SFWRTECH 4WP3 – Advanced Web Programming - Assignment #10

Weight

4% of total course grade

Due date

Tuesday April 12th at 11:59pm

Primary learning objectives

- Create a chatbot application.
- Use a cloud key-value database service in a web application.

Requirements

In this assignment you will create a chatbot for the **Hamilton Homes** web application developed in previous assignments. See the starter code provided for the chat interface itself, and the same backend as Assignment #4 is also provided as starter code.

When the user visits http://localhost:3000/ in the web browser they should be presented with the "home page" of your Hamilton Homes website. It should be styled as it has been on your previous assignments. Don't worry about making any other pages work, but if you would like to integrate the pages created for previous assignments that's up to you.

On the "home page" in the lower right corner there should now be a chat interface, using the same styles and technique as from the provided starter code. If you want to modify the styles slightly to match your design that's OK, but it should be very similar and the button options should be the same. When the user enters a message into the message area and clicks send, websockets should be used to send a message to the server, and the server should respond to that message. The server's response should be put into the list of chat messages so that the user can see all messages in the conversation with the chatbot.

No communication should happen with AJAX. The backend from Assignment #4 is provided as starter code as an example of performing simple searches on the "mock data" provided in the backend.

A "command" is some text entered by the user that conforms to a specific syntax, with some expected result in return. So these would be some potential commands:

help - shows a listing of all comands
bathrooms 3 – show all houses with 3 or more bathrooms
bathrooms 3 bedrooms 2 - show all houses with 3 or more bathrooms, 2 or more bedrooms

We would say that in the case of **bathrooms 3** that 3 is an "argument" to the command, like an argument to a function. We can use the JavaScript split() function to simply parse these commands in order to extract the command name (e.g. bathrooms) and argument(s) (e.g. 3): https://www.w3schools.com/jsref/jsref_split.asp.

Ideally in a situation like this we might make a more conversational chatbot that uses ML techniques among others to determine the intention of a question and respond appropriately, but that would be another class of lectures at least so we can't fit that in to this assignment.

Your chatbot should respond to at least 5 commands, with the following requirements:

- One command should be a **help** command (the text "help") that displays all of the commands and how they work in a neatly formatted way in the messages list.

- At least 3 commands should accept at least one argument that is used to present housing data from the "mock data" provided to the user in the messages list.
    - For example a command like **bathrooms 3** could provide a listing of houses with 3 or more bathrooms.

- The fifth command should do **something** useful within the context of the Hamilton Homes website, whether that's displaying more data about homes or something else (i.e. you want to get fun and creative with it)

Use a redis database to store the following information:

- A log of all commands that have been entered.
    - Store the string of each command entered by users using this schema:
        - **log:count –** where count is updated for each log entry
        - So for example...
            - log:1 – stores the first command ever entered
            - log:2 – stores the second command ever entered
            - … and so on ….
        - In order to implement this you will need to store the "count" value itself in the redis database, that your application knows where it should write the next log entry!
- The number of times each command has ever been requested. Use whatever key names you like for this.

As each command is received by the server, the redis database should be updated. None of this data should be "reset" every time the application runs, it should be an ongoing tally stored in the Redis database.

Submission

Delete the node_modules folder before submitting your work, but keep all the other files, including the package.json file. Zip your solution as a10_firstname_lastname.zip and upload it to the dropbox on Avenue.

Marking rubric

Component	**Description**	**Marks**
Help command	Displays list of all other commands in the messages list.	/10
Commands with arguments.	At least 3 commands that successfully use arguments for dynamic display of housing data.	/30
5 commands	At least 5 commands exist, the 5th command should do something meaningful in the context of the application	/10
Webscokets	Websockets used for all communication.	/20
Redis	Redis database used as required to store log of commands, number of times command requested.	/30
	Total:	/100