Twistlock – Test Report

Test strategy:

1. Sanity
   1. Check that server is able to accept correct query for correct page and returns result of correct format
2. Functionality
   1. Check for correct response of subsequent queries to same page
   2. Check for correct response for queries to different pages
   3. Check that behavior is consistent with naming
   4. Data validation
   5. Queries from multiple clients
3. Negative
   1. Check correct consistent response for invalid input (bad api name, bad page, bad password, bad username, bad format value in page, bad port)
   2. Check that after correct query, query with wrong user fails
   3. Check that after correct query, query with wrong password fails
   4. Check that empty auth query fails with correct response code
   5. Check POST api is not allowed
4. Stress
   1. Perform many queries for same/subsequent/random page, see that server is able to respond
5. Performance
   1. Check execution time for each page and see if they comparable
   2. Run each page for large amount of time and see if there is performance degradation
6. Fault tolerance
   1. Check that restarting the server doesn’t cause different data to be received

Bugs:

Preconditions: have a working setup with twtask already running

1. **Randomly some player names are missing in the responses sent by the server**

Priority:

**High** – getting reliable record required many queries which slows down the execution

Steps to reproduce:

* + Query a page (ex. GET localhost:8000/players?page=1) until any Name in the result is empty or 10 seconds have passed

Expected result:

No empty Names were found and timeout was reached

Actual result:

Pretty quickly there will be an empty Name

1. **There is a missing ID number between each 2 pages**

Priority:

**Low** – user can fix it at relatively low cost but, this logic needs to be documented if

released as is

Steps to reproduce:

* Query a page and save the response
* Query page +1 and save the response

Expected result:

First ID in second response is larger by one than the last ID of first response

Actual result:

First ID in second response is larger by two than the last ID of first response

1. **Players may have more than one ID in some pages**

Priority:

**Critical** – no feasible workaround and it might break some logical manipulations with the results

Steps to reproduce:

* Do a reliable query a page(because of bug1) and save the response
* Create new dictionary where Name is the key and value is list of IDs for this name according to the response
* Verify length of list for each key

Expected result:

All lists have one member

Actual result:

Some list have several members

1. **Server accepts partial auth(only user or only password) as correct credentials**

Priority:

**Critical** – security severely impacted

Steps to reproduce:

* Send a query with valid page and syntax but only provide partially correct auth (only password is correct or only username is correct)

Expected result:

Status code 401 is returned

Actual result:

Status code 200 is returned and page data is sent

1. **Server gives inconsistent response when queried about pages 0 and 18+**

Priority:

**Low** – might require additional documentation if released as is

Steps to reproduce:

* Send a query for pages 0 and 18 which are out of bounds

Expected result:

Both queries should return similar result

Actual result:

0 returns status 418

18 returns status 200 and data of first page

1. **Query of pages 7 and 8 return several hundred times slower than other pages**

Priority:

**High** – majorly degrades the quality of the server

Steps to reproduce:

* Query page 7 and page 8 and measure the time until the result is obtained
* Query any other page ex. 10

Expected result:

Execution time should be similar

Actual result:

For 7 and 8 the result takes 2-4 seconds to obtain while other pages take several microseconds

1. **POST/PUT/DELETE request to given API returns unexpected status code when invoked**

Priority:

**Low** – this behavior is undocumented and might be unintended and hence prone to

abuse

Steps to reproduce:

* Send post request to the API (POST localhost:8000/players?page=1) with correct auth credentials

Expected result:

As per documentation only GET allowed hence API should return 405 Method not

allowed

Actual result:

200 is returned

1. **Server crashes when receives two queries to in parallel**

Priority:

**Critical** – server is unusable in multiuser environment

Steps to reproduce:

* Start two threads that perform query

Expected result:

Data is returned in both threads

Actual result:

Server crashes

1. **Returns 418 instead of 404 on 0 and non-numerical input of page**

Priority:

**Low** – errors should be consistent and have intuitive meaning

Steps to reproduce:

* Query server with non-numerical value for page

Expected result:

Status code 404 is returned

Actual result:

Status code 418 is returned

1. **When query page that is out of bounds (18+) first page is returned instead of 404**

Priority:

**Medium** – out of bounds index shouldn’t return a valid value

Steps to reproduce:

* Query server with page 18+

Expected result:

Status code 404 is returned

Actual result:

First page is returned

1. **Returns 418 instead of 404 on invalid request on port 8000**

Priority:

**Low** – errors should be consistent and have intuitive meaning

Steps to reproduce:

* Query server with incorrect or invalid api request

Expected result:

Status code 404 is returned

Actual result:

Status code 418 is returned

1. **Some players have multiple IDs across the database**

Priority:

**Critical** – no feasible workaround and it might break some logical manipulations with the results

Steps to reproduce:

* Do a reliable query of every page and save the response in single json
* Create new dictionary where Name is the key and value is amount this name appears in the json
* Copy only the keys with value larger than 1 into a list

Expected result:

List is empty

Actual result:

List isn’t empty