

# Bank Management System

Name\_1 : Abhishek Gaur

SAP ID\_1 : 590022408

Name\_2: Neha Bisht

SAP ID\_2 : 590028765

## ● Problem Statement :

A Bank Account System is essential for managing personal finances, allowing users to securely store money, check balances, and transfer funds. Using C language, we can create an application that can manage the data of the Bank, we use basic knowledge of C like string, array, structures, etc.

The functionality of the Bank Management System Project is mentioned below:

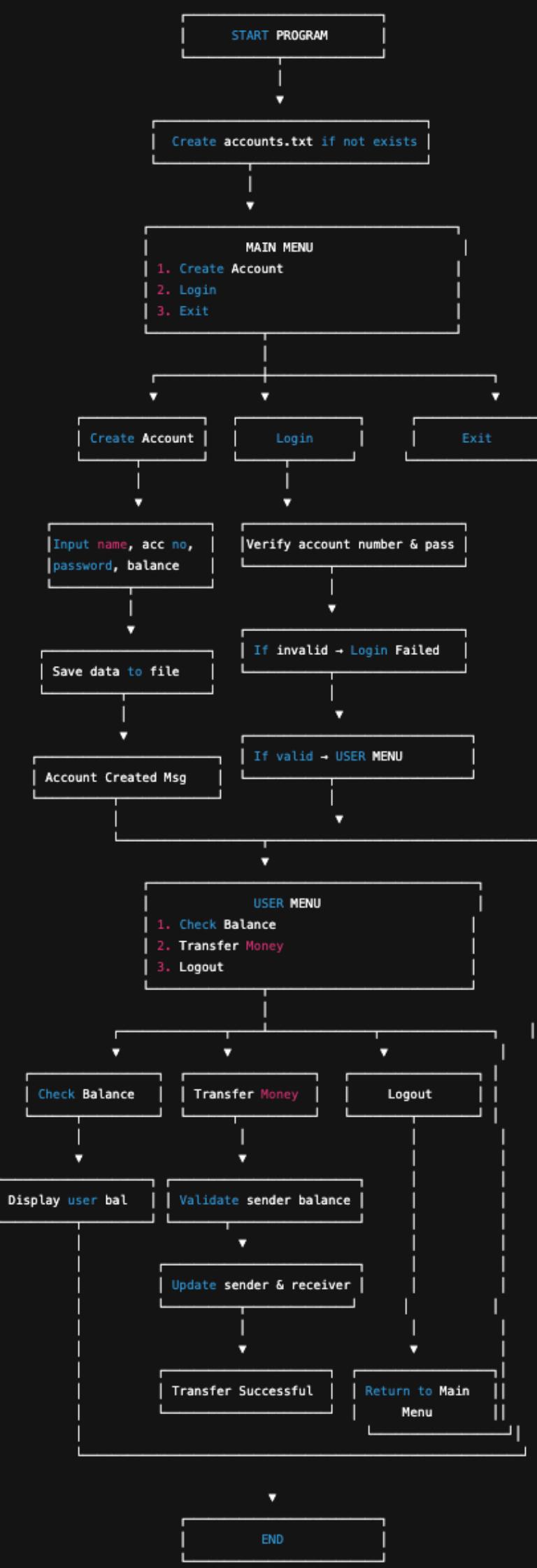
Transfer Money to the Account

Creation of Account

Check Amount

Login Functionality

## ● Flowchart :



## ● Algorithm :

**Start the program.**

**Check whether the file accounts.txt exists.**

- If it does not exist, create the file.

**Display the Main Menu:**

1. Create Account
2. Login
3. Exit

**If the user selects “Create Account”:**

1. Ask user to enter:
  - Name
  - Account Number
  - Password
  - Initial Balance
2. Store the details in the format:  
**account:password:name:balance**
3. Display message: “*Account Created Successfully.*”
4. Return to Main Menu.

**If the user selects “Login”:**

1. Ask user for Account Number.
2. Ask user for Password.
3. Search the accounts.txt file for matching record.
4. If no record matches:
  - Display “*Invalid Login.*”

## Project Report

- Return to Main Menu.
5. If login is successful:
- Display “*Login Successful.*”
  - Go to User Menu.
- User Menu Options:**
1. Check Balance
  2. Transfer Money
  3. Logout
- If the user selects “Check Balance”:**
1. Read balance from file.
  2. Display the current balance.
  3. Return to User Menu.
- If the user selects “Transfer Money”:**
1. Ask for Receiver’s Account Number.
  2. Ask for Transfer Amount.
  3. Check if receiver exists in file.
    - If not found → Display “*Receiver not found.*”
  4. Check if sender has sufficient balance.
    - If balance < amount → Display “*Insufficient Balance.*”
  5. Deduct amount from sender.
  6. Add amount to receiver.
  7. Update both records in `accounts.txt`.
  8. Display “*Transfer Successful.*”
  9. Return to User Menu.

**If the user selects “Logout”:**

- Return to Main Menu.

**If the user selects “Exit”:**

- Terminate the program.

**End.**

## ●Problems Faced :

### 1. Avoiding Duplicate Account Numbers

Ensuring that the same account number was not created again required searching the file each time. Handling this correctly took some time.

### 2. Debugging the Script

Shell scripts do not show detailed error messages, so finding mistakes like missing quotes, incorrect syntax, or wrong variable names became difficult. Debugging needed repeated testing.

### 3. Difficulty in Understanding Shell Commands

While developing the project, it was challenging to understand how different shell commands . Learning how to use these commands correctly took extra time.

# Project Report

## ● Code :

A screenshot of a terminal window titled "abhi\_gaur2705@abhishekgaur: ~". The terminal shows a shell script for managing accounts. The script starts by checking if "accounts.txt" exists; if not, it creates it. It then defines two functions: "create\_account()" and "login()". The "create\_account()" function prompts the user for account details (name, number, password, initial balance) and saves them to the file. The "login()" function prompts the user for their account number and password, then uses grep to search for a matching record in the file. If found, it prints a success message and returns. Otherwise, it prints an invalid login message and exits. The terminal window has a dark theme and includes a dock at the bottom with various icons.

```
ACCOUNTS="accounts.txt"

if [ ! -f $ACCOUNTS ]; then
    touch $ACCOUNTS
fi

create_account() {
    echo "Enter your Name:"
    read name
    echo "Enter Account Number:"
    read acc
    echo "Set Password:"
    read pass
    echo "Enter Initial Balance:"
    read bal

    # Save in file: acc:pass:name:bal
    echo "$acc:$pass:$name:$bal" >> $ACCOUNTS
    echo "Account Created Successfully!"
}

login() {
    echo "Enter Account Number:"
    read acc
    echo "Enter Password:"
    read pass

    record=$(grep "^$acc:$pass" $ACCOUNTS)

    if [ -z "$record" ]; then
        echo "Invalid Login!"
        return
    fi
}

1,1 Top
```

A screenshot of a terminal window titled "abhi\_gaur2705@abhishekgaur: ~". The terminal continues the shell script from the previous screenshot. It starts with the "fi" of the "login()" function, followed by an "echo" command for a successful login message and a "user\_menu" call. Then, it begins a "user\_menu()" function loop. Inside the loop, it prints a menu with three options: 1. Check Balance, 2. Transfer Money, and 3. Logout. It then prompts the user for a choice and reads it into "ch". A "case" statement handles the user's choice: if "ch" is 1, it calls "check\_balance" with arguments "\$acc" and "\$pass"; if it's 2, it calls "transfer" with the same arguments; if it's 3, it breaks out of the loop; otherwise, it prints an invalid choice message. After the case block, there's an "esac" keyword, a "done" keyword, and a closing brace for the "user\_menu()" function. Finally, there's a "check\_balance()" function definition that reads the account details from "accounts.txt" and prints them. The terminal window has a dark theme and includes a dock at the bottom with various icons.

```
fi
echo "Login Successful! Welcome"
user_menu "$acc" "$pass"
}

user_menu() {
    acc=$1
    pass=$2

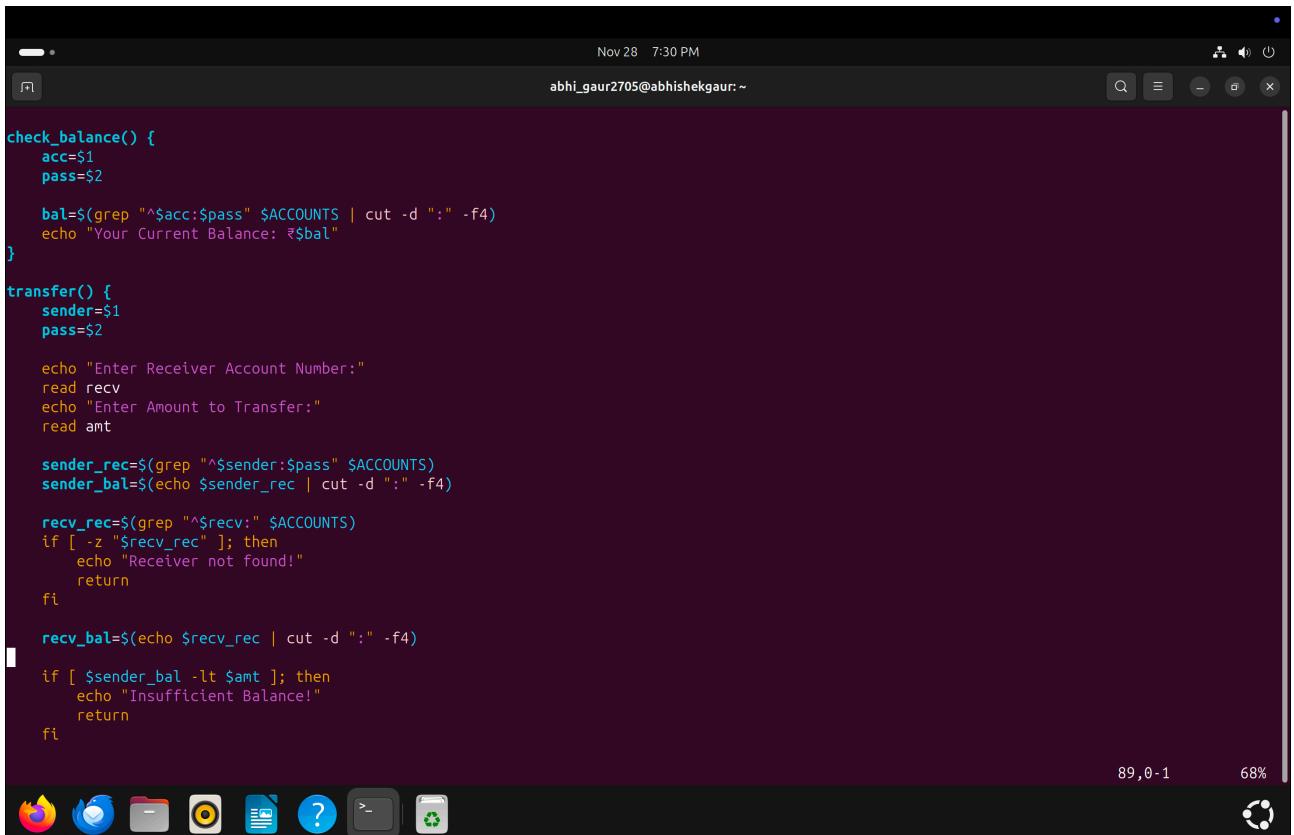
    while true
    do
        echo ""
        echo "===== USER MENU ====="
        echo "1. Check Balance"
        echo "2. Transfer Money"
        echo "3. Logout"
        echo "Enter your choice:"
        read ch

        case $ch in
            1) check_balance "$acc" "$pass" ;;
            2) transfer "$acc" "$pass" ;;
            3) break ;;
            *) echo "Invalid Choice!" ;;
        esac
    done
}

check_balance() {
    acc=$1
    pass=$2

    bal=$(grep "^$acc:$pass" $ACCOUNTS | cut -d ":" -f4)
61,0-1 36%
```

# Project Report



Nov 28 7:30 PM abhi\_gaur2705@abhishekgaur:~

```
check_balance() {
    acc=$1
    pass=$2

    bal=$(grep "^$acc:$pass" $ACCOUNTS | cut -d ":" -f4)
    echo "Your Current Balance: ₹$bal"
}

transfer() {
    sender=$1
    pass=$2

    echo "Enter Receiver Account Number:"
    read recv
    echo "Enter Amount to Transfer:"
    read amt

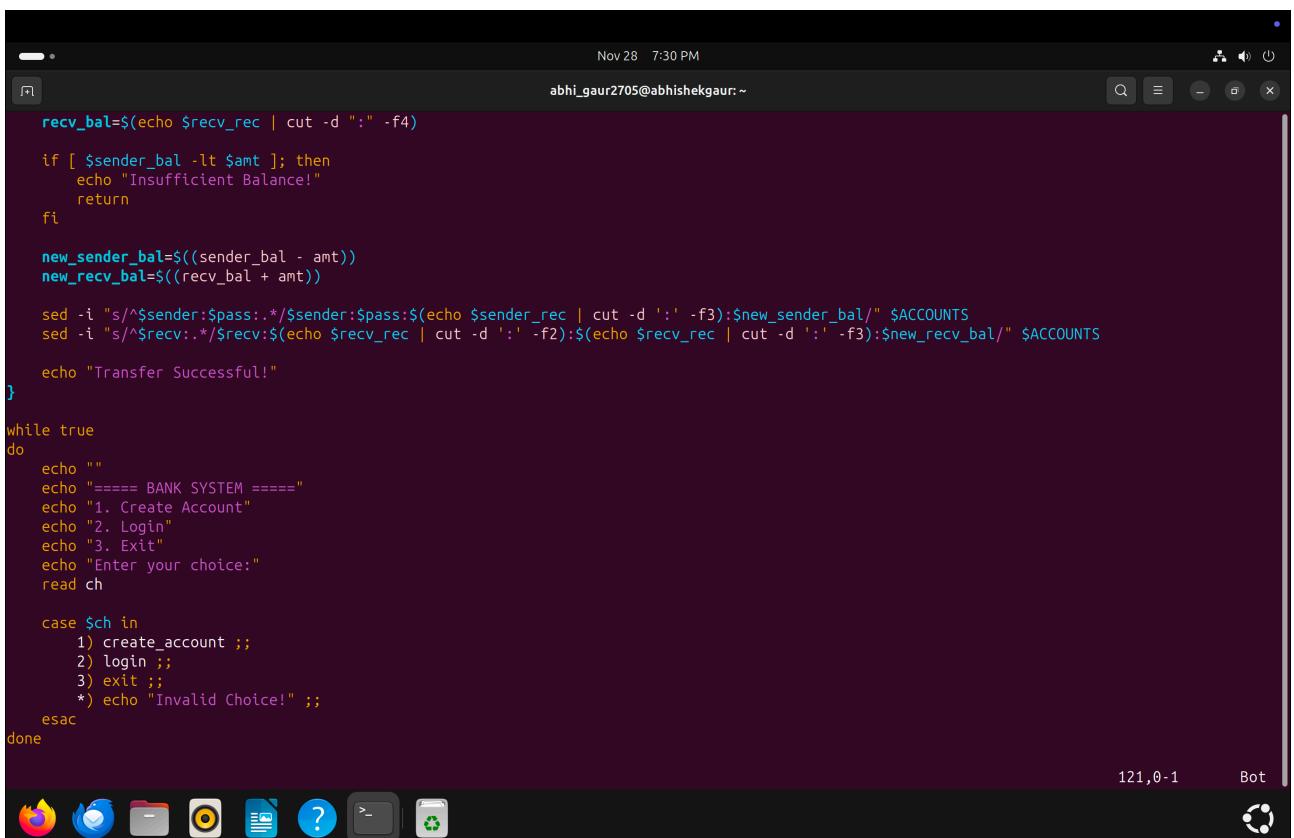
    sender_rec=$(grep "^$sender:$pass" $ACCOUNTS)
    sender_bal=$(echo $sender_rec | cut -d ":" -f4)

    recv_rec=$(grep "^$recv:" $ACCOUNTS)
    if [ -z "$recv_rec" ]; then
        echo "Receiver not found!"
        return
    fi

    recv_bal=$(echo $recv_rec | cut -d ":" -f4)

    if [ $sender_bal -lt $amt ]; then
        echo "Insufficient Balance!"
        return
    fi
}
```

89,0-1 68%



Nov 28 7:30 PM abhi\_gaur2705@abhishekgaur:~

```
recv_bal=$(echo $recv_rec | cut -d ":" -f4)

if [ $sender_bal -lt $amt ]; then
    echo "Insufficient Balance!"
    return
fi

new_sender_bal=$((sender_bal - amt))
new_recv_bal=$((recv_bal + amt))

sed -i "s/^$sender:$pass:.*$/$sender:$pass:$new_sender_bal/" $ACCOUNTS
sed -i "s/^$recv:$recv:$echo $recv_rec | cut -d ':' -f2:$new_recv_bal/" $ACCOUNTS

echo "Transfer Successful!"
}

while true
do
    echo ""
    echo "===== BANK SYSTEM ====="
    echo "1. Create Account"
    echo "2. Login"
    echo "3. Exit"
    echo "Enter your choice:"
    read ch

    case $ch in
        1) create_account ;;
        2) login ;;
        3) exit ;;
        *) echo "Invalid Choice!" ;;
    esac
done
```

121,0-1 Bot

## ● Output :

```
abhi_gaur2705@abhishekgaur:~$ ./project.sh
```

```
===== BANK SYSTEM =====
```

- 1. Create Account
- 2. Login
- 3. Exit

```
Enter your choice:
```

```
===== BANK SYSTEM =====
```

- 1. Create Account
- 2. Login
- 3. Exit

```
Enter your choice:
```

```
1
```

```
Enter your Name:
```

```
Abhishek Gaur
```

```
Enter Account Number:
```

```
1122334455
```

```
Set Password:
```

```
abhi
```

```
Enter Initial Balance:
```

```
50000
```

```
Account Created Successfully!
```

===== BANK SYSTEM =====

1. Create Account

2. Login

3. Exit

Enter your choice:

2

Enter Account Number:

1122334455

Enter Password:

abhi

Login Successful! Welcome

===== USER MENU =====

1. Check Balance

2. Transfer Money

3. Logout

Enter your choice:

1

Your Current Balance: ₹50000  
50000

## Project Report

```
===== BANK SYSTEM =====
1. Create Account
2. Login
3. Exit
Enter your choice:
2
Enter Account Number:
1122334455
Enter Password:
abhi
Login Successful! Welcome

===== USER MENU =====
1. Check Balance
2. Transfer Money
3. Logout
Enter your choice:
2
Enter Receiver Account Number:
6677889900
Enter Amount to Transfer:
10000
Transfer Successful!
```

```
===== USER MENU =====
1. Check Balance
2. Transfer Money
3. Logout
Enter your choice:
1
Your Current Balance: ₹40000
```