CS 586 Project Phase 1

Jiaqi Yan A20321362

1. MDA-EFSM

Meta Events

```
Open()
Login()
LoginFail()
Logout()
IncorrectPin(int max)
CorrectPinAboveMin()
AboveMin()
BelowMin()
Balance()
Withdraw()
WithdrawFail()
Deposit()
Lock()
LockFail()
Unlock()
UnlockFail()
Close()
```

Meta Actions

```
StorePin()              // store PIN with temp_pin
StoreBalance()          // store balance with temp_b
StoreId()               // store ID with temp_id
IncorrectPinMsg()       // print "incorrect pin" msg
IncorrectIdMsg()        // print "incorrect id" msg
TooManyAttemptsMsg()    // print "too many attempts" msg
PromptPin()             // prompt to input PIN
StoreAttempts()         // update attempts with temp_a
DisplayMenu()           // display menu
DoDeposit()             // increase balance by temp_d
NoFundMsg()             // print "no fund" msg
DisplayBalance()        // display account balance
DoWithdraw()            // decrease balance by temp_w
BelowMinMsg()           // print "balance below min" msg
```

2. Input Process for Account1

Based on the minimal balance in Account1, after `Deposit()`, `Withdraw()`, `Unlock()` and `CorrectPin()`, we need to check if we should invoke `AboveMin()` or `BelowMin()` event.

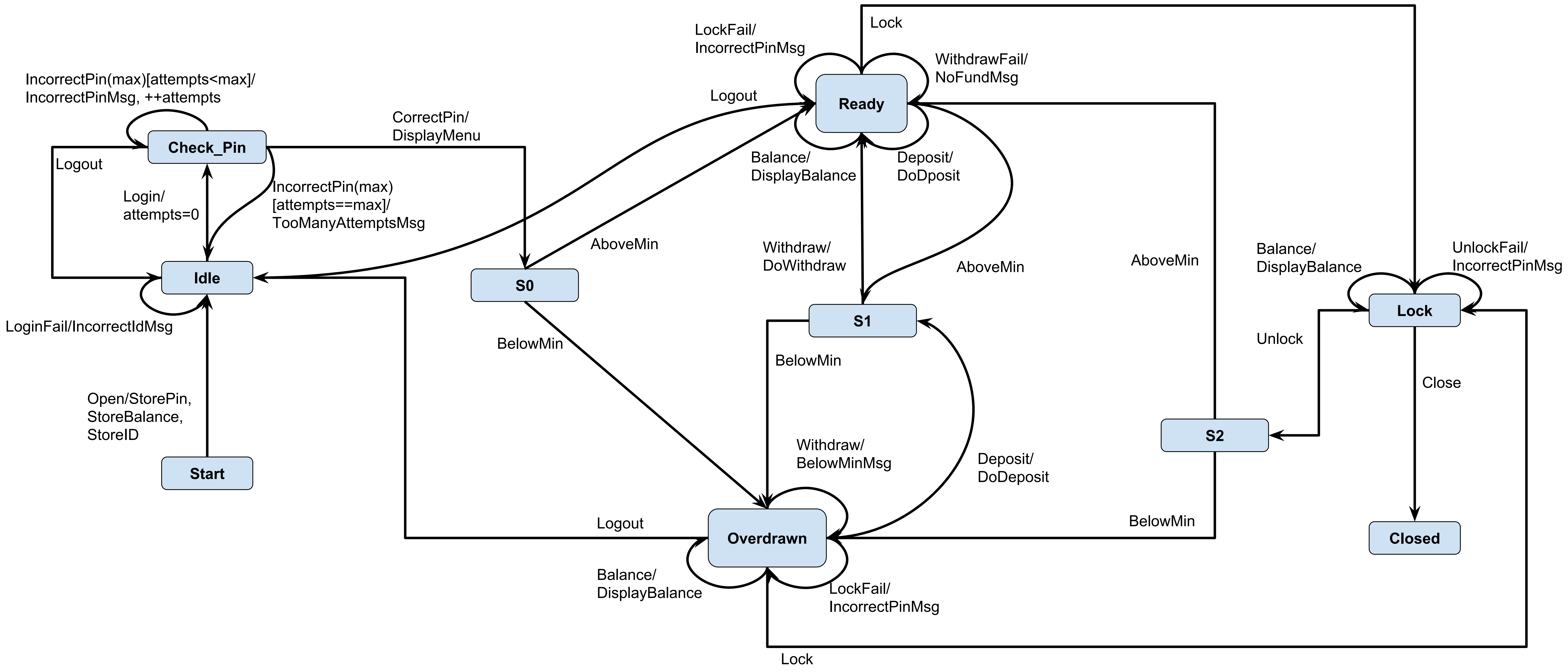
```
Account1::max_attempts = 3;
Account1::min_balance = 500;

void Account1::open(string p, string y, float a) {
    data->temp_pin = p;
    data->temp_id = y;
    data->temp_b = a;
    mda->Open()
}
```

```
void Account1::pin(string x) {
    if (x == data->pin) {
        mda->CorrectPin();
        if (data->b > min_balance) {
            mda->AboveMin();
        } else {
            mda->BelowMin();
        }
    } else {
        mda->IncorrectPin(max_attempts);
    }
}

void Account1::deposit(float d) {
    data->temp_d = d;
    mda->Deposit();
    if (data->b > min_balance) {
        mda->AboveMin();
    } else {
        mda->BelowMin();
    }
}

void Account1::withdraw(float w) {
    data->temp_w = w;
    if (data->b <= min_balance) {
        mda->WithdrawFail()
    } else {
        mda->Withdraw();
    }
    if (data->b > min_balance) {
        mda->AboveMin();
    } else {
        mda->BelowMin();
    }
}

void Account1::balance() {
    mda->Balance();
}

void Account1::login(string y) {
    if (y == data->id) {
        mda->Login();
    } else {
        mda->LoginFail();
    }
}

void Account1::logout() {
    mda->Logout();
}

void Account1::lock(string x) {
    if (x == data->pin) {
        mda->Lock();
        } else {
        mda->LockFail();
    }
}

void Account1::unlock(string x) {
    if (x == data->pin) {
        mda->Unlock();
        if (data->b > min_balance) {
            mda->AboveMin();
        } else {
            mda->BelowMin();
        }
    } else {
        mda->UnlockFail();
    }
}
```

3. Input Process for Account2

Since there is no minimal balance in Account2, after `Deposit()`, `Withdraw()`, `Unlock()` and `CorrectPin()`, we can always invoke `AboveMin()` event.

```
Account2::max_attempts = 2;
Account2::min_balance = 0;

void Account2::OPEN(int p, int y, int a) {
    data->temp_pin = p;
    data->temp_id = y;
    data->temp_b = a;
    mda->Open();
}

void Account2::PIN(int x) {
    if (x == data->pin) {
        mda->CorrectPin();
        mda->AboveMin();
    } else {
        mda->IncorrectPin(max_attempts);
    }
}

void Account2::DEPOSIT(int d) {
    data->temp_d = d;
    mda->Deposit();
    mda->AboveMin();
}

void Account2::WITHDRAW(int w) {
    data->temp_w = w;
    if (data->b > min_balance) {
        mda->Withdraw();
        mda->AboveMin();
    } else {
        mda->WithdrawFail();
    }
}

void Account2::BALANCE() {
    mda->Balance();
}

void Account2::LOGIN(int y) {
    if (y == data->id) {
        mda->Login();
    } else {
        mda->LoginFail();
    }
}

void Account2::LOGOUT() {
    mda->Logout();
}

void Account2::suspend() {
    mda->Lock();
}

void Account2::activate() {
    mda->Unlock();
    mda->AboveMin();
}

void Account2::close() {
    mda->Close();
}
```



EFSM-MDA State Diagram