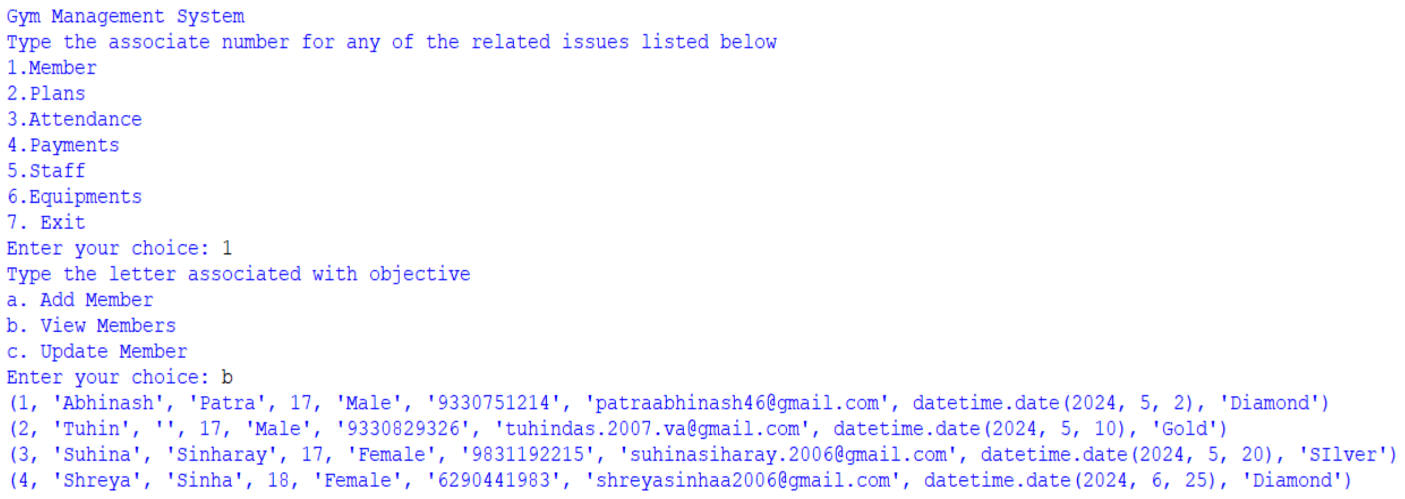
Accessing the management panel by admin login

```
Admin Login
Enter your SQL User ID: root
Enter your SQL password: 1234

Gym Management System
Type the associate number for any of the related issues listed below
1.Member
2.Plans
3.Attendance
4.Payments
5.Staff
6.Equipments
7. Exit
Enter your choice: 1
```

Adding a member details

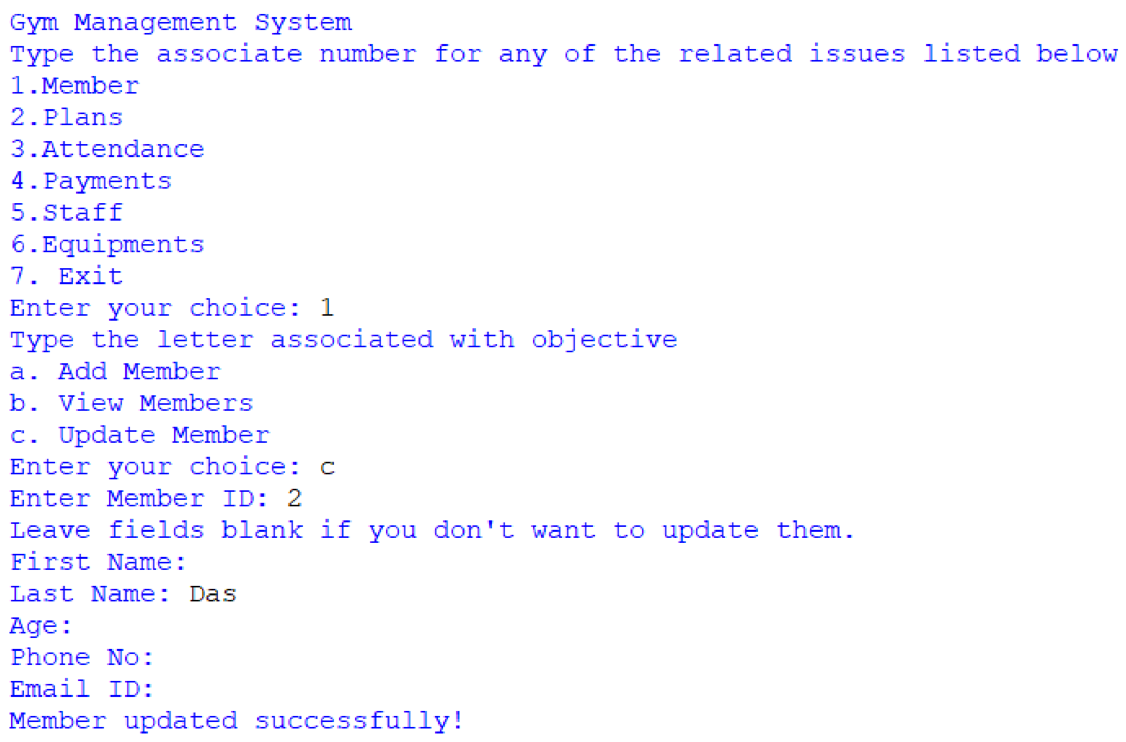
```
Gym Management System
Type the associate number for any of the related issues listed below
1.Member
2.Plans
3.Attendance
4.Payments
5.Staff
6.Equipments
7. Exit
Enter your choice: 1
Type the letter associated with objective
a. Add Member
b. View Members
c. Update Member
Enter your choice: a
First Name: Abhinash
Last Name: Patra
Age: 17
Gender: Male
Phone No: 9330751214
Email Id: patraabhinash46@gmail.com
Join Date (YYYY-MM-DD): 2024-05-02
Membership Plan (Silver/Gold/Diamond): Diamond
Member added successfully!
```

Viewing members details

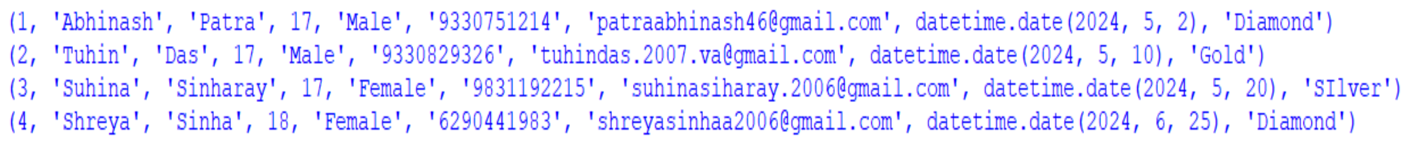


```
Gym Management System
Type the associate number for any of the related issues listed below
1.Member
2.Plans
3.Attendance
4.Payments
5.Staff
6.Equipments
7. Exit
Enter your choice: 1
Type the letter associated with objective
a. Add Member
b. View Members
c. Update Member
Enter your choice: b
(1, 'Abhinash', 'Patra', 17, 'Male', '9330751214', 'patraabhinash46@gmail.com', datetime.date(2024, 5, 2), 'Diamond')
(2, 'Tuhin', '', 17, 'Male', '9330829326', 'tuhindas.2007.va@gmail.com', datetime.date(2024, 5, 10), 'Gold')
(3, 'Suhina', 'Sinharay', 17, 'Female', '9831192215', 'suhinasiharay.2006@gmail.com', datetime.date(2024, 5, 20), 'SIlver')
(4, 'Shreya', 'Sinha', 18, 'Female', '6290441983', 'shreyasinhaa2006@gmail.com', datetime.date(2024, 6, 25), 'Diamond')
```

Updating a member details



```
Gym Management System
Type the associate number for any of the related issues listed below
1.Member
2.Plans
3.Attendance
4.Payments
5.Staff
6.Equipments
7. Exit
Enter your choice: 1
Type the letter associated with objective
a. Add Member
b. View Members
c. Update Member
Enter your choice: c
Enter Member ID: 2
Leave fields blank if you don't want to update them.
First Name:
Last Name: Das
Age:
Phone No:
Email ID:
Member updated successfully!
```



```
(1, 'Abhinash', 'Patra', 17, 'Male', '9330751214', 'patraabhinash46@gmail.com', datetime.date(2024, 5, 2), 'Diamond')
(2, 'Tuhin', 'Das', 17, 'Male', '9330829326', 'tuhindas.2007.va@gmail.com', datetime.date(2024, 5, 10), 'Gold')
(3, 'Suhina', 'Sinharay', 17, 'Female', '9831192215', 'suhinasiharay.2006@gmail.com', datetime.date(2024, 5, 20), 'SIlver')
(4, 'Shreya', 'Sinha', 18, 'Female', '6290441983', 'shreyasinhaa2006@gmail.com', datetime.date(2024, 6, 25), 'Diamond')
```

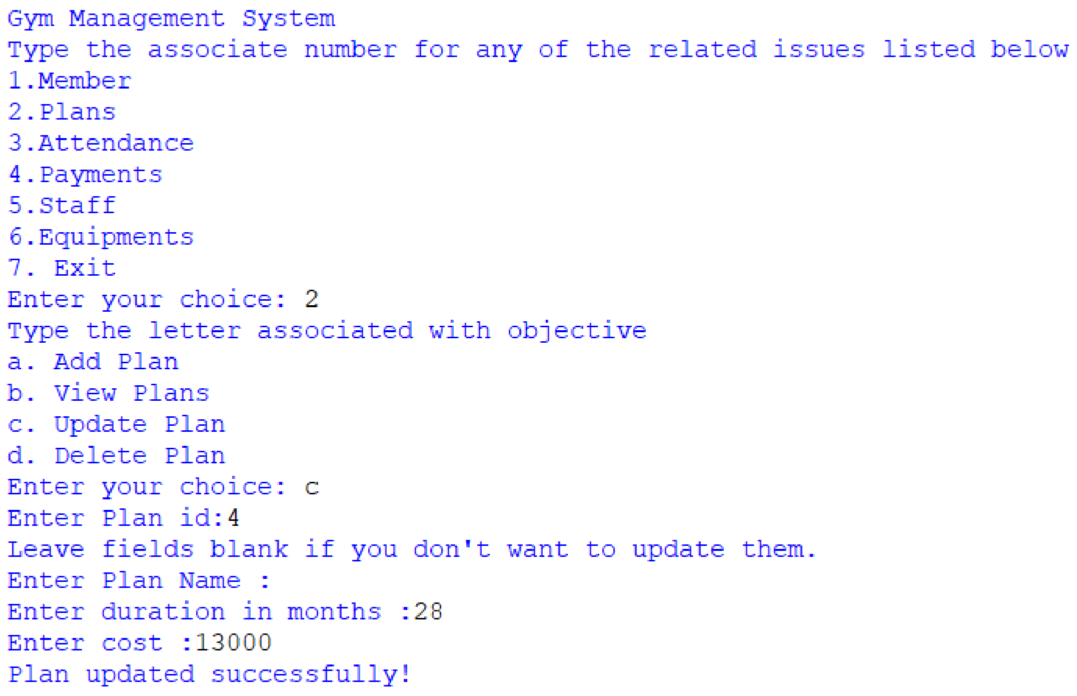
Adding a plan detail

```
Gym Management System
Type the associate number for any of the related issues listed below
1.Member
2.Plans
3.Attendance
4.Payments
5.Staff
6.Equipments
7. Exit
Enter your choice: 2
Type the letter associated with objective
a. Add Plan
b. View Plans
c. Update Plan
d. Delete Plan
Enter your choice: a
Plan Name :Silver
Duration in months :3
Enter cost :1500
Plan added successfully!
```

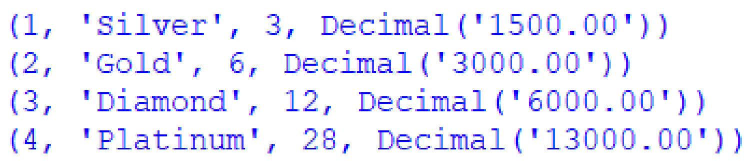
Viewing plan details

```
Gym Management System
Type the associate number for any of the related issues listed below
1.Member
2.Plans
3.Attendance
4.Payments
5.Staff
6.Equipments
7. Exit
Enter your choice: 2
Type the letter associated with objective
a. Add Plan
b. View Plans
c. Update Plan
d. Delete Plan
Enter your choice: b
(1, 'Silver', 3, Decimal('1500.00'))
(2, 'Gold', 6, Decimal('3000.00'))
(3, 'Diamond', 12, Decimal('6000.00'))
(4, 'Platinum', 24, Decimal('12000.00'))
```

Updating a plan details

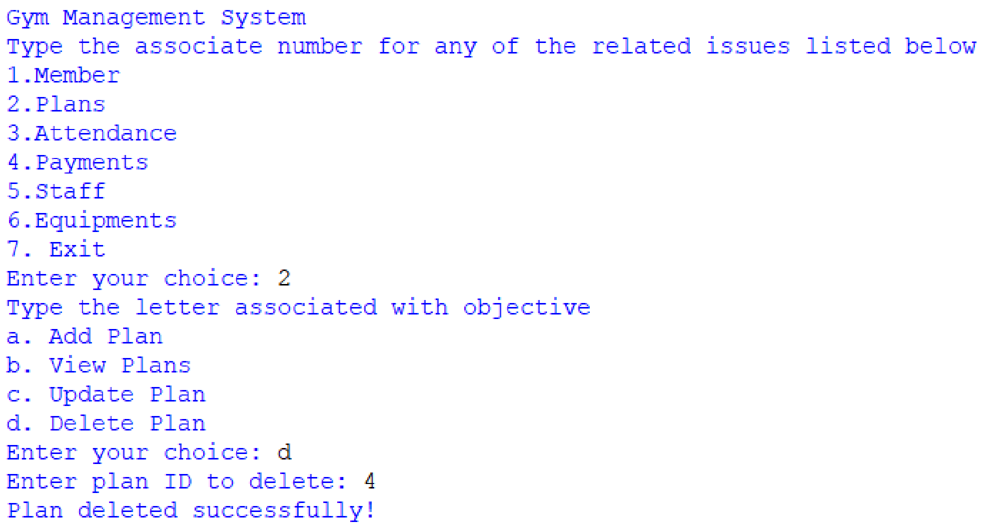


```
Gym Management System
Type the associate number for any of the related issues listed below
1.Member
2.Plans
3.Attendance
4.Payments
5.Staff
6.Equipments
7. Exit
Enter your choice: 2
Type the letter associated with objective
a. Add Plan
b. View Plans
c. Update Plan
d. Delete Plan
Enter your choice: c
Enter Plan id:4
Leave fields blank if you don't want to update them.
Enter Plan Name :
Enter duration in months :28
Enter cost :13000
Plan updated successfully!
```

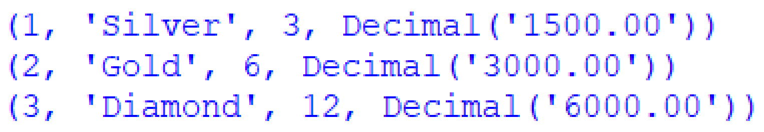


```
(1, 'Silver', 3, Decimal('1500.00'))
(2, 'Gold', 6, Decimal('3000.00'))
(3, 'Diamond', 12, Decimal('6000.00'))
(4, 'Platinum', 28, Decimal('13000.00'))
```

Deleting plans details



```
Gym Management System
Type the associate number for any of the related issues listed below
1.Member
2.Plans
3.Attendance
4.Payments
5.Staff
6.Equipments
7. Exit
Enter your choice: 2
Type the letter associated with objective
a. Add Plan
b. View Plans
c. Update Plan
d. Delete Plan
Enter your choice: d
Enter plan ID to delete: 4
Plan deleted successfully!
```



```
(1, 'Silver', 3, Decimal('1500.00'))
(2, 'Gold', 6, Decimal('3000.00'))
(3, 'Diamond', 12, Decimal('6000.00'))
```

Adding attendance of the members

```
Gym Management System
Type the associate number for any of the related issues listed below
1.Member
2.Plans
3.Attendance
4.Payments
5.Staff
6.Equipments
7. Exit
Enter your choice: 3
Type the letter associated with objective
a. Add Attendance
b. View Attendance
c. Delete Attendance
Enter your choice: a
Enter member ID: 1
 Present on (YYYY-MM-DD):2024-07-08
Attendance added successfully!
```

Viewing attendance of the members

```
Gym Management System
Type the associate number for any of the related issues listed below
1.Member
2.Plans
3.Attendance
4.Payments
5.Staff
6.Equipments
7. Exit
Enter your choice: 3
Type the letter associated with objective
a. Add Attendance
b. View Attendance
c. Delete Attendance
Enter your choice: b
(1, datetime.date(2024, 7, 8), 'Abhinash', 'Patra')
(2, datetime.date(2024, 6, 12), 'Tuhin', 'Das')
(3, datetime.date(2024, 7, 8), 'Suhina', 'Sinharay')
(4, datetime.date(2024, 7, 17), 'Shreya', 'Sinha')
```

Deleting attendance

```
Gym Management System
Type the associate number for any of the related issues listed below
1.Member
2.Plans
3.Attendance
4.Payments
5.Staff
6.Equipments
7. Exit
Enter your choice: 3
Type the letter associated with objective
a. Add Attendance
b. View Attendance
c. Delete Attendance
Enter your choice: c
Enter Attendance ID to delete: 2
Attendance deleted successfully!

(1, datetime.date(2024, 7, 8), 'Abhinash', 'Patra')
(3, datetime.date(2024, 7, 8), 'Suhina', 'Sinharay')
(4, datetime.date(2024, 7, 17), 'Shreya', 'Sinha')
```

Adding payments details of the members

```
Gym Management System
Type the associate number for any of the related issues listed below
1.Member
2.Plans
3.Attendance
4.Payments
5.Staff
6.Equipments
7. Exit
Enter your choice: 4

Manage Payments
a. Add Payment
b. View Payments
c. Delete Payment
Enter your choice: a
Enter member ID: 1
Enter amount: 12000
Payment added successfully!
```

Viewing payments details

```
Gym Management System
Type the associate number for any of the related issues listed below
1.Member
2.Plans
3.Attendance
4.Payments
5.Staff
6.Equipments
7. Exit
Enter your choice: 4

Manage Payments
a. Add Payment
b. View Payments
c. Delete Payment
Enter your choice: b
(1, 1, Decimal('12000.00'), datetime.date(2024, 9, 22))
(2, 2, Decimal('3000.00'), datetime.date(2024, 9, 22))
(3, 3, Decimal('1500.00'), datetime.date(2024, 9, 22))
(4, 4, Decimal('12000.00'), datetime.date(2024, 9, 22))
```

Deleting a payments details

```
Gym Management System
Type the associate number for any of the related issues listed below
1.Member
2.Plans
3.Attendance
4.Payments
5.Staff
6.Equipments
7. Exit
Enter your choice: 4

Manage Payments
a. Add Payment
b. View Payments
c. Delete Payment
Enter your choice: c
Enter payment ID to delete: 2
Payment deleted successfully!
```

```
(1, 1, Decimal('12000.00'), datetime.date(2024, 9, 22))
(3, 3, Decimal('1500.00'), datetime.date(2024, 9, 22))
(4, 4, Decimal('12000.00'), datetime.date(2024, 9, 22))
```

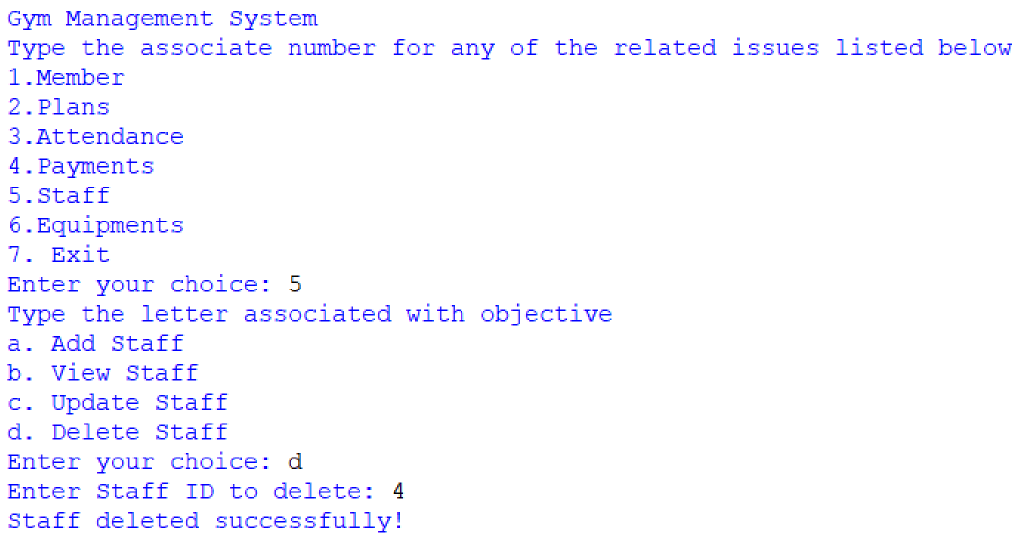
Adding a staff details

```
Gym Management System
Type the associate number for any of the related issues listed below
1.Member
2.Plans
3.Attendance
4.Payments
5.Staff
6.Equipments
7. Exit
Enter your choice: 5
Type the letter associated with objective
a. Add Staff
b. View Staff
c. Update Staff
d. Delete Staff
Enter your choice: a
Enter Name: Gaurav Singh
Enter position: Head Trainer
Enter contact number: 9335433805
Joining Date (YYYY-MM-DD)2020-05-06
Enter Salary: 15000
Staff added successfully!
```

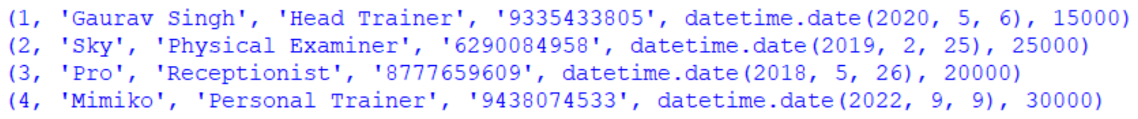
Viewing staff details

```
Gym Management System
Type the associate number for any of the related issues listed below
1.Member
2.Plans
3.Attendance
4.Payments
5.Staff
6.Equipments
7. Exit
Enter your choice: 5
Type the letter associated with objective
a. Add Staff
b. View Staff
c. Update Staff
d. Delete Staff
Enter your choice: a
Enter Name: Gaurav Singh
Enter position: Head Trainer
Enter contact number: 9335433805
Joining Date (YYYY-MM-DD)2020-05-06
Enter Salary: 15000
Staff added successfully!
```

Updating a staff details

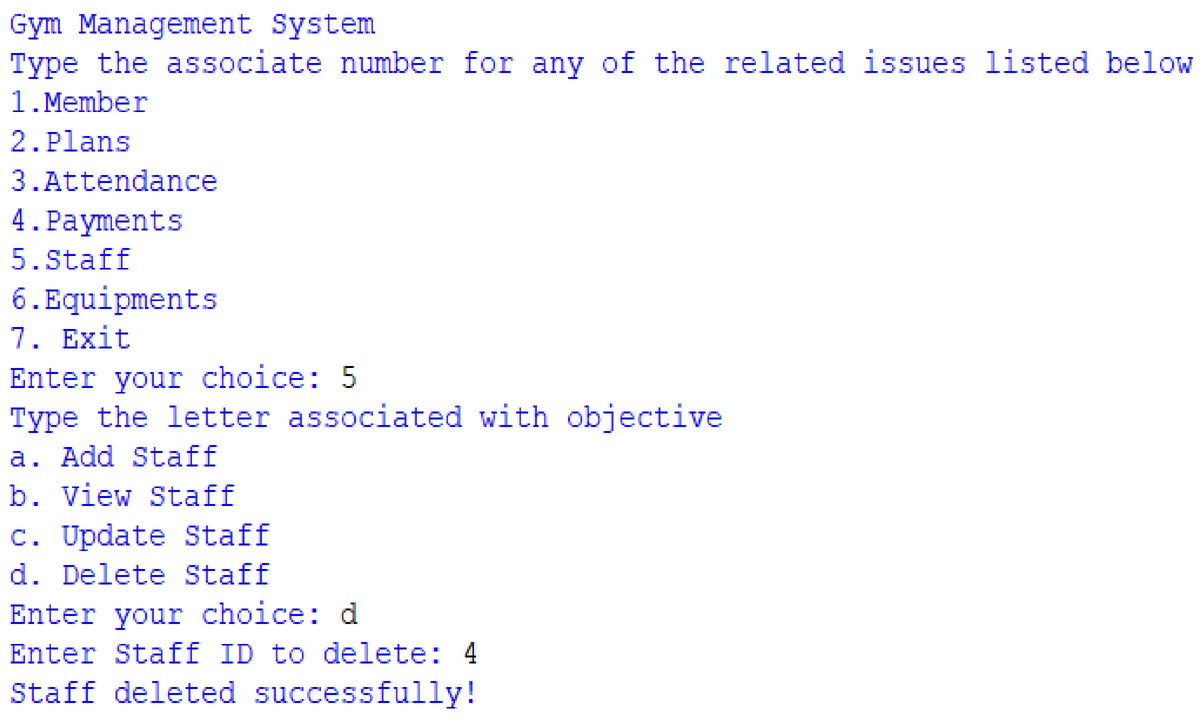


```
Gym Management System
Type the associate number for any of the related issues listed below
1.Member
2.Plans
3.Attendance
4.Payments
5.Staff
6.Equipments
7. Exit
Enter your choice: 5
Type the letter associated with objective
a. Add Staff
b. View Staff
c. Update Staff
d. Delete Staff
Enter your choice: d
Enter Staff ID to delete: 4
Staff deleted successfully!
```

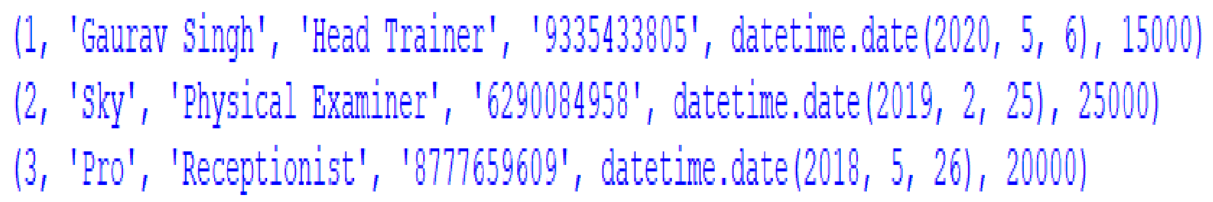


```
(1, 'Gaurav Singh', 'Head Trainer', '9335433805', datetime.date(2020, 5, 6), 15000)
(2, 'Sky', 'Physical Examiner', '6290084958', datetime.date(2019, 2, 25), 25000)
(3, 'Pro', 'Receptionist', '8777659609', datetime.date(2018, 5, 26), 20000)
(4, 'Mimiko', 'Personal Trainer', '9438074533', datetime.date(2022, 9, 9), 30000)
```

Deleting a staff details

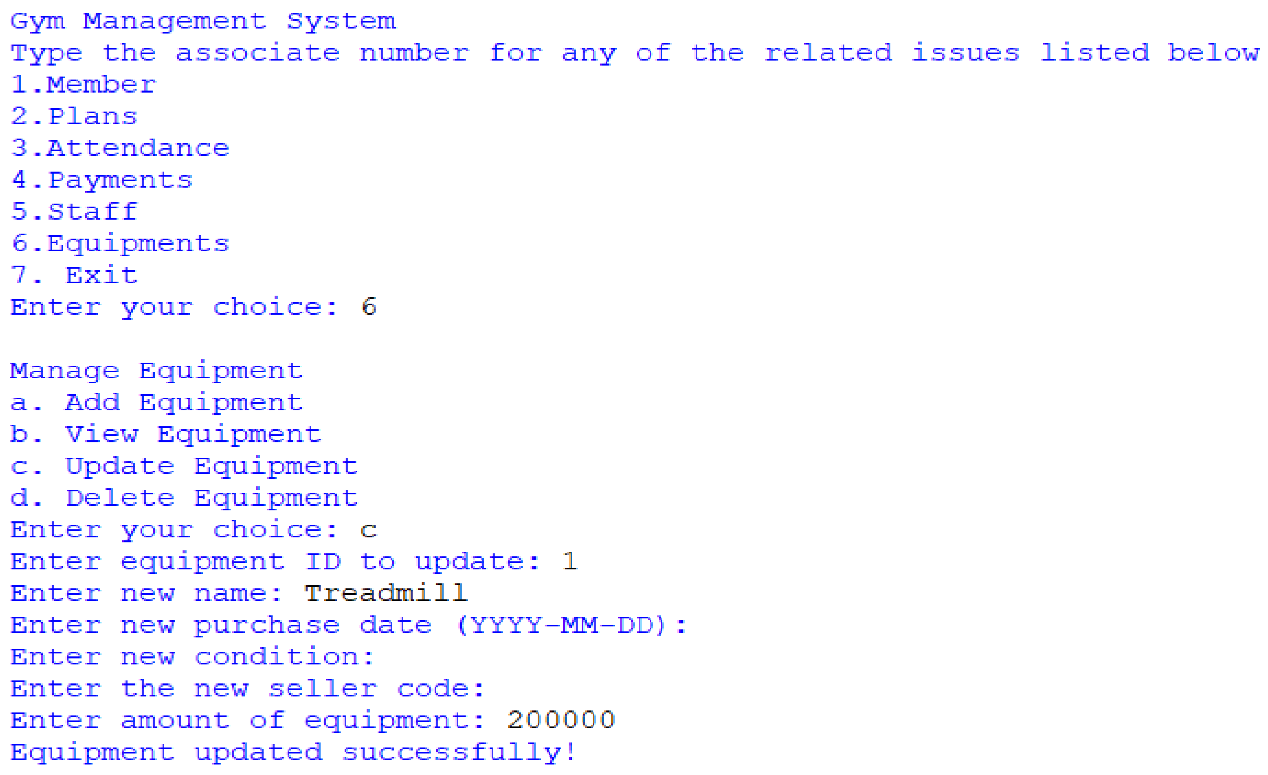


```
Gym Management System
Type the associate number for any of the related issues listed below
1.Member
2.Plans
3.Attendance
4.Payments
5.Staff
6.Equipments
7. Exit
Enter your choice: 5
Type the letter associated with objective
a. Add Staff
b. View Staff
c. Update Staff
d. Delete Staff
Enter your choice: d
Enter Staff ID to delete: 4
Staff deleted successfully!
```



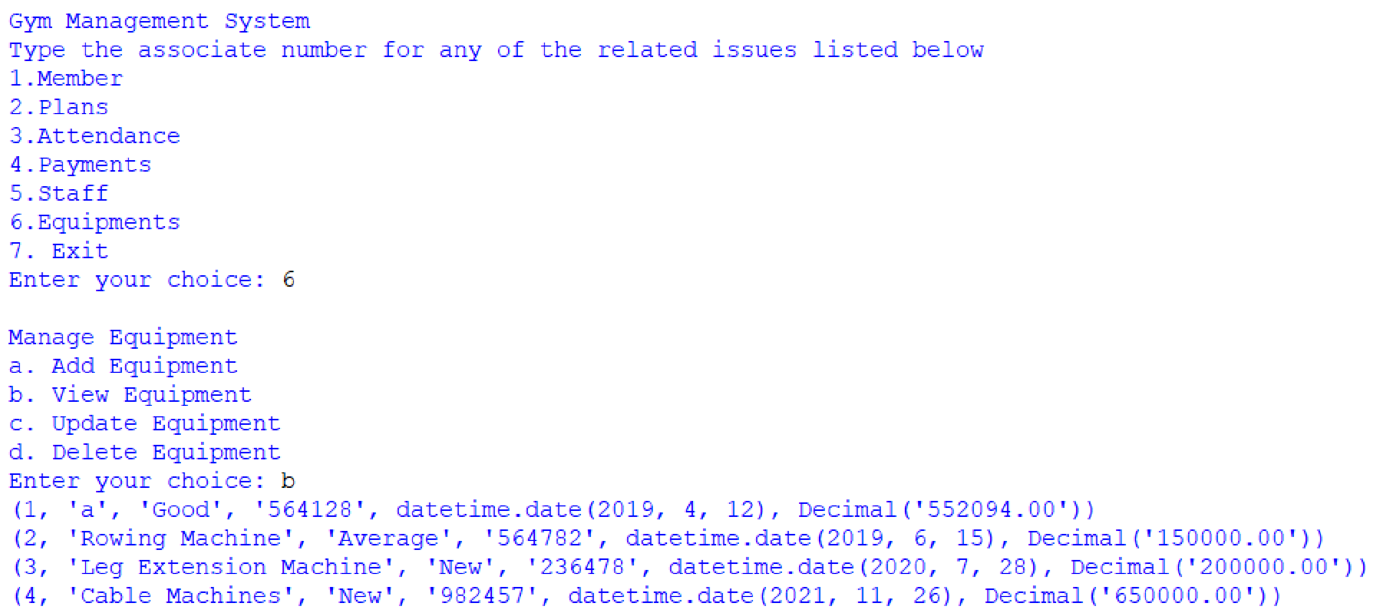
```
(1, 'Gaurav Singh', 'Head Trainer', '9335433805', datetime.date(2020, 5, 6), 15000)
(2, 'Sky', 'Physical Examiner', '6290084958', datetime.date(2019, 2, 25), 25000)
(3, 'Pro', 'Receptionist', '8777659609', datetime.date(2018, 5, 26), 20000)
```

Adding a equipment details



```
Gym Management System
Type the associate number for any of the related issues listed below
1.Member
2.Plans
3.Attendance
4.Payments
5.Staff
6.Equipments
7. Exit
Enter your choice: 6

Manage Equipment
a. Add Equipment
b. View Equipment
c. Update Equipment
d. Delete Equipment
Enter your choice: c
Enter equipment ID to update: 1
Enter new name: Treadmill
Enter new purchase date (YYYY-MM-DD):
Enter new condition:
Enter the new seller code:
Enter amount of equipment: 200000
Equipment updated successfully!
```

Viewing equipment details



```
Gym Management System
Type the associate number for any of the related issues listed below
1.Member
2.Plans
3.Attendance
4.Payments
5.Staff
6.Equipments
7. Exit
Enter your choice: 6

Manage Equipment
a. Add Equipment
b. View Equipment
c. Update Equipment
d. Delete Equipment
Enter your choice: b
(1, 'a', 'Good', '564128', datetime.date(2019, 4, 12), Decimal('552094.00'))
(2, 'Rowing Machine', 'Average', '564782', datetime.date(2019, 6, 15), Decimal('150000.00'))
(3, 'Leg Extension Machine', 'New', '236478', datetime.date(2020, 7, 28), Decimal('200000.00'))
(4, 'Cable Machines', 'New', '982457', datetime.date(2021, 11, 26), Decimal('650000.00'))
```

Updating a equipment details

```
Gym Management System
Type the associate number for any of the related issues listed below
1.Member
2.Plans
3.Attendance
4.Payments
5.Staff
6.Equipments
7. Exit
Enter your choice: 6

Manage Equipment
a. Add Equipment
b. View Equipment
c. Update Equipment
d. Delete Equipment
Enter your choice: c
Enter equipment ID to update: 1
Enter new name: Treadmill
Enter new purchase date (YYYY-MM-DD):
Enter new condition:
Enter the new seller code:
Enter amount of equipment: 200000
Equipment updated successfully!
```

```
(1, 'Treadmill', 'Good', '564128', datetime.date(2019, 4, 12), Decimal('200000.00'))
(2, 'Rowing Machine', 'Average', '564782', datetime.date(2019, 6, 15), Decimal('150000.00'))
(3, 'Leg Extension Machine', 'New', '236478', datetime.date(2020, 7, 28), Decimal('200000.00'))
(4, 'Cable Machines', 'New', '982457', datetime.date(2021, 11, 26), Decimal('650000.00'))
```

Deleting a equipment details

```
Gym Management System
Type the associate number for any of the related issues listed below
1.Member
2.Plans
3.Attendance
4.Payments
5.Staff
6.Equipments
7. Exit
Enter your choice: 6

Manage Equipment
a. Add Equipment
b. View Equipment
c. Update Equipment
d. Delete Equipment
Enter your choice: d
Enter equipment ID to delete: 1
Equipment deleted successfully!
```

```
(2, 'Rowing Machine', 'Average', '564782', datetime.date(2019, 6, 15), Decimal('150000.00'))
(3, 'Leg Extension Machine', 'New', '236478', datetime.date(2020, 7, 28), Decimal('200000.00'))
(4, 'Cable Machines', 'New', '982457', datetime.date(2021, 11, 26), Decimal('650000.00'))
```

Exiting the management system

```
Gym Management System
Type the associate number for any of the related issues listed below
1.Member
2.Plans
3.Attendance
4.Payments
5.Staff
6.Equipments
7. Exit
Enter your choice: 7
Thank You for using Gym Management System
```

TESTING

Software Testing is an empirical investigation conducted to provide stakeholders with information about the quality of the product or service under test[1] , with respect to the context in which it is intended to operate. Software Testing also provides an objective, independent view of the software to allow the business to appreciate and understand the risks at implementation of the software. Test techniques include, but are not limited to, the process of executing a program or application with the intent of finding software bugs.

It can also be stated as the process of validating and verifying that a software program/application/product meets the business and technical requirements that guided its design and development, so that it works as expected and can be implemented with the same characteristics. Software Testing, depending on the testing method employed, can be implemented at any time in the development process, however the most test effort is employed after the requirements have been defined and coding process has been completed.

TESTING METHODS

Software testing methods are traditionally divided into black box testing and white box testing. These two approaches are used to describe the point of view that a test engineer takes when designing test cases.

BLACK BOX TESTING

Black box testing treats the software as a "black box," without any knowledge of internal implementation. Black box testing methods include: equivalence partitioning, boundary value analysis, all-pairs testing, fuzz testing, model-based testing, traceability matrix, exploratory testing and specification-based testing.

SPECIFICATION BASED TESTING

Specification-based testing aims to test the functionality of software according to the applicable requirements.[16] Thus, the tester inputs data into, and only sees the output from, the test object. This level of testing usually requires thorough test cases to be provided to the tester, who then can simply verify that for a given input, the output value (or behaviour), either "is" or "is not" the same as the expected value specified in the test case. Specification-based testing is necessary, but it is insufficient to guard against certain risks.

ADVANTAGES AND DISADVANTAGES

The black box tester has no "bonds" with the code, and a tester's perception is very simple: a code must have bugs. Using the principle, "Ask and you shall receive," black box testers find bugs where programmers don't. But, on the other hand, black box testing has been said to be "like a walk in a dark labyrinth without a flashlight," because the tester doesn't know how the software being tested was actually constructed.

That's why there are situations when (1) a black box tester writes many test cases to check something that can be tested by only one test case, and/or (2) some parts of the back end are not tested at all. Therefore, black box testing has the advantage of "an unaffiliated opinion," on the one hand, and the disadvantage of "blind exploring," on the other.

WHITE BOX TESTING

White box testing, by contrast to black box testing, is when the tester has access to the internal data structures and algorithms (and the code that implement these).

Types of white box testing :-

The following types of white box testing exist:

- Api testing - Testing of the application using Public and Private APIs.
- Code coverage - creating tests to satisfy some criteria of code coverage.

For example, the test designer can create tests to cause all statements in the program to be executed at least once.

- Fault injection methods.
- Mutation testing methods.
- Static testing - White box testing includes all static testing.

CODE COMPLETENESS EVALUATION

White box testing methods can also be used to evaluate the completeness of a test suite that was created with black box testing methods. This allows the software team to examine parts of a system that are rarely tested and ensures that the most important function points have been tested.

Two common forms of code coverage are:

- Function Coverage: Which reports on functions executed and
- Statement Coverage: Which reports on the number of lines executed to complete the test. They both return coverage metric, measured as a percentage.

BIBLIOGRAPHY

- Computer science with python-class XII By: Sumita Arora
- https://www.geeksforgeeks.org
- https://www.w3schools.com
- https://github.com