Policies and Guidelines

- Assignments deadline come with a ***72 hours grace period***. If you submit the assignment during the grace period, it will be accepted with a ***20% penalty***. Submissions after the grace period will be accepted with a ***50% penalty***. ***No submission will be accepted after the last day of classes on Thursday, May 4, 2023, at 11:59 pm***.
- Every project has required submission guidelines. Please read the submission requirements carefully. Any deviations from specifications on projects will result in point deductions or incomplete grades.
- ***Instructors will not 'fix' your code to see what works***. If your code does not run due to any type of errors, it is considered non-functioning.
- ***You cannot resubmit 'fixed' code after the deadline***, regardless of how small the error is.
- If you use the University resources for development and/or testing, then keep in mind that others in the University may need to use the same resources, so please use these resources wisely. You should also follow the rules and restrictions associated with using the University resources.
- Using any external/third-party library and/or framework to implement the project requirements is not permitted.

Academic Honesty Expectations and Violation Penalty

- ***The programming assignments are team assignments; each team cannot have more than two members***. Each team must do the vast majority of the work on its own. It is permissible to consult with other teams to ask general questions, help discover and fix specific bugs, and talk about high-level approaches in general terms. ***Giving or receiving answers or solution details from other teams is not permissible***.
- You may research online for additional resources; however, ***you may not use code explicitly written to solve the problem you have been given***. ***You may not have anyone else help you write the code or solve the problem***. You may use code snippets found online, providing that they are appropriately and clearly cited within your submitted code.
- ***If you use a distributed version control service such as GitHub to maintain your project, you should always set your project repository to private***. If you make your project repository public during the semester or even after the semester, it may cause a violation of the academic honesty policy if other students find and use your solution.

- If you are involved in plagiarism or cheating, you will receive a score of "0" for the involved project for the first offense. You will receive an "F" grade for the course and be reported to the university for any additional offense.
- Students are required to strictly follow the rules and guidelines laid out in the Watson School Student Academic Honesty Code at the following link: http://www.binghamton.edu/watson/about/honesty-policy.pdf
- Please also review Computer Science Faculty Letter to Students Regarding Academic Honesty at the following link: http://www.cs.binghamton.edu/~ghyan/courses/CSHonestyLetterToStudents.pdf
- Cheating and copying will **NOT** be tolerated. Anything submitted as a programming assignment must be the student's original work.
- We reserve the right to use plagiarism detection tools to detect plagiarism in the programming assignments.

Implementation

Part-1

In this lab, you will learn the basics of socket programming for TCP connections in Python: how to create a socket, bind it to a specific address and port, as well as send and receive a HTTP packet. You will also learn some basics of HTTP header format.

You will develop a web server that handles one HTTP request at a time. Your web server should accept and parse the HTTP request, get the requested file from the server's file system, create an HTTP response message consisting of the requested file preceded by header lines, and then send the response directly to the client. If the requested file is not present in the server, the server should send an HTTP "404 Not Found" message back to the client.

Part-2

In Part-1, the web server handles only one HTTP request at a time. Implement a multithreaded server that is capable of serving multiple requests simultaneously. Using threading, first create a main thread in which your modified server listens for clients at a fixed socket. When it receives a TCP connection request from a client, it will set up the TCP connection through another socket and service the client request in a separate thread. There will be a separate TCP connection in a separate thread for each request/response pair.

Running the Web Server

Put the provided HTML file “home.html” in the same directory that the web server source file is in. Run the web server program. Determine the IP address of the host that is running the server (e.g., 128.238.251.26), and the port number you used for your web server (e.g., 28000). From another host, open a browser and provide the corresponding URL. For example:

http://128.238.251.26:28000/home.html

“home.html” is the name of the file you placed in the server directory. Note also the use of the port number after the colon. You need to replace this port number with whatever port you have used in the server code. In the above example, we have used the port number “28000”. The browser should then display the contents of “home.html”. If you omit ":28000", the browser will assume port “80” and you will get the web page from the server only if your server is listening at port “80”. Then try to get a file that is not present at the server. You should get a “404 Not Found” message.

For Part-2, where you should implement a multithreaded version of the web server, try requesting a large file, such as a PDF file, from multiple browser tabs simultaneously to test your implementation. For that, you may need to modify your web server code such that it can handle requests for a PDF file.

Submission

- Complete the "README.md". You may write “n/a” where applicable (bugs, references, etc.).
- Please submit the following files for Project-1 on Brightspace using the same file name and extension as here:
    - webserver1.py (Part-1)
    - webserver2.py (Part-2)
    - Screenshots (PNG) of your client browser, verifying that you actually receive the contents of the HTML file from the server.
    - README.md
- You must submit your project before the deadline to be considered on time. Otherwise, it will be considered late even if your project was completely working before the deadline. ***Please note that on Brightspace, we maintain all your submissions but only grade your most recent submission.***

Grading Rubric

Total: 100 points

- Part 1: 30 points
    - Demonstration: 15 points
    - Implementation: 15 points
- Part 2: 55 points
    - Demonstration: 30 points
    - Implementation: 25 points
- Submission: 15 points
    - README.md: 10 points
    - All other requirements: 5 points
- If submission didn't pass the plagiarism test(s): -100 points.