# Use Case Details - Structured

Final versions of the requirements documents used for walk-throughs with the main stakeholders.

IDs can be anything from consecutive numbers to meaningful acronyms about what kind of use case this is and what system it is a part of.

## UC1 Withdraw Cash

### General info

#### Description:

The actor uses bank card to access menu and choose withdraw cash. They request cash amount and ATM dispenses it with a receipt.

#### Actors: Customer

#### Other stakeholders: Bank

#### Type: System

#### Pre-conditions:

Rules for beginning this use case: state of system prevents usage, must be testable. Or in a business use case, this must be the current state that has met a goal through another use case that this use case can now follow.

* **ATM.CashOnHand** must be $200 or more.

### Scope info

#### Level: Goal

#### Includes:

The use cases that are extracted out of this use case and given a special name, so they can be reused. They are required to be a part of this use case. This use case can be considered a grouped use case if it includes one of a group of partial goal use cases.

#### Included in:

The use case(s) that uses this one as a necessary part of it.

#### Use cases grouped by this ID:

If it doesn’t have an included group above, then it will be a category for several use cases.

#### Grouped by:

The group that has others like this one.

### Tracking info

#### Author: BA class of 9/8/19

#### Date created: 9/13/19

#### Date revised:

### Project info

#### Design constraints: ATM chassis Wonderware 38374FF33

#### Priority: 9

#### Value to sponsor:

* Reduce overhead
* Free up personnel for other tasks

#### Sponsor:

Who is accountable for this use case being delivered successfully?

### Course of Events

The sequence of tasks in conversation format between actor and system. For best linking to other steps, start each number with a system task except for the trigger. Combine actor responses to system events when well. Rules are placed under the task unless they can be reused and then they are referenced and placed in a separate file.

The number of tasks per number is usually small and starts with the system or the role. Tasks are individually stated so the system/role can do multiple things but in separate sentences. The last task will prepare the state of the system so that this use case can be performed again. There will be no condition statements to branch into two separate use cases. There may be a section that is removed to a named partial use case and called an <<include>> to shorten the detailed use case.

References that can be used here to document anything other than a functional requirement are:

1. **T#** - Text file item number – used for error messages and small prompts mostly on forms.
2. **D#** - Design file item number – used for web pages, full screen menus, etc.
3. **R#** - Report file item number – used for printed or on-screen report formats
4. **\* -**  a Data Dictionary item – used to refer to data description and validation so that the detail doesn’t have to be specified here. Also bolded and colored is good.
5. **Rule#** - Rule file item number – used to refer to process rules. Generally, this will follow one path only and another use case will pick up any other options. Some data validation rules find their way here but should be collected under the Data Dictionary. Unnumbered rules are not reusable and will just be defined below their functional requirement.
6. The use case starts when the actor taps the screen.
7. The system prompts actor to insert card. The actor inserts the card.
8. The system prompts for **Account.PIN**. The actor enters PIN.
9. The system verifies PIN with bank. The system logs the communication. The bank responds with **Account** information.
10. The system displays menu options (SD#1). The actor selects Withdraw Cash option.
11. The system prompts for account to withdraw from. The actor select account.
12. The system prompts for amount to withdraw (SD#2). The actor enters amount to withdraw.
13. The system validates the amount.
    * RULE – **Available funds** - Withdrawal amount must be less or equal to the **Account.AvailableFunds**
    * RULE – $**20 increments** - Withdrawal amount must be in $20 increments.
    * RULE – withdrawal limit – no more than $300 per withdrawal
    * RULE - daily limit – no more than $1000 total withdrawals per day starting at midnight local time
14. The system asks the bank to process the withdrawal. The system logs the communication. The bank confirms.
15. The system dispenses the cash. The system prompts actor to take cash. The system deducts amount from **ATM.CashOnHand**. The actor takes the cash.
16. The system prompts user whether to print a receipt or not. The actor accepts.
17. The system prints the receipt. The system prompts actor to take receipt. The actor takes receipt.
18. The system prompts user for another transaction. The actor declines.
19. The system returns card. The system prompts actor to take card. The actor takes card.
20. The system prompts with a thank you screen.
21. The system starts an inactivity timer.

### Alternate flows (errors, exceptions)

The error flows are where a rule is broken, or something interrupts the normal “happy path” of the course of events. This often is during communication or other type of I/O.

* **Cancel key pressed** (2-5, 7, 9)
* **Bad PIN** (4) – The system prompts that it’s bad PIN. The actor confirms. The use continues at #3. If PIN is invalid three times then ATM retains card, prompts user to call bank for card. The actor confirms. The system prompts with further actions and actor confirms. The use case continues at #16.
* **Insufficient funds** (8r1) – System prompts user that account has insufficient funds and shows balance. The actor confirms. The use case continues at #7.

### Alternate flows (extension points)

An exception to branching is when there is an optional <<extends>> of a partial use case. But the use case returns to where the option was taken.

* **Check balance** (5-7) – The actor chooses to check available funds from an account.

### Post-conditions

What are your tests that tell you that this is a successful completion of a use case? It may be a repetition of one of the tasks or a file or document that has been completed. But there are minimal ways to complete the goal and there are very excellent ways to complete it. Put both down. Some people use MoSCow – must have, should have, could have instead of min and max conditions.

* Bank has received transaction and processed it.
* Customer has received cash.

### Notes/ Special Requirements

Any kind of quality, capacity, security, availability, disaster recovery information that is because of this use case. Maybe you also have ideas about design, or people who need to be checked with, etc.

# Screen Design

1. 
2. 

# Data dictionary

## Account

Balance

Available funds

PIN – 4 digits only

Customer->

Transaction->

Card->

## ATM

Inactivity time

Bank ID

Current time

Cash on hand

Customer messages

Greeting

Ads

Customer->