Method Selection and Planning

Question 1

For the programming language, our customer has asked for the project to be written in Java 17, ensuring the code is compatible and easy for the next group to take over. As a year group, we have already learned Java, so the next group is familiar with it, ensuring they can continue development smoothly. This was a requirement as specified in both the customer meeting and the project brief.

We have decided to use Whatsapp for group communication. This works for us as everyone already had accounts, and we were all familiar with the platform. We made a community, with different channels for each subgroup (each person was in multiple subgroups). A structure like this allows for clear communication and focus, without messages getting lost in one larger group chat. It also made it more simple to have subgroup meetings as we could start a Whatsapp call involving only the people required. We also considered using Discord, but not all the team members were familiar with that and we thought they had similar functionality.

For our document editing, we are using Google Docs and Google Drive. We made a shared drive, with different folders for organisational clarity. This is cloud-based, meaning we can access our work on both the lab computers and our personal computers, which is important for independent work and autosaving - no need to backup work to protect against data loss. Additionally, as it is cloud-based, it supports simultaneous editing, meaning we can all contribute at the same time. Google Docs also has a version history feature, which we can use to track edits, keep records of progress, and maintain accountability for contributions. Commenting and suggestion features are also easily accessible, supporting peer review and allowing us to refine our writing as a group. The platform can easily integrate diagrams, such as those made with PlantUML, as shown in this report.

We had considered alternatives, such as Microsoft Word, which is also cloud-based when used with OneDrive. We have found in the past that OneDrive takes a while to sync changes in comparison to Docs, which is not optimal for this variety of group work. LaTeX was another consideration, however everyone was already familiar with Google Docs, and it requires less technical knowledge to use. Finally, we looked at Markdown-based editors, however these did not have built-in commenting, and they appeared to be less intuitive for project documentation than Google Docs.

The final collaboration tool we are using is GitHub. As with Google Docs. This is cloud-based, so we are able to work on it regardless of which device we are using. This version control platform provides a clear history of commits, allowing us to easily track code changes and revert to previous versions. As we are university students, occasional coding errors may occur, and this platform means we can simply restore the last working version or create an issue to track the bug. Additionally, multiple team members are able to work simultaneously using branches, and it provides clear information of merge conflicts. Commits are linked to contributors, so it helps to maintain accountability within the team. We looked at alternatives such as GitLab and BitBucket, but decided on GitHub as it is one of the most commonly used version control systems, so many of us have already used it, and gaining experience in it is advantageous for future projects.

For our game engine, we selected LibGDX because it offered an effective balance between performance, flexibility, and control. As a lightweight, code-based framework rather than a visual editor, it allowed us to directly implement and manage core game systems such as input handling and asset management. The cross-platform capabilities of LibGDX were another major advantage, enabling us to build and test the same codebase across platforms such as Windows, Mac, and Android, allowing us all to contribute and work effectively. We also found the LibGDX API to be well-structured and consistent, making it straightforward to access features such as graphics and audio in a unified way. Its clear design helped streamline development and reduce platform-specific issues. Furthermore, because LibGDX is open source and built in Java, it aligned well with our existing programming knowledge and allowed for easy integration of additional libraries when needed.

We considered using Unity, however our customer requested a 2D game, and Unity is generally better for large-scale 3D projects. Another common Java game engine is jMonkeyEngine, however this is also more focused on 3D development. We also explored LWJGL, which is fast due to its low-level implementation, but this is difficult to develop with and requires a significant initial time investment to learn.

The IDE we decided to use is IntelliJ as it is built specifically for Java development. It utilises powerful tools, such as codebase-wide refactoring, advanced Java syntax highlighting, and robust debugging tools and error detection. It also contains built-in tools for both Gradle, which is the build automation tool used by LibGDX (our game engine), and Git, ensuring a seamless version control experience.

Alternatives considered include VS code and Eclipse. By using IntelliJ, we avoid the need for everyone to install multiple VS code extensions to get the same experience across the implementation team. Additionally, IntelliJ provides an out-of-the-box experience, unlike VS code which would take time to install the correct plugins. With regard to Eclipse, it is powerful but hard to learn, and as most team members are already familiar with IntelliJ, it would save valuable time to use that instead.

To create our map, we decided to use Tiled, a flexible tile-based level editor. We selected Tiled for multiple reasons, the main being its excellent compatibility with LibGDX, its large amount of built-in functions, and comprehensive documentation make it super easy to use. The simplicity and ease of learning were super important due to our project's time restrictions. Using Tiled allowed us to quickly develop a map using our tile sprites without any learning curve.

For our game’s assets, we initially used Sora AI to generate them. While sora produced some good images, inconsistencies across the assets made them look weird and unsuitable to be used. Given our time constraints, we were unable to create our own assets. We finally decided to use a free CC0 asset pack. This proved to be the best solution for a few reasons. The CC0 asset pack license allowed us to use the assets in production which meant no legal restrictions, and the customer had no requirements for the assets visual style. Additionally, we could import the sprites directly into tiled and start developing a map immediately.

Question 2

Key stakeholders for this project include the team members and our customer, Robert Jongeling. We had one meeting with our customer at the beginning of the semester, and have the ability to schedule follow-up meetings with him, or ask him short questions in our practical session every Wednesday.

Our team's approach to team organisation involves a more flat hierarchy, with no outright leader. With small groups (such as ours), this approach works well because it promotes collaboration and open communication, encouraging everyone to contribute. It also allows for faster problem solving, as everyone has the authority to make important decisions. Through valuing every person equally, our team is able to work more cohesively and create a product which reflects the collective strengths of the team. We also assigned a leader to each of the work packages, so we could keep track of progress.

While the flat structure encourages collaboration, our team also recognised the importance of clear organisation and equal participation. Therefore, to maintain accountability and ensure progress is still being made, we decided to make subteams based on the deliverables in the assessment. This allows different members of the team to focus on a smaller number of tasks, resulting in clearer task allocation and deadline monitoring. Decisions affecting the whole project are made collaboratively within the team and each subteam resolves their own internal questions or choices, improving efficiency. The subteams allow us to monitor each member's individual progress, providing support where required. As the project progressed, we changed who was working in which subteam, but the initial groupings can be seen below:

Website Subteam	Kaleb and Max handled the development and maintenance of the project's website.
Requirements Subteam	Kaleb, Alex, Maddie, and Fatima handled the requirements gathering, analysis and documentation.
Architecture/Diagrammer Subteam	Cory, Max, Kaleb, Alex, and Maddie handled the system architecture design and created technical diagrams.
Method Selector/Planner Subteam	Cory, Sarina, and Fatima handled the choice of development methodologies and project planning.
Implementation Subteam	Cory, Max, Alex, Sarina, and Maddie handled the development, coding and testing of the game and subsystems.
Risk Assessor Subteam	Sarina and Fatima handled the identification, analysis and documentation of the project's risks.

To co-ordinate between subteams and ensure clear communication, we follow an Agile Scrum approach during our weekly meetings. During these meetings, we discuss which tasks each member has completed, identify any blockers, and assign roles for the following week. This enables everyone to bring up issues and focus on accountability in a structured way. We also fill in a logbook after each meeting, to keep a record of what we discussed and achieved.

Question 3

ID	Title	Description	Dates	Dep.	Contributors
WP1	Requirement Elicitation	Deciding requirements for the project.			**Leader: Maddie**
T1.1	Create Interview Script	Write questions for the interview and complete a practice.	29/09/2025 - 08/10/2025		Everyone
T1.2	Conduct Interview	Complete customer interview. Record it and write notes.	08/10/2025	T1.1	Everyone
T1.3	Requirements Specification	Create a document listing the requirements.	09/10/2025 - 16/10/2025	T1.2	Kaleb, Alex, Maddie, Cory
D1.1	Requirements Document - Final Version	Lists the requirements for the project. **Visibility:** Customer	16/10/2025	T1.3	
WP2	Planning	Research, plan and schedule the project.			**Leader: Sarina**
T2.1	Research Tools	Research collaboration tools and game engines.	29/09/2025 - 05/10/2025		Everyone
T2.2	Assign Roles	Assign subteams and tasks within those.	06/10/2025 - 12/10/2025		Everyone
T2.3	Create Planning Document	Create a document listing the plan.	13/10/2025 - 02/11/2025	T2.1, T2.2	Sarina, Kaleb, Maddie, Cory
D2.1	Planning Document - Final Version	Lists the plan for the project. **Visibility:** Customer	02/11/2025	T2.3	
WP3	Risk Assessment	Assess and keep records of possible risks.			**Leader: Fatima**
T3.1	Create Risk Document	Create a document listing the initial risks.	29/09/2025 - 14/10/2025		Fatima
T3.2	Add New Risks	Add in new risks as we come across them.	16/10/2025 - 29/10/2025	T3.1	Fatima, Maddie, Sarina
D3.1	Risk Document - Final Version	Lists the possible risks for the project. **Visibility:** Customer	29/10/2025	T3.2	
WP4	Design	Create architecture diagrams and the website.			**Website: Max** **Arch.: Kaleb**
T4.1	Build Website	Create a basic website.	22/09/2025 - 28/09/2025		Kaleb, Max
T4.2	Add Final Links to Website	Add links for documents and diagrams to the website.	08/11/2025 - 10/11/2025	T4.1	Kaleb
T4.3	Initial Class Diagrams	Create initial class diagrams for architecture.	13/10/2025 - 16/10/2025	T1.2	Cory, Kaleb
T4.4	Architecture Changes	Architecture changes based on implementation	17/10/2025 - 26/10/2025	T4.3	Kaleb, Maddie, Alex

T4.5	Create Architecture Document	Create a document listing the architecture, including the class diagrams	27/10/2025 - 07/11/2025	T4.4	Kaleb, Maddie, Cory
D4.1	Website - Final Version	A website for our game, complete with links. **Visibility:** Customer	10/11/2025	T4.2	
D4.2	Architecture Document - Final Version	A document listing the architecture and class diagrams **Visibility:** Customer	07/11/2025	T4.5	
WP5	Implementation	Creating and coding the game			**Leader: Alex**
T5.1	Player Movement	Code focused on player movement	17/10/2025 - 23/10/2025	T4.3	Alex, Max, Kaleb
T5.2	Environment Creation	Creation of the maze, and implementation into the game	17/10/2025 - 23/10/2025	T4.3	Fatima, Kaleb
T5.3	Object and Entity Logic	Code focused on object and entity logic	17/10/2025 - 23/10/2025	T4.3	Alex, Max, Kaleb
T5.4	Player - Environment Interaction	Code focused on player and environment interaction	24/10/2025 - 07/11/2025	T5.1, T5.2, T5.3	Alex, Kaleb
T5.5	Menu Creation	Creating screens, such as menus	17/10/2025 - 07/11/2025		Max
T5.6	Create Implementation Document	Create a document, listing the processes and tools used for implementation	08/11/2025 - 10/11/2025	T5.4, T5.5	Kaleb
D5.1	Game and Code - Final Version	A working game, created to the specification provided. **Visibility:** Customer	07/11/2025	T5.4, T5.5	
D5.2	Implementation Document - Final Version	A document listing the resources and licences used to make the game **Visibility:** Customer	10/11/2025	T5.6	

Above is a table showing the work packages and tasks of the project, along with which members contributed to each task. A work breakdown structure (WBS) can also be found in Appendix C on the website, showing a more visual representation of the table above. Each work package was assigned a different leader, who was responsible for the progress of all the tasks within their package to distribute the leadership, and to maintain accountability within the subteams. For each task, we allocated members based on what needed to be done that week, and how much work everyone was already doing. The most vital tasks were allocated multiple people to avoid a low bus factor, optimally assigning three team members per task. To micro-plan what tasks were being completed each week, we filled in a logbook where we specified the activities and members involved.

Our initial schedule changed over time. A visual representation of this can be found using the Gantt charts in Appendix B on our website. We initially anticipated we would have more time for the implementation section of the project, however, we had dependencies on the interview (T1.2) for the architecture (T4.3), which we needed to complete before starting the implementation. This delayed the start of our implementation, resulting in less time for the written document. Despite this, we managed to assign more people to the implementation tasks, to maintain our final deadline. Both the WBS and Gantt charts were made using PlantUML.