[CS209A-23Fall] Final Project (100 points)

Background

In the process of software development, many questions will arise. Developers may resort to Q&A website to post questions and seek answers.

Stack Overflow is such a Q&A website for programmers, and it belongs to the Stack Exchange Network. Stack Overflow serves as a platform for users to ask and answer questions, and, through membership and active participation, to vote questions and answers up or down and edit questions and answers in a fashion similar to a wiki. Users of Stack Overflow can earn reputation points and "badges"; for example, a person is awarded 10 reputation points for receiving an "up" vote on a question or an answer to a question, and can receive badges for their valued contributions. Users unlock new privileges with an increase in reputation like the ability to vote, comment, and even edit other people's posts.

In this final project, we'll use Spring Boot to develop a web application that stores, analyzes, and visualizes Stack Overflow Q&A data w.r.t. `java programming`, with the purpose of understanding the common questions, answers, and resolution activities associated with Java programming.

Data Collection (10 points)

On Stack Overflow, questions related to Java programming are typically tagged `java`. You could use this `java` tag to identify java-related questions. A question and all of its answers and comments are together referred to as a `thread`.

For **java-related threads** on Stack Overflow, we are interested in answering a list of questions as described below. You should first collect proper data from Stack Overflow to answer these questions. Please check the official Stack Overflow REST API documentation to learn the REST APIs for collecting different types of data.

- You may need to create a Stack Overflow account in order to use its full REST API service.
- API requests are subject to rate limits. **Please carefully design and execute your requests, otherwise you may reach your daily quota quickly**.
- Connections to Stack Overflow REST service maybe unstable sometimes. So, **please start the data collection ASAP!**

There are over 1 million threads tagged with `java` on Stack Overflow. You DON'T have to collect them all. Yet, you should collect data for **at least 500 threads** in order to get meaningful insights from the data analysis.

Important:

Data collection is ***offline***, meaning that you need to collect and persist the data first. It is recommended that you use a database (e.g., PostgreSQL, MySQL, etc.) to store the data. However, it is also fine if you store the data in plain files. In other words, when users interact with your application, **the server should get the data from your local database** (or local files), instead of sending REST requests to Stack Overflow on the fly.

Hence, the data analysis for the below questions should be performed on the dataset you collected. That is, we first collect a subset of Stack Overflow data (e.g., 500 threads tagged `java`) and then answer the following questions using this subset.

Topic Popularity (20 points)

We have covered various topics in this course, e.g., generics, I/O, lambda, multithreading, socket, etc. It's interesting to compare the popularity (热度) of each topic on Stack Overflow. For this purpose, please:

- Choose at least 10 different topics covered in our course.
- Design at least 3 different metrics (e.g., # of average views of the threads tagged with the topic) to measure the popularity of a topic.
- Use proper visualizations to illustrate the popularity of these topics.

If you design the metrics and visualization well, it should be easy for the users of your application to compare the popularity of different Java topics on Stack Overflow.

Bug "Popularity" (20 points)

Developers make mistakes, which result in **bugs** in the code. Bugs manifest themselves as **errors** or **exceptions**, which can be roughly classified as:

- Syntax errors: errors detected by the compiler. Code with syntax errors cannot be successfully compiled.
- Fatal errors: errors like `OutOfMemoryError` that cannot be recovered at runtime.
- Exceptions: checked exceptions and runtime exceptions that can be handled programmatically by developers.

We are curious about the "popularity" of different errors and exceptions. Knowing this information help us understand the common mistakes made by Java developers. For this purpose, please:

- Design metrics and data analysis approaches to measure the "popularity" of errors and exceptions on Stack Overflow (e.g., how frequently they are being asked.). Note that, tags are high-level information; tags only may not include low-level errors or exceptions. You need to further analyze thread content (e.g., question text and answer text) to identify error or exception related information, probably using advanced techniques such as regular expression matching.
- Use proper visualizations to compare the popularity of different errors and exceptions. The comparison should be performed within category (e.g., compare different runtime exceptions) and between categories (e.g., compare syntax errors and fatal errors).

Related Topics (20 points)

Our application provides a query feature. Users can input any word (e.g., "spring") or phrase (e.g., "lambda expression") , and our application should display related topics on Stack Overflow.

- Topics can also be displayed as words or phrases.
- Design metrics to measure the "intimacy" of topic relations.
- Use proper visualization to illustrate all the related topics and their "intimacy" with the given input.

For example, if users input "multithreading", "Runnable" and "Thread pool" might be intimate topics on Stack Overflow while "lambda expression" might also be related but with much less intimacy.

RESTful Service (20 points)

Your application should also provide a *REST service* that answers the above three questions, so that users may use RESTful APIs to GET the answers they want. The required REST services include:

- Topic popularity: users could query for the popularity of a specific topic. Users could also query for the top N topics sorted by popularity.
- Bug popularity: users could query for the popularity of a specific error or exception. Users could also query for the top N errors or exceptions sorted by popularity.
- Related topics: users could query for the related topics of a specific input, sorted by their intimacy.

Responses of the REST requests should be in `json` format.

Logging (10 points)

Logging (日志) facilitate software servicing and maintenance at customer sites by producing log reports suitable for analysis by end users, system administrators, field service engineers, and software development teams.

Please implement a logging feature for your application, which:

- Log normal user activities.
- Log abnormal user activities or abnormal server states.

You can use any existing API, framework, or 3rd-party library for logging. However, `System.out.println("......")` DOES NOT count.

Notes

Web Framework

You should only use `Spring Boot` as the web framework.

Frontend

Frontend functionalities, such as data visualization and interactive controls, could be implemented in any programming language (e.g., JavaScript, Java, JSP, HTML, CSS, etc.) with any 3rd-party libraries or framework.

Dynamic generation of data analysis results

Data analysis results should be **dynamically generated** by the server everytime clients send a request. You **SHOULD NOT** precompute the results and stored it as a static content then simply display the precomputed static content on the frontend. 20 points will be deducted if you do so.

Visualization

You should use proper visualizations that best convey the idea, i.e., users can get the information they want instantly by looking at the visualization.

Take a look at the data visualization catalogue for inspirations.