Simple Messenger System [48 Hrs]

PLEASE DOCKERIZE YOUR SOLUTION AND INCLUDE DOCKER COMPOSE / DOCKERFILE

Submit a zipped source code folder with a readme on how to make and deploy the code, follow the API names given in the exhaustive examples below.

Include git history in the submission and make multiple commits so that progress can be tracked.

Design a simple peer-to-peer messaging server where a user can be registered with a unique human readable username and they can start message threads with one another. The server should also fetch chat history requested by the user. Data passed as JSON from the command line in the examples below.

Support the following:

1. Creating a user on the server: Build REST API /create/user

Request:
curl --request POST 'http://localhost:8080/create/user' \
--data-raw '{"username":"johnsmith007","passcode":"quirkyparrot"}'

Response: {"status":"success"}

Request:
curl --request POST 'http://localhost:8080/create/user' \
--data-raw '{"username":"johnsmith007","passcode":"fadnodebreath"}'

Response: {"status":"failure", "message":"User already exists"}

Request:
curl --request POST 'http://localhost:8080/create/user' \
--data-raw '{"username":"dummy009","passcode":"wawa009"}'

Response: {"status":"success"}

Request:
curl --request POST 'http://localhost:8080/create/user' \
--data-raw '{"username":"dummy008","passcode":"cocopuff"}'

Response: {"status":"success"}

2. Logging into user session: Build a REST API for authorising the user. Keep implementation simple and security features are not needed but can be considered a bonus. A user can only perform send and receive if he/she is logged in.

Request:
curl --request POST 'http://localhost:8080/login' \
--data-raw '{"username":"johnsmith007","passcode":"quirkyparrot"}'

Response: {"status":"success"}

3. Fetching Users on the server: Build an API to get the users on the server

Request:
curl --request GET 'http://localhost:8080/get/users' \

Response: {"status":"success", "data" : ["dummy008", "dummy009"]}

4. Fetch unread messages: Build REST API to fetch a user's unread messages

Request:
curl --request GET 'http://localhost:8080/get/unread' \
--data-raw '{"username":"johnsmith007"}'

Response: {"status":"success", "message": "You have message(s)", "data" : [{"username":"dummy009", "texts": ["thanks man!"]}]}

Request:
curl --request GET 'http://localhost:8080/get/unread' \
--data-raw '{"username":"johnsmith007"}'

Response: {"status":"success", "message" : "No new messages"}

5. Send a new message: API to send a specific user a message

Request:
curl --request POST 'http://localhost:8080/send/text/user' \
--data-raw '{"from":"johnsmith007", "to":"dummy009","text":"no problem"}'

Response: {"status":"success"}

6. Get chat history with a specific user

Request:
curl --request GET 'http://localhost:8080/get/history' \
--data-raw '{"friend":"dummy009", "user":"johnsmith007"}'

Response: {"status":"success", "texts":[{"dummy009":"thanks man!"},{"johnsmith007":"no problem"}]}

7. Logout the current client session

Request:
curl --request POST ‘http://localhost:8080/logout’ \
--data-raw ‘{“username”:”johnsmith007”}’

Response: {“status”:”success”}

Bonus:
implement send/text/group