SDM College of Engineering and Technology

Dhavalagiri, Dharwad-580 002. Karnataka State. India.

Email: principal@sdmcet.ac.in, cse.sdmcet@gmail.com

Ph: 0836-2447465/ 2448327 Fax: 0836-2464638 Website: sdmcet.ac.in

Department
Of
COMPUTER SCIENCE AND ENGINEERING

LAB-WORK REPORT

[21UCSL502-Database Management & Design Patterns]

Odd Semester: Sep-Jan-2023

Course Teacher: Prof. Anand D Vaidya

Topic: C-Program for Automated Teller Machine

2023

Submitted By

NAME	USN
PRAJWAL GANESH PAWAR	2SD21CS064
KIRANKUMAR R DEVARKONDI	2SD21CS045

5th Semester B division

Contents

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 processing, and dispenses cash to the customer. We assume that an ATM need not operate independently of the network.

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.

General Description

2.1 Product Perspective

The ATM network doesn't work independently. It has to work together with the computers/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 keeping 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 minimum cash in the ATM to permit a transaction t is the total fund in the ATM at start of day

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 is 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.

- 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

- 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.

–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 authorization.

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.

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 predefined 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.

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)

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.

  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.

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 records

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

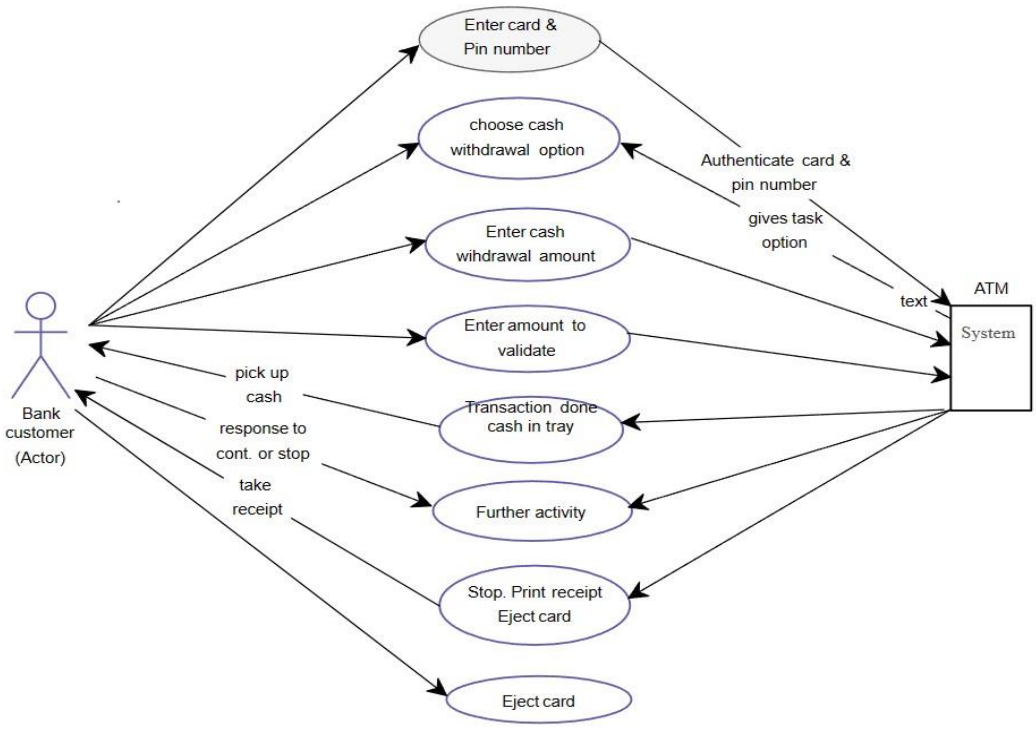
  If the amount exceeds the limit, the transaction will fail.

Functional requirement 10

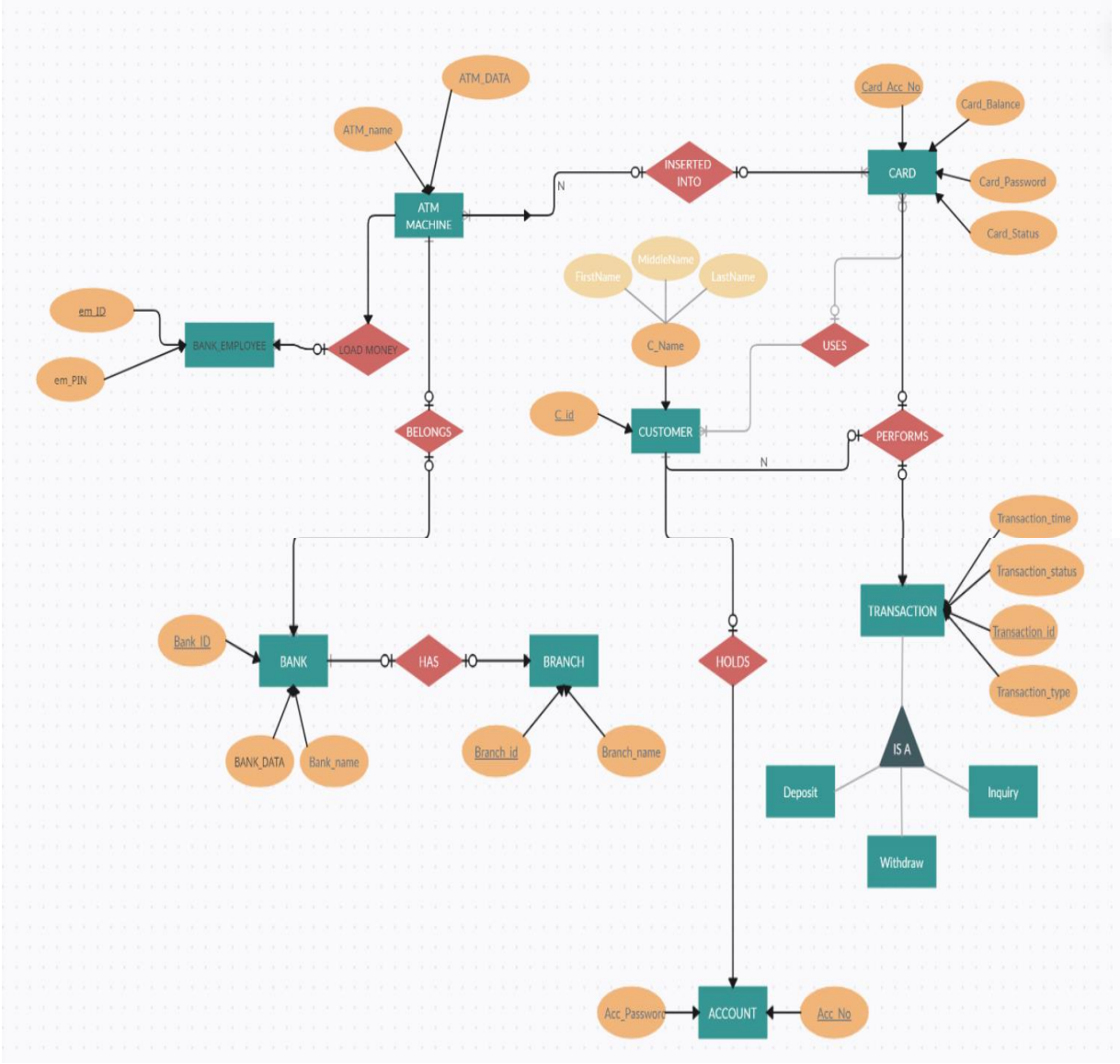
- Description

  The bank only provides security for their own computer and their own software.

USE CASE DIAGRAM



ER model


ATM_DATA
ATM_name
ATM MACHINE
INSERTED INTO
N
Card_Acc_No
Card_Balance
CARD
Card_Password
Card_Status
MiddleName
FirstName
LastName
C_Name
USES
em_ID
em_PIN
BANK_EMPLOYEE
LOAD MONEY
BELONGS
C_id
CUSTOMER
PERFORMS
N
Transaction_time
Transaction_status
TRANSACTION
Transaction_id
Transaction_type
Bank_ID
BANK
HAS
BRANCH
HOLDS
IS A
BANK_DATA
Bank_name
Branch_id
Branch_name
Deposit
Inquiry
Withdraw
Acc_Password
ACCOUNT
Acc_No

PROGRAM i) BANK

```
//Bank code
#include <stdio.h>
#include <stdlib.h>
#include <string.h>
#include <time.h>

struct Customer {
    char name[50];
    int account_number;
    double balance;
    int passlast;
};

int generateAccountNumber() {
    return rand() % 90000000 + 10000000;
}

int generate4DigitRandomNumber() {
    return rand() % 9000 + 1000; // Generates a random number between 1000 and 9999
}

void createCustomer() {
    struct Customer customer;
    FILE *file;

    printf("Enter customer name: ");
    scanf("%s", customer.name);

    customer.account_number = generateAccountNumber();
    customer.balance = 0.0;
    customer.passlast=generate4DigitRandomNumber();
```

```
    file = fopen("bank_data.txt", "a");
    if (file == NULL) {
        perror("Error opening file");
        exit(1);
    }

    fprintf(file, "%s %d %.2lf %d\n", customer.name, customer.account_number, customer.balance,
customer.passlast);

    fclose(file);

    printf("Customer created successfully. Account number: %d\n", customer.account_number);
}

void displayCustomer(int account_number) {
    struct Customer customer;
    FILE *file;

    file = fopen("bank_data.txt", "r");
    if (file == NULL) {
        perror("Error opening file");
        exit(1);
    }

    int found = 0;
    while (fscanf(file, "%s %d %lf %d", customer.name, &customer.account_number,
&customer.balance, &customer.passlast) != EOF) {
        if (customer.account_number == account_number) {
            printf("Customer Name: %s\n", customer.name);
            printf("Account Number: %d\n", customer.account_number);
            printf("Account Balance: %.2lf\n", customer.balance);
            found = 1;
            break;
```

```
        }
    }

    if (!found) {
        printf("Customer not found with account number: %d\n", account_number);
    }

    fclose(file);
}

void adminLogin() {
    int EnteredPIN1, choice;
    char EnteredID1[25];
    printf("\nGive login of an authorized employee of the bank\n");
    printf("Employee's ID\n");
    scanf("%s", EnteredID1);

    int c;
    while ((c = getchar()) != '\n' && c != EOF);
    printf("Employee's PIN\n");
    scanf("%d", &EnteredPIN1);

    FILE *file;
    file = fopen("data.txt", "r");
    if (file == NULL) {
        printf("File not found or unable to open the file.\n");
        return;
    }

    char ID1[25];
    int PIN1;
    fscanf(file, "ID1: %s\nPIN1: %d", ID1, &PIN1);
    fclose(file);
```

```
if (strcmp(ID1, EnteredID1) == 0 && PIN1 ==EnteredPIN1 ) {
   printf("\nEmployee LOGIN success\n");
} else {
   printf("Wrong ID or Wrong PIN of the employee !!!\n");
   return;
}

int cashLoadAmount = 0;
int atmBalance = 0;

while (1) {
   printf("\n1: Cash Load\n2: Check ATM Balance\n3: Logout\n");
   scanf("%d", &choice);

   switch (choice) {
      case 1:
         // Implement Cash Load functionality
         printf("Enter cash amount to load into ATM: ");
         scanf("%d", &cashLoadAmount);

         file = fopen("AtmBalance.txt", "r");
         if (file == NULL) {
            printf("ATM balance data not found or unable to open the file.\n");
            return;
         }

         fscanf(file, "ATM Balance : %d", &atmBalance);
         fclose(file);

         atmBalance += cashLoadAmount;

         file = fopen("AtmBalance.txt", "w");
```

```
            if (file == NULL) {
                printf("Error updating ATM balance.\n");
                return;
            }

            fprintf(file, "ATM Balance : %d", atmBalance);
            fclose(file);

            printf("Cash loaded successfully. Updated ATM balance: %d\n", atmBalance);
            break;

        case 2:
            // Check ATM Balance functionality
            file = fopen("AtmBalance.txt", "r");
            if (file == NULL) {
                printf("ATM balance data not found or unable to open the file.\n");
                return;
            }

            fscanf(file, "ATM Balance : %d", &atmBalance);
            fclose(file);

            if (atmBalance < 100000) {
                printf("ATM balance is less than 100000.\nThe ATM facility is currently Unavailable
!!!");
            } else {
                printf("ATM balance is sufficient: %d\n", atmBalance);
            }
            break;

        case 3:
            return;

        default:
```

```
            printf("Invalid choice. Please try again.\n");
        }
    }
}

void reviveBlocked_Account(int blocked_acc){
    return;
}
int main() {
    srand(time(NULL));

    int choice;
    int account_number, blocked_acc;

    while (1) {
        printf("\nBanking System Menu:\n");
        printf("1. Create Customer\n");
        printf("2. Display Customer Information\n");
        printf("3. For BANK ACCESS\n");
        printf("4. Revive Account\n");
        printf("5. Exit\n");
        printf("Enter your choice: ");
        scanf("%d", &choice);

        switch (choice) {
            case 1:
                createCustomer();
                break;
            case 2:
                printf("Enter account number: ");
                scanf("%d", &account_number);
                displayCustomer(account_number);
                break;
```

```
            case 3:
                adminLogin();
                break;
            case 4 :
                printf("Enter your Blocked account Number\n");
                scanf("%d",&blocked_acc);
                reviveBlocked_Account(blocked_acc);
                printf("Your Account is Unblocked\n");
                break;

            case 5:
                printf("Exiting the program.\n");
                exit(0);
            default:
                printf("Invalid choice. Please try again.\n");
        }
    }
    return 0;
}
```

PROGRAM ii) ATM

```
//ATM machine code
#include <stdio.h>
#include <stdlib.h>
#include <string.h>
#include <time.h>
#include <stdbool.h>

struct Customer {
    char name[50];
    int account_number;
    double balance;
    int passlast;
    int attempts;
};

struct transaction{
    int acc_num;
    double widraw_amt;
    char* currenttime;
};

char* getCurrentTime() {
    time_t currentTime;
    struct tm *timeInfo;
    static char timeString[100]; // Static buffer for the time string

    time(&currentTime); // Get the current time

    timeInfo = localtime(&currentTime); // Convert the current time to a more human-readable format

    // Format the current time as a string
```

```
    strftime(timeString, sizeof(timeString), "%Y-%m-%d_%H:%M:%S", timeInfo);

    return timeString;
}

//Function to Enter transaction details

void store_transaction(struct transaction t1,int account_nom,double deposit_amot){
    int account_no=account_nom;
    double deposit_amt=deposit_amot;

    char *currtame=getCurrentTime();
    t1.acc_num=account_no;
    t1.widraw_amt=deposit_amt;

    FILE *tranfile;
    tranfile=fopen("transactions.txt","a");

    if(tranfile==NULL){
        perror("Error Opening the File\n");
        exit(1);
    }

    fprintf(tranfile, "%d %.2lf %s\n",t1.acc_num,t1.widraw_amt,currtame);
    fclose(tranfile);
}

//Access transaction details

void transaction_details(int account_no) {
    struct transaction t1;
    int account_No=account_no;
    FILE *filedet;
```

```
    filedet=fopen("transactions.txt","r");

    if(filedet==NULL){
        perror("File is Empty\n");
        exit(0);
    }
    while(fscanf(filedet, "%d %lf %s",&t1.acc_num,&t1.widraw_amt,t1.currenttime)!=EOF){
            if(account_No==t1.acc_num){
                //printf("Account No : %d \n",t1.acc_num);
                printf("Transaction Amount : %.2lf \n",t1.widraw_amt);
                printf("Time of Transaction : %s \n",t1.currenttime);
            }
    }
    fclose(filedet);
}

void updateBalance(int account_number, double new_balance) {
    FILE *file, *tempFile;
    struct Customer customer;

    file = fopen("bank_data.txt", "r");
    if (file == NULL) {
        perror("Error opening file");
        exit(1);
    }

    tempFile = fopen("temp_bank_data.txt", "w");
    if (tempFile == NULL) {
        perror("Error creating temporary file");
        fclose(file);
        exit(1);
    }
```

```
    while (fscanf(file, "%s %d %lf %d", customer.name, &customer.account_number,
&customer.balance, &customer.passlast) != EOF) {
        if (customer.account_number == account_number) {
            customer.balance = new_balance;
        }
        fprintf(tempFile, "%s %d %.2lf %d\n", customer.name, customer.account_number,
customer.balance, customer.passlast);
    }
    fclose(file);
    fclose(tempFile);

    remove("bank_data.txt");
    rename("temp_bank_data.txt", "bank_data.txt");
}

void displayCustomer(int account_number) {
    FILE *file;
    struct Customer customer;

    file = fopen("bank_data.txt", "r");
    if (file == NULL) {
        perror("Error opening file");
        exit(1);
    }
    int found = 0;
    while (fscanf(file, "%s %d %lf", customer.name, &customer.account_number,
&customer.balance) != EOF) {
        if (customer.account_number == account_number) {
            printf("Customer Name: %s\n", customer.name);
            printf("Account Number: %d\n", customer.account_number);
            printf("Account Balance: %.2lf\n", customer.balance);
            found = 1;
            break;
        }
```

```
    }
    if (!found) {
        printf("Customer not found with account number: %d\n", account_number);
    }
    fclose(file);
}

void updateATMBalance(int deposit_amount) {
    int atmBalance;
    FILE *file = fopen("AtmBalance.txt", "r");

    if (file == NULL) {
        printf("ATM balance data not found or unable to open the file.\n");
        exit(1);
    }

    fscanf(file, "ATM Balance : %d", &atmBalance);
    fclose(file);

    int newAtmBal = atmBalance + deposit_amount;

    file = fopen("AtmBalance.txt", "w");
    if (file == NULL) {
        printf("Error updating ATM balance.\n");
        exit(1);
    }

    fprintf(file, "ATM Balance : %d", newAtmBal);
    fclose(file);
}

// these two functions helps to keep track of the begin of new day
int getStoredDay() {
```

```
    int storedDay = 0;
    FILE *file = fopen("stored_day.txt", "r");
    if (file != NULL) {
        fscanf(file, "%d", &storedDay);
        fclose(file);
    }
    return storedDay;
}

void updateStoredDay(int day) {
    FILE *file = fopen("stored_day.txt", "w");
    if (file != NULL) {
        fprintf(file, "%d", day);
        fclose(file);
    }
}

bool CheckATMDailyLimit(int wt_amount) {
    static int temp = 0; // Static variable to keep track of total withdrawal amount

    time_t currentTime = time(NULL);  // Get the current time
    struct tm *localTime = localtime(&currentTime);

    // Check if it's a new day, and reset the total withdrawal amount if necessary
    if (localTime->tm_mday != getStoredDay()) {
        temp = 0;
        updateStoredDay(localTime->tm_mday);
    }

    wt_amount = abs(wt_amount); // Take the absolute value of withdrawal amount
    temp = temp + wt_amount;

    if (temp >= 50000) {
```

```
        temp = temp - wt_amount;
        return false;
    } else {
        return true;
    }
}

int main() {

    int account_number;
    int logged_in = 0;
    int atmBalance;
    struct Customer customer;
    struct transaction t1;

    FILE *file = fopen("AtmBalance.txt", "r");
    if (file == NULL) {
        printf("ATM balance data not found or unable to open the file.\n");
        return 1;
    }

    fscanf(file, "ATM Balance : %d", &atmBalance);
    fclose(file);

    while (1) {
        if (!logged_in) {
            printf("\nWELCOME\nPLEASE INSERT YOUR CARD\n");
            char enter;
            int passgod;
            scanf("%c", &enter);
            printf("Enter your account number (0 to exit):\n");
            scanf("%d", &account_number);
            printf("Enter your 4 Digit Password\n");
```

```
        scanf("%d",&passgod);

        if (account_number == 0) {
            printf("\nExiting the program.\n");
            break;
        }

        FILE *file = fopen("bank_data.txt", "r");
        int found = 0;
        while (fscanf(file, "%s %d %lf %d", customer.name, &customer.account_number,
&customer.balance, &customer.passlast) != EOF) {
            if (customer.account_number == account_number && customer.passlast==passgod) {
                found = 1;
                break;
            }
        }
        fclose(file);

        if (found) {
            printf("Login successful.\n");
            logged_in = 1;
            customer.attempts = 0 ;
        }

        else if (customer.passlast != passgod) {
            printf("\nBAD PASSWORD, card is ejected !\n");
            logged_in = 0;
            customer.attempts ++ ;

            if (customer.attempts >= 3) {
                printf("!! ACCOUNT BLOCKED !!\n");
                logged_in = 0;
```

```
            // You might want to add additional logic here to handle the blocked account, such as updating a file/database.
            }
        }
        else {
            printf("\nBAD ACCOUNT, card is ejected !\n");
            logged_in = 0;
        }

    } else {
        printf("\nATM Menu:\n");
        printf("1. Display Customer Information\n");
        printf("2. Deposit\n");
        printf("3. Withdraw\n");
        printf("4. Check Balance\n");
        printf("5. Mini Statement\n");
        printf("6. Logout\n");
        printf("Enter your choice:");

        int choice;
        scanf("%d", &choice);

        switch (choice) {
            case 1:
                displayCustomer(account_number);
                break;
            case 2:
                printf("\nEnter the deposit amount: ");
                double deposit_amount;
                scanf("%lf", &deposit_amount);
                if (deposit_amount <= 0) {
                    printf("\nInvalid deposit amount. Please enter a positive value.\n");
                } else {
                    struct Customer customer;
```

```
                FILE *file = fopen("bank_data.txt", "r");
                int found = 0;
                double new_balance = 0.0;
                while (fscanf(file, "%s %d %lf", customer.name, &customer.account_number,
&customer.balance) != EOF) {
                    if (customer.account_number == account_number) {
                        found = 1;
                        new_balance = customer.balance + deposit_amount;
                        break;
                    }
                }
                fclose(file);
                if (found) {
                    updateBalance(account_number, new_balance);
                    updateATMBalance(deposit_amount);
                    store_transaction(t1,account_number,deposit_amount);
                    printf("\nDeposit successful. New balance: %.2lf\n", new_balance);
                } else {
                    printf("\nAccount not found. Logout and try again.\n");
                }
            }
            break;
        case 3:

            if (atmBalance <= 50000) {
                printf("\nInsufficient Balance in ATM MACHINE !!!\n Can't WITHDRAW, try Later\n");
                break;
            }

            printf("\nEnter the Withdrawal amount: ");
            int wt_amount;
            scanf("%d", &wt_amount);
            if (wt_amount <= 0) {
                printf("\nInvalid withdraw amount. Please enter a positive value.\n");
```

```
            break;
        }
        if( wt_amount >= 50000) {
            printf("\ncan't withdraw more than 50000 at a time\n");
            break;
        }

        bool b = CheckATMDailyLimit(wt_amount);
        if(!b) {
            printf("\nDaily ATM transaction limit has been exceeded\n");
            printf("TRY LATER !!\n");
            break;
        }

        wt_amount = -1 * wt_amount;
        struct Customer customer;
        FILE *file = fopen("bank_data.txt", "r");
        int found = 0;
        double new_balance = 0.0;
        while (fscanf(file, "%s %d %lf", customer.name, &customer.account_number,
&customer.balance) != EOF) {
            if (customer.account_number == account_number ) {
                found = 1;
                new_balance = customer.balance + wt_amount;
                break;
            }
        }
        fclose(file);

        if (customer.balance < (-1)*wt_amount) {
            printf("Insufficient balance for withdrawal.\n");
            break;
        }
```

```
            if (found) {
                updateBalance(account_number, new_balance);
                updateATMBalance(wt_amount);
                store_transaction(t1,account_number,wt_amount);
                printf("\nWithdrawal successful. New balance: %.2lf\n", new_balance);
            } else {
                printf("Account not found. Logout and try again.\n");
            }
            break;
        case 4:
            displayCustomer(account_number);
            break;
        case 5:
            // Mini Statement functionality can be added here
            transaction_details(account_number);
            break;
        case 6:
            printf("Logout successful.\n");
            logged_in = 0;
            break;
        default:
            printf("Invalid choice. Please try again.\n");
      }
    }
  }

  return 0;
}
```

OUTPUT: of Bank

Account Creation:

```
Banking System Menu:
1. Create Customer
2. Display Customer Information
3. For BANK ACCESS
4. Revive Account
5. Exit
Enter your choice:
```

```
Enter your choice: 1
Enter customer name: prajwal

Customer created successfully. Account number: 10026916
Password : 9383
```

Customer Information:

```
Enter your choice: 2
Enter account number: 10026916
Customer Name: prajwal
Account Number: 10026916
Account Balance: 0.00
```

Bank Access :

```
Enter your choice: 3

Give login of an authorized employee of the bank
Employee's ID
CNR245
Employee's PIN
7821

Employee LOGIN success

1: Cash Load
2: Check ATM Balance
3: Logout
```

AdminID&PIN.txt

```
1   ID1: CNR245
2   PIN1: 7821
```

Accounts in File:

output > bank_data.txt

```
1   projwal 101 125139.00 3921
2   raikar 1001 0.00 2698
3   shreyas 10017126 0.00 5889
4   pranav 10024225 0.00 2618
5   skldfj 10020904 0.00 4438
6   kiran 10026263 0.00 4389
7   prajwal 10026916 0.00 9383
```

OUTPUT: of ATM

Authentication & Menu:

```
WELCOME
PLEASE INSERT YOUR CARD

Enter your account number (0 to exit):
10026916
Enter your 4 Digit Password
9383
Login successful.

ATM Menu:
1. Display Customer Information
2. Deposit
3. Withdraw
4. Check Balance
5. Mini Statement
6. Logout
Enter your choice:
```

Deposit :

```
Enter your choice:2

Enter the deposit amount: 60000

Deposit successful. New balance: 60000.00
```

Balance Enquiry :

```
Enter your choice:1
Customer Name: prajwal
Account Number: 10026916
Account Balance: 60000.00
```

Withdraw :

```
Enter your choice:3

Enter the Withdrawal amount: 9000

Withdrawal successful. New balance: 11000.00

ATM Menu:
1. Display Customer Information
2. Deposit
3. Withdraw
4. Check Balance
5. Mini Statement
6. Logout
Enter your choice:4
Customer Name: prajwal
Account Number: 10026916
Account Balance: 11000.00
```

Daily Withdraw Limit:

```
Enter your choice:3

Enter the Withdrawal amount: 40000

Withdrawal successful. New balance: 20000.00

ATM Menu:
1. Display Customer Information
2. Deposit
3. Withdraw
4. Check Balance
5. Mini Statement
6. Logout
Enter your choice:3

Enter the Withdrawal amount: 15000

Daily ATM transaction limit has been exceeded
TRY LATER !!
```

Mini Statement :