SOFTWARE SPECIFICATION

Banking System

Group3

Author(D) : xxxx

Table of Contents

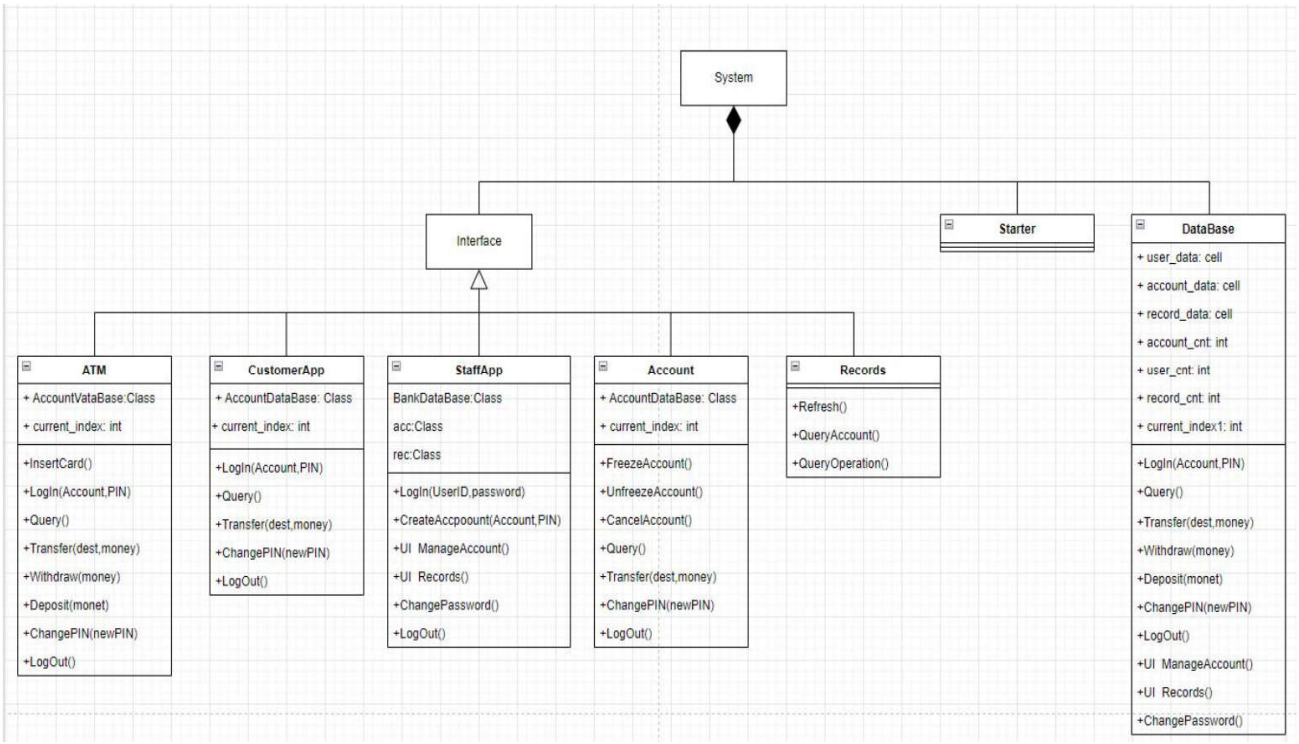
S3:Customer APP UI implementation

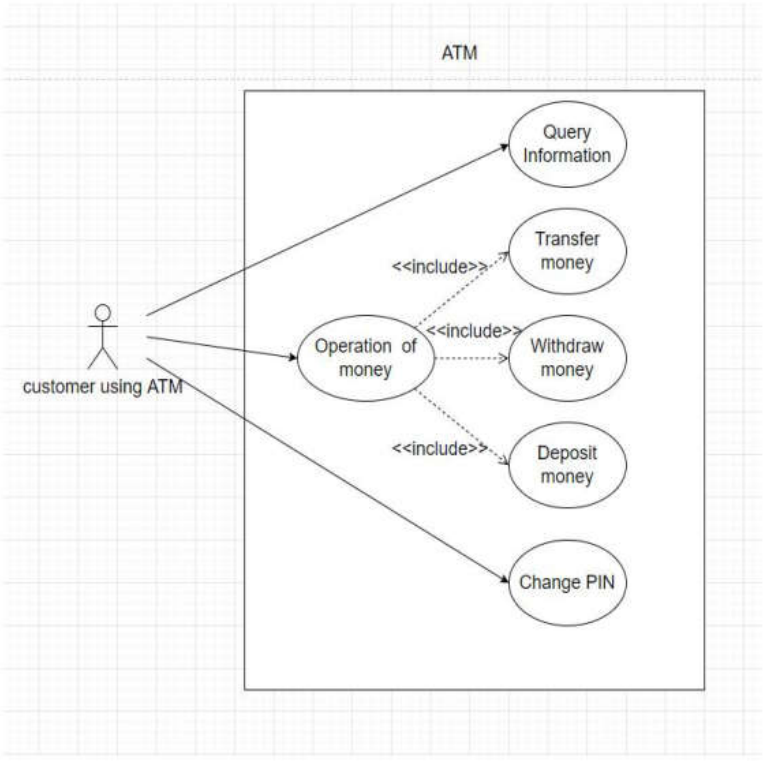
- S3.1: Verify account information
- S3.2: Enter customer APP transaction menu
- S3.3: Carry out customer APP transactions
  - S3.3.1: Customer APP Query
  - S3.3.2: Customer APP Transfer
  - S3.3.3: Customer APP Change account PIN
  - S3.3.4: Customer APP Logout

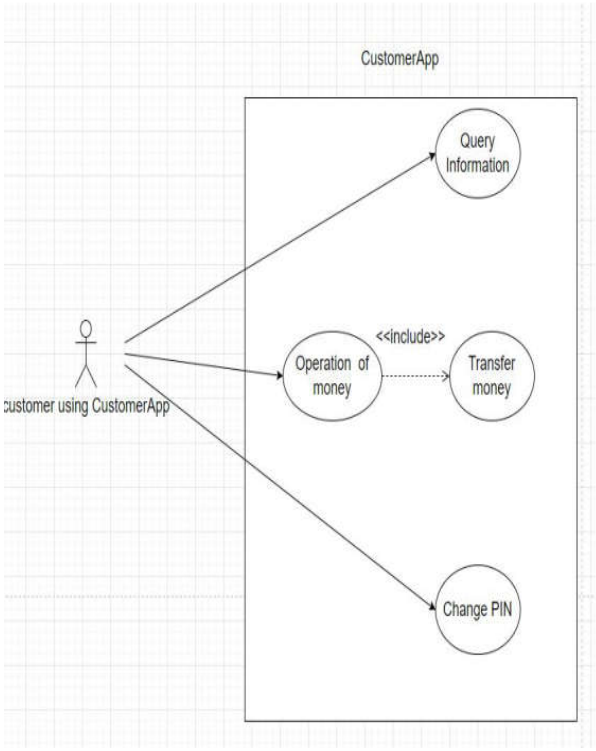
S4: Staff APP UI implementation

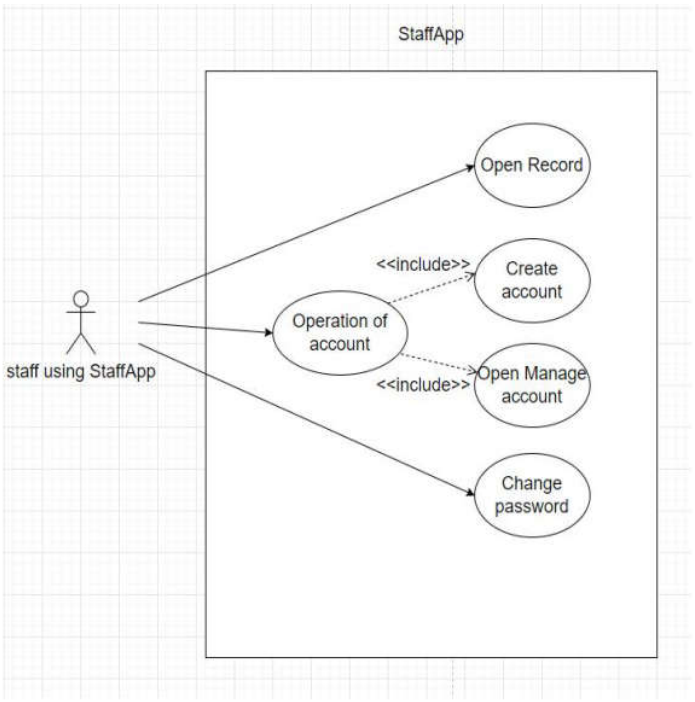
- S4.1:Verify user information
- S4.2: Register user account
- S4.3: Enter customer APP transaction menu
- S4.4: Create account
- S4.5: Manage account
  - S4.5.1: Staff APP Freeze/Unfreeze account
  - S4.5.2: Staff APP Cancel account
  - S4.5.3: Staff APP Query
  - S4.5.4: Staff APPTransfer
  - S4.5.5: Staff APPChange account PIN
  - S4.5.6: Staff APP Logout
- S4.6: View Transaction records
- S4.7: Change user password
- S4.8: Staff APP Logout

System Architecture






CustomerApp
Query Information
<<include>>
Operation of money
Transfer money
customer using CustomerApp
Change PIN


StaffApp
Open Record
<<include>>
Create account
Operation of account
Open Manage account
<<include>>
staff using StaffApp
Change password

S1: Database implementation

S1.1 :Database initialization

1.We use .xlsx file to store the data, including account data, user data(staff data) and transaction records.

As for the account_data.xlsx, the first column represents the customeraccount(6 bits), the second column represents the account PIN(6 bits), the third column represents the balance in this account, and the D1 cell represents the number of the accounts. It should be noted that a valid account has 6 bits, while an invalid account has 7 bits(if "1xxxxxx", it means the account is cancelled; if "2xxxxxx", it means the account is freezed).

	A	B	C	D
1	1100000	123456	1010	6
2	1100001	111111	255.99	
3	1100002	222222	88.01	
4	100003	123123	99998383.06	
5	100004	123456	819.25	
6	100005	123456	201	

As for the user_data.xlsx, the first column represents the bank administrator's user name, the second column represents the bank administrator'suser account(5 bits), the third column represents the bank administrator'suser account password, and the D1 cell represents the number of the user accounts.

	A	B	C	D	E
1	fgcc	10000	123456	6	
2	cst	10001	123456		
3	zdy	10002	222222		
4	xyz	10003	333333		
5	admin	10004	121212		

As for the records_data.xlsx, the first column represents the customer account which does the operation, the second column represents the operation itself("D"means deposit, "W" means withdraw, "T" means transfer), the third column represents the amount(for example, deposit 100 yuan, transfer 1000 yuan), the fourth column represents the account transferred to (if the operation is "T"), and the E1 cell represents the number of the records.

	A	B	C	D	E
1	100000	D	100	0	54
2	100000	W	1000	0	
3	100000	T	50	100001	
4	100001	T	100	100000	
5	100001	D	100	0	
6	100001	W	100	0	
7	100000	T	50	100001	

2. We read the data from those excel files, and use 1000*3 cells in matlab to store account_data, use 1000*3 cells in matlab to store user_data, use 1000*4 cells in matlab to store records_data, and use another 3 cells to store the numbers of customer accounts, user accounts(staff accounts) and the records.

```
classdef Database < handle
    properties
        user_data = cell(1000,3);
        account_data = cell(1000,3);
        records_data = cell(1000,4);
        account_cnt = 0;
        user_cnt = 0;
        records_cnt = 0;
        current_index1 = 0;
    end
    methods
        function BankDatabase = Database()
            BankDatabase.user_data = readcell('user_data.xlsx','Range','A1:C1000');
            BankDatabase.account_data = readcell('account_data.xlsx','Range','A1:C1000');
            BankDatabase.records_data = readcell('records_data.xlsx','Range','A1:D1000');
            cell2 = readcell('user_data.xlsx','Range','D1');
            BankDatabase.user_cnt = cell2{1};
            cell1 = readcell('account_data.xlsx','Range','D1');
            BankDatabase.account_cnt = cell1{1};
            cell3 =  readcell('records_data.xlsx','Range','E1');
            BankDatabase.records_cnt = cell3{1};
        end
```

Note: From S1.2 to S1.15, you may find in the beginning of each function, there would be codes below:

```
function addAccount(BankDatabase,data)
    BankDatabase.user_data = readcell('user_data.xlsx','Range','A1:C1000');
    BankDatabase.account_data = readcell('account_data.xlsx','Range','A1:C1000');
    cell2 = readcell('user_data.xlsx','Range','D1');
    BankDatabase.user_cnt = cell2{1};
    cell1 = readcell('account_data.xlsx','Range','D1');
    BankDatabase.account_cnt = cell1{1};
```

These codes are to update the data, which makes sure that the current data are the

newest.

S1.2: Add account

We pass parameter “data”, which is a 3*1 cell(like {Account, PIN, Balance})into this function.We write the “data” into the correct line in the 1000*3 account cells which are initialized in S1.1, and let the account_cnt plus one. Finally, we use writecell() and writematrix() in matlab to write them back to the excel file account_data.xlsx.

```
function addAccount(BankDatabase,data)
    BankDatabase.user_data = readcell('user_data.xlsx','Range','A1:C1000');
    BankDatabase.account_data = readcell('account_data.xlsx','Range','A1:C1000');
    cell2 = readcell('user_data.xlsx','Range','D1');
    BankDatabase.user_cnt = cell2{1};
    cell1 = readcell('account_data.xlsx','Range','D1');
    BankDatabase.account_cnt = cell1{1};

    BankDatabase.account_cnt = BankDatabase.account_cnt + 1;
    BankDatabase.account_data(BankDatabase.account_cnt, :) = data;
    writecell(BankDatabase.account_data, 'Data/account_data.xlsx');
    writematrix(BankDatabase.account_cnt, 'Data/account_data.xlsx', 'Range', 'D1');
    %xlswrite
end
```

S1.3: Create account

Generate an account(6 bits) by adding 100000 to account_cnt.

```
function new_account = createAccount(BankDatabase)
    BankDatabase.user_data = readcell('user_data.xlsx','Range','A1:C1000');
    BankDatabase.account_data = readcell('account_data.xlsx','Range','A1:C1000');
    cell2 = readcell('user_data.xlsx','Range','D1');
    BankDatabase.user_cnt = cell2{1};
    cell1 = readcell('account_data.xlsx','Range','D1');
    BankDatabase.account_cnt = cell1{1};

    new_account = BankDatabase.account_cnt+100000;
end
```

S1.4: Add user (bank staff)

We pass parameter “data”, which is a 3*1 cell (like {Name, Account, Password})into this function.We write the “data” into the correct line in the 1000*3 user cells which are initialized in S1.1, and let the user_cnt plus one. Finally, we use writecell() and writematrix() in matlab to write them back to the excel file user_data.xlsx.

```
function addUser(BankDatabase, data)
    BankDatabase.user_data = readcell('user_data.xlsx','Range','A1:C1000');
    BankDatabase.account_data = readcell('account_data.xlsx','Range','A1:C1000');
    cell2 = readcell('user_data.xlsx','Range','D1');
    BankDatabase.user_cnt = cell2{1};
    cell1 = readcell('account_data.xlsx','Range','D1');
    BankDatabase.account_cnt = cell1{1};

    BankDatabase.user_cnt = BankDatabase.user_cnt + 1;
    BankDatabase.user_data(BankDatabase.user_cnt, :) = data;
    writecell(BankDatabase.user_data, 'Data/user_data.xlsx');
    writematrix(BankDatabase.user_cnt, 'Data/user_data.xlsx', 'Range', 'D1');
end
```

S1.4: Create user (bank staff)

Generate a user account(5 bits) by adding 10000 to user_cnt.

```
function new_user = createUser(BankDatabase)
    BankDatabase.user_data = readcell('user_data.xlsx','Range','A1:C1000');
    BankDatabase.account_data = readcell('account_data.xlsx','Range','A1:C1000');
    cell2 = readcell('user_data.xlsx','Range','D1');
    BankDatabase.user_cnt = cell2{1};
    cell1 = readcell('account_data.xlsx','Range','D1');
    BankDatabase.account_cnt = cell1{1};

    new_user = BankDatabase.user_cnt+10000;
end
```

S1.6: Query

We pass parameter“current_index” (represent the position in the 1000*3 account data cells) to this function. Then we get the balance information from BankDatabase.account_data{current_index, 3} (row: current_index, column: 3).

```
function appTransfer(BankDatabase, current_index, account_index, amount)
    BankDatabase.user_data = readcell('user_data.xlsx','Range','A1:C1000');
    BankDatabase.account_data = readcell('account_data.xlsx','Range','A1:C1000');
    cell2 = readcell('user_data.xlsx','Range','D1');
    BankDatabase.user_cnt = cell2{1};
    cell1 = readcell('account_data.xlsx','Range','D1');
    BankDatabase.account_cnt = cell1{1};

    BankDatabase.account_data{current_index, 3} = BankDatabase.account_data{current_index, 3} - amount;
    BankDatabase.account_data{account_index, 3} = BankDatabase.account_data{account_index, 3} + amount;
    writecell(BankDatabase.account_data, 'Data/account_data.xlsx');
end
```

S1.7: Transfer

We pass 3 parameters “current_index, account_index, amount” to the function, “current_index” represents the index of current customer account, “account_index” represents the index of the account transferred to and “amount” represents the amount of money.We let the balance of the account transferred plus “amount”, let the balance of the current account minus “amount”(the modifications are all implemented on account_data cells).Finally, we write them back to the excel file account_data.xlsx.

```
function appTransfer(BankDatabase,current_index, account_index, amount)
    BankDatabase.user_data = readcell('user_data.xlsx','Range','A1:C1000');
    BankDatabase.account_data = readcell('account_data.xlsx','Range','A1:C1000');
    cell2 = readcell('user_data.xlsx','Range','D1');
    BankDatabase.user_cnt = cell2{1};
    cell1 = readcell('account_data.xlsx','Range','D1');
    BankDatabase.account_cnt = cell1{1};

    BankDatabase.account_data{current_index, 3} = BankDatabase.account_data{current_index, 3} - amount;
    BankDatabase.account_data{account_index, 3} = BankDatabase.account_data{account_index, 3} + amount;
    writecell(BankDatabase.account_data, 'Data/account_data.xlsx');
end
```

S1.8: Withdraw

We pass 2 parameters “current_index, amount” to the function, “current_index” represents the index of current customer account, and “amount” represents the amount of money. We let the balance of the current account minus “amount”(the modifications are all implemented on account_data cells). Finally, we write those cells in matlab back to the excel file account_data.xlsx.

```
function Withdraw(BankDatabase,current_index, amount)
    BankDatabase.user_data = readcell('user_data.xlsx','Range','A1:C1000');
    BankDatabase.account_data = readcell('account_data.xlsx','Range','A1:C1000');
    cell2 = readcell('user_data.xlsx','Range','D1');
    BankDatabase.user_cnt = cell2{1};
    cell1 = readcell('account_data.xlsx','Range','D1');
    BankDatabase.account_cnt = cell1{1};

    BankDatabase.account_data{current_index, 3} = BankDatabase.account_data{current_index, 3} - amount;
     writecell(BankDatabase.account_data, 'Data/account_data.xlsx');
end
```

S1.9: Deposit

We pass 2 parameters “current_index, amount” to the function, “current_index” represents the index of current customer account, and “amount” represents the amount of money. We let the balance of the current account plus “amount”(the modifications are all implemented on account_data cells).Finally, we write those cells in matlab back to the excel file account_data.xlsx.

```
function Deposit(BankDatabase,current_index, amount)
    BankDatabase.user_data = readcell('user_data.xlsx','Range','A1:C1000');
    BankDatabase.account_data = readcell('account_data.xlsx','Range','A1:C1000');
    cell2 = readcell('user_data.xlsx','Range','D1');
    BankDatabase.user_cnt = cell2{1};
    cell1 = readcell('account_data.xlsx','Range','D1');
    BankDatabase.account_cnt = cell1{1};

    BankDatabase.account_data{current_index, 3} = BankDatabase.account_data{current_index, 3} + amount;
    writecell(BankDatabase.account_data, 'Data/account_data.xlsx');
end
```

S1.10: Change user password

We pass 1 parameter “password” to the function, and we use this new “password” to cover the old one in the user_data cells. Finally, we write these cells in matlab back to the excel file user_data.xlsx.

```
function changeUserpassword(BankDatabase,password)
    BankDatabase.user_data = readcell('user_data.xlsx','Range','A1:C1000');
    BankDatabase.account_data = readcell('account_data.xlsx','Range','A1:C1000');
    cell2 = readcell('user_data.xlsx','Range','D1');
    BankDatabase.user_cnt = cell2{1};
    cell1 = readcell('account_data.xlsx','Range','D1');
    BankDatabase.account_cnt = cell1{1};

    BankDatabase.user_data{BankDatabase.current_index1, 3} = password;
    writecell(BankDatabase.user_data, 'Data/user_data.xlsx');
end
```

S1.11: Change account PIN

We pass 2 parameters “current_index, password” to the function, “current_index” represents the index of current customer account, and we use this new “password” to cover the old password of the current account in the account_data cells. Finally, we write these cells in matlab back to the excel file account_data.xlsx.

```
function changeAccountPIN(BankDatabase,current_index,password)
    BankDatabase.user_data = readcell('user_data.xlsx','Range','A1:C1000');
    BankDatabase.account_data = readcell('account_data.xlsx','Range','A1:C1000');
    cell2 = readcell('user_data.xlsx','Range','D1');
    BankDatabase.user_cnt = cell2{1};
    cell1 = readcell('account_data.xlsx','Range','D1');
    BankDatabase.account_cnt = cell1{1};

    BankDatabase.account_data{current_index, 2} = password;
    writecell(BankDatabase.account_data, 'Data/account_data.xlsx');
end
```

S1.12: Record

We pass 4 parameters “account, operation, amount, account1” to the function, the meanings of the 4 parameters is similar to those of the 4 columns of the records_data.xlsx, which are explained in **S1.1.** We write them to the records cells in matlab, and we let the records_cnt plus one. Finally, we write them back to the excel file records_data.xlsx.

```
function Record(BankDatabase, account, operation, amount, account1)
    BankDatabase.user_data = readcell('user_data.xlsx','Range','A1:C1000');
    BankDatabase.account_data = readcell('account_data.xlsx','Range','A1:C1000');
    BankDatabase.records_data = readcell('records_data.xlsx','Range','A1:D1000');
    cell2 = readcell('user_data.xlsx','Range','D1');
    BankDatabase.user_cnt = cell2{1};
    cell1 = readcell('account_data.xlsx','Range','D1');
    BankDatabase.account_cnt = cell1{1};
    cell3 =  readcell('records_data.xlsx','Range','E1');
    BankDatabase.records_cnt = cell3{1};

    BankDatabase.records_cnt = BankDatabase.records_cnt + 1;
    BankDatabase.records_data{BankDatabase.records_cnt, 1} = account;
    BankDatabase.records_data{BankDatabase.records_cnt, 2} = operation;
    BankDatabase.records_data{BankDatabase.records_cnt, 3} = amount;
    BankDatabase.records_data{BankDatabase.records_cnt, 4} = account1;
     writecell(BankDatabase.records_data, 'Data/records_data.xlsx');
     writematrix(BankDatabase.records_cnt, 'Data/records_data.xlsx', 'Range', 'E1');
end
```

S1.13: Cancel account

We pass 1 parameter “current_index” to the function, “current_index” represents the index of current customer account,and we add 1000000 to the customer account to show that the account is cancelled. Finally, we write it back to the excel file account_data.xlsx.

```
function Cancel(BankDatabase,current_index)
    BankDatabase.user_data = readcell('user_data.xlsx','Range','A1:C1000');
    BankDatabase.account_data = readcell('account_data.xlsx','Range','A1:C1000');
    cell2 = readcell('user_data.xlsx','Range','D1');
    BankDatabase.user_cnt = cell2{1};
    cell1 = readcell('account_data.xlsx','Range','D1');
    BankDatabase.account_cnt = cell1{1};

    BankDatabase.account_data{current_index, 1} = BankDatabase.account_data{current_index, 1} + 1000000;
    writecell(BankDatabase.account_data, 'Data/account_data.xlsx');
end

function Freeze(BankDatabase,current_index)
```

S1.14:Freeze account

We pass 1 parameter “current_index” to the function, “current_index” represents the index of current customer account,and we add 2000000 to the customer account to show that the account is freezed. Finally, we write it back to the excel file account_data.xlsx.

```
function Freeze(BankDatabase,current_index)
    BankDatabase.user_data = readcell('user_data.xlsx','Range','A1:C1000');
    BankDatabase.account_data = readcell('account_data.xlsx','Range','A1:C1000');
    cell2 = readcell('user_data.xlsx','Range','D1');
    BankDatabase.user_cnt = cell2{1};
    cell1 = readcell('account_data.xlsx','Range','D1');
    BankDatabase.account_cnt = cell1{1};

    BankDatabase.account_data{current_index, 1} = BankDatabase.account_data{current_index, 1} + 2000000;
     writecell(BankDatabase.account_data, 'Data/account_data.xlsx');
end
```

S1.15: Unfreeze account

We pass 1 parameter “current_index” to the function, “current_index” represents the index of current customer account,and we reduce 2000000 to the customer account to show that the account is unfreezed. Finally, we write it back to the excel file account_data.xlsx.

```
function Unfreeze(BankDatabase,current_index)
    BankDatabase.user_data = readcell('user_data.xlsx','Range','A1:C1000');
    BankDatabase.account_data = readcell('account_data.xlsx','Range','A1:C1000');
    cell2 = readcell('user_data.xlsx','Range','D1');
    BankDatabase.user_cnt = cell2{1};
    cell1 = readcell('account_data.xlsx','Range','D1');
    BankDatabase.account_cnt = cell1{1};

    BankDatabase.account_data{current_index, 1} = BankDatabase.account_data{current_index, 1} - 2000000;
    writecell(BankDatabase.account_data, 'Data/account_data.xlsx');
end

function flag = CheckAccount(BankDatabase,index, account)
```

S1.16: Check account state

We pass 2 parameters“index, account” to the function, “index” represents the index of current customer account,“account” represents the valid customer account, and we compare current account with the valid “account”.

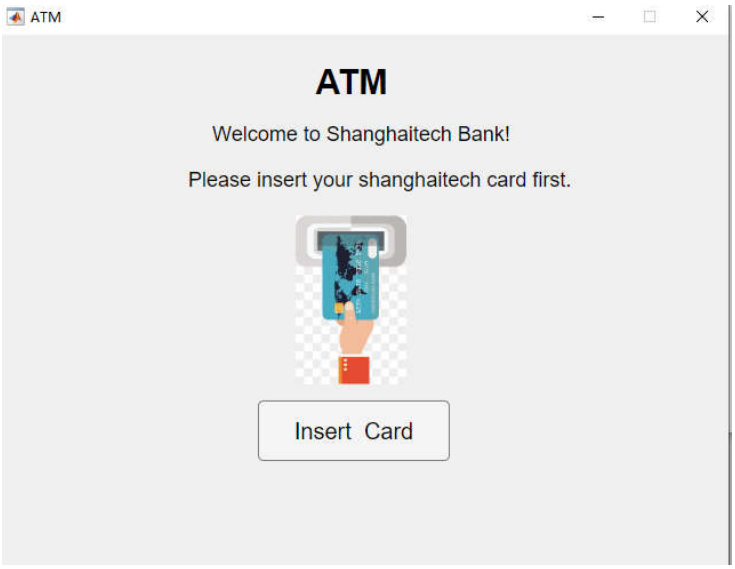
```
function flag = CheckAccount(BankDatabase,index, account)
    BankDatabase.user_data = readcell('user_data.xlsx','Range','A1:C1000');
    BankDatabase.account_data = readcell('account_data.xlsx','Range','A1:C1000');
    cell2 = readcell('user_data.xlsx','Range','D1');
    BankDatabase.user_cnt = cell2{1};
    cell1 = readcell('account_data.xlsx','Range','D1');
    BankDatabase.account_cnt = cell1{1};

    if BankDatabase.account_data{index, 1} == account
        flag = 1;
    else
        flag = 0;
    end

end
```

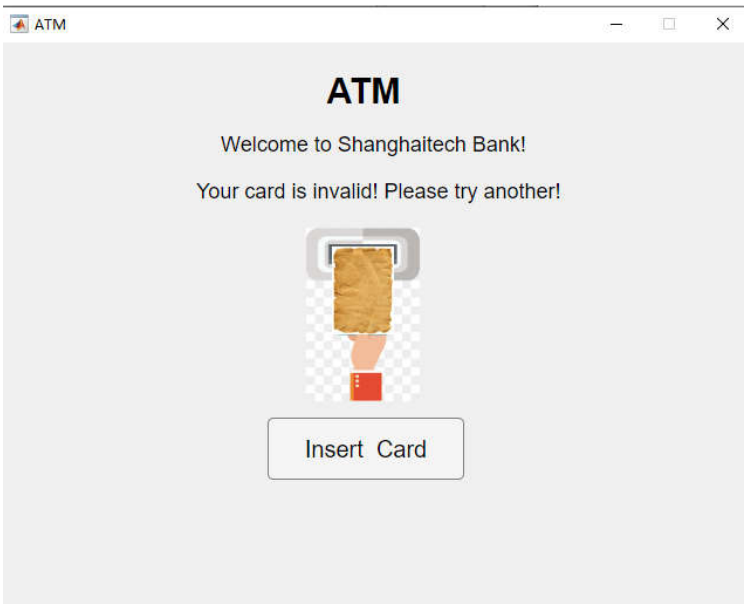
S2: ATM UI implementation

Note1: From S2 to S4, we will talk about UI implementation, there will be some code reuse in these 3 parts. So we may omit some specifications in S3,S4.
Note2: From S2 to S4, we will talk about UI implementation. We mostly update the UI figure by making some buttons, textareas and editfields invisible or visible. The codes about these may seemlong, but they are easy to understand.

S2.1: Insert card

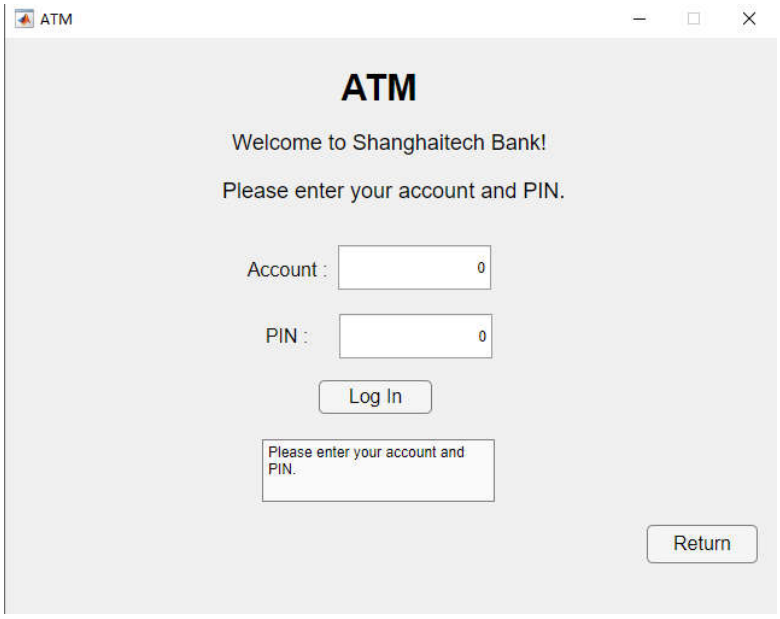
If the customer uses the valid card, after pressing insert button, he will enter the "verify account" window.
If the customer presses "n" on the keyboard, which means he uses the invalid card, after pressing insert button, he won't enter the "verify account" window and will get the "invalid card" information. He should press "c" to use the valid card.





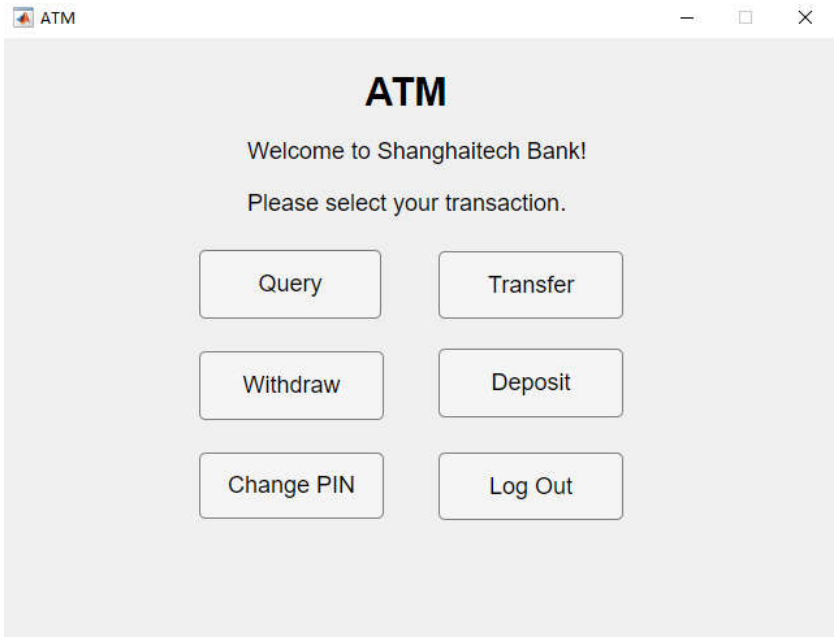
S2.2: Verify account information

The system will check the key-value pairs in the database. If the account and PIN are all correct and pairing successful, after pressing login button, the customer will log in.



S2.3: Enter ATM transaction menu

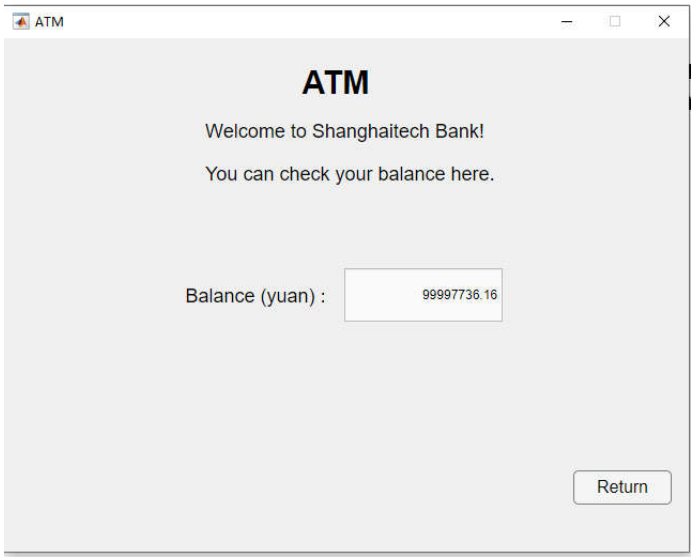
After logging in, the system will display the transaction menu for the customer to choose.



S2.4: Carry out ATM transactions

S2.4.1: ATM Query

After pressing query button in the transaction menu, the system will get data from the database and display the balance to the customer.



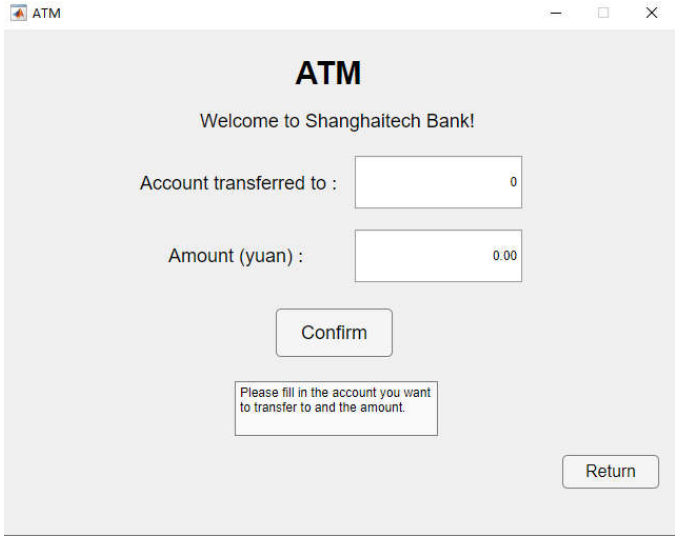
S2.4.2: ATM Transfer

After pressing transfer button in the transaction menu, the system willdisplay the transfer window to the customer.

The system will check if the account transferred to is valid and check if the balance is enough. If they are all valid, after pressing confirm button, the system will call the transfer() function in database to update the information in database.

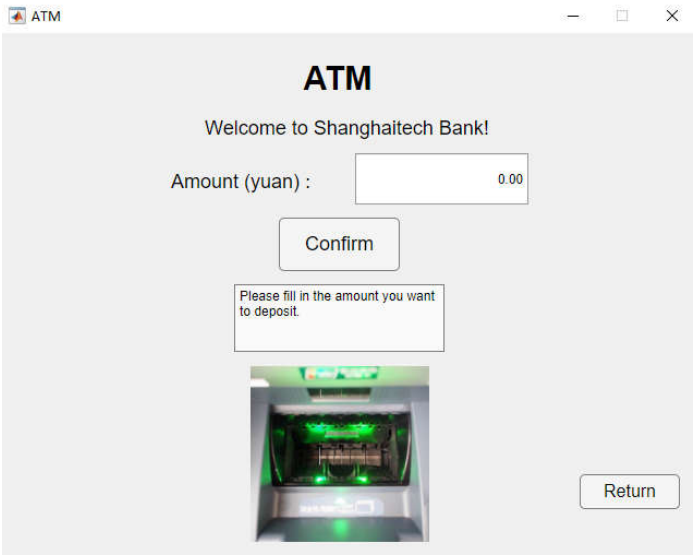
The textarea below the confirm button will tell the customer what to do.

If transferring successfully, the textarea will inform the customer and the editarea will be set non editable and the confirm button will be set unenabled.



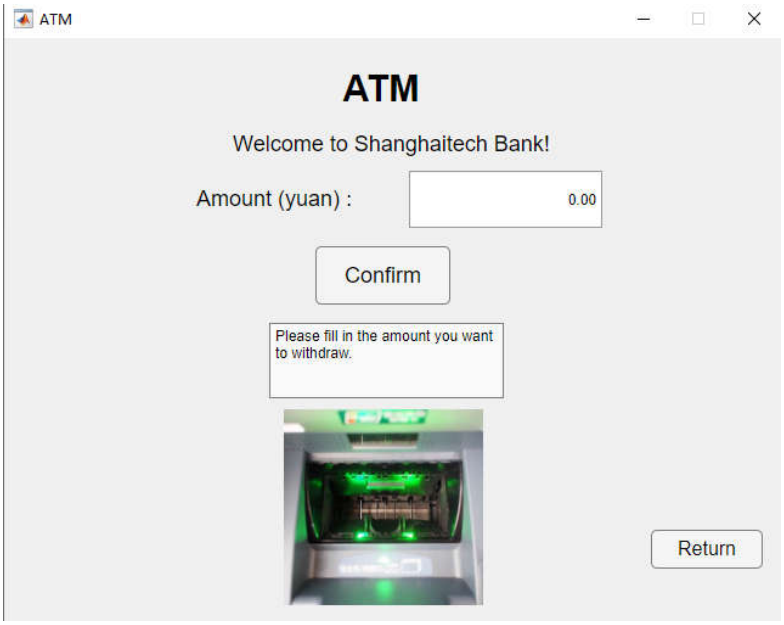
S2.4.3: ATM Deposit

After pressing deposit button in the transaction menu, the system will display the deposit window to the customer.
The system will check if the customer put money(press “m” on the keyboard) and check if the money is valid(pressing “n” on the keyboard simulates the customer put down invalid money) and check if the amount entered is multiples of 100. If they are all satisfied, after pressing confirm button, the system will call the deposit() function in database to update the information in database.
The textarea below the confirm button will tell the customer what to do.
If depositing successfully, the textarea will inform the customer and the confirm button will be set unenabled.



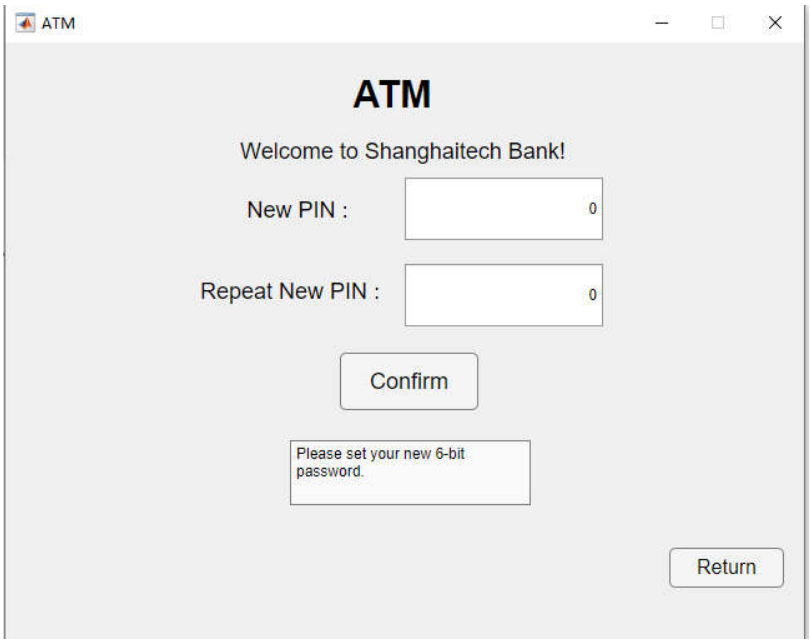
S2.4.4: ATM Withdraw

After pressing withdraw button in the transaction menu, the system will display the withdraw window to the customer.
The system will check if the amount entered is multiples of 100 and check if the balance is enough. If they are all satisfied, after pressing confirm button, the system will call the withdraw() function in database to update the information in database.
The textarea below the confirm button will tell the customer what to do.
If withdrawing successfully, the textarea will inform the customer and the confirm button will be set unenabled.
If customer presses the return button without taking away the money, the system won’t return and will inform the customer he don’t take away money.



S2.4.5: ATM Change account PIN

After pressingchangePIN button in the transaction menu, the system will display the changePIN window to the customer.
The system will check if the new PIN is 6 bits and check if the repeated new PIN is the same as new PIN. If they are all satisfied, after pressing confirm button, the system will call the changepassword() function in database to update the information in database.
The textarea below the confirm button will tell the customer what to do.
If changing successfully, the textarea will inform the customer and the editarea will be set non editableand the confirm button will be set unenabled.



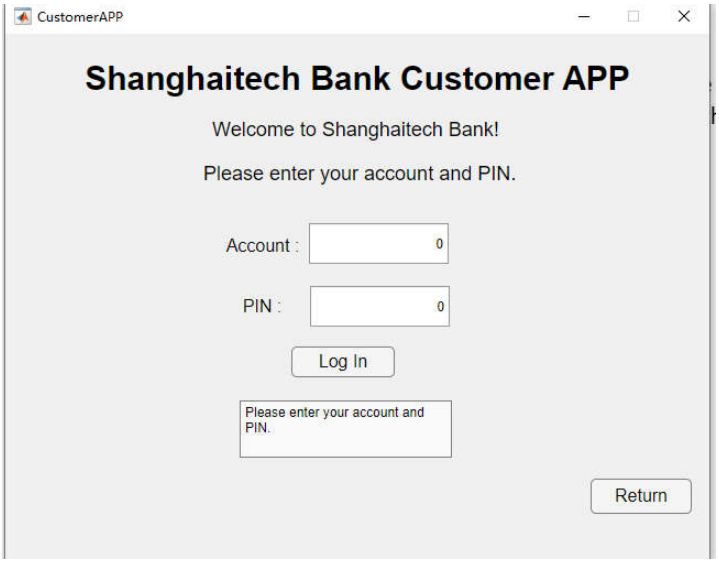
S2.4.6: ATM Logout

After pressing logout button in the transaction menu, the account will be logged out and the system will display the verify account window to the customer.

S3: Customer APP UI implementation

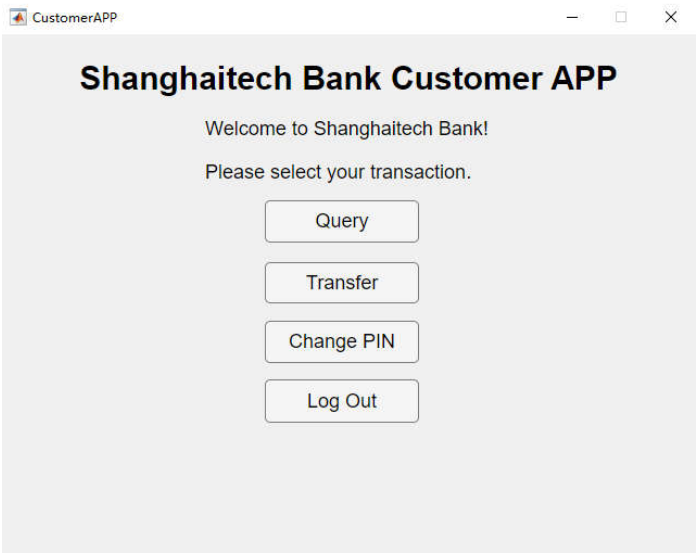
S3.1: Verify account information

It is the same as S2.2, please refer to **S2.2.**



S3.2: Enter customer APP transaction menu

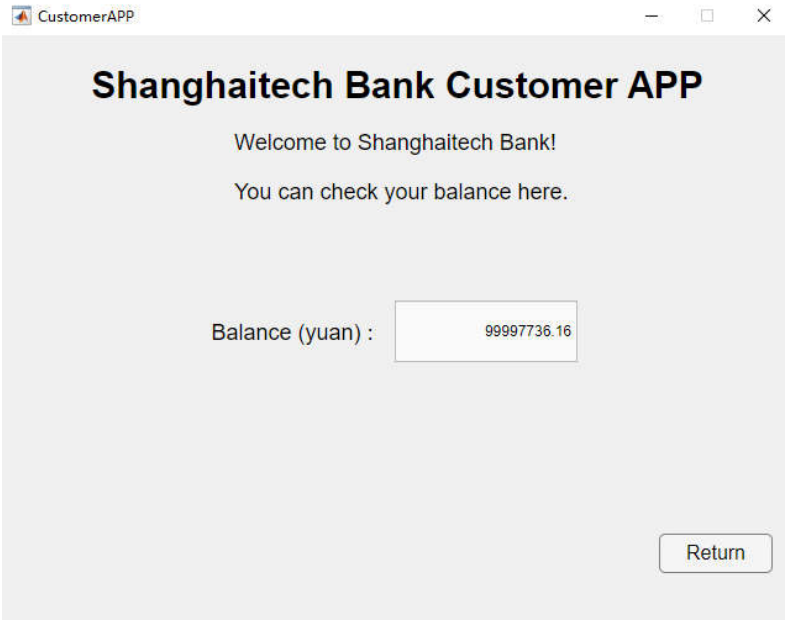
It is similar to S2.3, please refer to **S2.3**.



S3.3: Carry out customer APP transactions

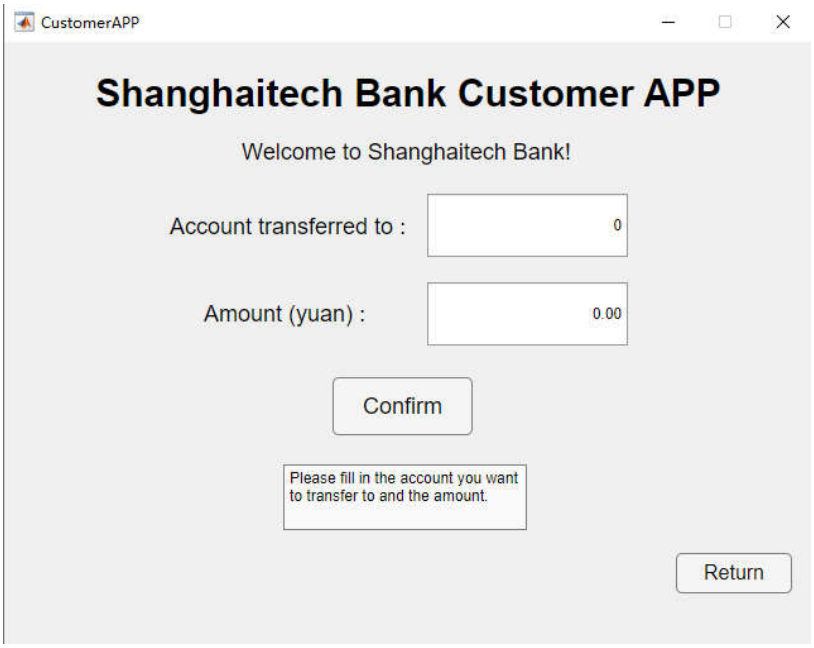
S3.3.1: Customer APP Query

It is the same as S2.4.1, please refer to **S2.4.1.**



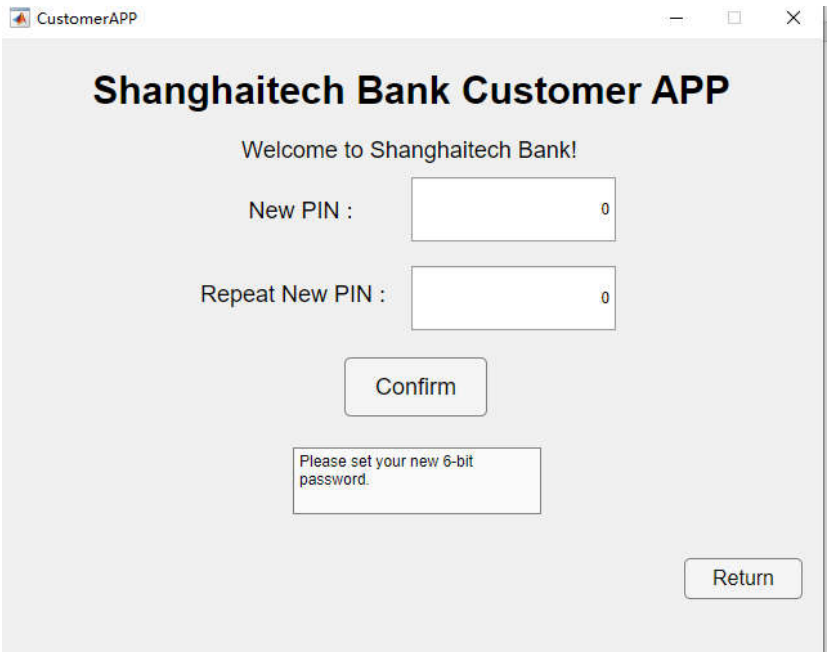
S3.3.2: Customer APP Transfer

It is the same as S2.4.2, please refer to **S2.4.2.**



S3.3.3: Customer APP Change account PIN

It is the same as S2.4.5, please refer to **S2.4.5.**



S3.3.4: Customer APP Logout

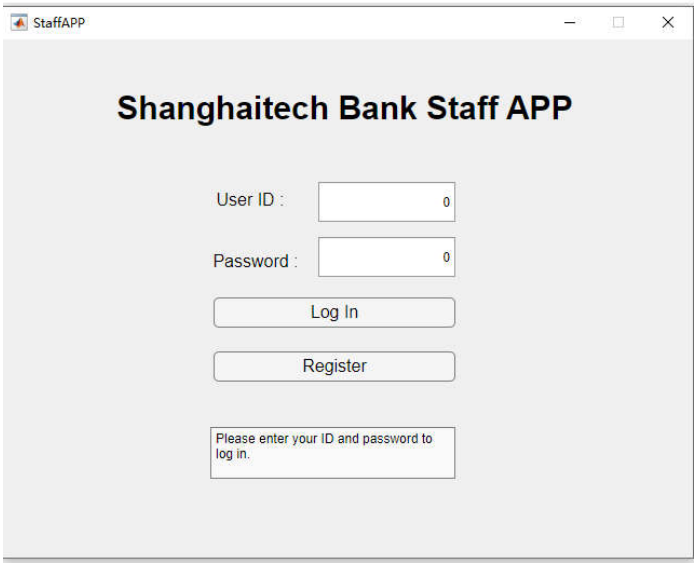
It is the same as S2.4.6, please refer to **S2.4.6.**

S4: Staff APP UI implementation

S4.1: Verify user information

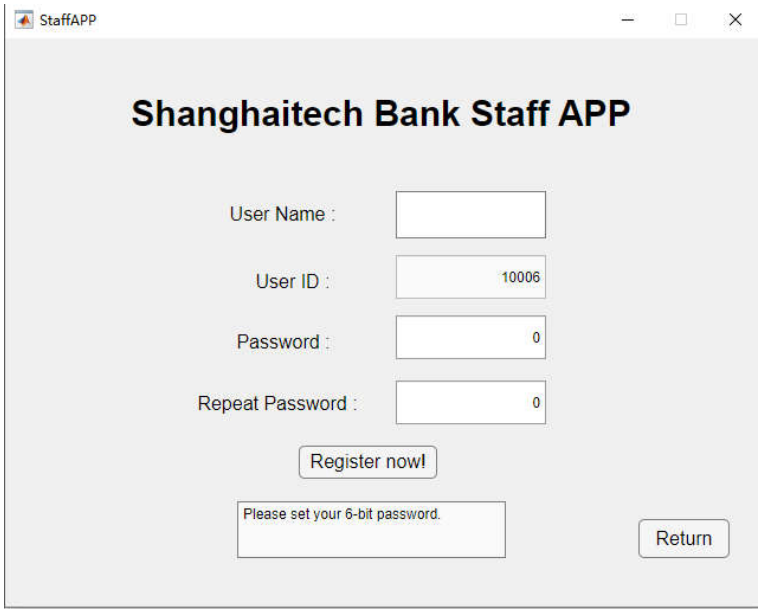
The system will check the key-value pairs in the database. If the 5 bits user account and user passwordare all correct and pairing successful, after pressing login button, the bank administrator will log in.

The textarea below the register button will tell the user what to do.



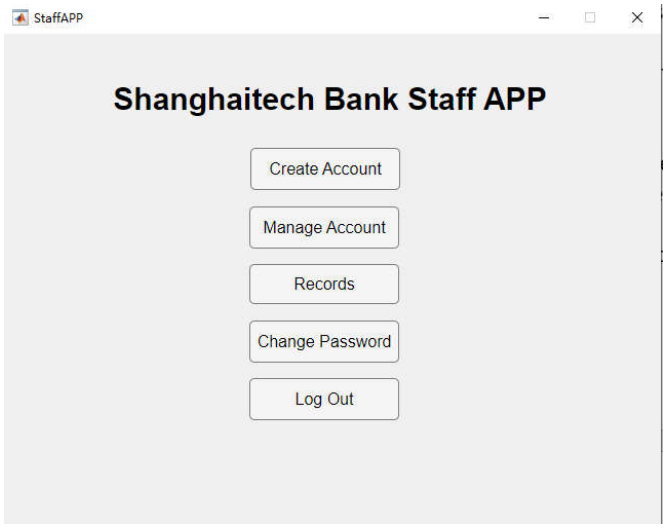
S4.2: Register user account

After pressing register button, the system will display the register window to the customer.
The database will distribute a 5 bitsuser ID to the user by calling function createuser() in database.
The system will check if the password is 6 bits and check if the repeated password is the same as the password. If they are all satisfied, after pressing register now button, the system will call the adduser() function in database to update the information in database.
The textarea below the register button will tell the user what to do.
If registering successfully, the textarea will inform the customer and the editarea will be set non editable and the register button will be set unenabled.



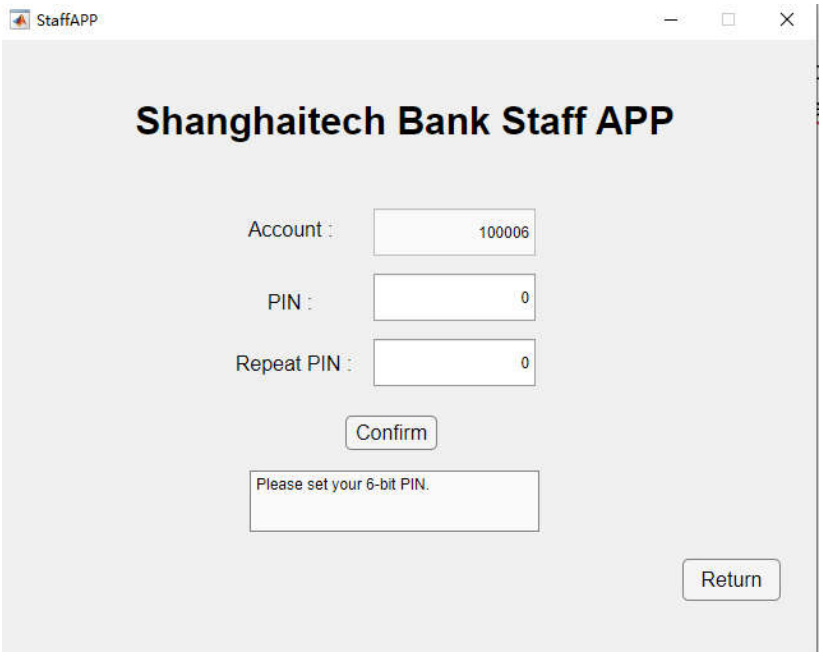
S4.3: Enter staff APP menu

It is similar to S2.3, please refer to **S2.3**.



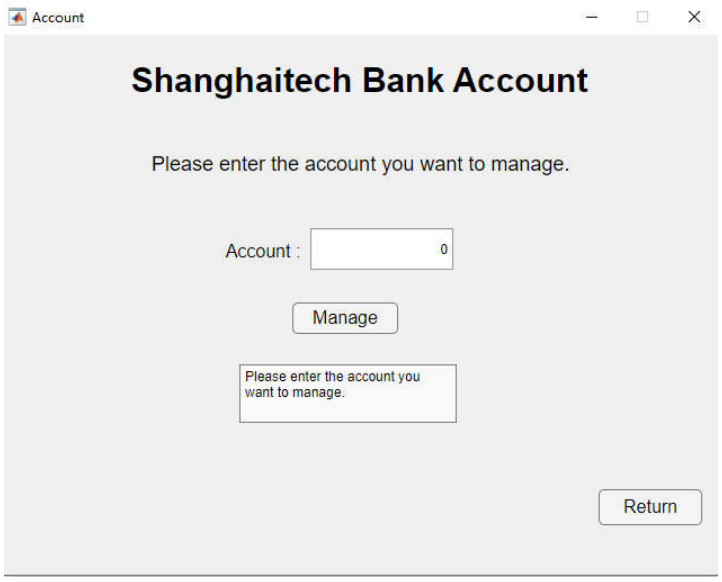
S4.4: Create account

After pressing create account button in the menu, the system will display thecreate account window to the customer.
The database will distribute a 6 bitsaccount ID by calling function createaccount() in database.
The system will check if the PIN is 6 bits and check if the repeated PIN is the same as the PIN. If they are all satisfied, after pressing confirm button, the system will call the addaccount() function in database to update the information in database.
The textarea below the confirm button will tell the user what to do.
If creating suucessfully, the textarea will inform the user and the editarea will be set non editable and the register button will be set unenabled.
After all the procedures, the bank staff will give the account information to the customer who comes to the bank counter. Then the customer can get his new account.



S4.5: Manage account

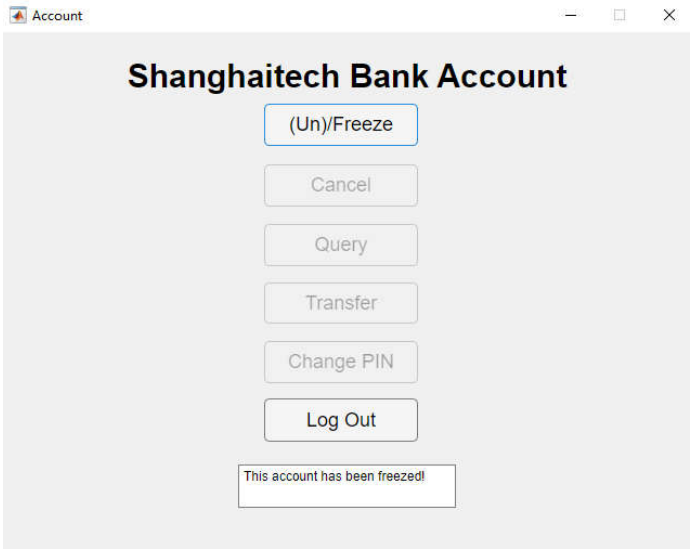
After pressing manage button in the menu, the system will display the account UIfigure to the bank administrator. Thebank administrator can enter the valid account ID to manage the accounts.



S4.5.1: Staff APP Freeze/Unfreeze account

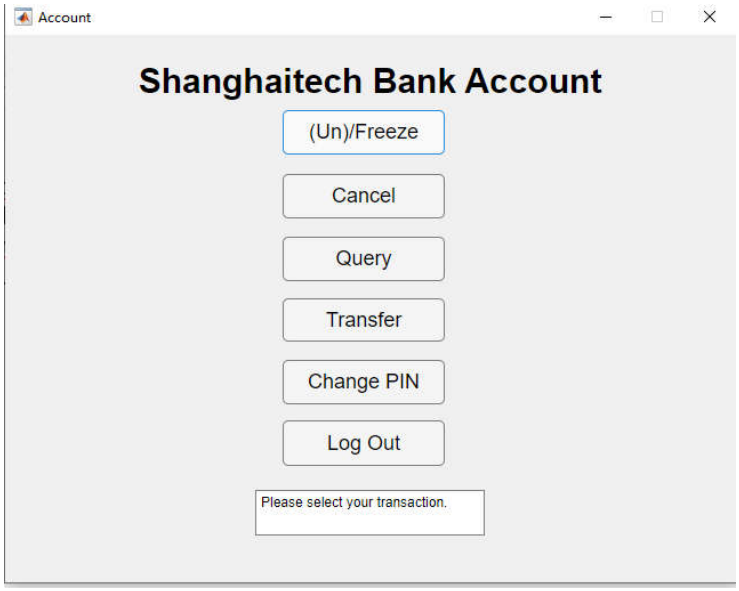
After pressing the freeze button, the system will call the freeze() function in the database and update the information in database. Then the account will be freezed.

All the other buttons are set unenabled after the account is freezed.



If the account is freezed, after pressing the freeze button, the system will call the unfreeze() function in the database and update the information in database. Then the account will be unfreezed.

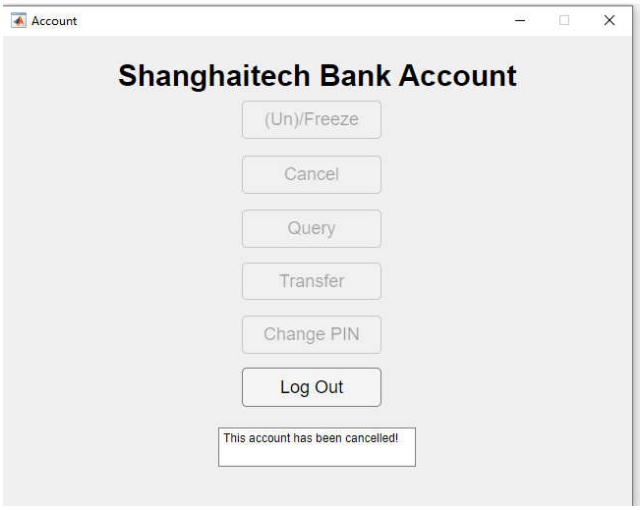
All the other buttons are set enabled after the account is unfreezed.



S4.5.2: Staff APP Cancel account

After pressing the cancel button, the system will call the cancel() function in the database and update the information in database. Then the account will be cancelled.

All the other buttons are set unenabled after the account is cancelled.



Note: From S4.5.3 to S4.5.6, they are the same as **S3.3.1—S3.3.4**. You can refer to them.

S4.5.3: Staff APP Query

S4.5.4: Staff APP Transfer

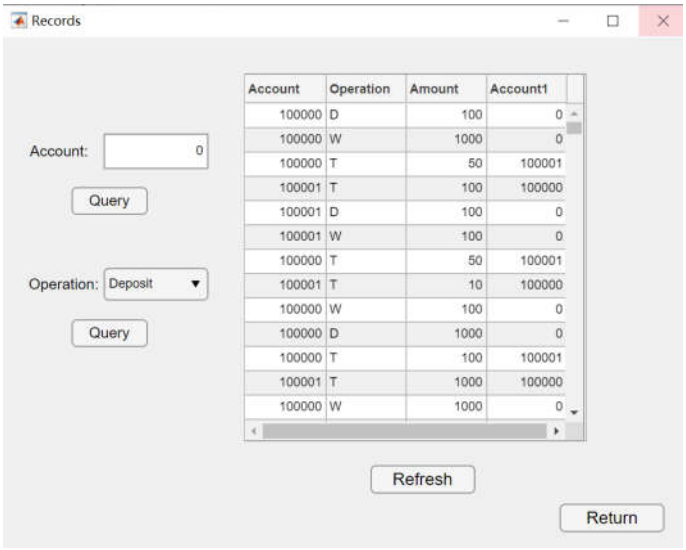
S4.5.5: Staff APPChange account PIN

S4.5.6: Staff APP Logout

S4.6: View Transaction records

After pressing records button in the menu, the records UIfigure will be displayed.

The user can find the specified account or find the specified operation.



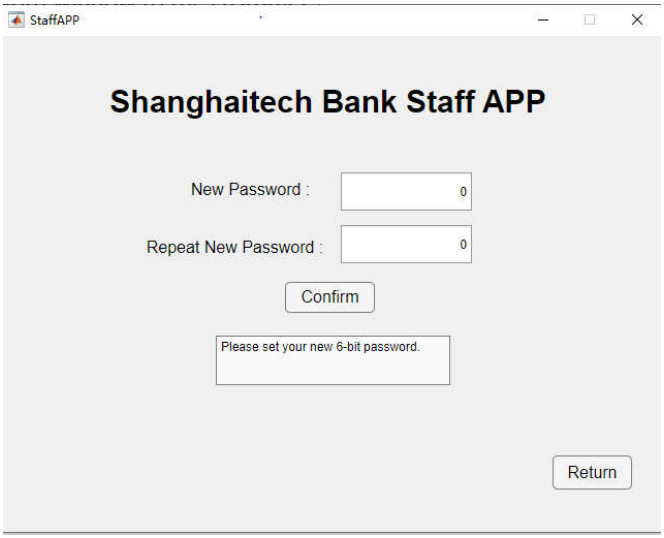
S4.7: Change user password

After pressing changeuser password button in the transaction menu, the system will display the changeuser passwordwindow to the customer.

The system will check if the new user password is 6 bits and check if the repeated new user password is the same as new user password. If they are all satisfied, after pressing confirm button, the system will call the changeuserpassword() function in database to update the information in database.

The textarea below the confirm button will tell the user what to do.

If changing successfully, the textarea will inform the user and the editarea will be set non editable and the confirm button will be set unenabled.



S4.8: Staff APP Logout

After pressing logout button in the menu, the account will be logged out and the system will display the verify user account window to the user.