Change Report

Team10
Ben Leaver
Gergely Gal
Ali Tekin
Faras Hbal
Chihun Park
Hux Stevenson
Utkarsh Singh

Summary

Before beginning anything else we decided it was most important to review the deliverables that we had inherited from the previous team. We did this in the form of a meeting and had the goal of deciding which deliverables we deemed were necessary, if we needed to change the definition of any or if we needed to add any additional deliverables that we thought were important to the final form of the project. We agreed this was most important to begin with as this was going to be a large defining factor of what we were going to develop and how it was going to change the shape of the project we had taken over. We also knew we would need to later link back to previous requirements and deliverables so it made sense to have a clear set of defined deliverables to then work from and be able to refer back to in a concise way.

Due to the increased size of the game and project during assessment 2 we sought to keep good track of where and how we had developed the project we had inherited from assessment 1. To keep track of changing and evolving code from the given project we applied thorough and detailed documentation of additional or modified code, in order to have changes clear and concise. This report shows exactly where and why we changed or added to the assessment for each section: requirements, architecture, method selection and planning and risk assessment and mitigation. As previously mentioned the changes made in our implementation are reflected by our rigorous documentation of our code.

One of the main conventions we underwent to help manage this change, was our weekly meeting schedule, this included our usual planning and organisation however we would additionally gather all of our collective changes to current material and reflect this in a table. This gave us a clear weekly overview that could be collected as one clear set of data, also helping us as a team to form a strong idea of how much we had improved the previous assessment material we inherited. This also greatly helped support our continuous integration work model, with an emphasis on steady work flow that was continuous across the project.

Requirements

New document: https://thedebugthugsuoy.github.io/files/Req2.pdf
Original document: https://thedebugthugs.github.io/files/Req1.pdf

Requirements Changes - New user requirements were given in a new statement of user needs which added the new requirement of a leaderboard and an achievement system to the game. These new requirements are reflected in the updated requirements 2 document which now has included UR_LEADERBOARD and UR_ACHIEVEMENTS which are added to the user requirements table as seen below.

UR_LEADERBOARD	The system shall store a list of highest player scores and names in descending order.	High	Gives the player a motivation to replay the game to achieve a higher score.
UR_ACHIEVEMENTS	The system shall store and track a series of achievements of the player.	High	Gives the player a motivation to replay the game to achieve a higher score.

Additionally the User requirements were adjusted as the UR_SCORE didn't appropriately reflect the dependency of having a score system for the leaderboard to use. The new score requirement is shown below.

UR_SCORE	The system shall generate a final score based on the player's escape time and the events they interacted with.	High	Gives the player a motivation to replay the game to achieve a higher score.

Functional Requirement Changes - The new user requirements were devised into 3 new functional requirements FR_LEADERBOARD_SCORE, FR_LEADERBOARD_MENU, FR_ACHIEVEMENTS and FR_TUTORIAL_ACCESS these requirements are appended to the already existing functional requirements table as it contained all other functional requirements needed for the project.

FR_LEADERBOARD_SCORE	The system shall save player scores to a JSON to persistently store the highest user scores.	UR_LEADERBOARD	Inspection, Test
FR_LEADERBOARD_MENU	The system shall display an ordered list when selecting the clicking the leaderboard menu button.	UR_LEADERBOARD	Inspection,
FR_ACHIEVEMENT	The system shall save players data to store a series of achievements they have unlocked that persists between play sessions	UR_ACHIEVEMENTS	Inspection,test

FR_TUTORIAL _ACCESS	The tutorial should be displayed to the user when accessed through the main menu by UI elements.	UR_AUDIENCE	Inspection, test

The functional requirements FR_GAMBIT_SHORTCUT and FR_GAMBIT_BONUS were removed from the tables as they did not seem like necessary requirements to specify or implement to the project. FR_SUBTITLES was removed from the original table since there were no audio cues for the player needed to understand and therefore no need for subtitles for accessibility. No additional changes were made to the functional requirements tables as the rest of the requirements remained suited to the new user requirements.

Non Functional Requirements - the NFR_TUTORIAL was added from UR_ACCESSIBLITY which covers the need for a clear introduction to the controls of the game for the user who has never played any games before and may find the system unintuitive.

Architecture Change Report

New Document: https://thedebugthugs.github.io/files/Arch2.pdf
Original Document: https://thedebugthugs.github.io/files/Arch1.pdf

Architectural Modifications:

The first thing to do with the updated architecture was to modify any parts of the previous team's design that we deemed needed to have changes made. As the architecture already had very good design coverage and clear evidence of the design journey, we did not see much need to change many parts of the original architecture as it already fit the requirements very well and gave a good representation of the game as a whole. Of the parts we did slightly edit, this included how the original design had one events class that related to the player and FirstScreen. As our implementation grew we decomposed this into lots of self-contained individual classes for each event, improving ease of debugging and testing, it also made more sense with the large amount of events that were being handled. This change can be found in the “Events” point in the “Evolution” section of the Arch2 document on the website. The last architectural change we made from the original was done in the 3.1 Game State Machine diagram, where we changed the SettingsScreen to be only accessible from the menu instead of FirstScreen as this better reflected the final game, we also showed the link from LoseScreen back to the menu as this was not previously included and was important for the player to have this functionality.

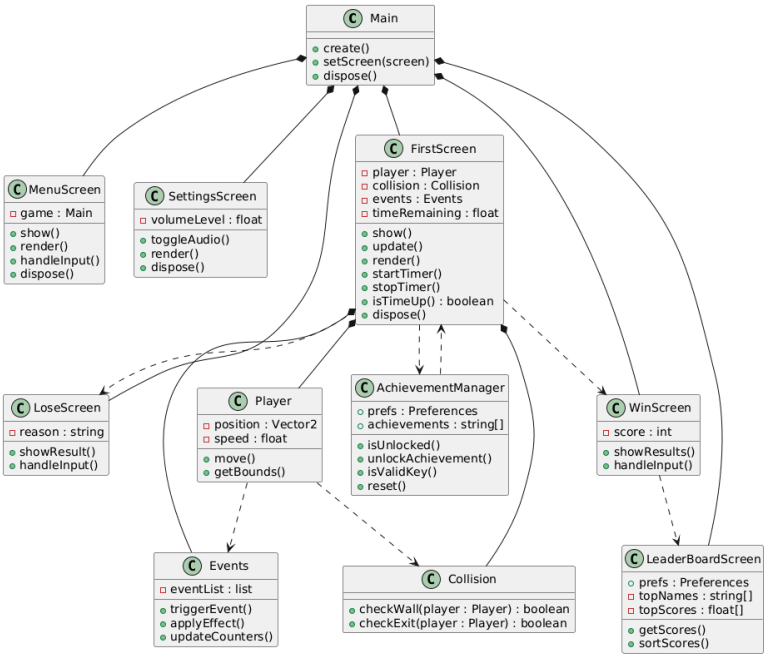
Architectural Additions:

This section makes up the largest portion of our architectural changes and involves design that we have added to the original architecture document. This mostly includes changing the architecture to fit the new requirements given to us (leaderboard and achievements) and

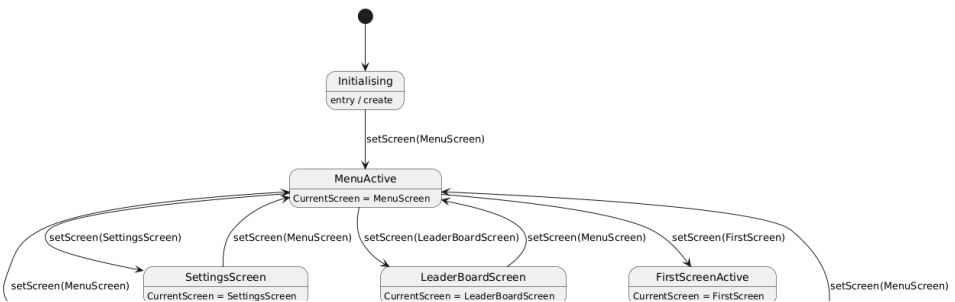
how this fits into the original designs. We have integrated these additions by making the following additions:

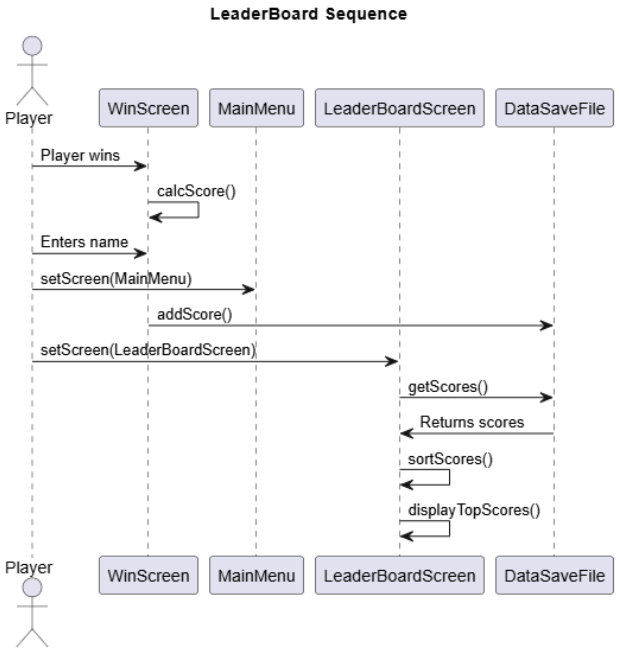
- **UML Diagrams** - We have extended the UML 2.0 class diagram to the UML 3.0 class diagram to give a strong structural view of the new additions and how it links to the old architecture. We have also updated 3.1 Game State Machine in order to include the screen displayed by LeaderBoardScreen in the game's current screen switching logic. We have created a new sequence diagram called 3.6 Leaderboard Sequence, this shows how the player interacts with the game to save and view their score on the leaderboard once the game has finished and is key in showing behaviour of this new addition. All of these updated and new UML diagrams can be found below and also under the “Architectural Diagrams” section of the Arch2 document on the website.
- **Requirements Traceability** - We have updated the requirements traceability table with both of these new features and shown how they link to the new requirements that can be found in the requirements change report. We have also updated the WinScreen section of this table as it is key in how the leaderboard works in terms of saving player data. These changes are all found in the table in “Requirements Traceability” in the Arch2 document on the website.

UML 3.0 class diagram showing the Structural View:



3.1 Game state machine:





3.6 Leaderboard Sequence

Method Selection & Planning

New document : https://thedebugthugsuoy.github.io/files/Plan2.pdf
Original document: https://thedebugthugs.github.io/files/Plan1.pdf

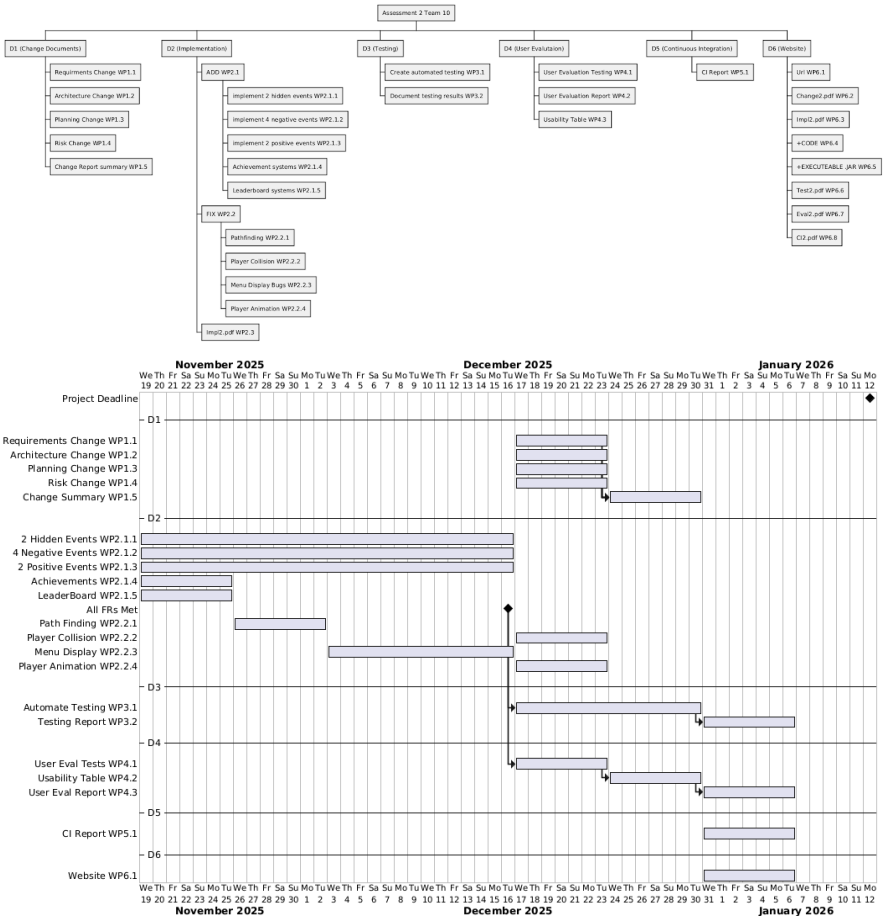
Method Selection Changes - Since the nature of the new project is very similar to the previous instance we didn't decide to change from the scrum inspired hybrid waterfall method previously used. This was because the method allowed for clear planning of deliverables and milestones needed to complete the project and also allowed for the agile iterative development of code which would suit implementation and testing of this project nicely.

Tool selection Changes - Previously Intellij IDEA was used for its compatibility with Java 17 and testing tools. Our team was more familiar with using Visual Studio Code which is also compatible with Java 17, LIBGdx and gradle using plugins allowing us to switch IDEs very easily. Visual Studio Code is also compatible with Github desktop allowing us to integrate with our continuous integration systems.

All other tools used previously seemed effective and intuitive for our team to continue to use such as Instagram, google meets, google drive and github as the team was already established with using these tools for the project so no other changes were made to tool selection.

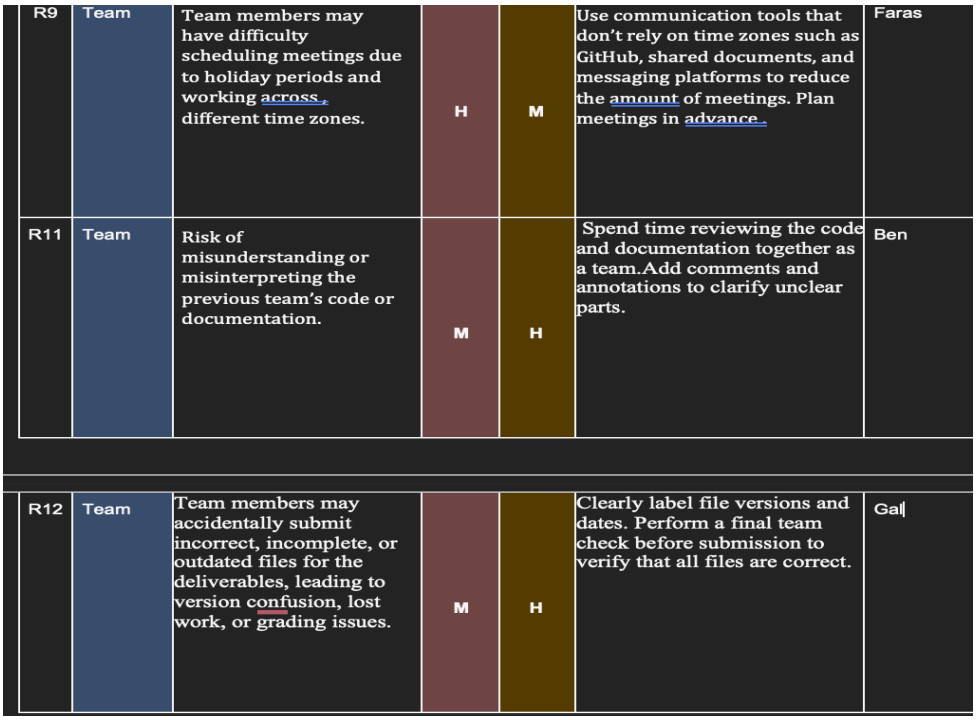
Team Organisation - we chose to not use any defined team roles and instead had each member always work on the most appropriate task at the time we divided the implementation work between us evenly to reduce risk of delays in development so our user testing can begin on schedule. This is no different from the previous team organisation. No change was made as using this loose team roles system allowed us to meet our deliverables previously.

Planning Changes - Since the project now had new requirements and some requirements were already completed and a new deadline an analysis of the current state of the project had to be done to design a new work breakdown structure to decompose the work into work packages for meeting all the new requirements. This was then scheduled into a gantt chart following the waterfall model to create an overview of the dependencies. We tracked what tasks we are working on a table on a shared document and had weekly meetings to update it.



Risk assessment and Mitigation

New Document: https://thedebugthugs.github.io/files/Risk2.pdf
Original Document: https://thedebugthugs.github.io/files/Risk1.pdf



R9	Team	Team members may have difficulty scheduling meetings due to holiday periods and working across different time zones.	H	M	Use communication tools that don't rely on time zones such as GitHub, shared documents, and messaging platforms to reduce the amount of meetings. Plan meetings in advance.	Faras
R11	Team	Risk of misunderstanding or misinterpreting the previous team's code or documentation.	M	H	Spend time reviewing the code and documentation together as a team.Add comments and annotations to clarify unclear parts.	Ben
R12	Team	Team members may accidentally submit incorrect, incomplete, or outdated files for the deliverables, leading to version confusion, lost work, or grading issues.	M	H	Clearly label file versions and dates. Perform a final team check before submission to verify that all files are correct.	Gal

Additional Team Risks Added

After taking over the previous team's Assessment 1 deliverables, we identified three additional team risks that were not included in the original risk register.

R10 – Scheduling difficulties due to holidays and time zones:

Since the submission of Assessment 2 is after the holidays, one of the risks that we think might happen is that team members are on holiday or located in different time zones, which makes arranging synchronous meetings difficult. This risk could reduce communication and delay decision-making. Mitigation focuses on using asynchronous communication tools (e.g., GitHub, shared documents, messaging platforms), scheduling meetings well in advance, and clearly documenting all decisions to ensure everyone stays informed.

R11 – Misunderstanding inherited code or documentation:

Inherited code and documentation could be misinterpreted, which could lead to incorrect implementation, wasted effort, and delays. Mitigation strategies include reviewing the code

as a team, adding comments, and conducting code reviews to make sure all team members correctly understand the inherited work.

R13 – Risk of incorrect or incomplete file submission:

Team members may accidentally submit wrong, outdated, or incomplete files for the project deliverables, which could cause confusion or require rework. Mitigation includes maintaining a central repository for all submissions, labeling file versions clearly, and performing a final team check before submission to verify that all files are correct.

These risks were added to the risk register and are actively monitored to reduce impact on project progress. They show our proactive approach to managing risks from team collaboration, inherited work, and project coordination.

Team Risks Removed

After reviewing the original risk register inherited from the previous team, we figured that this one risk was no longer applicable and has therefore been removed.

R1 – Customer dissatisfaction due to changing requirements:

This risk was originally due to the possibility of customer dissatisfaction or changes in requirements during development. However, this risk is no longer applicable because assessment 2 does not involve a meeting with a customer opposed to assessment 1. So the likelihood of it happening again is nonexistent.

R1	Product	The customer may not be happy with our game progress, requirements can change.	M	H	Ensure that we are receiving feedback from the customer regularly so if there are any issues they can be quickly resolved by the team. Everyone needs to be up to date on our requirements so we are all at hand to answer questions from the customer.	Rosie

References:

New Requirements Document: https://thedebugthugs.github.io/files/Req2.pdf
Original Requirements Document: https://thedebugthugs.github.io/files/Req1.pdf

New Architecture Document: https://thedebugthugs.github.io/files/Arch2.pdf
Original Architecture Document: https://thedebugthugs.github.io/files/Arch1.pdf

New Planning Document: https://thedebugthugs.github.io/files/Plan2.pdf
Original Planning Document: https://thedebugthugs.github.io/files/Plan1.pdf

New Risk Assessment Document: https://thedebugthugs.github.io/files/Risk2.pdf
Original Assessment Document: https://thedebugthugs.github.io/files/Risk1.pdf