Software engineering group project

Deliverable 5 - Final Report

	Name	ID
1	Marwan Altamimi	32489005
2	Juzheng Bai	32410964
3	Henry York	33113556
4	Jack Breslin	33055521
5	Aditya Bansal	33335052

Supervised by
Gerasim Tsonev

Teamwork Evaluation

At the end of the project, we believe that overall our team was successful in performing well together to create our Ad Auction System. As a team we were able to all collaborate together successfully with a good team effort all around, partly thanks to our efficient planning of our main design of the system's interface where we all discussed our ideas for how our system should look, and what features the interface will need to fulfil the requirements, such as having a bar graph to show data and buttons to upload files to the system. We all were able to collectively agree on a design that was appropriate and clean, which we thought suited the system's functionality well, through the use of storyboards and other design pieces that we could integrate into our system.

While our project was successful, we believe that some steps could've been done better along the project's timeline. In the beginning, it took us some time to start communicating and arrange the start of our project partly due to external circumstances and unclear communication, which did occur more often than we liked during arrangements of meetings together, and this used up some of our time in the start. However, once we were able to contact each other we were able to start quickly on our project. I think in hindsight, it would have been better to have asked each team member what form of communication they'd prefer at the start so there would be less trouble in communication. Thankfully, this didn't have much of an impact on the overall project.

At first we thought that it would be difficult as a team of five to create the system as how we envisioned in our first increment, but despite having less members we were able to work well and implement all the requirements for our Ad Auction dashboard, as well as include some of our own extra features such as saving images in our system. However, throughout the project we had issues with organising our sprint plans and the tasks associated, such as overestimating or underestimating certain tasks in our sprint plans that made it difficult to properly balance the work in our sprints and would have caused problems with time management later down the line. In order to mitigate this, we had to make some changes to our sprint plans and increment plan to mitigate this, but through this we were able to balance our overall workload better which helped to ensure our success in our project. Despite this, there were a few user stories that we had to sacrifice in order to keep inside our schedule, some of which being the interface themes for colour blindness and improved scaling of the system. These features would have improved the overall usability of the system so we had lost some value due to this sacrifice, but as they were low priority in our increment plan we were able to gain more time to work on our higher priority user stories.

Our team performed well together, but one thing we failed to take advantage of was each team member's individual skills and weaknesses during assigning our roles and responsibilities in the group. At times when we were assigning tasks for the team to do it was done quickly and without much thought or discussion, which led to some team members having some trouble with their assigned tasks. Thankfully we were able to have frequent meetings together to discuss this and ensure all team members were doing our work. In the future, I think as a team we should discuss our strong and weak points and be more proactive on which parts of the project we can handle and which we cannot.

We also think that our system could’ve used more varied testing methods during development. Our main approach to testing was integration testing, scenario testing and user story acceptability testing, but a smaller part was given to JUnit tests due to our lack of knowledge with it and our time constraints. While we were able to manually test in each increment and after each major addition to the code was made to make a complete and robust system, we believe that our JUnit tests for our backend processes could’ve covered more ground, for example increasing the variation of test data and the variety of test types used, including partition testing and boundary testing within our tests. It would have been useful for our test-driven development to have more automation which could have made our testing more reliable and effective.

In conclusion, we were all happy with the final design and were pleased that we were able to make it perform to our customer’s liking.

During the development, our team incorporated various agile methods into our design of our reports and system. An advantage of following these methods was that it was a very flexible and adaptive process. Our meetings could be adjusted and arranged to our liking which also allowed greater collaboration as a result and made sure everyone had an equal chance to participate in the team. In addition to this, our storyboard and UML design elements could be easily added upon or redesigned as development went along, which made changing our vision fast and efficient. It was also useful to have a customer that we could talk to at all times and ask for feedback and advice while we were making our system, which helped refine some holes in our design and guide us in the right direction.

However, at times, using this approach also gave a lack of predictability, mostly due to sudden risks and changes in the project that were wanted by the customer. While we were able to mitigate this predictability by making a risk assessment plan, it could be disorientating at times. The process of holding scrum meetings was valuable not only to engage with our team and occasionally our customer, but so we were all aware of the progress being made and the current state of the project and what to do next. Being held to a schedule was useful to us by ensuring that by following this process the project’s goal would be able to be reached in our given timeframe. On the other hand, a strict schedule also put a lot of pressure on us, which was slightly difficult as a newly formed team to manage. It may have been more helpful for an experienced team to follow these agile principles as the team would have more knowledge of each other’s strengths and how to perform well as a team under strict conditions. A high level of collaboration is also required to keep up with the schedule which can be daunting at times for new teams like us.

Agile planning also has the benefit of identifying and mitigating risk in the early stages of development such as our risk assessment table in our envisioning plan. Unlike a linear based workflow, if we came across any risks we had identified we were prepared to deal with them efficiently so that development could keep going as planned.

The application of Extreme Programming (XP) values within our project proved to be useful overall. The XP values we utilised include storyboards, pair programming, testing before implementation,on-site customer involvement & iteration.

The first XP value we used was storyboarding, which proved to be very useful for planning the design before attempting to implement the GUI. Having a visual reference of the user interface meant we could see how our users would interact with our system without committing to a specific design through code. This also meant we could identify potential issues which could arise and saved a lot of time having to change code. It was also an easier way to reach a consensus on potential design choices without taking the time to code out multiple different options.

We also had an on site customer (in the form of our supervisor) which meant we were able to receive real feedback on our system and make real-time improvements. This created a very useful feedback loop, allowing us to make sure our system was actually addressing the needs and expectations of the customer.

Writing tests before beginning implementation was another successful XP strategy we utilised to guide us through the development process. This gave us a clear pathway for adding features and implementing them in a way which met predefined test conditions. The main benefit of this was being able to catch and fix bugs early, ensuring our system was robust and reliable.

Pair programming was another XP strategy we implemented into our workflow, allowing us to collaborate on different areas of our system at the same time which helped spot potential issues or improvements we could address in our code as we worked together. We found this method greatly improved the efficiency and quality of our code. It also ensured all of our team members were familiar with the entire codebase, allowing everyone to learn from each other and improving our individual skills in the process.

In conclusion, the XP values provided us a structured framework we could use during the development process. This allowed us to adapt quickly, maintain quality and ultimately deliver a product which addressed our customer’s needs and requirements.

Time Expenditure

How much time did each person spend?
From our sprint logs, burndown chart, and taking into account time spent making our UML diagrams, storyboards, documents etc , we estimated that each member spent around 10 hours per week or 2 hours per working day on the overall project.

Which activities were most expensive?
We think that the activities that took most of our time in this project were designing the GUI, our tasks in the first iteration of our project, designing the compare screen in our system, testing and making our design elements.

Designing the GUI of our system took quite a lot of time due to it being a very extensive part of the user's interaction with the system. We had to spend time agreeing on a design that would be appropriate for our system with storyboards and take this design and implement it through our project's code. While it was easier to manage with the help of Scenebuilder we had to make sure each UI element was correctly sized and could fit in the theme of the system.

One part of our system that took a lot of time was the comparing graph feature, which involved having a separate screen that allowed different types of graphs to be displayed in a grid. We found that it involved a lot of separate parts, such as moving the graphs to the screen properly and each having a zoom in button to zoom in on the graphs which required a lot of programming to set up.

Also, we found that the user stories and tasks involved in the first increment were quite expensive as at that point, our system had to be built from the ground up and our increment plan also leaned on the starting point in terms of workload, involving creating the UI and computing all the key metrics in the data log files. Since the later increments built upon our first increment, we believe that out of the three the first increment was the most expensive to design.

Since our development was test-based, we found that testing was expensive in general thanks to the thoroughness of how we applied it to our system. Creating JUnit tests was quite lengthy as it involved creating a lot of test data alongside the tests, and when we needed to test our user stories and scenarios it involved going through the systems and checking each case of the tests related to them, which took some time for each member to do.

Finally, our design artefacts involved making storyboards, class diagrams and use case diagrams, which involved a lot of thought and discussion as they were the initial base to our system's features.

What type of effort estimation Did you use?
During the sprint plan we would make a rough estimate of how many hours it would take to complete each task, and then once the task was completed we would input the actual hours we spent on each task. We also used MoSCoW to decide what level of importance each task had and how we should prioritise them in our increment plans.

How did you balance the workload for each individual team member?
For each increment we evenly distributed the number of hours between team members, making sure everyone was okay with the amount of work they had. We also worked together on our tasks in a group most of the time. If there was an unforeseen circumstance causing a member to have to redistribute some of their tasks to another member, they would be assigned tasks that are estimated to take longer to complete for the next increment for fairness.

How could we have organised our time more efficiently?
In hindsight, we should have organised our next online scrum at the end of the meetings, rather than sending messages an hour or so before. We could have also done research into how to complete each task before we did group working sessions, as we would spend time reading documentation of different Java libraries and watching youtube videos during these sessions. If we did research beforehand, more progress on the tasks would have been able to have been made during the sessions, speeding up the project completion.

Tools & Communication

In this section we outline the tools that we used to finish our project, they are ordered by their effectiveness and usefulness, with the most important ones first. We also explain how we organised our in person meetings, and what we used them for.

Intellij
Code and development. This was our IDE of choice as everyone was familiar with using it to make JavaFX programs from our experience in programming II. It also supported group collaboration on code with the code-with-me feature, making Intellij a highly valuable tool.

Google Docs
We used google docs for making our increment documentation and report and organising handins. It allowed group collaboration so everyone could edit the documentation at the same time.

Gitlab
Repository for code development. We pushed each new version of code to Gitlab to keep up-to-date versions of the code available to everyone, and it allowed updates to be pushed by each team member and merged with the main project. Useful for mitigating any risk of data loss.

Discord
We used discord to host remote scrum meetings, as well our meetings with our supervisors to show our progress and quickly ask questions. We also used it for sharing links to the other tools and documentation for Java libraries, as well as general team communication such as arranging meetings and asking questions. This tool was essential, as without it the amount of meetings hosted would have been far less, due to at times we had group members away from Southampton and in different countries.

Scene Builder
This software was used for designing the GUI. This allowed us to develop our GUI to match how our storyboards looked and allowed us to easily apply our Java methods to the GUI buttons.

LucidChart
UML diagrams and burndown chart creation. It included collaboration tools for each team member to work on the diagrams at the same time and it was easy to import the diagrams. LucidChart allowed for modularity, as it was easy to add features and components, or to make slight modifications to our existing UML diagrams when we reached a new increment and when we implemented the feedback given from our supervisors.

Pen and Paper
We used these to draw our storyboards. We chose this over other tools as it allows for quick design without causing us any limitations that other online resources may have. It was somewhat useful as it was only used for the story board creation and could have been replaced with the likes of LucidChart / MS Publisher as pen and paper is not as modular to the likes of the previously mentioned LucidChart.

Trello
Task management. Useful for breaking down tasks and showing which parts have been finished to track progress clearly. Limited use outside of arranging tasks, and it came hand in hand with the burndown chart which both tracked progress over time.

Teams
Our initial method of reaching out to group members and supervisors, which was replaced by discord early on so it was not all that valuable to the team. As everyone had emails it was easy to contact group members this way and discuss our preferred method of communication from there.

Strategy of Physical meetings and how often we held them
We all agreed to meet up at a certain date and time beforehand using discord. We then booked rooms on highfield campus to meet up in. This would only occur a few times a week, as most of our meetings were held online to make it easier for everyone to take part if they were away from Southampton or had plans. We had an Initial meeting on discord to introduce ourselves, then we arranged a physical meeting to discuss designs of the project and create our initial storyboards during the envisioning process. We would also have other in person meetings for catchup. During these meetings each member would tell the group about what they have done since the last meeting, where the project is currently at in terms of progress, what is left to be done, review of tasks left to do and an estimate on how long it will take to complete. Finally we used the scrum meetings to assign tasks to each individual, and ask how each task was going, for example asking if someone was struggling to complete the task, or if tasks had been completed without any issues.

Advice

Learning from our experience of working as a team we have thought of some advice to give to the next year students when doing their SEG project in the future. This advice ranges from general team tips to the more technical side of the project.

Communication Methods

Establishing clear parameters for communication throughout the project is critical. For instance, using Discord or teams for group meetings is crucial to ensuring successful project cooperation. Additionally, this lessens confusion and helps to avoid misunderstandings.

Arranging Meetings

Make sure meetings are planned ahead of time, at least a few days prior, to give your team members notice, and time to let you know if they are unable to make them. Pushing meetings to be later and earlier is also possible if you need to reschedule.

Feedback

It's essential to actively seek out and learn from customer feedback. The reason for that is that It helps you meet their specific needs and expectations, you can spot and fix problems early on, and it helps in the long-term improvement of the software system.

Assign tasks

Break down the project into smaller tasks and assign them to team members according to their abilities and previous work. It is important because breaking down tasks to smaller ones helps with organisation efficiency and risk mitigation.

Deadlines

Set deadlines for each individual task. The reason for giving deadlines is to ensure that the team will be more accountable, better manage their time, and prioritise their duties. This can make it more likely that the project will be finished on schedule

Designing GUI

SceneBuilder is a useful software tool for designing the GUI and linking the controller with the interface. It saves valuable time during development because you can see your GUI as you add elements to it, rather than trying to create it with just your code. It's also more flexible compared to configuring each GUI element manually in the code, whereas it's more drag-and-drop in Scenebuilder which is easier to work with.

Task reevaluation

If you feel like task priority needs to change or some tasks need to be left out for time, make sure to justify why and provide an updated sprint/increment plan if necessary.

Check ups

Regular check up and scrum meeting to see how everyone is getting on with their assigned tasks, and if changes need to be made.This can make sure that everyone is aware of the project's objectives, deadlines, and expectations. And also help with project monitoring to identify any issues before it is too late

Notes
Take notes during supervisor meetings to keep a record of feedback and what could be changed, added or removed.

Adding features
Make sure that everyone is familiar with new additions to the system so all team members are on the same page.

Remote working
Use tools that have group collaboration features such as Intellij’s code with me which can be useful for pair programming, and google docs for collaborating on documents.

Communication
Informing other group members if you have been given a task that is too challenging for your particular skill set, as other group members may be willing to change tasks with you so everyone has a fair and doable workload.

Helping others
If your assigned tasks are quicker to complete then initially expected offer help to other members to speed up the completion of all tasks for the increment.

Follow up
Following up with customers after they offer feedback is an essential step in the feedback process. It assists in making sure that feedback is completely understood, highlights areas for development, and increases customer satisfaction.