Bank System Validation

System Architechture ____ 3

T1: Unit Test ____ 4

T1.1: CustomUI Unit Test ---- 4

T1.1.1 enqueue(app, ticket) 4

T1.1.2 result = dequeue(app) 4

T1.1.3 result = next(app, windowID) 5

T1.2: CashierUI Unit Test ---- 6

T1.2.1 result = checkPassword(app, password) 6

T1.3: ATMUI Unit Test ---- 7

T1.3.1 result = checkPassword(app, password) 7

T1.4: Account DB Unit Test ---- 8

T1.4.1 result = newID(app) 8

T1.4.2 newAccount(app,account,cause) 8

T1.4.3 result = queryInformation(app,accountID) 9

T1.4.4 result = changeBalance(app,account,amount,cause) 10

T1.4.5 [result,balance] = transfer(app,account,targetID,amount,cause) 10

T1.4.6 balance = deposit(app,account,amount,cause) 11

T1.4.7 [result,balance] = withdraw(app,account,amount,cause) 12

T1.4.8 resetPassword(app,account,newPassword,cause) 13

T1.4.9 result = deleteAccount(app,account,info,cause) 14

T2: Integration Test ____ 15

T2.1 ATM UI + Account DB Integration ---- 15

T2.1.1: Customer::Enter Account 15

T2.1.2: Customer::Transfer 17

T2.1.3: Customer::Deposit 19

T2.1.4: Customer::Withdraw 19

T2.1.5: Customer::Reset Password 20

T2.1.6: Customer::Query Information 20

T2.2 Cashier UI + Account DB Integration ---- 21

T2.2.1: Cashier::Enter Account 21

T2.2.2: Cashier::Transfer 23

T2.2.3: Cashier::Deposit 25

T2.2.4: Cashier::Withdraw 25

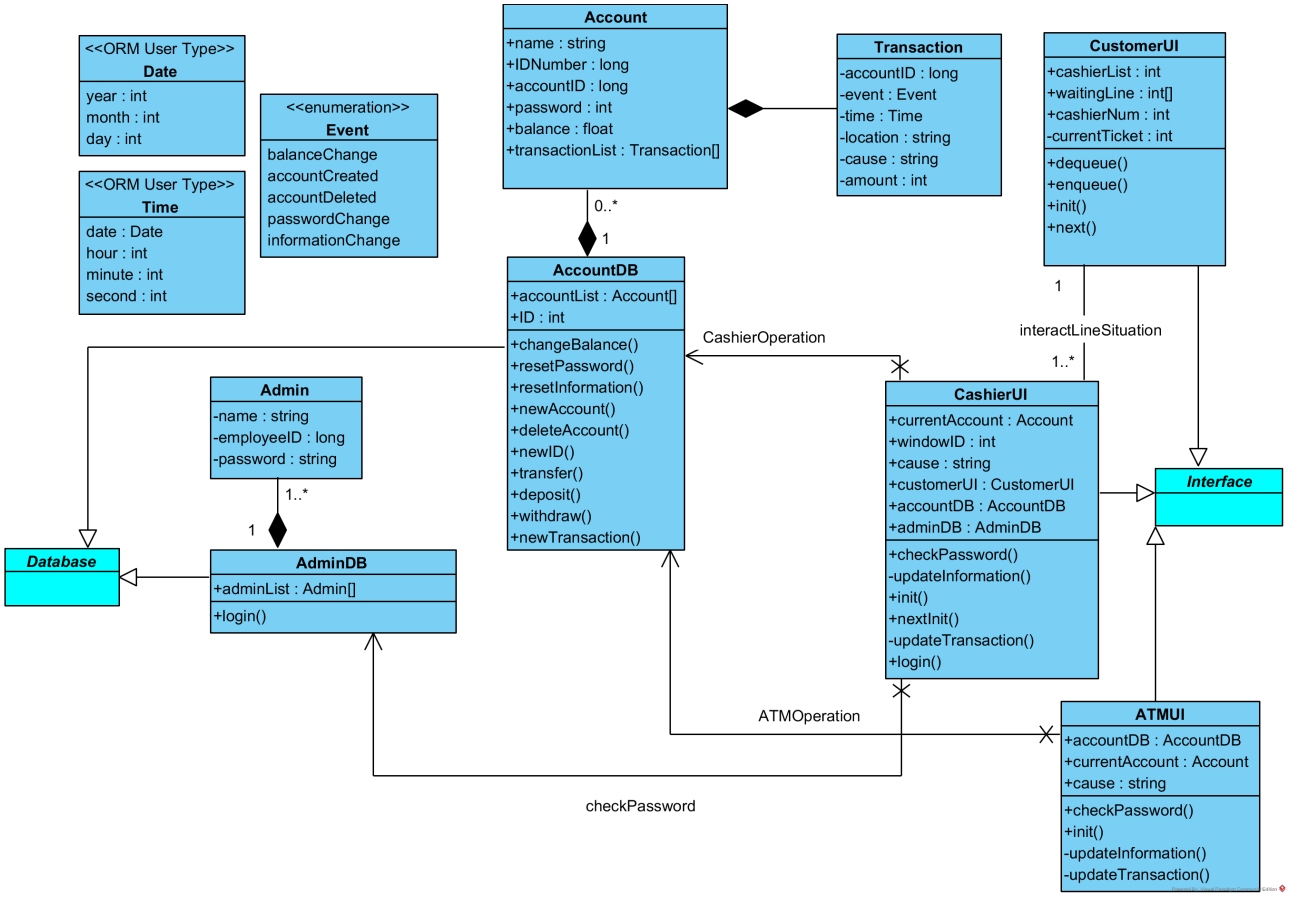
T2.2.5: Cashier::Reset Password 26

T2.2.6: Cashier::Query Information 26

T2.2.7: Cashier::Create Account 27

T2.2.8: Cashier::Reset Information 27
T2.2.9: Cashier::Close Account 28
T3: Functional Test ____29
T3.1: CustomUI Test ----29
T3.1.1: Use Case “Get ticket” 29
T3.1.2: Use Case “Get ticket” then “Cancel” 29
T3.2: ATM UI Test ----30
T3.2.1: Use Case “Enter Account” 30
T3.2.2: Use Case “Transfer” 32
T3.2.3: Use Case “Deposit” 35
T3.2.4: Use Case “Withdraw” 36
T3.2.5: Use Case “Reset Password” 38
T3.2.6: Use Case “Query Information” 40
T3.3: CashierUI Test ----41
T3.3.1: Use Case “Enter Account” 41
T3.3.2: Use Case “Transfer” 43
T3.3.3: Use Case “Deposit” 46
T3.3.4: Use Case “Withdraw” 47
T3.3.5: Use Case “Reset Password” 49
T3.3.6: Use Case “Query Information” 51
T3.3.7: Use Case “Create Account” 51
T3.3.8: Use Case “Reset Information” 54
T3.3.9: Use Case “Close Account” 55

System Architechture



T1: Unit Test
T1.1: CustomUI Unit Test

T1.1.1 enqueue(app, ticket)

```
function enqueue(app, ticket)
    for i = 1:app.cashierNum
        if (isempty(app.UITable.Data{i,2}))
            app.UITable.Data{i,2} = ticket;
            app.cashierList(i).WaitingLabel.Text = "Waiting: "+num2str(ticket);
            return;
        end
    end
    app.waitingLine = [app.waitingLine; ticket];
end
```

Coverage Criteria: Branch coverage

Test Case T1.1.1.1	
Coverage Item	Tcover 1.1.1.1
Input	add one 'ticket number'(etc. 2)
State	Waiting line is empty
Expected Output	the ticket number should be enqueued to the waitng line

Test coverage: 1/1=100%
Test result: 1 passed

T1.1.2 result = dequeue(app)

```
function result = dequeue(app)
    if (isempty(app.waitingLine))
        result = -1;
    else
        result = app.waitingLine{1};
        app.waitingLine(1) = [];
    end
end
```

Coverage Criteria: Branch coverage

Test Case T1.1.1.2	
Coverage Item	Tcover 1.1.1.2
Input	Empty
State	one member in waitingLine
Expected Output	the ticket should the dequeued ticket from the waitng line ket number should be enqueued to the waitng line

Test coverage: 1/1=100%
Test result: 1 passed

T1.1.3 result = next(app, windowID)

```
function result = next(app, windowID)
    app.UITable.Data{windowID,1} = app.UITable.Data{windowID,2};
    result = app.UITable.Data{windowID,1};
    tmp = app.dequeue();
    if (tmp ~= -1)
        app.UITable.Data{windowID,2} = tmp;
    else
        app.UITable.Data{windowID,2} = '';
        app.cashierList(windowID).CurrentLabel.Text = "Current: "+app.UITable.Data{windowID,1};
        app.cashierList(windowID).WaitingLabel.Text = "Waiting: ";
        return;
    end
    if (isempty(app.UITable.Data{windowID,1}))
        tmp = app.dequeue();
        app.UITable.Data{windowID,1} = app.UITable.Data{windowID,2};
        if (tmp ~= -1)
            app.UITable.Data{windowID,2} = tmp;
        else
            app.UITable.Data{windowID,2} = '';
        end
        result = app.UITable.Data{windowID,1};
    end
    app.cashierList(windowID).CurrentLabel.Text = "Current: "+app.UITable.Data{windowID,1};
    app.cashierList(windowID).WaitingLabel.Text = "Waiting: "+app.UITable.Data{windowID,2};
end
```

Coverage Criteria: Branch coverage

Test Case T1.1.1.3	
Coverage Item	Tcover 1.1.1.3
Input	window ID
State	one member in waitingLine
Expected Output	the next customer ticket number of the window

Test coverage: 1/1=100%
Test result: 1 passed

T1.2: CashierUI Unit Test

T1.2.1 result = checkPassword(app, password)

```
function result = checkPassword(app, password)
    result = (password == app.currentAccount.password);
end
```

Coverage Criteria: Branch coverage

Test Case T1.2.1.1	
Coverage Item	Tcover 1.2.1.1
Input	`'1'`
State	`An account with password: '1'`
Expected Output	`1('true')`

Test Case T1.2.1.2	
Coverage Item	Tcover 1.2.1.1
Input	`'0'`
State	`An account with password: '1'`
Expected Output	`0('false')`

Test coverage: 2/2=100%
Test result: 2 passed

T1.3: ATMUI Unit Test

T1.3.1 result = checkPassword(app, password)

```
function result = checkPassword(app, password)
    result = (password == app.currentAccount.password);
end
```

Coverage Criteria: Branch coverage

Test Case T1.2.1.1	
Coverage Item	Tcover 1.2.1.2
Input	`'911'`
State	`An account with password: '911'`
Expected Output	`[true true true]`

Test Case T1.2.1.2	
Coverage Item	Tcover 1.2.1.2
Input	`’120'`
State	`An account with password: '911'`
Expected Output	`[false false false]`

Test coverage: 2/2=100%
Test result: 2 passed

T1.4: Account DB Unit Test

T1.4.1 result = newID(app)

```
function result = newID(app)
    result = sprintf("%d",app.ID);
    app.ID = app.ID + 1;
end
```

Coverage Criteria: Branch coverage

Test Case T1.4.1.1	
Coverage Item	Tcover 1.4.1.1
Input	`Input: ID = 10`
State	`State: testCase.ID empty`
Expected Output	`Expected Output: ID + 1`

Test coverage: 1/1=100%
Test result: 1 passed

T1.4.2 newAccount(app,account,cause)

```
function newAccount(app,account,cause)
    app.accountList=[app.accountList;account];
    app.newTransaction(account,Event.accountCreate,0,cause);
end
```

Coverage Criteria: Branch coverage

Test Case T1.4.2.1	
Coverage Item	Tcover 1.4.2.1
Input	`newAccount, cause:'1'`
State	`accountList win one member newAccount`
Expected Output	`[newAccount, newAccount]`

Test coverage: 1/1=100%
Test result: 1 passed

T1.4.3 result = queryInformation(app,accountID)

```
function result = queryInformation(app,accountID)
    for i=1:length(app.accountList)
        if (app.accountList(i).accountID == accountID)
            result = app.accountList(i);
            return;
        end
    end
    result = -1;
    return;
end
```

Coverage Criteria: Branch coverage

Test Case T1.4.3.1	
Coverage Item	Tcover 1.4.3.1
Input	`account_id`
State	`accountList with one account(newAccount) whose accountID is account_id`
Expected Output	`newAccount`

Test Case T1.4.3.2	
Coverage Item	Tcover 1.4.3.2
Input	`account_id + 1`
State	`accountList with one account(newAccount) whose accountID is account_id`
Expected Output	`-1`

Test coverage: 2/2=100%
Test result: 2 passed

T1.4.4 result = changeBalance(app,account,amount,cause)

```
function result = changeBalance(app,account,amount,cause)
    account.balance = account.balance + amount;
    result = account.balance;
    app.newTransaction(account,Event.balanceChange,amount,cause);
end
```

Coverage Criteria: Branch coverage

Test Case T1.4.4.1	
Coverage Item	Tcover 1.4.4.1
Input	`amount, 'cause'`
State	`newAccount with balance 'originalBL'`
Expected Output	`originalBL + amount`

Test coverage: 1/1=100%
Test result: 1 passed

T1.4.5 [result,balance] = transfer(app,account,targetID,amount,cause)

```
function [result,balance] = transfer(app,account,targetID,amount,cause)
    result = app.queryInformation(targetID);
    balance = 0;
    if (result ~= -1)
        target = result;
        if (account.balance - amount < 0)
            result = -2;
            balance = account.balance;
            return;
        end
        result = 0;
        balance = app.changeBalance(account,-amount,cause);
        app.changeBalance(target,amount,cause);
    end
end
```

Coverage Criteria: Branch coverage

Test Case T1.4.5.1	
Coverage Item	Tcover 1.4.5.1
Input	`newAccount, targetID, amount,'cause'`
State	`currAccount with sufficient balance, target account exits.`
Expected Output	`0;originalBL - amount`

Test Case T1.4.5.2	
Coverage Item	Tcover 1.4.5.1
Input	`newAccount, targetID, amount,'cause'`
State	`currAccount with sufficient balance, target account not exits.`
Expected Output	`-1; 0`

Test Case T1.4.5.3	
Coverage Item	Tcover 1.4.5.1
Input	`newAccount, targetID, amount,'cause'`
State	`currAccount with infficient balance, target accountexits.`
Expected Output	`-2; originalBL`

Test coverage: 3/3=100%
Test result: 3 passed

T1.4.6 balance = deposit(app,account,amount,cause)

```
function balance = deposit(app,account,amount,cause)
    balance = app.changeBalance(account,amount,cause);
end
```

Coverage Criteria: Branch coverage

Test Case T1.4.6.1	
Coverage Item	Tcover 1.4.6.1
Input	`newAccount, amount, 'cause'`
State	`newAccount with originalBL`
Expected Output	`originalBL + amount`

Test Case T1.4.6.1	
Coverage Item	Tcover 1.4.6.1
Input	`newAccount, minor_amount, 'cause'`
State	`newAccount with originalBL`
Expected Output	`-1`

Test coverage: 2/2=100%
Test result: 1 passed, 1 failed (the input was controlled by UI Text Field, but not the function)

T1.4.7 [result,balance] = withdraw(app,account,amount,cause)

```
function [result,balance] = withdraw(app,account,amount,cause)
    if (account.balance - amount < 0)
        result = -1;
        balance = account.balance;
        return;
    end
    result = 0;
    balance = app.changeBalance(account,-amount,cause);
end
```

Coverage Criteria: Branch coverage

Test Case T1.4.7.1	
Coverage Item	Tcover 1.4.7.1
Input	`newAccount, amount,'cause'`
State	`newAccount with sufficient BL`
Expected Output	`0; originalBL - amount`

Test Case T1.4.7.2	
Coverage Item	Tcover 1.4.7.1
Input	`newAccount, bigger amount,'cause'`
State	`newAccount with sufficient BL`
Expected Output	`-1, originalBL;`

Test Case T1.4.7.3	
Coverage Item	Tcover 1.4.7.1
Input	`newAccount, minus amount,'cause'`
State	`newAccount with sufficient BL`
Expected Output	`-1, originalBL;`

Test coverage: 3/3=100%
Test result: 2 passed, 1failed (the input was controlled by UI Text Field, but not the function)

T1.4.8 resetPassword(app,account,newPassword,cause)

```
function resetPassword(app,account,newPassword,cause)
    account.password = newPassword;
    app.newTransaction(account,Event.passwordChange,0,cause);
end
```

Coverage Criteria: Branch coverage

Test Case T1.4.8.1	
Coverage Item	Tcover 1.4.8.1
Input	`911`
State	`An account with original password '110'`
Expected Output	`The later password is equal to input`

Test coverage: 1/1=100%
Test result: 1 passed

T1.4.9 result = deleteAccount(app,account,info,cause)

Coverage Branch

```
function result = deleteAccount(app,account,info,cause)
    result = 0;
    if (account.name ~= info.name)
        result = -1;
        return;
    end
    if (account.IDNumber ~= info.IDNumber)
        result = -2;
        return;
    end
    for i=1:length(app.accountList)
        if (app.accountList(i) == account)
            app.accountList(i) = [];
            app.newTransaction(account,Event.accountDelete,0,cause);
            return;
        end
    end
end
```

Criteria: coverage

Test Case T1.4.9.1	
Coverage Item	Tcover 1.4.1.1
Input	`newAccount, input, 'cause'`
State	`The account list has only account with name '996', IDnumber'991'`
Expected Output	`0`

Test Case T1.4.9.2	
Coverage Item	Tcover 1.4.1.1
Input	`Am account with name '997', ID '991', input, 'cause'`
State	`The account list has only account with name '996', IDnumber'991'`
Expected Output	`-1`

Test Case T1.4.9.3	
Coverage Item	Tcover 1.4.1.1
Input	`Am account with name '996', ID '110', input, 'cause'`
State	`The account list has only account with name '996', IDnumber'991'`
Expected Output	`-2`

Test coverage: 3/3=100%
Test result: 3 passed

T2: Integration Test
T2.1 ATM UI + Account DB Integration

T2.1.1: Customer::Enter Account

Test Case T2.1.1.1	
Coverage Item	Normal Input
Input	1. Enter account ID: 0 2. Click “Account ID Confirm ” button 3. Enter account password: 0 4. Click “Password confirm” button
State	The account database has only one account with ID ‘0’, password ‘0’
Expected Output	`testCase.atmUI.currentAccount == testCase.account`

Test Case T2.1.1.2	
Coverage Item	Error Input
Input	1. Enter account ID: 0 2. Click “Account ID Confirm ” button 3. Enter account password: 1 4. Click “Password confirm” button
State	The account database has only one account with ID ‘0’, password ‘0’
Expected Output	`testCase.atmUI.Message.Value{1}=='Wrong Password.'`

Test Case T2.1.1.3	
Coverage Item	Error Input
Input	1. Enter account ID: 1 2. Click “Account ID Confirm ” button
State	The account database has only one account with ID ‘0’, password ‘0’
Expected Output	`testCase.atmUI.Message.Value{1}=='Fail to identify.'`

Test Case T2.1.1.4	
Coverage Item	Error Input & Normal Input
Input	1. Enter account ID: 0 2. Click “Account ID Confirm ” button 3. Enter account password: 1 4. Click “Password confirm” button 5. Enter account password: 0 6. Click “Password confirm” button
State	The account database has only one account with ID ‘0’, password ‘0’
Expected Output	`1. testCase.atmUI.Message.Value{1}=='Wrong Password.'` `2. testCase.atmUI.currentAccount == testCase.account`

Test Case T2.1.1.5	
Coverage Item	Error Input & Normal Input
Input	1. Enter account ID: 1 2. Click “Account ID Confirm ” button 3. Enter account ID: 0 4. Click “Account ID Confirm ” button 5. Enter account password: 0 6. Click “Password confirm” button
State	The account database has only one account with ID ‘0’, password ‘0’
Expected Output	`1. testCase.atmUI.Message.Value{1}=='Fail to identify.'` `2. testCase.atmUI.currentAccount == testCase.account`

Test result: 5/5 passed

T2.1.2: Customer::Transfer

1. Coverage Criteria:
2. Test case

Test Case T2.1.2.1	
Coverage Item	Normal Input
Input	1. Transfer:: Target Account: 1 2. Amount: 50 3. Click ‘Confirm’ button
State	The account database has only two accounts: 1. Account 1{name, IDNumber, accountID, password, balance} = {0,0,0,0,100} 2. Account 2{name, IDNumber, accountID, password, balance} = {1,1,1,1,50}
Expected Output	`testCase.atmUI.Message.Value{1},'Success. Your balance is 50. '` Account 1: 1. Balance: 50 2. In Account 1 ATM UI: 1) atmUI.QueryTransactionUITable.Data{1,1} == "Balance Change” 2) atmUI.QueryTransactionUITable.Data{1,3} == “ATM" 3) atmUI.QueryTransactionUITable.Data{1,4} == “-50” Account 2: 1. Balance: 100 2. In Account 2 ATM UI: 1) atmUI.QueryTransactionUITable.Data{1,1} == "Balance Change” 2) atmUI.QueryTransactionUITable.Data{1,3} == “ATM" 3) atmUI.QueryTransactionUITable.Data{1,4} == “50"

Test Case T2.1.2.2	
Coverage Item	Error Input & Normal Input
Input	1. Transfer:: Target Account: 0 2. Amount: 50 3. Click ‘Confirm’ button 4. Transfer:: Target Account: 1 5. Amount: 50 6. Click ‘Confirm’ button
State	The account database has only two accounts: 1. Account 1{name, IDNumber, accountID, password, balance} = {0,0,0,0,100} 2. Account 2{name, IDNumber, accountID, password, balance} = {1,1,1,1,50}

Expected Output	`testCase.atmUI.Message.Value{1}=='Do not input your account.'`
	`testCase.atmUI.Message.Value{1}=='Success. Your balance is 50. '`
	
	Account 1:
	1. Balance: 50
	2. In Account 1 ATM UI:
	1) atmUI.QueryTransactionUITable.Data{1,1} == "Balance Change”
	2) atmUI.QueryTransactionUITable.Data{1,3} == “ATM"
	3) atmUI.QueryTransactionUITable.Data{1,4} == “-50”
	Account 2:
	1. Balance: 100
	2. In Account 2 ATM UI:
	1) atmUI.QueryTransactionUITable.Data{1,1} == "Balance Change”
	2) atmUI.QueryTransactionUITable.Data{1,3} == “ATM"
	3) atmUI.QueryTransactionUITable.Data{1,4} == “50"

Test Case T2.1.2.3	
Coverage Item	Error Input & Normal Input
Input	1. Transfer:: Target Account: 1 2. Amount: 150 3. Click ‘Confirm’ button
State	The account database has only two accounts: 1. Account 1{name, IDNumber, accountID, password, balance} = {0,0,0,0,100} 2. Account 2{name, IDNumber, accountID, password, balance} = {1,1,1,1,50}
Expected Output	`testCase.atmUI.Message.Value{1} == 'Insufficient Balance. Your balance is 100. '`

Test result: 3/3 passed

T2.1.3: Customer::Deposit

Test Case T2.1.3.1	
Coverage Item	Normal Input
Input	1. Deposit::Amount: 50 2. Click ‘Confirm’ button
State	The account database has only account: 1. Account 1{name, IDNumber, accountID, password, balance} = {0,0,0,0,100}
Expected Output	`testCase.atmUI.Message.Value{1}=='Success. Your balance is 150. '` Account 1: 1. Balance: 150 2. In Account 1 ATM UI: 1) atmUI.QueryTransactionUITable.Data{1,1} == "Balance Change” 2) atmUI.QueryTransactionUITable.Data{1,3} == “ATM" 3) atmUI.QueryTransactionUITable.Data{1,4} == “150”

Test result: 1/1 passed

T2.1.4: Customer::Withdraw

Test Case T2.1.4.1	
Coverage Item	Normal Input
Input	1. Withdraw::Amount: 50 2. Click ‘Confirm’ button
State	The account database has only account: 1. Account 1{name, IDNumber, accountID, password, balance} = {0,0,0,0,100}
Expected Output	`testCase.atmUI.Message.Value{1}=='Success. Your balance is 50. '` Account 1: 1. Balance: 50 2. In Account 1 ATM UI: 1) atmUI.QueryTransactionUITable.Data{1,1} == "Balance Change” 2) atmUI.QueryTransactionUITable.Data{1,3} == “ATM" 3) atmUI.QueryTransactionUITable.Data{1,4} == “-50”

Test result: 1/1 passed

T2.1.5: Customer::Reset Password

Test Case T2.1.5.1	
Coverage Item	Normal Input
Input	1. New Passwords: “1” 2. Confirm Passwords: “1”
State	The account database has only account: 1. Account 1{name, IDNumber, accountID, password, balance} = {0,0,0,0,100}

Expected Output	`testCase.atmUI.Message.Value{1}=='Success.'`
	
	`testCase.atmUI.currentAccount.password == "1"`

Test result: 1/1 passed

T2.1.6: Customer::Query Information

Test Case T2.1.6.1	
Coverage Item	Normal Input
Input	
State	The account database has only account: 1. Account 1{name, IDNumber, accountID, password, balance} = {0,0,0,0,100}
Expected Output	1. `testCase.atmUI.currentAccount.name == convertCharsToStrings(testCase.atmUI.QueryNameEditField.Value)` 2. `testCase.atmUI.currentAccount.IDNumber == convertCharsToStrings(testCase.atmUI.QueryIDNumberEditField.Value)` 3. `testCase.atmUI.currentAccount.accountID == convertCharsToStrings(testCase.atmUI.QueryAccountIDEditField.Value)` 4. Convert 1) `cmp = str2double(regexp(testCase.atmUI.QueryBalanceEditField.Value,'\d*','match'))` 2) `testCase.atmUI.currentAccount.balance == cmp`

T2.2 Cashier UI + Account DB Integration

T2.2.1: Cashier::Enter Account

Test Case T2.2.1.1	
Coverage Item	Normal Input
Input	1. Enter account ID: 0 2. Click “Account ID Confirm ” button 3. Enter account password: 0 4. Click “Password confirm” button
State	The account database has only one account with ID ‘0’, password ‘0’
Expected Output	`testCase.cashierUI1.currentAccount == testCase.account`

Test Case T2.1.1.2	
Coverage Item	Error Input
Input	1. Enter account ID: 0 2. Click “Account ID Confirm ” button 3. Enter account password: 1 4. Click “Password confirm” button
State	The account database has only one account with ID ‘0’, password ‘0’
Expected Output	`testCase.cashierUI1.Message.Value{1}==‘Wrong Password.'`

Test Case T2.2.1.3	
Coverage Item	Error Input
Input	1. Enter account ID: 1 2. Click “Account ID Confirm ” button
State	The account database has only one account with ID ‘0’, password ‘0’
Expected Output	`testCase.cashierUI1.Message.Value{1}=='Fail to identify.'`

Test Case T2.2.1.4	
Coverage Item	Error Input & Normal Input
Input	1. Enter account ID: 0 2. Click “Account ID Confirm ” button 3. Enter account password: 1 4. Click “Password confirm” button 5. Enter account password: 0 6. Click “Password confirm” button
State	The account database has only one account with ID ‘0’, password ‘0’
Expected Output	`1. testCase.cashierUI1.Message.Value{1}==‘Wrong Password.’` `2. testCase.cashierUI1.currentAccount == testCase.account`

Test Case T2.2.1.5	
Coverage Item	Error Input & Normal Input
Input	1. Enter account ID: 1 2. Click “Account ID Confirm ” button 3. Enter account ID: 0 4. Click “Account ID Confirm ” button 5. Enter account password: 0 6. Click “Password confirm” button
State	The account database has only one account with ID ‘0’, password ‘0’
Expected Output	`1. testCase.cashierUI1.Message.Value{1}==‘Fail to identify.’` `2. testCase.cashierUI1.currentAccount == testCase.account`

Test result: 5/5 passed

T2.2.2: Cashier::Transfer

Test Case T2.2.2.1	
Coverage Item	Normal Input
Input	1. Transfer:: Target Account: 1 2. Amount: 50 3. Click ‘Confirm’ button
State	The account database has only two accounts: 1. Account 1{name, IDNumber, accountID, password, balance} = {0,0,0,0,100} 2. Account 2{name, IDNumber, accountID, password, balance} = {1,1,1,1,50}
Expected Output	`testCase.atmUI.Message.Value{1},'Success. Your balance is 50. '` Account 1: 1. Balance: 50 2. In Account 1 ATM UI: 1) cashierUI1.QueryTransactionUITable.Data{1,1} == "Balance Change” 2) cashierUI1.QueryTransactionUITable.Data{1,3} == “ATM" 3) cashierUI1.QueryTransactionUITable.Data{1,4} == “-50” Account 2: 1. Balance: 100 2. In Account 2 ATM UI: 1) cashierUI1.QueryTransactionUITable.Data{1,1} == "Balance Change” 2) cashierUI1.QueryTransactionUITable.Data{1,3} == “ATM" 3) cashierUI1.QueryTransactionUITable.Data{1,4} == “50"

Test Case T2.2.2.2	
Coverage Item	Error Input & Normal Input
Input	1. Transfer:: Target Account: 0 2. Amount: 50 3. Click ‘Confirm’ button 4. Transfer:: Target Account: 1 5. Amount: 50 6. Click ‘Confirm’ button
State	The account database has only two accounts: 1. Account 1{name, IDNumber, accountID, password, balance} = {0,0,0,0,100} 2. Account 2{name, IDNumber, accountID, password, balance} = {1,1,1,1,50}

Expected Output	`testCase.cashierUI1.Message.Value{1}=='Do not input your account.'`
	`testCase.cashierUI1.Message.Value{1}==‘Success. Your balance is 50. '`
	
	Account 1:
	1. Balance: 50
	2. In Account 1 ATM UI:
	1) cashierUI1.QueryTransactionUITable.Data{1,1} == "Balance Change”
	2) cashierUI1.QueryTransactionUITable.Data{1,3} == “ATM"
	3) cashierUI1.QueryTransactionUITable.Data{1,4} == “-50”
	Account 2:
	1. Balance: 100
	2. In Account 2 ATM UI:
	1) cashierUI1.QueryTransactionUITable.Data{1,1} == "Balance Change”
	2) cashierUI1.QueryTransactionUITable.Data{1,3} == “ATM"
	3) cashierUI1.QueryTransactionUITable.Data{1,4} == “50"

Test Case T2.2.2.3	
Coverage Item	Error Input & Normal Input
Input	1. Transfer:: Target Account: 1 2. Amount: 150 3. Click ‘Confirm’ button
State	The account database has only two accounts: 1. Account 1{name, IDNumber, accountID, password, balance} = {0,0,0,0,100} 2. Account 2{name, IDNumber, accountID, password, balance} = {1,1,1,1,50}
Expected Output	`testCase.cashierUI1.Message.Value{1} == ‘Insufficient Balance. Your balance is 100. '`

Test result: 3/3 passed

T2.2.3: Cashier::Deposit

Test Case T2.2.3.1	
Coverage Item	Normal Input
Input	1. Deposit::Amount: 50 2. Click ‘Confirm’ button
State	The account database has only account: 1. Account 1{name, IDNumber, accountID, password, balance} = {0,0,0,0,100}
Expected Output	`testCase.cashierUI1.Message.Value{1}==‘Success. Your balance is 150. '` Account 1: 1. Balance: 150 2. In Account 1 ATM UI: 1) cashierUI1.QueryTransactionUITable.Data{1,1} == "Balance Change” 2) cashierUI1.QueryTransactionUITable.Data{1,3} == “ATM" 3) cashierUI1.QueryTransactionUITable.Data{1,4} == “150”

Test result: 1/1 passed

T2.2.4: Cashier::Withdraw

Test Case T2.2.4.1	
Coverage Item	Normal Input
Input	1. Withdraw::Amount: 50 2. Click ‘Confirm’ button
State	The account database has only account: 1. Account 1{name, IDNumber, accountID, password, balance} = {0,0,0,0,100}
Expected Output	`testCase.atmUI.Message.Value{1}==‘Success. Your balance is 50. '` Account 1: 1. Balance: 50 2. In Account 1 ATM UI: 1) cashierUI1.QueryTransactionUITable.Data{1,1} == "Balance Change” 2) cashierUI1.QueryTransactionUITable.Data{1,3} == “ATM" 3) cashierUI1.QueryTransactionUITable.Data{1,4} == “-50”

Test result: 1/1 passed

T2.2.5: Cashier::Reset Password

Test Case T2.2.5.1	
Coverage Item	Normal Input
Input	1. New Passwords: “1” 2. Confirm Passwords: “1”
State	The account database has only account: 1. Account 1{name, IDNumber, accountID, password, balance} = {0,0,0,0,100}

Expected Output	`testCase.cashierUI1.Message.Value{1}=='Success.'`
	
	`testCase.cashierUI1.currentAccount.password == "1"`

Test result: 1/1 passed

T2.2.6: Cashier::Query Information

Test Case T2.2.6.1	
Coverage Item	Normal Input
Input	
State	The account database has only account: 1. Account 1{name, IDNumber, accountID, password, balance} = {0,0,0,0,100}
Expected Output	1. `testCase.cashierUI1.currentAccount.name == convertCharsToStrings(testCase.cashierUI1.QueryNameEditField.Value)` 2. `testCase.cashierUI1.currentAccount.IDNumber == convertCharsToStrings(testCase.cashierUI1.QueryIDNumberEditField.Value)` 3. `testCase.cashierUI1.currentAccount.accountID == convertCharsToStrings(testCase.cashierUI1.QueryAccountIDEditField.Value)` 4. Convert 1) `cmp = str2double(regexp(testCase.cashierUI1.QueryBalanceEditField.Value,'\d*','match'))` 2) `testCase.cashierUI1.currentAccount.balance == cmp`

T2.2.7: Cashier::Create Account

Test Case T2.2.2.1	
Coverage Item	Normal Input
Input	1. Enter account name: 3 2. Enter account ID: 3 3. Enter Password: 3 4. Enter Confirm Password: 3 5. Click “Confirm ” button
State	Two exist accounts.

Expected Output	`testCase.accountDB.accountList(3).name == “3"`
	`testCase.accountDB.accountList(3).IDNumber, "3"`
	`testCase.accountDB.accountList(3).password, "3"`

Test result: 1/1 passed

T2.2.8: Cashier::Reset Information

Test Case T2.2.8.1	
Coverage Item	Normal Input
Input	1. Reset Information Tab 2. Enter account name: 9 3. Enter account ID: 9
State	Two exist accounts.
Expected Output	`testCase.accountDB.accountList(1).name == “9"` `testCase.accountDB.accountList(1).IDNumber ==  "9"`

Test result: 1/1 passed

T2.2.9: Cashier::Close Account

Test Case T2.2.9	
Coverage Item	Normal Input
Input	1. Reset Information Tab 2. Enter account name: 0 3. Enter account ID: 0
State	Two exist accounts.
Expected Output	`testCase.accountDB.accountList(1).name == “1"` `testCase.accountDB.accountList(1).IDNumber ==  “1"`

Test result: 1/1 passed

T3: Functional Test
T3.1: CustomUI Test

T3.1.1: Use Case “Get ticket”

Test Case T3.1.1.1	
Coverage Item	Normal Input
Input	1. Click “Get Ticket” button 2. Click “Confirm” button
State	The first customer click the “Get Ticket” button
Expected Output	1. `testCase.customerUI.GetTicketMessage.Value{1}==’Your ticket is 1.’` 2. `testCase.customerUI.UITable.Data{1,2}==1`

Test result: 1/1 passed

T3.1.2: Use Case “Get ticket” then “Cancel”

Test Case T3.1.2.1	
Coverage Item	Normal Input
Input	1. Click “Get Ticket” button 2. Click “Cancel” button
State	The first customer click the “Get Ticket” button
Expected Output	1. `testCase.customerUI.GetTicketMessage.Value{1}==’Your ticket is 1.’` 2. `testCase.customerUI.GetTicketMessage.Value{1}==’’`

Test result: 1/1 passed

T3.2: ATM UI Test

T3.2.1: Use Case “Enter Account”

Test Case T3.2.1.1	
Coverage Item	Normal Input
Input	1. Enter account ID: 0 2. Click “Account ID Confirm ” button 3. Enter account password: 0 4. Click “Password confirm” button
State	The account database has only one account with ID ‘0’, password ‘0’
Expected Output	`testCase.atmUI.currentAccount == testCase.account`

Test Case T3.2.1.2	
Coverage Item	Error Input
Input	1. Enter account ID: 0 2. Click “Account ID Confirm ” button 3. Enter account password: 1 4. Click “Password confirm” button
State	The account database has only one account with ID ‘0’, password ‘0’
Expected Output	`testCase.atmUI.Message.Value{1}=='Wrong Password.'`

Test Case T3.2.1.3	
Coverage Item	Error Input
Input	1. Enter account ID: 1 2. Click “Account ID Confirm ” button
State	The account database has only one account with ID ‘0’, password ‘0’
Expected Output	`testCase.atmUI.Message.Value{1}=='Fail to identify.'`

Test Case T3.2.1.4	
Coverage Item	Error Input & Normal Input
Input	1. Enter account ID: 0 2. Click “Account ID Confirm ” button 3. Enter account password: 1 4. Click “Password confirm” button 5. Enter account password: 0 6. Click “Password confirm” button
State	The account database has only one account with ID ‘0’, password ‘0’
Expected Output	`1. testCase.atmUI.Message.Value{1}=='Wrong Password.'` `2. testCase.atmUI.currentAccount == testCase.account`

Test Case T3.2.1.5	
Coverage Item	Error Input & Normal Input
Input	1. Enter account ID: 1 2. Click “Account ID Confirm ” button 3. Enter account ID: 0 4. Click “Account ID Confirm ” button 5. Enter account password: 0 6. Click “Password confirm” button
State	The account database has only one account with ID ‘0’, password ‘0’
Expected Output	`1. testCase.atmUI.Message.Value{1}=='Fail to identify.'` `2. testCase.atmUI.currentAccount == testCase.account`

Test result: 5/5 passed

T3.2.2: Use Case “Transfer”

Test Case T3.2.2.1	
Coverage Item	Normal Input
Input	1. Transfer:: Target Account: 1 2. Amount: 50 3. Click ‘Confirm’ button
State	The account database has only two accounts: 1. Account 1{name, IDNumber, accountID, password, balance} = {0,0,0,0,100} 2. Account 2{name, IDNumber, accountID, password, balance} = {1,1,1,1,50}
Expected Output	`testCase.atmUI.Message.Value{1},'Success. Your balance is 50. '` Account 1: 1. Balance: 50 2. In Account 1 ATM UI: 1) atmUI.QueryTransactionUITable.Data{1,1} == "Balance Change” 2) atmUI.QueryTransactionUITable.Data{1,3} == “ATM" 3) atmUI.QueryTransactionUITable.Data{1,4} == “-50” Account 2: 1. Balance: 100 2. In Account 2 ATM UI: 1) atmUI.QueryTransactionUITable.Data{1,1} == "Balance Change” 2) atmUI.QueryTransactionUITable.Data{1,3} == “ATM" 3) atmUI.QueryTransactionUITable.Data{1,4} == “50"

Test Case T3.2.2.2	
Coverage Item	Error Input
Input	1. Transfer:: Target Account: 0 2. Amount: 50 3. Click ‘Confirm’ button
State	The account database has only two accounts: 1. Account 1{name, IDNumber, accountID, password, balance} = {0,0,0,0,100} 2. Account 2{name, IDNumber, accountID, password, balance} = {1,1,1,1,50}
Expected Output	`testCase.atmUI.Message.Value{1} == ‘Do not input your account.'`

Test Case T3.2.2.3	
Coverage Item	Error Input & Normal Input
Input	1. Transfer:: Target Account: 0 2. Amount: 50 3. Click ‘Confirm’ button 4. Transfer:: Target Account: 1 5. Amount: 50 6. Click ‘Confirm’ button
State	The account database has only two accounts: 1. Account 1{name, IDNumber, accountID, password, balance} = {0,0,0,0,100} 2. Account 2{name, IDNumber, accountID, password, balance} = {1,1,1,1,50}

Expected Output	`testCase.atmUI.Message.Value{1}=='Do not input your account.'`
	`testCase.atmUI.Message.Value{1}==‘Success. Your balance is 50. '`
	
	Account 1:
	1. Balance: 50
	2. In Account 1 ATM UI:
	1) atmUI.QueryTransactionUITable.Data{1,1} == "Balance Change”
	2) atmUI.QueryTransactionUITable.Data{1,3} == “ATM"
	3) atmUI.QueryTransactionUITable.Data{1,4} == “-50”
	Account 2:
	1. Balance: 100
	2. In Account 2 ATM UI:
	1) atmUI.QueryTransactionUITable.Data{1,1} == "Balance Change”
	2) atmUI.QueryTransactionUITable.Data{1,3} == “ATM"
	3) atmUI.QueryTransactionUITable.Data{1,4} == “50"

Test Case T3.2.2.4	
Coverage Item	Error Input
Input	1. Transfer:: Target Account: 3 2. Amount: 50 3. Click ‘Confirm’ button
State	The account database has only two accounts: 1. Account 1{name, IDNumber, accountID, password, balance} = {0,0,0,0,100} 2. Account 2{name, IDNumber, accountID, password, balance} = {1,1,1,1,50}
Expected Output	`testCase.atmUI.Message.Value{1} == ‘Fail to identify.'`

Test Case T3.2.2.5	
Coverage Item	Error Input
Input	1. Transfer:: Target Account: 1 2. Amount: 150 3. Click ‘Confirm’ button
State	The account database has only two accounts: 1. Account 1{name, IDNumber, accountID, password, balance} = {0,0,0,0,100} 2. Account 2{name, IDNumber, accountID, password, balance} = {1,1,1,1,50}
Expected Output	`testCase.atmUI.Message.Value{1} == ‘Insufficient Balance. Your balance is 100. '`

Test Case T3.2.2.6	
Coverage Item	Error Input
Input	1. Transfer:: Target Account: 2. Amount: 50 3. Click ‘Confirm’ button
State	The account database has only two accounts: 1. Account 1{name, IDNumber, accountID, password, balance} = {0,0,0,0,100} 2. Account 2{name, IDNumber, accountID, password, balance} = {1,1,1,1,50}
Expected Output	`testCase.atmUI.Message.Value{1} == ‘Please input the target account.'`

Test Case T3.2.2.7	
Coverage Item	Error Input
Input	1. Transfer:: Target Account: 1 2. Amount: 3. Click ‘Confirm’ button
State	The account database has only two accounts: 1. Account 1{name, IDNumber, accountID, password, balance} = {0,0,0,0,100} 2. Account 2{name, IDNumber, accountID, password, balance} = {1,1,1,1,50}
Expected Output	`testCase.atmUI.Message.Value{1}==‘Please input the amount.'`

Test result: 6/6 passed

T3.2.3: Use Case “Deposit”

Test Case T3.2.3.1	
Coverage Item	Normal Input
Input	1. Deposit::Amount: 50 2. Click ‘Confirm’ button
State	The account database has only account: 1. Account 1{name, IDNumber, accountID, password, balance} = {0,0,0,0,100}
Expected Output	`testCase.atmUI.Message.Value{1}==’Success. Your balance is 150. '` Account 1: 1. Balance: 150 2. In Account 1 ATM UI: 1) atmUI.QueryTransactionUITable.Data{1,1} == "Balance Change” 2) atmUI.QueryTransactionUITable.Data{1,3} == “ATM" 3) atmUI.QueryTransactionUITable.Data{1,4} == “150”

Test Case T3.2.3.2	
Coverage Item	Error Input
Input	1. Deposit::Amount: -50 2. Click ‘Confirm’ button
State	The account database has only account: 1. Account 1{name, IDNumber, accountID, password, balance} = {0,0,0,0,100}
Expected Output	`testCase.atmUI.Message.Value{1}==’Please hand in the money.'`

Test Case T3.2.3.3	
Coverage Item	Error Input
Input	1. Deposit::Amount: (none) 2. Click ‘Confirm’ button
State	The account database has only account: 1. Account 1{name, IDNumber, accountID, password, balance} = {0,0,0,0,100}
Expected Output	`testCase.atmUI.Message.Value{1}==’Please hand in the money.'`

Test result: 3/3 passed

T3.2.4: Use Case “Withdraw”

Test Case T3.2.4.1	
Coverage Item	Normal Input
Input	1. Withdraw::Amount: 50 2. Click ‘Confirm’ button
State	The account database has only account: 1. Account 1{name, IDNumber, accountID, password, balance} = {0,0,0,0,100}
Expected Output	`testCase.atmUI.Message.Value{1}==‘Success. Your balance is 50. '` Account 1: 1. Balance: 50 2. In Account 1 ATM UI: 1) atmUI.QueryTransactionUITable.Data{1,1} == "Balance Change” 2) atmUI.QueryTransactionUITable.Data{1,3} == “ATM" 3) atmUI.QueryTransactionUITable.Data{1,4} == “-50”

Test Case T3.2.4.2	
Coverage Item	Error Input
Input	1. Withdraw::Amount: -50 2. Click ‘Confirm’ button
State	The account database has only account: 1. Account 1{name, IDNumber, accountID, password, balance} = {0,0,0,0,100}
Expected Output	`testCase.atmUI.Message.Value{1}=='Please input the amount.'`

Test Case T3.2.4.2	
Coverage Item	Error Input
Input	1. Withdraw::Amount: 150 2. Click ‘Confirm’ button
State	The account database has only account: 1. Account 1{name, IDNumber, accountID, password, balance} = {0,0,0,0,100}
Expected Output	`testCase.atmUI.Message.Value{1}=='Insufficient Balance. Your balance is 100. '`

Test Case T3.2.4.3	
Coverage Item	Error Input
Input	1. Withdraw::Amount: (none) 2. Click ‘Confirm’ button
State	The account database has only account: 1. Account 1{name, IDNumber, accountID, password, balance} = {0,0,0,0,100}
Expected Output	`testCase.atmUI.Message.Value{1}=='Please hand in the money.'`

Test result: 4/4 passed

T3.2.5: Use Case “Reset Password”

Test Case T3.2.4.1	
Coverage Item	Normal Input
Input	1. New Passwords: “1” 2. Confirm Passwords: “1”
State	The account database has only account: 1. Account 1{name, IDNumber, accountID, password, balance} = {0,0,0,0,100}

Expected Output	`testCase.atmUI.Message.Value{1}=='Success.'`
	
	`testCase.atmUI.currentAccount.password == "1"`

Test Case T3.2.4.2	
Coverage Item	Error Input
Input	1. New Passwords: “0” 2. Confirm Passwords: “0”
State	The account database has only account: 1. Account 1{name, IDNumber, accountID, password, balance} = {0,0,0,0,100}
Expected Output	`testCase.atmUI.Message.Value{1}=='The new password must be different from the original one.'`

Test Case T3.2.4.3	
Coverage Item	Error Input
Input	1. New Passwords: “1” 2. Confirm Passwords: “0”
State	The account database has only account: 1. Account 1{name, IDNumber, accountID, password, balance} = {0,0,0,0,100}
Expected Output	`testCase.atmUI.Message.Value{1}=='Two input passwords must be consistent.'`

Test Case T3.2.4.4	
Coverage Item	Error Input
Input	1. New Passwords: “” 2. Confirm Passwords: “”
State	The account database has only account: 1. Account 1{name, IDNumber, accountID, password, balance} = {0,0,0,0,100}
Expected Output	`testCase.atmUI.Message.Value{1}==’Please input the new password.'`

Test Case T3.2.4.5	
Coverage Item	Error Input
Input	1. New Passwords: “” 2. Confirm Passwords: “”
State	The account database has only account: 1. Account 1{name, IDNumber, accountID, password, balance} = {0,0,0,0,100}
Expected Output	`testCase.atmUI.Message.Value{1}==’Please comfirm the new password.'`

Test result: 5/5 passed

T3.2.6: Use Case “Query Information”

Test Case T3.2.4.1	
Coverage Item	Normal Input
Input	
State	The account database has only account: 1. Account 1{name, IDNumber, accountID, password, balance} = {0,0,0,0,100}
Expected Output	1. testCase.atmUI.currentAccount.name == convertCharsToStrings(testCase.atmUI.QueryNameEditField.Value) 2. testCase.atmUI.currentAccount.IDNumber == convertCharsToStrings(testCase.atmUI.QueryIDNumberEditField.Value) 3. testCase.atmUI.currentAccount.accountID == convertCharsToStrings(testCase.atmUI.QueryAccountIDEditField.Value) 4. Convert 1) cmp = str2double(regexp(testCase.atmUI.QueryBalanceEditField.Value,'\d*','match')) 2) testCase.atmUI.currentAccount.balance == cmp

Test result: 1/1 passed

T3.3: CashierUI Test

T3.3.1: Use Case “Enter Account”

Test Case T3.3.1.1	
Coverage Item	Normal Input
Input	1. Enter account ID: 0 2. Click “Account ID Confirm ” button 3. Enter account password: 0 4. Click “Password confirm” button
State	The account database has only one account with ID ‘0’, password ‘0’
Expected Output	`testCase.cashierUI1.currentAccount == testCase.account`

Test Case T3.3.1.2	
Coverage Item	Error Input
Input	1. Enter account ID: 0 2. Click “Account ID Confirm ” button 3. Enter account password: 1 4. Click “Password confirm” button
State	The account database has only one account with ID ‘0’, password ‘0’
Expected Output	`testCase.cashierUI1.Message.Value{1}==‘Wrong Password.'`

Test Case T3.3.1.3	
Coverage Item	Error Input
Input	1. Enter account ID: 1 2. Click “Account ID Confirm ” button
State	The account database has only one account with ID ‘0’, password ‘0’
Expected Output	`testCase.cashierUI1.Message.Value{1}=='Fail to identify.'`

Test Case T3.3.1.4	
Coverage Item	Error Input & Normal Input
Input	1. Enter account ID: 0 2. Click “Account ID Confirm ” button 3. Enter account password: 1 4. Click “Password confirm” button 5. Enter account password: 0 6. Click “Password confirm” button
State	The account database has only one account with ID ‘0’, password ‘0’
Expected Output	`1. testCase.cashierUI1.Message.Value{1}=='Wrong Password.'` `2. testCase.cashierUI1.currentAccount == testCase.account`

Test Case T3.3.1.5	
Coverage Item	Error Input & Normal Input
Input	1. Enter account ID: 1 2. Click “Account ID Confirm ” button 3. Enter account ID: 0 4. Click “Account ID Confirm ” button 5. Enter account password: 0 6. Click “Password confirm” button
State	The account database has only one account with ID ‘0’, password ‘0’
Expected Output	`1. testCase.cashierUI1.Message.Value{1}=='Fail to identify.'` `2. testCase.cashierUI1.currentAccount == testCase.account`

Test result: 5/5 passed

T3.3.2: Use Case “Transfer”

Test Case T3.3.2.1	
Coverage Item	Normal Input
Input	1. Transfer:: Target Account: 1 2. Amount: 50 3. Click ‘Confirm’ button
State	The account database has only two accounts: 1. Account 1{name, IDNumber, accountID, password, balance} = {0,0,0,0,100} 2. Account 2{name, IDNumber, accountID, password, balance} = {1,1,1,1,50}
Expected Output	`testCase.atmUI.Message.Value{1},'Success. Your balance is 50. '` Account 1: 1. Balance: 50 2. In Account 1 ATM UI: 1) atmUI.QueryTransactionUITable.Data{1,1} == "Balance Change” 2) atmUI.QueryTransactionUITable.Data{1,3} == “ATM" 3) atmUI.QueryTransactionUITable.Data{1,4} == “-50” Account 2: 1. Balance: 100 2. In Account 2 ATM UI: 1) atmUI.QueryTransactionUITable.Data{1,1} == "Balance Change” 2) atmUI.QueryTransactionUITable.Data{1,3} == “ATM" 3) atmUI.QueryTransactionUITable.Data{1,4} == “50"

Test Case T3.3.2.2	
Coverage Item	Error Input
Input	1. Transfer:: Target Account: 0 2. Amount: 50 3. Click ‘Confirm’ button
State	The account database has only two accounts: 1. Account 1{name, IDNumber, accountID, password, balance} = {0,0,0,0,100} 2. Account 2{name, IDNumber, accountID, password, balance} = {1,1,1,1,50}
Expected Output	`testCase.cashierUI1.Message.Value{1} == ‘Do not input your account.'`

Test Case T3.3.2.3	
Coverage Item	Error Input & Normal Input
Input	1. Transfer:: Target Account: 0 2. Amount: 50 3. Click ‘Confirm’ button 4. Transfer:: Target Account: 1 5. Amount: 50 6. Click ‘Confirm’ button
State	The account database has only two accounts: 1. Account 1{name, IDNumber, accountID, password, balance} = {0,0,0,0,100} 2. Account 2{name, IDNumber, accountID, password, balance} = {1,1,1,1,50}

Expected Output	`testCase.cashierUI1.Message.Value{1}=='Do not input your account.'`
	`testCase.cashierUI1.Message.Value{1}==‘Success. Your balance is 50. '`
	
	Account 1:
	1. Balance: 50
	2. In Account 1 ATM UI:
	1) cashierUI1.QueryTransactionUITable.Data{1,1} == "Balance Change”
	2) cashierUI1.QueryTransactionUITable.Data{1,3} == “ATM"
	3) cashierUI1.QueryTransactionUITable.Data{1,4} == “-50”
	Account 2:
	1. Balance: 100
	2. In Account 2 ATM UI:
	1) cashierUI1.QueryTransactionUITable.Data{1,1} == "Balance Change”
	2) cashierUI1.QueryTransactionUITable.Data{1,3} == “ATM"
	3) cashierUI1.QueryTransactionUITable.Data{1,4} == “50"

Test Case T3.3.2.4	
Coverage Item	Error Input
Input	1. Transfer:: Target Account: 3 2. Amount: 50 3. Click ‘Confirm’ button
State	The account database has only two accounts: 1. Account 1{name, IDNumber, accountID, password, balance} = {0,0,0,0,100} 2. Account 2{name, IDNumber, accountID, password, balance} = {1,1,1,1,50}
Expected Output	`testCase.cashierUI1.Message.Value{1} == ‘Fail to identify.'`

Test Case T3.3.2.5	
Coverage Item	Error Input
Input	1. Transfer:: Target Account: 1 2. Amount: 150 3. Click ‘Confirm’ button
State	The account database has only two accounts: 1. Account 1{name, IDNumber, accountID, password, balance} = {0,0,0,0,100} 2. Account 2{name, IDNumber, accountID, password, balance} = {1,1,1,1,50}
Expected Output	`testCase.cashierUI1.Message.Value{1} == ‘Insufficient Balance. Your balance is 100. '`

Test Case T3.3.2.6	
Coverage Item	Error Input
Input	1. Transfer:: Target Account: 2. Amount: 50 3. Click ‘Confirm’ button
State	The account database has only two accounts: 1. Account 1{name, IDNumber, accountID, password, balance} = {0,0,0,0,100} 2. Account 2{name, IDNumber, accountID, password, balance} = {1,1,1,1,50}
Expected Output	`testCase.cashierUI1.Message.Value{1} == ‘Please input the target account.'`

Test Case T3.3.2.7	
Coverage Item	Error Input
Input	1. Transfer:: Target Account: 1 2. Amount: 3. Click ‘Confirm’ button
State	The account database has only two accounts: 1. Account 1{name, IDNumber, accountID, password, balance} = {0,0,0,0,100} 2. Account 2{name, IDNumber, accountID, password, balance} = {1,1,1,1,50}
Expected Output	`testCase.cashierUI1.Message.Value{1}==‘Please input the amount.'`

Test result: 6/6 passed

T3.3.3: Use Case “Deposit”

Test Case T3.3.3.1	
Coverage Item	Normal Input
Input	1. Deposit::Amount: 50 2. Click ‘Confirm’ button
State	The account database has only account: 1. Account 1{name, IDNumber, accountID, password, balance} = {0,0,0,0,100}
Expected Output	`testCase.atmUI.Message.Value{1}==‘Success. Your balance is 150. '` Account 1: 1. Balance: 150 2. In Account 1 ATM UI: 1) cashierUI1.QueryTransactionUITable.Data{1,1} == "Balance Change” 2) cashierUI1.QueryTransactionUITable.Data{1,3} == “ATM" 3) cashierUI1.QueryTransactionUITable.Data{1,4} == “150”

Test Case T3.3.3.2	
Coverage Item	Error Input
Input	1. Deposit::Amount: -50 2. Click ‘Confirm’ button
State	The account database has only account: 1. Account 1{name, IDNumber, accountID, password, balance} = {0,0,0,0,100}
Expected Output	`testCase.cashierUI1.Message.Value{1}==‘Please hand in the money.'`

Test Case T3.3.3.3	
Coverage Item	Error Input
Input	1. Deposit::Amount: (none) 2. Click ‘Confirm’ button
State	The account database has only account: 1. Account 1{name, IDNumber, accountID, password, balance} = {0,0,0,0,100}
Expected Output	`testCase.cashierUI1.Message.Value{1}==‘Please hand in the money.'`

Test result: 3/3 passed

T3.3.4: Use Case “Withdraw”

Test Case T3.3.4.1	
Coverage Item	Normal Input
Input	1. Withdraw::Amount: 50 2. Click ‘Confirm’ button
State	The account database has only account: 1. Account 1{name, IDNumber, accountID, password, balance} = {0,0,0,0,100}
Expected Output	`testCase.atmUI.Message.Value{1}=='Success. Your balance is 50. '` Account 1: 1. Balance: 50 2. In Account 1 ATM UI: 1) cashierUI1.QueryTransactionUITable.Data{1,1} == "Balance Change” 2) cashierUI1.QueryTransactionUITable.Data{1,3} == “ATM" 3) cashierUI1.QueryTransactionUITable.Data{1,4} == “-50”

Test Case T3.3.4.2	
Coverage Item	Error Input
Input	1. Withdraw::Amount: -50 2. Click ‘Confirm’ button
State	The account database has only account: 1. Account 1{name, IDNumber, accountID, password, balance} = {0,0,0,0,100}
Expected Output	`testCase.cashierUI1.Message.Value{1}=='Please input the amount.'`

Test Case T3.3.4.2	
Coverage Item	Error Input
Input	1. Withdraw::Amount: 150 2. Click ‘Confirm’ button
State	The account database has only account: 1. Account 1{name, IDNumber, accountID, password, balance} = {0,0,0,0,100}
Expected Output	`testCase.cashierUI1.Message.Value{1}=='Insufficient Balance. Your balance is 100. '`

Test Case T3.3.4.3	
Coverage Item	Error Input
Input	1. Withdraw::Amount: (none) 2. Click ‘Confirm’ button
State	The account database has only account: 1. Account 1{name, IDNumber, accountID, password, balance} = {0,0,0,0,100}
Expected Output	`testCase.cashierUI1.Message.Value{1}==‘Please hand in the money.'`

Test result: 4/4 passed

T3.3.5: Use Case “Reset Password”

Test Case T3.3.5.1	
Coverage Item	Normal Input
Input	1. New Passwords: “1” 2. Confirm Passwords: “1”
State	The account database has only account: 1. Account 1{name, IDNumber, accountID, password, balance} = {0,0,0,0,100}

Expected Output	`testCase.cashierUI1.Message.Value{1}==‘Success.'`
	
	`testCase.cashierUI1.currentAccount.password == "1"`

Test Case T3.3.5.2	
Coverage Item	Error Input
Input	1. New Passwords: “0” 2. Confirm Passwords: “0”
State	The account database has only account: 1. Account 1{name, IDNumber, accountID, password, balance} = {0,0,0,0,100}
Expected Output	`testCase.cashierUI1.Message.Value{1}==‘The new password must be different from the original one.'`

Test Case T3.3.5.3	
Coverage Item	Error Input
Input	1. New Passwords: “1” 2. Confirm Passwords: “0”
State	The account database has only account: 1. Account 1{name, IDNumber, accountID, password, balance} = {0,0,0,0,100}
Expected Output	`testCase.cashierUI1.Message.Value{1}==‘Two input passwords must be consistent.'`

Test Case T3.3.5.4	
Coverage Item	Error Input
Input	1. New Passwords: “” 2. Confirm Passwords: “”
State	The account database has only account: 1. Account 1{name, IDNumber, accountID, password, balance} = {0,0,0,0,100}
Expected Output	`testCase.cashierUI1.Message.Value{1}==‘Please input the new password.'`

Test Case T3.3.5.5	
Coverage Item	Error Input
Input	1. New Passwords: “” 2. Confirm Passwords: “”
State	The account database has only account: 1. Account 1{name, IDNumber, accountID, password, balance} = {0,0,0,0,100}
Expected Output	`testCase.cashierUI1.Message.Value{1}==‘Please comfirm the new password.'`

Test result: 5/5 passed

T3.3.6: Use Case “Query Information”

Test Case T3.3.6.1	
Coverage Item	Normal Input
Input	
State	The account database has only account: 1. Account 1{name, IDNumber, accountID, password, balance} = {0,0,0,0,100}
Expected Output	1. `testCase.cashierUI1.currentAccount.name == convertCharsToStrings(testCase.cashierUI1.QueryNameEditField.Value)` 2. `testCase.cashierUI1.currentAccount.IDNumber == convertCharsToStrings(testCase.cashierUI1.QueryIDNumberEditField.Value)` 3. `testCase.cashierUI1.currentAccount.accountID == convertCharsToStrings(testCase.cashierUI1.QueryAccountIDEditField.Value)` 4. Convert 1) `cmp = str2double(regexp(testCase.cashierUI1.QueryBalanceEditField.Value,'\d*','match'))` 2) `testCase.cashierUI1.currentAccount.balance == cmp`

Test result: 1/1 passed

T3.3.7: Use Case “Create Account”

Test Case T3.3.7.1	
Coverage Item	Normal Input
Input	1. Enter account name: 3 2. Enter account ID: 3 3. Enter Password: 3 4. Enter Confirm Password: 3 5. Click “Confirm ” button
State	Two exist accounts.

Expected Output	`testCase.accountDB.accountList(3).name == “3"`
	`testCase.accountDB.accountList(3).IDNumber, "3"`
	`testCase.accountDB.accountList(3).password, "3"`

Test Case T3.3.7.2	
Coverage Item	Error Input
Input	1. Enter account ID: 3 2. Enter Password: 3 3. Enter Confirm Password: 3 4. Click “Confirm ” button
State	Two exist accounts.
Expected Output	`testCase.cashierUI1.Message.Value{1}== 'Please input the name.'`

Test Case T3.3.7.3	
Coverage Item	Error Input
Input	1. Enter account name: 3 2. Enter account ID: 3 3. Enter Password: 3 4. Enter Confirm Password: 4 5. Click “Confirm ” button
State	Two exist accounts.
Expected Output	`testCase.cashierUI1.Message.Value{1}== 'Two input password must be consistent.'`

Test Case T3.3.7.4	
Coverage Item	Error Input
Input	1. Enter account name: 3 2. Enter account ID: 3. Enter Password: 3 4. Enter Confirm Password: 3 5. Click “Confirm ” button
State	Two exist accounts.
Expected Output	`testCase.cashierUI1.Message.Value{1} == ‘Please input the ID number.')`

Test Case T3.3.7.5	
Coverage Item	Error Input
Input	1. Enter account name: 3 2. Enter account ID: 3. Enter Password: 3 4. Enter Confirm Password: 3 5. Click “Confirm ” button
State	Two exist accounts.
Expected Output	`testCase.cashierUI1.Message.Value{1} == 'Please comfirm the password.'`

Test result: 5/5 passed

T3.3.8: Use Case “Reset Information”

Test Case T3.3.8.1	
Coverage Item	Normal Input
Input	1. Reset Information Tab 2. Enter account name: 9 3. Enter account ID: 9
State	Two exist accounts.
Expected Output	`testCase.accountDB.accountList(1).name == “9"` `testCase.accountDB.accountList(1).IDNumber ==  "9"`

Test Case T3.3.8.2	
Coverage Item	Error Input
Input	1. Reset Information Tab 2. Enter account name: 3. Enter account ID: 9
State	Two exist accounts.
Expected Output	`testCase.cashierUI1.Message.Value{1} == 'Please input the name.'`

Test Case T3.3.8.3	
Coverage Item	Error Input
Input	1. Reset Information Tab 2. Enter account name: 9 3. Enter account ID:
State	Two exist accounts.
Expected Output	`testCase.cashierUI1.Message.Value{1} == 'Please input the ID number.'`

Test result: 3/3 passed

T3.3.9: Use Case “Close Account”

Test Case T3.3.9.1	
Coverage Item	Normal Input
Input	1. Reset Information Tab 2. Enter account name: 0 3. Enter account ID: 0
State	Two exist accounts.
Expected Output	`testCase.accountDB.accountList(1).name == “1"` `testCase.accountDB.accountList(1).IDNumber == “1"`

Test Case T3.3.9.2	
Coverage Item	Error Input
Input	1. Reset Information Tab 2. Enter account name: 3. Enter account ID: 0
State	Two exist accounts.
Expected Output	`testCase.cashierUI1.Message.Value{1} == 'Please input the name.'`

Test Case T3.3.9.3	
Coverage Item	Error Input
Input	1. Reset Information Tab 2. Enter account name: 0 3. Enter account ID:
State	Two exist accounts.
Expected Output	`testCase.cashierUI1.Message.Value{1} == 'Please input the ID number.'`