Report on Computational Thinking and the ATM Withdrawal System

1. Overview

A system known as an Automated Teller Machine (ATM) enables customers to conduct banking operations without a human teller present, including cash withdrawals and account balance checks. One of the best examples of computational thinking is the ATM withdrawal procedure. Clear inputs (card, PIN, amount), processing (authentication, balance check, transaction update), and output (cash dispensing, updated balance) are all part of it. This system demonstrates how automation and logical reasoning simplify routine financial transactions.

2. Analysis of the Problem a) Abstraction

Abstraction eliminates extraneous details while emphasizing the crucial ones. Important components consist of: - The user inserts the card and inputs the PIN.

The system checks the database for the PIN.

The user inputs the amount of the withdrawal.

The system verifies cash availability and balance.

The balance is updated and cash is disbursed.

Among the overlooked details are the internal configuration of the bank server.
The ATM's internal cash counting mechanism.

The backend transaction records of the bank.

b) Disintegration
The ATM withdrawal procedure is divided into more manageable, smaller steps:
1. Card Insertion & Authentication: Find the card and check the PIN.
2. Menu Display: Display options such as Mini Statement, Check Balance, and Withdraw.
3. Withdrawal Request: The user enters the sum.
4. Balance Verification: Verify that there is sufficient funds in the account.
5. Cash Dispensing: Give out the specified sum.
6. Update Balance & Print Receipt: Show the remaining balance after deducting the amount that was taken out.
7. Exit & Card Ejection: Complete the session by safely ejecting the card.

c) Identifying Patterns

This procedure is similar to other transaction systems in the following ways:
User authentication is comparable to PIN/password online logins.

Balance checking is similar to using shopping apps to check inventory before sales.
Transaction Confirmation: In e-commerce, this is comparable to payment confirmation.
Understanding these patterns facilitates the reuse of code and logic across different systems.

3. Design of the Solution
Description of a Flowchart (Text Representation)
Start → Insert Card → Enter PIN → Verify PIN → If Not Correct → Show "Invalid PIN" → Try Again/Exit → If Correct → Show Menu → Choose "Withdraw Cash" → Enter Amount → Verify Balance → If not enough, show "Insufficient Funds"; if enough, show Cash Disbursement → Balance Update → New Balance Display → Print Receipt → Card Ejection → Finish

Pseudocode:

```
BEGIN

DISPLAY "Insert Card"

READ card_number

DISPLAY "Enter PIN"

READ pin

IF pin == stored_pin THEN

DISPLAY "Select Transaction: 1. Withdraw 2. Check Balance"

READ choice

IF choice == 1 THEN

DISPLAY "Enter Amount"

READ amount

IF amount <= balance THEN

balance = balance - amount

DISPLAY "Collect Your Cash"

DISPLAY "Remaining Balance:", balance

ELSE

DISPLAY "Insufficient Balance"
```

```
ENDIF
ELSE IF choice == 2 THEN
DISPLAY "Your Balance:", balance
ENDIF
ELSE
DISPLAY "Invalid PIN"
ENDIF
END
```

. Implementation (Python Code Example)

```
# Simple ATM Withdrawal Simulation
# Author:

balance = 10000  # initial account balance
stored_pin = "1234"  # pre-set PIN
pin = input("Enter your PIN: ")
if pin == stored_pin:
    print("1. Withdraw Cash")
    print("2. Check Balance")
    choice = input("Enter choice: ")
    if choice == "1":
        amount = int(input("Enter amount to withdraw: "))
        if amount <= balance:
            balance -= amount
            print("Please collect your cash.")
            print("Remaining Balance:", balance)
        else:
            print("Insufficient funds.")
    elif choice == "2":
```

```
        print("Your current balance is:", balance)
    else:
        print("Invalid choice.")
else:
    print("Invalid PIN. Transaction failed.")
```

5. Reflection

I was able to observe how computational thinking streamlines practical procedures by developing the ATM withdrawal system. Finding the necessary components while disregarding extraneous hardware details was the biggest obstacle. Through decomposition, I discovered how to divide the system into smaller modules, which simplified design and coding. I was able to see how the logic of login and transaction systems is similar by identifying patterns. My ability to think algorithmically and write Python code was enhanced by this exercise.