Change Report - Group 10 Cohort 1

In the pre-planning of our project we held a group meeting to discuss the current state of the inherited project and the areas which needed the most changes to grasp an idea of where to start. In that meeting we took notes and delegated tasks to different group members whilst retaining our group roles from assessment 1. We documented these plans in a Discord server, which we initially created for our previous assessment with modifications to support the development of the new deliverables. We also created a google sheets table which we populated with ideas for changes we wanted to make, with columns for the change number (for tracking purposes), the change title and description and whether the change had been made yet.

Our methodology towards development remained the same as not only our previous project, but also as that of the group whose project we picked up. The scrum approach was found to be the best way to plan development week by week and review our progress later in each week. Meetings were scheduled ahead of time each week with a room being booked for in-person meetings. As the deadline drew closer group members went home over the Christmas period, at which point our group meetings moved to Discord.

We created a shared google drive to store all of the inherited deliverables, which we also decided would further be useful to store updated/additional deliverables. Our file manager, James Farrow, also stored a backup locally as a precaution for the possibility of file corruption or loss of access to data due to network issues.

At numerous stages in development we reviewed changes made and consequently further changes which we would need to make. We did this through further group meetings to review where we were currently at and what we needed to work on further. These meetings took place both in person and virtually. When changes were made to the implementation we submitted them to a google form, which nicely categorised the author of the change, the change made and the reasons for the change.

In order to review changes made to the implementation, we used Continuous Integration on our github repository. Whenever code was pushed onto the main branch, CI would automatically build the project and run it for different test cases, which we set ourselves. Any issues brought up by a failed build or error produced by CI would then be immediately brought to the attention of the implementation team, allowing them to address the issue.

With development of the project the requirement came around to update deliverables to reflect our development and adaptation of the project. With these changes came the need to keep track of anything we updated through documentation, which is the purpose of this current document - Change2. Below the changes to each of the 4 key deliverables have been specified, with explanations for changes as well as things that we left the same, and the reasoning behind each one.

**Requirements**

When picking up another team's project we understood that the requirements would be different to the ones they had previously followed, and as such we'd need to identify them through comparing their project to the updated brief. This is stated in a section of the *eliciting our requirements* header starting with "assessment 2" in brackets to indicate the additions we've made. This is consistent between each additional information we've added in subcategories.

Assessment 2 required us to add additional functionality to the game of our choice. As such the requirements deliverable had to be updated to reflect what was further required to meet the client requirements. We adopted the previous group's layout of requirements and added some new user requirements as specified, which contained different categories of requirements in tabular form, totalling in 4 tables. The requirements in each table are ordered based on priority from highest to lowest.

Given that we were picking up the previous team's requirements document, we decided to follow the same format and layout of their requirements, we felt that we didn't need to add a large description of how we're continuing with their method. As such, we included a single sentence to explain our thought process.

In the Single Statement of Need, additional information was added to reflect the updated requirements. These additions are reflected in bold text which has been added to the statement of the previous group.

One of the requirements was a leaderboard that shows the top 5 scores with names of each user who gained them. This was added as an essential requirement to the table of user requirements as well as the system requirements. As this addition was categorised under features that the game *must* contain, the significance of completing the task is listed as essential. Below is the user requirement for the addition of a leaderboard.

| | | | |
|---|---|---|---|
| UR_LEADERBOARD | The user should be able to view a leaderboard of the top 5 scores and the names of the people with those scores. If the user scores a top 5 score, they should be asked to input their name to be added to the leaderboard. | Essential | SR_LEADERBOARD | APPROVED |

The leaderboard also has an associated system requirement; the game would store the data of the top 5 scores and the user who earned each one. When required the game should also display this information to the user. Below is the system requirement entry for the leaderboard.

| | | | |
|---|---|---|---|
| SR_LEADERBOARD | The game must include a leaderboard, which stores and displays the top 5 scores with a name connected to each score. If a user's final score places within the top 5 they should be asked to input their name to be added to the leaderboard. | Essential | UR_LEADERBOARD | APPROVED |

Given that this requirement does not change any major components of the game, no changes are required to the non-functional or constraint requirements.

Another requirement included in the updated client brief states that the game should feature achievements for completing certain tasks within the game. The brief proposes that these achievements could influence the player's final score either positively or negatively, assumedly dependent on the nature of the achievement. The associated user requirement in the deliverable is shown below.

| | | | |
|---|---|---|---|
| UR_ACHIEVEMENTS | The user must be able to earn achievements whilst playing the game if they meet certain conditions. These achievements could influence the final score of the player, either positively or negatively. | Essential SR_ACHIEVEMENTS | APPROVED |

The corresponding system requirement can be seen here:

| | | | |
|---|---|---|---|
| SR_ACHIEVEMENTS | The game must feature achievements which can be earned in a playthrough with each achievement having a criteria to unlock. The game should keep track of unlocked achievements to influence score and to ensure the same achievement is not earned multiple times. | Essential UR_ACHIEVEMENTS | APPROVED |

On top of these specific additions, the requirement for the final product has changed. Instead of only having to meet certain requirements specified in the initial brief, every requirement mentioned in the brief must be fully complete. Therefore, every user and system requirement had their priority raised to essential to reflect this. In order to still reflect some level of hierarchy when it comes to tasks, tasks are ordered in the table from first to complete to last (which was already followed to an extent by the previous group).

Change Report - Group 10 Cohort 1

## Architecture

The architecture deliverable of the previous group contains well-made graphs and diagrams representing the different features, classes and other components of the game. There is however an issue with these diagrams, being that it does not reflect the updated state of the game including new functionality. Given this fact, the diagrams had to be reconstructed to reflect our implementation.

The document begins with an *Introduction and Overview of the Tools Used*, of which we did not make any changes. This is because the listed tools match the tools we used for development, being Java as the programming language, LibGDX as the library implemented to build the project, and using PlantUML for UML diagrams. During core implementation all of the programs used by our group have already been listed in the deliverable.

Moving on, the *Architectural Diagrams and Structure* subheader features a UML graph of the architecture. Given that we've picked up this group's project, the UML diagram produced by them is still highly relevant. However, in proceeding further with implementation towards a finished product, new elements have been introduced which should be reflected in the UML graph. As such, our group made an updated graph which can be seen just below the original in the deliverable. Following this, each component has a short description for each of its inclusions which we once again updated to reflect new additions to the graph. This is located at the bottom of the preexisting deliverable.

An activity diagram was also created to represent the behavioural architecture of the project. This activity diagram is a flow chart which features all of the processes occurring within the game as it runs, with different branches to represent process states and conditions. Once again, this diagram still held high significance towards the project but still had to be updated, especially with consideration towards new additions introduced in the updated brief. Our updated graph is again provided just below the preexisting one to allow for a clear comparison between the obtained project and the changes introduced by us. Key actions are listed alongside the diagram, however these were slightly barebones and lacked functionalities that we chose to add including events and also achievements, which were newly introduced by us.

Alongside the other graphs, we've added some specialised graphs to reflect our new additions to the project. Several diagrams are included here: an abstracted diagram which has attributes and methods removed, an unpackaged diagram, a UI package, an achievements package, a graphs package and a map package. All of these diagrams are a great way to represent different parts of the project through individual components and also allows us to reflect on certain details, such as a new addition in the achievements package. They're all at the bottom of the graph with brief descriptions of what each diagram shows.

The group included the *evolution of their architecture* as 7 distinct stages, each with more development towards the intended final outcome. We updated this with 2 additional iterations, describing how we built upon the preexisting project and introduced new features as specified on the brief. We also reflected on how we revisited some of the preexisting features and built upon them due to them not being fully implemented as required.

*Justification* was made for all core components of the architecture created by the previous group. This didn't explore specific implementations but instead explains design choices and organisation with reference to how the game is composed. We made no major changes in this regard, so we did not update this part of the deliverable.

Finally, *requirements traceability* is a section of the deliverable which explains how core functionalities align with user and system requirements. The general explanation here was suitable and did not need any updating, however an additional point was added to reflect traceability of newly added functionality.

## Method Selection and Planning

When picking up another group's project, it was recognised that the previous group likely used different methods and/or planning processes during their development of the project. With that in mind, we reviewed their method and planning techniques and updated the document where necessary to reflect our methodologies we'd be moving forward with.

The first category was *the chosen methodology*. Coincidentally, the group we chose opted for the Scrum approach, which is the same method we've adopted for development. Given this fact, there was no requirement for us to update the method selection section of the document. However, we did further consider a new method in reflection of the more clear final goal. This method was the plan-driven approach; with a clear picture of an end product now in sight, it would be plausible to map out everything we had to do and work towards a final product. Despite opting for the Scrum approach going forward regardless, it would be useful to reflect our consideration in the updated deliverable.

Moving onto *Collaboration and Documentation tools*, we added additional tools which we used to their documentation. These tools included Discord for communication as opposed to Slack due to familiarity, and whatsapp as an additional communication tool to organise meetings. A small paragraph was added to separate their tools and ours, however we followed the same format and explained why our chosen tools were better suited for our project than the previous group's tools. We chose not to remove the tools listed by the previous group as they remain a reflection of the project's development process regardless of whether we adopted them or not.

As a side note, like a lot of the other graphs created by the previous group in the deliverables, the Development Tools table produced by the previous team corrupted when converting the PDF documents to google docs. As such we had to rebuild the table however we've taken the effort to reconstruct this table as closely to the original as possible.

With regards to the *development tools* used, these were essentially identical to the tools we planned to use for further development, which was a motivating factor for the choice of project we went with. However, under version control where Github is referenced, we specified our use of continuous integration through github, adding it onto the information already included on the previous team's use. However, the previous team never mentioned

PlantUML here despite it being a significant component in the development of the project, therefore we added an additional table entry being PlantUML.

The previous group had a different approach to *Team Roles and Organisation* compared to us, with their roles being tailed more towards monitoring specific tasks than generalised roles within a team setting that allow the team to work efficiently. To reflect this, we updated the documentation to reflect our team roles (carried over from assignment 1) with reasoning as to why these roles are better suited to our project than the previous group's roles.

The *systematic plan* used by us and the previous group both consisted of a gantt chart used to manage time spent on different parts of the project. Since our systematic plans were effectively the same there was no need to majorly change this section of the deliverable. However, our group created another gantt chart and added it at the bottom of the document to reflect how we further planned out the time to be spent on different tasks.

## Risk Assessment and Mitigation

When inheriting this document table data would become corrupted; multiple attempts were made to convert the PDF in different formats, however the table would always break when attempting to do so. An example of this is seen below:

| Likelihood (%) | Severity (%) Mitigation Strategy |
|---|---|
| **40% -** Due to the winter months this project is completed in, it is reasonably likely that a team member | **40% -** If the illness is communicated, the team members task can be easily picked up by other team | The Project Lead will reassign tasks for short-term periods of unavailability. For longer periods, they will check in |

In light of this, our first priority was to completely reconstruct the table, keeping text the same but manually copying it over from the provided PDF. Due to this some slight inconsistencies may exist between the previous document and our updated version. Once the table was fully restored we began considering updates to the design and contents of the document.

Like our group, the prior group also opted to use a risk register in tabular form as a way of keeping track of different potential risks. We decided that their risks were suitable, however some additional risks had to be considered as we proceeded with the project, especially with the new functionality being implemented. As for the preexisting risks, we had to reassign risk owners for each risk. These owners have been stated in bold below the previous risk owners for each risk in the table.

With the introduction of Continuous Integration to our project came a new risk to consider. Continuous integration helped in the testing of the project, however there always came the

risk that it would suffer a failure, either due to improper setup or an issue in the code. As such, a new risk was added to the bottom of the risk table to reflect this.

Pre-existing risks had mitigation strategies that, whilst effective, required some slight tweaking to work for our team. For example, some of their mitigation strategies involved use of Slack; an application for team communication which we opted not to continue with. As such, where Slack is referenced we've included reference to our alternate software of choice (in this example that would be Discord).

Similarly, some risks were handled based on different team roles and the responsibilities they held. In the table wherever a team role is mentioned our equivalent role is listed in brackets for the sake of clarity. Furthermore, some risks that they opted to handle as a team were tied to a specific individual for our development for effective organizational purposes.

A feature which was missing from the previous team's table was clearer indication of the different risk levels of tasks - previously risks were represented solely as percentages. Our group preferred an approach that was easier to understand visually, for which we colour coded different risks based on their likelihood and severity.