# CSE3310 Project

The Social Network

# Iteration II

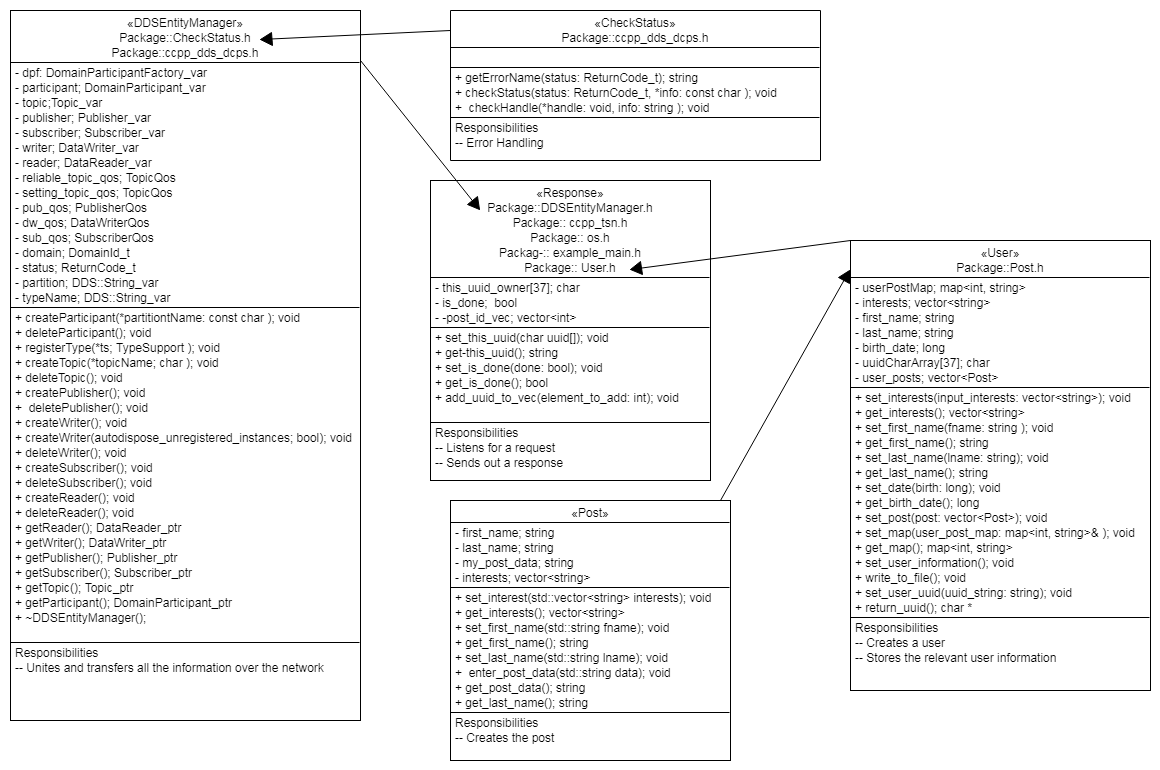
## Requirements

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| REQID | F/NF | Notes |  | Requirements |
| 1 | F | List Users | User Req | The list user option shall list all the users in an orderly fashion. |
| 2 | F | Show User | User Req | The show users option shall show the specific user and the posts they have made. |
| 3 | F | Edit | User Req | The edit option shall let the user edit the information entered, by default it is saved to a disk file and read in on program start. |
| 4 | F | Resync | User Req | The resync option shall forget all saved data and reset the sending of data. |
| 5 | F | Post | User Req | The post option should let user post their content. |
| 6 | F | Start | System Req | On start the program should print user’s personal information which will be published not more than once per 30 seconds. |
| 7 | F | Request | User/Sys Req | After receiving request, data will be sent out 1 per minute, which contains vector of UUID and serial numbers. |
| 8 | F | Post | System Req | The serial number of the post will be the unique identifier for the post. |
| 9 | F | Request | System Req | In order to make a request there must be nodes online with data you do not have. |
| 10 | F | User | System Req | The program shall ask for the user information only for the first time. |
| 11 | F | User | System Req | A user must be able to post, exit, and rejoin at any time they want. |
| 12 | F | Posts | System Req | Posts are to be sent only when requested to save network traffic. |
| 13 | NF | User ID | System Req | User id must be of length 37 generated by Boost stored in a char array |
| 14 | F | Posts | System Req | All post must be stored on local disk |
| 15 | F | Statistics | User Req | The Stats option shall print out how many nodes are known and how much content is available in this node listed as a percentage |
| 16 | F | Posts | System Req | Local disk file must be encrypted |
| 17 | F | Menu | System Req | The menu will display the user commands. |
| 18 | F | Serial Number | System Req | All serial numbers generated will start at 0 then generate new one every time post is made |
| 19 | F | UUID | System Req | UUID generated will be stored in a file. |
| 20 | NF | OS | System Req | This program should be compiled and run in Linux OS. |
| 21 | F | Request | System Req | An application will only satisfy requests for data it owns. |
| 22 | F | Function/Command | User Req | Program should support command to list all the users |
| 23 | F | Function/Command | User Req | Program should support command to reset the sending of data. (resync) |
| 24 | NF | Prog. Language | System req | TSN will be coded in C++11 Language |
| 25 | NF | IDL | System Req | Must use the IDL provided by the instructor. |
| 26 | F | User | System Req | Notification should be displayed if new post is made or user is online. |
| 27 | F | Post | System Req | The posts will be matched based on the interests the user selects. |
| 28 | F | User | System Req | The program should support Direct messaging. |

## Design (4 points)

Provide:

* A class diagram.



* Description of all important data structures and why they were selected (text and diagrams).

string – uuid\_string

Every time the user tries to create a post, the program will automatically generate a UUID of the post that will be stored in uuid\_string.

string first\_name

When ever a user tires to make a post certain information is required before they could make a post, their first name will be stored in first\_name.

string last\_name

In a similar fashion the last name of the user will be stored in last\_name.

And on the built in data structures side we have:

1.Arrays:

uuidCharArray

When ever a post is made it will get a UUID which will be stored in uuidCharArray array.

2.Vector:

Interests /input\_interests

While creating a post the program will ask for user interests. After the user enters the interests they will be stored in a vector Interests and input\_interests vector will contain same data as interest vector.

* Overall description of the program and how it works (text).

TITLE: The Social Network

This is a peer to peer social network command line program. This will be the first Iteration of the project. In this program a user will be prompted with a menu with options to List users, Show users, Edit, Resync, Post, Show statistics, Exit. The first option list user will list all the users of the program saved in a file. The second option Show user should show the information about the particular user based on the Name/UUID entered. Third option Edit will let user edit their information. Similarly Resync will forget all the data and reset the sending of the data. The other option show statistics will Prints out how many nodes are known and how much content is available in this node listed as a percentage. The last option will let the user Exit out of the network.

The second iteration of the project will add more functionality and easy user interface in the program. The two primary functionalities are user being able to send private message to another user on the network and getting notification if any user is online. This version of TSN is the modification on the previous version which is better and easy to use.

## Testing

Describe the testing process, in detail, used to verify the program.

Assessment:

* Coverage of requirements.
  + The way we tested our code was by:
    - Run make to compile the code.
    - Run main executable, this is the driver of the software.
    - Main has all the functions, like post resync, list, etc.
    - Open another terminal if you want to publish or subscribe to either a Response or a Request
    - In Request, we made sure our hello.tsn file was defined if not it will throw an error.
    - Then we verified the user input, our options were to enter a user number given from a list.
    - Once a user is chosen, a serial number is asked and we send out a request over the network.
    - The program will then enter a subscriber mode listening for a request
* Repeatability (can the grader repeat the tests?)
  + During the requirements process we found out that our Response Subscriber wasn’t working and have yet to fix it. A full report will be given on iteration 2. Also, inside our Request code we had a bug where our UUIDs were not printing correctly.