Project Report

Project Name: A simple bank account

System Design and Implementation

BankAccount Class Definition

I have created a first-class **BankAccount**, in which I have defined:

- **Account Holder**
- **Balance**

The first initialization (`__init__`) creates a new account with the name Satyam and a balance of 0. There are two variables here: `self.account_holder` and `self.balance`.

Figure 1: BankAccount class initialization

Figure 2: Class constructor with account holder and balance attributes

```
class BankAccount:
    def __init__(self, Satyam, balance=0.0):
        self.account_holder = Satyam
        self.balance = balance
```

Core Functions

Deposit Function

After this there is a **deposit function** that accepts a new deposit and adds it to `self.balance`. If the entered amount is negative, then it shows an error with a clear message.

```
def deposit(self, amount):
    if amount > 0:
        self.balance += amount
        print(f"Deposited {amount}. New balance is {self.balance}")
    else:
        print("Deposited amount must be positive.")
```

Figure 3: Deposit function implementation with validation

Withdraw Function

After this **withdraw function**, which adds balance to the account according to the amount entered by the user. If the entered amount is more than the balance, then it shows an error saying "Insufficient Balance."

```
def withdraw(self, amount):
    if amount <= 0:
        print("Withdraw must be positive.")
    elif amount > self.balance:
        print("Insufficient funds for this withdrawal!")
    else:
        self.balance -= amount
        print(f"Withdraw successful of amount {amount}. New balance is {self.balance}")
```

Figure 4: Withdraw function with balance validation

Display Balance Function

After this there is a **display balance function**, which shows the new balance after the transaction is done.

```
def display_balance(self):
    print(f"{self.account_holder}'s balance is {self.balance}")
```

User Interface Implementation

User Input for Account Creation

Now I have created a variable to get user input for account name.

```
account = BankAccount(input("Enter account holder name: "))
```

Figure 5: User input prompt and account creation

Transaction Loop and Menu System

Main Transaction Loop

Now comes the loop to do multiple transactions.

Here the code will first print 4 options for users to select; after users' selection, it will execute the corresponding function:

Menu Options

1. **Deposit** - The first condition checks for input "1" and deposits the entered amount by running the deposit function
2. **Withdraw** - The second condition checks for input "2" and withdraws the entered amount by running the withdraw function
3. **Check Balance** - The third condition checks for input "3," and if the input is correct, it displays the balance of the account by running the display balance function
4. **Exit** - The fourth condition checks for input "4" and breaks the loop if the input is correct by printing "Logging out..."
5. **Invalid Input** - The last one prints "Invalid Choice" if none of the given inputs is selected

Figure 6: Menu system implementation and Transaction loop with conditional logic

```python
while True:
    print("Choose a transaction:")
    print("1. Deposit")
    print("2. Withdraw")
    print("3. Show Balance")
    print("4. Log out")
    choice = input("Select action 1/2/3/4: ")

    if choice == "1":
        amount = float(input("Enter amount you want to deposit: "))
        account.deposit(amount)
    elif choice == "2":
        amount = float(input("Enter the amount you want to withdraw: "))
        account.withdraw(amount)
    elif choice == "3":
        account.display_balance()
    elif choice == "4":
        print("Logging out..")
        break
    else:
        print("Invalid choice, Please enter correct one ")
```

Conclusion

This simple bank account system demonstrates fundamental programming concepts including:

- **Object-Oriented Programming** through class-based design
- **Input validation** and error handling
- **User interface** with menu-driven interaction
- **Control flow** using loops and conditional statements
- **Function encapsulation** for modular code organization

The project successfully implements core banking operations with proper validation and user-friendly feedback mechanisms.

Report and project done by: Satyam Kumar
Student ID: NIU-25-7003
Team : Syntax Squad