Social Connecting App

Author:

Muhammad Ahmad (F2022266093)

University:

University of Management and Technology

Dept:

School of Science and Technology (SST)

Degree:

Bachelor of Computer Science

Email:

f2022266093@umt.edu.pk / mahmadmalik0301@gmail.com

Summary:

This code has various functionalities, including a friendly user interface that manages multiple parts of this code. This code has 5 out of 6 required functionalities. All this program lacks is an undo functionality.

Users are connected through a link list, but friends are connected through a graph. Each user has a unique username, which helps find the desired chat box and post. The tree implements a posting system in which roots represent a post. In contrast, these children represent a reply to the message. Gain Link list is used again to store various chat boxes, which are later fetched according to the user who called them. This code has 6 custom header files and one main cpp file. Each header file is used to implement 5 of the required functionalities, and one header file includes all the header files to make the main body cleaner. Some built-in header files are used to make the code better.

This program ensures the proper use of data structures for each implementation. The sequential implementation of the program will be explained later. This code was written in visual code with the latest

compiler, so some functionality might need to be fixed on dev C++.

Sequential Implementation:

The first header file is allHeaders. This header file has the header files used in that program, which include this header files

#include <iostream>

#include <list>

#include <vector>

#include <windows.h>

#include <ctime>

#include <cstdlib>

#include <regex>

#include "accounts.h"

#include "chat.h"

#include "groupChat.h."

#include "post.h"

#include "friendSystem.h."

<list> is used to access STL-linked list functions, and < vector> is used to access vector functions. <window.h>, <cstdlib> are supporting headerfiles. <ctime> is used to store the time of the post created and an account created. <regex\> is used to check the input validation of the email.

A function mainManager() manages everything to let users access different code modules like chatting. This ensures a cleaner interference for the user. The function uses a while loop, so users can't exit until they press a specific key.

In the main body, 4 link lists store posts, group chat, private chat, and user accounts. A single graph connects different users, representing the friendship between users.

In the main body, the user can create an account or log in to an existing one. After logging in with a password, the object in the user link list returns to an object in the main body by value for security purposes. NA d that object is passed to the mainManager() function and the other link lists so that the user can control various exceptions on the code.

Now, let's talk about each of the Modules.

User Account: "accounts. h" has a class named user account. These accounts have various data members like usernames and other things. And the functions to manipulate the objects. It also ensures that each user has a unique name and function to change a current detail. And to print all the user data available. It also has a functioning called login(), a friend to the class user account. This function asks the user to enter a name and check if the username exists in the link list, which stores user accounts. Yes, then it contains the password related to that username, and using a range-based for loop, it returns that object to the main body so that the user can do his stuff. It also has a few more functions that use a range-based loop to ensure that a username or email already exists.

Private Chat: Each chatbox is represented by an object. The class name is chatBox. It has a link list that stores all the messages between users and 2 usernames—one of the sender and the other of the receiver.

A function chat() pushes messages to a link list of strings with the username of the sender. chatMananging() is used to manage all the functionality related to chatting. It asks the user to enter the name of the

person they want to talk with and check if a chatbox exists for both users. If not, it creates a new chatBox object and pushes it into the link list that stores the chatbox.

GroupChat: This works the same as the Private chat, but each chatbox has a unique ID and a vector of people in the group chat.

Grouchat can be either public or private. Users can also browse the Public group chat and join them, but they can't join the group even though he has their ID.

Posting implementation: Class postingArea represents a tree, with each tree representing a post and its reply. Unlike a common BST, each node has a vector of replies pointing to a node in the answer.

Each root represents a post, and its child means the replies to it, and so on. A function (findPost) finds a post with its ID. It goes through a message's whole vector of replies nodes to find it and uses recursive calling for faster execution. Suppose the post is not found it returns NULL. It goes through the whole tree to locate the post with the same ID the user entered.

Add reply () function adds a reply to the PostID user entered. It uses the findPost function to locate the post or the answer and then add the reply to its vector of replies.

printTree() function is used to print the whole tree, which includes the main post and all of its replies in a readable manner. Again, in this code, recursive calling is used. And each recall depth is increased by 1 so that it creates a very readable format.

Afterward, a function named deletedPost() locates the post with findPost() and then deletes it and all its replies permanently.

After the class, a function postManager() is used. This uses a base loop to view all the posts stored in the link list. If the root (the main post) is deleted, it also gets removed from the link list that holds all the posts.

Friend system: A class friend graph is used to implement the friend system. It used unordered_map header files. The first part (represents a string, g, and the other part is the lin,k list, which is the list of the user represents all the vertex links.

Different functions are used so that users can make a proper manual. For sending requests, the other user has to accept the request so that they both get added to each other friend list. The program also has other functions to send, receive, and delete requests and delete.

Reference:

- cplusplus.com
- GeeksforGeeks - C++ Programming Language
- Codecademy - Learn C++
- cplusplus.org
- Stack Overflow