UNIVERSITY OF YORK

DEPARTMENT OF COMPUTER SCIENCE

Change Report

Cohort 2 - Group 17 (Rich Tea-m 17)

Group Members:

George Jopson
Ben Slater
Meg Tierney
William Potts
Jamie Burgess
Seyi Towolawi
Alex Staicu

Introduction

When we first took over Team 16's project, we knew there were several different changes we needed to make to the original deliverables, as well as parts of the project we would have to update when the updated project brief was released. We made sure to agree on a change management system that would allow everyone to consistently keep track of the changes made within each separate part of the project. We used a Google Doc where we updated each change with key information, like which part of the project it concerned, when we made the change, and what type of change it was. We also ensured that these changes were colour coded so that we could quickly see if something had been added, updated, deleted or changed at a glance.

Change Request Form	
Change Name:	Risk Assessment + Mitigation: Add Brownfield Development Risks
Date:	22/04/2024
Requested Change:	Add risks to the risk register that consider the fact that we are now handling a brownfield project.
Change Assessment (priority/implementation effort/etc):	This will take a reasonable amount of effort, but is a high priority task as the major change that needs to be considered in the updated risk register is that we are now dealing with brownfield development.
Change Acceptance (Yes/No):	Yes

Fig. 1 - An example of a change request.

Setting out each change request in this way also ensured that there was no confusion with each change, as every change would have consistent and concise answers as to what exactly was being changed in the project.

Using a Google Doc to store these changes was a decision we came to because we could see who made each change in the project so that if we had any questions about something that was being changed, we could easily see the person making that change and ask them about it separately.

We also kept track of changes by giving a brief summary of any progress or updates we had each made at the beginning of each group meeting. This allowed us to have a good idea of where each person was in the project and if everything was moving within the planned time frame.

The following document is a summary of the changes we have made to the Requirements, Architecture, Method Selection & Planning, and Risk Assessment & Mitigation documents in the second part of the project. Changes are fully described and partially shown, with links to the updated documents that display the changes in full. Some examples of changes have been given in this document in order to give a brief idea of the changes we have made in the project documents.

Requirements

Original Doc: https://megant2004.github.io/ENG1-T17-Website/Part2/Assets/Deliverables/Req1.pdf
New Doc: https://megant2004.github.io/ENG1-T17-Website/Part2/Assets/Deliverables/Req2.pdf

Due to new features being outlined in the project brief for part 2, it was important that the project requirements changed so that they could be implemented properly. We changed most of the naming conventions that Team 16 used, as they weren't so clear in places. All of these changes have been documented in the new requirements document on the project website.

Minor Changes:

- We modified two functional requirements to better suit our changing requirements. **FR-MENU-2** was changed from no games will be saved to the end score of a game will be saved for the purpose of adding this score to the leaderboard. **FR-GAME-PLAY4** was also changed from one leisure location being needed to three, as per the updated project brief.
- We modified the names of several requirements that we felt they weren't immediately descriptive of what the requirement was, especially in the cases where requirements were numbered (e.g. FR-GAME-PLAY 1 to 4 became **FR-SLEEP-LOCATION, FR-STUDY-LOCATION, FR-EATING-LOCATION** and **FR-LEISURE-LOCATION**). These are clearly marked in the new requirements document alongside the old name in brackets so the implementation team could ensure there was no confusion during the changeover of names.
- Some descriptions of requirements have been updated to explain more about the requirement (i.e. **UR-SETTINGS** or **UR-ACCESSIBILITY**) or to better reflect the requirement (i.e. **FR-DEVICE**). These are clearly marked in the new document.
- We merged **FR-SLEEP2** and **FR-ENERGY2** into **FR-NO-ENERGY** as they were the same requirement worded in reverse of each other. We also merged **FR-MENU1** and **FR-MENU4** into **FR-OPTIONS**.

We didn't feel the need to add any new Non-Functional Requirements as the system still largely needed to do the same things as before, even with the added features.

As pointed out by Sommerville in "*Software Engineering*", it is essential to consider "if the benefits of implementing new requirements are justified by the costs of implementation" during change management [1]. For this project, the addition of new features to the existing product brief meant that it was essential to implement new requirements. These two new features, the addition of a leaderboard and streaks/achievements were the source of all major changes and additions to the requirements.

Major Changes:

- **Change: Requirements: Additional User Requirements - New Specified Features**
  The new features in the project facilitated the addition of two new User Requirements. These were **UR-LEADERBOARD** and **UR-STREAKS**. We felt it was important to include these as requirements we shall be implementing since they were specified within the updated project brief.
- **Change: Requirements: Additional Functional Requirements - Leaderboards**
  When deciding on the new functional requirements for the project we decided that there were three main aspects to the leaderboards that would be displayed. Those were displaying the leaderboard featuring the top 10 user's scores, saving the score of a user after their playthrough ends (which we incorporated into the change for **FR-SAVED-DATA,** prev. FR-MENU-2), and updating the leaderboard if the saved score is above any of the existing scores on the leaderboard. These changes correspond to **FR-LEADERBOARD-DISPLAY,** and **FR-HIGH-SCORE** respectively.
- **Change: Requirements: Additional Functional Requirements - Streaks/Achievements**

For the streaks, we needed to consider the functional requirements for each different type of streak we'd want to have in the game, two examples of which have been outlined below. We also needed requirements considering how a streak would be counted and decided that they must be done (at most) three times on independent days to count, so you can't gain any given streak on day 1 of the game. This is **FR-STREAK-ACTIVE.** The final thing to consider was the displaying of achievements at the end of the game depending on the streaks you'd earned. Once the player completes an action a certain amount of times, the corresponding achievement will be gained and subsequently displayed at the end of the game. These new requirements are **FR-ACHIEVEMENT-UNLOCK** and **FR-ACHIEVEMENT-DISPLAY.**

- **Change: Requirements: Additional Functional Requirements - Scoring System**
  In order for the game to have a leaderboard, we need to have a score in the first place. Team 16's game displayed the number of times an activity was done. But we need to give each activity a number of points so that we can get a score displayed at the end of the game. We added **FR-SCORE** and **FR-END-SCREEN** to describe this.

New User Requirements

UR-LEADERBOARD	The user shall be able to save/see their own and other user's scores on a leaderboard.	Shall
UR-STREAK	The user shall be able to gain secret achievements throughout the game by completing actions repeatedly.	Shall

New Functional Requirements

FR-LEADERBOARD-DISPLAY	Top 10 scores from previous playthroughs will be displayed.	UR-LEADERBOARD
FR-HIGH-SCORE	The system shall update the leaderboard if a new high score is achieved.	UR-LEADERBOARD
FR-STREAK-FLOWERS**	The player will earn a streak by smelling the flowers on campus 5 days in a row.	UR-STREAK, UR-INTERACT
FR-STREAK-STUDY	The player will earn a streak by studying for their exams 5 days in a row.	UR-STREAK, UR-INTERACT
FR-STREAK-FLOWERS	The player will earn a streak by smelling the flowers on campus 5 days in a row.	UR-STREAK, UR-INTERACT
FR-STREAK-ACTIVE	The system will only count actions towards a streak if the action is performed on consecutive days rather than all in one day.	UR-STREAK, UR-INTERACT
FR-ACHIEVEMENT-UNLOCK	The system shall award the player achievements if they successfully complete a streak.	UR-STREAK
FR-ACHIEVEMENT-DISPLAY	The system shall display all achievements gained by the player at the end of a playthrough.	UR-STREAK
FR-SCORE	The system will keep a score for the player, this will increase when the player completes an activity.	UR-MENU
FR-END-SCREEN	The system shall display the score of a player and how many times they completed an activity at the end of the game.	UR-MENU

Architecture

Original Doc: https://megant2004.github.io/ENG1-T17-Website/Part2/Assets/Deliverables/Arch1.pdf
New Doc: https://megant2004.github.io/ENG1-T17-Website/Part2/Assets/Deliverables/Arch2.pdf

Due to the new features outlined in the project brief for part 2 of the project, the architecture for this project must be changed in order to fulfil these new features. Since the initial architecture document was finalised, we have made many additions and slight alterations to the architecture.

Major Changes

- **Change: Architecture: New Screen Classes**
  To better match the updated requirements, we created more screens to add the newly required functionality. These new screens are LeaderboardScreen and SaveScreen. Both new screens implement Screen, and are used to allow the user to save their high score along with their name, and view the top ten scores in the new leaderboard menu, which can be accessed via a new button on the main menu screen. Fig. 1 - Final Class Diagram in the architecture document has been updated to show how the new classes work within the Responsibility-Driven Design architecture.
- **Change: Architecture: New Events in EventManager**
  To add more functionality to the game we added a few more events to the game for the player to participate in. These include flowersEvent, busStopEvent, gymEvent, duckPondEvent, libraryEvent and townBusStopEvent. These new events along with the initial events already implemented are shown in Fig. 2. All of the new events, in the same way as the previously implemented events make use of a DialogueBox when the player interacts with them. For example, FlowersEvent is used to allow the player to 'smell the flowers' for a chosen period of time, and is classed as a recreational activity in the game logic.
- **Change: Architecture: Achievements in EventManager and GameOverScreen**
  As the player completes the related events the number of times they are achieved is tracked by EventManger, then at the end passed to GameOverScreen to check if they meet the minimum quantity requirements. The achievements achieved by the player will be displayed around the GameOverScreen window border, with text underneath explaining how they got the achievement.
- **Change: Architecture: Changes to State Diagram for Screens**
  To account for the newly added screens, we had to change the Fig. 5a - Final Screen State Diagram, as well as 'Fig. 5b - Final Screen State Sub-Diagram for Starting game to returning to menu' to allow for navigation between the new menus. For example, from MenuScreen, you can now go to CreditScreen by clicking on the Credits button, and you can go back to the main menu again by pressing the exit button from within the CreditScreen.

Relating Architecture to Requirements

New User Requirements

UR-LEADERBOARD	The LeaderboardScreen class manages the display of player rankings and scores. It interfaces with the game's data management systems to retrieve and sort scores, providing a dynamic and competitive aspect to the game. This screen is accessible from the main menu.
UR-STREAK	The game includes a system to track activity streaks, using a HashMap *'streaks'* to log what streaks have been completed by the player.

New Functional Requirements

FR-STREAK	EventManager stores that the player has completed a specific activity, allowing them to complete the same actions in a row to earn a streak.

FR-STREAK-STUDY	The player can earn a studying streak by studying every day. Streaks progress is recorded in EventManager, and are presented on the gameOverScreen.
FR-STREAK-FLOWERS	The player can earn a streak by completing the 'smelling the flowers' action 5 days in a row in their playthrough. Streaks progress is recorded in EventManager, and are presented on the gameOverScreen.
FR-STREAK-TOWN	The player can earn a streak by going into town 5 days in a row in their playthrough. Streaks progress is recorded in EventManager, and are presented on the gameOverScreen.
FR-STREAK-SHOP	The player can earn a streak by buying something from Nisa at least 5 days in a row in their playthrough. Streaks progress is recorded in EventManager, and are presented on the gameOverScreen.
FR-STREAK-LIBRARY	The player can earn a streak by buying studying at the library at least 5 days in a row in their playthrough. Streaks progress is recorded in EventManager, and are presented on the gameOverScreen.
FR-STREAK-NIGHT-OWL	The player can earn a streak by being awake, at least 5 times in a row, between the hours of 20:00 and 00:00 throughout their playthrough. Streaks progress is recorded in EventManager, and are presented on the gameOverScreen.
FR-STREAK-EARLY-BIRD	The player can earn a streak by being awake, at least 5 times in a row, between the hours of 00:00 and 08:00 throughout their playthrough. Streaks progress is recorded in EventManager, and are presented on the gameOverScreen.
FR-STREAK-EATING	The player will earn a streak by eating 3 meals a days 3 days in a row. Streaks progress is recorded in EventManager, and are presented on the gameOverScreen.
FR-ACHIEVEMENT-DISPLAY	Achievements unlocked are displayed within the GameOverScreen, showing detailed icons for each achievement which was achieved.
FR-SCORE	The score variable in GameOverScreen calculates a final score based on player performance and game events. This class also interacts with the LeaderboardScreen to update high scores and rank players.
FR-END-SCREEN	The GameOverScreen class presents the final game results, including score summary, achievements unlocked during the session, and options to restart the game or exit. This screen transitions from the GameScreen upon game completion.
FR-LEADERBOARD-DISPLAY	The LeaderboardScreen class retrieves player scores from a scores.csv file and displays them in a sorted list. This screen is accessible from the main menu and updates upon game completion.
FR-HIGH-SCORE	The LeaderboardScreen will show the player who has scored the highest at the top, until their score is beaten by another player, at which point they will move down the leaderboard.

Method Selection and Planning

Old Doc: https://megant2004.github.io/ENG1-T17-Website/Part2/Assets/Deliverables/Plan1.pdf
New Doc: https://megant2004.github.io/ENG1-T17-Website/Part2/Assets/Deliverables/Plan2.pdf

Some significant changes have been made to this document, due to the new deliverables and work required to be completed for this assignment, as well as the switch to a Brownfield software project. We kept the structure of this document the same to the original, as we felt they outlined the details of their planning and organisation clearly.

Major Changes

- **Change: Method Selection and Planning: New Work Breakdown diagram**
  Created a new work breakdown diagram to reflect the new work requirements of the assessment. For each deliverable described in the assessment brief, we have broken them down into separate tasks according to the descriptions of each deliverable in the brief. For example, for the change report deliverable, the representative tasks are to summarise the changes made and to justify changes for each deliverable, as this is how the change report deliverable is described in the brief. The previous team states in the document that they broke down each deliverable in a similar way, so I followed a similar methodology as I felt their work breakdown document was informative.
- **Change: Method Selection and Planning: New deliverable and task tables**
  Replaced the old deliverable and task tables with new tables that reflect the tasks stated in the new product brief. Each task represents a node in the work breakdown document. For example, tasks T5.1-T5.4 represent tasks relevant to the user evaluation deliverable, as the fifth branch on the work breakdown diagram represents user evaluation and that branch has 4 nodes. Due to the switch to a Brownfield project, all of the tasks stated in these tables reflected work that had already been completed, and this meant new tables had to be created to reflect the new work required.
- **Change: Method Selection and Planning: New Gantt charts**
  A new Gantt chart will now be created and edited for each sprint that reflects the work completed during assessment 2. The bottom paragraph will be updated weekly with the progress the team is making. Each task described in the Gantt chart was taken from tasks assigned during Sprint reviews, as well as tasks described in the work breakdown document. For example, the testing section of the Gantt chart concerns the test2 deliverable, and the tasks described in that section both include tasks assigned during sprint meetings (e.g. perform tests for new requirements) but also tasks described in the work breakdown (e.g. produce report). We had to make new Gantt charts because, due to the switch to a Brownfield project, all of the tasks described in the old Gantt charts represented work that had already been completed.
- **Change: Method Selection and Planning: Organisational roles**
  Added details about team organisational roles to the work organisation section. These were selected according to the scrum guide [2] and are as follows (full descriptions are in the document):
    - George/Seyi: Scrum master and team leadership
    - Seyi/Will: Product owner
    - Jamie/Ben: Secretarial tasks
    - Meg/George: Project editing

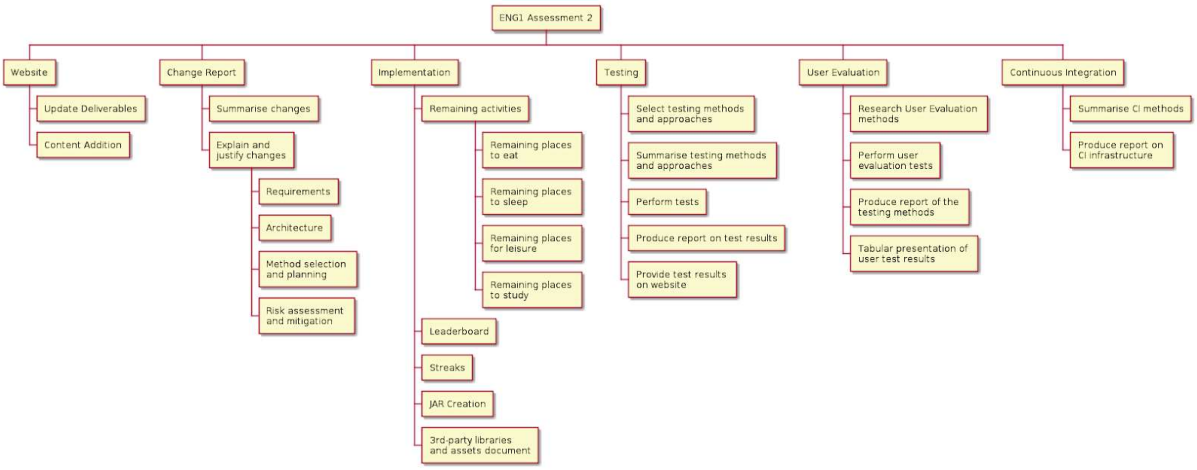
  The idea is that the first person acts as the main person responsible for that role in team organisation, and the second person operates as a co-leader if the first person is unable to complete their tasks. These roles were selected according to how our team was organised during assessment 1, and these changes were necessary because the previous team's organisation

Minor Changes

- **Change: Notion description and justification**
  Added a short description and justification for the use of Notion, this was missing from the original document and since we use Notion to help us schedule tasks I feel this change was necessary.
- **Change: Method Selection and Planning: Update deliverable names**
  Updated the deliverable names and lead roles to reflect both the current team and the new deliverables required. The old names were redundant and needed to be changed.
- **Change: Method Selection and Planning: Work Organisation**
  Rewrote the work organisation section of the document so that both the names of the deliverables being worked on and the names of the people completing them now reflect the current team and assessment.

Team 16 also used the scrum methodology, with an identical method for planning, completing and reviewing sprints, so nothing was changed about this section. We also used the same tools as Team 16 in this part of the assessment, so most of the development and collaboration tools section of the document has been left unaltered.

New Work Breakdown document



New Deliverables table

ID	Title	Due date	Description	Visibility	Relevant Tasks
D1	url2.txt	22/05	Website	Shared	T1, T4.5
D2	Change2.pdf	22/05	Change Report	Shared	T2
D3	Impl2.pdf	22/05	Implementation	Shared	T3
D3.1	Code	22/05	Implementation	Shared	T3
D3.2	Executable JAR	22/05	Implementation	Shared	T3.4
D4	Test2.pdf	22/05	Testing	Shared	T4
D5	User2.pdf	22/05	User Evaluation	Shared	T5
D6	CI2.pdf	22/05	Continuous Integration	Shared	T6

Risk Assessment and Mitigation

Original Doc: https://megant2004.github.io/ENG1-T17-Website/Part2/Assets/Deliverables/Risk1.pdf
New Doc: https://megant2004.github.io/ENG1-T17-Website/Part2/Assets/Deliverables/Risk2.pdf

For the risk assessment and mitigation we made some minor changes to the project:

- Update risk owners to reflect the current team. Risk ownership was assigned based on what team members were working on the area of the project the risk was based on. This ensures that any changes to the risks are spotted, as the team member in charge of the risk will be intimately involved in the area involving it.

There were some areas of potential risk that due to the project choice were avoided. As Sommerville points out [3], taking over documentation can be problematic if teams used different styles of methodologies and so produced wildly different types of documentation. This can make the handover process problematic as teams might have to maintain documentation that they don't find useful. However, the previous team used Scrum similar to us and so this wasn't a problem.

We also made some more major changes:

- **Change: Risk Assessment + Mitigation: Update Risk Categories**
  The risks in the inherited project were split into Project and Product risks. These categories are very broad, and so don't provide a huge amount of information about what style of risk is in that category. It also could lead to risks being missed, and we didn't consider all aspects of the project and just considered "Project" and "Product" risks. To solve this, we further categorised risks suggested by Sommerville [4]:
  - **Estimation:** Arising from incorrect estimates of time/resources needed to build project
  - **People:** Arising from people in development team
  - **Requirement:** Arising from changing requirements
  - **Technology:** Arising from risks using external developed hardware/software
  - **Tools:** Arising for software tools (like IDEs) used to develop project

  From this I discovered that estimation and tool risks weren't analysed thoroughly (with only 1 tool risk and no estimation risks). Therefore, for the next change we further analysed these areas to ensure all risks were properly identified.
- **Change: Risk Assessment + Mitigation: Analyse Estimation and Tool risks**
  We added the following risks to reflect the tool and estimation risks identified. This ensured that the tool and estimation risks were thoroughly considered, and we had mitigation strategies if anything went wrong in this area. The added risks are risks 18-20.
- **Change: Risk Assessment + Mitigation: Add Brownfield Development Risks**
  One major change was adding risks to reflect the fact that we have transitioned from greenfield to brownfield development. As Feathers points out [5], that working with code written by previous teams, especially when that code is untested like in our case, can be very risky.
  Feathers highlights that "The first risk is that we make some gross mistake when we refactor that leads us to think that the system is doing something that it isn't." This is addressed in risk 21, which shows how as a team we will deal with possible misunderstandings of the system. The added risks are risks 21-27

18	Project – Estimation	Time to implement all requirements is underestimated, perhaps due to initially misunderstanding requirements.	This will lead to a rushed end of development, putting undue pressure on the team. This rush will also likely cause a drop in standards, sacrificing the quality of the final product.	4	3	12	Have a list of high-effort low-impact changes to the project. These can be dropped in the case of an under-estimation of how long it will take to do work. Therefore, the development team has time to focus on the important work. Also regularly review work left at each team meeting, to re-evaluate any estimates with gained knowledge. If it seems like we under-estimated the time taken to complete a section of work, that work can be shifted to the "buffer" week of time planned at the end of the project.	Ben	Weekly
19	Project - Estimation	Amount of effort needed to complete different aspects of the project are incorrectly estimated.	This will lead to some team members getting overloaded with work while others having nothing to do.	3	3	9	Agilely respond to new information about team members workloads, by moving work to team members who managed to complete their work quickly.	Ben	Weekly
20	Project - Tool	IDE is slow/inefficient to use, or not suited to the team's workflow.	This would lead to the team's development slowing down due to issues dealing with the IDE. This could lead to all implementation taking significantly longer	4	2	8	Have back-up IDEs (such as Visual Studio Code) that members of the team are already familiar with, that can be switched to if the initial IDE (IntelliJ) is causing problems.	Alex	Weekly
21	Project - Technology	Team misunderstands legacy code (perhaps due to poor documentation from previous team)	This can lead to initial features being developed in a way that could break already implemented systems. Or new systems could be made in a way that doesn't interact well with legacy systems.	4	4	16	Have time planned into initial development to account for mistakes made in taking over code. Then version control systems can be used to roll back ill-advised new feature implementations.	Seyi	Weekly for first 2 weeks of project
22	Project - Requirements	Previous team missed major requirements from assessment 1.	This will lead to extra work implementing these original requirements.	4	2	8	Use an extra "buffer" week allocated at the end of the project to finish these requirements.	Will	Weekly

References

[1] I. Sommerville, “SOFTWARE ENGINEERING Ninth Edition,” 2011 pp. 113. [Online] Available: https://engineering.futureuniversity.com/BOOKS%20FOR%20IT/Software-Engineering-9th-Edition-by-Ian-Sommerville.pdf

[2] “The 2020 Scrum Guide”, scrumguides, 2020, Nov. Available: https://scrumguides.org/scrum-guide.html

[3] I. Sommerville, “SOFTWARE ENGINEERING Ninth Edition,” 2011 pp. 261. [Online] Available: https://engineering.futureuniversity.com/BOOKS%20FOR%20IT/Software-Engineering-9th-Edition-by-Ian-Sommerville.pdf

[4] I. Sommerville, “SOFTWARE ENGINEERING Ninth Edition,” 2011 pp. 647. [Online] Available: https://engineering.futureuniversity.com/BOOKS%20FOR%20IT/Software-Engineering-9th-Edition-by-Ian-Sommerville.pdf

[5] M. Feathers, Working Effectively with Legacy Code. Prentice Hall Professional, 2004 pp. 18.