PHYTON PROJECT

BANKING MANAGMENT SYSTEM

Prepared by :

HAPPY

(25MEI10080)

Course: [CSE1021 - Introduction to Problem Solving and Programming]
Institution: [VIT BHOPAL]
Submitted By: [HAPPY]
Roll Number: [25MEI10080]

GitHub Repository:
https://github.com/happy25mei10080-glitch/vityarthiproject/edit/main/README.md

1. Introduction

i made a project on banking system which operates some bank operation like withdraw, deposite and check balance while withdrawn .This project includes user friendly interfaces and user input handling, conditional logic, loops, and basic financial transaction processing. the system helps the bank to manage all the things easily and in very less time.

2. Problem Statement

There is a need for a simple, accessible banking system that allows users to perform basic transactions without unnecessary complex thing so that they will not confuses any more . This project addresses the need for:

1 Quick access to banking operations
2. Simple user interface for all age groups
3. Basic transaction processing capabilities
4. Educational tool for understanding banking operations

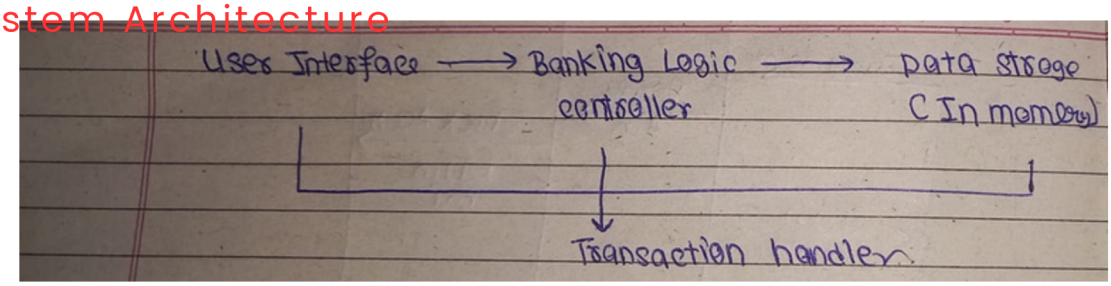
3. Functional Requirements

- **User Authentication**: verifies the account of user and verifies name
- **Deposit Functionality**: allows to deposite users money easily
- **Withdrawal Functionality**: allows to withdraw user money with balance checks
- **Balance Inquiry**: Provide account statement in every step except exit
- **Transaction Validation**: Ensure all transactions are valid and within limits
- **User-friendly Interface**: interface is kept very user friendly so there is no thing where user face difficulty
-

4. Non-functional Requirements

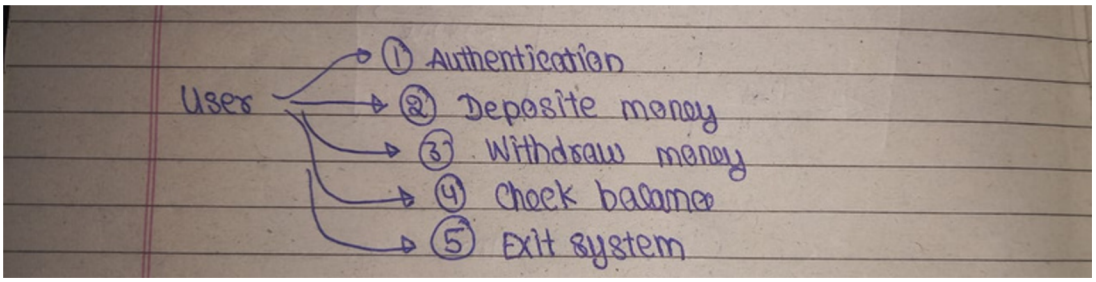
Performance:
Reliability:
Usability:
Security:
Maintainability:
Portability:

5. System Architecture

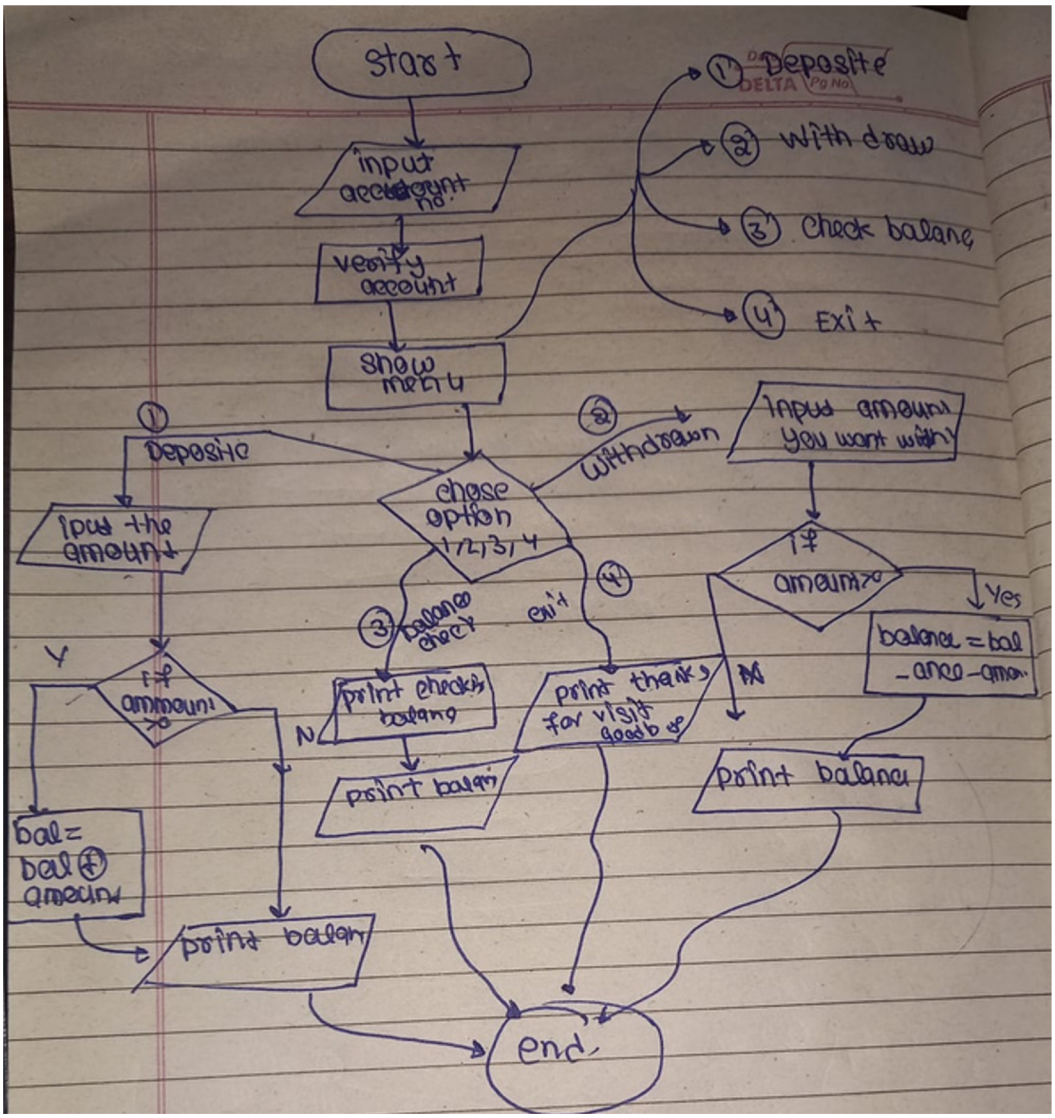


6. Design Diagrams

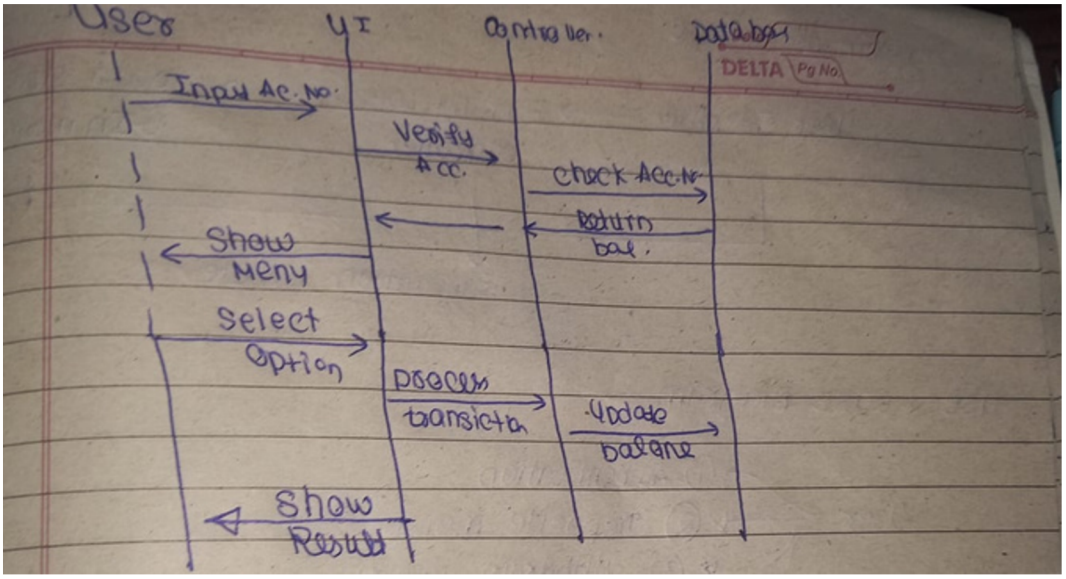
Use Case Diagram



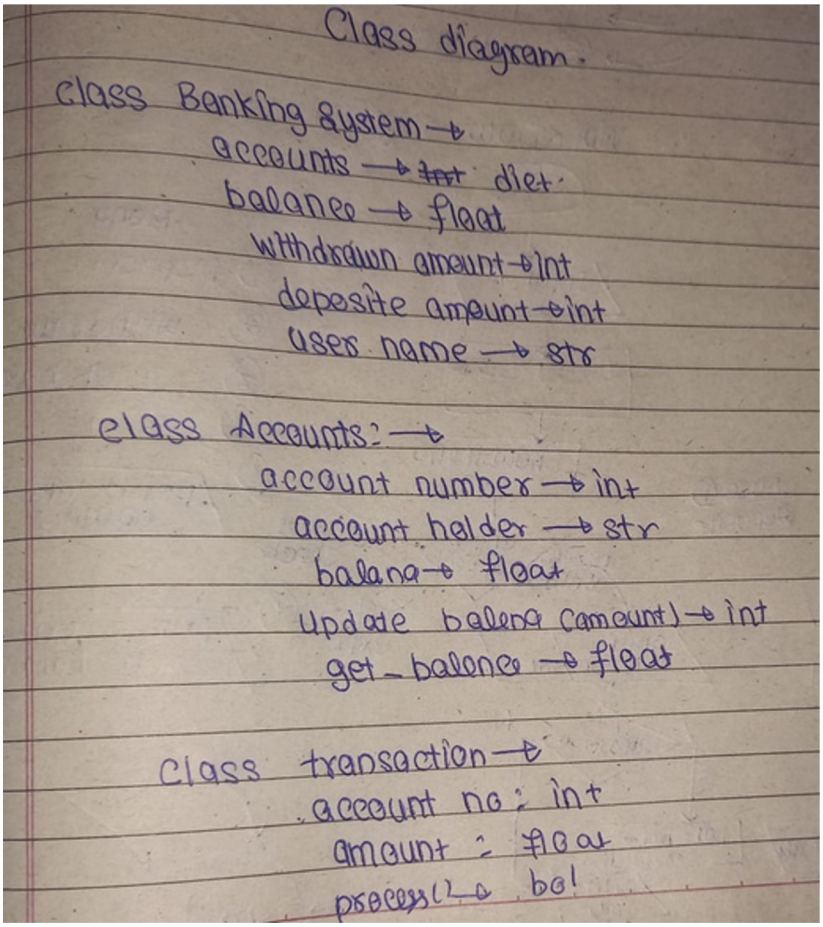
Workflow Diagram



Sequence Diagram



Class Diagram



7. Design Decisions & Rationale

Technology Choices

- **Python**: Selected for its simplicity
- **Console Interface**: Chosen for simplicity and focus on core logic
- **In-Memory Storage**: Used for prototype demonstration; easily replaceable with database

Architectural Decisions

- **Input Validation**: user can input there money in withdraw and deposite casses
- **Error Handling**: Comprehensive exception handling for user-friendly experience

9. Implementation Details

```python
def main() -> None:
    name=input('enter account holders name')
    acc_number=int(input('enter your account number'))
    if acc_number==12345:
        balance=23410
    elif acc_number==23456:
        balance=89023
    elif acc_number==469978:
        balance=33200
    elif acc_number==74978:
        balance=11003

    print(name,'welcome to our bank')
    #balance = 6666760.0  # Initialize balance

    while True:
        print("\n---BANKING SYSTEM---")
        print("1. Deposite")
        print("2. withdraw")
        print("3. Check balance")
        print("4.Exit")

        choice =input("choose option")

        if choice =='1':
            amount=float(input("ENTER amount"))
            if amount>0:
                balance=balance+amount
                print("amount deposited sucessfully",amount)
                print("amount balance ",balance)
        elif choice == '2':
            amount=float(input("ENTER amount want withdraw"))
            if amount<= balance:
                balance = balance-amount
                print('your amount',amount,' withdran sussefully\n ')
                print("amount balance ",balance)
            else:
                print("insufficient balance")
        elif choice =='3':
            print('Checking your balance')
            print("YOUR BALNCE IS :",balance)
        elif choice=='4':
            print("exiting your bank project. goodbye")
            break
        else:
            print("not getting any command, try again")
main()
```

Key Features Implemented

- Account verification system
- Transaction validation
- Balance management
- User session handling
- Input sanitization

10. Screenshots / Results

```
PS C:\code,s> python -u "c:\code,s\.vscode\b management system.py"
 enter account holders nameHappy
 enter your account number74978
 Happy welcome to our bank

 ---BANKING SYSTEM---
 1. Deposite
 2. withdraw
 3. Check balance
 4.Exit
 choose option1
 ENTER amount2345
 amount deposited sucessfully 2345.0
 amount balance  13348.0
```

```
---BANKING SYSTEM---
1. Deposite
2. withdraw
3. Check balance
4.Exit
choose option2
ENTER amount want withdraw3267
your amount 3267.0  withdran sussefully

amount balance  10081.0
```

```
---BANKING SYSTEM---
1. Deposite
2. withdraw
3. Check balance
4.Exit
choose optionpython -u "c:\code,s\.vscode\b management system.py"
not getting any command, try again

---BANKING SYSTEM---
1. Deposite
2. withdraw
3. Check balance
4.Exit
choose option2
ENTER amount want withdraw3267
your amount 3267.0  withdran sussefully

amount balance  10081.0

---BANKING SYSTEM---
1. Deposite
2. withdraw
3. Check balance
4.Exit
choose option3
Checking your balance
YOUR BALNCE IS : 10081.0
```

```
---BANKING SYSTEM---
1. Deposite
2. withdraw
3. Check balance
4.Exit
choose option3
Checking your balance
YOUR BALNCE IS : 10081.0

---BANKING SYSTEM---
1. Deposite
2. withdraw
3. Check balance
4.Exit
choose option4
exiting your bank project. goodbye
```

11. Testing Approach

Test Cases Executed

Account Verification

- Valid account number
- Invalid account number

Deposit Functionality

- Positive amounts
- Zero amount
- Negative amount (error case)
- Non-numeric input (error case)

Withdrawal Functionality

- Amount within balance
- Amount exceeding balance
- Zero amount
- Negative amount

Balance Inquiry

- Initial balance display
- Balance after transactions

12. Challenges Faced

Technical Challenges

Input Validation: Handling edge cases for user input
State Management: Maintaining account statements across transactions
Code Organization: Structuring code for readability and maintainability and syntax error at some cases

Solutions Implemented

- Implemented input sanitization
- Modularized transaction processing logic

13. Learnings & Key Takeaways

Enhanced Python programming skills
Better understanding of control structures
recognised syntax error in python
Code organization and documentation practices
Problem-solving approach
Project planning and execution
Documentation skills

14. Future Enhancements

- Add PIN-based authentication
- Implement transaction history
- Add fund transfer between accounts
- Advanced security features (encryption, OTP)

15. References

[VITYARHI]
[Banking System Design Patterns]