WebSockets 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 / Python

General Information & Licensing

Code Repository	https://github.com/emilydesantis/cse312-because-the-internet
License Type	MIT
License Description	● Permission to use, copy, modify, and distribute this software and its documentation for any purpose and without fee is hereby granted, provided that the above copyright notice appear in all copies and that both that copyright notice and this permission notice appear in supporting documentation, and that the name of M.I.T. not be used in advertising or publicity pertaining to distribution of the software without specific, written prior permission. ● M.I.T. makes no representations about the suitability of this software for any purpose. It is provided "as is" without express or implied warranty.
License Restrictions	● Redistributions of source code must retain the above copyright notice, this list of conditions and the following disclaimer. ● 2. 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. ● 3. 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. ● 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.

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? Please explain this in detail, starting from after the TCP socket is created
- 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 will likely grow beyond the page

Flask is a lightweight WSGI web application framework. It is designed to make getting started quick and easy, with the ability to scale up to complex applications. It began as a simple wrapper around Werkzeug and Jinja and has become one of the most popular Python web application frameworks.

Flask is designed to be a “simple extension to implement WebSocket communication between a client and the server”.We generally utilize socketio.on, a decorator that registers our events in the game.

In line 16 of server.py, a new instance of the Flask class is created to the app variable.
https://flask-socketio.readthedocs.io/en/latest/getting_started.html#initialization

In line 137 of server.py, @socketio.on('create_room') registers a handler for the 'create_room' event. So when a client sends a 'create_room' event to the server, the handle_create_room function will be called. Applying the user data, creates a new room and adds the user passed in the parameter, then sends events to all clients in other rooms for updates in UI and navigation.
https://flask-socketio.readthedocs.io/en/latest/getting_started.html?highlight=socketio.on#connection-events

In line 150 of server.py, @socketio.on('join_room') registers a handler for the 'join_room' event. So when a client sends a 'join_room' event to the server, the handle_join_room function will be called. Handling the addition of a new player to a previously existing room, and sends events to all clients for updates in UI and navigation.
https://flask-socketio.readthedocs.io/en/latest/getting_started.html?highlight=socketio.on#connection-events

In line 166 of server.py, @socketio.on('join_lobby') registers a handler for the 'join_lobby' event. So when a client sends a 'join_lobby' event to the server, the join_lobby function will be called. Handling the addition of a new player to the lobby of a specific room and sends events to all clients in that room to update the UI with the current list of users in the lobby.
https://flask-socketio.readthedocs.io/en/latest/getting_started.html?highlight=socketio.on#connection-events

In line 184 of server.py, @socketio.on('submit_word') registers a handler for the 'submit_word' event. So when a client sends a 'submit_word' event to the server, the handle_submit_word function is called. Handling the submission of a new word for players to guess in a specific room. It sets the game state for the room accordingly, additionally starting a new round of the game.
https://flask-socketio.readthedocs.io/en/latest/getting_started.html?highlight=socketio.on#connection-events

On line 196 of server.py, @socketio.on('submit_question_or_guess') registers a handler for the 'submit_question_or_guess' event. So when a client sends a 'submit_question_or_guess' event to the server, the handle_submit_question_or_guess function is called. Handling the submission of a question or guess by a player in a specific room. It checks if the guess is correct and sends the appropriate events to all clients in the room. If the guess is incorrect, this function decrements the questions_left count in the game_state dictionary then sends events to continue the game. When questions_left reaches 0, the function sends a 'game_over' event to all clients in the room.
https://flask-socketio.readthedocs.io/en/latest/getting_started.html?highlight=socketio.on#connection-events

On line 224 of server.py, @socketio.on('submit_answer') registers a handler for the 'submit_answer' event. So when a client sends a submit_answer event to the server, the handle_submit_answer function is called. Handling the submission of a player's answer to the current round of the game. As well as updating the UI so it is disabling the yes and no buttons, the event registered on line 147 of page3.html and displays the correct answer for all players in the room.
https://flask-socketio.readthedocs.io/en/latest/getting_started.html?highlight=socketio.on#connection-events

On line 232 of server.py, @socketio.on('start_game') registers a handler for the 'start_game' event. So when a client sends a 'start_game' event to the server, the handle_start_game function is called. Handling the 'navigate_to_page3' event to all clients in the room_name room with an empty dictionary as its data argument. This event signals the clients to navigate to the third page of the game.
https://flask-socketio.readthedocs.io/en/latest/getting_started.html?highlight=socketio.on#connection-events

On line 246 of server.py, @socketio.on('start_question_round') registers a handler for the 'start_question_round' event on line 112 of page3.html. So when a client sends a 'start_question_round' event to the server, the handle_start_question_round function is called. Handling the start of a new round of the game in a specific room. And sends necessary events to clients as a means to signal the start of the round and to disable the "yes" and "no" buttons until a question is asked.
https://flask-socketio.readthedocs.io/en/latest/getting_started.html?highlight=socketio.on#connection-events

On line 255 of server.py, @socketio.on('correct_guess') registers a handler for the 'correct_guess' event on line 93 of lobby.html. So when a client sends a 'correct_guess' event to the server, the handle_correct_guess function is called. Handling correct guesses for the word by a player in a specific room. It sends a 'game_over' event registered in line 164 of page3.html to all clients in the room with a win result.
https://flask-socketio.readthedocs.io/en/latest/getting_started.html?highlight=socketio.on#connection-events

On line 267 of server.py, @socketio.on('select_role') registers a handler for the 'select_role' event on line 76 and 85 of lobby.html. So when a client sends a 'select_role'

event to the server, the handle_select_role function will be called. When a role selected by the client is 'select_word', the function sends a 'choose_word' event to the room_name room. This event signals to the client with the 'select_word' role to submit a word for the other player to guess.
https://flask-socketio.readthedocs.io/en/latest/getting_started.html?highlight=socketio.on#connection-events

On line 285 of server.py, @socketio.on('hide_ask') registers a handler for the hide_ask event on line 88 of lobby.html. So when a client sends a hide_ask event to the server, the handle_hide_ask function will be called. Handling the hide_ask event by hiding the "Ask" button on the clients' UI in a specific room.
https://flask-socketio.readthedocs.io/en/latest/getting_started.html?highlight=socketio.on#connection-events

On line 291 of server.py, @socketio.on('hide_select') registers a handler for the 'hide_select' event on line 79 of lobby.html. So when a client sends a 'hide_select' event to the server, the handle_hide_select function will be called. Handling the event where a player has selected a word and hiding the UI hat is allowing word selection for all players in the room.
https://flask-socketio.readthedocs.io/en/latest/getting_started.html?highlight=socketio.on#connection-events

On line 62 of page3.html,.connect() is utilized to create a new variable to our WebSocket connection object to the server, used to send data between the client and server in real time.
https://flask-socketio.readthedocs.io/en/latest/getting_started.html?highlight=socket.emit#initialization

On line 68 of page3.html, socket.emit is utilized to send a message to the server containing user data with the ‘join_lobby’ event allowing them to join the room when the server responds with either creating a new lobby, adding a user to the lobby or sending a message back indicating the user has successfully joined.
https://flask-socketio.readthedocs.io/en/latest/getting_started.html?highlight=socket.emit#initialization

On line 82 of page3.html, socket.emit is utilized to send a message to the server containing user data with the ‘submit_word’ event, allowing the word to be checked and validated, update the game state and send that to the clients in that room.
https://flask-socketio.readthedocs.io/en/latest/getting_started.html?highlight=socket.emit#initialization

On line 93 of page3.html, socket.emit is utilized to send a message to the server containing data with the ‘correct_guess’ event when the guess matches the submitted word.
https://flask-socketio.readthedocs.io/en/latest/getting_started.html?highlight=socket.emit#initialization

On line 97 of page3.html, socket.emit is utilized to send a message to the server containing data with the ‘submit_question_or_guess’ event when a player inputs a guess or their guess does not match the current word.
https://flask-socketio.readthedocs.io/en/latest/getting_started.html?highlight=socket.emit#initialization

On line 103 of page3.html, socket.emit is utilized to send a message to the server containing data with the ‘submit_answer’ event when the user clocks on the ‘yes-button’ element and updates the UI to reflect the user’s answer “Yes”.
https://flask-socketio.readthedocs.io/en/latest/getting_started.html?highlight=socket.emit#initialization

On line 108 of page.html, On line 103 of page3.html, socket.emit is utilized to send a message to the server containing data with the ‘submit_answer’ event when the user clocks on the ‘no-button’ element and updates the UI to reflect the user’s answer “No”.
https://flask-socketio.readthedocs.io/en/latest/getting_started.html?highlight=socket.emit#initialization