***EXERCISE 5*** 🡪 <https://web.uettaxila.edu.pk/CMS/AUT2010/seSREbs/notes%5CLecture_12%20SRE.pdf>

**PROMPT:** The following should be textually analyzed and a use case diagram created containing several use cases. Identify the actors, use cases and associations. Please, use the user goal level approach. Also, please consider any possible generalization relationship between use cases or between actors, and any possible "include" or "extend" relationship between use cases. Please give me the PlantUML code for the use case diagram corresponding to the following text:

***Problem Description: A bank has several automated teller machines (ATMs), which are geographically distributed and connected via a wide area network to a central server. Each ATM machine has a card reader, cash dispenser, a keyboard/display, and a receipt printer. By using the ATM machine, a customer can withdraw cash from either a checking or savings account, query the balance of an account, or transfer funds from one account to another. A transaction is initiated when a customer inserts an ATM card into the card reader. Encoded on the magnetic strip on the back of the ATM card are the card number, the start date, and the expiration date. Assuming the card is recognized, the system validates the ATM card to determine: that the expiration date has not passed, that the user-entered PIN matches the PIN maintained by the system, that the card is not lost or stolen. The customer is allowed three attempts to enter the correct PIN; the card is confiscated if the third attempt fails. Cards that have been reported lost or stolen are also confiscated. If the PIN is validated satisfactorily, the customer is prompted for a withdrawal, query, or transfer transaction. Before a withdrawal transaction can be approved, the system determines that sufficient funds exist in the requested account, that the maximum daily limit will not be exceeded, and that there are sufficient funds available at the local cash dispenser. If the transaction is approved, the requested amount of cash is dispensed, a receipt is printed containing information about the transaction, and the card is ejected. Before a transfer transaction can be approved, the system determines: that the customer has at least two accounts and that there are sufficient funds in the account to be debited. For approved query and transfer requests, a receipt is printed, and the card ejected. A customer may cancel a transaction at any time; the transaction is terminated, and the card is ejected. Customer records, account records, and debit card records are all maintained at the server. An ATM operator may start up and close down the ATM to replenish the ATM cash dispenser and for routine maintenance. It is assumed that functionality to open and close accounts and to create, update, and delete customer and debit card records is provided by an existing system and is not part of this problem.***

ChatGPT

**ChatGPT**

Based on the provided text, I've identified the actors, use cases, and associations for the ATM system. Here is the PlantUML code for the corresponding use case diagram:

@startuml

left to right direction

actor "Customer" as Customer

actor "ATM Operator" as Operator

database "Central Server" {

folder "Customer Records" {

[Maintain Customer Records]

[Maintain Debit Card Records]

}

[Maintain Account Records]

}

rectangle "ATM Machine" {

usecase "Withdraw Cash" as Withdraw

usecase "Query Balance" as QueryBalance

usecase "Transfer Funds" as TransferFunds

usecase "Cancel Transaction" as CancelTransaction

}

Operator --> [Maintain Customer Records]

Operator --> [Maintain Debit Card Records]

Operator --> [Maintain Account Records]

Customer --> Withdraw

Customer --> QueryBalance

Customer --> TransferFunds

Customer --> CancelTransaction

Withdraw --> [Validate ATM Card]

QueryBalance --> [Validate ATM Card]

TransferFunds --> [Validate ATM Card]

CancelTransaction --> [Terminate Transaction]

[Validate ATM Card] --> [Check Expiration Date]

[Validate ATM Card] --> [Check PIN]

[Validate ATM Card] --> [Check Card Status]

[Check PIN] --> [Card Confiscated]

[Check PIN] --> Withdraw

[Check PIN] --> QueryBalance

[Check PIN] --> TransferFunds

[Withdraw] --> [Check Account Balance]

[Withdraw] --> [Check Daily Limit]

[Withdraw] --> [Check Cash Availability]

[Withdraw] --> [Dispense Cash]

[Withdraw] --> [Print Receipt]

[Withdraw] --> [Eject Card]

[TransferFunds] --> [Check Multiple Accounts]

[TransferFunds] --> [Check Debit Account Balance]

[TransferFunds] --> [Print Receipt]

[TransferFunds] --> [Eject Card]

[CancelTransaction] --> [Terminate Transaction]

[CancelTransaction] --> [Eject Card]

@enduml

**PROMPT:** This is too messy, Please consider the following summarized information of the problem: The customer is able to: Withdraws funds from the checking or savings account, Query the balance of an account, Transfer funds from one account to another. Consider that the action of PIN validation is common to all these 3 actions. The ATM customer interacts with the system via the ATM card reader, keyboard/display, cash dispenser, and receipt printer. Regarding the operator interactions: Shuts down the ATM, Replenishes the ATM cash dispenser, and Starts the ATM.

ChatGPT

**ChatGPT**

Certainly! Here is a more concise and simplified PlantUML code based on the summarized information:

@startuml

left to right direction

actor "Customer" as Customer

actor "ATM Operator" as Operator

rectangle "ATM System" {

usecase "Withdraw Funds" as Withdraw

usecase "Query Balance" as QueryBalance

usecase "Transfer Funds" as TransferFunds

usecase "Shutdown ATM" as ShutdownATM

usecase "Replenish Cash" as ReplenishCash

usecase "Start ATM" as StartATM

}

Customer --> Withdraw

Customer --> QueryBalance

Customer --> TransferFunds

Withdraw --> [Validate PIN]

QueryBalance --> [Validate PIN]

TransferFunds --> [Validate PIN]

[Validate PIN] --> [Common Validation]

Operator --> ShutdownATM

Operator --> ReplenishCash

Operator --> StartATM

@enduml

**PROMPT:** It is good, just that I think that the relationship between the 3 use cases of the customer and the Validate PIN use case should be with an "include" dashed arrow, since the PIN validation use case is included within the other 3 use cases, meaning that in order to achieve those actions, I first need to validate pin.

ChatGPT

**ChatGPT**

Absolutely, I appreciate the clarification. Here is the updated PlantUML code with "include" relationships between the customer actions and the "Validate PIN" use case:

@startuml

left to right direction

actor "Customer" as Customer

actor "ATM Operator" as Operator

rectangle "ATM System" {

usecase "Withdraw Funds" as Withdraw

usecase "Query Balance" as QueryBalance

usecase "Transfer Funds" as TransferFunds

usecase "Validate PIN" as ValidatePIN

usecase "Shutdown ATM" as ShutdownATM

usecase "Replenish Cash" as ReplenishCash

usecase "Start ATM" as StartATM

}

Customer --> Withdraw

Customer --> QueryBalance

Customer --> TransferFunds

Withdraw --> (ValidatePIN)

QueryBalance --> (ValidatePIN)

TransferFunds --> (ValidatePIN)

(ValidatePIN) --> [Common Validation] : <<include>>

Operator --> ShutdownATM

Operator --> ReplenishCash

Operator --> StartATM

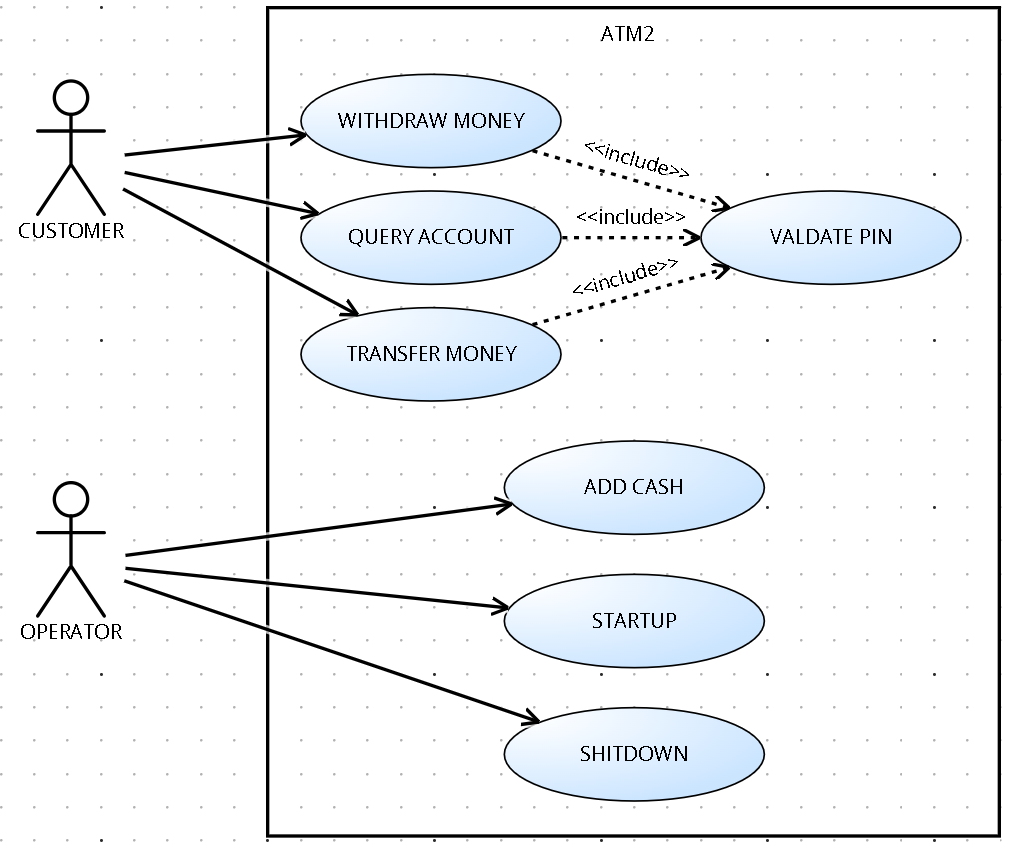
@enduml

**Analysis:**

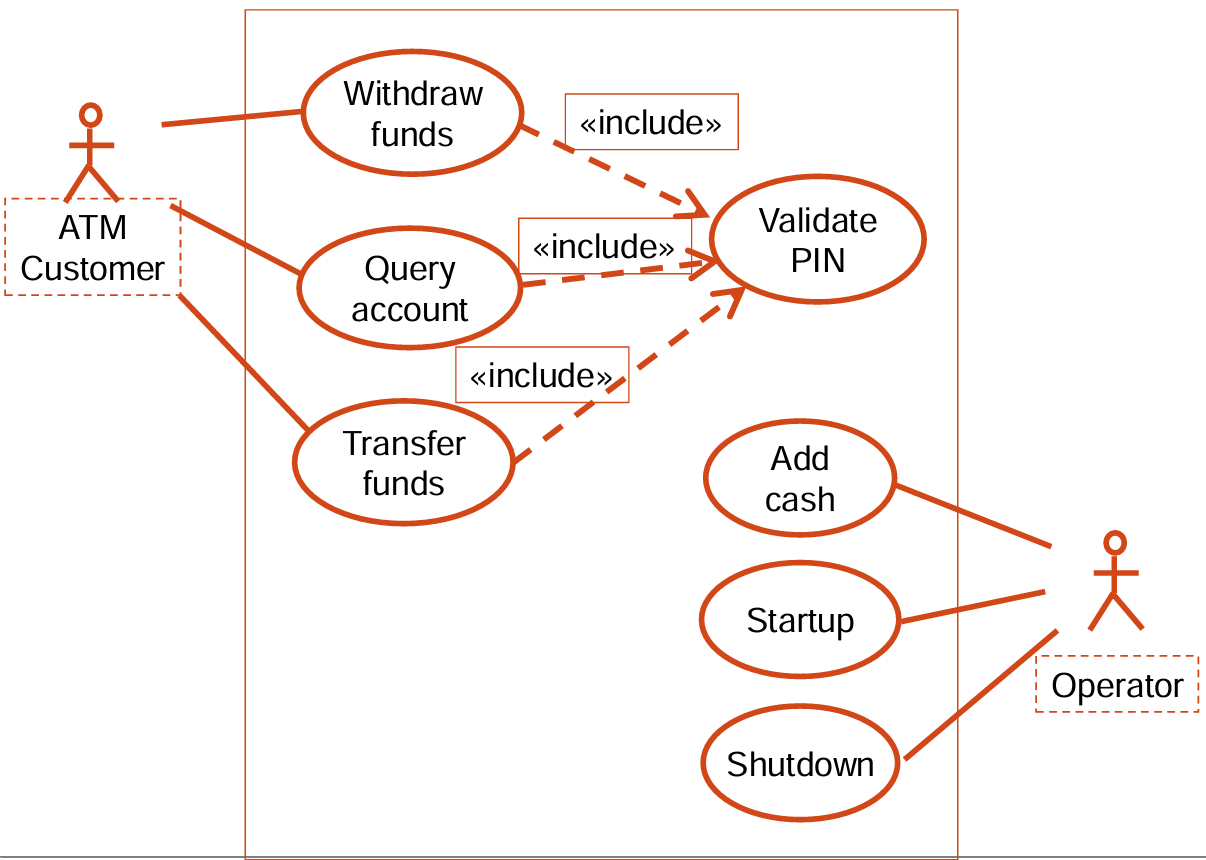
This exercise mirrors EXERCISE 4, although with slight variations in some use cases, actors, and the textual description of the problem. The approach taken involved testing ChatGPT's capabilities with two different methods. Initially, the entire problem description was presented, and ChatGPT was tasked with providing the relevant code. The resulting code in this first case was exceedingly disorderly, rendering the Use Case Diagram (UCD) incomprehensible. It featured unnecessary associations and an excessive number of use cases that were entirely superfluous. Dissatisfied with this outcome, I prompted ChatGPT again, opting for the second approach, which involved analyzing the provided actors and use cases from the description.

In this second case, the generated code exhibited a notable improvement over the first one in terms of both readability and correctness. However, a minor error persisted, specifically with the "inclusion" relation arrow, which was not coded as a dashed arrow. Nevertheless, the word "inclusion" was written above the arrow, rendering it understandable for the user. This solution closely resembled the original UCD solution provided by the exercise, with the refined code aligning closely with the analyzed description of actors and use cases.

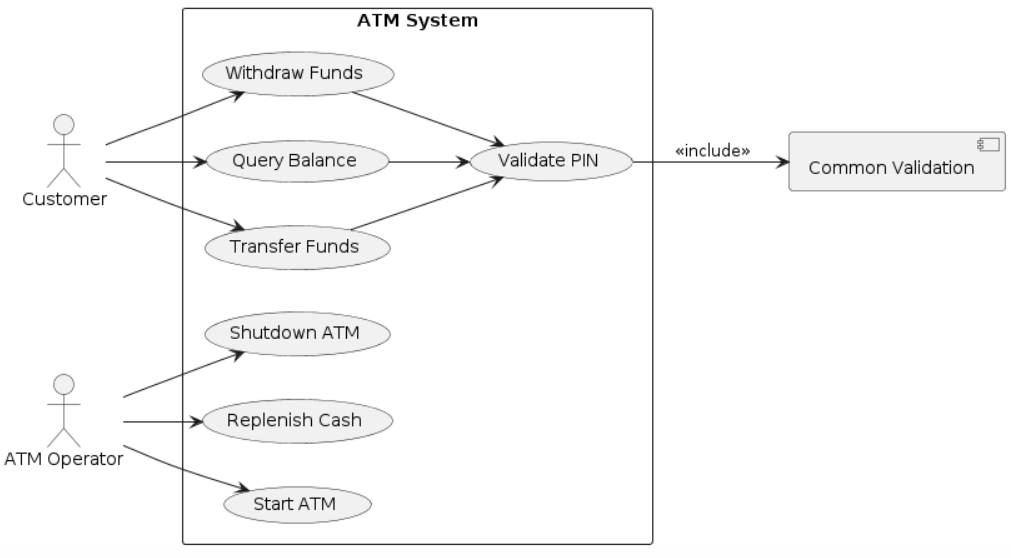
**MY SOLUTION:**

****

**REFERENCE SOLUTION:**

****

**LLM SOLUTION:**

****