# [Flask]

## General Information & Licensing

| | |
|---|---|
| Code Repository | https://github.com/pallets/flask |
| License Type | BSD |
| License Description | ● Redistributions of source code must retain the above copyright notice, this list of conditions and the following disclaimer.<br>● 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.<br>● Neither the name of the copyright holder nor the names of its contributors may be used to endorse or promote products derived from this software without specific prior written permission. |
| License Restrictions | ● It prohibits the names of the authors from being used to endorse or promote products relating to the software.<br>● It prohibits others from using the name of the copyright holder or its contributors to promote derived products without written consent.<br>● This software is provided by the copyright holders and contributors “as is” and any express or implied warranties, including, but not limited to, the implied warranties of merchantability and fitness for a particular purpose are disclaimed. in no event shall the copyright holder or contributors be liable for any direct, indirect, incidental, special, exemplary, or consequential damages (including, but not limited to, procurement of substitute goods or services; loss of use, data, or profits; or business interruption) however caused and on any theory of liability, whether in contract, strict liability, or tort (including negligence or otherwise) arising in any way out of the use of this software, even if advised of the possibility of such damage. (liability and warranty) |

# [Socketio]

## General Information & Licensing

| | |
|---|---|
| Code Repository | https://github.com/miguelgrinberg/Flask-SocketIO |
| License Type | MIT |
| License Description | ● A free software license for a copyrighted work that offers freedoms such as publishing a work to the public domain.<br>● Licensed works, modifications, and larger works may be distributed under different terms and without source code.<br>● Grants use rights, including right to relicense (allows |

| | |
|---|---|
| | prioritization, license compatibility) |
| License Restrictions | ● It carries only minimal restrictions on how the software can be used, modified, and redistributed, usually including a warranty disclaimer.<br>● The software is provided "as is", without warranty of any kind, express or implied, including but not limited to the warranties of merchantability, fitness for a particular purpose and non infringement. In no event shall the authors or copyright holders be liable for any claim, damages or other liability, whether in an action of contract, tort or otherwise, arising from, out of or in connection with the software or the use or other dealings in the software. (liability and warranty) |

## *Magic*

For prasing HTTP header, we don't do the same with as we did on homewor which is getting header and prasing the header by position and "\r\n". Instead, we use two parts to deal with the header. First, we use @app.route() in Flask to prase the request header and jump between pages. Second, we use @socketio.on() in SocketIO to prase the response header that socketio send out when operating in the pages.

Flask.route:
https://flask.palletsprojects.com/en/2.2.x/api/#flask.Flask.route

Socketio.on:
https://flask-socketio.readthedocs.io/en/latest/getting_started.html

**Flask.route**

Flask.route is deafultly setting to deal with "GET" request. It takes two parameter, "rule" and "options". "Rule" represents the URL rule string that we want. "Options" represents the extra options passed to the rule object.

In our website, we mainly use two routes for the website, "/" and "/game". More specific, render_template() accepts one positional argument, which is the name of the template found in our templates folder, in this case, we set it to "/static" folder. When you go to "/" root page, the Flask.route will leads you to the "lobby.html" page, similarly, if you go to "/game" page, it will leads you to the "index.html" page.

The following picture shows the implementation of Flask.route uses in our website.

```python
app = Flask(__name__, static_url_path="/static")
app.config['SECRET'] = "secret!123"
socketio = SocketIO(app, cors_allowed_origins="*")

@app.route('/')
def index():  # put application's code here
    # users_account.drop()
    return render_template("lobby.html")

@app.route('/game')
def game():  # put application's code here
    return render_template("index.html", amount_ready=str(len(game_engine.ready_list)))
```

**SocketIO.on**

SocketIO.on collaborates with amount of functions such as “socket.send”, “socket.emit”, ect. If we apply socketio on server-side, then we should use socketio.on() in Python, if we apply it on client-side, then we should use socket.on() in JavaScript. Both of the function create event handler for define events. In order to recevie header, the socket must send a header first. Therefore, the following code sends the message that once a user press the “ready” button in the game for instance, it sends out the message that includes “User Ready!” to indicate that.

```javascript
// user ready
$("#ready-button").click(function () {
    socket.send(socket_id.toString() + ": User ready!");
})
```

Then, the following code gives an example that socketio deal with the “message” event happens on “/game/” page, which prase the receive header and check if the header message is from “/game” and if include “User Ready!”. If so, then do the defined operations.

```python
@socketio.on('message', namespace='/game')
def handle_message(message):
    # print("Received message: " + message)
    game_engine.alert_status = []
    if "User connected!" in message:
        game_engine.users.append(int(message.split(":")[0]))
        # print("Connect Successful")
    elif "User ready!" in message:
        user_id = int(message.split(":")[0])
        if user_id not in game_engine.ready_list:
            game_engine.ready_list.append(user_id)
            send(json.dumps(["ready", game_engine.ready_list]), broadcast=True)
        if len(game_engine.ready_list) == 4:
            for i in range(0, len(game_engine.ready_list)):
                game_engine.users_info[i]["user_id"] = game_engine.ready_list[i]
                game_engine.users = game_engine.ready_list
            send(json.dumps(["start", {"roll_num": 1, "user": game_engine.users_info}]), broadcast=
```

Besides the above exmaple, we also have plenty of route that handle distinctive response such as, “signup”, “login”, “roll dice”, etc. SocketIO helps to prase the header and directly walk in the function we needed to implement.