AggieFit: Message-Board Architecture and User Instruction Details

The time taken to complete this task: 3 hours

The requirement for the AggieFit message-board platform is to support the following functions:

1. “select”: which selects the board specified
2. “read”: which reads all the messages in the selected board; returns a message to select a board if not selected
3. “write”: which writes messages to the selected board; returns a message to select a board if not selected
4. “listen”: which listens to messages which are published to a board

The architecture behind the implementation of the AggieFit Message-Board uses the following technologies:

1. MongoDB for persistence of the messages
2. REDIS which provides the Pub-Sub functionality for real-time listening of the messages

The implementation is as follows:

1. When the application is brought up, connections to a MongoDB server and a REDIS server are established and the user is prompted to enter a command from a set of 5 commands (select, read, write, listen, quit)
2. For simplification purposes, this application is executed as a loop which runs until interrupted.
3. When the user selects a board by providing a name, a variable in the instance of the application running on his machine is set to the name of the board.
4. The user uses the write command to write to the board. When he is in fact writing to the board, the message is being published to REDIS and also persisted in a MongoDB collection. The publish in REDIS is happening to a channel named by the board name. The persistence to Mongo is happening to a collection as a document with name key as board name and a message key with the message.
    a. data = {“name”: “board-name”, “message”: “write-message”}
5. The user uses the read command to read all messages. When a read command is executed, all the messages in the selected board are fetched from the MongoDB collection (using the find query, with the board name as parameter) and displayed to the user.
6. The user uses the listen command to subscribe to the channel he selected. Immediately, the REDIS subscribe is executed in the background and any writes by any other user are immediately displayed to the current user.
7. The user uses the quit command to exit the application.
8. Errors are handled appropriately. Wrong inputs are the main form of error. On screen messages are shown to the user prompting them to enter the correct messages.

User Instructions:

1. Start up a REDIS server on the localhost on port 6379.
2. Make sure the Mongo connection on 34.233.78.56 is up and running.
3. The database is named message_board_826006434.
4. The collection is also named message_board_826006434.
5. Using the command line, navigate to the folder where the “task2” folder is extracted.
6. Run the command “python main.py”
7. The CLI will prompt you to enter a command.
8. You can choose from the following 5 commands:
    a. select <board-name>
    b. read
    c. listen
    d. write <write-message>
    e. quit
9. Make sure you run the select command first. This selects the message board to operate on.
10. “read” reads all the messages in the selected board.
11. “write” <write-message> write write-message to the selected board.
12. “listen” starts up a subscription. Any other writes on the selected board will be displayed immediately on the screen.
13. Use Ctrl+C to exit out of the subscription.
14. “quit” exits the application.