REPORT ON BANK MANAGEMENT SYSTEM

Almas Shafqat(006)

BACKGROUND:

Before the advent of computerized Bank Management Systems, the operations of banks were predominantly manual and paper-based. The management of banking activities relied heavily on physical records, manual calculations, and human intervention. Here is an overview of the background of bank management before the implementation of any management system:

1. **Manual Record Keeping:** Banks maintained extensive paperwork for various aspects of their operations, including customer accounts, transactions, loans, and other financial activities. Each customer's account information, transaction history, and other details were recorded manually in ledger books.

2. **Limited Automation:** Early on, some basic automation tools were introduced, such as mechanical calculators and typewriters. However, these tools were limited in their capability and did not provide comprehensive solutions for managing the growing complexity of banking operations.

3. **Paper-based Transactions:** Every banking transaction, whether it involved depositing money, withdrawing funds, or transferring funds between accounts, required paper documentation. Checks, deposit slips, and withdrawal forms were commonly used for these transactions.

4. **Queue-based Customer Service**: Customers had to visit the bank in person for most transactions, leading to long queues and wait times. The interaction with bank staff was primarily face-to-face, and the availability of services was often restricted to regular banking hours.

5. **Manual Calculations:** Financial calculations, including interest calculations, loan approvals, and account balances, were performed manually by bank employees. This manual approach increased the likelihood of errors and required significant time and effort.

6. **Limited Accessibility:** Access to account information and banking services was limited to the physical presence of customers at the bank branch. There were no electronic channels or online platforms for customers to access their accounts remotely.

7. **Security Challenges:** The security of customer data relied on physical measures such as locked vaults and secure storage of paper records. The absence of advanced security measures made the banking system susceptible to risks associated with physical theft or unauthorized access.

8. **Time-Consuming Processes:** Tasks that are now automated, such as account updates, transaction verifications, and account reconciliation, were time-consuming when performed manually. This often resulted in delays and inefficiencies in the overall banking processes.

The transition to computerized Bank Management Systems marked a significant advancement in the efficiency, accuracy, and accessibility of banking operations. The introduction of technology streamlined processes, enhanced security, and paved the way for the modern banking systems we use today.

Introduction:

The Bank Management System is a meticulously crafted console application in C language, providing comprehensive functionality for the efficient management of crucial operations within a banking environment. It is designed to handle account operations, search, update, and delete operations within a bank. The system employs a linked list structure to efficiently store account information.

At its core, the system utilizes a sophisticated linked list structure to systematically store and organize account-related data, ensuring optimal efficiency in data management and retrieval processes.

The significance of the Bank Management System lies in its ability to streamline and automate essential tasks associated with account management within a bank, providing a robust foundation for efficient banking administration. Through the integration of advanced data structuring techniques, specifically the utilization of linked lists, the system optimizes the storage and retrieval of account-specific information, enhancing the overall operational effectiveness of bank staff.

The subsequent sections of this project report will delve into the intricate details of the project structure, user interface design, bank management code logic, code structure, project execution flow, working environment specifications, employed header files, and conclude by summarizing the significance of the implemented Bank Management System

Project Structure:

The project structure consists of a main source code file, `main.c`, containing the logic for a simple console-based bank management system. Transaction details are stored in a text file named `Account.txt`. The root directory, named "BankManagementSystem," also includes a README.md file to provide information about the program's usage, dependencies, and any other relevant details. This straightforward structure is suitable for a single-file project, with potential for expansion as the project grows in complexity or additional features are added.

User Interface:

The bank management system uses a simple console interface where users input their name and account number. It offers options like deposit, withdrawal, transfer, and account details through a numeric menu. The program operates in a loop, executing the chosen functions and displaying transaction details in a text file. Users can interact with the basic banking features until choosing to exit.

Bank Management Code Logic:

1. **Global Variables:**
    - **name**: An array to store the customer's name.
    - **dip_amt**, **with_amt**, **trans_amt**: Variables to store deposit, withdrawal, and transfer amounts.

- **amt**: Variable to store the current account balance.
- **acc_no**: Variable to store the account number.
- **count**: A counter to keep track of the number of transactions.

2. **Function Prototypes:**
    - **menu()**: Displays the main menu options.
    - **deposit_money()**: Handles the logic for depositing money into the account.
    - **withdraw_money()**: Handles the logic for withdrawing money from the account.
    - **transfer_money()**: Handles the logic for transferring money between accounts.
    - **account_details()**: Displays the details of the account, including name, account number, and balance.
    - **transaction_details()**: Displays recent transaction details from a file.
3. **Main Function (main()):**
        - Takes user input for name and account number.
        - Enters an infinite loop displaying the main menu and waits for user input.
        - Based on the user's choice, it calls the corresponding function:
        - **deposit_money()**: Takes an amount as input, adds it to the account balance, and records the transaction details.
        - **withdraw_money()**: Takes an amount as input, subtracts it from the account balance, and records the transaction details.
        - **transfer_money()**: Takes an amount and the recipient's account number, transfers the amount if the balance is sufficient, and records the transaction details.
        - **account_details()**: Displays the customer's name, account number, and current balance.
        - **transaction_details()**: Reads and displays recent transaction details from a file.
4. **File Handling:**
    - The program uses file handling to store and retrieve transaction details in a text file (**Account.txt**).

- For each transaction, it appends the details to the file, including the transaction amount, type, date, and time.

5. **Loop and Exit:**
    - The program continues to display the main menu until the user chooses to exit (option 6).

Code Structure:

The code follows a straightforward structure, beginning with the declaration of global variables, including those for user details and account balances. It includes function prototypes for various operations such as deposit, withdrawal, transfer, and displaying account and transaction details. The `**main()**` function initializes user information and enters an infinite loop presenting a menu, where user choices trigger corresponding functions. These functions handle user interactions, update balances, and record transaction details in a text file. The code utilizes file handling for reading and writing transaction information. While relatively simple, the program incorporates an organized flow, modular functions, and file management to create a functional console-based bank management system.

Project Execution:

The program starts by asking account number and name from user..

Users interact with the system by choosing options from the main menu.

Based on the user's choice, the program performs corresponding actions using methods from the Bank structure.

The program continues to run until the user chooses to exit.

Working Environment:

The project is a console-based application in C language, suitable for execution in a command-line environment. It does not rely on any external dependencies.

Header Files:

The code includes the necessary standard C headers:

<stdio.h> for input and output operations.

< ctime> for the **time_t** type,

<stdlib.h> for dynamic memory allocation.

Conclusion:

The Bank Management System provides a basic yet effective way to manage account operations within a bank. The use of linked lists allows for efficient storage and retrieval of

account information. The console-based interface ensures user-friendly interaction. The modular code structure allows for easy maintenance and future enhancements.

Output Screenshots:

```
Enter your name: Sara
Enter your account number: 1223

*****************MAIN MENU*****************
1. Deposit money
2. Withdraw money
3. Transfer money
4. Account details
5. Transaction details
6. Exit
Enter your choice:1
-----------------------------------------------
*****************DEPOSITING MONEY*****************
-----------------------------------------------
Enter the amount you want to deposit: 500
-----------------------------------------------
*****************MONEY DEPOSITED*****************
-----------------------------------------------
Current balance: 10500
```

```
*****************MAIN MENU*****************
1. Deposit money
2. Withdraw money
3. Transfer money
4. Account details
5. Transaction details
6. Exit
Enter your choice:2
-----------------------------------------------
*****************WITHDRAW MONEY*****************
-----------------------------------------------
Enter the amount you want to withdraw:500
-----------------------------------------------
*****************MONEY WITHDRAWN*****************
-----------------------------------------------
Current balance: 9500
```

```
**********MAIN MENU**********
1. Deposit money
2. Withdraw money
3. Transfer money
4. Account details
5. Transaction details
6. Exit
Enter your choice:3
-------------------------------------------------
**********TRANSFERRING MONEY**********
-------------------------------------------------
Enter the amount you want to transfer: 1500
Enter the account number in which you want to transfer: 1255
-------------------------------------------------
**********MONEY TRANSFERRED**********
-------------------------------------------------
Current balance: 8500
```

```
**********MAIN MENU**********
1. Deposit money
2. Withdraw money
3. Transfer money
4. Account details
5. Transaction details
6. Exit
Enter your choice:4
-------------------------------------------------
**********ACCOUNT DETAILS**********
-------------------------------------------------
Name: Sara
Account Number: 1223
Total balance: 10000
```

```
*****************MAIN MENU*****************
1. Deposit money
2. Withdraw money
3. Transfer money
4. Account details
5. Transaction details
6. Exit
Enter your choice:5

******************TRANSACTION DETAILS*****************

Rs100 has been deposited to your account
Date/Time of transaction :  Fri Jan 12 10:18:20 2024

Rs100 has been withdrawn from your account
Date/Time of transaction :  Fri Jan 12 10:18:27 2024

Rs100 has been transferred from your account to 1244
Date/Time of transaction :  Fri Jan 12 10:18:34 2024
```