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

**Design Document**

**For AB-SERVICE**

*Prepared for*

**Thursday, May 28, 20155**

**Version 0.1**

*Prepared by*

**Team JavaFresher**

Table of Contents

[1 Introduction 4](#_Toc420583090)

[1.1 Purpose 4](#_Toc420583091)

[1.2 Scope 4](#_Toc420583092)

[1.3 Intended Audiences and Document Organization 4](#_Toc420583093)

[1.4 Acronyms and Abbreviations 4](#_Toc420583094)

[2 Architecture design 5](#_Toc420583095)

[2.1 Application Logical Architecture 5](#_Toc420583096)

[2.1 User case view 5](#_Toc420583097)

[2.1.1 Overview 5](#_Toc420583098)

[2.1.2 Description 6](#_Toc420583099)

[2.2 Architectural Representation 6](#_Toc420583100)

[2.2.1 Presentation Layer 7](#_Toc420583101)

[2.2.2 Business Layer 7](#_Toc420583102)

[2.2.3 Data Access Layer 8](#_Toc420583103)

[2.3 Packages/Components view 8](#_Toc420583104)

[2.2.4 UI Components 8](#_Toc420583105)

[2.2.5 Business Entities 9](#_Toc420583106)

[2.2.6 Data Access Object 9](#_Toc420583107)

[2.2.7 Exceptions Lib 9](#_Toc420583108)

[2.2.8 General Lib 9](#_Toc420583109)

[2.2.9 Logging Lib 9](#_Toc420583110)

[3 Technical Solutions 10](#_Toc420583111)

[3.1 Exception handling mechanism 10](#_Toc420583112)

[3.2 Loging mechanis 10](#_Toc420583113)

[4 Database design HuyVQ4 10](#_Toc420583114)

[4.1 Entity Relationship Diagram 10](#_Toc420583115)

[4.2 Schema 10](#_Toc420583116)

[4.3 Detail Schema 11](#_Toc420583117)

[4.3.1 Customer 11](#_Toc420583118)

[4.3.2 Account 11](#_Toc420583119)

[4.3.3 Card 11](#_Toc420583120)

[4.3.4 OverDraft Limit 11](#_Toc420583121)

[4.3.5 WithDraw Limit 11](#_Toc420583122)

[4.3.6 ATM 12](#_Toc420583123)

[4.3.7 Money 12](#_Toc420583124)

[4.3.8 Stock 12](#_Toc420583125)

[4.3.9 LogType 12](#_Toc420583126)

[4.3.10 Log 12](#_Toc420583127)

[4.3.11 Config 13](#_Toc420583128)

[5 Application Security 13](#_Toc420583129)

[5.1 . Main functions ‘s security matrix 13](#_Toc420583130)

[6 Interface Design 14](#_Toc420583131)

[6.1 Layout 14](#_Toc420583132)

[6.2 Themes 14](#_Toc420583133)

[7 Details function design 14](#_Toc420583134)

[7.1 Use case 01: Validation – HuyVQ4 14](#_Toc420583135)

[7.1.1 Validate Card 14](#_Toc420583136)

[7.1.2 Authentication 19](#_Toc420583137)

[7.2 Use case 02: Withdraw Money - LuyenNV 24](#_Toc420583138)

[7.2.1 Withdraw 24](#_Toc420583139)

[7.3 Use case 03: Check Balance – LuyenNV 30](#_Toc420583140)

[8 Configuration 34](#_Toc420583141)

[8.1 Application Configuration 34](#_Toc420583142)

[8.2 System Configuration 34](#_Toc420583143)

[9 Packaging and Deployment – HuyVQ4 35](#_Toc420583144)

# 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 Spring MVC framework with Java 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 simulator application in scope of the Assignment of FRESHER30.

## 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 | BE | Business Entities |

# 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 | Withdraw | Allow customer to withdraw money |
| UC03 | Check balance | Allow customer to check their account balance |

## 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 MVC 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 classes will be wisely used in the modules.

### Logging Lib

This package includes implemented logging classes.

# Technical Solutions

## Exception handling mechanism

The try-catch clauses should be kept to a minimum.

Whenever an exception raise , system will log exception detail to Windows event and redirect user to Error page to display standard error message!

## Loging mechanis

# Database design

## Entity Relationship Diagram

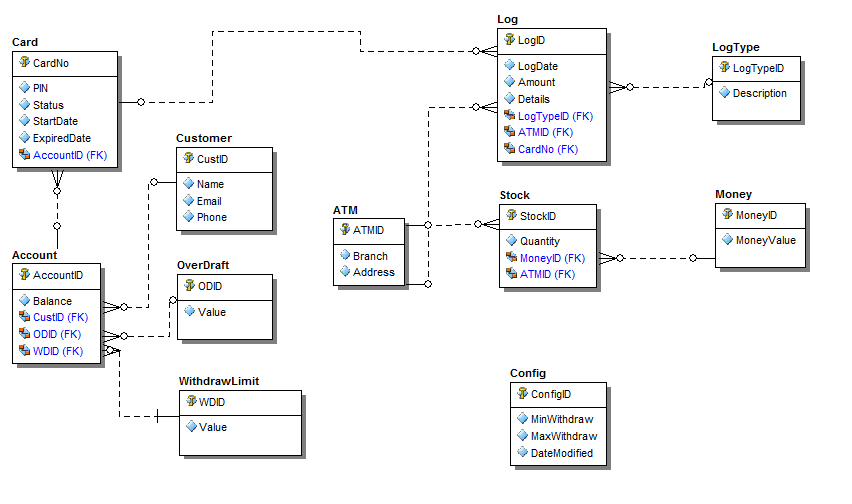


Figure 3 – Data overview

## Schema

**Overview**

|  |  |  |
| --- | --- | --- |
| **STT** | **Name** | **Description** |
| 1 | Customer | List all Customer |
| 2 | Account | List all account use in system |
| 3 | Card | List all ATM card use in system |
| 4 | OverDraft | Amount of money that an account can over draft |
| 5 | WithdrawLimit | Limit of amount that an account can withdraw for a day |
| 6 | ATM | List all ATM use in system |
| 7 | Money | Kind of money and value |
| 8 | Stock | Kind of money and quantity of each kind stored in each ATM |
| 9 | Log | Log any transaction of customer |
| 10 | LogType | Kind of Log: Withdraw, transfer, check balance, change pin |
| 11 | Config | Store all system config: minwithdraw, maxwithdraw, number records per seach results. |

## 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 | Varchar | 50 | Store phone number of Customer |
| 4 | Email | Varchar | 100 | Store email address of Customer |
| 5 | Addr | Nvarchar | 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 | AccountNo | Varchar | 50 | Store Account Number |
| 4 | ODID | Int |  | Store ID of OverDraft Limit |
| 5 | WDID | Int |  | Store ID of WithDraw Limit |
| 6 | Balance | Decimal |  | Store amount of money left in account |

### Card

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Card | | | | |
|  | **Field name** | **Type** | **Max Length** | **Descrition** |
| 1 | CardNo | Varchar | 16 | Store Card number |
| 2 | Status | Varchar | 30 | Store status of card: block, normal … |
| 3 | AccountID | Int |  | Store Account ID |
| 4 | PIN | Varchar | 6 | Store the PIN code of Card |
| 5 | StartDate | Datetime |  | Date of created |
| 6 | ExpiredDate | Datetime |  | Date of expire. |
| 7 | Attempt | Int |  | Entered invalid PIN times of customer |

### OverDraft Limit

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| OverDraft | | | | |
|  | **Field name** | **Type** | **Max Length** | **Descrition** |
| 1 | ODID | Int |  | ID of overdraft limit |
| 2 | Value | Decimal |  | Value of overdraft limit that an account can overdraft |

### WithDraw Limit

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| WithDraw | | | | |
|  | **Field name** | **Type** | **Max Length** | **Descrition** |
| 1 | WDID | Int |  | ID of withdraw limit |
| 2 | Value | Decimal |  | Value of withdraw limit that an 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 |

### Money

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Money | | | | |
|  | **Field name** | **Type** | **Max Length** | **Descrition** |
| 1 | MoneyID | Int |  | ID of money use in system |
| 2 | MoneyValue | Decimal |  | Value of kind of money |

### Stock

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Stock | | | | |
|  | **Field name** | **Type** | **Max Length** | **Descrition** |
| 1 | StockID | Int |  | ID of stock record |
| 2 | MoneyID | Int |  | Store ID of kind of money |
| 3 | ATMID | Int |  | Store ID of ATM |
| 4 | Quantity | Int |  | Store quantity of each kind of money in each ATM |

### 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 |

### 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 | CardNo | Varchar | 16 | Number of ATM Card |
| 5 | LogDate | Datetime |  | Date when transaction happen. |
| 6 | Amount | Decimal |  | Amount of transaction |
| 7 | Details | Varchar | 100 | Description about transaction |

### Config

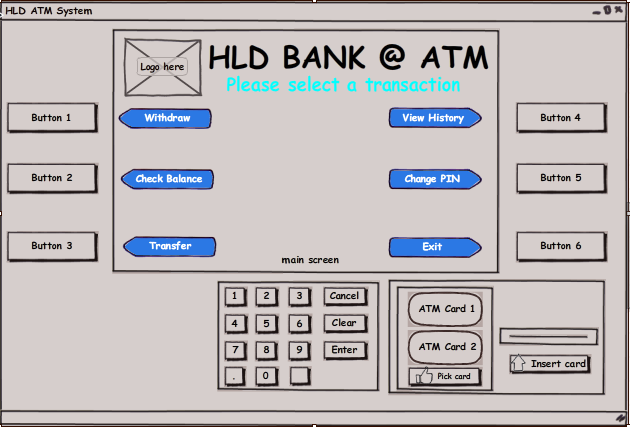
|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Config | | | | |
|  | **Field name** | **Type** | **Max Length** | **Descrition** |
| 1 | DateModified | Datetime |  | Date of modified |
| 2 | MinWithDraw | Decimal |  | Limit minimun of each withdrawal |
| 3 | MaxWithDraw | Decimal |  | Limit maximum of withdrawal for a day |
| 4 | NumPerPage | Int |  | Number of record will display per page in search results screen |

# Application Security

## Main functions‘s security matrix

|  |  |  |  |
| --- | --- | --- | --- |
| **Actor**  **Usercase** |  | **Customer** | **ATM** |
| Validation |  |  | x |
| Withdraw |  | x | x |
| CheckBalance |  | x | x |

# Interface Design



## Layout

* 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.

## Themes

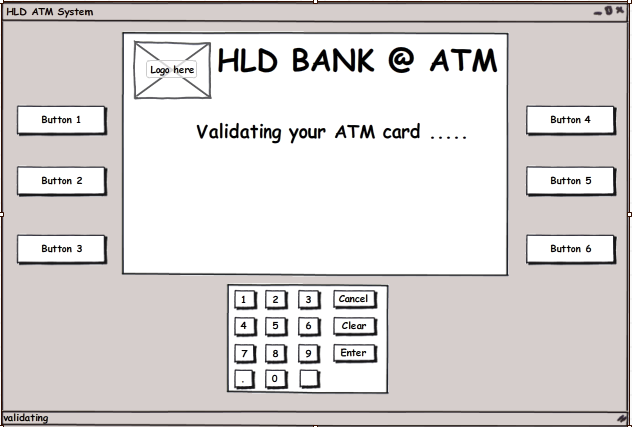
# Details function design

## Use case 01: Validation

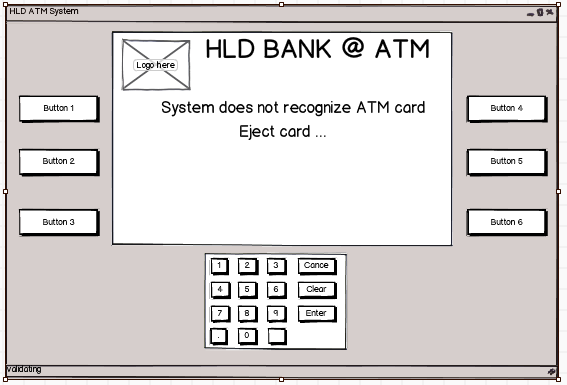
### Validate Card

|  |  |
| --- | --- |
| **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 clicked on ‘Insert Card’ button at main screen. |
| **Pre-condition** | The card has been inputted into ATM machine. |
| **Post-condition** | If the card is valid then next step “Authenticaton” is activated, eject the card if it is invalid. |

#### Screen Design & Data Description



SC 01 : Main screen when system validating customer’s card



SC 02 : Screen when system does not recognize customer’s card is a ATM card or customer insert a card this bank system does not support.

| **Item** | **Type** | **Description** |
| --- | --- | --- |
|  | button | Do nothing at this screen |
|  | button | Do nothing at this screen |
|  | button | Do nothing at this screen |
|  | button | Do nothing at this screen |
|  | button | Do nothing at this screen |
|  | button | Do nothing at this screen |
|  | button | Do nothing at this screen |
|  | button | Do nothing at this screen |
|  | button | Do nothing at this screen |
|  | button | Do nothing at this screen |
|  | button | Do nothing at this screen |
|  | button | Do nothing at this screen |
|  | button | Do nothing at this screen |
|  | button | Do nothing at this screen |
|  | button | Do nothing at this screen |
|  | button | Do nothing at this screen |
|  | button | Do nothing at this screen |
|  | button | Do nothing at this screen |
|  | button | Do nothing at this screen |

#### Activities Flow



Figure 4.1:Activities Flow

#### Detail Processing

|  |  |  |
| --- | --- | --- |
| Activity | BR Code | Description |
| (3) | BR01 | Checking rules:   * IF <Card Reader> cannot read the card’s number THEN   + Set <<ShowedScreen>> = [Wrong Card Screen].   + Send request to eject card. |
| (7) | BR02 | Checking rules:   * When user inserted the right card into ATM   + Get card information from database with card number like the number read from the card was inserted by customer. * IF Card number does not match with any card number in database THEN   + Set <<ShowedScreen>> = [Invalid Card Screen].   + Send request to eject card. |

#### Class diagram



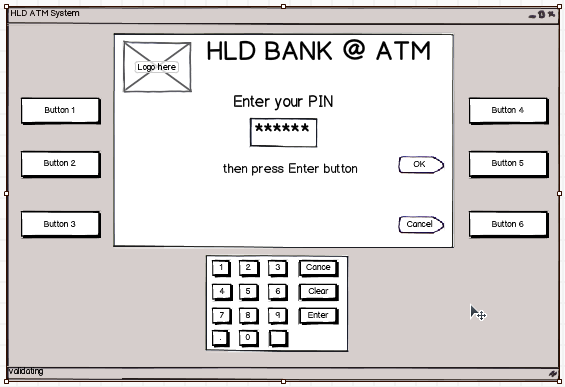
#### Sequense diagram



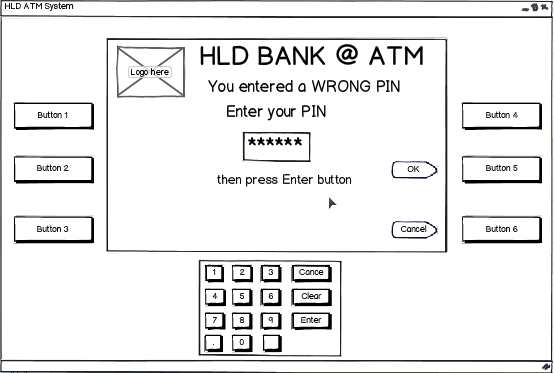
### Authentication

|  |  |
| --- | --- |
| **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 clicked on ‘Enter’ button or ‘Submit’ button at [Input PIN] screen. |
| **Pre-condition** | The card has been inputted into ATM machine. |
| **Post-condition** | Customer was authenticated successfully, ATM system display the select transaction screen. |

#### Screen Design & Data Description



SC 01 : Screen when customer enter their PIN



| **Item** | **Type** | **Description** |
| --- | --- | --- |
|  | button | Submit customer PIN |
|  | button | Cancel function, return to pre-screen |
|  | button | Cancel function, return to pre-screen |
|  | button | Clear text at text box |
|  | button | Submit customer PIN |
|  | Textbox | Input customer PIN |
| Others button | Button | Do nothing at this screen |
|  | button | Accept input of customer |
|  | button | Accept input of customer |
|  | button | Accept input of customer |
|  | button | Accept input of customer |
|  | button | Accept input of customer |
|  | button | Accept input of customer |
|  | button | Accept input of customer |
|  | button | Accept input of customer |
|  | button | Accept input of customer |
|  | button | Accept input of customer |

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



SC 03 : Screen when system validating customer PIN.

| **Item** | **Type** | **Description** |
| --- | --- | --- |
| All button | Button | Do nothing at this screen |

#### 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 <<ShowedScreen>> = [Wrong PIN Screen].   + Prompt customer to re-enter PIN. * IF the customer has inputted three times wrong PIN THEN   + Set <<ShowedScreen>> = [Block Card Screen].   + Set <Status> of Card = “Block”.   + Swallow card. |

#### Class diagram



#### Sequense diagram

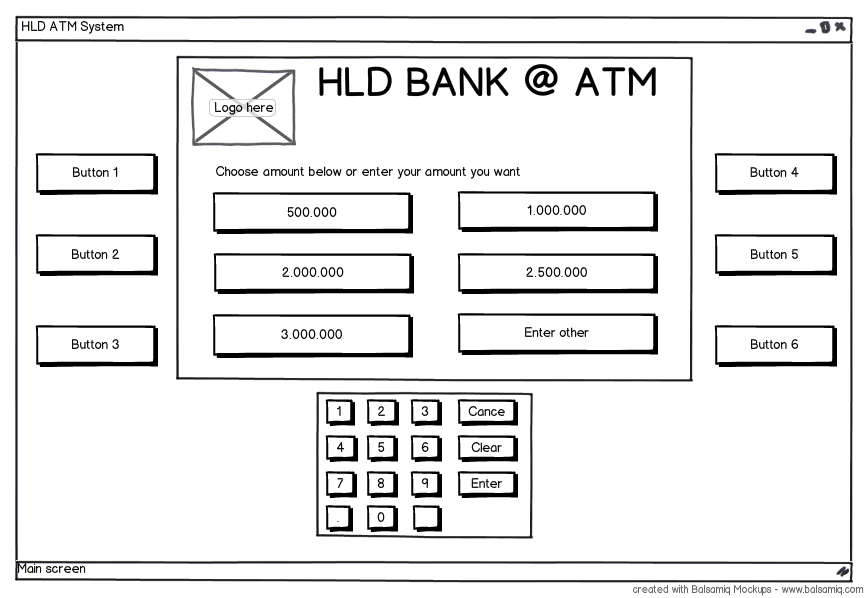


## Use case 02: Withdraw Money

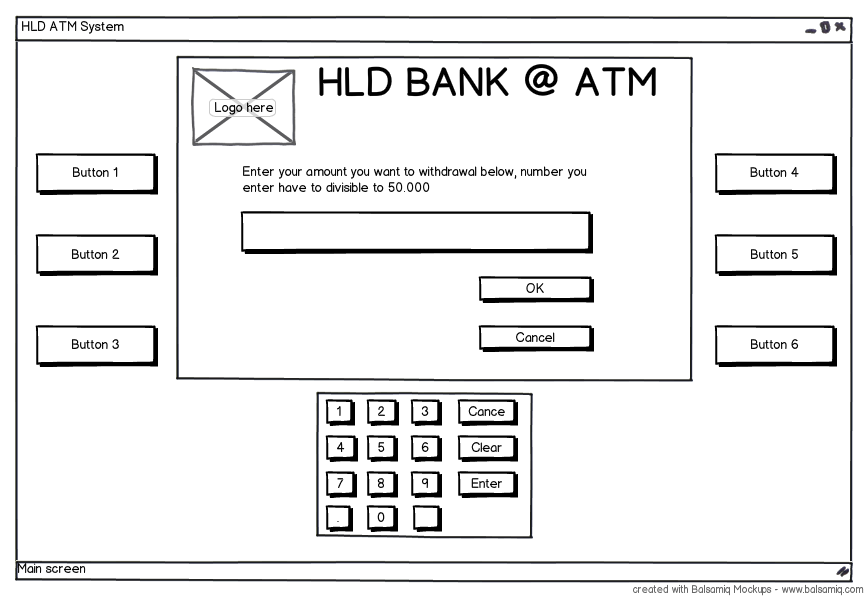
### Withdraw

|  |  |
| --- | --- |
| **Name** | Withdraw money |
| **Description** | This use case allows customer to withdraw moneys. |
| **Actor** | Customers |
| **Trigger** | When user clicking on ‘Withdraw’ button in the screen. |
| **Pre-condition** | After Validation success, customer input amount money which they want to withdraw. |
| **Post-condition** | Receive money, write log into system, decide print receipt. |

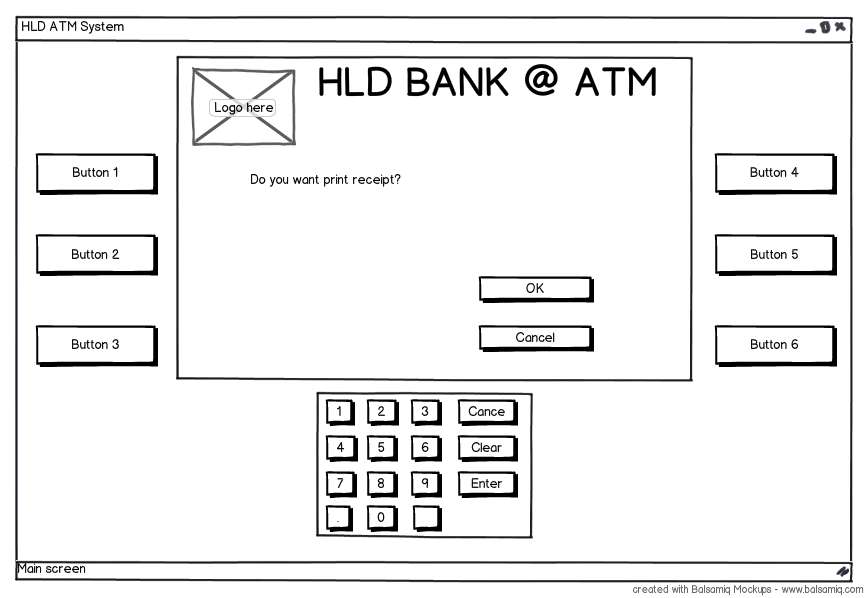
#### Screen Design & Data Description



SC01 : Main screen when customer access “Withdraw”



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



SC03: Screen confirm “Print receipt” when customer choose “OK” from SC 01 screen or “OK” from SC 02 screen

| **Item** | **Type** | **Description** |
| --- | --- | --- |
|  | button | Do nothing at this screen |
|  | button | Do nothing at this screen |
|  | button | Do nothing at this screen |
|  | button | Do nothing at this screen |
|  | button | Function “OK” |
|  | button | Function “Cancel” |
|  | button | Number “1” SC03 |
|  | button | Number “2” SC03 |
|  | button | Number “3” SC03 |
|  | button | Number “4” SC03 |
|  | button | Number “5” SC03 |
|  | button | Number “6” |
|  | button | Number “7” |
|  | button | Number “8” |
|  | button | Number “9” |
|  | button | Number “0” |
|  | button | Function “Cancel” |
|  | button | Clear text input |
|  | button | Function “OK” |

#### Activities Flow

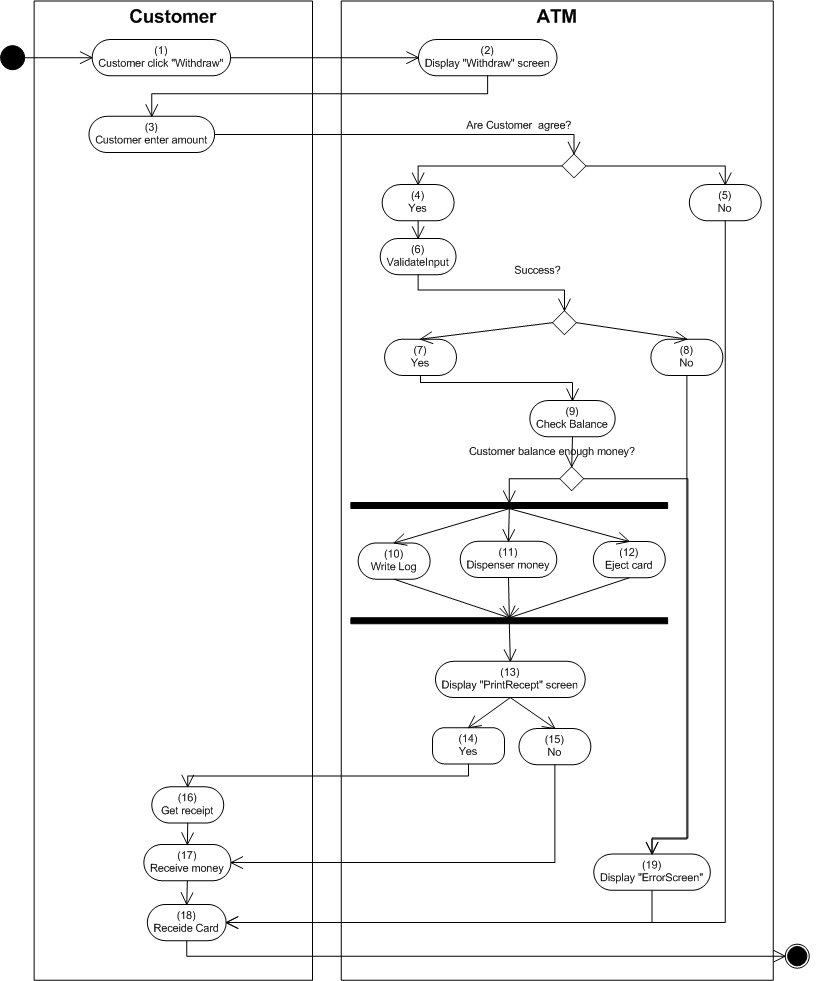


Figure 4.3:Activities flow of withdraw

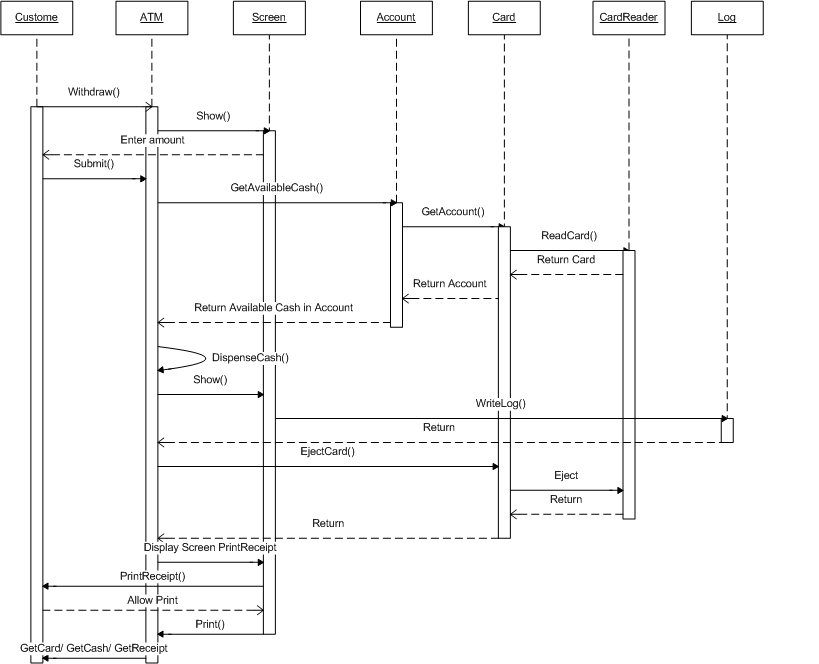
#### Detail Processing

|  |  |  |
| --- | --- | --- |
| Activity | BR Code | Description |
| (6) | BR01 | ValidateInput:   * System check balance: * IF enterCash > MinValue * OR enterCash <MaxValue * 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



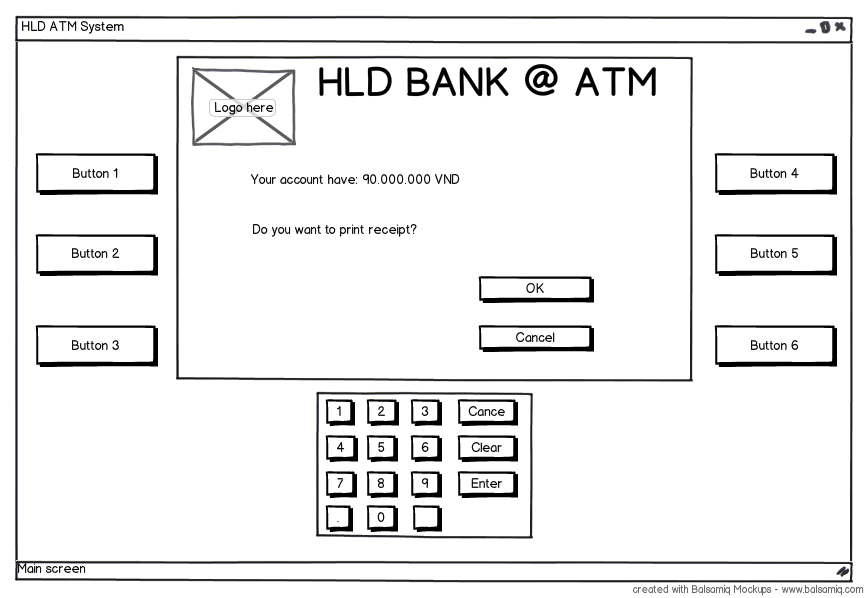
#### Sequense diagram



## Use case 03: Check Balance – LuyenNV

|  |  |
| --- | --- |
| **Name** | CheckBalance |
| **Description** | This use case allows Customer to check their Balance. |
| **Actor** | Customer |
| **Trigger** | When Customer click “Check balance” button in MainScreen |
| **Pre-condition** | Customer had Validation to ATM. |
| **Post-condition** | ATM system display balance of Customer. |

#### Screen Design & Data Description



SC 01 : Display Balance of customer and confirm PrintReceipt

| **Item** | **Type** | **Description** |
| --- | --- | --- |
|  | button | Do nothing at this screen |
|  | button | Do nothing at this screen |
|  | button | Do nothing at this screen |
|  | button | Do nothing at this screen |
|  | button | Do not print receipt |
|  | button | Allow print receipt |
|  | button | Do nothing at this screen |
|  | button | Do nothing at this screen |
|  | button | Do nothing at this screen |
|  | button | Do nothing at this screen |
|  | button | Do nothing at this screen |
|  | button | Do nothing at this screen |
|  | button | Do nothing at this screen |
|  | button | Do nothing at this screen |
|  | button | Do nothing at this screen |
|  | button | Do nothing at this screen |
|  | button | Do not print receipt |
|  | button | Do nothing at this screen |
|  | button | Allow print receipt |

#### Activities Flow

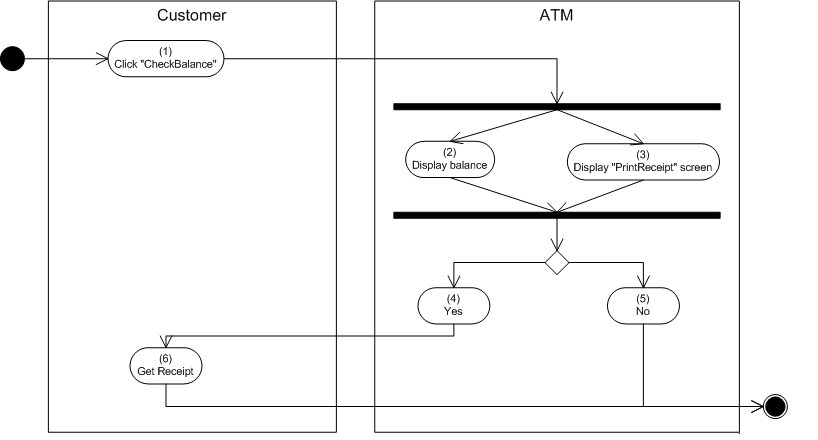
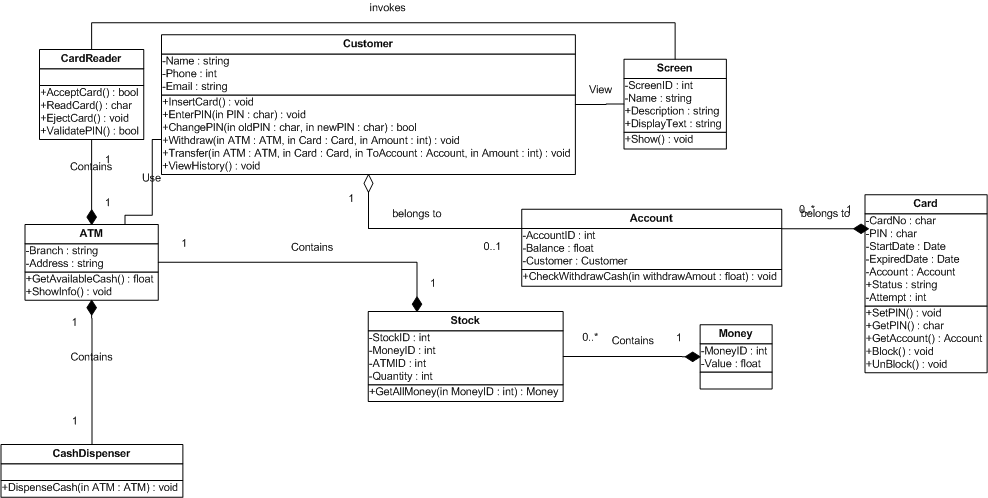


Figure 4.4:Activities flow of check balance

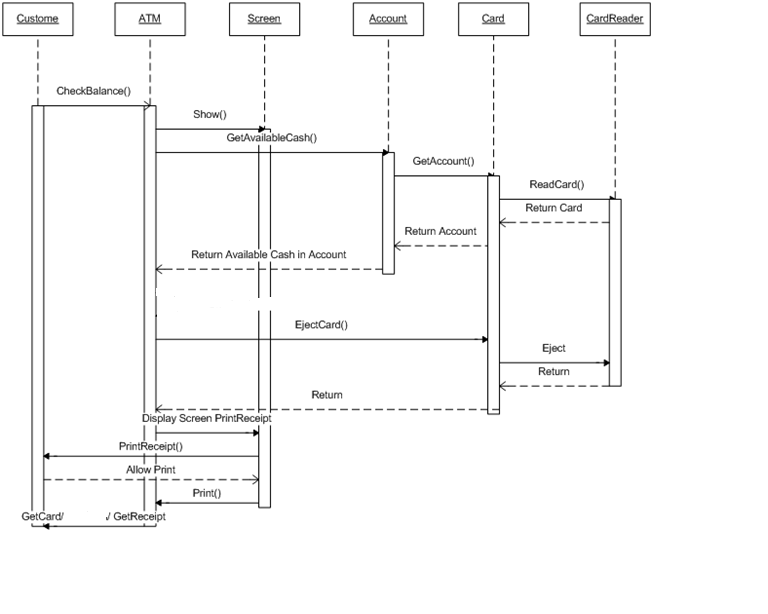
#### Detail Processing

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

#### 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



Figure 5.1:Deployment diagram