Line Server Problem

In this exercise, you will build and document a system that is capable of serving lines out of a file to network clients. You may do this in any language (although Java, Python are the most used at Circle). You may use any reference and any open-source software you can find to help you build this system, so long as you document your use. However, you should not actively collaborate with others.

Specification

Your system should act as a network server that serves individual lines of an immutable text file over the network to clients using the following simple REST API:

- GET /lines/<line index>
    - Returns an HTTP status of 200 and the text of the requested line or an HTTP 413 status if the requested line is beyond the end of the file.

Your server should support multiple simultaneous clients.
The system should perform well for small and large files.
The system should perform well as the number of GET requests per unit time increases.
You may pre-process the text file in any way that you wish so long as the server behaves correctly.
The text file will have the following properties:

- Each line is terminated with a newline ("\n").
- Any given line will fit into memory.
- The line is valid ASCII (e.g. not Unicode).

Design considerations:

- What techniques would you use to achieve maximum uptime? Scenarios to consider: code deployment, server failures, disk failures, etc…
- How extensible is your service / architecture? Can we add more endpoints for additional features?

What to submit

You should provide access to a public source code repository (or submit a zip file) that contains shell scripts to build and run your system, documentation for your system, and the source code for the system itself.

- **build.sh** - A script that can be invoked to build your system. A good usage would be to run unit tests and fail the build if they don't pass. You may invoke another tool such as Maven, Ant, etc. with this script. You may download and install any libraries or other programs you feel are necessary to help you build your system. (Please list out these dependencies in the README).

- **run.sh** - A script that takes a single command-line parameter which is the name of the file to serve. Ultimately, it should start the server you have built.
- **README** - A text file that answers the following questions:
    - How does your system work? (if not addressed in comments in source)
    - What do we need to build your system?
    - How will your system perform with a 1 GB file? a 10 GB file? a 100 GB file?
    - How will your system perform with 100 users? 10000 users? 1000000 users?
    - What documentation, websites, papers, etc did you consult in doing this assignment?
    - What third-party libraries or other tools does the system use? How did you choose each library or framework you used?
    - How long did you spend on this exercise? If you had unlimited more time to spend on this, how would you spend it and how would you prioritize each item?
    - If you were to critique your code, what would you have to say about it?

The remainder of the files in your tree should be the source-code for your system.