[CS209A-24Fall] 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 1000 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., 1000 threads tagged `java`) and then answer the following questions using this subset.

Part I: Data Analysis (70 points)

For each question from this part, you should:

- Figure out which data is needed to answer the question
- Design and implement the data analysis on the backend
- Visualize the results on the frontend using proper charts.

In other words, when interacting with your web application from the browser, users could select interested analysis, which sends requests to the server; the server performs corresponding data analysis and returns the results back to the frontend, which visualizes the results on the webpages.

Your work will be evaluated by:

- whether the data analysis is meaningful and relevant, i.e., it can indeed answer the question with proper analysis on proper data. There could be multiple ways to answer a question. Be creative!
- whether the visualizations effectively 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.

1. Java Topics (10 points)

We have covered various topics in this course, e.g., generics, collections, I/O, lambda, multithreading, socket, etc. It's interesting to know, what are the top N (N>1, you may choose a proper N depending on your data and your UI design, same below) topics that are most frequently asked on Stack Overflow?

2. User Engagement (15 points)

What are the top N topics that have the most engagement from users with higher reputation scores? User engagement means any user activity (e.g., edit, answer, comment, upvote, downvote, etc.) on the thread.

3. Common Mistakes (15 points)

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

- Fatal errors: errors like `OutOfMemoryError` that cannot be recovered at runtime.
- Exceptions: checked exceptions and runtime exceptions that can be handled programmatically by developers.

What are the top N errors and exceptions that are frequently discussed by Java developers?

Note that, tags are high-level information and may not include low-level errors or exceptions. Hence, for this question, you cannot only use tag information. 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.

4. Answer Quality (30 points)

We consider an answer to be "high-quality" if it is accepted or has many upvotes. It's useful to know, what factors contribute to high-quality answers?

Please investigate the following factors:

- The elapsed time between question creation and answer creation (e.g., whether the first posted answer tends to be accepted?).
- The reputation of the user that creates the answer (e.g., whether answers created by high-reputation users tend to be accepted or have more upvotes?).

In addition to these 2 factors, you should also propose another 1 factor that may contribute to the quality of answers.

For each of the 3 factors, use proper data analysis and visualizations to demonstrate whether the factor contributes to high-quality answers or not.

Part II: RESTful Service (20 points)

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

- Topic frequency: users could query for the frequency of a specific topic. Users could also query for the top N topics sorted by frequency.
- Bug frequency: users could query for the frequency of a specific error or exception. Users could also query for the top N errors or exceptions sorted by frequency.

Here, you could reuse the data analysis from Part I.

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

Requirements

Data Analysis

You should implement the data analysis by yourself, using Java features such as Collections, Lambda, and Stream.

You **CANNOT** feed the data to AI, ask AI to do the analysis, and use AI responses as your data analysis results. **You will get 0 point for the question if you do so**.

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.**

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, HTML, CSS, etc.) with any 3rd-party libraries or framework.