

# **BANKING SYSTEM PROJECT REPORT**

**Submitted By Pratik**

**Mehra**

**Under the guidance of**

**Dr. Srinivasan Ramchandran**

**School of Computer Science**

**University of Petroleum and Energy Studies**

## Abstract

This project is a simple command-line banking system built in the C programming language. It supports basic operations like creating an account, depositing money, withdrawing money, and viewing balance.

The program uses **file handling** to save and load account data from a text file, allowing the system to retain information between runs.

---

## Problem Definition

A small bank wants a simple terminal-based system to manage basic customer transactions. The system should allow multiple accounts and store all data permanently.

---

## Objectives

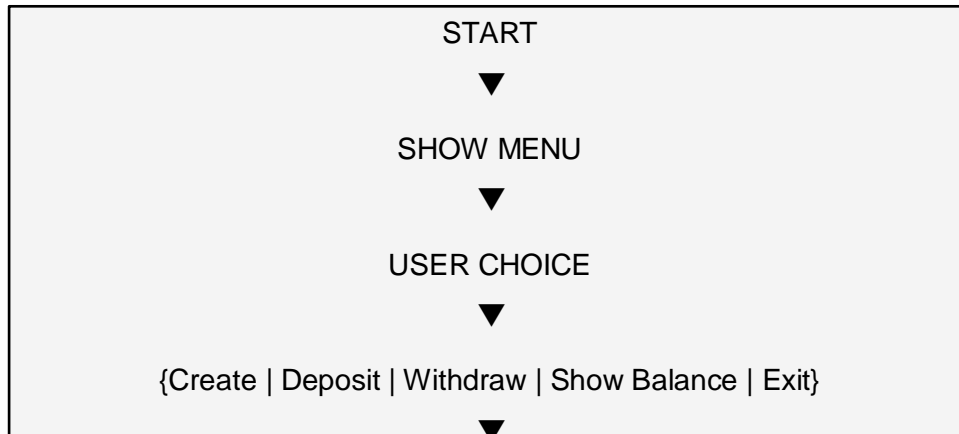
- Create a simple and understandable banking application
  - Use structures in C to represent accounts
  - Implement file handling to save account data
  - Provide user interaction using a menu system
- 

## System Requirements

- C compiler (GCC)
  - VS Code or any code editor
  - Linux/Windows
- 

## System Design

## Flowchart



---

## Algorithm

### Create Account

1. Ask user for name
2. Generate new account number
3. Store account in array
4. Save array to file

### Deposit

1. Enter account number
2. Search account
3. Add amount
4. Save changes

### Withdraw

1. Enter account number
2. Check balance
3. Deduct amount
4. Save changes

### Show Balance

1. Enter account number
2. Display name and balance

#### **File Load**

1. Open accounts.txt
2. Read each line
3. Fill array

#### **File Save**

1. Open file in write mode
2. Write all accounts

---

## **Implementation**

### **Language and Approach**

The project is made in C using a menu-driven program.

Functions are used for each banking operation and file handling is used to save accounts.

### **Structure Used**

```
typedef struct {  
    int accountNumber;  
    char name[50];  
    float balance;  
} Account;
```

### **Files in the Project**

- bank.h – declarations
- bank.c – functions
- main.c – menu + calling functions
- accounts.txt – file where data is saved

## File Handling

Two main functions handle permanent storage:

Load accounts at program start:

```
int loadAccounts(Account a[]) {  
    FILE *f = fopen("accounts.txt", "r");  
    if(!f) return 0;  
    int c=0;  
    while(fscanf(f, "%d %s %f", &a[c].accountNumber, a[c].name, &a[c].balance) == 3)  
        c++;  
    fclose(f);  
    return c;  
}
```

Save accounts after any change:

```
void saveAccounts(Account a[], int n) {  
    FILE *f = fopen("accounts.txt", "w");  
    for(int i=0; i<n; i++)  
        fprintf(f, "%d %s %.2f\n", a[i].accountNumber, a[i].name, a[i].balance);  
    fclose(f);  
}
```

## Menu Logic

1. Create Account
2. Deposit
3. Withdraw
4. Show Balance
5. Exit

Each choice calls its respective function.

## Functions Implemented

- **createAccount()** – stores name + acc number
  - **deposit()** – adds amount
  - **withdraw()** – deducts amount
  - **showBalance()** – prints details
  - **saveAccounts() / loadAccounts()** – file storage
- 

## Code Snippets

### Structure

```
typedef struct {  
    int accountNumber;  
    char name[50];  
    float balance;  
} Account;
```

### File Save Function

```
void saveAccounts(Account accounts[], int count);
```

### File Load Function

```
int loadAccounts(Account accounts[]);
```

---

## Testing

### Test Case 1: Create Account

Input:

Name = Raj

Output: Account Created: 1000

```
--- Banking System ---
1. Create Account
2. Deposit
3. Withdraw
4. Show Balance
5. Exit
Enter choice: 1
Enter your name: Raj
Account created and saved.

--- Banking System ---
1. Create Account
2. Deposit
3. Withdraw
4. Show Balance
5. Exit
Enter choice: 5
Data saved. Exiting...
```

### Test Case 2: Deposit

Input:

Account: 1000

Amount: 500

Output: Deposit done.

```
--- Banking System ---  
1. Create Account  
2. Deposit  
3. Withdraw  
4. Show Balance  
5. Exit  
Enter choice: 2  
Account Number: 1000  
Amount: 500  
Deposit done and saved.
```

```
--- Banking System ---  
1. Create Account  
2. Deposit  
3. Withdraw  
4. Show Balance  
5. Exit  
Enter choice: 5  
Data saved. Exiting...
```

### Test Case 3: Withdraw

Input:

Account: 1000

Amount: 200

Output: Withdraw done.



```
--- Banking System ---
1. Create Account
2. Deposit
3. Withdraw
4. Show Balance
5. Exit
Enter choice: 3
Account Number: 1000
Amount: 200
If successful, changes saved.

--- Banking System ---
1. Create Account
2. Deposit
3. Withdraw
4. Show Balance
5. Exit
Enter choice: 5
Data saved. Exiting...
```

---

## Sample File Output

Inside accounts.txt:

1000 Raj 300.00

---

## Conclusion

The project successfully demonstrates how file handling, structures, and functions can be combined to create a functional banking system in C.

It is easy to extend and modify and gives a good understanding of basic C programming concepts.

---

## References

- Class lecture notes
- GCC manual

- Basic C Programming Books
-