159.352 Computer Work I

PART B: Test Report

Mitchell Fenwick

19037806

Listed below are the test I have manually made in order to make sure my server is working as intended:

Requirement 1: Status Update

1. When visiting the update.html page, the page is rendered properly in Google Chrome, Mozilla Firefox, and Microsoft Edge.
2. When the user has typed out a sentence and clicks on the reset (clear) button, the typed content is removed
3. When the user presses the post button without entering a status, the page does not send any requests and prompts the user to enter something in the text field
4. When the user has entered some text and presses the Post button, the page is refreshed and the text in the text field is added to the status.json file with appropriate timestamp and likes array.

Requirement 2: Show Friends

1. When the User visits the friends.html page the content is generated dynamically and shows all information about friends (name, picture, recent status update, number of likes it has received)
2. The User request to get friends content is authenticated by creating a list of IP addresses and then checking to see if the request came from an friendly IP address.
3. If the User navigates to the friends.html page but one or more of the friends servers are not running then it will replace their status update with their name and some text to say they are offline.
4. If the user visits the friends.html page and one or more of their friends have no status updates, then they will simply be shown with a name and picture with no status.

Requirement 3: Like

1. When the User clicks on the like button for a friend’s status the page will refresh and add 1 to the likes total
2. When the User clicks on the like button, the post request will check if the user’s IP address in in the list of likes and if so then it will not increment the likes counter.
3. When the User has click on the like button, the like button will then be disabled so that the user will no longer be able to click on it. It will only become enabled again if the user’s IP address is removed from the friends likes list.

Requirement 4: System Design

1. Server.py is completely modular and the only difference between each user's file is the port number
2. Adding more users is easy, you just have to copy an existing user folder and change the Server.py to a new unique port number and change the friends.json file to include the other users. If you then want to view the new user on an old user's friend.html page, their friends.json file will need to be updated with the new users IP address. You can also replace the picture if you want, just make sure it is called profile.jpg
3. When a user tries to enter an address that does not exist (e.g. "localhost:8082/home.html") then they will be given an 404 Not Found error
4. At no point in the server is a request sent to another user from the web browser. All request are made via direct contact between Users.

Requirement 5: Documentation

1. I have documented the source code thoroughly and tried to make each piece easy to understand
2. I believe my usage report is clear enough to get the servers up and running.

Bonus Points: Caching

While I have not implemented the traditional method of caching with the Last-Modified and If-Modified-Since headers, I have implemented a simple caching system that stores data on the local disk rather than in the web browser. This is done by the requests-cache package.

Whenever a request is made using the requests package, the response will be cached for 60 seconds. If another request for that same response is made within 60 seconds than the data is retrieved from a local file rather then sending another request to a user.