**PROJECT REPORT**

**Submitted for :**

**DATABASE MANAGEMENT SYSTEM (UCS310)**


| SUBMITTED BY | ROLL NUMBER |
|---|---|
| VASHISTHA | 102203447 |
| SAMARTH SINGH | 102203487 |
| GAUTAM | 102203498 |
| VIDHI GOYAL | 102203659 |


# TOPIC: VIRTUAL TRADING PLATFORM

**Submitted to-**
**Mr. Shashank Singh**

THAPAR INSTITUTE
OF ENGINEERING & TECHNOLOGY
(Deemed to be University)


**Computer Science and Engineering DepartmentTIET, Patiala**

**(JAN-MAY, 2024)**

# ABSTRACT

## Development and Implementation of a Virtual Trading Platform using MySQL and Python

**The abstract of this report encapsulates the development and implementation of a virtual trading platform leveraging MySQL and Python. In an increasingly digitalized world, virtual trading platforms offer individuals a risk-free environment to hone their investment skills, test strategies, and explore financial markets. This report outlines the architectural design, functionality, and implementation details of the virtual trading platform.**

**The platform's foundation rests upon the integration of MySQL, a robust relational database management system, and Python, a versatile programming language renowned for its simplicity and efficiency. MySQL serves as the backbone for storing and managing crucial data such as user information, stock market data, transaction history, and portfolio details.**

**Throughout the development process, emphasis was placed on scalability, security, and user experience. MySQL's scalability features ensure the platform can accommodate a growing user base and increasing volumes of data. Security measures, including encryption protocols and secure authentication mechanisms, safeguard users' sensitive information and transactions. User experience enhancements, such as intuitive navigation, responsive design, and real-time updates, contribute to a seamless and engaging trading environment.**

# INDEX

# INTRODUCTION

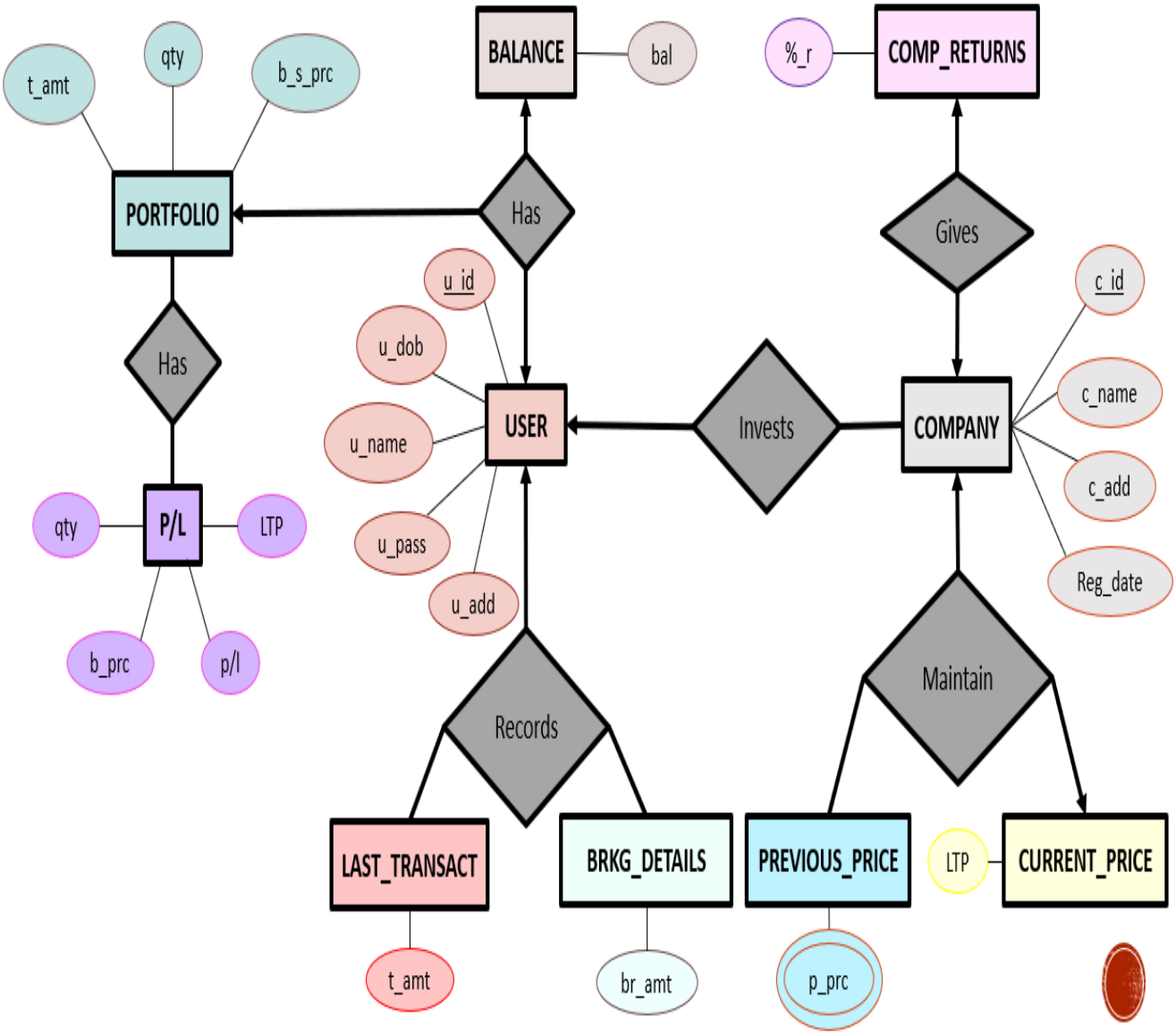
**A trading platform is a software system used to trade securities. It allows investors to open, close, and manage market positions online through a financial intermediary, such as an online broker.**

**Online trading platforms are frequently offered by brokers either for free or at a discount in exchange for maintaining a funded account and/or making a specified number of trades per month. The best trading platforms offer a mix of robust features and low fees.**

**Online Trading is a method that facilitates buying and selling of financial instruments such as mutual funds, equities, bonds, Sovereign gold bonds, derivatives, stocks, ETFs and commodities through an electronic interface. Online Trading has simplified a complex process into a few clicks.**

# ER Diagram

**An Entity-Relationship (ER) diagram is a visual representation of the data model that describes how entities are related to each other within a system. It's a conceptual modeling technique used in software engineering to design databases in a graphical format.**



# ER TO TABLES

| USER | | | | |
|---|---|---|---|---|
| u_id | u_name | u_dob | u_add | u_pass |
| | | | | |

| BALANCE | |
|---|---|
| u_id | bal |
| | |

| BRKG_DETAILS | |
|---|---|
| br_amt | u_id |
| | |

| LAST_TRANSACT | |
|---|---|
| lt_amt | u_id |
| | |

| COMPANY | | | |
|---|---|---|---|
| c_id | c_name | c_add | Reg_date |
| | | | |

| PREVIOUS_PRICE | |
|---|---|
| p_prc | c_id |
| | |

| P_prc | |
|---|---|
| p_prc | c_id |
| | |

| COMP_RETURNS | |
|---|---|
| %_r | c_id |
| | |

| CURRENT_PRICE | |
|---|---|
| ltp | c_id |
| | |

| COMPANY-INVESTS | | | | |
|---|---|---|---|---|
| c_id | c_name | c_add | Reg_date | u_id |
| | | | | |

| PORTFOLIO | | | | |
|---|---|---|---|---|
| u_id | t_amt | c_id | qty | b_s_prc |
| | | | | |

| P/L | | | | | |
|---|---|---|---|---|---|
| u_id | pyl | c_id | qty | B_prc | ltp |
| | | | | | |

# NORMALIZATION

**Normalization is a database design technique used to organize tables and their relationships in a way that reduces redundancy and dependency. In this code, normalization principles are applied in several places:**

1. **User Information (user_info) Table:**

**The user_info table is in the third normal form. There are no transitive dependencies present. Each non-key attribute (columns other than the primary key) is dependent only on the primary key U_id.**

2. **Company Information (company_info) Table:**

**Similarly, the company_info table is in the third normal form. All non-key attributes are directly dependent on the primary key C_id.**

3. **Portfolio Table:**

**The portfolio table, which maintains users' stock holdings, appears to be in 3NF. The attributes such as B_S_price, Qty, and Total_amt are directly related to the primary key (C_id, U_id), and there are no transitive dependencies.**

**Normalization to 3NF ensures that there are no non-prime attributes dependent on other non-prime attributes, i.e., all attributes are functionally dependent only on the primary key. This reduces data redundancy and helps maintain data integrity.**

# PROJECT CODE

```
import mysql.connector
import random
import matplotlib.pyplot as plt
p_count=0

mydb = mysql.connector.connect(
  host="localhost",
  user="root",
  password="GAUTAM",
  database="TRADING_PLATFORM"    )
mycursor = mydb.cursor()


price_list=[]

mycursor.execute("CREATE DATABASE TRADING_PLATFORM")
mycursor.execute("CREATE TABLE user_info (U_name VARCHAR(255),U_id
int primary key, U_address VARCHAR(255),U_dob DATE,U_pass
VARCHAR(255))")
mycursor.execute("CREATE TABLE Brkg_details (U_id int, Br_amt int ,
FOREIGN KEY (U_id) REFERENCES user_info(U_id))")
mycursor.execute("CREATE TABLE Comp_returns (C_id int, Prec_return
int , FOREIGN KEY (C_id) REFERENCES company_info(C_id))")
mycursor.execute("CREATE TABLE PL (U_id int,C_id int , Qty int , P_L
int , LTP_buy_price int , Sell_Price int)")
mycursor.execute("CREATE TABLE Last_Transact (U_id int ,T_amt int ,
FOREIGN KEY (U_id) REFERENCES user_info(U_id))")
mycursor.execute("CREATE TABLE Prev_Price (C_id int, Pre_Price int ,
FOREIGN KEY (C_id) REFERENCES company_info(C_id))")
mycursor.execute("CREATE TABLE company_info (C_name
VARCHAR(255),C_id int primary key , C_address VARCHAR(255),Reg_date
DATE)")
mycursor.execute("CREATE TABLE Balance (U_id int, Balance int ,
FOREIGN KEY (U_id) REFERENCES user_info(U_id))")
mycursor.execute("CREATE TABLE Stock_price_current (C_id int, LTP
int , FOREIGN KEY (C_id) REFERENCES company_info(C_id))")
mycursor.execute("CREATE TABLE portfolio (C_id int,U_id int ,
B_S_price int ,Qty int,Total_amt int, FOREIGN KEY (C_id) REFERENCES
company_info(C_id),FOREIGN KEY (U_id) REFERENCES
user_info(U_id))")
```

```
def add_user():
    print("Welcome to the PLATFORM")
    uname = input("Enter the User Name: ")
    uid = int(input("Enter the User Id: "))
    udob = input("Enter the Dob: ")
    uadd = input("Enter the address: ")
    upass = input("Enter the password: ")
    sql = "INSERT INTO user_info (U_name, U_id, U_address, U_dob,
U_pass) VALUES (%s, %s, %s, %s, %s)"
    val = (uname, uid, uadd, udob, upass)
    mycursor.execute(sql, val)
    mydb.commit()
    print("User added successfully!")
    print()
    sql = "INSERT INTO Balance (U_id,Balance) VALUES (%s, %s)"
    val = (uid,0)
    mycursor.execute(sql, val)
    mydb.commit()



def mod_user():
    uid=int(input("Enter the uid to modify : "))
    uname = input("Enter the New User Name: ")
    udob = input("Enter the New Dob: ")
    uadd = input("Enter the New address: ")
    upass = input("Enter the New password: ")
    sql="update user_info set U_name=%s
,U_dob=%s,U_address=%s,U_pass=%s where U_id=%s"
    val=(uname,udob,uadd,upass,uid)
    mycursor.execute(sql, val)
    mydb.commit()
    print("User modified successfully!")
    print()



def add_comapny():
```

```
    print("Welcome to the PLATFORM")
    cname = input("Enter the Comapny Name: ")
    cid = int(input("Enter the Comapny Id: "))
    cdate = input("Enter the Reg Date : ")
    cadd = input("Enter the Company address: ")
    price=int(input("Enter the listed(initial) price : "))

    sql = "INSERT INTO company_info (C_name, C_id, C_address,
Reg_date) VALUES (%s, %s, %s, %s)"
    val = (cname, cid, cadd, cdate)
    mycursor.execute(sql, val)
    mydb.commit()
    print("Company added successfully!")
    print()


    sql = "INSERT INTO Stock_price_current (C_id,LTP) VALUES (%s,
%s)"
    val = (cid,price)
    mycursor.execute(sql, val)
    mydb.commit()




def mod_company():
    cid=int(input("Enter the cid to modify : "))
    cname = input("Enter the New Company Name: ")
    cdate = input("Enter the New Reg date: ")
    cadd = input("Enter the New address: ")
    sql="update company_info set C_name=%s ,Reg_date=%s,C_address=%s
where C_id=%s"
    val=(cname,cdate,cadd,cid)
    mycursor.execute(sql, val)
    mydb.commit()
    print("Company modified successfully!")
    print()
```

```
def add_balance():
    uid=input("Enter the User id to add money : ")
    amt=int(input("Enter the amount to be added : "))
    sql="select Balance from Balance where U_id=%s"
    mycursor.execute(sql, (uid,))
    prev=mycursor.fetchone()
    prev_bal=int(prev[0])
    new_bal=prev_bal+amt
    sql="update Balance set Balance=%s where U_id=%s"
    val=(new_bal,uid)
    mycursor.execute(sql, val)
    mydb.commit()
    print("Money Added successfully!")
    print()



    sql = "INSERT INTO last_transact (U_id,T_amt) VALUES (%s, %s)"
    val = (uid, amt)
    mycursor.execute(sql, val)
    mydb.commit()




def withdraw_bal():


    uid=input("Enter the User id  : ")
    amt=int(input("Enter the amount to withdraw : "))

    sql="select Balance from Balance where U_id=%s"
    mycursor.execute(sql, (uid,))
    cur=mycursor.fetchone()
    cur_bal=int(cur[0])
    if cur_bal > amt:
        new_bal=cur_bal-amt
        sql="update Balance set Balance=%s where U_id=%s"
```

```
        val=(new_bal,uid)
        mycursor.execute(sql, val)
        mydb.commit()
        print("Money Withdrawed successfully!")



        sql = "INSERT INTO last_transact (U_id,T_amt) VALUES (%s,
%s)"
        val = (uid, -amt)
        mycursor.execute(sql,val)
        mydb.commit()




    else:
        print("Insufficient Money in account ")



def update_ltp():
    cid=int(input("enter the Cid : "))
    price_move = random.randint(1, 15) * 5
    price_trend=random.randint(0,1)
    arr=["+","-"]
    char=arr[price_trend]
    sql="select LTP from Stock_price_current where C_id=%s"
    mycursor.execute(sql, (cid,))
    cur=mycursor.fetchone()

    prev_price=int(cur[0])

    if(char == '+'):
        new_p=prev_price+price_move

    else:
        new_p=prev_price-price_move


    sql="update Stock_price_current set LTP=%s where C_id=%s"
    val=(new_p,cid)
```

```
    mycursor.execute(sql, val)
    mydb.commit()
    price_list.append(new_p)
    global p_count;
    p_count=p_count+1


    sql="select C_name from company_info where C_id=%s"
    mycursor.execute(sql, (cid,))
    cur=mycursor.fetchone()
    print("Current stock price of company :",cur[0], " : ",new_p)







    sql = "INSERT INTO prev_price (C_id,Pre_Price) VALUES (%s,
%s)"
    val = ( cid,new_p)
    mycursor.execute(sql, val)
    mydb.commit()



def Chart():
    update_ltp()
    update_ltp()
    update_ltp()
    update_ltp()
    update_ltp()



    y=price_list
    x=[]
    for i in range(0,p_count,1):
        x.append(i*5)
```

```
    plt.plot(x, y)
    plt.xlabel('TIME')
    plt.ylabel('PRICE')
    plt.title('PRICE CHART')
    plt.show()




def buy_share():

    uid=int(input("Enter the user id :"))
    cid=input("Enter the company id : ")
    qty=int(input("Enter the qty :"))
    sql="select LTP  from Stock_price_current where C_id=%s"
    mycursor.execute(sql, (cid,))
    cur=mycursor.fetchone()
    bs_price=cur[0]
    f_amt=bs_price*qty

    sql="select Balance from Balance where U_id=%s"
    mycursor.execute(sql, (uid,))
    cur=mycursor.fetchone()
    ava_val=int(cur[0])


    print("Brokerage : 5 %")
    bro_amt=0.05*f_amt


    if(ava_val > f_amt+bro_amt):
        sql = "INSERT INTO portfolio
(C_id,U_id,B_S_price,Qty,Total_amt) VALUES (%s, %s, %s,
%s,%s)"
        val = ( cid,uid,bs_price,qty,f_amt)
        mycursor.execute(sql, val)
        mydb.commit()
        print("Share purchased successfully!")
        print("")

        sql="update Balance set Balance=%s where U_id=%s"
```

```
        r_bal=ava_val-(f_amt+bro_amt)
        val=(r_bal,uid)
        mycursor.execute(sql, val)
        mydb.commit()
        print("Balance updated successfully!")
        print("")



        sql = "INSERT INTO Brkg_details (U_id,Br_amt) VALUES (%s,
%s)"
        val = ( uid,bro_amt)
        mycursor.execute(sql, val)
        mydb.commit()
        print("Brokerage Received")


        pl_amt=0
        sp=0
        sql = "INSERT INTO pl
(U_id,C_id,Qty,P_L,LTP_buy_price,Sell_price) VALUES (%s,%s, %s, %s,
%s,%s)"
        val = ( uid,cid,qty,pl_amt,bs_price,sp)
        mycursor.execute(sql, val)
        mydb.commit()



        sql = "INSERT INTO last_transact (U_id,T_amt) VALUES (%s,
%s)"
        val = (uid, -(f_amt+bro_amt))
        mycursor.execute(sql, val)
        mydb.commit()

    else :
        print("Insufficient Money")
```

```
def sell_share():
    uid=int(input("Enter the user id :"))
    cid=input("Enter the company id : ")
    qty=int(input("Enter the qty to sell :"))


    sql="select Qty from portfolio where U_id=%s and C_id=%s "
    val=(uid,cid)
    mycursor.execute(sql,
val)
    cur=mycursor.fetchone()
    ava_qty=int(cur[0])




    print("Brokerage : 2.5 %")


    update_ltp()
    sql="select LTP from Stock_price_current where C_id=%s"
    mycursor.execute(sql, (cid,))
    cur=mycursor.fetchone()
    c_price=int(cur[0])




    if(ava_qty >= qty):

        amt_with=c_price*qty
        bro_amt=0.025*amt_with
        sql="select Balance from Balance where U_id=%s "
        value=(uid,)
        mycursor.execute(sql, value)
        cur=mycursor.fetchone()
        ava_val=int(cur[0])
        sql="update Balance set Balance=%s where U_id=%s"
        r_bal=ava_val+amt_with-bro_amt
```

```
        val=(r_bal,uid)


        mycursor.execute(sql, val)
        mydb.commit()


        print("Balance updated successfully!")
        print("")



        sql="update portfolio set  Qty=%s where U_id=%s and C_id=%s
"
        new_qty=ava_qty-qty
        val=(new_qty,uid,cid)
        mycursor.execute(sql, val)
        print("Qty updated successfully !!")
        mydb.commit()



        sql="select  LTP_buy_price from pl where U_id=%s and
C_id=%s"
        value=(uid,cid)
        mycursor.execute(sql, value)
        cur=mycursor.fetchone()
        bp_amt=int(cur[0])

        plamt=(c_price-bp_amt)*qty




        sql="update pl set sell_price=%s, p_l=%s,qty=%s where
U_id=%s and C_id=%s "
        val=(c_price,plamt,new_qty,uid,cid)
        mycursor.execute(sql, val)
        print("P/L Statement updated successfully !!")
        mydb.commit()
```

```
        sql = "INSERT INTO last_transact (U_id,T_amt) VALUES (%s,
%s)"
        val = (uid, (amt_with-bro_amt))
        mycursor.execute(sql, val)
        mydb.commit()



    else:
        print("Insufficient Shares")




def Comapny_return():
    cid=int(input("Enter the company id : "))



    sql="select  LTP_buy_price from pl where C_id=%s"
    value=(cid,)
    mycursor.execute(sql, value)
    cur=mycursor.fetchone()
    buy_amt=int(cur[0])



    sql="select  Sell_Price from pl where C_id=%s"
    value=(cid,)
    mycursor.execute(sql, value)
    cur=mycursor.fetchone()
    sell_amt=int(cur[0])


    per=((sell_amt-buy_amt)/buy_amt)*100



    sql = "INSERT INTO comp_returns (C_id,Prec_return) VALUES (%s,
%s)"
    val = (cid,per)
    mycursor.execute(sql, val)
```

```
    mydb.commit()
    print("Percentage return is : ", per , "%")



def Display_Previous_price():


    cid=int(input("Enter the cid : "))



    sql="select  C_name from company_info where C_id=%s "
    value=(cid,)
    mycursor.execute(sql, value)
    cur=mycursor.fetchone()
    print("Company :  ",cur[0])




    print("Company id : ",cid )
    sql="select  Pre_Price from prev_price where C_id=%s"
    value=(cid,)
    mycursor.execute(sql, value)
    cur=mycursor.fetchall()
    for i in cur:
        print("Price ",i[0])



def login():
    print("Enter the following details for login : ")
    ui=int(input("Enter the use id :"))
    pas=input("Enter the pass :")
    sql="select U_pass from user_info where U_id=%s"
    value=(ui,)
    mycursor.execute(sql, value)
    oe_pass=mycursor.fetchone()

    l_pass=oe_pass[0]
```

```
    if(l_pass == pas):
        print("Logged in successfully")
        return 1


    else:
        print("INVALID DETAILS")
        return 0



print(" $$ WELCOME TO THE VIRTUAL TRADING PLATFORM  $$ ")




print("Making the DEMO USER : ")
add_user()


a=login()

while(a):
    print("Enter the option to proceed with the platform :- ")


    print("Option 1 : TO ADD NEW USER" )
    print("")



    print("Option 2 : TO MODIFY USER" )
    print("")



    print("Option 3 : TO ADD NEW COMPANY" )
    print("")




    print("Option 4 : TO MODIFY COMPANY" )
    print("")
```

```
print("Option 5 : TO ADD BALANCE" )
print("")



print("Option 6 : TO WITHDRAW BALANCE" )
print("")



print("Option 7 : TO SHOW LTP OF A STOCK" )
print("")



print("Option 8 : TO DISPLAY PRICE CHART" )
print("")



print("Option 9 : TO BUY SHARE" )
print("")



print("Option 10 : TO SELL SHARE" )
print("")
```

```
    print("Option 11 : TO DISPLAY COMPANY RETURNS" )
    print("")




    print("Option 12 : TO DISPLAY PREVIOUS PRICE OF A STOCK" )
    print("")




    print("Option 13 : TO LOGOUT" )
    print("")


    op=int(input("Enter Desired option : "))


    match op:

        case 1:
            add_user()



        case 2:
            mod_user()

        case 3:

            add_comapny()

        case 4 :
            mod_company()


        case 5 :
            add_balance()
```

```
        case 6 :
            withdraw_bal()


        case 7:
            update_ltp()


        case 8:
            Chart()



        case 9:

            buy_share()


        case 10 :
            sell_share()


        case 11:
            Comapny_return()


        case 12:
            Display_Previous_price()









        case 13 :
            a=0
            print("Logged out successfully")



        case _ :
            print("Invalid option selected :")
```

# OUTPUT (VS CODE)

## 1) ADD USER

```
Welcome to the PLATFORM
Enter the User Name: GAUTAM
Enter the User Id: 1
Enter the Dob: 2002-10-16
Enter the address: SGNR
Enter the password: GD@123
User added successfully!
```

```
Enter the following details for login :
Enter the use id :1
Enter the pass :GD@123
Logged in successfully
Enter the option to proceed with the platform :-
Option 1 : TO ADD NEW USER

Option 2 : TO MODIFY USER

Option 3 : TO ADD NEW COMPANY

Option 4 : TO MODIFY COMPANY

Option 5 : TO ADD BALANCE

Option 6 : TO WITHDRAW BALANCE

Option 7 : TO SHOW LTP OF A STOCK

Option 8 : TO DISPLAY PRICE CHART

Option 9 : TO BUY SHARE

Option 10 : TO SELL SHARE

Option 11 : TO DISPLAY COMPANY RETURNS

Option 12 : TO DISPLAY PREVIOUS PRICE OF A STOCK

Option 13 : TO LOGOUT

Enter Desired option :
```

```
Enter Desired option : 1
Welcome to the PLATFORM
Enter the User Name: VASHISTHA
Enter the User Id: 2
Enter the Dob: 2003-07-21
Enter the address: BARMER
Enter the password: VC@123
User added successfully!
```

**2) MODIFY USER**

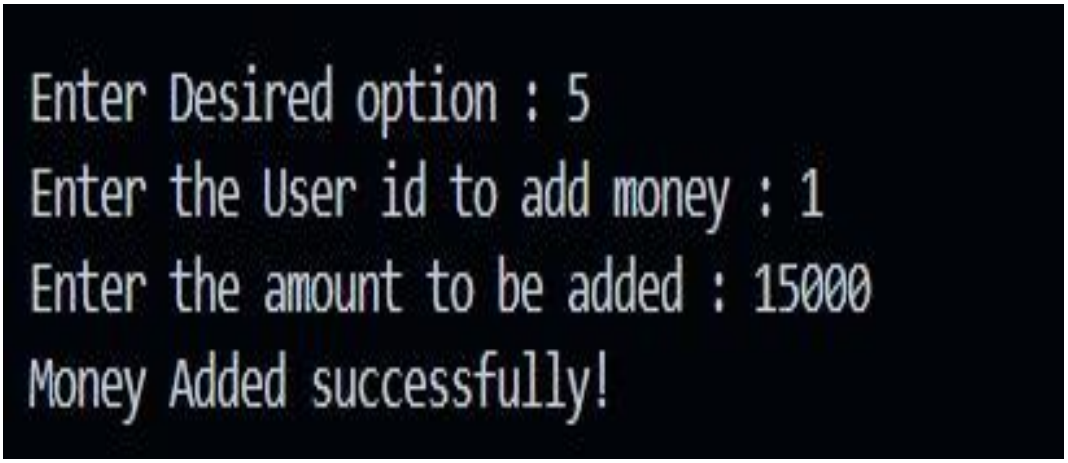
```
Enter Desired option : 2
Enter the uid to modify : 2
Enter the New User Name: VIDHI
Enter the New Dob: 2004-10-18
Enter the New address: SGNR
Enter the New password: VG@123
User modified successfully!
```

**3) ADD COMPANY**

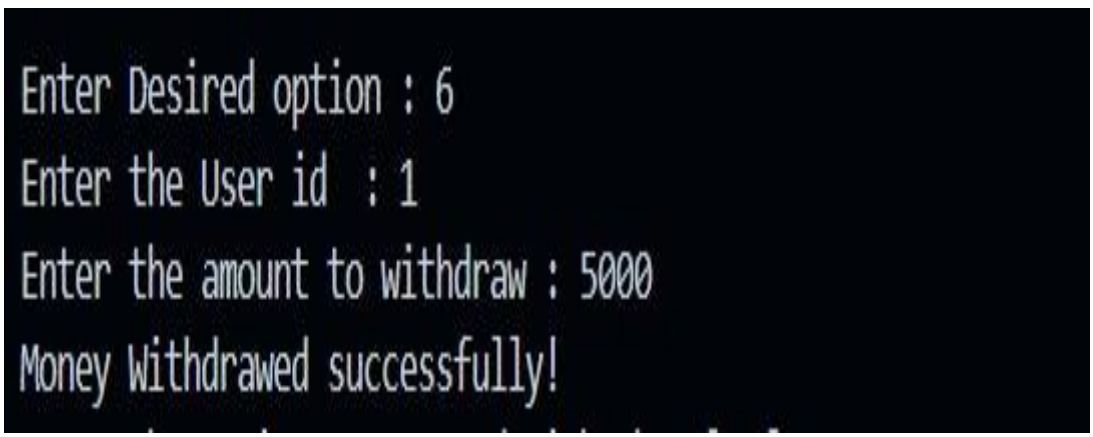
```
Enter Desired option : 3
Welcome to the PLATFORM
Enter the Comapny Name: HDFC BANK
Enter the Comapny Id: 100
Enter the Reg Date : 2000-10-10
Enter the Company address: DELHI
Enter the listed(initial) price : 2500
Company added successfully!
```

**4) MODIFY COMPANY**


**5) ADD BALANCE**



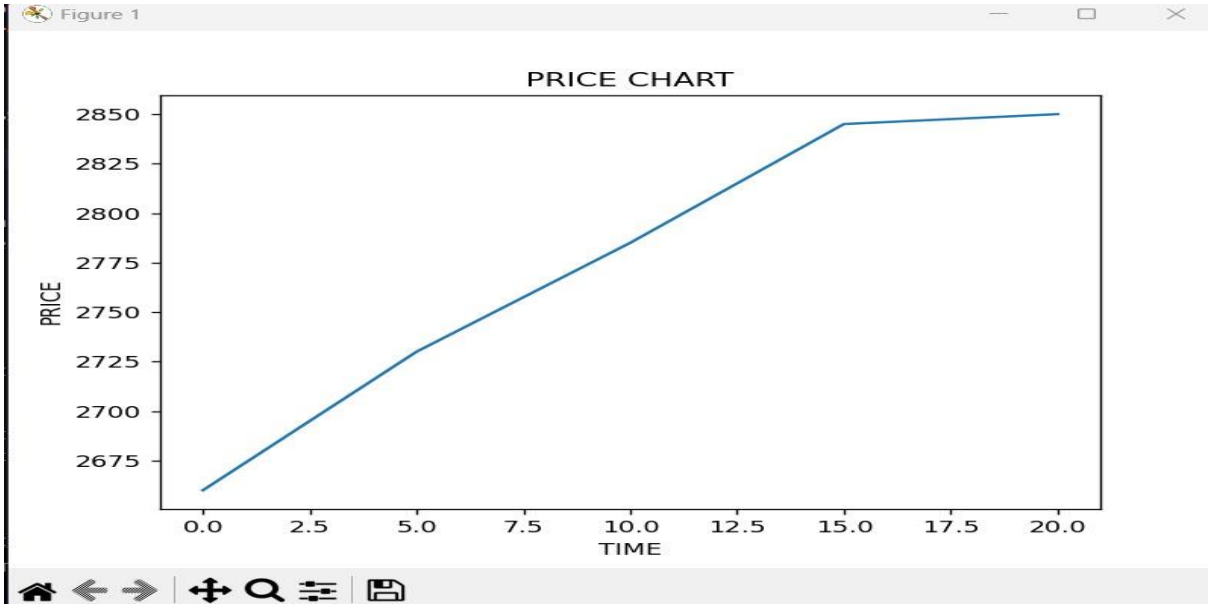

**6) WITHDRAW BALANCE**



## 7) DISPLAY LTP

```
Enter Desired option : 7
enter the Cid : 100
Current stock price of company : HDFC PVT BANK  :  2555
```

## 8) CHART




```
Enter Desired option : 8
enter the Cid : 100
Current stock price of company : HDFC PVT BANK  :  2660
enter the Cid : 100
Current stock price of company : HDFC PVT BANK  :  2730
enter the Cid : 100
Current stock price of company : HDFC PVT BANK  :  2785
enter the Cid : 100
Current stock price of company : HDFC PVT BANK  :  2845
enter the Cid : 100
Current stock price of company : HDFC PVT BANK  :  2850
```

## 9) BUY SHARE

```
Enter Desired option : 9
Enter the user id :1
Enter the company id : 100
Enter the qty :2
Brokerage : 5 %
Share purchased successfully!

Balance updated successfully!

Brokerage Received
```

## 10) SELL SHARE

```
Enter Desired option : 10
Enter the user id :1
Enter the company id : 100
Enter the qty to sell :1
Brokerage : 2.5 %
enter the Cid : 100
Current stock price of company : HDFC PVT BANK  :  2835
Balance updated successfully!

Qty updated successfully !!
P/L Statement updated successfully !!
```

## 11) COMPANY RETURN

```
Enter Desired option : 11
Enter the company id : 100
Percentage return is :  -1.9298245614035088 %
```

## 12) DISPLAY PREVIOUS PRICES

```
Enter Desired option : 12
Enter the cid : 100
Company :   HDFC PVT BANK
Company id :  100
Price  2555
Price  2590
Price  2660
Price  2610
Price  2625
Price  2630
Price  2725
Price  2795
Price  2855
Price  2780
Price  2710
Price  2660
Price  2730
Price  2785
Price  2845
Price  2850
Price  2835
Price  2795
```

## 13) LOGOUT

```
Enter Desired option : 13
Logged out successfully
PS C:\Users\gauta>
```

# OUTPUT (DATABASE)

```
mysql> select * from user_info;
+-----------+-----+-----------+-----------+--------+
| U_name    | U_id | U_address | U_dob     | U_pass |
+-----------+-----+-----------+-----------+--------+
| GAUTAM    |    1 | SGNR      | 2002-10-16 | GD@123 |
| VIDHI     |    2 | SGNR      | 2004-10-18 | VG@123 |
| VASHISTHA |    3 | BMR       | 2002-04-09 | VC@123 |
| SAMARTH   |    4 | JK        | 2001-06-04 | SS@123 |
+-----------+-----+-----------+-----------+--------+
4 rows in set (0.00 sec)
```

```
mysql> select * from company_info;
+--------------+-----+-----------+-----------+
| C_name       | C_id | C_address | Reg_date  |
+--------------+-----+-----------+-----------+
| HDFC PVT BANK |  100 | NEW DELHI | 2001-05-06 |
+--------------+-----+-----------+-----------+
1 row in set (0.00 sec)
```

```
mysql> select * from portfolio;
+------+------+-----------+------+-----------+
| C_id | U_id | B_S_price | Qty  | Total_amt |
+------+------+-----------+------+-----------+
|  100 |    1 |      2850 |    0 |      5700 |
+------+------+-----------+------+-----------+
1 row in set (0.00 sec)
```

```
mysql> select * from pl
    -> ;
+------+------+------+------+--------------+------------+
| U_id | C_id | Qty  | P_L  | LTP_buy_price | Sell_Price |
+------+------+------+------+--------------+------------+
|    1 |  100 |    0 |  -55 |         2850 |       2795 |
+------+------+------+------+--------------+------------+
1 row in set (0.01 sec)
```

```
mysql> select * from last_transact;
+------+-------+
| U_id | T_amt |
+------+-------+
|    1 | 15000 |
|    1 | -5000 |
|    1 | -5985 |
|    1 |  2764 |
|    1 |  2725 |
+------+-------+
5 rows in set (0.01 sec)
```

```
mysql> select * from comp_returns;
+------+-------------+
| C_id | Prec_return |
+------+-------------+
|  100 |          -2 |
|  100 |          -2 |
+------+-------------+
2 rows in set (0.00 sec)
```

```
mysql> select * from brkg_details;
+------+--------+
| U_id | Br_amt |
+------+--------+
|    1 |    285 |
+------+--------+
1 row in set (0.00 sec)
```

```
mysql> select * from balance;
+------+---------+
| U_id | Balance |
+------+---------+
|    1 |    9504 |
|    2 |       0 |
|    3 |       0 |
|    4 |       0 |
+------+---------+
4 rows in set (0.00 sec)
```

```
mysql> select * from prev_price;
+------+-----------+
| C_id | Pre_Price |
+------+-----------+
|  100 |      2555 |
|  100 |      2590 |
|  100 |      2660 |
|  100 |      2610 |
|  100 |      2625 |
|  100 |      2630 |
|  100 |      2725 |
|  100 |      2795 |
|  100 |      2855 |
|  100 |      2780 |
|  100 |      2710 |
|  100 |      2660 |
|  100 |      2730 |
|  100 |      2785 |
|  100 |      2845 |
|  100 |      2850 |
|  100 |      2835 |
|  100 |      2795 |
+------+-----------+
18 rows in set (0.00 sec)
```

```
mysql> select * from stock_price_current;
+------+------+
| C_id | LTP  |
+------+------+
|  100 | 2795 |
+------+------+
1 row in set (0.00 sec)
```

# CONCLUSION

**In conclusion, the development and implementation of this virtual trading platform underscore the fusion of MySQL and Python as powerful tools for creating innovative financial applications. By providing users with a realistic yet risk-free avenue for exploring financial markets, the platform aims to democratize access to investment education and empower individuals to make informed financial decisions.**