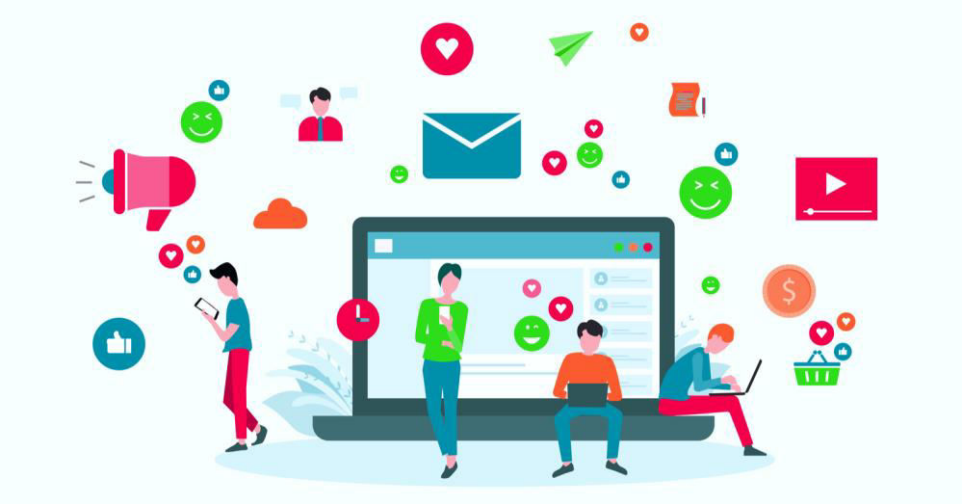
TECHNO INTERNATIONAL NEW TOWN

DSA PROJECT

TOPIC – FIND 5 CLOSEST FRIENDS ON A SOCIAL MEDIA PROFILE DEPENDING UPON LATEST ACTIVITIES



GROUP MEMBERS -

ANWARUL HAQUE

PIYUSH KUMAR BYAHUT

TAHSEEN ATIQUE ALI

POUSHALI GHOSH

INTRODUCTION

SOCIAL MEDIA:

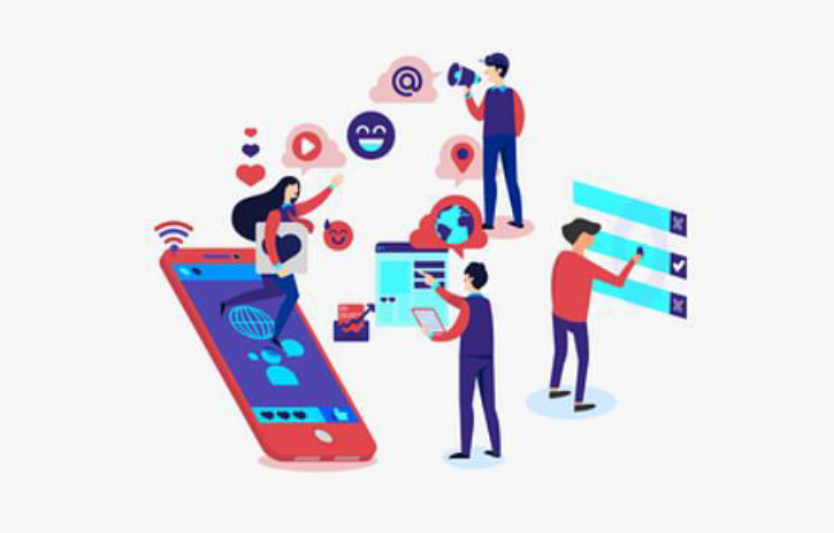
SOCIAL MEDIA REFERS TO THE MEANS OF INTERACTIONS AMONG PEOPLE IN WHICH THEY CREATE, SHARE, AND/OR EXCHANGE INFORMATION AND IDEAS IN VIRTUAL COMMUNITIES AND NETWORKS.

SCOPE OF STUDY:

On a personal level, social media allows you to communicate with friends and family, learn new things, develop your interests, and be entertained. On a professional level, you can use social media to broaden your knowledge in a particular field and build your professional network by connecting with other professionals in your industry.

PROBLEM STATEMENT:

THIS PROJECT TAKES AN IN-DEPTH LOOK AT THE ROLE OF SOCIAL MEDIA IN COMMUNICATION THROUGHOUT THE WORLD AND ALLOWS PEOPLE TO CONNECT FROM ANY CORNER OF THE GLOBE ANYTIME.



REAL CASE APPLICATIONS OF SOCIAL MEDIA:

1. Building Relationships and Staying Connected

Social media can make it easy to find groups of like-minded people or make new friends. Finding a close-knit community can help us feel valued and accepted. Social media is also an easy way to nurture existing relationships with family and friends who have moved away. Send messages, share photos, call, or host video chats to stay in touch.

2.SPREADING NEWS

News from just about any part of the world can spread like wildfire on social media. While this can be overwhelming at times, it can also keep us in tune with important events. This can be an outstanding benefit if you need to get the word out fast about something. For example, if someone from a small town lost their dog, they could get the message out on social media. Everyone in the area could keep an eye out and report back with information instantly.

3.Showing Empathy and Kindness

People often share things online that are personal or that they're struggling with. When you relate, you can show empathy by supporting and encouraging others through messages and comments. Show your friends (and even those you've never met) that you care about their situation and understand where they're coming from. Seeing others work through tough times can also be inspiring and help us see things from a new perspective.

ALGORITHM

START:

MAIN()

1. INITIALIZE AN INTEGER VARIABLE 'OPTION'

2. START AN INFINITE LOOP

3. DISPLAY THE MENU OPTIONS: "CREATE ACCOUNT", "LOGIN", "EXIT"

4. PROMPT THE USER TO ENTER AN OPTION

5. READ THE USER'S OPTION AND STORE IT IN THE 'OPTION' VARIABLE

6. USE A SWITCH STATEMENT TO EVALUATE THE 'OPTION' VARIABLE

7. IF 'OPTION' IS 1, CALL THE 'CREATE_ACCOUNT' FUNCTION

8. IF 'OPTION' IS 2, CALL THE 'LOGIN' FUNCTION

9. IF 'OPTION' IS 3, PRINT "EXITING..." AND TERMINATE THE PROGRAM WITH EXIT CODE 0

10. IF 'OPTION' IS ANY OTHER VALUE, PRINT "INVALID OPTION, TRY AGAIN."

11. END THE LOOP

12. RETURN 0

CREATE_ACCOUNT()

1. DECLARE TWO CHAR ARRAYS 'USER' AND 'PASSWORD' WITH THE MAXIMUM ALLOWED LENGTHS 'MAX_USERNAME_LENGTH' AND 'MAX_PASSWORD_LENGTH' RESPECTIVELY.

2. PROMPT THE USER TO ENTER A USERNAME AND READ IT INTO THE 'USER' VARIABLE.

3. USE A WHILE LOOP TO VALIDATE THE USERNAME WITH THE 'ISVALIDUSERNAME' FUNCTION. IF THE USERNAME IS INVALID, PROMPT THE

USER TO RE-ENTER A USERNAME. REPEAT UNTIL A VALID USERNAME IS ENTERED.

4. USE ANOTHER WHILE LOOP TO CHECK IF THE USERNAME IS ALREADY TAKEN WITH THE 'ISUSERNAMETAKEN' FUNCTION. IF IT'S TAKEN, PROMPT THE USER TO TRY A NEW ONE. REPEAT UNTIL A UNIQUE USERNAME IS ENTERED.

5. PROMPT THE USER TO ENTER A PASSWORD AND READ IT INTO THE 'PASSWORD' VARIABLE.

6. USE A WHILE LOOP TO VALIDATE THE PASSWORD WITH THE 'ISVALIDPASSWORD' FUNCTION. IF THE PASSWORD IS INVALID, PROMPT THE USER TO RE-ENTER A PASSWORD. REPEAT UNTIL A VALID PASSWORD IS ENTERED.

7. OPEN A FILE 'ACCOUNTS.CSV' IN "APPEND" MODE USING THE 'FOPEN' FUNCTION.

8. WRITE THE 'USER' AND 'PASSWORD' VARIABLES TO THE FILE WITH THE 'FPRINTF' FUNCTION, SEPARATED BY A COMMA.

9. PRINT "ACCOUNT CREATED SUCCESSFULLY".

10. CLOSE THE FILE USING THE 'FCLOSE' FUNCTION.

*ISVALIDUSERNAME(CHAR *USER)*

1. INPUT: A STRING USER

2. INITIALIZE THREE VARIABLES A, B AND C TO 1, TO INDICATE THE VALIDITY OF USERNAME LENGTH, FIRST CHARACTER AND CHARACTER TYPE, RESPECTIVELY.

3. CHECK THE LENGTH OF USER:

A. IF THE LENGTH OF USER IS LESS THAN MIN_LENGTH, PRINT AN ERROR MESSAGE "ERROR: USERNAME IS TOO SHORT." AND SET A = 0.

B. IF THE LENGTH OF USER IS MORE THAN MAX_LENGTH, PRINT AN ERROR MESSAGE "ERROR: USERNAME IS TOO LONG." AND SET A = 0.

4. CHECK IF THE FIRST CHARACTER OF USER IS AN ALPHABET OR NOT:

A. IF THE FIRST CHARACTER IS NOT AN ALPHABET, PRINT AN ERROR MESSAGE "ERROR: FIRST CHARACTER MUST BE A LETTER." AND SET B = 0.

5. LOOP THROUGH ALL CHARACTERS OF USER:

A. IF THE CURRENT CHARACTER IS NOT AN ALPHANUMERIC CHARACTER OR AN UNDERSCORE,

I. IF THE CURRENT CHARACTER IS A SPACE, PRINT AN ERROR MESSAGE "ERROR: SPACE NOT ALLOWED IN THE USERNAME." AND SET C = 0.

II. IF THE CURRENT CHARACTER IS NOT A SPACE, PRINT AN ERROR MESSAGE "ERROR: INVALID CHARACTER IN USERNAME. ONLY LETTERS, NUMBERS, AND _ ARE ALLOWED." AND SET C = 0.

6. IF A, B, AND C ARE ALL EQUAL TO 1, RETURN 1, INDICATING THAT THE USER IS VALID.

7. ELSE, RETURN 0, INDICATING THAT THE USER IS NOT VALID.

*ISUSERNAMETAKEN(CHAR *USER)*

1. CHECK IF THE FILE "ACCOUNTS.CSV" EXISTS.

- *A. IF IT EXISTS, GO TO STEP 2.*
- *B. IF IT DOESN'T EXIST, RETURN 0.*

2. OPEN THE FILE "ACCOUNTS.CSV" IN READ MODE AND ASSIGN THE FILE POINTER TO "FILE".

- *A. IF THE FILE CANNOT BE OPENED, PRINT AN ERROR MESSAGE AND RETURN -1.*

3. CREATE TWO STRINGS "CURRENTUSERNAME" AND "CURRENTPASSWORD" TO STORE THE CURRENT USERNAME AND PASSWORD BEING READ.

4. READ THE FILE LINE BY LINE USING “FSCANF” UNTIL THE END OF FILE IS REACHED.

5. SPLIT EACH LINE READ INTO THE USERNAME AND PASSWORD USING “FSCANF” AND STORE THEM IN "CURRENTUSERNAME" AND "CURRENTPASSWORD".

6. COMPARE THE "USER" ARGUMENT PASSED TO THE FUNCTION WITH THE "CURRENTUSERNAME".

- A. IF THEY MATCH, CLOSE THE FILE AND RETURN 1.

7. REPEAT STEPS 4-6 FOR EACH LINE UNTIL THE END OF FILE IS REACHED.

8. IF THE END OF THE FILE IS REACHED AND NO MATCHING USERNAME IS FOUND, CLOSE THE FILE AND RETURN 0.

*ISVALIDPASSWORD(CHAR *PASSWORD)*

ALGORITHM FOR ISVALIDPASSWORD(PASSWORD):

1. INITIALIZE LEN TO LENGTH OF THE PASSWORD.

2. CHECK IF LENGTH OF THE PASSWORD IS LESS THAN MIN_PASSWORD_LENGTH, IF YES PRINT ERROR MESSAGE.

3. INITIALIZE HASLOWER, HASUPPER, HASDIGIT AND HASSPECIAL AS FALSE.

4. ITERATE THROUGH THE PASSWORD STRING AND SET THE RESPECTIVE FLAG TO TRUE IF THE CHARACTER IS LOWERCASE, UPPERCASE, DIGIT, OR SPECIAL CHARACTER RESPECTIVELY.

5. CHECK IF HASLOWER, HASUPPER, HASDIGIT AND HASSPECIAL ARE FALSE AND PRINT RESPECTIVE ERROR MESSAGES IF ANY.

6. IF ALL FLAGS ARE TRUE, RETURN 1, ELSE RETURN 0.

LOGIN()

1. DECLARE VARIABLES FOR THE USERNAME, PASSWORD, FILE USERNAME, FILE PASSWORD, AND A FLAG FOR VALIDITY

2. PROMPT THE USER TO ENTER THEIR USERNAME

3. READ THE INPUTTED USERNAME AND STORE IT IN THE 'USER' VARIABLE

4. PROMPT THE USER TO ENTER THEIR PASSWORD

5. READ THE INPUTTED PASSWORD AND STORE IT IN THE 'PASSWORD' VARIABLE

6. OPEN THE "ACCOUNTS.CSV" FILE IN READ MODE AND ASSIGN IT TO THE FILE POINTER 'FP'

7. CHECK IF THE FILE IS SUCCESSFULLY OPENED. IF NOT, PRINT AN ERROR MESSAGE AND RETURN

8. READ THE FILE USING FSCANF, STORING THE USERNAME AND PASSWORD SEPARATED BY A COMMA INTO 'FILE_USER' AND 'FILE_PASSWORD' RESPECTIVELY

9. COMPARE THE 'FILE_USER' AND 'USER' VARIABLES. IF THEY MATCH, COMPARE 'FILE_PASSWORD' AND 'PASSWORD'. IF BOTH MATCH, SET 'VALID' FLAG TO 1 AND BREAK THE LOOP

10. REPEAT STEPS 8-9 UNTIL END OF FILE (EOF)

11. CLOSE THE FILE

12. CHECK THE VALUE OF THE 'VALID' FLAG. IF IT'S 1, PRINT A WELCOME MESSAGE WITH THE USERNAME AND CALL THE 'OPERATIONS' FUNCTION WITH THE USERNAME AS AN ARGUMENT. IF IT'S 0, PRINT AN ERROR MESSAGE

13. END THE FUNCTION

*OPERATIONS(CHAR *USER)*

*ALGORITHM FOR THE FUNCTION "OPERATIONS(CHAR *USER)":*

1. INITIALIZE AN ARRAY “DATA" OF SIZE 100 TO STORE RECORDS AND SET A VARIABLE “RECORDS” TO 0.

2. REPEAT THE FOLLOWING STEPS UNTIL USER CHOOSES TO EXIT (OPTION 7):

A. CALL THE "PRINTMENU" FUNCTION TO PRINT THE MENU OF OPTIONS.

B. READ USER'S CHOICE AS AN INTEGER.

C. PERFORM THE CORRESPONDING ACTION BASED ON THE USER'S CHOICE:

I. IF USER SELECTS OPTION 1, CALL THE "GENERATE_DATA" FUNCTION WITH "USER" AS ARGUMENT.

II. IF USER SELECTS OPTION 2, CALL THE "READFILE" FUNCTION WITH "DATA", 100, AND "USER" AS ARGUMENTS AND STORE THE RETURNED VALUE IN "RECORDS". IF "RECORDS" IS GREATER THAN 0, PRINT THE NUMBER OF LINES OF DATA READ.

III. IF USER SELECTS OPTION 3, CHECK IF "RECORDS" IS GREATER THAN 0. IF TRUE, CALL THE "PRINTDATA" FUNCTION WITH "DATA" AND "RECORDS" AS ARGUMENTS. IF FALSE, PRINT A MESSAGE INDICATING THAT THE USER NEEDS TO PERFORM STEPS 1 AND 2 FIRST.

IV. IF USER SELECTS OPTION 4, PRINT A MESSAGE EXPLAINING THE POINT CALCULATION MATRIX.

V. IF USER SELECTS OPTION 5, CHECK IF "RECORDS" IS GREATER THAN 0. IF TRUE, CALL THE "CALCULATEPOINTS" FUNCTION WITH "DATA" AND "RECORDS" AS ARGUMENTS, SORT THE DATA USING THE "SORTDATA" FUNCTION WITH "DATA" AND "RECORDS" AS ARGUMENTS, AND PRINT THE CALCULATED POINTS FOR EACH USER. IF FALSE, PRINT A MESSAGE INDICATING THAT THE USER NEEDS TO PERFORM STEPS 1 AND 2 FIRST.

VI. IF USER SELECTS OPTION 6, CHECK IF "RECORDS" IS GREATER THAN 0. IF TRUE, CALL THE "CALCULATEPOINTS" FUNCTION WITH "DATA" AND "RECORDS" AS ARGUMENTS, SORT THE DATA USING THE "SORTDATA" FUNCTION WITH "DATA" AND "RECORDS" AS ARGUMENTS, AND PRINT THE TOP 5 USERS WITH THE HIGHEST POINTS USING THE "PRINTTOP5" FUNCTION WITH "DATA" AND "RECORDS" AS ARGUMENTS. IF FALSE, PRINT A MESSAGE INDICATING THAT THE USER NEEDS TO PERFORM STEPS 1 AND 2 FIRST.

VII. IF USER SELECTS OPTION 7, RETURN 0.

VIII. IF USER SELECTS AN INVALID OPTION, PRINT A MESSAGE INDICATING THE INVALID OPTION.

3. RETURN 0.

ALGORITHM FOR "GENERATE_DATA" FUNCTION:

1. CHECK IF THE FOLDER SPECIFIED BY "USER" ARGUMENT EXISTS USING "ACCESS" FUNCTION.

IF IT EXISTS, PRINT "DATA ALREADY EXISTS." AND RETURN.

2. CREATE THE FOLDER USING "SYSTEM" FUNCTION WITH A COMMAND STRING BUILT USING "SPRINTF" WITH "USER" ARGUMENT.

IF THE "SYSTEM" CALL FAILS, PRINT "ERROR: FAILED TO CREATE FOLDER." AND RETURN.

3. BUILD THE FILE NAME BY CONCATENATING "USER" AND "/RECENT-ACTIVITIES.CSV".

4. OPEN THE FILE FOR WRITING USING "FOPEN" AND STORE THE FILE POINTER IN A "FILE" TYPE VARIABLE.

IF THE FILE FAILS TO OPEN, PRINT AN ERROR MESSAGE AND RETURN.

5. FOR "NUM_ROWS" ITERATIONS DO THE FOLLOWING:

A. GENERATE A RANDOM NAME USING "SPRINTF" BY COMBINING A RANDOM "FIRST_NAMES" AND "SURNAMES" ELEMENT.

B. GENERATE 5 RANDOM VALUES FOR "LIKES", "COMMENTS", "CHATS", "PHOTOS", AND "COMMENTS_MENTIONED".

C. WRITE THE GENERATED VALUES TO THE FILE AS A COMMA-SEPARATED STRING.

6. CLOSE THE FILE USING "FCLOSE".

7. PRINT "DATA HAS BEEN GENERATED SUCCESSFULLY!"

READFILE(DATA DATA[]*, INT MAXRECORDS, CHAR *USER)*

1. DEFINE A CHAR ARRAY "FILENAME" WITH A SIZE OF 200 AND AN INT "READ" AND "RECORDS".

2. CREATE A STRING USING SPRINTF FUNCTION WITH "FILENAME" AS THE FIRST ARGUMENT, "%S/RECENT-ACTIVITIES.CSV" AS THE SECOND ARGUMENT, AND "USER" AS THE THIRD ARGUMENT.

3. OPEN THE FILE USING FOPEN WITH "FILENAME" AS THE FIRST ARGUMENT AND "R" AS THE SECOND ARGUMENT AND STORE THE RESULT IN "FILE".

4. CHECK IF THE FILE POINTER IS NULL, IF TRUE THEN PRINT "GENERATE DATA THEN READ IT." AND RETURN -1.

5. READ THE DATA FROM THE FILE USING FSCANF FUNCTION WITH "FILE" AS THE FIRST ARGUMENT, AND THE FORMAT STRING "%30[^,],%D,%D,%D,%D,%D\N". STORE THE RESULT IN "READ".

6. CHECK IF "READ" IS EQUAL TO 6, IF TRUE INCREMENT "RECORDS".

7. CHECK IF "READ" IS NOT EQUAL TO 6 AND FILE END-OF-FILE IS NOT REACHED, IF TRUE THEN PRINT "FILE FORMAT INCORRECT." AND RETURN -1.

8. CHECK IF THERE IS AN ERROR READING THE FILE USING FERROR FUNCTION, IF TRUE THEN PRINT "ERROR READING FILE." AND RETURN -1.

9. REPEAT STEPS 5 TO 8 UNTIL THE END-OF-FILE IS REACHED.

10. CLOSE THE FILE USING FCLOSE WITH "FILE" AS THE ARGUMENT.

11. RETURN "RECORDS".

ALGORITHM FOR THE FUNCTION "CALCULATEPOINTS(DATA DATA[], INT RECORDS)"

1. INITIALIZE A VARIABLE "I" AS 0

2. REPEAT THE FOLLOWING STEPS FOR "I" RANGING FROM 0 TO "RECORDS-1"

A. CALCULATE THE POINTS FOR EACH DATA OBJECT AS FOLLOWS:

*I. DATA[I].POINTS = DATA[I].LIKES + DATA[I].COMMENTS * 1.5 + DATA[I].CHATS / 5 + DATA[I].TAGGED_PHOTOS * 2 + DATA[I].COMMENTS_MENTIONED * 3;*

B. INCREMENT "I" BY 1

3. END THE LOOP WHEN "I" IS EQUAL TO "RECORDS"

ALGORITHM: SORTDATA

1. INITIALIZE TWO LOOP COUNTERS "I" AND "J" AND SET "I" TO 0 AND "J" TO "I" + 1.

2. CHECK IF DATA[I].POINTS IS LESS THAN DATA[J].POINTS.

3. IF DATA[I].POINTS IS LESS THAN DATA[J].POINTS, SWAP DATA[I] AND DATA[J] USING A TEMPORARY VARIABLE "TEMP".

4. INCREMENT "J" BY 1.

5. REPEAT STEPS 2 TO 4 UNTIL "J" IS LESS THAN "RECORDS".

6. INCREMENT "I" BY 1.

7. REPEAT STEPS 2 TO 6 UNTIL "I" IS LESS THAN "RECORDS" - 1.

8. END THE ALGORITHM.

ALGORITHM FOR "PRINTTOP5" FUNCTION:

1. INITIALIZE INTEGER VARIABLE "N" TO "5".

2. IF "RECORDS" IS LESS THAN "5", SET "N" TO "RECORDS".

3. PRINT MESSAGE "THESE ARE YOUR N CLOSEST FRIENDS:"

4. LOOP FROM "I"=0 TO "N-1" (INCLUSIVE)

A. PRINT THE RANK, USERNAME AND POINTS OF THE I-TH RECORD IN THE FORMAT "I+1. USERNAME - POINTS POINTS"

5. END THE LOOP.

END

RESULTS AND ANALYSIS WITH SOME DIFFERENT TERMINAL CASES: (CASE: 1)

```
PS C:\Users\Anwarul> cd "c:\Socio"
PS C:\Socio> & .\"Code_V2.exe"
--------------------------------------------------------------------------------------------------
MENU
1. Create account
2. Login
3. Exit
--------------------------------------------------------------------------------------------------
Enter an option: 1
--------------------------------------------------------------------------------------------------
Enter a username: Anwarul2002
--------------------------------------------------------------------------------------------------
Enter a password: Anwarul@1234

Account created successfully
--------------------------------------------------------------------------------------------------
MENU
1. Create account
2. Login
3. Exit
--------------------------------------------------------------------------------------------------
Enter an option: 2
--------------------------------------------------------------------------------------------------
Enter your username: Anwarul2002
Enter your password: Anwarul@1234
--------------------------------------------------------------------------------------------------
Welcome, Anwarul2002!
--------------------------------------------------------------------------------------------------

MENU
1. Generate Data
2. Read Data
3. View Data
4. View Point Calculation Criteria
5. View Calculated Points
6. View Result
7. Logout
--------------------------------------------------------------------------------------------------
Select an option:1
--------------------------------------------------------------------------------------------------
Data has been generated successfully!
--------------------------------------------------------------------------------------------------
MENU
1. Generate Data
2. Read Data
3. View Data
4. View Point Calculation Criteria
5. View Calculated Points
6. View Result
7. Logout
--------------------------------------------------------------------------------------------------
Select an option:6
--------------------------------------------------------------------------------------------------
1st Perform these 2 steps:
                Generate Data
                Read Data
--------------------------------------------------------------------------------------------------

MENU
1. Generate Data
2. Read Data
3. View Data
4. View Point Calculation Criteria
5. View Calculated Points
6. View Result
7. Logout
--------------------------------------------------------------------------------------------------
Select an option:2
--------------------------------------------------------------------------------------------------
100 line of data has been read.
--------------------------------------------------------------------------------------------------
MENU
1. Generate Data
2. Read Data
3. View Data
4. View Point Calculation Criteria
5. View Calculated Points
6. View Result
7. Logout
--------------------------------------------------------------------------------------------------
Select an option:6
--------------------------------------------------------------------------------------------------

These are your 5 closest friends:
1. Anjali Tripathi - 113 points
2. Prashant Tripathi - 106 points
3. Rajesh Kumar - 103 points
4. Aarti Anand - 102 points
5. Ravi Shukla - 101 points
--------------------------------------------------------------------------------------------------
```

```
--------------------------------------------------------------------------------
MENU
1. Generate Data
2. Read Data
3. View Data
4. View Point Calculation Criteria
5. View Calculated Points
6. View Result
7. Logout
--------------------------------------------------------------------------------
Select an option:7
--------------------------------------------------------------------------------
--------------------------------------------------------------------------------
MENU
1. Create account
2. Login
3. Exit
--------------------------------------------------------------------------------
Enter an option: 3
--------------------------------------------------------------------------------
Exiting...
```

(CASE: 2)

```
PS C:\Socio> cd "c:\Socio"
PS C:\Socio> & .\"Code_V2.exe"
--------------------------------------------------------------------------------
MENU
1. Create account
2. Login
3. Exit
--------------------------------------------------------------------------------
Enter an option: 1
--------------------------------------------------------------------------------
Enter a username: Piyush2002
--------------------------------------------------------------------------------
Enter a password: piyush@2002
Error: Password must contain at least one uppercase letter.
Invalid password
--------------------------------------------------------------------------------
re-enter a password: Piyush@2002

Account created successfully
--------------------------------------------------------------------------------
MENU
1. Create account
2. Login
3. Exit
--------------------------------------------------------------------------------
Enter an option: 2
--------------------------------------------------------------------------------
Enter your username: Piyush2002
Enter your password: Piyush@2002
--------------------------------------------------------------------------------
Welcome, Piyush2002!
```

```
--------------------------------------------------------------------------------------------
MENU
1. Generate Data
2. Read Data
3. View Data
4. View Point Calculation Criteria
5. View Calculated Points
6. View Result
7. Logout
--------------------------------------------------------------------------------------------
Select an option:1
--------------------------------------------------------------------------------------------
Data has been generated successfully!
--------------------------------------------------------------------------------------------
MENU
1. Generate Data
2. Read Data
3. View Data
4. View Point Calculation Criteria
5. View Calculated Points
6. View Result
7. Logout
--------------------------------------------------------------------------------------------
Select an option:2
--------------------------------------------------------------------------------------------
100 line of data has been read.
--------------------------------------------------------------------------------------------
```

```
MENU
1. Generate Data
2. Read Data
3. View Data
4. View Point Calculation Criteria
5. View Calculated Points
6. View Result
7. Logout
--------------------------------------------------------------------------------------------
Select an option:6
--------------------------------------------------------------------------------------------

These are your 5 closest friends:
1. Prashant Malhotra - 113 points
2. Anil Sinha - 107 points
3. Megha Aggarwal - 104 points
4. Amit Gupta - 101 points
5. Jyoti Sharma - 98 points
--------------------------------------------------------------------------------------------
MENU
1. Generate Data
2. Read Data
3. View Data
4. View Point Calculation Criteria
5. View Calculated Points
6. View Result
7. Logout
--------------------------------------------------------------------------------------------
Select an option:7
--------------------------------------------------------------------------------------------
--------------------------------------------------------------------------------------------
MENU
1. Create account
2. Login
3. Exit
--------------------------------------------------------------------------------------------
Enter an option: 3
--------------------------------------------------------------------------------------------
Exiting...
```

APPENDIX

CODE:

```
#include <stdio.h>
#include <stdlib.h>
#include <string.h>
#include <ctype.h>
#include <time.h>
#include <unistd.h>


#define MIN_LENGTH 8
#define MAX_LENGTH 30
#define MIN_PASSWORD_LENGTH 8
#define MAX_USERNAME_LENGTH 50
#define MAX_PASSWORD_LENGTH 50
#define NUM_ROWS 100
#define NUM_COLUMNS 6


typedef struct
{
    char username[100];
    int likes;
    int comments;
    int chats;
    int tagged_photos;
    int comments_mentioned;
    int points;
} Data;
```

```c
void create_account();
int isValidUsername(char *user);
int isUsernameTaken(char *user);
int isValidPassword(char *password);


void login();


int operations(char *user);
void printMenu();
void generate_data(char *user);
int readFile(Data data[], int maxRecords, char *user);
void printData(Data data[], int records);
void calculatePoints(Data data[], int records);
void sortData(Data data[], int records);
void printTop5(Data data[], int records);


int main()
{
    int option;
    while (1)
    {
        printf("-----------------------------------------------------------------
------------------------\n");
        printf("MENU\n");
        printf("1. Create account\n");
        printf("2. Login\n");
        printf("3. Exit\n");
        printf("-----------------------------------------------------------------
------------------------\n");
        printf("Enter an option: ");
```

```
        fflush(stdin);
        scanf("%d", &option);
        printf("-----------------------------------------------------------------
------------------------\n");
        switch (option)
        {
        case 1:
            create_account();
            break;
        case 2:
            login();
            break;
        case 3:
            printf("Exiting...\n");
            exit(0);
            break;
        default:
            printf("Invalid option, try again.\n");
            break;
        }
    }


    return 0;
}


void create_account()
{
    char user[MAX_USERNAME_LENGTH];
    char password[MAX_PASSWORD_LENGTH];
    printf("Enter a username: ");
```

```
    fflush(stdin);
    scanf("%[^\n]", user);
    while (!isValidUsername(user))
    {
        printf("Invalid username\n");
        printf("------------------------------------------------------------------
------------------------\n");
        printf("re-enter a username: ");

        fflush(stdin);
        scanf("%[^\n]", user);
    }

    while (isUsernameTaken(user))
    {
        printf("Sorry, '%s' is already taken.\n", user);
        printf("------------------------------------------------------------------
------------------------\n");
        printf("Please try a new one: ");
        fflush(stdin);
        scanf("%[^\n]", user);
    }

    printf("------------------------------------------------------------------
--------------------\n");
    printf("Enter a password: ");
    fflush(stdin);
    scanf("%[^\n]", password);
    while (!isValidPassword(password))
    {
```

```
        printf("Invalid password\n");
        printf("-----------------------------------------------------------------
------------------------\n");
        printf("re-enter a password: ");
        fflush(stdin);
        scanf("%[^\n]", password);
    }
    FILE *fp;
    fp = fopen("accounts.csv", "a");
    fprintf(fp, "%s,%s\n", user, password);
    printf("\nAccount created successfully\n");
    fclose(fp);
}


int isValidUsername(char *user)
{
    int len = strlen(user);
    int a, b, c = 1;
    if (len < MIN_LENGTH || len > MAX_LENGTH)
    {
        if (len < MIN_LENGTH)
        {
            printf("Error: Username is too short. MIN_LENGTH = %d and
MAX_LENGTH = %d\n", MIN_LENGTH, MAX_LENGTH);
            a = 0;
        }
        else
        {
            printf("Error: Username is too long. MIN_LENGTH = %d and
MAX_LENGTH = %d\n", MIN_LENGTH, MAX_LENGTH);
        }
    }
```

```
    if (!isalpha(user[0]))
    {
        printf("Error: First character must be a letter.\n");
        b = 0;
    }
    for (int i = 0; i < len; i++)
    {
        if (!isalnum(user[i]) && user[i] != '_')
        {
            if (isspace(user[i]))
            {
                printf("Error: Space not allowed in the username.\n");
                c = 0;
            }
            else
            {
                printf("Error: Invalid character '%c' in username. Only
letters, numbers and _ are allowed.\n", user[i]);
                c = 0;
            }
        }
    }
    if (!(a && b && c))
    {
        return 0;
    }


    return 1;
}
```

```c
int isUsernameTaken(char *user)
{
    if (access("accounts.csv", F_OK) != -1)
    {
        FILE *file = fopen("accounts.csv", "r");
        if (file == NULL)
        {
            printf("Error opening file.\n");
            return -1;
        }
        char currentUsername[100], currentPassword[100];
        while (fscanf(file, "%30[^,],%s\n", currentUsername, currentPassword)
== 2)
        {
            if (strcmp(currentUsername, user) == 0)
            {
                fclose(file);
                return 1;
            }
        }
        fclose(file);
        return 0;
    }
    else
    {
        return 0;
    }
}


const char special_characters[] = "!@#$%^&*()-+";
```

```c
int isValidPassword(char *password)
{
    int len = strlen(password);
    if (len < MIN_PASSWORD_LENGTH)
    {
        printf("Error: Password is too short. It must be at least %d characters long.\n", MIN_PASSWORD_LENGTH);
    }

    int hasLower = 0, hasUpper = 0, hasDigit = 0, hasSpecial = 0;

    for (int i = 0; i < len; i++)
    {
        if (islower(password[i]))
            hasLower = 1;
        else if (isupper(password[i]))
            hasUpper = 1;
        else if (isdigit(password[i]))
            hasDigit = 1;
        else if (strchr(special_characters, password[i]) != NULL)
            hasSpecial = 1;
    }
    if (!hasLower)
        printf("Error: Password must contain at least one lowercase letter.\n");
    if (!hasUpper)
        printf("Error: Password must contain at least one uppercase letter.\n");
    if (!hasDigit)
```

```
        printf("Error: Password must contain at least one digit.\n");
    if (!hasSpecial)
        printf("Error: Password must contain at least one special
character.\n");


    if (hasLower && hasUpper && hasDigit && hasSpecial)
    {
        return 1;
    }
    return 0;
}


void login()
{
    char user[MAX_USERNAME_LENGTH];
    char password[MAX_PASSWORD_LENGTH];
    char file_user[MAX_USERNAME_LENGTH];
    char file_password[MAX_PASSWORD_LENGTH];
    int valid = 0;
    printf("Enter your username: ");
    fflush(stdin);
    scanf("%[^\n]", user);
    printf("Enter your password: ");
    fflush(stdin);
    scanf("%[^\n]", password);


    FILE *fp;
    fp = fopen("accounts.csv", "r");
    if (fp == NULL)
    {
```

```
        printf("\nError opening file.\n");
        return;
    }
    while (fscanf(fp, "%30[^,],%s\n", file_user, file_password) != EOF)
    {
        if (strcmp(file_user, user) == 0)
        {
            if (strcmp(file_password, password) == 0)
            {
                valid = 1;
                break;
            }
        }
    }
    fclose(fp);


    if (valid)
    {
        printf("-----------------------------------------------------------------
------------------------\n");
        printf("Welcome, %s!\n", user);
        operations(user);
    }
    else
    {
        printf("Invalid username or password.\n");
    }
}


int operations(char *user)
```

```
{
    Data data[100];
    int records = 0;
    int choice;
    while (1)
    {
        printMenu();
        fflush(stdin);
        scanf("%d", &choice);
        printf("------------------------------------------------------------------
------------------------\n");
        switch (choice)
        {
        case 1:
            generate_data(user);
            break;
        case 2:
            records = readFile(data, 100, user);
            if (records > 0)
            {
                printf("%d line of data has been read.\n", records);
            }
            break;
        case 3:
            if (records > 0)
            {
                printData(data, records);
            }
            else
            {
                printf("1st Perform these 2 steps:\n\t\tGenerate
Data\n\t\tRead Data\n");
```

```
            }
            break;
        case 4:
            printf("Point Calculation Matrix:\n");
            printf("1 Like = 1 point\n");
            printf("1 Comment = 1.5 points\n");
            printf("1 Chat = 0.2 points\n");
            printf("1 Tagged photo = 2 points\n");
            printf("1 Comment mentioned = 3 points\n");
            break;
        case 5:
            if (records > 0)
            {
                calculatePoints(data, records);
                sortData(data, records);
                printf("\nCalculated Points:\n");
                printf("------------------------------\n");
                printf("|      %-14s| %-7s|\n", "Username", "Points");
                printf("------------------------------\n");
                for (int i = 0; i < records; i++)
                    printf("|  %-18s|   %-5d|\n", data[i].username,
data[i].points);
                printf("------------------------------\n");
            }
            else
            {
                printf("1st Perform these 2 steps:\n\t\tGenerate
Data\n\t\tRead Data\n");
            }
            break;
        case 6:
            if (records > 0)
```

```
            {
                calculatePoints(data, records);
                sortData(data, records);
                printTop5(data, records);
            }
            else
            {
                printf("1st Perform these 2 steps:\n\t\tGenerate
Data\n\t\tRead Data\n");
            }
            break;
        case 7:
            return 0;
            break;
        default:
            printf("\nInvalid option selected. Please try again.\n");
            break;
        }
    }
    return 0;
}


void printMenu()
{
    printf("-----------------------------------------------------------------
--------------------\n");
    printf("MENU\n");
    printf("1. Generate Data\n");
    printf("2. Read Data\n");
    printf("3. View Data\n");
    printf("4. View Point Calculation Criteria\n");
```

```
    printf("5. View Calculated Points\n");
    printf("6. View Result\n");
    printf("7. Logout\n");
    printf("------------------------------------------------------------------
--------------------\n");
    printf("Select an option:");
}


const char *first_names[] = {
    "Amit", "Anil", "Anjali", "Ankita", "Asha", "Aarti",
    "Dinesh", "Jasmine", "Jyoti", "Kamal", "Manish", "Megha",
    "Monika", "Mukesh", "Nandini", "Neha", "Nidhi", "Praveen",
    "Preeti", "Priyanka", "Prashant", "Rajat", "Rajesh", "Rakesh",
    "Rashmi", "Reena", "Ravi", "Sanjay", "Sanjeev", "Shilpa",
    "Shobha", "Shruti", "Sneha", "Suresh", "Vikas", "Vinod"};


const char *surnames[] = {
    "Sharma", "Devi", "Singh", "Kumar", "Yadav", "Gupta",
    "Verma", "Chaudhary", "Jain", "Aggarwal", "Rao", "Patel",
    "Bhatia", "Choudhary", "Bhardwaj", "Tripathi", "Kapoor", "Mehta",
    "Dixit", "Jha", "Prasad", "Anand", "Malhotra", "Arora",
    "Chauhan", "Rajput", "Sinha", "Mathur", "Varma", "Chawla",
    "Khanna", "Meena", "Khatri", "Nath", "Soni", "Bajaj",
    "Pandey", "Rastogi", "Goel", "Desai", "Chopra", "Dutta",
    "Sethi", "Bose", "Jadhav", "Shah", "Shukla", "Thakur", "Bedi"};


void generate_data(char *user)
{
    srand(time(NULL));
```

```c
if (access(user, F_OK) != -1)
{
    printf("Data already exists.\n");
    return;
}
else
{

    char command[200];
    sprintf(command, "mkdir %s", user);
    if (system(command) != 0)
    {
        printf("Error: Failed to create folder.\n");
        return;
    }

    char fileName[200];
    sprintf(fileName, "%s/recent-activities.csv", user);

    FILE *file = fopen(fileName, "w");
    if (file == NULL)
    {
        perror("Error opening file");
        return;
    }

    for (int i = 0; i < NUM_ROWS; i++)
```

```
        {
            char name[32];
            sprintf(name, "%s %s", first_names[rand() % (sizeof(first_names) /
sizeof(char *))],
                    surnames[rand() % (sizeof(surnames) / sizeof(char *))]);
            int likes = rand() % 50;
            int comments = rand() % 30;
            int chats = (i < 20) ? (70 + rand() % 29) : (rand() % 50);
            int photos = rand() % 8;
            int comments_mentioned = rand() % 8;
            fprintf(file, "%s,%d,%d,%d,%d,%d\n", name, likes, comments, chats,
photos, comments_mentioned);
        }


        fclose(file);


        printf("Data has been generated successfully!\n");
    }
}


int readFile(Data data[], int maxRecords, char *user)
{
    char fileName[200];
    sprintf(fileName, "%s/recent-activities.csv", user);
    FILE *file = fopen(fileName, "r");
    if (file == NULL)
    {
        printf("\n1st Generate data then read it.\n");
        return -1;
    }
```

```
    int read = 0;
    int records = 0;


    do
    {
        read = fscanf(file, "%30[^,],%d,%d,%d,%d,%d\n",
data[records].username,
                                                    &data[records].likes,

&data[records].comments,
                                                    &data[records].chats,

&data[records].tagged_photos,

&data[records].comments_mentioned);
        if (read == 6)
            records++;
        if (read != 6 && !feof(file))
        {
            printf("\nFile format incorrect.\n");
            return -1;
        }
        if (ferror(file))
        {
            printf("\nError reading file.\n");
            return -1;
        }
    } while (!feof(file));


    fclose(file);
```

```
    return records;
}


void printData(Data data[], int records)
{
    printf("\nData : \n");
    printf("-------------------------------------------------------------------
--------------------\n");
    printf("|      %-14s| %-6s| %-9s| %-6s| %-14s| %-19s|\n", "Username",
                                                              "Likes",
                                                              "Comments",
                                                              "Chats",
                                                              "Tagged photos",
                                                              "Comments
mentioned");
    printf("-------------------------------------------------------------------
--------------------\n");
    for (int i = 0; i < records; i++)
        printf("|  %-18s|  %-5d|    %-6d|  %-5d|       %-8d|         %-
11d|\n", data[i].username,

data[i].likes,

data[i].comments,

data[i].chats,

data[i].tagged_photos,

data[i].comments_mentioned);
    printf("-------------------------------------------------------------------
--------------------\n");
}
```

```
void calculatePoints(Data data[], int records)
{
    for (int i = 0; i < records; i++)
    {
        data[i].points = data[i].likes +
                         data[i].comments * 1.5 +
                         data[i].chats / 5 +
                         data[i].tagged_photos * 2 +
                         data[i].comments_mentioned * 3;
    }
}


void sortData(Data data[], int records)
{
    for (int i = 0; i < records - 1; i++)
    {
        for (int j = i + 1; j < records; j++)
        {
            if (data[i].points < data[j].points)
            {
                Data temp = data[i];
                data[i] = data[j];
                data[j] = temp;
            }
        }
    }
}
```

```c
void printTop5(Data data[], int records)
{
    int n = 5;
    if (records < 5)
    {
        n = records;
    }
    printf("\nThese are your %d closest friends:\n", n);
    for (int i = 0; i < n; i++)
    {
        printf("%d. %s - %d points\n", i + 1, data[i].username,
data[i].points);
    }
}
```