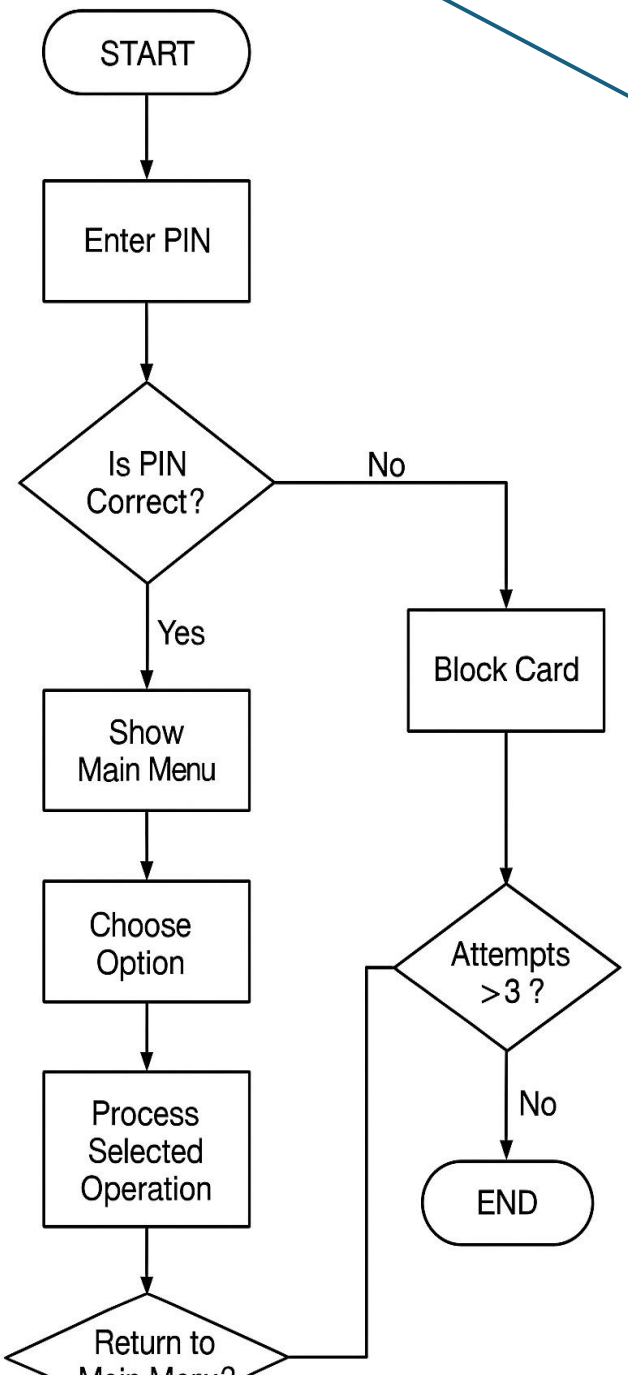
2. FLOWCHART OF ATM SIMULATION

Why 2 because in no. 1 I have created a folder named atm_project file and created the text files of python and readme.md etc.



[Grab your reader's attention with a great quote from the document or use this space to emphasize a key point. To place this text box anywhere on the page, just drag it.]

The ATM system begins with the user entering a PIN. The system checks if the PIN is correct.

If the user enters the wrong PIN more than three times, the session ends. If the PIN is correct,

the user is taken to the main menu where they can select different operations such as checking

balance, withdrawing money, depositing money, or viewing transaction history. After completing

each operation, the user is returned to the main menu until they choose to exit.

---

3. FUNCTIONAL AND NON-FUNCTIONAL REQUIREMENTS

3.1 Functional Requirements

1. PIN Authentication:
   The system must allow the user to enter a PIN and validate it before accessing services.
2. Balance Inquiry:
   The user should be able to view their current account balance.

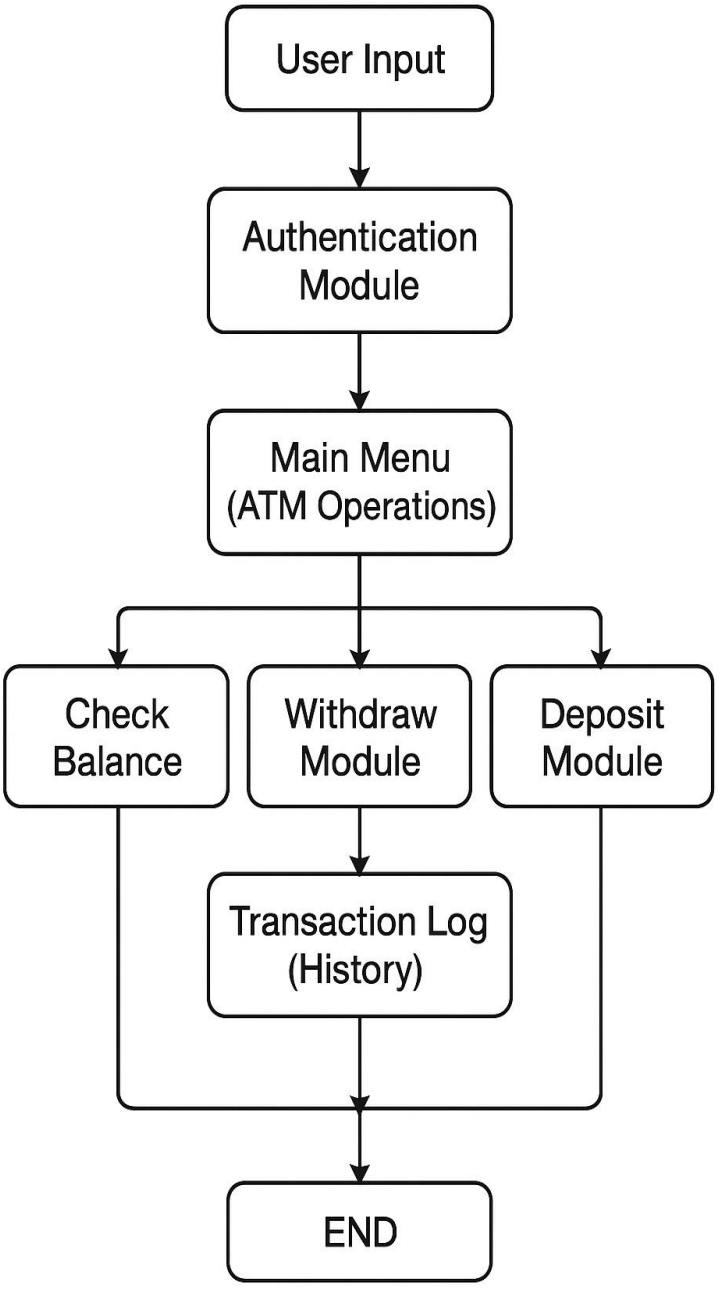
3. **Withdraw Money:**
   The system must allow the user to withdraw a valid amount, ensuring the balance is sufficient.
4. **Deposit Money:**
   The user should be able to deposit an amount into the account.
5. **Transaction History:**
   The system should record and display all previous transactions.
6. **Exit Option:**
   The user should be able to exit the ATM session anytime.

3.2 Non-Functional Requirements

1. **Usability:**
   The ATM system should be easy to use with a clear, simple menu interface.
2. **Performance:**
   All operations should be processed quickly without delays.
3. **Security:**
   The PIN must be validated carefully, and access must be blocked after three incorrect attempts.
4. **Reliability:**
   The system should handle invalid inputs and prevent crashes.

5. **Maintainability:**
   The system should be written using modular code with functions for easy updates.

4. SYSTEM ARCHITECTURE AND WORKFLOW



The user begins by entering the PIN. The system validates the PIN. If the PIN is incorrect

more than three times, the session ends. If the PIN is correct, the user is shown the main

ATM menu. The user can choose from options such as checking balance, withdrawing money,

depositing money, or viewing transaction history. Each operation is processed by its

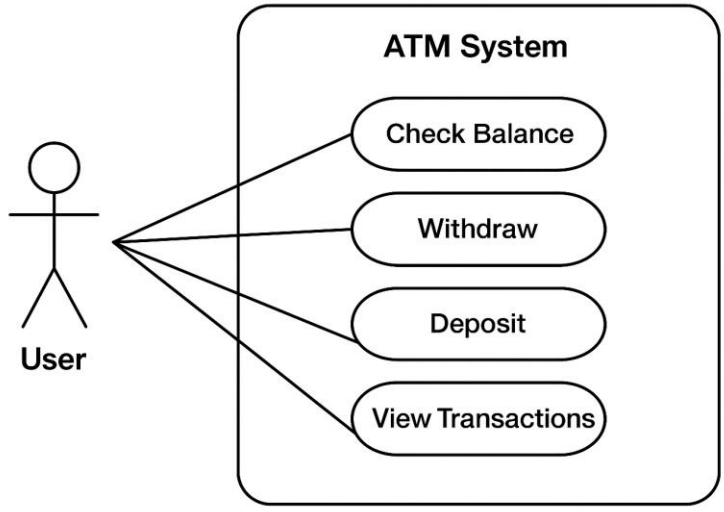
respective module. After each operation, the system returns to the main menu until the

user selects the exit option.

5. UML DIAGRAMS

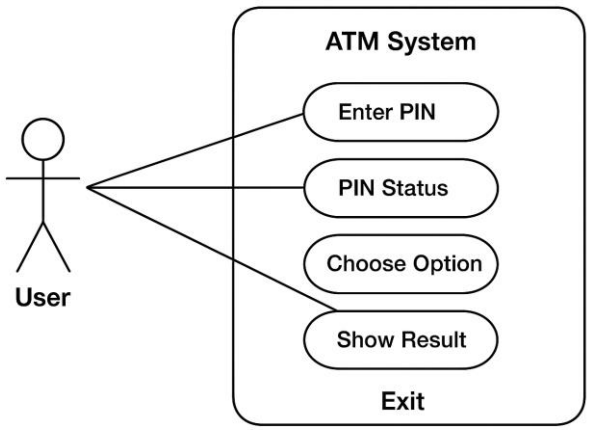
5.1 Use Case Diagram

The user interacts with the ATM System and performs actions such as checking balance, withdrawing money, depositing money, viewing transactions, and exiting the system.



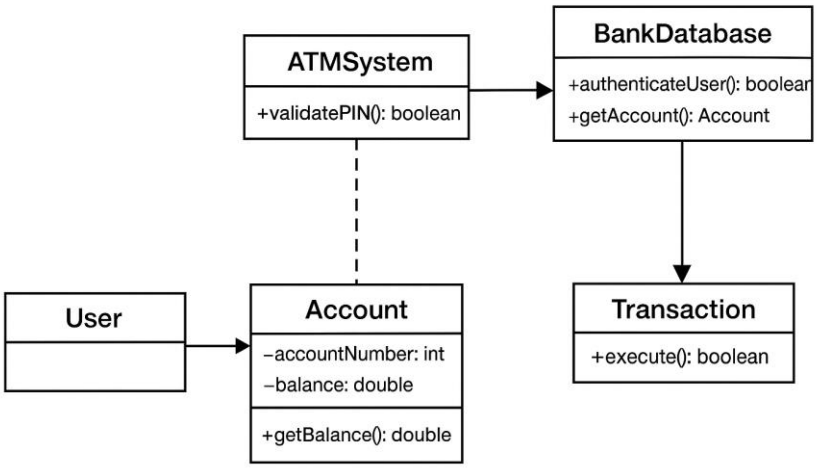
5.2 Sequence Diagram

This diagram shows how the user and the ATM system interact step-by-step during a session.



5.3 Class/Component Diagram

The ATM System class contains account data and provides methods for all ATM functions.



6. IMPLEMENTATION (PYTHON CODE)

--The following Python code implements the ATM Simulation System.

```
# ATM Simulation Program

balance = 10000
pin = "1234"
transactions = []

def authenticate():
    attempts = 0
    while attempts < 3:
        user_pin = input("Enter your PIN: ")

        if user_pin == pin:
            print("PIN verified successfully.\n")
            return True

        else:
            attempts += 1
            print("Incorrect PIN. Attempts left:", 3 - attempts)

    print("Too many incorrect attempts. Card blocked.")
    return False


def check_balance():
    print("\nYour current balance is: ₹", balance)
    transactions.append("Checked Balance")


def withdraw():
    global balance
    amount = int(input("\nEnter amount to withdraw: "))

    if amount <= 0:
        print("Invalid amount!")
        return

    if amount > balance:
        print("Insufficient funds!")
    else:
        balance -= amount
        print("Withdrawal successful! New balance: ₹", balance)
        transactions.append(f"Withdrew ₹{amount}")


def deposit():
    global balance
    amount = int(input("\nEnter amount to deposit: "))

    if amount <= 0:
```

```
        print("Invalid amount!")
    else:
        balance += amount
        print("Deposit successful! New balance: ₹", balance)
        transactions.append(f"Deposited ₹{amount}")


def show_transactions():
    print("\nTransaction History:")
    if len(transactions) == 0:
        print("No transactions yet.")
    else:
        for t in transactions:
            print("-", t)


def main_menu():
    while True:
        print("\n===== ATM MENU =====")
        print("1. Check Balance")
        print("2. Withdraw")
        print("3. Deposit")
        print("4. Transaction History")
        print("5. Exit")

        choice = input("Choose an option (1-5): ")

        if choice == "1":
            check_balance()
        elif choice == "2":
            withdraw()
        elif choice == "3":
            deposit()
        elif choice == "4":
            show_transactions()
        elif choice == "5":
            print("Thank you for using the ATM. Goodbye!")
            break
        else:
            print("Invalid choice. Please select again.")


# MAIN PROGRAM FLOW
print("----- Welcome to the ATM Simulation -----")

if authenticate():
    main_menu()
```

Code Explanation

1. authenticate()
   Verifies the user PIN with a maximum of three attempts.
2. check_balance()
   Displays the current account balance.
3. withdraw()
   Allows the user to withdraw money if funds are sufficient.

4. **deposit()**
   Allows the user to deposit money to the account.
5. **show_transactions()**
   Displays a list of all previous transactions.
6. **main_menu()**
   Displays the ATM menu and routes user actions.
7. **Main Program**
   The program starts with authentication and then opens the main menu.

7. TESTING AND RESULTS

7.1 TEST CASES

Test Case	Input	Expected Output	Actual Output	Result
PIN Check	Correct PIN (1234)	Access Granted	Works correctly	PASS
Wrong PIN	Wrong PIN 3 times	Access Blocked	Works correctly	PASS
Balance Inquiry	Option 1	Show Current Balance	Works correctly	PASS
Withdraw	Option 2, amount = 100	Deduct ₹100	Works correctly	PASS
Deposit	Option 3, amount = 200	Add ₹200	Works correctly	PASS
Transaction History	Option 4	Show List	Works correctly	PASS
Exit	Option 5	Program Ends	Works correctly	PASS

7.2 Screenshots:-

[*]:
```python
# ATM Simulation Program

balance = 10000
pin = "1234"
transactions = []

def authenticate():
    attempts = 0
    while attempts < 3:
        user_pin = input("Enter your PIN: ")

        if user_pin == pin:
            print("PIN verified successfully.\n")
            return True

        else:
            attempts += 1
            print("Incorrect PIN. Attempts left:", 3 - attempts)

    print("Too many incorrect attempts. Card blocked.")
    return False


def check_balance():
    print("\nYour current balance is: ₹", balance)
    transactions.append("Checked Balance")


def withdraw():
    global balance
    amount = int(input("\nEnter amount to withdraw: "))

    if amount <= 0:
        print("Invalid amount!")
        return

    if amount > balance:
        print("Insufficient funds!")
    else:
        balance -= amount
        print("Withdrawal successful! New balance: ₹", balance)
        transactions.append(f"Withdrew ₹{amount}")


def deposit():
    global balance
    amount = int(input("\nEnter amount to deposit: "))

    if amount <= 0:
        print("Invalid amount!")
    else:
        balance += amount
        print("Deposit successful! New balance: ₹", balance)
        transactions.append(f"Deposited ₹{amount}")


def show_transactions():
    print("\nTransaction History:")
    if len(transactions) == 0:
        print("No transactions yet.")
    else:
        for t in transactions:
            print("-", t)


def main_menu():
    while True:
        print("\n===== ATM MENU =====")
        print("1. Check Balance")
        print("2. Withdraw")
        print("3. Deposit")
        print("4. Transaction History")
        print("5. Exit")

        choice = input("Choose an option (1-5): ")

        if choice == "1":
            check_balance()
        elif choice == "2":
            withdraw()
        elif choice == "3":
            deposit()
        elif choice == "4":
            show_transactions()
        elif choice == "5":
            print("Thank you for using the ATM. Goodbye!")
            break
        else:
            print("Invalid choice. Please select again.")


# MAIN PROGRAM FLOW
print("----- Welcome to the ATM Simulation -----")

if authenticate():
    main_menu()
```

```
----- Welcome to the ATM Simulation -----
Enter your PIN:
```

```
----- Welcome to the ATM Simulation -----
Enter your PIN:  1234
PIN verified successfully.


===== ATM MENU =====
1. Check Balance
2. Withdraw
3. Deposit
4. Transaction History
5. Exit
Choose an option (1-5): ↑↓ for history. Search history with c-↑/c-↓


 ===== ATM MENU =====
 1. Check Balance
 2. Withdraw
 3. Deposit
 4. Transaction History
 5. Exit
 Choose an option (1-5):  1

 Your current balance is: ₹ 10000

 ===== ATM MENU =====
 1. Check Balance
 2. Withdraw
 3. Deposit
 4. Transaction History
 5. Exit
Choose an option (1-5):
===== ATM MENU =====
1. Check Balance
2. Withdraw
3. Deposit
4. Transaction History
5. Exit
Choose an option (1-5):  2

Enter amount to withdraw:  100
Withdrawal successful! New balance: ₹ 9900

===== ATM MENU =====
1. Check Balance
2. Withdraw
3. Deposit
4. Transaction History
5. Exit
Choose an option (1-5):  3

Enter amount to deposit:  50
Deposit successful! New balance: ₹ 9950

===== ATM MENU =====
1. Check Balance
2. Withdraw
3. Deposit
4. Transaction History
5. Exit
Choose an option (1-5): ↑↓ for history. Search history with c-↑/c-↓
```

```
===== ATM MENU =====
1. Check Balance
2. Withdraw
3. Deposit
4. Transaction History
5. Exit
Choose an option (1-5):  4

Transaction History:
- Checked Balance
- Withdrew ₹100
- Deposited ₹50

===== ATM MENU =====
1. Check Balance
2. Withdraw
3. Deposit
4. Transaction History
5. Exit
Choose an option (1-5): ↑↓ for history. Search history with c-↑/c-↓
```

The above screenshots demonstrate the working of the ATM Simulation System.

They show successful program execution including authentication, balance inquiry,

withdrawal, deposit, and transaction history operations. All functionalities

were tested and verified to be working correctly.

7.3 Challenges Faced

1. Understanding the flow of ATM operations and breaking them down into separate modules like authentication, withdrawal, and deposit was initially challenging.

2. Creating the flowchart and UML diagrams required careful planning to ensure the project structure was clear and easy to explain.

3. Running the Python script from Notepad and CMD was difficult because of issues with

file paths, file extensions, and Python environment setup.

4. Jupyter Notebook sometimes caused input handling problems such as kernel freeze or deadlock when using the input() function.

5. Ensuring the menu loop worked correctly without crashing required proper use of loops, conditions, and functions.

6. Formatting the report, adding screenshots, diagrams, and test cases took additional time and effort to maintain a clean structure.

7. Debugging withdrawal and deposit logic was necessary to handle invalid inputs, insufficient balance, and negative values.

7.4 Conclusion

The ATM Simulation project helped me understand how real-world banking systems work internally. Through this project, I learned about breaking a large problem into smaller modules such as authentication, balance inquiry, withdrawal, deposit, and transaction history. I also learned the importance of flowcharts, UML diagrams, and test cases in structuring and documenting a software project.

Implementing the project in Python improved my skills in loops, conditions, functions, user input handling, and error checking. I also gained practical experience in debugging, testing, and ensuring smooth program execution. Overall, the project strengthened my problem-solving skills and gave me a clear understanding of basic software development practices.

7.5 References

1. Python Official Documentation
   https://docs.python.org/

2. UML Diagram Basics
   https://www.uml-diagrams.org/

3. ATM System Concepts and Examples
   Various online educational resources

4. Jupyter Notebook and Python Input Handling
   https://jupyter.org/

5. Classroom Notes and Syllabus Provided by Faculty