Rams Bank International

Automated Teller Machine (ATM) Software

Requirements Specifications

Version 1.0.0
February 12, 2016

TEAM
Baheer Kamal
Alessandro Profenna
Tolaz Hewa
Markus Jaeger
Otto Barbalho
Mikhail Korchevskiy

Table of Contents

1. INTRODUCTION

1.1 Purpose

The purpose of this document is to specify the software requirements and logical model of RBI's Automated Teller Machine (ATM) system in a clear and consistent manner.

1.2 Scope

This software will be used to implement the basic functions of a public Automated Teller Machine (ATM). This ATM will be equipped with a screen, a card reader, an NFC receiver, a small printer, a cash dispenser, and a keypad. The keypad will have numeric buttons with an enter key, cancel key, and clear key. Main functionalities include logging in with a valid bank account number and PIN, withdrawing/depositing funds, checking account balance, and printing a receipt.

1.3 Acronyms

This section contains the definitions of all used acronyms in this document:

ATM - Automated Teller Machine
RBI - Rams Bank International
GUI - Graphical User Interface
IEEE - Institute of Electrical and Electronics Engineers
PIN - Personal Identification Number
NFC - Near Field Communication

1.4 Overview

This document is structured according to the standards described by the IEEE. In Section 2, a description of the ATM software is given. In Section 3, the details of the software's functional requirements are given as use cases, along with the non-functional and user interface requirements.

2. GENERAL DESCRIPTION

2.1 Product Perspective

RBI's ATM Software is a standalone system that interacts only with a set of Canadian banking databases with access to account holders' banking numbers, PINs, and balance information.

2.2 Product Functions

The following is a list of the main functions of the ATM.
This software must be able to:

1. Receive and validate the user's 16-digit bank account number (either through the card reader, NFC receiver, or keypad)
2. Receive and validate the user's 4-digit PIN through the keypad
3. Display the available funds in the user's bank account
4. Withdraw the requested amount from the user's bank account and and dispense the value in cash accordingly
5. Accept cash/cheques and deposit their value in the user's bank account
6. Print a receipt with the details of the user's transaction

2.3. User Characteristics

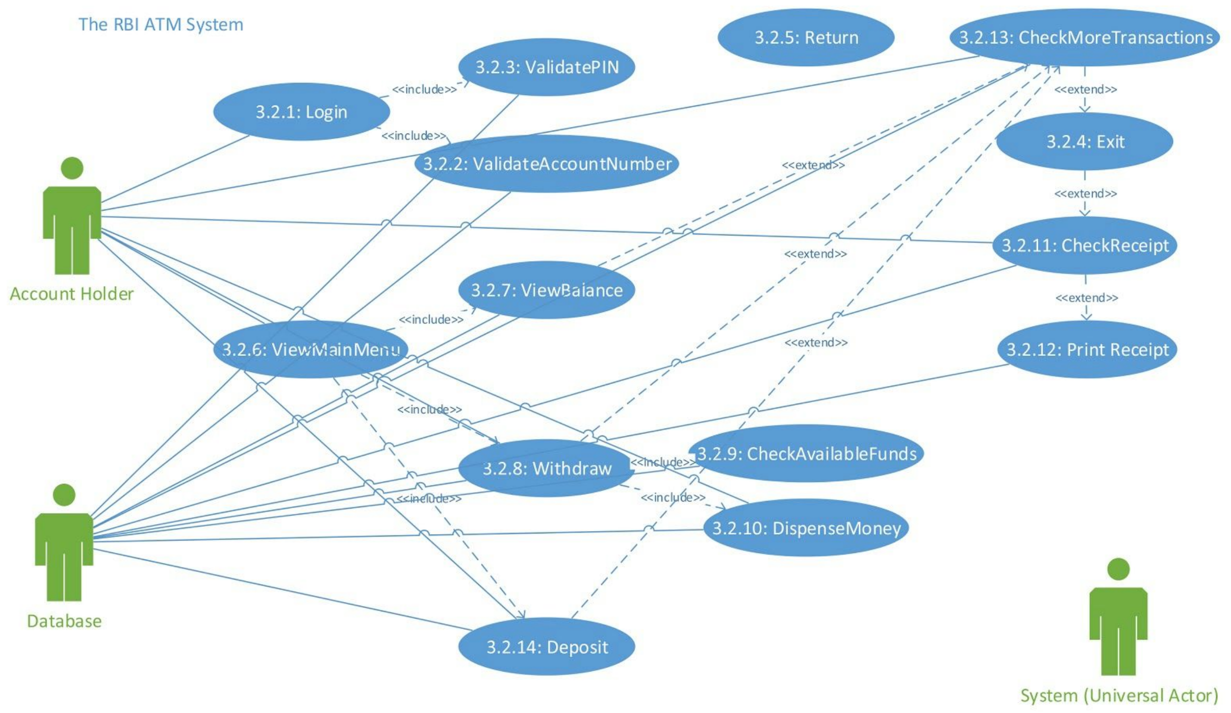
The users of this software will be any member of the general public with a valid banking card registered with a financial institution within our system.
The ATM is to be used by only one person at a time.

2.4. General Constraints

This ATM software must be designed in a highly compact and user-friendly way. The developers should also keep security as a top priority, in all aspects of account verification, transactions, and privacy protection.

3. SPECIFIC REQUIREMENTS

3.1. Use Case Model



*System is a universal actor, meaning it acts upon all the use cases.

3.2. Functional Requirements - Use Cases

In this section, all of the functional requirements of the software to be developed are given in the form of use cases. Each requirement is accompanied by a priority level. Each priority level is given an integer from 1 to 5, where 1 stands for the highest priority and 5 stands for the lowest priority. Requirements marked with a priority level 4 or 5 will only be implemented if time allows.

3.2.1 Login (include UC 3.2.2 ValidateAccountNumber and UC 3.2.3 ValidatePin)

Priority: 1
Brief Description: The user inputs login information and the system checks validity
Actors: Account Holder, System
Triggers: Inserting Card, NFC Card, Entering Account Number
Flow of Events:

Basic Flow – Login:

1. The use case starts when the user provides an account number
2. Include (UC 3.2.2 ValidateAccountNumber)
3. While System does not validate account number.
    - 3.1. Prompt to user that the account does not exist.
    - 3.2. User must input a new number to be checked.
    - 3.3. Include (UC 3.2.2 ValidateAccountNumber)
4. Else Switch to PIN entering window
5. Include (UC 3.2.3 ValidatePIN)
6. While System does not validate account PIN and tries are less than 3.
    - 6.1. Prompt to user that the account PIN is invalid.
    - 6.2. User must input a new PIN to be checked.
    - 6.3. Include (UC 3.2.3 ValidatePin)

Alternative Flow:

1. User presses ‘exit’ button (UC 3.2.4 Exit)

Special Requirements: The card number has to be 16 digits; PIN must be 4 digits long
Preconditions: Application must be running
Postconditions:

Success Postconditions:

1. Account is verified and open to ATM specific actions.

Failure Postconditions:

1. Account is not verified, therefore system goes back to log in.

Extension Points: none

3.2.2. ValidateAccountNumber

Priority: 1
Brief Description: System validates the customer account number by checking the database for existing matching account.
Actors: System, Database
Triggers: Customer has inputted an account number to be validated and presses ‘OK’
Flow of Events:

Basic Flow – Validate Account Number:

1. The Use Case starts when the system has received a 16 digit account number to be validated.
2. System searches for appropriate account corresponding to 16 digit account number in the database.
3. If account number exists
   3.1 Exit Use Case and return to Login menu indicating a successful validation.
4. Else account number does not exist
   4.1 Exit Use Case and return to Login menu indicating a failed validation.

Alternative Flow: None

Special Requirements: None
Preconditions: Application must be running, account number must be entered
Postconditions:

Success Postconditions:
1. Account is verified and open to ATM specific actions.

Failure Postconditions:
1. Account is not verified, therefore system goes back to log in.

Extension Points: none

3.2.3. ValidatePin

Priority: 1
Brief Description: System validates the inputted PIN by checking the database for correct a corresponding PIN number.
Actors: System, Database
Triggers: Customer has inputted a PIN number to be validated and attempts to log in.
Flow of Events:

Basic Flow – Validate Pin:

1. The Use Case starts when the system has received a 4 digit PIN number to be validated.
2. System compares inputted 4 digit PIN number corresponding to PIN stored within the 16 digit account number.
3. If inputted PIN is equal to account's stored pin
    3.1 Exit Use Case and return to Login menu indicating a successful validation.
4. Else inputted PIN is not equal to account's stored PIN
    4.1 Exit Use Case and return to Login menu indicating a failed validation.

Alternative Flow: None

Special Requirements: None
Preconditions: Application must be running and account number must be validated.
Postconditions:

Success Postconditions:
1. Account is verified and open to ATM specific actions.

Failure Postconditions:
1. Account is not verified, therefore system goes back to log in.

Extension Points: none

3.2.4. <u>Exit</u>

Priority: 1
Brief Description: Exits the application
Actors: System
Triggers: Pressing the ‘exit’ button
Flow of Events:

Basic Flow – View MainMenu:

1. The use case starts when account holder presses the ‘exit’ button
2. Prompt asking, “ Would you like to exit ATM now?”
3. If user selects “No”
   3.1 return to previous screen
4. System checks if transactions have been made
   4.1 if a transaction have been made, Extension point (UC 3.1.11 CheckReceipt)
5. System switches to a new window
6. The new window prints “Thank you for banking with RBI”
7. The application exits

Alternative Flow: None

Special Requirements: None
Preconditions: Application must be running
Postconditions:

Success Postconditions:
1. Application exits

Failure Postconditions:
1. Application does not exit

Extension Points: none

3.2.5. <u>Return</u>

Priority: 2
Brief Description: Goes back to the previous page
Actors: System
Triggers: When the user presses the ‘Return’ key on screen.
Flow of Events:

Basic Flow – Return:

1. Go to previous page

Alternative Flow: None

Special Requirements: None
Preconditions: Application must be running
Postconditions:

Success Postconditions:

1. Have returned to previous page.

Failure Postconditions:

1. Fails to return to previous page.

Extension Points: none

3.2.6. ViewMainMenu

Priority: 1
Brief Description: Displays the main menu of options for the Account Holder
Actors: Account Holder, System
Triggers: Successful login, Account Holder selects to do another transaction
Flow of Events:

Basic Flow – View MainMenu:

1. The use case starts when one of the triggers occur
2. The system displays the main menu of option buttons
3. The system waits until the Account Holder selects one of the buttons (Withdraw, Deposit, View Balance, Exit)
4. The system calls the selected button's function

Alternative Flow:

1. User presses 'exit' button (UC 3.2.4 Exit)

Special Requirements: None
Preconditions: Application must be running, Account must be logged in
Postconditions:

Success Postconditions:

1. Main menu is displayed and usable

Failure Postconditions:

1. Main menu is not displayed

Extension Points: none

3.2.7. ViewBalance

Priority: 2
Brief Description: Fetches balance from database and displays it on screen
Actors: System, Database
Triggers: Account Holder request account balance
Flow of Events:

Basic Flow - View Balance:

1. Request Account balance from database
2. Display Account balance.

Alternative Flow:

1. User presses ‘exit’ button (UC 3.2.4 Exit)

Special Requirements: none
Preconditions: Application must be running, account must be logged in
Postconditions:

Success Postconditions:

1. Account balance is displayed

Failure Postconditions:

1. Account balance is not displayed

Extension Points: none

3.2.8. Withdraw (include UC 3.2.9 CheckAvailableFunds and UC 3.2.10 DispenseMoney)

Priority: 1
Brief Description: The Account Holder withdraws a specific amount from their account
Actors: Account Holder, System, Database
Triggers: Selecting 'Withdraw' from the main menu
Flow of Events:

Basic Flow – Withdraw:

1. The use case starts when Account Holder selects the 'withdraw' button on the main menu
2. The System displays a window with predetermined amounts to withdraw and an option to input a custom amount (must be a multiple of 20)
3. If a custom amount is entered and 'enter' is pushed
    - 3.1. While the amount is not divisible by 20
        - 3.1.1. Display an invalid input message
        - 3.1.2. Clear the amount in the textbox
    - 3.2 Continue with custom amount.
4. Else if the account holder selects a predetermined amount to withdraw
    - 4.1 Continue with the predetermined value
5. Include (UC 3.1.9 CheckAvailableFunds)
6. Access database and withdraw the funds.
7. Include (UC 3.1.10 DispenseMoney)
8. The balance is decreased by amount selected/inputted through database

Alternative Flow:

1. User presses 'exit' button (UC 3.1.4 Exit)
2. User presses 'return' button (UC 3.1.5 Return)

Special Requirements: None
Preconditions: Application must be running, account must be logged in
Postconditions:

Success Postconditions:

1. The selected amount of money is withdrawn and the balance is updated

Failure Postconditions:

1. The selected amount of money cannot be withdrawn

Extension Points: none

3.2.9. CheckAvailableFunds (included in UC 3.2.8 Withdraw)

Priority: 2
Brief Description: The System checks if the selected amount to be withdrawn is within the account's balance
Actors: System, Database
Triggers: Withdraw amount selected
Flow of Events:

Basic Flow – Check Available Funds:

1. The use case starts when the System receives the amount to be withdrawn from (UC 3.2.8 Step #5)
2. The System gets the active account's balance
3. If the withdraw amount is greater than the account's balance
   - 3.1 Display error message "withdraw amount exceeds balance"
   - 3.2 Return to (UC 3.2.8 Step #2)
4. Return to (UC 3.2.8 Step #6)

Alternative Flow: None

Special Requirements: None
Preconditions: Application must be running, account must be logged in, withdrawal amount must be selected and passed
Postconditions:

Success Postconditions:

1. The selected amount of money is confirmed available to be withdrawn

Failure Postconditions:

1. The selected amount of money cannot be withdrawn

Extension Points: none

3.2.10 DispenseMoney (included in UC 3.2.8 Withdraw)

Priority: 2
Brief Description: The System dispenses the money through the money dispenser
Actors: Account Holder, System, Bank Database
Triggers: Verified withdrawal amount passed
Flow of Events:

Basic Flow – Dispense Money:

1. The use case starts when the System receives the amount to be dispensed from (UC 3.2.8 Step #6)
2. If the System does not have sufficient cash money
    - 2.1 Display error message “Sorry, this machine is out of money.”
    - 2.2 Return to UC 3.1.7
3. The System gets the required amount of money
4. The System dispenses the money through the money dispenser
5. While the money has not been removed from the dispenser
    - 5.1. The System displays a remove money message

Alternative Flow:

1. User presses ‘exit’ button (UC 3.2.4 Exit)

Special Requirements: None
Preconditions: Application must be running, account must be logged in, withdrawal amount must be confirmed and passed
Postconditions:

Success Postconditions:

1. The selected amount of money is dispensed

Failure Postconditions:

1. The selected amount of money cannot be dispensed

Extension Points: none

3.2.11 CheckReceipt

Priority: 3
Brief Description: Asks the Account Holder if they would like a receipt of their previous transactions
Actors: Account Holder, System, Database
Triggers: When the Account Holder decided to Exit or stop using the ATM
Flow of Events:

Basic Flow – Check for a Receipt:

1. The use case starts when the user has decided to exit the system and ATM
2. The System asks the Account Holder if they would like a printed receipt.
3. If Account holder says yes
   3.1. Extension Point (UC 3.2.12 PrintReceipt)
4. Exit use case and return to (UC 3.2.4 Exit)

Alternative Flow: None

Special Requirements: None
Preconditions: Application must be running, account must be logged in
Postconditions:

Success Postconditions:
1. Receipt will be printed.

Failure Postconditions:
1. DIsplay error

Extension Points: none

3.2.12 PrintReceipt

Priority: 3
Brief Description: Will print receipt with all transaction done on it.
Actors: System, Database
Triggers: When the Account Holder decides to have a receipt printed
Flow of Events:

Basic Flow – Print Receipt:

1. The Use Case starts when the user has decided to have a printed receipt.
2. System checks for sufficient paper to print receipt.
3. If there is enough paper to print receipt.
    3.1. Print receipt and the return to UC 3.2.4 Exit.
4. Else show prompt saying "Sorry but there is not enough paper to print a receipt at this time. Please excuse this inconvenience."
5. Return to UC 3.2.4 Exit.

Alternative Flow: None

Special Requirements: None
Preconditions: Application must be running, account must be logged in, transactions must have been made
Postconditions:

Success Postconditions:
1. Receipt will have been printed with all transactions done this session

Failure Postconditions:
1. Display error

Extension Points: none

3.2.13 CheckMoreTransactions

Priority: 3
Brief Description: The System checks if the Account Holder wants to do another transaction after completing a transaction
Actors: Account Holder, System, Database
Triggers: After a transaction completes
Flow of Events:

Basic Flow – Check for a Receipt:

1. The Use Case starts when a transaction completes
2. The system asks the Account Holder whether or not they want to do another transaction
3. If the Account Holder selects 'Yes'
    - 3.1. Return to (UC 3.2.6 ViewMainMenu)
4. Else
    - 4.1. Go to (UC 3.2.4 Exit)

Alternative Flow: None

Special Requirements: None
Preconditions: A transaction has been completed successfully
Postconditions:

Success Postconditions:
1. System returns to main menu or exits

Failure Postconditions:
1. Display error

Extension Points: none

3.2.14 Deposit

Priority: 4
Brief Description: The user gives the System an amount to deposit into their account (either cash or cheque through the deposit slot)
Actors: Account Holder, System, Database
Triggers: When the Account Holder selects 'Deposit' from the main menu
Flow of Events:

Basic Flow – Deposit:

1. The use case starts when the user selects 'Deposit' from the main menu
2. The system prompts the user to enter the amount they wish to deposit
3. The user enters the amount to deposit
4. While the System is waiting for the deposit to be put into the deposit slot
   4.1 The System prompts the Account Holder to put in their cash or cheque via envelope
5. The System updates the active Account's balance on database

Alternative Flow:

1. User presses 'Exit' button (UC 3.2.4 Exit)
2. User presses 'Return' button (UC 3.2.5 Return)

Special Requirements: None
Preconditions: Application must be running, account must be logged in, the user selects the 'Deposit' button on the main menu
Postconditions:

Success Postconditions:
1. The Account receives the deposit successfully
Failure Postconditions:
1. The deposit is not completed

Extension Points: none

3.3 Non-functional Requirements

3.3.1 The account number data field must accept exactly 16 numerical digits.

3.3.2 The PIN data field must accept exactly 4 numerical digits.

3.3.3 The PIN must be entered within 30 seconds.

3.3.4 The ATM must allow up to 3 tries to get the PIN right.

3.3.5 The ATM suspends the card if PIN is entered incorrectly 3 times.

3.3.6 The ATM must be able to be shut down and restarted.

3.3.7 The ATM must be able to respond to interactions and inputs within 100 ms.

3.3.8 The withdraw process must take less than 10 seconds when OK is pressed. Otherwise, the withdrawal process is cancelled.

3.3.9 The deposit function must wait for user to insert an envelope.

3.3.10 The ATM has to be able to read any valid account number within the database.

3.3.11 In case the ATM is unable to connect to the database, display error and retry every second for 5 seconds. If it is still unable to connect to the database, the atm software returns to the previous screen.

3.3.12 The ATM should be able to generate the receipt within 5 seconds.

3.3.13 After session exited, the ATM should be able to return to the main screen within 3 seconds.

3.3.14 If there is any communication error between the ATM and the database when withdrawing/depositing, an error message will pop up and the withdrawal/deposit will be canceled.

3.3.15 The system will attempt to read a card three times before producing an error.

3.4 User Interface Requirements

The following are high-level requirements for the ATM software's user interface. Specific details will be determined in the design stage.

3.4.1 Each editable field must be accompanied by a clickable keypad with clear, enter, and cancel buttons along with keys for numbers 0-9.

3.4.2 The 'login' screen will have an editable field for entering an account number and enter button.

3.4.3 The 'PIN' screen will have an editable field for entering a PIN, an enter button and a cancel button.

3.4.4 The 'main menu' screen will have buttons for withdraw, deposit, display balance, exit.

3.4.5 The 'withdraw' screen will have the values $40, $60, $80, $100, return, and input custom value in the buttons with an editable field.

3.4.6 The 'deposit' screen will have an editable field with the options to enter and return.

3.4.7 The 'more transactions' screen will have buttons for 'yes' and 'no' to the question, 'would you like to do another transaction?'

3.4.8 The 'print receipt' screen will have buttons for 'yes' and 'no' to the question 'would you like a receipt?'

3.4.9 The 'display balance' screen will show the balance and have a button to return to the main menu.