**Application under test:**

Password hashing application used to hash a given password

**Requirements:**

1. When the app is launched it should wait for http connections
2. It should answer on the PORT specified in the PORT environment variable
3. Supports 3 endpoints:
   1. POST to /hash should accept a password and return a job identifier immediately. Wait for 5 sec and compute the password hash. Hash algorithm used is SHA512
   2. GET to /hash should accept a job identifier. It should return the base64 encoded password for the corresponding POST request
   3. GET to /stats should accept no data. Should return JSON data structure for total hash requests since the server started and average time of a hash request in milli sec.
4. Software should be able to process multiple connections simultaneously
5. Software should support a graceful shutdown request.
   1. Allow an in-flight password hashing to complete
   2. Reject any new requests
   3. Respond with a 200 and shutdown
   4. No additional requests should be allowed when shutdown is pending.

**Test case summary:**

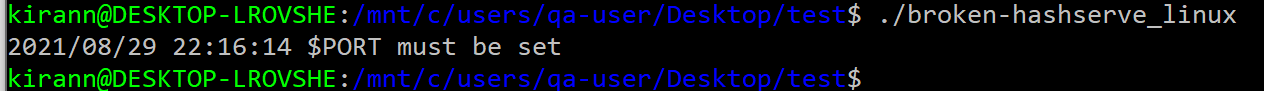
|  |  |  |  |
| --- | --- | --- | --- |
| **Sl.No.** | **Feature** | **Test case** | **Expected Result** |
| 1 | **Launching the app** | Without setting port | The app will not work (it crashes) |
|  |  | With setting Port | The app should wait for http requests |
|  |  |  |  |
| 2 | **POST to /hash** | Use a regular password like angrymonkey | Job identifier returned immediately. Wait for 5min to calculate password hash. SHA512 algorithm used for calculating hash |
|  |  | Send an empty request | Error-handling should be present to indicate that the requirement parameter is missing |
|  |  | Send a blank password | Error-handling should be present to indicate that the password is missing |
|  |  | Send a password with special characters and numbers | Job identifier returned immediately. Wait for 5min to calculate password hash. SHA512 algorithm used for calculating hash |
|  |  | Send an invalid json than the expected one like pwd:angrymonkey | An error should be thrown saying invalid/malformed input |
|  |  | Send a password with only numeric values like password:12345 | Job identifier returned immediately. Wait for 5min to calculate password hash. SHA512 algorithm used for calculating hash |
|  |  |  |  |
| 3 | **GET to /hash** | Use the job identifer from all the above requests (whereever applicable) | Verify the base64 encoded password (use online tools to computer the SHA512 hash for the password input and then the base64 encoder to get the base64 encoded password from the derived SHA512 hash. Verify that it matches with the one generated by the app |
|  |  | Use a job identifier that does not exist | Returns hash not found |
|  |  | Use a job identifier that is alphanumeric | Not exactly sure what the expected result should be as per requirements, but from the output it looks like only numeric values are allowed |
|  |  | Use a job identifier that is a spacial character | Not exactly sure what the expected result should be as per requirements, but from the output it looks like only numeric values are allowed |
|  |  | No job identfier sent | There should be an error saying parameter missing |
|  |  | Send a job identifier to find the max value allowed for job identifier | To check the max limit allowed for hashing passwords in a session (got out of range for 9999999999999999999 |
|  |  |  |  |
| 4 | **GET to /stats** | Get the stats when only 1 password hash request has been sent | The total requests should show as 1 and the average time should be the time taken for 1 request |
|  |  | Get the stats when 2 password hash requests have been sent | Verify total requetss is 2 and average time value is as expected in milli sec |
|  |  |  |  |
| 5 | **Shutdown** | Send the shutdown request | Verify that the app shutsdown and is no longer listening at the port |
|  |  | Send a POST request to /hash and issue the shutdown command at the same time | Verify that the job identifier is returned |
|  |  | Send a GET to /hash request and issue the shutdown command at the same time | Verify that the base64 encoded password is returned |
|  |  | Send a /stats request and issue the shutdown command at the same time | Verify that stats are returned first and then the app is shutdown successfully. |
|  |  | Send the shutdown request first and then the POST to /hash | Verify no job identifier is returned |
|  |  | Send the shutdown request first and then the GET to /hash | Verify no base64 encoded password is returned |
|  |  | Send the shutdown request first and then the stats request | Verify no stats are returned |
|  |  |  |  |
| 6 | **Processing multiple connections** | Run multiple POST to /hash requests from different windows | Verify that all return a job identifier |
|  |  | Run multiple GET to /hash requests from different windows | Verify that the corresponding base64 encoded password is returned corresponding to the job identifier |
|  |  | Run multiple /stat requests from different windows | Verify that the stats are returned |

**Test cases and execution results:**

**Launching the app**

1. Launch the app without setting the Port

I tried this in WSL and got a message telling me that the port must be set



But in windows it does not provide this message. Ideally we would not want an app to crash, instead provide error handling if the required input is not provided (in this case the PORT is not set)

1. Launch the app after setting the port

Verified that the app is waiting for http connections. I used port 44430 and when I ran the exe, I saw that it was running in the desired port

Text

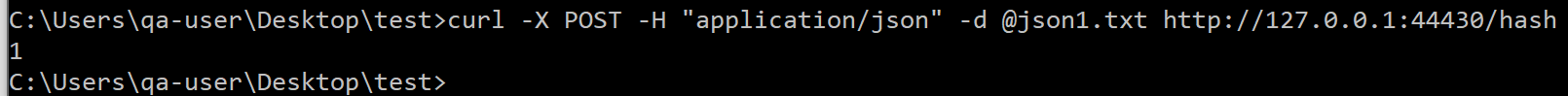
Description automatically generated

**Supported Endpoints**

**POST to /hash**

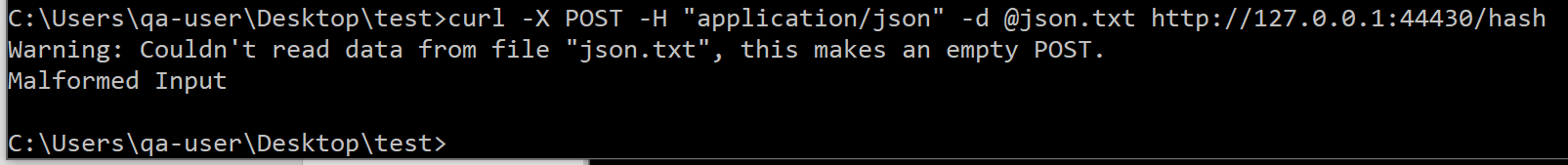
1. Hash the password where the password is {"password": "angrymonkey"}

curl -X POST -H "application/json" -d @json1.txt <http://127.0.0.1:44430/hash>



Observations: Although I do get the job identifier, I get it after 5seconds and not immediately as stated in the requirements. Probably the job identifier is returned after the hash is calculated.

1. Hash the password where the password file does not exist. Here we get ‘Malformed Input’. In this scenario, we could probably improve error to say what is missing, since ‘Malformed input’ is generic.



1. Hash the password where password is blank, json5.txt = {"password": ""}

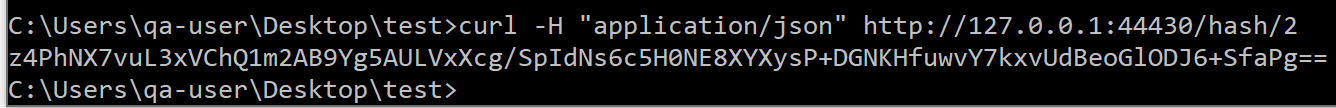
In this case, I checked the SHA512 algorithm which returns the following hash for an empty string:

cf83e1357eefb8bdf1542850d66d8007d620e4050b5715dc83f4a921d36ce9ce47d0d13c5d85f2b0ff8318d2877eec2f63b931bd47417a81a538327af927da3e

and the corresponding base 64 encoded value was

z4PhNX7vuL3xVChQ1m2AB9Yg5AULVxXcg/SpIdNs6c5H0NE8XYXysP+DGNKHfuwvY7kxvUdBeoGlODJ6+SfaPg==

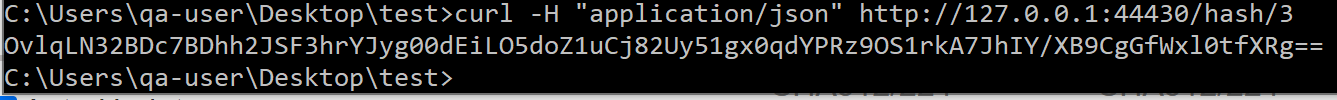
which matched with what was received from GET /hash



So the app is encoding a blank password, which should actually have been rejected. Before using the blank password, there should be error-handling done to check if the password that was passed in the request is valid and conforms to the standard password requirements.

1. Hash the password where special characters and numbers are used, json6.txt = {"password": "@ngrym0nkey"}

The app accepts special characters and numbers and hashes them appropriately as seen below.



Graphical user interface, text, application

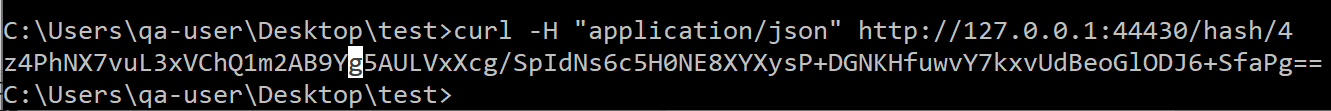
Description automatically generated

Graphical user interface, text, application, email

Description automatically generated

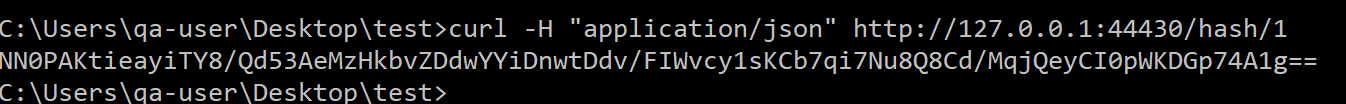
1. Provide the input using a different name for password in json input. json4.txt = {"pwd": "angrymonkey"}

Job identifier returned but when using GET /hash, I see this as hash value. Ideally this should have been same as what was got when we used {“password” : “angrymonkey”}



**GET to /hash**

1. Use the job identifier from the POST request above to get the hashed password



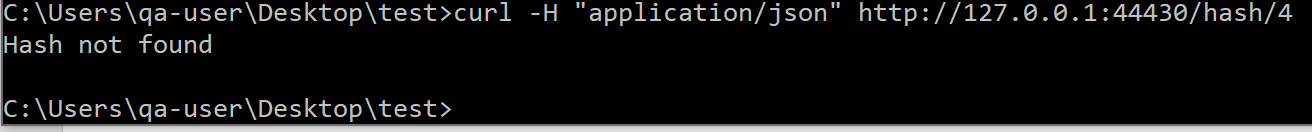
I checked online (<https://sha512.online/> was one of the multiple sites I checked just to make sure I get the same value in all) as to what the password hash using the SHA512 algorithm would turn out to be and it was this:

34dd0f00ab6279aca24d8f3f41de7701e3331e46ef6437706188839f0b4376ffc5216bdccb5b0a09beea8bb36ef10f0277f32a8d07b2088d2958a0c6a7be00d6

Used the above value as input in the base64 encoder(<https://emn178.github.io/online-tools/base64_encode.html> ) and it returned me this: NN0PAKtieayiTY8/Qd53AeMzHkbvZDdwYYiDnwtDdv/FIWvcy1sKCb7qi7Nu8Q8Cd/MqjQeyCI0pWKDGp74A1g==

This did match with the output of GET to /hash

1. Use a job identifier that does not exist.



Valid result when the job identifier does not exist.

1. Use a job identifier that is alpha numeric/special character

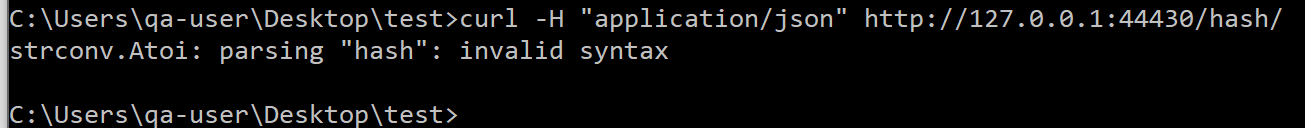
As per requirements, the job identifier does not need to conform to any standards, so I would expect alphanumeric values allowed too. But does not seem like it allows alpha-numeric or special characters since I get invalid syntax

Text

Description automatically generated

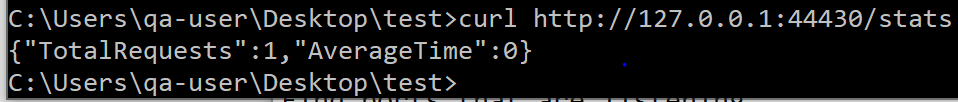
1. No job identifier passed in the request.

If we do not pass any job identifier, we get invalid syntax. An improvement on this could be a message like ‘No job identifier passed’



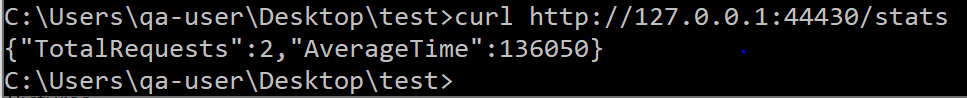
**GET to /stats**

1. Get the stats when only 1 request has been completed since server started.



This does not look right, since the average time should not have been 0 when 1 request is completed.

1. Get the stats when 2 requests have been completed since server started.



Here although we do not get a 0 anymore, the number 136050 (supposed to be in milli sec) does not look right (which is like 136 sec i.e. almost 2 min) not really sure how the time is calculated in this case.

**Bugs/Improvement Summary:**

1. Job identifier is not returned immediately but instead it seems to wait for 5 secs. Probably it is returned after the hash is calculated. I used

2. When sending no data, like

**curl -v -X POST -H "Content-Type: application/json" http://127.0.0.1:44430/hash**

it still spends 5 seconds and eventually returns 'Malformed input', instead if there was validation done earlier for the input it would be better.

3. Also when the above malformed input is sent and we try to get stats, we still get a response. In this case, not sure what should happen. If such requests need to be considered in the stats or not?

**curl http://127.0.0.1:44430/stats**

4. Blank password gets hashed without returning any error

5. If json string has incorrect name the password gets hashed as well. In #4 and 5 not exactly sure since the base64 encoded value for blank password as well as incorrect name for the password returns the same. We probably want to get the job identifier and hash passwords of only valid passwords

6. Not sure what is supposed to happen if an alphanumeric or special character is entered as job identifier. But there should be proper error-handling to indicate what is a valid job identifier. Probably a message indicate valid job identifiers for that session

7. If no job identifier is passed in GET to /hash, error handling should indicate missing parameter

8. Get stats shows average time as 0 when stats are retrieved after 1 request is complete. It should show the actual value, but it shows 0

9. The value of the average time by /stats, does not look right. Got 136050 (supposed to be in milli sec) does not look right (which is like 136 sec i.e. almost 2 min) not really sure how the time is calculated in this case.