# Change 2

## Group 2 - Vikingz

Damian Boruch

Tommy Burholt

Elliott Bryce

James Butterfield

Ren Herring

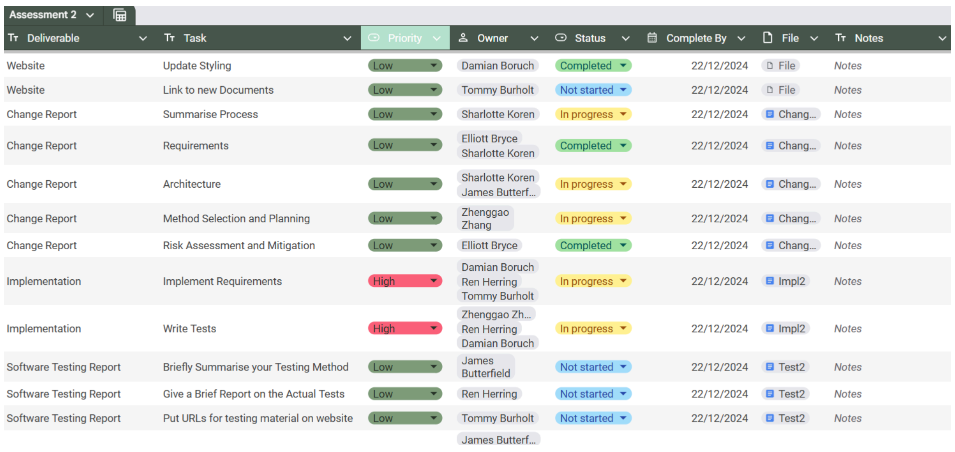
Zhenggao Zhang

Sharlotte Koren

# Our Approach to Managing Assessment 1 Deliverables

## Processes

Our team started by familiarising ourselves with the code and deliverables we inherited. We wanted to understand the original team's thought process and how their approach aligned with ours. This helped us identify similarities, pinpoint areas that needed improvement, and decide how to proceed. To stay organised and ensure cohesion, we assigned responsibilities based on our roles from Assessment 1, so everyone knew what they were accountable for. We made sure to work in an agile manner, allowing us to receive consistent feedback from each other. We documented each assignment in an Excel sheet, which made it easy to keep track of tasks and progress, as shown below.



Assessment 2

| Deliverable | Task | Priority | Owner | Status | Complete By | File | Notes |
|---|---|---|---|---|---|---|---|
| Website | Update Styling | Low | Damian Boruch | Completed | 22/12/2024 | File | Notes |
| Website | Link to new Documents | Low | Tommy Burholt | Not started | 22/12/2024 | File | Notes |
| Change Report | Summarise Process | Low | Sharlotte Koren | In progress | 22/12/2024 | Chang... | Notes |
| Change Report | Requirements | Low | Elliott Bryce, Sharlotte Koren | Completed | 22/12/2024 | Chang... | Notes |
| Change Report | Architecture | Low | Sharlotte Koren, James Butterf... | In progress | 22/12/2024 | Chang... | Notes |
| Change Report | Method Selection and Planning | Low | Zhenggao Zhang | In progress | 22/12/2024 | Chang... | Notes |
| Change Report | Risk Assessment and Mitigation | Low | Elliott Bryce | Completed | 22/12/2024 | Chang... | Notes |
| Implementation | Implement Requirements | High | Damian Boruch, Ren Herring, Tommy Burholt | In progress | 22/12/2024 | Impl2 | Notes |
| Implementation | Write Tests | High | Zhenggao Zh..., Ren Herring, Damian Boruch | In progress | 22/12/2024 | Impl2 | Notes |
| Software Testing Report | Briefly Summarise your Testing Method | Low | James Butterfield | Not started | 22/12/2024 | Test2 | Notes |
| Software Testing Report | Give a Brief Report on the Actual Tests | Low | Ren Herring | Not started | 22/12/2024 | Test2 | Notes |
| Software Testing Report | Put URLs for testing material on website | Low | Tommy Burholt, James Butterf... | Not started | 22/12/2024 | Test2 | Notes |

**Figure 1**: team management Excel sheet

## Tools

For tools, we used GitHub as our main version control system, ensuring all changes were tracked and managed efficiently. Excel was a key tool for organising roles and tasks, while WhatsApp helped us stay in touch and communicate quickly when needed. To collaborate on deliverables, we used Google Docs and maintained a shared Google Drive to store and access all files easily.

To review and manage changes, we used pull requests on GitHub. These allowed the team to check each other's work and ensure everything met our standards before being merged. Regular team meetings in person, provided an opportunity to discuss progress, give feedback, and address any issues.

## Conventions

We also made an effort to follow consistent conventions for both coding and documentation. This ensured the code was intuitive and easy to understand, not just for us but for anyone working on it in the future. Our commit messages were short, clear, and descriptive, which made tracking changes simple and transparent.

# Changing Assessment 1’s Deliverables

## Requirements

One significant change we made to the Requirements document was simplifying and combining two “Single Statements of Need” into one. The original team had two separate statements, when the whole purpose of this was to only have one. To make this clearer, we merged them into one: “The game shall enable players to create and manage their own interactive university campus through construction of buildings, responses to events, and management of budget, in order to maximise student satisfaction.”

Another key update we made was removing references to Assessment 1-specific needs that were no longer relevant. For example, the original requirement introduction stated: “The rest of our cohort is a primary stakeholder, as their interest in the project is potentially choosing it for use in the second assessment, based on their judgement of how promising the project is”. Since the project has now been chosen, this statement’s relevance is no longer applicable, we removed it to make sure that the requirements are focused purely on the needs of the customer rather than assessment-specific objectives.

We also felt that some user requirements were better suited as functional requirements, encapsulated under a single user requirement. This change will aid us in having easily testable requirements. For example, the user requirements we inherited contained UR_MUTE (be able to mute game music) and UR_SIZE_CHANGE (be able to change the size of the game window). We decided to refactor these requirements into FR_SIZE_CHANGE, FR_MUTE, FR_MUSIC (game should have music) and place them under the banner of UR_UX (The system shall have a pleasant user experience) as all these requirements contribute to a pleasant user experience. This opened up better requirement testing opportunities later in the project i.e. it is easy to test that the music is mutable.

The main addition we made to this document was adding new requirements specific to Assessment 2, ensuring that the game meets the new expanded brief. Key additions include UR_BUILD_LIMITS (and corresponding FR_MAP_LIMITS, FR_MAP_COLLISIONS, FR_BUILD_COLLISIONS) which helps to define constraints on the map and on building placement, and UR_SATISFACTION and FR_SAT_CALCULATION which helps to track and calculate player satisfaction based on their gameplay decisions. Minimal updates were made to UR_PLACE_BUILD_RLX, specifying that at least two types of recreational buildings must be included. We added FR_EVENTS which adds events to the game and, UR_LEADERBOARD, FR_LEADERBOARD, UR_ACHIEVEMENTS and FR_ACHIEVEMENTS, which bring the goal-oriented and competitive elements to the game. These updates ensure that the requirements now fully reflect the expanded deliverables and scope for Assessment 2 while still aligning with the project’s goals.

We also added requirements that were needed for Assessment 1 but were not included, for example the need to have a pause menu. The previous team did not have the ability to pause, so therefore we added in FR_PAUSE to ensure that this gets done.

The last change we made was filling in several missing requirements to fill the gaps and ensure the requirements specification is complete. This included NFR_GAME_END, which now states clear conditions for how the game ends, NFR_OPERABILITY which specifies the game's ease of use and FR_MONEY_DISPLAY, which clarifies how the game displays the player's financial resources. These additions were necessary to ensure we have fully captured the functional and non-functional requirements of the game, which creates a stronger foundation for implementation, architecture and testing.

Original Requirements deliverable: Req1
Updated Requirements deliverable: Req1.docx

## Architecture

One of the changes that we have made to the architecture document is an update to the constructor of the class “Relaxation” from the Building UML diagram. Currently, to be able to expand this class and have multiple different types of relaxation buildings, we would have had to create a new class for each one due to the fact that the texture file path was hard-coded into the class. To improve this implementation, we changed the static String variable “defaultImage” to a String array, allowing for multiple texture file paths in the same variable. As a result of this, we also had to update the Relaxation constructor to take the parameter “textureIndex” which allows us to specify the texture file path we want to use, ultimately enabling us to have multiple different types of relaxation buildings.

Another change that we have made to the architecture document is to include behavioural diagrams. Whilst the existing documentation had a structural diagram (class diagram), a behavioural diagram had not been included which we felt should have been in order to show how certain parts of the program behave to user interaction. The behavioural diagram that we decided to use was a use case diagram. This was the most suitable diagram to show how the game should respond to certain user interactions, more specifically, the screens that should be displayed when a user presses a button, or the action that should take place when the user interacts with the screen with a building on their mouse (placing a building). Based on the existing architectural documentation and access to running the game in its current state, we were able to develop an initial use case diagram that showed the basic structure and flow from the start screen, to placing buildings, to the timer running out. We believe that this initial diagram met the criteria and requirements for Assessment 1. Once we had created this initial use case diagram, we expanded on it to meet the requirements for Assessment 2. This included showing how the game responds to the user viewing the leaderboard, as well as the random events system.

We also decided to include another behaviour diagram, which was a sequence diagram. First of all, we designed an initial sequence diagram based on how the game ran when we took it over, as well as using their existing documentation to ensure the diagram was as accurate as possible. The reason we decided that a sequence diagram was necessary to include is because it shows a clear timeline of how the game progresses based on user interactions, such as what screens should be displayed when certain buttons are clicked. This is important as it gives us a clear understanding of how the game should be developed to ensure a smooth user experience, and that all user interactions behave in a way that the user would expect. Once we had designed the initial sequence diagram, we expanded on it to create the final sequence diagram which also included the requirements for Assessment 2. The key changes made here from the initial sequence diagram was including the event pop-up system and leaderboard. The event pop-up system shows that a user must respond to a random event once it is triggered, and depending on the outcome of the user’s response the game state is updated (positively or negatively). Similarly, the final sequence diagram also shows that when the user interacts with the leaderboard button, the leaderboard is displayed to them as well as the option to return to the start screen once they have finished viewing the leaderboard.

Another change we have made to the architecture document is a final class diagram representing all the changes we have made to their game’s code structure, as well as everything necessary to meet the requirements of Assessment 2. Within this final class diagram, we have removed any methods we deemed unnecessary, as well as restructured the way some of the classes were represented. For example, we have significantly simplified the previous group’s final building class diagram from Assessment 1, removing classes such as BuildingCounter being represented by the Building Class, and moving it to GameLogic instead. Now, the building class just contains the different types of buildings (such as accommodation, recreational, etc…). These changes to each class have allowed us to simplify the entire architectural structure of the game which as a result has made it easier to understand and much more intuitive. Furthermore, we have also added the leaderboard and events classes into the class diagram in order to meet the requirements of Assessment 2.

A minor change that we have made to the architecture document is a general clean-up. Originally, we felt that the documentation was a little difficult to follow and unorganised, therefore we have added proper headings and subheadings to separate different sections of the architecture. Overall, this was a great benefit to us as it made their architecture documentation more readable.

We have also linked the final class diagram on our website here.

Original Architecture deliverable: Arch1
Updated Architecture deliverable: Arch1.docx

## Method Selection and Planning

For the first section “Outline and Justification of Teams Software Engineering Methods”, one of the key additions we made was mentioning that other software tools were considered, such as Trello, before deciding on GitHub for task management. This shows that alternatives were considered before settling on the most suitable tool for the project. Including this information justifies why GitHub was the best fit for the team, with its great integration with the code and its ability to streamline collaboration.

We also incorporated information specific to Assessment 2 to reflect how our methods evolved. For example, we explained how we continued using GitHub for its existing setup and efficiency, making it easier for us to adapt to the new code without starting from scratch. In addition, we realised that while the previous team used Discord for communication, we opted to use our existing WhatsApp group for convenience. This change was based on what worked best for our team dynamics and ensured our communication was seamless without requiring everyone to switch to a new platform.

For the second section “Outline To Teams Organisation”, we decided that no changes were necessary to the original team’s approach. Their method of dividing the workload among team members, with multiple people assigned to each section, seemed well thought out and reasonable. It aligned with our own approach in Assessment 1, so we decided to carry it forward with no changes. Increasing the bus factor by having multiple members working on each section of the project was very effective in minimising unexpected delays or absences.

However, for Assessment 2, we added details about how our team was organised for this section of the project. We added a new section explaining how tasks were assigned to team members based on their previous contributions in Assessment 1. This decision was driven by practicality, as it allowed people to continue working on areas they were already familiar with, reducing the time required to get familiar with the content. Additionally, we included an Excel snapshot of our workflow progress tracking system, which shows all the key tasks, the team members responsible for each, and the corresponding deadlines. This shows clearly how we built upon the original team’s framework while adjusting it to meet the specific needs of Assessment 2.

For the third section “Systematic Plan For Project”, we found that the original content was good and didn’t require many changes. The plan laid out the key tasks, their dependencies, and the general timeline in a logical way, which worked well for the project. The only adjustment we made was to improve the readability of the document by structuring the weekly updates with clear headings for each week. This made the document easier to follow, ensuring that progress for each stage of the project was more visible. To reflect our work in Assessment 2, we added more Gantt charts for the general overview as well as on a weekly basis. These charts provided a clear snapshot of how we were progressing.

Original Method Selection and Planning deliverable: Plan1
Updated Method Selection and Planning deliverable: Plan1.docx

## Risk Assessment and Mitigation

We made minimal changes to the justification of the risk management process, as it followed a very similar process to the process that our group used in assessment 1. There were some minor changes needed to fix incorrect grammar and improve the flow of sentences. Overall we were happy with the process they outlined and thought it fit the project well.

In the risk register, we looked to remove risks that were no longer relevant to the project. The majority of the risks were well thought out and had good mitigation strategies, likelihood and severity levels. Due to this, we only removed one risk as it was a duplicate of two other risks. We removed risk 12, “Unexpected bugs in the codebase”, as the register also contains “Unexpected (minor) bugs in the codebase” (risk 7) and “Unexpected (major) bugs in the codebase” (risk 8).

We then changed the risk owners to match members of our team. We made sure that all members were happy with updating and tracking the risks that they were assigned.

The final thing we had to think about was whether there are any new risks that we might encounter in assessment 2 that are not already included in the risk register. The risk register that we inherited already includes many risks that are relevant to assessment 2. A good example of this is risk 14, “Requirement Changes”, which already accounts for any changes to requirements that the consumer might send us. The team who wrote the Risk Assessment did a good job and made our work rather easy when changing the risk register as there was not much to do.

In the end, we only added one extra risk, reusing the risk ID 12 as we had previously removed the old one. We felt that the risk assessment was not acknowledging possible conflict between group members so we added ‘Arguments/conflict between team members’. We implemented a basic mitigation strategy for this risk and decided it was best to have all team members as the owner of this risk, with every person looking out for tension between team members - whether it’s an argument involving themselves or a dispute between other members.

Original Risk Assessment and Mitigation deliverable: Risk1
Updated Risk Assessment and Mitigation deliverable: Risk1.docx