Group 48: Final Group Report

Songyang Ding (sd8g22@soton.ac.uk)
Samuel Hyder (sh5g22@soton.ac.uk)
Andrew Shipway (as27g22@soton.ac.uk)
Muhammad Izi Arman (maia1g22@soton.ac.uk)
Samuel Wiles (sw14g22@soton.ac.uk)

August 29, 2025

1 Introduction

This document presents a comprehensive analysis of our software engineering group project. It examines our collaborative processes, the tools employed, and offers reflections on various aspects of our project. The structure of this report is designed to first evaluate our collective teamwork, followed by an analysis of time allocation both as a group and individually. Subsequently, we will discuss the tools and communication methods we utilized, and conclude with some recommendations for future students. Throughout this report, we will provide examples from our experiences to enhance understanding and support illustration.

2 Evaluation of Team Work

In this section, we critically evaluate our teamwork and collective experience. We will outline the aspects of the project that were successful, as well as identify areas where improvements were necessary. This balanced assessment aims to highlight our strengths and also acknowledging areas for growth and enhancement in our collaborative processes.

2.1 Successes

- **Time Management:** Our time management skills were a cornerstone of our project's success. We effectively prioritized tasks and adhered strictly to our planned schedule, which allowed us to complete all sprint targets ahead of the deadlines. This approach not only ensured that we had more time for thorough testing but also provided us the opportunity to enhance our project with additional functionalities. The ability to finalize our work early significantly reduced stress and contributed to a high-quality final product, demonstrating our team's capability to manage time efficiently.
- **Communication:** We also did very well in communicating. Throughout the semester, our near-constant communication played a pivotal role in the success of our project. This continuous exchange allowed us to efficiently divide tasks, seek and provide feedback, and maintain a high level of mutual respect within the team. By keeping open lines of communication, we ensured that everyone remained aligned and engaged, significantly enhancing our collaboration and overall productivity.
- **Strong Leadership:** Leadership played a relatively vital role in our project too as the leader for each increment was given a fair share of responsibility. Despite this everyone managed their leadership expertly. All members had shown strong leadership, stayed punctual with bookings and meetings, organised others and gave guidance where necessary.
- **Extreme Programming Values (XP):** One of our biggest successes was our implementation of XP values, which led to efficient and structural product development. XP values will be discussed in more detail in the following sections.

2.2 Challenges

- **Lack of experience:** Unfortunately, due to this being most of our first group coding work, we were slightly ignorant of maintaining proper coding practices. This lead to some confusing variable and class names, subpar documenting which hindered progress later on in the project.
- **Member adjustment:** Our group was reduced from 5 members to 4, which introduced some challenges in teamwork. Firstly, the workload per team member increased, which led to burnout and decreased productivity when not managed effectively. With fewer people, there was also a smaller range of expertise and perspectives available. This was particularly painful to us as we were made aware of their preexisting skills using GIT, JIRA, and Java.

- **Integrating contributions:** Initially, as a smaller group, we assigned specific members to focus on coding or report writing to foster expertise in those areas. This strategy posed challenges later in the project when we all needed to engage in coding tasks. This led to a collective need to refamiliarise ourselves with our code, which consumed valuable time. In hindsight, involving everyone in all aspects of the project from the start would have been more beneficial.

2.3 Agile Methodologies

Throughout the project, we utilized agile methodologies to achieve a more efficient and adaptive construction process. We integrated scrum development into our project, along with comprehensive sprint plans for each deliverable. During weekly meetings with our supervisor, every team member actively participated and asked relevant questions, especially the leader for that increment. These interactions sometimes presented new challenges that we had to quickly adapt to and overcome. Additionally, we leveraged incremental delivery to plan future increments more effectively, allowing us to envision subsequent steps with greater ease. Below, we evaluate agile methodologies based on their benefits and limitations, along with our team experience:

2.3.1 Advantages

- **Customer Participation:** We had regular, punctual meetings with our supervisors, who acted as customers in this project. During these sessions, we always presented our most up-to-date report and a stable version of the software, providing valuable insights for the customer. This transparency in communication allowed customers to voice their opinions and concerns, which were essential for project development.

- **Flexibility:** The team consistently reviewed and adjusted the progress plan. Any essential changes to the project, such as requirement modifications or unexpected workloads, were handled immediately.

- **Productivity:** Regular daily meetings fostered enthusiasm within the team and drove us towards our objectives. These meetings helped us stay focused on development tasks and increased our productivity.

2.3.2 Disadvantages

Overall, we had a smooth experience adapting to this methodology. However, we also encountered some challenges associated with agile methodologies:

- **Loss of Focus:** While frequent team meetings generated many practical ideas, they sometimes blurred the actual user requirements and distracted our focus. From our experience, during the design process, our team was excited about discussing the design and scene transfer logic, which occasionally caused us to overlook the primary goal of making the software user-friendly and easy to comprehend.

- **Time Management:** Scheduling regular meetings became challenging as additional workloads from other modules arose towards the final period. Discussions about availability sometimes hindered our ability to fully utilize our communication and resource management.

2.4 Extreme Programming

The group's approach to Extreme Programming (XP) is outlined below according to its five key principles, detailing how each principle was followed and the benefits they brought to the development process.

2.4.1 Communication

Firstly, we preferred having team meetings be in person as this ensured the most effective and direct communication. Therefore, we made that a priority for all meetings. This consisted of planning when and where we would meet dynamically on our WhatsApp group chat. Any important updates were notified to everyone via the group chat. As such everyone was always made aware of any updates that could affect others. Additionally, communication with our supervisor was always taken seriously and enhanced with the use of well-thought out questions to specifically address areas of uncertainty.

2.4.2 Simplicity

By focusing on simplicity, a team can streamline their workflow, prioritise essential tasks, and deliver functional products more swiftly. At the beginning of each week we discussed which user stories we could break down further into smaller concrete parts. This helped to abstract the task into easier to understand individual work that would build back up into a working whole. Furthermore, it was easier to transition smoothly between these simpler tasks than a big complex one. It also helps the team member working on that task to understand it more easily.

2.4.3 Feedback

Feedback is crucial for continuous improvement and adaptability. Regular, constructive feedback loops—whether through code reviews, pair programming, or via our supervisor allowed us to quickly address issues or improve already great work further. Often, we would comment on other team members work and add our own helpful, constructive criticism. Sometimes, conveying feedback kindly can be difficult. To address this, we took steps to explain why certain initiatives were implemented. We also made sure to clarify why we believed a particular change would improve the overall product. Therefore, we could identify weaknesses early and make appropriate changes. Our supervisors' feedback was always prioritized and taken extremely seriously. Their insights were invaluable in guiding our development process.

2.4.4 Courage

Despite being one team member down, everyone stepped up and contributed beyond what was expected of them. We all demonstrated courage by willingly taking on additional tasks, pushing ourselves beyond our comfort zones. This impacted our project and guided us to more innovative solutions that required more thought. However, ambition is very easy to succumb to and this occasionally hindered progress because of how easy it is to become fixated and refuse help. Consequently, sometimes individual tasks could take longer than expected.

2.4.5 Respect

Respect was manifested through how we interacted with each other and acknowledged each team member's contributions and opinions. Everyone was asked for input for all major decisions such that everyones opinions and thoughts were equally valued and considered. This fostered a positive working environment where no one was afraid to give ideas to others. Through time this built up and meant more respect was earnt and thus given to each team member because we were all working towards the same goal.

3 Time Expenditure

In this section, we will assess our time management throughout the project. Firstly, we will provide a general overview of the overall time spent. Subsequently, we will delve into more detailed evaluations of each team member's contributions and specific tasks.

3.1 Overview

Our team successfully completed all scheduled tasks on time for each sprint. This achievement reflects the considerable effort and dedication invested by each team member. More importantly, our team had only four members from the start of the project, making each person aware of the additional workload and time pressures throughout the development. This heightened awareness helped us remain conscious of the extra effort required and encouraged more frequent communication and meetings.

3.2 Planning Strategy

During the planning stage in envisioning, after extensive discussions and consulting with our supervisor, we decided to complete most functionalities by Sprint 2. This decision was influenced by more upcoming coursework deadlines from other modules and the limited availability of team members during the Easter break. During the sprint development, we committed to daily meetings where we worked collaboratively for two to three hours and we often increased the frequency of these meetings if a sprint deadline approached. Sometimes, we met over the weekends to catch up if we feel we are behind schedule.

3.2.1 Time Estimation

When planning the tasks for the following increment, we tended to estimate more time for each task, adhering to the principle that it's better to overestimate than underestimate. Our initial time estimation strategy for each task is based on a combination of task importance and estimated difficulty. However, as many design techniques and program libraries are relatively new to us, we also factor in the time needed to research, learn, and thoroughly test them before implementation. This approach has occasionally led to overestimation, particularly for tasks that are deemed very important but are not as difficult to learn and implement as anticipated. The tendency to overestimate task duration necessitated the reallocation of our workforce during increments 1 and 2. Recognizing this, we improved our prediction processes to be more realistic and our estimation time was becoming more accurate as the team gained a deeper understanding in JavaFX through the development.

3.2.2 Challenging Tasks

There were some tasks we anticipated to be challenging, particularly testing, which we expected to become more time-consuming as the project progressed. Our team consistently scheduled at least 5 hours for testing in each sprint, and we assigned a member who had not been heavily involved in the development to conduct the tests as a tester. Although this approach required additional time for communication and explanation, we believe it provided significant benefits. It helped ensure that our software was accessible to clients, who have limited knowledge of development.

There were tasks whose difficulties we underestimated. Overall, implementing the user interface in JavaFX proved to be far more time-consuming than expected. Throughout the development, we encountered numerous challenges in aligning the system with our designs, as the styles of the components could not be easily manipulated, and significant time was spent testing and ensuring the appearance was correct. After recognizing this difficulty in the early sprints, we allocated more time for front end design implementation in later sprints.

3.3 Workload Distribution

Figure 1 below illustrates the continuous adjustment in workload distribution for each team member. As previously discussed, our goal was to complete the majority of tasks before the Easter break, leading to a substantial allocation of our efforts during increment 2. This figure demonstrates a continuous process of refining workload distribution: the member who contributes the most in one sprint is assigned less demanding tasks in the following sprint, and vice versa. We also took into account personal task preferences and expertise. While this occasionally influenced distribution outcomes, we aimed to spread tasks evenly among all members, ensuring that everyone accumulated a similar total number of working hours.

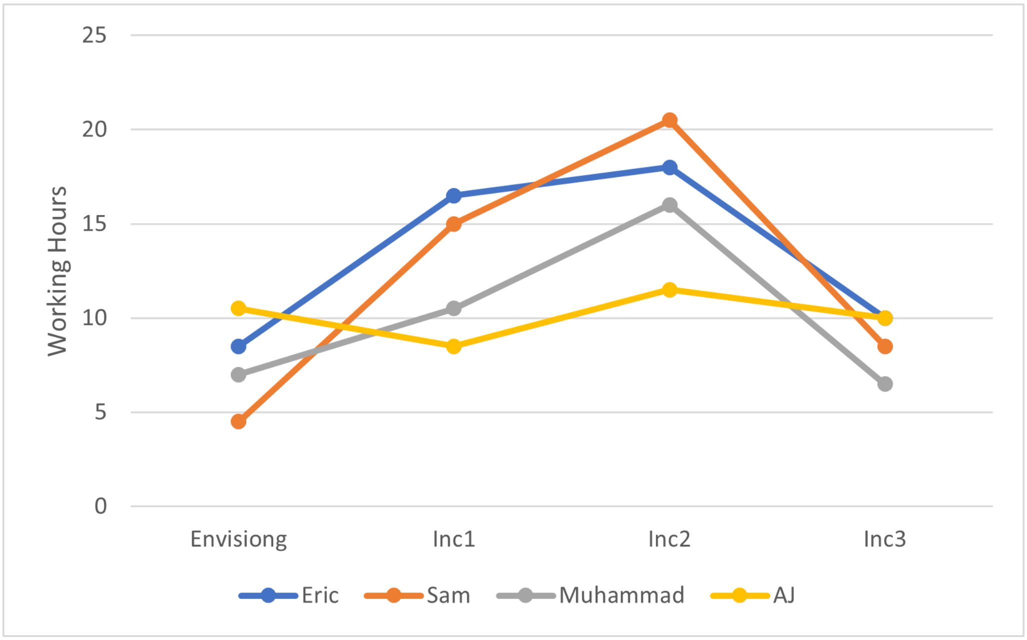

Figure 1: Working Hours

3.4 Potential Improvements

There are areas where we could improve our time management if we were to undertake this project again.

- **Sprint Breaks:** We typically allowed ourselves a break after each sprint deadline on Thursday. However, our weekly meetings with our supervisor, which were extremely helpful in clarifying user requirements and were really useful for our design ideas, were scheduled every Tuesday. This often meant that we had limited progress to discuss in these meetings and did not maximise the benefits of support provided by the supervisor.

- **Work Distribution:** Each time we planned and distributed tasks for the next increment, we estimated only the time required for implementing user requirements. Other essential tasks such as report writing, creating backlogs, and preparing for marking meetings were often overlooked. These tasks are equally important and should be included in the planning stage.

4 Tools and Communication

In the course of our project, we leveraged a suite of tools tailored to optimize our software engineering processes, foster collaboration, and streamline communication. Each tool served a distinct purpose, contributing to the overall efficiency and effectiveness of our project execution. Below is an evaluation of the tools utilized, delineating their respective value and highlighting those that emerged as particularly impactful:

1. **WhatsApp:** WhatsApp served as our primary means of communication within the team. Its real-time messaging feature allowed us to quickly coordinate meetings, share updates on our progress, and address any immediate concerns. While WhatsApp lacks sophisticated project

management features, its simplicity and widespread usage made it highly accessible and efficient for day-to-day communication.

2. **Overleaf:** Overleaf proved invaluable for collaborative writing and document management, particularly for our reports and self-logbooks. Its LaTeX-based platform enabled seamless collaboration on complex documents, ensuring consistency and version control across multiple contributors. Overleaf's real-time preview feature and extensive LaTeX support enhanced our productivity and the quality of our written deliverables.

3. **IntelliJ:** As our primary integrated development environment (IDE), IntelliJ provided robust support for code development and debugging. Its intuitive interface, advanced code analysis tools, and extensive plugin ecosystem streamlined our software development process. IntelliJ's seamless integration with version control systems like Git further facilitated collaboration and code management within the team.

4. **Microsoft Teams:** Microsoft Teams served as our primary platform for formal communication with our supervisor and for scheduling meetings, presentations, and soliciting project feedback as well as for submission of every deliverable to the supervisor. While Microsoft Teams provided a structured environment for formal communication, its effectiveness was somewhat hindered by occasional usability issues and dependence on stable internet connectivity.

5. **Excel and Jira:** Excel and Jira were instrumental in managing our project's scrums, product backlog, and sprint planning. Excel provided a familiar and flexible platform for organizing and visualizing project data, while Jira offered more specialized features tailored to Agile project management, such as sprint burndown charts and issue tracking. The combination of Excel and Jira allowed us to effectively plan and track our project progress, although the learning curve associated with Jira may have posed challenges for some team members.

6. **Lucid:** Lucidchart was utilized for creating UML diagrams to visualize the architecture and design of our software. Its intuitive interface and extensive library of shapes and templates simplified the creation of complex diagrams, aiding in the communication of our design concepts within the team.

7. **Microsoft Powerpoint:** Microsoft PowerPoint was used to design the storyboard of our program, providing a visual representation of the user interface and user interactions. Its familiar interface and rich feature set enabled us to create polished presentations for showcasing our project progress and findings.

8. **GitHub:** GitHub was initially utilized for version control, with the repository being cloned into IntelliJ for code development. While some progress was committed and pushed to GitHub, regular use of the platform was not maintained. Instead, team members frequently exchanged code via WhatsApp group chat. We believe that we could have spent more time using GitHub so that our code development would be more efficient. Regular and disciplined use of the platform would facilitate real-time updates, enhance version control, streamline code integration, and mitigate risks associated with manual file sharing.

In summary, each tool played a crucial role in supporting different aspects of our project, with varying degrees of effectiveness and value. While **WhatsApp** and Overleaf stood out for their simplicity and collaborative features, tools like **IntelliJ**, **Jira**, and **Excel** provided essential support for software development and project management. **Microsoft Teams** served as a centralized platform for formal communication and scheduling, albeit with some usability limitations. **Lucidchart** and **Microsoft PowerPoint** contributed to visual design and presentation tasks, albeit with differing levels of interactivity and integration. Overall, the combination of these tools enabled our team to effectively collaborate, communicate, and deliver on our project goals. **GitHub**, while initially underutilized, proved to be a vital tool for version control and collaboration, highlighting the need for more consistent use in future projects. Overall, the combination of these tools enabled our team to effectively collaborate, communicate, and deliver on our project goals.

5 Advice to Next Year's Students

As a group, we have included below the recommendations that would benefit future teams undertaking the software engineering group project. These recommendations derive from our own experiences and challenges faced whilst completing the runway redeclaration tool, however, we believe these would additionally translate to the ad auction project. The advice is intended to guide next year's students, equipping them with strategies that foster success:

- **Communicate frequently and in-person:** Communication amongst team members is a vital aspect of the group coursework and we found that regular, face-to-face meetings were essential for team cohesion and ensuring that all members were consistently aligned with the project's current status and goals. Meeting in person allowed us to have direct interactions which led to a clearer understanding of one another's ideas or current situations that they may need assistance for, such as errors in code, design issues etc.

- **Effective time management:** Time management helps avoid piling on an unreasonable amount of workload until the last minute. As mentioned in our reflections from previous sections, we recommend start the next sprint straight away after each hand in session with the markers. It helps the group understand what tasks may be more difficult to complete and it will give the group time, if necessary, to adjust the sprint plan.

- **Active engagement with the group supervisor:** Utilise the group supervisor as a key resource. Our supervisor played a critical role in addressing any issues or queries we had regarding the coursework and would always provide clear explanations and reply quickly. Additionally, we found it useful to, before each weekly marking meeting, prepare a list of anything that we may need assistance with that might be hard to articulate through messaging on teams. By doing this we made sure to optimise how useful each of these meetings could be.

- **Commitment to agile methodologies:** Agile methodologies will help enhance responsibilities and allow teams to adapt to changing requirements. As a group utilising scrums and iterative development was extremely useful. In our team reflection, we didn't use agile tools to its full capacity, so we would recommend researching and learning such tools and trying to implement them from the start of the project.

- **In-depth understanding of the specification:** We recommend continually revisiting the user requirements throughout the development process. A thorough comprehension of each aspect of the specifications can serve as a reminder of the requirements. During programming, there may be areas that seem straightforward to code initially, but you might realize that your previous code is not sufficiently adaptable to your new ideas. A comprehensive understanding of the project will enable you to code with consideration for future enhancements, ensuring greater flexibility and scalability.

- **Preparation for the marking meeting:** Before each biweekly marking meeting, we found that preparing through doing "mock" sessions was useful, we would practice running through our product showing all of the updated features along with any relevant testing. From doing these it allowed us to have consistently smooth marking sessions.

To conclude, employing these strategies will enhance the overall project outcome as well as contribute to a greater learning experience.