Change Report

Megan Miles
Alistair Foggin
Kajol Dodia
James Wild
Isselmou Boye
Matt Fitzpatrick

Part A

From the other team we inherited the deliverables, documentation and code. To make Assessment 2 successful we had to use processes, tools and conventions to plan, make, keep track of, and review changes from the other team.

Processes

- Understanding: Before making any changes to the documentation provided by the other team, we reviewed the documentation and code to understand the project's architecture, risks, and any other issues.
- Communication: Talked with the other team to understand any issues encountered. So we talked to them in person to explain functions that didn't have documentation.
- Introduced planning software: The other team used 'Monday.com' to track and plan so we went through the process of breaking down the documentation and code into subtasks.
- Testing all parts: Test changes thoroughly to ensure they do not introduce new issues or break existing functionality.
- Maintain documentation: Update and keep track project documentation to reflect any changes made to the code or new project requirements by listing the new features.

Tools

- Documentation planning: We used the project planning system 'Monday.com' to plan and make the different deliverables.
- Testing code tools: Github project plan to test existing code and ensure that changes do not introduce new issues or bugs. Breaking down the original code and categorising the parts to test by size.
- Documentation creation tools: Created a new shared Google drive to collaborate, plan and review changes to the Assessment 1 deliverables, documentation and code.
- Automated testing tools: We have an automated unit test so that whenever a pull request or commit is pushed into the main branch and before it merges all unit tests need to pass. This ensures that changes do not introduce new issues or bugs.
- Communication tools: We used discord to communicate within the team and email to contact the previous team to help make the new code.

Conventions

- Testing conventions: It is convention to test all parts of the code to ensure that the code inherited is bug free.

- Communication: It is important to establish new requirements from the client to make sure that the code is working towards the same goals.
- Documentation conventions: Establish conventions for documenting the project, including document structure, formatting, and style. Consistent documentation conventions can make the project easier to make, and review changes.
- Commenting conventions: Comment code thoroughly to help other developers understand the purpose and functionality of the code.

Part B

Report Type		
Requirements	Original Report	New Report
Architecture	Original Report	New Report
Method Planning and Selection	Original Report	New Report
Risk Assessment and Mitigation	Original Report	New Report

The changes that were made to each report are detailed below:

i) Requirements Introduction

The requirements were thoroughly completed for assessment 1, and very similar to the requirements that we had for assessment 1. This made it easier for us to take over and identify areas for improvement. The assessment 1 requirements were thoroughly done, meaning there were less changes to make. However, with the additional customer requirements and changes to the game (e.g. adding cooks and endless mode) some edits needed to be made to address these changes. These are stated below.

Additions

Continuing the project required additional requirements to be added, due to new features that the customer wanted implemented. All the requirements that have been added are highlighted in bold in the changed requirement report.

UR_COMPLETION was the only requirement that was added since it was missing earlier in the project. In requirement FR_COMPLETION_TIME it links to the user requirement UR_TIMING, however, this requirement does not appear to exist. To mitigate this, UR_COMPLETION was added rather than UR_TIMING. This decision was made since there are now two modes to think about, and FR_COMPLETION_TIME only indicates the end of scenario mode. Adding, UR_COMPLETION means that there can be functional requirements added to address the completion of Endless mode.

Changes

Since some features are being added others need to be changed to adhere to the new requirements. For example, there are no longer just 2 chefs, there are three, this needs to be addressed in the requirements.

Changed Functional Requirements

ID	Description	User Requirements	Reason for change

FR_STATION_NUMBERS	The user shall have access to 2 cooking and ingredient stations for each cook.(6 in total)	UR_GAME_SIMPLE	Since now there are three cooks, there need to be 6 in total not 4
New ID: FR_SCENARIO_MODE Old ID: FR_COMPLETION	The game shall support scenario mode that has a default of 5 customers and the user has to complete all the orders in the quickest time.	UR_SUPPORT_MULTIPLE_MODES	Since there are more modes in this version of the game, completion means more than one thing.
FR_RECIPES	There shall be 4 recipes: salad, burger, pizza and jacket potato	UR_GAME_PLAY	The new game needs to support more recipes
FR_COOKS	The player shall have access to 3 cooks which the player can switch between and which the player can move	UR_GAME_SIMPLE	The new game requires the additional cook
New ID: FR_SCENARIO_COMPLETION_TIME Old ID: FR_COMPLETION_TIME	The game shall return the time taken to serve all 5 customers in scenario mode	UR_COMPLETION	Since there are more modes in this version of the game, completion means more than one thing.
New ID: FR_TIMING_SCENARIO_MODE Old ID: FR_TIMING	The game shall take approx. 5 minutes, due to high flow of users	UR_EXHIBIT	Since there are more modes being introduced to this game the timing will vary depending on the mode.

Deletions

There were a few requirements that were not implemented in assessment 1 and as a group we needed to decide the importance of these requirements to decipher if they need to be implemented. After a discussion, one requirement was removed. This is stated below, with reasons why:

ID	Description	User Requirements	Reason for change
FR_COUNTER	When a dish is complete, the dish shall be able to be delivered by being placed on the counter	UR_GAME_PLAY	In the current implementation, the chef serves the customer directly rather than using a counter. As a team we decided that the current implementation works for the brief and there is no need for a counter to be implemented. Thus, the requirement can be removed.

ii) Architecture Introduction

Upon taking over team 26's project, we decided to keep the architecture of their project relatively the same, and only make changes to accommodate the new requirements of assessment 2. One of the things that stood out in their project was their use of ECS, which enabled us to alter their architecture with minimal effort; for instance, adding the powerUpSystem allowed us to implement the new requirement of having power-ups without altering any of the already existing systems. Thus, no major changes to their architecture were made, and consequently few changes and additions were made to their deliverable.

To visualise the architecture, we used the same procedure that team 26 had, that is using Intellij IDEA Ultimate to generate PlantUML code and then use diagrams.net to organise the diagrams.

Additions

A few additions were added to the other team's deliverable, and they can be identified in the new diagrams and are later discussed in further detail in the text (specifically in the relevant classes section of the deliverable). As previously mentioned, we added the powerUpSystem to implement power-ups, which in turn helped satisfy the following functional requirements: FR_PREP_TIME_DECREASE, FR_CHOP_TIME_DECREASE, FR_INCREASE_MONEY, FR_INCREASE_CUSTOMER_PAITENCE, and FR_FASTER_COOKS . A game state was created to allow the saving of games. The following were made saveable by making a new class for each entity : cook, customer, CustomerAisystem, food, PowerUpSystem, station, and timer. This allowed us to meet FR_SAVE_STATE and FR_RESUME_PLAY. Moreover, a difficulty class was made to implement the requirement of having game difficulty modes; thus, meeting the requirements FR_EASY, FR_MEDIUM, FR_HARD. Lastly, two game screens were created, one for the scenario mode and the other to implement the endless mode, they both extend a new class that replaced the game screen from the previous architecture. This aided in meeting the following new functional requirements: FR_ENDLESS_MODE, FR_CHANGE_DEFAULT_SCENARIO. All of these changes can be seen in the modified version of team 26's architecture diagrams- at the beginning of the document.

Additionally, we made a state diagram and sequence diagram, since the team didn't make any in the last assessment. The state diagram shows a user choosing between the scenario mode and the endless mode. On the other hand, the sequence diagram show the process of making a burger in the game. They were also added to the document.

Changes

Apart from changing the diagrams in the document and fixing references for requirements' consistency, no other changes have been made to the other team's document. Diagrams were changed to either show additions to the architecture, or reveal new fields and methods added to existing classes. For instance, many of the components in the ECS were changed to help implement the power-ups, and also new methods and fields were added to systems to implement further requirements. New fields were added to components such as the station component in order to be able to modify chopping and preparation speeds of food. This can be seen in the diagrams in the changed deliverable.

Deletion

All the architecture for assessment 1, is consistent with the assessment 1 code and therefore we decided that there was no need to delete any of this report. It is important to keep a record of the edits made to the architecture and keeping previous diagrams. On account of other developers who take over the code have a clear understanding of the history of the code.

iii) Method Planning and Selection Introduction

There are a few similarities between our own Method Planning and Selection and Dev Charles'. Specifically, in the team organisation and the tools used. For example, in our team organisation for assessment 1, we both split into two groups, one group completing the implementation and the other group continuing with the deliverables. Despite this, the assessment will be slightly different, due to the additional requirements and the completion of software testing which was not included in the first assessment, meaning there might be a need for a change. Further, there are some tools and methods we did not implement and will ultimately decide whether we want to use them continuing this project.

Additions

Software Engineering Methods

We added the use of Google Drive to the report, since this is a really useful tool to keep a track of all the resources needed for the assessment. Documents like the Product Brief, original assessment 1 reports from the other team, our own reports from assessment 1 and so on. This is something that as a team we used in the first assessment and found it useful so we decided to continue using it.

Team Organisation

We added further roles into the organisation of the team. In the last project, we had assigned roles (Meeting Chair, Secretary, Librarian and Report Editor) to members in the team. This allowed us to have more structure and guidance in the last project, since all members knew what they were doing. We decided to continue with the roles that were assigned in the last assessment in this assessment for the same reasons.

Systematic Plan

We added the systematic plan that was followed for assessment 2 underneath the assessment 1 plan. The inclusion of this allows others looking at the report to understand the intricacies of producing assessment 2 concisely.

Changes

Software Engineering Methods

Although the previous team ran into some issues with Monday.com we did not have the same problem. Therefore we continued to use it as a tool for our own project. This was not a tool that we had heard of in the previous assessment but we decided it is an invaluable tool and allows us to create gantt charts in a more user friendly manner. Further, it has useful features like:

- Allowing all users involved to see the gantt chart online
- Assigning tasks to an person that has an account
- Assigning a status to the task so the whole team can see when progress is being made
- Assigning dependencies for tasks

They are just some of the many reasons we decided to continue to implement Monday.com.

Moving forward for assessment two we decided that it would be best to use only one IDE, such that everyone is familiar with the same one. Therefore, if someone is having issues with their own code or machine, others in the group will be able to use their own experience with the IDE to help out. The IDE that the group decided to continue with is IntelliJ. The reason being that this was the IDE used for assessment and therefore there is a familiarity with the software.

Additionally, we would no longer want to continue following the waterfall methodology. This is because completing one task after another is not the most effective way to manage a project. This is because some things can be completed simultaneously, for example, some of the change reports can happen at the same time as the software testing for assessment 1. Further, using waterfall would break our continuous integration methods, new function, new tests. This could not be implemented in waterfall as implementation and testing would be completed in two different sections. As a result the group has decided to use a more agile approach.

Systematic Plan

No changes were made to this section of the report since our team had no involvement in assessment 1.

Team Organisation

No changes were made to this section of the report since our team had no involvement in assessment 1.

Formatting

The formatting of the document felt a bit random in the report; there were bullet points for reasoning of some tools but not others, the spacing of the paragraphs were a bit inconsistent, and the font size and margins are inconsistent with other reports. This was tidied up for cohesion. Further, to fit with the report as a whole, some sentence structures and paragraphs were edited. These edits were not recorded (e.g. not in bold) since the content is still the same.

Deletion

There were no deletions for this report since the report details the other teams method planning and selection. We felt that everything was relevant for helping others understand the organisation for the project.

iv) Risk Assessment and Mitigation Introduction

Fortunately, the inherited Risk Register was very detailed and included very similar risks that we had developed for our own risk register in assessment 1. As a result there were a few edits that needed to be made to the risks and mitigation plans themselves, the edits will come from the formatting of the Risk Register.

Changes

There are no changes that need to be made to the Risk Register, since it is cohesive and appropriately describes the risks and mitigation plans.

Additions

We first evaluated the previous teams risks, we came to the conclusion that most risks described are still valid. However, taking on another group's project can present both opportunities and challenges. As a group we then discussed the new possible risks of taking on another group's project. Then we also discussed the potential mitigations to minimise the risks.

ID	Description	Likelihood	Severity	Mitigation	Owner
New_Risk_1	A file is accidentally deleted or corrupted	low	high	Testing the previous groups code first before adding new features to reduce the field of error	Alistair

ID	Description	Likelihood	Severity	Mitigation	Owner
	when the repository is forked.				
New_Risk_2	New features creating a clash with the current architecture mean we would need to make major changes	high	high	Understand the current architecture and manipulate the feature to fit the architecture	James
New_Risk_3	Lack of understandin g	High	low	Communicate with previous team to discuss any ambiguity	Kajol
New_Risk_4	Issues with taking on new code that was missed when in the testing stage	High	moderat e	Test all parts of code before adding new features	Matt

Another addition which is vital is the new ownership of the risks. In the original report the owners are all from team 26, however continuing this assessment they will no longer be involved in the project. As a result, new owners need to be assigned.

Although the Risk Register is detailed and covers a wide range of risks, there is one thing ultimately missing from the table which allows for traceability. That is a status, which describes if the risk has ever occurred or not, or if the risk is still a valid risk. This change is reflected in the new Risk Assessment and Mitigation report, unfortunately, we do not have any information regarding the status of the risks during assessment 1. Thus the status only reflects the status during assessment 2.

Deletions

There were no deletions in this report since all the risks were relevant and it is important to keep continuity