

Bank Of E: A Banking System based on Python and MySQL

**Mehul Arora, XII-SA2 Roll no. 25**

Modern Vidya Niketan Sr. Sec. School

**Certificate**

This is to certify that the project entitled **“Bank Of E: A Banking System based on Python and MySQL”**, which deals with creating a system for making accounts in a bank and provides for withdrawal, deposit and transaction functions in a bank has been submitted by Mehul Arora. This project is bonafide piece of work carried out with the consultation of the supervisor.

Mehul Arora

XII-SA2

Roll no. 25

**Acknowledgement**

The satisfaction and euphoria that accompanies the successful completion of any task would be incomplete without mentioning the name of the people whose constant guidance and ancouragement has crowed all my efforts with success.

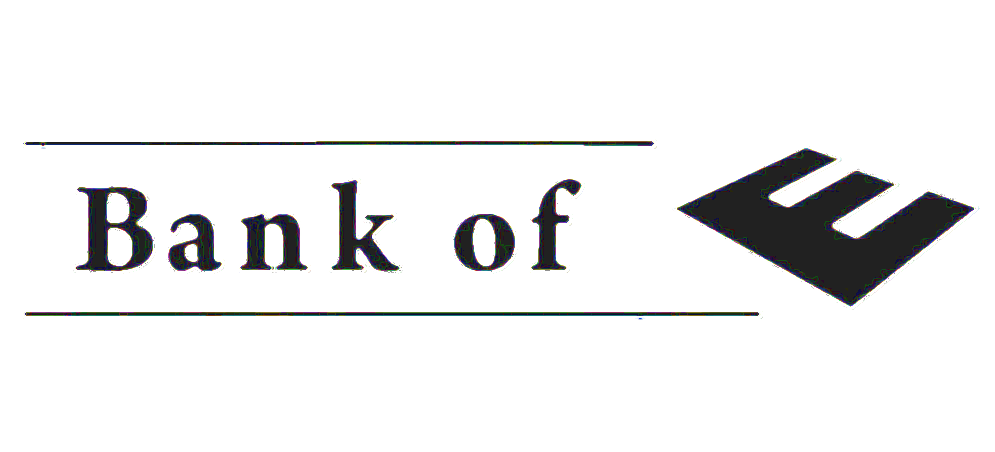
I own my sincere and whole hearted thanks to **Modern Vidya Niketan Sr. Sec. School** who allowed me to make project report. I would also like to express my gratitude to **Mrs. Pragya Gulati,** my teacher for constantly guiding me and helping me tackle a variety of hurdles with extreme patience throughout my project. She has infused in me a great inspiration and confidence in taking up this study in right direction.

Words cannot express my sincere thanks to the **Central Board of Senior Education** who has given us a chance to make this project.

Lastly, I would like to convey my heartiest thanks to my parents and my friends with whose support and guidance this assignment of mine has been succesfully accomplished.

Mehul Arora

**System Description**



**Bank Of E** provides a system that allows customers to create accounts, deposit or withdraw money, or transfer money to other accounts. The program also allows for a user with admin-levell priviledges to log in and check details related to customers and transactions.

The program starts with a menu that allows for:

1. Customer Login
2. Account Creation
3. Admin Login
4. Exit

The user can select either of the options from **Main Menu** form which lead to a **Login form** or **Account Creation form.**

If the user selects “Admin Login”, then the program also checks if the username-password combination provided in the Login form is assigned admin status.

The customer log in panel has the following features:

1. Deposit money
2. Withdraw Money
3. Transfer Money (Requires KYC to be done)
4. Update KYC
5. Check Balance
6. Log Out

The admin log in panel has the following features:

1. Find Customer
2. View Ledger
3. See Transaction Table
4. Find Transaction
5. Log Out

The provided functions allow for the user to seamlessly make changes to their bank account and also allow for the admin to traverse data according to their need. The program is equipped to ensure that one customer cannot access the data of other customers, thus upholding privacy.

**DATABASE STRUCTURE**

The python program refers to the database **“bankofe”** made in MySQL. The database contains the following tables:

# A.CUSTOMERDETAILS

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| Field | Type | Null | Key | Default | Extra |
| cust\_id | int | NO | PRI | NULL |  |
| name | varchar(50) | NO |  | NULL |  |
| address | varchar(50) | YES |  | NULL |  |
| dob | date | NO |  | NULL |  |
| kyc\_docnum | varchar(50) | YES |  | NULL |  |
| gender | int(1) | NO |  | NULL |  |
| phone | varchar(10) | YES |  | NULL |  |
| email\_id | varchar(50) | YES |  | NULL |  |
| occupation | varchar(30) | YES |  | NULL |  |

CREATE TABLE customerdetails (

cust\_id INT(5) NOT NULL PRIMARY KEY,

name VARCHAR(50) NOT NULL,

address VARCHAR(150),

dob DATE NOT NULL,

kvc\_docnum VARCHAR(20),

gender INT(1) NOT NULL,

phone VARCHAR(10) NOT NULL,

email\_id VARCHAR(50),

occupation VARCHAR(30) );

# B.CUSTOMERBALANCE

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| Field | Type | Null | Key | Default | Extra |
| cust\_id | int | NO | PRI | NULL |  |
| balance | int | NO |  | NULL |  |
| last\_transaction\_time | timestamp | YES |  | NULL |  |

CREATE TABLE customerbalance (

cust\_id INT(5) NOT NULL PRIMARY KEY,

balance INT(20),

last\_transaction\_time TIMESTAMP );

# C.TRANSACTIONS

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| Field | Type | Null | Key | Default | Extra |
| transac\_id | int | NO | PRI | NULL |  |
| dcust | int | YES |  | NULL |  |
| ccust | int | YES |  | NULL |  |
| amount | int | NO |  | NULL |  |
| time | timestamp | NO |  | NULL |  |

CREATE TABLE transactions(

transac\_id INT(9) NOT NULL PRIMARY KEY,

dcust INT(5) NOT NULL,

ccust INT(5) NOT NULL,

amount BIGINT(20) NOT NULL,

time TIMESTAMP NOT NULL );

# D.CREDENTIALS

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| Field | Type | Null | Key | Default | Extra |
| username | varchar(20) | NO | PRI | NULL |  |
| pwd | varchar(30) | NO |  | NULL |  |
| cust\_id | int | NO |  | NULL |  |
| admin | int | NO |  | NULL |  |

CREATE TABLE credentials (

userid VARCHAR(20) NOT NULL PRIMARY KEY,

pwd VARCHAR(30) NOT NULL,

cust\_id INT(5) NOT NULL,

admin INT(1) NOT NULL );

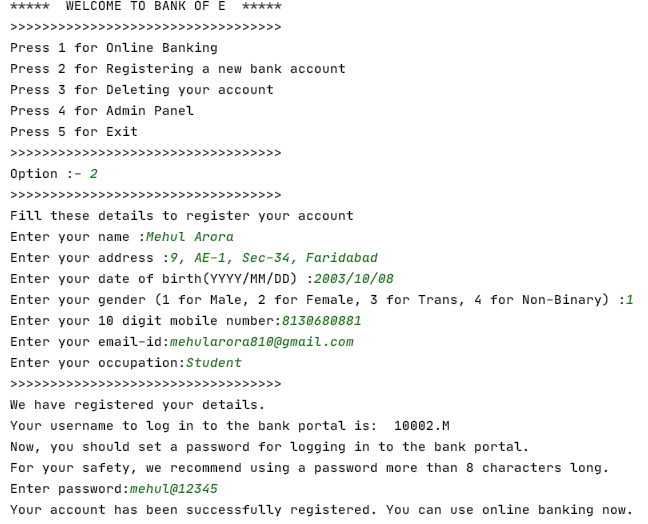
**Python Code**

**import** mysql.connector  
  
**'''  
BANK OF E : Online Banking Platform  
Copyright: Mehul Arora 2020  
Test Admin ID Password:  
 sociallyencrypted, mrrobot59  
 tyrellwellick, joanna66  
'''**db = mysql.connector.connect(host=**'localhost'**, username=**'root'**, password=**'mysql12345'**, database=**'bankofe'**)  
cursor = db.cursor()  
ctr = 0 *# Checks if login was successful*last = 1 *# Used to calculate last customer/transaction ID***def** isKYCDone(cid):  
 *"""Check if a customer has filled their KYC.  
  
 Keyword arguments:  
 cid -- customer ID  
 """* cursor.execute((**"SELECT kyc\_docnum FROM customerdetails WHERE cust\_id={}"**.format(cid)))  
 a = cursor.fetchall()  
 r = a[0]  
 **for** x **in** r:  
 condition = x  
 **return** condition  
  
  
**def** updatedBalance(cid):  
 *"""Returns updated balance after a transaction.  
  
 Keyword arguments:  
 cid -- customer ID  
 """* cursor.execute((**"SELECT balance FROM customerbalance WHERE cust\_id={} ;"**.format(cid)))  
 a = cursor.fetchall()[0]  
 **for** x **in** a:  
 print(**"Updated Balance :- "**, x)  
 print(**'>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>'**)  
  
  
**def** transaction(dcust, ccust, amount):  
 *"""Fills details of the transaction in to the TRANSACTIONS table of database BANKOFE  
  
 Keyword arguments:  
 dcust -- the customer ID from which the amount has been debited  
 dcust -- the customer ID to which the amount has been credited  
 amount -- value being transferred in rs.  
  
 Note:  
 A "NULL" value means there is a deposit or withdrawal by the customer.  
 """* cursor.execute(**"SELECT MAX(transac\_id) FROM transactions;"**)  
 a = cursor.fetchall()  
 a = a[0]  
 **for** x **in** a:  
 last = x  
 **if** last **is None**:  
 transac\_id = 111111  
 **else**:  
 transac\_id = 1 + last  
 cursor.execute((**"INSERT INTO transactions VALUES ({},{},{},{},NOW());"**.format(transac\_id, dcust, ccust, amount)))  
 db.commit()  
  
  
**def** moneyDeposit(cid):  
 *"""Function to handle money deposits.  
  
 Keyword arguments:  
 cid -- customer ID  
 """* money\_deposited = int(input(**'Amount to be deposited :- '**))  
 print(**'>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>'**)  
 cursor.execute((**"UPDATE customerbalance SET balance = balance + {} WHERE cust\_id={};"**.format(money\_deposited, cid)))  
 cursor.execute((**"UPDATE customerbalance SET last\_transaction\_time=NOW() WHERE cust\_id={};"**.format(cid)))  
 db.commit()  
 transaction(**"NULL"**, cid, money\_deposited)  
 updatedBalance(cid)  
  
  
**def** moneyWithdraw(cid):  
 *"""Function to handle money withdrawals.  
  
 Keyword arguments:  
 cid -- customer ID  
 """* money\_withdrawn = int(input(**'Amount to be withdrawn :- '**))  
 print(**'>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>'**)  
 cursor.execute((**"UPDATE customerbalance SET balance = balance - {} WHERE cust\_id={};"**.format(money\_withdrawn, cid)))  
 cursor.execute((**"UPDATE customerbalance SET last\_transaction\_time=NOW() WHERE cust\_id={};"**.format(cid)))  
 transaction(cid, **"NULL"**, money\_withdrawn)  
 updatedBalance(cid)  
  
  
**def** moneyTransfer(cid):  
 *"""Function to handle money transfers.  
  
 Keyword arguments:  
 cid -- customer ID  
 """* money\_transferred = int(input(**'Amount to be transferred :- '**))  
 depositAcc = int(input(**"Enter account ID to which money is to be transferred"**))  
 print(**'>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>'**)  
 cursor.execute((**"UPDATE customerbalance SET balance = balance - {} where cust\_id={}"**.format(  
 money\_transferred, cid)))  
 cursor.execute((**"UPDATE customerbalance SET balance = balance + {} where cust\_id={}"**.format(  
 money\_transferred, depositAcc)))  
 cursor.execute((  
 **"UPDATE customerbalance SET last\_transaction\_time=NOW() WHERE cust\_id = {};)"**.format(  
 cid)), multi=**True**)  
 cursor.execute((  
 **"UPDATE customerbalance SET last\_transaction\_time=NOW() WHERE cust\_id = {};)"**.format(  
 depositAcc)), multi=**True**)  
 db.commit()  
 transaction(cid, depositAcc, money\_transferred)  
 updatedBalance(cid)  
  
  
**def** welcome\_message():  
 print(**'>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>'**)  
 print(**'\*\*\*\*\* WELCOME TO BANK OF E \*\*\*\*\*'**)  
 print(**'>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>'**)  
  
  
**def** deleteAcc():  
 *"""Function to handle account deletion."""* us = input(**"Enter your username :- "**)  
 p = (input(**"Enter your password :-"**))  
 print(**'>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>'**)  
 cursor.execute(**"SELECT \* FROM credentials WHERE userid='{}' AND pwd='{}';"**.format(us, p))  
 data\_login = cursor.fetchall()  
 cursor.execute(**"DELETE FROM customerdetails WHERE cust\_id={};"**.format(data\_login[0][2]))  
 cursor.execute(**"DELETE FROM customerbalance WHERE cust\_id={};"**.format(data\_login[0][2]))  
 cursor.execute((**"DELETE FROM credentials WHERE userid ='{}'"**.format(us)))  
 db.commit()  
 print(**"Account deleted."**)  
  
  
**def** login():  
 *"""Function to handle customer login.  
  
 Returns:  
 data\_login -- A list containing matching records from the table CREDENTIALS of database BANKOFE  
 """* **while True**:  
 us = input(**"Enter your username :- "**)  
 p = (input(**"Enter your password :-"**))  
 print(**'>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>'**)  
 cursor.execute((**"SELECT \* FROM credentials where userid='{}' and pwd='{}';"**.format(us, p)))  
 data\_login = cursor.fetchall()  
 **if** len(data\_login) != 0:  
 globals()[**'ctr'**] = 1  
 **break  
 else**:  
 print(**'LOGIN FAILED'**)  
 print(**"USERNAME OR PASSWORD IS WRONG"**)  
 print(**'>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>'**)  
  
 **return** data\_login  
  
  
**def** userinterface():  
 *"""Interface through which customer interacts with the database"""* **global** condition  
 cdetails = login()  
 **if** globals()[**'ctr'**] == 1:  
 cust\_id = cdetails[0][2]  
 print(**"LOGIN SUCCESSFUL"**)  
 print(**'>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>'**)  
 **while True**:  
 print(**'Press 1 for depositing money'**)  
 print(**'Press 2 for withdrawing money'**)  
 print(**'Press 3 for doing kyc'**)  
 print(**'Press 4 for checking balance'**)  
 print(**'Press 5 to transfer money to other account'**)  
 print(**'Press 6 for logging out'**)  
 print(**'>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>'**)  
 ch = int(input(**"Enter your option :- "**))  
 **if** ch == 1:  
 moneyDeposit(cust\_id)  
 **elif** ch == 2:  
 moneyWithdraw(cust\_id)  
 **elif** ch == 3:  
 cond = isKYCDone(cust\_id)  
 **if** cond **is None**:  
 print(**'For KYC you need to provide either one of the following Government IDs.'**)  
 print(**'Press 1 for Aadhar Card'**)  
 print(**'Press 2 for Voter ID Card'**)  
 print(**'Press 3 for PAN Card'**)  
 print(**'Press 4 for Driving License'**)  
 print(**'>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>'**)  
 cho = int(input(**"Enter your choice :- "**))  
 print(**'>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>'**)  
 **if** cho == 1:  
 ad = (input(**"Aadhar Number :- "**))  
 cursor.execute(  
 (**"UPDATE customerdetails set kyc\_docnum='{}' where cust\_id={}"**.format(ad, cust\_id)))  
 db.commit()  
 print(**"KYC Done"**)  
 **elif** cho == 2:  
 vi = (input(**"Voter Id Number :- "**))  
 cursor.execute(  
 (**"UPDATE customerdetails SET kyc\_docnum='{}'' WHERE cust\_id={}"**.format(vi, cust\_id)))  
 db.commit()  
 print(**"KYC Done"**)  
 **elif** cho == 3:  
 pc = (input(**"Pan Card Number :- "**))  
 cursor.execute(  
 (**"UPDATE customerdetails SET kyc\_docnum='{}' WHERE cust\_id={}"**.format(pc, cust\_id)))  
 db.commit()  
 print(**"KYC Done"**)  
 **elif** ch == 4:  
 dl = (input(**"Driving License Number :- "**))  
 cursor.execute(  
 (**"UPDATE customerdetails SET kyc\_docnum='{}' WHERE cust\_id={}"**.format(dl, cust\_id)))  
 db.commit()  
 print(**"KYC Done"**)  
 **else**:  
 print(**'Wrong Choice'**)  
 **else**:  
 print(**'KYC Already Done'**)  
 print(**'>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>'**)  
 **elif** ch == 4:  
 q = **"SELECT balance FROM customerbalance WHERE cust\_id={};"**.format(cust\_id)  
 cursor.execute(q)  
 a = cursor.fetchall()  
 a = a[0]  
 **for** x **in** a:  
 print(**"Balance :- "**, x)  
 print(**'>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>'**)  
 **elif** ch == 5:  
 cond1 = isKYCDone(cust\_id)  
 **if** cond1 **is None**:  
 print()  
 print(**'>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>'**)  
 print(**"IMPORTANT:"**)  
 print(**"Update KYC first to make payments."**)  
 print(**'>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>'**)  
 **else**:  
 moneyTransfer(cust\_id)  
 **elif** ch == 6:  
 **break  
 else**:  
 print(**"Wrong Option "**)  
 print(**'>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>'**)  
  
  
**def** admininterface():  
 **while True**:  
 print(**'Press 1 for finding customer'**)  
 print(**'Press 2 for viewing ledger'**)  
 print(**'Press 3 for viewing transaction table'**)  
 print(**'Press 4 for locating transaction'**)  
 print(**'Press 5 to log out'**)  
 print(**'>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>'**)  
 ch = int(input(**"Enter your option :- "**))  
 **if** ch == 1:  
 checkCust = int(input(**"Enter Customer ID"**))  
 cursor.execute((**"SELECT \* FROM customerdetails WHERE cust\_id={};"**.format(checkCust)))  
 res = cursor.fetchall()  
 **for** x **in** res:  
 **for** i **in** x:  
 print(i)  
 print(**"\n"**)  
 **elif** ch == 2:  
 cursor.execute(**"SELECT \* FROM customerbalance;"**)  
 print(**"Customer ID| Balance| Last Transaction Time"**)  
 print(cursor.fetchall())  
 **elif** ch == 3:  
 cursor.execute(**"SELECT \* FROM transactions;"**)  
 print(**"Transaction ID| Deposited to| Credited From| Amount| Transaction Time"**)  
 res = cursor.fetchall()  
 **for** i **in** res:  
 print(i)  
 **elif** ch == 4:  
 checkTrnsc = int(input(**"Enter Transaction ID"**))  
 cursor.execute((**"SELECT \* FROM transactions WHERE transac\_id={};"**.format(checkTrnsc)))  
 res = cursor.fetchall()  
 **for** x **in** res:  
 **for** i **in** x:  
 print(i)  
 print(**"\n"**)  
 **elif** ch == 5:  
 **break**

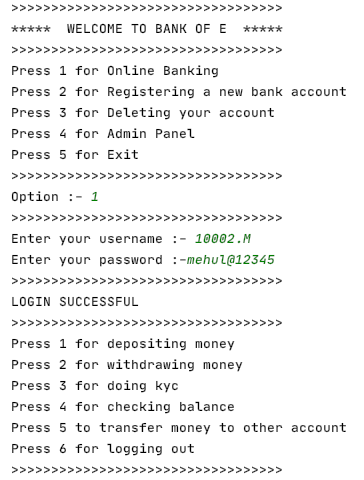
**while True**:  
 welcome\_message()  
 print(**'Press 1 for Online Banking'**)  
 print(**'Press 2 for Registering a new bank account'**)  
 print(**'Press 3 for Deleting your account'**)  
 print(**'Press 4 for Admin Panel'**)  
 print(**'Press 5 for Exit'**)  
 print(**'>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>'**)  
 choice = int(input(**"Option :- "**))  
 print(**'>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>'**)  
 **if** choice == 1:  
 userinterface()  
 **elif** choice == 2:  
 cursor.execute(**"SELECT MAX(cust\_id) FROM customerdetails;"**)  
 a = cursor.fetchall()  
 a = a[0]  
 **for** x **in** a:  
 last = x  
 **if** last **is None**:  
 custid = 111111  
 **else**:  
 custid = 1 + last  
 print(**'Fill these details to register your account '**)  
 name = input(**"Enter your name :"**)  
 address = (input(**"Enter your address :"**))  
 dob = (input(**"Enter your date of birth(YYYY/MM/DD) :"**))  
 gender = int(input(**"Enter your gender (1 for Male, 2 for Female, 3 for Trans, 4 for Non-Binary) :"**))  
 phone = input(**'Enter your 10 digit mobile number:'**)  
 emailid = input(**"Enter your email-id:"**)  
 occupation = input(**"Enter your occupation:"**)  
 user = str(custid) + **"."** + name[0].capitalize()  
 print(**'>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>'**)  
 kyc = **'false'** query = **"INSERT INTO customerdetails VALUES({},'{}','{}','{}',NULL,{},'{}','{}','{}');"**.format(custid, name,  
 address,  
 dob, gender,  
 phone,  
 emailid,  
 occupation)  
 cursor.execute(query)  
 print(**"We have registered your details."**)  
 print(**"Your username to log in to the bank portal is: "**, user)  
 print(**"Now, you should set a password for logging in to the bank portal."**)  
 print(**"For your safety, we recommend using a password more than 8 characters long."**)  
 pwd = input(**"Enter password:"**)  
 query = **"INSERT INTO credentials VALUES('{}','{}',{},0);"**.format(user, pwd, custid)  
 cursor.execute(query)  
 cursor.execute((**"INSERT INTO customerbalance VALUES ({},0,NULL)"**.format(custid)))  
 print(**"Your account has been successfully registered. You can use online banking now."**)  
 db.commit()  
 **elif** choice == 3:  
 deleteAcc()  
 **elif** choice == 4:  
 *# Handle access to the admin interface.* adetails = login()  
 **if** globals()[**'ctr'**] == 1:  
 admin = adetails[0][3]  
 **if** admin == 1:  
 print(**"LOGIN SUCCESSFUL"**)  
 admininterface()  
 **else**:  
 print(**"This account is not authorised to access this page."**)  
 print(**'>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>'**)  
 **elif** choice == 5:  
 **break  
 else**:  
 print(**'Wrong Option'**)  
 print(**'>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>'**)

**Screenshots**

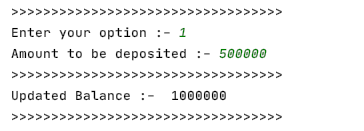
# Creating an Account



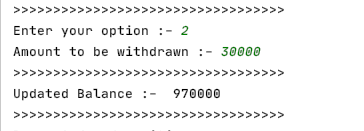
# Online Banking



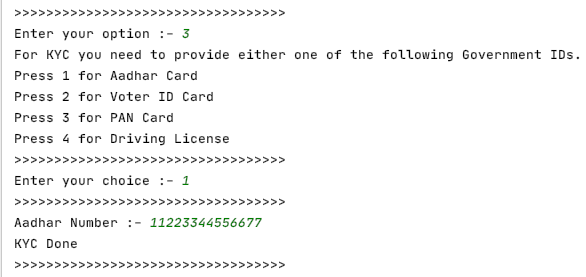
## Depositing Money



## Withdrawing Money

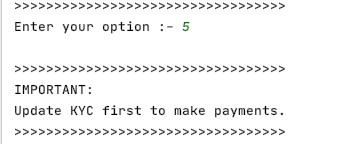


## Doing KYC

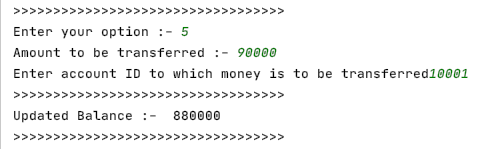


## Transferring Money

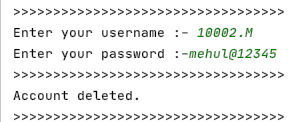
### When KYC is not done



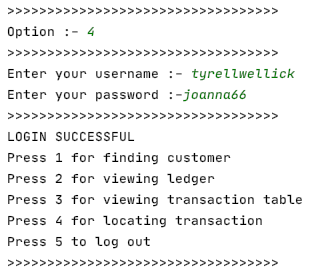
1. When KYC is done



# Account Deletion



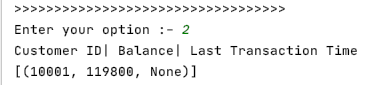
# Admin Panel



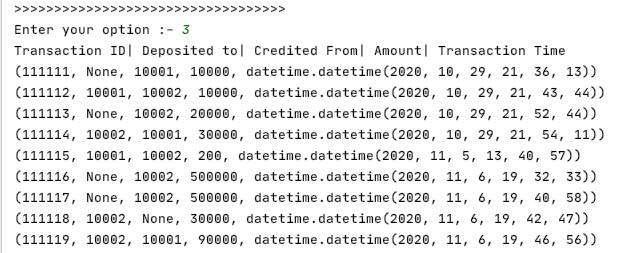
## Finding Customer



## Viewing Ledger



## Viewing Transaction Table



## Locating Transaction

