Change Report

Cohort 2 - Group 7 - ‘Mikey and the Freemans’

Members: Louis Burdon, Varun Nayak, Alex Nevard, Sam Russell, Gaoman Zhu, Vivaan Kampani, Teva Geffen, George Overton

Planning:
Firstly, we defined the project's scope - a crucial first step. This mainly for us consisted of reviewing and analysing team 10's deliverables and GitHub repository. Before choosing this group we did a surface level analysis on every group in the cohort. After this group was chosen we dove deeper into how things stood, as this would really define the direction of our project. After the deliverable review we broke the anticipated work into manageable chunks and allocated this to team members. We did this using a google docs table and meetings to ensure deadlines, task scope, overall document quality and ownership. Furthermore, as a team, we committed to a weekly meeting, with supplementary video meetings for the available team members. The meetings would ensure any updates could happen and where possible the team was in the loop. The schedule as a team was intentionally designed to be lightweight such that we could reschedule for any unavailable team members.

Tools:
Physical whiteboard, Instagram (for the people who couldn't attend the meeting in person), plantUML, Google docs tables, GitHub, mockito and piksel.

Conventions:
We generally use diagrams to explain our thought processes due to the nature of the project, but otherwise, we mostly used Instagram to communicate our thoughts. We took on a structured approach to our team meetings, accounting for who is present and who isn't so that we could then inform the missing parties of any progress made - which was done using meeting summary documents and also verbal updates - the tasks assigned to them or chase them up on the portion of the work they were due to complete for that week.

Tracking changes:
For tracking changes, we forked the version control repository of the previous team, and the development and testing team did a full runthrough of the code in its current state at the time of takeover. It meant that the group of people working on the code fully understood it and could move to planning out the strategy for the new features we were going to implement. We were able to track code changes through the commit history tab on the forked repository, but also kept a record of any updates we made in our own time through our team's chosen platform for communication, Instagram.

Reviewing work:
To ensure a high quality of work, we conducted structured peer reviews of code and documentation during our meetings. We committed to code to branches which were thoroughly reviewed by a team member before pushing to the main branch. Automated tests ensured code validity and that our functionality remained intact - As this was implemented early on, we limited the amount of manual tests that had to be completed. After each development sprint phase, in our team meetings we retrospectively would identify what worked well during the sprint and what needed improvement. Finally after reviews updated the project documentation to keep it up to date and aligned with all new amendments.

Integration and finalisation: We used the GitHub repo and a thorough branch protocol, commit messages and group chat messages - coupled with group meetings to assign roles and tasks.

Requirements

Original URL: https://alistudentgit.github.io/team10projectsite/docs/Req1.pdf
Updated-URL:https://docs.google.com/document/u/0/d/1Tg1Gclezq-MDrnzKd3zT0Lgh6ZaX3m1OKEraxmqHFK4/edit

We made several changes to the requirements document, as we were not present for the elicitation of requirements for this project, the main work was in refining the requirements to make sure that they matched the brief and updating the layout with researched presentation methods.

We started by adding a title page as outlined in the brief. We made the introduction much more concise by removing information not required in the brief such as individual requirements rather than how requirements were elicited. We used our own research into requirements specification and presentation as they still hold true and there was no existing research in the requirements by the previous group.

There were multiple ambiguous user requirements - these were updated to better reflect their requirements.

There were several requirements listed in the user requirement that did not fall under the user requirement banner; UR_Negative, UR_Positive UR_Hidden which were previously found in the user requirement have now been rewritten as functional requirements. There were also a number of requirements listed as user requirements that fell under the non-functional requirements heading; UR_Audience, UR_Licensed_Assets, UR_Standard_Computer, UR_Desktop_Inputs. These were rewritten with the correct user requirement in to cover that section and the rest of the requirements moved to the non-functional requirements section

FR_event counter was removed as there was no user requirement for it. There is no user requirement for a pause menu as a result the functional requirement for a pause menu was also removed. We combined the two dean elements as there was no clear need to have two requirements. FR_Fast was combined with FR_score as both covered the same user requirement.

There was a user requirement UR_University without a functional requirement so this was added. As we intended to add both achievement and leaderboard functionality to the game we wrote user requirements and functional requirements to reflect this

All of the descriptions of the non-functional requirements were not particularly concise or relevant so they were written to better reflect the requirements. We deleted NFR_time_constraint as it is a functional requirement not a non-functional requirement where it was already listed. Non-functional requirements were also poorly tied to user requirements so a more concise user requirement, UR_Game was added which reflected the user requirements better.

Architecture

Original URL: https://alistudentgit.github.io/team10projectsite/docs/Arch1.pdf
Updated-URL:https://docs.google.com/document/u/0/d/18xKvHkdMsuFw3Xl7MR5m13AJrqACPes A4gdmk2l01hA/edit

A huge number of changes were made to the Architecture document, as we drastically changed the structure of the code and all the diagrams in the original document were just images so we couldn't edit the existing diagrams. The code was significantly refactored so that features like event creation were far simpler to implement for our own benefit. This meant that we had to create an updated version of almost every diagram from scratch. There was also little explanation into what the diagrams represented and which requirements they linked to, so we added much more detail to that. The formatting of the original document was also largely irregular, making it difficult to read, so we almost entirely re-formatted the document.

We started by adding a title page, as this was something that was missing before. We then added an for an introduction, using a paragraph that was previously mislabeled, and largely extended what was previously written. This was to give an overview into the decisions we'd made when designing the architecture, such as using plantUML for our diagrams and justifying why we made those decisions.

Next, we improved upon the paragraph that explained the use of the class diagram to go into more detail. This also included discussing some of the most important classes within the game, such as the Event and RoomManger classes, as well as a brief discussion of which requirements they may relate to. We also mentioned the tweaks we'd made to the framework as well as the addition of some new classes, notably the Leaderboard and Achievements classes. Then, we used this information to create a new class UML diagram, laying it out in a clearer format to what was done before. There were a number of unnecessary methods such as getters and setters, so we removed these alongside a few connections between classes that didn't add any important detail. This meant that our new diagram is much more readable and it's clear to see how each class is connected to other classes. We also removed the notes on the diagram, as we felt that these were unnecessary at this point, as everything mentioned would be discussed in later diagrams. Reformatting the original UML diagram so it could fit on one page and leaving it in the document was a decision we made so as to make it easy to compare the new and old side by side and easily tell the differences.

After this was the Sequence Diagrams, to which we changed entirely. They previously had 5 diagrams that were hard to read and were loosely linked to different requirements. They also had 3 diagrams dedicated to different types of events which we condensed into 1 diagram as they were almost identical in layout. Now, we have 6 different Sequence Diagrams that comprehensively outline all key interactions that the user will come across when playing the game. These include all events that occur, the game being paused, using the main menu, the game starting, running out of time and successfully completing the game. On top of this, we've added a brief description of each diagram to clear up any potential confusion, and make it even clearer how the user and system can interact at different points. We also greatly improved upon how they linked it to their requirements, as after each diagram we

have a list of any requirements that have been achieved, as well as describing how they have done so. By doing this, we have managed to achieve every requirement within the 6 diagrams, other than the UR_Dean requirement, which was a creative decision. We reformatted how they'd presented their section of this project as well, by putting each diagram on a new page, with each one having a short description and a list of relevant requirements below it. This has made it much easier to read the diagrams as well as a lot clearer which descriptions are for which diagram.

There was much less change needed to the state diagrams. We did make new diagrams so we could add certain elements of the game but they are largely the same as the previous groups. We did the same for the component diagram, as we found that all the pre-existing diagrams were well-made. We did however remove the state diagram showing when the game ends, as we felt it was arbitrary and had already been discussed. Also, what the group had done previously was link these diagrams to more requirements, but we felt that the links were negligible in places, and we'd already discussed this in detail. So instead, we wrote an explanation for each of these diagrams into how they work, and how this impacted how a user would play the game. We felt that this was much more useful in giving a comprehensive overview of the game in its entirety.

Method selection and planning

Original URL: https://alistudentgit.github.io/team10projectsite/docs/Plan1.pdf
Updated-URL:https://docs.google.com/document/u/0/d/1TZR307NGy5QmUPvmAiwoOzqYpBWhgxfrtBh-JDMDelk/edit

Multiple changes were made to the method selection and planning document, due to the project acquisition. However, we kept some areas the same - such as some of the applications used, LibGDX framework for the development, Git and GitHub for version control, Google Docs to write all documents, and Instagram for all forms of communication. Since we were all in the same cohort, we had the same meeting times, and that stayed consistent throughout the project. All changes made are discussed below.

New roles, and areas were assigned to each person in the group - we followed a Scrum approach, and to avoid any confusion, each member of the team was assigned a task to work on initially. However, team members branched to different aspects to contribute in, to maximize efficiency and keep the project running well.

The document was rewritten in multiple areas, we started by adding a title page, all names/mentions of the previous team were removed to preserve continuity. We used PlantUML for the Gantt Chart, and Piksel to design assets. The applications used by Group 10 were not mentioned in the document. However we felt the need to mention them, in order to stay consistent with the product brief.

For communications, we primarily used Instagram, using the chat and call functionality, as opposed to Google meets, whenever realtime collaboration was needed. We decided to use one app only to maximize efficiency and felt it was unnecessary to use separate apps for texting and video communication.

We found that a Scrum methodology was better suited for this project compared to a Kanban approach, due to the iterative nature of the project. We found ourselves to be working in sprints, with weekly meetings to discuss what is needed from each member of the team, and reviewing progress the next week, compared to a continuous flow system.

We also decided to remove the UML work breakdown and Trello board diagrams. We believed that the Trello board was not the best approach to map out project tasks any longer, as it was better suited for a Kanban methodology, while we followed an agile Scrum approach. The UML work breakdown diagram was removed as we believed that we had appropriately explained all the work within the text, and in our Gantt chart diagram.

We created a new Gantt chart diagram, as there was a new timeline for the project, and many completely new tasks. We also made sure to add dates in our Gantt chart for better understanding of the timeframe, and the general workflow of the project - which tasks were being worked on and when. The Gantt chart was updated throughout the course of the project. We also added a References table, to allow readers to track what our sources are, which was consistent with our initial document as well.

Risk assessment and mitigation

Original URL: https://alistudentgit.github.io/team10projectsite/docs/Risk1.pdf
Updated-URL:https://docs.google.com/document/u/0/d/1LbXvc_7LQPnhAQ3DFHDITWTPxZoZDiuhBOFTmBKy4vQ/edit

When we inherited the project from the other groups, we really wanted to balance deliverable quality and content with that of the code. In that regard sometimes there are tradeoffs in some regards. On the whole, the risk register, and quality of it was very comprehensive - part of the reason we chose the group.

The first element of the risk assessment we updated was the structure and representation of the risks. The document downloaded as a pdf, and converted onto google docs, meant that editing the document would be challenging. Coupled with this, I found the structure of their table, and clarity and wording were not optimal. To combat this I restructured the table, updating the relevant fields and colour coded the risk significance and likelihood for quicker and easier data representation. The reason we as a team decided this was optimal was firstly for our regular group meetings. Every week (and sometimes twice weekly) we would review the old risks, updating their significance and likelihood, and so having colour coding for each risk really streamlined this process. Secondarily, we updated much of the risk wording and risk assessment process on the first page of the document. This was really important to the functionality of our team - as we found ourselves not in complete agreement with the previous team's processes as we thought they were slightly ambiguous and not optimal. So by updating the wording here really reflected our team's inner workings and strengths. The table's restructure meant that no time in our risk evaluation would be spent finding risks or decrypting previous work - as they had been overhauled and represented clearly. We also added a title page - something the previous team had missed.

The next element of the document we focused on was risk ownership. With the next chapter of the project, it was really important to allocate ownership to risks - and equally important to remove the previous team's ownership. With the change in project objectives, and even team member roles, it was important to re-evaluate each member's involvement in different sections of the assessment. With that would come new risk allocation to a given team member. Because we implemented this update, it meant during the development of our project, as we worked in sprints meaning for a week sprint to achieve an objective in project development we needed to be aware of who is managing a given risk, which the updated risk ownership was vital for. Overall this meant for our group we could effectively manage the risks identified, and with our regular meetings we could allocate any new risks we found to an owner with like-risks to their roles and already assigned risks.

Finally, upon evaluation of the previous team's risk register we found it while comprehensive - lacking in scope. Especially with new objectives for the second half of the project, we needed some new risks for each given risk category. In total we identified 5 risks mostly of medium likelihood and medium-high significance. These were promptly added to the risk register and allocated owners. The risks ranged from: performance drops(technology risks) to more user focused risks like screen motion causing disorientation - however most risks we defined were development focused (see risk register **R33-R38**) . The reason we did this was it would ensure for the remainder of the project all the likely risks had been identified, resulting in the most productive and high quality project delivery we could achieve as a team - ultimately it would solidify our success.