Open-Source Technology Use Report

Proof of knowing your stuff in CSE312

Instagram 2.0 CSE 312 Project

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	BSD
License Description	• It gives the permission for anyone else to use it for commercial use, modification, distribution, and private use. • The copyright holder or contributors are not liable for any damages
License Restrictions	• The name of the copyright holder nor the names of the contributors can be used to endorse or promote products derived from Flask without written permission. • Redistributions in binary form must reproduce the

	copyright notice ● Redistribution of source code needs the copyright notice
Who worked with this?	Tirth, Vedant, Aditya, Abi

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.

@app.route(“”)

Purpose

What does this tech do for you in your project? Flask allows us to route our requests to the appropriate files in our public directory. More specifically, we have the `@app.route(“/”)` to route to path “/”, and so on for all the other paths
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 @app.route() function is being used in our route.py file to route requests via their appropriate paths.

Magic

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.

App.route works off the class created by calling flask and when the .route function is called the path is added to the class and the function add_url_rule is called, which checks the input and whether or not it is properly formed as well as ensuring that the methods are formatted correctly such as making the method uppercase and storing a variable which then creates a 'url_rule_class' which is a function called from werkzeug.routing calling a class 'rule'. This class contains the methods that are provided like 'GET' or 'POST' and ensures that they are callable methods and that are needed, as well as the path and sets them to class variables and associates them to each other by having them in the same class. Overall the path is set by a class call in werkzeug routing storing the path, methods, and other possible variables that are applied in the function.

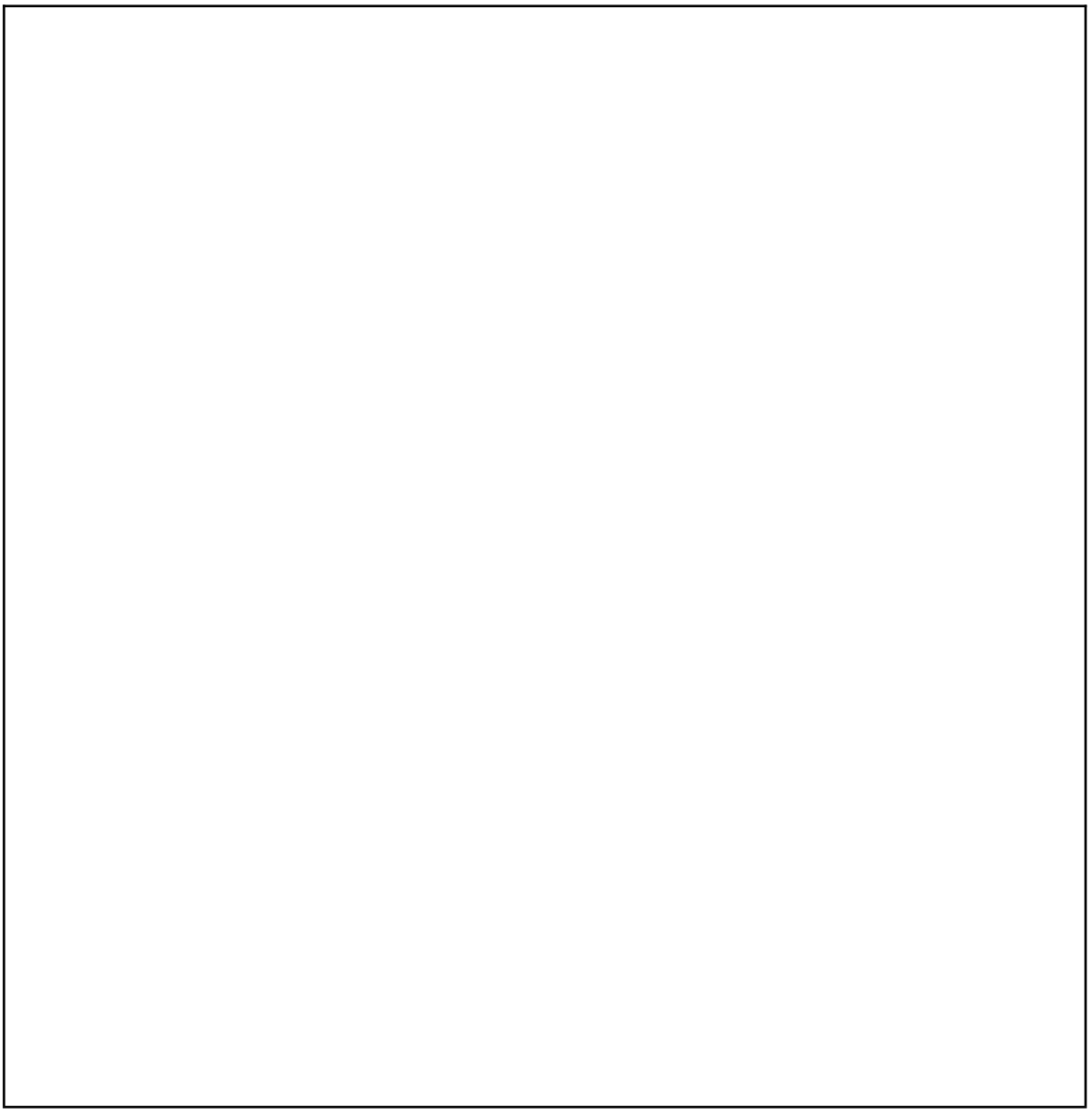
Where is the specific code that does what you use the tech for? You ***must*** provide a link to the specific file in the repository for your tech with a line number or number range. through all these libraries with links to all the involved code.

https://github.com/hahshtrit/solid-server/blob/master/routes.py

https://github.com/pallets/flask/blob/fac630379d31baaa1667894c2b5613304285a4bb/src/flask/app.py#L25
Lines 414~
https://github.com/pallets/flask/blob/fac630379d31baaa1667894c2b561330

https://github.com/pallets/werkzeug/blob/bb21bf90b0b121e3ed45b9950b823e4b43a81fd8/src/werkzeug/routing.py#L558
.

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



Flask.make_response() -> Flask.render_template()

Purpose

Used Extensively in Routes.py
Render_template(): Replaces the index.html pages' placeholder with the appropriate strings. This is used in routes.py, after each call to the path in order to render the page.

Render_template() also calls a function named make_response(), which is also used in our code when we need to set cookies as in the response as well.

Template inheritance allowed us to import templates from a base file called base.html,

which contained our nav-bar and pop-up messages. This mean significantly less clutter html files. Basically, the extends command tells the compiler to inject the html skeleton document. Each specified block {%%} is a container which can be filled with stuff in the child templates. The {% %} format is used in html files, and helps us use other html files and adds onto this.

Magic

1. Explain Purpose
    a. The function make_response() is called in routes.py, which is where we handle the different routes that the client's request.
    b. From the render_template function, the function is sent into a helper function that renders the html.
    c. Jinja template is used as well to render our template. It works in similarity to python code where it reads the file, and finds phrases to execute. It uses { } to differentiate its text.
    d. This is sent to the make_response function which takes the path as the input. The helper function is called through an object called current_app, which is the the initialized app file after the server is begun.
    e. Make_response parses the argument, which is a tuple after it has been processed by the current_app object. This object is checked for headers. If these headers are there, a func call is made to rv.headers.update(headers), which is where the html file specified by the route given in make_response().
    f. This is passed onto update, which replaces the keywords in the html with the given arguments in make_response() and allow us to render the template
    g. This is passed back to the render_template() function which
    h. This is passed back to the make_respnse function, which now returns our rendered template and returns it at the route function

https://www.geeksforgeeks.org/template-inheritance-in-flask/

Flask.set_cookie

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?
  This allows us to set_cookie to our headers when we send out the data back 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.
  We use this feature in our routes.py file where we set the cookie to the number of visits that have been to this page.

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.

1. Purpose

This function takes in the alteast 2 arguments that is key and value, and more cookies arguments that you might want to pass. It then adds these values to the headers in the response object. It calls a dump_cookie() function which creates a 'Set-Cookie' header without the 'set-cookie' prefix.' which then returns the headers in bytes.

2. Links
   https://github.com/pallets/werkzeug/blob/main/src/werkzeug/http.py
   line~ 1218 onwards
   https://github.com/pallets/werkzeug/blob/main/src/werkzeug/sansio/response.py
   line ~ 232 onwards as well as line 127
   https://github.com/hahshtrit/solid-server/blob/master/routes.py
   line ~ 51 onwards

Flask Forms management

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?

This tech allows us to receive the data that the clients submit, so that we can host it on our webpage. It collects the headers from the form and parses the relevant data, and creates a filestorage object that holds all our data.

- Where specifically is this tech used in your project?

This global variable is used in our route.py file, and used to receive forms that users upload to their account.

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.
  This technology takes in the request that the TCP sends into its request class, and then that class contains multiple methods for parsing forms and different type of data. The specific method that we use are form and files. The form method calls _load_form_data(), and this function parses all the data that is received from the client and stores all the relevant information. Then the form function returns that and it is a dictionary. For the files function, it is similar to form, but it instead creates a `ImmutableMultiDict[str, FileStorage]`, and that stores the name of the input you had on the form, and creates a filestorage class, which stores all relevant details of the files. The function files calls another function _load_form_data, which creates a FormDataParser object, which stores, the filename, filesize, file type and all relevant information related to the file. We then call file read, which essentials reads the data in the file.

- 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.

  https://github.com/hahshtrit/solid-server/blob/master/routes.py ~ https://github.com/hahshtrit/solid-server/blob/master/templates/signup.html ~ 6 onwards
  https://github.com/pallets/werkzeug/blob/main/src/werkzeug/datastructures.py

~848 onwards
https://github.com/pallets/flask/blob/main/src/flask/wrappers.py ~15
https://github.com/pallets/flask/blob/main/src/flask/globals.py line 55~

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

Flask.redirect()

Purpose

- This allows us to redirect the file, by sending a response that elicits a 302 response code, which is better than entirely rendering the template from scratch. Basically the same thing as we did in the redirects for HW 1

Magic

1. This function is called in some of the routes in routes.py, mostly to handle redirects after a form has been submitted, or just page refreshes in general.
2. The redirect function is in utils.py under werkzeug. This redirect function constructs the response with the appropriate headers, such as the location, code of 302 and an optional response object.
3. The response object is then constructed with the appropriate headers and the link, which is also html escaped before being send as a redirect request. Finally the header's location parameter is set as the location of the redirect, and this function returns the appropriate redirect request.