SOCIAL NETWORKING MANAGEMENT

QUESTION:-

Social Network Management System:

Data Structures: Graph (adjacency list) for user connections, linked list for user profiles.

Functionality:

Create user profiles (name, interests, etc.).

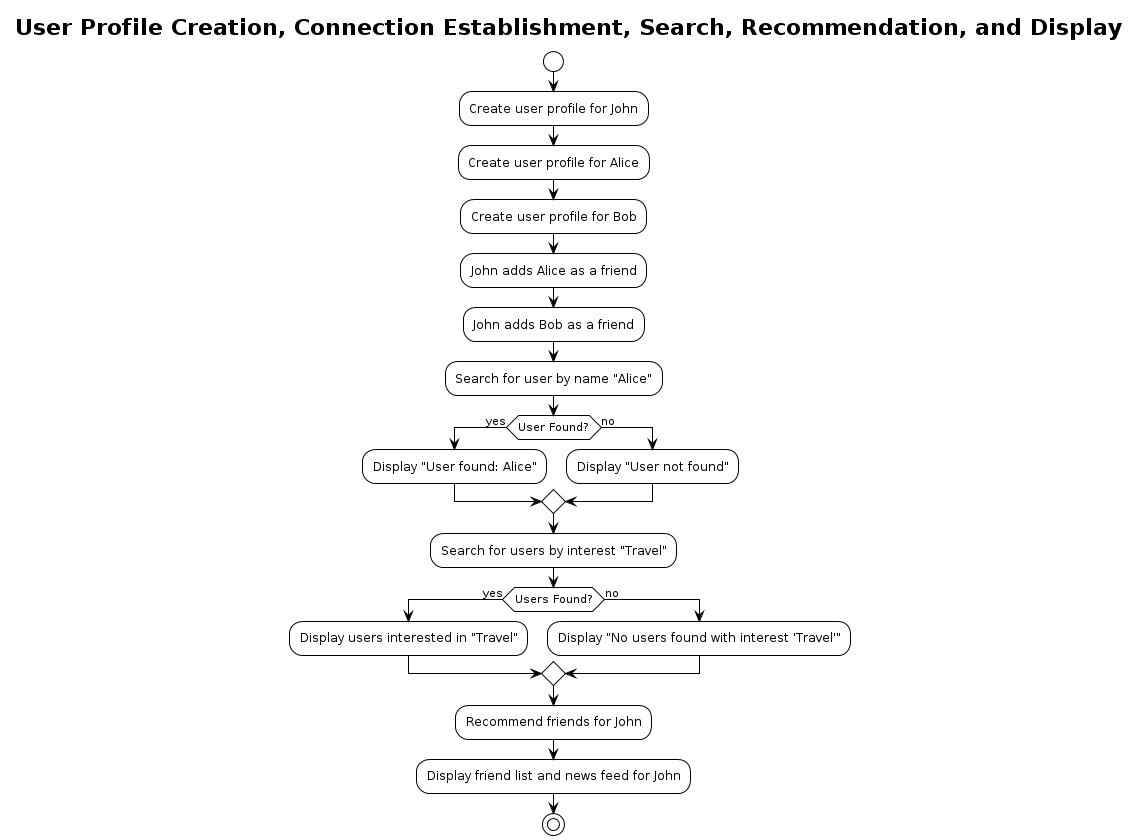
Add friends (connect users in the graph).

Search for users by name or interest.

Recommend friends based on mutual connections or interests.

Display a user's friend list and news feed (simulated data or integration with an external API).

FLOWCHART:-



MODULAR CODE

network.h

#ifndef SOCIAL\_NETWORK\_H

#define SOCIAL\_NETWORK\_H

#define MAX\_USERS 100

#define MAX\_NAME\_LENGTH 50

#define MAX\_INTEREST\_LENGTH 100

typedef struct User

{

char name[MAX\_NAME\_LENGTH];

char interests[MAX\_INTEREST\_LENGTH];

struct User \*friends[MAX\_USERS];

int num\_friends;

} User;

typedef struct Graph {

User users[MAX\_USERS];

int num\_users;

} Graph;

void add\_user(Graph \*graph, const char \*name, const char \*interests);

void add\_friendship(Graph \*graph, const char \*user1\_name, const char \*user2\_name);

void search\_users\_by\_name(Graph \*graph, const char \*name);

void search\_users\_by\_interest(Graph \*graph, const char \*interest);

void recommend\_friends(Graph \*graph, const char \*name);

void display\_user\_friends(User \*user);

void display\_news\_feed(User \*user);

#endif

Netwok.c

#include <stdio.h>

#include <string.h>

#include "social\_network.h"

void add\_user(Graph \*graph, const char \*name, const char \*interests) {

if (graph->num\_users >= MAX\_USERS) {

printf("Cannot add more users.\n");

return;

}

User \*user = &graph->users[graph->num\_users++];

strcpy(user->name, name);

strcpy(user->interests, interests);

user->num\_friends = 0;

}

void add\_friendship(Graph \*graph, const char \*user1\_name, const char \*user2\_name) {

User \*user1 = NULL, \*user2 = NULL;

for (int i = 0; i < graph->num\_users; i++) {

if (strcmp(graph->users[i].name, user1\_name) == 0) {

user1 = &graph->users[i];

}

if (strcmp(graph->users[i].name, user2\_name) == 0) {

user2 = &graph->users[i];

}

}

if (user1 && user2) {

user1->friends[user1->num\_friends++] = user2;

user2->friends[user2->num\_friends++] = user1;

}

}

void search\_users\_by\_name(Graph \*graph, const char \*name) {

printf("Users with name '%s':\n", name);

for (int i = 0; i < graph->num\_users; i++) {

if (strstr(graph->users[i].name, name)) {

printf("%s\n", graph->users[i].name);

}

}

}

void search\_users\_by\_interest(Graph \*graph, const char \*interest) {

printf("Users interested in '%s':\n", interest);

for (int i = 0; i < graph->num\_users; i++) {

if (strstr(graph->users[i].interests, interest)) {

printf("%s\n", graph->users[i].name);

}

}

}

void recommend\_friends(Graph \*graph, const char \*name) {

printf("Recommended friends for %s:\n", name);

for (int i = 0; i < graph->num\_users; i++) {

if (strcmp(graph->users[i].name, name) == 0) {

User \*user = &graph->users[i];

for (int j = 0; j < user->num\_friends; j++) {

User \*friend = user->friends[j];

for (int k = 0; k < friend->num\_friends; k++) {

User \*potential\_friend = friend->friends[k];

if (strcmp(potential\_friend->name, name) != 0 &&

!is\_friend\_of\_user(user, potential\_friend)) {

printf("%s\n", potential\_friend->name);

}

}

}

break;

}

}

}

bool is\_friend\_of\_user(User \*user, User \*potential\_friend) {

for (int i = 0; i < user->num\_friends; i++) {

if (user->friends[i] == potential\_friend) {

return true;

}

}

return false;

}

Networkmain.c

#include <stdio.h>

#include <stdlib.h>

#include "social\_network.h"

#include <time.h>

int main() {

Graph social\_network;

social\_network.num\_users = 0;

// Create users

add\_user(&social\_network, "Alice", "coding, reading");

add\_user(&social\_network, "Bob", "sports, cooking");

add\_user(&social\_network, "Charlie", "coding, gaming");

// Add friendships

add\_friendship(&social\_network, "Alice", "Bob");

add\_friendship(&social\_network, "Bob", "Charlie");

// Search for users

search\_users\_by\_name(&social\_network, "Alice");

search\_users\_by\_interest(&social\_network, "coding");

// Recommend friends

recommend\_friends(&social\_network, "Alice");

return 0;

}

EXPLINATION:-

Each user profile can be represented by a node in a graph.

Additionally, user profile information such as name, interests, etc., can be stored in a linked list or a similar data structure attached to each node.

Graph for User Connections:

The user connections can be represented using a graph data structure with an adjacency list implementation.

Each user is a node in the graph, and edges represent connections (friendships) between users.

The adjacency list for each user node contains references to other users to whom they are connected.

Creating User Profiles:

When creating a user profile, you would add a new node to the graph to represent the user.

The user's profile information can be stored in a linked list or similar data structure attached to the user node.

Adding Friends:

Adding a friend connection between two users involves adding an edge between their respective nodes in the graph.

Searching for Users:

Searching for users by name or interest involves traversing the graph and looking for nodes that match the search criteria.

This can be done using various graph traversal algorithms like depth-first search (DFS) or breadth-first search (BFS).

Recommend Friends:

Friend recommendations can be based on mutual connections or shared interests.

To recommend friends based on mutual connections, you can look for users who are connected to multiple mutual friends.

To recommend friends based on shared interests, you can look for users who have similar interests to the target user.

Displaying Friend List and News Feed:

A user's friend list can be obtained by traversing the adjacency list of their node in the graph.

The news feed can be simulated data generated within the system or integrated with an external API to fetch real-time updates from friends.

By implementing these functionalities, you can create a Social Network Management System that allows users to connect with each other, search for other users, receive friend recommendations, and interact with their friends' content.

OUTPUT:-

