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 License
License Description	• A BSD license are a family of free software licenses • A BSD license imposes minimal restrictions on the use and distribution of the software. • These do not have share-alike requirements like other licenses.
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

This library is a lightweight web application framework. This allows for web servers to pass requests to web application or other frameworks. Flask accomplishes its purpose through multiple different aspects. We will examine how Flask operates once the TCP socket is created.
Once the TCP socket is created, Flask receives incoming HTTP requests through the socket which is then processed by the Flask application and generates an HTTP response. We can break these general steps down into more specific tasks. First, an instance of the Flask application is created by calling the Flask class in python. Then an HTTP request is received by the socket, Flask then matches this request to the corresponding function that is written in the python code. This function often look like "*@app.route(/example-path)*". The function processes the HTTP request and once the desired actions are completed then an HTTP response is sent back to the client through the socket. In order to communicate with the web server, Flask uses a web server gateway interface(WSGI). When a request takes place then the web server passes it to the WSGI which then passes it to the Flask software. The response is then sent back through the WSGI which sends back to the client.
The specific code that we use this technology for is accepting and senfing a response to and from a web server which is either to or from a client. The code that sends the response can be found in the helpers.py file in the

repository(https://github.com/pallets/flask/blob/main/src/flask/helpers.py). This helper function returns a response object with all of the necessary headers (https://github.com/pallets/flask/blob/d0bf462866289ad8bfe29b6e4e1e0f531003ab34/src/flask/helpers.py#L132). This helper calls another class called current_app which is in the globals.py(https://github.com/pallets/flask/blob/d0bf462866289ad8bfe29b6e4e1e0f531003ab34/src/flask/globals.py#L48) file. This class spawns another class called Flask(https://github.com/pallets/flask/blob/57e926c7916ef5fd7d4abac965273ea185e497d2/src/flask/app.py#L92). This then calls a response method(https://github.com/pallets/flask/blob/d0bf462866289ad8bfe29b6e4e1e0f531003ab34/src/flask/app.py#L1719). These two methods is where the request is handled and the response headers are created. The response goes through a series of multiple method which eventually reaches the “finalize_request” function (https://github.com/pallets/flask/blob/d0bf462866289ad8bfe29b6e4e1e0f531003ab34/src/flask/app.py#L1489) . This Flask file imports mostly imports other files that are part of the repository but there are a few additional classes imported such as “os”, “logging”, and “sys” which are used to perform lower level operations.
In conclusion, when there is a request for the web server, the web server routes through all of the methods mentioned above and then returns a response to the request.

We use this technology in our project to listen for an HTTP request coming from the client using the line “@app.route('/')”. This example is listening for a request to the home path. We then respond to this request with a response by using the line “return render_template('index.html')”.