Software Engineering Group Project

Group 36: Final Team Project Deliverable

Connor Calkin, Daniel Braghis, Benjamin Lewis, Lucas Sayers, Pingding He

1. EVALUATION OF TEAMWORK

1.1 What Went Well

The team's effective utilization of agile methodologies contributed to efficient collaboration and minimized conflicts. Our team implemented multiple branches, consistent coding styles, and techniques that allowed for parallel progress on various components of the product. This approach ensured that the final product not only met all necessary requirements but also surpassed expectations. The team's dedication to their work and adherence to standards were key factors that contributed to the successful outcome of the project.

1.2 Areas of improvement

While the team demonstrated a strong focus on deliverability, there were instances where we deviated from the original plan. To improve future project outcomes, it would be to adopt a more balanced approach that prioritizes both deliverability and adherence to the plan. For instance, the initial plan for increment 2 focused on delivering a large portion of the UI. However, because of incorrect prioritisation, the UI components developed where of subpar quality to include in the increment. In hindsight, the components should have been included in the deliverable to bring value to the client.

1.3 Advantages of Agile Methodology

The utilization of Agile methodology provided several advantages throughout the software development process. The frequent communication with the client facilitated continuous feedback and allowed for timely adjustments during the development process. This iterative approach ensured that the final product met the client's expectations and minimized the risk of wasted effort. By maintaining open lines of communication with the client, the team was able to gain a deeper understanding of the client's needs and make appropriate adjustments accordingly. Frequent feedback was crucial in getting the project back on track and literally saved the project following the second increment in our team’s case.

1.4 Disadvantages of Agile Methodology

The agile methodology has a steep learning curve, especially for inexperienced and new development teams like ours. Many techniques used assume a very good understanding of one's ability to deliver the work in time and up to the client's standards. This wouldn't be a problem for experienced programmers but for many university students this is their first exposure to a robust and stable code base with rigorous testing.

Moreover, the time constraints of each sprint were challenging, as the team had to work quickly to achieve their goals. In some instances, this could lead to fatigue and burnout if not managed properly. The team can mitigate these issues by setting more realistic goals or adopting a more flexible approach to work. Effective management is necessary to ensure team members have enough time to rest and recharge.

1.5 Disadvantages of Agile Methodology

The team successfully followed the XP value of simplicity by prioritizing code that was easy to understand and maintain. The resulting code was modular and easy to test in isolation to ensure its quality throughout the development process. The team consistently sought feedback to ensure they met the requirements. However, the team can still improve in the areas of communication and confidence in their abilities. By fostering open communication and experimenting with new approaches, the team can enhance their effectiveness and productivity. This will allow them to tackle problems more effectively and efficiently.

Overall, by leveraging the advantages of Agile methodology and adhering to XP values, the team successfully delivered a product that met the client's expectations. The team can improve their approach moving forward by balancing deliverability and adherence to the plan, promoting communication and experimentation, and adequately managing the team's workload to prevent burnout. This will ensure that future projects are delivered on time and to a high standard.

2. TIME EXPENDITURE

2.1 Breakdown by team member

Time Spent (hours)				
Team Member	**Sprint 1**	**Sprint 2**	**Sprint 3**	**Total**
Connor Calkin:	**9**	**12**	**10**	**31**
Daniel Braghis	**11**	**22**	**21**	**54**
Pingding He	**17**	**8**	**8**	**33**
Benjamin Lewis	**18**	**8**	**13**	**39**
Lucas Sayers	**8**	**20**	**12**	**40**

2.2 Most time expensive activities

The tasks that took the longest to complete were done more on the back-end side of the development. Implementing SQL into the system and attempting to make the system run as fast as possible to ensure that the user would have the best outcome possible. It was also the creation of graphs that took a long time. We had one implementation, however, on larger data sets, it would be far too slow and would therefore not add much value to the stakeholders, it, therefore, took a long time to attempt to optimise this task and develop it using better SQL querying.

2.3 Time estimation

The utilization of Fibonacci numbers for task sizing proved valuable in our project. By leveraging their exponential nature, we maintained a realistic perspective on the sizes of larger tasks, preventing overly optimistic estimations. This approach also facilitated better teamwork by reducing disagreements and ensuring a shared understanding of task sizes among team members.

To further enhance agreement and inclusivity, we employed the agile planning poker technique. This involved the whole team in the estimation process, valuing each team member's input equally. By incorporating diverse perspectives, we achieved more accurate estimations and fostered a collaborative environment.

Over time, our estimation accuracy improved. Initially, we encountered challenges as tasks often took longer than anticipated. However, as we gained experience, our estimations became more accurate, allowing for better workload allocation and ultimately delivering a comprehensive project. For example, the creation of graphs, initially estimated as a medium-sized task (5), turned out to be significantly larger. Nonetheless, our estimations became more aligned with the actual task sizes as the project progressed.

In conclusion, the use of Fibonacci numbers and agile planning poker greatly benefited our project. They provided us with more accurate task division and a clearer understanding of the workload, enabling us to meet each deliverable's deadline effectively.

2.4 Division of workload

We divided the workload using a first-come, first-serve system where team members chose tasks they were comfortable with. This approach promoted efficiency and reduced the chances of individuals struggling with their assigned tasks. Effective communication was crucial in coordinating our efforts, ensuring everyone knew what others were working on. As we gained more experience, communication improved, leading to a smoother workflow.

In hindsight, it may have been more efficient to define tasks earlier to ensure a balanced workload. Assigning tasks to pairs or practicing pair programming could have further optimized our workflow by leveraging individual strengths and enhancing communication within smaller groups.

Overall, our first-come, first-serve approach allowed for task allocation based on individual preferences, supporting an efficient process. Improved communication and earlier task delineation could have enhanced efficiency and balance within the team.

3. TOOLS AND COMMUNICATION

3.1 Software Engineering Tools

We used IntelliJ IDEA as our integrated development environment (IDE) for its robust code editing, debugging, and refactoring features. It integrated seamlessly with Maven, our chosen build tool, which provided standardized building and automated processes, saving development time, and reducing version conflicts.

For version control, we relied on GitHub, allowing us to work on independent branches, conduct code reviews through pull requests, and easily track and revert to historical versions.

To create use case and UML class diagrams, we utilized Lucidchart for its intuitive interface, templates, and multi-user collaboration. Lucidchart's versioning feature ensured easy tracking of diagram changes and provided the ability to revert to previous versions.

Overall, our tool choices, including IntelliJ IDEA, Maven, GitHub, and Lucidchart, supported our software development project by enhancing development efficiency, enabling version control, facilitating collaboration, and promoting effective design documentation.

3.2 Collaboration Tools

We adopted Trello as our project schedule management tool. On Trello, tasks are divided into three categories: not started, in progress, and completed, to clearly classify the work progress. This helped us assign tasks and track project progress.

We actively utilized Google Documents as a document collaboration tool. It can standardize the font, text size, line spacing and other document formatiing. Additionally, its support for live editing and commenting on documents made our collaboration more efficient and improved the flow of the development process.

We used Teams as a tool for remote meetings. Teams can share files and send messages. At the same time, it allows us to share screens and collaborate via online meeting as a group would face-to-face. Teams allows us to communicate and collaborate without being limited by time and place.

3.3 Communication Tools

For instant messaging, we use WhatsApp. On WhatsApp, all information is transmitted point-to-point, which ensures the security of our information and excludes the possibility that development information will be leaked by the communication software during the development process. Moreover, WhatsApp message reminders are worth mentioning. Even if the user is offline, that user can still receive the notification immediately about someone sending him a message. This ensures that we can contact the corresponding developers conveniently and in a timely manner during the development process, which is great helpful to our development progress.

3.4 Meetings Strategy

Our team held three to four 12 to 15 minute Scrum meetings per week, conducted one 1-hour work analysis meeting, and met with our project advisor for 40 minutes each week. Additionally, after delivering each stage of the product, we had 30-minute review meetings with the customer.

During the Scrum meetings, team members reported on completed and upcoming work, analyzed the burndown chart, and resolved any confusion or challenges encountered during the development process. The work analysis meetings focused on setting goals for each deliverable, designing class diagrams, and ensuring the smooth progress of the project. In the advisor meetings, we presented our work to receive feedback and suggestions from the advisor, allowing us to iterate and align the project with the customer's needs. The review meetings with the customer provided valuable feedback to ensure that the delivered products consistently met their requirements.

3.5 Effectiveness Evaluation

Maven and the IntelliJ IDEA integrated development environment (IDE) played crucial roles in our development process. They were indispensable tools that significantly contributed to project integration and efficient coding. While GitHub and Lucidchart also played important roles in project development and design, they were of slightly lower priority compared to Maven and the IDE. Although these tools provided valuable support, we could still carry out the project successfully without them.

For collaboration and communication, we found Microsoft Teams to be irreplaceable. Teams offered exceptional features for remote meetings, screen sharing, and file sharing. It also provided support for offline messaging and online editing, meeting our collaboration and communication needs effectively. By leveraging Teams, we were able to coordinate and address complex tasks in an organized manner. Additionally, the other software tools we utilized further enhanced our development efficiency.

In conclusion, the tools we adopted played vital roles in our development, collaboration, and communication processes. They facilitated effective coordination of tasks and ensured accurate iterations based on customer needs. Ultimately, we successfully delivered a product that met our customers' expectations within the scheduled timeframe.

4. ADVICE TO NEXT YEAR'S STUDENTS

4.1 Agile Methodologies

As you only have a few lectures for the module, it's vital you attend them all. The theory you learn about in those few lectures, you need to apply as part of the project. You will be assessed on how you follow the agile methodologies, so you need to understand them both theoretically and practically. During every step of development, you should be thinking "Am I meeting agile development methods?". A key part of this is making sure that during each sprint, you deliver value to the customer. This includes balancing front-end and back-end development to make sure you can provide functionality that works well, and the customer can see.

4.2 Dividing Responsibilities

When meeting as a group for the first time, it's a good idea to get to know each other, especially around your goals for the project and your skills. It's vital to be honest and to let people know your weaknesses just as much as your strengths. As it is a full software development project, there should be roles accessible to all skill sets - even some roles with limited programming or development responsibilities. By discussing this early on, you can build confidence as a group. However, equally you are doing this project to learn new skills, so although you may be weak at something, it's important during at least one sprint you practice and develop these weaker skills.

4.3 Initial Team Meeting

One of the most important parts of your teamwork is the first time you meet. This is when you set out your communication strategies and tools you plan to use. If you start this incorrectly, it can be difficult to recover as you will have started using the other tools and won't want to move to new ones.

In terms of communication tools reflect on things like, does everyone in the team have it, does it allow for file sharing, is it permanent and what devices can it be used on? Our group chose to use WhatsApp which worked well for most of these questions, but as it's traditionally meant for phones it was difficult to use on laptops and computers.

Remember this project is also an opportunity to try out new things and tools. You will discuss and be advised to use tools that you may have never used before such as Trello or Github. It's important not to fear using them, but to give them a go as they may be better than the tools you currently use. Equally they may not work as well as previous tools you have used - that is equally fine, do not feel you must use certain tools if you find other ones as a team work better.

4.4 Supervisor

The most valuable tool you have throughout your project is your supervisor. They have great knowledge and experience, so you should aim to get regular feedback from them throughout the project - more often than just at the end of each sprint. You must also learn from each sprint. Hopefully over each sprint you should improve as a team together.

Remember you are also assessed on professionalism - this is something you should be focusing on from the first deliverable. They are easy skills to refine and easy marks to gain, so make sure to be punctual and professional.

4.5 Overall Advice

Throughout your project you will be receiving advice from a range of sources, lecturers, supervisors, team members, other groups. They are giving you advice for a reason, so even though in the short term it may seem like extra effort, in the long term it will help you so do listen. You can always try out their advice, and then revert if it doesn't work for your team.

This project is an opportunity to make mistakes with minimal impact and lots of opportunities to improve so enjoy the project, even if it may be stressful at points. It is a fantastic experience for the real world and try to get everything you can out of it.