Method selection and Planning

Group 9

Group name: KOWLAAB

Aiden Sayer
Lurvïsh Polodoo
Ben Salkield
Oliver Rogers
Alex Sharman
William Roebuck
Kiyan Eiles

Outline and justification of software engineering methods:

After having a meeting where we discussed the possible methods we could use in the development of the project. We finalised on using the Agile methodology as it is best suited for our specific project requirements. Agile development consists of the four fundamental values:

- individuals and interactions over processes and tools,
- working software over comprehensive documentation.
- Customer collaboration over contract negotiation.
- Responding to change by following a plan

This aligns with our objective of developing a solution that meets the requirements in a short period of time as it concentrates on developing software quickly. This applies to this project as the time constraint adds extra pressure to get the product to meet the customer requirements. This style also concentrates on collaborating with customers frequently and being able to adapt to changes easily. As we have been given a sizable freedom in the development of features for the product, a Agile methodology will allow us to develop code more effectively and efficiently as we can adapt to any restraints that may appear during development.

During the meeting we alternatively considered the Waterfall model. This model is based on the idea of taking a sequential, ordered plan to develop a product in stages. However this method lacks flexibility which is a key factor in this project's development due to the time constraints and freedom of the requirements. We decided that the Waterfall model wasn't the best option, because we knew we would make multiple changes to the plan and we didn't have time to restart the project due to the deadline.

Development and Communication tools:

To assist with communication, we chose to use Discord for file transfer and more formal planning. For informal, quick exchanges we rely on WhatsApp, as it is easily accessible to all team members and convenient for quick updates and reminders. This allowed us to maintain continuous communication which is important for an agile team.

When choosing the IDE, we considered VScode as everyone was already familiar with it but one of the group members suggested this isn't the best option as it is not used for game development in java. The alternative chosen was IntelliJ due to its vast plugin ecosystem, excellent refactoring tools as well as the version control integration as we are using Git through GitHub. GitHub was the obvious choice for version control as most of the group has had experience with it in the past and it is easy to learn for people that haven't.

Another tool we are using for streamlining issues is Linear. It enables us to assign tasks to different people and track progress throughout the project. We can also change the priority and status of separate tasks making it easier to concentrate on key stages of our project. This tool is incredibly useful as it allows us to ensure that the project is organised throughout the development process where tasks change and we have to adapt based on the situation.

To help with progress tracking, issues that are created on Linear are automatically transferred to Github. This functionality of Linear linking to GitHub pull requests allows us as a team to better automate workflow and ensure that all issues are synced over Linear and Github. Using this method and software for organisation aligns with our choice to go with the agile method of approach to this project. There were many alternatives tools and software that we could have used for this project. However after discussing it as a group we landed on Linear being the best option for our specific needs for the project.

Approach to team organisation:

During our preparation stage we created a plan and a list of functional and non-functional requirements needed in the final product. This allowed us to outline the key stages of our project and estimate the time required for each stage while ensuring that the key features that are required from the data gathered from the product brief and customer interview.

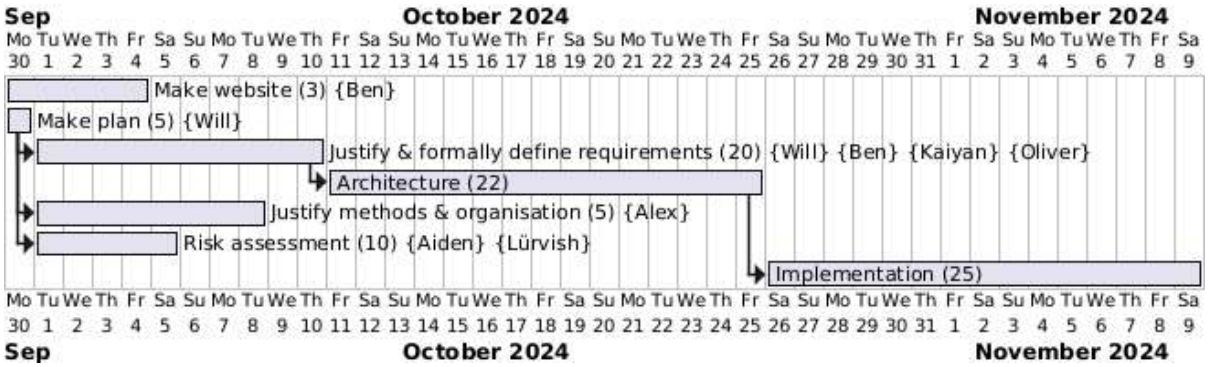
Our design process follows an agile software engineering inspired sprint model, which divides the project into manageable weekly segments. We hold sprint planning meetings every Thursday, where we assign tasks based on priority and each team member's strengths. During these meetings, we collaboratively decide which functionalities to tackle in the coming week, breaking down the project into smaller, more manageable stages. This planning structure allows us to make strategic choices, ensuring that important tasks are completed on time and without overwhelming the team.

Our approach based on clear planning, regular progress tracking, and communication has proven effective in maintaining momentum and creating a supportive team environment. The sprint structure keeps us agile, ensuring that we can meet project milestones outlined by our plan while adjusting to challenges or new ideas as they arise. This organisational style is well-suited for both our team and the project, promoting productivity and adaptability, essential qualities for successful software development.

Systematic plan for the project:

Initially, we set out a rough plan using a PlantUML diagram. We based the timescale on the number of marks, and discussed which parts we'd all like to do. As a group we decided we should all work on the two largest parts, architecture and implementation. To ensure an even distribution of marks, we made sure that all members had an equal number of marks before the start of architecture.

Figure 11 on https://eng1-cohort2-group9.github.io/Website/



During our next meeting, we realised that many of our tasks were dependent on the stakeholder meeting and adjusted the plan below.

Figure 12 on https://eng1-cohort2-group9.github.io/Website/

For each meeting during the architecture section we went through the steps of RDD and discussed the overall system architecture. This process is detailed inside the architecture document.

During development, we made use of linear.app to break down large tasks and track priorities and dependencies. Once the libGDX project was completed and all of our tools were set up, the highest priority task was to add files and empty methods for all classes planned in the architecture document, as all other tasks depended on this. After this was complete, we set about adding the implementation details. Certain larger problems, like the UI design, were created as an issue and given multiple sub-issues which could be assigned to different people.

A selection of screenshots throughout development can be seen on the website. (https://eng1-cohort2-group9.github.io/Website/).