Banking System Project Report

1. Introduction:

This report outlines the simple banking system developed using C programming language. The system facilitates similar functions to that of a real time banking system such as creating accounts, logging into the accounts, performing actions such as credit, debit or balance checking only when user is logged in. This project showcases the use of file handling, data structures, loops, functions, etc.

2. Project Objectives

The primary objectives of this banking system project are:

- **To create a user-friendly banking interface**
- **To provide basic banking operations (deposits, withdrawals, balance enquiry)**
- **To maintain persistent data storage for user accounts**
- **To demonstrate practical application of C programming concepts**

3. System Features

The banking system includes the following key features:

3.1 Account Management

- **Account creation with unique account numbers**
- **Secure login with password authentication**
- **Session management with logout functionality**

3.2 Transaction Operations

- **Balance inquiry**
- **Deposit funds (credit)**
- **Withdraw funds (debit)**

3.3 Data Persistence

- **Account information stored in text files**
- **Real-time data updates for transactions**

4. System Design

4.1 Data Structure

The core data structure used in this system is the Account struct, which holds:

- **Account number (10-digit unique identifier)**
- **Password (max 15 characters)**
- **Account balance**

```
struct Account {
    char accNumber[MAX_ACC_LEN + 1];
    char password[MAX_PASS_LEN + 1];
    float balance;
};
```

4.2 Global Variables

The system uses two global variables for session management:

- **isLoggedIn: A flag to track user authentication status**
- **currentUser: The currently active user account**

4.3 File Structure

Account data is stored in a text file named "accounts.txt" with each line containing:

- **Account number**
- **Password**
- **Current balance**

5. Implementation Details

5.1 Account Creation

The account creation process includes:

1. **Generating a random 10-digit account number**
2. **Taking user input for password**
3. **Initializing balance to zero**

4. **Writing account information to "accounts.txt"**

5.2 User Authentication

The login process:

1. **Prompts for account number and password**
2. **Validates credentials against stored accounts**
3. **Sets the logged-in state and loads user information**

5.3 Transaction Handling

Transactions are implemented with appropriate validations:

- **Credit: Validates positive amounts before adding to balance**
- **Debit: Checks for sufficient funds before deduction**
- **Both operations update account information in the file**

5.4 Data Management

File operations ensure data persistence:

- **Reading account information during login**
- **Creating a temporary file for updates**
- **Replacing the original file with updated information**

6. User Interface

The system implements a simple menu-driven interface. This design provides intuitive navigation through different banking operations.

1. **Create Account**
2. **Login**
3. **Check Balance**
4. **Credit Amount**
5. **Debit Amount**
6. **Log Out**
7. **Exit**