How we made changes to the documents

When deciding what needs to be altered on the original documents, two team members reviewed each document and both separately created lists of changes in accordance to the assessment brief. These lists were then compared and then a finalised set of changes was created and agreed upon on a shared document. This is an efficient way to plan as it was done within the first week which means for the following weeks of the project we could all have a clear understanding of how the documents should develop. Also, as there are two people reviewing each document then we could reduce the risk of unintentional bias and have the changes that are suggested be unbiased and necessary.

We then started allocating who will alter each document with every document having two team members on at one time so they can crosscheck what was being added to the document. Once anything was added to the document we added it to a list of things that needed to be included within the change document so that when it came time to write the actual change document we had a readily available reference to everything that had been changed. If at any point while writing the change report you wanted to see exactly what has been added we could use the version history feature on google docs which allows you to go back to see when they added it and what exactly they added. This was very helpful for our team as we also used google docs to make the changes and thus having these two tools be on the same platform was really easy and saved a lot of time.

While editing the documents we tried hard to maintain the original conventions of the documents and only changed stuff if we deemed it necessary. For example, we kept the font size the same and did all the headings the same way. This is easier for us all to upkeep the documents as we won't have to change the font if we all used the original font. We also tried to keep the structures of the documents similar to how they originally were.

<u>Document links</u>

Requirements:
Assessment 1:
https://mforrow310.github.io/glitchgobblers.github.io/Docs/Requirements.pdf
Assessment 2:
https://mforrow310.github.io/glitchgobblers.github.io/Docs/Requirements2.pdf

Architecture:
Assessment 1:
https://mforrow310.github.io/glitchgobblers.github.io/Docs/Architecture.pdf
Assessment 2:
https://mforrow310.github.io/glitchgobblers.github.io/Docs/Architecture2.pdf

Method selection and planning:
Assessment 1:
https://mforrow310.github.io/glitchgobblers.github.io/Docs/Method%20selection%20and%20planning.pdf
Assessment 2:
https://mforrow310.github.io/glitchgobblers.github.io/Docs/Method%20selection%20and%20planning2.pdf

Risk:
Assessment 1:
https://mforrow310.github.io/glitchgobblers.github.io/Docs/Risk%20assessment%20and%20mitigation.pdf
Assessment 2:
https://mforrow310.github.io/glitchgobblers.github.io/Docs/Risk.pdf

i) Requirements

For the requirements document, the majority of the requirements remained unchanged as they were mostly relevant to the game even though we added functionality and events. However as all of the requirements didn't fit the updated game we had to add new requirements and even changed a few existing elements so they would correspond to the game in its current state. However, to preserve the work that the team before us made, all of these changes were completed on a copied version of the table at the end of the document.

One of the key requirements that were added to the table was the requirements that related to the Dean as the Dean was not previously implemented from the other group, they had no need to add any requirements for it other than the user requirements. This is because it was not implemented and thus the functional requirements were not created as there was no functionality regarding the Dean within the game.
The original requirement UR_DEAN was marked as will not have and no implementation of it was planned. Once changes were made an implementation for the Dean requirement was written that corresponds with the part the Dean plays in the game and when the Dean is triggered. The priority of the dean was also changed from Will not have to Must have. New validation criteria is included in the functional requirements table including spawning and chase behaviour under the functional requirement FR_DEAN_AI. The Dean provides the game's main obstacle and creates a system to lose points. Adding this ensures the game includes a required challenge and a clear condition to lose points.

Another major change to the requirements was the addition of the completely random check-in code and the requirements that are needed to implement it. As the previous game did not have a check-in code event we needed to create some requirements that define its implementation and its priority. To complete this task we added a user requirement UR_CHECKIN with the priority must have and a functional requirement FR_CODE with a validation criteria which links to its implementation within the game. This is a major component of the game as it is what triggers the Dean if it is inputted incorrectly otherwise the player gains points from inputting the correct code.

Added a new user requirement UR_LEADERBOARD which takes the top five scores from players after they have completed their run and entered their name. The previous team's implementation included partial leaderboard logic. To fully integrate and justify the feature, requirements were added for traceability and validation. To implement this we had to update and improve some of the validation criteria in some of the requirements (UR_SCORE and UR_EVENT_COUNTER). Ensuring the requirement is specific and measurable, enabling consistent verification.

Also added a new functional requirement FR_ACHIEVEMENTS which was not previously present to reflect the new game feature of the achievements. Added to define achievement-triggering logic and optional score modifications. This requirement must be present to ensure the achievement system is properly defined and can be checked during testing.

Two new non-functional requirements were added, one of which was NFR_USABILITY which ensures new user interface elements remain clear and accessible to players. This requirement ensures that all new interface elements are easy to read and interact with, supporting a smoother gameplay experience. The other requirement added for this section was NFR_DATA_PERSISTENCE which adds support for saving leaderboard entries between game sessions. As scores must be saved so the leaderboard is consistent when the game is closed and restarted even on a different machine.

ii) Architecture

<u>Document changes</u>

The introduction to the documents remains the same for our team since we will also be using LibGDX and IntelliJ for the second part of the assessment. We are also using GitHub in the same way as the previous team, using it to manage branches, track changes and also to plan our project through the issues section.

Our team updated the architecture diagram to include all the new methods and variables used in our code. We also added 2 new classes that were needed for this part of the project. The class Dean and GameUI were added to fulfill the requirement for a Dean chasing the player and the GameUI class was added to improve clarity of MapManager.

The structural architecture of the game has stayed the same, so the diagram was not edited in our documentation. This stays the same since the components of the game such as the entities and systems are still interacting with each other as they were in the previous part of the assessment.

The Process of Designing Architecture stays the same as before for each of the iterations below:
Figure 2a diagram - The player movement and collision chart stays the same since the input W/A/S/D is still used for our player. Collisions also happen the same as in the previous part with the player checking if the tile it will move into is safe and moving if it is or stop moving if it isn't.
Figure 2b diagram - The event interaction and counter diagram is the same since we are still using the same system where the player triggers an event and then the event triggers the EventManager.event() which increments the counter by 1. It also triggers the modifyScore.MapManager() which increases the score when an event is triggered.
Figure 2c diagram - The timer still acts as it did in the previous assessment, as does the victory/gameover screens. The only change is that the Victory screen now includes a place to input your name so your score is saved on the leaderboard. However the diagram did not need to be changed since this is part of the Victory screen which is present already on the diagram.
Figure 2d diagram - We continued to use ESC as the key which pauses the game, and the system has not changed since the last assessment so this diagram remains unchanged.
We updated the Table to show how we had implemented the requirements. " rows were added, a row for the FR_DEAN_AI and FR_ACHIEVEMENTS requirement. Both of these requirements are demonstrated in the diagram 3a and both interact with MapManager. FR_ACHIEVEMENTS uses the LeaderboardScreen element as well.
We also updated the FR_MAIN_MENU row to include what now happens when the player wins the game. Now they are taken to a screen where they can input their name to be added to the leaderboard rather than just showing the victory screen.
In the FR_INTERACT_EVENTS row we changed the number of events the player needs to interact with in the description to reflect the new requirements for this half of the assessment.
In the table where MainMenuScreen is referenced in the previous table, it has been changed to SplashScreen since this is the main screen now.

Code changes

We made changes to our code to implement the new requirements for the second assessment. The previous team had hardcoded some of their project, so our first action was to fix this. We changed it so we use Tiled to store positions of interactables including the crated and keys.
To clean up MapManager (which is the main class for the game) we moved all the code relating to UI into its own class GameUI which MapManager uses. This made the MapManager code easier to read.Another way we cleaned the code up is by removing redundant code and fixed some uses of "magic strings" by replacing them with enums.
We made a new class Dean to fulfill the requirement for a dean chasing the player. In the Dean class we added code to spawn and unspawn the dean at the correct time. We also added code to check for collisions with the dean, because the number of times the player collided with the Dean was needed to assess whether the player had won the achievement. Code was also added to update the boolean values for achievements so the achievements could be displayed on the leaderboard on the end.

We added some new events to the game including the player collecting a check in Codes from the noticeboard and having to submit them to collect the stones. To implement this we added a showCode UI screen and methods to generate random check in codes.
We added more helper methods to Event manager and we updated the object check method so it now also checks for the new objects we added. Input handling is now improved, showing cleaner switching between UI pop ups and the game.

iii)Method selection and planning

During the entirety of the second part of the assignment we had to make sure that the plan document was kept up to date and included all the activities undertaken by our team. A lot of the previous team's work was not altered as it provided a step by step guide as to how they approached the first part and thus it was necessary to keep it in accordance to having a completed plan of the entire project for the stakeholders.

The most important thing that needed to be done was to add our team's plan to the weekly plan table which documents what tasks were completed, the dates and the people involved. This was extremely important to monitor what every team member was doing throughout the project as well as keeping a physical record of the work completed, a description of the tasks, the priority and the dependencies these tasks have. For this section we kept the same structure and format that the previous team used to keep it uniform for the report.

We also added our team organisation to the document in the form of a table. For this we split the project into 5 sections (documentation, coding ,testing, design and website) and wrote down who was responsible for each section. As well as adding how we as a team communicated outside of the scheduled meetings and how we as a group delegated roles. The previous team used a more rigid structure of team organisation with each person having a fixed role within the team (i.e. Secretary, Project lead). However, we decided against this pretty early on within taking over the project as it would require us to create new roles to fit the expanding criteria and requirements for the project. So, instead we allocated people roles based on what they were comfortable with and always made sure to have at least 2 people on each task so they can check with the other team members and brainstorm with them.

We had to add developments on two sections of the previous plan document one of which being the methodology used for our team during the project. As we used a similar methodology as the previous team but had a different reason to justify it we decided to write a section about why we chose to use agile methodology. Another development was how the plan changed throughout the project as it only had the previous teams changes and we had to add our own at the end of the project. For this we had to look at what had changed throughout the entirety of our development cycle and write it up.

iv)Risk assessment and mitigation

Before we began the next stage of the project we had to edit the risk assessment and mitigation document to make sure it was up to date. Our risk management process was the same as the previous teams because we felt it was a good way to work out and assess potential risks.

We copied the original risks into another table where we could write our own mitigations for these risks, because the way we are planning on mitigating the risks is different to the previous team. The mitigations for the second assessment stayed quite similar apart from the following risks. The mitigations for R2 and R6 were changed to mention that we were researching the other teams software rather than researching to find software to use.
The biggest difference is for risk R7 our mitigation includes having a member of our group who oversees all the documentation in our project, monitoring its progress and completion. This is different to the previous teams approach where they summarised in a document the documentation that had been completed.
In R9 our mitigation was the same apart from we added the extra precaution of having 2 team members to each document and task to make sure they were getting completed on time and at the level we expected.
The final notable change in our mitigations table for assessment 2 was in R10 where we added that more rigorous testing would be done in this assessment which can help mitigate crashes even more.
.
We also had to make sure that the likelihood and severity were correct for the next part of the assessment. We decided the severity of the risks would be the same as for the previous assessment but some of the likelihoods were changed:

- R2 (Medium -> Low) - The likelihood of using incompatible software is lower for this part of the assessment because the software has been researched and tested by the previous team and we will continue to use their software.
- R13 (High -> Medium) - The likelihood of insufficient code testing is reduced because we are completing a large amount of testing in this section of the assessment.

In our updated risk assessment table we had to assign an owner from our group to each of the risks to make sure that we had someone to oversee each risk so it can be prevented.

We added a new risk (R15) to the risk table. Since the assessment is due just after the christmas holidays we wanted to make sure we had mitigations so us being virtual does not affect whether or not we submit in time. These mitigations include moving the majority of tasks to the last couple of weeks before term ends and ensuring good communication is happening over the holidays. We assessed the likelihood for this to be high because it is very likely communication may reduce during the festive period, but the severity at a medium because with proper mitigation it shouldn’t have a massive impact.