SE 3200: Web Application Development I

Assignment: Message Log

Requirements

- Using Python, create a server web application which implements an API that receives, stores, and returns simple messages, according to the following specifications:
  - Two RESTful routes:
    - `POST /messages`: receives a message within the body of the request and records the message in the message log, by writing (appending) the message to a file on the local filesystem. The server should respond appropriately with the status code `201 Created`. No response body content is necessary.
    - `GET /messages`: returns all messages contained within the message log, by reading the messages from the file on the local filesystem. The data should be returned as a JSON array within the response body, and the server should respond appropriately with the status code `200 OK` and the `Content-Type` response header set correctly.
    - Both routes should be implemented according to REST standards. Test both of your routes using a tool such as Postman or curl.
  - CORS should be implemented server-wide in order to support Ajax requests from client applications.
  - If a `GET` or `POST` request is received that does not conform to the paths defined above (or any others that you choose to implement), then the server should return an appropriate Not Found response, with the correct status code, and content that properly explains the reason for this response. The content type may be plain text or HTML; set the response header correctly.
- Using JavaScript, create a client web application that communicates with your server application, using its API, on behalf of the user, with the following:
  - A simple form that allows the user to enter a message and click a button to record their message. Upon submission, the message should be sent to the server API using the Fetch API, using the appriorate API route above. The form should be styled to allow the user to easily enter a message of any length.
  - A list of messages that contains all messages returned by the server API, displayed in the order they are given. The messages should be requested from the server API using the Fetch API, using the appropriate API route above. The list should be styled to cleanly display a large list of messages, containing messages of any length.
  - After the user submits a new message to the server API, the list should be refreshed to reflect the new message. After the server responds from the initial request which sent the message, a subsequent request should be made to refresh the list.
  - All data sent and received to and from the server API should be implemented using Ajax requests.
- You may take liberties to modify your application's features and purpose from that described above, but the overall specifications and structure listed above should still be met.
- Make your application look professional and presentable. Use valid HTML and CSS to structure and style your application.
- No third-party JavaScript or CSS libraries or frameworks may be used without prior instructor permission.

Submission

1. Submit your project using Git and GitHub. Start by creating a repo for this assignment here.
2. Show your completed assignment to the instructor during class or office hours to receive credit.