CS 344 Spring 2019 Semester Project

Due: April 25, 2019 (last day of class), by 2:00 pm

Presentation of project:

Monday, April 29, 2019, from 1700 to 1900 (5:00 pm - 7:00 pm)

Documents due by 2 pm on April 25, 2019:

Provide a folder with a printout of all of your code, including header files. Organize the folder neatly so that I can find things fast.

Have a cover page with CS 344, Spring 2019, Semester Project, and the names of each team member on it.

Following the cover page should be a table of contents with the name of each file and the team member (or members) who was (were) responsible for the code in that file.

At the top of each file should be the names of the team members responsible for the code in that file followed by a short description, and then followed by a list of the functions in the file if the file contains C code.

Description:

You have decided to write a client/server application to keep track of work projects. The project must have the following features:

1. When the client initiates a connection the server must provide an initial screen asking the user if he/she wants to sign up or sign in.
2. If the user wants to sign up, the server must collect an email address as a username, a password, and a full name. This information about user name, password (unencrypted for this exercise), and full name must be stored in an external text file on the server. The server must not allow any user to sign up more than once. If someone tries to sign up again, it must present them with a login screen. Authentication is based on username and password. The password must be at least 8 characters in length. The username is at most 30 characters long and must not contain spaces or start with a number. The full name should be no more than 50 characters in length.
3. The server must require users to sign in ONLY IF they have already signed up. That is, when a user attempts to sign in the server must check the user file to see if the user exists. If the user does not exist, the server must ask the user to sign up.
4. The server must allow users to create, edit, delete, save, and display project information by presenting the user with an appropriate menu of choices which also includes an option to quit the session. Editing is not fancy. It simply consists of asking the user for new values to replace existing ones.
5. The project information should be stored in a file and read into a list as soon as the user logs in. The entire list should be re-saved in the file, overwriting its contents, as soon as the user chooses to quit the session. The file name should be constructed from the

user's username, followed by the word proj as an extension. There should be no spaces in the filename. For example: bethelmd@erau.edu.proj

6. The server should define the project information as a list of structures where the data part of each node is itself a structure containing
    a. a project name of at most 100 characters,
    b. a project description of at most 1000 characters,
    c. a date that the project was created as an 8-character array that will hold a 2-character month, a 2-character day, and a four-character year,
    d. a due date that looks just like the date created,
    e. a byte that will keep track of the number of members on the project's team, and
    f. a linked list of names of project members, including the current user's name. Each name is no more than 50 characters in length.
7. Only the data part of the project information should be saved. That is, you should not save any pointers to data. A zero should follow the data for each project in the file in order to delimit one project's information from another.
8. The client must be designed to interact appropriately with the server.

Project specifications:

Your server must spawn processes to handle each communication session with a client.

The code should be modular and should consist of functions that do one logical thing and one thing only. This may mean that a function has to call other functions to help it.

Functions should be grouped into logical units where each logical unit exists in its own C file with its own header file.

A makefile should be used to compile and link the project files.

Notes:

You will need to carefully plan your application-level protocol so that you will know what data is to be transmitted between the client and the server. You will also need to plan your data structures and your transmission data format very well.