Manthan Kansagara

22BCE140

Bank Management System Report

Introduction

The **Bank Management System** is a console-based application designed to perform basic banking operations. These include creating an account, depositing money, withdrawing money, and checking the balance. The program leverages file handling for persistent storage of account data.

This document explains the functionality of the system, the programming constructs used, and the constraints imposed during development.

Functionalities

1. Create Account

- **Purpose**: Allows users to create a new bank account.
- **Process**:
    - User enters their name and account number.
    - Initial balance is set to 0.
    - Account details are stored in a file (account.dat).
- **Key Functions Used**:
    - fopen(): Opens the file in append mode (a) to add new records.
    - fprintf(): Writes account details in a formatted manner to the file.
    - fclose(): Closes the file after writing.

2. Deposit Money

- **Purpose**: Enables users to deposit money into their account.
- **Process**:
    - User enters their account number and the deposit amount.
    - The system searches for the account in the file.
    - If found, the balance is updated, and the new details are written to a temporary file.

    - The original file is replaced with the temporary file.
- **Key Functions Used**:
    - fscanf(): Reads account details from the file.
    - fprintf(): Writes updated account details to the temporary file.
    - remove() and rename(): Replaces the original file with the updated file.

3. Withdraw Money

- **Purpose**: Allows users to withdraw money from their account.
- **Process**:
    - User enters their account number and the withdrawal amount.
    - The system searches for the account in the file.
    - If the account exists and has sufficient balance, the balance is reduced.
    - The updated details are written to a temporary file, which replaces the original.
- **Key Functions Used**:
    - Same as those in the deposit functionality.

4. Check Balance

- **Purpose**: Displays the current balance of a user's account.
- **Process**:
    - User enters their account number.
    - The system searches for the account in the file and displays the balance if the account exists.
- **Key Functions Used**:
    - fscanf(): Reads account details from the file.
    - fclose(): Closes the file after reading.

Programming Constructs and Their Use

1. **File Handling**:
    - Used to store account details persistently.
    - Files allow data to remain available even after the program exits.
    - Modes used:

        - a (append): For adding new records.
        - r (read): For reading existing records.
        - w (write): For creating temporary files.

2. **Structures**:
    - The Account structure is used to group account-related data (name, account number, balance).
    - This improves code organization and makes it easier to handle related data.
3. **Temporary Files**:
    - Temporary files (temp.dat) are used to update existing records.
    - This ensures data consistency and avoids overwriting issues.
4. **Conditional Statements**:
    - if and else are used for decision-making, such as checking if an account exists or if a balance is sufficient for withdrawal.
5. **Loops**:
    - while loops are used to read multiple records from the file and to allow users to perform multiple operations without restarting the program.
6. **Error Handling**:
    - File opening operations are checked to ensure the file exists and is accessible.
    - Appropriate messages are displayed when operations fail (e.g., account not found).

Constraints and Assumptions

1. **Account Number**:
    - Each account must have a unique account number.
    - The system does not enforce uniqueness programmatically, so the user must ensure no duplicates.
2. **File Integrity**:
    - The program assumes that the account.dat file exists and contains correctly formatted data.
    - Corrupt or manually edited files may cause unexpected behavior.
3. **Data Validation**:

- User input is not rigorously validated (e.g., entering non-numeric values for account numbers).
- Future enhancements could include stricter input validation.

4. **Concurrency**:
    - The system does not support simultaneous access by multiple users.
    - Changes made by one instance of the program may not be reflected immediately in another.
5. **Storage Format**:
    - Data is stored in a text file in a simple format. This is sufficient for small-scale use but may be inefficient for larger datasets.
6. **Insufficient Balance**:
    - Withdrawals are only allowed if the account balance is greater than or equal to the withdrawal amount.

Conclusion

The Bank Management System demonstrates basic file handling and account management operations in C. While functional for small-scale use, future enhancements could make it more robust, secure, and user-friendly.