**Summary and Reflection**

This chapter will introduce how the project was managed. Section 1 will include a discussion on specific tools and methodologies the team used. Reflective remarks will also be demonstrated in Section 2 of this chapter.

**1 Project management**

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

**1.1 Supporting Tools**

The team utilised some practical tools to manage the team and the project. The usage of Git in this project will be included in Section 1.1.1. In section 1.1.2, a documentation tool is introduced. 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 that can trace code changes. The utilisation of Git allows the team to spot positions of bugs easily and notice what change has affected the software. Commit messages also makes communication more efficient. Git provides branch to simplify team collaboration. Every time a member needs to develop a new feature, creating a new branch would help ensure the new feature will not affect the main software code.

**1.1.2 Documentation – Style Guidist**

The team takes JavaScript as the programming language. Noticing that JavaDoc is a powerful tool for demonstrating the usage of code with proper documentation, the team decided to adopt a similar tool. Style Guidist is used as the documentation tool for further maintenance. With an extra Markdown file in each folder, Style Guidist will automatically generate a JavaScriptDoc in the web. This JavaScriptDoc would help the future maintenance team or anyone who is interested in our design to understand the software code.

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

GitHub works not only as a remote repository but also a teamwork organisation tool. CI is applied by a feature called Action on GitHub. Action will automatically run all the tests and build the project after GitHub receives a push operation. It helps automate testing and prevent potential long-term errors.

The team assigns tasks and raises questions and bugs by issues. Kanban will automatically take issues and assign them into corresponding columns. Milestones will display each sprint’s tasks and encourage members to complete tasks soon. All members could have a clear view of the current state.

**1.2 Task and Responsibility Distribution**

Designing and developing software from scratch not only need the ability of programming, the skills in User Interface (UI) design, structure design and testing are also required. The team explored each member’s unique skills and strength, which allows the team to make the most of members’ ability. For example, people who are creative are assigned to design the software structure first. Experienced UI designers are responsible for designing UI. Members who are familiar with programming will implement designed features. After the design was confirmed, the team of designers will then join the quality assurance team. Clear responsibilities make members focused on their work, and the whole team become efficient.

**1.3 Agile Methodology with Requirements Documentation**

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

During the development, the team wrote user stories to specify features needed to be built. Based on 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 every week’s formal meeting, we confirmed the week’s achievements with the supervisor. In this case, the level of customer engagement was high, and any things that go to a wrong direction could be fixed quickly.

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

**2 Reflection**

The team gained valuable experience from the project. This section will share some of the reflective remarks and future expectations.

**2.1 Requirements Engineering – strong & weak**

In order to build software from scratch, requirements engineering is considered as the most fundamental and important thing. The team found requirements engineering was a relatively prosperous part of the project. Team 10 learned how to perform a survey, focus group and interview through this process. Survey gave us an overall user preference for 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 software's platform and style and come up with practical and innovational features. The focus group even aspired us to design a tutorial section for newcomers to programming. The interview collected opinions from a lecturer who teaches algorithm courses, and he helped us spread the survey to many our stakeholders so the survey result became more convincing.

As for the the team’s shortcoming in 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 207 students have been involved in the survey, more focused questions may help more.

**2.2 Technical Research – weak but learned a lesson**

Technical research was not conducted smoothly. It was assigned to be a week's research task on programming languages and tools we can use. However, because the task was declared vaguely, it turned out that no one had done any useful technical research. After that, the team discussed aspects of technical research and distributed work to different people. Technical research was partly done and enough for the interim report. Nevertheless, it was far from enough to help us decide the programming language at that time. The team has worked on completing the part as asked to do, but did not implement the process in order and choose technical language and tools properly. This mistake caused a waste of time before actual programming, since all of us found it is hard to do coding with the tools we decided in the interim report.

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

**2.3 Software Engineering Methodology - strong**

It is not easy for team members who are not experienced in software engineering to decide which software engineering method to use, traditional or agile. Firstly, we reviewed the related lectures of the Software Engineering module and listed both methods' advantages and disadvantages. 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 on time. According to the supervisor's advice and suggestions after each sprint, the software was accumulatively being more robust and equipped with more features.

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

**2.4 Time Management Issues**

**Late of Starting Coding – weak but learned a lesson**

Because we did technical research again after interim report for finding a more suitable programming language, coding was postponed. The team also spent another three weeks learning React and Electron. It turned out that software development finished later than planned, but the quality of software and overall time were guaranteed by rescheduling time plan. This made the process more flexible and agile.

**2.5 Team Management Issues**

**Late of Finishing Tasks - weak but learned a lesson**

Some of the team members could not finish their tasks on time. This may potentially affect the progress of the whole team because tasks have dependencies. The first attempt is to separate tasks to reduce coupling and assign individual tasks to those who may take more time. This attempt was not an ultimate solution and still caused some issues since the situation that the whole team was waiting for one task to be finished is still possible to exist. The second attempt was trying to treat the root. The team leader had conversations on the issue and tried to find out the reason behind it. This attempt worked better and encouraged the team member to take responsibility. Hence, it is significant for a team to have communications. Spotting issues and addressing them immediately is vital for further work.

**Stand-up and Task Assignment - strong**

As we applied agile methodology, a stand-up meeting was conducted every day except a break day of week, informal and formal meetings. Daily stand-up pushed us to keep pace with the overall progress and finish tasks on time. This also allows every member to have a global view of the project. During the Spring Festival, stand-up functions well in keeping everyone learning React and related knowledge. As for task assignment, different members are responsible for different work in this project. However, it is difficult for members with cooperation to keep track of the progress of each other. To solve that problem, we decided to use Kanban in GitHub. At the same time, we assign issues in GitHub to ensure that every member knows their tasks.

**2.6 Technical Issues**

**Way of Peer Programming - strong**

We learned that peer programming had several merits. However, during the break, it was hard to program together. We successfully found an extension called Live Share in Visual Studio Code, which allow us to edit on the same workplace and even share the terminal. Since the project is web-based, a browser sharing tool was also used to view real-time changes. By performing peer programming, each task was assigned to two of the technical team members. One would be responsible for the coding, and the other will monitor the process and check whether there was anything wrong with commenting, naming, etc. Peer programming improves code quality considerably and increases efficiency since members can exchange ideas between each other.

**Testing – weak to strong**

The team realised the importance of testing and spent a week learning how to perform testing on React JavaScript project. At the early stage, Test Driven Development (TDD) was not taken seriously by some of the team members. After TDD was stressed to be vital, the group added unit tests for the component. Compared to human eyes and testing manually, automated unit testing helps design the code and prevents potential mistakes by checking components each time they are modified.

**2.8 Future Work**

**Language Support**

Team 10 cares about people who speak different languages. According to the survey, most participants claim that they need a Chinese version to assist them in understanding the content. Besides, it is also noticeable that those widely used similar software only support English. We have considered it as one of the additional requirements. However, translation needs to be conducted with professional assistant and references. It is hard to find such help and reference, so we have not done the Chinese version at the end. We hope to develop an upgrade version of iCanSort which also provides Chinese to help more students who are struggling with learning sorting algorithms.

**Accessibility Support**

Team 10 noticed that web contents have a special attribute called aria with which browsers can read out content to help people who cannot see the content. For future work, we would like to develop an idea to help people with disability in visions.

**Multiple Platform Support**

As a web-based project, it is highly possible to be deployed in a web server and allow people to use this web app by accessing a website. Also, it could be compatible with smartphones, which needs further work.

**3 Risk Management**



**4 time plan**