Banking System

(Technical Documentation)

Group Members:

Sufiyaan Usmani (21K – 3195)
Qasim Hasan (21K - 3210)
Ahsan Ashraf (21K – 3186)
Talha Shaikh (21K – 4564)

Project Start Date: October 21, 2021

Links:

Banking System

Version History

Documentation and User Guide

Proposal and Final Report

Header files used in the program

```
#include <stdio.h>
FUNCTIONS:
(printf(), scanf(), gets(), puts(), getchar(), fflush())
#include <stdlib.h>
FUNCTIONS:
(srand(), rand(), exit(), system())
#include <string.h>
FUNCTIONS:
(strcat(), strlen(), strcpy(), strcmp())
#include <math.h>
FUNCTIONS:
(ceil(), sqrt(), pow(), log(), exp())
#include <conio.h>
FUNCTIONS:
(getch(), clrscr())
#include <time.h>
FUNCTIONS:
(time())
#include <ctype.h>
FUNCTIONS:
(isdigit())
#include <windows.h>
FUNCTIONS:
(sleep())
```

FUNCTIONS THAT WE HAVE TAKEN FROM INTERNET:

GOTOXY:

```
void gotoxy (int x, int y) {

    COORD c;
    c.X = x;
    c.Y = y;
    SetConsoleCursorPosition(GetStdHandle(STD_OUTPUT_HANDLE), c);
}
```

We have taken this function from google its basic use is to move the cursor to a specific point (Used mainly for presentation of the program and it is done with precision by running and correcting mainly times)

```
enum Status adminPass;
enum Status accountNoGenerated;
enum Status customerID;
enum Status customerPass;
enum Status accountFound;
```

COLOUR:

```
Another code we took from the google is setcolour to put color into the program
interface and make it more interactive.

void Set Color (int ForgC)
{
    WORD wColor;
    HANDLE hStdOut = GetStdHandle(STD_OUTPUT_HANDLE);
    CONSOLE_SCREEN_BUFFER_INFO csbi;   //We use csbi for the
wAttributes word.
    if (GetConsoleScreenBufferInfo(hStdOut, &csbi)) {
    //Mask out all but the background attribute, and add in the
foreground color
        wColor = (csbi. WAttributes & 0xF0) + (ForgC & 0x0F);
        SetConsoleTextAttribute(hStdOut, wColor);
    }
}
```

CURRENT DATE AND TIME:

```
void currentDateAndTime()
{
    time_t t; // not a primitive datatype
    time(&t);

    printf("%s", ctime(&t));
}
```

Enum:

An enum is a datatype that contains fixed set of constants.
(SUCCESS, FAIL) defined beforehand in enum AS:

```
enum Status
{
    SUCCESS,
    FAIL
};
enum Status adminID;
enum Status adminPass;
enum Status accountNoGenerated;
enum Status customerID;
enum Status customerPass;
enum Status accountFound;
```

STRUCT:

A struct (or structure) is a collection of variables (can be of different types) under a single name.

```
struct CustomerInfo customer;
struct SenderInfo sender;
struct ReceiverInfo receiver;
struct AdminInfo admin;
struct Update update;
struct CurrencyInfo currency;
```

These are some of the structure are coded in which a list of variables is defined.

```
// general structure
struct CustomerInfo
{
    int acountNo;           // 6 digit
    char firstName[15];     // max 15
characters
    char lastName[15];      // max 15
characters
    int age;                // 2 digits >=
18
    char contactNumber [11]; // exactly 11
                               characters
    int accountStatus;      // 0 -
blocked,
                                1 – active
    int amount;              // max 20
digits
    char password[9];
};

struct Update
{
    int acountNo;           // 6 digit
    char firstName[15];     // max 15
characters
    char lastName[15];      // max 15
characters
    int age;                // 2 digits >=
18
    char contactNumber[11]; // exactly 11
```

```
                                characters
    int accountStatus;      // 0 - blocked,
                                1 - active
    int amount;             // max 20
digits
    char password[9];
};

/* for money transfer structures */

// sender
struct SenderInfo
{
    int acountNo;
    char firstName[15];
    char lastName[15];




    int age;
    char contactNumber[11];
    int accountStatus;
    int amount;
    char password[9];
};

// receiver
```

```
struct ReceiverInfo
{
    int acountNo;
    char firstName[15];
    char lastName[15];
    int age;
    char contactNumber[11];
    int accountStatus;
    int amount;
    char password[9];
};

struct AdminInfo
{
    int id;
    char firstName[15];
    char lastName[15];
    char password[9];
};

struct CurrencyInfo
{
    int cNo;
    char first[11];
    char last[11];
    char code[4];
    float rate;
}
```

FUNCTIONS USED IN THE PROGRAM:

```
// function prototypes
void gotoxy (int x, int y);
void currentDateAndTime ();
int mainMenu ();
void loginAsAdmin ();
void createNewAccount ();
int generateAccountNumber ();
void loginAsCustomer ();
void customerPortal ();
void depositMoney ();
int customerPortalMenu ();
void loading Animation ();
```

```
void withdraw Amount ();
void transfer Amount ();
void delete Account ();
int integerInputOnly();
void viewTransactionHistory();
void viewCurrencyRates();
void adminDeleteAccount();
void adminPortal();
int adminPortalMenu();
void viewCurrentAccInfo();
void aboutUs();
void updateCurrencyRates();
void createCustomerDataBaseBackup();
void createCustomerDataBaseBackupAnimation();
void viewMyTransactionHistory();
void searchCustomer();
void searchByAccountNumber();
void searchByName();
void bankPolicy();
void sortAsc();
void sortDes();
```

POINTERS USED:

```
FILE *fp;
FILE *temp;
FILE *transaction;
FILE *backup;
```

The **main** of the program:

```
int main()
{
    int mainMenuChoice;
    while (1)
    {
```

```
mainMenuChoice = mainMenu ();
```
here mainMenu is used to access the interface. (FUNCTION)

```
switch (mainMenuChoice)
{
case 1:
    system("cls");
    system("title Admin Login");
    loginAsAdmin();
    break;
case 2:
    system("cls");
    system("title Customer Login");
    loginAsCustomer();
    break;
case 3:
    system("cls");
    system("title Create New Account");
    createNewAccount();
    break;
case 4:
    system("cls");
    system("title Today's Currency Rates");
    viewCurrencyRates();
    break;
case 5:
    system("cls");
    system("About Us");
    aboutUs();
    break;
case 6:
    system("cls");
    gotoxy(40, 20);
    printf("Thankyou for using our service, :)");
    Sleep(2000);
    exit(0);
    break;
default:
    system("cls");
```

```
            system("title ERROR :(");
            gotoxy(0, 0);
            system("color 04");
            printf("Wrong choice entered, try again! \a");
            Sleep(1500);
            break;
        }
    }
    return 0;
}
```

Now the program will keep running with the help of while loop until default switch case or case 6 aren't entered. Here default and case 6 both are using sleep (): The sleep() function will return void which means it will not return anything and the use for it to display a code on the screen for given number of time.

Here the main menu will go back to the main body and display the interface

```
int mainMenu()
{
    int choice;
    system("color 0F");
    system("cls");
    system("title FAST NUCES BANK - MAIN MENU");
    gotoxy(0, 0);
    SetColor(10);
    currentDateAndTime();
    gotoxy(32, 3);
    SetColor(11);

printf("\xB2\xB2\xB2\xB2\xB2\xB2\xB2\xB2\xB2\xB2\xB2\xB2\xB2\xB2\xB2\xB2\xB2\xB2\xB2\xB2\xB2\xB2\xB2\xB2\xB2\xB2\xB2\xB2\xB2\xB2\xB2\xB2\xB2\xB2\xB2\xB2\xB2\xB2\xB2 FAST-NUCES BANK
\xB2\xB2\xB2\xB2\xB2\xB2\xB2\xB2\xB2\xB2\xB2\xB2\xB2\xB2\xB2\xB2\xB2\xB2\xB2\xB2\xB2\xB2\xB2\xB2\xB2\xB2\xB2\xB2\xB2\xB2\xB2\xB2\xB2\xB2\xB2\xB2\xB2\xB2\xB2");
Loop are used to produce a menu
with gotoxy
for (i = 1; i <= 7; i++)
    {
        gotoxy(31, 3 + i);
        printf("|");
    }
    for (i = 1; i <= 7; i++)
    {
        gotoxy(91, 3 + i);
        printf("|");
    }
    for (i = 1; i <= 60; i++)
    {
        gotoxy(31 + i, 11);
        printf("-");
    }
```

```
    SetColor(15);
    gotoxy(33, 5);
    printf("1. Login as Admin");

    gotoxy(33, 6);
    printf("2.Login as Customer");
    gotoxy(33, 7);
    printf("3.Create new account");
    gotoxy(33, 8);
    printf("4. Check today'currency
                          rates");
    gotoxy(33, 9);
    printf("5. About Us");
    gotoxy(33, 10);
    printf("6. Exit");
    gotoxy(32, 15);
    printf("Enter your choice: ");
    fflush(stdin);
    scanf("%d", &choice);
    return choice;
}
```

FUNCTION WHICH IS DEFINED TO SHOW THE CURRENT DATE AND TIME:

```
void currentDateAndTime ()
{
    time_t t; // not a primitive datatype
    time(&t);

    printf("%s", ctime(&t));
}
```

CASE 1 OF SWITCH(mainMenuchoice):

```
void loginAsAdmin()
{
    adminID = FAIL;
    adminPass = FAIL;
    int id;
    char password[9];
    printf("Enter ID: ");
    fflush(stdin);
    scanf(" %d", &id);
    system("cls");
    fp = fopen("admin.txt", "r");
    fseek(fp, 0, SEEK_SET);
    if (fp == NULL)
    {
        system("cls");
        perror("Error");
        exit(1);
    }
```

```
while(fscanf(fp,"%6d %15s %15s
%8s\n", &admin.id, admin.firstName,
admin.lastName,admin.password)!= EOF)
    {
    if (id == admin.id)
        {
       adminID = SUCCESS;
       printf("Enter password: ");
    for (i = 0; i <= 7;)
        {
        ch = getch();
if ((ch >= 'a' && ch <= 'z') ||
    (ch >= 'A' && ch <= 'Z') ||
    (ch >= '0' && ch <= '9') ||
    (ch >= 33 && ch <= 47))
        {
        password[i] = ch;
        ch = '*';
        printf("%c", ch);
```

```
        i++;
                }
        }
     password[8] = '\0';
if((strcmp(password,admin.password))
== 0)
     {
      adminPass = SUCCESS;
      system("cls");
      fclose(fp);
      loadingAnimation();
      adminPortal();
      break;
     }
   }
 }
Fclose(fp);
    if (adminID == FAIL)
    {
 system("cls");
 printf("This ID does not exists, try
                          again \a");
        Sleep(1500);
        system("cls");
        loginAsAdmin();
    }
    if (adminPass == FAIL)
    {
        system("cls");
        printf("Wrong Password, try
                          again \a");
        Sleep(1500);
        system("cls");
        loginAsAdmin();
    }
}
```

Explanation:

Firstly, when login in as admin adminID will reset then enter the ID of the admin then check the ID in the file(admin.txt) Then scan the entire file until EOF. If the id exists, then admin will enter his password in an array. If password matches the password saved in admin file, then give excess to **adminPortal ().** If admin password or admin id is not active or is not correct then recall the same function **loginAsAdmin()** keep entering admin id and password until it matches thein the admin file .txt
when both condition are fulfilled close the admin .txt file.

```
void adminPortal()
{
 system("cls");
int adminPortalChoice, id = admin.id;

 while (1)
    {
fp = fopen("admin.txt", "r");
fseek(fp, 0, SEEK_SET);
while (fscanf(fp, "%6d %15s %15s
%8s\n", &admin.id, admin.firstName,
admin.lastName, admin.password) !=
EOF)
    {
    if (id = admin.id)
    {
    break;
    }
    }
    fclose(fp);
adminPortalChoice =adminPortalMenu();
        switch (adminPortalChoice)
        {
--        case 1:
            system("color 0B");
            system("cls");
printf("Name:%s%s\n",admin.firstName,
admin.lastName);
printf("ID: %d\n", admin.id);
```

```
printf("\nPress any key to go to your
portal\n");
            getch();
            break;
        case 2:
            system("cls");
            viewCurrentAccInfo();
            break;
        case 3:
            system("cls");
            viewTransactionHistory();
            break;
        case 4:
            system("cls");
            adminDeleteAccount();
            break;
        case 5:

            system("cls");
            updateCurrencyRates();
            break;
        case 6:
            system("cls");

createCustomerDataBaseBackupAnimation
();
createCustomerDataBaseBackup();
            break;
        case 7:
            main();
            break;
        default:
            system("cls");
            gotoxy(0, 0);
            system("color 04");
printf("Wrong choice entered, try
                    again! \a");
            Sleep(1500);
            break;

        }
    }
```

```
int adminPortalMenu()
{
    int choice;
    system("color 0F");
```

```
}
```

Explanation:

Now after admin has successfully entered his id and password now he will have access to different functions. Now after entering admin.txt will again be called to check which admin id is active at the time and then the file will close. After that the adminPortalMenu() will run explained in the next page:

Case 1: will lead to the active admin information
Case 2: will open account info of the customer
Case 3: will open the transaction history of the customers
Case 4: will give admin the power to delete customer accounts
All of these cases are explained in the further text.
Case 5: will update the currencies rate.
Case 6: will allow admin to search and sort customers.
Case 7: backup of database
Case 8: go back to the main interface
All of these cases code and explanation are given further below the text.

```
    system("cls");
    system("title ADMIN PORTAL");
    gotoxy(0, 0);
    currentDateAndTime();
    gotoxy(0, 2);
    printf("Welcome %s %s\n",
admin.firstName, admin.lastName);
    gotoxy(32, 5);

printf("\xB2\xB2\xB2\xB2\xB2\xB2\xB2\
xB2\xB2\xB2\xB2\xB2\xB2\xB2\xB2\xB2\x
B2\xB2\xB2\xB2\xB2 FAST-NUCES BANK
\xB2\xB2\xB2\xB2\xB2\xB2\xB2\xB2\xB2\
xB2\xB2\xB2\xB2\xB2\xB2\xB2\xB2\xB2\x
B2\xB2\xB2");
    for (i = 1; i <= 12; i++)
    {
        gotoxy(31, 5 + i);
        printf("|");
    }
    for (i = 1; i <= 12; i++)
    {
        gotoxy(91, 5 + i);
        printf("|");
    }
    for (i = 1; i <= 60; i++)
    {
     gotoxy(31 + i, 18);
     printf("-");
    }
    gotoxy(33, 7);
    printf("1. View my information");
    gotoxy(33, 9);
    printf("2. View current accounts
                        information");
    gotoxy(33, 11);
    printf("3. View Transaction
                        History");
    gotoxy(33, 13);
    printf("4. Delete Account");
    gotoxy(33, 15);
 printf("5. Update Currency Rates");
    gotoxy(33, 17);
printf("6. Create Customer Database
Backup");
    gotoxy(33, 19);
    printf("7. Logout");
    gotoxy(32, 21);
    printf("Enter your choice: ");
    fflush(stdin);
    scanf(" %d", &choice);
    return choice;

}
```

Explanation:

Now again we will use gotoxy to code the interface of admin menu same as we did in the bank interface and return the chosen option.

```
void viewCurrentAccInfo()
{
    fp = fopen("customer.txt", "r");
    if (fp == NULL)
    {
        perror("Error");
    }
    fseek(fp, 0, SEEK_SET);
```

```
    system("cls");
printf("Account Number  First Name   Last Name  Age  Status   Amount     \n\n");
while (fscanf(fp, "%6d  %15s %15s %2d %11s %d %9d %8s\n", &update.acountNo,
update.firstName, update.lastName, &update.age, update.contactNumber,
&update.accountStatus, &update.amount, update.password) != EOF)
    {
printf("%-14d  %-15s %-15s %-3d %6s   %-9d\n", update.acountNo, update.firstName,
update.lastName, update.age, update.accountStatus == 0 ? "Active" : "Blocked",
update.amount);
    }
    printf("\n\nPress any key to continue ");
    getch();
    fclose(fp);
}
```

Explanation:

Admin has the access to the customer info so we will then open the customer.txt file where all the information is being stored detail codes and explanation is in case 2 of (mainMenuchoice) and then all customer's basic information will be printed as we will print each line of the file by loop until the end of file.

```
void viewTransactionHistory()
{
    char transactionType[10];
    transaction = fopen("transaction_history.txt", "r");
    fseek(fp, 0, SEEK_SET);
    system("cls");
printf("Account Number    First Name     Last Name    Amount     Type    \n\n");

while (fscanf(transaction, "%6d %15s %15s %9d %s\n", &update.acountNo,
update.firstName, update.lastName, &update.amount, transactionType) != EOF)
    {
printf("%14d %15s %15s %9d %s\n", update.acountNo, update.firstName,
update.lastName, update.amount, transactionType);
    }
printf("\n\nPress any key to continue ");
getch();
    fclose(transaction);
}
```

Explanation:

Admin also have access to all the transactions of the customers for this we will open transaction.txt file and print all the transaction with the help of loop till the end of file.

```
void adminDeleteAccount()
{
    int accountNoToDelete, accountFound = 0;
    system("cls");
    printf("Enter account number to delete: ");
    fflush(stdin);
    scanf("%d", &accountNoToDelete);
fp = fopen("customer.txt", "r");
 while (fscanf(fp, "%6d %15s %15s %2d %11s %d %9d %8s\n", &update.acountNo,
update.firstName, update.lastName, &update.age, update.contactNumber,
&update.accountStatus, &update.amount, update.password) != EOF)
    {
        if (update.acountNo == accountNoToDelete)
        {
            accountFound = 1;
            break;
        }
    }
    fclose(fp);
    if (accountFound == 0)
    {
        system("cls");
        SetColor(4);
        printf("\aAccount does not exists");
        Sleep(1000);
        system("color 0F");
        goto doNothing;
    }
    if (update.amount >= 0)
    {
        printf("Are you sure you want to delete this account? [y,n]: ");
        fflush(stdin);
        ch = getche();
        if (ch == 'y' || ch == 'Y')
        {
            fp = fopen("customer.txt", "r");
            temp = fopen("temp.txt", "w");
            if (fp == NULL || temp == NULL)
```

```
            {
                perror("Error");
                Sleep(1000);
            }
            fseek(fp, 0, SEEK_SET);
            while (fscanf(fp, "%6d %15s %15s %2d %11s %d %9d %8s\n",
&update.acountNo, update.firstName, update.lastName, &update.age,
update.contactNumber, &update.accountStatus, &update.amount, update.password) !=
EOF)
            {
                if (update.acountNo != accountNoToDelete)
                {
                    fprintf(temp, "%-6d %-15s %-15s %-2d %-11s %-d %-9d %-8s\n",
update.acountNo, update.firstName, update.lastName, update.age,
update.contactNumber, update.accountStatus, update.amount, update.password);
                }
            }
            fclose(fp);
            fclose(temp);
            remove("customer.txt");
            rename("temp.txt", "customer.txt");
            system("cls");
            Sleep(2000);
            adminPortal();
        }
        else if (ch == 'n' || ch == 'N')
        {
            system("cls");
            adminPortal();
        }
        else
        {
            printf("Wrong choice entered, please enter a valid choice");
            system("cls");
            Sleep(1000);
            adminDeleteAccount();
        }
    }
    else
    {
        printf("You can not delete you account as you have negative balance\nPay
the balance first to delete your account\n");
    }
doNothing:
    printf("Loading");
```

```
    system("cls");
}
```

Explanation:

In the admin portal, when admin choose to delete an account, he enters the account number he wants to delete. When he enters the account number, the program checks whether there exists an account holding entered account number. If account is not found, then a message is printed that “account does not exist”. If account exists, then admin is asked if he really wants to delete the account. If he presses “no”, then the program is directed to admin portal. If he presses “yes”, then the customer file is read by program, when the entered account number appears, it is omitted, and rest of the data is written on a temp file. Once all the data is written the temp file is renamed to “customer.txt” file and the previous file is deleted.

Admin does not have the authority to delete his or another admin’s account.

```
void updateCurrencyRates()
{
    int cNoToUpdate, isFound;
    float newRate;
    char ch;
    cNoToUpdate = 0;
    isFound = 0;
    viewCurrencyRates();

    system("cls");
    printf("Enter currency number to update: ");
    fflush(stdin);
    while (1)
    {
        ch = getch();
        if (ch >= '0' && ch <= '9')
        {
            printf("%c", ch);
            cNoToUpdate = (cNoToUpdate * 10) + (ch - 48);
        }
        else if (ch == 13)
        {
            break;
        }
    }
```

```
    fp = fopen("currency_rates.txt", "a+");
    while (fscanf(fp, "%2d %10s %10s %3s %6f\n", &currency.cNo, currency.first,
currency.last, currency.code, &currency.rate) != EOF)
    {
        if (cNoToUpdate == currency.cNo)
        {
            isFound = 1;
            break;
        }
    }
    fclose(fp);
    system("cls");
    if (isFound == 1)
    {
        while (1)
        {
            printf("Enter new rates: ");
            fflush(stdin);
            scanf("%f", &newRate);
            if (newRate > 0)
            {
                break;
            }
            else
            {
                system("cls");
                printf("Error: Rates can not be negative");
                Sleep(1000);
                system("cls");
            }
        }
        fp = fopen("currency_rates.txt", "r");
        temp = fopen("currencyTemp.txt", "w");
        while (fscanf(fp, "%2d %10s %10s %3s %6f\n", &currency.cNo,
currency.first, currency.last, currency.code, &currency.rate) != EOF)
        {
            if (currency.cNo == cNoToUpdate)
            {
                fprintf(temp, "%-2d %-10s %-10s %-3s %-6.2f\n", currency.cNo,
currency.first, currency.last, currency.code, newRate);
            }
            else
            {
                fprintf(temp, "%-2d %-10s %-10s %-3s %-6.2f\n", currency.cNo,
currency.first, currency.last, currency.code, currency.rate);
```

```
            }
        }
        fclose(fp);
        fclose(temp);
        remove("currency_rates.txt");
        rename("currencyTemp.txt", "currency_rates.txt");
    }
    else
    {
        printf("Error: Currency not found\n\nRedirecting");
        Sleep(500);
    }
}
```

Explanation:

Due to this function, admin can update currency rates. He selects the currency number; he wants to update. After this, the currency.txt file is opened, and it is checked that the entered number exists or not. The file is then closed, then admin enters the updated amount of that currency and again after the same file reading, writing, and deleting process is done and currency is updated.

void searchCustomer()

```
{
    int searchChoice;
    printf("1. Search by Account Number\n");
    printf("2. Search by Name\n");
    printf("3. Sort by amount (ASCENDING)\n");
    printf("4. Sort by amount (DESCENDING)\n");
    printf("5. Go back\n\n");
    SetColor(10);
    printf("Enter your choice: ");
    fflush(stdin);
    SetColor(15);
    scanf("%d", &searchChoice);

    switch (searchChoice)
    {
    case 1:
        searchByAccountNumber();
        break;
    case 2:
```

```
        searchByName();
        break;
    case 3:
        sortAsc();
        break;
    case 4:
        sortDes();
        break;
    case 5:
        goto searchCustomerEnd;
        break;
    default:
        printf("Error, wrong choice entered");
        Sleep(1000);
        break;
    }
    searchCustomerEnd:
        system("cls");
}
```

Explanation:

When admin presses 6 to sort and search customers. The menu appears which gives users multiple options of sorting and searching. Each of them is listed and explained below.

```
void searchByAccountNumber()
{
    int accNo, flag;
    flag = 0;
    system("cls");
    while (1)
    {
        printf("Enter account number (-1 to go back) : ");
        fflush(stdin);
        scanf("%d", &accNo);
        if (accNo != -1)
        {
            if (accNo >= 100000 && accNo <= 999999)
            {
                fp = fopen("customer.txt", "r");
                while (fscanf(fp, "%6d %15s %15s %2d %11s %d %9d %8s\n",
&update.acountNo, update.firstName, update.lastName, &update.age,
```

```
update.contactNumber, &update.accountStatus, &update.amount, update.password) !=
EOF)
                {
                    if (accNo == update.acountNo)
                    {
                        flag = 1;
                        system("color 5F");
                        system("cls");
                        printf("Name            : ");
                        printf("%s %s\n", update.firstName, update.lastName);
                        printf("Account Number  : %d\n", update.acountNo);
                        printf("Age             : %d\n", update.age);
                        printf("Contact Number  : %s\n", update.contactNumber);
                        printf("Account Status  : %s\n", (update.accountStatus ==
0 ? "Active" : "Blocked"));
                        printf("Account Balance : %d\n", update.amount);
                        printf("\n\nPress any key to go to your portal\n");
                        getch();
                        system("color 0F");
                        goto searchEnd;
                    }
                }
                if (flag == 0)
                {
                    system("cls");
                    printf("This account does not exists\n");
                    Sleep(2000);
                    system("cls");
                }
            }
            else
            {
                system("cls");
                printf("Error, account number must be of 6 digits");
                Sleep(2000);
                system("cls");
            }
        }
        else
        {
            goto searchEnd;
        }
    }
searchEnd:
    fclose(fp);
```

```
    system("cls");
}
```

Explanation:

After user selects the option to search account information by Account Number, the admin enters the account number he wants to search, this account number is then checked whether it is a valid account number or not, then the customer.txt file is opened, and it is read until the entered account number is found, when entered account number is found the program prints all the information related to that account number.

```
void searchByName()
{
    int flag = 0, count = 0;
    char name[30], fullName[30];
    system("cls");
        printf("Enter full name: ");
        fflush(stdin);
        gets(name);
        strupr(name);
        system("cls");
        fp = fopen("customer.txt", "r");
        while (fscanf(fp, "%6d %15s %15s %2d %11s %d %9d %8s\n",
&update.acountNo, update.firstName, update.lastName, &update.age,
update.contactNumber, &update.accountStatus, &update.amount, update.password) !=
EOF)
        {
            strcpy(fullName, update.firstName);
            strcat(fullName, " ");
            strcat(fullName, update.lastName);
            strupr(fullName);
            if (strcmp(fullName, name) == 0)
            {
                flag = 1;
                system("color 5F");
                printf("Name            : ");
                printf("%s %s\n", update.firstName, update.lastName);
                printf("Account Number  : %d\n", update.acountNo);
                printf("Age             : %d\n", update.age);
```

```
                printf("Contact Number  : %s\n", update.contactNumber);
                printf("Account Status  : %s\n", (update.accountStatus == 0 ?
"Active" : "Blocked"));
                printf("Account Balance : %d\n\n\n\n", update.amount);
                count++;
            }
        }
        if (flag == 1)
        {
            printf("There are currently %d account(s) with this name\n", count);
            printf("Press any key to continue");
            getch();
        }
        else
        {
            printf("This name does not exists");
            Sleep(2000);
        }
    fclose(fp);
    system("cls");
}
```

Explanation:

This function searches the account by using name of the account holder. The algorithm is that the admin first enters the full name of the account holder, then customer.txt file is opened and read, when the name appears in the file, all the data related to that name is printed.

This function has the ability to print the data of multiple accounts having same name.

```
void sortAsc(){
    int size = 0, i = 0, round;
    fp = fopen("customer.txt", "r");
    while(fscanf(fp, "%6d %15s %15s %2d %11s %d %9d %8s\n", &update.acountNo,
update.firstName, update.lastName, &update.age, update.contactNumber,
&update.accountStatus, &update.amount, update.password) != EOF){
        size++;
    }
```

```
    fclose(fp);
    system("color F1");
    struct CustomerInfo sort[size], temp1;

    fp = fopen("customer.txt", "r");
    i = 0;
    while(fscanf(fp, "%6d %15s %15s %2d %11s %d %9d %8s\n", &temp1.acountNo,
temp1.firstName, temp1.lastName, &temp1.age, temp1.contactNumber,
&temp1.accountStatus, &temp1.amount, temp1.password) != EOF){
        sort[i] = temp1;
        i++;
    }

    for(round=1;round<size;round++){
        for(i=0;i<size-round;i++){
            if(sort[i].amount > sort[i+1].amount){
                temp1 = sort[i];
                sort[i] = sort[i+1];
                sort[i+1] = temp1;
            }
        }
    }
    fclose(fp);
    system("cls");
    SetColor(1);
    printf("Account Number  First Name      Last Name      Age  Status   Amount
\n\n");
    SetColor(0);
    i = 0;
    for(i=0;i<size;i++)
    {
        printf("%-14d  %-15s %-15s %-3d %6s   %-9d\n", sort[i].acountNo,
sort[i].firstName, sort[i].lastName, sort[i].age, sort[i].accountStatus == 0 ?
"Active" : "Blocked", sort[i].amount);

    }
    printf("\nPress any key to continue...");
    getch();
    system("cls");
}
```

Explanation:

This function just sorts the customer data in ascending order with respect to amount in the customer's account. For this **Bubble Sort** is used.

```
void sortDes(){
    int size = 0, i = 0, round;
    fp = fopen("customer.txt", "r");
    while(fscanf(fp, "%6d %15s %15s %2d %11s %d %9d %8s\n", &update.acountNo,
update.firstName, update.lastName, &update.age, update.contactNumber,
&update.accountStatus, &update.amount, update.password) != EOF){
        size++;
    }
    fclose(fp);
    system("color F1");
    struct CustomerInfo sort[size], temp1;

    fp = fopen("customer.txt", "r");
    i = 0;
    while(fscanf(fp, "%6d %15s %15s %2d %11s %d %9d %8s\n", &temp1.acountNo,
temp1.firstName, temp1.lastName, &temp1.age, temp1.contactNumber,
&temp1.accountStatus, &temp1.amount, temp1.password) != EOF){
        sort[i] = temp1;
        i++;
    }

    for(round=1;round<size;round++){
        for(i=0;i<size-round;i++){
            if(sort[i].amount < sort[i+1].amount){
                temp1 = sort[i];
                sort[i] = sort[i+1];
                sort[i+1] = temp1;
            }
        }
    }
    fclose(fp);
    system("cls");
    SetColor(1);
    printf("Account Number  First Name      Last Name      Age  Status   Amount
\n\n");
    SetColor(0);
    i = 0;
    for(i=0;i<size;i++)
    {
```

```
        printf("%-14d  %-15s %-15s %-3d %6s   %-9d\n", sort[i].acountNo,
sort[i].firstName, sort[i].lastName, sort[i].age, sort[i].accountStatus == 0 ?
"Active": "Blocked", sort[i].amount);

    }
    printf("\nPress any key to continue...");
    getch();
    system("cls");
}
```

Explanation:

This function just sorts the customer data in ascending order with respect to amount in the customer's account. For this **Bubble Sort** is used.

```
Void
createCustomerDataBaseBackup
(){
  char fname[16];
  char date[50];
  char fileName[29] = "./backup/";

  time_t t; // not a primitive
datatype
  time(&t);

  strcpy(date, ctime(&t));
  fname[0] = date[0];
  fname[1] = date[1];
  fname[2] = date[2];
  fname[3] = ' ';
  fname[4] = date[4];
  fname[5] = date[5];
  fname[6] = date[6];
  fname[7] = ' ';
  fname[8] = date[8];
  fname[9] = date[9];
  fname[10] = ' ';
  fname[11] = date[20];
  fname[12] = date[21];
  fname[13] = date[22];
  fname[14] = date[23];
  fname[15] = '\0';

  strcat(fileName, fname);
  strcat(fileName, ".txt");
  backup = fopen(fileName,"w");
  fp = fopen("customer.txt", "r");
  while(fscanf(fp, "%6d %15s %15s %2d
%11s %d %9d %8s\n", &update.acountNo,
update.firstName, update.lastName,
&update.age, update.contactNumber,
&update.accountStatus,
&update.amount, update.password) !=
EOF){
      fprintf(backup, "%-6d %-15s %-
15s %-2d %-11s %-d %-9d %-8s\n",
update.acountNo, update.firstName,
update.lastName, update.age,
update.contactNumber,
update.accountStatus, update.amount,
update.password);
  }
  fclose(fp);
  fclose(backup);
}
```

```
void
createCustomerDataBaseBackupAnimation
(){
    int i;
    system("cls");
    system("color 09");
    gotoxy(50, 10);
    printf("Creating Backup");
    gotoxy(40, 12);
    printf("[");
    gotoxy(78, 12);
    printf("]");
    gotoxy(41, 12);
    for (i = 1; i <= 37; i++)
    {
        printf("%c", 177);
    }
    gotoxy(41, 12);
    for (i = 1; i <= 37; i++)
    {
        Sleep(35);
        printf("%c", 219);
    }
    gotoxy(0, 0);
    system("cls");
    system("color 0A");
    printf("Backup Created
Successfully");
    Sleep(2000);
    system("cls");
}
```

CASE 2 OF SWITCH(mainMenuchoice):

```
void loginAsCustomer()
{
    int backSpaceCount = 0;
    customerID = FAIL;
    customerPass = FAIL;
    int accNo = 0;
    char password[9];
    printf("Enter Account Number: ");
    SetColor(11);
    fflush(stdin);
    scanf("%d", &accNo);
    system("cls");
    SetColor(15);
    fp = fopen("customer.txt", "r");
    fseek(fp, 0, SEEK_SET);
    if (fp == NULL)
    {
        system("cls");
        perror("Error");
        exit(1);
    }
    while (fscanf(fp, "%6d %15s %15s
%2d %11s %d %9d %8s\n",
&customer.acountNo,
customer.firstName,
customer.lastName, &customer.age,
customer.contactNumber,
&customer.accountStatus,
&customer.amount, customer.password)
!= EOF)
    {
        if(accNo ==customer.acountNo)
        {
            customerID = SUCCESS;
```

```
            system("color 0F");
            printf("Enter password: ");
            SetColor(11);
            for (i=0;i<8;)
            {
                ch = getch();
                if ((ch >= 'a' && ch <= 'z') || (ch >= 'A' && ch <= 'Z') || (ch >= '0' && ch <= '9') || (ch >= 33 && ch <= 47))
                {
                    password[i] = ch;
                    ch = '*';
                    printf("%c", ch);
                    i++;
                    backSpaceCount++;
                }
                else if (ch == 8 && backSpaceCount > 0)
                {
                    printf("\b \b");
                    backSpaceCount--;
                    i--;
                }
            }
            password[8] = '\0';
            if((strcmp(password, customer.password)) == 0)
            {
                customerPass = SUCCESS;
                system("cls");
                fclose(fp);
                loadingAnimation();
                customerPortal();
                break;
            }
        }
    }
    fclose(fp);
    if (customerID == FAIL)
    {
        system("cls");
        printf("This ID does not exists, try again \a");
        sleep(1500);
        system("cls");
        loginAsCustomer();
    }
    if (customerPass == FAIL)
    {
        system("cls");
        printf("Error: Wrong Password, retry");
        sleep(1500);
        system("cls");
        loginAsCustomer();
    }
}
```

Explanation:

Firstly, when login in as customer

customerID will reset then enter the ID of the customer then check the ID in the file(customer.txt) Then scan the entire file until EOF. If the id exists, then customer will enter his password in an array. If password matches the password saved in customer file, then give excess to **customerPortal ().** If customer password or customer id is not active or is not correct then recall the same function **loginAsCustomer()** keep entering customer id and password until it matches the id and password in the customer file .txt
when both conditions are fulfilled close the customer.txt file.

```
void customerPortal()
{
    system("cls");
    int customerPortalChoice, acc_No = customer.acountNo;

    while (1)
    {
        fp = fopen("customer.txt", "r");
        fseek(fp, 0, SEEK_SET);
        while (fscanf(fp, "%6d %15s %15s %2d %11s %d %9d %8s\n", &customer.acountNo, customer.firstName, customer.lastName, &customer.age, customer.contactNumber, &customer.accountStatus, &customer.amount, customer.password) != EOF)
        {

if (acc_No == customer.acountNo)
{
   break;
 }
        }
        fclose(fp);
        customerPortalChoice = customerPortalMenu();
        switch (customerPortalChoice)
        {
        case 1:

            system("color 0B");
            system("cls");
            system("title MY INFO");
            printf("Name: ");
            printf("%s %s\n", customer.firstName, customer.lastName);
            printf("Account Number  : %d\n", customer.acountNo);
            printf("Age             : %d\n", customer.age);
            printf("Contact Number  : %s\n", customer.contactNumber);
            printf("Account Status  : %s\n", (customer.accountStatus == 0 ? "Active" : "Blocked"));
            printf("Account Balance : %d\n", customer.amount);

            printf("\nPress any key to go to your portal\n");
            getch();
            system("color 0F");
            // customerPortal();
            break;
        case 2:
            system("cls");
            system("title DEPOSIT AMOUNT");
            depositMoney();
            break;
        case 3:
            system("cls");
            system("title WITHDRAW AMOUNT");
            withdrawAmount();
```

```
            break;
        case 4:
            system("cls");
            system("title TRANSFER
AMOUNT");
            transferAmount();
            break;
        case 5:
            system("cls");
            system("title VIEW
TRANSACTION HISTORY");

viewMyTransactionHistory();
            break;
        case 6:
            system("cls");
            system("title DELETE
ACCOUNT");
            deleteAccount();
            break;
        case 7:
            main();
            break;
        default:
            system("cls");
            system("title ERROR");
            gotoxy(0, 0);
            system("color 4F");
            printf("\aWrong choice
entered, try again! \a");
            Sleep(1500);
            system("color 0F");
            break;
        }
    }
}
```

Explanation:

Now after customer has successfully entered his id and password now he will have access to different functions. Now after entering customer.txt will again be called to check which customer id is active at the time and then the file will close. After that the customerPortalMenu() will run explained in the next page:

Case 1: will lead to the active customer information.

Case 2: will lead to depositMoney() to deposit money in the account.

Case 3: will lead to withdrawAmount() to withdraw money from his account.

Case 4: will lead to transferAmount() to transfer money to another account.

Case 5: will lead to viewMyTransactionHistory() to see his transaction history.

Case 6: will lead to deleteAccount() to delete account.

Case 7: go back to the main interface.
All of these cases code and explanation are given further below the text.

int **customerPortalMenu()**

{

int choice;

```
    system("color 0F");
    system("cls");
    system("title MY PORTAL");
    gotoxy(0, 0);
    SetColor(10);
    currentDateAndTime();
    gotoxy(0, 2);
    SetColor(9);
    printf("Welcome, Mr %s %s\n",
customer.firstName, customer.lastName);
    printf("Current Balance: Rs. %d",
customer.amount);
    gotoxy(32, 5);
    SetColor(11);

printf("\xB2\xB2\xB2\xB2\xB2\xB2\xB2\xB2\xB2\
xB2\xB2\xB2\xB2\xB2\xB2\xB2\xB2\xB2\xB2\xB2\
xB2 FAST-NUCES BANK
\xB2\xB2\xB2\xB2\xB2\xB2\xB2\xB2\xB2\xB2\xB2
\xB2\xB2\xB2\xB2\xB2\xB2\xB2\xB2\xB2\xB2");
    for (i = 1; i <= 14; i++)
    {
        gotoxy(31, 5 + i);
        printf("|");
    }
    for (i = 1; i <= 14; i++)
    {
        gotoxy(91, 5 + i);
        printf("|");
    }
    for (i = 1; i <= 60; i++)
    {
        gotoxy(31 + i, 20);
        printf("-");
    }
    SetColor(15);
    gotoxy(33, 7);
    printf("1. View my information");
    gotoxy(33, 9);
    printf("2. Deposit Money");
    gotoxy(33, 11);
    printf("3. Withdraw Money");
    gotoxy(33, 13);
    printf("4. Transfer Money");
    gotoxy(33, 15);
    printf("5. View My Transaction History");
    gotoxy(33, 17);
    printf("6. Delete Account");
    gotoxy(33, 19);
    printf("7. Logout");
    gotoxy(32, 22);
    SetColor(5);
    printf("Enter your choice: ");
    fflush(stdin);
    scanf("%d", &choice);
    system("color 0F");
    return choice;
}
```

Explanation:

Now again we will use gotoxy() to code the interface of admin menu same as we did in the bank interface and return the chosen option.

void **depositMoney()**

```
{
   int amountToDeposit = 0, newAmount, tempAcc;
   tempAcc = customer.acountNo;
   system("cls");
   do
   {
      system("cls");
      printf("Enter amount to deposit(-1 to go back): Rs. ");
      fflush(stdin);
      scanf(" %d", &amountToDeposit);
      if ((customer.amount + amountToDeposit) > 999999999)
      {
         system("cls");
         printf("Error: Account Limit reached. Enter a smaller amount");
         Sleep(1000);
      }
      else if (amountToDeposit < 0 && amountToDeposit != -1)
      {
         system("cls");
         printf("Error: Amount can not be negative");
         Sleep(1000);
      }
      else if (amountToDeposit == -1)
      {
         system("cls");
         goto depositMoneyEnd;
      }
   } while ((customer.amount + amountToDeposit > 999999999) || amountToDeposit < 0);
   fp = fopen("customer.txt", "r");
   temp = fopen("customerTemp.txt", "w");
   transaction = fopen("transaction_history.txt", "a");
   if (fp == NULL)
   {
      perror("Error");
      exit(1);
   }
   if (temp == NULL)
   {
      perror("Error");
      exit(1);
   }
   newAmount = customer.amount + amountToDeposit;
   fseek(fp, 0, SEEK_SET);
   while (fscanf(fp, "%6d %15s %15s %2d %11s %d %9d %8s\n", &update.acountNo, update.firstName, update.lastName, &update.age, update.contactNumber,
```

```
&update.accountStatus, &update.amount,
update.password) != EOF)
  {
    if (tempAcc == update.acountNo)
    {
      fprintf(temp, "%-6d %-15s %-15s %-2d %-11s %-d %-9d %-8s\n", update.acountNo, update.firstName, update.lastName, update.age, update.contactNumber, update.accountStatus, newAmount, update.password);
      fprintf(transaction, "%-6d %-15s %-15s %-9d Deposit\n", update.acountNo, update.firstName, update.lastName, amountToDeposit);
    }
    else
    {
      fprintf(temp, "%-6d %-15s %-15s %-2d %-11s %-d %-9d %-8s\n", update.acountNo, update.firstName, update.lastName, update.age, update.contactNumber, update.accountStatus, update.amount, update.password);}}
  system("cls");
  printf("Rs. %d deposited successfully in your account\n", amountToDeposit);
  Sleep(2000);
  fclose(temp);
  fclose(fp);
  fclose(transaction);
  remove("customer.txt");
  rename("customerTemp.txt", "customer.txt");
depositMoneyEnd:
  system("cls");
}
```

Explanation:

Firstly, user will enter his account number and password, if he logins successfully then he is given the option to deposit money in his account. When he enters an amount, the entered amount is summed up with the existing amount, if the summed amount does not exceed the limit of the account, then money is successfully deposited in his account. Negative amounts will not be accepted. -1 is the only value which will be accepted, and it redirects the customer to customer portal.

void **withdrawAmount()**

```
{
  int amountToWithdraw = 0, newAmount, tempAcc;
  tempAcc = customer.acountNo;
  system("cls");
  do
  {
    system("cls");
    printf("Enter amount to withdraw(-1 to go back): Rs. ");
    fflush(stdin);
    scanf(" %d", &amountToWithdraw);
    if (amountToWithdraw > customer.amount)
    {
      system("cls");
      // add red color here
      printf("Error: Not enough account balance");
```

```
        Sleep(1000);
    }
    else if (amountToWithdraw < 0 &&
amountToWithdraw != -1)
    {
        system("cls");
        printf("Error: Amount cannot be negative");
        Sleep(1000);
    }
    else if (amountToWithdraw == -1)
    {
        goto withdrawAmountEnd;
    }
  } while (amountToWithdraw >
customer.amount || amountToWithdraw < 0);
  fp = fopen("customer.txt", "r");
  temp = fopen("customerTemp.txt", "w");
  transaction = fopen("transaction_history.txt",
"a");
  if (fp == NULL)
  {
    perror("Error");
    exit(1);
  }
  if (temp == NULL)
  {
    perror("Error");
    exit(1);
  }
  newAmount = customer.amount -
amountToWithdraw;
  fseek(fp, 0, SEEK_SET);
  while (fscanf(fp, "%6d %15s %15s %2d %11s %d
%9d %8s\n", &update.acountNo,
update.firstName, update.lastName,
&update.age, update.contactNumber,
&update.accountStatus, &update.amount,
update.password) != EOF)
  {
    if (tempAc == update.acountNo)
    {
      fprintf(temp, "%-6d %-15s %-15s %-2d %-
11s %-d %-9d %-8s\n", update.acountNo,
update.firstName, update.lastName, update.age,
update.contactNumber, update.accountStatus,
newAmount, update.password);
      fprintf(transaction, "%-6d %-15s %-15s %-
9d Withdraw\n", update.acountNo,
update.firstName, update.lastName,
amountToWithdraw);
    }
    else
    {
      fprintf(temp, "%-6d %-15s %-15s %-2d %-
11s %-d %-9d %-8s\n", update.acountNo,
update.firstName, update.lastName, update.age,
update.contactNumber, update.accountStatus,
update.amount, update.password);
    }
  }
  system("cls");
  printf("Rs. %d withdrawn successfully from your
account\n", amountToWithdraw);
  Sleep(2000);
  fclose(temp);
  fclose(fp);
  fclose(transaction);
```

```
    remove("customer.txt");
    // if (remove("customer.txt") == 0)
    //     printf("Success\n");
    // else
    //     perror("Unable to delete the file");
    rename("customerTemp.txt", "customer.txt");
withdrawAmountEnd:
    system("cls");
}
```

Explanation:

Firstly, user will enter his account number and password, if he logins successfully then he is given the option to withdraw money from his account. When he enters an amount, the entered amount is checked against his enough balance, if there is enough balance in his account, then the entered amount is subtracted from the balance and the amount is withdrawn. If the user does not have enough balance, then the message pops up that there is not enough balance in your account. Negative amounts will not be accepted. -1 is the only value which will be accepted, and it redirects the customer-to-customer portal.

void transferAmount()

```
{
    int accNoReceiver, amountToTransfer;
    accountFound = FAIL;
    sender.acountNo = customer.acountNo;
    strcpy(sender.firstName, customer.firstName);
    strcpy(sender.lastName, customer.lastName);
    sender.age = customer.age;
    strcpy(sender.contactNumber,
customer.contactNumber);
    sender.accountStatus =
customer.accountStatus;
    sender.amount = customer.amount;
    strcpy(sender.password, customer.password);
    system("cls");
    while (1)
    {
        printf("Enter receiver's account number (-1 to
go back): ");
        fflush(stdin);
        scanf("%d", &accNoReceiver);
        if (accNoReceiver == -1)
        {
            goto transferAmountEnd;
        }
        else if (accNoReceiver != sender.acountNo)
        {
            break;
        }
        else
```

```
    {
        system("cls");
        system("color 04");
        printf("\aError! You cannot transfer to
yourself, please enter a valid account no\n");
        Sleep(1000);
        system("cls");
        system("color 0F");
    }
  }
  fp = fopen("customer.txt", "a+");
  fseek(fp, 0, SEEK_SET);
  while (fscanf(fp, "%6d %15s %15s %2d %11s %d
%9d %8s\n", &receiver.acountNo,
receiver.firstName, receiver.lastName,
&receiver.age, receiver.contactNumber,
&receiver.accountStatus, &receiver.amount,
receiver.password) != EOF)
  {
    if (accNoReceiver == receiver.acountNo)
    {
      accountFound = SUCCESS;
      break;
    }
    else
    {
      accountFound = FAIL;
    }
  }

  fseek(fp, 0, SEEK_SET);
  if (accountFound == SUCCESS)
  {
    printf("Enter amount to transfer: ");
    fflush(stdin);
    scanf("%d", &amountToTransfer);
    if (sender.amount >= amountToTransfer)
    {
      if (receiver.amount + amountToTransfer <=
999999999)
      {
        temp = fopen("customerTemp.txt", "w");
        transaction =
fopen("transaction_history.txt", "a");
        fprintf(transaction, "%-6d %-15s %-15s
%-9d Transfer\n", sender.acountNo,
sender.firstName, sender.lastName,
amountToTransfer);
        while (fscanf(fp, "%6d %15s %15s %2d
%11s %d %9d %8s\n", &update.acountNo,
update.firstName, update.lastName,
&update.age, update.contactNumber,
&update.accountStatus, &update.amount,
update.password) != EOF)
        {
          if (update.acountNo ==
sender.acountNo)
          {
            fprintf(temp, "%-6d %-15s %-15s %-
2d %-11s %-d %-9d %-8s\n", sender.acountNo,
sender.firstName, sender.lastName, sender.age,
sender.contactNumber, sender.accountStatus,
sender.amount - amountToTransfer,
sender.password);
          }
          else if (update.acountNo ==
receiver.acountNo)
          {
```

```
                fprintf(temp, "%-6d %-15s %-15s %-2d %-11s %-d %-9d %-8s\n", receiver.acountNo, receiver.firstName, receiver.lastName, receiver.age, receiver.contactNumber, receiver.accountStatus, receiver.amount + amountToTransfer, receiver.password);
            }
            else
            {
                fprintf(temp, "%-6d %-15s %-15s %-2d %-11s %-d %-9d %-8s\n", update.acountNo, update.firstName, update.lastName, update.age, update.contactNumber, update.accountStatus, update.amount, update.password);
            }
        }
        system("cls");
        SetColor(15);
        printf("%d ", amountToTransfer);
        SetColor(10);
        printf("transferred successfully to ");
        SetColor(11);
        printf("%s %s\n", receiver.firstName, receiver.lastName);
        Sleep(1500);
        system("cls");
        system("color 0F");
        fclose(fp);
        fclose(temp);
        fclose(transaction);
        remove("customer.txt");
        rename("customerTemp.txt", "customer.txt");
    }
        else
        {
            system("cls");
            printf("Account limit of %s %s reached maximum", receiver.firstName, receiver.lastName);
            Sleep(500);
        }
    }
    else
    {
        system("cls");
        printf("You don't have enough balance");
        Sleep(500);
    }
  }
  else
  {
    system("cls");
    printf("Account does not exists");
    Sleep(500);
  }
  fclose(fp);
transferAmountEnd:
  system("cls");
}
```

Explanation:

After, the user enters his portal by entering correct user and password, he is given the

option of transfer amount. On entering transfer amount portal, he enters the account number to which he wants to transfer the money. If the account exists in the bank, he is asked to enter the amount he wants to transfer. If he has enough money in his account, then the amount is successfully transferred in the person's account whose account number was previously mentioned. Some protocols are that:

- Negative amount cannot be entered. -1 is the only value which will be accepted, and it redirects the customer-to-customer portal.
- Account entered (to which money is to transfer) should exist in the bank.

void **viewMyTransactionHistory()**

```
{
   int flag = 0;
   char transactionType[10];
   transaction = fopen("transaction_history.txt", "r");
   while (fscanf(transaction, "%6d %15s %15s %9d %s\n", &update.acountNo, update.firstName, update.lastName, &update.amount, transactionType) != EOF)
   {
      if (customer.acountNo == update.acountNo)
      {
         flag = 1;
         break;
      }
   }
   fclose(transaction);
   if (flag == 1)
   {
      printf("Amount   Type   \n\n");
      transaction = fopen("transaction_history.txt", "r");
      while (fscanf(transaction, "%6d %15s %15s %9d %s\n", &update.acountNo, update.firstName, update.lastName, &update.amount, transactionType) != EOF)
      {
         if (customer.acountNo == update.acountNo)
         {
            printf("%-9d %-s\n", update.amount, transactionType);
         }
      }
      fclose(transaction);
   }
   else
   {
      system("cls");
      printf("You currently have no transactions\n");
      Sleep(2000);
   }
   printf("\nPress any key to continue");
   getch();}
```

Explanation:

After user enters his account Id and password, he is given the option to check his transaction history. This process involves, the reading of file: transaction.txt. Reading involves checking the user's account and password. When his account is found, then the transaction history corresponding to it is printed.

void **deleteAccount()**

```
{
   char ch;
   int accountToDelete = customer.acountNo;
   system("cls");
   if (customer.amount >= 0)
   {
      printf("Are you sure you want to delete your account? [y,n]: ");
      fflush(stdin);
      ch = getche();
      if (ch == 'y' || ch == 'Y')
      {
         fp = fopen("customer.txt", "r");
         temp = fopen("temp.txt", "w");
         if (fp == NULL || temp == NULL)
         {
            perror("Error");
            Sleep(1000);
         }
         fseek(fp, 0, SEEK_SET);
         while (fscanf(fp, "%6d %15s %15s %2d %11s %d %9d %8s\n", &update.acountNo, update.firstName, update.lastName, &update.age, update.contactNumber, &update.accountStatus, &update.amount, update.password) != EOF)
         {
            if (update.acountNo != accountToDelete)
            {
               fprintf(temp, "%-6d %-15s %-15s %-2d %-11s %-d %-9d %-8s\n", update.acountNo, update.firstName, update.lastName, update.age, update.contactNumber, update.accountStatus, update.amount, update.password);
            }
         }
         fclose(fp);
         fclose(temp);
         remove("customer.txt");
         rename("temp.txt", "customer.txt");
         system("cls");
         printf("Account Deleted Successfully, Redirecting to Main Menu");
         Sleep(2000);
         loadingAnimation();
         fflush(stdin);
         main();
      }
      else if (ch == 'n' || ch == 'N')
```

```
    {
      system("cls");
      customerPortal();}
    else{
      printf("Wrong choice entered, please enter a valid choice");
      system("cls");
      Sleep(1000);
      deleteAccount();
    }
  }
  else
  {
    printf("You can not delete you account as you have negative balance\nPay the balance first to delete your account\n");
  }}
```

Explanation:

In the customer portal, when customer choose to delete hi account, he is asked again whether he wants to delete it or not, if customer chooses “y”, then the program reads the customer.txt file and makes a temp.txt file to write all the data except the account which is to be deleted in it. Finally, the temp.txt file is renamed to customer.txt file again, and the file is closed. After closing file, the program is redirected to main menu.

CASE 3 OF SWITCH(mainMenuchoice):

This function (createNewAccount)will create a new account for the user and the function after execution will go back to the main body and in the third switch case of(mainMenuChoice)

```
void createNewAccount()
{
    char ch, choice; //variables
system("title CREATE NEW ACCOUNT");
customer.acountNo=generateAccountNu
                              -mber();
system("cls");
// FIRST LOOP:
    do
    {
    system("cls");
 printf("Enter your first name: ");
 fflush(stdin);
 scanf("%s", customer.firstName);
if(strlen(customer.firstName))>=15)
    {
 printf("\nLength of name must be less than or equal to 15 \a");
        Sleep(1500);
        }
    } while ((strlen(customer.firstName))>=15);
//SECOND LOOP:
    do
    {
system("cls");
printf("Enter your last name: ");
fflush(stdin);
```

```
scanf("%s", customer.lastName);
if(strlen(customer.firstName))>=15)
    {
printf("\nLength of name must be
less than or equal to 15 \a");
            Sleep(1500);
        }
    } while
((strlen(customer.lastName))>=15);
```

// THIRD LOOP:

```
    do
    {
system("cls");
printf("Enter your age: ");
fflush(stdin);
// scanf("%d", &customer.age);*
customer.age = integerInputOnly();
if
(customer.age<18||customer.age>99)
   {
printf("\nPeople between the age 18
and 99 can only create an
account\nTry Again! \a");
Sleep(1000);
   }
    }
While
(customer.age<18||customer.age>99);
```

// FOURTH LOOP:

```
    do
    {
system("cls");
printf("Enteryour phone number: ");
fflush(stdin);
scanf("%s",customer.contactNumber);
if
((strlen(customer.contactNumber))
!=11)
     {
printf("\nInvalidcontactnumber\a");
Sleep(1000);
     }
     }while
((strlen(customer.contactNumber))
!=11);
customer.accountStatus = 0;
customer.amount = 0;
system("cls");
    printf("Enter your password
(must be exactly 8 characters): ");
//FIFTH LOOP:
    for (i = 0; i <= 7;)
    {
    ch = getch();
if   ((ch >= 'a' && ch <= 'z') ||
      (ch >= 'A' && ch <= 'Z') ||
      (ch >= '0' && ch <= '9') ||
      (ch >= 33 && ch <= 47))
        {
customer.password[i] = ch;
 ch = '*';
 printf("%c", ch);
 i++;
        }
    }
customer.password[8] = '\0';
system("color 0B");
system("cls");
printf("Name: ");
printf("%s %s\n",
customer.firstName,
customer.lastName);
printf("Account Number  : %d\n",
customer.acountNo);
printf("Age             : %d\n",
customer.age);
printf("Contact Number  : %s\n",
customer.contactNumber);
printf("Account Status  : %s\n",
(customer.accountStatus == 0 ?
"Active" : "Blocked"));
printf("Account Balance : %d\n",
customer.amount);

printf("\nAre your sure you want to
create your account: [y/n]: ");

    while (1)
    {
        ch = getch();
```

```
        if (ch == 'y' || ch == 'Y'
|| ch == 'n' || ch == 'N')
        {
            choice = ch;
            printf("%c", ch);
            break;
        }
    }

    system("color 0F");
if (choice == 'y' || choice == 'Y')
    {
 fp = fopen("customer.txt", "a");
fprintf(fp, "%-6d %-15s %-15s %-2d
%-11s %-d %-9d %-8s\n",
customer.acountNo,
customer.firstName,
customer.lastName, customer.age,
customer.contactNumber,
customer.accountStatus,
customer.amount,
customer.password);
        fclose(fp);
        system("cls");
        system("color 0A");
printf("AccountcreatedSuccesfully);
Sleep(1000);
    }
}
```

NOTE: in every loop fflush(stdin) is used to remove garbage values

Explanation for case 3:

Firstly, when we make an account we need an account number for every user so we created ***generateAccountNumber()*** and stored **in *customer.accountNo***
Here generateAccountnumber is coded further in the text.
//First loop:
Now we will enter first name for that only 15 or less length names are accepted and stored in **customer.firstName**

//Second Loop:
Now we will enter last name for that only 15 or less length name are accepted and stored in **customer.lastName**
//Third Loop:
Now we will enter the age between 18-99 knowing that only integer value can be stored in **customer.age**
Here **integerinputonly()** is an other function explained in the end of text

//Fourth Loop:
Now enter your phone number which cannot exceed 11 digits and stored in ***customer.contactNumber***.
//Fifth Loop:
After getting the basic information ask the user to enter password exactly of 8 character or digits
And then store the password in ***customer.password[i] = ch*** then * is used to hide when a person is entering his passcode instead of just showing the as it is.

customer.accountStatus = 0
if is equal to 0 then the account would be active

customer.amount = 0 this can be updated by excessing case 2.

Then we finalize this function by asking yes or no to create the account if said yes all the data obtained by the user will be stored in a file by using filing after storing we will close the file and we will be back to the main interface.

`int` **generateAccountNumber()**

```
{
 int acc;
fp = fopen("customer.txt", "r");
check:
fseek(fp, 0, SEEK_SET);
srand(time(0));
    do
    {
    acc = (rand() % 1000) + 100000;
    }
while(acc<100000 || acc> 99999);



while (fscanf(fp, "%6d %15s %15s
%2d %11s %d %9d %8s\n",
&customer.acountNo,
customer.firstName,
customer.lastName, &customer.age,
customer.contactNumber,
&customer.accountStatus,
&customer.amount,
customer.password) != EOF)
    {
if (acc == customer.acountNo)
        {
   goto check;
        }
    else
        {
   goto ganend;
        }
    }
ganend:
    fclose(fp);
    return acc;
}
```

Explanation for generateAccountNumber:

We will save every account number generated and the data that we input in each account stored in a file and can be excessed by customer and admin to view all the information and credits.

If the account Number Entered in the program is equal to the account number generated for the user than the file will open and show all the information if not, then the file will not open and accountnumber will be asked again.

```
void loginScreenAnimation()
{
    system("cls");
    system("color 01");
    printf("Loading");
    for (i = 1; i <= 3; i++)
    {
        Sleep(500);
        printf("\xB2");
    }
    system("cls");
    system("color 0F");
}
```

Explanation:

This function is made to display a loading bar mainly for user interaction at by using sleep()

```
int integerInputOnly()
{
    int x = 0;
    char ch;
    int count;

    for (count = 1; count <= 2;)
    {
        ch = getch();
        if (ch >= '0' && ch <= '9')
```

```
        {
            printf("%c", ch);
            x = (x * 10) + (ch - 48);
            count++;
        }
        if (ch == 13)
        {
            break;
        }
    }
    return x;
}
```

Explanation:

We are going to check the password to be an integer more than one time in this enter program, so we made a function to check whether the password entered every time falls on the condition.

CASE 4 OF SWITCH(mainMenuchoice):

```
void viewCurrencyRates()
{
fp=fopen("currency_rates.txt", "r");
system("cls");
system("color F0");
printf("Currency Symbol Rate\n");

while (fscanf(fp, "%10s %10s %3s
%6f\n", currency.first,
currency.last, currency.code,
&currency.rate) != EOF)
    {
printf("%-10s %-10s %6s %-6.2f\n",
currency.first, currency.last,
currency.code, currency.rate);
    }
printf("\nPRESS ANY KEY TO
CONTINUE\n");
 getch();
 fclose(fp);
 system("color 0F");
}
```

Explanation:

Saving all the currencies rates in a file which can be excessed by the admin and can be changed according to international rates an easy option to view currency rate of other countries with respect to rupees. Used filing to open and checking the file until end of file before displaying the rates.

CASE 5 OF SWITCH(mainMenuchoice):

```
void aboutUs()
{
 system("cls");
 for (j = 21; j >= 0; j--)
    {
 if(j == 21){
    system("color 01");
    }
 if(j == 19){
    system("color 02");
        }
 if(j == 18){
    system("color 03");
        }
 if(j == 17){
    system("color 04");
        }
 if(j == 16){
    system("color 05");
        }
 if(j == 15){
```

```
    system("color 06");
       }
 if(j == 14){
    system("color 07");
       }
 if(j == 13){
    system("color 08");
       }
 if(j == 12){
    system("color 09");
       }
 if(j == 11){
    system("color 0A");
       }
 if(j == 10){
    system("color 0B");
       }
 if(j == 9){
    system("color 0C");
       }
if(j == 8){
  system("color 0D");
   }
if(j == 7){
  system("color 0E");
   }
if(j == 6){
  system("color 0F");
   }
if(j == 5){
  system("color 01");
   }
if(j == 4){
   system("color 02");
   }
if(j == 3){
   system("color 03");
   }
if(j == 2){
   system("color 0C");
   }
if(j == 1){
   system("color 0E");
   }
if(j == 0){
   system("color 0F");
   }
system("cls");
gotoxy(40, j);
printf("THIS SYSTEM IS DESIGNED BY");
gotoxy(40, j + 1);
printf("Sufiyaan Usmani (21K-3195)");
gotoxy(40, j + 2);
printf("Ahsan Ashraf    (21K-3186)");
gotoxy(40, j + 3);
printf("Qasim Hasan     (21K-3210)");
gotoxy(40, j + 4);
printf("Talha Shaikh    (21K-4546)");
gotoxy(0,0);
Sleep(200);
    }
}
```

Explanation:

Using gotoxy() to print the name of team mates in making this bank management project and using system (to make it more presentable.