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.

[websockets]

General Information & Licensing

Code Repository	python-socketio/server.py at main · miguelgrinberg/python-socketio · GitHub
License Type	BSD License
License Description	Redistribution and use of in source and binary forms, with or without modification, are permitted provided certain conditions are met
License Restrictions	● Redistributions must retain the same copyright notice whether it is in source code or binary form ● Neither the name of the copyright holder nor the names of its contributor may be used to endorse or promote produces derived from this software without specific prior

	written permission

Magic ⋆★⁺₊˚⋅˚☽˚⁀➷｡˚★彡⁺⋆༄

On line 17 of our code we create an instance of socketIO.This brings us to Flask-SocketIO/__init__.py at main · miguelgrinberg/Flask-SocketIO · GitHub, where this library is middleware to create a bridge between Flask framework and python's socket-io library. On line 21, we see a reference to socketio library. On initiation of the SocketIO class it calls init_app, which on line 243, it sets its own property "server" to socketio's server. It implements the same functions as socket io, however, it is just made so that it works with the flask environment and the functions often call functions from socket io, so I will be explaining how socket io works in the following paragraphs.

Socket.IO is library that enables low-latency, bidirectional communication, between client and server. This is achieved using an event based communication, where the client and server both listens to each other continuously until they receive an event that contains the message.

To do this, the server must be able to set events, to decide which events it decides to respond to. This is handled in line 172 - 219. Each event is given a name stored in namespace, which then is used as a key of the handler dictionary. Or it also provides a way to store multiple events and their respective handlers using the register_namespace() from line 256-269.

The main way for socketIO to send messages to multiple people is through the idea of using rooms. SocketIO will know all the rooms available and each clients can participate to be in one of the rooms to receive message as a group.

The main function used to broadcast message to all participants is emit() in with lines 271-321, it uses another python file called asyncio_manger.py which manages the client list for our server. It then is used to loop through its own name space and broadcast messages to all participants in the room (lines 11 - 34). It also is able to disconnect a client from the room, close the entire room, it relies on python's asyncio library and base_manager.

Lines 424 - 475, of Socket Server.py, contains the functions enter_room(), leave_room(), close_room(), and rooms() all use functions declared in asyncio manager, which is extends base manager. Base_manager.py is a data structure that contains the all the logs, server, rooms, sid, callback, and disconnect as dictionaries. Calling functions to add server or logs simply adds them to their respective dictionary, however the most important part in base_manager is how rooms is structured. It uses a 3-d array looking like room[namespace][room][sio_sid] = eio_sid. Namespace is the event to be listened on and room then contains all the participants listening to that specific event and in the same room (line 21). SIO_SID is the id given by socketio to the client for the sesssion which is mapped onto, EIO_SID which is the instance python's engine.IO which handles the transmission of messages.

On the client, it also implements on(), event(), and register_namespace() to listen on events coming in (lines 166-259). However, it is different from the server in that it does not manage the participants in the room, but it can call connect() to participate in one of those rooms(Lines 261 - 350) or disconnect to leave the room(lines 476-483). The emit(), send(), and call() are the same as ones in the server, however the client side is only concerned with sending message to the server and not other clients (lines 367 - 462)