University of Nottingham
UK | CHINA | MALAYSIA

COMP2043.GRP Indivial Report

Webpages monitor robot

Shujun JIANG(ssysj1)

Other group members:
Ruizhe HUANG(scyrh1)
Yuhao MA(scyym4)
Chi MA(scycm2)
Haowei GAO(scyhg2)

Supervised by Jiawei Li

School of Computer Science

University of Nottingham

Submit date: 23/04/2023

Contents

Chapter 1

Introduction

The market competition for web page monitoring products is fierce. Previously, our stakeholder commissioned a team to create a web page monitoring product to help teachers and students track updates to school modules. This year, the stakeholder hires our team to develop a web monitoring robot specifically for academic content, as well as entertainment and current affairs information.

Compared to traditional web monitoring robot, our product pays more attention to accuracy, timeliness and quickness. The aim of our project is to make a small program that can be used on the mobile terminal to facilitate users to use at any time. After the user enters the keyword and refresh time in the monitor, the user will be notified periodically by email if new content appears.

Due to the small size of our team, each member has to carry a heavier workload. My initially assigned main team roles are as an implementer and tester. However, due to the flexibility of software developer deployment, each member of our team is responsible for many other team roles across domains. Ultimately, in the software development process, I actually participate in aspects of the team roles of monitor evaluator, resource investigator, and completer finisher.

In addition to development work, I am also responsible for updating and maintaining the team webpage on a daily basis. During our weekly formal meetings, the team takes turns acting as the host and meeting minutes-taker.

Chapter 2

Self Contribution

2.1 Self Responsibility

As stated in chapter 1, my assigned responsibilities enabled me to assume multiple roles within the team. My main involvement in software development was centered on the backend implementation of Bing Scholar and Sina's bottom-level source code, as well as the frontend development of the Academic and Sina modules. Furthermore, after completing the functionalities of each module, I conducted tests and made necessary modifications to ensure that users receive a fully functional software package. I also ensured that the User manual was written comprehensively so that users can follow the instructions to deploy and use the software.

2.2 Contribution on the Requirement

At the outset of my software development project, I was tasked with developing a news monitoring module. To fulfill this requirement, I conducted preliminary research on a variety of news websites, including BBC News, Global Times, Securities Times, CNN News, and Sina News. However, due to the unique anti-crawling mechanisms of the Global Times and CNN websites, it was impossible for me to obtain a large amount of data from these two websites. Additionally, it was difficult to obtain ordered data from BBC News, leading to the data volume being highly unstable. Securities Times contained only partial news articles related to finance, and thus was determined to be inadequate.

Therefore, I ultimately chose Sina News as the monitored website for this module.

During our communication with the stakeholders, we learned that they wished for the software to primarily focus on academic aspects with additional functionalities as secondary priorities. As a result, we decided to add a monitoring website for the Academic module. Our candidate websites included Baidu Scholar, Bing Scholar, AMS Journals, and CNKI Scholar. After considering crawling technology, popularity, and data quantity, I ultimately chose Bing Scholar to be added to this module.

Through the UI design work that I and another colleague have done for the front-end, especially for the design of three monitoring modules, more detailed requirements have been demanded for both front-end and back-end code. These requirements include the ability to toggle monitoring modules on and off, to preserve original data and display it and to allow for detailed page navigation. Our UI design efforts led to more well-defined functional requirements for each module, resulting in a more user-friendly and reasonable user experience.

2.3 Contribution on Development

My programming productivity was comparatively high. Before the winter break, I completed the development of the back-end source code for Sina using Python. During the winter break, up to the first week of school, I utilized Taro-React to complete the front-end development of Google Scholar and Sina News.

After updating the requirements for the Academic section, I quickly developed the back-end source code for monitoring Bing Scholar and updated the front-end of Google Scholar to the front-end of the current Academic module so that users can choose which monitor to use (or both).

Around the same time, other parts of the software were partially completed, and I started working on testing these features. When the front-end of the three monitoring modules was completed, I started to merge the front-end of these three modules with another student in the group. During this time, we discovered that the front-end libraries Taro-ui and OSSAui were not compatible with each other. After I unified all the UI libraries into OSSAui, and embellished the page UI somewhat. I continued to test the front-end including component testing, functional testing, rendering testing, performance testing, stress testing, etc. I also evaluated the mini-program with Wechat DevTools and got a good score.

During back-end testing, we found that the Bilibili monitor was not working properly and determined that it was caused by an update to Bilibili's anti-crawl policy. To solve this problem required redesigning the bottom-end crawl source code based on the new anti-crawl policy. Based on my experience in developing the bottom-end source code for the Bing Scholar and Sina monitors, I provided an executable bottom-end source code for deployment on the Bilibili search page, and this student merged the code he developed for deployment on the video URL detail page to become the bottom-end source code for the current Bilibili monitor.

2.4 Contribution on Testing

I am primarily responsible for front-end and partial back-end testing. Using WeChat DevTools, I conducted front-end performance testing, rendering testing, stress testing, compatibility testing, parallel testing, etc. The test results met expectations. It fully validated the functions of each component and provided conditions for integration. It verified the reliability, stability, accuracy and concurrency of the small program. These tests indicate that the small program is currently able to correctly perform all expected functions and is a product that can be deployed once server support is in place. The program has also established a sound foundation for potential demands that may be added in the future, and reserved appropriate interfaces.

In addition, I formulated Beta testing standards and organized Beta testing. Through these tests, we obtained more software usage feedback and resolved a large volume of potential issues with software usage, improving product quality.

2.5 Contribution on Artifacts

Product packaging is a relatively difficult process, as our development is for WeChat mini-programs. Most developers directly upload the product for approval on the WeChat platform instead of packaging it. Consequently, there is very little information about packaging WeChat mini-programs online. I repeated the packaging process on my computer nearly 30 times, finally identifying the necessary environment, libraries, and plugins required for our program packaging. I organized them into the package.json in the compressed package, so users only need to follow our user manual to deploy a locally identical WeChat mini-program.

Chapter 3

Reflection

3.1 Advantage and achievements summarize

The first point is that I have a clear understanding of my responsibilities and tasks in software development and communicate effectively. I often communicate and discuss with my team members to ensure that everyone is clear about their responsibilities and tasks. This good communication mechanism enables me to complete projects faster and improve work efficiency. For the second thing, I chose to use Taro-React for front-end development. Compared to other front-end development frameworks for mini-programs, Taro is an innovative development framework with the feature of unified multi-end development. In addition to being adaptable to WeChat mini-programs, it can also be used on QQ, Baidu, Alipay, H5 and other platforms. This is something that traditional WeChat mini-program development frameworks cannot achieve. The third advantage is that I have decided to use Python to develop the back-end. The Flask framework makes our back-end architecture more methodical and provides good development capabilities, leaving interfaces for us to deal with new requirements during development. Also, I have gained experience in developing software similar to web crawlers during my previous internships and am well-versed in writing low-level source code. Additionally, I have participated in a product symposium hosted by a company where their showcased product shared similarities with the software we were developing. This experience has facilitated my understanding of our work. In addition, I have good learning and resource searching abilities. As mentioned above, using the Taro framework for front-end development is an innovative behavior, which also means that there are few resources about Taro on the Internet and even many incorrect materials. However, with my good learning ability, I quickly learned the basic syntax of Taro and could write almost all the functions we needed in development. With my excellent resource searching ability, when I encountered some difficulties that I could not solve, I went to GitHub to find some other people's projects and learned some writing methods that were not in the official documentation from their code. In addition, I also found some excellent third-party libraries through this method, which made some of our functions more convenient.

For the achievement, first I obtained a suitable software developed by ourselves. Secondly, in this software development cooperation, I have a deeper understanding of the normative process of software development and a deeper understanding of software engineering. Third, in this development, I learned more development-related technologies and had a higher proficiency. Fourth, I improved my professional knowledge in software engineering, learned how to effectively communicate with team members and stakeholder, and also had a deeper understanding of the various requirements required for the development of WeChat mini-programs. In terms of technology, because I participated in most of the development work, I have involved in and learned all the key technologies of this software, such as how to make the software can be used by multiple users, how to make the software have the ability to cache, and how to make the software have the ability to run multi-threads and multi-monitors at the same time. Now I can skillfully use Python to develop the crawler back-end and project framework, and in the front-end aspect, especially in the field of Taro front-end development, I have my own unique insights, and also have a deeper understanding of API data transmission between the front and back ends. My other main job is testing. In this job, I learned how to set up the test project reasonably, how to plan the Beta test as a whole, how to set up the issue log and solve the problems according to the log plan, how to ignore the foreseeable errors and return to the front-end to guide the user to solve, how to add the non-negligible errors to the modification plan and solve. So that in the end, when our software is submitted, there are no abnormal conditions that will stop the normal operation of our software.

3.2 Aspects not achieved

In this project, it is a pity that we could not let the user experience the program we developed on the mobile phone, and could only show it to them on the computer. If we want to make this small program available on mobile phones, we need to submit the software to WeChat for review, and then rent a server to deploy our software on a server that can access the Chinese Internet and foreign Internet at the same time. Since one of the modules we monitor is Google Scholar, which can only be accessed by foreign networks, and Sina News, which denies access to overseas IP addresses, the server provider we asked cannot provide such services.

On the other hand, we originally planned to directly jump to the target information when users clicked on the searched items when using our software, but we ended up copying the URL and letting users search on the browser by themselves. It is also a pity that we need to obtain the permission of our monitoring website and submit it to WeChat to realize this function. We eventually abandoned the idea of a direct jump.

The real-time performance of our software is not strong enough. Due to the lack of experience in developing monitoring software, we focus on developing a focused web crawler and parallel crawler instead of combining the advantages of incremental crawler when writing crawler, which also makes the new data not directly rendered on the page. It requires the user to click the refresh button once for it to render.

3.3 Unexpected accident

There were three situations that exceeded my expectations. First, when monitoring Google Scholar, only my computer would be blocked when running this function, so no data could be obtained. We made changes to the bottom code and added some randomness to better simulate human operations to solve this problem, but there was still a very small probability of being temporarily blocked. Second, when monitoring Bing Scholar, there are six computers in our team, and only three of them can run the source code. The common point is that these three computers are all Lenovo brand, so we wonder whether Lenovo information was written in the simulation table header, and this problem was solved after re-matching the Cookie. The third point is that about a week before we submitted the software, Bilibili's anti-crawling strategy was updated, which made our original monitoring script unable to continue to be used. I wrote a new monitoring script and merged it with the deep crawling script written by another classmate to solve this problem. The last problem comes from packaging. As I wrote in 2.5, it is not a common operation to package WeChat mini-program. The mutual exclusion of some libraries increases the difficulty of packaging.

3.4 Project could do better

If the project is restarted, first, I will choose the website again. In my opinion, Bing scholar is not a comprehensive choice for academic monitor's website selection. This part of the research should not be as small as the sample size presented in our final report and not representative and universal. Second, I will consider the development of the bottom crawler more comprehensively, combining the advantages of more types of crawlers. Third, for front-end development, I can better implement OOP programming, so that more code can be abstracted into components to reduce code repetition rate and improve utilization. Fourth, in terms of the choice of database, I still want to change to MySQL or SQL server instead of continuing to use SQLite, because if our product is deployed on the server, MySQL or SQL server will be more convenient and secure, which is not guaranteed by the lightweight database SQLite. Also, maybe I could improve the aesthetics of the front-end UI.

3.5 Future improvement

If the development continues in the future, we can make changes to the product according to 3.4, such as changing the database, refactoring the code, etc. We can also add more monitoring objects to the software to provide users with customized monitoring crawlers. In addition, we also need to create an administrator system to maintain the security of the product and database regular testing, etc.

Appendix A

Peer Assessment