Test Plan

A] **Test Project:**

The project is about testing JumpCloud's password hashing application in Golang. This application hashes the password provided by the user using SHA512 algorithm and then encodes it using base64 encoding. The application (broken-hashserve.tar) has binaries that could be unpacked and installed.

B] **Test Strategy**:

1) **Test Objective**: Test the password hashing application and various endpoints are tested based on the specifications provided in read me file. Identify the bugs and state clearly with screenshots.

2) **Scope**: In order to test the password hashing application, four given endpoints will be used for testing.
--POST call that sends password to the http://127.0.0.1:8088/hash, which in response provides the job identifier and computes the hash.
--GET call to /hash uses the job identifier generated in the POST call and returns the base64 encoded hash for the password for the job id
--GET to /stats provides the total hash requests sent when server started and average time of the hash request
--POST call to shut down the application

3) **Features tested**:

- Length and complexity of the password is tested and checked against the hash generated in various scenarios. [**Assumption** made as requirements were not mentioned for password rules]
- Multiple requests (Post call) are made sequentially and checked the time spent and average response time of a request.
- Multiple connections were established to the server and GET to/hash tested
- Check the hash is produced for valid passwords and not for null or empty strings or too lengthy passwords. [Assumption made as requirements were not mentioned for password rules]
- Checked that hash is not generated for invalid job identifier
- Verified stats—Total requests value along with average time of a hash request
- Verified that inflight hash requests are executed when shut down call is made

4) **Features not tested:**

- Authentication/Authorization is not tested fully as no requirements were provided.
- Database storage of hashes are not tested
- Load testing/stress testing isn't performed—except with max 50 requests POST call
- Due to immediate shutdown of app, allowing of additional password requests couldn't be tested

5)**Test tools/environment:** Windows machine is used with GIT Bash CL utility to execute curl commands for the four endpoints mentioned. Manual tests executed.

C] **Test deliverables:**

- Test Plan in a pdf format
- Test cases in excel spreadsheet with screenshots and status/messages/responses

- Read Me file

D] **Automation scripts steps** that can be used for testing:

1. POST the password to the http://127.0.0.1:8088/hash in json format
2. Parse the job identifier from the response of the POST call
3. Use that job identifier to make a GET call to/hash and capture the hash value
4. Post set number of hash requests to the server and then make a GET call for stats and capture the response for the TotalRequests and verify the count
5. Use either csv or dataprovider (if RESTAssured java library) to POST different password values in POST to /hash call –data driven testing
6. Parse the response of GET to hash call and verify the hash for the same passwords vs different password and also for the same job identifier.
7. Run the POST shutdown and verify the exe is closed and PORT is not listening

Automation scripts can be executed using Rest assured java libraries with eclipse and set as a maven project, using TestNg for data providers and test annotations to prioritize the run of tests in sequence needed.