SDM College of Engineering and Technology
Dharwad-580002

DEPARTMENT OF COMPUTER SCIENCE AND ENGINEERING

Database management System laboratory

Course Instructor: Prof. Anand Vaidya

2023-24

Report on
" ATM system using C "

Submitted by

USN	Name
2SD21CS006	Aditya A Joshi
2SD21CS088	Shashidhar V G

Index:

1.Assumptions:

Credentials: Users are identified by a combination of Account number, Card ID and PIN. The system assumes a fixed length for Account Number (up to 12 digits), Card IDs (up to 10 digits) and PINs (up to 4 digits). And user should use the card number and the PIN as login credentials which had been given by the admin.

Initial ATM Parameters : The ATM starts the day with initial parameters:

a. Total fund in the ATM (`t`) is set to 100000/-
b. Maximum withdrawal per day and account (`k`) is set to 20000/-
c. Maximum withdrawal per transaction (`m`) is set to 20000/-
d. Minimum cash in the ATM to permit a transaction (`n`) is set to 100/-

Failed Login Attempts : If a user enters an incorrect PIN more than 3 times, their card is blocked. And can be unblocked only by the admin of the bank.

Reset Failed Login Attempts : Failed login attempts are reset to zero once a successful login occurs.

Invalid Input Handling : The code assumes that users will input valid data, and there is minimal error checking. For example, it doesn't handle situations where users might enter non-numeric values for amounts.

Fixed User IDs : User IDs are assigned incrementally starting from 1. This simplifies user management but may not be suitable for a real-world application.

Blocking a Card : When a user exceeds the maximum allowed failed login attempts, the code blocks the card. However, there is no provision for unblocking the card.

Limited ATM Functionality : The ATM system has limited functionality, including withdrawal, balance checking, depositing, viewing the remaining ATM balance, and viewing transaction logs. It doesn't cover other common banking operations, like transferring money between accounts or changing the PIN.

2 . Structure used :

The **Bank** structure is used to represent an individual user of the ATM system. It includes information such as:

- name: The name of the user.
- acNo: A unique identifier for the user.
- cardNo: The card ID associated with the user.
- passWord: The PIN associated with the card.
- totBal: The current balance of the user.
- bal1,bal2,bal3,bal4,bal5:The last 5 transactions of the user.

3. Modules Implementation :

User Profiles:

- Each user has a name, an acNo, cardNo, passWord, totbal, and other relevant information.
- User authentication is implemented based on the cardNo and passWord.
- Users can withdraw, deposit, and check balance.
- There are maximum daily withdrawal limits and per-transaction withdrawal limits.

Bank Module:

- A Bank has the ability to log in and create a new user.
- The Bank name and password are hardcoded in the system.

ATM Module:

- Shows the current balance left in the ATM.

● Displays a transaction log that contains a list of all transactions. Transaction Logs:

● All transactions, including balance checks, are logged with a timestamp.

Data Handling :

● Users' data is saved to a file (bankAdmin.txt) to ensure data persistence across different runs of the program.

● Transaction logs are saved to a file (accounts.txt).

4. Flow of program(Working)

1. User Structure : - The code defines a `User` structure to store user-related information, including name, user acNo, cardNo, passWord, totBal, bal1,bal2,bal3,bal4,bal5.

2. Main Function : - The `main` function is the entry point of the program. It initializes parameters, such as ATM balance, maximum daily withdrawal limit, and more. - It presents a menu to the user with four options: Deposit money, Withdraw money, Mini statement, Balance enquiry, Exit- Based on the user's choice, it calls different modules: ` depositMoney`, ` withDrawal`, ` miniStatement`,' balanceEnquiry' or exits the program.

3. Admin Program :

- The `admin program` allows bank administrators to create new user accounts.

-The admin can create a new user by entering a name, and a unique card ID and PIN are generated.

-The admin can also unblock the card by entering the card number and account number.

4. User Program :

- The `user program` is where a customer can interact with the ATM.

- Users enter their cardNo and passWord for authentication.

- If the login is successful, they are presented with a menu that includes options to withdraw, check balance, deposit, miniStatement, balanceEnquiry.

5. Create User :

- The `createUser` function is used by bank administrators to create new user accounts. It generates a random 12-digit acNo, 10-digit cardNo 4-digit password for each user.

6. Authentication :

- The `authenticateUser` function verifies a user's identity by comparing the entered cardNo and passWord with stored values.

- It also tracks login attempts and can block the card if too many login failures occur.

7. Display Menu :

- The `main` function displays the user menu, where customers can choose withdraw, deposit, miniStatement, balanceEnquiry.

8. Withdrawal :

- The `withdraw` function allows users to withdraw a specified amount from their account. It checks for various limits (ATM balance, daily withdrawal limit, and per-transaction limit) before processing the withdrawal.

9. Balance Enquiry :

- The `checkBalance` function displays the user's account balance.

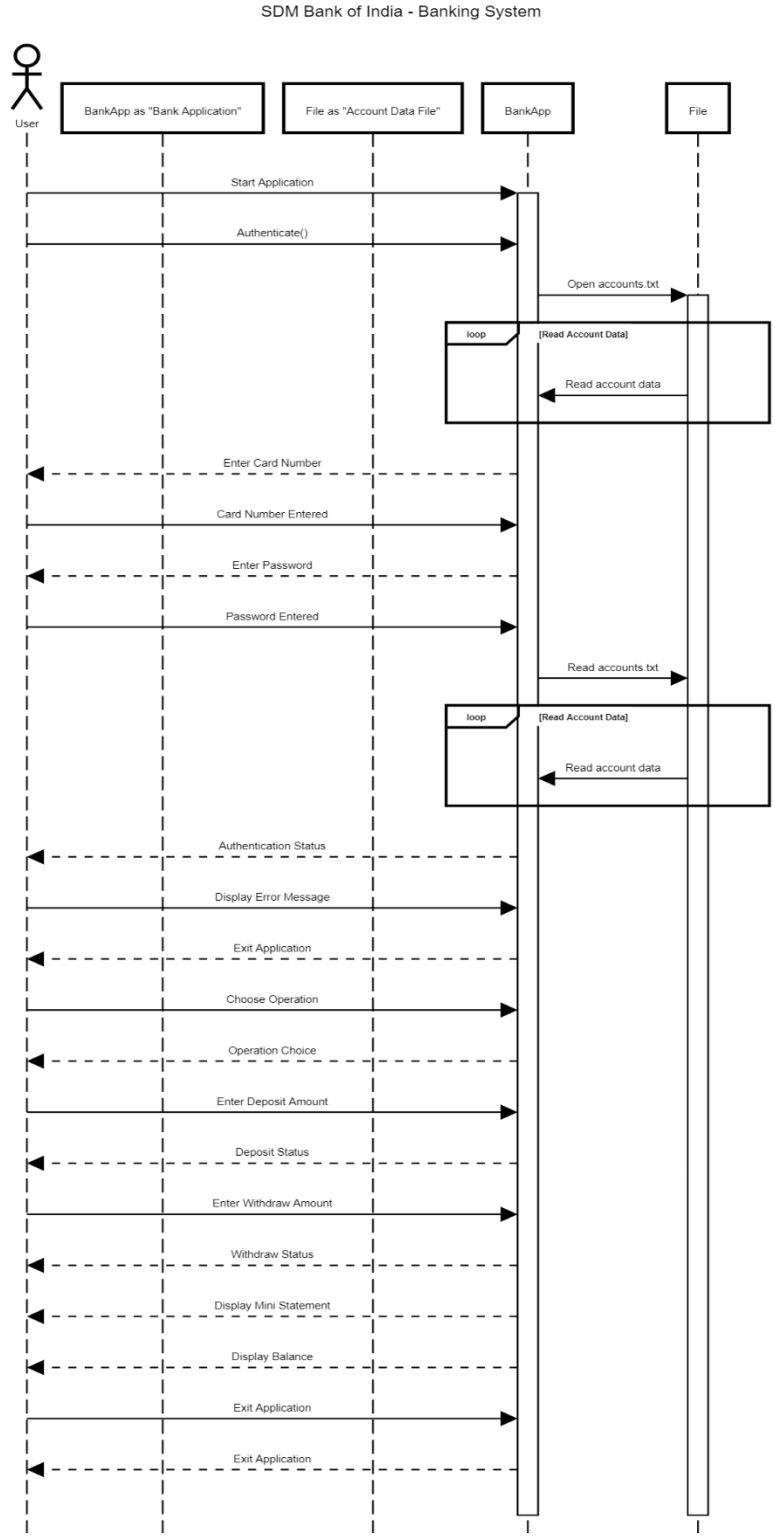
10. Deposit :

- The `deposit` function allows users to deposit money into their account.

11.Mini Statement :

- Added the `miniStatement` function that keeps track of a customer's recent transactions in a dedicated statement file.

5. Sequence diagram :



Chapter 1

Introduction

1.1 Purpose

This document describes the software requirements for an automated teller machine network (ATM). It is intended for the designer, developer and maintainer of the ATM.

1.2 Scope

The function of the ATM is to support a computerized banking network.

1.3 Overview

The remainder of this document is organized as follows: There will be some definitions of important terms. Section 2 contains a general description of the ATM. Section 3 identifies the specific functional requirements, the external interfaces and performance requirements of the ATM.

1.4 Definitions

Account

a single account in a bank against which transactions can be applied. Accounts may be of various types with at least checking and savings. A customer can hold more than one account.

ATM

A station that allows customers to enter their own transactions using cash cards as identification. The ATM interacts with the customer to gather transaction information, sends the transaction information to the central computer for validation and process-ing, and dispenses cash to the customer. We assume that an ATM need not operate independently of the network.

5

Bank

A financial institution that holds accounts for customers and that issues cash cards authorizing access to accounts over the ATM network.

Bank computer

The computer owned by a bank that interfaces with the ATM network and the bank's own cashier stations. A bank may actually have its own internal network of computers to process accounts, but we are only concerned with the one that interacts with the network.

Cash Card

A card assigned to a bank customer that authorizes access to accounts using an ATM machine. Each card contains a bank code and a card number, coded in accordance with national standards on credit cards and cash cards. The bank code uniquely identifies the bank within the consortium. The card number determines the accounts that the card can access. A card does not necessarily access all of a customer's accounts. Each cash card is owned by a single customer, but multiple copies of it may exist, so the possibility of simultaneous use of the same card from different machines must be considered.

Customer

The holder of one or more accounts in a bank. A customer can consist of one or more persons or corporations the correspondence is not relevant to this problem. The same person holding an account at a different bank is considered a different customer.

Transaction

a single integral request for operations on the accounts of a single customer. We only specified that ATMs must dispense cash, but we should not preclude the possibility of printing checks or accepting cash or checks. We may also want to provide the flexibility to operate on accounts of different customers, although it is not required yet. The different operations must balance properly.

6

7

Chapter 2

General Description

2.1 Product Perspective

The ATM network doesn't work independently. It has to work together with the comput ers/software owned by banks. There are clearly defined interfaces for the different systems.

2.2 Product Functions

The software should support a computerized banking network. Each bank provides its own computer to maintain its own accounts and process transactions against them. Automatic teller machines communicate with the banks' computers. An automatic teller machine accepts a cash card, interacts with the user, communicates with the bank computer to carry out the transaction, dispenses cash and prints receipts. The system requires appropriate record keep-ing and security provisions.

The system must handle concurrent access to the same account correctly. The banks will provide their own software for their own computers. The cost of the shared system will be apportioned to the banks according to the number of customers with cash cards.

2.3 User Characteristics

There are several users of the ATM network:

Customer

The customer interacts with the ATM network via the ATM. It must be very easy for them to use the ATM. They should be supported by the system in every possible way.

Maintainer

It should be easy to maintain the whole system. The maintainer should be the only person that is allowed to connect a new ATM to the network.

2.4 Abbreviations

Throughout this document the following abbreviations are used: k is the maximum withdrawal per day and account m is the maximum withdrawal per transaction n is the minimum cash in the ATM to permit a transaction t is the total fund in the ATM at start of day

8

Chapter 3

Specific Requirements

3.1 Functional Requirements

The functional requirements are organized in two sections: First requirements of the ATM and second requirements of the bank.

3.1.1 Requirements of the automated teller machine

The requirements for the automated teller machine are organized in the following way: General requirements, requirements for authorization, requirements for a transaction.

General

Functional requirement 1

_ Description

Initialize parameters t,k,m,n

Input

ATM is initialized with t dollars. k,m,n are entered

Processing

Storing the parameters

Output Parameters are set.

Functional requirement 2

_ Description

If no cash card is in the ATM, the system should display initial display.

Functional requirement 3

Description

If the ATM is running out of money, no card should be accepted. An error message is displayed.

_

Input

A card is entered.

_ Processing

The amount of cash is less than t.

9

_ Output

Display an error message. Return cash card.

Authorization

The authorization starts after a customer has entered his card in the ATM.

Functional requirement 4

_ Description

The ATM has to check if the entered card is a valid cash-card.

_

Input

Customer enters the cash card.

_ Processing

Check if it is a valid cash card. It will be valid if

1. the information on the card can be read.

2. it is not expired.

Output

Display error message and return cash card if it is invalid.

Functional requirement 5

_ Description

If the cash card is valid, the ATM should read the serial number and bank code.

_

Input

Valid cash card.

_ Processing

Read the serial number.

_ Output

Initiate authorization dialog

Functional requirement 6

_ Description

The serial number should be logged.

_

Input

Serial number from cash card

10

_ Processing

Log the number.

_ Output

Update to log _le.

Functional requirement 7

_ Description

Authorization dialog: The user is requested to enter his password. The ATM verifies the bank code and password with the bank computer

_

Input

Password from user, bank code from cash card.

_ Processing

Send serial number and password to bank computer, receive response from bank.

_ Output

Accept or reject authorization from bank.

Functional requirement 8

_ Description

Different negative answers from bank computer for authorization dialog.

_

Input

Response from bank or authorization dialog:

“bad password" if the password was wrong.

“bad bank code" if the cash card of the bank is not supported by the ATM. {

“bad account" if there are problems with the account.

Processing

If the ATM gets any of these messages from the bank computer, the card will be ejected and the user will get the relevant error message.

_ Output

Card is ejected and error message is displayed.

Functional requirement 9

_ Description

If password and serial number are ok, the authorization process is finished.

11

_

Input

The ATM gets accept from the bank computer from authorization process.

_ Processing

Finishing authorization

_ Output

Start transaction dialog

Functional requirement 10

_ Description

If a card was entered more than three times in a row at any ATM and the password was wrong each time, the card is kept by the ATM. A message will be displayed that the customer should call the bank.

_

Input

Entering a wrong password for the fourth time in succession

_ Processing

Initiate authorization process. Response from bank computer is to keep the card.

_ Output

Display error message that the customer should call the bank.

Functions

These are the requirements for the different functions the ATM should provide after autho rization.

Functional requirement 11

_ Description

The kind of transactions the ATM offers is: withdrawal

_

Input

Authorization successfully completed. Enter the amount to withdraw.

_Processing

Amount entered is compared with m.

_ Output

Amount of money to be dispensed is displayed. Begin initial withdrawal sequence.

12

Functional requirement 12

_ Description

Initial withdrawal sequence: If it is too much withdrawal redo the transaction.

_

Input

Customer has entered the amount of money.

_ Processing

Error if the amount is greater than m.

_ Output

Start transaction or re-initiate transaction dialog if the amount is not within the pre defined transaction policy

Functional requirement 13

_ Description

Perform transaction.

_

Input

Initial withdrawal sequence successful

_ Processing

Send request to the bank computer.

_ Output

Wait for response from the bank computer.

Functional requirement 14

_ Description

If the transaction is successful, the money is dispensed.

_

Input

ATM gets message \transaction succeeded" from the bank computer.

_ Processing

ATM prints receipt, updates t and ejects the card. Dialog: Customer should take the card.

_ Output

After the Customer has taken the card the money is dispensed.

13

Functional requirement 15

_ Description

If the money is dispensed, the amount is logged

_

Input

The number of $20 bills requested is dispensed to the customer.

_ Processing

Log the amount of money against the serial number of the card.

_ Output

Amount logged together with the serial number. Response sent to bank for money dispensed.

Functional requirement 16

_ Description

If the transaction is not successful, an error message should be displayed. The card should be ejected.

_

Input

ATM gets message \transaction not successful" from the bank computer

_ Processing

ATM displays error message. Dialog: Customer should take the card.

_ Output

Eject card.

3.1.2 Requirements of the bank computer for the ATM

Authorization

The bank computer gets a request from the ATM to verify an account.

Functional requirement 1

_ Description

The bank computer checks if the bank code is valid. A bank code is valid if the cash card was issued by the bank.

_

Input

Request from the ATM to verify card (Serial number and password)

14

_ Processing

Check if the cash card was issued by the bank.

_ Output

Valid or invalid bank code.

Functional requirement 2

_ Description

If it is not a valid bank code, the bank computer will send a message to the ATM.

_

Input

Invalid bank code

_ Processing

Process message

_ Output

The bank computer sends the message \bad bank code" to the ATM.

Functional requirement 3

_ Description

The bank computer checks if the password is valid for a valid cash card.

_

Input

Request from the ATM to verify password.

_ Processing

Check password of the customer.

_ Output

Valid or invalid password

Functional requirement 4

_ Description

If it is not a valid password, the bank computer will send a message to the ATM.

_

Input

Invalid password

_ Processing

Process message. Update count for invalid password for the account.

15

_ Output

The bank computer sends the message \bad password" to the ATM.

Functional requirement 5

_ Description

If it is a valid cash card and a valid password but there are problems with the account, the bank will send a message to the ATM that there are problems.

_

Input

Valid cash card and password

_ Processing

Process message

_ Output

The bank sends \bad account" to the ATM.

Functional requirement 6

_ Description

If it is a valid cash card, a valid password and there are no problems with the account the bank computer will send a message to the ATM that everything is ok

_

Input

Valid cash card, password and account

_ Processing

Process message.

_ Output

Send “account ok" to the ATM.

Transaction

The bank computer gets a request to process a transaction from the ATM

Functional requirement 7

_ Description

After a request the bank computer processes the transaction.

_

Input

Request to process a transaction on an account and amount m to withdraw.

16

_ Processing

Process transaction (together with the software of the bank). Update k for amount

_ Output

If transaction succeeded, the bank computer sends the message \transaction succeeded" to the ATM. If not, it will send \transaction failed".

Functional requirement 8

_ Description

Update account after money is dispensed

Input Response from ATM about money dispensed.

Processing

Updates account

_ Output

New account record

Functional requirement 9

_ Description

Each bank has a limit k for each account about the amount of money that is available via cash card each day/monthly.

_

Input

Request to process transaction.

_ Processing

Check if the amount of money doesn't exceed k

_ Output

If the amount exceeds the limit, the transaction will fail.

Functional requirement 10

_ Description

The bank only provides security for their own computer and their own softwar

6. Code implementaytion

bankAdmin.c

```
#include<stdio.h>
#include<string.h>
#include<time.h>
#include<math.h>
struct Bank
{
  char name[30];
  long long acNo;
  long cardNo;
  int passWord;
  float bal1,bal2,bal3,bal4,bal5,totBal;
};
struct Bank bank;
FILE* adminFile,*user,*read;
void createAccount()
{
    char line[1000];
    char line2[200];
    double bal1=0,bal2=0,bal3=0,bal4=0;
    long long upper=1000000000000-1,lower=100000000000;
    user =fopen("accounts.txt","a");
    adminFile = fopen("adminFile.txt","a");
    printf("Enter the name\n");
    scanf("%s",bank.name);
```

```
    printf("Enter the amount to be added\n");
    scanf("%f",&bank.totBal);
    srand(time(NULL));
    bank.acNo=(rand()%(upper-lower+1)+lower);
    bank.cardNo=(rand()%(9900000000-1000000000+1)+1000000000);
    bank.passWord=(rand()%(9900-1000+1)+1000);
    sprintf(line,"%-20s||%-14lld||%-10ld||%-
6d\n",bank.name,bank.acNo,bank.cardNo,bank.passWord);
    sprintf(line2,"active||%-10ld||%-6d||%-8f||%-8f||%-8f||%-8f||+%-8f||+%-
8f\n",bank.cardNo,bank.passWord,0.0,0.0,0.0,0.0,bank.totBal,bank.totBal);
    fwrite(line,1,strlen(line),adminFile);
    fwrite(line2,1,strlen(line2),user);
     fclose(adminFile);
     fclose(user);
}
void viewDetails()
{
    read = fopen("adminFile.txt","r");
    printf("Account details are\n");
    char line[100];
    while(fgets(line,100,read)!=NULL)
    {
      printf("%s\n",line);
    }
    fclose(read);
}
void unBlock()
{
   float bal1,bal2,bal3,bal4,bal5,totBal,depo;
```

```
FILE *read,*write;
write=fopen("temp.txt","w");
read=fopen("accounts.txt","r");
int lineUpdated=0;
long card;
printf("Enter the card number to be unblocked\n");
scanf("%ld",card);
 char newline[100],line2[100],line[100];
 while (fgets(line, 100, read) != NULL)
 {
   strcpy(line2,line);
   char *status1=strtok(line,"||");
   char *token = strtok(NULL, "||");
   char *pass=strtok(NULL,"||");
   sscanf(pass, "%d", &bank.passWord);
   if((sscanf(token, "%ld", &bank.cardNo) == 1)&&card==bank.cardNo)
   {
     char* cBal1=strtok(NULL,"||");
     char* cBal2=strtok(NULL,"||");
     char* cBal3=strtok(NULL,"||");
     char* cBal4=strtok(NULL,"||");
     char* cBal5=strtok(NULL,"||");
     char* cTotBal=strtok(NULL,"||");
     sscanf(cBal1, "%f",&bank.bal1 );
     sscanf(cBal2, "%f",&bank.bal2 );
     sscanf(cBal3, "%f",&bank.bal3 );
      sscanf(cBal4, "%f",&bank.bal4 );
     sscanf(cBal5, "%f",&bank.bal5 );
```

```
        sscanf(cTotBal, "%f",&bank.totBal );
        sprintf(newline,"active||%-10ld||%-6d||%-8f||%-8f||%-8f||%-8f||+%-8f||%-
8f\n",bank.cardNo,bank.passWord,bank.bal1,bank.bal2,bank.bal3,bank.bal4,bank.bal5,bank.totB
al);
        fprintf(write,"%s",newline);
       }
       else
       {
        fprintf(write,"%s",line2);
       }
      }
       fclose(write);
       fclose(read);
       remove("accounts.txt");
       rename("temp.txt","accounts.txt");
}
void main()
{
  int choice;
  while(1)
  {
   printf("Emter the choice\n1-Create account 2-View accounts 3-Unblock 0-Exit\n");
   scanf("%d",&choice);
   switch(choice)
   {
    case 1:createAccount();
           break;
    case 2:viewDetails();
           break;
```

```
    case 3:unBlock();
    case 0:
    return;
    default:printf("Invalid choice\n");
            break;
   }
  }
}
```

atm.c

```
#include<stdio.h>
#include<string.h>

char line[100];
struct Bank
{
  int status;
  long cardNo;
  int pass,attempts;
  float bal1,bal2,bal3,bal4,bal5,totBal;
};
struct Bank bank;
int authenticate()
{
   FILE *read,*write;
   write=fopen("temp.txt","w");
   read=fopen("accounts.txt","r");
   char newline[100],line2[100],line[100];
```

```
long cardNo;
int flag=0;
printf("Enter the card number\n");
scanf("%ld",&cardNo);
char *cardStr = (char)cardNo;
while(fgets(line,100,read)!=NULL)
{
  strcpy(line2,line);
  char *status=strtok(line,"||");
  char *token=strtok(NULL,"||");

  if(strcmp(status,"active")==0)
    {
      bank.attempts=0;
    }
    else
    {
     printf("Account has been blocked\n");
     return 0;
    }
    long card;
    char pass[6];
  if (sscanf(token, "%ld", &card) == 1)
  {
    char* passW=strtok(NULL,"||");
    int opass;
    sscanf(passW, "%d", &opass);
    sscanf(passW, "%d", &bank.pass);
```

```
if(card==cardNo)
{
  bank.cardNo=card;

    while(bank.attempts<4){
  printf("Enter the Password\n");
  scanf("%s",pass);
  char ch;
  int count =0;
  int epass,opass;
  sscanf(pass, "%d", &epass);
 if(epass==opass)
 {
  bank.pass=epass;
  printf("Authenticated\n");
  flag=1;
 }
else
{
   bank.attempts++;
   printf("Invalid password.....\n");
}
  }
}
 if(bank.attempts>=3)
  {
   fprintf(write,"in");
   fprintf(write,"%s",line2);
```

```
            printf("Your card has been blocked\n");
          }
        else{
            fprintf(write,"%s",line2);
         }
      }
    }
fclose(read);
fclose(write);
remove("accounts.txt");
rename("temp.txt","accounts.txt");
if(flag==0)
{
  printf("Account doesnot exist!\n");
  fclose(read);
  fclose(write);
  return 0;
}
else
{
    return 1;
}
}
void depositMoney()
{
     float bal1,bal2,bal3,bal4,bal5,totBal,depo;
     FILE *read,*write;
     write=fopen("temp.txt","w");
```

```
read=fopen("accounts.txt","r");
int lineUpdated=0;
 char newline[100],line2[100];
 while (fgets(line, 100, read) != NULL)
 {
  strcpy(line2,line);
  long card;
  char *status1=strtok(line,"||");
  char *token = strtok(NULL, "||");
  char *pass=strtok(NULL,"||");
  if((sscanf(token, "%ld", &card) == 1)&&card==bank.cardNo)
  {
   char* cBal1=strtok(NULL,"||");
   char* cBal2=strtok(NULL,"||");
   char* cBal3=strtok(NULL,"||");
   char* cBal4=strtok(NULL,"||");
   char* cBal5=strtok(NULL,"||");
   char* cTotBal=strtok(NULL,"||");
   sscanf(cBal1, "%f",&bank.bal1 );
   sscanf(cBal2, "%f",&bank.bal2 );
   sscanf(cBal3, "%f",&bank.bal3 );
    sscanf(cBal4, "%f",&bank.bal4 );
   sscanf(cBal5, "%f",&bank.bal5 );
   sscanf(cTotBal, "%f",&bank.totBal );
   printf("Enter the amount to be deposited\n");
   scanf("%f",&depo);
   bank.totBal+=depo;
   bank.bal1=bank.bal2;
```

```
        bank.bal2=bank.bal3;
        bank.bal3=bank.bal4;
        bank.bal4=bank.bal5;
        bank.bal5=depo;
        lineUpdated=1;
        printf("Balance is:%f",bank.totBal);
        sprintf(newline,"active||%-10ld||%-6d||%-8f||%-8f||%-8f||%-8f||+%-8f||%-
8f\n",bank.cardNo,bank.pass,bank.bal1,bank.bal2,bank.bal3,bank.bal4,bank.bal5,bank.totBal);
        fprintf(write,"%s",newline);

      }
      else{
        fprintf(write,"%s",line2);
      }
    }
     fclose(read);
     fclose(write);

    if(lineUpdated)
    {
     if(remove("accounts.txt")!=0)
      perror("Unable to remove the file\n");

     if(rename("temp.txt","accounts.txt")!=0)
      perror("Unable to rename the file\n");
    }
    else{
     printf("Account not found\n");
    }
```

```
}
void withDrawal()
{
  float money;
   FILE *read,*write;
   write=fopen("temp.txt","w");
   read=fopen("accounts.txt","r");
   int lineUpdated=0;
   char newline[100],line2[100];
   while (fgets(line, 100, read) != NULL)
   {
       strcpy(line2,line);
       long card;
        char *status1=strtok(line,"||");
       char *token = strtok(NULL, "||");
       char *pass=strtok(NULL,"||");
       if((sscanf(token, "%ld", &card) == 1)&&card==bank.cardNo)
       {
         char* cBal1=strtok(NULL,"||");
         char* cBal2=strtok(NULL,"||");
         char* cBal3=strtok(NULL,"||");
         char* cBal4=strtok(NULL,"||");
         char* cBal5=strtok(NULL,"||");
         char* cTotBal=strtok(NULL,"||");
         sscanf(cBal1, "%f",&bank.bal1 );
         sscanf(cBal2, "%f",&bank.bal2 );
         sscanf(cBal3, "%f",&bank.bal3 );
```

```
 sscanf(cBal4, "%f",&bank.bal4 );
sscanf(cBal5, "%f",&bank.bal5 );
sscanf(cTotBal, "%f",&bank.totBal );

printf("Enter the amount to be withdrawn\n");
scanf("%f",&money);

if(money>(bank.totBal+500))
{
  printf("Insufficient balance\n");
  return;
}
if(money>25000)
{
  printf("Exceeding the maximum limit\n");
  return;
}
bank.totBal-=money;
bank.bal1=bank.bal2;
bank.bal2=bank.bal3;
bank.bal3=bank.bal4;
bank.bal4=bank.bal5;
bank.bal5=money;
printf("Balance is:%f",bank.totBal);
sprintf(newline,"active||%-10ld||%-6d||%-8f||%-8f||%-8f||%-8f||-%-8f||%-
8f\n",bank.cardNo,bank.pass,bank.bal1,bank.bal2,bank.bal3,bank.bal4,bank.bal5,bank.totBal);
fprintf(write,"%s",newline);
lineUpdated=1;
}
```

```
        else{
         fprintf(write,"%s",line2);
        }
      }
      fflush(read);
fclose(read);
fflush(write);
fclose(write);

    if(lineUpdated)
    {
      if(remove("accounts.txt")!=0)
       perror("Unable to remove the file\n");

      if(rename("temp.txt","accounts.txt")!=0)
       perror("Unable to rename the file\n");
    }
    else{
     printf("Account not found\n");
    }
}
void miniStatement()
{
  float money;
  FILE *read,*write;
  read=fopen("accounts.txt","r");
  int lineUpdated=0;
     char newline[100],line2[100];
```

```
while (fgets(line, 100, read) != NULL)
{
 strcpy(line2,line);
 long card;
  char *status1=strtok(line,"||");
 char *token = strtok(NULL, "||");
 char *pass=strtok(NULL,"||");
 if((sscanf(token, "%ld", &card) == 1)&&card==bank.cardNo)
 {
  char* cBal1=strtok(NULL,"||");
  char* cBal2=strtok(NULL,"||");
  char* cBal3=strtok(NULL,"||");
  char* cBal4=strtok(NULL,"||");
  char* cBal5=strtok(NULL,"||");
  char* cTotBal=strtok(NULL,"||");
  sscanf(cBal1, "%f",&bank.bal1 );
  sscanf(cBal2, "%f",&bank.bal2 );
  sscanf(cBal3, "%f",&bank.bal3 );
   sscanf(cBal4, "%f",&bank.bal4 );
  sscanf(cBal5, "%f",&bank.bal5 );
  sscanf(cTotBal, "%f",&bank.totBal );
  printf("Mini statement\nCredit/Debit||Amount\n");
  if(bank.bal1>=0)
   printf("Credit  ||  %f \n",bank.bal1);
  else
   printf("Debit  ||  %f \n",bank.bal1);
  if(bank.bal2>=0)
   printf("Credit  ||  %f \n",bank.bal2);
```

```
        else
         printf("Debit  ||  %f \n",bank.bal2);
        if(bank.bal3>=0)
         printf("Credit  ||  %f \n",bank.bal3);
        else
         printf("Debit  ||  %f \n",bank.bal3);
        if(bank.bal4>=0)
         printf("Credit  ||  %f \n",bank.bal4);
        else
         printf("Debit  ||  %f \n",bank.bal4);
        if(bank.bal5>=0)
         printf("Credit  ||  %f \n",bank.bal5);
        else
         printf("Debit  ||  %f  \n",bank.bal5);
        printf("The balance is:%f\n",bank.totBal);
      }
    }
}

void balanceEnquiry()
{
  float money;
 FILE *read,*write;
 read=fopen("accounts.txt","r");
   char newline[100],line2[100];
     while (fgets(line, 100, read) != NULL)
     {
       strcpy(line2,line);
```

```
        long card;
        char *status1=strtok(line,"||");
        char *token = strtok(NULL, "||");
        char *pass=strtok(NULL,"||");
        if((sscanf(token, "%ld", &card) == 1)&&card==bank.cardNo)
        {
          char* cBal1=strtok(NULL,"||");
          char* cBal2=strtok(NULL,"||");
          char* cBal3=strtok(NULL,"||");
          char* cBal4=strtok(NULL,"||");
          char* cBal5=strtok(NULL,"||");
          char* cTotBal=strtok(NULL,"||");
          sscanf(cTotBal, "%f",&bank.totBal );
          printf("The balance is:%f\n",bank.totBal);
        }
      }
  }
int main()
{
  int choice;
  bank.attempts=0;
  printf("\n<---Welcome to SDM Bank of India--->\n");
  if(authenticate()){
    printf("Enter the choice\n1-Deposit money 2-Withdraw money 3-Mini statement 4-Balance
enquiry 0-Exit\n");
    scanf("%d",&choice);
    switch(choice)
    {
    case 1:depositMoney();
```

```
        break;
    case 2:withDrawal();
          break;
    case 3:miniStatement();
          break;
    case 4:balanceEnquiry();
          break;
    case 0:
    printf("\n\n\nThank you for Banking with us..\n<--SDM Bank of INDIA-->\n");
    return 0;
    }
  }
  else{
    printf("\n\n\nThank you for Banking with us..\n<--SDM Bank of INDIA-->\n");
    return 0;
  }

   return 0;
}
```

6. Output Snapshot :

Admin Module

6.1. Home Screen :

```
Emter the choice
1-Create account 2-View accounts 3-Unblock 0-Exit
```

6.2 Create new user:

```
Emter the choice
1-Create account 2-View accounts 3-Unblock 0-Exit
1
Enter the name
Aditya
Enter the amount to be added
5000
```

6.3 Unblock the user:

```
Emter the choice
1-Create account 2-View accounts 3-Unblock 0-Exit
3
Enter the card number to be unblocked
1000029314
```

User module:

6.4 Authentication:

```
<---Welcome to SDM Bank of India--->
Enter the card number
1000007925
2337  Enter the Password
2337
Authenticated
Enter the choice
1-Deposit money 2-Withdraw money 3-Mini statement 4-Balance enquiry 0-Exit
```

6.5 Deposit:

```
<---Welcome to SDM Bank of India--->
Enter the card number
1000007925
2337  Enter the Password
2337
Authenticated
Enter the choice
1-Deposit money 2-Withdraw money 3-Mini statement 4-Balance enquiry 0-Exit
1
Enter the amount to be deposited
8000
Balance is:13000.000000
```

6.6 Withdraw

```
<---Welcome to SDM Bank of India--->
Enter the card number
1000007925
2337  Enter the Password
2337
Authenticated
Enter the choice
1-Deposit money 2-Withdraw money 3-Mini statement 4-Balance enquiry 0-Exit
2
Enter the amount to be withdrawn
5000
Balance is:8000.000000
```

6.7 Mini statement

```
<---Welcome to SDM Bank of India--->
Enter the card number
1000007925
2337  Enter the Password
2337
Authenticated
Enter the choice
1-Deposit money 2-Withdraw money 3-Mini statement 4-Balance enquiry 0-Exit
3
Mini statement
Credit/Debit||Amount
Credit  ||  0.000000
Credit  ||  0.000000
Credit  ||  5000.000000
Credit  ||  8000.000000
Debit  ||  -5000.000000
The balance is:8000.000000
```

6.8 Balance enquiry

```
<---Welcome to SDM Bank of India--->
Enter the card number
1000007925
2337  Enter the Password
2337
Authenticated
Enter the choice
1-Deposit money 2-Withdraw money 3-Mini statement 4-Balance enquiry 0-Exit
4
The balance is:8000.000000
```