**Summary and Reflection**

Six members in Team 10 have worked together on this project for around six months. We have gained valuable experience and learned a lot during the process.

This chapter will provide a summary of the group project in Section 1. Members’ reflection on both project management and technical perspectives will be presented in Section 2. Team 10 also developed risk management before the project started to help solve some possible problems effectively.

**1.Summary**

This section summarises the final product and experience of the group project in Section 1.1 and Section 1.2. Possible future work is demonstrated in Section 1.3.

1.1 Product Summary

The final product is a desktop application named iCanSort, which can be installed on both PC and Mac to help users learn sorting algorithms and their correctness. The software is currently released on the team’s GitHub release page and can be downloaded freely. The product is also open source under MIT license.

The team finishes all the core features from the requirement, so the software is complete and ready to use. Team 10 is proud of the software’s step-by-step learning structure, by which users who have little knowledge to plentiful knowledge can all learn sorting algorithms easily. The user interface is also carefully designed in order to provide a simple, beautiful and user-friendly experience during the learning process.

1.2 Experience of the Group Project

Building this software from scratch is quite a challenge. However, the team have successfully done both the software engineering and project presentation. We have experienced requirement engineering with our stakeholders and gradually built a clearer view of the software. The interim report was successfully finished, and the team learned LaTeX for the first time. During the Spring Festival, the team kept working in the form of peer programming, learning new knowledge and building the software effectively.

Agile made the process more flexible, and ten sprints in total were conducted during the software development. Due to the high level of customer engagement, we also learned to design the software from the customer’s standpoint. We have a specialised quality assurance team to monitor the coding process. The team has realised the importance of testing by this project as well. Conflicts could not be avoided, but the team learned the importance of communication and worked more efficiently. A group project is not a work of one. How to make the best of everyone and manage the relationship appropriately are another two lessons the team learned.

1.3 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 the help of professional assistant and references, which is hard for us to find. In that case, 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 it. 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, the software is possible to be deployed on a web server. In that case, people can use this web app by accessing a website. Also, it could be compatible with smartphones, which needs developers’ further work.

**2.Reflection**

This section will share some of the reflective remarks learned from this project in aspects of project management and technical issues.

2.1Project Management

2.1.1Software Engineering

**Requirement Engineering**

For building software from scratch, requirements engineering is considered the most fundamental and significant part. Team 10 learned how to perform a survey, focus group and interview throughout the requirements engineering process. Survey gave us a general view of users’ preference for the software, while the focus group allows users to share their specific ideas about the software with us. Those results helped the team decide the software’s platform and interface style. They also inspired us to come up with some practical and innovative features, such as a tutorial section for newcomers to programming. The interview collected opinions from a lecturer who teaches algorithm courses, and he helped spread the survey to many of our stakeholders. In this case, the survey result became more convincing.

As for the team’s shortcoming in requirements engineering, the questionnaire contains some questions which are not helpful in the software design, such as “Through what you learn algorithms?”. Since 207 students have been involved in the survey, questions should be more precise.

**Insufficient Technical Research**

Technical research was not conducted smoothly. At first, the task that finding suitable programming languages and tools for the project was declared vaguely, and it was assigned to every team member. Then 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 members. Technical research was partly done at that time, and the research result was declared in interim report. Nevertheless, as we start the implementation phase, we found it is hard to do coding with the language and tools we decided in the interim report. This mistake forced us to spend extra time finding a more suitable programming language, so coding was postponed. Then the team spent another three weeks learning React and Electron. It turned out that software development was finished later than planned.

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

**Suitable Software Engineering Methodology**

It is not easy for team members who have no related experience in software engineering to decide the software engineering method, traditional one or agile. Hence, we reviewed the related lectures of the Software Engineering module and 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.

The software we built is components based, and it 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 we adopted agile method, even requirements changes could be handled well.

**Sprints and Customer Engagement**

With agile method, we implemented the software engineering process sprint by sprint. A sprint began with each week’s informal meeting, where the team discussed features to achieve based on the detailed requirements documentation. After that, we developed the features during the week’s sprint. In the formal meeting, we confirmed the week’s achievements and design details with the supervisor. Then we would do some modification to the software according to the supervisor’s advice and suggestions, which is the last part of that week’s sprint. Every sprint went through this process. In this case, customer engagement level was high, and any things that go to the wrong direction could be fixed quickly.

电脑屏幕的截图

描述已自动生成

2.1.2Team management issues

**Job Allocation**

Designing and developing software from scratch does 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 and tried to make the most of members’ ability. For example, creative people 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 increase the whole team’s work efficiency.

**Poor Time Management**

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 to finish their tasks. This attempt was not an ultimate solution because the situation that the whole team was waiting for one task to be finished is still possible to exist. The second attempt is to treat the root. Team leader had conversations on the issue with members 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 in a short time is vital for future cooperation.

**Stand-Up Meeting**

Stand-up meeting is a significant part in implementing agile method. As we applied agile methodology, a stand-up meeting which is a short meeting of about 5 minutes was conducted almost every day. Members were asked to report their daily progress and problems encountered, which helps the team finish tasks in time and solve those problems. This also allows each member to have a global view of the project and keep pace with the overall progress. During the Spring Festival, stand-up functions well in keeping everyone learning React and related knowledge.

**Task Assignment – GitHub**

In our team, tasks are assigned to both individuals and a group of members. It is difficult for members to remember all the tasks and keep track of others working progress. To solve that problem, we decided to use Kanban in GitHub. Tasks are assigned as issues in GitHub to ensure that every member knows their tasks. Members also raise questions and report 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 before the deadline. With Kanban, all members could have a clear view of the current state.

**图形用户界面, 应用程序, Teams

描述已自动生成**

2.2Technical

**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.

**Continuous Integration (CI)**

GitHub works as not only a remote repository but also a useful CI 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.

\begin{figure}[H]

\centering

\includegraphics[width=.6\textwidth]{images/Test/test-2-integrateTest.png}

\caption{Continuous integration}

\label{fig:Continuous integration}

\end{figure}

**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 additional Markdown file in each folder, Style Guidist will automatically generate a JavaScriptDoc on the web. This JavaScriptDoc would help the future maintenance team or anyone interested in our design to understand the software code.

图形用户界面, 文本, 应用程序

描述已自动生成

**Peer Programming**

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 or naming. Peer programming improves code quality considerably and increases efficiency since members can exchange ideas between each other.

**Testing**

The team realised the importance of testing and spent a week learning how to perform testing on React JavaScript project. Test Driven Development (TDD) was not taken seriously by some of the team members at the early stage. 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.

**3.Risk Management**

**报纸上的文字

中度可信度描述已自动生成**

**4.Time Plan**