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

> Question Design: Yida Tao
>
> Demo & Data: Yilun Qiu, Xiaofeng Wu
>
> Evaluation: Qiujiang Chen
>
> Git & Code styles: Yunxiang Yan

Early submission: May 23, 23:55pm (Tuesday at week 15).

Final submission: May 30, 23:55pm (Tuesday at week 16).

Late submissions after the final deadline will NOT be accepted.

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 (source: wikipedia).

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 processes associated with Java programming.

Basic Requirements (60 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 the following questions.

Number of Answers (15 points)

- What percentage of questions don't have any answer?
- What is the average and maximum number of answers?
- What is the distribution of the number of answers?

Accepted Answers (15 points)

- What percentage of questions have accepted answers (one question could only have one accepted answer)?
- What is the distribution of question resolution time (i.e., the duration between the question posting time and the posting time of the accepted answer)?
- What percentage of questions have non-accepted answers (i.e., answers that are not marked as accepted) that have received more upvotes than the accepted answers?

Tags (15 points)

- Which tags frequently appear together with the `java` tag?
- Which tags or tag combinations receive the most upvotes?
- Which tags or tag combinations receive the most views?

Users (15 points)

- Many users could participate in a thread discussion. What is the distribution of such participation (i.e., the number of distinct users who post the question, answers, or comments in a thread)?
- Which are the most active users who frequently participate in thread discussions?

Your web application, opened in a browser, should be able to answer these questions **with proper visualization**. It's your job to design the web application and choose proper visualization, so that users could comfortably use your application to get the answers they want.

Requirements for Implementation (25 points)

Data Collection & Storage (10 points)

You should collect proper data from Stack Overflow to answer the above 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.

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.

Web Framework (10 points)

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

Frontend (5 points)

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.

Advanced Requirements (12 points)

Frequently discussed Java APIs (8 points)

Which Java APIs (e.g., classes, methods) are frequently discussed on Stack Overflow? To answer this question, you may need to extract code snippets from thread content (including posts, answers, and comments) and further identify class names or method names.

REST services (4 points)

You should build a *web service* that answers the above questions, so that users may use RESTful APIs to get the answers they want. The web service may include questions defined in this documentation, or you may also define new questions. Nevertheless, your web service should provide at least 3 different RESTful endpoints that answer 3 different types of questions (e.g., `GET https://your.rest.server/java/answers?status=accepted` will return the data of all accepted answers for questions tagged with `java`).

Documentation (3 points)

You should provide a written report that describes the data you collected for this project. The written report should also introduce the architecture design of your project, as well as the important classes, fields, and methods. Finally, your report should highlight the insights you obtained from the data analysis results, e.g., what topics of Java programming are asked the most, etc.

Teamwork

We encourage you to work in a team for this final project. The preferred team size is 2, while a team of 3 or a team of only 1 student is also allowed. However, teams of size 3 CANNOT be consisted of only CS students. In addition, teams of size 3 will get a 90% discount on their project scores, because the average workload for each student decreases. Teams of only 1 student will NOT get a bonus, because s/he doesn't have to make the communication efforts that are costly but crucial for a teamwork.

Please find your teammates as soon as possible, and fill in your team information in this form: 【腾讯文档】CS209A-23S-项目组队 https://docs.qq.com/sheet/DQ0FLdmN2TEhickJF?tab=BB08J2

Demo

We provide a simple demo, which could be accessed here.

Note that the data of this demo is Stack Overflow threads with `rust` tags, meaning that you CANNOT directly reuse this data.

Submission

Please submit a zip file named "StudentID-Name-Project.zip" to Sakai before the deadline. The submitted zip should include two parts:

1. The project folder, which includes all the source code and other relevant files necessary for running the project.
2. A written report (.pdf).

Presentation

Each team should present your project during the lab sessions on either May 24 (Week 15) or May 31 (Week 16). **You can only team up with someone from the same lab session**. In addition, all team members must be present for the project presentation (points will be deducted if you don't show up in the presentation).

To present at May 24 (Week 15), your team needs to submit the project before the early submission date (May 23). Teams that have submitted and presented the project at week 15 will get a 1 point bonus to the **overall** course grade.

In addition, teams that perform well at week 15 will have a chance to present the project during the lectures (Tuesday) at week 16. Such teams will get **at most 1 point** bonus to the **overall** course grade.

Evaluation

- **Functionalities**: Each team must present the project during lab sessions (see above), and we'll check whether you've accomplished the required functionalities onsite.
- **Version Control**: You should use `GitHub` to manage the code changes of your project (see lab 1 for further details of how to use `git`). You should made **at least 2 commits**. Your remote repo on GitHub **should be set to `private` before deadline, so that no one else will see your code.**
- **Coding Style**: You should pay attention to write readable and maintainable code along the way. See lab 1 for how to use `CheckStyle` for that purpose. **After the deadline**, you can set your GitHub repo to `public`, and we'll check whether any of your commits have reduced `CheckStyle` warnings according to `google_checks.xml`.