Developer Documentation for Facebook Community Management System.

**Solution Explanation**

The Facebook Community Management System is a program designed to manage a community of members. The system supports functionalities like adding new members, searching for members, deleting members, listing all members, and listing members invited by a specific user. The program stores the member data persistently in a text file called “community\_db.txt”, ensuring data is saved and loaded across sessions.

The program utilizes a linked list for dynamic management of the members in memory.

**Modules**

1. **Main Module**

Responsible for user interaction and driving the program through a menu system.

1. **Database Management Module**

Functions to load and save the member database from/to the file system.

1. **Member Management Module**

Functions to add, search, delete and list members, ensuring efficient management and query operations.

1. **Memory Management Module**

Handles memory allocation and cleanup to prevent memory leaks.

**Data Structures**

The Program uses Struct and Linked List.

**Member (Struct)**

The primary data structure that represents each member. It includes:

**username:** Unique identifier for the member.

**familyName:** Member's family name.

**givenName:** Member's given name.

**birthYear, birthMonth, birthDay:** Date of birth.

**birthPlace:** Place of birth.

**inviter**: Username of the person who invited the member.

**next**: Pointer to the next Member in the linked list.

**Algorithms**

1. **Linked List for Member Management**
2. Dynamic allocation for members.
3. New members are added at the head of the list for efficient insertion.
4. Traversal is used for searching, listing, and deleting members.
5. **File Operations for Persistence**
6. The file is read sequentially to recreate the linked list on program startup.
7. The list is saved to the file at shutdown or upon explicit user request.

**List of Functions and Their Interface**

1. **Member \*loadDatabase()**
   * Input: None.
   * Output: Returns the head pointer to the linked list of members.
   * Description: Loads member data from “community\_db.txt” and constructs the linked list.
2. **void saveDatabase(Member \*head)**
   * Input: head (Pointer to the head of the linked list).
   * Output: None.
   * Description: Writes all members' data “to community\_db.txt”.
3. **Member \*addMember(Member \*head)**
   * Input: head (Pointer to the head of the linked list).
   * Output: Returns the updated head pointer.
   * Description: Collects input for a new member, adds it to the list, and returns the updated list.
4. **void searchMember(Member \*head)**
   * Input: head (Pointer to the head of the linked list).
   * Output: None.
   * Description: Searches for a member by username or family name and prints details.
5. **Member \*deleteMember(Member \*head)**
   * Input: head (Pointer to the head of the linked list).
   * Output: Returns the updated head pointer.
   * Description: Deletes a member by username and adjusts the list accordingly.
6. **void listInvitedMembers(Member \*head)**
   * Input: head (Pointer to the head of the linked list).
   * Output: None.
   * Description: Lists all members invited by a specific username.
7. **void listAllMembers(Member \*head)**
   * Input: head (Pointer to the head of the linked list).
   * Output: None.
   * Description: Lists all members with their details.
8. **void freeMemory(Member \*head)**
   * Input: head (Pointer to the head of the linked list).
   * Output: None.
   * Description: Frees all memory allocated for the linked list.

**How To Run the Project**

**Prerequisites:**

1. **C compiler. Ensure a C compiler (e.g., GCC) is installed.**
2. **Text Editor or IDE. Any text editor or IDE for c programming (e.g., Visual Studio Code)**

**Steps to Compile and Run:**

1. **Ensure the following files are in the same directory.**
2. **Compile the Project: Use the following command to compile the project.**

**gcc -o community\_manager main.c database.c member.c**

1. **Run the Program: Execute the compiled program:**

**./community\_manager**