University of York Department of Computer Science
Engineering 1

Change Report

Cohort 3 Team 5 - alltheeb5t

Aaron Heald
Alex Gu
Arun Hill
Jade Stokes
Maksim Soshchin
Meg Tierney
Will Hall

Introduction

After taking over Group 2's project, we discussed and reviewed each deliverable as a team and decided on major changes that needed to be made to each, whether that was adding new items to the requirements and architecture, or changing the type of team organisations in the planning. We split into teams to handle each different deliverable who could then properly organise the specific changes to the deliverables or code. We used Github Projects to plan and organise priority of changes to the code, and we created a Change Log in Google Docs to do the same with the changes to the deliverables. This log was manually filled in by the team about the details of the change and its priority. This made it easier to look back when writing the Change Report to see which changes were made to the old deliverables and when.

Change Request Form	
Change Name:	Requirements - Adding New Requirements
Change Request Date:	23/11/24
Requested Change:	Adding the new user and system requirements for the second part of the project.
Priority Details (importance/how long it will take to implement/etc.):	This should be done quickly so it can influence the new architecture and implementation, it should take a few days to implement so should be completed quickly.
Accepted (Yes/No):	Yes

Fig. 1 - Example Change Log entry.

To track the progress of each change, we made use of Github commits and branches, as well as Github issues to track and communicate problems with the code whenever they were discovered. With the deliverables, we used Google Docs edit history and version control to see which changes were made to existing documentation and when, although we were careful to keep an original version of the deliverables too.

When reviewing the changes made to each part of the project, we would consult the Change Log or Github pull requests, as well as discuss with the team any changes we thought were appropriate so that everyone was on the same page about each deliverable's progress. This was especially important when writing the architecture and the requirements as this was largely happening in tandem with the development, so we made sure to make sure everyone was informed and kept the Change Log as up to date as we could by reviewing it at the beginning of each meeting.

Requirements

Original Document: https://alltheeb5t.github.io/assessment-2.github.io/assessment1/pdf/Req1.pdf
New Document: https://alltheeb5t.github.io/assessment-2.github.io/assessment2/change_report.html

Since the requirements for this project had been expanded due to the updated brief, there were a lot of new requirements to add to the project, as well as changes that needed to be made to the original requirements. We also decided to change the classifications of certain requirements as we felt that they were better suited in different categories. All of these changes have been documented in the new requirements document linked above.

Group 2 had already added in requirements for a leaderboard as their added feature in part one of the assessment, so we ended up just changing the wording on most of their requirements surrounding that to match the current outline in the project brief.

Minor Changes:

- We decided to reword the descriptions of some of the User Requirements as we felt that they were too focused on the system rather than the user, for example in **UR_EXPERIENCE**
- We updated the descriptions that Group 2 had already put down for the new Leaderboard requirements (**UR_LEADERBOARD, FR_LEADERBOARD, NFR_SYSTEM_REQUIREMENTS, FR_SAVES)** so that they would better fit the current scope of the project.
- We changed the categorisation of certain requirements that seemed to fit better elsewhere, for example **NFR_SAVES** was describing a save system for the scores of each player, which was a function of the game that was able to be properly implemented rather than an outside non-functional requirement, so it became **FR_SAVES** with a similar description. The opposite was true of **FR_SYSTEM_REQUIREMENTS** which was about the type of hardware the game can run on. Therefore, we changed it to **NFR_SYSTEM_REQUIREMENTS** instead.

Major Changes:

- **Requirements: Additional Requirements - Already Implemented Features**
  We added some requirements that had already been implemented by Group 2 but weren't listed, such as finances and satisfaction. This made it easier for our development team to properly follow along when they were fixing any bugs to do with these features. These new requirements include **UR_SATISFACTION, FR_COUNT,** and **NFR_METRICS.**
- **Requirements: Additional User Requirements - New Specifications**
  Since the project brief called for new user requirements, we added **UR_EVENTS** and **UR_ACHIEVEMENTS**. Since they were specifically mentioned in the project brief, we listed them as 'shall' requirements.
- **Requirements: Additional Functional Requirements - Events**
  We had lots of ideas for different in-game events, and so there were a lot of specific requirements that pertained to how these events would be executed, some of which are shown below with the full list in the new document linked above, as well as the general functionality of events within the game, such as how often they would appear, how they would be displayed, the result of the event, and how players could make choices during an event (**FR_EVENT_GENERATOR, FR_EVENT_DISPLAY, FR_EVENT_RESULT** and **FR_EVENT_CHOICE** respectively).
- **Requirements: Additional Functional Requirements - Achievements**
  Just like with the events, we had lots of ideas for different achievements that a player could earn in-game that would enhance the challenge and user experience of playing the game over to earn a higher score. Some of these are outlined in the table below.

We wanted to add a mix of achievements for different aspects of the game, such as running out of money, keeping a steady satisfaction, or focusing on different building types. We also added a requirement on how achievements would be displayed to the player. We settled on having an in-game menu that players could browse through to see the achievements they unlocked (**FR_ACHIEVEMENT_MENU**)

New User Requirements

UR_MENU	The user shall be able to interact with several menu screens throughout the game.	Shall
UR_EVENTS	The user shall be able to experience random events in the game that will affect their playthrough differently.	Shall
UR_SATISFACTION	The user will be able to increase or decrease a metric for student satisfaction which will influence their final score.	Shall
UR_FINANCE	The user will be able to gain money as the game progresses that will allow them to build more things on campus.	Shall
UR_ACHIEVEMENTS	The user shall be able to unlock several achievements within the game based on their actions when playing.	Shall

New Functional Requirements

FR_SATISFACTION	The system shall keep track of student satisfaction as a percentage and update based on the environment, displaying this to the player.	UR_SATISFACTION
FR_PAUSE	The system shall allow the user to pause the game timer and view a menu of options at any point.	UR_MENU, UR_TIME
FR_TUTORIAL	The system shall display a brief introduction of the game with controls to the user upon starting a new playthrough	UR_MENU
FR_STUDENT_FINANCE	At the beginning of each in-game semester the system will give the user a lump sum of money to spend on new buildings.	UR_TIME, UR_FINANCE
FR_BUILD_TIME	The system shall take a set amount of time to 'build' a new building.	UR_BUILDINGS
FR_INCOME	The system shall give users a small amount of money over time for each Eating Building built.	FR_BUILDINGS, FR_COUNTER, FR_FINANCE
FR_COUNT	The system shall keep track of the total number of buildings currently on the map for the user to see.	UR_BUILDINGS, UR_COUNTER
FR_EVENT_GENERATOR	The system will randomly choose an event to occur every in-game year	UR_EVENTS
FR_EVENT_RES	The system will assign a positive, negative or neutral	UR_EVENTS,

ULT	result after an event has occurred	UR_SATISFACTION
FR_EVENT_DISPLAY	The system will display a brief description of the event that is currently occurring and the effect of it on satisfaction and finances.	UR_EVENTS
FR_EVENT_CHOICE	For certain events the system will allow the user to choose between two options to affect their game.	UR_EVENTS
FR_STRIKE_EVENT	The system shall display an event where lecturers go on strike. Any buildings being built will pause until the strike ends and satisfaction will decrease.	UR_EVENTS UR_BUILDINGS UR_SATISFACTION
FR_STRIKE_CHOICE	The system shall give the user a choice to end the strike early by paying the lecturers more money (and buildings will cost more to maintain) or letting it run its course which decreases satisfaction until it ends.	UR_EVENTS UR_BUILDINGS UR_FINANCE UR_SATISFACTION
FR_ACHIVEMENT_MENU	The system shall have a menu that the player can view containing all possible achievements in game.	UR_ACHIEVEMENTS
FR_BM_UNLOCK	The system shall unlock the achievement “Bare Minimum” if the user only places 1 of each building type throughout the game.	UR_ACHIEVEMENTS, UR_COUNT
FR_UNLUCKY_UNLOCK	The system shall unlock the achievement “Unlucky” if the player encounters 3 negative events in one game.	UR_ACHIEVEMENTS, UR_EVENTS
FR_GAME_END	The system shall end the game after 5 minutes.	UR_TIME
FR_END_SCREEN	The system shall display the final satisfaction score to the player at the end of the game.	UR_SATISFACTION, UR_MENU
FR_WIN	The system shall show the player whether they have ‘won’ the game if their satisfaction is above 70% at the end of the game.	UR_SATISFACTION

New Non-Functional Requirements

NFR_METRICS	The system will update metrics such as student satisfaction and finances consistently.	Metrics are updated on the screen within a second of changing.	UR_FINANCE, UR_SATISFACTION

Architecture

Original Document: https://alltheeb5t.github.io/assessment-2.github.io/assessment1/pdf/Arch1.pdf
New Document: https://alltheeb5t.github.io/assessment-2.github.io/assessment2/change_report.html

Due to the need to implement the full product brief for Assessment 2, the architecture document required some major changes to reflect the new structure and functionality of the game.

As a result of this, new sections were added to show the evolution of the system's architecture between Assessment 1 and 2. These sections detailed the structure of several of the game's main features such as achievements, in-game random events, the leaderboard, buildings and the tracking of key game metrics (i.e. time, student satisfaction, money).

Some minor changes were also made to the explanation of the architecture and design process for Assessment 1 in order to maximise clarity and brevity, as well as a justification for using the PlantUML tool to model class diagrams.

Minor Changes:

- Some minor formatting changes were made to the structure of the explanations for each class diagram in the document in order to increase readability.
- The narratives detailing design evolution and the creation of the use-case diagram have been rewritten to make them as clear and succinct as possible.
- A justification for the use of PlantUML to model class diagrams has been added under 'Class Diagrams' on the document in order to more closely follow the brief set out for Architecture in the assessment paper.

Major Changes:

- Throughout the entire document, many direct references to the game's requirements have been added in order to properly justify the features of the game's design.
- New class diagrams (and their descriptions) have been added to the end of the document in order to fully explain the structure and behaviour of the system now that the product brief has been fully implemented for Assessment 2. These new sections include:
    - The overall final structure of the game
    - How singleton classes are used to maintain and manipulate the game's state
    - How buildings are created, managed and destroyed
    - How achievements are tracked and displayed
    - How events are triggered and their effects enacted
    - How key game metrics such as the in-game time, student satisfaction and the player's money are monitored and changed throughout the course of each playthrough
    - How a leaderboard is maintained and made available for users to view

Method Selection and Planning

Original Document: https://alltheeb5t.github.io/assessment-2.github.io/assessment1/pdf/Plan1.pdf
New Document: https://alltheeb5t.github.io/assessment-2.github.io/assessment2/change_report.html

Since all the members of the team for this project have changed, some significant changes had to be made to the Methods and Planning to adjust for this. However, some things in this section did not need to be changed, as Group 2's had the same preference that we did for the scrum methodology, which is what they seemed to describe in the original methods and planning document. We also decided to keep the same meeting dates that Group 2 had, those being on a Monday and a Friday, and we also still used GitHub for version control and the Google Suite for documentation and housing a shared drive of work.

Minor Changes:

- The document has been rewritten in places where names of team members and deliverables need to be changed in order to keep continuity with the rest of this part of the project.
- We decided to use a mix of IntelliJ and VSCode rather than solely VSCode for the IDE's in our project as our development team were specialising in different areas of the project and we wanted everyone to just use whatever they were most comfortable with at this point.
- We used different programs to create the assets for our project, mainly using Microsoft Paint rather than Aseprite and Photoshop as the team were not familiar with these programs.
- When creating diagrams for the project deliverables, the team only used PlantUML instead of a mix of PlantUML and Goodnotes 6 as we were familiar with the former only.
- For communication, we decided to use Discord as well as WhatsApp as we found that sending images and having more focused conversations about certain aspects of the project was a lot easier over Discord.

Major Changes:

- **Methods & Planning: New Gantt Charts**
  Since there was a new timeframe for the project with many new tasks and a new team, new Gantt Charts had to be created to organise the team's workflow. We tried to keep the Gantt Chart in as realistic of a timeline as we could, so that people were not rushing to complete tasks but that everybody should have had enough to do throughout the duration of the project. These new Gantt Charts were updated with progress and any changes throughout the project.
- **Methods & Planning: Team Organisation Method**
  We decided to use a different organisational method for our team when deciding how people would be assigned parts of the project. Rather than allocating roles based on marks, we tried to keep everyone in a role similar to the one they were in for the first part of the project, except for the new deliverables where we simply asked members of the team which part they'd prefer to be working on. We felt that although this didn't give everyone an exactly equal number of marks to work on, everyone would be more comfortable in an area that they know most about already and can just get on with. For example, our implementation team stayed the same as the first part of the project.
- **Methods & Planning: Role Allocation**
  As mentioned above, we decided to keep people in roles that they felt the most comfortable in. We decided also to try and keep the number of team members working on one section of the project as low as we could, so that everyone had enough to do. Our final role allocations for this part of the project were:

- **Website: Maksim**
- **Change Report: Meg** (Requirements, Methods) **& Arun** (Architecture, Risk Assessment)
- **Implementation: Will, Jade, Aaron**
- **Testing: Alex**
- **User Evaluation: Maksim**
- **Continuous Integration: Will**

Risk Assessment and Mitigation

Original Document: https://alltheeb5t.github.io/assessment-2.github.io/assessment1/pdf/Risk1.pdf
New Document: https://alltheeb5t.github.io/assessment-2.github.io/assessment2/change_report.html

The Risk Assessment and Mitigation report was updated in some significant ways due to the change of team working on the project and the need to produce a different set of deliverables for Assessment 2. As a result of this, all risk owners had to be changed to match the new team and a new risk was added for tracking changes to the code using continuous integration methods.

Apart from this, the structure of the risk register was modified to aid clarity and readability by refining risk types and rating each risk using a risk matrix (1-9 scale) to more clearly signify overall significance.

Small edits to some risks' Impact and Mitigation sections were also made to increase the overall readability of the risk register and the clarity of those sections.

Minor Changes:

- The description of the risk management process undertaken during the project was edited to better explain it by adding an image of a flowchart demonstrating each step of the process.
- The phrasing of the Impact sections of R1, 2, 5, 6 and 8, as well as the Mitigation sections of R2, 4, 6 and 7 were modified to increase the clarity and brevity of those risks' descriptions.
- The formatting of the risk register table has been modified slightly to make it easier to read and assess the risks listed.
- References one and three have been added to the report to properly cite the new changes made.

Major Changes:

- The risk types were changed to include Schedule, Team, Requirements, Code and Tools instead of Human and Product in order to help refine the classification of each risk and therefore better describe them.

- The Risk Matrix metric was added to the risk register in order to help readers quickly assess the overall significance of a risk. This change was implemented by rating each risk on a scale of 1-9 using the risk matrix (shown in the new risk report) to represent each risk's likelihood and potential impact in a single value. These values were then added to the register table in a new column, 'Risk Matrix', that was colour-coded (red, amber and green) to clearly demonstrate the risk's importance.

- Each risk's owners were updated to correspond with our team members as opposed to those of the previous team. This therefore ensured that, for every risk, one or more team members were assigned to handle its migitation.

- The risk R9 was added in order to highlight the potential issue of tracking and merging conflicting code versions, which could become a problem while the implementation team refactored, debugged and extended the previous team's code. The mitigation strategy for this was to keep commits small and regular and to use continuous integration methods to preserve code integrity and prevent branch divergence while team members worked on implementing or testing different aspects of the product brief simultaneously.