CSE278 Systems I

Programming Assignment#3: Client/Server Programming in C++ using BSD TCP Sockets

The objective of this assignment is to design and develop a novel client/server application in C++ using BSD TCP Sockets. Your client and server code must be well organized and documented, include a makefile, compile error-free, pass cpplint, produce expected results and present them nicely formatted output.

The Task

Design and develop a novel, easy to use, and useful client/server application of your choice. It must accomplish something useful and has some novel features. The server must provide at least 4 significant functionalities or services for clients to use – the services can be exposed using a header file. One of the functionalities could be authentication (the server authenticates the client before the client is able to use any of the services).

Note: taking one of the client/server programs presented in class (e.g. MathServer) and adding a few simple methods then submitting it as your solution is not an acceptable application for this assignment. If you do so, the maximum **total** mark for this kind of application is 15% for your effort. Please think of a novel idea.

Please *accept the assignment on Github (see link below) and note that I am not providing any starter code this time, but you are free to use any of the client/server applications we have developed in class as a starting point for client/server communication.*

Guidelines – Must…

- Your application consists of a server capable of handling multiple client requests, and a client.
- Use C++ BSD TCP Sockets. You are free to use the wrapper classes (from PracticalSocket) or the POCO library – your choice.
- Your server continues to handle client's requests until it is manually terminated.
- You provide a header file of the four services the server provides – so that the code is nicely organized.
- Provide a one-page Word or PDF document (readme) describing your client/server application. This can also be provided as a readme file on Github.
- Provide a fully functional makefile that can be used to compile the server and client, and run them. For example:
    - make server (will build the server executable)
    - make client (will build the client executable)
    - For the purpose of testing, you may hard code the server port number…so you can easily run them as:
        - make run-server

- make run-client (you may hard code server host: localhost or 127.0.0.1)

No need to submit your idea to the instructor for approval, just make sure it satisfies the above criteria…and I am available during office hours to discuss and brainstorm your idea. If there is a particular library you'd like to use in your application, please check with the instructor and get approval first.

Submission Guidelines

Submit your source code, readme file, support files, and makefile on Github **by 11:59pm on Friday, Nov 4**. Eight-hour grace period (so if you submit by 7:59am on Sat, Nov 5 there is no 10% penalty). Submissions after that time must be made via Canvas email (to Zhewen and CC Dr.Q) and are subject to 10% penalty/day. Here is the Github assignment link – ensure you select the correct name if needed before you proceed:
https://classroom.github.com/a/55Lt4J11

Grading Rubrics
The following rubrics will be used as a guideline:

Item (%)	**Excellent (full mark)**	**Good (75%)**	**Satisfactory (50%)**	**Unsatisfactory (25%)**	**Zero (zero)**
Novelty and functionality (50%)	Novel idea and fully functional, output is nicely formatted, no errors or warnings.	Re-inventing the wheel with no value-added, output is poorly formatted.	Basic idea and functionality.	Nothing special, error messages during runtime.	Doesn't compile or run.
Readme file (10%) & make file (10%)	Clear details about the objective of the client/server application, how it works, and how to run it.	Sufficient details are provided about the purpose of the application, how it works, and how to run it	Little details on the purpose of the application, how it works, or how to run it.	It is not clear what is the purpose of the application, how it works, or how to run it.	No readme file is provided, or blank Github readme file.
Source code and documentation (20%)	Follows coding standards. Fully documented, including purpose of program.	Readable source code, doesn't follow coding standards, not everything is documented.	Spaghetti code, basic minimal documentation.	Code provided is incomplete or does not make sense, documentation does not help understand the code.	No source code provided or the source code is not accessible, no documentation.
Linter - cpplint (10%)	No errors.	One error.	Two errors.	Three errors.	Four or more errors.