School of Electronics and Computer Science
Faculty of Physical Sciences and Engineering
University of Southampton

COMP2211 - Team Project Experience Evaluation

Sobczak *Konrad K (kks1g19@soton.ac.uk)*
Scrase *James M (jms1g19@soton.ac.uk)*
Umavassee *Sharan G (sgu1n19@soton.ac.uk)*
Nooli *Karthik (kn3g19@soton.ac.uk)*
Protopapas *Christos (cp2n19@soton.ac.uk)*
Tarajia *Ammar (at3g19@soton.ac.uk)*

Team **17**

14^{th} May 2021

1 Evaluation of Teamwork

1.1 Successes

Our team worked cohesively as a group, having regular meetings, and continuously updated one another through various group communications platforms such as Discord, and via other social media; for example, to help with updating one another on code written, we implemented a bot that would automatically notify the group whenever a commit was pushed to our GitLab repository.

1.2 Future Improvements

To improve our teamwork, we should have set up more regular meetings during the early stages of the project to maximise our output for early increments and facilitate later ones. Maintaining communication in between meetings to update each other about any progress is also an essential part that we took some time to implement.

1.3 Agile Methodologies

We used Agile methodologies in our development process, namely Scrums and Kanban, as well as some tenets of XP, for example pair programming, iterative design and prototyping, supervisor feedback and test-driven development amongst others.

1.3.1 Advantages

Scrum helped break down the workload into manageable chunks that could be assigned to the various members of the team during the regular scrum meetings, in which we also checked on each others' progress. Our sprint plans helped to ensure our objectives were met for each increment, and allowed us to adjust the number of team members working on certain tasks based on their difficulty, current progress, and time remaining until the next increment.

This in turn allowed us to more effectively manage our time and resources. One key example of this was the most recent sprint plan in which the deadline we assumed to be correct was moved closer by one week. To this extent we had to re-prioritise our objectives, removing a few of our largest projected features in order to ensure delivery of critical features.

Throughout the project we engaged in pair programming, which allowed us to more rapidly develop and test features, since any piece of code that was written could be concurrently reviewed by their programming partner. This concurrent reviewing allowed for the consistency of code quality to be maintained throughout the document.

After feedback from our supervisor in the meeting for Increment 1, we introduced test-driven development as a key aspect of our working. This ensured that our features functioned correctly, and ensured the integrity of the code-base allowing us to utilise continuous integration, as well as reducing the time spent debugging later in the project. In retrospect, test-driven development should have been implemented from the start, as the team still ran into problems with code written from the first increment that did not utilise this strategy.

Using Agile methodologies also allowed us to modify the design, as many of our initial features could not be implemented, or would have been very difficult to implement, due to the limitations of the software being used (JavaFX). Changing the design in Scrum only required a meeting to resolve, as opposed to developing using a linear-sequential life-cycle model, such as the Waterfall Model which would have required a complete reversion of the project back to the design phase.

1.3.2 Disadvantages

One disadvantage we ran into with Agile methodologies was the pace and tempo of the project was vastly increased. For example, if a member was ill for several days, as happened within the project numerous times, it became very difficult for them to keep track of what was going on within the team as everything

moves so quickly and up to several meetings may have already taken place reallocating responsibilities and tasks of both themselves and the teammates they may have been collaborating with.

Following up from this, if a member is unavailable for an unspecified amount of time, planning for the next sprint, and reallocating tasks during the current sprint becomes a very difficult process, as certain team members may have to juggle large amounts of responsibility and switch between tasks very quickly, giving them no time to adjust. This can affect the quality of the product, as if a team member is thrown into a portion of the code-base they are unfamiliar with, they may be required to both familiarise themselves with, and complete work-in-progress code.

Despite the value that test-driven development added, it also presented various challenges. Mostly in the form of tightened time-constraints, as after Increment 1, post implementation of test-driven development, we were required to both post-implement tests for existing code, as well as leave enough time in the following increment deadline to both write the tests for the code, and the code itself - ensuring the code passed the tests we specified.

2 Time Expenditure

2.1 Rough Time Estimates

Increment 1 - 70 hours **Increment 2** - 150 hours **Increment 3** - 30 hours
The project was broken down to tasks and assigned to people based on the skill relevance, but also well distributed among all group members. Due to a change in deadline, the early completion of all must haves for the project and Easter holidays between Increment 1 and Increment 2, actual time spent on the project is not evenly spread across all 3 increments. The development phase was completed in approximately 250 hours. Given 63 days and normal working hours (between 09:00 to 17:00), our efficiency factor gives a ratio of 2:1 for actual time taken to expected completion time. Each group member averages 4.63 hours per week, based on total number of hours divided by total number of weeks and size of group.

2.2 Most Expensive Activities

Setting up, instantiating, and optimising our database management classes to a level we considered sufficient was one of the most challenging tasks, both in terms of complexity and time expenditure, along with constructing robust templates for queries and filling them based on user input from the application.

2.3 Effort Estimation

We used the T-Shirt sizes technique to estimate time and effort for the project. User stories and tasks were analysed and given a size from the following:

Extra Small(**XS**): < 2 hours Small(**S**): 2-4 hours Medium(**M**): 4-8 hours
Large(**L**): 8-12 hours Extra Large(**XL**): > 12 hours

We broke down most user stories into smaller tasks - usually around the Small or Medium estimation categories - to better showcase the steps needed to achieve a user story target. However, some tasks were considered large to represent their overall difficulty.

2.4 Workload Balance

We tried to assign tasks during meetings based on each person's capabilities and considering how much work one already has assigned in order to make it as balanced and fair as we could. However, in cases where a member's assigned workload was less than the others and could complete it faster, said member still helped the rest of the team complete their respective tasks in any way they could to avoid remaining idle for the remaining duration of the increment.

2.5 Future Improvements in Time Management

For further projects, we will use the experience gained while creating our Ad Auction Dashboard to streamline our time management process. Although we split tasks as evenly as possible, whilst trying to assign the most suitable tasks for each individual, we found that some people were doing more work than others. If we were doing this again, we would do our best to assign tasks not based solely on skill set, but also on the size of the task, and if required, offer assistance to the person working on the task if the task turns out to be too large.

3 Tools and Communication

3.1 Design and Creation

Figma was used to design the application, and iterate over our designs. We used Figma as it allows for multiple users to be working on the same design concurrently and allows for plugins to be used so that we can import icons and images directly without having to go to other websites and downloading them etc., vastly speeding up the design process. It also allows for the developers to see the colour codes/icons and other design features directly in CSS so that the developer can view the details directly and ensure maximal parity between the design and product.

3.2 Communication

Due to the COVID-19 pandemic we were not able to meet face-to-face, and were forced to organise all our meetings online.

Many groups may have elected to use WhatsApp, however we decided that a WhatsApp group would be hard to follow and lead to unnecessary disorganisation when we had various platforms at our disposal more conducive to the task at hand; for the sake of openness and keeping things systematic, we chose Discord. Discord allowed us to have group voice / video calls, during the course of which we could also cast our screens in real-time, and we used it to keep track of upcoming meetings using the multiple text channels for various purposes. This feature in particular was very useful for us as it allowed us to easily and naturally organise ourselves in the context of communication.

We setup the Discord server to have a variety of channels; `#general`, `#changes`, `#session-planning`, and `#git`. The `#general` channel is our main text channel where we discuss about the project. The `#session-planning` channel is where we'll post reminders of meeting times and their specifics and topics to be discussed. The `#git` channel was linked to the GitLab such that we could see any pushes/commits that any of members of the team had uploaded in real-time. The `#changes` channel was used to keep track of committed changes to the code. This would be useful to reduce the number of conflicts each commit would get, as well as keeping track of the progress of individual members as the program responsible for posting the messages in the channel also posted the commit message alongside the notifying message.

Moreover, for communication and sprint reviews with our supervisors, we used the Microsoft Teams server set up by the University. It allowed us to schedule meetings, share files, cast our screens for presentation and most importantly - record meetings with our supervisor and download the transcripts after the matter.

Our communication strategy revolved around regular weekly meetings on Discord. Design and sprint planning are carried out in the first few meetings of each increment after discussing all relevant details for each task and their development methods.People are assigned to tasks in such a way as to carry out pair programming as much as possible.
Reviews are carried out in meetings where each assignee will update the team on their progress on each specific task, explain what they will do in the future and ask for help if needed. We will also discuss all tasks together to brainstorm ideas and facilitate the process for all of us.
Assignees work out their own working schedule to work on relevant tasks. Team meetings are arranged to be at least 2 times per week for regular updates on the running of the project.

3.3 Project Management

We chose Notion to keep track of the progress of our project. We also used Notion to keep track of files, tasks, and deadlines. In addition, the university's instance of GitLab was used for management, version control, and distribution of code to all group members.

We used Notion due to its litany of applicable features, such as real time updates, timelines (for planning purposes), tables (for planning), unlimited files, and a flexible organizational structure. It also has Kanban boards, and all of this allowed us to plan the project the way we wanted to, rather than conform to a more generically structured plan.

4 Advice to Next Years Students

4.1 Time-Management

Keep on track. Plan out each session before the meeting, and make notes about what needs to be done for the next session and also assign people to those tasks. Have a meeting at a minimum of twice a week and make sure that you are staying on track with the tasks.

4.2 Supervisors

Arrange frequent meetings with your supervisor and have a detailed plan of who is going to discuss different aspects of a given increment or a the whole project. Make sure to set up a relatively simple communication channel with your supervisor, so that you can resolve issues together effectively. Their feedback can more than often prove valuable for the future of the development process.

4.3 Project Management

When working in a group, accidents are the norm - issues like commits overriding your work and others can happen at any time, so be sure that the entire group has a satisfactory level of knowledge of your selected tool chain (like Git etc.).
Be sure to provide your team members with a way to post live updates of their work and do not hesitate to help each other with your respective tasks, especially if someone is falling behind.

4.4 Unexpected Problems

Many issues will unavoidably arise during development of your application, including and not limited to technical difficulties and changes in the deadline dates. Make sure you have discussed and planned with your group for such occurrences so that you are not caught by surprise when your group encounters such issues.