KENDRIYA VIDYALAYA JANAKPURI

तत् त्वं पूषन् अपावृणु

केन्द्रीय विद्यालय संगठन

PROJECT DETAILS

PROJECT: GYM MANAGEMENT SYSTEM USING PYTHON AND MYSQL

GROUP LEAD: ADARSH (12A – 03), ROLL NO: 26633791

GROUP MEMBERS:

- ADARSH (12A – 03)
  ROLL NO: 26633791
- ANKIT KUMAR (12A – 11)
  ROLL NO: 26633799
- DHRUV VERMA (12A – 21)
  ROLL NO: 26633807
- SHUBHAM KUMAR (12A – 47)
  ROLL NO: 26633836

INDEX

CERTIFICATE

This is to Certify that, Adarsh, Ankit Kumar Dhruv Verma, Shubham Kumar have successfully completed the project on the topic “GYM MANAGEMENT SYSTEM USING MYSQL AND PYTHON”, under my guidance.

Teacher’s Signature:

ACKNOWLEDGEMENT

We would like to thank our teacher Mrs. Sonia Sharma for helping us with this project. She allowed us to work on this project. Along with that, we would also like to give our special thanks to our school principal wholeheartedly to provide us with resources.

WHAT IS PYTHON?

- *-Written by Ankit Kumar*

Technically defined, Python is an object-oriented, interpreted, high-level programming language, developed by Guido van Rossum and originally released in 1991. ‘Python’ has an easy-going ring to it and so is its operation. Python has a reputation for being a beginner-friendly language.

One answer to the question of ‘What is Python’ can be that it is a language that is now replacing Java for it handles the complexity for the user, allows the newbies to focus on grasping the concepts, and produce result-driven codes.

The use case of python includes server-side web development, software development, mathematics and system scripting. Python is popular for Rapid Application Development. Did you know that

Python is also called a ‘Glue Language’ as it is also used to tie existing components? Python has made it possible because of its high-level, built-in data structures, dynamic binding and typing.

One more reason why Python is so famous is that with its use, program maintenance costs go down. It happens due to the easily learned syntax and emphasis on readability.

One more thing that makes Python so special is that it is an open-source language; every day, hundreds and thousands of programmers are building libraries and functionalities to make it even better.

To answer the question of ‘What is Python?’, we need to travel back the time and understand its history too.

The founder of the Python Programming Language is Guido van Rossum, who rolled out this language for the first time in 1991.

While creating Python, the only vision he had was that he wanted to build something that could resolve the issues the programmers were facing at that time. It took him 5 years, but yes, he rolled out the first version of Python Programming Language.

Python solved the dual purpose. It not only resolved the problems, the programmers were facing, but also had an easy-to-understand and readable syntax. The founder named this programming language ‘Python’ in honor of his favorite comedian, ‘The Pythons’ in the ‘Monty Python’s Flying Circus’ show.

It was until Google announced that it has used Python for its internal development programming that Python started getting recognized for what a wonder it is.

Thereafter, many programmers and developers around the world started using Python as their primary programming language. It became so prominent that it eventually became the best programming language for any Data Science Projects, Machine Learning Algorithms, Calculations, and whatnot.

Today, Python is one of the most used programming languages in the world.

There are indeed a number of reasons to choose python over any other language. Here are some of them.

- Popularity

According to the Stack Overflow Developer Survey 2022, Python is the 4th most popular and fastest-growing programming language. It is being used by the companies like Google, Instagram, Netflix, and Spotify.

- Interpretation

Python, being an interpreted language passes straight to the interpreter, making the execution simple and quick; unlike compilers where the machine code has to be generated from the source code before running.

- Opensource

One of the best facts about Python is that it is a free language developed under an OSI-approved open-source license.

- Portability

Major trouble comes in transferring a code from one platform to another without making blunders in the command. Python programming language, being a portable code can easily be transferred without making any errors.

- Simplicity

Python is the only programming language which is quite English-like. It is quite easy to read and understand. Compared to C++ or Java, Python programming language uses fewer keywords. Hence, python language has come to the top of the preferences of developers across the world.

Advantages of Python

- Python is an open-source and free programming language which can be downloaded by anyone at zero cost.
- It is a high-level programming language with an English-like Syntax, which is hence easy to understand for the beginners
- Productivity is comparatively higher as the code is simple.
- Python halts the code whenever there is an error, and keeps it over there until the code is error-free, this helps in avoiding the waste of time while creating error-free code.
- It is a portable code, which means that you do not need to change the code while moving it onto some other platform.
- One of the biggest advantages of python being an open-source programming language is that there are a million developers who are adding to the vast library helping others ease their work.

There are multiple uses of Python. Let’s talk about some of them in brief.

1.) AI and Machine Learning
All thanks to the simple and easy-to-understand syntaxes, Python is considered the best programming language for Artificial Intelligence (AI) and Machine Learning (ML).

2.) Data Analytics
Python has been used to create some of the most popular data mining and analytics tools. Hence, it can be said that it is an excellent tool for data science. Not only the big data, Python can also help organizations learn more about themselves, their offerings and their customers.

3.) Web Development
Python is a versatile backend programming language. Its simplicity is its key factor. Due to its vast usage, there are a number of frameworks that can be used for backend development including Django.

4.) Search Engine Optimization (SEO)
Python also helps SEO professionals automate their tasks and helps them in extracting and analyzing large amounts of data. Python programming language can be used to identify broken links, automate solutions and remove human error.

5.) Blockchain
Along with some of the most prominent programming languages used for blockchain development, like JavaScript, C++, Java, and more, Python is giving them tough competition. Python has high flexibility and functionality, reinforced by its security, which makes it the best choice for the blockchain industry.

6.) Game Development
Python might not be the most popular programming language when it comes to game development, but it is still incredibly useful, owing to its simplicity. Python can be used to build prototypes and develop ideas.

7.) Automation
One of the most popular uses of Python is automation. It can interact with huge sets of data and automate human tasks avoiding any kind of errors. Learning Python programming language would not just save you time, but help you throughout your life.

Which companies use Python?
Here is a list of top companies which use Python in their daily tasks. You might be surprised to see some of the names here on the list!

- Google
- Facebook
- Instagram
- Dropbox
- Spotify
- Reddit
- Uber
- Netflix

python™

WHAT IS MYSQL?

-Written by Dhruv Verma

MySQL is the world's most popular open-source database. According to DB-Engines, MySQL ranks as the second-most-popular database, behind Oracle Database. MySQL powers many of the most accessed applications, including Facebook, Twitter, Netflix, Uber, Airbnb, Shopify, and Booking.com. Since MySQL is open source, it includes numerous features developed in close cooperation with users over more than 25 years. So, it's very likely that your favorite application or programming language is supported by MySQL Database.

Databases are the essential data repository for all software applications. For example, whenever someone conducts a web search, logs in to an account, or completes a transaction, a database system is storing the information so it can be accessed in the future.
A relational database stores data in separate tables rather than putting all the data in one big storeroom. The database structure is organized into physical files optimized for speed. The logical data model, with objects such as data tables, views, rows, and columns, offers a flexible programming environment. You set up rules governing the relationships between different data fields, such as one to one, one to many, unique, required, or optional, and "pointers" between different tables. The database enforces these rules so that with a well-designed database your application never sees data that's inconsistent, duplicated, orphaned, out of date, or missing.
The "SQL" part of "MySQL" stands for "Structured Query Language." SQL is the most common standardized language used to access databases. Depending on your programming environment, you might enter SQL directly (for example, to generate reports), embed SQL statements into code written in another language, or use a language-specific API that hides the SQL syntax.
Open source means it's possible for anyone to use and modify the software. Anybody can download MySQL software from the internet and use it without paying for it. You can also change its source code to suit your needs. MySQL software uses the GNU General Public License (GPL) to

define what you may and may not do with the software in different situations.
If you feel uncomfortable with the GNU GPL or need to embed MySQL code into a commercial application, you can buy a commercially licensed version from Oracle. See the MySQL Licensing Information section for more information.
MySQL is fast, reliable, scalable, and easy to use. It was originally developed to handle large databases quickly and has been used in highly demanding production environments for many years.

Although MySQL is under constant development, it offers a rich and useful set of functions. MySQL's connectivity, speed, and security make it highly suited for accessing databases on the internet.

MySQL's key benefits include

Ease of use: Developers can install MySQL in minutes, and the database is easy to manage.
Reliability: MySQL is one of the most mature and widely used databases. It has been tested in a wide variety of scenarios for more than 25 years, including by many of the world's largest companies. Organizations depend on MySQL to run business-critical applications because of its reliability.
Scalability: MySQL scales to meet the demands of the most accessed applications. MySQL's native replication architecture enables organizations such as Facebook to scale applications to support billions of users.
Performance: MySQL Heatwave is faster and less expensive than other database services, as demonstrated by multiple standard industry benchmarks, including TPC-H, TPC-DS, and CH-benchmark.
High availability: MySQL delivers a complete set of native, fully integrated replication technologies for high availability and disaster recovery. For business-critical applications, and to meet service-level agreement commitments, customers can achieve

- Recovery point objective = 0 (zero data loss)

- Recovery time objective = seconds (automatic failover)

Security: Data security entails protection and compliance with industry and government regulations, including the European Union General Data Protection Regulation, the Payment Card Industry Data Security Standard, the Health Insurance Portability and Accountability Act, and the Defense Information Systems Agency's Security Technical Implementation Guides. MySQL Enterprise Edition provides advanced security features, including authentication/authorization, transparent data encryption, auditing, data masking, and a database firewall.
Flexibility: The MySQL Document Store gives users maximum flexibility in developing traditional SQL and NoSQL schema-free database applications. Developers can mix and match relational data and JSON documents in the same database and application.

MySQL is extremely popular for

Ecommerce: Many of the world’s largest ecommerce applications (for example, Shopify, Uber, and Booking.com) run their transactional systems on MySQL. It’s a popular choice for managing user profiles, credentials, user content, financial data including payments, and fraud detection.
Social platforms: Facebook, Twitter, and LinkedIn are among the world’s largest social networks that rely on MySQL.
Content management: Unlike single-purpose document databases, MySQL enables both SQL and NoSQL with a single database. The MySQL Document Store enables CRUD operations and the power of SQL to query data from JSON documents for reporting and analytics.
SaaS and ISVs: More than 2,000 ISVs, OEMs, and VARs, including Ericsson, F5, and IBM, rely on MySQL as their embedded database to make their applications, hardware, and appliances more competitive, bring them to market faster, and lower their cost of goods sold. MySQL is also the database behind popular SaaS applications, including Zendesk and HubSpot.
Other popular applications using MySQL include ones for online gaming, digital marketing, retail point-of-sale systems, and Internet of Things monitoring systems.

On-premises applications with MySQL Enterprise Edition: MySQL Enterprise Edition includes the most comprehensive set of advanced features, management tools, and technical support to achieve the highest levels of MySQL scalability, security, reliability, and uptime. It reduces the risk, cost, and complexity in developing, deploying, and managing business-critical MySQL applications. It provides security features, including MySQL Enterprise Backup, Monitor, Firewall, Audit, Transparent Data Encryption, and Authentication, to help customers protect data and achieve regulatory and industry compliance.

MySQL™

NEED FOR PROJECT?

-Written By Shubham Kumar

Gym management software is a type of software that provides fitness businesses the functionality needed to manage all aspects of their business and efficiently operate their studio. Gym management software can also be referred to as club management software, fitness software, or gym scheduling software.

Regardless of the nomenclature, these platforms all share similar feature sets and are used for the same purposes. Gym management software helps fitness owners and operators manage their class and trainer scheduling, keep track of their members, communicate with clients, and process payments.

While the software category includes the word “gym”, gym or club management softwares are used by many types of business. Health clubs, boutique fitness studios (including specific disciplines like spin, barre, Pilates, yoga and HIIT), health spas, CrossFit gyms, martial arts studios, fitness centers, personal trainers, and sports performance businesses all fall into the same category when it comes to software needs.

From gym owners and operators to the trainers and front-desk staff, all employees of a fitness business interact with the software. However, depending on their role, how they utilize the software will be different.

Trainers and instructors will use the software to manage their availability, check their schedules, and view member information. Front-desk staff (which consists of trainers at many studios, especially boutique fitness) will use the software to enroll clients in memberships, schedule classes for members, and process payments. Owners and operators will utilize the software to view reports, create new classes and programs, manage marketing communications, and run instructor payroll.

However, staff members are not the only users of club management software. The clients of the business also do. Gym management softwares have a client facing portal, where members can log in, enroll in memberships, schedule new classes, and transact with the studio.

This means that when choosing a software, fitness studios must understand that their decision is not just about their internal operations, but also their members. Using a gym software that provides a poor customer experience could make that member look elsewhere.

So, we've covered what fitness software is and the functions it serves… but why do clubs and studios need an all-in-one management system?

It helps optimize studio operations

Simply put, club management software helps a studio operate more efficiently. While spreadsheets work when starting the business, eventually the administrative workload becomes too much. Gym management software helps automate routine tasks like data entry, member check-ins, schedule management, and membership renewals. By streamlining the day-to-day operations at the studio, staff now have more time, focus, and energy to devote to other areas of the business.

It helps increase member satisfaction

One of the most important benefits of a fitness software is that they can help increase member retention. Assuming clients enjoy the workouts and the sense of community at your studio, if the software is user friendly when they're scheduling from a mobile app or on their computer, then they'll continue to come back.

Where software can hurt a business is when their members have difficulty using it on their own. If it isn't intuitive to purchase and schedule, or there is too much technology friction, then they'll be forced to call in and have a staff member deal with it. Eventually, they'll just look to take their business somewhere else.

However, there is more to it than just the client experience. Many gyms management softwares have electronic waiver capabilities, which allow new members to skip the hassle at the front desk and enjoy the studio's services when they walk in. In addition, the ability to increase member engagement with marketing workflows is important. Studio owners can set up marketing automations based on various data points like last visit, birthdays, and number of classes attended, which makes members feel valued and appreciated.

It helps make better business decisions

A third reason why club management software is important is they help owners and operators make quicker and more informed decisions. With real-time data and insightful analytics, studios can gain more insights on things like membership performance over time, track retail sales, view outstanding balances, and visualize business KPIs over time.

Since the payments platform is integrated with the business software, each transaction adds more data points and gives a clearer view of the overall state of the business. It also eliminates the need to export data from various sources to put in a spreadsheet and run calculations.

While there are a number of features to consider when looking at a club management software, I'm going to touch a few of the more important functions that your solution should include. Most software providers possess these features in one way or another.

Scheduling, booking, and registration

One of the main functions of a gym management software is the ability to manage scheduling and registration for your programs. Whether your main revenue streams include ongoing classes, private sessions, six-week programs or one day events, each program you offer has different needs when it comes to scheduling, and it's important to understand whether the software can handle those use cases.

For example, if your studio offers personal training, then you'll need the ability to create availability for your trainers that clients can block out. You'll also need to consider HOW clients are able to book those sessions. Many softwares let you set up various credit packages that can be associated with specific open-booking events for clients to purchase and schedule.

When it comes to pre-scheduled programs (e.g., 4-week, Monday/Wednesday class), this would be geared more towards registration, since clients aren't scheduling time slots but signing up for the entire program. If this is more of what your gym offers, then I would recommend looking into how payments can be configured. Upper Hand's fitness software provides the flexibility to create any type of program with a variety of payment options, including one-time payments for the entire program, single-session purchases (including proration), and even payment plans for higher-dollar programs.

Lastly, you'll want to understand how you're able to manage scheduling on the backend. There will be instances where you'll have to refund credits or remove someone from a class, and some softwares may have limitations on whether you can credit those sessions or refund those payments. The key is having important attendance and credit records. You'll also want to understand how staff members are assigned to those sessions and what limitations there are when having to update scheduling on the fly.

Related: Why the In-Studio Experience Begins Online

Membership management

Since auto-renewal memberships is one of the main revenue streams of fitness clubs, gyms, and studios, understanding the functionality of memberships is crucial.

Your membership management software should automatically charge a member's card on file on their renewal date, whether that is monthly, quarterly, or annually.

Customizing the benefits of your memberships is another big aspect to consider. What are you offering to entice your clients to enroll? Upper Hand has the most flexible membership creation process, allowing studios to combine any number of member discounts, retail perks, automate class credits, and member-only programs.

Additional membership features and workflows to consider include one-time join fees, commitment lengths, and what the process is for canceling or suspending memberships entails.

Staff management

To keep operations running efficiently, you'll need the ability to manage your staff. The software should provide a staff portal that they can use to manage their availability, check their schedules, and manage client information.

When it comes to payroll, your software should be able to manage employee timesheets so that calculating payroll is an automated process and not a headache.

Client retention tools

Attracting new clients is only half of the battle; retaining those clients is equally important. Your club management software should give you tools to automate emails, segment your clients into contact groups, and provide the ability to run successful member promotions.

In addition, fitness software gives you the ability to track important client details on their profiles so you have easy access to critical information.

Integrated payments

Many fitness softwares differ when it comes to payments. Some include a payment gateway and merchant account, while others make you connect a third-party payment system. Either way, you should have the ability to accept payments how you want and where you want.

If you have a front desk, then consider which point-of-sale hardwares the software integrates with. You'll need the ability to perform refunds, offer online payment options for clients, manage recurring billing and one-time transactions, and reconcile financial transactions with detailed financial reports.

UP Payments is Upper Hand's in-house payment processor that gives businesses all of the flexibility they need to efficiently and accurately manage their finances.

Retail management

Another feature of gym softwares is retail and inventory management. The software should allow you to sell your branded swag online and in-studio and provide up-to-date inventory metrics. Upper Hand gives studios the ability to set low-quantity alerts to help automate reorders, as well as barcode scanners and SKU printers for easy inventory checks.

Member portal

Lastly, club management software gives your members a way to engage with your business. Whether it’s on their computer or via a mobile app, clients have access to browse your programs, manage scheduling and payment details, and enroll in new programs. Upper Hand also provides clients the ability to manage other members of their family, meaning no duplicate contacts and a more friendly user-experience.

MY SQL CONNECTOR MODULE

-Written By Ankit Kumar

MYSQL CONNECTOR MODULE

MySQL is a Relational Database Management System (RDBMS) whereas the structured Query Language (SQL) is the language used for handling the RDBMS using commands i.e., Creating, Inserting, Updating and Deleting the data from the databases. SQL commands are case insensitive i.e., CREATE and create signify the same command.

In this article, we will be discussing the MySQL Connector module of Python, how to install this module and a piece of code on how to connect this with the MySQL database. For any application, it is very important to store the database on a server for easy data access.

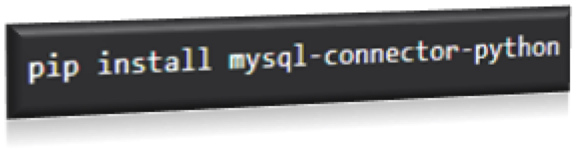
MySQL Connector/Python enables Python programs to access MySQL databases, using an API that is compliant with the Python Database API Specification v2.0 (PEP 249). It is written in pure Python and does not have any dependencies except for the Python Standard Library.

REQUIREMENTS

-Written By Dhruv Verma

MYSQL CONNECTOR MODULE

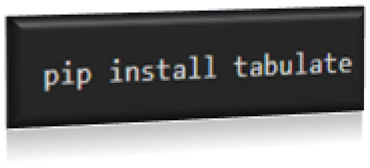
This module does not come built-in with Python. To install it type the below command in the terminal.



```
pip install mysql-connector-python
```

TABULATE

Tabulate is a module that allows you to display table data beautifully. It is not part of standard Python library, so tabulate needs to be installed:



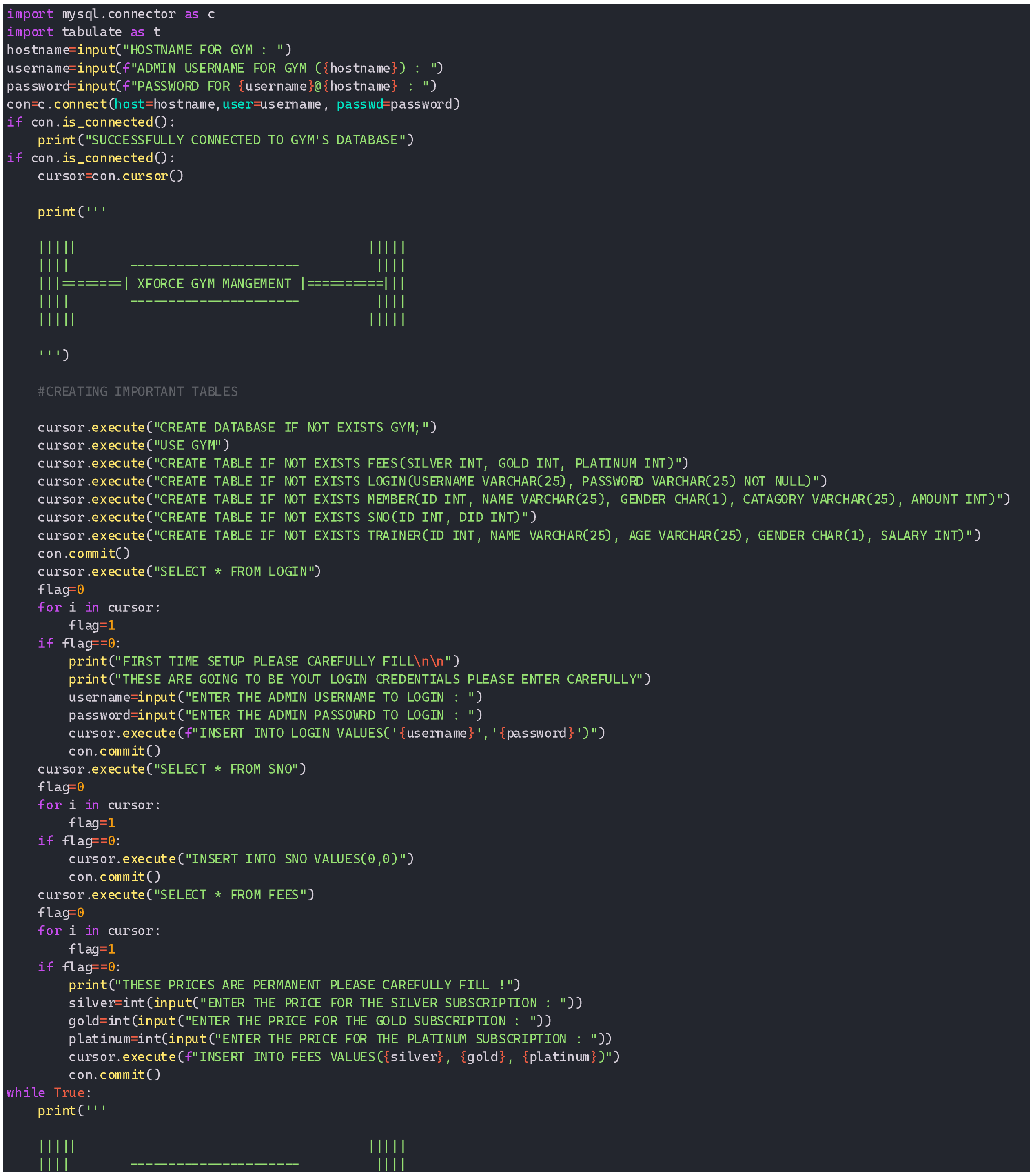
```
pip install tabulate
```

Module supports such tabular data types as:

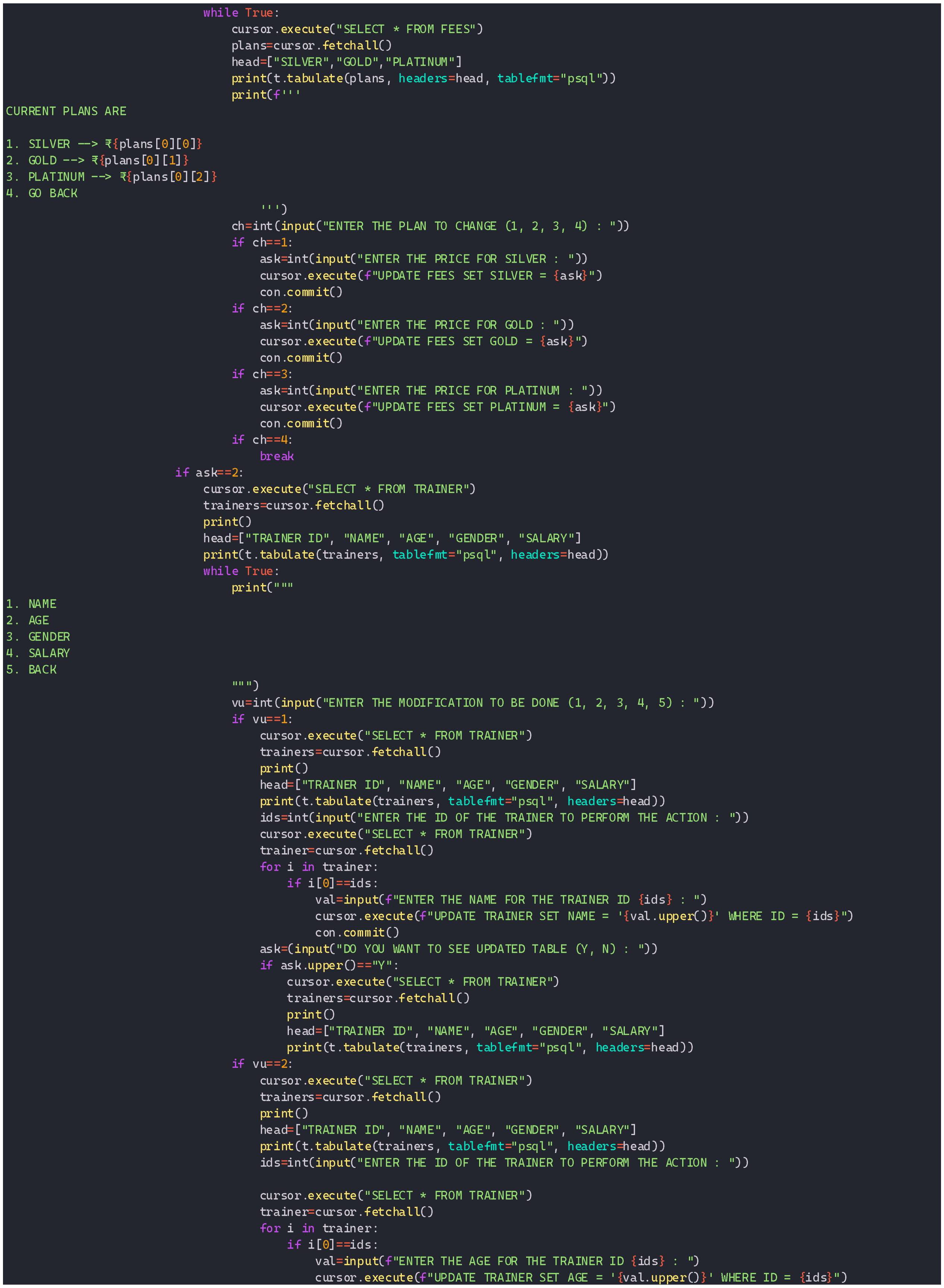
- list of lists (in general case – iterables of iterables)
- dictionary list (or any other iterables object with dictionaries). Keys are used as column names
- dictionary with iterables objects. Keys are used as column names

PROGRAM

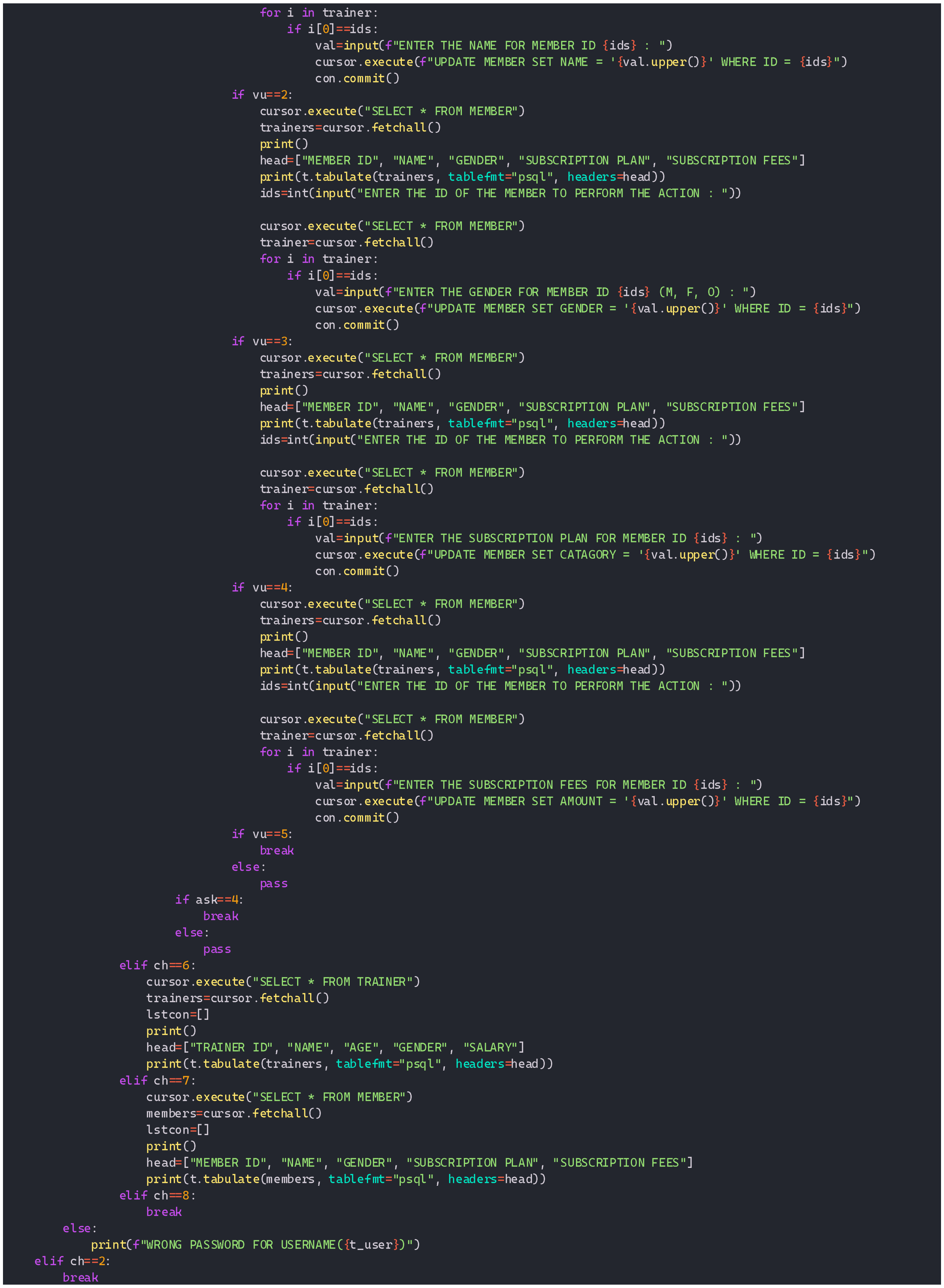
-Code Written by Adarsh



```python
import mysql.connector as c
import tabulate as t
hostname=input("HOSTNAME FOR GYM : ")
username=input(f"ADMIN USERNAME FOR GYM ({hostname}) : ")
password=input(f"PASSWORD FOR {username}@{hostname} : ")
con=c.connect(host=hostname,user=username, passwd=password)
if con.is_connected():
    print("SUCCESSFULLY CONNECTED TO GYM'S DATABASE")
if con.is_connected():
    cursor=con.cursor()

    print('''

    |||||                                      |||||
    ||||        ----------------------          ||||
    |||========| XFORCE GYM MANGEMENT |=========|||
    ||||        ----------------------          ||||
    |||||                                      |||||

    ''')

    #CREATING IMPORTANT TABLES

    cursor.execute("CREATE DATABASE IF NOT EXISTS GYM;")
    cursor.execute("USE GYM")
    cursor.execute("CREATE TABLE IF NOT EXISTS FEES(SILVER INT, GOLD INT, PLATINUM INT)")
    cursor.execute("CREATE TABLE IF NOT EXISTS LOGIN(USERNAME VARCHAR(25), PASSWORD VARCHAR(25) NOT NULL)")
    cursor.execute("CREATE TABLE IF NOT EXISTS MEMBER(ID INT, NAME VARCHAR(25), GENDER CHAR(1), CATAGORY VARCHAR(25), AMOUNT INT)")
    cursor.execute("CREATE TABLE IF NOT EXISTS SNO(ID INT, DID INT)")
    cursor.execute("CREATE TABLE IF NOT EXISTS TRAINER(ID INT, NAME VARCHAR(25), AGE VARCHAR(25), GENDER CHAR(1), SALARY INT)")
    con.commit()
    cursor.execute("SELECT * FROM LOGIN")
    flag=0
    for i in cursor:
        flag=1
    if flag==0:
        print("FIRST TIME SETUP PLEASE CAREFULLY FILL\n\n")
        print("THESE ARE GOING TO BE YOUT LOGIN CREDENTIALS PLEASE ENTER CAREFULLY")
        username=input("ENTER THE ADMIN USERNAME TO LOGIN : ")
        password=input("ENTER THE ADMIN PASSOWRD TO LOGIN : ")
        cursor.execute(f"INSERT INTO LOGIN VALUES('{username}','{password}')")
        con.commit()
    cursor.execute("SELECT * FROM SNO")
    flag=0
    for i in cursor:
        flag=1
    if flag==0:
        cursor.execute("INSERT INTO SNO VALUES(0,0)")
        con.commit()
    cursor.execute("SELECT * FROM FEES")
    flag=0
    for i in cursor:
        flag=1
    if flag==0:
        print("THESE PRICES ARE PERMANENT PLEASE CAREFULLY FILL !")
        silver=int(input("ENTER THE PRICE FOR THE SILVER SUBSCRIPTION : "))
        gold=int(input("ENTER THE PRICE FOR THE GOLD SUBSCRIPTION : "))
        platinum=int(input("ENTER THE PRICE FOR THE PLATINUM SUBSCRIPTION : "))
        cursor.execute(f"INSERT INTO FEES VALUES({silver}, {gold}, {platinum})")
        con.commit()
while True:
    print('''

    |||||                                      |||||
    ||||        ----------------------          ||||
    |||=======|  XFORCE ADMIN LOGIN  |==========|||
    ||||        ---------------------          ||||
    |||||                                     |||||

    ''')
    print("""

    1. LOGIN
    2. EXIT

    """)
    ch=int(input("ENTER THE ACTION TO PERFORM (1, 2) : "))
    if ch==1:
        cursor.execute("SELECT * FROM LOGIN")
        for i in cursor:
            t_user, t_pass=i
        password=input(f"ENTER THE ADMIN PASSWORD FOR ADMIN USERNAME ({t_user}) : ")
        if password==t_pass:
            while True :
                print('''
        |||||                                     |||||
        ||||        ---------------------          ||||
        |||=======|   XFORCE GYM HOME    |==========|||
        ||||        ---------------------          ||||
        |||||                                     |||||

        ''')
                print("""

                1. ADD TRAINER
                2. ADD MEMBER
                3. REMOVE TRAINER
                4. REMOVE MEMBER
                5. MODIFY
                6. SHOW TRAINERS
                7. SHOW MEMBERS
                8. GO BACK

                """)
                ch=int(input("ENTER THE ACTION TO PERFORM (1, 2, 3, 4, 5, 6, 7, 8) : "))
                if ch==1:
                    trainer_name=input("NAME OF THE TRAINER : ")
                    trainer_age=input(f"ENTER YOUR AGE OF {trainer_name.upper()} : ")
                    trainer_gender=input(f"ENTER THE GENDER OF {trainer_name.upper()} (M , F, O) : ")
                    trainer_salary=input(f"ENTER THE SALARY FOR {trainer_name.upper()} : ")
                    cursor.execute("SELECT * FROM SNO")
                    for i in cursor:
                        t_id,t_did=i
                    t_id+=1
                    cursor.execute(f"INSERT INTO TRAINER VALUES({t_id},'{trainer_name.upper()}', '{trainer_age.upper()}', 
'{trainer_gender.upper()}', {trainer_salary})")
                    cursor.execute(f"UPDATE SNO SET ID={t_id}")
                    con.commit()
                    print(f"""

                    TRAINER ADDED WITH FOLLOWING DETAILS :
                    NAME : {trainer_name}
                    AGE : {trainer_age}
                    GENDER : {trainer_gender}
                    SALARY : ₹{trainer_salary}

                    UNIQUE ID FOR {trainer_name} : {t_id}

                    """)
                elif ch==2:
                    member_name=input("ENTER THE NAME OF THE MEMBER : ")
                    member_gender=input(f"ENTER THE GENDER OF {member_name.upper()}(M, F, O) : ")
                    member_catagory=""
                    cursor.execute("SELECT * FROM FEES")
                    show=cursor.fetchall()
                    print(show)

                    print(f"""

                    1. SILVER --> ₹{show[0][0]}
                    2. GOLD --> ₹{show[0][1]}
                    3. PLATINUM --> ₹{show[0][2]}
                    """)
                    ch=int(input("ENTER THE CHOICE : "))
                    if ch==1:
                        member_catagory="SILVER"
                    if ch==2:
                        member_catagory="GOLD"
                    if ch==3:
                        member_catagory="PLATINUM"
                    cursor.execute(f"SELECT {member_catagory} FROM FEES")
                    member_amount=cursor.fetchall()
                    print(member_amount)
                    member_amount=member_amount[0][0]
                    cursor.execute("SELECT * FROM SNO")
                    for i in cursor:
                        t_id,t_did=i
                    t_did+=1
                    cursor.execute(f"UPDATE SNO SET DID={t_did}")
                    cursor.execute(f"INSERT INTO MEMBER VALUES({t_did}, '{member_name.upper()}', '{member_gender.upper()}',
'{member_catagory}' , {member_amount})")
                    con.commit()
                    print(f"""

                    MEMBER ADDED WITH FOLLOWING DETAILS :
                    NAME : {member_name.upper()}
                    GENDER : {member_gender.upper()}
                    SUBSCRIPTIOM : {member_catagory}
                    AMOUNT : ₹{member_amount}

                    UNIQUE ID FOR {member_name.upper()} : {t_did}

                    # """)
                elif ch==3:
                    ch=input("DO YOU WANT TO SHOW THE TRAINERS (Y, N) : ")
                    if ch.upper()=="Y":
                        cursor.execute("SELECT * FROM TRAINER")
                        trainers=cursor.fetchall()
                        lstcon=[]
                        print()
                        head=["TRAINER ID", "NAME", "AGE", "GENDER", "SALARY"]
                        print(t.tabulate(trainers, tablefmt="psql", headers=head))
                    print()
                    ch=int(input("ENTER THE ID OF TRAINER TO REMOVE : "))
                    cursor.execute(f"DELETE FROM TRAINER WHERE ID={ch}")
                    print(f"TRAINER WITH TRAINER ID : {ch} REMOVED")
                    con.commit()
                elif ch==4:
                    ch=input("DO YOU WANT TO SHOW THE TRAINERS (Y, N) : ")
                    if ch.upper()=="Y":
                        cursor.execute("SELECT * FROM MEMBER")
                        members=cursor.fetchall()
                        lstcon=[]
                        print()
                        head=["MEMBER ID", "NAME", "GENDER", "SUBSCRIPTION PLAN", "SUBSCRIPTION FEES"]
                        print(t.tabulate(members, tablefmt="psql", headers=head))
                    ch=int(input("ENTER THE ID OF MEMBER TO REMOVE : "))
                    cursor.execute(f"DELETE FROM MEMBER WHERE ID={ch}")
                    print(f"MEMBER WITH MEMBER ID : {ch} REMOVED")
                    con.commit()
                elif ch==5:
                    while True:
                        print('''
        |||||                                    |||||
        ||||        ---------------------         ||||
        |||=======| XFORCE MODIFICTAIONS |=========|||
        ||||        ---------------------         ||||
        |||||                                    |||||

        ''')
                        print("""

                    1. PLANS
                    2. TRAINER
                    3. MEMBER
                    4. GO BACK

                    """)
                        ask=int(input("ENTER THE ACTION TO PERFORM (1, 2, 3, 4): "))
                        if ask==1:
```



```
                            while True:
                                cursor.execute("SELECT * FROM FEES")
                                plans=cursor.fetchall()
                                head=["SILVER","GOLD","PLATINUM"]
                                print(t.tabulate(plans, headers=head, tablefmt="psql"))
                                print(f'''
CURRENT PLANS ARE

1. SILVER --> ₹{plans[0][0]}
2. GOLD --> ₹{plans[0][1]}
3. PLATINUM --> ₹{plans[0][2]}
4. GO BACK
                                    ''')
                                ch=int(input("ENTER THE PLAN TO CHANGE (1, 2, 3, 4) : "))
                                if ch==1:
                                    ask=int(input("ENTER THE PRICE FOR SILVER : "))
                                    cursor.execute(f"UPDATE FEES SET SILVER = {ask}")
                                    con.commit()
                                if ch==2:
                                    ask=int(input("ENTER THE PRICE FOR GOLD : "))
                                    cursor.execute(f"UPDATE FEES SET GOLD = {ask}")
                                    con.commit()
                                if ch==3:
                                    ask=int(input("ENTER THE PRICE FOR PLATINUM : "))
                                    cursor.execute(f"UPDATE FEES SET PLATINUM = {ask}")
                                    con.commit()
                                if ch==4:
                                    break
                        if ask==2:
                            cursor.execute("SELECT * FROM TRAINER")
                            trainers=cursor.fetchall()
                            print()
                            head=["TRAINER ID", "NAME", "AGE", "GENDER", "SALARY"]
                            print(t.tabulate(trainers, tablefmt="psql", headers=head))
                            while True:
                                print("""
1. NAME
2. AGE
3. GENDER
4. SALARY
5. BACK
                                """)
                                vu=int(input("ENTER THE MODIFICATION TO BE DONE (1, 2, 3, 4, 5) : "))
                                if vu==1:
                                    cursor.execute("SELECT * FROM TRAINER")
                                    trainers=cursor.fetchall()
                                    print()
                                    head=["TRAINER ID", "NAME", "AGE", "GENDER", "SALARY"]
                                    print(t.tabulate(trainers, tablefmt="psql", headers=head))
                                    ids=int(input("ENTER THE ID OF THE TRAINER TO PERFORM THE ACTION : "))
                                    cursor.execute("SELECT * FROM TRAINER")
                                    trainer=cursor.fetchall()
                                    for i in trainer:
                                        if i[0]==ids:
                                            val=input(f"ENTER THE NAME FOR THE TRAINER ID {ids} : ")
                                            cursor.execute(f"UPDATE TRAINER SET NAME = '{val.upper()}' WHERE ID = {ids}")
                                            con.commit()
                                    ask=(input("DO YOU WANT TO SEE UPDATED TABLE (Y, N) : "))
                                    if ask.upper()=="Y":
                                        cursor.execute("SELECT * FROM TRAINER")
                                        trainers=cursor.fetchall()
                                        print()
                                        head=["TRAINER ID", "NAME", "AGE", "GENDER", "SALARY"]
                                        print(t.tabulate(trainers, tablefmt="psql", headers=head))
                                if vu==2:
                                    cursor.execute("SELECT * FROM TRAINER")
                                    trainers=cursor.fetchall()
                                    print()
                                    head=["TRAINER ID", "NAME", "AGE", "GENDER", "SALARY"]
                                    print(t.tabulate(trainers, tablefmt="psql", headers=head))
                                    ids=int(input("ENTER THE ID OF THE TRAINER TO PERFORM THE ACTION : "))

                                    cursor.execute("SELECT * FROM TRAINER")
                                    trainer=cursor.fetchall()
                                    for i in trainer:
                                        if i[0]==ids:
                                            val=input(f"ENTER THE AGE FOR THE TRAINER ID {ids} : ")
                                            cursor.execute(f"UPDATE TRAINER SET AGE = '{val.upper()}' WHERE ID = {ids}")
                                            con.commit()
                                    ask=(input("DO YOU WANT TO SEE UPDATED TABLE (Y, N) : "))
                                    if ask.upper()=="Y":
                                        cursor.execute("SELECT * FROM TRAINER")
                                        trainers=cursor.fetchall()
                                        print()
                                        head=["TRAINER ID", "NAME", "AGE", "GENDER", "SALARY"]
                                        print(t.tabulate(trainers, tablefmt="psql", headers=head))
                                if vu==3:
                                    cursor.execute("SELECT * FROM TRAINER")
                                    trainers=cursor.fetchall()
                                    print()
                                    head=["TRAINER ID", "NAME", "AGE", "GENDER", "SALARY"]
                                    print(t.tabulate(trainers, tablefmt="psql", headers=head))
                                    ids=int(input("ENTER THE ID OF THE TRAINER TO PERFORM THE ACTION : "))

                                    cursor.execute("SELECT * FROM TRAINER")
                                    trainer=cursor.fetchall()
                                    for i in trainer:
                                        if i[0]==ids:
                                            val=input(f"ENTER THE GENDER FOR THE TRAINER ID {ids} : ")
                                            cursor.execute(f"UPDATE TRAINER SET GENDER = '{val.upper()}' WHERE ID = {ids}")
                                            con.commit()

                                    ask=(input("DO YOU WANT TO SEE UPDATED TABLE (Y, N) : "))
                                    if ask.upper()=="Y":
                                        cursor.execute("SELECT * FROM TRAINER")
                                        trainers=cursor.fetchall()
                                        print()
                                        head=["TRAINER ID", "NAME", "AGE", "GENDER", "SALARY"]
                                        print(t.tabulate(trainers, tablefmt="psql", headers=head))
                                if vu==4:
                                    cursor.execute("SELECT * FROM TRAINER")
                                    trainers=cursor.fetchall()
                                    print()
                                    head=["TRAINER ID", "NAME", "AGE", "GENDER", "SALARY"]
                                    print(t.tabulate(trainers, tablefmt="psql", headers=head))
                                    ids=int(input("ENTER THE ID OF THE TRAINER TO PERFORM THE ACTION : "))

                                    cursor.execute("SELECT * FROM TRAINER")
                                    trainer=cursor.fetchall()
                                    for i in trainer:
                                        if i[0]==ids:
                                            val=input(f"ENTER THE SALARY FOR THE TRAINER ID {ids} : ")
                                            cursor.execute(f"UPDATE TRAINER SET SALARY = {val.upper()} WHERE ID = {ids}")
                                            con.commit()

                                    ask=(input("DO YOU WANT TO SEE UPDATED TABLE (Y, N) : "))
                                    if ask.upper()=="Y":
                                        cursor.execute("SELECT * FROM TRAINER")
                                        trainers=cursor.fetchall()
                                        print()
                                        head=["TRAINER ID", "NAME", "AGE", "GENDER", "SALARY"]
                                        print(t.tabulate(trainers, tablefmt="psql", headers=head))
                                if vu==5:
                                    break
                                else:
                                    pass
                        if ask==3:
                            while True:
                                print("""
1. NAME
2. GENDER
3. SUBSCRIPTION PLAN
4. SUBSCRIPTION FEES
5. BACK
                                """)
                                vu=int(input("ENTER THE MODIFICATION TO BE DONE (1, 2, 3, 4, 5) : "))
                                if vu==1:
                                    cursor.execute("SELECT * FROM MEMBER")
                                    trainers=cursor.fetchall()
                                    print()
                                    head=["MEMBER ID", "NAME", "GENDER", "SUBSCRIPTION PLAN", "SUBSCRIPTION FEES"]
                                    print(t.tabulate(trainers, tablefmt="psql", headers=head))
                                    ids=int(input("ENTER THE ID OF THE MEMBER TO PERFORM THE ACTION : "))

                                    cursor.execute("SELECT * FROM MEMBER")
                                    trainer=cursor.fetchall()
```



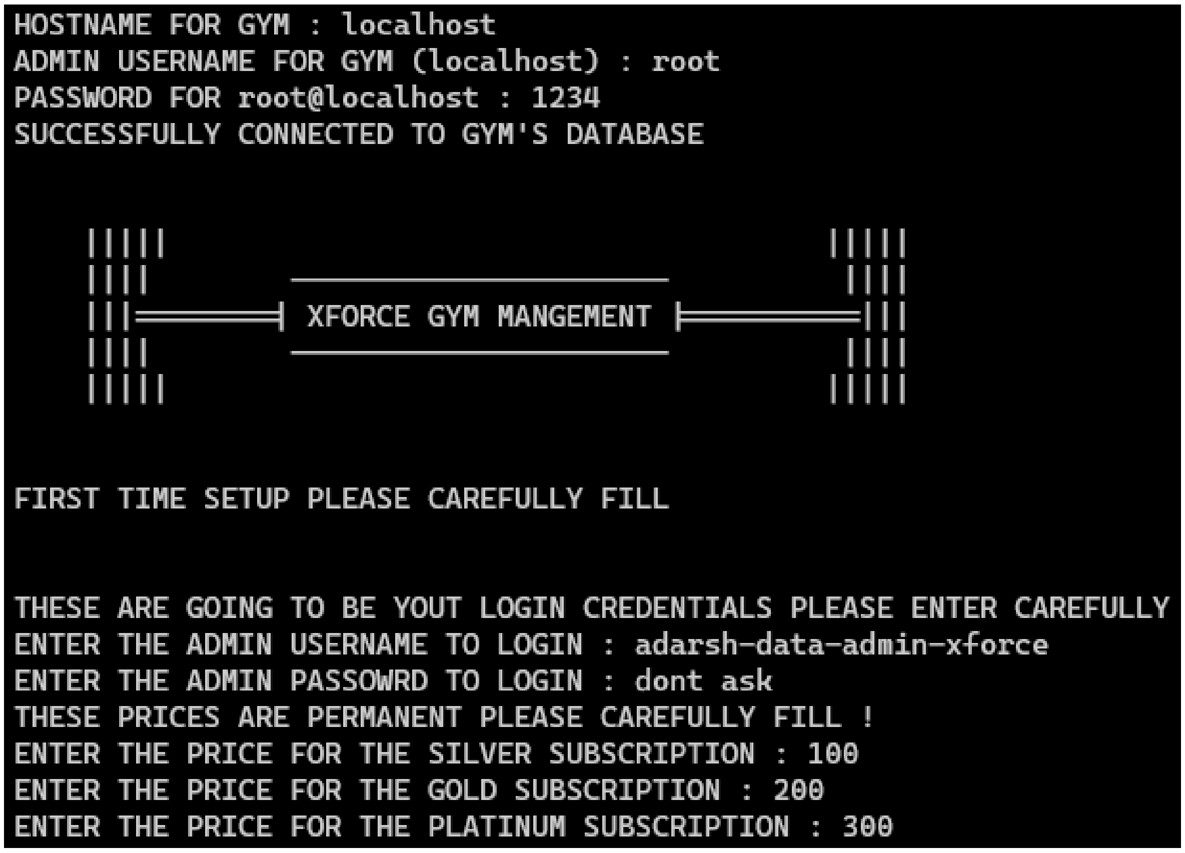
```
                                    for i in trainer:
                                        if i[0]==ids:
                                            val=input(f"ENTER THE NAME FOR MEMBER ID {ids} : ")
                                            cursor.execute(f"UPDATE MEMBER SET NAME = '{val.upper()}' WHERE ID = {ids}")
                                            con.commit()
                                if vu==2:
                                    cursor.execute("SELECT * FROM MEMBER")
                                    trainers=cursor.fetchall()
                                    print()
                                    head=["MEMBER ID", "NAME", "GENDER", "SUBSCRIPTION PLAN", "SUBSCRIPTION FEES"]
                                    print(t.tabulate(trainers, tablefmt="psql", headers=head))
                                    ids=int(input("ENTER THE ID OF THE MEMBER TO PERFORM THE ACTION : "))

                                    cursor.execute("SELECT * FROM MEMBER")
                                    trainer=cursor.fetchall()
                                    for i in trainer:
                                        if i[0]==ids:
                                            val=input(f"ENTER THE GENDER FOR MEMBER ID {ids} (M, F, O) : ")
                                            cursor.execute(f"UPDATE MEMBER SET GENDER = '{val.upper()}' WHERE ID = {ids}")
                                            con.commit()
                                if vu==3:
                                    cursor.execute("SELECT * FROM MEMBER")
                                    trainers=cursor.fetchall()
                                    print()
                                    head=["MEMBER ID", "NAME", "GENDER", "SUBSCRIPTION PLAN", "SUBSCRIPTION FEES"]
                                    print(t.tabulate(trainers, tablefmt="psql", headers=head))
                                    ids=int(input("ENTER THE ID OF THE MEMBER TO PERFORM THE ACTION : "))

                                    cursor.execute("SELECT * FROM MEMBER")
                                    trainer=cursor.fetchall()
                                    for i in trainer:
                                        if i[0]==ids:
                                            val=input(f"ENTER THE SUBSCRIPTION PLAN FOR MEMBER ID {ids} : ")
                                            cursor.execute(f"UPDATE MEMBER SET CATAGORY = '{val.upper()}' WHERE ID = {ids}")
                                            con.commit()
                                if vu==4:
                                    cursor.execute("SELECT * FROM MEMBER")
                                    trainers=cursor.fetchall()
                                    print()
                                    head=["MEMBER ID", "NAME", "GENDER", "SUBSCRIPTION PLAN", "SUBSCRIPTION FEES"]
                                    print(t.tabulate(trainers, tablefmt="psql", headers=head))
                                    ids=int(input("ENTER THE ID OF THE MEMBER TO PERFORM THE ACTION : "))

                                    cursor.execute("SELECT * FROM MEMBER")
                                    trainer=cursor.fetchall()
                                    for i in trainer:
                                        if i[0]==ids:
                                            val=input(f"ENTER THE SUBSCRIPTION FEES FOR MEMBER ID {ids} : ")
                                            cursor.execute(f"UPDATE MEMBER SET AMOUNT = '{val.upper()}' WHERE ID = {ids}")
                                            con.commit()
                                if vu==5:
                                    break
                                else:
                                    pass
                        if ask==4:
                            break
                        else:
                            pass
                elif ch==6:
                    cursor.execute("SELECT * FROM TRAINER")
                    trainers=cursor.fetchall()
                    lstcon=[]
                    print()
                    head=["TRAINER ID", "NAME", "AGE", "GENDER", "SALARY"]
                    print(t.tabulate(trainers, tablefmt="psql", headers=head))
                elif ch==7:
                    cursor.execute("SELECT * FROM MEMBER")
                    members=cursor.fetchall()
                    lstcon=[]
                    print()
                    head=["MEMBER ID", "NAME", "GENDER", "SUBSCRIPTION PLAN", "SUBSCRIPTION FEES"]
                    print(t.tabulate(members, tablefmt="psql", headers=head))
                elif ch==8:
                    break
        else:
            print(f"WRONG PASSWORD FOR USERNAME({t_user})")
    elif ch==2:
        break
```

SCAN QR CODE TO ACCESS THE CODE FILE

SCAN ME

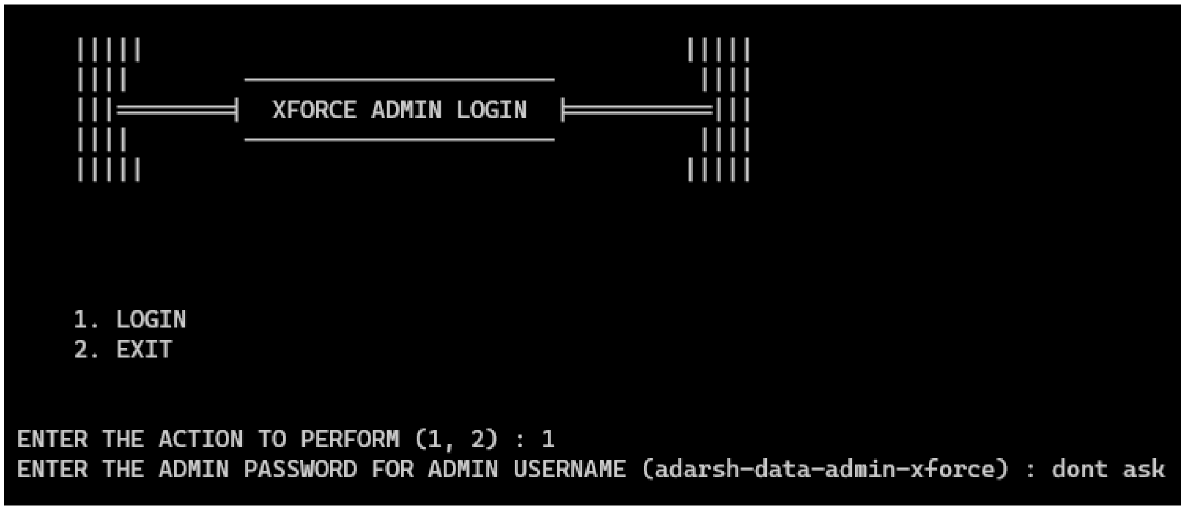
OUTPUT

First Time Setup:



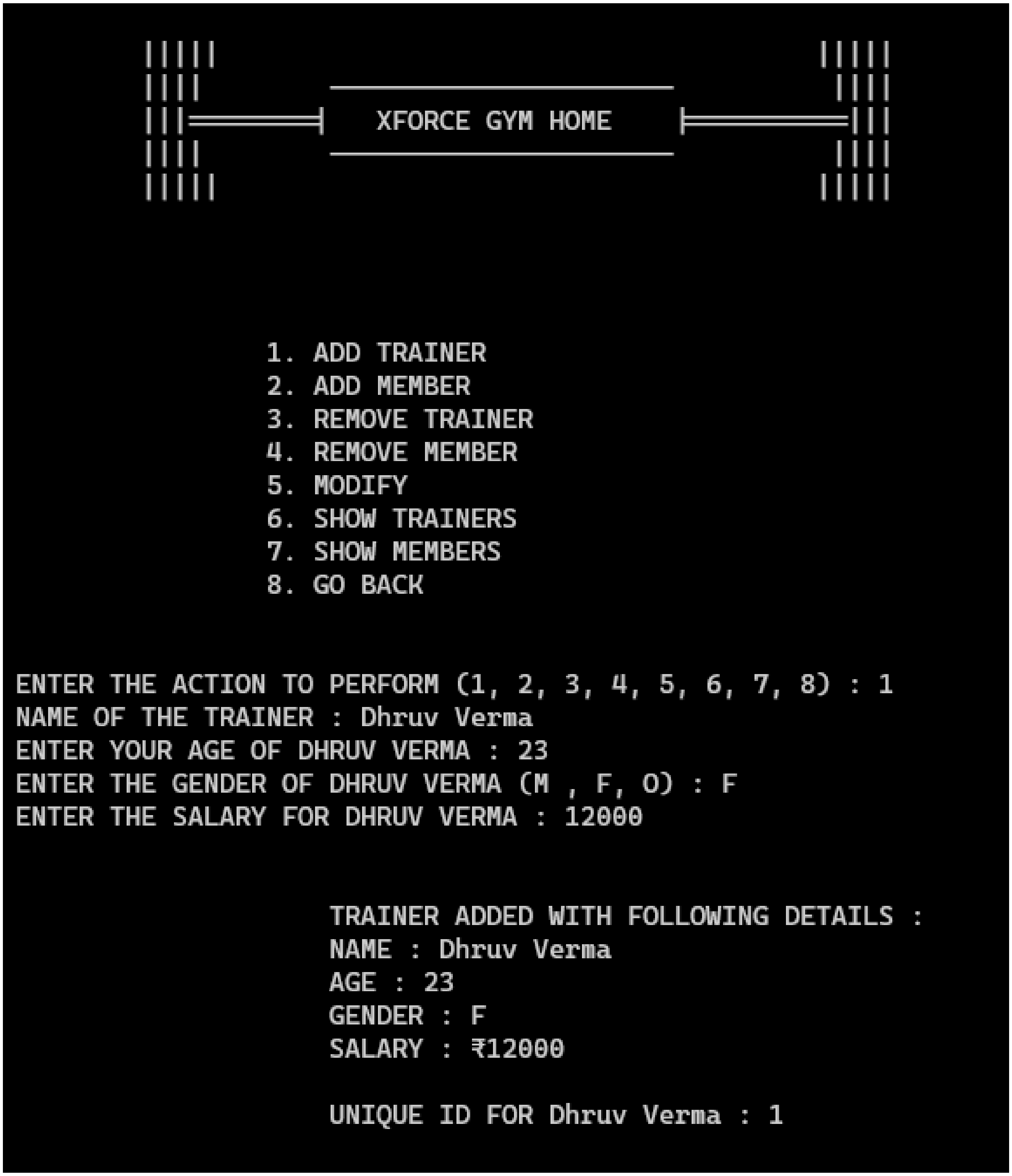
```
HOSTNAME FOR GYM : localhost
ADMIN USERNAME FOR GYM (localhost) : root
PASSWORD FOR root@localhost : 1234
SUCCESSFULLY CONNECTED TO GYM'S DATABASE


    |||||                                   |||||
    ||||       _____________________         ||||
    |||=======| XFORCE GYM MANGEMENT |=======|||
    ||||       ---------------------         ||||
    |||||                                   |||||


FIRST TIME SETUP PLEASE CAREFULLY FILL


THESE ARE GOING TO BE YOUT LOGIN CREDENTIALS PLEASE ENTER CAREFULLY
ENTER THE ADMIN USERNAME TO LOGIN : adarsh-data-admin-xforce
ENTER THE ADMIN PASSOWRD TO LOGIN : dont ask
THESE PRICES ARE PERMANENT PLEASE CAREFULLY FILL !
ENTER THE PRICE FOR THE SILVER SUBSCRIPTION : 100
ENTER THE PRICE FOR THE GOLD SUBSCRIPTION : 200
ENTER THE PRICE FOR THE PLATINUM SUBSCRIPTION : 300
```

Login to the GYM MANAGEMENT SYSTEM:



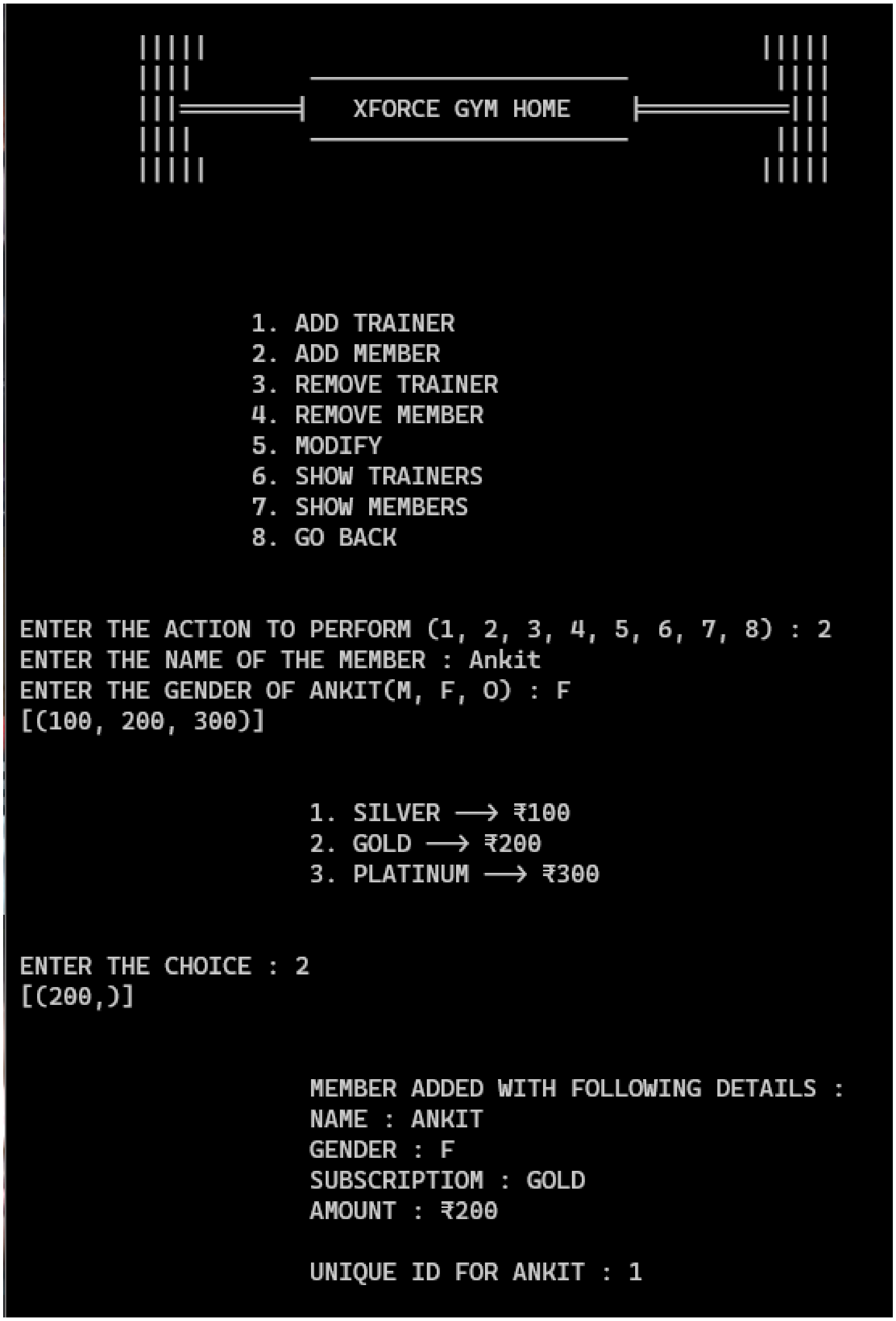
```
    |||||                                   |||||
    ||||       _____________________         ||||
    |||=======|  XFORCE ADMIN LOGIN  |=======|||
    ||||       ---------------------         ||||
    |||||                                   |||||



    1. LOGIN
    2. EXIT


ENTER THE ACTION TO PERFORM (1, 2) : 1
ENTER THE ADMIN PASSWORD FOR ADMIN USERNAME (adarsh-data-admin-xforce) : dont ask
```

Adding Trainer:



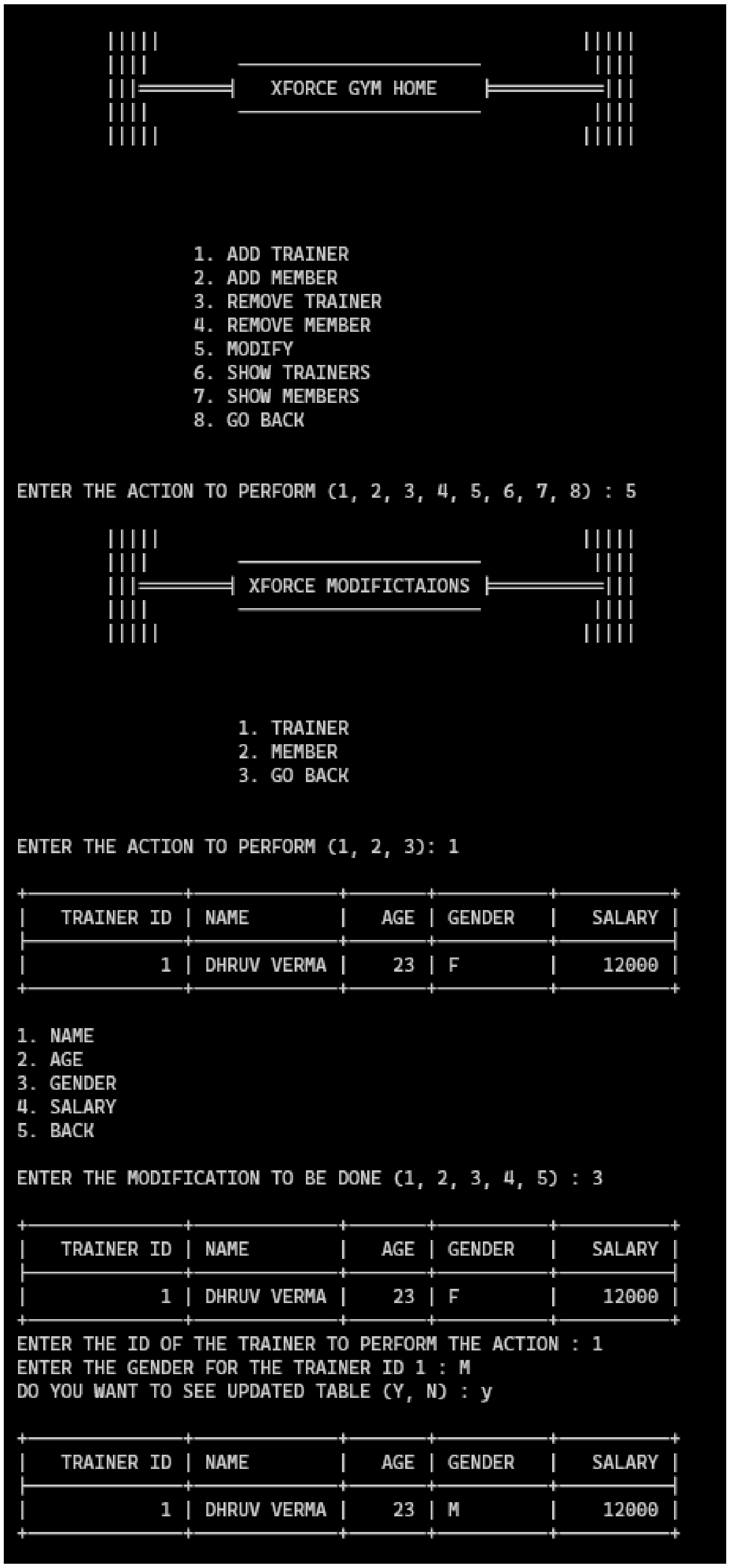
```
        |||||                                             |||||
        ||||        _______________________________        ||||
        |||╞═════════╡    XFORCE GYM HOME    ╞═══════════|||
        ||||        _______________________________        ||||
        |||||                                             |||||




                1. ADD TRAINER
                2. ADD MEMBER
                3. REMOVE TRAINER
                4. REMOVE MEMBER
                5. MODIFY
                6. SHOW TRAINERS
                7. SHOW MEMBERS
                8. GO BACK


ENTER THE ACTION TO PERFORM (1, 2, 3, 4, 5, 6, 7, 8) : 1
NAME OF THE TRAINER : Dhruv Verma
ENTER YOUR AGE OF DHRUV VERMA : 23
ENTER THE GENDER OF DHRUV VERMA (M , F, O) : F
ENTER THE SALARY FOR DHRUV VERMA : 12000


                    TRAINER ADDED WITH FOLLOWING DETAILS :
                    NAME : Dhruv Verma
                    AGE : 23
                    GENDER : F
                    SALARY : ₹12000

                    UNIQUE ID FOR Dhruv Verma : 1
```

Adding Member:



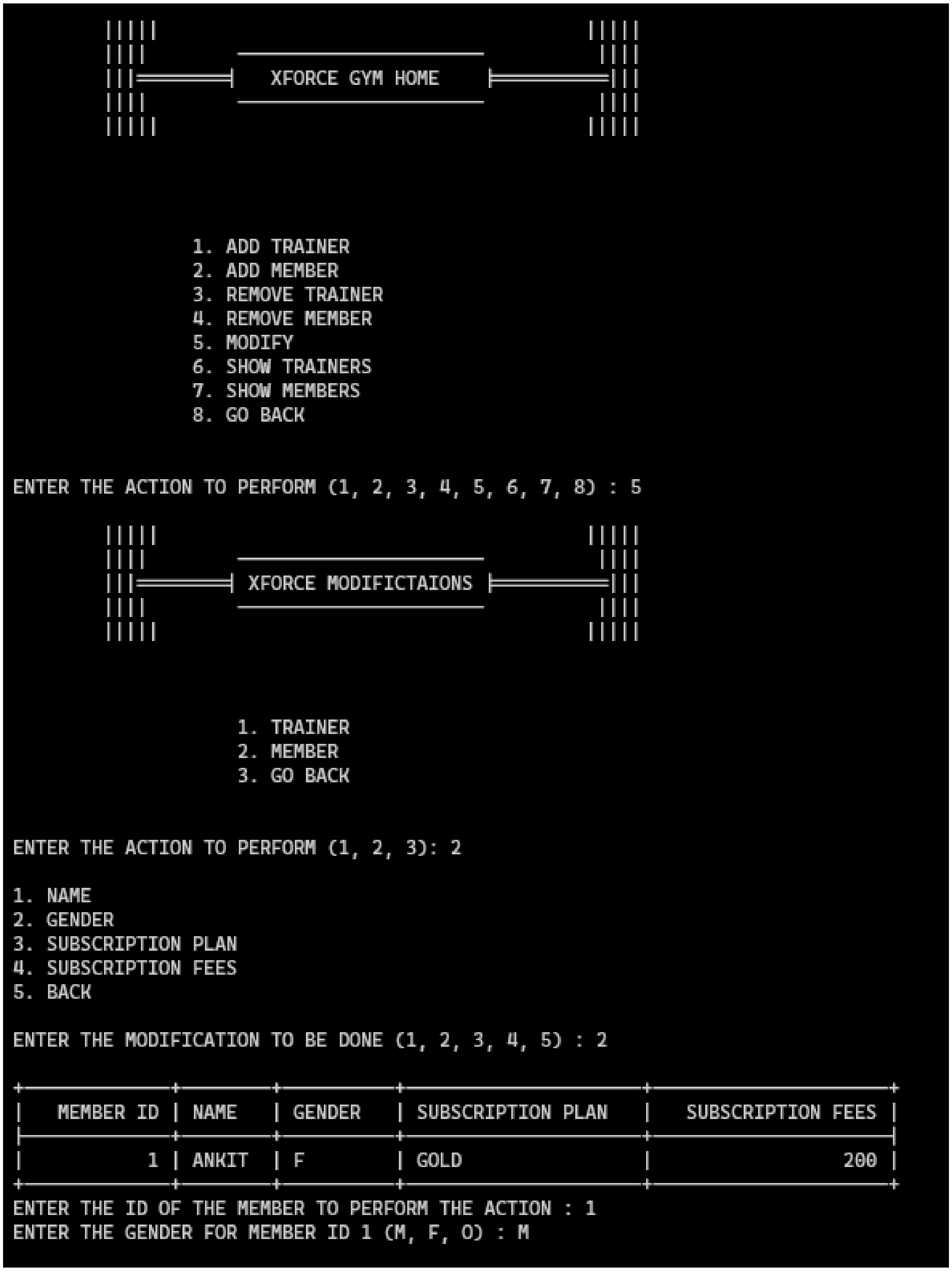
```
        |||||                                          |||||
        ||||       _______________________              ||||
        |||════════╡    XFORCE GYM HOME    ╞══════════|||
        ||||       ‾‾‾‾‾‾‾‾‾‾‾‾‾‾‾‾‾‾‾‾‾‾‾              ||||
        |||||                                          |||||




                1. ADD TRAINER
                2. ADD MEMBER
                3. REMOVE TRAINER
                4. REMOVE MEMBER
                5. MODIFY
                6. SHOW TRAINERS
                7. SHOW MEMBERS
                8. GO BACK


ENTER THE ACTION TO PERFORM (1, 2, 3, 4, 5, 6, 7, 8) : 2
ENTER THE NAME OF THE MEMBER : Ankit
ENTER THE GENDER OF ANKIT(M, F, O) : F
[(100, 200, 300)]


                    1. SILVER ⟶ ₹100
                    2. GOLD ⟶ ₹200
                    3. PLATINUM ⟶ ₹300


ENTER THE CHOICE : 2
[(200,)]


                    MEMBER ADDED WITH FOLLOWING DETAILS :
                    NAME : ANKIT
                    GENDER : F
                    SUBSCRIPTIOM : GOLD
                    AMOUNT : ₹200

                    UNIQUE ID FOR ANKIT : 1
```

Modifying Trainer:



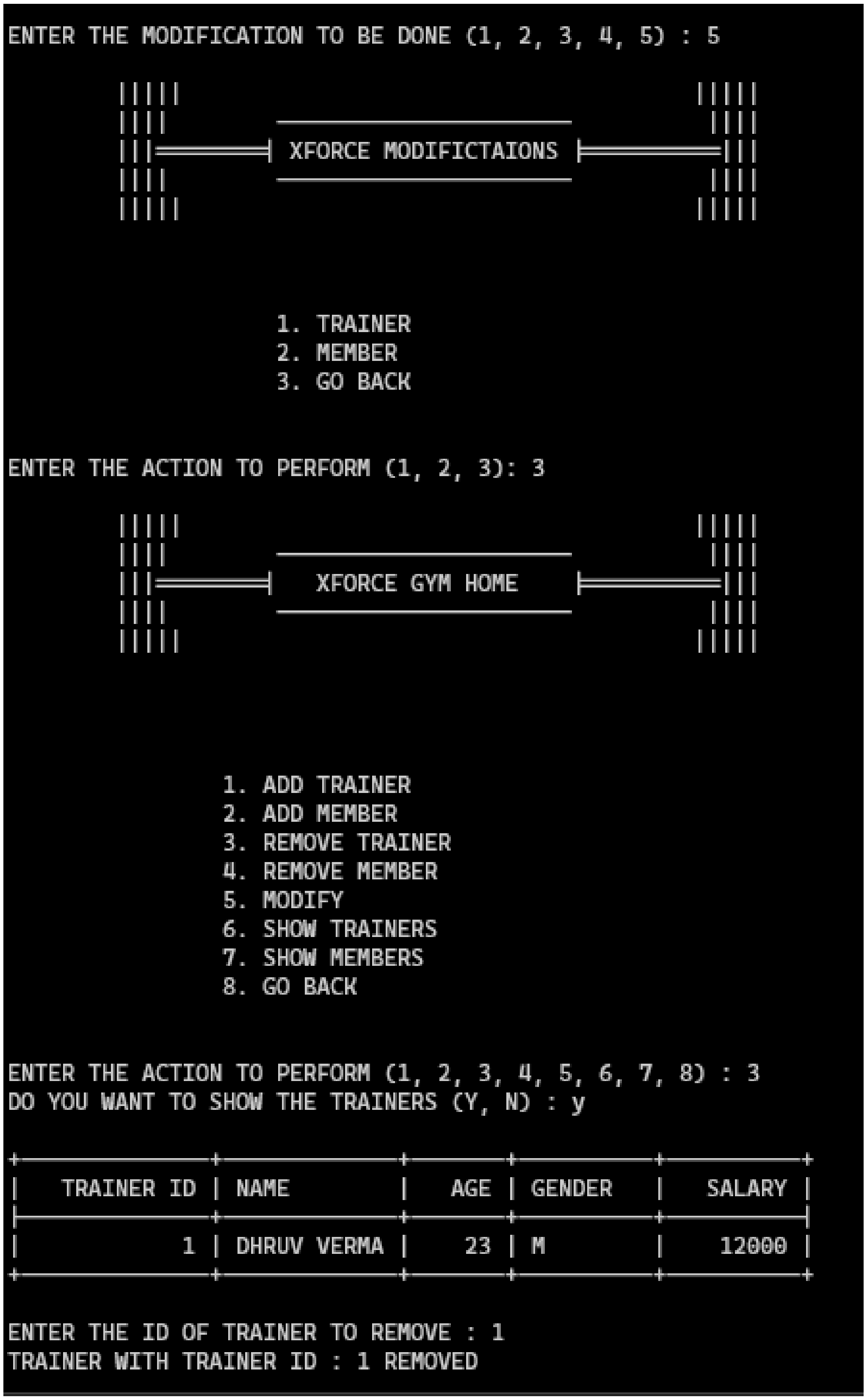
```
        |||||                                       |||||
        ||||      ──────────────────────             ||||
        |||═══════╡  XFORCE GYM HOME    ╞══════════|||
        ||||      ──────────────────────             ||||
        |||||                                       |||||



                1. ADD TRAINER
                2. ADD MEMBER
                3. REMOVE TRAINER
                4. REMOVE MEMBER
                5. MODIFY
                6. SHOW TRAINERS
                7. SHOW MEMBERS
                8. GO BACK


ENTER THE ACTION TO PERFORM (1, 2, 3, 4, 5, 6, 7, 8) : 5
        |||||                                       |||||
        ||||      ──────────────────────             ||||
        |||═══════╡ XFORCE MODIFICTAIONS ╞═════════|||
        ||||      ──────────────────────             ||||
        |||||                                       |||||



                    1. TRAINER
                    2. MEMBER
                    3. GO BACK


ENTER THE ACTION TO PERFORM (1, 2, 3): 1

+──────────────+─────────────+───────+──────────+─────────+
|   TRAINER ID | NAME        |   AGE | GENDER   |  SALARY |
├──────────────+─────────────+───────+──────────+─────────┤
|            1 | DHRUV VERMA |    23 | F        |   12000 |
+──────────────+─────────────+───────+──────────+─────────+

1. NAME
2. AGE
3. GENDER
4. SALARY
5. BACK

ENTER THE MODIFICATION TO BE DONE (1, 2, 3, 4, 5) : 3

+──────────────+─────────────+───────+──────────+─────────+
|   TRAINER ID | NAME        |   AGE | GENDER   |  SALARY |
├──────────────+─────────────+───────+──────────+─────────┤
|            1 | DHRUV VERMA |    23 | F        |   12000 |
+──────────────+─────────────+───────+──────────+─────────+
ENTER THE ID OF THE TRAINER TO PERFORM THE ACTION : 1
ENTER THE GENDER FOR THE TRAINER ID 1 : M
DO YOU WANT TO SEE UPDATED TABLE (Y, N) : y

+──────────────+─────────────+───────+──────────+─────────+
|   TRAINER ID | NAME        |   AGE | GENDER   |  SALARY |
├──────────────+─────────────+───────+──────────+─────────┤
|            1 | DHRUV VERMA |    23 | M        |   12000 |
+──────────────+─────────────+───────+──────────+─────────+
```

Modifying Member:



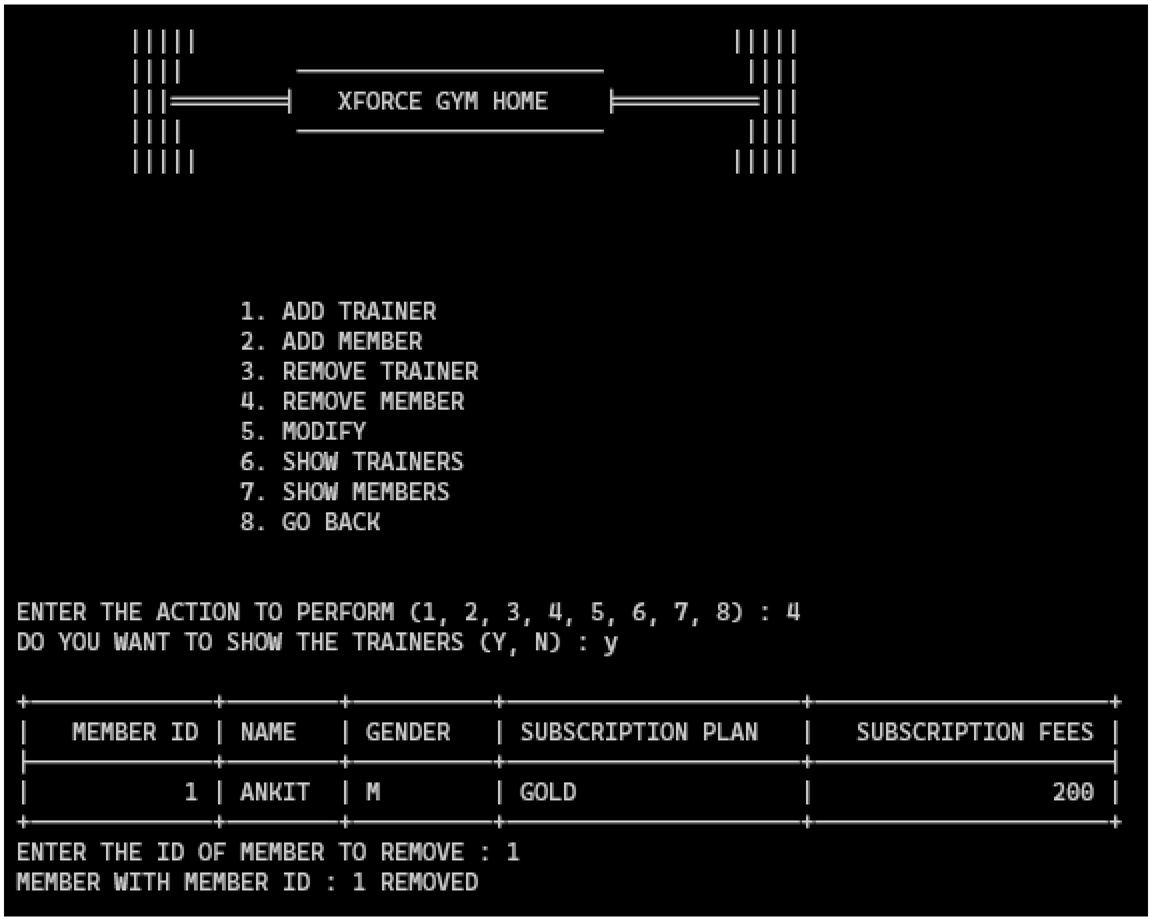
```
        |||||                                           |||||
        ||||      ______________________                 ||||
        |||══════╡      XFORCE GYM HOME      ╞══════════|||
        ||||      ______________________                 ||||
        |||||                                           |||||




                1. ADD TRAINER
                2. ADD MEMBER
                3. REMOVE TRAINER
                4. REMOVE MEMBER
                5. MODIFY
                6. SHOW TRAINERS
                7. SHOW MEMBERS
                8. GO BACK


ENTER THE ACTION TO PERFORM (1, 2, 3, 4, 5, 6, 7, 8) : 5

        |||||                                           |||||
        ||||      ______________________                 ||||
        |||══════╡ XFORCE MODIFICTAIONS ╞══════════|||
        ||||      ______________________                 ||||
        |||||                                           |||||




                    1. TRAINER
                    2. MEMBER
                    3. GO BACK


ENTER THE ACTION TO PERFORM (1, 2, 3): 2

1. NAME
2. GENDER
3. SUBSCRIPTION PLAN
4. SUBSCRIPTION FEES
5. BACK

ENTER THE MODIFICATION TO BE DONE (1, 2, 3, 4, 5) : 2

+-------------+---------+----------+----------------------+---------------------+
|   MEMBER ID | NAME    | GENDER   | SUBSCRIPTION PLAN    |   SUBSCRIPTION FEES |
|-------------+---------+----------+----------------------+---------------------|
|           1 | ANKIT   | F        | GOLD                 |                 200 |
+-------------+---------+----------+----------------------+---------------------+
ENTER THE ID OF THE MEMBER TO PERFORM THE ACTION : 1
ENTER THE GENDER FOR MEMBER ID 1 (M, F, O) : M
```

Remove Trainer:



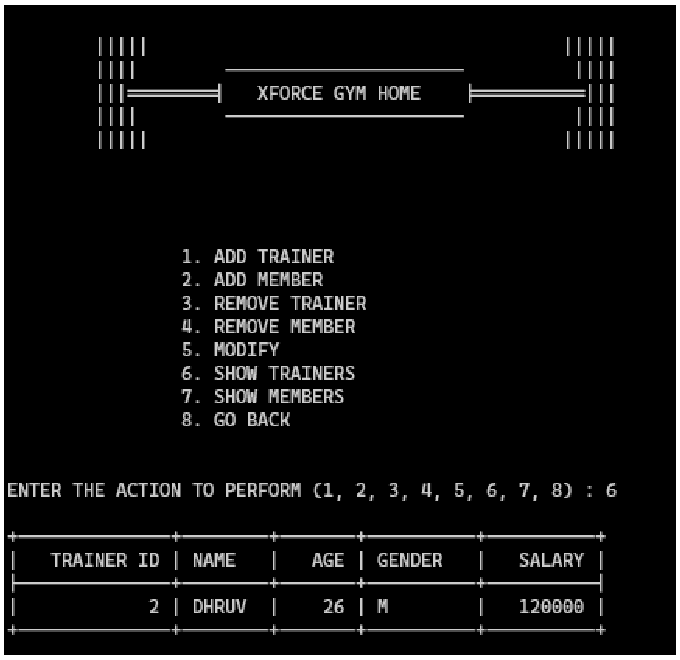
```
ENTER THE MODIFICATION TO BE DONE (1, 2, 3, 4, 5) : 5

        |||||                                     |||||
        ||||       ______________________          ||||
        |||═══════╡ XFORCE MODIFICTAIONS ╞═══════════|||
        ||||       ‾‾‾‾‾‾‾‾‾‾‾‾‾‾‾‾‾‾‾‾‾‾          ||||
        |||||                                     |||||



                    1. TRAINER
                    2. MEMBER
                    3. GO BACK


ENTER THE ACTION TO PERFORM (1, 2, 3): 3

        |||||                                     |||||
        ||||       ______________________          ||||
        |||═══════╡   XFORCE GYM HOME    ╞═══════════|||
        ||||       ‾‾‾‾‾‾‾‾‾‾‾‾‾‾‾‾‾‾‾‾‾‾          ||||
        |||||                                     |||||




                1. ADD TRAINER
                2. ADD MEMBER
                3. REMOVE TRAINER
                4. REMOVE MEMBER
                5. MODIFY
                6. SHOW TRAINERS
                7. SHOW MEMBERS
                8. GO BACK


ENTER THE ACTION TO PERFORM (1, 2, 3, 4, 5, 6, 7, 8) : 3
DO YOU WANT TO SHOW THE TRAINERS (Y, N) : y

+--------------+-------------+-------+------------+----------+
|   TRAINER ID | NAME        |   AGE | GENDER     |   SALARY |
|--------------+-------------+-------+------------+----------|
|            1 | DHRUV VERMA |    23 | M          |    12000 |
+--------------+-------------+-------+------------+----------+

ENTER THE ID OF TRAINER TO REMOVE : 1
TRAINER WITH TRAINER ID : 1 REMOVED
```

Remove Member:



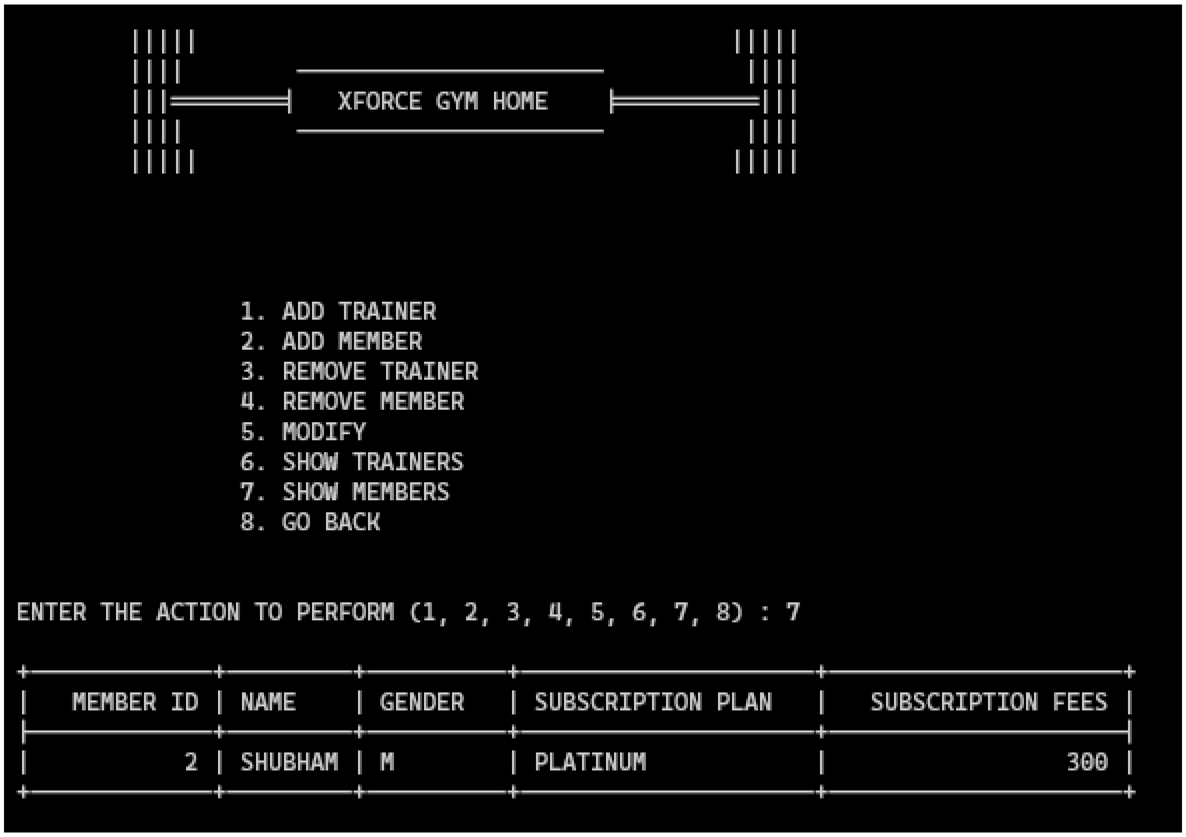
```
        |||||                                      |||||
        ||||      ______________________            ||||
        |||======|   XFORCE GYM HOME    |==========|||
        ||||      ______________________            ||||
        |||||                                      |||||



                1. ADD TRAINER
                2. ADD MEMBER
                3. REMOVE TRAINER
                4. REMOVE MEMBER
                5. MODIFY
                6. SHOW TRAINERS
                7. SHOW MEMBERS
                8. GO BACK


ENTER THE ACTION TO PERFORM (1, 2, 3, 4, 5, 6, 7, 8) : 4
DO YOU WANT TO SHOW THE TRAINERS (Y, N) : y

+-------------+--------+----------+---------------------+---------------------+
|   MEMBER ID | NAME   | GENDER   | SUBSCRIPTION PLAN   |   SUBSCRIPTION FEES |
|-------------+--------+----------+---------------------+---------------------|
|           1 | ANKIT  | M        | GOLD                |                 200 |
+-------------+--------+----------+---------------------+---------------------+
ENTER THE ID OF MEMBER TO REMOVE : 1
MEMBER WITH MEMBER ID : 1 REMOVED
```

Displaying Trainers:



```
        |||||                                      |||||
        ||||      ______________________            ||||
        |||======|   XFORCE GYM HOME    |==========|||
        ||||      ______________________            ||||
        |||||                                      |||||



                1. ADD TRAINER
                2. ADD MEMBER
                3. REMOVE TRAINER
                4. REMOVE MEMBER
                5. MODIFY
                6. SHOW TRAINERS
                7. SHOW MEMBERS
                8. GO BACK


ENTER THE ACTION TO PERFORM (1, 2, 3, 4, 5, 6, 7, 8) : 6

+--------------+--------+-------+----------+----------+
|   TRAINER ID | NAME   |   AGE | GENDER   |   SALARY |
|--------------+--------+-------+----------+----------|
|            2 | DHRUV  |    26 | M        |   120000 |
+--------------+--------+-------+----------+----------+
```

Displaying Members:



```
        |||||                                       |||||
        ||||        ______________________           ||||
        |||═══════╡    XFORCE GYM HOME     ╞══════════|||
        ||||        ______________________           ||||
        |||||                                       |||||



                1. ADD TRAINER
                2. ADD MEMBER
                3. REMOVE TRAINER
                4. REMOVE MEMBER
                5. MODIFY
                6. SHOW TRAINERS
                7. SHOW MEMBERS
                8. GO BACK


ENTER THE ACTION TO PERFORM (1, 2, 3, 4, 5, 6, 7, 8) : 7

+─────────────+─────────+──────────+──────────────────────+──────────────────────+
|   MEMBER ID | NAME    | GENDER   | SUBSCRIPTION PLAN    |    SUBSCRIPTION FEES |
├─────────────+─────────+──────────+──────────────────────+──────────────────────┤
|           2 | SHUBHAM | M        | PLATINUM             |                  300 |
+─────────────+─────────+──────────+──────────────────────+──────────────────────+
```

BIBLIOGRAPHY

1. www.geeksforgeeks.org
2. www.w3schools.com
3. www.stackoverflow.com
4. www.python.org
5. www.mysql.com
6. www.wikipedia.com
7. www.hostinger.com
8. www.oracle.com
9. www.pypi.org
10. www.askpython.com
11. www.analyticsindiamag.com
12. www.pyend.readthedocs.io
13. www.github.com
14. www.qr.io
15. www.codeimage.dev