1. Team details: Clearly state the names and netids of your team members (usually there are 2 of you).
    a. Arushi Pradhan (ap2132) and Shahrez Ahmad(ssa136)
2. Collaboration: Who did you collaborate with on this project? What resources and references did you consult? Please also specify on what aspect of the project you collaborated or consulted.
    a. For references, we consulted W3Schools for HTTP requests concepts, basic Python syntax and file handling, Google for Python's official documentation for understanding HTTP server socket setup, and GeeksforGeeks for examples of dictionary operations and HTTP parsing techniques.
3. Is there any portion of your code that does not work as required in the description above?Please explain.
    a. All portions of the code work as required by the project description.
4. Did you encounter any difficulties? If so, explain.
    a. In the optional part of the project, we were told to dynamically adjust the host name in the form's submit URL based on the header received in the HTTP request. Initially, when testing on the iLab servers, this did not seem necessary because the iLab servers have fixed domain names that remain consistent so hardcoding the hostname did not cause any issues in that environment. However, when we ran the server locally on our personal computer, we realized that the hostname and the localhost domain could mismatch, especially when the server was accessed from a different machine. This made it clear why dynamically adjusting the host was important: it ensures that the server can properly serve forms even when the client and server are running on different machines. It took some trial and error, as well as carefully reading the instructions again, to identify exactly where in the server code the host should be updated dynamically. Ultimately, we understood that the correct way to handle this is to read the host value from incoming requests and adjust the form action URL accordingly. Lastly, a client outside of the ilab domain cannot connect to a server.py running on the ilab domain for some reason. Perhaps Rutgers has it blocked for privacy reasons? It took time and tinkering to isolate this issue. Finally, when we tested our dynamically changed host (the host was changed as per the instructions), we actually observed more unexpected behavior than before with the localhost call. We were unable to resolve this in a timely manner without needing to perhaps use code not suggested to us by the instructors. So, using git, we reverted to our version of code that used the localhost call and used that as our final

version. This was from the optional part of the assignment and as stated above everything works on the ilab servers so it shouldn't be an issue.

5. Please discuss two observations or facts you learned about HTTP and cookies in the process of working on this project. Please be specific and technical in your response.
    a. Through this project, we learned two important technical facts about HTTP and cookies. First, we observed what HTTP being stateless looked like in actual implementation. Each HTTP request is completely independent, and servers do not remember any previous interactions by default. This makes cookies essential for managing user sessions, as they allow the server to recognize returning clients by associating a random token with their identity. Additionally, we also observed that browser behavior regarding cookies can vary slightly depending on settings and security policies. For example, when testing with Mozilla Firefox, we found that cookies were sometimes not presented back to the server after a successful login unless specific cookie attributes were correctly set. Browsers are becoming increasingly strict with cookie handling, especially for localhost environments, to protect user privacy. Therefore, while basic cookie functionality worked for this project, a bigger server would need to set these attributes explicitly to ensure consistent behavior across different browsers. Lastly, we gained experience and knowledge in the difference between HTTP GET and POST requests, as well as some knowledge and introduction to the rest of the HTTP methods.