**Summary and reflections**

In this chapter, we will introduce how the project was managed by specific tools and methodologies in Section 1. Reflective remarks will also be demonstrated in Section 2 of this chapter.

1 risk management

**1 Project management**

This section introduces three main supporting tools for managing the project and how the team distributes jobs and utilises agile methodology.

**1.1 Supporting Tools**

The team utilised some useful tools to manage the team and project. Section 1.1.1 will introduce the usage of Git. In section 1.12, a documentation tool is explained. Section 1.1.3 shows how the team is managed with the help of GitHub.

**1.1.1 Version Control – Git**

Git is a version control tool which can trace any changes in code. Each commit will record all the code status at a specific time node. The utilisation of Git allows the team to easily spot positions of bugs and notice what has affected the software. With commit message, it also makes communication more efficient. Git provides a convenient feature branch as well. Branch simplifies team collaboration. Each time a member needs to develop a new feature, a new branch would be of great help to ensure the new feature will not affect the main software.

**1.1.2 Documentation – Style Guidist**

The team takes JavaScript as programming language. However, we noticed that JavaDoc is powerful tool for demonstrating usage of code with proper documentation. For further maintenance, the team uses Style Guidist as the documentation tool. With an extra Markdown file in each folder, Style Guidist will generate a JavaScriptDoc automatically in the form of web. It would help maintenance team to understand and help anyone who are interested in our design.

**1.1.3 Teamwork Organisation and Remote Repository – GitHub**

GitHub works not only as a remote repository but also a teamwork organization tool. To share the code among all the members, GitHub stores all the branches and records on the cloud as a remote repository. CI is also applied by a feature called Action on GitHub. Action will automatically run all the tests and build the project each time when GitHub receives a push operation to automate testing and prevent potential long-term errors. GitHub also provides issues, milestones and Kanban. The team assigns tasks and rises questions and bugs by issues. Kanban will automatically take issues and assign them into TODO, In Progress and Done columns. Milestone will display each sprint’s tasks and encourage members to complete tasks soon. These three features help a lot in task assignment and time management. All the members could have a clear view on current state.

**1.2 Task and Responsibility Distribution**

Design and develop a software from scratch not only need the ability of programming. The skills of UI design, structure design and testing are also required. The team explored each member’s unique skills and strength, which allows the team to take advantages of every member. E.g., people who are creative designs the software structure first. Experienced UI designer is responsible for designing UI. Members who are familiar with programming will implement designed features. After design was confirmed, the team of design will also join the quality assurance team. Clear responsibilities make members focused and the whole team become efficient. Labor division is as follows.

Team Leader: Shiliang Chen

UI Design Team: Yiming Tang, Yani Huang

Technical Team: Shiliang Chen, Yijie Lu, Yuting Jiang, Ruizi Han

Quality Assurance Team: Yiming Tang, Yani Huang

Report Editor: Yuting Jiang, Yijie Lu

Meetings are conducted twice a week, formally and informally. Chairperson and secretary are changed in turn. Tasks are distributed after meeting and minutes are in appendix X.

**1.3 Agile Methodology with Requirements Documentation**

The team decided to apply agile methodology for the development of software. However, at the beginning of the project, the team has made a detailed requirements documentation to clarify all the basic requirements from stakeholders. The requirements documentation also functions as a checklist with our supervisor.

During the development, the team wrote user stories to specify features needed to be built. With those user stories, the team discussed features to achieve in each week’s informal meeting and develop the features during the week’s sprint. In the formal meeting of the week, we confirmed that week’s achievements with supervisor. In this case, the level of customer engagement was high, and any unsatisfied things could be fixed very soon.

The team also utilises stand-up, which is short meeting of about 5 minutes held every day except those days with meetings. Stand-up allows everyone report daily progress and problems encountered, which helps the team clear tasks in time and handle issues and problems in time.

Peer programming was applied strictly during the spring festival. Each task was assigned to two of the technical team members. One would be responsible for coding and the another will take care of monitoring and checking whether there was anything wrong with commenting, naming, etc. Peer programming improves code quality considerably and increases the efficiency by exchanging ideas between peers.

**2 Reflection**

The team learned a lot of experience from the project. This section will share some of the reflective remarks from this project and future expectations on it.

**2.1 Requirements Engineering**

To build a software from scratch, requirements engineering is the most fundamental thing but also most important. The team finds requirements engineering is a relatively successful part of the project. Team 10 learned how to perform survey, focus group and interview through this process. Survey gave us an overall user preference to the software, while the focus group allows users to share more specific ideas about the software. Survey and focus group helped the team decide the platform and style of the software and come up with practical and innovational features. Focus group even aspired us to design a tutorial section for newcomers of programming. The interview collected opinions from a lecturer and helped us spread the survey widely to conclude more general ideas.

As for the weakness of requirements engineering, the questionnaire contains some questions which may not be that useful in the software design such as “Through what you learn algorithms?” Since overall 207 students has involved in the survey, more valuable questions may help more.

**2.2 Technical Research**

Technical research was not conducted smoothly. It was assigned to be a week’s research on programming languages and tools we can use. However, because the task was made vaguely, i.e., everyone go to do technical research, it turned out that no one had done anything useful related to technical research. After that, the team discussed about aspects of technical research and distributed work to different people with different aspects. Technical research was done and enough for interim report at the end, however, it was far from enough to help us decide programming language at that time. The team has worked on complete the part as asked to do it, but not in order to actually choose some technical things properly. This was a big mistake and caused more waste of time before actual programming, since all of us found it hard to do coding with the tools decided in interim report.

As a reflective remark, we would do a comprehensive and focused technical research to choose a most suitable programming language and tools for a project.

**2.3 Software Engineering Methodology**

For team members who are not experienced in software engineering, it is not easy to decide which software engineering method to use, traditional or agile. Firstly, we reviewed the related lectures of the Software Engineering module and listed the advantages and dis-advantages of both methods. Also, we referred to the advice of the supervisor. Finally, we decided to combine two methods to develop this software. The traditional development method was used to detailly record the requirement documents, while the agile development method was used in the design and development stages. Agile helped improve customer engagement. We were able to contact and confirm any details with our supervisor in time. According to the advice and suggestions given by supervisor after each sprint, the software was accumulatively being stronger and equipped with more features.

The software we built is components based and is relatively suitable for agile method. We could always see the progress of the software, which made the team more energetic and willing to develop. As the method was agile, even requirements change could be handled well.

**2.4 Time Management Issues**

**Meeting Time Management**

**The supervisor emphasized that each formal meeting must be controlled within 30 min- utes. However, sometimes we may have lots of content expected to discuss with the supervisor. That sometimes led to the meeting longer than 30 minutes. To solve this problem, one member pays attention to the time in each formal meeting. The member will remind everyone if time is exceeded. Further, we carefully choose more important content to discuss in formal meetings.**

**Late of Starting coding**

Compared to other groups, our overall progress seems a bit slow. The reason may be that we need to do more work to collect user requirements. Some members have raised concerns about whether we could finish the project on time. Then, we showed module convenor our time plan and asked him if it was a reasonable arrangement. The result is that he reminded us to focus on ourselves because it is meaningless to compare with different groups. We should follow our time plan. 但是最后证明 时间的安排是没有问题的

**2.5 Team Management Issues**

**Late of Finishing Tasks**

私聊等措施得知原因

**Meeting and Task assignment**

Different members are responsible for different work in this project. However, it is difficult for members with cooperation to keep track of progress of each other. To solve that problem, we decided to use the Kanban function in GitHub. At the same time, we assign issues in GitHub to ensure that each member know about tasks.

**Stand-up in Spring Festival**

预防性严格执行 加速了进城和理解

**Disagreement**

A severe disagreement occurred during the prototype design stage. Different members had different opinions about the process of learning sorting algorithms. This kind of situation is not a surprise. We have predicted it in the risk management section. To deal with this problem, all members must keep calm. Then, we decide to finish the sequence diagram together, and the disagreement above solved finally.

2.6 Technical Issues

**Way of Peer Programming**

Live share

**Testing**

严格执行 得到了好处

**2.7 Lexical Issues**

The supervisor pointed out that there were several grammar errors in documentation files. Because all members of team 10 are Chinese, there indeed exists difficulties in writing formal reports in English. To deal with this problem, we decided to double-check all documentation and pay more attention to vocabulary usage and grammar errors when writing reports.

**2.8 Future Work**