Open-Source Technology Use Report

Proof of knowing your stuff in CSE312

Flask(__name_)	Anthony
Request parsing and routing	Imon
Http response encoding	Anson

Guidelines

Provided below is a template you must use to write your report for each of the technologies you use in 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'd like 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.
- **Who worked with this?**: It's not necessary for the entire team to work with every technology used, but we'd like to know who worked with what.

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	Copyright 2010 Pallets
License Description	THIS SOFTWARE IS PROVIDED BY THE COPYRIGHT HOLDERS AND CONTRIBUTORS "AS IS" AND ANY EXPRESS OR IMPLIED WARRANTIES, INCLUDING, BUT NOT LIMITED TO, THE IMPLIED WARRANTIES OF MERCHANTABILITY AND FITNESS FOR A PARTICULAR PURPOSE ARE DISCLAIMED. IN NO EVENT SHALL THE COPYRIGHT HOLDER OR CONTRIBUTORS BE LIABLE FOR ANY DIRECT, INDIRECT, INCIDENTAL, SPECIAL, EXEMPLARY, OR CONSEQUENTIAL DAMAGES (INCLUDING, BUT NOT LIMITED TO, PROCUREMENT OF SUBSTITUTE GOODS OR SERVICES; LOSS OF USE, DATA, OR PROFITS; OR BUSINESS INTERRUPTION) HOWEVER CAUSED AND ON ANY THEORY OF LIABILITY, WHETHER IN CONTRACT, STRICT LIABILITY, OR TORT (INCLUDING NEGLIGENCE OR OTHERWISE) ARISING IN ANY WAY OUT OF THE USE OF THIS SOFTWARE, EVEN IF ADVISED OF THE POSSIBILITY OF SUCH DAMAGE.
License Restrictions	Redistribution and use in source and binary forms, with or without modification, are permitted provided that the following conditions are met: ● Redistributions of source code must retain the above copyright notice, this list of conditions and the following disclaimer. ● Redistributions in binary form must reproduce the above copyright notice, this list of conditions and the following disclaimer in the documentation and/or other materials provided with the distribution. ● Neither the name of the copyright holder nor the names of its contributors may be used to endorse or promote products derived from this software without specific prior written permission.
Who worked with this?	Anson Dai

Use as many of the sections below as needed, or create more, to explain every function, method, class, or object type you used from this library/framework.

Flask()

Purpose

What does this tech do for you in your project?
- Per the docs, this call instantiates the Flask class which implements a Web Server Gateway Interface (WSGI) application to run web applications in Python.

Where specifically is this tech used in your project? Give us some details like file location and line number, if applicable. If too cumbersome, a general description of where it's used for a given purpose is fine as well.
- This is used in UB-CSE-312-Project-Jesse/server.py/ Line 3. It serves as the starting point for our Flask implementation of the project.

Magic ⋆★˚｡⋆°✦°☽°⌒➘｡°★彡✭ ༄

Dispel the magic of this technology. Replace this text with some that answers the following questions for the above tech:

How does this technology do what it does for you in the **Purpose** section of this report? Please explain this in detail, starting from after the TCP socket is created. Remember, to be allowed to use a technology in your project, you must be able to know how it works.
- The call creates an instance of the WSGI application which uses a TCP connection.

Where is the specific code that does what you use the tech for? You ***must*** provide a link to the specific file in the repository for your tech with a line number or number range.
- https://github.com/pallets/flask/blob/7620cb70dbcbf71bca651e6f2eef3cbb05999272/src/flask/app.py#L97

*This section may grow beyond the page for many features.

Last Updated: Friday, November 12

return …

Purpose

Replace this text with some that answers the following questions for the above tech:

- What does this tech do for you in your project?
    - The tech creates the http response to send to the client
- Where specifically is this tech used in your project? Give us some details like file location and line number, if applicable. If too cumbersome, a general description of where it's used for a given purpose is fine as well.
    - To conclude every function, a return is used to send http data to the client for the web page.

Magic ⋆★˚｡⋆˚⋆☽˚⁀➷｡˚★彡⋆ ༄

Dispel the magic of this technology. Replace this text with some that answers the following questions for the above tech:

- How does this technology do what it does for you in the **Purpose** section of this report? Please explain this in detail, starting from after the TCP socket is created. Remember, to be allowed to use a technology in your project, you must be able to know how it works.
    - This section of the Werkzeug library sends all the necessary HTTP headers outside of the data.
- Where is the specific code that does what you use the tech for? You ***must*** provide a link to the specific file in the repository for your tech with a line number or number range.
    - https://github.com/pallets/werkzeug/blob/main/src/werkzeug/serving.py (lines 154-422)

*This section may grow beyond the page for many features.

@app.route()

Purpose

Replace this text with some that answers the following questions for the above tech:

- A Python decorator that runs the wrapped function on a given URL route and HTTP request method type(s).
- Where specifically is this tech used in your project? Give us some details like file location and line number, if applicable. If too cumbersome, a general description of where it's used for a given purpose is fine as well.
    - The tech is used in server.py on multiple places. In essence the function defined immediately after @app.route() is executed whenever the HTTP request's path and type matches the parameters passed to @app.route()

Magic ⋆★˚｡⋆ ˚｡⋆☽˚⁀➷｡˚★彡⋆ ༄

Dispel the magic of this technology. Replace this text with some that answers the following questions for the above tech:

- How does this technology do what it does for you in the **Purpose** section of this report? Please explain this in detail, starting from after the TCP socket is created. Remember, to be allowed to use a technology in your project, you must be able to know how it works.
    - The function wrapped by a @app.route() is called a "view function". With the parameters passed to @app.route(), Flask creates an instance of a Rule class and then adds that to a Map class. The view function is stored in a dictionary with the URL endpoint as the key. Whenever a request is received, the dispatch_request() function matches the request url and returns the return value of the view function associated with the request url.
- Where is the specific code that does what you use the tech for? You ***must*** provide a link to the specific file in the repository for your tech with a line number or number range.
    - @app.route() calls Scaffold.route() (scaffold.py Line 413) which in turn calls Scaffold.add_url_rule() (scaffold.py line 445) which is defined in the Flask (app.py line 1038) class which extends the Scaffold class.
    - In add_url_rule() an instance of Rule class is created (app.py line 1083) with the rule (URL path) and endpoint (name of the view_function unless endpoint explicitly defined).
    - The Rule instance is then added to a Map class (app.py line 1086).
    - The view_function is added to a dictionary with the endpoint as the key (app.py line 1094).
    - The dispatch_request (app.py line 1480) matches the req URL and returns the return value of the view_function if a view_function is defined for that endpoint (app.py line 1502).

*This section may grow beyond the page for many features.

[insert method/function/class/object name here]

Purpose

Replace this text with some that answers the following questions for the above tech:

- What does this tech do for you in your project?
- Where specifically is this tech used in your project? Give us some details like file location and line number, if applicable. If too cumbersome, a general description of where it's used for a given purpose is fine as well.

Magic

Dispel the magic of this technology. Replace this text with some that answers the following questions for the above tech:

- How does this technology do what it does for you in the **Purpose** section of this report? Please explain this in detail, starting from after the TCP socket is created. Remember, to be allowed to use a technology in your project, you must be able to know how it works.
- Where is the specific code that does what you use the tech for? You ***must*** provide a link to the specific file in the repository for your tech with a line number or number range.
  - If there is more than one step in the chain of calls *(hint: there will be)*, you must provide links for the entire chain of calls from your code, to the library code that actually accomplishes the task for you.
  - Example: If you use an object of type HttpRequest in your code which contains the headers of the request, you must show exactly how that object parsed the original headers from the TCP socket. This will often involve tracing through multiple libraries and you must show the entire trace through all these libraries with links to all the involved code.

*This section may grow beyond the page for many features.

Last Updated: Friday, November 12