Change Report Introduction
Our Group inherited a codebase and documentation derived from Escape from University Assessment 1 and iterated upon it as a brownfield project, forking the public repository into a private one. We adopted a structured extension and maintenance development process, working to prioritise the core features and complete the full product brief. We used a scrum-based agile workflow, conducting regular sprint planning as well as weekly meetings, which allowed us to fully implement the product brief within the timeline set.

Planning and Tracking: We performed a gap analysis between the inherited Assessment 1 deliverables and the full Assessment 2 product brief, allowing us to identify and quantify the vital and missing features. An example of this is transitioning from a single event of each type to the full requirements of five negative, three positive, and three hidden events, where we prioritised the core game mechanics, such as a leaderboard, whilst dividing up vital features among our coding team, ensuring equitable work allocation. We aligned a task list to the requirement IDs to track our progress on our development, meeting weekly to discuss it.

Tools and Review: Since we utilised the GitHub platform to host our repository, we were able to simultaneously develop and iterate upon the code. This meant that git commits were regularly reviewed by team members, splitting off into different branches when features had little to no overlap. We would peer review each other's complete classes to ensure that we met the specification and that the logic remained stable as we integrated each of the required features

Conventions for Tracking Changes: We adhered to strict Javadoc standards, using the tags // NEW: or // MODIFIED on every altered method to ensure traceability, as this allowed us to easily view where we made changes. This led to effective, high maintainability of the code. Versioning was controlled utilising git, which allowed us to utilise branching and revert the codebase to stable builds if required. We cross-referenced our git commits with the requirement IDs using internal communication, which allowed us to work on the most vital features and divide the labour amongst us.

Quality Assurance - CI/CD: We utilised GitHub Actions to implement a continuous integration pipeline where we configured a custom main.yml file to trigger its workflow on every git push, automatically building the project with Gradle, which allowed us to catch compilation errors early. JUnit 5 unit tests were executed to verify critical logic, such as the score calculation and achievement triggers, and generated JaCoCo coverage reports as downloadable build artefacts, which allowed us to conduct post-failure inspections, additionally supporting reproducibility. This allowed us to maintain the functionality of the code whilst developing upon it, as we were given immediate feedback on the health of the codebase.

These approaches allowed us to plan, implement, track, and review all changes in alignment with the full Assessment 2 product brief, working through shared Google Docs to review the original Assessment 1’s architecture, requirements, and planning to inform our decisions, adapted to the Assessment 2 brief

Requirements
Original File: Req1.pdf | https://zoey-ahmed-uni.github.io/assets/PDFs/Req1.pdf
Update Version: Req2.pdf https://mattholl26.github.io/assets/PDFs/Req2.pdf

1. Requirements Changes Overview

Assessment 2 requires extending the Assessment 1 prototype to meet the full product brief, such as expanding from 3 events to 11 events and making the map bigger to fit them all.

Table 1: Requirements Changes Overview

Category	Assessment 1	Assessment 2	Changes
User Requirements	14	14	0 (Modified UR_EVENTS scope from 3 to 11 events)
Functional Requirements	19	31	+12 (Added 2 positive events, 4 negative events, and 2 hidden events, plus the leaderboard, name entry, and achievements)
Non-Functional Requirements	15	16	+1 (Added non-functional map requirement)

2. Modified Requirements

2.1 UR_EVENTS (Req1.pdf page 2 -> Req2.pdf page 2)

- In Assessment 1 Req1.pdf under UR_EVENTS, it is stated that the system should have exactly one of each of a positive, negative, and hidden event, alongside a counter showing how many of each event the user has encountered.
- The updated version in Assessment 2 Req2.pdf under UR_EVENTS is modified to include multiple events of each type with a minimum of 3 positive events, 5 negative events, and 3 hidden events, alongside a counter showing progress in X/Y format (e.g., “2/3 positive events found”).

Table 2: Event System Expansion

Event Type	Assessment 1	Assessment 2	Changes
Positive	1. Speed boost	1. Speed boost 2. Freeze dean 3. Extra time	Kept 1 positive event and added 2 positive events
Negative	1. Dean chase	1. Dean chase 2. Slow down 3. Drown 4. Decrease Time 5. Patrol Deans	Kept 1 negative event and added 4 negative events
Hidden	1. Bus pass	1. Bus pass 2. Teleport 3. Questionnaire	Kept 1 hidden event and added 2 hidden events

3. Added Requirements (Req2.pdf pages 4 - 7)

3.1 Positive Events (added 2 events)

- FR_POSITIVE_EVENT_FREEZE_DEAN: The player discovers lab materials that freeze all deans for 30 seconds when activated with the E key. This is a one-time use power-up.
- The event was added because the user feedback indicated that players found it difficult to avoid the dean. This feature gives the player time to get past it, temporarily giving players a defensive option. This is implemented in Freeze_Dean.java.
- FR_POSITIVE_EVENT_EXTRA_TIME: The player finds an NPC that adds 30 seconds to the timer when collided with. Auto-collects on contact.
- The map is now 50% bigger, which means players need more time to explore and find all the events. This event helps players who are running out of time, so it counteracts the time decrease trap, providing balance between losing and gaining time. This is implemented in Extra_Time.java.

3.2 Negative Events (added 4 events)

- FR_NEGATIVE_EVENT_SLOW: The player collides with a bush that reduces movement speed by 50% for 20 seconds.
- The event was added to give a disruptive but not game-breaking experience, like getting caught by the dean. Making the player slower for 20 seconds adds a little challenge without making it punishing. This is implemented in Slow_Down.java.
- FR_NEGATIVE_EVENT_DROWN: The player falls into water tiles, triggering respawn at the starting position with a -10 score penalty.
- The event added is meant to make the players pay attention to where they're going by adding water tiles to the map, giving environmental obstacles instead of using just the dean. This is implemented in Drown.java.
- FR_NEGATIVE_EVENT_DECREASE_TIME: The player collides with a tree that reduces remaining time by 30 seconds.
- The event is meant to add time pressure to the game. Unlike the dean who chases the player around, this trap will make the players more aware when exploring the map. This is implemented in Decrease_Time.java.
- FR_NEGATIVE_EVENT_PATROL_DEAN: There will be three deans patrol vertically in the bottom room (Y coordinates 90-260) at 0.7f speed. Each reverses direction at the boundaries. If a player fails the questionnaire, a fourth patrol dean spawns as a penalty.
- The event creates a danger zone for players who can see it coming and find alternative routes just to avoid the deans. This is implemented in Patrol_Dean.java (3 instances in GameScreen.java) .

3.3 Hidden Events (added 2 events)

- FR_HIDDEN_EVENT_TELEPORT: The player discovers a hidden science lab entrance that teleports them to a shortcut location in the maze that could reduce the player's travel time to reach the exit.
- The event gives players who find teleportation an advantage to complete the game quicker. This is implemented in Teleport.java.
- FR_HIDDEN_EVENT_QUESTIONNAIRE: The player discovers a hidden questionnaire that provides bonus points when the player answers correctly. However, failing the questionnaire will trigger the game to spawn a fourth patrol dean as a penalty.
- This is a bonus hidden event that doesn't affect the core gameplay but adds extra points like a fun Easter egg for players who actually try to explore everything on the map, and consequences for wrong answers. This is implemented in Questionnaire.java.

3.4 User Interface and Data Persistence

- FR_LEADERBOARD_SAVE: The top 5 scores are saved into a JSON file and displayed on a leaderboard screen. This was essential to meet the product brief for a

persistent high score, allowing people to compete for the high score and increasing the competitiveness. This is implemented in Save_Leaderboard.java
- FR_NAME_ENTRY: A new screen is displayed with an onscreen keyboard that allows the user to input their names before playing. This ensures that the user who got the high score can be identified. This is implemented in NameScreen.java

3.5 Map Size
- NFR_MAP_SIZE: The map dimensions have expanded by approximately 50% to accommodate event locations while maintaining the 5-minute completion target, referring to (NFR_GAME_COMPLETION) non-functional requirements. Current map size: 640x640 pixels.
- With 11 events instead of 3, the map has to be bigger to give out elements of surprise, and the events wouldn't spawn too close together. If events are clustered, it wouldn't make the experience enjoyable. The 50% increase gives enough space for all events while still keeping the 5-minute completion time from Assessment 1.

4. Impact on other deliverables
- The event expansion required creating 8 new event classes and updating GameScreen.java to manage them all (see Arch2.pdf).
- Automated testing increased from 0 to 228 tests since each event needs 8-24 test cases (see Test2.pdf).
- The team added a penalty system, which is -5 points for dean catches, -10 for drowning, and -30 seconds for the time trap, with respawn. (see Impl2.pdf).
- Implementation changes were tracked by requirement ID in commit messages and via // NEW: / // MODIFIED: tags in code

5. Deferred Requirements
UR_SETTINGS / FR_SETTINGS_OPTION: The volume controls and the resolution settings (which are listed in Req2) were deferred as given the time constraints, we prioritised the implementation of the core gameplay loop (11 events and leaderboard) over these configuration menus. This ensured that we implemented a fully playable MVP, delivered by the deadline.

All of our requirement changes have been documented above. No other requirement updates were necessary for the completion of the assessment.

Architecture
Original Version: Arch1.pdf | https://zoey-ahmed-uni.github.io/assets/PDFs/Arch1.pdf
Updated Version: Arch2.pdf | https://mattholl26.github.io/assets/PDFs/Arch2.pdf

Overall Changes:
Considering changes to the requirements document had to be made to include the full amount of events, achievements and also the leaderboard, the architecture also reflects these changes as new classes and relationships had to be formed. The original layered architecture was preserved, which allowed us to extend the project without unnecessary dependencies.

Evolution:
A sentence was added to this section, "Upon the extension of requirements, another package diagram was added and extended to include all the new classes," to clarify that an extra diagram would be added to reflect the new requirements.

Updated Layered Architecture Table:
The following components were added to the existing layers, which allowed us to fulfil the new functional requirements. Existing framework layers were retained.

Layer	Components	Responsibilities
Presentation layer	LeaderboardScreen, NameScreen	Handles the new User Interface (UI) for the name entry and high score display.
Game Logic layer	Achievement, Save_Leaderboard, Patrol_Dean, Teleport, Questionnaire, Freeze_Dean, Slow_Down, Drown	Contains some of the core game rules, entity behaviours for some new events, enemy behaviour, item interactions and scoring logic

CRC cards:
These rows were added onto the existing CRC cards to further cover all potential classes needed to fulfil the requirements for the game.

Class	Responsibilities	Collaborators
Achievement	Tracks the player's achievements (positive/negative impact)	MyGame
Decrease_Time	A Negative event which reduces the game time upon collision with a tree	GameScreen, SpriteBatch
Drown	A Negative event, which is a water hazard. It resets the player's position when the player walks over to its area.	GameScreen, SpriteBatch
Extra_Time	A Positive event which adds a permanent time boost to the timer when a player interacts with a specific NPC.	GameScreen, SpriteBatch
Freeze_Dean	A Positive event that freezes all deans for 30 seconds once the player has interacted with the materials area.	MyGame, GameScreen
LeaderboardScreen	Displays the top 5 scores from a local JSON file. The player can then return to the main menu screen or quit	MyGame, GameScreen
NameScreen	Implements a name entry screen using a simple on-screen keyboard. The player enters at least a mandatory first name	MyGame, GameScreen
Patrol_Dean	A Negative event, which is an enemy that patrols vertically between the Y-axis bounds.	GameScreen
Questionnaire	A Hidden event which renders the quiz prompt, questionnaire questions, and the result text.	GameScreen, SpriteBatch
Save_Leaderboard	Saves leaderboard scores to the local JSON file. Existing file contents are overwritten.	MyGame
Slow_Down	A Negative event that reduces player speed by 50%	GameScreen
Teleport	A Hidden event that teleports the player to a random location	GameScreen

Traceability To Requirements:

The new functional requirements that would need to be seen within the game's code were mapped to the specific architectural components to ensure full coverage.

Requirement ID	Related Component(s)	Architectural Rationale
FR_POSITIVE_EVENT_LOCKER	Locker.update(), Locker.isBoostActive(), GameScreen.handleInput()	Locker.update() detects player proximity and E key presses, activates temporary speed boost. Movement speed is updated when a boost is active
FR_POSITIVE_EVENT_FREEZE_DEAN	GameScreen.freezeAllDeans(), playerRect.overlaps()	If the playerRect overlaps with the materials and E is pressed, the game screen freezes all the deans.
FR_POSITIVE_EVENT_EXTRA_TIME	Player.getPosition() timer.addTime()	If Player.getPosition() finds that it is overlapping with the NPC, then time is added
FR_NEGATIVE_EVENT_PATROL_DEAN	gameScreen.isCellBlocked()	If the deans that are patrolling get stuck in a cell found by isCellBlocked(), they can be reverted to get unstuck
FR_NEGATIVE_EVENT_SLOW_DOWN	player.GetPosition()	If .GetPosition() finds that it is colliding with the bush, debuffs can be applied
FR_NEGATIVE_EVENT_DROWN	Player.getPosition() obj.getRectangle()	If .GetPosition() of both the player and the rectangle overlap; the player can be respawned.
FR_NEGATIVE_EVENT_DEAN_CHASE	Player.getPosition() gameScreen.isCellBlockedForDean() tryMoveDiagonally()	.GetPosition() can help guide the dean towards the player via diagonal movements.
FR_NEGATIVE_EVENT_DECREASE_TIME	Player.getPosition() timer.addTime()	If .GetPosition() of the player is colliding with tree, then time can be added via .addTime()
FR_HIDDEN_EVENT_BUS_PASS	BusTicket.isCollected(), NPC.showMessage, BusTicket.render()	The bus ticket is initially hidden, revealed via BusTicket.discover() when the player gets close, the NPC provides a hint through a dialogue
FR_HIDDEN_EVENT_TELEPORT	player.getPosition() getRandomSafePosition()	If the player triggers teleport, then getRandomSafePosition() means they can be teleported to a new set of coordinates
FR_HIDDEN_EVENT_QUESTIONNAIRE	player.getPosition() Keyboard Input gameScreen.spawnSecondDean()	If the player gets a question wrong via keyboard input, the deans get spawned via .spawnSecondDean()
FR_ACHIEVEMENT_SYSTEM	Achievements	Achievements get added through gameplay
FR_LEADERBOARD_SAVE	loadScores() finalScores.add() finalScores.sort() saveScores() Integer.compare()	Scores can be loaded and added after the game is finished.

	finalScores.removeIndex ScoreFile.readString()	
FR_LEADERBOARD_DISPLAY	Render leaderboard leaderboard.getHighScores(); 	Loads scores from the file and ranks them.
FR_NAME_ENTRY	ScreenRender() CreateKeyboard()	CreateKeyboard() lets a player type their name in via the onscreen keyboard.

No architectural layers were removed. The changes above were limited to adding components and classes required by Req2.

Method Section - Changes and Justification
Original doc: Plan1 | https://zoey-ahmed-uni.github.io/assets/PDFs/Plan1.pdf
Updated doc: Plan2 | https://mattholl26.github.io/assets/PDFs/Plan2.pdf

1.0 Reasons for Minimal Changes to the other team's original plan

After receiving the original team's documents from Assessment 1, we decided that no major changes were needed to the method and plan because it was very similar to our own approach in Assessment 1. So we decided to stick with an Agile approach, but only changing the team roles to fit Assessment 2 better. Because the methodology we used was very similar, changing this would have resulted in unnecessary overhead and meant we would have started on the main parts of the project later than we wanted to.

This plan provided us with a good structure throughout the project, allowing us to be focused on our roles while giving us a way to easily communicate any issues. It also allowed us to get started on our main project quickly, such as implementation and the change reports.

2.0 Changes: Agile Manifesto

2.1 Working software over significant documentation:
We increased our emphasis on the documentation's clarity alongside implementation, as with receiving documents from another team, there could be ambiguity about roles and requirements that were set out. So clear documentation was required to ensure none of these issues occurred, as everyone knew their role and tasks set out for them to do, reducing handover ambiguity and supporting traceability.

2.2 Individuals and Interactions over processes and tools
We kept this because it prioritises communication and collaboration over anything else. This will be helpful to keep every role informed about what is happening with each part of the project.

2.3 Responding to change over following a plan
Again, we kept this because it means any complication that arises will be easy for us to adapt to and change. Also, with effective communication, this means it's easy to do this.

2.4 Customer collaboration over contract negotiation
Finally, we kept this one as it means everything we do will be traced back to the original stakeholder questions, and means we should be on track to hit our goals through the project.

3.0 Role Allocation Changes and Justification

Roles were assigned based on individual strengths and workload balance. Roles were changed to prevent an over-reliance on a single person. We changed it so there is a clear

justification for each person's role and what tasks they are doing, and also shared ownership of the larger tasks between multiple people, so no singular person is doing more than their share. This allows us to give each other immediate feedback on what we are currently working on.

We designated each role based on each other's experience, so people with better technical skills were assigned to implementation, and people with better organizational skills were given tasks like risk management. We also assigned the scrum master to the person with the best communication skills, so everyone had someone to update their progress to. These roles weren't fixed and could be changed throughout the project if needed, for example, if we were running behind on a task but ahead on another, a person could be switched from one task to another. We can do this because of our effective communication and focus on documentation, meaning it is easy to get up to speed.

4.0 Task Additions and Justification

Tasks 18-25 were added to the original task list to reduce risks introduced by the project handover.

- **Task 18** - **Receive Documents**
  Added to account for the handover process and reduce risk while receiving the documents.
- **Task 19** - **Discussing Requirements Updates**
  Added to reassess requirements and ensure they match up with the overall project requirements.
- **Task 20** - **Deciding Team Roles**
  Added to make sure everyone knows their responsibility early and reduce the risk of confusion over tasks that need to be completed.
- **Task 21** - **Updating Change Reports**
  Added to make sure we update the original documents, improve, and ensure implementation aligns with documented changes.
- **Task 22** - **Implementation and Testing**
  Scheduled after the start of documentation updates to reduce risk caused by unclear requirements and documentation.
- **Task 23** - **User Evaluation**
  Added to reduce the risk of the final product not meeting requirements.
- **Task 24** - **Continuous Integration**
  Added to mitigate integration risk.
- **Task 25 - Final Check and Deployment**
  Added to ensure all documents and code meet requirements before submission.

Conclusion

The original plan required minimal changes as it already aligned with our team's original Agile approach. Changes were made to improve documentation quality and role coverage. Emphasis on documentation was increased to reduce misunderstanding and support changes during development. Roles were changed to ensure each member is doing their fair share and not overly reliant on a singular person. These changes improve communication, reduce dependency on individuals, and support consistent project delivery. While the core Agile methodology was retained from Assessment 1, we refined the team roles and task allocations of each member, allowing us to effectively tackle any challenges faced during the Brownfield handover.

Risk Assessment and Mitigation

Original File: Risk1 | https://zoey-ahmed-uni.github.io/assets/PDFs/Risk1.pdf
Updated Version: Risk2 | https://mattholl26.github.io/assets/PDFs/Risk2.pdf

1.0 Requirements Change Overview
1.1 Estimation section added: as this includes many risks
Added an estimation risk section - This adds any risks related to overestimating or underestimating parts of the project. This can have a big influence on the project if anything is extremely underestimated or overestimated. Such as underestimating the time the project will take to complete.

1.2 Changed impact key + likelihood

Removed their original impact key and changed it to low, medium, and high, so it is easier to understand and reduces the variation difficulty down to three worded levels, to improve readability for the stakeholders, removing the need to check a number against a key. Also changed the likelihood to high, medium, and low to additionally improve readability for stakeholders

Changed the likelihood on TR1 to high as the new team has to get used to the old team's tools, which would impact the project more, as it takes more time to get used to tools when compared to the old team, who were used to the tools.

1.3 Frequency of Changes

Added continuous integration to TR2, as this falls into integration issues. Also changed weekly to daily in order to reduce the risk of problems with integration, as the changes will be added more frequently to prevent a week's worth of problems affecting the code (so worst case scenario, only one day's worth would affect the code)

1.4 Structural and Classification Changes:

- Changed TR2 members do not show up to meetings, to Organisation and Team Risk, as it is NOT a technical risk as originally categorised.
- Also added very obvious + impactful technical requirements that are very likely to happen. Such as software bugs, which are extremely likely to occur.
- Changed the ownership column presentation so it is clearer who owns what, and we have justified every person who owns something.
- OTR1 has been removed as ‘people are sick’ falls under people not attending meetings.
- Removed the scheduling and delivery section, as we will make an estimation section that more tasks will fit under - including the scheduling and delivery, as before, there were only 2 items in this section.
- Moved QRR3 and QRR2 into technical requirements, as testing and traceability are both technical problems.
- Added QRR2 ‘the product does not meet the final requirements’ as this shows the risk of the game not meeting any required standards, which would be an extreme setback.
- Added estimation table, as both over- and underestimating the final product can have extreme setbacks on the final product. Any underestimations would mean the final product’s requirements are not met.
- Changed the ownership so there is less ownership on one person (the scrum master), so the workload is spread more evenly and is not all dependent on one person if a risk occurs.
  We retained the existing risk framework but refined it to clarify the impact levels, enhance the clarity, and address the new integration and estimation factors.