Open-Source Technology Use Report

Proof of knowing your stuff in CSE312

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-socketio

General Information & Licensing

Code Repository	https://github.com/miguelgrinberg/flask-socketio
License Type	MIT
License Description	• It gives the permission for anyone else to use it without restriction where they have rights to use, copy, modify, merge, publish, distribute, sublicense, and/or sell copies of the • The copyright holder or contributors are not liable for any damages or claim
License Restrictions	• Redistribution of source code needs the copyright notice and license
Who worked with this?	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.

@socketio.on(“”, namespace=””)

Purpose

- What does this tech do for you in your project?
  This tech allows us to automatically read some websocket requests from the client based on the event name in the parameter. This even allows us to specify namespaces to split our websocket for specific purposes which helps with organization and clarity. In our project, we specifically use this to map certain server response functions to certain websocket events that the client requests. For example, we have ones for “draw”, “stop_drawing”, “connect_user”, and “direct_message” along with their respective namespaces.
- 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 specifically used in the routes.py where we have @socketio.on(“”, namespace=””) for specific events and their respective namespaces. For example on lines 195, 201, 207, 223 in routes.py.

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 just sets similar functionality as flask routes where the corresponding function will be mapped to this event call.
- 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.

https://github.com/hahshtrit/solid-server/blob/e705a3599735c638853742a1ae4b96b40b56763b/routes.py#L195

https://github.com/miguelgrinberg/Flask-SocketIO/blob/f7ca69af129e6575f82142f27fbf9054522e969d/src/flask_socketio/__init__.py#L258
https://github.com/miguelgrinberg/Flask-SocketIO/blob/f7ca69af129e6575f82142f27fbf9054522e969d/src/flask_socketio/__init__.py#L286
Then it enters the socketio package, where in the server.py it has defined on() functionality.

emit()

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 send some data to the client to some event that the client expects. This even allows us to specify what data is sent over the websocket connection and gives us the ability to broadcast to all open sockets. It even gives us an option to select which websockets to send it to if we want to be more specific. In our project, we use this to respond to some websocket event that the requests. We send our drawing data and direct message data using this function. We specified to broadcast to all for drawing data and specified a room (which specifies a socket) for direct

messages.

- 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 routes.py. We use it on line 198, 204, 240.

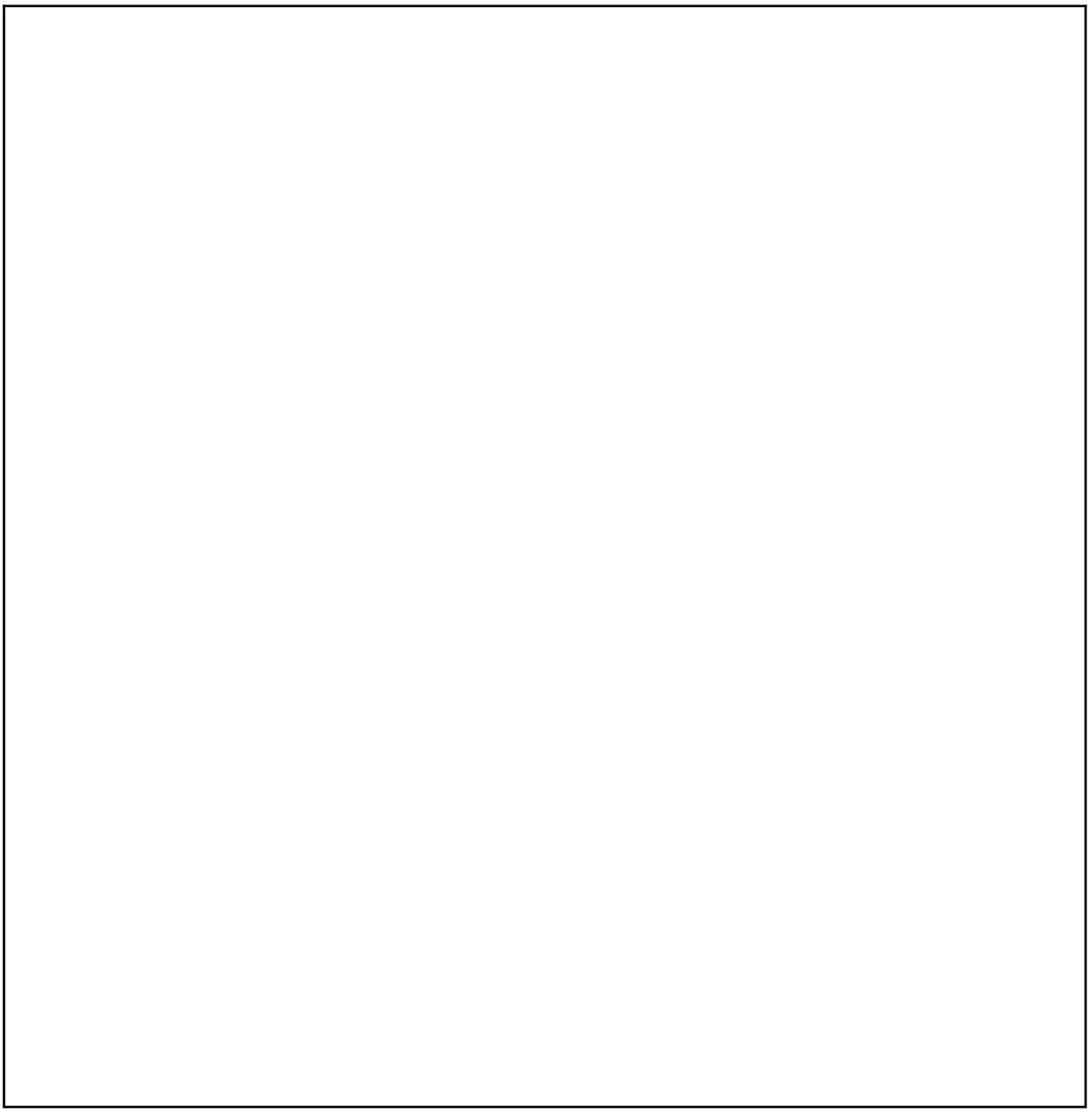
Magic

1. This is called in our routes.py file, and it routes all socket request that may fall under the specified namespace, as given in the above report for @socket.on()
2. Once it is called in routes.py, it calls an emit function, which takes in the event parameter which is the name of the user event to emit.
3. The emit function parses all of the parameters of the original request, and allows broadcast to all the other receiving sockets. The receiving sockets can be anything for the individual rooms or the drawers in our user interaction.
4. This is done thru the parent object, the object which was initialized when the user starting to use to client

- 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.
https://github.com/hahshtrit/solid-server/blob/e705a3599735c638853742a1ae4b96b40b56763b/routes.py#L198

https://github.com/miguelgrinberg/Flask-SocketIO/blob/f7ca69af129e6575f82142f27fbf9054522e969d/src/flask_socketio/__init__.py#L401



socketio.run()

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?
  Runs the entire flask app but now also creates and establishes a WebSocket connection along with the flask.route() function calls.
- 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 run.py in order to run the flask app with WebSocket functionality. It is found on line 7.

Magic ★★ ° ˚ ☽ ˚ ⌒ ˚ ★ ≡ ★ ≈

Dispel the magic of this technology. Replace this text with some that answer 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 technology in your project, you must be able to know how it works.
  After the TCP socket is created, it creates a wrapper for the WebSocket file. This WebSocket file handles basically all the WebSocket functionality we did in the homework. It will upgrade the TCP socket to a WebSocket after it receives the request. Then it has a buffer where it will continue reading from the WebSocket. It will parse these frames to successfully get the payload. It will also send response frames and even close the WebSocket if prompted to. This will all run continuously until the server is closed.
- 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.
https://github.com/hahshtrit/solid-server/blob/e705a3599735c638853742a1ae4b96b40b56763b/run.py#L7

https://github.com/miguelgrinberg/Flask-SocketIO/blob/f7ca69af129e6575f82142f27fbf9054522e969d/src/flask_socketio/__init__.py#L516
https://github.com/miguelgrinberg/Flask-SocketIO/blob/f7ca69af129e6575f82142f27fbf9054522e969d/src/flask_socketio/__init__.py#L624
Then it goes to the eventlet library to do the WebSocket parsing work
https://github.com/eventlet/eventlet/blob/88ec603404b2ed25c610dead75d4693c7b3e8072/eventlet/WSGI.py#L884
https://github.com/eventlet/eventlet/blob/88ec603404b2ed25c610dead75d4693c7b3e8072/eventlet/WebSocket.py#L43
https://github.com/eventlet/eventlet/blob/88ec603404b2ed25c610dead75d4693c7b3e8072/eventlet/WebSocket.py#L65
This line and below covers the parsing of the WebSocket frames.

SocketIO

Purpose

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

- What does this tech do for you in your project?
  This tech creates a Flask-SocketIO server. This takes in a flask application instance as a parameter. It allows us to take our original flask application and add WebSocket functionality.
- 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.
  It is used specifically in __init__.py on line 7.

Magic

Dispel the magic of this technology. Replace this text with some that answer 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 technology in your project, you must be able to know how it works. SocketIO creates an object of SocketIO
  It calls SocketIO which establishes a TCP connection that takes in data to be read and parsed according to the requirements. It handles buffering to read arbitrarily large 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.
  - 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.
https://github.com/hahshtrit/solid-server/blob/e705a3599735c638853742a1ae4b96b40b56763b/__init__.py#L7
https://github.com/miguelgrinberg/Flask-SocketIO/blob/f7ca69af129e6575f82142f27fbf9054522e969d/src/flask_socketio/__init__.py#L54
https://github.com/miguelgrinberg/Flask-SocketIO/blob/f7ca69af129e6575f82142f27fbf9054522e969d/src/flask_socketio/__init__.py#L191