PROJECT REPORT

SIMPLE BANK ACCOUNT SYSTEM USING PYTHON

TEAM NAME: CodeXworld

This project report has been prepared to demonstrate our understanding of core Python programming concepts, with a special focus on Object-Oriented Programming (OOP). Through the development of a "Simple Bank Account System," we aim to showcase how classes, objects, methods, and decision-making structures can be applied to create practical, real-world applications. This work reflects our analytical skills, technical learning, and collaborative effort in translating theoretical knowledge into a functional software model.

TEAM MEMBERS

- RAJ SINGH (GitHub Account: rajsingh0327)
- MONU(GitHub Account: iamharish621-sys)
- PRIYANSHU KUMAR (GitHub Account: Priyanshu9040)
- SALONI PANWAR (GitHub Account: salonipanwarr001)
- RITOBROTO RAY (GitHub Account: ritobrotoray-ui)

Project Report: Simple Bank Account System Using Python

1. Introduction

A bank account system is one of the most basic applications used to understand Object-Oriented Programming (OOP) concepts in Python.

This project demonstrates how to create a simple bank account using a Python class, which allows the user to deposit and withdraw money. The system also includes important features such as checking for sufficient balance and allowing multiple transactions through a loop.

The aim of the project is to learn how classes, objects, methods, and conditional statements work together in Python.

2. Objectives of the Project

The main objectives of this project are:

1. To understand how to create and use classes and objects in Python.
2. To implement methods for deposit and withdrawal operations.
3. To use conditional statements to check for sufficient balance.
4. To build a menu-driven program that allows multiple transactions.
5. To provide a simple and user-friendly system for basic banking operations.

3. Methodology

This project follows the Object-Oriented Programming approach. The BankAccount class is used to represent a bank account with its properties and behavior.

3.1 Class Design

Class Name: `BankAccount`

Attribute: `balance` – Stores the current balance of the user.

Methods:

1. `__init__()` – Constructor to initialize the account balance.
2. `deposit(amount)` – Adds money to the account balance.
3. `withdraw(amount)` – Withdraws money after checking for sufficient balance.

4. System Features

4.1 Deposit Money

• Allows the user to enter an amount to deposit.

• Updates the balance.

• Displays the updated balance.

4.2 Withdraw Money

• Accepts an amount from the user.

• Checks if the account has enough balance.

• Performs withdrawal only if balance is sufficient.

• Shows a warning message if funds are low.

4.3 Check Balance

• Displays the current account balance to the user.

4.4 Multiple Transactions

• A loop-driven menu allows the user to perform transactions repeatedly until they select “Exit”.

5. Python Code Implementation

```
class BankAccount:
    def __init__(self, balance=0):
        self.balance = balance

    def deposit(self, amount):
        self.balance += amount
        print(f"₹{amount} deposited successfully.")
        print(f"Current Balance: ₹{self.balance}\n")

    def withdraw(self, amount):
        if amount > self.balance:
            print("Insufficient Balance! Withdrawal Failed.\n")
        else:
            self.balance -= amount
            print(f"₹{amount} withdrawn successfully.")
            print(f"Current Balance: ₹{self.balance}\n")


# Main Program
account = BankAccount()
```

```
while True:
    print("\n--- Simple Bank Menu ---")
    print("1. Deposit Money")
    print("2. Withdraw Money")
    print("3. Check Balance")
    print("4. Exit")

    choice = int(input("Enter your choice: "))

    if choice == 1:
        amt = float(input("Enter amount to deposit: "))
        account.deposit(amt)

    elif choice == 2:
        amt = float(input("Enter amount to withdraw: "))
        account.withdraw(amt)

    elif choice == 3:
        print(f"Your Current Balance is: ₹{account.balance}\n")

    elif choice == 4:
        print("Thank you for using the bank system!")
        break

    else:
        print("Invalid Choice! Please try again.\n")
```

6. Output Snapshot (Explanation)

When the program runs, the user sees a menu:

--- Simple Bank Menu ---

1. Deposit Money

2. Withdraw Money

3. Check Balance

4. Exit

• If the user selects Deposit, they enter the amount and the balance increases.

• If they select Withdraw, the system checks if enough balance exists.

• The balance can be checked anytime.

• Selecting Exit stops the program.

7. Advantages of the System

• Simple and easy to understand.

• Demonstrates important OOP concepts.

• User-friendly menu-driven program.

• Ensures safe withdrawal using balance checking.

9. Conclusion

This project successfully demonstrates how a simple banking system can be created using Python's Object-Oriented Programming features. It includes all essential operations like deposit, withdrawal, and balance check. The system is efficient, easy to use, and helps in understanding class design, loops, and conditional statements.