Databases and Web Programming: Project 1

Individual project
Due: 30 August at 12:00 pm
Version 1.0

Preamble: The Great StackOverflow Scavenger Hunt

You may skip this section.

Title: “The Great StackOverflow Scavenger Hunt”

Once upon a time, in the mystical land of Codeberg, there lived a quirky developer named Alice. Alice was known for her wild ideas and unconventional solutions. She had a pet rubber duck named Quackers, who doubled as her debugging companion.

One sunny morning, as Alice sipped her coffee (extra strong, just like her opinions on semicolons), she received an urgent message from the Grand Council of Developers. The message was written in Comic Sans, which could only mean one thing: a dire situation.

The council explained that the entire internet had been swallowed by a giant black hole (probably caused by a rogue AI trying to optimize its gradient descent). Websites vanished faster than a junior developer’s confidence during a live demo. Panic ensued. Cats and dogs started collaborating on pull requests. Chaos reigned supreme.

But there was hope! The council revealed a prophecy: “Only by scraping StackOverflow can we restore the lost knowledge and save our digital civilization.” Alice’s heart raced. She was the chosen one—the StackOverflow Scavenger!

With her trusty laptop and a USB-powered lightsaber, Alice embarked on her quest. Here’s how it unfolded:

1. **The Quest for the Golden Snippet:** Alice entered the StackOverflow Forest, where trees whispered Python secrets. She encountered a wise old owl who said, “To find the Golden Snippet, you must answer three

riddles: ‘What’s the difference between == and ===?’ ‘How do you pronounce “GIF”?’ and ‘Why does JavaScript exist?’” Alice pondered, then replied, “The Golden Snippet is a one-liner that solves any problem. It’s like finding a unicorn riding a rainbow.”

2. **The Treacherous Tag Wars:** In the Tag Wars Dungeon, Alice faced tag trolls who argued about whether a question belonged in “python-2.7” or “python-3.x.” She shouted, “Enough! I’ll create a new tag: ‘python-42’—the answer to life, the universe, and debugging.”
3. **The Scroll of Infinite Scrolling:** Deep within the StackOverflow Caves, Alice found the Scroll of Infinite Scrolling. It contained a JavaScript spell to load more answers. But beware! If she scrolled too far, she’d encounter the dreaded “closed as duplicate” vortex. Alice chanted, “Scroll, scroll, little div, show me the wisdom of the hive mind!” And behold, the answers flowed like a waterfall of curly braces.
4. **The Legendary Reputation Points:** At the peak of Mount Upvote, Alice met a guru named Karma. He said, “Earn reputation points by answering questions. But beware vanity metrics—they’re like glitter: shiny but useless.” Alice replied, “I seek wisdom, not badges!” And she gained +42 enlightenment.
5. **The Final Boss: The Captcha Sphinx.** In the heart of StackOverflow Castle, Alice faced the Captcha Sphinx. It challenged her with obscure programming languages: “Write a Hello World in LOLCODE, using only emojis.” Alice cracked her knuckles, typed furiously, and summoned a Hello World that made the Sphinx weep. It granted her access to the API endpoints.

And so, Alice returned to Codeberg, API endpoints in hand. The internet reassembled itself, and kittens resumed their daily dose of cat memes. The Grand Council thanked her, and Quackers quacked in approval.

From that day on, developers whispered tales of Alice, the StackOverflow Scavenger. And if you listen closely, you might hear her mantra echoing through the IDEs: “When in doubt, Ctrl+C, Ctrl+V from StackOverflow!”

This preamble was partly generated by AI.

Specification

You must create a REST API that recreates the functionality of a set of endpoints described in the StackExchange API specification. Use Python and Flask to build the API. The data returned by your API should be scraped from the StackOverflow website using BeautifulSoup.

Your API must implement:

- all 4 built in filters described by the StackExchange API specification.
- the `min`, `max`, `fromdate`, `todate`, and `sort` parameters as well as any other parameters available for each endpoint.
- paging as described by the StackExchange API specification.

Your API does not need to implement:

- returning object fields that require authentication.
- any parameters that are not available for an endpoint (including `min`, `max`, `fromdate`, `todate`, and `sort`).
- the `quota_max` and `quota_remaining` fields.

You do not need to scrape StackExchange websites not part of StackOverflow. The order of the JSON fields returned is not assessed - but all objects must be nested correctly.

StackExchange's endpoints include an API version number as part of the path. Your endpoints must not include a version number as part of the path. For example, if the documentation describes an endpoint /foo then the /foo path must be available directly after the root.

If an invalid resource is requested from your API or a request is made using the wrong HTTP method, the response must use an appropriate HTTP response error code (such as `400 Bad Request` when it applies).

Before starting, explore the StackOverflow website and try to find tricky inputs such as this one. Additionally, you should review the API documentation and try out the API.

The Flask API is to be created in a file named `stackoverflow_scraper.py` and expose itself on the port specified by the `STACKOVERFLOW_API_PORT` environment variable. This environment variable should not be set by your code. Your API will rely

on web scraping functions to be defined in a file named `stackoverflow_scraper.py`. All Python dependencies (as output by `pip freeze`) should be specified in `requirements.txt`. Projects which fail to adhere to these instructions will not be marked. The following BASH script can be used to verify adherence.

```
#!/usr/bin/env bash
SCRIPT="stackoverflow_scraper.py"
TIMEOUT=30
PORT=23467
export STACKOVERFLOW_API_PORT=$PORT
python -m venv .venv
source .venv/bin/activate
pip install -r requirements.txt
python $SCRIPT &
while ! $(curl -s localhost:$PORT > /dev/null) && [ $TIMEOUT -gt 0 ] do
sleep 1; ((TIMEOUT--)) ; done
pkill -f $SCRIPT
deactivate
if [ $TIMEOUT -gt 0 ]; then echo "Verified"; else echo "Failed"; fi
```

You may assume your program will have stable internet access and do not need to handle network exceptions or incorrect user input.

Each time your API returns invalid JSON, you will receive a penalty. Use an appropriate JSON library and test your program thoroughly to ensure this does not happen. Each time your API returns an inappropriate HTTP response code or throws an exception you will receive a penalty.

Since you are attempting to clone the StackExchange API, if there are any inconsistencies between the API specification and the API you should mimic the behavior of the API as implemented by StackExchange. If you find API behavior is unpredictable then check Teams and SUNLearn announcements or similar reports. If it has not been addressed, please contact me via Teams or email at 24718076@sun.ac.za. In your message include as much information as possible such as the URL, the endpoint, field and any observations you have about when inconsistencies occur.

Endpoints

`/collectives`

Returns a list of Collective objects. Collectives are smaller communities on StackExchange websites.

The parameters `fromdate` and `todate` do not apply to collectives. The `sort` parameter has the possible values `asc` and `desc`. These must work (as implemented by the API) regardless of the documentation.

/questions

Returns a list of Question objects.

/questions/{ids}

Returns a list of Question objects identified by ids.

/answers/{ids}

Returns a list of Answer objects identified by ids.

Example:


```
{
 "items": [
   {
     "owner": {
       "account_id": 32490916,
       "reputation": 33,
       "user_id": 25242974,
       "user_type": "registered",
       "profile_image": "https://i.sstatic.net/oTQjUQWA.jpg?s=256",
       "display_name": "Ammighorbani",
       "link": "https://stackoverflow.com/users/25242974/ammighorbani"
     },
     "is_accepted": false,
     "score": -2,
     "last_activity_date": 1718575160,
     "creation_date": 1718575160,
     "answer_id": 78630353,
     "question_id": 419163,
     "content_license": "CC BY-SA 4.0",
     "body": "<p>Hi this code will check if that file this code is running
was main file and this code wasn't imported in another file it will do your
conditions after this code for you↙p>\n<p>Have a great day.↙p>\n"
```

```
    }
  ],
  "has_more": false,
  "quota_max": 300,
  "quota_remaining": 227
}
```

`/questions/{ids}/answers`

Returns a list of Answer objects.

Resources

- Beautiful Soup: Build a Web Scraper With Python
- Encoding URLs in Python
- The Hitchhiker's Guide to Python!

Marking rubric

Specification	Marks	Section total
Get collectives	6	6
Get questions		25
Fields (including filters)	15	
Query parameters	10	
Get questions by ID		25
Fields (including filters)	15	
Query parameters	10	
Get answers by ID		20
Fields (including filters)	12	
Query parameters	8	
Get answers by question ID		20
Fields (including filters)	12	
Query parameters	8	

Specification	Marks	Section total
Appropriate HTTP response codes	4	4
Total		100

Patch notes

29 July v1.0 - Project released