TCP 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/python]

General Information & Licensing

Code Repository	https://github.com/pallets/flask
License Type	**BSD-3-Clause License**
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	• 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.

Magic

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

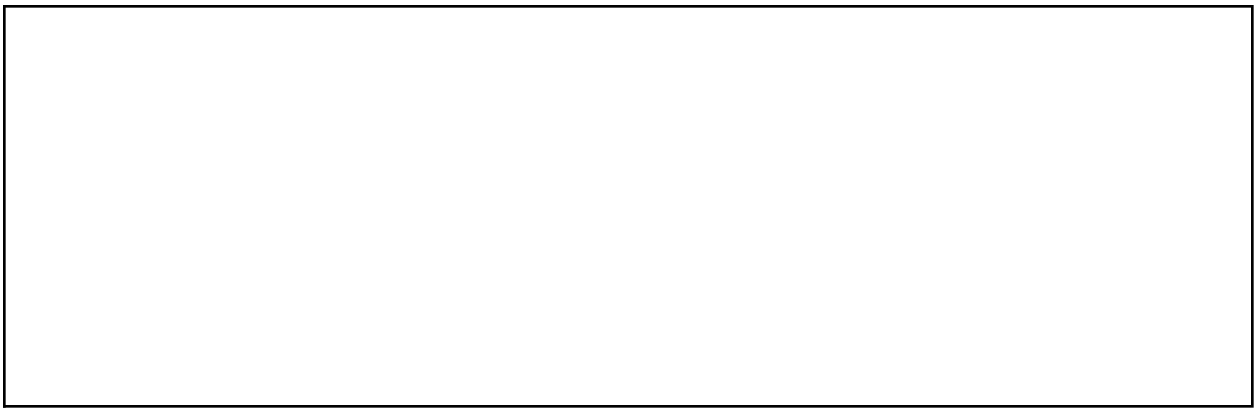
- How does this technology do what it does? Please explain this in detail, starting from after the TCP socket is created
- Where is the specific code that does what you use the tech for? You ***must*** provide a link to the specific file in the repository for your tech with a line number or number range.
    - 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 will likely grow beyond the page

IMPORTS:
from flask import Flask, and request(https://github.com/pallets/flask/blob/ed5b240417414cbd0322efa95c91f759928ba154/src/flask/wrappers.py#L15) The request object is a:class:`~werkzeug.wrappers.Request` subclass and provides all of the attributes Werkzeug defines plus a few Flaskspecific ones. In render_template, is taken from the src/flask/templating.py branch,(https://github.com/pallets/flask/blob/d0bf462866289ad8bfe29b6e4e1e0f531003ab34/src/flask/templating.py#L139). This is used in our code to render a template by name with the given context. param template_name_or_list: The name of the template to render. If a list is given, the first name to exist will be rendered.: param context: The variables to make available in the template. and redirect is also imported (https://github.com/pallets/flask/blob/ed5b240417414cbd0322efa95c91f759928ba154/src/flask/app.py#L2041) this creates a redirect response object. And finally url_for is also imported (https://github.com/pallets/flask/blob/ed5b240417414cbd0322efa95c91f759928ba154/src/flask/app.py#L1917) lines 1917 -1926, Functions can be decorated with meth:`url_defaults` to modify keyword arguments before the URL is built. then from wekzeug.security import generate_password_hash (https://github.com/pallets/werkzeug/blob/76b049dd45fd072fb62a54bccc0e8d513b03f4d8/src/werkzeug/security.py#L83) from line 83. Which Securely hash a password for storage. A password can be compared to a stored hash using:func:`check_password_hash. This works by taking advantage of gen_salt line 18 which generates a random string of SALT_CHARS with a specified length and _hash_interval at line 26. Next to be imported is check_password_hash(https://github.com/pallets/werkzeug/blob/76b049dd45fd072fb62a54bccc0e8d513b03f4d8/src/werkzeug/security.py#L120)its functionality Securely check that the given stored password hash, previously generated using:func:`generate_password_hash`, matches the given

password.

CODE:
After the imports the code starts off with creating the variable app with Flask (parameters: __name__ and template_folder='template') taken from (https://github.com/pallets/flask/blob/ed5b240417414cbd0322efa95c91f759928ba154/src/flask/app.py#LL110C10-L110C10) line 110 The flask object implements a WSGI application and acts as the central object. It is passed the name of the module or package of the application. Once it is created it will act as a central registry for the view functions, the URL rules, template configuration, and much more by taking the parameters of Scaffold on line 55 that has common behavior shared between class`~flask.Flask`and:class:`~flask.blueprints.Blueprint. Then @app.route is called which takes '/' as an input. Directing to the path of /. it does this in def route (https://github.com/pallets/flask/blob/ed5b240417414cbd0322efa95c91f759928ba154/src/flask/scaffold.py#L423) line 423 -452 this function Decorates a view function to register it with the given URL rule and options. Calls:meth:`add_url_rule`, which has more details about the implementation. Route contains a function within it named decorator on line 447, this function takes the input of type T_route and returns type T_route, where on line 42 it is defined, then the function takes the input 'self' which in this case would be '/' and calls function add_url_rule on line 455 which Register a rule for routing incoming requests and building URLs. The :meth:`route` decorator is a shortcut to call this with the ``view_func`` argument. These are equivalent: to the regular call and to the call with app.add_url_rule. The endpoint name for the route defaults to the name of the view function if the ``endpoint`` parameter isn't passed. An error will be raised if a function has already been registered for the endpoint.The ``methods`` parameter defaults to ``["GET"]``. ``HEAD`` is always added automatically, and ``OPTIONS`` is added automatically by default. When this is called in the decorator it is called with the rule, endpoint, f, and **options from the route call input. Then inside the def index in our code, a call to render_template is made, with the input of ' login.html'. With this, the template is rendered from the templates folder, and render_template acts as previously stated. (https://github.com/pallets/flask/blob/d0bf462866289ad8bfe29b6e4e1e0f531003ab34/src/flask/templating.py#L139).
The next @app.route follows the same format as the last one called except it has an extra input for the rule, being methods=[get and post] which allows for the usage of post and get requests in the app route for /register.same goes for /login ,/echo,/accpunt,/feed,/bid,and /post_item.etc



Websocket 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-Sock]

General Information & Licensing

Code Repository	https://github.com/miguelgrinberg/flask-sock/tree/main
License Type	**MIT License (MIT)**
License Description	• Permission is hereby granted, free of charge, to any person obtaining a copy of • this software and associated documentation files (the "Software"), to deal in • the Software without restriction, including without limitation the rights to

	● use, copy, modify, merge, publish, distribute, sublicense, and/or sell copies of ● the Software, and to permit persons to whom the Software is furnished to do so, ● subject to the following conditions: ●
License Restrictions	The above copyright notice and this permission notice shall be included in all copies or substantial portions of the Software. THE SOFTWARE IS PROVIDED "AS IS", WITHOUT WARRANTY OF ANY KIND, EXPRESS OR IMPLIED, INCLUDING BUT NOT LIMITED TO THE WARRANTIES OF MERCHANTABILITY, FITNESS FOR A PARTICULAR PURPOSE AND NONINFRINGEMENT. IN NO EVENT SHALL THE AUTHORS OR COPYRIGHT HOLDERS BE LIABLE FOR ANY CLAIM, DAMAGES OR OTHER LIABILITY, WHETHER IN AN ACTION OF CONTRACT, TORT OR OTHERWISE, ARISING FROM, OUT OF OR IN CONNECTION WITH THE SOFTWARE OR THE USE OR OTHER DEALINGS IN THE SOFTWARE.

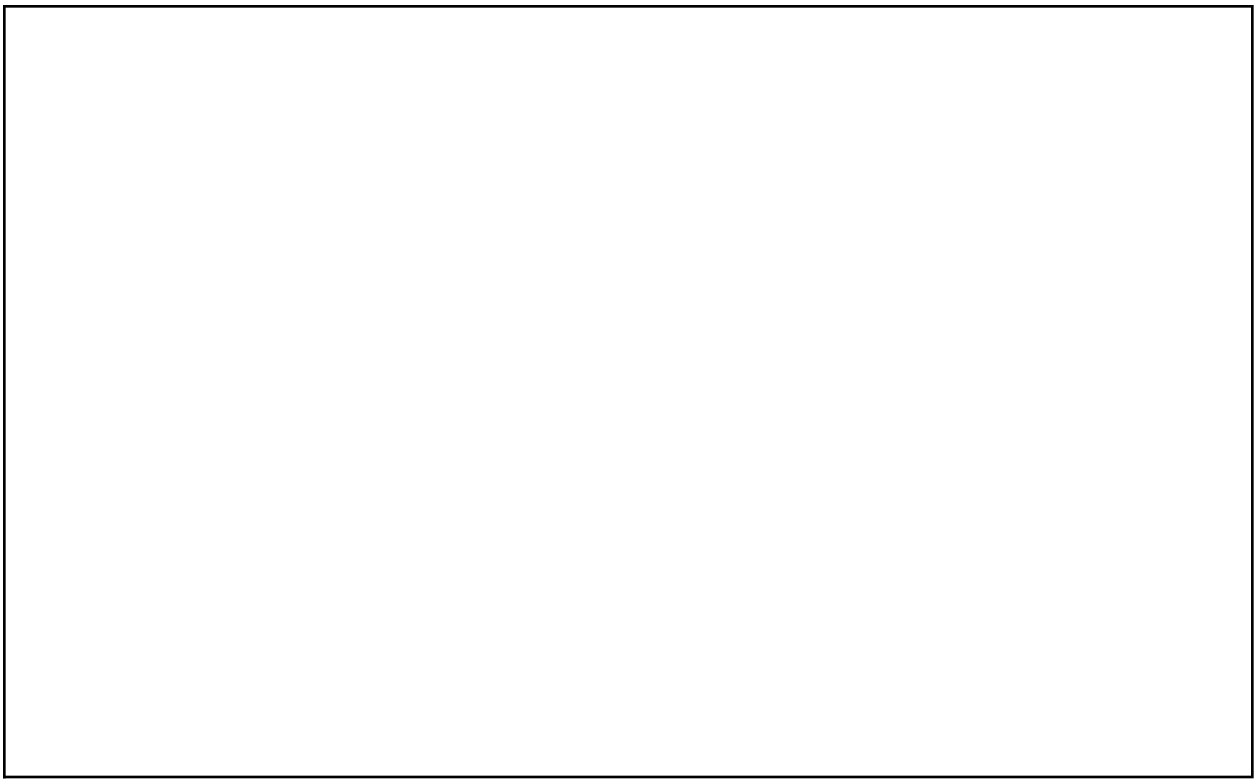
Magic ★★ ° ˚ ☽ ˚ ⌒ ˚ ★ ≡ ★ ≫

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? Please explain this in detail, starting from after the TCP socket is created
- 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 will likely grow beyond the page

import :
From flask_sock import Sock , this line in our code allows us access to the library. Class Sock is imported from(https://github.com/miguelgrinberg/flask-sock/blob/cb728fc1bededa29c49141744313a13f505cab53/src/flask_sock/__init__.py#L6) starting at line 6 working in tandem with the Flask application. This provides us with the library to use throughout the program. Within the echo route in our code on line 81 def echo takes input of object sock. And within that funcion is a while true loop where data is set to sock.receive where .recive receive is used to receive data sent by the remote socket. The library accomplishes this by taking advantage of the def route function in (https://github.com/miguelgrinberg/flask-sock/blob/cb728fc1bededa29c49141744313a13f505cab53/src/flask_sock/__init__.py#L31) this function will be invoked when a WebSocket client establishes a connection, with a WebSocket connection object passed as an argument.this function also contains function decorator on line 56, and def websocet_route on line 58 (https://github.com/miguelgrinberg/flask-sock/blob/cb728fc1bededa29c49141744313a13f505cab53/src/flask_sock/__init__.py#L58). If the route has variable components, the ``ws`` argument needs to be included before them. It also takes a blueprint(referenced in tcp report) on which route to register. If this isnt passed the route is attached directly to the Flask application instance.
After this in our code line 84 sock.send(data) follows a similar process to the previous line but instead of receiving its sending the received data. This creates the websocket connection.



HEADER PARSING Open-Source Report

Proof of knowing your stuff in CSE312

[Flask / Python]

General Information & Licensing

Code Repository	https://github.com/pallets/flask
License Type	**BSD-3-Clause License**
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	• 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.

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? Please explain this in detail, starting from after the TCP socket is created
- 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 will likely grow beyond the page

The code being talked about in this report is used to parse the headers. We used this to parse and retrieve the HTTP headers for a request. We used it to get the parts of the content we want, for example, username, password, image, etc…

We import:
From flask import Flask, request
In our code we use in our app.py file:

- “request.method” to get what type of request is being requested from the incoming HTTP request
    - Based on the certain type of request we do what is needed
        - “Request.method” == POST
        - “request.method” == GET
- We use “request.form.get” to get the names from the username, that would be within the headers of the request
- For example:
    - “request.form.get(“the header wanted”)
    - For example when we want to parse through the headers and get the username of the a user when they login or register we do:
        - “request.form.get(“username”)

Now to discuss how we traced through the repo to find what we can use to benefit us when it comes to header parsing.

https://github.com/pallets/flask/blob/d0bf462866289ad8bfe29b6e4e1e0f531003ab34/src/flask/app.py# The Request class is called. This class is created in the werkzeug.wrappers module. The Werkzeug is a comprehensive WSGI application library that is used to create the framework Flask.

https://github.com/pallets/flask/blob/ed5b240417414cbd0322efa95c91f759928ba154/src/flask/wrappers.py#L15
“The request object used by default in Flask. Remembers the matched endpoint and view arguments. It is what ends up as :class:`~flask.request`. If you want to replace the request object used you can subclass this and set:attr:`~flask.Flask.request_class` to your subclass. The request object is a :class:`~werkzeug.wrappers.Request` subclass and provides all of the attributes Werkzeug defines plus a few Flask specific ones.” (lines 16 - 23)
The Request class is a subclass of the BaseRequest class, which is also created in the werkzeug.wrappers module.

https://github.com/pallets/werkzeug/blob/2b2c4c3dd3cf7389e9f4aa06371b7332257c6289/src/werkzeug/wrappers/base_request.py#L29

The Base Request class defines most of the classes to use for different types of HTTP requests.
"Very basic request object. This does not implement advanced stuff like
    entity tag parsing or cache controls. The request object is created with
    the WSGI environment as first argument and will add itself to the WSGI
    environment as ``'werkzeug.request'`` unless it's created with
    `populate_request` set to False.” (lines 30 - 34)

The Request class is created from this class with more functionality.