UE21CS351A - DATABASE MANAGEMENT SYSTEM

MINI PROJECT

WEALTHWISE: AN EFFICIENT FUND AND CUSTOMER DATA MANAGEMENT SYSTEM FOR ASSET MANAGEMENT COMPANIES

SECTION: D

TEAM MEMBERS:

1. **Gagan H R - PES1UG21CS196**
2. **Ganta Srilalitha - PES1UG21CS200**
3. **G Vishnupriya Reddy - PES1UG21CS194**

TABLE OF CONTENTS

INTRODUCTION

PURPOSE OF THE PROJECT

The primary purpose of the "WealthWise" project is to develop a robust and efficient database management system tailored to the specific needs of asset management companies. An asset management company (AMC) is a financial institution or organization that specializes in managing, investing and overseeing various types of assets for clients. These clients can be individuals, institutions or trusts. The AMC invests in various financial instruments, including stocks, bonds, real estate and mutual funds. The primary goal of this project is to develop a platform with an efficient database which can help customers grow and preserve their wealth by making informed investment decisions. This innovative system aims to streamline and enhance the management of assets, investments, and client information. The platform also provides asset managers with the tools they need to enhance efficiency, precision, and client services.

SCOPE OF THE PROJECT

The scope of the "WealthWise" project is comprehensive involving development of a robust database management system tailored to the specific needs of asset management companies. This scope includes designing and building a scalable relational database, creating an intuitive web-based user interface, enabling efficient tracking and management of various asset funds, maintaining detailed client profiles along with their investments, ensuring data integrity, and implementing features of user access control. The project also manages funds investments in various asset classes. Additionally, the platform also ensures that the bank account’s of the asset management company are efficiently managed. Furthermore, fund managers and their certificates can also be easily tracked and managed with this relational database. The project's comprehensive scope ensures that "WealthWise" is a complete solution for enhancing the efficiency and the precision of managing funds, fund managers and client services of asset management companies within the competitive finance industry.

PROJECT DESCRIPTION

PROJECT OVERVIEW

The "WealthWise" project represents a comprehensive solution in the field of financial services, tailored to meet the unique needs of asset management companies (AMCs), fund managers, and their clients. This innovative database management system is designed to centralize, streamline, and enhance the management of diverse financial assets, investment funds, transactions, and client relationships, beginning a new era of efficiency and precision in the management of funds.

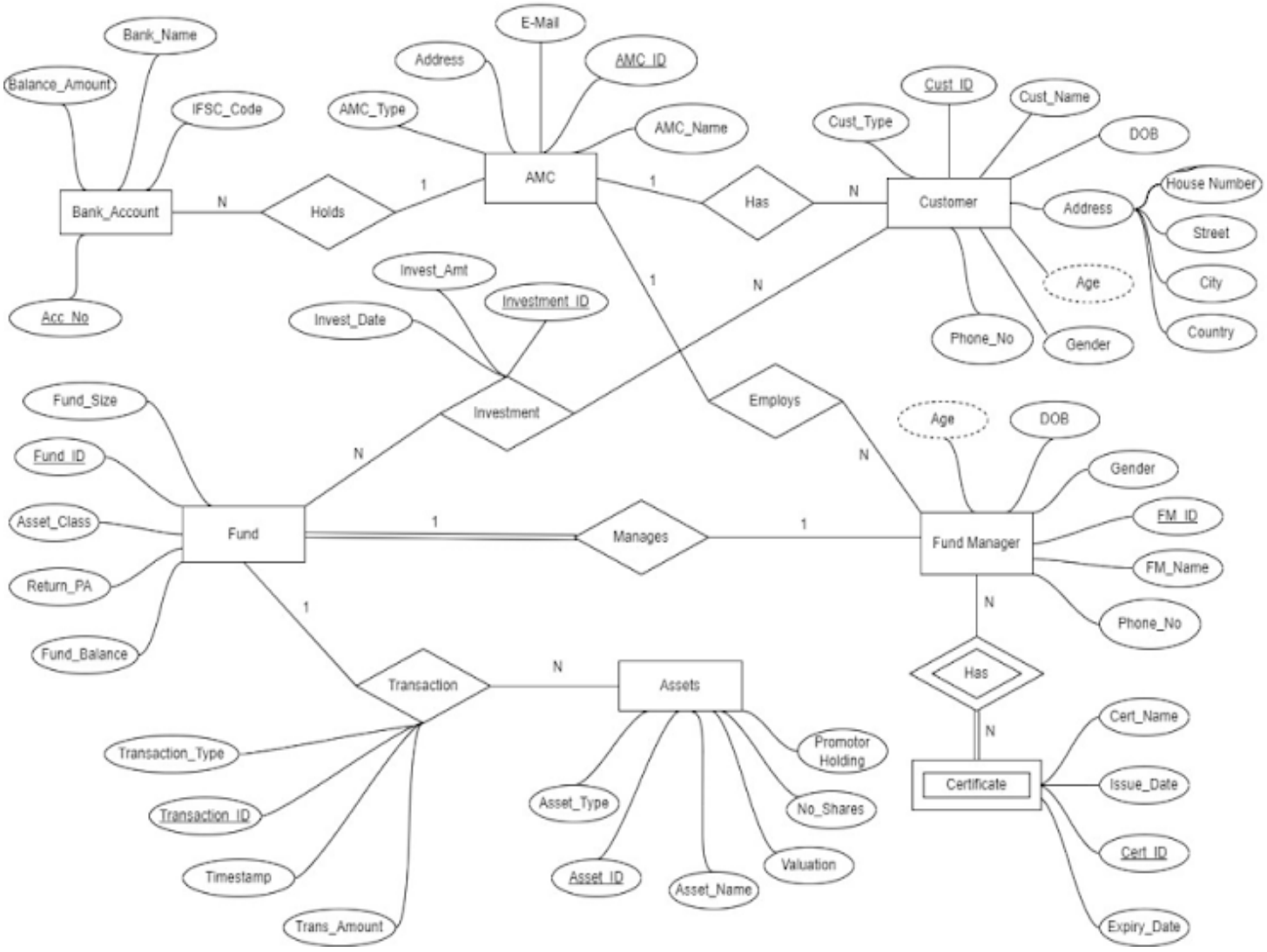
The project is developed with the creation of a robust relational database capable of efficiently storing and managing a wide array of data related to assets, investment funds, transactions, and client information. The database serves as a foundational infrastructure, enabling the centralization and secure storage of critical data. Accessible through an intuitive web-based interface, the system provides authorized users within banks, AMCs, and fund management firms with the tools necessary to optimize asset management and client services.

The project encompasses a broad spectrum of functionalities, from asset and fund management to transactions and client investments and management. It empowers fund managers to effectively track and manage a diverse portfolio of assets, including equities, bonds, real estate, and more. Keeping track of all investments and assets in the AMC ensures regulatory compliance, crucial for financial institutions' commitment to industry standards.

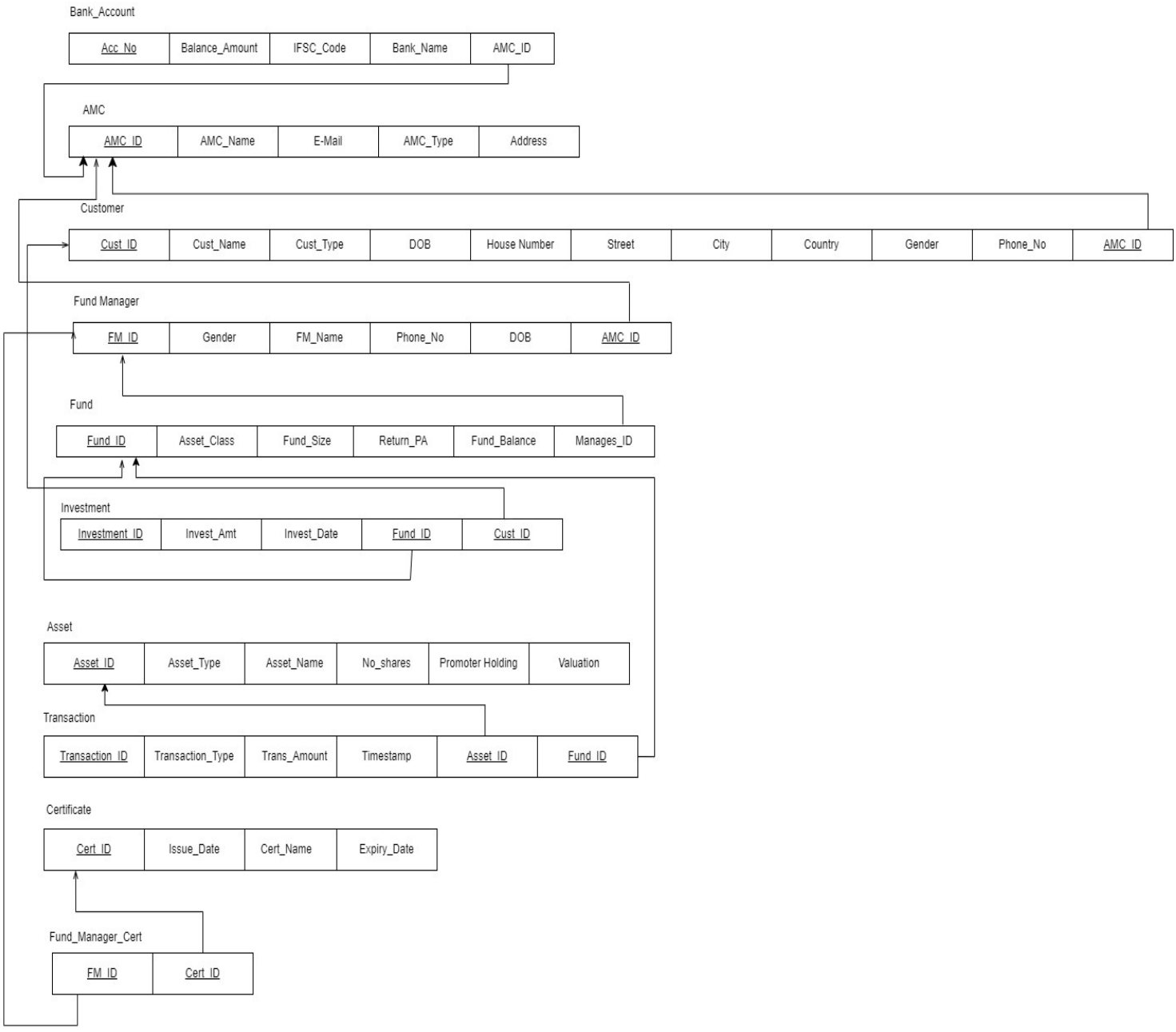
SOFTWARE USED

Programming Language: Python - Used for interaction with the database
Frontend: Python Tkinter
Database: MySQL

ENTITY RELATIONSHIP DIAGRAM



RELATIONAL SCHEMA



DDL SQL COMMANDS

The following are the DDL SQL Commands used in the WealthWise - DBMS for Wealth and Customer Management project:

The create commands are in the following order:

1. Create AMC table
2. Create Asset table
3. Create Bank table
4. Create Certificate table
5. Create Customer table
6. Create Fund Manager table
7. Create Fund table
8. Create Investment table
9. Create Transaction table

The drop commands are in the following order:

1. Drop a user
2. Drop a trigger
3. Drop a procedure

The following are the detailed SQL Commands for the above mentioned queries:

```
CREATE DATABASE IF NOT EXISTS dbms;
USE dbms;

CREATE TABLE IF NOT EXISTS amc_rec (
    amc_id INT NOT NULL PRIMARY KEY,
    amc_name VARCHAR(255),
    address VARCHAR(255),
    amc_type VARCHAR(255)
);

CREATE TABLE IF NOT EXISTS asset (
    asset_id INT NOT NULL PRIMARY KEY,
```

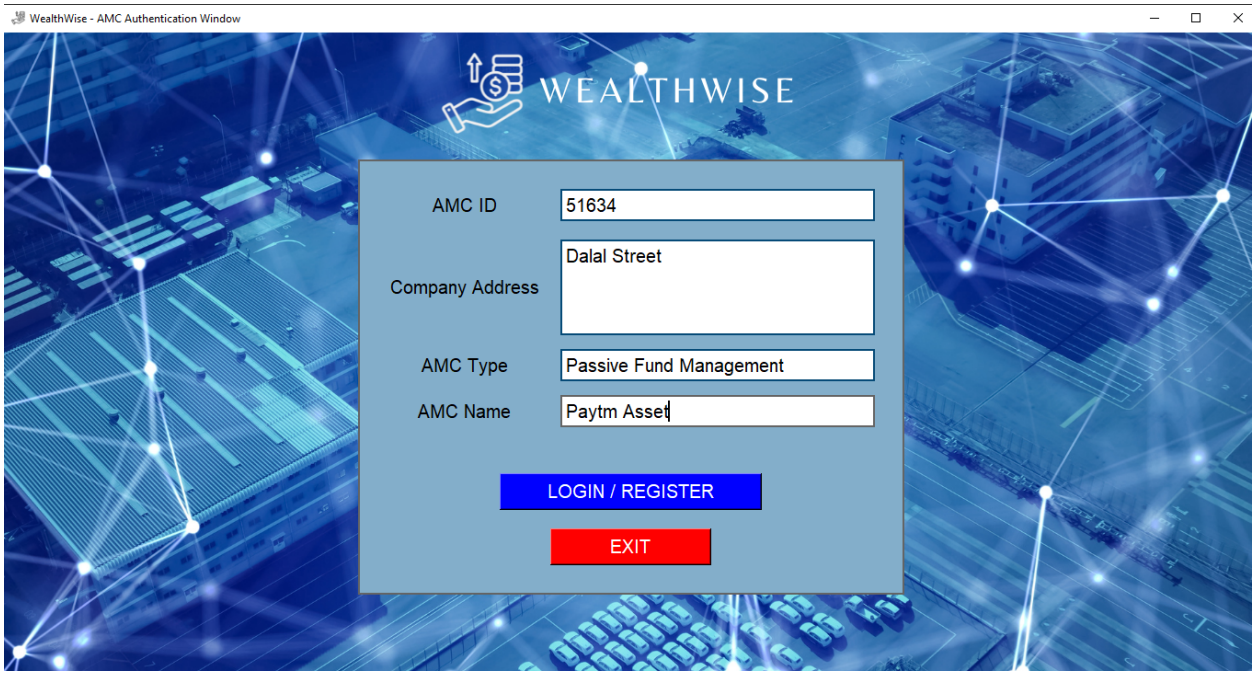
```
    asset_type VARCHAR(255),
    asset_name VARCHAR(255),
    no_of_shares INT,
    promoter_holding VARCHAR(255),
    valuation VARCHAR(255)
);

CREATE TABLE IF NOT EXISTS bank (
    ifsc_code VARCHAR(20),
    bank_name VARCHAR(255),
    account_no INT NOT NULL PRIMARY KEY,
    balance_amount DECIMAL(10, 2),
    amc_id INT,
    FOREIGN KEY (amc_id) REFERENCES amc_rec(amc_id)
);

CREATE TABLE IF NOT EXISTS certificate (
    cert_id INT NOT NULL PRIMARY KEY,
    cert_name VARCHAR(255),
    issue_date DATE,
    expiry_date DATE
);

CREATE TABLE IF NOT EXISTS cus (
    cus_id INT NOT NULL PRIMARY KEY,
    cus_dob DATE,
    cus_type VARCHAR(255),
    cus_name VARCHAR(255),
    cus_addr VARCHAR(255),
    cus_g VARCHAR(1),
    cus_ph CHAR(10),
    cus_age INT,
    amc_id INT,
    FOREIGN KEY (amc_id) REFERENCES amc_rec(amc_id)
);

CREATE TABLE IF NOT EXISTS fund_manager (
    fm_id INT NOT NULL PRIMARY KEY,
    amc_id INT,
    FOREIGN KEY (amc_id) REFERENCES amc_rec(amc_id)
```

```
);

CREATE TABLE IF NOT EXISTS fund (
    fund_id INT NOT NULL PRIMARY KEY,
    asset_class VARCHAR(255),
    fund_size INT,
    return_pa INT,
    fund_balance INT,
    manages_id INT,
    FOREIGN KEY (manages_id) REFERENCES fund_manager(fm_id)
);

CREATE TABLE IF NOT EXISTS investment (
    inv_id INT NOT NULL PRIMARY KEY,
    invest_amt INT,
    fund_id INT,
    cus_id INT,
    invest_date DATE,
    FOREIGN KEY (fund_id) REFERENCES fund(fund_id),
    FOREIGN KEY (cus_id) REFERENCES cus(cus_id)
);

CREATE TABLE IF NOT EXISTS transaction (
    transaction_id INT NOT NULL PRIMARY KEY,
    timestamp DATETIME,
    transaction_amount INT,
    transaction_type VARCHAR(255),
    asset_id INT,
    FOREIGN KEY (asset_id) REFERENCES asset(asset_id)
);


-- Stored procedure to add a certificate to the database
CREATE PROCEDURE addcert(
    IN cert_id_param INT,
    IN cert_name_param VARCHAR(255),
    IN issue_date_param DATE,
    IN expiry_date_param DATE
)
```

```
-- Query to create a user with specified privileges
CREATE USER '{username}'@'localhost' IDENTIFIED BY '{password}';

-- Drop a user
DROP USER '{username}'@'localhost';

-- Drop a trigger
DROP TRIGGER IF EXISTS update_fund_balance;
DROP TRIGGER IF EXISTS log_transaction;

-- Drop a stored procedure
DROP PROCEDURE IF EXISTS addcert;

-- Drop a table
DROP TABLE IF EXISTS transaction_log;
```

CRUD OPERATIONS

1. Registration of Asset Management Companies in AMC Registration Table




```
+--------+-----------------------+--------------+--------------------------+
| amc_id | amc_name              | address      | amc_type                 |
+--------+-----------------------+--------------+--------------------------+
|      1 | a                     | a            | a                        |
|  12343 | Axis Asset Management | Delhi        | Mutual Funds             |
|  12356 | Zerodha               | Bangalore    | Active Fund Management   |
|  51634 | Paytm Asset           | Dalal Street | Passive Fund Management  |
+--------+-----------------------+--------------+--------------------------+
4 rows in set (0.00 sec)
```

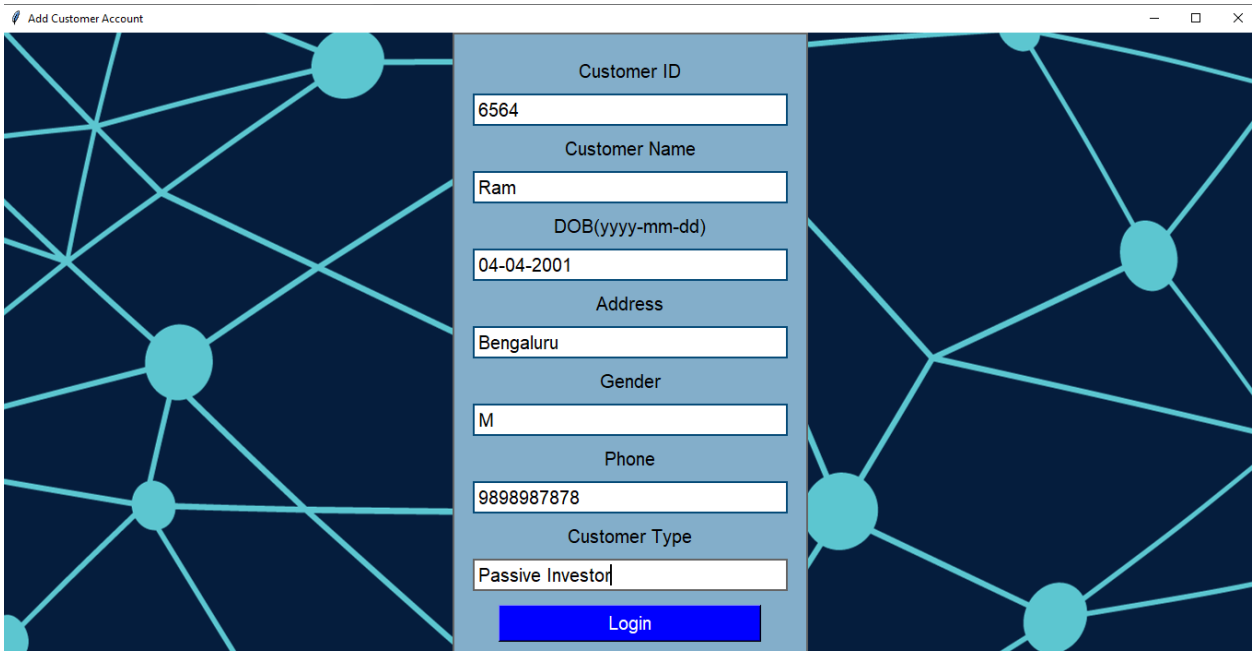
2. Addition of new bank accounts to Asset Management Company

```
+------------+------------+------------+----------------+--------+
| ifsc_code  | bank_name  | account_no | balance_amount | amc_id |
+------------+------------+------------+----------------+--------+
| SBI1234    | SBI        |      12345 |       10000.00 |  51634 |
| BOB94      | BOB        |      94949 |       50000.00 |      1 |
| HDFC99     | HDFC       |      99999 |      100000.00 |      1 |
| HDFC00GT65 | HDFC       |   67532947 |      500000.00 |  12356 |
| UB001IN90  | Union Bank |   99657485 |      100000.00 |  12356 |
+------------+------------+------------+----------------+--------+
5 rows in set (0.00 sec)
```

3. View Bank Account Details - READ

4. Searching for a particular bank account - READ

5. Addition of new customers and investors



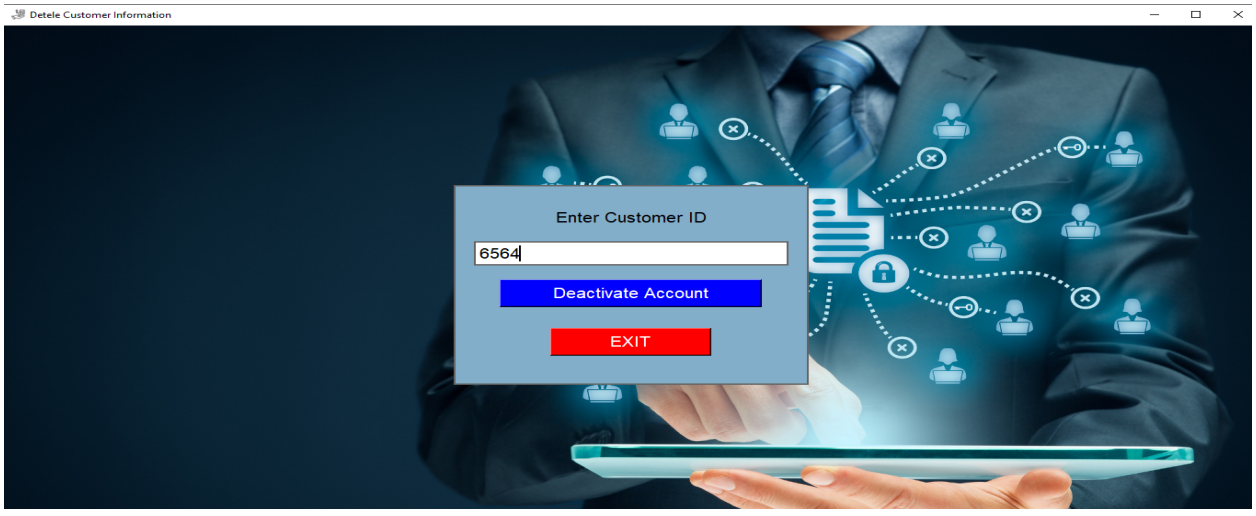

```
mysql> select * from cus;
+--------+------------+--------------------+----------+-----------+-------+------------+----------+--------+
| cus_id | cus_dob    | cus_type           | cus_name | cus_addr  | cus_g | cus_ph     | cus_age | amc_id |
+--------+------------+--------------------+----------+-----------+-------+------------+----------+--------+
|   1999 | 2000-01-01 | Long-Term Investor | Kumar    | Bangalore | M     | 9898998989 |      23 |      1 |
|   4444 | 2000-01-01 | Permanent          | Kumar    | Mumbai    | M     | 8888888888 |      23 |  51634 |
+--------+------------+--------------------+----------+-----------+-------+------------+----------+--------+
2 rows in set (0.05 sec)

mysql> select * from cus;
+--------+------------+--------------------+----------+-----------+-------+------------+----------+--------+
| cus_id | cus_dob    | cus_type           | cus_name | cus_addr  | cus_g | cus_ph     | cus_age | amc_id |
+--------+------------+--------------------+----------+-----------+-------+------------+----------+--------+
|   1999 | 2000-01-01 | Long-Term Investor | Kumar    | Bangalore | M     | 9898998989 |      23 |      1 |
|   4444 | 2000-01-01 | Permanent          | Kumar    | Mumbai    | M     | 8888888888 |      23 |  51634 |
|   6564 | 2001-04-04 | Passive Investor   | Ram      | Bengaluru | M     | 9898987878 |      22 |  51634 |
+--------+------------+--------------------+----------+-----------+-------+------------+----------+--------+
3 rows in set (0.00 sec)
```

6. Update personal details (address) of the customer in AMC


```
mysql> select * from cus;
+--------+------------+--------------------+----------+-----------+-------+------------+----------+--------+
| cus_id | cus_dob    | cus_type           | cus_name | cus_addr  | cus_g | cus_ph     | cus_age | amc_id |
+--------+------------+--------------------+----------+-----------+-------+------------+----------+--------+
|   1999 | 2000-01-01 | Long-Term Investor | Kumar    | Bangalore | M     | 9898998989 |       23 |      1 |
|   4444 | 2000-01-01 | Permanent          | Kumar    | Mumbai    | M     | 8888888888 |       23 |  51634 |
|   6564 | 2001-04-04 | Passive Investor   | Ram      | Bengaluru | M     | 9898987878 |       22 |  51634 |
+--------+------------+--------------------+----------+-----------+-------+------------+----------+--------+
3 rows in set (0.00 sec)

mysql> select * from cus;
+--------+------------+--------------------+----------+-----------+-------+------------+----------+--------+
| cus_id | cus_dob    | cus_type           | cus_name | cus_addr  | cus_g | cus_ph     | cus_age | amc_id |
+--------+------------+--------------------+----------+-----------+-------+------------+----------+--------+
|   1999 | 2000-01-01 | Long-Term Investor | Kumar    | Bangalore | M     | 9898998989 |       23 |      1 |
|   4444 | 2000-01-01 | Permanent          | Kumar    | Mumbai    | M     | 8888888888 |       23 |  51634 |
|   6564 | 2001-04-04 | Passive Investor   | Ram      | Kashmir   | M     | 9898987878 |       22 |  51634 |
+--------+------------+--------------------+----------+-----------+-------+------------+----------+--------+
3 rows in set (0.00 sec)
```

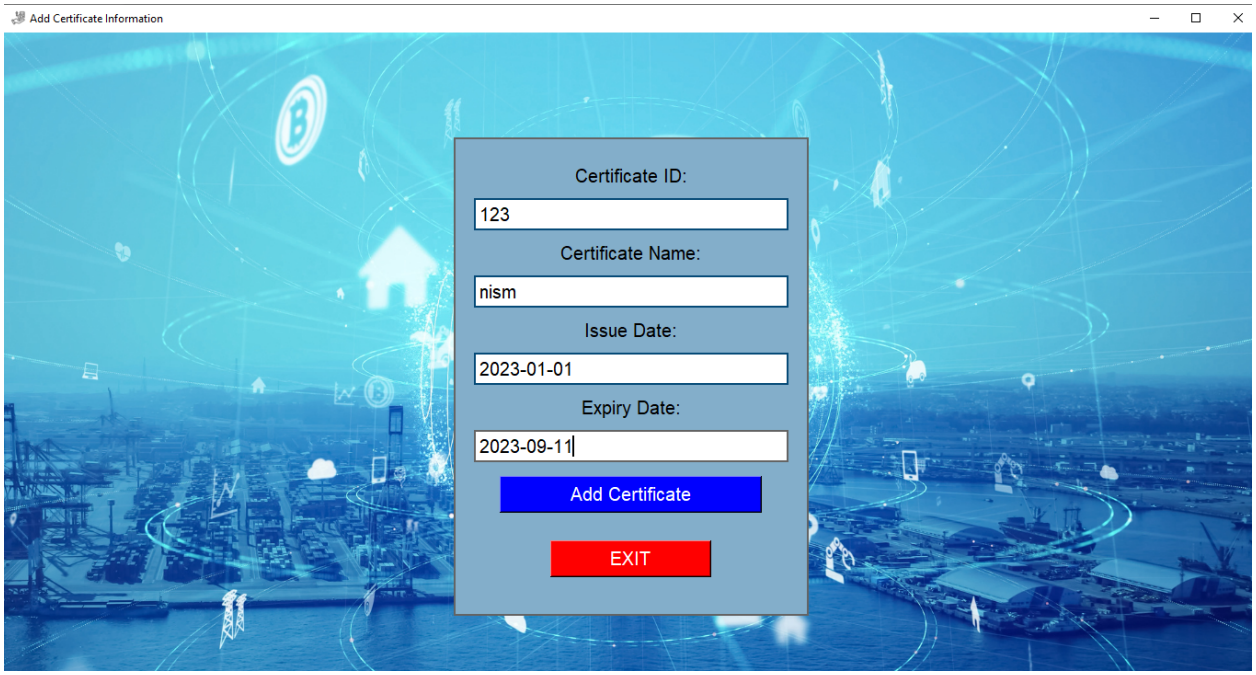
Address changed from Bengaluru to Kashmir using the update command

7. Delete or deactive a customer's account



```
mysql> select * from cus;
+--------+------------+--------------------+----------+-----------+-------+------------+---------+--------+
| cus_id | cus_dob    | cus_type           | cus_name | cus_addr  | cus_g | cus_ph     | cus_age | amc_id |
+--------+------------+--------------------+----------+-----------+-------+------------+---------+--------+
|   1999 | 2000-01-01 | Long-Term Investor | Kumar    | Bangalore | M     | 9898998989 |      23 |      1 |
|   4444 | 2000-01-01 | Permanent          | Kumar    | Mumbai    | M     | 8888888888 |      23 |  51634 |
|   6564 | 2001-04-04 | Passive Investor   | Ram      | Kashmir   | M     | 9898987878 |      22 |  51634 |
+--------+------------+--------------------+----------+-----------+-------+------------+---------+--------+
3 rows in set (0.00 sec)

mysql> select * from cus;
+--------+------------+--------------------+----------+-----------+-------+------------+---------+--------+
| cus_id | cus_dob    | cus_type           | cus_name | cus_addr  | cus_g | cus_ph     | cus_age | amc_id |
+--------+------------+--------------------+----------+-----------+-------+------------+---------+--------+
|   1999 | 2000-01-01 | Long-Term Investor | Kumar    | Bangalore | M     | 9898998989 |      23 |      1 |
|   4444 | 2000-01-01 | Permanent          | Kumar    | Mumbai    | M     | 8888888888 |      23 |  51634 |
+--------+------------+--------------------+----------+-----------+-------+------------+---------+--------+
2 rows in set (0.00 sec)
```

8. Addition of new certificates.




```
mysql> select * from certificate;
+---------+-----------+-------------+-------------+
| cert_id | cert_name | issue_date  | expiry_date |
+---------+-----------+-------------+-------------+
|     123 | nism      | 2023-01-01  | 2023-09-11  |
+---------+-----------+-------------+-------------+
1 row in set (0.16 sec)
```

All four operations of CRUD - Create, Read, Update and Delete have been demonstrated, with frontend and tables of before and after as well. This project includes many more CRUD operations which have been listed under the functionalities section of this report.

MAJOR PROJECT FUNCTIONALITIES

Key functionalities include comprehensive asset management , investment fund creation and management, managing certification and roles of the fund manager, transaction tracking, and client and customer management for maintaining client profiles and tracking investment histories.

Client and customer management are at the forefront of the system, enabling banks and AMCs to maintain detailed client profiles and track investment histories. In addition, the project incorporates features to track transactions, investments, and financial products. All investments made by the client can be easily tracked.

The system allows for the addition, modification, and archiving of various asset types, encompassing equities, fixed income, real estate, and an array of financial instruments. Fund managers are assigned funds. All decisions to buy or sell investment instruments are made by the fund manager. The clients invest in these funds. The amount invested is used to make investments in several asset classes as per the funds requirements.

The asset management company will also have to manage one or more bank accounts to deposit and withdraw funds. The amount is kept in the bank by the AMC’s for various reasons. Our project keeps track of all bank transactions as well. A major functionality of the asset management company is to also keep track of its fund managers. Every fund manager is given a fund to make investment decisions. As per the regulations, fund managers need to have one or more certificates to manage certain kinds of funds. The project keeps track of all these certificates, their dates of issue and expiry.

Every transaction made by the fund to buy or sell an asset is recorded with complete details about the transaction. The database will have complete details of all the funds in the asset management company, the details of the companies in which the funds have invested, the amount invested and the returns earned per annum on the investments made. The project also makes sure that the funds of a certain asset class can only make investments in those assets/instruments.

FUNCTIONALITIES WITH SCREENSHOTS

1. User Authentication and Access Control:

- User authentication with role-based access control.
- Role-based permissions to restrict access to sensitive data.
- Customer, AMC and Fund Managers will have different permissions to access information in the database.

2. Fund Management:

- Create, update, and track investment funds
- Allocate assets to funds and rebalance funds
- Assign fund managers to the funds
- Support for various asset classes such as equities, bonds, real estate, and alternative investments.
- Clear and detailed information about each asset/instrument on which the fund has invested can be obtained.
- Utilize investments from clients to purchase assets which match the fund's permissible asset classes.

3. Client Information Management:

- Maintain a comprehensive client database.
- Store personal and financial information, which includes their contact details.
- Enable the client to invest in various funds of various asset classes.
- Track the client's investments in various funds, the amount invested in those funds and the returns the client has earned.

4. Transaction Management:

- Record buy/sell transactions of assets in each fund.
- Keep track of the type of asset traded.
- Calculate transaction costs, update portfolio values, cash balance in the fund, returns gained from the entire buy and sell of the instrument.
- Record the time and date of the transaction.
- Check if the fund has sufficient balance to carry out the asset purchase.
- Also confirming if the fund has a certain asset before selling it.

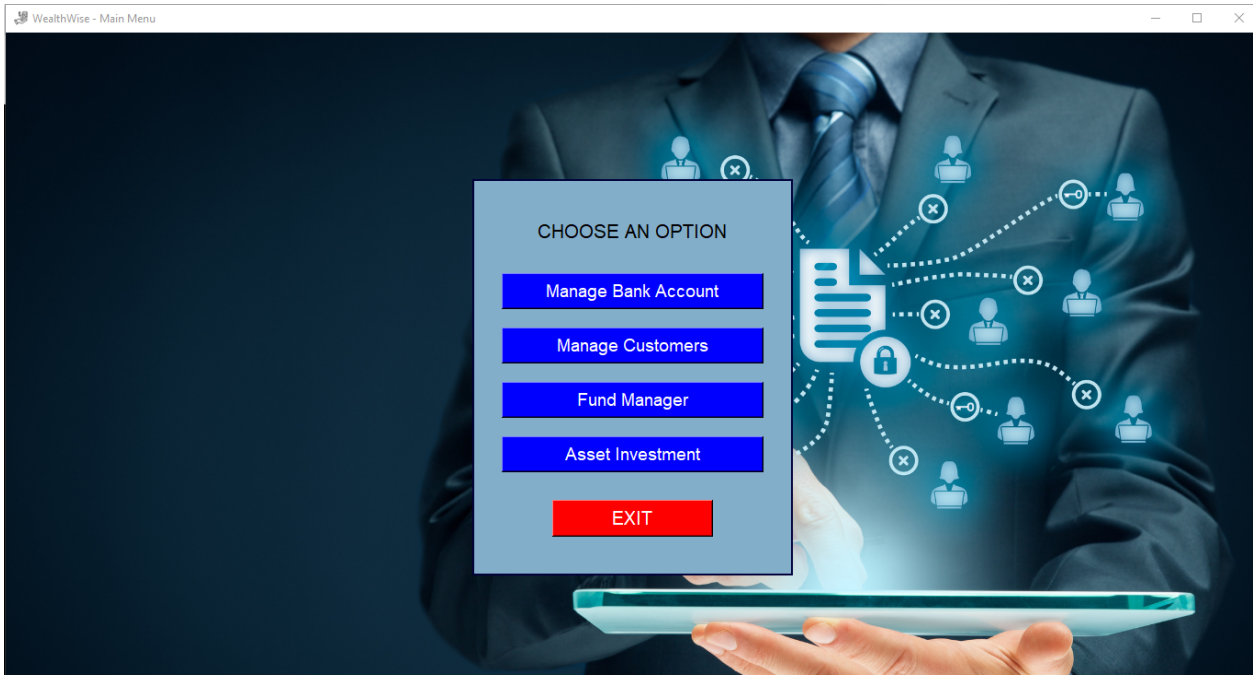
5. Fund Manager and Certifications:

- Maintain a database of fund managers and their relevant certifications.
- Track the expiration dates of certifications to ensure compliance with regulatory requirements.
- Assign handling of a fund to the fund manager.

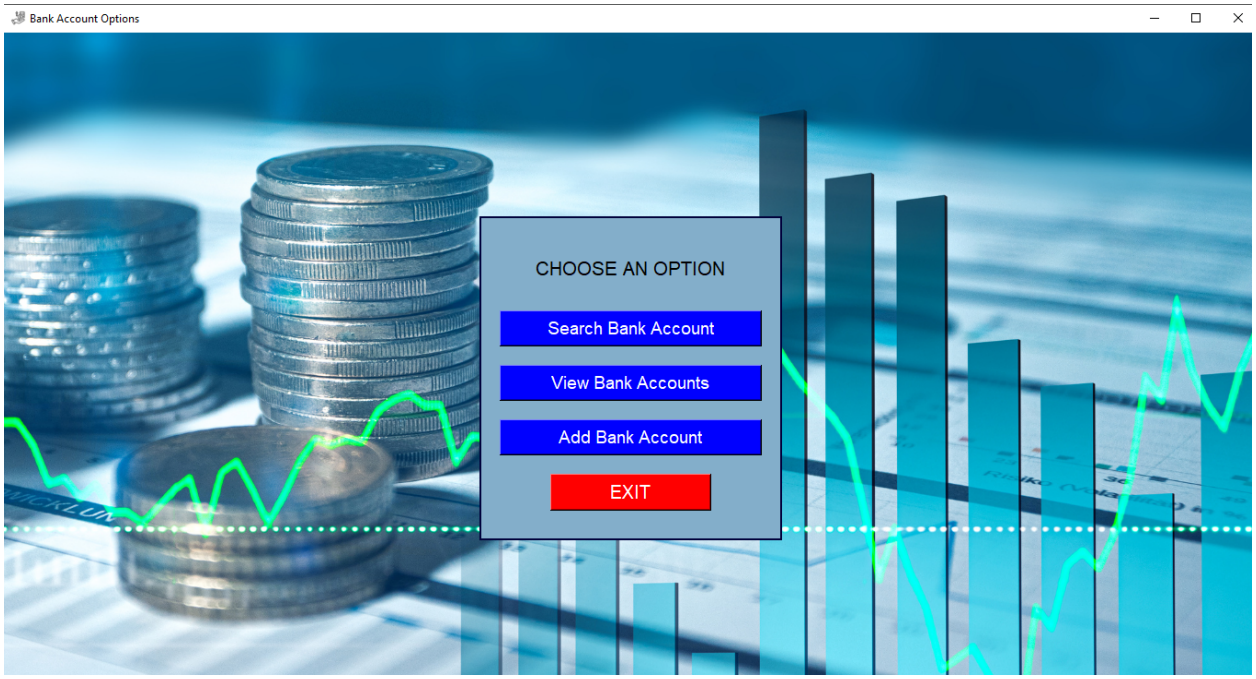
HOME SCREEN:

AMC LOGIN/REGISTER WINDOW:

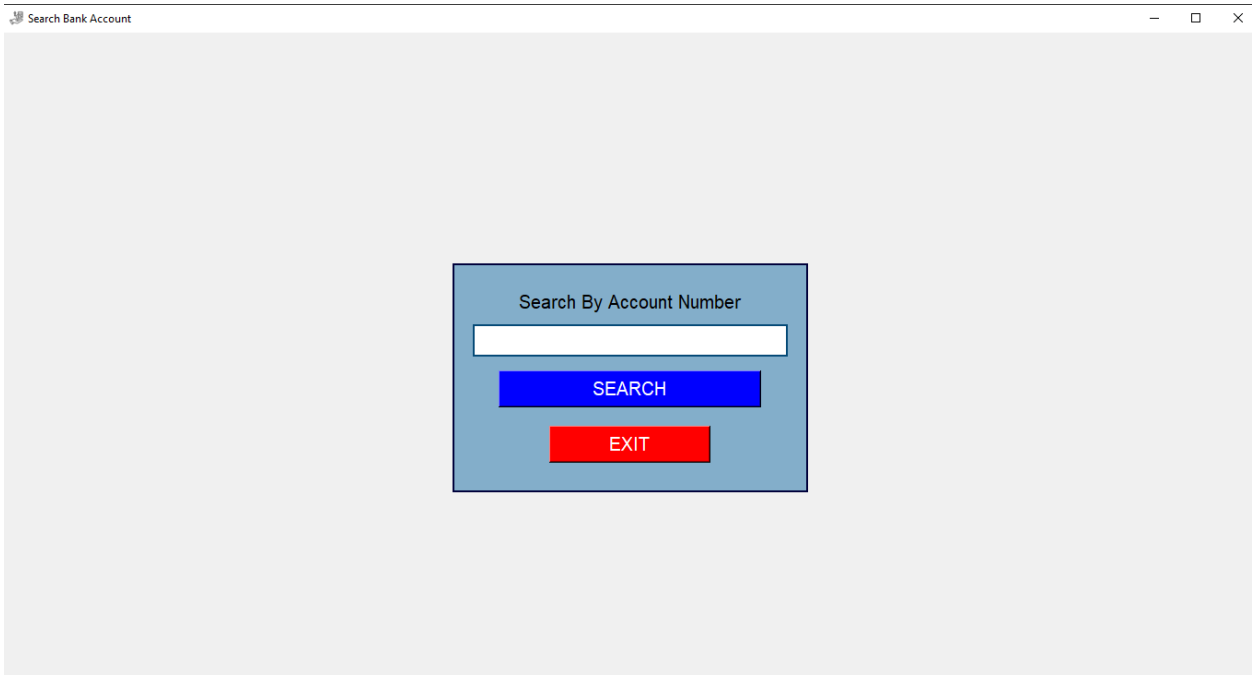
MAIN MENU WINDOW:



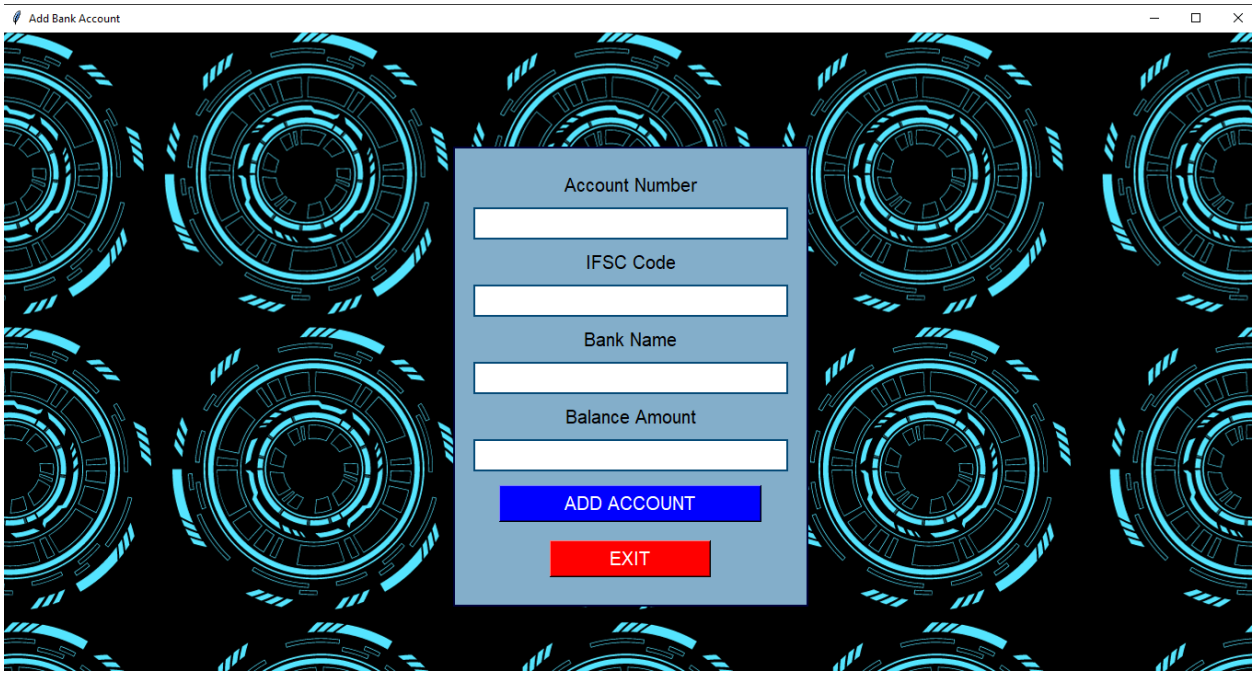
BANK OPTIONS MENU:



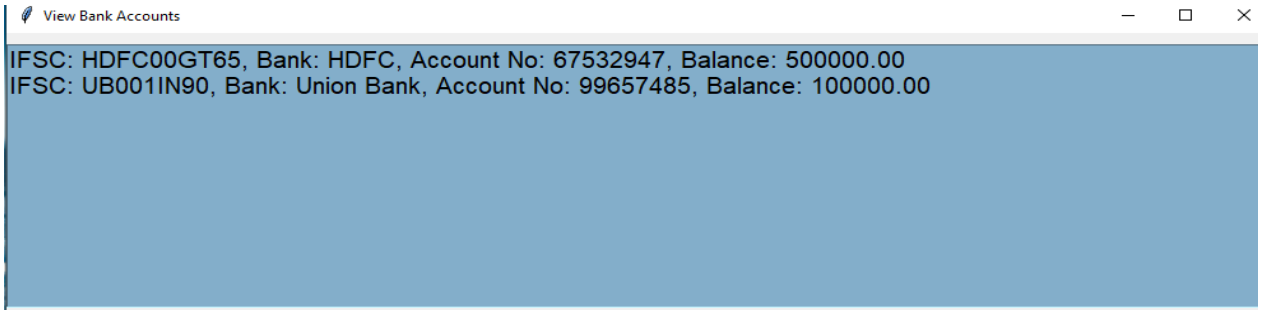
SEARCH ACCOUNT NUMBER WINDOW:



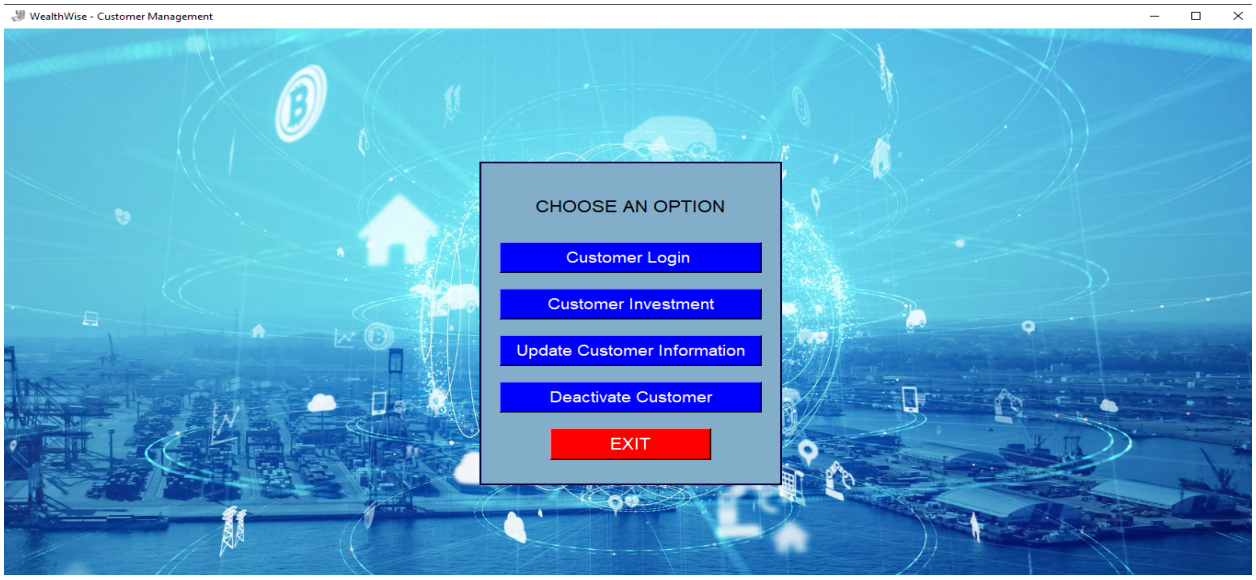
ADD BANK ACCOUNT WINDOW:



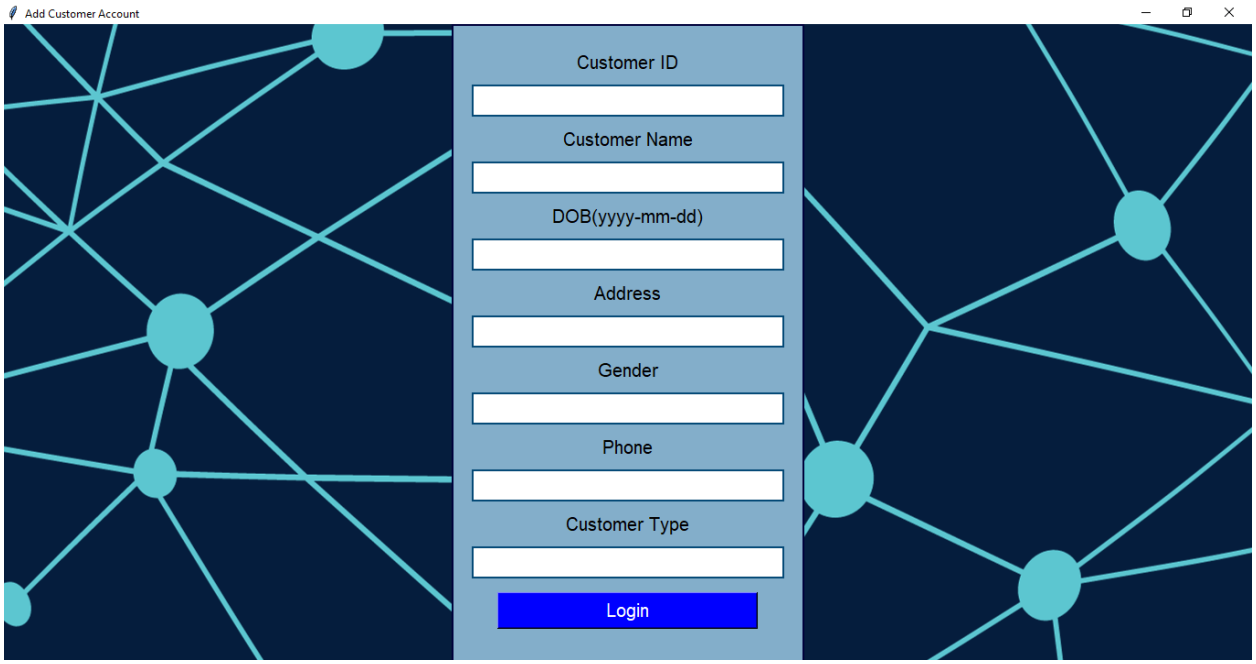
VIEW BANK ACCOUNT WINDOW:



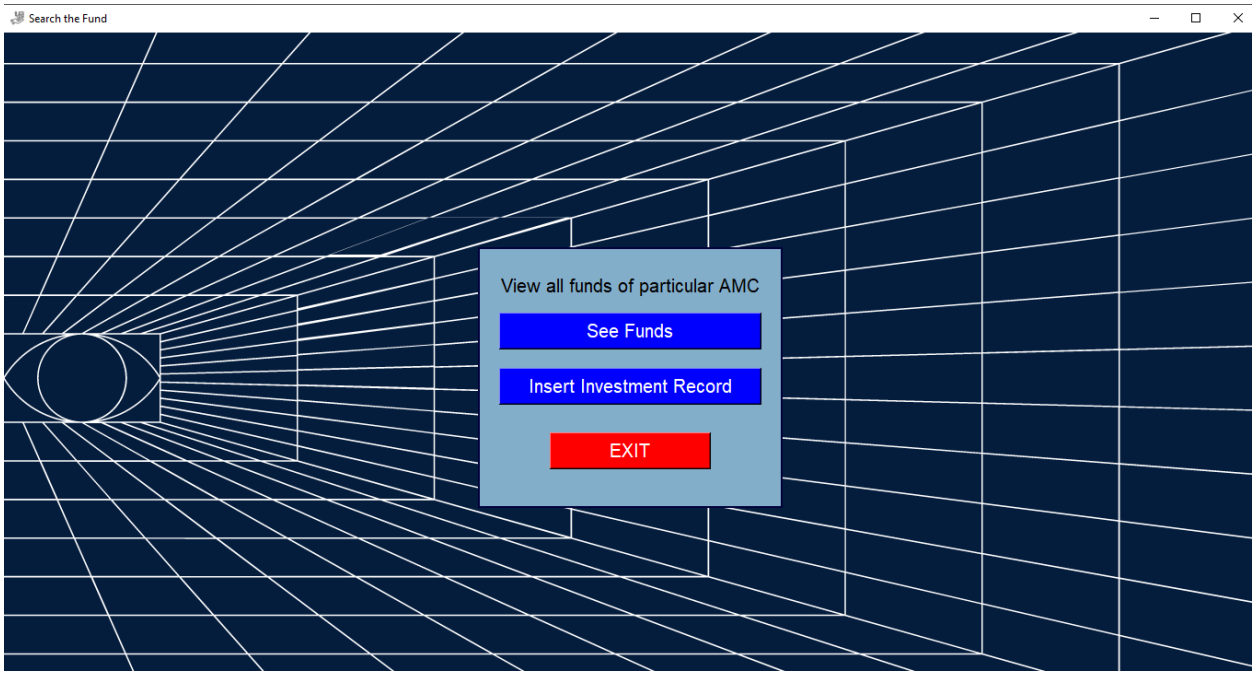
CUSTOMER MANAGEMENT FUNCTIONALITY MENU:



ADD NEW CUSTOMER FEATURE:

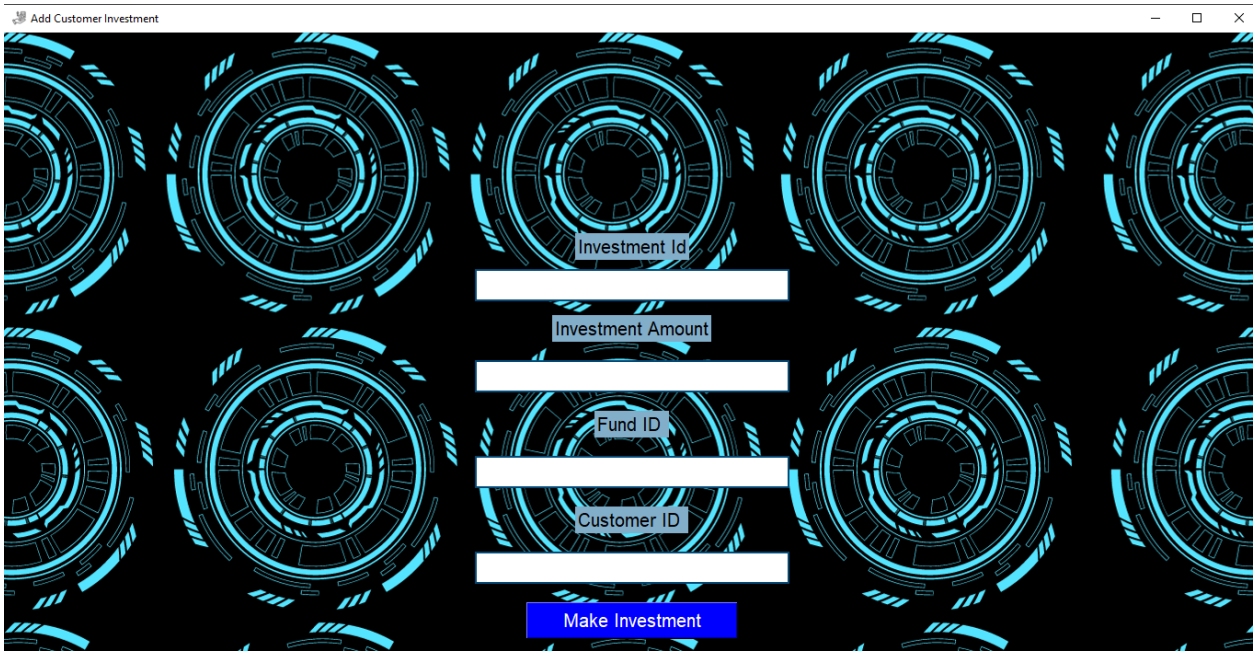


FEATURE TO VIEW ALL FUNDS AND MAKE INVESTMENT IN FUNDS:




```
mysql> select * from fund;
+---------+-------------+-----------+-----------+--------------+------------+
| fund_id | asset_class | fund_size | return_pa | fund_balance | manages_id |
+---------+-------------+-----------+-----------+--------------+------------+
|     111 | a           |        27 |        55 |            3 |          1 |
|     112 | b           |         4 |         2 |           19 |          1 |
|     113 | c           |        43 |        25 |           18 |          2 |
|     114 | b           |        45 |         3 |           20 |          1 |
+---------+-------------+-----------+-----------+--------------+------------+
4 rows in set (0.12 sec)
```

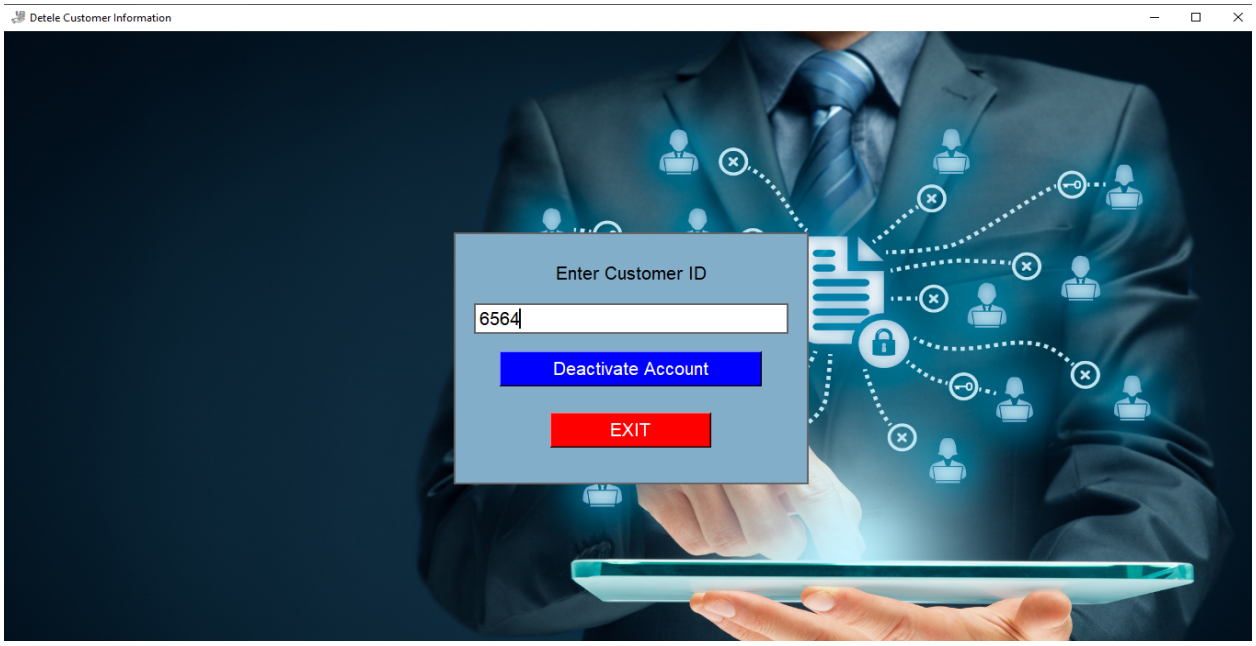
CUSTOMER INVESTMENT IN NEW FUNDS FEATURE:



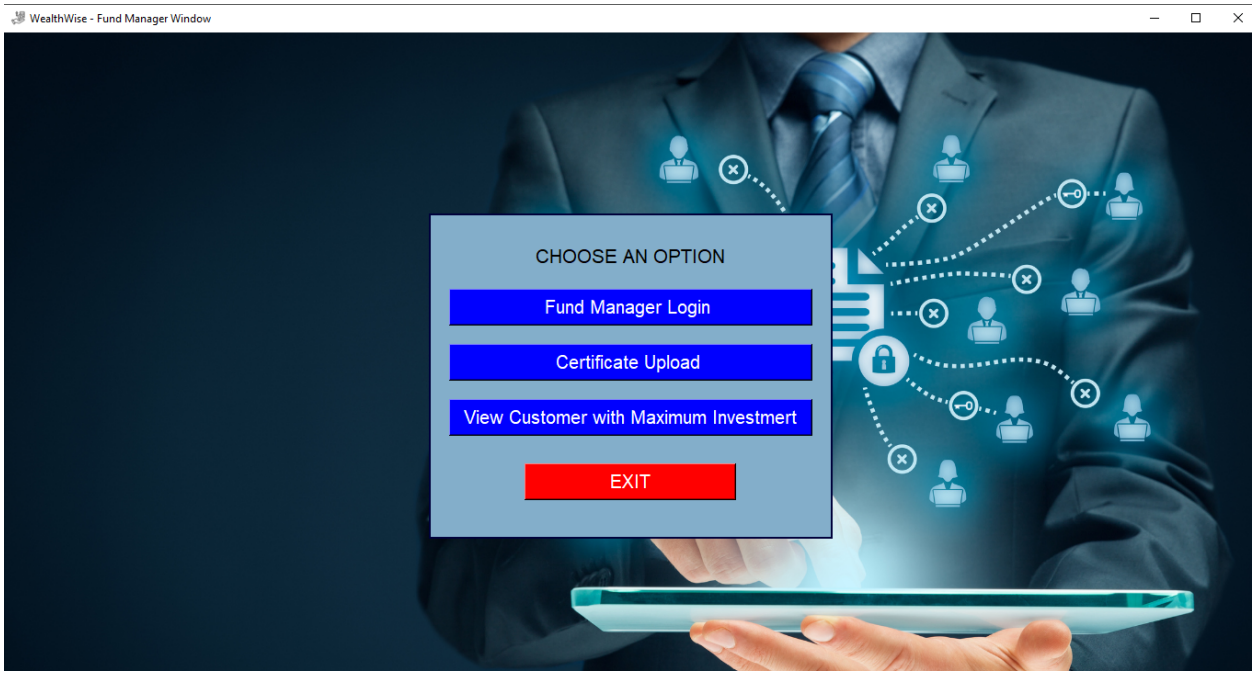

```
mysql> select * from investment;
+--------+-------------+---------+--------+-------------+
| inv_id | invest_amt  | fund_id | cus_id | invest_date |
+--------+-------------+---------+--------+-------------+
|     14 |         890 |     112 |     11 | 2023-10-29  |
|     15 |         500 |     112 |     11 | 2023-10-29  |
|     16 |         400 |     112 |     11 | 2023-11-13  |
|     17 |         600 |     113 |     11 | 2023-11-21  |
|  11156 |          23 |     112 |     11 | 2023-11-21  |
+--------+-------------+---------+--------+-------------+
5 rows in set (0.21 sec)
```

UPDATION OF CUSTOMER PERSONAL DETAILS:

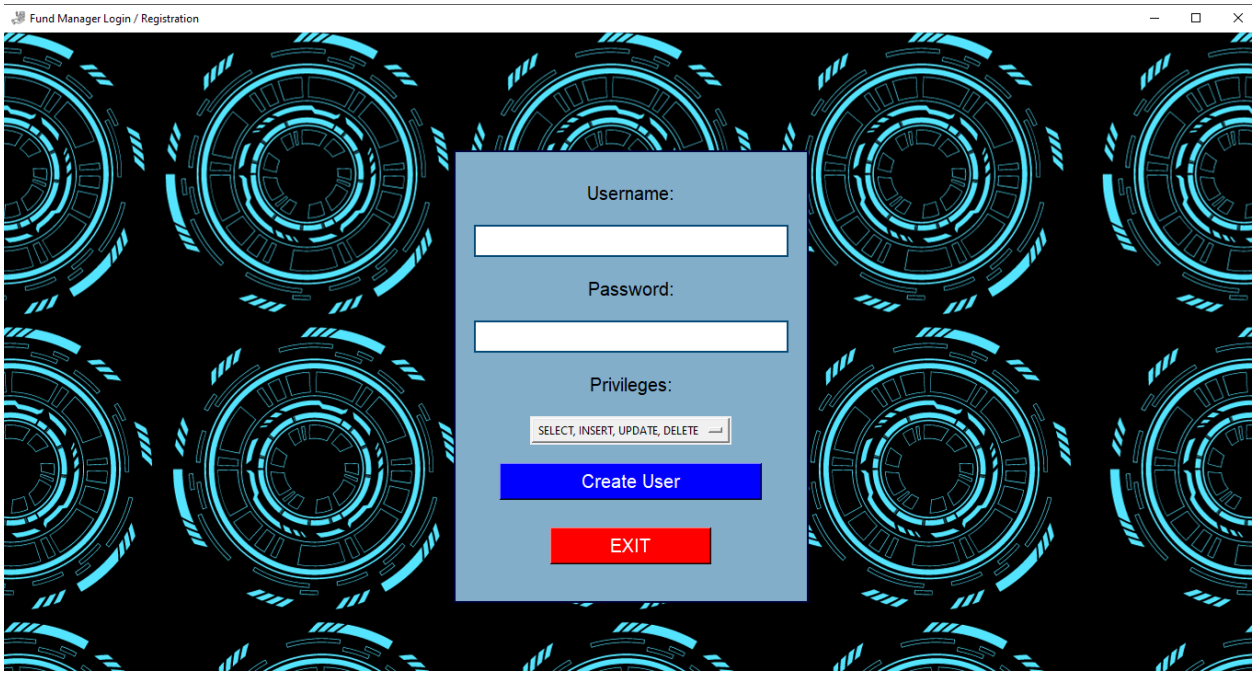
DELETION OR DEACTIVATION OF CUSTOMER ACCOUNT:



FUND MANAGER OPTIONS:

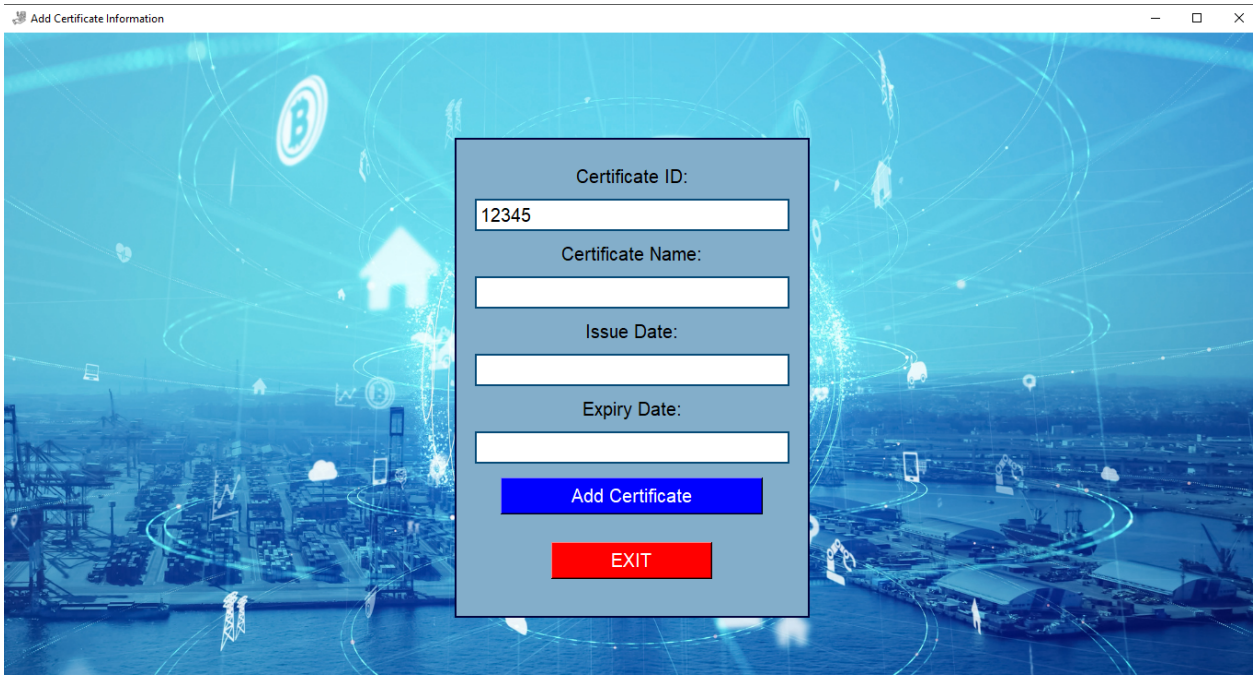


FUND MANAGER LOGIN/REGISTRATION:



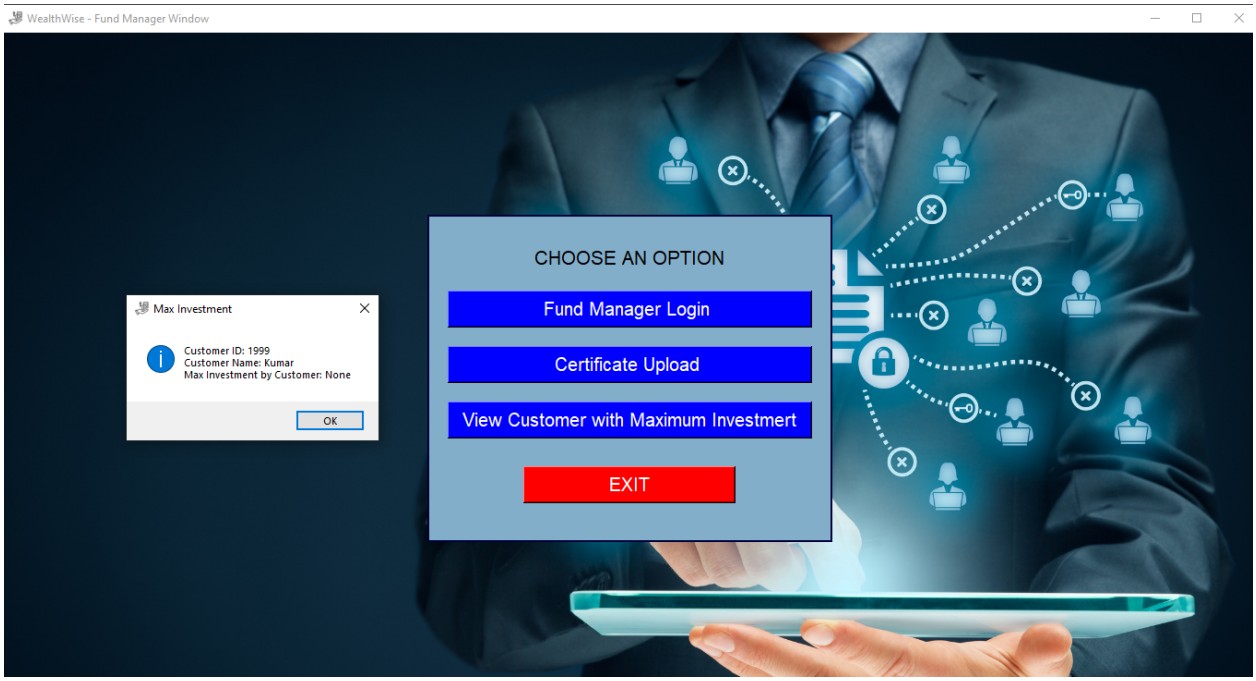
The complete output of this functionality is explained in detail in this report under the “USER PRIVILEGES” section.

CERTIFICATE UPLOAD FOR FUND MANAGERS:

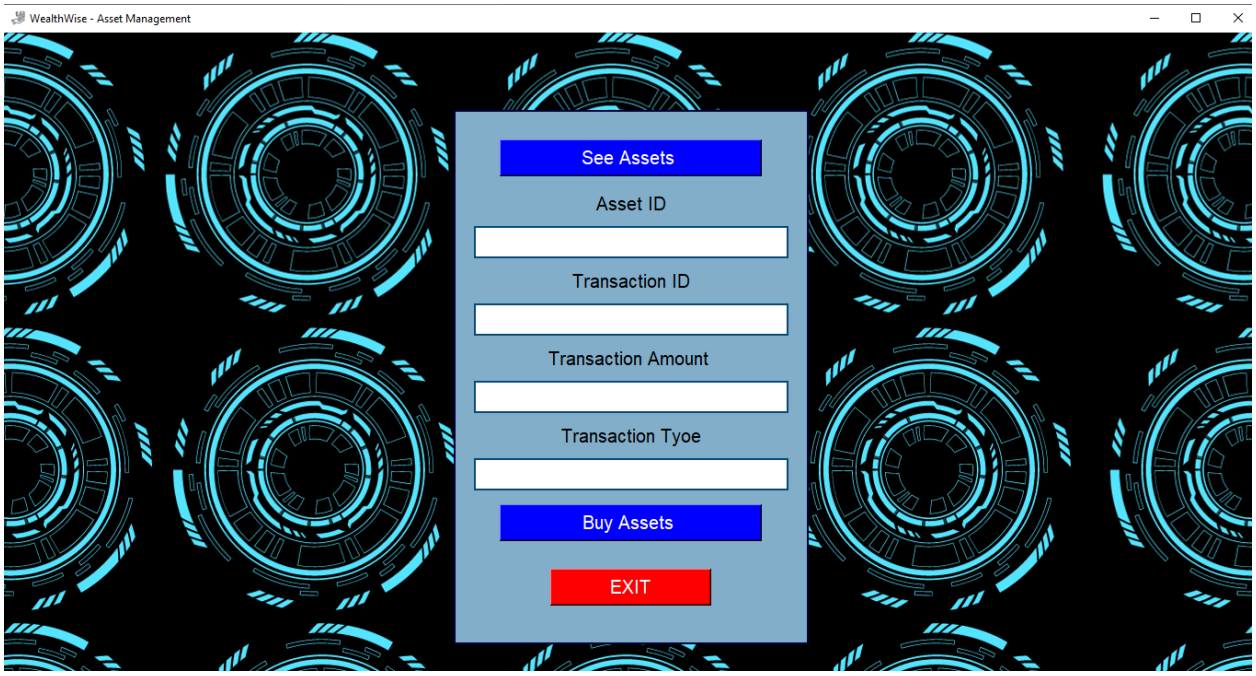


PDF has to be uploaded to the software. The platform reads the unique certificate number from the PDF.

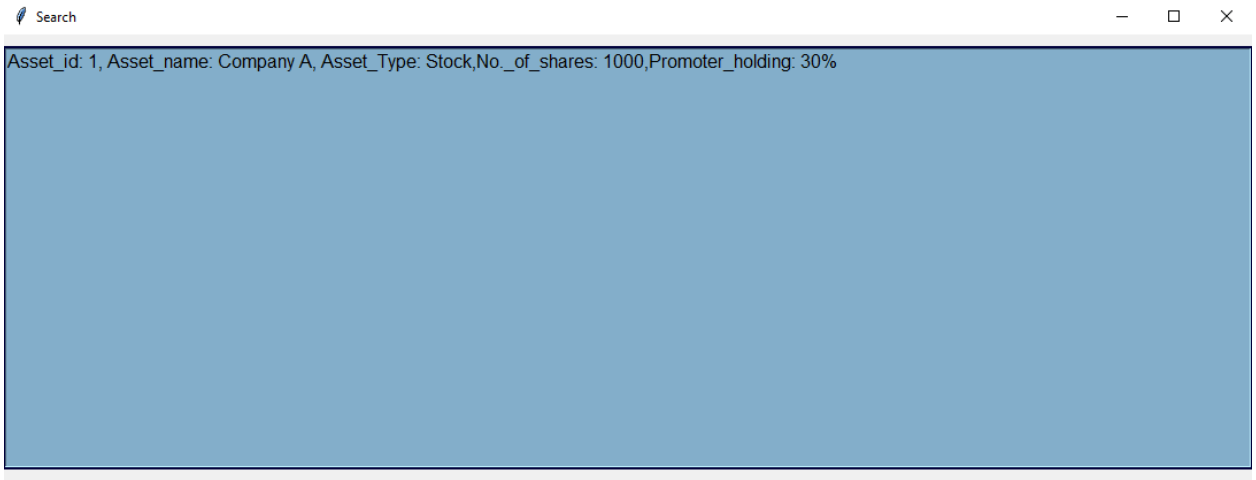
VIEW MAXIMUM INVESTMENT BY A CUSTOMER:



ASSET TRANSACTION WINDOW:



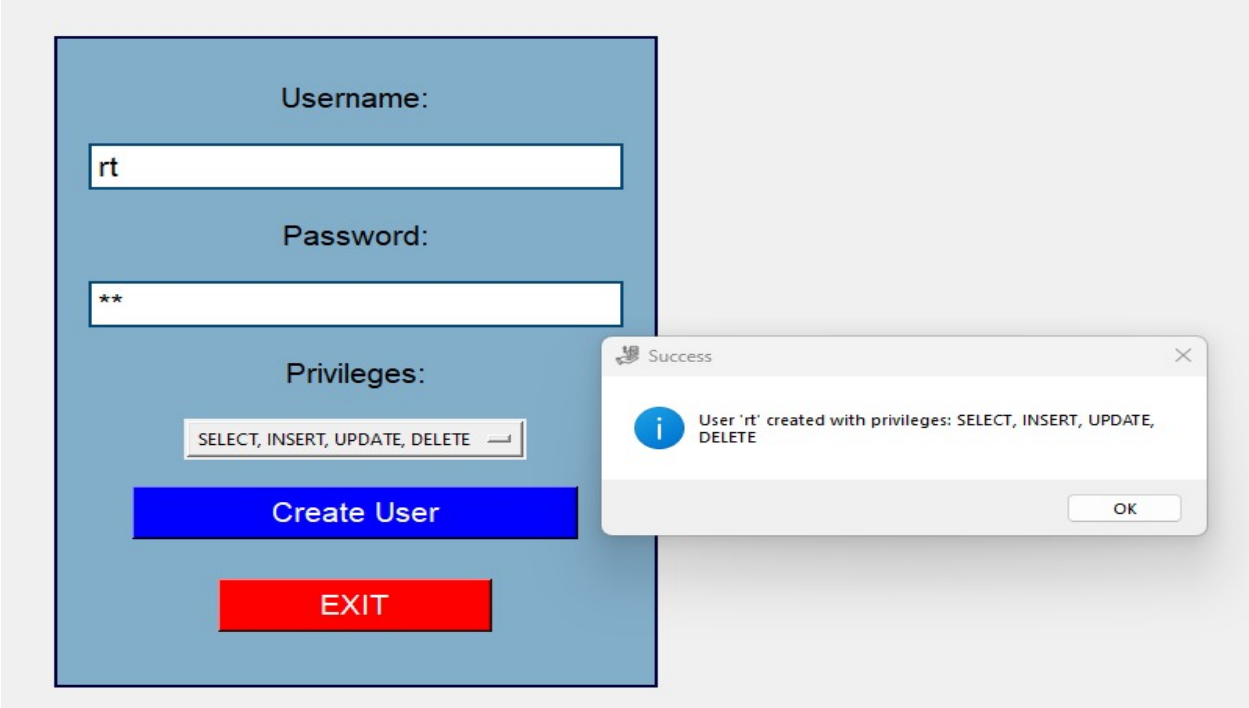
VIEW ASSET FUNCTION:



TRANSACTION CONFIRMATION:

USER CREATION AND USER PRIVILEGES

Fund Managers, based on their experience and certifications are given varied access to the database.



CODE:

```
def create_user():
    username = username_entry.get()
    password = password_entry.get()
    privileges = privileges_var.get()

    try:
        # Create the user
        cursor.execute(f"CREATE USER '{username}'@'localhost' IDENTIFIED BY '{password}'")

        # Grant privileges to the user
        cursor.execute(f"GRANT {privileges} ON your_database.* TO '{username}'@'localhost'")

        # Flush privileges to apply changes
        cursor.execute("FLUSH PRIVILEGES")

        messagebox.showinfo("Success", f"User '{username}' created with privileges: {privileges}")
        ins_fund()
    except mysql.connector.Error as err:
        messagebox.showerror("Error", f"Error creating user: {err}")
```

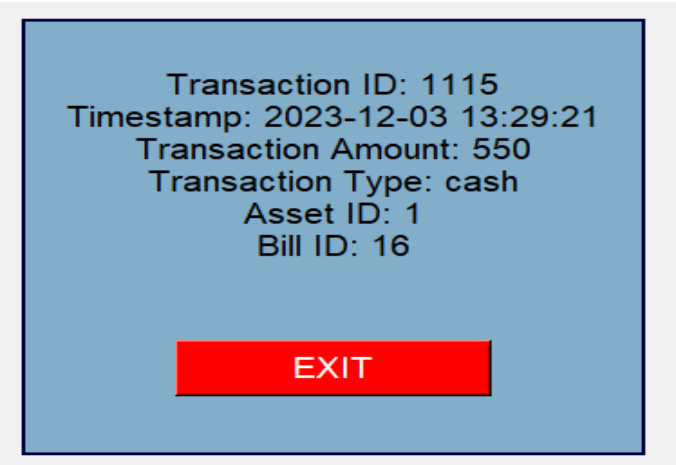
TRIGGERS

Trigger generates a bill(acknowledgement) when a successful transaction has been made.This transaction is a buy/sell order from the fund manager on behalf of the asset management company for funds managed under the AMC. Funds buy or sell assets like gold, silver, bonds and stocks available in the real world. To keep track of all the investments made, the trigger generates a bill, which involves a unique transaction number. This unique transaction number can be used to keep track of all fund investments.

TRIGGER CODE:

```
cursor.execute("""CREATE TABLE IF NOT EXISTS transaction (transaction_id INT, timestamp DATETIME, transaction_amount DECIMAL(10, 2),
                transaction_type VARCHAR(255), asset_id INT)""")
cursor.execute("""CREATE TABLE IF NOT EXISTS bill (bill_id INT AUTO_INCREMENT PRIMARY KEY,
                transaction_id INT,
                timestamp DATETIME, transaction_amount DECIMAL(10, 2),
                transaction_type VARCHAR(255), asset_id INT, FOREIGN KEY (asset_id) REFERENCES asset(asset_id))""")
cursor.execute("""
CREATE TRIGGER bill_trigger3
AFTER INSERT ON transaction
FOR EACH ROW
BEGIN
    INSERT INTO bill (transaction_id, timestamp, transaction_amount, transaction_type, asset_id)
    VALUES (NEW.transaction_id, NEW.timestamp, NEW.transaction_amount, NEW.transaction_type, NEW.asset_id);
END;
""")
# Insert data into the transaction table
cursor.execute("INSERT INTO transaction (transaction_id, timestamp, transaction_amount, transaction_type, asset_id) VALUES (%s, %s, %s, %s,
            (transaction_id, current_timestamp, transaction_amount, transaction_type, asset_id))
db.commit()
display_generated_bill(transaction_id, current_timestamp, transaction_amount, transaction_type, asset_id)
```

TRIGGER OUTPUT :



PROCEDURES

To fetch certificate information once it is uploaded & add its details to the certificate table.

```
def add_certificate(cert_id, cert_name, issue_date, expiry_date):
    try:
        # Call the stored procedure
        cursor = db.cursor()
        #cursor.callproc('add_certificate', (cert_id, cert_name, issue_date, expiry_date))
        #db.commit()
        cursor.execute("""

        CREATE PROCEDURE addcert1(
            IN cert_id_param INT,
            IN cert_name_param VARCHAR(255),
            IN issue_date_param DATE,
            IN expiry_date_param DATE
        )
        BEGIN
            INSERT INTO certificate (cert_id, cert_name, issue_date, expiry_date)
            VALUES (cert_id_param, cert_name_param, issue_date_param, expiry_date_param);
        END

        """)
        db.commit()
```

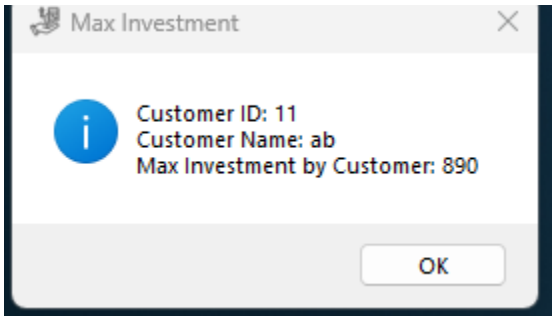
NESTED QUERIES

Feature to obtain maximum investment made till now by a specific customer. This feature can be used by AMC's to keep track of their High Net-Worth/Important Clients, who have to be given more importance. This feature also helps the company keep track of such customers, and contact them for fresh investments into new funds.

NESTED QUERY CODE:

```
# Nested query to calculate the sum of investment amounts for each customer
    query = """
        SELECT cus_id, cus_name, (
            SELECT MAX(invest_amt) FROM investment WHERE investment.cus_id = cus.cus_id
        ) AS total_investment
        FROM cus
    """
    cursor.execute(query)
    result = cursor.fetchall() # check this
```

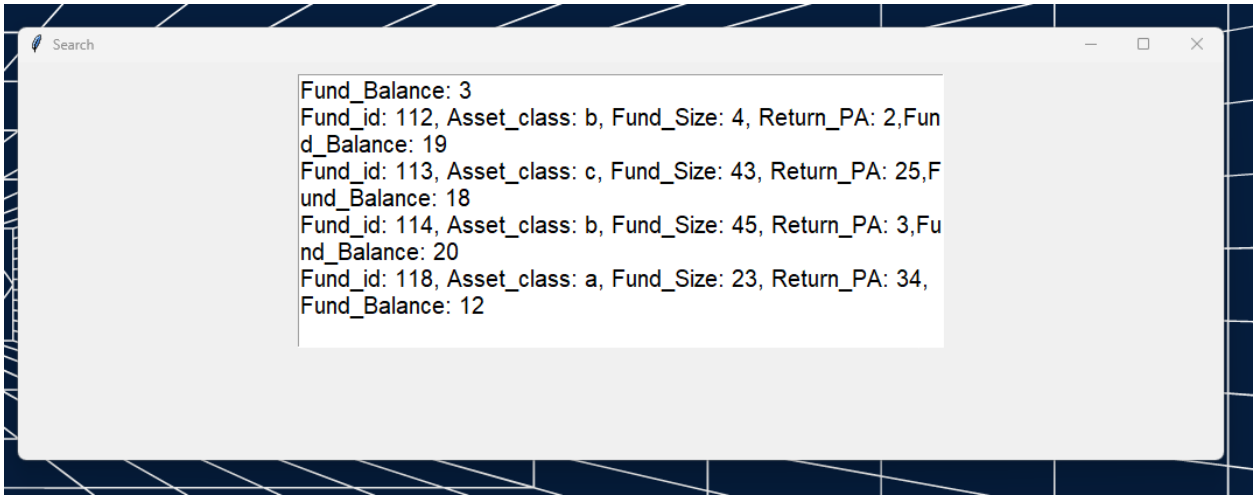
NESTED QUERY OUTPUT:



JOIN QUERIES

Feature to fetch fund information managed by a fund manager. Helpful for AMC's to keep track of which fund is managed by one or several fund managers. If a fund manager is already occupied with managing several funds, they will not be assigned new funds by the AMC.

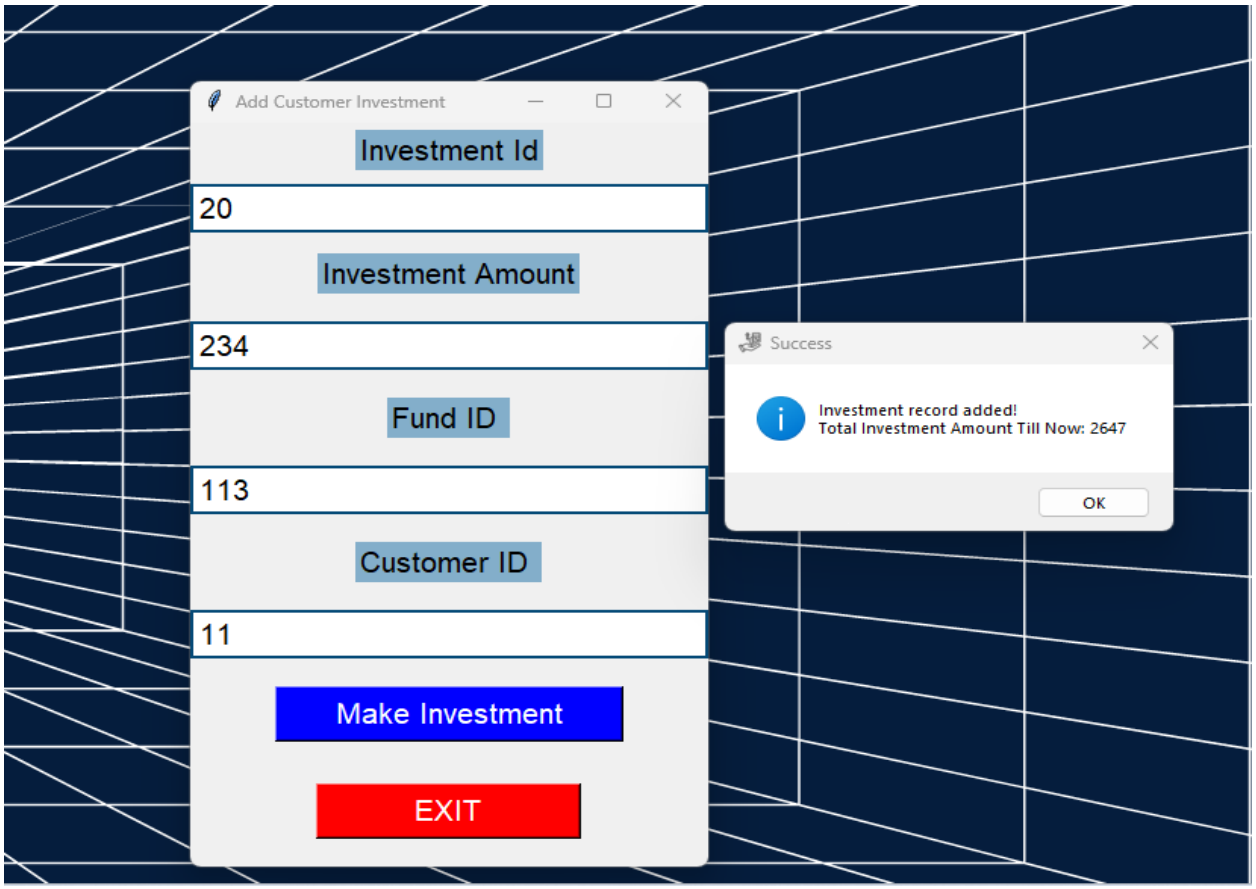
```
def see():
    query = """
    SELECT f.fund_id, f.asset_class, f.fund_size, f.return_pa, f.fund_balance
    FROM fund AS f
    JOIN fund_manager AS fm ON f.manages_id = fm.fm_id
    WHERE fm.amc_id = %s
    """
    cursor.execute(query, (current_amc_id,))
```



AGGREGATE QUERIES

Feature to get the total of all investments made by a customer. Helpful feature for AMC's to look into a client's total exposure in a certain kind of funds, which will help the client buy more/ sell existing holdings based on the global/local investment conditions.

```
sum_query = "SELECT SUM(invest_amt) FROM investment WHERE cus_id = %s"
cursor.execute(sum_query, (cus_id,))
total_investment = cursor.fetchone()[0]
messagebox.showinfo("Success", f"Investment record added!\nTotal Investment Amount Till Now: {total_investment}")
```



SQL FILE

```sql
-- Create a MySQL database connection
CREATE DATABASE IF NOT EXISTS dbms;
USE dbms;

CREATE TABLE IF NOT EXISTS amc_rec (
    amc_id INT NOT NULL PRIMARY KEY,
    amc_name VARCHAR(255),
    address VARCHAR(255),
    amc_type VARCHAR(255)
);

CREATE TABLE IF NOT EXISTS asset (
    asset_id INT NOT NULL PRIMARY KEY,
    asset_type VARCHAR(255),
    asset_name VARCHAR(255),
    no_of_shares INT,
    promoter_holding VARCHAR(255),
    valuation VARCHAR(255)
);

CREATE TABLE IF NOT EXISTS bank (
    ifsc_code VARCHAR(20),
    bank_name VARCHAR(255),
    account_no INT NOT NULL PRIMARY KEY,
    balance_amount DECIMAL(10, 2),
    amc_id INT,
    FOREIGN KEY (amc_id) REFERENCES amc_rec(amc_id)
);

CREATE TABLE IF NOT EXISTS certificate (
    cert_id INT NOT NULL PRIMARY KEY,
    cert_name VARCHAR(255),
    issue_date DATE,
```

```
    expiry_date DATE
);

CREATE TABLE IF NOT EXISTS cus (
    cus_id INT NOT NULL PRIMARY KEY,
    cus_dob DATE,
    cus_type VARCHAR(255),
    cus_name VARCHAR(255),
    cus_addr VARCHAR(255),
    cus_g VARCHAR(1),
    cus_ph CHAR(10),
    cus_age INT,
    amc_id INT,
    FOREIGN KEY (amc_id) REFERENCES amc_rec(amc_id)
);

CREATE TABLE IF NOT EXISTS fund_manager (
    fm_id INT NOT NULL PRIMARY KEY,
    amc_id INT,
    FOREIGN KEY (amc_id) REFERENCES amc_rec(amc_id)
);

CREATE TABLE IF NOT EXISTS fund (
    fund_id INT NOT NULL PRIMARY KEY,
    asset_class VARCHAR(255),
    fund_size INT,
    return_pa INT,
    fund_balance INT,
    manages_id INT,
    FOREIGN KEY (manages_id) REFERENCES fund_manager(fm_id)
);

CREATE TABLE IF NOT EXISTS investment (
    inv_id INT NOT NULL PRIMARY KEY,
    invest_amt INT,
    fund_id INT,
```

```
    cus_id INT,
    invest_date DATE,
    FOREIGN KEY (fund_id) REFERENCES fund(fund_id),
    FOREIGN KEY (cus_id) REFERENCES cus(cus_id)
);

CREATE TABLE IF NOT EXISTS transaction (
    transaction_id INT NOT NULL PRIMARY KEY,
    timestamp DATETIME,
    transaction_amount INT,
    transaction_type VARCHAR(255),
    asset_id INT,
    FOREIGN KEY (asset_id) REFERENCES asset(asset_id)
);

-- Create a trigger to update fund balance after each investment
DELIMITER //
CREATE TRIGGER update_fund_balance AFTER INSERT ON investment
FOR EACH ROW
BEGIN
    UPDATE fund
    SET fund_balance = fund_balance - NEW.invest_amt
    WHERE fund_id = NEW.fund_id;
END;
//
DELIMITER ;

-- Create a trigger to log transactions
DELIMITER //
CREATE TRIGGER log_transaction AFTER INSERT ON transaction
FOR EACH ROW
BEGIN
    INSERT INTO transaction_log (transaction_id, timestamp, transaction_type,
asset_id)
    VALUES (NEW.transaction_id, NEW.timestamp, NEW.transaction_type, NEW.asset_id);
END;
```

```
//
DELIMITER ;



SELECT
    cus_id,
    cus_name,
    (
        SELECT MAX(invest_amt) FROM investment WHERE investment.cus_id = cus.cus_id
    ) AS total_investment
FROM cus;

-- Stored procedure to add a certificate to the database
DELIMITER //
CREATE PROCEDURE addcert(
    IN cert_id_param INT,
    IN cert_name_param VARCHAR(255),
    IN issue_date_param DATE,
    IN expiry_date_param DATE
)
BEGIN
    INSERT INTO certificate (cert_id, cert_name, issue_date, expiry_date)
    VALUES (cert_id_param, cert_name_param, issue_date_param, expiry_date_param);
END;
//
DELIMITER ;

-- Query to retrieve fund information for a specific asset management company (AMC)
SELECT
    f.fund_id,
    f.asset_class,
    f.fund_size,
    f.return_pa,
    f.fund_balance
FROM fund AS f
JOIN fund_manager AS fm ON f.manages_id = fm.fm_id
```

```
WHERE fm.amc_id = %s;

-- Query to insert investment details into the investment table
INSERT INTO investment (inv_id, invest_date, invest_amt, fund_id, cus_id)
VALUES (%s, %s, %s, %s, %s);

-- Query to calculate the total investment amount for a specific customer
SELECT SUM(invest_amt) FROM investment WHERE cus_id = %s;

-- Query to create a user with specified privileges
CREATE USER '{username}'@'localhost' IDENTIFIED BY '{password}';

-- Query to grant privileges to a user
GRANT {privileges} ON your_database.* TO '{username}'@'localhost';

-- Query to flush privileges to apply changes
FLUSH PRIVILEGES;

-- Query to create a new fund manager and insert it into the fund_manager table
INSERT INTO fund_manager (fm_id, amc_id) VALUES (%s, %s);

-- Query to insert fund details into the fund table
INSERT INTO fund (fund_id, asset_class, fund_size, return_pa, fund_balance,
manages_id)
VALUES (%s, %s, %s, %s, %s, %s);

-- Inserting a new customer record into the 'cus' table
INSERT INTO cus (cus_id, cus_dob, cus_type, cus_name, cus_addr, cus_g, cus_ph,
cus_age, amc_id)
VALUES (%s, %s, %s, %s, %s, %s, %s, %s, %s);

-- Checking if a customer record already exists
SELECT * FROM cus WHERE cus_id = %s;
-- Searching for a bank account by account number in the 'bank' table
SELECT * FROM bank WHERE account_no = %s;
```

```
-- Retrieving all bank accounts associated with a specific AMC from the 'bank'
table
SELECT * FROM bank WHERE amc_id = %s;

-- Inserting a new bank account record into the 'bank' table
INSERT INTO bank (ifsc_code, bank_name, account_no, balance_amount, amc_id)
VALUES (%s, %s, %s, %s, %s);



-- Change user password
SET PASSWORD FOR '{username}'@'localhost' = PASSWORD('{new_password}');

-- Update customer address
UPDATE cus SET cus_addr = %s WHERE cus_id = %s;

-- Delete a customer account
DELETE FROM cus WHERE cus_id = %s;

-- Drop a user
DROP USER '{username}'@'localhost';

-- Drop a trigger
DROP TRIGGER IF EXISTS update_fund_balance;
DROP TRIGGER IF EXISTS log_transaction;
-- Drop a stored procedure
DROP PROCEDURE IF EXISTS addcert;
-- Drop a table
DROP TABLE IF EXISTS transaction_log;
```

GitHub Repository

GitHub Repository Link:

https://github.com/gaganhr94/WealthWise