به نام خدا

پروژه

بانکداری

استاد مربوطه: استاد بخشنده

حسین الف پوریان و کیمیا حسن زاده

مدلسازی:

ابتدا نیاز به چهار کلاس و متد های مربوطه داریم به گونه ای که :

Account:
User
Account Type
Account ID
Balance
Time

User:
Name
Last name
Father name
Birthdate
City
Mobile
National ID
Phone
User ID

Admin:
Username
Password

Bank:
Username
Password

اکنون کد نویسی رو شروع میکنیم:

ابتدا چون برنامه نیاز به دیتابیس(mysql) و کتابخانه مخصوص تاریخ(datetime)دارد،این دو را ایمپورت میکنیم

سپس اولین کلاسمان به اسم Account را به سه رفتار (Deposit ,Withdraw ,Showing Account Details) به صورت زیر میسازیم:

```
        #Account Class
    class Account:
        def __init__(self,user,typeOfAccount,id):
            self.user=user
            self.typeOfAccount= typeOfAccount
            self.accountId=id
            self.balance=0
            self.date=datetime.now()

            def withdraw(self,amount): #function for withdrawing and after all it shows
    how much is remaining in balance
                if amount>50000000 :
                    print("Sorry,your amount is above the limit...")
                else :
                    if amount<=self.balance:
                        self.balance-=amount
                        print(f"Withdraw done successfully . Your balance now is
    {self.balance}")
                    else :
                      print("Withdraw wasn`t successful ,because your balance is lower
    than amount...")

            def deposit(self,amount): #function for depositing and after all it skows
    how much is remainig in balance
                self.balance+=amount
                print(f"Deposit done successfully . Your balance now is {self.balance}")

            def accountDetails(self): #function that print name, created time and
    balance
                print(f"Account owner name is :{self.user}\nDate of creation is
    :{self.date}\nBalance is :{self.balance}")
```

سپس سراغ کلاس User میرویم با رفتار :

(accountDetails)

```
        #Users class
    class User:
        def __init__(self, name, lastName, fatherName, birthDate, city, phoneNumber,
    idCode, phone=None, userId=None):
        self.name = name
        self.lastName = lastName
        self.fatherName = fatherName
        self.birthDate = birthDate
        self.city = city
        self.phoneNumber = phoneNumber
        self.idCode = idCode
        self.phone = phone
        self.userId = userId
    def showDetails(self):
        print(f"Name is : {self.name} {self.lastName}\nFather Name is :
{self.fatherName}\nBirth Date is: {self.birthDate}\nCity is : {self.city}\nphone
number is : {self.phone_number}\nNational ID is: {self.id_code}\nUser Id
is:{self.userId}")
        if self.phone:
            print(f"Phone is: {self.phone}")
```

و کلاس ادمین با رفتار :

(Verify)

```
        #Admins class

class Admin:
    def __init__(self, username="admin", password="admin"):
        self.username = username
        self.password = password

    def verify(self, username, password):    # functioon to see if you are typing
the right username and pass
        return self.username == username and self.password == password
```

و در اخر میرسیم به کلاس اخر و اصلی که نیاز به کتابخانه و دیتابیس دارد و رفتار های زیادی در خودش جای داده که عبارت اند از:

1.CREAT TABLE	**6.LOGIN**
2. CREAT USER	**7.REGISTER**
3.CREAT ACCOUNT	**8.DEPOSIT**
4.FIND USER BY ID	**9.WITHDRAW**
5. FIND USER BY N.ID	**10.SHOW DETAILS**
	11.START THE APP

```
    #Bank class
class Bank:
    def __init__(self):
        self.users = []
        self.accounts = []
        self.admin = Admin()
        self.mydb = sqlite3.connect('Online Banking.db')
        self.create_tables()

    def create_tables(self):
        cursor = self.mydb.cursor() #A cursor in database is a construct which
allows you to iterate/traversal the records of a table
        cursor.execute('''CREATE TABLE IF NOT EXISTS users
                          (userId INTEGER PRIMARY KEY AUTOINCREMENT,
                           name TEXT,lastName TEXT,
                           fatherName TEXT,
                           birthDate TEXT,
                           city TEXT,
                           phoneNumber TEXT,
                           idCode TEXT,
                           phone TEXT)''')# frist  if user doesn`t exsit the table
starts to create .
                                          #Use execute for writting command to the
data base ,Primary is specefic for each users
        cursor.execute('''CREATE TABLE IF NOT EXISTS accounts
                          (accountId INTEGER PRIMARY KEY AUTOINCREMENT,
                           userId INTEGER,
                           accountType TEXT,
                           date TEXT,
                           balance INTEGER,
                           FOREIGN KEY(userId) REFERENCES users(userId))''')
#account id and user id is connection bridge between two tables
        self.mydb.commit() #Use commit to save changes on db

        #Define new useer and save it in database
    def createUser(self, name, lastName, fatherName, birthDate, city, phoneNumber,
idCode, phone=None):
        cursor = self.mydb.cursor()
        sql = "INSERT INTO users (name, lastName, fatherName, birthDate, city,
phoneNumber, idCode, phone) VALUES (?, ?, ?, ?, ?, ?, ?, ?)" #? shows null
        val = (name, lastName, fatherName, birthDate, city, phoneNumber, idCode,
phone)
        cursor.execute(sql, val)
        self.mydb.commit()
        userId = cursor.lastrowid #This read-only property returns the value
generated for an AUTO_INCREMENT column by the previous INSERT or UPDATE statement or
None when there is no such value available.
        user = User(name, lastName, fatherName, birthDate, city, phoneNumber,
idCode, phone, userId)
        self.users.append(user)
        return user
```

```
        #Define new account and save in database
    def create_account(self, user, accountType):
        cursor = self.mydb.cursor()
        sql = "INSERT INTO accounts (userId, accountType, date, balance) VALUES (?,
?, ?, ?)"
        val = (user.userId, accountType, datetime.now(), 0)
        cursor.execute(sql, val)
        self.mydb.commit()
        global accountId
        accountId = cursor.lastrowid
        account = Account(user, accountType, accountId)
        self.accounts.append(account)
        return account
        #Search with ID code
    def find_user_by_id_code(self, idCode):
        for user in self.users:
            if user.idCode == idCode:
                return user
        return None

        #Search with account ID
    def find_account_by_id(self, accountId):
        for account in self.accounts:
            if account.accountId == accountId:
                return account
        return None

        #Users login
    def login(self):
        idCode = input("Enter your national ID: ")
        user = self.find_user_by_id_code(idCode)
        if user:
            accountId = input("Enter your account ID: ")
            account = self.find_account_by_id(int(accountId))
            if account and account.user.userId == user.userId:
                print("Login successful")
                return account
            else:
                print("Incorrect account ID")
        else:
            print("User not found")
```

```
    def register(self):
        name = input("Enter your name: ")
        lastName = input("Enter your last name: ")
        fatherName = input("Enter your father name: ")
        birthDate = input("Enter your birth date (yyyy-mm-dd): ")
        city = input("Enter your city: ")
        phoneNumber = input("Enter your phone_number number: ")
        idCode = input("Enter your national ID: ")
        phone = input("Enter your phone number (optional): ")
        user = self.createUser(name, lastName, fatherName, birthDate, city,
phoneNumber, idCode, phone)
        print("User created successfully")
        print("Please create an account")
        accountType = input("Enter account type: ")
        account = self.create_account(user, accountType)
        print("Account created successfully")
        print(f"Account Create with Account ID : {accountId}")

        #Deposting
    def deposit(self, account):
        amount = int(input("Enter amount to deposit: "))
        account.deposit(amount)
        cursor = self.mydb.cursor()
        sql = "UPDATE accounts SET balance = ? WHERE account_id = ?"
        val = (account.balance, account.accountId)
        cursor.execute(sql, val)
        self.mydb.commit()

        #Withdrawing
    def withdraw(self, account):
        amount = int(input("Enter amount to withdraw: "))
        account.withdraw(amount)
        cursor = self.mydb.cursor()
        sql = "UPDATE accounts SET balance = ? WHERE account_id = ?"
        val = (account.balance, account.account_id)
        cursor.execute(sql, val)
        self.mydb.commit()

        #Showing account detail
    def show_account_details(self, account):
        account.show_details()

        #Showing user detail
    def show_user_details(self, user):
        user.show_details()
```

```
#-----------------------------
#-----STARTING APLICATION-----

    def start(self):
        while True:
            print("\nWelcome to our Online Banking ")
            print("1. Login")
            print("2. Register")
            print("3. Logout")
            print("Developing by : Mr.Alefporian and Ms.Hassanzade")
            choice = input("Enter your choice: ")
            if choice == "1":
                account = self.login()
                if account:
                    while True:
                        print("\n1. Deposit")
                        print("2. Withdraw")
                        print("3. Show Account Details")
                        print("4. Show User Details")
                        print("5. Logout")
                        choice = input("Enter your choice: ")
                        if choice == "1":
                            self.deposit(account)
                        elif choice == "2":
                            self.withdraw(account)
                        elif choice == "3":
                            self.show_account_details(account)
                        elif choice == "4":
                            self.show_user_details(account.user)
                        elif choice == "5":
                            break
                        else:
                            print("Invalid choice, please try again")
            elif choice == "2":
                self.register()
            elif choice == "3":
                break
            else:
                print("Invalid choice")

if __name__ == "__main__": #It Allows You to Execute Code When the File Runs as a
Script,
    bank = Bank()
    bank.start()
```

در ابن گزارش سعی نمودیم تا هر چه را استاد ممکن بود در ارائه برایشان سوال شود را با کامنت گذاشتن تا حدی توضیح دادیم...

با تشکر از استاد زهرا بخشنده