

Manual Test Plan for sp21-CS242-assignment 2.2

Table of Contents

- **Environment Setup**
- **Test - Scraping/Database Handling**
- **Test - API Requests**
- **Test - UI**

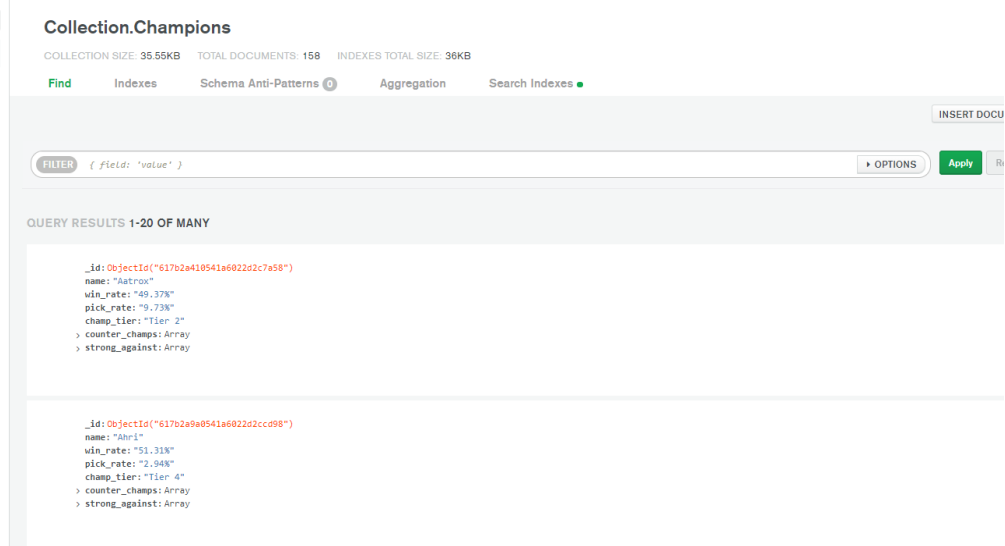
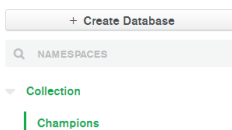
Environment Setup

- **Python 3.0**
- **unittest**
- **Javascript**
- **HTML**
- **D3 and svg**
- **Eclipse - 4.20.0**
- **Windows 10**

Test : Scraping/Database Handling

Test 1.1 Basic Program Run

```
1 from scraper import Scraper
2 import time
3 from database import get_key
4 from database import database_handler
5
6 """
7 Implements the scraping of data of all the champions from op.gg
8 each champions data is based off their current meta role
9 e.g aatrox meta role is top lane so we scrape data based of his top lane stats
10 """
11 def main():
12     s = Scraper()
13     arr = s.scrape_champion_links()
14     c_counter = 0
15     N = len(arr)
16     while(c_counter < N):
17         retArr = s.scrape_champion_page(arr[c_counter])
18         if retArr is None:
19             continue
20         c_counter+=1;
21         time.sleep(10)
22         database_handler(retArr)
23
24 if __name__ == "__main__":
25     main()
```



@Test 1.1

- Users can run the python module 'scraper_exe' to execute the Scraper class functionality and push all champions listed on the website op.gg to the MongoDB database.

Test - API Requests

Test 1.2 - Get Request

http://127.0.0.1:5000/champion?name=Aatrox Save

GET ▼ http://127.0.0.1:5000/champion?name=Aatrox Send

Params ● Authorization Headers (7) Body Pre-request Script Tests Settings Co

Query Params

	KEY	VALUE	DESCRIPTION	...	Bi
<input checked="" type="checkbox"/>	name	Aatrox			
	Key	Value	Description		

Body Cookies Headers (4) Test Results 🌐 Status: 200 OK Time: 4.68 s Size: 294 B Save Respo

Pretty Raw Preview Visualize JSON ▼

```
1 {
2   "counter_champs": [
3     "Zac",
4     "Malphite",
5     "Kled"
6   ],
7   "name": "Aatrox",
8   "pick_rate": "9.73%",
9   "strong_against": [
10    "Ryze",
11    "Sylas",
12    "Renekton"
13  ],
14   "win_rate": "49.37%"
15 }
16
17 }
```

@Test 1.2

- Users can use the GET API Request by first entering the command 'set FLASK_APP=src/api' and then flask run to start the local server.
- Specify the Request to be GET and then enter a champion's name to start the request. Upon a valid name being given, the json object should contain all the fields that are in the MongoDB database.

Test 1.3 - Put Request

The screenshot shows a REST client interface with the following details:

- URL:** `http://127.0.0.1:5000/champion?name=Aatrox`
- Method:** PUT
- Body (JSON):**

```
1 { "name": "Aatrox",
2   "pick_rate": "5.7%",
3   "win_rate": "79%",
4   "champ_tier": "Tier 0",
5   "counter_champs": ["Alistar", "Akali", "Zed"],
6   "strong_against": ["Yasuo", "Yone", "Riven"] }
```
- Response:** Status: 200 OK, Time: 4.58 s, Size: 183 B
- Response Body (Pretty):**

```
1 updated champion entry: Aatrox
```

@Test 1.3

- Users can use the PUT API Request by first entering the command 'set FLASK_APP=src/api' and then flask run to start the local server.
- Specify the Request to be PUT and then enter a champion's name to start the request. Upon a valid name being given along with a valid json object that updates any or all fields, the server will return a message depending on if the request was successful.

Test 1.4 - Post Request

The screenshot displays a REST client interface with the following components:

- URL Bar:** `http://127.0.0.1:5000/champion`
- Method:** `POST`
- Body:** A JSON object representing a champion's stats:

```
{
  "name": "Lex",
  "pick_rate": "5.7%",
  "win_rate": "79%",
  "champ_tier": "Tier 0",
  "counter_champs": ["Alistar", "Akali", "Zed"],
  "strong_against": ["Yasuo", "Yone", "Riven"]
}
```
- Response:** The client shows a `200 OK` status with a JSON response:

```
{
  "result": {
    "champ_tier": "Tier 0",
    "counter_champs": [
      "Alistar",
      "Akali",
      "Zed"
    ],
    "name": "Lex",
    "pick_rate": "5.7%",
    "strong_against": [
      "Yasuo",
      "Yone",
      "Riven"
    ],
    "win_rate": "79%"
  }
}
```

@Test 1.4

- Users can use the POST API Request by first entering the command `'set FLASK_APP=src/api'` and then `flask run` to start the local server.
- Specify the Request to be DELETE and then enter a champion's name to start the request. Upon a valid name being given, the

Test 1.4 - Delete Request

http://127.0.0.1:5000/champion?name=Lex

DELETE http://127.0.0.1:5000/champion?name=Lex

Params Authorization Headers (7) Body Pre-request Script Tests Settings Cookies

Query Params

	KEY	VALUE	DESCRIPTION	...	Bulk Edit
<input checked="" type="checkbox"/>	name	Lex			
	Key	Value	Description		

Body Cookies Headers (4) Test Results

Status: 200 OK Time: 4.51 s Size: 183 B Save Response

Pretty Raw Preview Visualize HTML

```
1 deleted champion with name Lex
```

@Test 1.4

- Users can use the DELETE API Request by first entering the command 'set FLASK_APP=src/api' and then flask run to start the local server.
- Specify the Request to be DELETE and then enter a champion's name to start the request. Upon a valid name being given, the MongoDB database will delete the specified entry by name and return a message depending on if the request was successful.

Test - UI

Test 1.5 - Static UI

← → ↻ ⓘ 127.0.0.1:5000

📱 Apps 📖 Assignments - CS 2... 🔍 Schedule Appointm... 📺 (45) A quick tour of...

Hello, Welcome to our App

[Top Champions by Pick Rate](#) | [Top Champions by Win Rate](#)

Input below

Champion Name:

Win rate:

Pick rate:

Tier:

Counter Champs:

Strong Champs:

Results below

@Test 1.5

- **Static UI home page that is yet to be connected to the backend query parser that has been implemented.**