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.

[insert library/framework name here]

General Information & Licensing

Code Repository	https://github.com/animalman33/cse312-auction/
License Type	BSD
License Description	• Permissive open-source license that originated from the University of California, Berkeley • Allows users to freely use, modify, distribute, and sublicense the software, whether for commercial or non-commercial purposes
License Restrictions	• Places minimal restrictions on the usage of the code, providing a considerable amount of freedom to developers and users

Magic

- How does this technology do what it does? Please explain this in detail, starting from after the TCP socket is created

To facilitate the auction functionality, the code defines several event handlers using the @socketio.on decorator. These event handlers handle specific events sent from the client-side.

The join_auction event handler is triggered when a user wants to join an auction. It receives the user object and the auction ID as parameters. Inside the handler, it stores the user's information in the users dictionary, joins the specified auction room using join_room(auctionID), sends a confirmation message to the user using send, and broadcasts a message to all connected clients indicating that the user has joined the auction.

The create_auction event handler is triggered when a user wants to create an auction. It receives the user object, the auction price, and the auction name. In this handler, it retrieves the username based on the session ID, adds the auction to the database using Database.add_auc, and performs any necessary actions related to auction creation.

The leave_auction event handler is triggered when a user wants to leave an auction. It takes the user object and the auction ID as parameters. Within the handler, it ensures the user leaves the specified auction room using leave_room(auctionID), sends a confirmation message to the user indicating their departure using send, and broadcasts a message to all connected clients indicating that the user has left the auction.

The new_bid event handler is triggered when a user places a new bid. It receives the bid amount as a parameter. Inside the handler, it retrieves the username based on the session ID, adds the bid to the database using Database.add_bid, and broadcasts a message to all connected clients indicating the username and the amount of the new bid.

These event handlers collectively enable the functionality of joining auctions, creating auctions, leaving auctions, and placing bids using websockets. By utilizing Flask-SocketIO, the code establishes bidirectional communication between the server and the clients, allowing real-time updates and interactions in the auction house application.

- 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.

join_auction event handler:

https://github.com/animalman33/cse312-auction/blob/Backend-WebSockets/auction.py

```
@socketio.on('join_auction')
@current_user
def join_auction(user, auctionID):
    users[user] = request.sid
    join_room(auctionID)
    send("You have entered!", to=auctionID)
    emit("feed", {user} + ' has joined.', broadcast=True)
```

Lines 14-20

create_auction event handler:

https://github.com/animalman33/cse312-auction/blob/Backend-WebSockets/auction.py

```
@socketio.on('create_auction')
@current_user
def create_auction(user, price, name):
    # Gets the user based off session ID
    username = None
    for user in users:
        if users[user] == request.sid:
            username = user
    Database.add_auc(username, Database.get_user_info(username)[2], name, price)
```

Lines 23-31

leave_auction event handler:

https://github.com/animalman33/cse312-auction/blob/Backend-WebSockets/auction.py

```
@socketio.on('leave')
@current_user
def leave_auction(user, auctionID):
    leave_room(auctionID)
    send("You have left!", to="/user/home")
    emit("feed", {user} + ' has left.', broadcast=True)
```

Lines 34-39

new_bid event handler:

```
@socketio.on('new_bid')
```

```
@current_user
def new_bid(amount):
    # Gets the user based off session ID
    username = None
    for user in users:
        if users[user] == request.sid:
            username = user
    # Add bid and send to feed
    Database.add_bid(username, amount, rooms(request.sid),
Database.get_user_info(username)[2])
    emit("feed", {username} + ' has bid ' + {str(amount)}, broadcast=True)
```

Lines 42-52

https://github.com/animalman33/cse312-auction/blob/Backend-WebSockets/auction.py