Steps to Run-

1 Clone the project from https://github.com/manishpaul26/pizza-server
2 Import into Eclipse/IntelliJ
3 Build the project using mvn clean install
4 Open the class "PizzaServer" and click on run
   a. You can provide the following run time arguments to the class (default in brackets)
      i. -DpoolSize=8 – The thread pool size
      ii. -DmaxPoolSize=20 - The Maximum thread pool size
      iii. -DqueueSize=200 - The queue size for the Thread pool
      iv. -DsocketTimeOut=4000 – Socket time out
      v. -DwriteToSameFile=false – Post request concurrency behaviour
      vi. -Dverbose=false – Print all logs- turn to true
      vii. –Dport = 4444 – Port number
5 Once you see the message "Starting Pizza server on port..", it is ready to use.
6 Hit the url localhost:<port>
   a. Default port is localhost:4444
7 This will show the default index.html file.

What all has been implemented-

1. GET request-
   a. /index.html -> homepage
   b. /upload.html -> Page to upload images

2. POST Request-
   a. /upload.html -> Upload an image and click on "Upload Image" button. This will upload the image to the "content" folder in the repository with the name of the uploaded file.
   b. If –DwriteToSameFile=true, then all post requests from this page will write to the same file (**new.jpg**).

3. To solve the problem mentioned in 2(b), all writes to the same file have been synchronized by maintaining a ConcurrentHashMap of files that are currently being written. If a file has been opened for writing, no other thread can open this file until the previous thread releases the file and updates this map (FileIO.java).

4. Thread Pool with configurable limits
   a. Use the run time arguments to configure the thread pool

5. A JMeter script "PizzaServerTest.jmx" is present in the root folder. Import this into JMeter and just click on the "Run" icon to simulate multi-threaded scenario with GET and POST Requests. This script does the following-
   a. Hits the homepage – 200 OK
   b. Hits the file "content/miffy.jpg"

    c. Hits 404 paths – 404 Not Found
    d. Posts the **new.jpg** file – Enable –DwriteToSameFile=true

6. HTTP Implementations-
    a. 200 OK
    b. 404 Not Found
    c. Keep-Alive
    d. Content-Disposition – to download files
    e. Content Types-
        i. Images
        ii. Html
        iii. Icons
        iv. MP4

7. **POST Requests-**
    a. A servlet framework has been created using custom Annotations.
    b. You can create your own servlet using "@ChocoServlet" annotation and a "path" parameter.
    c. During startup, all servlets with this annotation are registered and stored in a map with key as path and value as the Class itself
    d. At runtime, POST Requests check whether there is any servlet at that path. If yes, then use Reflections (org.reflections – dependency) to invoke the doPost method of that servlet.
    e. This can be seen with the "DownloadServlet" which is invoked on the click of the "download" button on the homepage.

8. **Performance boost-**
    a. The file miffy.jpg is cached in memory on startup to enable faster reads and avoid repeated I/O operations during the JMeter script execution. This has been done to improve the read performance when multiple threads are at work.