Method selection and Planning

Group Number: 4
Group Name: THEEMD
Members: Mikaella Loppnow, Tom Daly, Harriet Kirby, Ethan Buss, Dillon Pandya, Ereife Odusi

Methods

We have chosen to follow an agile methodology after discussing the options and deciding what values we wanted to put at the heart of our team. We agreed with the key principles prioritising individuals and responding to change throughout the project as opposed to making a strict plan and adhering to it, as stated in the Agile Manifesto [1].

Development Tools

We considered using different IDEs and a number of different game engines - the most notable alternatives to LibGDX considered were LITIENGINE and jMonkeyEngine.

Ultimately, we decided on using IntelliJ IDEA for our IDE and LibGDX for our game engine.

We chose to use LibGDX due to its reputable and well maintained status; boasting frequent updates and a large community of users [2], making it a lower risk to use than several of the alternatives we considered. The game engine's mention in lectures as well as the amount of other development teams known to be using it also held influence over our decision to utilise it. LibGDX is also lightweight and efficient, both of which are things we aim to achieve with our final product, as well as being described as easy to learn - especially with the number of tutorials for members to follow.

We also wanted to maximise the chances of potential further development of our project by other teams (this decision was reached after a number of team members expressed concern about using engines less well known by the general cohort); and also to take some pressure of our implementation team, as it prevents them from having to spend time learning multiple game engines after we pick up another project.

Alternatives Considered

Engines:

The main reasoning behind our interest in LITIENGINE was its marketing as being "beginner friendly", open-source (specifically under an MIT licence) and cross platform [3]. However, the main issue we found with LITIENGINE was that it was significantly less widely used than the alternatives and had less documentation, which could greatly hinder our development should issues arise that cannot be easily resolved.

We also considered jMonkeyEngine as it is free and open source as well as providing its own IDE with thorough documentation. However, on looking into using it, we found that it had a much steeper learning curve, which was something we wanted to avoid since our team was working with a tight deadline and spending too much time learning the engine could take away from progress in other areas. In addition, we knew most teams were using LibGDX and thus we ultimately decided against jMonkeyEngine.

IDEs:

We chose to use IntelliJ IDEA as our IDE due to it being visually intuitive to use as well as due to its excellent compatibility with both LibGDX and Git, which would allow for a more streamlined and convenient approach to version control throughout the implementation process. Having looked at other game development operations, we saw that many of them

utilised and recommended IntelliJ IDEA, as well as a large number of people within our cohort.

We did also consider using Eclipse and VSCode, however they both had their shortcomings that ultimately led to our decisions against them. We found Eclipse less intuitive to use than either of the others and members of the team who had and had not used it before felt it had a steep learning curve when compared to IntelliJ. VS Code lacks build support and requires a number of extensions to function properly with LibGDX. Ultimately, IntelliJ IDEA seemed to be the best decision and thus we chose to move forward with it.

- **Collaboration Tools**
    - **WhatsApp** - simple messaging service to provide first point of contact outside of University arranged practicals, enables easy communication allowing for flexibility.
    - **Discord** - enables members of the group to communicate and screen share progress and speak in a voice channel without having to physically attend an in-person meeting.
    - **GitHub** - essential for efficient version control between the different members of the programming team.
    - **Monday.com** - used for scheduling and to mark clear progress throughout our project, ensuring deadlines are met.
    - **Google Drive** - used to share documents throughout the development process and enable collaborative development of the non-code deliverables.

Team Organisation

Our team made a collaborative decision early in the development process that we would avoid officially assigning specific roles to each team member; instead we chose to favour a more flexible approach wherein members all contributed equally. This does not mean that we did not favour unofficial positions in the team, which were filled naturally as the development proceeded.

For example, some members were more confident in speaking - particularly in the first few meetings - and so naturally rose to unofficial leadership roles within the team. Other members were more creative and had confidence in those skills, allowing them to participate in the team's branding (i.e., our name and logo): this gave us a clear identity as THEEMD from the early stages of our development. Other members were less outspoken but more than happy to pick up work on various aspects of the project, leading to an overall comfortable environment.

We also prioritised ice-breaking activities in our first few meetings, which allowed less outspoken members to feel comfortable voicing their opinions, enabling our team's key ethic of equality between members to thrive. The team thus did not consist of a hierarchy, but instead consisted of a "round-table" approach.

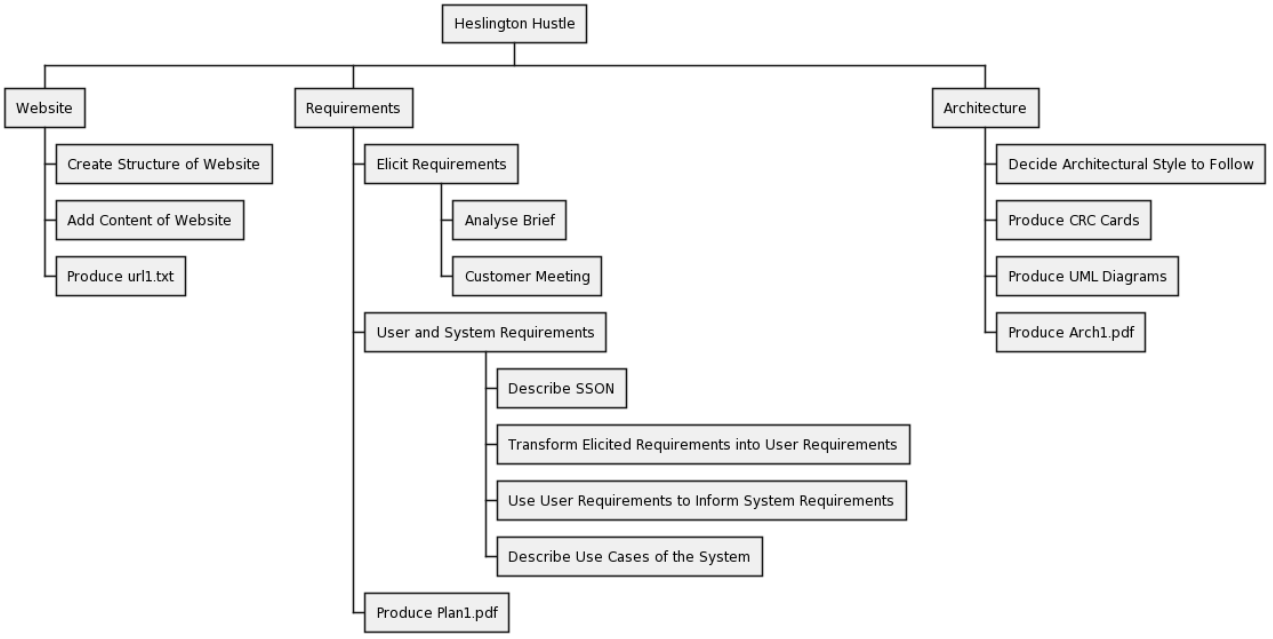
We also decided on a set time and location for weekly meetings outside of the practical session, which all group members made an effort to attend - these opportunities to check in on progress in person as well as the lower pressure environment outside of an allocated teaching environment lead to our team becoming comfortable with one another faster and also enabled issues / developments to be discussed in person. We chose to value in person communication where possible, as it is often more comfortable and less formal than communicating via either WhatsApp or Discord.

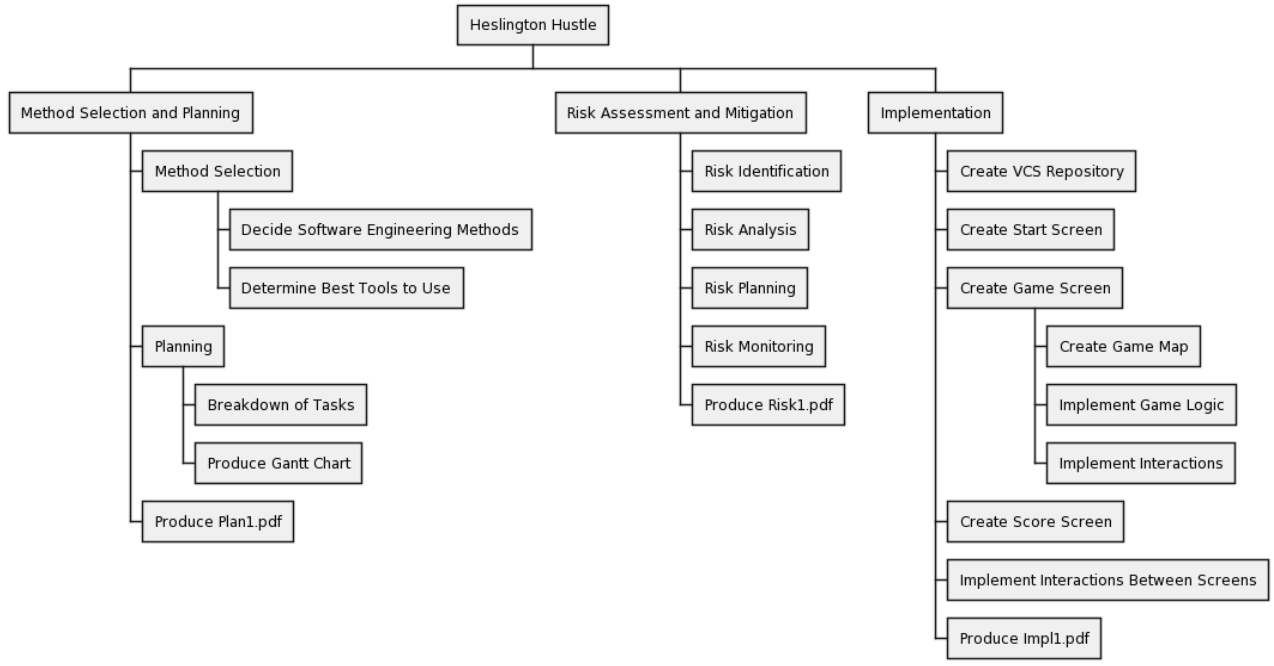
Overall, this was massively beneficial to our team's motivation and created a comfortable atmosphere to work in. We ensured that all team members felt that their opinions could be voiced and would be valued just as much as anyone else. Our team ended up really feeling like a "team" instead of just a group of students set to work together to achieve a common goal.

This overall approach enabled all group members to focus on areas most befitting their strengths and thus each member was able to utilise their full potential, as they felt comfortable to express an interest in what best appealed to them. This allowed people to develop aspects of the project that they genuinely cared about, leading to an overall higher quality of work as well as motivation amongst our team. As we made an effort to value each other's strengths and weaknesses, and to work together when things didn't go as planned instead of attempting to blame each other, the team became a significantly more productive environment and far less stressful than it could have been.

Workflow Plan

Our team first determined the key tasks that would need to be completed for each section of the project. We decided to illustrate these using a Work Breakdown diagram, which we produced using PlantUML. This diagram is shown below, split into two figures for readability:

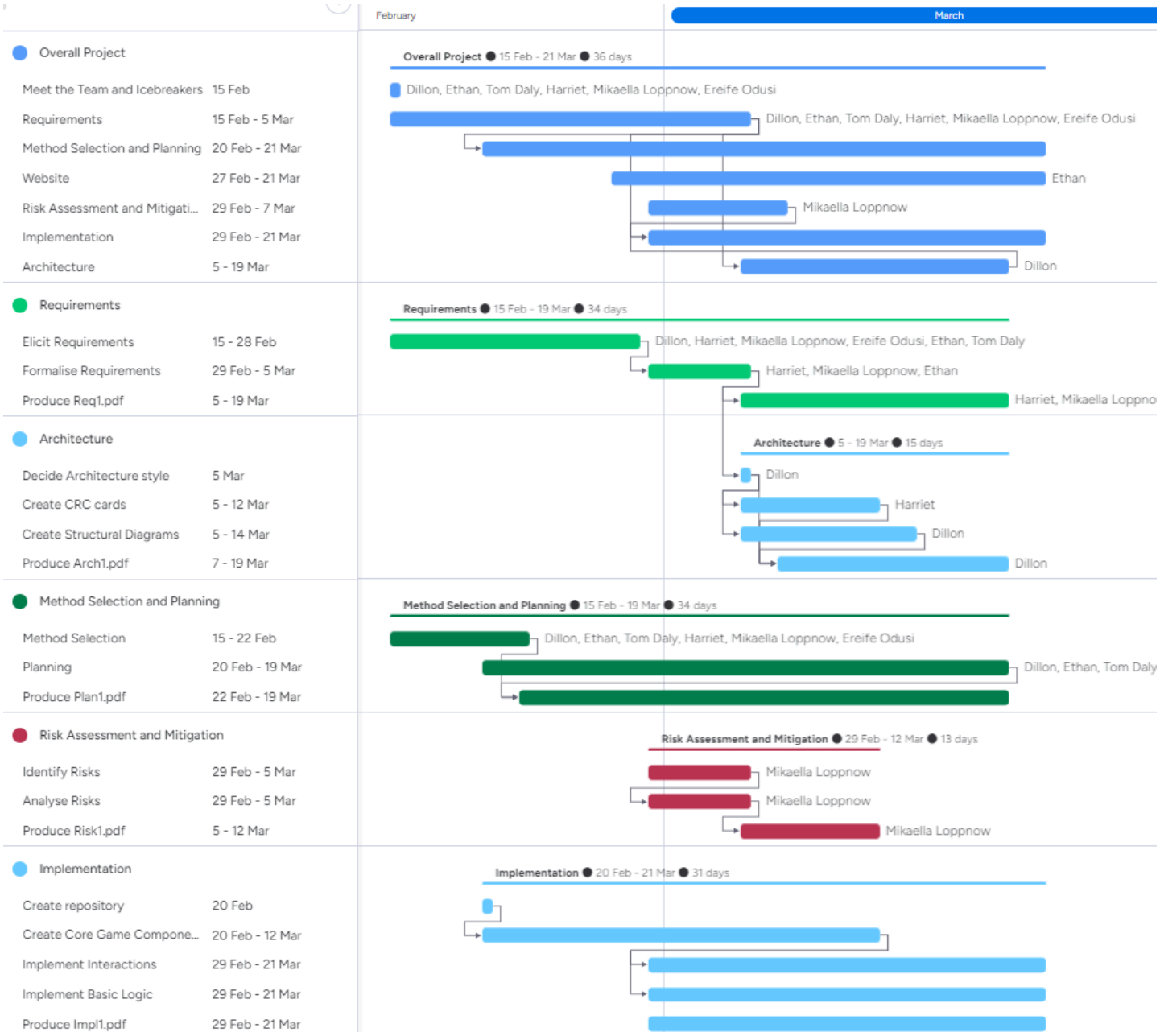




We then used the Work Breakdown diagram to inform the production of a Gantt chart for the timeframe of our project. We took into account the expected lengths of each task and any dependencies of a task on others. We produced this Gantt chart using Monday.com, as it offers many features that make it suitable for the purposes of our project, e.g. the ability to include sub-tasks and sections, as well as the ability to assign tasks to specific team members.

The Gantt chart is shown below, which shows the tasks as well as their timeframes and dependencies (illustrated by arrows from one task to another). We prioritised tasks on which

others were dependent, since it was important that we started on each task as early as we could in case of delays. For example, we critically prioritised ‘Elicit Requirements’ and ‘Formalise Requirements’, as ‘Decide Architectural Style’ was dependent on them.



The breakdown of tasks did not evolve much throughout the project as we considered the majority of tasks that needed to be completed from the start. However, we adapted the timeline of the project significantly over the course of the project as a result of tasks taking longer or shorter to complete than the expected time, which in turn also affected dependent tasks. This can be seen in the ‘Create Core Game Components’ task in the Implementation section of the Gantt chart. Initially, the finish date of the task was sooner but we extended it as we were still completing it as we started ‘Implement Interactions’ and ‘Implement Basic Logic’.

Bibliography

[1]
“Manifesto For Agile Software Development,” *Agile Manifesto*, 2001. [Online]. Available: https://agilemanifesto.org/.
[2]
LibGDX, “LibGDX,” *LibGDX*, 2009. [Online]. Available: https://libgdx.com/.
[3]
Litiengine, “LITIENGINE 🎮 Free And Open Source Java 2D Game Engine,” *LITIENGINE 🎮*. [Online]. Available: https://litiengine.com/.