

Method Selection and Planning 1

Cohort 1, Group 6

Group Members:

Hussain Alhabib

Ellen Matthews

Minnie Poon

Jason Ruan

Daniel Smith

Owen Smith

Software Engineering Methodologies

For our project, we selected the Agile methodology to support an iterative, flexible, and collaborative development process. Below, we outline our reasoning and the alternatives considered.

Agile

Agile's adaptability and iterative nature make it ideal for UniSim. It allows for incremental development with continuous client feedback, which is crucial as requirements evolve based on testing. Agile promotes strong team collaboration and ensures our alignment with deliverables. Regular review will help us adjust priorities quickly, keeping the project on track and ensuring the clients needs are met.

Challenges

Its frequent change requests could overwhelm the team, particularly with our small team and tight timeline. Therefore, we must manage the scope carefully to ensure steady progress.

Waterfall

We also considered Waterfall - a traditional methodology with a structured, sequential process (design, development, testing). It works well for projects with fixed requirements. However, Waterfall's rigidity and lack of flexibility make it unsuitable for UniSim's evolving features, requiring frequent iterations. Also, testing only occurs after development which risks bugs becoming deeply embedded. Therefore, with a limited timeline and a small team, Waterfall isn't suitable for our needs.

Collaboration & Documentation Tools

To support our chosen methodology, we use Slack for communication and Google Drive for file management. Below is our reasoning and the alternative tools we considered.

Slack

Slack is our primary communication tool. It enables real-time messaging, file sharing, and integrates with relevant tools like Google Drive and GitHub. Slack's channel-based structure helps organise discussions by task or deliverable which is ideal for Agile workflows. Its ease of use supports the fast-paced and collaborative nature of our project.

Challenges

Slack's free plan limits app integrations and message history, but these restrictions are manageable given our team's size and the project's short timeline.

Google Drive

Google Drive serves as our central file management platform. It allows real-time collaboration and integrates with other Google Workspace tools, enabling smooth document sharing and version control.

Challenges

The team's Google Drive can become cluttered due to the many contributors, but this is manageable through proper file management (administered via our Secretary).

Microsoft Workspace

We considered the Microsoft Suite (One Drive, Teams, and Microsoft 365), but the free plan limits features and our team is more familiar with Google Drive and Slack. Also, Slack offers more professional communication features compared to Microsoft Teams.

Development Tools

We selected IntelliJ IDEA as our IDE, LibGDX for game development, and GitHub for code management. Below is why we chose these tools and the alternatives we considered.

	Chosen Tool	Challenges	Alternatives Considered
IDEs	For Java 17, we chose IntelliJ IDEA for its robust support for Java development, including intelligent code completion, debugging, and GitHub integration.	IntelliJ is feature-rich, which can be resource-heavy on lower-end systems, but this is unlikely to be a major issue for our project.	We considered Visual Studio Code , but it lacks the dedicated Java tools of IntelliJ, making it less suitable for our needs.
Game Development Framework	We are using LibGDX for UniSim. It is a flexible, well-documented framework ideal for building 2D games, supporting graphics, input handling, and audio - critical features for our project.	LibGDX may be difficult for beginners, and some advanced features can take longer to implement.	We considered Unity , but its focus on 3D development and use of C# made it less suited for our 2D-focused game. Additionally, our team is more experienced with LibGDX, which also offers cross-platform capabilities.
Version Control	We use GitHub for version control, providing a centralised repository for code management. Its branching and pull requests enable parallel development, preventing conflicts. The version history feature tracks changes and supports code reviews.	GitHub's free plan limits private repositories, and while its interface is user-friendly, some advanced features require learning.	We also considered BitBucket , which offers unlimited free private repositories, but GitHub's larger use base, better integration with tools like Slack and GitHub Actions, and stronger support make it a better choice for our project.

Team Roles & Organisation

As a team of 6, we adopted a deliverable-focused approach, in which each team member is assigned as a Lead for one of the six project deliverables. This approach promotes ownership and accountability, enabling each team member to focus on a deliverable aligned with their expertise or interest. Each Deliverable Lead oversees the planning and completion of their respective deliverable and also contributes to other roles and deliverables as needed to support the team's progress.

To further support the team's workflow and maintain efficiency, each team member has a secondary role that addresses various operational needs within the project. Below is an overview of these roles:

- 1. Project Lead:** Responsible for overseeing the project's timeline and ensuring that all deliverables stay on schedule as per the Gantt Chart. They check in on team members' progress at the beginning of each meeting and help maintain focus on tasks.
- 2. Head Developer:** Manages code production, coordinates feature integration, and establishes coding standards and formatting. They ensure that development aligns with the team's chosen methodology and oversees the technical aspects of the project.
- 3. Quality Assurance:** Handles testing to ensure the product meets agreed-upon requirements, assists the Head Developer with identifying and fixing bugs, and ensures the quality of features within the product.
- 4. Report Editor:** Oversee the finalisation of all written reports and deliverable documents, ensuring they meet the standards outlined in the assessment brief.
- 5. Secretary:** This role is divided into two different positions:
 - a. Meeting Secretary:** Manages meeting documentation, prepares agenda, and takes notes. They also organise the team's working directory to ensure documents are up-to-date and easy to find.
 - b. Logistics and Communications Secretary:** Coordinates meeting logistics, including booking rooms and arranging schedules, and manages communication with the client by relaying project updates and feedback.

Our Team

Hussain Alhabib
Risk Assessment and
Mitigation Lead
Quality Assurance

Jason Ruan
Architecture Lead
Project Lead

Ellen Matthews
Requirements Lead
Meetings Secretary

Daniel Smith
Website Lead
*Logistics and Communications
Secretary*

Minnie Poon
Method Selection and
Planning Lead
Report Editor

Owen Smith
Implementation Lead
Head Developer

Our Systematic Plan

To efficiently manage the team's tasks and track progress, we created a Gantt Chart in Google Sheets with additional features to support collaboration and ensure task transparency and ownership.

The Gantt chart is organised into seven main sections: one for general project tasks and six for each project deliverable. Within each section, tasks outline the objectives for each deliverable - detailing their estimated duration, status, weight (in marks and pages), assigned team members, planned start and end dates, relevant files, and dependencies.

Initially, each task was assigned a provisional start and end date, along with an estimated duration to help provide a clear timeline for the project - providing clarity to team members. Once tasks began, their timings were updated, and team members were assigned to oversee their completion. Task dependencies were incorporated to ensure an efficient workflow for the project. For example, most architecture tasks were dependent on requirement tasks, and some website tasks required the completion of certain architecture and implementation tasks.

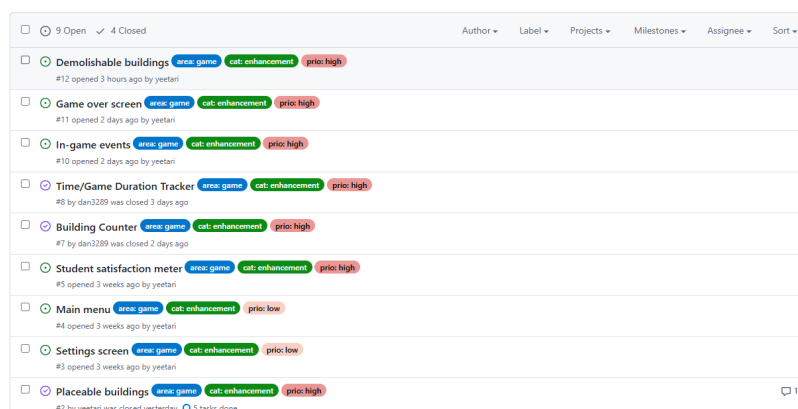
As the program's development began, we opted to use GitHub Actions and Projects to manage coding tasks instead of adding them directly to the Gantt Chart. This allowed us to streamline feature implementation and prioritise tasks according to our Agile Methodology. Each task in GitHub was categorised by priority, type, and game area.

To maintain transparency, weekly screenshots of our project's progress have been added to our team's website [<https://uoy-team-six.github.io/>] under the "Weekly Screenshots" tab, where the final version of our Gantt Chart can also be viewed.

Key Evolutions and Adjustments to the Gantt Chart:

- **Utilisation of GitHub for Programming Task Management:** We added programming tasks to GitHub, reducing potential clutter on the Gantt Chart and ensuring we focus on the priority of coding tasks essential for meeting the project's requirements.
- **Addition of Task:** In Week 4, we added "2.5 Negotiate requirements with client" to the Requirements section. This change led to us adjusting the forecasted completion dates for related tasks as we incorporated client feedback and refined the project's requirements.

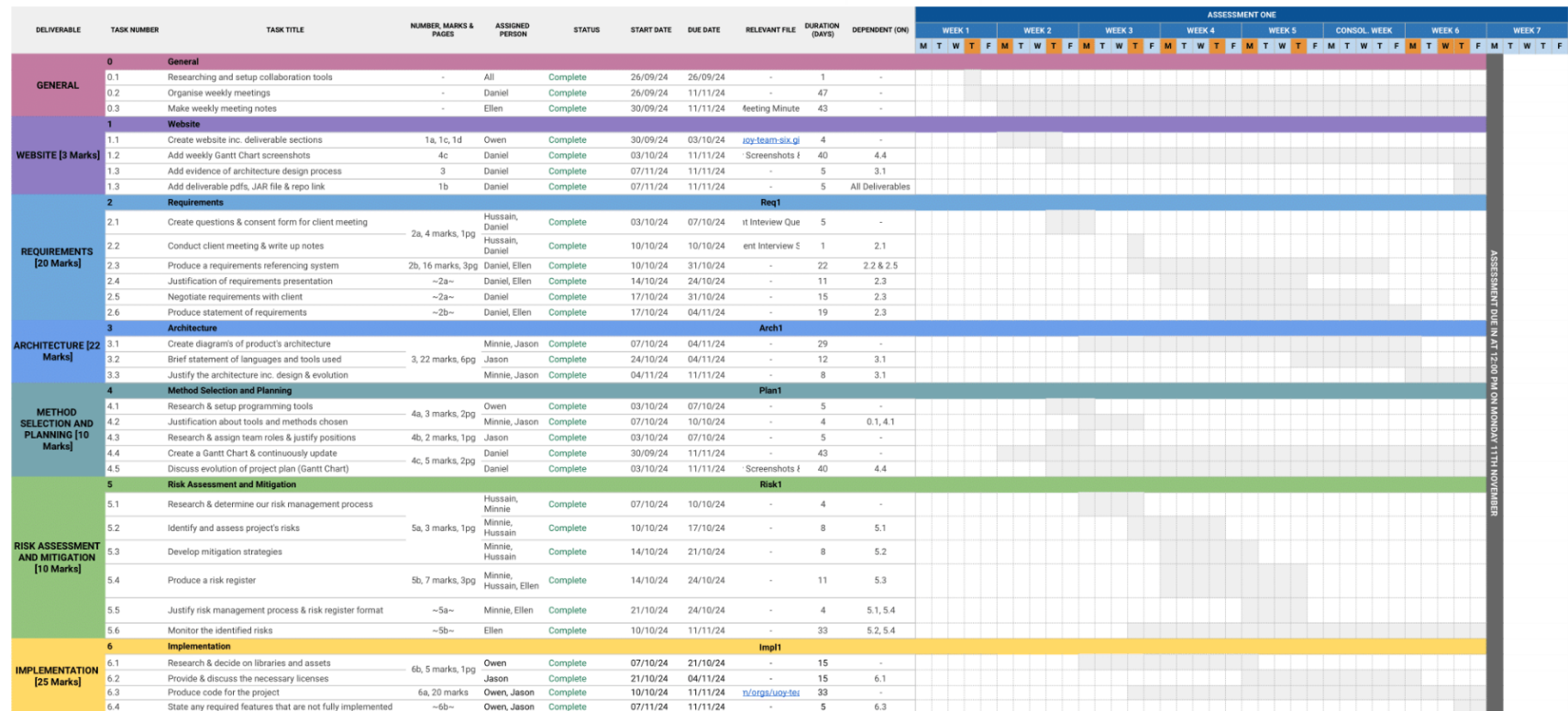
An Excerpt of Our GitHub Issues:



Each task (issue) in GitHub was categorised by priority (high, medium, or low), area (mainly "game"), and category (enhancement or bug).

Below is the final version of our Gantt Chart, reflecting the completion of our project.

Team Meetings (Completed/Upcoming)
Duration of Task



An electronic copy of this Gantt Chart is available on the website [<https://uoy-team-six.github.io/>] under the “Download” tab.