Simple Bank Account System - Project Report

Executive Summary

This project demonstrates the implementation of a simple yet functional bank account management system using Object-Oriented Programming (OOP) principles in Python. The system allows users to perform fundamental banking operations including depositing funds, withdrawing money, and checking account balance through an interactive menu-driven interface.

Project Overview

Project Name: Simple Bank Account System
Objective: Create a class-based banking application with deposit, withdrawal, and balance inquiry functionalities

The project utilizes Python's class structure to encapsulate banking operations, implementing proper error handling for invalid transactions and a user-friendly command-line interface for account management.

System Architecture

Class Design: BankAccount

The foundation of this project is the **BankAccount class**, which encapsulates all account-related operations and data. This class-based approach promotes code reusability, maintainability, and follows OOP best practices.

Core Attributes

The BankAccount class contains two primary instance variables:

- **account_holder**: Stores the name of the account owner (initialized as "Satyam")
- **balance**: Maintains the current account balance (initialized to 0)

These attributes are initialized through the constructor method `__init__()`, which creates a new account instance with specified parameters and establishes the initial state of the account.

Functional Components

1. Account Initialization (`__init__` Method)

The initialization method serves as the constructor for creating new BankAccount instances. It accepts the account holder's name and starting balance as parameters, setting up the foundational attributes needed for all subsequent operations.

```
def __init__(self, account_holder, initial_balance=0):
    self.account_holder = account_holder
    self.balance = initial_balance
```

This method ensures that every account is created with a valid name and a defined starting balance.

2. Deposit Function

The **deposit function** enables users to add funds to their account. This method accepts a deposit amount, validates that the amount is positive, and updates the account balance accordingly.

Key Features:

- Accepts monetary amount as input
- Validates that deposits are non-negative values
- Displays error message if user attempts negative deposit
- Updates balance upon successful transaction
- Provides transaction confirmation

The validation mechanism prevents logical errors such as negative deposits, ensuring data integrity throughout the transaction history.

3. Withdrawal Function

The **withdraw function** allows users to remove funds from their account with comprehensive validation checks. This method ensures sufficient balance exists before processing any withdrawal request.

Key Features:

- Accepts withdrawal amount as input
- Verifies that account balance is sufficient
- Displays error message if balance is insufficient
- Updates balance upon successful transaction
- Prevents overdrafts through balance validation

This function serves as a critical safeguard against overdraft scenarios, maintaining account integrity by ensuring withdrawals do not result in negative balances.

4. Display Balance Function

The **display_balance function** provides users with real-time information about their current account balance. This function is called after every transaction to confirm the transaction status and show the updated balance.

Key Features:

- Retrieves current account balance
- Displays balance in readable format
- Called after each transaction for confirmation
- Provides transparency in account management

User Interface Implementation

Account Setup

The system begins by prompting the user to input their account holder name. This input is captured and used to instantiate a new BankAccount object with the provided name and an initial balance of zero.

Interactive Menu System

The program implements a menu-driven loop that allows users to perform multiple transactions within a single session. The main menu presents four options to the user:

Menu Options:

Option	Function	Description
1	Deposit	Add funds to the account
2	Withdraw	Remove funds from the account
3	Check Balance	Display current account balance
4	Exit	Logout and terminate the session

Transaction Flow

The transaction loop operates as follows:

1. **Display Menu**: The program presents all available options to the user
2. **Accept Input**: User selects an option by entering 1, 2, 3, or 4
3. **Process Request**: A switch-case conditional structure routes the input to the appropriate function
4. **Execute Function**: The selected operation (deposit, withdraw, or balance check) is executed
5. **Confirm Transaction**: Balance is displayed after each operation

6. **Loop Control**: The program either returns to the menu or exits based on user selection

Error Handling

The system includes robust error handling mechanisms:

- **Deposit Validation**: Rejects negative deposit amounts with an appropriate error message
- **Withdrawal Validation**: Prevents withdrawals exceeding available balance
- **Input Validation**: Displays "Invalid Choice" message for unrecognized menu selections
- **Clear Error Messages**: Each error includes a descriptive message explaining the issue

Program Logic Summary

The program follows this sequential workflow:

1. Create a new BankAccount instance with user-provided name
2. Enter the main transaction loop
3. Display menu options for user selection
4. Evaluate user input using conditional logic:
    - Input "1": Execute deposit function with user-provided amount
    - Input "2": Execute withdraw function with user-provided amount
    - Input "3": Execute display_balance function
    - Input "4": Break loop and display "Logging out..." message
    - Any other input: Display "Invalid Choice" and return to menu
5. Repeat steps 3-4 until user selects option 4 (Exit)

Key Achievements

✓ Successfully implemented OOP principles through class-based design
✓ Comprehensive input validation for all financial transactions
✓ User-friendly menu-driven interface for intuitive navigation
✓ Proper error handling with descriptive feedback messages
✓ Clean, modular code structure promoting maintainability
✓ Real-time balance updates and transaction confirmation

Conclusion

This simple bank account system demonstrates fundamental programming concepts including object-oriented design, function definition, user input handling, conditional logic, and error validation. The project serves as an excellent foundation for understanding how financial applications manage core banking operations while maintaining data integrity and providing a user-centric experience.

The modular design allows for future enhancements such as transaction history logging, multi-account management, interest calculation, and persistent data storage.

Project completed by: Satyam Kumar
ID: NIU-25-7003
[1]

⁂

1. Project-Report.docx