DS5760 NoSQL for Modern Data Science Applications

Mini Project 1 – Building a Redis Pub/Sub-Based Chatbot

## I. Purpose:

* Creating a simple chatbot using Redis's Pub/Sub mechanism
* Through practice, obtaining a solid understanding of related concepts such as real-time messaging, Redis Pub/Sub, and integrating Redis with Python
* Interacting with Docker containers and becoming proficient in the use of docker-compose

## II. Assignment Tasks [100 POINTS]:

1. [10 POINTS] Setting up two Docker containers using docker-compose

* One container: Python client (see class example)
* Another container: Redis

1. [5 POINTS] Chatbot Initialization

* When the chatbot starts, it should introduce itself and provide a list of commands that the user can use (see all required commands in step 6).

1. [10 POINTS] User Identification:

* A user should be able to identify themselves with a username.
* Store user information, including their name, age, gender, and location, in Redis DB.

1. [20 POINTS] Channels:

* Users should be able to join or leave a channel.
* When a user sends a message to a channel, all users subscribed to that channel should receive the message.
* You do not have to implement a mechanism to stop listening to a channel

1. [20 POINTS] The chatbot should recognize the following special commands and provide automated responses to them.

* !help: Provide a list of available commands.
* !weather <city>: Provide a mock weather update for the given city. You can store some data in Redis or store some static data in your Python code (I recommend the former as it gives you more practice with Redis ☺)
* !fact: Provide a random fun fact. You can store some data in Redis or store some static data in your Python code. (I recommend the former just because ☺)
* !whoami: Provide information about the user based on their username

1. [20 POINTS] Documentation

* [10 POINTS] Using comments throughout your code, indicate how your program meets the requirements of each assignment task in this file.
* [10 POINTS] In a README file, include the following:
  + Instructions on how to set up and use your chatbot and details of your development process. You can provide screenshots as examples
  + Monitoring: Use the redis-cli monitor command to monitor interactions in real-time. Screenshot the output for at least 3 different user inputs to your bot.
  + Keep in mind that this assignment is designed to help you create a project on GitHub that can be used later to showcase your NoSQL skills, so be as comprehensive as you want with the README
* If you leveraged Generative AI (GenAI) in any way, you are required to disclose how you used it, including how much work was your own vs GenAI output – refer to the course syllabus for the course policy on the use of GenAI.

1. **[EXTRA CREDIT] I hope you will have a lot of fun working on this assignment and make it your own project that can be added to your portfolio.** Anything you do beyond what you are asked of above, we will reward you with extra points between 1-20 points depending on the complexity of the extra features. For example, a simple way to earn 2 extra points would be to cleanly format the output text. A slightly more complex example feature may be implementing multiple channel subscriptions, which can earn you 7 points. Something that can earn you 20 points would be a new command or a new feature that makes the chatbot more user friendly.

**Submission:**

A zip file containing the items below:

* Your docker-compose.yml file:
* Your .py client file
* README file

**Sample output**

* The following samples are from my terminal output after running an example solution I created. Your output does not have to match 100% of what I have. I encourage you to be creative! **A screenshot of a computer program

  Description automatically generatedA screenshot of a computer

  Description automatically generatedA screenshot of a computer program

  Description automatically generated**

**Additional Notes**

* When evaluating your mini project submission, the TAs will be asked to look for the following attributes in your submission:
  + **Functionality**: Does the chatbot function as intended? Are there observable bugs?
  + **Code Quality**: Is the code well-structured, organized, and commented? Are best practices adhered to?
  + **Documentation**: Is the provided documentation clear and comprehensive?