**ATM Requirements**

For this task you are going to implement an ATM system from which users can manage their accounts. There are two types of users: Customers and Administrators. Both are presented with their own menus (after login). Customers can use the system to withdraw cash, deposit cash and get their current balance. For simplicity, assume a customer can have only 1 account (so they can't have a savings AND checking account or a Joint account with a spouse for example). Administrators can view, create, delete, and update accounts of customers. All data should be stored in a MySQL database and the program is a console application (not a web app).

Login: When your program starts, it should display a login screen. Customers/Admins will be asked to enter a login and 5-digit pin code. The system verifies the login and pin and displays an error if it's incorrect.

Enter login: Adnan123
Enter Pin code: 12345

Note: The writing in green is entered by the customer/administrator.

Customer: A customer is then taken to the customer option menu where they will select one of the following options:

1----Withdraw Cash
3----Deposit Cash
4----Display Balance
5----Exit

**b) Withdraw Cash**
In this case, the customer will be asked the amount they wish to withdraw. Note, the amount must be valid.

Enter the withdrawal amount: 3600
Cash Successfully Withdrawn
Account #12
Date: 01/29/2024
Withdrawn: 3600
Balance: 154,500

**3----Deposit Cash**
If the customer selects deposit cash, they are asked to specify the amount to deposit into the account.

Enter the cash amount to deposit: 12562

Cash Deposited Successfully.
Account #12
Date: 01/29/2024
Deposited: 12562
Balance: 154,500

**4----Display Balance**
This displays the balance on the screen.

Account #12
Date: 01/29/2024
Balance: 154,500

Administrator: When logged in as an administrator they should be presented with the following menu:

1----Create New Account
2----Delete Existing Account
3----Update Account Information
4----Search for Account
6----Exit

**1----Create New Account**

If the admin selects this option, they are asked to enter account information. The system should check the validity of the data i.e. Pin Code is an integer of length 5.

Login: Javed123
Pin Code: 12345
Holders Name: XYZ
Starting Balance: 6000
Status: Active

Account Successfully Created – the account number assigned is: 19

**2---Delete Existing Account**

If the admin selects this option, they are asked to enter an account number.

Enter the account number to which you want to delete: 15
You wish to delete the account held by John Doe. If this information is correct, please re-enter the account number: 15
Account Deleted Successfully

**3---Update Account Information**

The program should first ask the admin to enter the account number to be updated:

Enter the Account Number: 15

The fields the admin can update are listed in green.

Account # 15
Holder: Mr James
Balance: 50,000
Status: Disabled
Login: dotNet66
Pin Code: 45678

**4---Search for Account**

This will display a menu asking the admin to enter the account number of a user. It will then display the account information:

Enter Account number: 15
The account information is:
Account # 15
Holder: Mr James

Balance: 50,000
Status: Disabled
Login: dotNet66
Pin Code: 45678

Additional Requirements:

- This is a command line application (console app).
- The system should be robust: valid input, min/max values, exception handling etc.
- We are not concerned with security for this project (like SQL injection or encrypting user information).
- Store data using MySQL.
- Implement code using Java or C#.
- Use dependency injection (DI) along with a DI framework of your choice (such as ninject).
- For this project unit tests are optional (you are not required to turn them in or show them to me). This is because we have not discussed testing.
- I want to see you hiding your data correctly. For example, make everything private and only when needed make it public. As another example, don't have a getter for every attribute in your class.
- I want stateless and functional style programming. Obviously, you will need state and OO programing! This is an OO class! I'm just saying be very strategic how you use state and immutability. For example, don't have a setter for every attribute in your class. As another example, leverage immutability by not passing in a list as an IN/OUT (pass by reference in C++ style), instead pass it in and return a new list (NOTE: I'm assuming this does not impact performance significantly).
- Leverage the SOLID design principles. I've mentioned before you don't need to implement SOLID perfectly, there are times it over complicates the application and it's not worth it. I understand this is a simple application, so do your best at balancing over complicating the application with still using SOILD in this application to learn and practice it!