INTRODUCTION TO PROBLEM SOLVING AND PROGRAMMING PYTHON PROJECT

(BANK MANAGEMENT SYSTEM)

BY:-SAMANT KUMAR(25BCE11230)

INTRODUCTION:-

- A Bank Management System (BMS) serves as the central technological backbone of the bank, moving beyond manual paper-based processes to provide a secure, efficient, and user-friendly platform for both bank personnel and customers.

PROBLEM STATEMENT:-

- **PROBLEM:-**

The existing manual or fragmented bank management processes are characterized by critical inefficiencies, including data inaccuracy due to human error, slow customer service wait times, poor data security and an inability to generate timely, comprehensive reports.

- **SOLUTION:-**

Implement a centralized, computerized Bank Management System (BMS) featuring a secure database, automated transaction processing, and a multi-tiered access control structure.

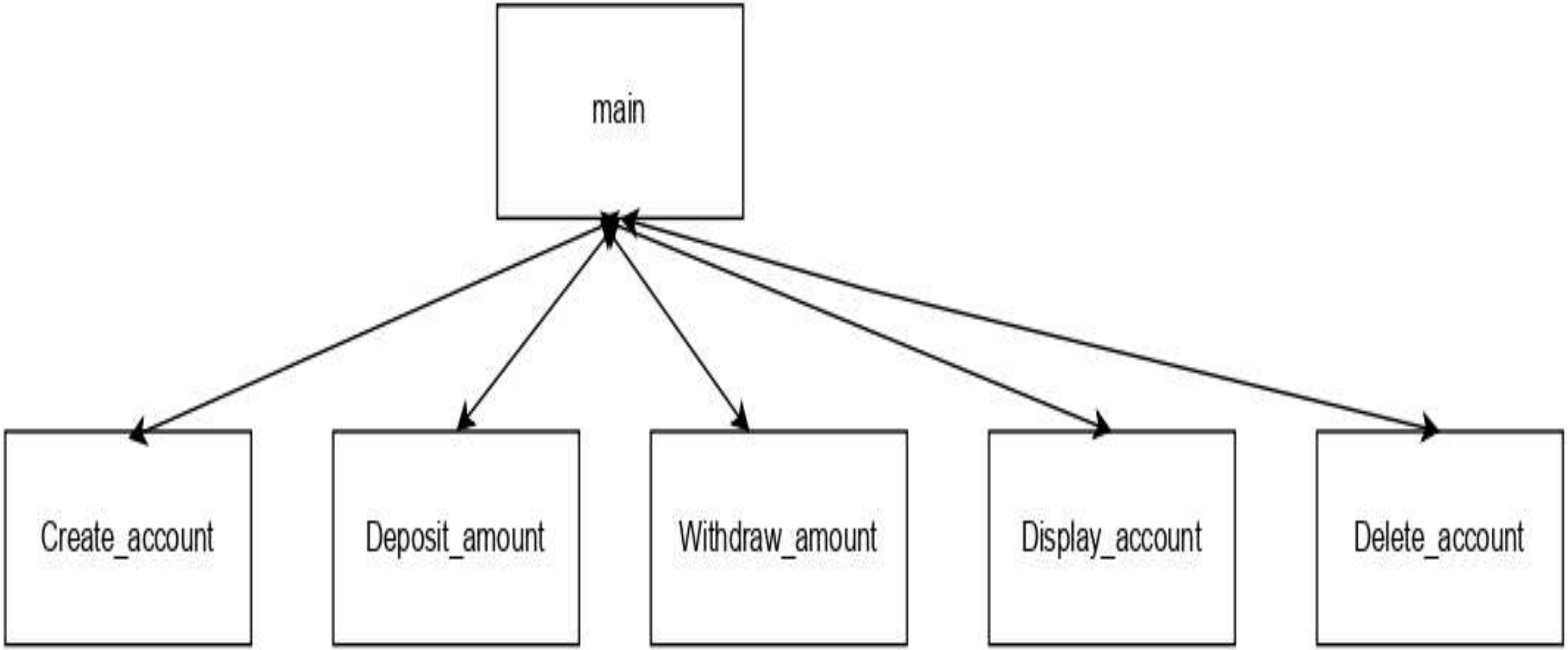
FUNCTIONAL REQUIREMENTS:-

- CUSTOMER MANAGEMENT MODULE:-The system must allow a bank employee to create a new customer profile and capture personal information.
- TRANSACTION MANAGEMENT MODULE:-The system must process cash deposit and withdrawal transactions.
- ACCOUNT MANAGEMENT MODULE:-The system must allow bank to automatically generate a unique account number and to close an existing bank account.

NON-FUNCTIONAL REQUIREMENTS:-

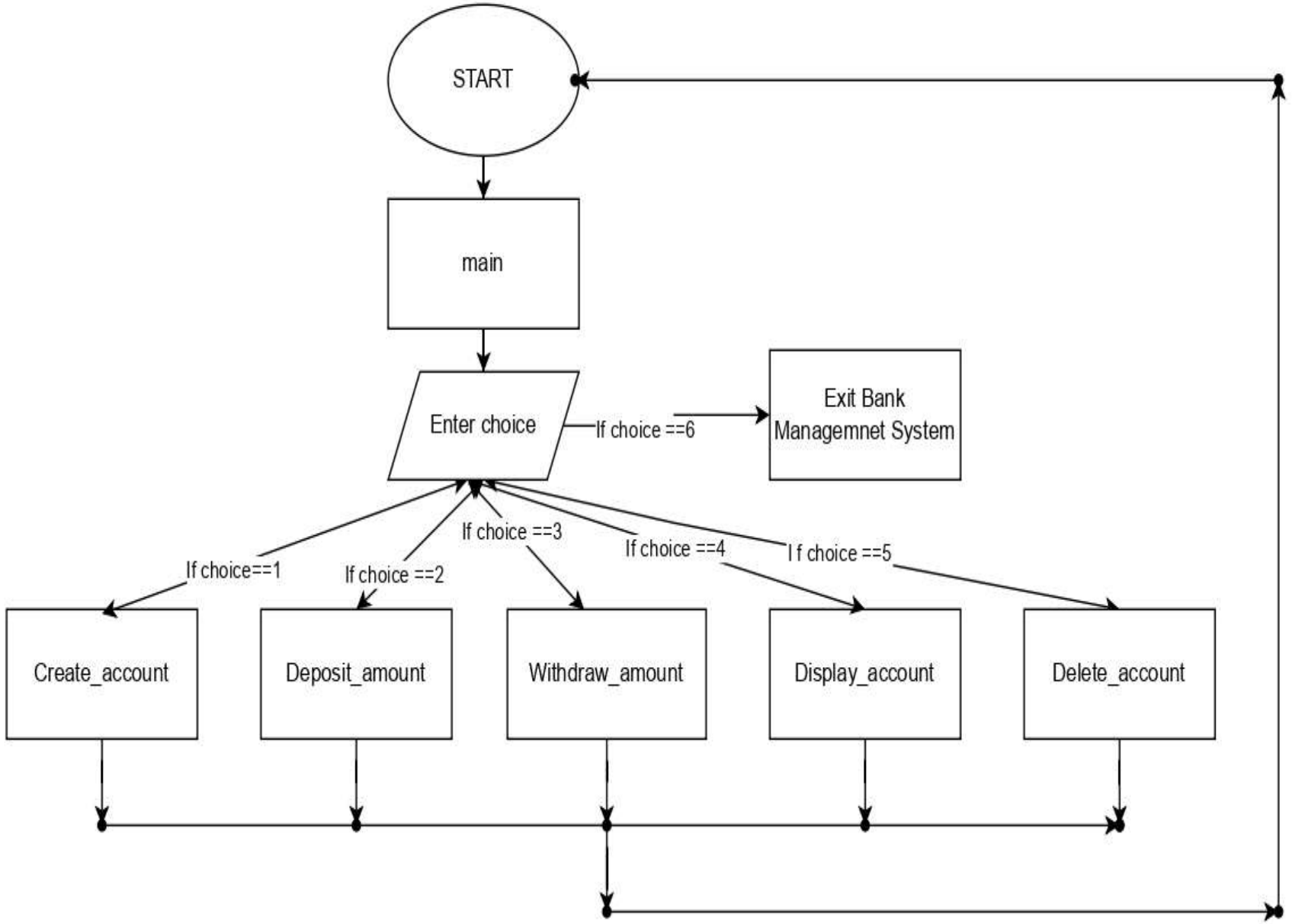
1. **USABILITY:-** The interface must be a simple Command Line Interface (CLI) navigable by non-technical users.
2. **RELIABILITY:-** The system must handle invalid inputs gracefully without crashing (e.g., inputs other than specific options).
3. **PERFORMANCE:-** The system must respond to user inputs instantly (< 0.1 seconds).
4. **OFFLINE AVAILABILITY:-** The system must function 100% without an internet connection.

SYSTEM ARCHITECTURE:-



DESIGN DIAGRAMS

(WORKFLOW DIAGRAM):-



DESIGN DECISIONS & RATIONALE :-

- USE OF PYTHON DICTIONARIES:-We chose nested dictionaries to represent the symptom tree.

- RATIONALE:-This allows for O(1) lookup time and makes it incredibly easy to add new symptoms without modifying the core code logic.

- COMMAND LINE INTERFACE (CLI):-We chose a text-based interface.

- RATIONALE:-It ensures the application is lightweight and can run on older hardware often found in resource-constrained environments.

IMPLEMENTATION DETAILS:-

- The project is implemented in Python 3.10 using a modular folder structure:

1. main:-The entry point containing the main `while' loop.
2. Create_account:-It helps in creating an account.
3. Deposit_amount:-It helps in depositing amount to an account.
4. Withdraw_amount:-It helps in withdrawing amount from an account.
5. Display_account:-It helps in displaying information of an account.
6. Delete_account:-It helps to delete an existing account.

OUTPUTS:-

CREATING AN ACCOUNT:-

```
1:Create
2:Deposit
3:Withdraw
4:Display
5:Delete
6:Exit
Enter your choice:1
Enter your name:samant kumar
Enter your age:18
Enter initial deposit amount:1000
Account created successfully! Your account number is: 1
```

DEPOSITING AMOUNT:-

```
1:Create
2:Deposit
3:Withdraw
4:Display
5:Delete
6:Exit
Enter your choice:2
Enter your account number:1
Enter amount to deposit:100
Amount deposit successful
```

OUTPUTS:-

WITHDRAWING AMOUNT:-

```
1:Create
2:Deposit
3:Withdraw
4:Display
5:Delete
6:Exit
Enter your choice:3
Enter your account number:1
Enter amount to withdraw:500
Amount withdrawl successful
```

DISPLAYING ACCOUNT:-

```
1:Create
2:Deposit
3:Withdraw
4:Display
5:Delete
6:Exit
Enter your choice:4
Enter your account number:1
Name :  samant kumar
Age :  18
Amount :  1000.0
Account displayed successfully
```

OUTPUTS:-

DELETING ACCOUNT:-

```
1:Create
2:Deposit
3:Withdraw
4:Display
5:Delete
6:Exit
Enter your choice:5
Enter your account number:1
Account deleted successfully
```

EXITING BMS:-

```
1:Create
2:Deposit
3:Withdraw
4:Display
5:Delete
6:Exit
Enter your choice:6
Bank management system exited
Thank you for using bank management system
```

TESTING APPROACH :-

- MONITORING AND LIVE TESTING:-Observing system behavior in the production environment.

- REGRESSION TESTING:-Ensuring new code changes don’t break existing functionality.

- PERFORMANCE TESTING:-System must handle extreme load .

CHALLENGES FACED:-

- ZERO TOLERANCE FOR ERROR:-Any calculation or reporting error can lead to massive fines, legal liabilities or reputational damage.

- MANAGING CUSTOMER EXPECTATIONS:-The system must provide a consistent, high quality and fast user experience across all platforms.

- EVOLVING REGULATORY:-The system must be designed to be flexible enough to rapidly adapt to any change without a major overhaul.

LEARNINGS & TAKEAWAYS :-

- Learnt how to structure a Python project with packages.

- Gained deeper knowledge of Data Structure, specifically Dictionaries, and how they could represent complex decision trees.

- Learnt the importance of separating "Logic" from "Data" in making code maintainable.

FUTURE ENHANCEMENTS :-

- **USE OF AI AND ML:-** AI and ML move the system from reactive management to proactive management.

- **BLOCKCHAIN:-** Blockchain enhances security,transparency and speed in specific banking areas.

REFERENCES :-

1. Books and Academic references

2. Core banking system(CBS) Architecture and Design

3. Python 3 Documentation