|  |  |
| --- | --- |
|  | **Fsoft logo for Words-Small.gif** |

**Design Document**

**For AB-SERVICE**

*Prepared for*

**C/C++ Team**

**Tuesday, July 31, 2012**

**Version 0.1**

*Prepared by*

**Pham Trung Hai**

Revision and Signoff Sheet

Change Record

|  |  |  |  |
| --- | --- | --- | --- |
| **Date** | **Author** | **Version** | **Change reference** |
| 07/25/2012 | ChienTQ1 | 0.1 | 4, 5,6.3 |
| 07/25/2012 | NhungDTH1 | 0.1 | 6.2,6.5 |
| 07/25/2012 | LuongBV1 | 0.1 | 6.4 |
| 07/26/2012 | HongLTB | 0.1 | 6.1 |
| 07/26/2012 | C/C++ Teeam | 0.1 | 1,2,3,7,8,9 |

Reviewers

|  |  |  |  |
| --- | --- | --- | --- |
| **Name** | **Version** | **Position** | **Date** |
| C/C++ Team | 0.1 | QA | 07//26/2012 |
| C/C++ Team | 0.1 | QA | 07/26/2012 |

Table of Contents

[1 Introduction 31](#_Toc322064171)

[1.1 Purpose 31](#_Toc322064172)

[1.2 Scope 31](#_Toc322064173)

[1.3 Intended Audiences and Document Organization 31](#_Toc322064174)

[1.4 Acronyms and Abbreviations 31](#_Toc322064175)

[1.5 References 32](#_Toc322064176)

[2 Architecture design 32](#_Toc322064177)

[2.1 Application Logical Architecture 32](#_Toc322064178)

[2.1 User case view 32](#_Toc322064179)

[2.1.1 Overview 32](#_Toc322064180)

[2.1.2 Description 33](#_Toc322064181)

[2.2 Architectural Representation 33](#_Toc322064182)

[2.2.1 Presentation Layer 34](#_Toc322064183)

[2.2.2 Business Layer 34](#_Toc322064184)

[2.2.3 Data Access Layer 35](#_Toc322064185)

[2.3 Packages/Components view 35](#_Toc322064186)

[2.2.4 UI Components 35](#_Toc322064187)

[2.2.5 Business Entities 36](#_Toc322064188)

[2.2.6 Data Access Object 36](#_Toc322064189)

[2.2.7 Exceptions Lib 36](#_Toc322064190)

[2.2.8 General Lib 36](#_Toc322064191)

[2.2.9 Logging Lib 36](#_Toc322064192)

[3 Technical Solutions 37](#_Toc322064193)

[3.1 Exception handling mechanism 37](#_Toc322064194)

[3.2 Loging mechanis 37](#_Toc322064195)

[4 Database design HuyVQ4 37](#_Toc322064196)

[4.1 Entity Relationship Diagram 37](#_Toc322064197)

[4.2 Schema 37](#_Toc322064198)

[4.3 Detail Schema 38](#_Toc322064199)

[4.3.1 Customer 38](#_Toc322064200)

[4.3.2 Account 38](#_Toc322064201)

[4.3.3 Card 38](#_Toc322064202)

[4.3.4 OverDraft Limit 38](#_Toc322064203)

[4.3.5 WithDraw Limit 38](#_Toc322064204)

[4.3.6 ATM 39](#_Toc322064205)

[4.3.7 Money 39](#_Toc322064206)

[4.3.8 Stock 39](#_Toc322064207)

[4.3.9 LogType 39](#_Toc322064208)

[4.3.10 Log 39](#_Toc322064209)

[4.3.11 Config 40](#_Toc322064210)

[5 Application Security 40](#_Toc322064211)

[5.1 . Main functions ‘s security matrix 40](#_Toc322064212)

[6 Interface Design 41](#_Toc322064213)

[6.1 Layout 41](#_Toc322064214)

[6.2 Themes 41](#_Toc322064215)

[7 Details function design 41](#_Toc322064216)

[7.1 Use case 01: Validation – HuyVQ4 41](#_Toc322064217)

[7.1.1 Validate Card 41](#_Toc322064218)

[7.1.2 Authentication 46](#_Toc322064219)

[7.2 Use case 02: Withdraw Money - LuyenNV 51](#_Toc322064220)

[7.2.1 Withdraw 51](#_Toc322064221)

[7.2.2 Dispense Cash 57](#_Toc322064222)

[7.3 Use case 03: Check Balance – LuyenNV 57](#_Toc322064223)

[7.4 Use case 04: View History – HuyVQ4 61](#_Toc322064224)

[7.4.1 View History 61](#_Toc322064225)

[7.5 Use case 05: Cash Transfer – DucNA9 66](#_Toc322064226)

[7.6 Use case 06: Change PIN – DucNA9 79](#_Toc322064227)

[7.7 Use case 07: Logging – HuyVQ4 85](#_Toc322064228)

[7.7.1 Logging 85](#_Toc322064229)

[8 Configuration 87](#_Toc322064230)

[8.1 Application Configuration 87](#_Toc322064231)

[8.2 System Configuration 88](#_Toc322064232)

[9 Packaging and Deployment 88](#_Toc322064233)

[10 Appendix 88](#_Toc322064234)

[10.1 Micrsoft Enterprise library 88](#_Toc322064235)

# Introduction

## Purpose

This document provides a technical approach for implementing ATM application, using a number of different architectural diagrams to view different aspects of the system. It also presents the decisions that will govern how the application will be built from with C/C++ programming perspective.

This document’s aim is to describe the architecture at high level, in which the system will be described as a set of functional layers, with technologies suggested for each layers.

## Scope

This document is prepared for the ATM console application in scope of the Assignemtn of Fasttrack C/C++ FRESHER19.

## Intended Audiences and Document Organization

This document is intended for:

* Development team: Developers
* Customer Representatives: Responsible to review & approve the document.

Below are main sections of the document:

* **Introduction** : This section describes the general introduction of this document
* **Architecture Design :** This section describes the high-level technical assessments and decisions for the application.
* **Data Design**: This section describesin detail how data is structured and manipulated in this application.
* **Interface Design:** This section describesin detail how UI is designed in general ( layout , theme ).
* **Application Security**: This section describles security matrix in detail
* **Configuration:** This section describes all configuration needed for the application to function properly.
* **Packaging and Deployment:** This section describles how applications could be packaged and deployed.

## Acronyms and Abbreviations

|  |  |  |
| --- | --- | --- |
| # | Item | Description |
| 1 | ATM | Automated Teller Machine |
| 2 | PL | Presentation Layer |
| 3 | BLL | Business Logic Layer |
| 4 | DAL | Data Access Layer |
| 5 | DAO | Data Access Object, this object is responsible for attaching to a system, extracting some information, based on specific requirements, and creating a value object. |
| 6 | VO | Value Object |
| 7 | OD | Over draft |

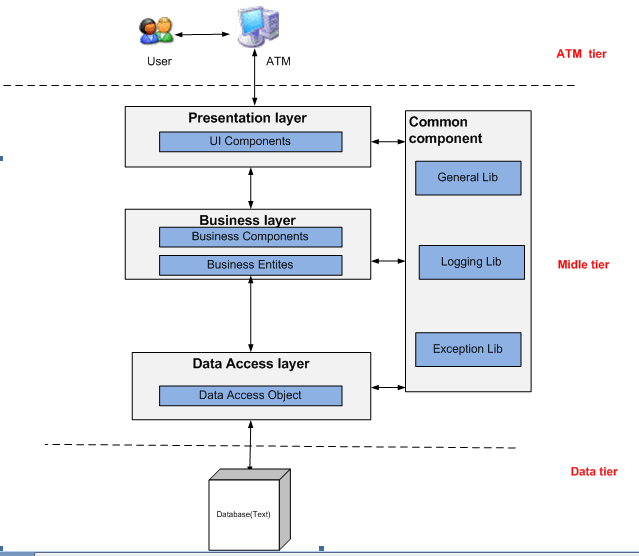
## References

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| # | Title | Version | File Name / Link | Description |
| 1 | C/C++ Project\_Software Design Document | 1.0 | **.**C/C++  **Project\_Software Design Document.docx** |  |
| 2 |  |  |  |  |

Table 1.1: List of References

# Architecture design

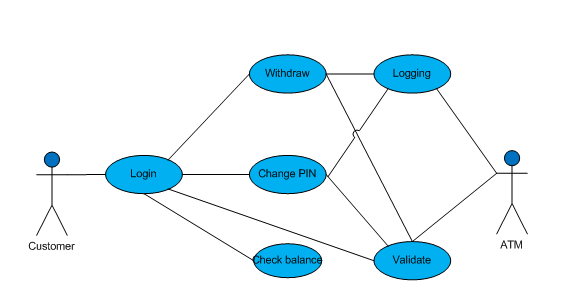
## Application Logical Architecture



## User case view

### Overview

The following use cases are required to be to be included in the ATM simulator application



### Description

|  |  |  |
| --- | --- | --- |
| **UC** | **Name** | **Description** |
| UC01 | Validation | Validate customer card and PIN customer enterred |
| UC02 | WithdrawMoney | Allow customer to withdraw money |
| UC03 | View Balance | Allow customer to check their account balance |
| UC04 | Login | Allow customer to login to ATM |
| UC05 | Change PIN | Allow customer to change their PIN |
| UC05 | Logging | System write log |

## Architectural Representation

The following diagram shows the primary tiers in the proposed n-tier architecture. This diagram shows the main layers in this architecture and the vision of how they fit together.

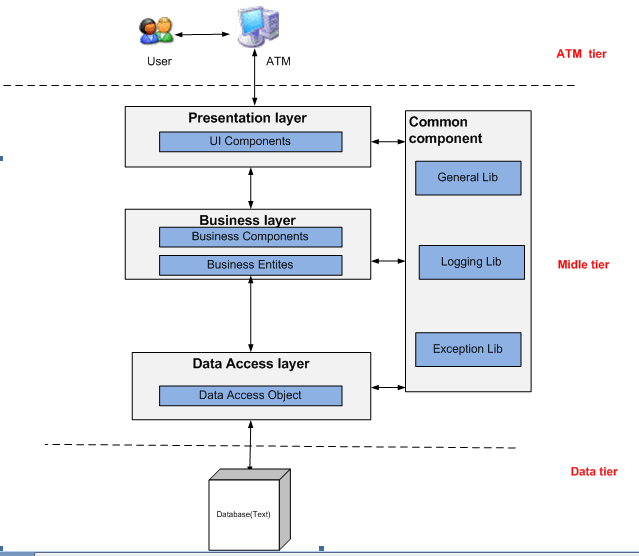


Figure 1 – N-tier architecture of ATM simulator system

### Presentation Layer

This layer controls the display to the end user. The development framework is based on .NET Model architecture. The framework is responsible for:

* Managing requests/responses from/to the clients.
* Controlling display to the end user.
* Performing UI validation.
* Handling exceptions from other layers.

### Business Layer

This layer manages the business processing rules and logic.

* Handling application business logic and business validation.
* Managing transactions.
* Allowing interfaces for interaction with other layers.
* Managing dependencies between business level objects.
* Adding flexibility between the presentation and the persistence layer so they do not directly communicate with each other.
* Exposing a context to the business layer from the presentation layer to obtain business services.
* Managing implementations from the business logic to the persistence layer.

### Data Access Layer

This layer manages access to persistent storage. The primary reason to separate data access from the rest of the application is that it is easier to switch data sources and share Data Access Objects (DAOs) between applications.

This layer manages reading, writing, updating, and deleting stored data.

## 2.3 Packages/Components view

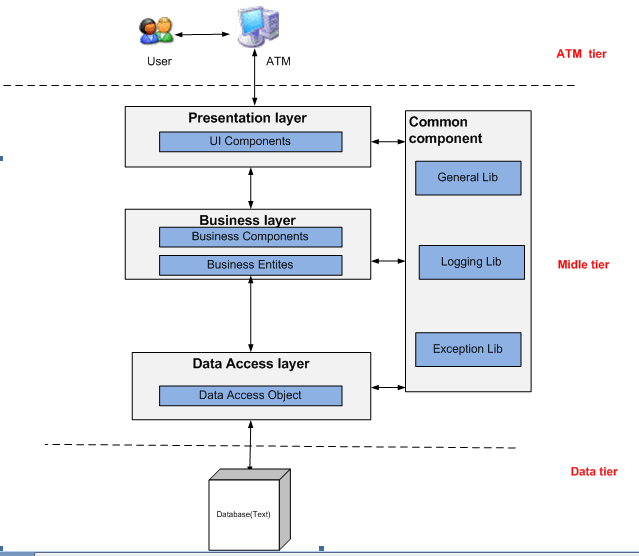


Figure 2 – Packages/Component view of ATM simulator system

### UI Components

This package includes the implementation for the .NET architecture proposed to be used in the Presentation Layer to handle the display to the end user.

**Validation**: All validation of incoming requests parameters to the server should be validated using .NET client side control .

### Business Entities

This package includes the implementation of business objects. **Business Entities** (BE) layer is used to perform the business operations. The Business Entities layer will access the DAO to access database. Transactions should be managed within this business layer.

### Data Access Object

This package includes the implementation of Data Access Object. Using Linq to SQL classes here to make the application more flexible to access database. Linq to SQL classes includes basic functions to work with database: *select, insert, update, delete*.

### Exceptions Lib

This package will include all general exceptions that will typically used by more than one package. The try-catch clauses should be kept to a minimum.

### General Lib

This package includes all utilities .NET classes will be wisely used in the modules.

### Logging Lib

This package includes implemented logging classes.

# Technical Solutions

## Exception handling mechanism

* Error user
* When user input card, ATM checkvalid, all error of this process :

+Type:0

+ID : NVL

+Mess:”Card is not valid”

* When user input card,card is valid, all error of process’s withdraw money :

+Type:0

+ID : NWD

+Mess:”Can not withdraw money because your balance is not enough ”

* When user input card,card is valid, ATM not enough money :

+Type:0

+ID : NEM

+Mess:”Can not withdraw money because your balance is not enough ”

* Error system
* ATM can not reject card :

+Type:1

+ID : NRC

+Mess:”error system ”

|  |  |  |
| --- | --- | --- |
| **STT** | **Name** | **Description** |
| 1 | Type :0 | Show error direct |
| 2 | Type:1 | Show error general “error system” on screen and save error into database |
| 3 | NVL | Error not valid |
| 4 | NWD | Error not withdraw money |
| 5 | NEM | Error not enough money |
| 6 | NRC | Error can not reject card |

## Loging mechanis

\_ Write to file:

*#include <iostream>*

*#include <fstream>*

|  |  |
| --- | --- |
| *using* *namespace* std;  *int* main () {  ofstream myfile;  myfile.open ("example.txt");  myfile << "Writing this to a file.\n";  myfile.close();  *return* 0;  } | [file example.txt]  Writing this to a file. |

# This project creat variable, write to file by transfer from variable to file

\_Read from file

#include<iostream>

#include<fstream>

using namespace std;

int main() {

ifstream myReadFile;

myReadFile.open("text.txt");

char output[100];

if (myReadFile.is\_open()) {

while (!myReadFile.eof()) {

myReadFile >> output;

cout<<output;

}

}

myReadFile.close();

return 0;

}

# This project creat variable, read from file by transfer from file to variable

# Database design HuyVQ4

## Entity Relationship Diagram



Figure 3 – Data overview

## Schema

**Overview**

|  |  |  |
| --- | --- | --- |
| **STT** | **Name** | **Description** |
| 1 | Customer | List all customers |
| 2 | Account | List all bank accounts, 1 customer can have more than one account |
| 3 | Card | List all ATM card use in system |
| 4 | CheckingACC | List all type of checking account and their OD value |
| 5 | SavingACC | List all type of saving account and their withdraw day limit value |
| 6 | ATM | List all ATM use in system |
| 7 | Log | Log any transaction of customer |
| 8 | LogType | Kind of Log: Withdraw, Check balance, Change PIN |

## Detail Schema

### Customer

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Customer | | | | |
|  | **Field name** | **Type** | **Max Length** | **Descrition** |
| 1 | CustID | int |  | Store ID of Customer |
| 2 | Name | nvarchar | 100 | Store name of Customer |
| 3 | Phone | nvarchar | 20 | Store phone number of Customer |
| 4 | Email | nvarchar | 100 | Store email address of Customer |
| 5 | IDCard | nvarchar | 20 | IDCard number of Customer |
| 6 | Addr | nvvarchar | 200 | Address of Customer |

### Account

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Account | | | | |
|  | **Field name** | **Type** | **Max Length** | **Descrition** |
| 1 | AccountID | int |  | Store ID of Account |
| 2 | CustID | int |  | Store ID of Customer |
| 3 | TypeID | int |  | Type ID of account (to know checking account or saving account) |
| 4 | Balance | float |  | Store amount of money left in account |

### Card

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Card | | | | |
|  | **Field name** | **Type** | **Max Length** | **Descrition** |
| 1 | CardID | nvarchar | 16 | Store Card number |
| 2 | Status | nvarchar | 30 | Store status of card: valid, blocked temporarily, blocked permanently |
| 3 | AccountID | int |  | Store Account ID that owns the card |
| 4 | PIN | nvarchar | 6 | Store the PIN code of Card |
| 5 | StartDate | datetime |  | Date of created |
| 6 | ExpiredDate | datetime |  | Date of expire. |

### CheckingACC

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| OverDraft | | | | |
|  | **Field name** | **Type** | **Max Length** | **Descrition** |
| 1 | TypeID | int |  | ID of type of checking account |
| 2 | OD | float |  | Value of overdraft limit that an account can overdraft for each type of checking accounts |

### SavingACC

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| OverDraft | | | | |
|  | **Field name** | **Type** | **Max Length** | **Descrition** |
| 1 | TypeID | int |  | ID of type of saving account |
| 2 | WDL | float |  | Value of withdraw limit for that an saving account can withdraw for a day. |

### ATM

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| ATM | | | | |
|  | **Field name** | **Type** | **Max Length** | **Descrition** |
| 1 | ATMID | int |  | ID of ATM |
| 2 | Branch | nvarchar | 50 | Branch name |
| 3 | Address | nvarchar | 100 | Location of ATM |
| 4 | Name | nvarchar | 100 | Name of ATM |
| 5 | Balance | float |  | Amount of money that remain in ATM currently |
| 6 | WithdrawLimit | float |  | Limit amount of money for each time the customer using ATM withdraw service |
| 7 | Qo50K | Int |  | Quantity of 50.000 Bill. |
| 8 | Qo100K | Int |  | Quantity of 100.000 Bill. |
| 9 | Qo200K | Int |  | Quantity of 200.000 Bill. |
| 10 | Qo500K | Int |  | Quantity of 500.000 Bill. |

### LogType

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| LogType | | | | |
|  | **Field name** | **Type** | **Max Length** | **Descrition** |
| 1 | LogTypeID | Int |  | ID of log type use in system |
| 2 | Description | nvarchar | 100 | Name or details of log type (Withdraw Money, Change PIN, View Balance) |

### Log

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Log | | | | |
|  | **Field name** | **Type** | **Max Length** | **Descrition** |
| 1 | LogID | Int |  | ID of log record |
| 2 | LogTypeID | Int |  | ID of log type |
| 3 | ATMID | Int |  | ID of ATM |
| 4 | CardID | nvarchar | 16 | ID of ATM Card |
| 5 | LogDate | datetime |  | Date when transaction happen. |
| 6 | Amount | float |  | Amount of transaction |
| 7 | Details | nvarchar | 100 | Description about transaction |

# Application Security

## . Main functions ‘s security matrix

|  |  |  |  |
| --- | --- | --- | --- |
| **Actor**  **Usercase** |  | **Customer** | **ATM** |
| Validation |  |  | x |
| Withdraw |  | x | x |
| CashTransfer |  | x | x |
| CheckBalance |  | x | x |
| ChangePIN |  | x | x |
| ViewHistory |  | x | x |
| Logging |  |  | x |

## Layout – ChienTQ1

* A Main screen is aligmented top center: use to display all the screen transaction to the customer
  + Bank logo on the top left of main screen
  + Bank Name on the top center of main screen
* There are six (6) fixed button (Button1 -> 6): 3 on the left and 3 on the right: diffirent function with each screen transaction
* Caculator keyboards are under the main screen: button 0 to 9 to input number 0 to 9.
  + Cancel button: used to cancel any transaction.
  + Clear button: used to refresh the input text.
  + Enter button: used to submit the input or confirm customer action.

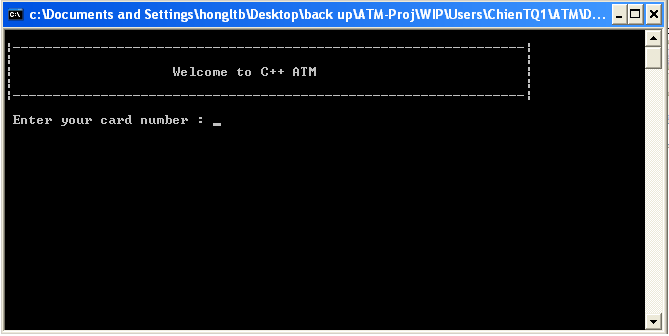
# Details function design

## Use case 01: Validation – HongLTB

### Validate Card

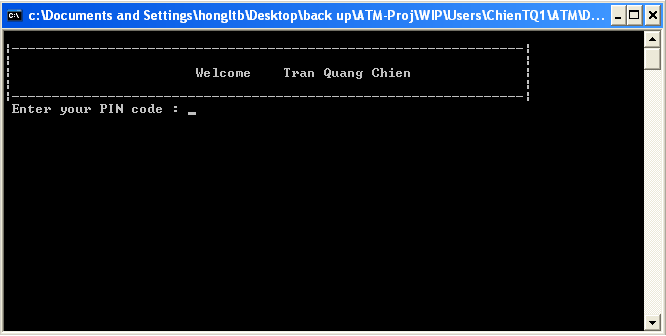
|  |  |
| --- | --- |
| **Name** | Validate Card |
| **Description** | This use case allows ATM system to check the cards which is inputted by user is valid or not. |
| **Actor** | ATM System |
| **Trigger** | When user press any key at the welcome screen |
| **Pre-condition** | The machine has started |
| **Post-condition** | If the card is valid then next step “Authenticaton” is activated, else eject the card. |

#### Screen Design & Data Description



**SC 01** : Main screen when system ask customer to enter card number

| **Item** | **Type** | **Description** |
| --- | --- | --- |
|  | Text | Display welcome message |
|  | Cursor | Position to type card number |



**SC 02** :Screen when system authenticate the customer

| **Item** | **Type** | **Description** |
| --- | --- | --- |
|  | Text | Display the name of card owner |

#### Activities Flow

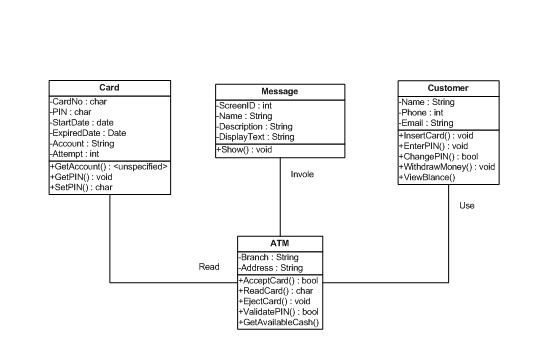


Figure 4.1:Activities Flow

#### Detail Processing

|  |  |  |
| --- | --- | --- |
| Activity | BR Code | Description |
| (3) | BR01 | Checking rules:   * IF <Card Reader> card’s number does not exist in DB THEN   + Set <<ShowedMessage>> = [Wrong Card Message].   + Send request to eject card. |
| (7) | BR02 | Checking rules:   * When user inserted the right card number into ATM   + Get card status information from database with card number inserted by customer. * IF the status is Blocked permanently THEN   + Set <<ShowedMessage>> = [Blocked Card & Swallow Message].   + Send request to swallow card. * IF the status is Blocked temporarily THEN   + Set <<ShowedMessage>> = [Blocked Card & Eject Message].   + Send request to eject card. |

#### Class diagram



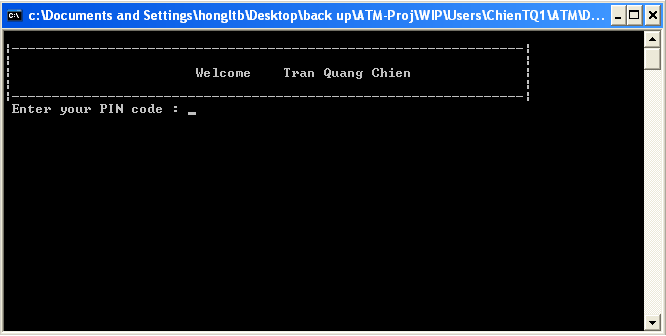
#### Sequense diagram



### Authentication

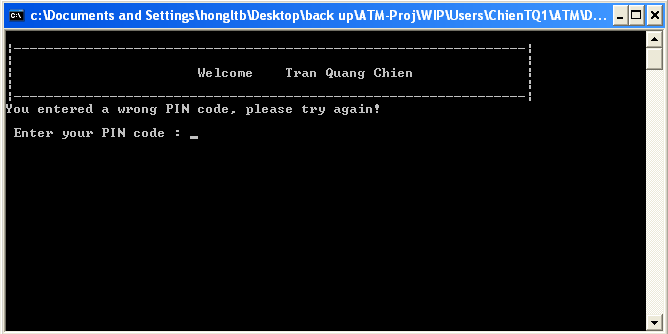
|  |  |
| --- | --- |
| **Name** | Authentication |
| **Description** | This use case allows ATM system to check the PIN which is inputted by customer is valid or not. |
| **Actor** | ATM System |
| **Trigger** | When user press ‘Enter’ button on the keyboard after input PIN code. |
| **Pre-condition** | The right card number has been inputted into ATM machine. |
| **Post-condition** | Customer was authenticated successfully, ATM system display the select transaction screen (main menu screen). |

#### Screen Design & Data Description



**SC 02** : Screen for customer enter their PIN

| **Item** | **Type** | **Description** |
| --- | --- | --- |
|  | Text | Display the name of card owner |

 SC 02 : Screen when customer enter WRONG PIN, system requires customer to re-enter PIN.

| **Item** | **Type** | **Description** |
| --- | --- | --- |
|  | Text | Message showed when customer input wrong PIN code |

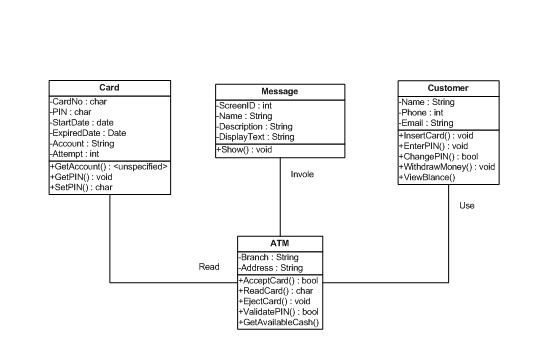
#### Activities Flow

Figure 4.2:Activities flow for authentication

#### Detail Processing

|  |  |  |
| --- | --- | --- |
| Activity | BR Code | Description |
| (4) | BR01 | Checking rules:   * Check PIN   + Get the PIN of Customer Card from Database.   + Compare the PIN get from database with the PIN customer just enterred. * IF the PIN customer inputted does not match with the PIN in database of customer Card THEN   + Set <<ShowedMessage>> = [Wrong PIN Message].   + Prompt customer to re-enter PIN. * IF the customer has inputted three times wrong PIN THEN   + Set <<ShowedMessage>> = [Eject Card Message].   + Eject card. |

#### Class diagram



#### Sequense diagram



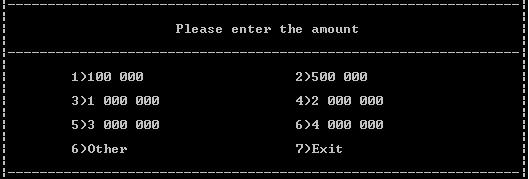
## Use case 02: Withdraw Money – NhungDTH1

### Withdraw

|  |  |
| --- | --- |
| **Name** | Withdraw Money |
| **Description** | This use case allows customer to withdraw moneys. |
| **Actor** | Customers |
| **Trigger** | When user chose ‘Withdraw’ in the screen. |
| **Pre-condition** | After Validation success, customer input amount money which they want to withdraw. |
| **Post-condition** | Receive money, Receive card , decide print receipt, transaction information saved in database. |

#### Screen Design & Data Description

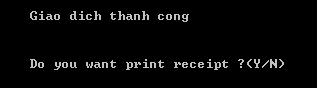
SC01 : Main screen when customer access “Withdraw”



SC02: Screen display when customer choose “Enter other” from SC01 screen.

C:\Documents and Settings\nhungdth2\Desktop\money2.JPG

SC03: Screen message transaction successful “Do you want print receipt”



#### Activities Flow

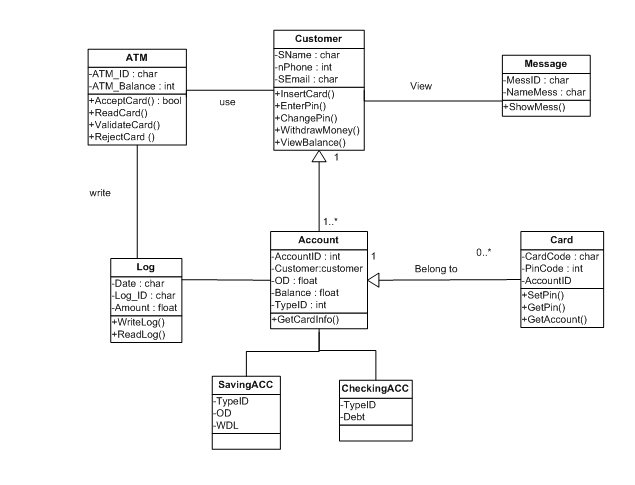


Figure 4.3:Activities flow of withdraw

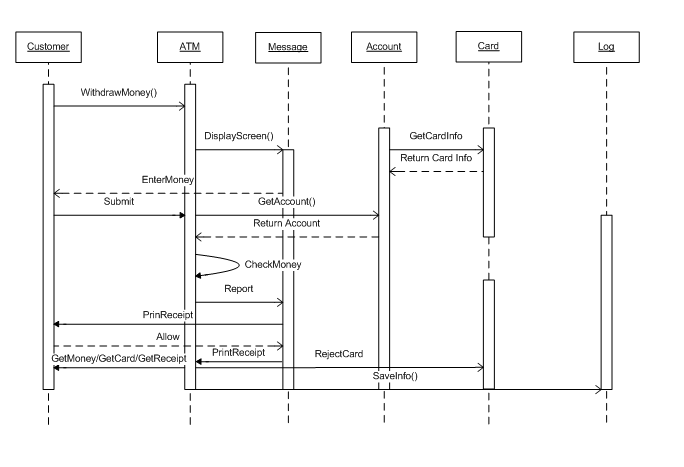
#### Detail Processing

|  |  |  |
| --- | --- | --- |
| Activity | BR Code | Description |
| (6) | BR01 | ValidateInput:   * System check balance: * IF enterCash > MaxValue * OR enterCash > CardValue + CardOD * OR enterCash mod 50.000 <> 0 THEN   + Set <<ShowedScreen>> = [Withdraw Failed Screen]   + Return FALSE |
| (9) | BR02 | Check balance:   * IF enterCash < AccountBalance THEN   + Set <<Account Balance>> = <<Account Balance>> - enterCash   + Write Log. * ELSE   + Set <<ShowedScreen>> = [Withdraw Failed Screen] |
| (11) | BR03 | Dispenser money:   * Calculate enterCash customer enter and MoneyType and Value, number of this MoneyType in this ATM, Return Cash to customer |

#### Class diagram



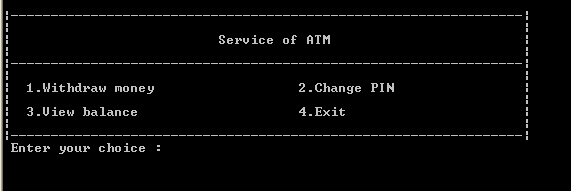
#### Sequense diagram



## Use case 03: ViewBalance – ChienTQ!

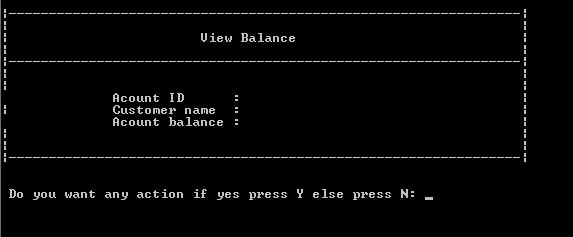
|  |  |
| --- | --- |
| **Name** | ViewBalance |
| **Description** | This use case allows Customer to view their Balance. |
| **Actor** | Customer |
| **Trigger** | When Customer choose “View balance” but in MenuScreen |
| **Pre-condition** | Customer had Validation to ATM. |
| **Post-condition** | ATM system display balance of Customer. |

#### Screen Design & Data Description



MenuScreen

| **Item** | **Type** | **Description** |
| --- | --- | --- |
| Service of ATM | Text | Name of screen |
| 1.Withdraw money | option | Move to withdraw function |
| 2.Change PIN | option | Move to change PIN function |
| 3.View balance | option | Move to view balance function |
| 4.Exit | option | Move to welcome screen |
| Enter your choice | Text | Enter number of your option |



BalanceScreen

| **Item** | **Type** | **Description** |
| --- | --- | --- |
| View Balance | Text | Name of screen |
| Account ID | Text | Show ID of the account |
| Customer name | Text | Show name of the customer |
| Account balance | Text | Show balance of the account |

#### Activities Flow

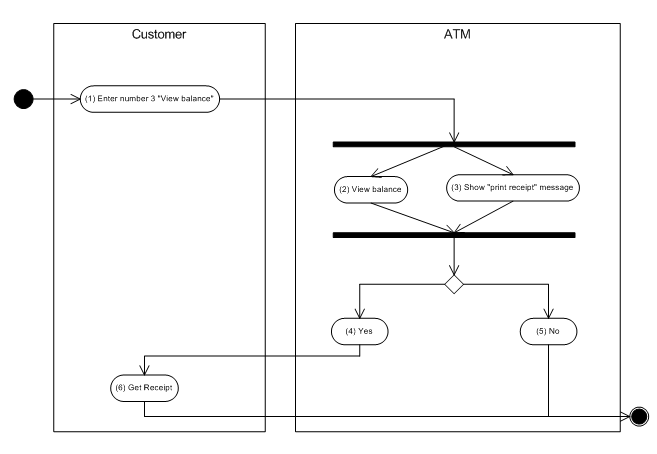
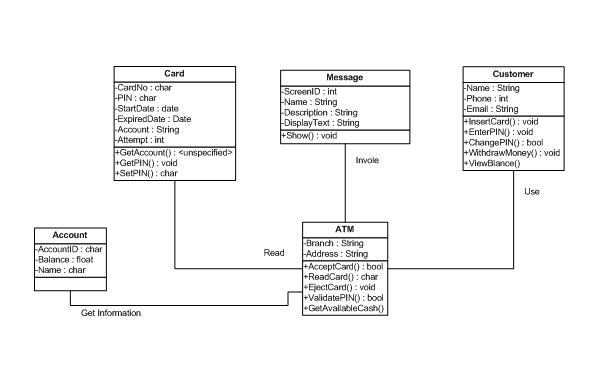


Figure 4.4:Activities flow of view balance

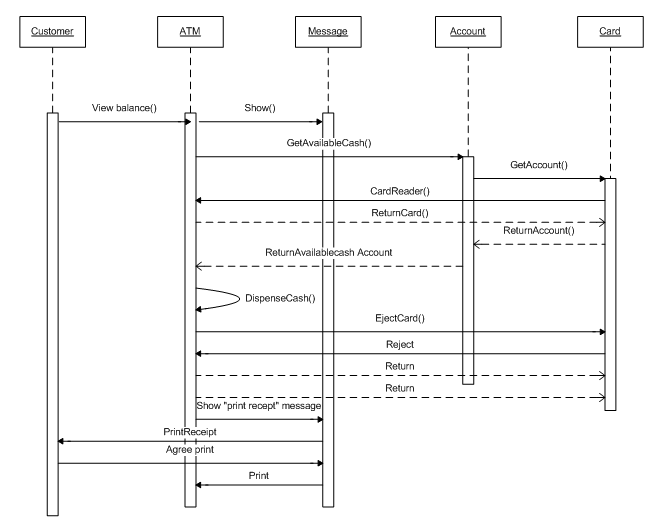
#### Detail Processing

|  |  |  |
| --- | --- | --- |
| Activity | BR Code | Description |
| (2) | BR01 | Display balance:   * + Get balance of customer from database and display to screen.   + Show customer name.   + Show customer ID. |

#### Class diagram



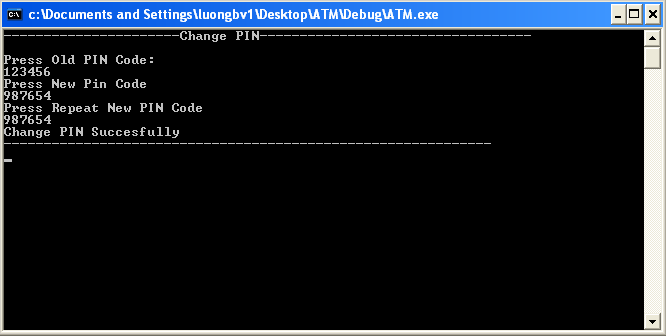
#### Sequense diagram



## Use case 04: Change PIN – LuongBV1

|  |  |
| --- | --- |
| **Name** | Change PIN |
| **Description** | This use case allows customer to change his/her PIN |
| **Actor** | Customers |
| **Trigger** | When user selcect 3 from main menu screen |
| **Pre-condition** | After Validation success |
| **Post-condition** | PIN of customer will change |

#### Screen Design & Data Description



**Screen 01:** Display screen require customer enter new PIN

| **Item** | **Type** | **Description** |
| --- | --- | --- |
|  | Text | Display welcome message |
|  | Cursor | Position to type PIN code number |
|  | Cursor | Position to repeat type PIN code number |
|  | Text | When change PIN succesful |

#### Activities Flow



Figure 4.7:Activities flow of Change PIN

#### Detail Processing

|  |  |  |  |
| --- | --- | --- | --- |
| Activity | BR Code | | Description |
| (4) & (8) &(11) | | BR01 | Check Syntax   * Get old PIN of customer in database and compare with old PIN he/she have just entered * IF old PIN have just entered haven’t in database or length of all PIN he/she just entered different 6 * Display “ERROR” * Return “Main Menu” screen |
| (12) | | BR02 | Change PIN   * Get new PIN and compare with new PIN which he/she re-enter * IF 2 new PINs are match THEN   + Display “Chage PIN Succsesfully”   + Return “Main Menu” |

#### Class diagram



#### Sequense diagram



## Use case 05: Logging – NhungDTH1

### Logging

|  |  |
| --- | --- |
| **Name** | Logging |
| **Description** | This use case allows ATM system to log all transaction has been executed by customer |
| **Actor** | ATM System |
| **Trigger** | When user finish any transaction with ATM system |
| **Pre-condition** | A transaction has finished. |
| **Post-condition** | New record will be inserted to table Log in database stored information about customer transaction: date, transaction type, amount … |

#### Screen Design & Data Description

No screen.

#### Activities Flow



Figure 4.8:Activities flow of Logging

#### Detail Processing

|  |  |  |
| --- | --- | --- |
| Activity | BR Code | Description |
| (3) | BR01 | * Insert new Log   + Set <<LogDate>> = [Current Time]   + Set <<LogType>> = [Transaction Type]   + Set <<ATM>> = [Current ATM Machine]   + Set <<Card>> = [Current Card]   + Set <<Amount>> = [Amount] |

#### Class diagram



#### Sequense diagram



# Configuration

## Application Configuration

|  |  |  |
| --- | --- | --- |
| Field | Values | Remark |
| MinWithDraw | 50.000 |  |
| MaxWithDraw | 10.000.000 |  |
| Number Record per Page | 5 |  |

## System Configuration

|  |  |  |
| --- | --- | --- |
| Field | Values | Remark |
| Date format | dd/MM/yyyy |  |
| Time format | hh:mm:ss |  |
| Format money | ##,###,###.00 |  |

# Packaging and Deployment – HuyVQ4



Figure 5.1:Deployment diagram

# Appendix

## Micrsoft Enterprise library