Cash Flow Minimizer System

1. Introduction

The Cash Flow Minimizer System is a software application designed to reduce the number of financial transactions required among multiple banks by optimizing how funds are transferred. This system is especially useful in global financial environments where different banks use various payment methods and may not be directly compatible with each other. The system uses a central World Bank as an intermediary to facilitate transactions between banks without a shared mode of payment.

2. Objective

The goal of this project is to:
- Minimize the number of cash transactions among a network of banks.
- Ensure that payments are routed efficiently considering common payment modes.
- Introduce an intermediary (World Bank) to resolve compatibility issues between banks with no common payment mode.
- Visualize the final optimized transaction paths clearly.

3. Key Features

- Input handling for banks with different payment modes.
- Construction of a debt graph based on transaction history.
- Calculation of net balances for each bank.
- Resolution of debts through the minimum number of transactions.
- Payment mode matching using common types.
- Use of World Bank as fallback mechanism.
- Clear output for final settlements.

4. System Design

4.1 Data Structures

- Class `bank`: Stores the name, net balance, and a set of payment types for each bank.
- 2D vector `graph`: Represents the amount owed from one bank to another.
- `ansGraph`: Stores the minimized set of final transactions including the amount and mode of payment.
- `unordered_map indexOf`: Maps bank names to their indices for quick access.

4.2 Algorithms

Net Balance Calculation: For each bank:
netAmount = sum of money received - sum of money paid

Debt Simplification:
- Identify the bank with the maximum net credit and maximum net debt.
- If they share a common payment mode, make a transaction directly.
- Otherwise, route the transaction through the World Bank using two steps.

Payment Mode Matching:
Uses `set_intersection()` to find common payment modes between banks.

5. Implementation Workflow

Step 1: Input - Total number of banks and their payment modes.
Step 2: Graph Construction - Build a 2D matrix representing the debt relationships.
Step 3: Minimization Logic - Repeatedly match debtor-creditor pairs.
Step 4: Output - Final set of optimized transactions using a clear and readable format.

6. Sample Input/Output

Input:
Number of banks: 3
Bank 0: WorldBank 3 cash cheque crypto
Bank 1: BankA 2 cash cheque
Bank 2: BankB 1 crypto
Number of transactions: 2
BankA BankB 100
BankB BankA 30

Output:
The transactions for minimum cash flow are as follows:
BankA pays Rs 70 to BankB via cash

7. Benefits of This Approach

- Reduces financial overhead
- Improves performance
- Supports real-world constraints
- Scalable to large networks

8. Code Highlights

- `getMaxIndex`: Matches banks using shared payment types.
- `minimizeCashFlow`: Main logic for minimizing net balances.
- `printAns`: Outputs final simplified transactions.

9. Limitations

- Assumes no spaces in bank names or payment types.
- Matching uses brute-force logic.
- No dynamic updates during runtime.

10. Future Enhancements

- UI Integration
- Database Support
- Real-time Optimization
- Enhanced Security

11. Conclusion

The Cash Flow Minimizer System demonstrates a practical and efficient method for reducing the complexity of financial transactions across a global network. It balances debts with the least number of transfers while addressing compatibility of payment modes via a World Bank intermediary.

12. Tools Used

- Language: C++
- STL Features: set, vector, unordered_map, set_intersection
- IDE: VS Code / CodeBlocks / g++

13. References

- C++ STL documentation: https://cplusplus.com/
- Greedy Algorithm Principles
- Graph Theory – Net Flow Balancing