Bank Info Use Case

Document Information

|  |  |
| --- | --- |
| **Document Title** | Access Bank Information |
| **Document Owner** | Marvel Okafor |
| **Version** | 1 |
| **Status** | Completed |
| **Date** | 1/31/2022 |

1. Brief Description

This program starts when a user enters their bank ID from numbers 1 to 10. It shows them a main menu with the option to check their bank balance, withdraw or deposit money into their account and the option to exit. It ends when the user choses the option to exit by closing the program.

1. Actors

List any roles or systems involved with this process or use case. A person or system fulfilling a role will be the actor in one of the steps.

* Customer/User

1. Pre-Conditions

List anything that must be true before this process or functionality begins. Preconditions should be states that a system can validate to be true. A common example is that a specific Actor has logged into the System.

* The customer has to log in with an id from 0 to 9
* The customer has to create a name and password to access their account with the respective ID they chose
* The customer has to put positive numbers for withdrawing and depositing into their account
* The customer cannot withdraw above the money they have in their account

1. Basic Flow

The basic flow is the normal course of events, otherwise called the “happy path.” Ask yourself, what happens most of the time and you’ll discover the steps that belong here. You’ll want your basic flow to cover the full scope of activities between the starts when and ends when.

Create a numbered list of each step below. I recommend using the Word “numbered list” functionality to automatically number the list.

1. The Program starts and asks the user for a bank ID information.
2. The user puts a number between 0 and 9.
3. The program asks the user to create a name and password
4. The program calls the account class and hands the user that information which is a number and a default starting balance of $100.
5. The program calls the namePassword class and hands the user information which is the name and password and stores it in an array.
6. The program displays a menu with 4 options: Check Balance, Withdraw, Deposit, and exit.
7. If the user chooses the check balance option, the getBalance method is called by the program and displays the user’s current balance.
8. After showing the user’s current balance, the program displays the options listed in step 4.
9. If the user chooses the withdraw option, the program calls the checkWIthDrawal method in the Calculation class and asks the user for an amount to withdraw and then calls the withdraw method in the account class which does the subtraction from the account.
10. After the amount has been subtracted from the user’s current balance, the program displays the options listed in step 4.
11. If the user choses the deposit option, the program calls the checkDeposit method from the Calculations class and asks the user for an amount to deposit to the account. After that, the program calls the deposit method from the deposit class and does the addition to the account.
12. After the amount has been added to the user’s current balance, the program displays the options listed in step 4.
13. If the user choses the exit option, the user is sent back to the main screen with the option to log back in or exit the program totally.
14. Here, another user can log in with another number as long as it is between 0 and 9 and it has not been taken by a previous user.
15. Alternate/Exception Flows

An alternate flow is a variation from the basic flow. Alternatives can be triggered at any step in the basic flow and often reinsert the actors back into the basic flow.

An exception flow is an error, or a negative condition. When an exception is encountered, it prevents the process from finishing through to its conclusion until it’s addressed.

Number your alternate and exception flows to indicate the step at which the variation occurs. For example, a variation on step 3 could be listed as 3a and a second variation as 3b, and so forth.

Describe the alternate functionality and then identify at what step in the basic flow this variation picks back up. For exception flows that result in the use case ending, simply write, “Use Case Ends.”

6a. If the user chooses the check balance option, the getBalance method is called by the program and displays the user’s current balance. After this, the program returns to the main menu in step 6 of basic flow.

6b. If the user chooses the withdraw option, the program calls the checkWIthDrawal method in the Calculation class and asks the user for an amount to withdraw and then calls the withdraw method in the account class which does the subtraction from the account. After this, the program returns to the main menu in step 6 of basic flow.

6c. If the user chooses the deposit option, the program calls the checkDeposit method from the Calculations class and asks the user for an amount to deposit to the account. After that, the program calls the deposit method from the deposit class and does the addition to the account. After this, the program returns to the main menu in step 6 of basic flow.

6d. If the user choses the exit option, the user is sent back to the main screen with the option to log back in or exit the program totally.

6d. if the user inputs a number not in the specified range (1,2,3,4) then it tells the user to try again.

6b. InputMismatch Exception. If the user puts a string, the program catches the exception and ask the user to try again. If the user puts a negative number, the program asks the user for a positive number, and if the user puts a number greater than the user’s current balance the program asks the user to try again and displays what the user inputted and the user’s current balance.

6c. InputMismatch Exception. If the user puts a string, the program catches the exception and asks the user to put a number. If the user puts a negative number, the program asks the user for a positive number.

1. Post Conditions

Post-conditions indicate what must be true of the state of the system after the steps of the use case are complete. These should be true for the basic flow and all alternate flows. Exception flows may have different post-conditions or none at all.

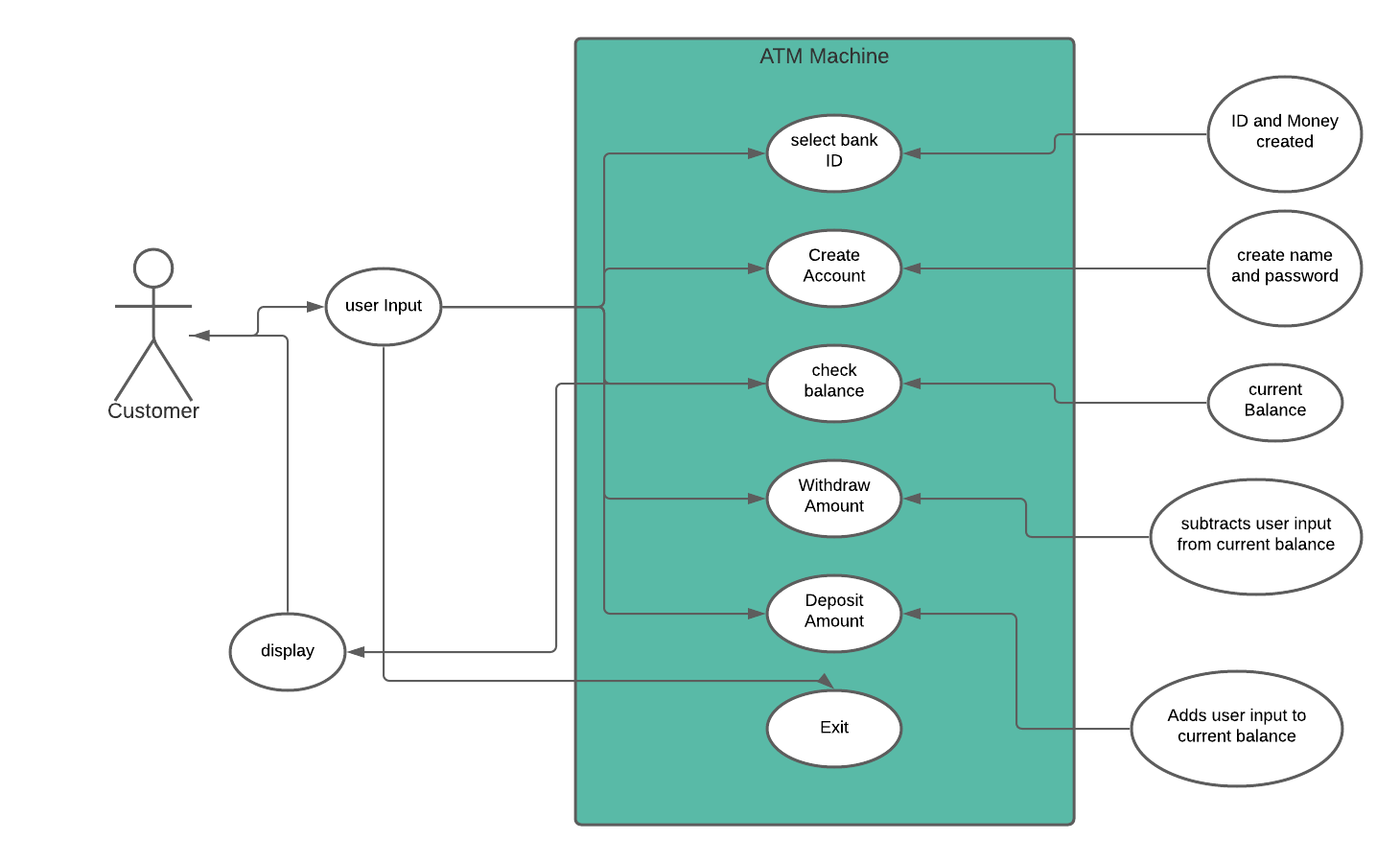
* Each user can still access their accounts as long as they don’t press 14 to end their program from the main menu.
* Each user can go back if they remember their ID and enter their name and password to access their account.

1. Supplemental Requirements

This is a special section I use to hold miscellaneous requirements related to the use case. Often you’ll find BAs including a Business Rules section or other collection of information related to the use case. These may or may not be actual requirements – you’ll want to establish a clear pattern and communicate that clearly and ensure it’s consistent with how your organization documents this type of requirement. I’ve also used this section to capture the most salient decisions and notes so they are stored right with the use case for future consideration.

1. They might be the option for the user to reset their password.
2. It is also possible for the user’s account to remain once the program ends totally but that will take a while to create and will need the work of an actual Database server.
3. Visual Model

Many use cases are enhanced by a visual model. A simple work-flow diagram can be used to visually show the sequence of steps and alternate and exception flows. A user interface mock-up can be used to show a possible representation of these user requirements in an interface (or a desired representation). In some organizations, a more formal UML diagram may be appropriate.



Revision History

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| V. | Date | Author | Description | Status |
| 1 | 1/30/2022 | Marvel | Finishing up the use case | In Progress |
|  |  |  |  |  |