Use Case Documentation for ATM

Developed by

Kasi Periyasamy

Created on Aug 10, 2008
Modified on Sep 25, 2014
Modified again on Feb 05, 2015
Modified again on Oct 29, 2018

1. About the document

This document describes the use case narratives for a simple ATM system. There are two types of users in this system – customers and managers. Any user (manager or customer) requires to login into the system before using it. The system will include four transactions that can be invoked by customers – *deposit*, *withdraw*, *check balance* and *transfer*. While withdrawing an amount from an account, a user is not permitted to withdraw more than the balance available in that account. Managers have all privileges as customers and hence they can invoke all use cases that customers do. In addition, managers will be able to *create* and *delete* accounts. The use case diagram, shown in Figure 1, includes three additional use cases – *validate account*, *update account* and *check account* These use cases are not accessible directly by the user but are used by other use cases.

1.1 Assumptions

1. Each account is assumed to be associated with a unique account identifier and a password so that a user will login to the account using these two identifications.
2. All transactions can be performed only after successful login into the system. However, the dependency relationship between the 'login' use case and other use cases will not be shown in the use case diagram because the other use cases will not directly implement the 'login' use case; rather, they will only use the result of login.
3. The database is assumed to be available all the time and hence there is no use case to connect to the database and/or to check the connection. This will be taken care of in the design phase.

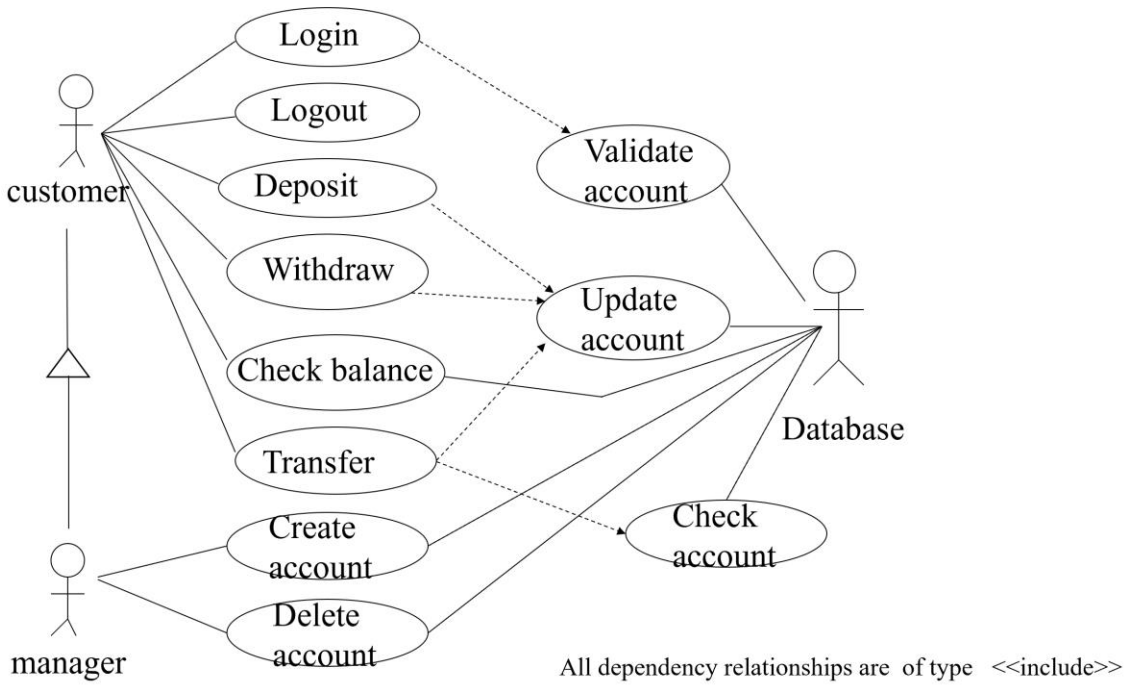

Figure 1: Use Case Diagram for ATM

2. Use Case Narratives

Use Case #:	ATM-L1
Use Case name:	Login
Purpose:	To login into an account.
Primary actor:	Customer
Secondary actor:	None
Input parameters:	Account number, Password
Output parameters:	None
Precondition:	Input parameters must be in appropriate format. Both input parameters must be given.
Post-condition:	If successful, a confirmation is sent to the user for successful login.
Successful scenario:	1. User submits the two parameters. 2. The two parameters are validated for format and completeness. 3. The two parameters are passed to the 'Validate account' functionality to validate user access. 4. The result of 'Validate account', a confirmation, is passed back to the user.
Exceptions:	One or both parameters are missing. One or both parameters are in incorrect format.
Additional remarks:	A welcome page may be displayed after successful login. Depending on the type of user, each such welcome page may be different.

Use Case #:	ATM-L2
Use Case name:	Logout
Purpose:	To logout from an account.
Primary actor:	Customer
Secondary actor:	None
Input parameters:	None
Output parameters:	None
Precondition:	User must have logged in.
Post-condition:	If successful, user will be logged out of the system and will not be able to access any functionality.
Successful scenario:	1. User submits request for logout. 2. The system will terminate all accesses for this user.
Exceptions:	User did not login.
Additional remarks:	None.

Use Case #:	ATM-D
Use Case name:	Deposit
Purpose:	To deposit a positive amount into an account.
Primary actor:	Customer
Secondary actor:	None.
Input parameters:	Amount
Output parameters:	None
Precondition:	Amount must be numeric and must be positive.
Post-condition:	If successful, the given amount is added to the balance of the account.
Successful scenario:	1. Ensure that the user has logged in. 2. User requests for deposit and provides a positive amount. 4. If the parameter is valid (format and value), the current balance in the logged in account is updated by adding the amount to the balance. The updated account is stored back into the database.
Exceptions:	User did not login into any account. Format error in amount parameter. Amount is zero or negative.
Additional remarks:	None

Use Case #:	ATM-W
Use Case name:	Withdraw
Purpose:	To withdraw a positive amount from an account.
Primary actor:	Customer
Secondary actor:	None.
Input parameters:	Amount
Output parameters:	None
Precondition:	Amount must be numeric and must be positive. Amount must be less than or equal to the available balance in the account.
Post-condition:	If successful, the given amount is subtracted from the balance of the account.
Successful scenario:	1. Ensure that the user has logged in. 2. User requests for withdrawal and provides a positive amount. 3. Ensure that the parameter is valid (format and value) and the balance in the account is greater than or equal to Amount. 3. If so, the current balance in the logged in account is updated by subtracting the amount from the balance. The updated account is

stored back into the database.

Exceptions:	User did not login into any account. Format error in amount parameter. Amount is zero or negative. Amount is more than the available balance in the account.
Additional remarks:	None

Use Case #:	ATM-C
Use Case name:	Check Balance
Purpose:	To retrieve the balance in an account.
Primary actor:	Customer
Secondary actor:	None
Input parameters:	None
Output parameters:	Balance
Precondition:	None
Post-condition:	If successful, the balance in the account is displayed to the user.
Successful scenario:	1. Ensure that the user has logged in. 2. User requests for check balance transaction. 2. System displays the balance in the account currently logged in.
Exceptions:	User did not login into any account.
Additional remarks:	The balance may be displayed on a pop-up window.

Use Case #:	ATM-T
Use Case name:	Transfer
Purpose:	To transfer a positive amount from one account to another account.
Primary actor:	Customer
Secondary actor:	None
Input parameters:	Destination account number, Amount
Output parameters:	None
Precondition:	Destination account number must be in proper format and must exist in the database. Amount must be numeric and must be positive.
Post-condition:	If successful, the given amount is subtracted from the balance of the logged in account and added to the balance of the destination account.
Successful scenario:	1. Ensure that the user has logged in. 2. User requests for transfer, and provides the destination account number and a positive amount as parameters. 3. The functionality ‘Validate account’ is invoked to check the

	format and existence of the destination account. 4. If 'Validate account' returns true, then Amount is checked for Format and is ensured to be positive. 5. If so, the balance in the logged in account is updated by subtracting the amount from the balance. In addition, the balance in the destination account is updated by adding the amount to the balance. Both the logged in account and the destination account are stored back into the database.
Exceptions:	User did not login into any account. Format error in amount parameter. Format error in destination account number. Destination account does not exist. Amount is zero or negative. Amount is more than the available balance in the logged in account.
Additional remarks:	None

Use Case #:	ATM-V
Use Case name:	Validate Account
Purpose:	To validate an account against the stored accounts in the database during login.
Primary actor:	None
Secondary actor:	Database
Input parameters:	Account number, Password
Output parameters:	Account
Precondition:	Both parameters are provided. Both parameters are in proper format.
Post-condition:	If successful, the account is retrieved from the database for further transactions.
Successful scenario:	
	1. One of the other use cases invokes this use case by passing the Account number and Password. 2. Ensure that both parameters are in proper format. 3. The account is retrieved from the database and is returned to the calling use case.
Exceptions:	One or both parameters are missing. One or both parameters are in incorrect format. Account does not exist for the given account number and password.
Additional remarks:	None

Use Case #:	ATM-C
Use Case name:	Check Account
Purpose:	To validate an account against the stored accounts in the database.

Primary actor:	None
Secondary actor:	Database
Input parameters:	Account number
Output parameters:	Account
Precondition:	Account number is in proper format.
Post-condition:	If successful, the account is retrieved from the database for further transactions.
Successful scenario:	1. One of the other use cases invokes this use case by passing the Account number. 2. The account is retrieved from the database and is returned to the calling use case.
Exceptions:	Account number format is invalid. Account does not exist for the given account number.
Additional remarks:	None

Use Case #:	ATM-U
Use Case name:	Update Account
Purpose:	To write an account back to the database.
Primary actor:	None
Secondary actor:	Database
Input parameters:	Account
Output parameters:	None
Precondition:	User must have logged into an account.
Post-condition:	If successful, the account is written back onto the database.
Successful scenario:	1. One of the other use cases invokes this use case. 2. The account that is currently logged into is stored back into the database.
Exceptions:	User did not login into any account.
Additional remarks:	None

Use Case #:	ATM-Cr
Use Case name:	Create Account
Purpose:	To create a new account.
Primary actor:	Manager
Secondary actor:	Database
Input parameters:	None
Output parameters:	Account
Precondition:	None
Post-condition:	If successful, a new account will be created and stored in the database.
Successful scenario:	

	1. Ensure that the user is logged in and the user is a manager. 2. A new account is created with a unique account number and an initial password. The account is then stored in the database.
Exceptions:	User did not login into any account. User is not a manager.
Additional remarks:	Account number can be auto-generated to ensure that it is unique. Initial password can also be auto-generated. The user may be asked to change the password when he/she uses it for the first time.

Use Case #:	ATM-Dl
Use Case name:	Delete Account
Purpose:	To delete an existing account.
Primary actor:	Manager
Secondary actor:	Database
Input parameters:	Account number
Output parameters:	Account
Precondition:	Account number is provided. Account number is in proper format. An account exists with the account number.
Post-condition:	If successful, the account with the given account number will be deleted from the database.
Successful scenario:	
	1. Ensure that the user is logged in and the user is a manager. 2. Ensure that Account number is provided and is in correct format. 3. Ensure that an account exists with the given Account number. 4. If so, delete the account from the database so that it is no longer accessible.
Exceptions:	Account number is not provided. Account number is in incorrect format. No account exists with given Account number. User did not login into any account. User is not a manager.
Additional remarks:	The selected account may be ‘soft-deleted’ meaning it is archived but is not accessible by any user to do any transaction. If the selected account has any balance left, it may be returned to the owner of the account but that communication is not part of this software.

.