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-SocketIO

General Information & Licensing

Code Repository	https://github.com/tylerdak/CSE312GroupProject
License Type	MIT
License Description	• The license gives permission to use, copy, modify, merge, publish, distribute, and sublicense the software so long as the copyright notice as well as the permission notice are included alongside it.
License Restrictions	• The license is very open-source friendly, the only restriction is that the author can't be held liable for how the technology is used.

Magic

What does this technology (library/framework/service) accomplish for you?

SocketIO gives our project access to low latency communication between the clients and the server. The benefit of using SocketIO is that using regular WebSockets may require implementing features such as reconnection and broadcasting for example which are already included in SocketIO.

How does this technology accomplish what it does?

Additional trace of libraries SocketIO can build off of:

- eventlet: will be called upon for its websocket transports.
- gevent: provides additional websocket support and long polling(however we will disable long polling due to project requirements)
- Werkzeug - a comprehensive web application library.

Adding flask-SocketIO to a flask application:

```
from flask import Flask, render_template
from flask_socketio import SocketIO

app = Flask(__name__)
app.config['SECRET_KEY'] = 'secret!'
socketio = SocketIO(app)

if __name__ == '__main__':

    socketio.run(app)
```

These lines of code allow the server to have access to the SocketIO library and allows the application to be accessible on a port with the additional benefits of the socketio library.

The application must also serve a page to the client that loads the SocketIO library. Below is javascript within an html that can be used to establish that connection with the client.

```
<script
src="https://cdnjs.cloudflare.com/ajax/libs/socket.io/4.0.1/socket.io.js"
integrity="sha512-q/dWJ3kcmjBLU4Qc47E4A9kTB4m3wuTY7vkFJDTZKjTs8jhyGQnaUrxa0
Ytd0ssMZhbNua9hE+E7Qv1j+DyZwA==" crossorigin="anonymous"></script>
<script type="text/javascript" charset="utf-8">
    var socket = io();
    socket.on('connect', function() {
        socket.emit('my event', {data: 'I\'m connected!'});
    });

</script>
```

The ‘src’ imports the library into the script, and socket.on() can be used to read and write any data you need.

The messaging system we are required to implement is supported by SocketIO. This can be done by creating an event handler. Below is an example of receiving a message.

```
@socketio.on('my event', namespace='/test')

def handle_my_custom_namespace_event(json):
```

```
    print('received json: ' + str(json))
```

The event handler works just like an endpoint. The namespace is the path, and ‘my_event’ is the data that is received, which can be used to handle the message.

Sending a message is similar, but has slight changes.

```
@socketio.on('my event')
def handle_my_custom_event(json):

    emit('my response', ('foo', 'bar', json), namespace='/chat')
```

Sending messages utilizes the send() or emit() functions. emit() allows you to create and transfer unnamed custom events, and send() will transfer named events.

Another feature that SocketIO helps implement is broadcasting.

```
@socketio.on('my event')
def handle_my_custom_event(data):

    emit('my response', data, broadcast=True)
```

When a message is sent with ‘broadcast=True’ all clients connected to the namespace receive it.

SocketIO also supports a room feature where a subset of users can communicate amongst each other.

```
from flask_socketio import join_room, leave_room

@socketio.on('join')
def on_join(data):
    username = data['username']
    room = data['room']
    join_room(room)
    send(username + ' has entered the room.', to=room)

@socketio.on('leave')
def on_leave(data):
    username = data['username']
    room = data['room']
    leave_room(room)

    send(username + ' has left the room.', to=room)
```

The join_room and leave_room functions create a connection with all the users, and provides a session ID for it.