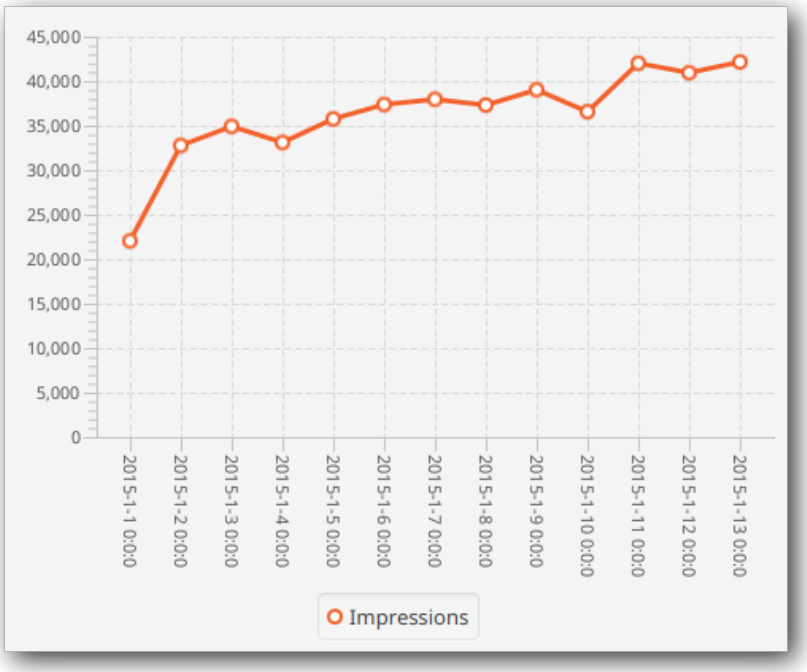
Ben Hadlington (bh9g22)
David Afonso Shepherd (das1g22)
Alejandro Andino Martinez (aam1g22)
Patrick Jfremov-Kustov (pjk1g22)
Kun Chen (kc4a22)

COMP2211
SEng Group Project

Ad Auction Dashboard
(2024)



Group 24
Version 1
Hand-in 5

1 Team Work

1.1 What Went Well

We believe that we displayed great teamwork; we completed all of our assigned tasks in a timely manner and made sure to keep up regular communication with each other as we did so (e.g. informing other members of tasks that we had completed that may come hand-in-hand with their tasks).

Through use of SCRUM meetings, we were able to manage our time and progress effectively, also helping us to allocate people to work in groups (peer programming) where additional support was required. These regular meetings also helped us to keep on track, especially while balancing our workloads around multiple other courseworks; if someone needed some extra help with their stories as they didn't have much free time throughout the increment, we were able to work around this. Furthermore, we believe that the use of SCRUM meetings helped us to ensure the final product was up to the standards set by our supervisor (acting as a sort of quality control), constantly reviewing the feedback we had previously been given and making design decisions accordingly.

Grouping of stories into "epics" helped us to track progress of different fundamental features of our piece of software. These are essentially groups of stories that all contribute to the same overlying concept. For example, we were able to plan our sprints around these epics - effectively targeting data processing and display (2 different epics) during the first few sprints. This helped us to understand what it was that we were actually working towards, as well as improved planning around features that depended on completion of other (more fundamental) epics (e.g. filtering was dependent on data processing).

Prioritisation of specific tasks using the MoSCoW notion in collaboration with story points helped to smooth the development process. This allowed us to focus on what was really important for each individual sprint, also providing a semi-accurate estimation of the amount of effort required for each story (which became more accurate through experience in the later increments). This, again, helped us to meet our deadlines in a timely fashion whilst also helping with the even distribution of our project's workload.

We split our team effectively, assigning one person to be in charge of the front-end and one to be in charge of the back-end. These specialised developers helped coordinate the rest of the team using their knowledge, who acted as full-stack. This came hand-in-hand with effective communication as we were able to understand, for example, how to link specific back-end features with the front-end interface. Although most people in industry are encouraged to be full-stack, we found that this worked extremely well for our use case as it provided a divide between the two ends whilst also enabling us to easily blend the two parts with help from the full-stack members of the team.

1.2 Improvements

With the Easter break period landing in the middle of our coursework, we dropped off most of our communication throughout. This wasn't good as it meant that we were unable to assess each other's progress throughout this time, leading to lots of last-minute changes and collaborative work before the submission deadline for the increment. We believe that if we had planned around this break we could've ensured that the same 2 week period was spent doing work together rather than over the span of 6 weeks individually.

Although SCRUM meetings helped to ensure the quality of the final piece of software, they were also difficult to plan. This was due to the fact that we had taken multiple different modules and, due to this, we were unavailable at a lot of times. There were instances where we had other commitments and thus some of us couldn't attend, but also we tried to plan around this wherever possible. We believe that we did the best we could going into this without prior experience, but if we had we planned our SCRUM meetings further in advance we could've effectively attuned them to our schedules and thus improved turnout.

We believe that we had split up our stories to the best of our abilities; however, there was still instances of dependency between some. This affected our burndown charts which could've been a cause for demotivation as it seemed as though little progress was being made (when this wasn't the case). For example, there is inherit dependencies between data processing and filtering, where once has to be done before the other. We attempted to remedy this by ensuring dependent stories were put in separate sprints, but this wasn't always possible based on our progress. With more effective planning, we feel that we could've avoided this issue entirely but it would be difficult to achieve.

1.3 Extreme Programming (XP) Values

Simplicity involves doing as is required, but nothing more. This was probably the weakest of the principles that we followed, as we found ourselves slightly ahead of the specification. Where we thought we could easily implement something (such as the loading of multiple different campaigns into the software, rather than just one) during the process, we did so. We did, however, mitigate failures as they happened and take small steps towards our goal through story sub-tasking. We found that although we deviated slightly from the principle of simplicity, we had great success with meeting the project specification and including all of our supervisor's feedback.

Communication involves regular face-to-face communication and teamwork. We were unable to do daily face-to-face communication, though we hosted weekly SCRUM meetings and maintained regular communication through WhatsApp between these times. All of us worked together to accomplish our goals.

Feedback involves responding to suggestions from our project supervisor and delivering working solutions at the end of each increment. We always made it a goal to deliver of something of value, and never included half-finished features within our review meetings. We noted down lists of feedback from our supervisor during meetings and used this as a criteria to work against and strive towards.

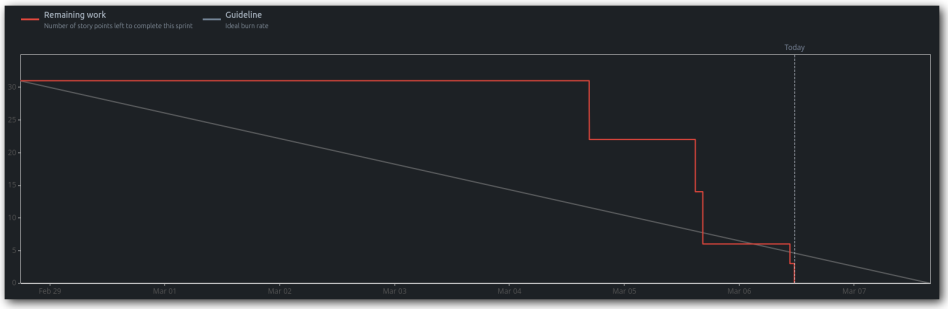
Respect involves giving everyone on the team the respect they deserve, also making sure to complete our own tasks as not to undermine the value of other people's work as this would be disrespectful. We always completed our own stories, even helping people with theirs where required. We never disrespected anyone, and weren't afraid to give people feedback on their work in a respectful manner.

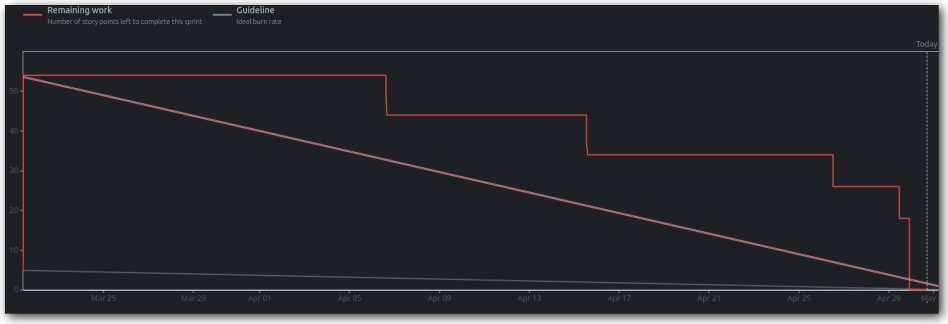
Courage involves being truthful about progress and estimates, without excuses. We always stuck to our plans and were always honest on our progress. If someone was struggling with their specific stories and were honest about this, we found other tasks for them to do or helped them to meet the requirements (typically using peer programming).

2 Time Expenditure

2.1 Time Management

Initially, we found that our time management wasn't perfect as we were leaving story completion until about a week in and then sweeping through stories at once. This wasn't ideal as it meant that we could easily get overwhelmed with things to implement and thus fall behind on stories that we had listed as requirements within our sprint plan. We did; however, improve with this over time (see our first burndown chart and final burndown chart below), burning stories from an earlier date and making steady progress.





2.2 Estimation Strategy

We opted for a Fibonacci-based effort estimation strategy for story points. This was incredibly useful as it meant that sums of story points were always proportional, helping us to allocate similar / identical numbers of story points to different team members wherever possible.

2.3 Expenditure Breakdown

Member	Estimated Time Spent (*hours/week*)
BH	9
DAS	9
PJ	8.5
AM	8.5
KC	8

2.4 Expensive Stories

Story	Points	Brief Description
Filter Metrics	8	Required the filtering of loaded metric data. Needed implementation on both the back-end and the front-end, including checkboxes for selecting different filters and adjusting the display based on these. Very involved and required strong communication between the front-end leader and back-end leader to properly implement.
Login	8	Separate story for both clients and agency members. Was very involved as it required setting up an external database to hold login information and access levels. Also involved adjusting the interface and permissions based on the logged in user's access level (i.e. allowing agency members to register new users).

2.5 Balancing the Workload

Using the aforementioned strategy of Fibonacci effort estimation, we were able to effectively balance the workload with little margin of error. Though there were other factors to take into consideration, such as time spent working on the report, these were always taken into account to ensure that a fair distribution of work was achieved. We could've, however, done this more effectively through use of different planning strategies, such as planning poker, in order to improve our point estimations for different stories.
We believe that although our initial effort estimations were wrong, we did improve at making these estimates during later increments and it was a matter of experience.

3 Tools & Communication

3.1 Tools

3.1.1 Development

IntelliJ was our choice of integrated development environment (IDE). We all opted to use this as it has great Maven integration, allowing us all to easily install project dependencies and thus work on the project on different machines. It also has built-in FXML scene builder support, which proved incredibly useful when designing the front-end system. Furthermore, it has amazing GitHub integration which helped us to effectively merge our code and collaboratively develop the final product.

GitHub was used for version control and collaborative programming. This proved to be incredibly useful as it allowed us to enforce branch protection rules that reduced the occurrence of merge conflicts and thus let us focus on programming. We regularly pulled in updates from the project and performed merges on our local branches, as this would further reduce the frequency of conflicts.

MongoDB was used for hosting our user database required by the project. We originally used Docker to create a local MySQL container, but this meant that everyone needed to set up a sample database on their own machine to access the application. Due to this, use of MongoDB was extremely useful as it provided a 24/7 cloud-based alternative that suited our needs perfectly (as we weren't meeting any data / request limits).

3.1.2 Progression

Jira was used to create sprint plans, generate burndown charts, and track our progress. This was incredibly useful as it was able to create burndown charts for us based on our sprint progress all within one piece of software. This meant that there was no need to input data elsewhere in order to generate sprint reports and thus made tracking progress extremely easy. We originally attempted to use Trello, but we found Jira to be so much more useful throughout the process as Trello couldn't perform the level of report-generation that Jira provides.

3.1.3 Documentation

Overleaf was our choice of text editor for writing our documentation. This was preferable to other text editors, such as Word, as it allowed us to programmatically generate our reports and format them with ease.

Photoshop was used for the creation of our storyboards. Though Adobe software is expensive, we happened to have a group member with access to it. This was incredibly useful as creating our storyboards was easy through using it, but we'd recommend using prototyping software such as Figma in the future as it's better suited for the purpose.

Lucidchart was our choice of specialised software used for creating UML diagrams. It supports multi-user collaboration, which allowed us to work on diagrams together and thus improved our efficiency greatly.

3.1.4 Communication

WhatsApp was used to maintain communication between SCRUM meetings. This meant that we were always aware of what each other were doing and could track progress verbally (e.g. how far someone is into a story, rather than a simple done / not done as specified by Jira), as well as help other team members if required.

3.2 SCRUM Meetings

We had semi-regular face-to-face SCRUM meetings, aiming to have them once weekly. We would've liked to have performed these more often, though with extra commitments this wasn't always possible. Whenever someone was unable to attend physically, we'd either call them through Teams and involve them with the meeting or inform them of what was discussed and ask for a simple progress check. This was effective as it meant that everyone was involved, regardless of whether they could attend.

3.3 Evaluation

We believe that most of the tools we used throughout this process worked extremely well together and helped to streamline the agile process. Jira was extremely fit for purpose, being a piece of software that is actually used in industry for tracking agile projects. Furthermore, the integration between GitHub and IntelliJ made handling merging of code and working collaboratively far less daunting. There were cases where there were other viable alternatives to software, such as using Figma instead of Photoshop for storyboards, but overall we believe that all of the tools we used complimented each other well. Furthermore, we believe that through using these tools and getting accustomed to them we will have a far better time adjusting to an actual work environment in which many of these different tools will be used together.

4 Advice for Future Students

4.1 Do

SCRUM MEETINGS: Always make an effort to engage in SCRUM meetings as much as possible. The more regular you can have these the better, as we found it to be extremely motivating to check progress and have constant goals to work towards. It may seem like a pain at first as you need to find time to meet together face-to-face, but it's more than worth it.

COMMUNICATION: Ensure that you maintain communication with your group members, even between SCRUM meetings. It's important to keep talking regularly, asking for help where needed and informing others of your progress. This will help with managing your time and with any collaborative work in the future.

TOOLS: Make use of existing tools and technologies that'll make your life easier. We can't recommend Jira enough as it'll handle all things sprint planning related, including stories, epics, burndown charts, sprint planning, and so much more. Use of these tools will help you to focus on the software-side of the project and reduce the workload required for the documentation and planning aspects of each increment.

FEEDBACK: Your supervisor is there to guide you, not work against you. They are ultimately the one that decides how you're getting along in the project and it's important that you attempt to incorporate most (if not all) feedback that they give you. Yes, it's important to allocate time to work on your must / should user stories, but it's also just as important to spend sessions working through any feedback that your supervisor might've given you.

4.2 Don't

COMPILATION: Avoid leaving compilation of the project and packaging it to a jar to right before hand-in deadlines. A lot of groups, ours included, had issues with this and it's far better to get this set up in advance to any actual programming. Ensure that the jar is also compatible with multiple operating systems, including Mac, UNIX-based, and Windows. We recommend using shade to do this.

TESTS: Avoid incorporating your tests after the implementation of code. Strive for test-driven development, creating method signatures and then creating the tests with expected input / output before writing the actual implementation.