Method selection and planning

4a)

For the project, the team selected the Agile method which focuses on collaboration, adaptation and continuous improvement. More specifically, we selected Scrum where the development is divided into sprints for respective tasks.

Scrum allows regular iterations which is helpful for our project that involves different phases (requirements gathering, implementation, architecture, etc). It also gives us the flexibility to adapt to new challenges that come up during the project. It gives continuous feedback since it breaks the project down into tasks per week, also making it more simpler to manage as a relatively small team. As well as enhancing transparency and equitable workload distribution.

In each meeting, we report what people have worked on and finished, what people are currently working on and we will continue working for the week, and risks or problems that need to be addressed. At the end of each spring, we review what has been completed and what not, what needs priority and what to do differently for the next spring.

The tools we used for our project were:

- Google docs: used for recording meeting minutes and practical sessions. It kept track of important documents of different tasks of the project, and its collaborative environment enabled all team members to contribute to the project on a real time basis.
- Plant UML(Tasks Management): it was used to create the Gantt chart that contains a clear view of the different tasks assigned, their start and finish dates, and each team's member contribution. The visual format made it easier to track projects and see how individual tasks affected other parts of the project and categorise them in priority.
- Github(Version Control): it was chosen for source code management. It is where the code is stored and the tool facilitates collaboration of all team members. It helped ensure that the code was constantly up to date.
- IntelliJ IDE: it was the IDE chosen to develop the code. Its features like code completion, debugging and refactoring tools improved productivity and allowed us to focus on building the game.
- Java 17: we used it to develop the core components of the game.
- LibGDX: it was our chosen game development framework. It provides the necessary tools for building 2D games. It also offers high flexibility for managing graphics, input handling, and game physics, making it suitable to develop our game.
- Discord: it was our chosen platform for communication, it facilitated smooth communication between the team members during the development of the project.

Alternatives considered:

- Notion (Tasks Management): It helped us plan in a collaborative way since everyone in the team could access it and modify the files. There we kept track of priority tasks and deadlines. It gave us a visual representation of our sprints and made tracking easier. However, after consulting it with our practical professor, Notion was not providing an ideal scalable gantt chart. Hence, it was recommended to use PlantUML to give a clearer graphical depiction of each member's details alongside the deliverables and their tasks.

- JMonkey: We considered it as a library for game development but ultimately we chose LibGDX since it has more extensive documentation and is better suited for 2D games.

4b)

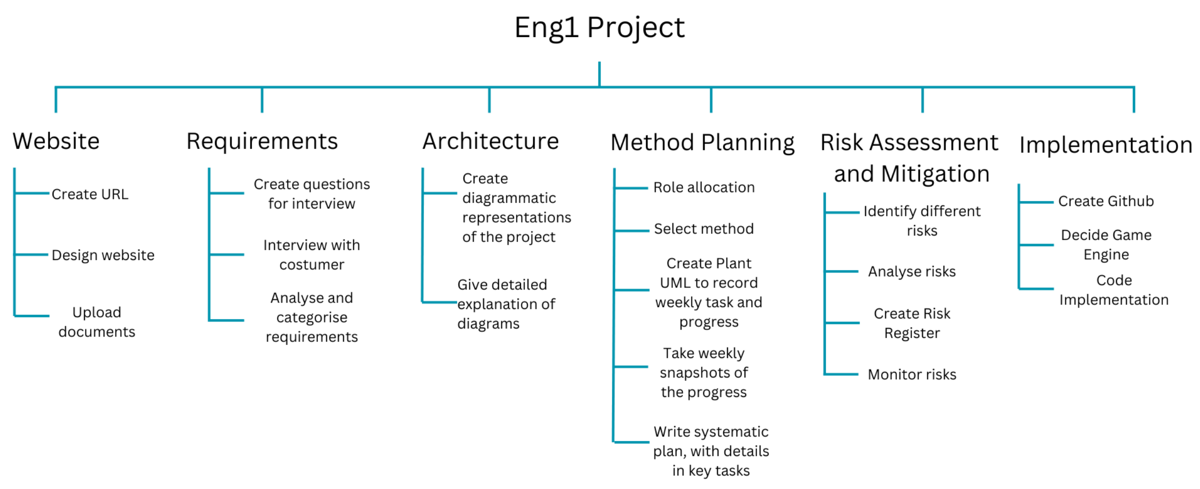
In the first week of the project we discussed the allocation of different roles and responsibilities. To begin with, we used various frameworks such as the York Strengths and Myers Briggs Type Indicator to get a better grasp of our individual strengths and weaknesses. We then had an open discussion where everyone took turns to express their abilities and what they felt most comfortable working on, for example, some felt more inclined to work on the code implementation section and the others on either project management and eliciting requirements. Alongside this process, we took into account the mark contribution for the corresponding sections, hence negotiation for certain roles were conducted to ensure everyone had roughly the same mark contribution average. All members agreed unanimously on their final distinct roles.

Throughout the allocation process we kept in mind to allocate at least two people for each deliverable section to ensure we had a low bus factor to mitigate risks across the project development. Particularly the code implementation, three members were assigned as it was most critical to the overall project development. In addition, within each section, we avoided joint-leadership and assigned the same leader of the deliverable section to oversee its following tasks. This was to maintain constant work style and quality throughout and reduce overlapping and confusion of each member's work process.

We felt that overall, this was an appropriate approach to the team and project as it allowed members to work on deliverables they either set to challenge themselves or which aligned with their personal strengths and interests. In general, allowing our members to choose their dedicated deliverables themselves increased their overall motivation towards the project.

4c)

Work Breakdown:


Eng1 Project
Website
Create URL
Design website
Upload documents
Requirements
Create questions for interview
Interview with costumer
Analyse and categorise requirements
Architecture
Create diagrammatic representations of the project
Give detailed explanation of diagrams
Method Planning
Role allocation
Select method
Create Plant UML to record weekly task and progress
Take weekly snapshots of the progress
Write systematic plan, with details in key tasks
Risk Assessment and Mitigation
Identify different risks
Analyse risks
Create Risk Register
Monitor risks
Implementation
Create Github
Decide Game Engine
Code Implementation

<u>Phase 1: Initial Setup (Friday 27th September):</u>
Tasks:
- Team setup: assign roles and responsibilities
- Create shared drive, choose communication platforms, create github.
- Select method planning: Agile, Scrum

<u>Phase 2: Website Setup and Requirements Gathering (September 30 - October 14):</u>
Tasks:
- Set up the website to store key documents and updates.
- Brainstorm requirements and conduct stakeholder interviews.
- Analyse and categorise gathered requirements.
- Document the requirements and prepare reference material.

Dependencies:
- Website and shared documents must be set up before documenting requirements.

Priority: High

<u>Phase 3: Method Planning and Risk Assessment (October 3 - October 17):</u>
- Record each sprint and task updates weekly on PlantUML
- Write the risk management plan and mitigate identified risks

Dependencies:
- Competition of requirements gathering

Priority: Medium to High

<u>Phase 4: Architecture Design and Diagram Representation (October 10 - October 24):</u>
Tasks:
- Begin diagrammatic representation of game architecture
- Finalise and explain detailed diagrams

Dependencies:
- Analysis of requirements informs the design process

Priority: High

<u>Phase 5: Core Game Implementation and Testing (October 18 - November 7):</u>

Tasks:
- Decide the game engine (libGDX) and begin coding features like the grid and camera
- Begin implementation of UI elements (eg., menu, clock, progress bar) and core gameplay features.

Dependencies:
- Architecture must be in place before core features are developed

Priority: High

<u>Phase 6: Risk Review and Final Submission (November 8 - November 10):</u>
- Review and finalise risk management, and update the register
- Package the code and submit final documents and game assets

Dependencies:
- All core features and implementation tasks must be completed

Priority: High