Change report

Cohort 1, Group 6 - M6

Members:

Mir Baydemir
Adam Fraulo
Azib Hj Abu Akmar
Agata Pittarello
Esther Scanlon
Jazz Stubbins
Sonia Szetela

Introduction:

When choosing the project to inherit, the team conducted an initial analysis of the game's derivables, documentation and code. During this early review the team was able to identify overall inconsistencies between the documentation provided and the code itself as well as missing features and requirements. We then planned how to organise and split the work given each member's ability and the quantity of changes required to be made. This plan was also made in light of the new SSON, which was analysed and discussed with the whole team, and that led to making a quick first version of modified requirements.

A shared Google Drive was used to make all derivables changes and reviews. Using this tool it made it possible to track all document versions in each stage of the project and know who was responsible for every amendment made.
This tool also allowed members to comment on all derivables, suggest improvements or flag potential problems that needed a quick resolution, and also revert to previous versions of the documents if needed.
The updated derivables were given the same name as their first iteration with the "1" replaced by a "2". This clearly indicated that they were the modified documents while ensuring that everyone could easily recognise them.

GitHub was instead used for code related work. The previous team's repository was forked, made private and afterwards shared with every member of the team. Most members worked on local copies and afterwards pushed changes to the main branch to prevent merge conflicts. Some members instead preferred to work through dedicated branches. All the different changes and versions of the game were tracked using GitHub's version control features.

The documentation followed a javadoc style, building on the previous team's work and other comments were added to improve maintainability and readability.

Overall, using these tools has proven key to planning, making, keeping track of, and reviewing changes to the Assessment 1 deliverables, documentation and code we inherited from team 9.

I. Requirements

Original URL: https://eng1-group9.github.io/website/Req1.pdf
New URL: https://keybordkat.github.io/websiteGroup6/Req2.pdf

What changes have been made and why?:

Derivable and additional materials of team 9

The team 9 Requirement document acquired was a 6 page document that included the initial cover page and a final references page. In the website there were also two additional documents the USE CASES and the REQUIREMENTS REFERENCING SYSTEM.

The critical issues found

1) In the main document the team presented all requirements discursively without using the referencing system.
   The requirements were written more as notes describing the client's requests and considerations, and the reasoning behind the team's choices of design and architecture, than as clearly listed, technical and unambiguous requirements.
2) The use cases were referenced at the end of the very long document getting lost in all the pages. The use cases themselves are adequately done and were only slightly adjusted to better express the necessary requirements.
3) The Requirements Referencing System was not referenced in all the document. In the tables some cells were left with ellipsis indicating that the team did not complete them. Some of the requirements included were phrased improperly, incorrect or too specific. These were then removed or modified in the new documentation.

An example of an overly specific and incorrect "constraint requirement" that was removed is CR_TIMEFRAME:

CR_TIMEFRAME	A finished, fuly working first iteration of the game must be completed and tested by 10/11/2025	All core FR's implemented and tested by first assessment milestone: (10/11/2025).

An example of an incorrect and redundant "user requirement" that was modified was UR_SCORE:

UR_SCORE	The player 'wins' the game by escapeing the maze within the time and a score is given based on the time.	should

Which not only partially repeated the UR_ESCAPE requirement,

UR_ESCAPE	The player needs to be able to escape the maze within 5 minutes and fails otherwise	shall

but as well failed to mention that also the events (and achievements) influence the score. This was modified to the current UR_SCORE which resolves the issues mentioned and which is better specified in the FR_SCORE_CALC:

UR_SCORE	Once completed the game a score will be shown based on the in-game actions performed and the time elapsed.	shall

FR_SCORE_CALC	The system shall calculate the score based on the player's escape time, events and achievements completed. Only if the victory condition is reached the score will be saved and displayed in the ending screen.	**UR_SCORE**

Changes made

The new requirement document was changed as follows to solve 1), 2) and 3) issues:

1) The previous main document was compressed to form an introduction.
   The way the previous team elicited the requirements both from the SSON and the client meeting was slightly reformulated and compacted to make it more readable and clear.
   The discussion about the requirements was instead used as an example to show how, from the meeting and brief, the team was able to produce the various requirements.
   The requirements table was added after this introduction

2) A direct reference to the use case was added at the end of the introduction separating it from the preceding paragraph and including the URL directly in the document making it more clearly referenced. The use cases were slightly modified and added to the previous document and helped in the formulation and identification of the new requirements and requirements that needed to be modified.

3) The requirement table present in the website was then added to the main document where it was modified as specified in the previous paragraph (by modifying wording and by deleting excessively specific or incorrect requirements) .
   Afterwards, after performing an ulterior analysis of the brief and project and use cases, missing requirements were added.
   For example "non-functional requirements" for maintainability, extensibility and documentation were added.

NFR_MAINTAINABILITY	The game's code should be easy to understand, modify and maintain.	**UR_TECHNICAL**	The game's code should be structured clearly and modularly making it possible for other developers to understand, modify and

			maintain the system.
NFR_EXTENDABILITY	The game should be designed to support expansions.	**UR_TECHNICAL**	The game's architecture should support expanding the map or adding events and achievements with only minimal changes to existing code.
NFR_DOCUMENTATION	The game's code and project should have ample documentation to support them.	**UR_TECHNICAL**	The game's code shall include appropriate comments and the project's documentation shall be detailed enough to support future maintenance or expansions.

Other requirements were added as the SSON was modified by the client, for example requirements for the leaderboard, achievements had to be added.

UR_ACHIEVEMENTS	Through the game the player may achieve in game objectives that might alter the score.	shall
UR_LEADERBOARD	A leaderboard with the name and the top 5 scores will be shown to the player once completed the game.	shall

II. Architecture

Original URL: https://eng1-group9.github.io/website/Arch1.pdf
New URL:https://keybordkat.github.io/websiteGroup6/Arch2.pdf

What changes have been made and why?:

- Created new diagram to reflect the changes in made in implementation
- Updated any references to requirements in accordance with the changes made

1) To begin, quite a few of their architectural claims were incorrect (although legacy names were kept for clarity). Although it may look like an ECS at first, the system is actually a standard OOP inheritance hierarchy. There is no component separation, no true system operating on components (methods are instance methods), and uses traditional inheritance (extends vs composition in traditional ECS). However their diagrams actually showed OOP so those did not need to be changed, just their descriptions. After revisions the architecture style was decided to be “a monolithic game loop with OOP hierarchy and a helper system”. It is not entity based, it is map based.Some further references to ECS were removed.
2) Edited Figure 2 and its description to highlight the fact that Longboi, the scroll and potion are not true entities since they are revealed as themselves through the rendering system, and are therefore not part of the entity layer.
3) An EventStatus component does not exist. There is a tracked event state but it is just scattered around parts of Main.java. Removed this from the architectural document as including it is inaccurate.
4) They named movement as an example of a system (it is not in any of the systems, the systems just interact with each other as a result of movement) so this was removed from the systems description as it is inaccurate.
5) They do not actually have layer isolation. Their Main.java file causes a God Object Anti-pattern. This anti-pattern was documented so that Main.java is acknowledged as the central controller.
6) We added a new leaderboard system to create our leaderboard component because it is required by the new assessment brief.

III. Method Selection & Planning

Original URL: https://eng1-group9.github.io/website/Plan1.pdf

New URL: https://keybordkat.github.io/websiteGroup6/Plan2.pdf

What changes have been made and why?:

Method Selection:

- A paragraph is added explaining how Agile methodology applies to the Brownfield development context. This paragraph is added because Assessment 2 involves working with the inherited code and we needed a new paragraph explaining how the Agile approach adapted to this new situation.
- Specific XP practices that we adopted have been added with their explanation.(Pair Programming, Test-Driven Development, Refactoring).We added this because although the inherited document mentioned XP, it did not specify specific practices being followed. I wanted to be more explicit about practices that directly address Brownfield development challenges.

- Tools:
  I removed the Discord section from Communication Tools. This is because our team only used Whatsapp for communication.
  I added a new section called testing tools. In which I added Junit 5 for unit tests and Mockito. This is because assessment 2 requires automated testing.

Project Plan:

I replaced the other team's week 2-6 timeline with the week 7-13 timeline. This is because assessment 2 has a different timeline and different deliverables.
I replaced Group 9's Gantt chart with our own Gantt chart.
I added Screenshoot of the Google sheets diagram which shows who is responsible for which deliverables.

IV. Risk Assessment & Mitigation

Original URL: https://eng1-group9.github.io/website/Risk1.pdf
New URL: https://keybordkat.github.io/websiteGroup6/Risk2.pdf

What changes have been made and why?:

Derivable and additional materials of team 9

The team 9 Risk Assessment and Management document acquired was a 6 page document that included the initial cover page and a final black references page.

The critical issues found

1) The risk register was not colour coded.
2) A “Monitoring” section was included. This section reduced the register readability and was unnecessary as the “Mitigation and Avoidance” section already provides all the relevant information needed.
3) Some risks were owned by multiple people in the team.
4) Most risks were allocated H/H M/H H/M severities, which are too drastic considering that this is a small project in a learning environment.
5) Estimation risks, risks on the game itself and two other risks involving the technology and project were missing.

Changes made

The new Risk Assessment and Mitigation document was changed as follows to solve 1), 2), 3), 4) and 5) issues:

1) The register is now colour coded and a simple reading key has been added to the “Risk Analysis” introduction section.
2) The "Monitoring" section was eliminated and its contents were integrated in the “Mitigation and Avoidance” section where needed.
3) Each risk is now owned by a singular team member of Group 6.
4) All risk severity was re-assessed and more balanced severities and likelihoods were assigned.
5) Estimation risks, risks on the game itself and two other risks involving the technology and project were added. In particular two new categories, for identification purposes, were created: G standing for the product(game) and E standing for estimation. The risks added were: TP4, T5, G1, G2, G3, E1, E2, E3.

The introduction of the document was left unchanged as already well done. The only changes were the removal of the “Monotiring” section description and the addition of the colour coding key.

V. Implementation

Original URL: https://keybordkat.github.io/websiteGroup6/Impl1.pdf
New URL: https://keybordkat.github.io/websiteGroup6/Impl2.pdf

Derivable and Additional Materials of Team 9

The Team 9 maze game implementation we received was a working, fairly simple game that already included player movement, a Dean antagonist with basic pathfinding, collision detection, a timer, and a pause feature. The codebase was organised into entity classes (Player, Dean, MovingEntity, AnimatedEntity, Entity) and system classes (RenderingSystem, CollisionSystem, InputSystem, TimerSystem, ToastSystem, TriggerSystem).

Critical issues we found in their implementation:
Many methods untestable because static: Timer and Toast systems only used static methods, which meant they couldn't really be tested properly with pure JUnit. The Player's invisibility timer directly called LibGDX's getDeltaTime() instead of taking it as a parameter, making it impossible to unit test. The Dean variable in Main was private, so tests couldn't access it. The Entity class had lots of unused methods that just added unnecessary clutter to the codebase.

Missing Core Features(due to new deliverables): There was no system for entering a player name. There was no achievement tracking or leaderboard system. Event variety was very limited, there was only one power-up (scroll) and no negative events at all. There were no layer changing or elevation mechanics to allow game expansion(not required but we wanted our game to be expandabe).

Changes Made:
Testability Refactoring:
Refactored TimerSystem and ToastSystem from static utility classes into proper instantiated objects, allowing dependency injection for testing.
Updated the Player's update() method so it now accepts deltaTime as a parameter instead of calling Gdx.graphics.getDeltaTime() internally.
Changed the Dean variable in Main from private to protected so tests can access it.
Removed unused methods from the Entity class, including setX(), setY(), getPosition(), toggleCollision(), isColliding(Entity other), and distanceTo().
Added getter methods to Dean (getMoveNum(), getNextTileDistance()) so its path state can be tested.

New features:
Implemented a player name input system with a text box and cursor in InputSystem:1–150.
Added a full achievement system with 11 achievements: "Press F", "Boss Fight", "Must've been the wind", "No cheese wheels?", "Treasure trove", "To infinity and beyond", "Librarian", "Book Worm", "Retep", "...no cutscene?", "Smooth sailing", "Occupant Immorality 8: Town".
Created LeaderboardSystem to track player scores with persistent file storage.
Added a book power-up (negative event) that slows the player for 15 seconds using the becomeSlow() method in Player:143–167.

Implemented a multi-layer collision system using changeLayerAtChair(), goDownLayer(), and handleChestRoomCases() to support attic exploration.
Added Dean phase 2 progression via deanLevelUp(), where the Dean randomly changes direction after a specific point in the game.
Added bidirectional pathfinding with changeDirection(), allowing the Dean to reverse direction along paths.
Refactored the Dean's movement from tile-based steps to distance-based movement using delta time for smoother motion.
Added stuck-frame detection that tracks the Dean's position and triggers syncPos() after 120 stuck frames to avoid softlocks.
Implemented position synchronisation that snaps the Dean to the nearest tile boundary when stuck, keeping it aligned to the path after direction changes.
Added deanVicinityInvis() to detect invisible players near the Dean for achievement tracking.

More events:
Added five new events: LongBoi's hidden passage (openHiddenPassage()), evil duck dialogue (secondDuckDialogue()), book power-up (negative), Dean phase 2 (negative), and attic discovery (hidden). Integrated achievement unlocks into all existing events (chest room door, exit door, scroll, spike switch). Added event counters: hiddenEventCounter, positiveEventCounter, negativeEventCounter for statistics tracking in Main:38–40.

System Integration:
Instantiated ToastSystem and TimerSystem as object fields rather than static classes in Main:42–43. Modified the InputSystem constructor so it now accepts a RenderingSystem dependency for proper integration.
Extended CollisionSystem's init() method to accept a layer name parameter, enabling dynamic switching between "Collision" and "atticCollision". Added staticWorldState() helper method to preserve world state (opened doors, lowered spikes) across layer transitions.
Integrated toast notifications across all game events, including achievement pop-ups.
Additional Improvements:
Added JavaDoc documentation to all classes to improve readability and maintainability.
Enhanced ToastSystem with customisable positioning and colours using Colour parameters.
Added a groundFloor boolean flag to track player elevation. Added nine-patch bubble graphics for better toast rendering. Created persistent storage files for achievements and the leaderboard (ach_list.txt, Leaderboard.txt).