Part A

Software Engineering Methods

Our chosen Software Engineering method was Agile, where we used the Scrum framework. The team was divided into two smaller teams, namely, Implementation and Documentation Team. We then came up with a project “road-map” early in the development process as illustrated below. As we deicded to the Scrum framework, our road-map shows a sprint of 2-weeks for each task followed by a review stage where we would discuss our progress and work on solutions to any problems the team is having. Once we had an idea of the main blocks of work that needed to be done, the team discussed all the smaller tasks that needed to be completed in order to complete a main block of work and we then used GitHub Issues where each member from the two teams were assigned tasks that need to be done. Having all the requirements for the final game from the customer, our roadmap helped us efficiently plan the execution and completion of this project, with the aim of delivering a high-quality game and all its supporting documents on time.

We organised frequent team meetings, on average twice a week. These meetings served two key purposes:

- Providing a platform to discuss and agree upon key development decisions, such as game design and documentation.
- Allowing each team member to communicate their progress on their assigned tasks, and for new tasks to be assigned when necessary.

As discussed earlier, we broke down the tasks into sub-tasks according to the marks and assigned each member ~15 marks worth of tasks based on their skillset and our experience from the first assessment, so that every member had an equal amount of contribution towards the project. The frequent team meetings alongside the bi-weekly review meetings, in which we discussed task progression, allowed us to keep each other accountable for our productivity and ensured that we stay on track.

Tools Used

Communication and Collaboration

- For our team meetings we either organised a discord call or met in-person. Using discord was an intuitive choice as it is a platform that we all have lots of experience with and therefore avoided an unnecessary learning curve.
- We created a Discord server for general communications throughout the project. This was crucial in allowing team members to ask clarifying questions without having to wait for the next meeting, avoiding any unnecessary obstructions to task progression. Moreover, we created a "To-do list" channel in our server where after every meeting notes were posted regarding what was discussed in the meeting and what needed to be done.
- We used GitHub Issues in order to organise and keep track of our tasks throughout the project. Alongside the team meetings, this was crucial in keeping us organised by providing a numerical representation of tasks that were completed (closed issues) and tasks that were in progress or yet to begin (open issues).

Website

- We used GitHub pages to develop our website considering the teams previous experience with using the tool.

Architecture

- We used PlantUML in order to create the abstract and concrete architecture diagrams as the team had prior experience in using this tool.

Implementation

- We chose to use IntelliJ as our IDE. This was due to both the ease of use offered by the tool, along with the fact that several team members had previous experience with the tool: it felt like the ideal choice to avoid unnecessary and time-consuming learning curves.
- We utilised the libGDX game development framework during the implementation of our game. Similarly, to our choice of IDE, this was largely influenced by the previous experience of the team. We were also aware that LibGDX was a popular choice amongst other teams and therefore our use of this framework may make it easier for another team to take over and expand our code.

Alternatives considered

- We considered using other IDEs such as Eclipse for software implementation, however as mentioned previously we chose to use IntelliJ due to team members having previous experience with this IDE.
- We also considered game engines other than libGDX in order to enable software development:
  - We considered using Unity, however we were discouraged from this choice by factors such as Unity's reputation for largely outdated/incomplete documentation and the fact that many useful features are behind a paywall.
  - We also considered using the Unreal game engine, but quickly decided against this as it seemed inappropriate for developing what is a relatively small game and would cause the resulting game to be unnecessarily bloated.

Part B

Team Roles

During our initial team meeting we discussed assigning the following team roles.

- Meeting Chair: Ensuring organised and efficient team meetings that covered all necessary updates and decisions
- Secretary: Recording team decisions and keeping notes of the content discussed in each team meeting
- Librarian: Keeping track of documents and other resources, particularly ensuring that in-progress documents were regularly uploaded to the team shared google drive
- Report Editor: Overseeing document production, in particular ensuring that documentation progress was largely in line with the initial working plan

During this discussion we decided to combine all the roles as they seemed like largely interdependent tasks, hence forming the main role of Team Leader. We then added two addition roles, Implementation leader and Documentation leader, and discussed who would be most appropriate for each role and agreed on the following assignments:

- Team Leader – Katie
- Implementation Leader – Jacob
- Documentation Leader – Saud

Assigning these roles enabled a smooth and efficient team working process and helped to keep track of team progress.

Task Assignments

Task assignment took place once we had gathered all the requirements for the game as mentioned above. We went through all the deliverables required in Assessment 2 and discussed each major task that had to be done. We then calculated how much each task was worth and assigned each member ~15 marks worth of tasks using GitHub Issues based on their skill set and the groups experience from Assessment 1, in order to ensure efficiency and timely completion of tasks. We also made sure that each member's work was reviewed by at least one other team member so that appropriate feedback could be given before the approval of a merge request and the member working on it could make any necessary changes. We then added sub-tasks to these major tasks and such sub-tasks were added till our last review session in order to ensure that each part of the deliverable was completed, and nothing was forgotten.

Part C

Intro

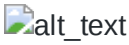
Our first step in planning this assessment was to create a task breakdown table, as shown below (note that time estimates were rounded up to the nearest week e.g., 1.a. which we believed would only need one team meeting are 1 week)

Main Task No.	**Main Task**	**Subtask letter.**	**Subtask**	**Dependencies**	**Estimated time (weeks)**	**Estimated Start and End Time (in Term Weeks)**
1	Team Forming	a	Team introductions and familiarisation	-	1	Aut/3
		b	Assigning team roles	-	1	Aut/3
2	Identifying risks and requirements	a	Meeting to discuss initial ideas about requirements and risks, and compile a list of client questions	-	1	Aut/4

		b	Client meeting to discuss requirements and ask formulated questions	2a	1	Aut/4
		c	Team meeting to agree upon necessary requirements and relevant risks	2b	1	Aut/4
		d	Creation of formal requirements representation (req1.b)	2c	1	Aut/5
		e	Creation of formal risk representation (risk1.b)	2c	1	Aut/5
		f	Written explanations of our requirement and risk process (req1.and risk1.a)	2d	1	Aut/5
3	Website	a	Create initial website (not yet populated with necessary documents)	-	3	Aut/6
		b	Add all necessary documents to website	5b, 3a	1	Spr/3
4	Game Design	a	Team meeting to discuss and agree upon game design ideas	2c	1	Aut/6
		b	Creation of	4a	2	Aut/7

			abstract architecture diagrams			
5	Implementation	a	Creation of functional, commented code	4b	5	Spr/2
		b	Explanation of non-implemented features (impl1)	4a	1	Spr/2
		c	Creation of reflective concrete architecture diagrams	5a	1	Spr/3
6	Submission	a	Completing outstanding reflective documentation	5b	1	Spr/3
		b	Compiling documentation into PDFs, and combining this with code and website into a submittable zip file	6a	1	Spr/3
		c	Completing self and peer assessment	-	1	Spr/3

We then used this table in order to create an initial Gantt chart to show a theoretical schedule for when each task would be completed for assessment 1(as shown below;)



At each team meeting our assigned secretary, Jarred, would create meeting notes which he would then upload to our shared google drive. At the end of each week, we then used these notes in order to create a ‘snapshot’ of our team progress (shown on website snapshot page).

These snap-shots would be created by taking a condensed version of the task breakdown table and colour coding the tasks which were due to be done at this point in the project. With the following colour connotations:

- Red – Not started

- Orange – In progress
- Green – Completed