For this question, I was a little surprised to see web development. I chose to use Flask to quickly create a simple web service using Python. In order to expose the service to the world, I found a little tool called serveo that creates a url and tunnels into the correct port on my localhost, so the url required for my example is currently:

- http://dimarco_paxos.serveo.net

The hint asks for when the order of messages I POST vs digests I GET matter. If the server is to go down, then the GET will not receive the digest since they are stored in the server's memory and volatile and will no longer exist. Additionally, if there are collisions between any two messages, then the GET will be the more recent message.

The performance question asks what the bottleneck is in my implementation as the service acquires more users, and how I could scale it. Being precise, my service is running on an Ubuntu Virtual machine with 4GB of allocated memory, so that is certainly the bottleneck since 4GB won't hold many hash digests. A more realistic web service would have a full database. If our service became extremely popular, then we would need to introduce more nodes to handle traffic. Each node could handle its own computation of the hash digest then write it to some external repository. When looking up a hash digest, it would again need to interact with the repository and most likely require a caching of the more popular message digests.

To run: call "flask run" from paxos1/
To create url: autossh -M 0 -o ServerAliveInterval=60 -R dimarco_paxos.serveo.net:80:localhost:5000 serveo.net