1. Title Page:

• Project Title : Cash Flow Minimizer System

• Name of the Student(s) :
1. Shravani Sawant - 24MCS0045
2. Aarthi M - 24MCS0017

• Course : MCSE501L
• Date of Submission : 14/11/2024

2. Objective

The objective of this mini project is to implement a **Cash Flow Minimizer System** that reduces the number of transactions among multiple banks by optimizing payments using different payment modes. This system aims to minimize the total number of transactions while ensuring that all debts are settled efficiently, even if some banks do not share common payment methods.

3. Introduction:

In today's digital economy, financial institutions are interconnected through various transactions involving multiple banks and payment channels. Optimizing these inter-bank transactions is crucial to reduce processing time, lower transaction fees, and improve the efficiency of financial networks. With advancements in payment technology, banks now support a variety of payment modes such as GPay, Paytm, PhonePe, etc., which can make direct transactions between them challenging if they do not share common modes of payment. To address this challenge, the problem involves creating an optimized solution that minimizes the number of transactions between banks while also considering payment compatibility. Additionally, a central entity known as the "World Bank" serves as an intermediary when no direct payment mode is shared between the banks, facilitating transactions that would otherwise be impossible due to payment incompatibility.

Understanding the Problem Context

In a typical network of banks, transactions occur frequently as banks lend and borrow funds to balance their respective accounts. For instance, Bank A may owe Bank B, and Bank C might owe Bank A. In such cases, an efficient settlement system can significantly reduce the number of individual transactions by identifying the minimum number of cash flows necessary to balance all accounts. Instead of having multiple individual transactions between each debtor and creditor, the system should ideally aggregate debts and credits, reducing transaction redundancy. The overall goal is to achieve "net settlement," where each bank either pays or receives a single net amount, thus minimizing transaction counts and operational costs.

However, the complication arises when banks do not share common payment methods. In this situation, even if Bank A owes Bank B, they cannot complete the transaction if they do not have compatible payment modes (for example, Bank A only uses GPay and Paytm, while Bank B uses PhonePe). Here, the World Bank steps in as a central intermediary that has access to all payment modes, enabling it to facilitate the transaction. The World Bank can receive money from the debtor in a mode it supports and transfer it to the creditor using a compatible mode. This role is crucial in ensuring that all transactions are possible, regardless of direct compatibility between banks.

Key Elements of the Solution

1. **Minimizing Transactions**: The primary goal of this solution is to minimize the total number of transactions required to settle all debts across the network. This is achieved by computing the net balance for each bank — the difference between what it owes and what it is owed. For example, if Bank A owes a total of $500 across various banks and is also owed $300, it will have a net payable amount of $200. This netting process ensures that

each bank only makes one transaction to settle its total debt, rather than multiple smaller transactions.

2. **Optimizing Cash Flow**: After calculating net balances, the system identifies the minimum number of transactions needed for settlement. The algorithm will determine the banks with net credits and net debits and match them up to settle amounts efficiently. For instance, if Bank A has a net debt of $200 and Bank B has a net credit of $200, they can directly settle their accounts, thus eliminating unnecessary intermediate transactions.
3. **Handling Payment Mode Compatibility**: Since banks may use different payment modes, each transaction needs to be checked for compatibility. If both the debtor and creditor bank have a common payment method (e.g., both support GPay), they can settle directly. However, if there's no shared payment mode, the World Bank intervenes. The debtor bank transfers the amount to the World Bank using its available payment mode, and the World Bank subsequently transfers the amount to the creditor bank using a mode compatible with the creditor. This approach ensures seamless transactions without forcing all banks to adopt a uniform payment method.
4. **Role of the World Bank as an Intermediary**: The World Bank acts as a central clearinghouse for incompatible transactions, enabling efficient settlements without requiring banks to adjust their payment infrastructures. This centralized role helps reduce transaction delays and processing complexities that would arise from individual banks needing to coordinate with each other over multiple payment methods. It also mitigates the problem of incompatible payment systems, which is a common issue in decentralized financial networks.

4. Methodology

The Cash Flow Minimizer System utilizes specific data structures and a Greedy Algorithm approach to efficiently reduce the number of transactions among banks. The chosen data structures and algorithm ensure the solution is both efficient and scalable.

1. **Chosen Data Structures**:
    - **Class Bank**: Represents each bank with properties like name, netAmount (indicating the net credit or debit), and types (a set of payment modes supported by the bank). This structure encapsulates all necessary attributes, allowing easy access to each bank's details.
    - **Set**: Used within the Bank class to store each bank's unique payment modes. A set ensures that duplicate payment modes are avoided, maintaining a clear list of available transaction methods.
    - **Vector**: Employed to represent the transaction graph as a dynamic array that holds details of the transactions between banks, such as the amounts involved and the debtor-creditor relationships.
    - **Map (unordered_map)**: Provides quick access to bank indices by mapping each bank's name to its position in the transaction graph, improving lookup efficiency.
2. **Algorithm**: The algorithm is based on a Greedy Approach for minimizing cash flow between banks:
    - **Calculate Net Amount**: Compute the net balance (credit or debit) for each bank by summarizing all transactions. This balance forms the basis for determining which banks need to pay or receive funds.
    - **Identify Min and Max Index**: Determine the banks with the highest credit and the highest debt. This identification helps in matching creditors with debtors to minimize the number of necessary transactions.
    - **Transaction Minimization**: Using shared payment modes, the algorithm iteratively reduces debts by creating optimal transactions that settle accounts with minimal exchanges.
    - **Print Results**: Finally, the minimized transactions are displayed, detailing the optimized cash flow between banks.
3. **Development Environment**:
    - **Programming Language**: C++
    - **IDE**: Visual Studio Code
    - **Compiler**: GCC

4. **Relevance to Course of Study**

This project aligns with the objectives of MCSE501L by emphasizing data structures, algorithms, and problem-solving techniques, particularly in the context of optimization and efficient resource management. By applying these concepts to real-world banking transactions, the project demonstrates practical skills in computational efficiency and resource optimization.

5. Implementation Details:

The program starts by initializing the list of banks and their respective payment modes. It reads the number of banks, payment modes, and transaction details. The minimizeCashFlow function computes the net amount for each bank and reduces the number of transactions using the getMinIndex and getMaxIndex functions. The minimized transactions are then displayed.

Code Snippets:

Finding the Bank with Minimum Debt:

```
int getMinIndex(bank listOfNetAmounts[], int numBanks) {
    int min = INT_MAX, minIndex = -1;
    for (int i = 0; i < numBanks; i++) {
                    if (listOfNetAmounts[i].netAmount < min &&
listOfNetAmounts[i].netAmount != 0) {
            minIndex = i;
            min = listOfNetAmounts[i].netAmount;
        }
    }
    return minIndex;
}
```

Transaction Minimization:

```
void minimizeCashFlow(int numBanks, bank input[],
unordered_map<string, int>& indexOf, int numTransactions,
vector<vector<int>>& graph, int maxNumTypes) {
    bank listOfNetAmounts[numBanks];
    // Calculate net amounts and minimize transactions
    printAns(ansGraph, numBanks, input);
}
```

Challenges

- **Handling Different Payment Modes:** Matching payment modes between banks was complex and required the use of set intersections.
- **Optimizing Performance:** Iterating through banks and finding intersections had a high time complexity, which was addressed by reducing redundant operations.

6. Results:

```
Enter the number of banks participating in the transactions.
6
Enter the details of the banks and transactions as stated:
Bank name ,number of payment modes it has and the payment modes.
Bank name and payment modes should not contain spaces
World Bank : Reserve_bank 3 gpay paytm phonepay
Bank 1 : State_bank 2 gpay paytm
Bank 2 : Iob 1 paytm
Bank 3 : Canara 2 gpay phonepay
Bank 4 : federal 1 phonepay
Bank 5 : Boi 2 paytm phonepay
Enter number of transactions.
9
Enter the details of each transaction as stated:Debtor Bank , creditor Bank and amount
The transactions can be in any order
0 th transaction : federal Reserve_bank 100
1 th transaction : federal State_bank 300
2 th transaction : federal Iob 100
3 th transaction : federal Canara 100
4 th transaction : Boi Reserve_bank 300
5 th transaction : Boi Iob 100
6 th transaction : Reserve_bank State_bank 400
7 th transaction : State_bank Iob 200
8 th transaction : Iob Canara 500

The transactions for minimum cash flow are as follows :

Iob pays Rs 100 to State_bank via paytm
Boi pays Rs 400 to State_bank via paytm
federal pays Rs 600 to Canara via phonepay
```

7. Testing:

Input Details

1. **Number of Banks Participating**: 6
2. **Bank Details**:
    - **World Bank**: Payment modes - gpay, paytm, phonepay
    - **Bank 1** (State_bank): Payment modes - gpay, paytm
    - **Bank 2** (Iob): Payment mode - paytm
    - **Bank 3** (Canara): Payment modes - gpay, phonepay
    - **Bank 4** (federal): Payment mode - phonepay
    - **Bank 5** (Boi): Payment modes - paytm, phonepay
3. **Number of Transactions**: 9
4. **Transaction Details**:
    - **Transaction 0**: Debtor - federal, Creditor - Reserve_bank, Amount - 100
    - **Transaction 1**: Debtor - federal, Creditor - State_bank, Amount - 300
    - **Transaction 2**: Debtor - federal, Creditor - Iob, Amount - 100
    - **Transaction 3**: Debtor - federal, Creditor - Canara, Amount - 100
    - **Transaction 4**: Debtor - Boi, Creditor - Reserve_bank, Amount - 300
    - **Transaction 5**: Debtor - Boi, Creditor - Iob, Amount - 100
    - **Transaction 6**: Debtor - Reserve_bank, Creditor - State_bank, Amount - 400
    - **Transaction 7**: Debtor - State_bank, Creditor - Iob, Amount - 200
    - **Transaction 8**: Debtor - Iob, Creditor - Canara, Amount - 500

Output Details

- **Transactions for Minimum Cash Flow**:
    - Iob pays Rs 100 to State_bank via paytm
    - Boi pays Rs 400 to State_bank via paytm
    - federal pays Rs 600 to Canara via phonepay

This test case involves calculating the minimum cash flow between banks based on the transactions provided, while ensuring the transactions use valid payment modes available to both debtor and creditor banks. The goal is to minimize the number of transactions required for settlements.

8. Conclusion:

The Cash Flow Minimizer System efficiently reduces inter-bank transactions by using structured data management and a Greedy Algorithm. By employing data structures like classes, sets, vectors, and maps, it organizes transaction data, calculates net balances for each bank, and minimizes redundant cash flows. The World Bank acts as an intermediary, allowing banks with different payment modes to complete transactions without modifying their systems. This project demonstrates real-world applications of data structures and algorithms, emphasizing optimization and resource management—a core focus of the MCSE501L curriculum. Overall, the system showcases practical computer science concepts relevant to efficient financial transaction networks.