University of Petroleum and Energy Studies

School of Computer Science

Project Report

Topic: ATM Simulator (with security features)

Submitted by:

Name:Prakhar Goyal

SAP ID:590027648

Batch-59

Course Code: CSEG1041

Course:Programming in C

Semester-1

Academic Year: 2025–2026

Program:B.Tech(Computer Science and Engineering)

Submitted to:

Instructor Name:Dr.Prashant Trivedi

Abstract

This project on Automated Teller Machine(ATM) is made using C Programming language.The primary and main goal of this project was to create a hands-on and practical demonstration on almost all the concepts in the 6 units taught to me in the course of 'Programming in C' in my first semester at UPES.

This is a simulation of real ATM,it allows a single user to perform various everyday needed operations like ,logging in via a pin(with max 3 wrong attempts for Security feature),checking account balance,depositing and withdrawing money,changing atm pin,changing username,printing receipt and mini transaction history (on screen) using file handling concepts.It also allows to reset account for improving usability of project and also making testing easier during development.

I tried to write clean code and using modular programming approach that is easy to understand.The project helped in practicing important concepts of loops, structures, functions, file handling.

Builiding this project on ATM simulator made me understand how actual ATM machines work in real life.This project made me realized that how combining important concepts and simple logic building approach can be used for making everyday tools.This ATM project gave me the confidence that how a beginner level idea can grow into meaningful things with patience and consistency.

Problem Definition

In today's life,ATMs have become an important tool in providing everyday banking operations facility to people so that they can access and handle their money conveniently. Normally,understanding the concept that how an ATM works is commonly unnoticed.Every time we check balance,deposit,withdraw money,a series of invisible hidden steps happen instatntly and accurately inside the ATM.This project tries to bring those hidden steps into open picture by creating and making a small and easy to use ATM simulator in C.

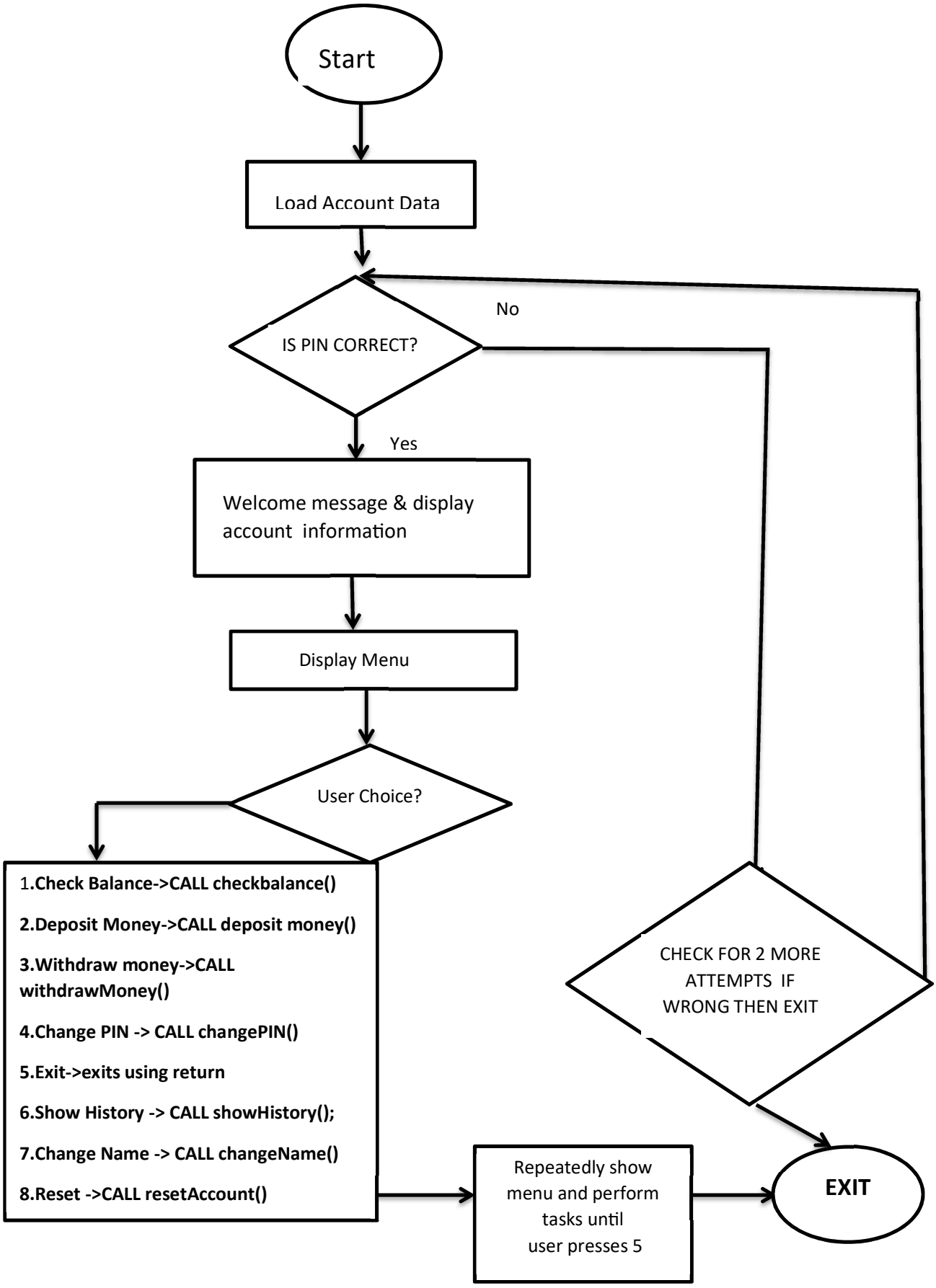
To solve this we should focus on creating a simpler and easy to use ATM simulator that works like a actual real life ATM.It should allow user to login successfully using a PIN ,carry operations like depositing, withdrawing money, changing pin, username , displaying mini transaction history on screen, and resetting account for easy testing and development of this code on our systems. It also allows user for only three PIN input attempts for Security features

The challenge is to make these operations/transactions smooth,cleaner and error free feeling like real life ATM.

The purpose of this project is not just to write a program,it's to demonstrate how day to day baking operations work using the concepts we learned in class lectures like structures, loops,file handling, functions,conditional statements.

Flowchart and Algorithm

Flowchart(login to menu navigation):



Algorithm:

1.Start

2.load the saved account details from the file if no file exist , creating a defult account with preset default values

3.Display a welcome and enetering into the while loop

4.initialiazing the no of attempts counter to 0

5.Repeat until the correct PIN is entered by user or attempts reach max i.e. 3

- Asking user to enter 4 digit pin
- If input is not number then clearing it and asking again
- If entered pin matches with stored the PIN:
    - Display a welcome message and showing account details
    - Break out of the loop
- Otherwise,we increase the no of wrong attempts counter and tell the no of attempts left

6.If the user enters the **wrong PIN three times** the a display account locked message and **End** the program here.

7.If it is a successful login,the main menu is displayed with the options:

1. Check Balance
2. Deposit Money
3. Withdraw Money
4. Change PIN
5. Exit
6. Transaction History
7. Change Account Holder Name
8. Reset Account

8.Repeat until the user chooses Exit (option 5):

a) Printing the menu and Reading the users menu choice
b) Operating and calling functions as needed:
   - **checkBalance():** To show balance and give low-balance warning if balance is low
   - **depositMoney():**It validates the amount to be deposited ,update balance,optionally print of receipt, and save transaction history
   - **withdrawMoney():**It validates amount to be withdrawn,ensure sufficient funds,update balance, optional printing of receipt, and save the transcation in the history file
   - **changePIN():** It asks the user for a new PIN twice and update if both are same and valid.
   - **Exit:** Not a function but a switch case and option in menu which exits the prog by return command
   - **showHistory():**Display all the recorded transactions
   - **changeName():**it changes the the account holder name and accept new name and save it
   - **resetAccount():** It is used mainly for testing the code for different input and outputs and restoring default account values and it also clears transactions recorded(history).
   - **Invalid Choice:** Not a function but a switch case Display an error message

9.When the user selects Exit ,display a thank you message and return.

10.End

Function-wise Algorithm for the important functions:

Algorithm of checkBalance() function:

1. Start
2. Display the current balance stored in memory
3. Dispalying a warning message if balance is less than 1000
4. End

Algorithm of depositMoney() function:

1. Start
2. Asking the user to enter the amount to be deposited
3. Checking that the amount should be >0 if not then displaying error message.
4. If it is a valid amount
    - Then updating the balance
    - Asking user for receipt
    - Printing receipt if user type y or Y i.e (yes)
    - Adding the transaction details to history file using file handling tools
5. Return the structure acc
6. End

Algorithm of withdrawMoney() function:

1. Start
2. Asking the user to enter the amount to be withdrawn
3. Checking that the amount should be >0 if not then displaying error message
4. If it is a valid amount
    - Then updating the balance
    - Asking user for receipt
    - Printing receipt if user type y or Y i.e (yes)
    - Adding the transaction details to history file using file handling tools
5. Return the structure ‘acc’
6. End

Algorithm of changePIN() function:

1. Start
2. Asking user to enter new pin
3. Asking him to re enter the new pin for re-confirming
4. Checking if the new pin entered is 4 digit and the two PINs matched
    - If matched then updating the PIN
    - Else displaying the message that PIN not changed
5. Return the ‘acc, structure
6. End

Algorithm of changeName() function:

1. Start
2. Asking the user to enter a new account holder name and checking that a name is entered (i.e. there is no empty input)
3. If no input than displaying appropriate message and returning the acc structure
4. If a valid input then Replacing the store name with the new user entered name
5. Saving the updated name in account file and returning it
6. End

Algorithm of addHistory() function:

1. Start
2. It receives the transaction and stores in a string pointer
3. Opening the history file in append mode using file pointer
4. If file does not exit then displaying error message
5. Else printing the transaction entry in the file
6. Closing the file
7. End

Algorithm of showHistory() function:

1. Start
2. Opening the history file in read mode
3. Checking if exists or not ,if doesnot exist then displaying error message
4. If file exists then printing the file contents on screen and tracking using empty variable
5. If nothing was printed then printing that No transactions recorded yet
6. Closing the file
7. End

Algorithm of resetAccount():

1. Start
2. Opening the account.txt in write mode
3. If file does not exist then printing error message
4. If file exists then Setting the balance,pin and name, bank name,branch name,account type to default fixed values

5.Closing the account.txt file

6.No deleting the data of old transactions in history.txt file by opening it in write mode

7.Closing this history file

8.Informing the user that account has reset

9.End

Algorithm of saveToFile() Function:

(This function saves the updated account details back into account.txt)

1. Start
2. Opening the account.txt in write mode
3. If the file cannot be opened,displaying an error message and returning
4. If the file opens
    - Writing the details of the account in the file using fprintf
5. Closing the file
6. End

Algorithm of loadFromFile() Function:

(This function loads the saved content from account.txt)

1. Start
2. Opening the account.txt in read mode
3. If the file does not exist ,then creating an account with default preset values nd saving it using saveToFile()
4. If the file opens
    - Read the details of the account one by one in the file using fgets and fscanf
5. Store these in account structure
6. Closing the account.txt file
7. End

IMPLEMENTATION

This ATM simulator is implemented using the C language. I used structured and modular programming approach to make the code easier to understand.Instead of writing the entire code into the main I divided the code into blocks i.e. functions so that they can be reused where required which is better than typing the same code again.The entire program was divided into multiple functios-each responsible for different operations like cecking account balance,depositing and withdrawing money,changing the PIN,saving transactions,and displaying transaction history.I kept the code oraganized,easier to read and debug.

To make the code feel like real life ATM tool Simulation the code stores all account details to 2 files:
account.txt ->this stores the account holder's name,PIN,balance, bank name,branch name and account type

history.txt ->This stores the transactions history to show on user request.It is like a mini statement

I used structure to hold account data:

```
struct account
{
   char name[50];        //to store acc holder name
   int pin;              //to store the pin
   float balance;        //to store account balance
   char bankName[50];    //to store bank name
   char branch[50];      //to store branch
   char type[20];        //to store the account type
};
```

On Starting the program, the **loadFromFile()** function loads the saved account from account.txt. If the file is missing ,my program automatically creates a default account with preset default values using **saveToFile()** function.

After loading the data from the respective files,the login process starts now.First there is PIN checking with 3 max attempts for security purposes and (allowing 4 digit PIN) only.

Only if the PIN is correct then only the welcome message with user details and main menu is displayed which is important feature for security.

Function-wise Implementation of ATM Simulator:

- **main()**
  The main function is the central controller of the ATM simulator.It loads the account.txt data (or creates default account if account.txt is missing).The main handles user login and PIN verification with max 3 attempts security feature and displays account details,menu to process operations for each choice and carrying operations using switch case and loops until user exit.It ensures modular and organized processing.
  Code of main():

```
int main()
{
   struct account user;
   user=loadFromFile();   // to load saved account (or create new)

   int enteredPin,choice;
   int attempts=0;

   printf("\n- Welcome to State Bank ATM -\n");

   while (1)
```

```
{
   printf("Enter your 4 digit pin: ");
   if (scanf("%d", &enteredPin) != 1)
   {
   printf("Invalid input! Numbers only.\n");
    while (getchar()!='\n'); // to clear the buffer
   continue;
   }

   if (enteredPin==user.pin)
   {
      //showing user and acc information
      printf("Welcome, %s\n",user.name);
      printf("Login successful.\n");
      printf("Bank:%s\n",user.bankName);
      printf("Branch:%s\n",user.branch);
      printf("Account Type:%s\n",user.type);

      // low balance warning
      if (user.balance < 1000)
         printf("\n WARNING:Your balance is very low.Please deposit soon.\n");
      break;
   }
   else
   {
      attempts++;
      printf("Wrong pin entered...Attempts left: %d\n",3-attempts);
   }
   if(attempts==3) //exiting if more than 3 wrong pins entered
   {
      printf("You tried wrong pin as too many times.\n");
      printf("Account locked for safety.\n");
      return 0;
   }
}

// printing Menu and using switch case to execute the menu choices
do {
   printf("\n-- MENU --\n");
   printf("1.Check Balance\n");
   printf("2.Deposit Money\n");
   printf("3.Withdraw Money\n");
   printf("4.Change PIN\n");
   printf("5.Exit\n");
   printf("6.Transaction History\n");
   printf("7.Change Account Holder Name\n");
   printf("8.Reset Account\n");
   printf("Enter choice: ");
   scanf("%d", &choice);   //taking input from user
   while(getchar()!='\n');   // to clear the buffer
```

```
    switch (choice)
    {
        case 1:
            checkBalance(user);
            break;

        case 2:
            user=depositMoney(user);
            saveToFile(user);
            break;

        case 3:
            user = withdrawMoney(user);
            saveToFile(user);
            break;

        case 4:
            user = changePIN(user);
            saveToFile(user);
            break;

        case 5:
            printf("\nThank you for using the ATM.\n");
            break;
        case 6:
              showHistory();
               break;// to avoid fall through condition
        case 7:
              user=changeName(user);
              saveToFile(user);
              break;

        case 8:  user=resetAccount(user);
               saveToFile(user);
              break;

        default:
            printf("Invalid option. Try again.\n");
    }

} while (choice != 5);

return 0;
}
```

(P.T.O)

- **checkBalance(struct account acc)**

  This function displays current account balance.Thsi also shows a balance warning if balance is less than 1000Rs.
  code of checkBalance:

```
void checkBalance(struct account acc)
{
   printf("\nYour balance: Rs %.2f\n", acc.balance);//printing acc balance
   if (acc.balance < 1000)
          { printf("\n WARNING:Your balance is very low.Please deposit
soon.\n");
          }
}
```

- **depositMoney(struct account acc)**

  This function allow user to deposit money into their account.It also checks for valid deposit amount.It updates the balance and on asking user prints a receipt and records the transaction in history.txt file. And returns the updated account structure.
  <u>code of depositMoney(struct account acc):</u>

```
struct account depositMoney(struct account acc)
 {
    float amt;
    printf("Enter amount to deposit: ");  //taking amount to be deposited
    scanf("%f",&amt);

    if(amt<=0)  //if amt less than or equal to 0
    {
       printf("Invalid amount.\n");
    }
    else
    {
       acc.balance+=amt;    //updating the balance
       printf("Deposited Rs %.2f\n", amt);
       while (getchar() !='\n');  //to clear the buffer
       char ch;
       printf("Do you want a receipt?(y/n):"); //asking for receipt printing
       scanf(" %c",&ch);
       if(ch=='y'|| ch=='Y')
       {
          printf("\n----Receipt-----\n");
          printf("Transaction :deposited\n");
```

```
        printf("Amount: Rs %.2f\n", amt);
        printf("New balance:Rs %.2f\n",acc.balance);

      }
       //adding transaction to history
      addHistory("Deposited:");
      FILE *fp =fopen("history.txt", "a");    //opening file in append mode to
add transaction info
      fprintf(fp, " Rs %.2f | Balance: Rs %.2f\n", amt, acc.balance);
      fclose(fp);   //closing the file

   }
   return acc;
}
```

- **withdrawMoney(struct account acc)**
  This function allows user to withdraw money and checks for valid money withdrawn.It also displays warning if entered money is greater than balance.It also update the balance and on user request print a receipt.It also saves the transaction in history.txt.And returns the updated account structure.
  code of withdrawMoney(struct account acc):

```
struct account withdrawMoney(struct account acc)
{
   float amt;
   printf("Enter amount to withdraw: ");
   scanf("%f", &amt);   //inputting withdrawl amount
   while (getchar() != '\n');
   if (amt<=0)
   {
      printf("Invalid amount.\n");
   }
   else if(amt>acc.balance)
   {   //if amount in account is less than withdrawl request
      printf("Not enough balance.\n");
   }
   else
   {
      acc.balance-=amt;    //updating the balance
      printf("Please collect Rs %.2f\n", amt);
       char ch;
      printf("Do you want a receipt?(y/n):");  //asking for receipt
      scanf(" %c",&ch);
```

```
        if(ch=='y'|| ch=='Y')
        {
            printf("\n----Receipt-----\n");
            printf("Transaction :withdrawal\n");
            printf("Amount:Rs %.2f\n",amt);
            printf("Remaining Balance: Rs %.2f\n",acc.balance);

        }
        //adding transaction to history
        addHistory("Withdrawn:");
        FILE *fp= fopen("history.txt", "a");  //opening file in append mode
to add transaction info
        fprintf(fp, " Rs %.2f | Balance: Rs %.2f\n", amt, acc.balance);
        fclose(fp);  //closing the file

    }

    return acc;
}
```

- **changePIN(struct account acc)**

  This function enable sthe user to change their 4 digit pin.It asks for retype of new PIN also for validation.And returns the updated structure account.

  code of changePIN(struct account acc):

```
struct account changePIN(struct account acc)
{
    int newPin,again;
    printf("Enter new PIN: ");
    scanf("%d", &newPin);
    while(getchar()!='\n');   // to clear buffer

    printf("Re-enter new PIN: ");   //for re-confirming the pin
    scanf("%d", &again);
    while(getchar()!='\n');   // to clear buffer

    if (newPin==again && newPin>=1000 && newPin<= 9999) {
        acc.pin=newPin;
        printf("PIN changed successfully.\n");
    }
    else
    {   // if the entered new pins doesnot match
        printf("PIN not changed. Either mismatch or not 4 digits.\n");
    }

    return acc;}
```

- **changeName(struct account acc)**

This function allows user to change the account hoder name .It also updates the name in the memory and saves it into account.txt file.It also Returns the updated account structure.

Code of changeName(struct account acc):

```
struct account changeName(struct account acc)
{
   char temp[50];

   printf("Enter new Account Holder name: ");


   // Read the name safely using fgets()
   if (fgets(temp, sizeof(temp), stdin) != NULL) {
      // Remove trailing newline character left by fgets()
      temp[strcspn(temp, "\n")] = '\0';

      if (strlen(temp) == 0) { // Check for empty input (e.g., if user just pressed
Enter)
         printf("No name entered. Name unchanged.\n");
         return acc;
      }

      // Update account name
      strcpy(acc.name, temp);
      printf("Account Holder name updated successfully: %s\n", acc.name);
   } else {
      printf("Error reading name.\n");
   }

   return acc;
}
```

- **addHistory(char *entry)**

  This function adds the entry of transaction into history.txt file.It is used by deposit and withdrawal functions to maintain a list of recent transactions.

Code of addHistory(char *entry):

```
void addHistory(char *entry)
{
   FILE *fp = fopen("history.txt", "a");
   if (fp ==NULL) //if file does not exist
   {
      printf("Error writing to history.\n");
      return;
   }
   fprintf(fp,"%s\n",entry);//adding transaction to file
   fclose(fp); //closing pointer
}
```

- **showHistory()**

  This function is used to display the transaction history stored in history.txt file.It also informs if no transactions are recorded.

  Code of showHistory():

```
void showHistory()
{
   FILE *fp = fopen("history.txt", "r");
   char line[200];
   int empty=1; //to check if history.txt is empty or not

   if (fp==NULL) //if file does not exist
   {
      printf("\nNo transactions recorded yet.\n");
      return;
   }

   printf("\n--- MINI TRANSACTION HISTORY ---\n");
   while (fgets(line,sizeof(line),fp) != NULL)
   {    empty=0;
      printf("%s", line);
   }
   if(empty)  //if file exist but empty
   {
      printf("No transactions recorded yet.\n");
   }
   fclose(fp);
}
```

- **saveToFile(struct account acc)**

  This function is used here to save all account details to account.txt file to ensure that memory changes after transactions nd pinchange and name change or account resetting.

  Code of saveToFile(struct account acc):

```
void saveToFile(struct account acc)
{
   FILE *fp = fopen("account.txt", "w");
   if (fp == NULL)
   {
      printf("Error saving data!\n");
      return;
   }
   //printing information to file account.txt
   fprintf(fp,
"%s\n%d\n%.2f\n%s\n%s\n%s\n",acc.name,acc.pin,acc.balance,acc.bankNa
me,acc.branch,acc.type);
   fclose(fp);  //closing file
}
```

- **loadFromFile()**

  This loads account details from the account.txt file into the account structure.It also creates a default account with preset default values if the file does not exist.It also returns the account structure.

  Code of loadFromFile():

```
struct account loadFromFile()
{
   struct account acc;
   FILE *fp = fopen("account.txt", "r");

   if (fp==NULL) //if file is empty then storing the default values
   {
      strcpy(acc.name,"Default User");
      acc.pin=1234;
      acc.balance=10000.0;
      strcpy(acc.bankName,"State Bank of India");
      strcpy(acc.branch,"Dehradun Main Branch");
      strcpy(acc.type,"savings");
      saveToFile(acc);
      return acc;
   }
   //if file is not empty Then storing the stored data in structure vriables
   fgets(acc.name,sizeof(acc.name),fp); //reading name
```

```
    acc.name[strcspn(acc.name,"\n")]='\0';

    fscanf(fp,"%d",&acc.pin);
    fscanf(fp,"%f",&acc.balance);
    fgetc(fp);
    fgets(acc.bankName, sizeof(acc.bankName), fp);
    acc.bankName[strcspn(acc.bankName,"\n")]='\0';
    fgets(acc.branch, sizeof(acc.branch), fp);
    acc.branch[strcspn(acc.branch,"\n")]='\0';
    fgets(acc.type,sizeof(acc.type), fp);
    acc.type[strcspn(acc.type,"\n")]='\0';

    fclose(fp);   //closing the file
    return acc;
}
```

- **resetAccount(struct account acc)**

  This function is used in my project to reset all the values in account to default preset values.It also clears the history.txt file to remove transaction history.It also reloads the default data into memory and return sthe reset account structure.

  Code of resetAccount(struct account acc):

```
struct account resetAccount(struct account user)
{   //reseting the account
    FILE *fp=fopen("account.txt","w");   //first emptying and then writing
    if(fp==NULL)
    {
        printf("Error resetting account!\n");
        return user;
    }
    fprintf(fp,"Default User\n1234\n10000.00\n");
    fprintf(fp,"State Bank of India\n");
    fprintf(fp,"Dehradun Main Branch\n");
    fprintf(fp,"Savings\n");
    fclose(fp);
    // Reset History File
    FILE *history=fopen("history.txt", "w");  // opening in write mode clears
the file
    fclose(history);
    printf("Transaction history cleared.\n");
    printf("Account reset successfully.\n");
    // Reload default values into memory
    user=loadFromFile();
    return user;
}
```

TESTING

I tested my code of ATM simulator using a variety of valid and invalid inputs to ensure that the program works correctly in different situations.Watching the output sin the terminal helped me understand whether each block of code was working as expected or not.I also captured screenshot salong the way to record the results and show the program's performance clearly.

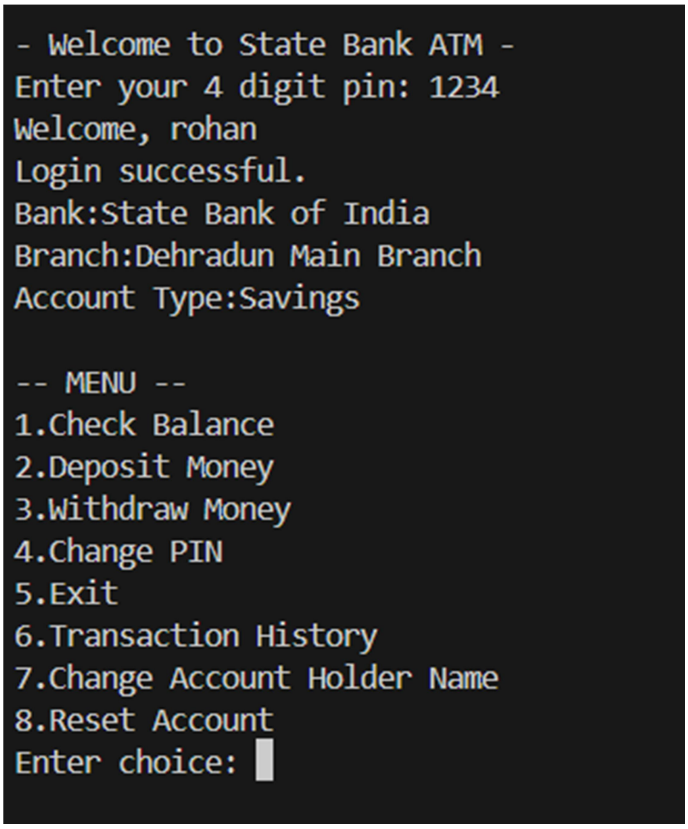
Test Case table

Test Case	What I tested	What I expected to Happen	What Actually Happened	Result
1.Correct PIN Entry	PIN=correct value	Login successful	Login successful	Pass
2.Incorrect PIN Entry	PIN=wrong value (3 times)	Access Denied after 3 attempts	Access denied	Pass
3.Deposit Valid Amount	Deposit=500	Balance increases by 500	Balance updated correctly	Pass
4. Deposit Invalid Amount	Deposit=-50 or 0	Show error message	Error shown	Pass
5.Withdraw Valid amount	Withdraw= amount<= balance	Balance reduced correctly	Works as expected	Pass
6.Withdraw More than Balance	Withdraw> balance	Show insufficient balance	Error Shown	Pass

7.Check Balance	Select balance option	Display current balance	Balance displayed	Pass
8.Change PIN(valid)	Enter new Valid PIN	PIN updated	PIN updated	Pass
9.Change PIN (invalid)	Entering mismatched pin in retyping pin	Should show error	Error message shown	Pass
10.Save Data to file	Select save option	File should be created/updated	File saved successfully	Pass
11.Load Data from file	Select Load option	Data should load correctly	Data loaded successfully	Pass
12.Mini Transaction History	Performing few operations	Should show last few transactions	Correct list displayed	Pass
13.Change account holder name	Entering a new valid name	Should update the name when a valid name is entered and save to file	Name updated to new name and saved in account.txt	Pass
14.Reset Account	Select Reset option	Balance and PIN reset to default	Reset successful	Pass
15.Exit	Selecting exit option from main menu	Program shouls terminate gracefully with a thank you message	Program exited successfully and displayed expected message	Pass

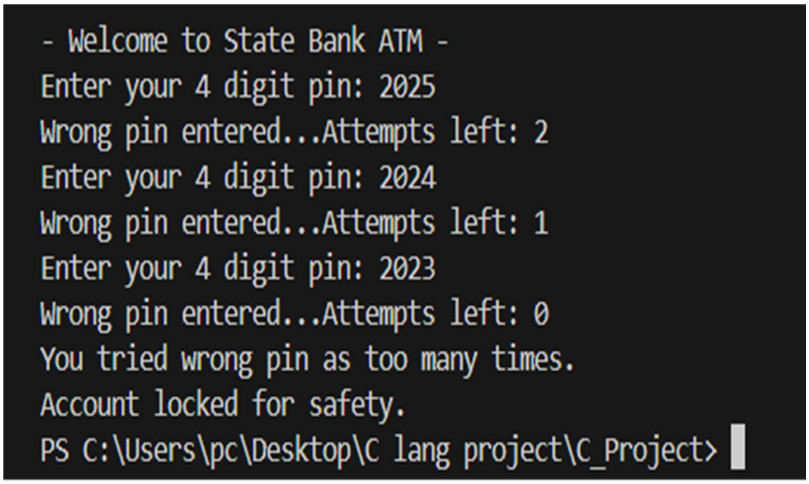
Screenshot of Test Cases:

Test case-1 screenshot : Correct PIN Entry



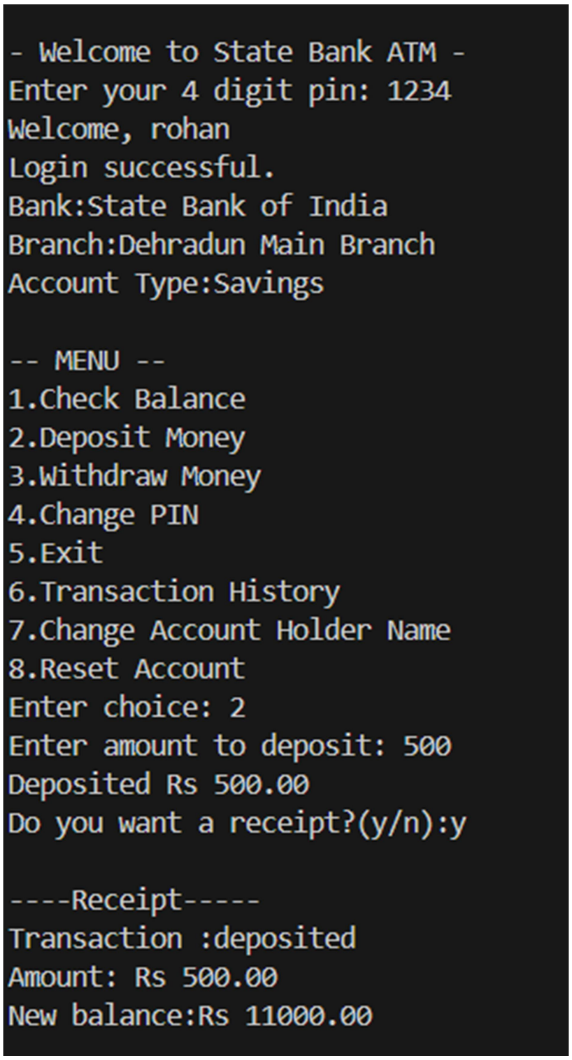
Explanation: The user entered the correct PIN .The system verified it successfully and displayed the welcome message along with account details.

Test case-2 screenshot : Incorrect PIN Entry



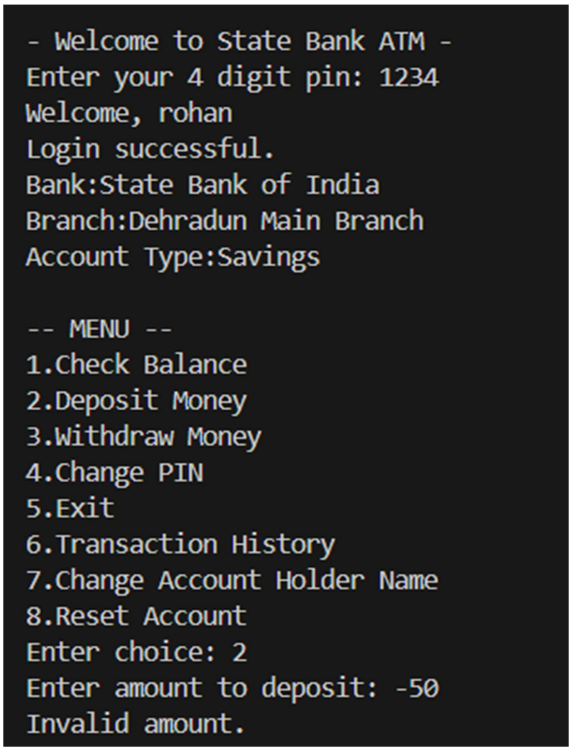
Explanation: The user entered an incorrect PIN three times. After the third attempt,the system denied access and displayed an account lock warning.

Test case-3 screenshot : Deposit Valid Amount



Explanation:The user deposited a valid amount (Rs.500) .The account balance updated correctly and a receipt was optionally displayed.

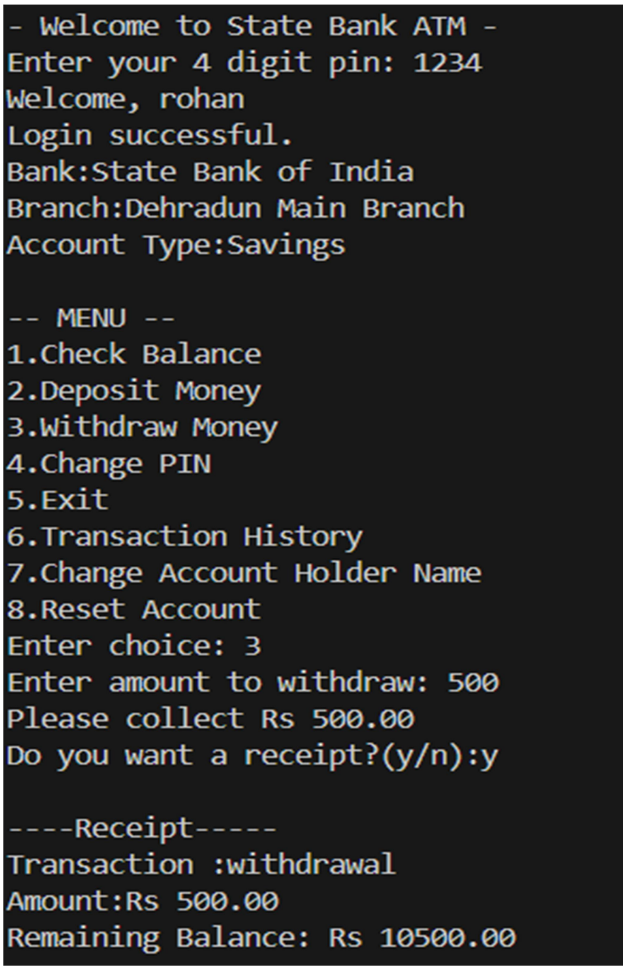
Test case-4 screenshot : Deposit Invalid Amount



Explanation: The user tried to deposit an invalid amount of -50 .But the system displayed an error message and did not change the balance.

(P.T.O)

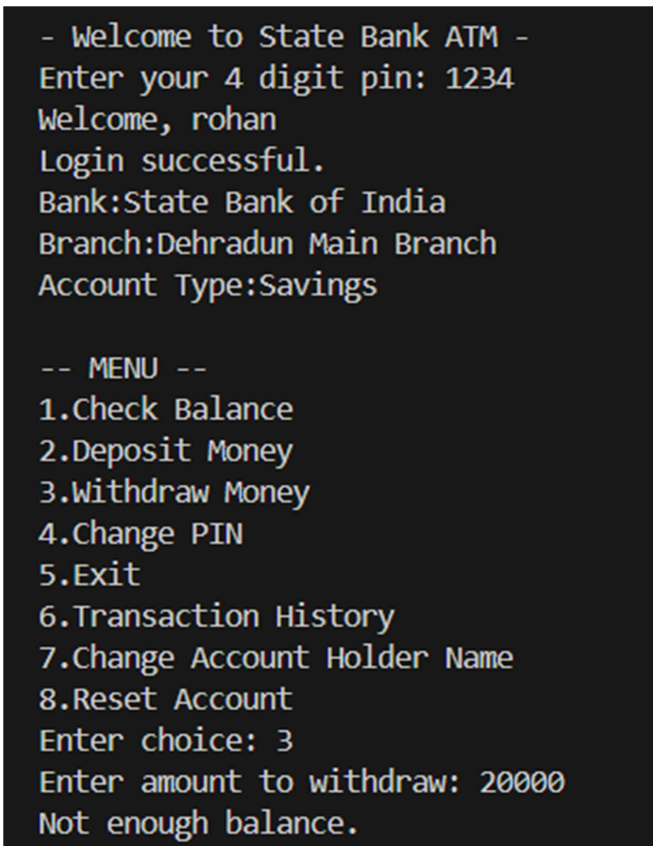
Test case-5 screenshot : Withdraw Valid amount



Explanation:The user a valid amount within the available balance.The balance decreased accordingly and a receipt was optionally displayed.

(P.T.O)

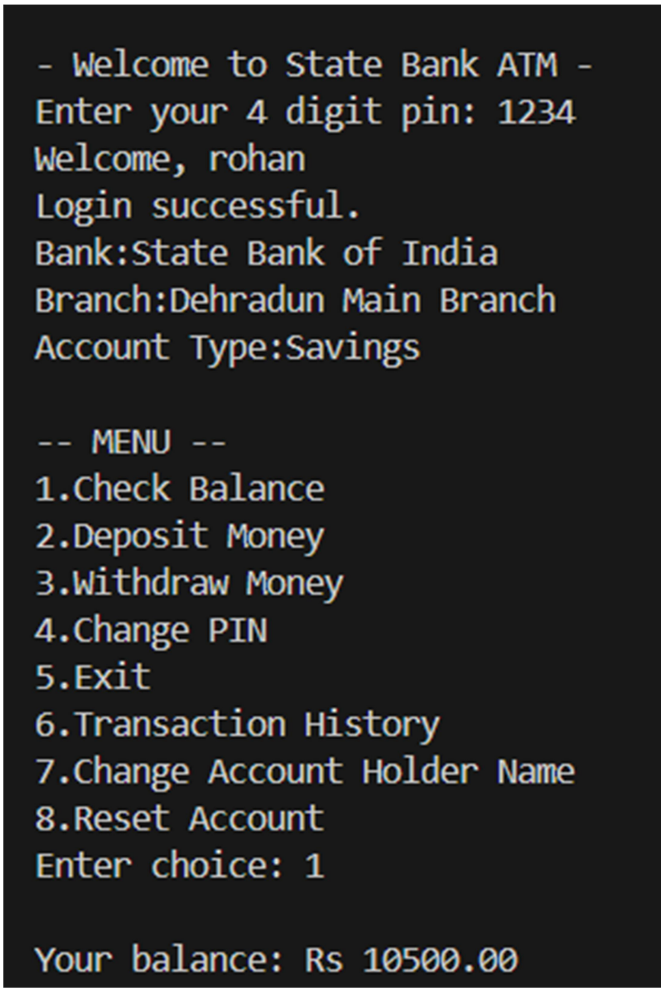
Test case-6 screenshot : Withdraw More than Balance



Explanation:The user tried to withdraw(20000) more than the available balance .The system prevented the withdrawal and showed a warning of insufficient balance on screen.

(P.T.O)

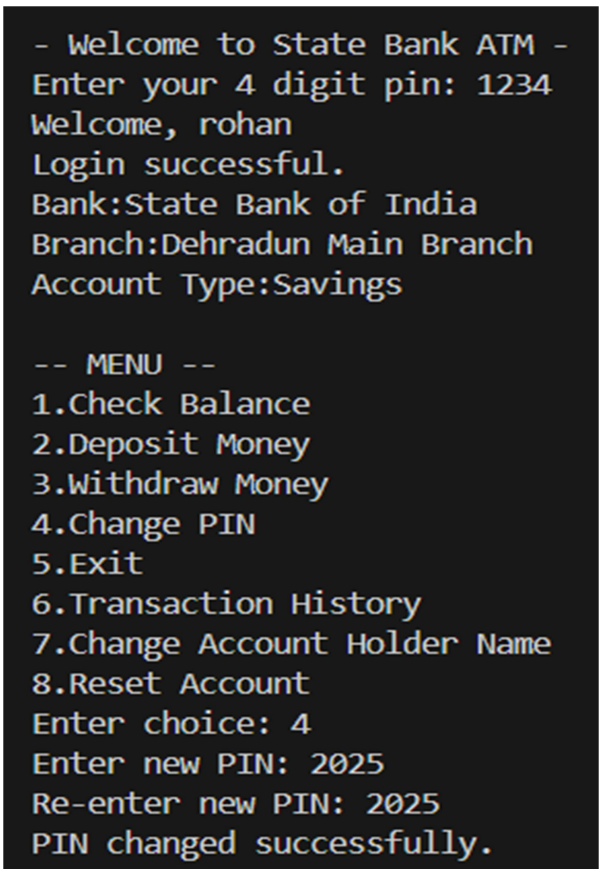
Test case-7 screenshot : Check Balance



Explanation:The User selected the Check Balance option and the current was displayed correctly.

(P.T.O)

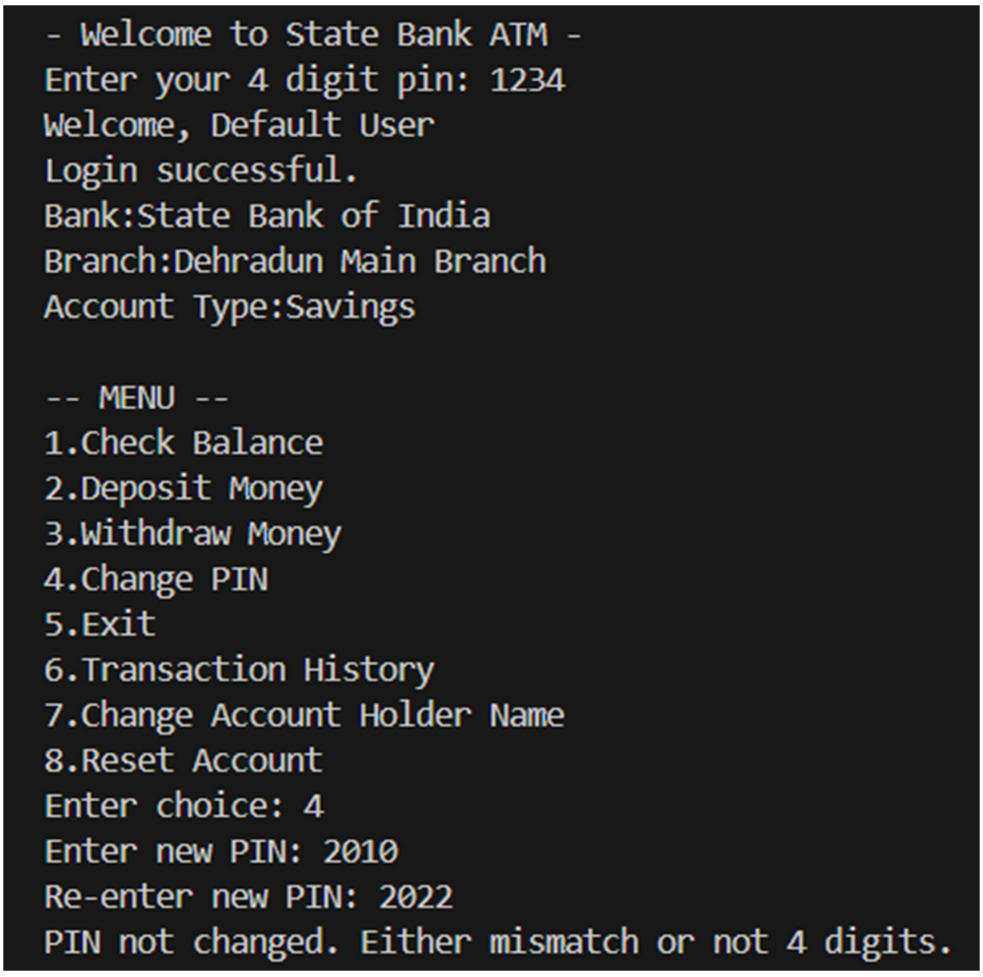
Test case-8 screenshot : Change PIN(valid)



Explanation:The user successfully changed the PIN by entering a valid 4 digit number twice.Confirmation message was displayed.

(P.T.O)

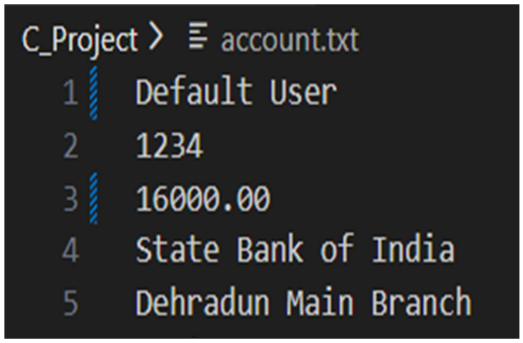
Test case-9 screenshot : Change PIN (invalid)



Explanation: The user attempted to change PIN with invalid input(e.g. mismatch) .The system rejected it and displayed an error.

Test case-10 screenshot : Save Data to file

```
- Welcome to State Bank ATM -
Enter your 4 digit pin: 1234
Welcome, Default User
Login successful.
Bank:State Bank of India
Branch:Dehradun Main Branch
Account Type:Savings

-- MENU --
1.Check Balance
2.Deposit Money
3.Withdraw Money
4.Change PIN
5.Exit
6.Transaction History
7.Change Account Holder Name
8.Reset Account
Enter choice: 1

Your balance: Rs 11000.00

-- MENU --
1.Check Balance
2.Deposit Money
3.Withdraw Money
4.Change PIN
5.Exit
6.Transaction History
7.Change Account Holder Name
8.Reset Account
Enter choice: 2
Enter amount to deposit: 5000
Deposited Rs 5000.00
Do you want a receipt?(y/n):y

----Receipt-----
Transaction :deposited
Amount: Rs 5000.00
New balance:Rs 16000.00
```



Explanation:After performing transactions,the system saved updated account details.The file was verified successfully

Test case-11 screenshot : Load Data from file

```
- Welcome to State Bank ATM -
Enter your 4 digit pin: 1234
Welcome, aman
Login successful.
Bank:State Bank of India
Branch:Dehradun Main Branch
Account Type:Savings

-- MENU --
1.Check Balance
2.Deposit Money
3.Withdraw Money
4.Change PIN
5.Exit
6.Transaction History
7.Change Account Holder Name
8.Reset Account
Enter choice:
```

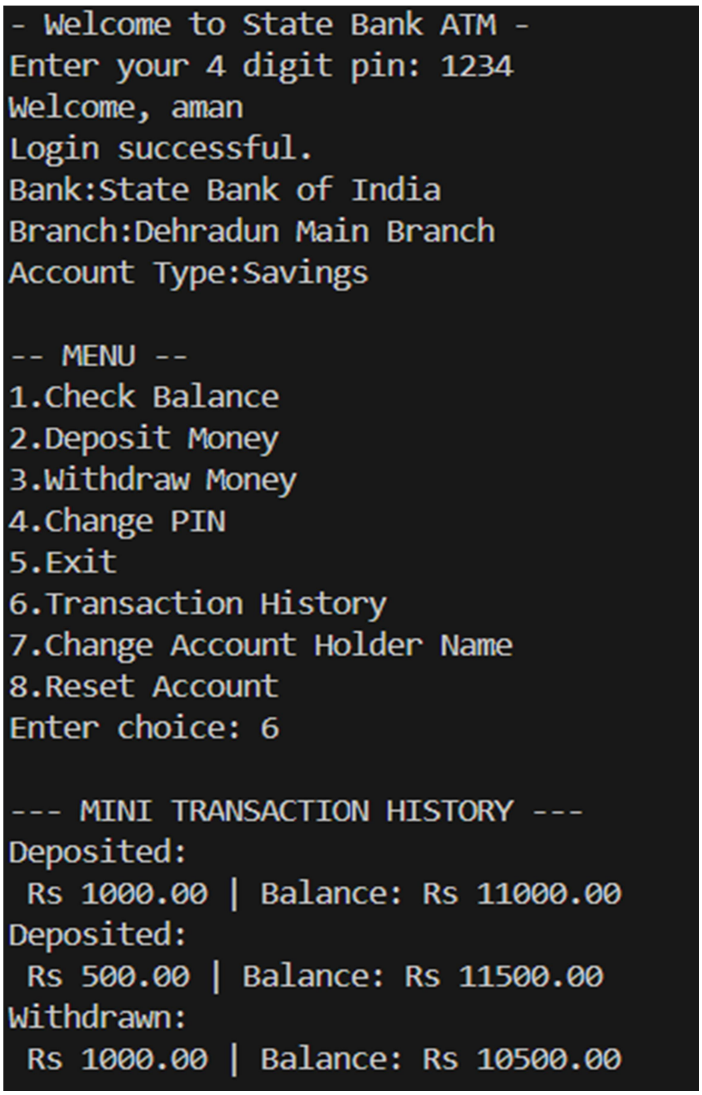
C_Project > account.txt

```
1  aman
2  1234
3  10000.00
4  State Bank of India
5  Dehradun Main Branch
6  Savings
```

Explanation:account data was loaded correctly from account.txt,ensuring that previous transactions and details persisted.

(P.T.O.)

Test case-12 screenshot : Mini Transaction History



Explanation: The system displayed a list of recent transactions correctly in mini statement format.

(P.T.O.)

Test case-13 screenshot : Change account holder name

```
- Welcome to State Bank ATM -
Enter your 4 digit pin: 1234
Welcome, Default User
Login successful.
Bank:State Bank of India
Branch:Dehradun Main Branch
Account Type:Savings

-- MENU --
1.Check Balance
2.Deposit Money
3.Withdraw Money
4.Change PIN
5.Exit
6.Transaction History
7.Change Account Holder Name
8.Reset Account
Enter choice: 7
Enter new Account Holder name: aman
Account Holder name updated successfully: aman
```

C_Project > account.txt

```
1  aman
2  1234
3  10000.00
4  State Bank of India
5  Dehradun Main Branch
6  Savings
```

Explanation:The user enetered a new valid name.The system updated the account name and successfully saved it to account.txt.

(P.T.O)

Test case-14 screenshot : Reset Account

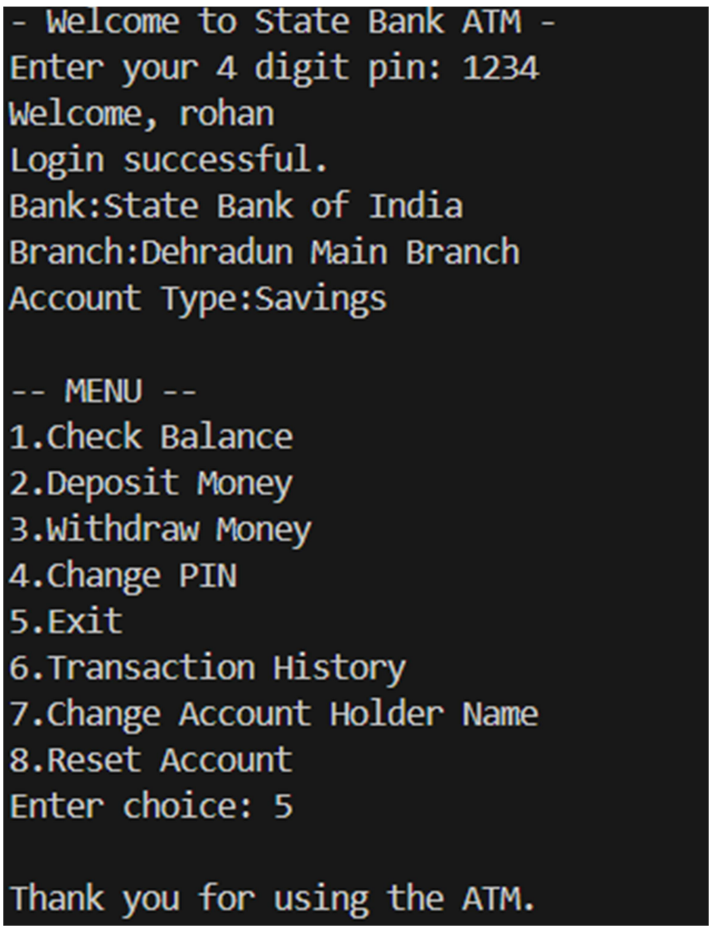
```
- Welcome to State Bank ATM -
Enter your 4 digit pin: 1234
Welcome, Default User
Login successful.
Bank:State Bank of India
Branch:Dehradun Main Branch
Account Type:Savings

-- MENU --
1.Check Balance
2.Deposit Money
3.Withdraw Money
4.Change PIN
5.Exit
6.Transaction History
7.Change Account Holder Name
8.Reset Account
Enter choice: 1

Your balance: Rs 16000.00

-- MENU --
1.Check Balance
2.Deposit Money
3.Withdraw Money
4.Change PIN
5.Exit
6.Transaction History
7.Change Account Holder Name
8.Reset Account
Enter choice: 8
Transaction history cleared.
Account reset successfully.

-- MENU --
1.Check Balance
2.Deposit Money
3.Withdraw Money
4.Change PIN
5.Exit
6.Transaction History
7.Change Account Holder Name
8.Reset Account
Enter choice: 1

Your balance: Rs 10000.00
```

C_Project > account.txt

```
1  Default User
2  1234
3  16000.00
4  State Bank of India
5  Dehradun Main Branch
6  Savings
```

Explanation:The user selected the Reset Account .The balance,PIN and other account details were reset to default values,and transaction history was cleared.

Test case-15 screenshot :Exit



Explanation:
The user selected the “Exit” option and the program terminated gracefully and displayed the message:Thank you for using the ATM

(P.T.O)

CONCLUSION

This ATM Simulator project helped me understand how real-word banking systems work in real life .It has been a greta learning experience.By breaking the program into small and reusable functions ,I understood the importance of modular programming approach.I divided the tasks like checking balance,depositing and withdrawing money,changing PIN,saving and loading data to files and showing a mini transaction history.It helped me to learn much about the file handling concepts.

I used concepts from almost all six units taught to me in the lecture classes by our professor.from this project,I got a hand-on experience on ATM features and using PIN verification with max attempt for verification.The resetAccount() function helped me test my code easily.These made the program feel more like an actual ATM and helped me understand the applications of C programming.

Overall,this project has not only helped improve my coding skilss but also given me a sense of how programming can be used to solve real life problems.By completing this project ,it has boosted my confidence in solving these bigger and complex projects in the future .In future,I can also imagine taking this ATM simulator further like supporting multiple users,adding more banking features,etc. It has been a rewarding experience that strengthened my skills and sparked my curiosity to explore more practical applications of programming.I would sincerely like to thank my professor Dr.Prashant Trivedi for his continuous guidance and support throughout this project.

Appendix

1.Source Code

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

struct account
{
    char name[50];   //to store acc holder name
    int pin;          //to store the pin
    float balance;    //to store account balance
    char bankName[50];   //to store bank name
    char branch[50];    //to store branch
    char type[20];       //to store the account type
};


void checkBalance(struct account acc);
struct account depositMoney(struct account acc);
struct account withdrawMoney(struct account acc);
struct account changePIN(struct account acc);
void addHistory(char *entry);
void showHistory();
struct account changeName(struct account acc);
void saveToFile(struct account acc);
struct account loadFromFile();
struct account resetAccount(struct account user);

int main()
{
    struct account user;
    user=loadFromFile();   // to load saved account (or create new)

    int enteredPin,choice;
    int attempts=0;

    printf("\n- Welcome to State Bank ATM -\n");

    while (1)
    {
        printf("Enter your 4 digit pin: ");
        if (scanf("%d", &enteredPin) != 1)
        {
        printf("Invalid input! Numbers only.\n");
         while (getchar()!='\n'); // to clear the buffer
        continue;
        }
```

```
        if (enteredPin==user.pin)
        {
            //showing user and acc information
            printf("Welcome, %s\n",user.name);
            printf("Login successful.\n");
            printf("Bank:%s\n",user.bankName);
            printf("Branch:%s\n",user.branch);
            printf("Account Type:%s\n",user.type);

            // low balance warning
            if (user.balance < 1000)
                printf("\n WARNING:Your balance is very low.Please deposit
soon.\n");
            break;
        }
        else
        {
            attempts++;
            printf("Wrong pin entered...Attempts left: %d\n",3-attempts);
        }
        if(attempts==3) //exiting if more than 3 wrong pins entered
        {
            printf("You tried wrong pin as too many times.\n");
            printf("Account locked for safety.\n");
            return 0;
        }
    }

    // printing Menu and using switch case to execute the menu choices
    do {
        printf("\n-- MENU --\n");
        printf("1.Check Balance\n");
        printf("2.Deposit Money\n");
        printf("3.Withdraw Money\n");
        printf("4.Change PIN\n");
        printf("5.Exit\n");
        printf("6.Transaction History\n");
        printf("7.Change Account Holder Name\n");
        printf("8.Reset Account\n");
        printf("Enter choice: ");
        scanf("%d", &choice);   //taking input from user
        while(getchar()!='\n');   // to clear the buffer

        switch (choice)
        {
            case 1:
                checkBalance(user);
                break;

            case 2:
                user=depositMoney(user);
                saveToFile(user);
```

```
                break;

            case 3:
                user = withdrawMoney(user);
                saveToFile(user);
                break;

            case 4:
                user = changePIN(user);
                saveToFile(user);
                break;

            case 5:
                printf("\nThank you for using the ATM.\n");
                break;
            case 6:
                    showHistory();
                     break;// to avoid fall through condition
            case 7:
                    user=changeName(user);
                    saveToFile(user);
                    break;

            case 8:  user=resetAccount(user);
                     saveToFile(user);
                    break;


            default:
                printf("Invalid option. Try again.\n");
        }

    } while (choice != 5);

    return 0;
}


// Save account to file
void saveToFile(struct account acc)
{
    FILE *fp = fopen("account.txt", "w");
    if (fp == NULL)
    {
        printf("Error saving data!\n");
        return;
    }
    //printing information to file account.txt
    fprintf(fp,
"%s\n%d\n%.2f\n%s\n%s\n%s\n",acc.name,acc.pin,acc.balance,acc.bankName,acc.bra
nch,acc.type);
    fclose(fp);  //closing file
```

```c
}

// Load account from file (or create default)
struct account loadFromFile()
{
    struct account acc;
    FILE *fp = fopen("account.txt", "r");

    if (fp==NULL) //if file is empty then storing the default values
    {
        strcpy(acc.name,"Default User");
        acc.pin=1234;
        acc.balance=10000.0;
        strcpy(acc.bankName,"State Bank of India");
        strcpy(acc.branch,"Dehradun Main Branch");
        strcpy(acc.type,"savings");
        saveToFile(acc);
        return acc;
    }
    //if file is not empty Then storing the stored data in structure vriables
    fgets(acc.name,sizeof(acc.name),fp); //reading name
    acc.name[strcspn(acc.name,"\n")]='\0';

    fscanf(fp,"%d",&acc.pin);
    fscanf(fp,"%f",&acc.balance);
    fgetc(fp);
    fgets(acc.bankName, sizeof(acc.bankName), fp);
    acc.bankName[strcspn(acc.bankName,"\n")]='\0';
    fgets(acc.branch, sizeof(acc.branch), fp);
    acc.branch[strcspn(acc.branch,"\n")]='\0';
    fgets(acc.type,sizeof(acc.type), fp);
    acc.type[strcspn(acc.type,"\n")]='\0';

    fclose(fp);   //closing the file
    return acc;
}



// NORMAL ATM FUNCTIONS
void checkBalance(struct account acc)
{
    printf("\nYour balance: Rs %.2f\n", acc.balance);//printing acc balance
    if (acc.balance < 1000)
                { printf("\n WARNING:Your balance is very low.Please deposit
soon.\n");
                }
}

struct account depositMoney(struct account acc)
{
    float amt;
```

```
    printf("Enter amount to deposit: ");  //taking amount to be deposited
    scanf("%f",&amt);

    if(amt<=0)  //if amt less than or equal to 0
    {
        printf("Invalid amount.\n");
    }
    else
    {
        acc.balance+=amt;    //updating the balance
        printf("Deposited Rs %.2f\n", amt);
        while (getchar() !='\n');  //to clear the buffer
        char ch;
        printf("Do you want a receipt?(y/n):"); //asking for receipt printing
        scanf(" %c",&ch);
        if(ch=='y'|| ch=='Y')
        {
            printf("\n----Receipt-----\n");
            printf("Transaction :deposited\n");
            printf("Amount: Rs %.2f\n", amt);
            printf("New balance:Rs %.2f\n",acc.balance);

        }
         //adding transaction to history
        addHistory("Deposited:");
        FILE *fp =fopen("history.txt", "a");    //opening file in append mode
to add transaction info
        fprintf(fp, " Rs %.2f | Balance: Rs %.2f\n", amt, acc.balance);
        fclose(fp);   //closing the file

    }
    return acc;
}

struct account withdrawMoney(struct account acc)
{
    float amt;
    printf("Enter amount to withdraw: ");
    scanf("%f", &amt);   //inputting withdrawl amount
    while (getchar() != '\n');
    if (amt<=0)
    {
        printf("Invalid amount.\n");
    }
    else if(amt>acc.balance)
    {   //if amount in account is less than withdrawl request
        printf("Not enough balance.\n");
    }
    else
    {
        acc.balance-=amt;    //updating the balance
        printf("Please collect Rs %.2f\n", amt);
```

```
         char ch;
        printf("Do you want a receipt?(y/n):");  //asking for receipt
        scanf(" %c",&ch);
        if(ch=='y'|| ch=='Y')
        {
            printf("\n----Receipt-----\n");
            printf("Transaction :withdrawal\n");
            printf("Amount:Rs %.2f\n",amt);
            printf("Remaining Balance: Rs %.2f\n",acc.balance);

        }
        //adding transaction to history
        addHistory("Withdrawn:");
        FILE *fp= fopen("history.txt", "a");  //opening file in append mode to
add transaction info
        fprintf(fp, " Rs %.2f | Balance: Rs %.2f\n", amt, acc.balance);
        fclose(fp);  //closing the file

    }

    return acc;
}

struct account changePIN(struct account acc)
{
    int newPin,again;
    printf("Enter new PIN: ");
    scanf("%d", &newPin);
    while(getchar()!='\n');   // to clear buffer

    printf("Re-enter new PIN: ");   //for re-confirming the pin
    scanf("%d", &again);
    while(getchar()!='\n');   // to clear buffer

    if (newPin==again && newPin>=1000 && newPin<= 9999) {
        acc.pin=newPin;
        printf("PIN changed successfully.\n");
    }
    else
    {   // if the entered new pins doesnot match
        printf("PIN not changed. Either mismatch or not 4 digits.\n");
    }

    return acc;
}
void addHistory(char *entry)
{
    FILE *fp = fopen("history.txt", "a");
    if (fp ==NULL) //if file does not exist
    {
        printf("Error writing to history.\n");
        return;
```

```c
    }
    fprintf(fp,"%s\n",entry);//adding transaction to file
    fclose(fp); //closing pointer
}
void showHistory()
{
    FILE *fp = fopen("history.txt", "r");
    char line[200];
    int empty=1; //to check if history.txt is empty or not

    if (fp==NULL) //if file does not exist
    {
        printf("\nNo transactions recorded yet.\n");
        return;
    }

    printf("\n--- MINI TRANSACTION HISTORY ---\n");
    while (fgets(line,sizeof(line),fp) != NULL)
    {    empty=0;
        printf("%s", line);
    }
    if(empty)  //if file exist but empty
    {
        printf("No transactions recorded yet.\n");
    }
    fclose(fp);
}
struct account changeName(struct account acc)
{
    char temp[50];

    printf("Enter new Account Holder name: ");


    // Read the name safely using fgets()
    if (fgets(temp, sizeof(temp), stdin) != NULL) {
        // Remove trailing newline character left by fgets()
        temp[strcspn(temp, "\n")] = '\0';

        if (strlen(temp) == 0) { // Check for empty input (e.g., if user just
pressed Enter)
            printf("No name entered. Name unchanged.\n");
            return acc;
        }

        // Update account name
        strcpy(acc.name, temp);
        printf("Account Holder name updated successfully: %s\n", acc.name);
    } else {
        printf("Error reading name.\n");
    }
```

```
    return acc;
}


struct account resetAccount(struct account user)
{    //reseting the account
    FILE *fp=fopen("account.txt","w");   //first emptying and then writing
    if(fp==NULL)
    {
        printf("Error resetting account!\n");
        return user;
    }
    fprintf(fp,"Default User\n1234\n10000.00\n");
    fprintf(fp,"State Bank of India\n");
    fprintf(fp,"Dehradun Main Branch\n");
    fprintf(fp,"Savings\n");
    fclose(fp);
    // Reset History File
    FILE *history=fopen("history.txt", "w");  // opening in write mode clears
the file
    fclose(history);
    printf("Transaction history cleared.\n");
    printf("Account reset successfully.\n");
    // Reload default values into memory
    user=loadFromFile();
    return user;
}
```

2.References

- lecture note s and class materials provided by my professor
- Let us C Book by Yashavant Kanetkar
- GeeksforGeeks