[Flask] (render_template)

General Information & Licensing

Code Repository	https://github.com/pallets/flask https://github.com/pallets/jinja
License Type	BSD 3-Clause
License Description	● Redistribution of source code and in binary form is allowed if the copyright notice is there ● 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.
License Restrictions	● Endorsements and promotions for products made with flask is not allowed unless prior permission is given
Who worked with this?	Tariq

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.

render_template

Purpose

What does this tech do for you in your project?	• This function renders an HTML page for a route specified in the decorator above. The HTML must be included in the templates/ folder. • The function also has a template engine that will provide a rendered form of the HTML to the user, with any variables, conditionals, or loops in the template replaced with the appropriate values or removed.
Where specifically is this tech used in your project?	Throughout flask_server.py (ex. Line 30)

Magic ⋆★˚⋆｡˚☽˚｡⁀➷˚｡★彡⋆༄

Documentation	• https://flask.palletsprojects.com/en/2.1.x/tutorial/templates/ • https://jinja.palletsprojects.com/en/3.1.x/templates/
How does this technology do what it does for you in the Purpose section of this report?	• The jinja2 template engine works similarly to the template engine shown in class. To start, there is the jinja2 Environment class, which defines the config variables. For my purpose I did not need to change any of these variables so I used the default configuration. This sets '{{' to be the beginning of variable, '}}' to be the end, and '{%' and '%}' for blocks. Within my html, I can use these brackets to specify variables I want to change. • This is done by calling render_template with the name of the HTML template along with named variables to be changed within the template. First, a jinja environment is created based on the app context. Jinja will then load the template and return a Template object which can then be rendered. (Template, context - dictionary containing the values to be replaced as well as other values needed for the template context processor, and app - the AppContext, containing various config variables for the server itself) are passed as parameters to _render. The .send functions do nothing as they are a part of the FakeSignal class. However, the return variable is a rendered string of the html template with placeholders replaced. This is done by calling the template's render function with the aforementioned context. Within render, placeholders are replaced using the environment's concat function. The template is iterated through and getattr is called for line. If there is a placeholder variable, here is where it is replaced

	appropriately. Once finished, this string with replaced placeholders is returned, eventually returning up the stack calls until it is served to the socket.
Where is the specific code that does what you use the tech for?	● flask - render_template() - templating.py 133 ● flask - update_template_context() - app.py 732 ● jinja - get_or_select_template() - environment.py 1057 ● jinja - get_template() - environment.py 966 ● jinja - Template - environment.py 1120 ● flask - _render() - templating.py 124 ● jinja - render() - environment.py 1259 ● jinja - getattr() - environment.py 480