README

1. Purpose
   This is a README document for JumpCloud QA Assignment. This document describes the documentation, Test execution summary and Automation Test suite created for password hashing application password hashing application.

2. Documentation Summary
   Following documents were created as part of Test Design process :
   Requirements Traceability Matrix : This document maps each requirement described in functional specifications to the High Level Test Scenarios and Test Cases. This document helps to ensures 100% Test coverage for functional as well as non-functional test cases.

   Test Case Template : Documentation of Test cases and their attributes including Ids, Titles, Pre-reqs, Test Steps, Test Data, Expected and Actual Test Results, Priority, Status, Tester Name and whether the test case is been added to Automation suite or not.

   Bug Report : Documentation of Defects in detail including the Priority/Severity.

3. Test Execution Summary

Total No. of Test Cases	No. of Test cases Executed	No. of Test cases passed	No. of Test cases failed
19	19	12	7

   (Test cases were executed manually as well as automated way, using terminal commands, Pycharm IDE and Postman)

4. Defect Summary

Total Defects	P1 - Showstopper	P2 - Major	P3 - Important
5	1	3	1

5. Test Automation Summary
   The Automation Test Suite is been created using Pytest framework.
   Below are the Test scripts and their brief description :
   a. Conftest.py – This file holds the global variables and common modules used across the other test scripts

b. Test_password_hash.py – This file includes various test modules corresponding to test cases documented in Test case template. I have tried to match the module names to the test cases.
c. Test_async_POST_requests.py – This program is to test multiple POST request sent asynchronously. This is not fully functional and getting into runtime erros. However, the goal was to test programmatically whether Application is able to handle asynchronous requests successfully. Hope it helps to understand the approach and provides psudo code.
d. Test_inflight_requests_during_shutdown.py – Attempted to test this functionality programmatically using multiprocessing /process and Queues
e. Test_error_post_shutdown_request.py - Attempted to test this functionality programmatically using multiprocessing /process and Queues
f. Report.html – Pytest allows to generate html report giving the summary of automation test execution. (pytest -v test_password_hash.py --html=report.html)

Please note that, I have marked test cases to be skipped (@pytest.mark.skip) which triggers the shutdown to avoid interreference to remaining test execution. However, those can be run individually.

Assumptions :
Since the expected and actual password hash is not matching, It is not clear that GET request should return the the base64 encoded password hash for the corresponding password or SHA512 password hash computed in the POST request. Hence assumption is made that application is expected to generate SHA512 password hash after the POST request. Then for GET request - It'll return base64 encoded password hash of the SHA512 password hash.

6. Decision – **No GO**
   Based on the Test passed % - 63.15 and due to the outstanding P1/P2 defects which are affecting the required functionality to not work as expected, it's a No GO decision to move this code to production.