Date: 13/01/25

UniTycoon

CHANGE REPORT

Group 6

TEAM6 Game Studios

Work Undertaken by:

Thomas Koukouris
Adam Khan
Oliver Herron
Sam Jordan
Nathan Hopper
Fergus Irvine

Summary of Processes, Tools and Conventions

To keep track of changes to the Assessment 1 deliverables we created new documents for each deliverable and completed them as we worked on our project. For example a new requirements document was created for the new requirements that had to be implemented into the project.

Once a new version of the document was completed the team member working on the document would compare the new document with the old document. Any changes made to the original document would then be catalogued, either within a table or explained with bullet points/diagrams. We made sure that the documents would be compared multiple times and all changes were recorded and reasons provided. For any situations where there were no changes to the original deliverables we have made sure to explain why.

New branches in the GitHub repository inherited from Team 9 were created to separate our coding files from theirs. We then merged their most up-to-date branch with our newly created ones. Thus we had access to the old version of the code at any given time without overwriting their work allowing us to revert back to the old version if our changes didn't work. Separate repository branches were also created for the team members who worked on the coding implementation. Every member uploaded to their branch only and once checked by other team members it was merged to a main default branch ensuring proper version control and mitigating the risk of loss of work. The website repository was worked on by one member and was kept entirely separate from the implementation repository but under the same organisation in GitHub.

❖ i. Requirements

Changes to Requirements

In this page we are outlining the changes we found necessary to male to the requirements document from Team 9. We decided to present the changes in a tabular format for ease of readability. Each change has been titled and a description has been added to specify exactly what the change was and why we thought it was necessary.

Original Document

The original requirements document was overall well-produced but a lot of the requirements were repeated and incorrectly placed. Some missing requirements were added along with further requirements to meet the updated product brief.
The original requirements document (Team 9) can be accessed here:
https://team6unissim.github.io/Assessment2Website/docs/Req1.pdf

#	Changes	Description
1	Requirements presentation & spelling.	Coloured table cells, capitalised titles of different requirement types, modified page margins, corrected some spelling errors. Added a short description of each requirement type next to their title.
2	UR_SCORE	This user requirement was moved to functional requirements.
3	UR_PAUSE	Priority was changed from 'should' to 'shall' as the product brief clearly states this as a game requirement.

4	UR_YEARLY_REPORTS	Changed priority from ‘should’ to ‘may’ as this is not really required by the product brief.
5	UR_END_OF_GAME_REPORT	This initially indicated that the user shall be given a report of how they performed over the course of the game and that the game shall save player’s scores to compare them with the next games. Moreover, the title of this requirement indicates that this is about the ‘end of the game’ but the requirement description never mentions an end report of player performance. The client during our first meeting had specifically mentioned “no progression or rewards or storing things as it is a 5 minute game”. However, both parts of the requirement mentioned here were later added in product brief 2 in the form of a **leaderboard** and **achievements**. I decided to change this completely by only focusing on the end of the game report and changing to ‘should’ priority. The other parts were removed as they will be added separately.
6	UR_EVENTS	This initially stated the game shall only have random events but the brief clearly states there should be unplanned (random) and planned events, therefore this was revised.
7	UR_DEPENDENT_EVENTS	This was combined into the UR_EVENTS requirement.
8	UR_PERFORMANCE	This was placed incorrectly under user requirements as it describes how the system should work - should be in NFRs. Also, it was aso repeated in non-functional requirements. Therefore, i removed from here.
9	UR_PLAYABILITY	Similarly, I moved this from user to non-functional requirements and combined it with FR_ACCESSIBILITY.
10	UR_ISOMETRIC	Similarly, I moved this from user to non-functional requirements.
11	UR_MAP	This one initially included details that should not be under user requirements. I solved this by splitting it into two distinct requirements, the first half stayed in user requirements but the second half was moved to FR_TILEABLE_MAP.
12	UR_BUILDINGS	Renamed it to UR_ADD_BUILDINGS. I expanded on this to match the product brief requirements of minimum amount for each building required. Removing buildings was not specifically requested by the client and so was added as a ‘may’ user requirement.
13	UR_BUILDING_LOCATION	This requirement talks about how the system handles the building placement and shouldn’t be in user requirements. It does exist, however, in functional requirements already so this is redundant - removed.
14	UR_BUILDING_VARIANTS	This was never mentioned in the brief. I touched on this on the UR_BUILDINGS requirement, therefore this was removed altogether.
15	UR_JOVIAL	This originally was not an objective requirement and was modified to UR_REPLAY.
16	UR_SATISFACTION	This was added as it was missing from original user requirements.
17	FR_PAUSE	This functional requirement was overly descriptive, thus it was made more concise.

18	FR_TIMING	This was reworded.
19	FR_REAL_TIME	This was redundant as it is already mentioned in user requirements, therefore it was removed from here.
20	FR_INPUT	This was initially mentioned in UR_PLAYABILITY which I moved to non-functional requirements, it was removed from here.
21	FR_ACCESSIBILITY	This was moved to non-functional requirements from functional ones as it describes **how** the system shall be.
22	FR_MAP_FEATURES	This directly affects the user so it was also placed in user requirements as UR_TERRAIN and was refined here and renamed as FR_TERRAIN.
23	FR_CONSTRUCTION	This was reworded.
24	FR_DESTRUCTION	This is a user requirement and so was moved there from here and given a ‘may’ priority. However, a functional requirement was also added specifying what the system should do.
25	FR_BUILDING_TYPES	Was removed as already mentioned in UR_ADD_BUILDINGS.
26	FR_BUILDING_RELATIONSHIP	Combined FR_BUILDING_RELATIONSHIP and FR_SATISFACTION_ALGORITHM into one FR_STUDENT_SATISFACTION_BUILDINGS.
27	FR_SATISFACTION_ALGORITHM	This was changed and renamed to FR_STUDENT_SATISFACTION_EVENTS
28	FR_PLANNED_EVENTS	This was renamed to better describe its function.
29	FR_DEPENDENT_EVENTS	This was reworded and renamed to better match the user requirement UR_EVENTS
30	FR_MUSIC	This was renamed to FR_SOUND and UR_SOUND was added in user requirements to base this one on it
31	FR_UPGRADES	This was removed as not specified in the brief.
32	NFR link to UR	A few NFRs links to URs were changed to UR_UX as a lot of the previously linked requirements had been removed/altered.
33	NFR_INTUITIVE	This was renamed to NFR_TUTORIAL and the description changed accordingly. This requirement was already mentioned in the UR_UX and so was redundant here but a tutorial was never mentioned.
34	Further additions of NFRs	Added NFR_ACCESSIBILITY, NFR_COMPATIBILITY and NFR_SYS_REPONSE_SPEED to clarify further system requirements.
35	UR_LEADERBOARD	This user requirement was created for the new **leaderboard** requirement that had to be implemented with ‘shall’ priority.
36	UR_ACHIEVEMENTS	This user requirement was created for the new **achievement** requirement that had to be implemented with ‘shall’ priority.

37	FR_LEADERBOARD	This was added to explain what the system must do to facilitate the UR_LEADERBOARD requirement introduced by the updated brief.
38	FR_ACHIEVEMENTS	This was added to explain what the system must do to facilitate the UR_ACHIEVEMENTS requirement introduced by the updated brief.

Conclusion

A lot of requirements were misplaced and so had to be moved within the table. Some were renamed or had their descriptions shortened or modified. This created major conflicts as there is a strong linkage between types of requirements. This meant further alterations had to be done to make sure everything was referenced correctly. Moreover, a couple new user requirements were introduced UR_LEADERBOARD and UR_ACHIEVEMENTS to meet the updated brief requirements. This also introduced new functional requirements FR_LEADERBOARD and FR_ACHIEVEMENTS that explained how the system would tackle these user requirements.

❖ ii. Architecture

Changes to Architecture

The architecture document was updated with new UML Class diagrams to reflect the changes that have been made to the architecture to meet the new requirements.

The original document of Team 9's Architecture can be found here:
https://eng1-cohort2-group9.github.io/Website/docs/Arch1.pdf

❖ iii. Method Selection and Planning

Changes to Method Selection and Planning
The Changes have been shown in the table below each section's header, as well as attached reasons for the changes. The table format allows for clarity and readability for each change.

Original Document
The Original document of Team 9's Method Selection and Planning can be found here:
https://eng1-cohort2-group9.github.io/Website/docs/Plan1.pdf

Outline and justification of software engineering methods:

#	Change	Reason
1	Communication	For our method of communication within our group we decided to use WhatsApp. This allows fast, reliable communication as it is a mobile app. We primarily used WhatsApp to arrange meetings, with communication about the project taking place

		within the meetings, however we would be able to discuss small parts of the project quickly with WhatsApp
2	File Transfer	For transferring files we used Google Drive, as it is easy to use and keep organised. This allowed us to all be able to access any files at all times
3	IDE	For our IDE we chose to use VScode as all members of the group are familiar with it and know how to use it. For our Code sharing we used Github as some of our group were familiar with it, and due to its wide usage other members could easily get used to how it works.
4	Task Planning	To keep track of tasks that needed to be done we used Trello. This let us set different tasks for each deliverable and we could all see which tasks were being actively worked on and which were already completed
5	Creating Diagrams	For creating various diagrams such as gantt charts we used LucidChart as it was easy to use for what it was needed for

Approach to team organisation

#	Change	Reason
1	Engineering Method	We still used an agile method however we divided the deliverables at the start of the project and reallocated people as tasks got completed, this allowed us to ensure we each understood our section of the project, and when complete we could move to other deliverables
2	Meetings	Rather than just meeting every Thursday we meet on Mondays and Thursdays, sometimes the days may change due to availability

❖ iv. Risk Assessment and Mitigation

Introduction

The updated risk register addresses gaps identified in Team 9's original document and aligns the risk management process with the evolving scope of the project. This report gives a thorough account and an argumentation between the changes introduced, in terms of new, modified, and removed risks. The changes are intended to enhance project robustness, guarantee compliance with specifications and manage the risks related to inherited deliverables and new project stages.

Original Document

The Original document of Team 9's Risk Assessment and Mitigation can be found here: https://eng1-cohort2-group9.github.io/Website/docs/Risk1.pdf

Updated Document
The updated document, excluding introduction, of the Risk Assessment and Mitigation can be found here:
https://team6unissim.github.io/Assessment2Website/docs/Risk2.pdf

1. Added Risks

- **Ethical compliance and legal issues:** Added risks associated with the use of inherited third-party libraries and external data sources based on ethical and legal compliance.
    - ***Justification:*** Legal compliance is essential to prevent future lawsuits or project failure. The original document lacked this consideration, which is especially important in the current phase where inherited assets may pose unforeseen risks.
- **Scalability of architecture:** Presented a risk with regard to the scalability of the inherited design.
    - ***Justification:*** When new requirements are added, the scalability of the system has to be assessed. Failure to consider scalability from the beginning could result in considerable refactoring at later stages.
- **Delayed feedback on prototype:** Added a risk for delays in stakeholder feedback
    - ***Justification:*** Stakeholder feedback is paramount to ensure the project delivers in response to user needs. Without timely feedback, the development process may deviate from expectations, leading to rework.
- **New features conflicting with architecture:** Incorporated risks for future conflictions when planning to enhance the architecture
    - ***Justification:*** Development of new features should be carefully controlled so as to prevent architectural mismatches that may contribute to technical debt, or performance degradation.

2. Updated Risks

- **Deadlines:** Revised the likelihood and mitigation strategy for missed deadlines
    - ***Justification:*** Due to the rise in complexity of the Assessment 2, deadlines are more likely to be postponed. Mitigational strategies including reassessing priorities and reprogramming resources have been implemented.
- **Coding style consistency:** Updated mitigation strategies to include mandatory adoption of a style guide and regular code reviews
    - ***Justification:*** Good coding styles are critical to collaborative projects, and in particular when potentially combining inherited code. Regular reviews ensure adherence to agreed standards.
- **Testing:** Improved the mitigation strategy for poor testing to include both unit and playtesting
    - ***Justification:*** Testing is a major factor in ensuring code quality, so refining the risk register this way should help emphasise this.

3. Removed Risks

- **IDE Compatibility:** Removed the risk related to IDE compatibility
    - ***Justification:*** This risk was already mitigated in assessment 1 by meeting a requirement for IDE compatibility which is no longer relevant

Summary of Changes
The changes were guided by the following:

Alignment with project requirements: The revised register ensures that risks represent the wider scope and complexity of the project.

Addressing inherited deliverables: Risks specific to the inherited deliverables, such as outdated libraries or unclear documentation, were prioritised.

Enhanced mitigation strategies: Each risk also contains clearly defined and implementable mitigation strategies along with the enhancement of the project's resilience to problems.

Conclusion

Through these enhancements, the risk management process is made robust, comprehensive, and responsive to the changing requirements of the project. The changes address both inherited and new risks, improving the likelihood of project success while adhering to the principles of effective software engineering.