ATM BANKING SIMULATION

Project Title: ATM Banking Simulation
Author: Disha K
Course / Subject: Python Programming
Date: 13-1-2026

Abstract

The ATM Banking Simulation project is a console-based application developed using Python to simulate the basic operations of an Automated Teller Machine (ATM). The system allows a user to securely log in using a Personal Identification Number (PIN) and perform essential banking operations such as balance inquiry, depositing money, and withdrawing money. The project demonstrates the use of fundamental Python programming concepts including dictionaries, functions, loops, and conditional statements. This simulation helps beginners understand how real-world banking logic can be implemented using simple programming constructs.

Keywords

Keywords: Python, ATM Simulation, Dictionary, Functions, Loops, Conditional Statements

Introduction

An Automated Teller Machine (ATM) is an electronic banking outlet that enables customers to perform financial transactions without visiting a bank branch. ATMs provide services such as cash withdrawal, cash deposit, and balance inquiry at any time of the day. This project simulates a basic ATM system using Python. It focuses on implementing secure PIN validation and transaction handling using a menu-driven program. The project is designed to help beginners apply programming concepts to a real-life scenario.

Objectives

The main objectives of this project are:

- To simulate basic ATM operations using Python
- To implement secure PIN authentication
- To allow users to check balance, deposit money, and withdraw money
- To understand the use of dictionaries for storing user data
- To practice functions, loops, and conditional statements

System Requirements

Software Requirements

- Python 3.x
- Any Python IDE or text editor (VS Code, PyCharm, IDLE)

Hardware Requirements

- A computer or laptop with keyboard and display

System Design

The ATM system follows a menu-driven approach. After successful PIN authentication, the user is presented with a menu of options. Each option corresponds to a specific banking operation. The program continues to display the menu until the user chooses to exit.

User Data Structure

The user account details are stored in a dictionary:

```
user = {
    "Name": "Disha",
    "PIN": 54321,
    "Balance": 5000
}
```

```
user = {"Name": "Disha","PIN": 54321,"Balance": 5000}
```

Functional Modules

- **Check Balance:** Displays the current account balance
- **Deposit Money:** Adds a specified amount to the balance
- **Withdraw Money:** Deducts a specified amount if sufficient balance is available
- **Exit:** Terminates the program

Implementation

PIN Authentication Logic

```
PIN=int(input("Enter PIN: "))
if PIN!=user["PIN"]:
    print("Invalid PIN")
else:
    while True:
        print("\n1.Check Balance\t2.Deposit\t3.Withdraw\t4.Exit\n")
        choice=int(input("Choose: "))

        if choice==1:
            check_balance()
        elif choice==2:
            deposit()
        elif choice==3:
            withdraw()
        elif choice==4:
            break
        else:
            print("Invalid choice")
```

Fig 1: PIN Authentication Code

This logic compares the user-entered PIN with the stored PIN. Access to the ATM menu is granted only if the PIN matches; otherwise, an error message is displayed.

Deposit Function

```
def deposit():  1 usage
    amount=int(input("Enter amount to deposit:"))
    if amount > 0:
        user["Balance"]=amount+user["Balance"]
        print("Deposit successful")
        print("The amount deposited is",amount)
```

Fig 2: Deposit Function Code

The deposit function prompts the user to enter an amount. If the amount is positive, it is added to the existing balance and a confirmation message is displayed.

Withdraw Function

```
def withdraw():  1 usage
    amount = int(input("Enter amount to withdraw:"))
    if amount > user["Balance"]:
        print("Insufficient balance")
    elif amount > 0:
        user["Balance"]=user["Balance"]-amount
        print("The amount withdrawn is",amount)
```

Fig 3: Withdraw Function Code

The withdraw function checks whether the requested amount is less than or equal to the available balance. If sufficient funds exist, the amount is deducted; otherwise, an insufficient balance message is shown.

Balance Check Function

```
def check_balance():  1 usage
    print("Balance:",user["Balance"])
```

Fig 4: Balance Check Function Code

This function simply displays the current balance stored in the user dictionary.

Testing

Test Case 1: Valid PIN Login

- **Expected Result:** Access granted
- **Actual Result:** Access granted
- **Status:** Pass

```
Enter PIN: 54321

1.Check Balance 2.Deposit   3.Withdraw  4.Exit

Choose:
```

Test Case 2: Deposit Money

- **Expected Result:** Balance increases
- **Actual Result:** Balance updated successfully
- **Status:** Pass

```
1.Check Balance 2.Deposit   3.Withdraw  4.Exit

Choose: 2
Enter amount to deposit:1000
Deposit successful
The amount deposited is 1000
```

Test Case 3: Withdraw with Insufficient Balance

- **Expected Result:** Transaction denied
- **Actual Result:** Insufficient balance message displayed
- **Status:** Pass

```
1.Check Balance 2.Deposit   3.Withdraw  4.Exit

Choose: 3
Enter amount to withdraw:6000
Insufficient balance
```

Test Case 4: Exit Option

- **Expected Result:** Program terminates
- **Actual Result:** Program exits successfully
- **Status:** Pass

```
1.Check Balance 2.Deposit   3.Withdraw  4.Exit

Choose: 4

Process finished with exit code 0
```

Results

The ATM Banking Simulation performed all operations correctly. PIN validation worked as expected, deposits and withdrawals updated the balance accurately, and invalid transactions were prevented. The menu-driven system allowed smooth navigation between different operations.

Conclusion

This project successfully demonstrates a simple ATM banking system using Python. It highlights the practical use of dictionaries, functions, loops, and conditional statements. The simulation provides a clear understanding of how basic banking operations can be implemented programmatically and serves as a strong foundation for more advanced projects.

Future Scope

The project can be enhanced in the following ways:

- Support for multiple user accounts
- Integration with a database for data persistence
- Implementation of a graphical user interface (GUI)
- Secure PIN storage using encryption techniques
- Transaction history and mini statement generation

References

- Python Official Documentation https://docs.python.org/3/
- Python Banking Programs Menu-Driven Bank Management System in Python: https://www.geeksforgeeks.org/menu-driven-program-for-bank-management-system/ (GeeksforGeeks)
- Investopedia – Understanding ATM Systems:https://www.investopedia.com/terms/a/atm.asp (investopedia.com)