Open-Source Report

Proof of knowing your stuff in CSE312

Guidelines

Provided below is a template you must use to write your reports for your project.

Here are some things to note when working on your report, specifically about the **General Information & Licensing** section for each technology.

- **Code Repository**: Please link the code and not the documentation. If you'd like to refer to the documentation in the **Magic** section, you're more than welcome to, but we need to see the code you're referring to as well.
- **License Type**: Three letter acronym is fine.
- **License Description**: No need for the entire license here, just what separates it from the rest.
- **License Restrictions**: What can you *not* do as a result of using this technology in your project? Some licenses prevent you from using the project for commercial use, for example.

Also, feel free to extend the cell of any section if you feel you need more room.

If there's anything we can clarify, please don't hesitate to reach out! You can reach us using the methods outlined on the course website or see us during our office hours.

[Flask]

General Information & Licensing

Code Repository	https://github.com/pallets/flask
License Type	BSD-3-Clause license
License Description	It is an open-source software license used by the Flask project and many other open-source projects. The license allows for free use, modification, and distribution of the software, as long as certain conditions are met.
License Restrictions	1. Redistributions of the source code must retain the copyright notice, the list of conditions, and the disclaimer. 2. Redistributions in binary form must reproduce the copyright notice, the list of conditions, and the disclaimer in the documentation and/or other materials provided with the distribution. 3. The name of the copyright holder and the names of its contributors cannot be used to endorse or promote products derived from this software without specific prior written permission.

Magic

Brief description:

The line of code **token = request.cookies.get("auth_token")** in your Flask application is used to retrieve a cookie named "auth_token" from the HTTP request. This operation is performed by Flask's request object, which is an instance of the **Request** class from the Werkzeug library.
While this example focuses on cookies, the parsing and handling of other HTTP headers in Flask applications follows a similar process. The **Request** object provides dictionary-like access to all the headers in the HTTP request, allowing for easy retrieval of header values.

The whole chain from framework code to HW code:

1. We call **token = request.cookies.get("auth_token")** at: https://github.com/fhuang566/CSE-312/blob/37c7873f40566230956b02c78422c705ee50470d/code/app.py - L28
    - The request object is an instance of Flask's Request class, and it is automatically created by Flask at the beginning of each request. The cookies attribute is a werkzeug.datastructures.Headers object that provides dictionary-like access to the request's headers.

2. Flask's **Request** class calls Werkzeug's **Request** class: https://github.com/pallets/werkzeug/blob/76b049dd45fd072fb62a54bccc0e8d513b03f4d8/src/werkzeug/sansio/request.py - L359
    - Werkzeug's **Request** class provides the **cookies** attribute, which is a dictionary-like object that provides access to the request's cookies. The cookies are parsed from the **Cookie** header in the request.

3. Werkzeug's **Request** class uses **werkzeug.http.parse_cookie**: https://github.com/pallets/werkzeug/blob/76b049dd45fd072fb62a54bccc0e8d513b03f4d8/src/werkzeug/sansio/http.py - L124
    - The **parse_cookie** function parses the **Cookie** header into a dictionary-like object.

4. In the process of combining the "Cookie" headers, the **getlist** method of the **Headers** class from the **werkzeug.datastructures** module is called:https://github.com/pallets/werkzeug/blob/76b049dd45fd072fb62a54bccc0e8d513b03f4d8/src/werkzeug/sansio/request.py - L362
    - The **getlist** method retrieves all the values associated with a given key from the headers. In this case, it retrieves all the "Cookie" headers from the HTTP request.

Stack trace:

1. **Application code:**
   - token = request.cookies.get("auth_token")

2. **Flask's Request object:**
   - The request object is an instance of Flask's Request class. It's automatically created by Flask at the beginning of each request.

3. **Werkzeug's Request class:**
   - The **request** object is an instance of Flask's **Request** class. It's automatically created by Flask at the beginning of each request.

4. **Parsing the Cookie header:**
   - The **cookies** attribute is a **cached_property** which means the parsing is done once and then cached for subsequent access. The parsing happens in the **cookies** method where it joins all the Cookie headers and then calls **werkzeug.http.parse_cookie** to parse the Cookie header into a dictionary-like object.

5. **Werkzeug's parse_cookie function:**
   - **parse_cookie** function is used to parse the HTTP Cookie header. It takes the raw Cookie header as an argument and returns a dictionary-like object representing the cookies.