Programmers:

List of Java APIs

Name	Acronym	Functionality	Links
Java Abstract Window Toolkit	Java AWT	-Provides GUI components including Button, Label, TextField, Menu, Choice, List, Canvas, etc. -2D Graphics Drawing (points, lines, image, etc.)	[1] Oracle. “Java AWT Tutorial." URL: AWT Tutorial [2]TurorialsPoint.”Java AWT Tutorial for Beginners." URI: Java AWT tutorial for beginners
Java 2D API		-Graphics 2D supports draw points, Line, rectangle, etc; enable translation, rotation, scaling and shearing. - Advanced Rendering and Anti-Aliasing which provide rendering optimization parameters (interpolation, shadows, gradients, etc.) to enhance graphic quality.	[3]Oracle.”GRAphics2D(Java platform SE 8).” URL: Graphics2D (Java Platform SE 8) [4]Oracle. “Overview of the Java 2D API.” URL: Java 2D API
LibGDX		-Provides 2D/3D graphics rendering interfaces, including texture drawing, sprites, animations frames, shape rendering. -Built-in Box2D engine, implementing rigid body physics, collision detection, gravity friction, etc. -Provides a GUI framework featuring a Stage and Actor model, supporting controls such as buttons, labels, and menus.	[5]LibGDX Documentation Team.”LibGDX official website.” URL: libGDX - libGDX
jMonkeyEngine	JME	-Provides a complete 3D rendering system based on OpenGL (via LWJGL). Supports lighting, shadows, materials, shaders and processing effects. -Integrated Bullet Physics Engine (available in native or pure Java versions), supporting rigid bodies, collision detection , gravity, friction, vehicle controllers, and character controllers.	[7] jMonkeyEngine Team.”jMonkeyEngine Official Site.” URL: jMonkeyEngine

Software Engineering Method Selection:

The waterfall model itself is relatively rigid, requiring all requirements to be finalized early on. However, our project undergoes frequent revisions and experiments with different visual representations during the prototyping phase. Consequently, the waterfall process would constrain our development and is therefore not well-suited for our needs. Therefore we used agile development because Agile teams can respond more quickly to changing priorities,.This feedback reduces the risk of building incorrect solutions, and fosters effective collaboration and communication within the team.

Category	Tools	Purpose & Justification
Programming Language	Java (JDK 17)	Robust OOP support, an extensive ecosystem, and cross platform probability.
Graphics & Game APIs	LibGDX/Tiled	Tiled programs providing resource-efficient,elegant art style and fixed-size puzzle pieces can improve collision detection,libGDX for advanced spring rendering and physics support (BOX 2D). These API enable for rapid prototyping and efficient rendering.
IDE	IntelliJ IDEA/Eclipse	Simplify Java project management and debugging workflows while integrating Gradle/Maven dependency management.
Version Control	Git + GitHub	Supports collaborative development, problems solving and pull request review workflows.
communication	WhatsApp/ Google doc	Quick technical issue discussions, organizing meetings, and task assignments.
Task Management	WhatsAPP/Google doc	Team members report work progress as requested by the leader and display it in Google Docs.

The selected Java APIs (LibGDX) align with our interactive agile workflow due to their support for modular development:

The primary advantage of using LibGDX is it supports existing testtable game templates that can provide high-quality resources and creative inspiration.

This layered modular design enables us to incrementally expand functionality and aligns with the agile development approach adopted by the project.

During development, LibGDX can leverage GitHUb to streamline the entire development, while GitHUb manages code storage, version control, and team collaboration.

Comparison of Alternative Solutions:
We chose LibGDX over jMonkey because LibGDX is more focused on 2D game development while jMonkey leans toward 3D game development. Additionally LibGDX offers advantages in being more lightweight, flexible, and easier to learn than jMonkey.

Conclusions:

In summary, our team ultimately adopted an interactive agile development approach, utilizing collaboration tools such as GitHub and WhatsAPP, along with the java-based LibGDX framework to complete the game development. Additionally, we employed Tiled to build the level maps. This combination of tools not only supported our frequent visual testing but also made modular iterations more efficient, proving highly suitable for this 2D Java game project.

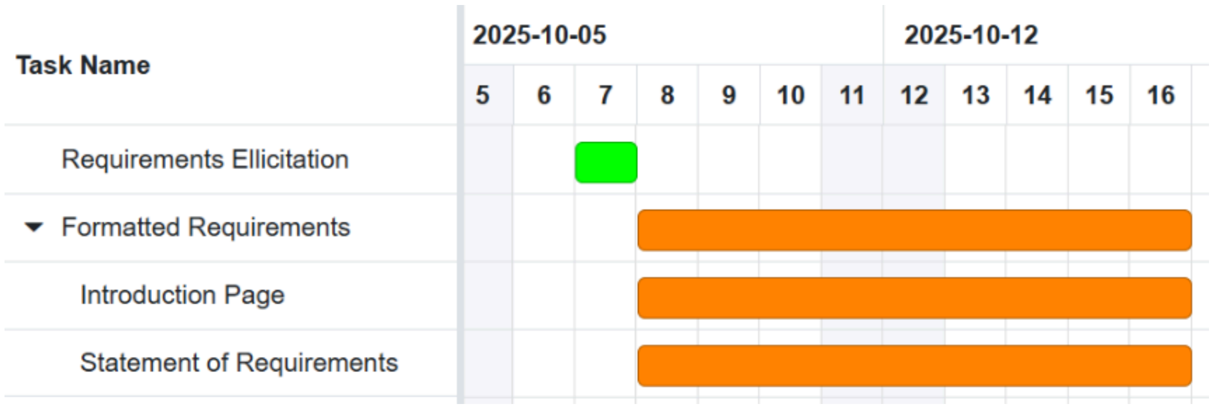
<u>Project Plan - Weekly Snapshots</u>

Week 1

- Task: Team Building
- Priority: N/A
- Dependencies: None

Week 2

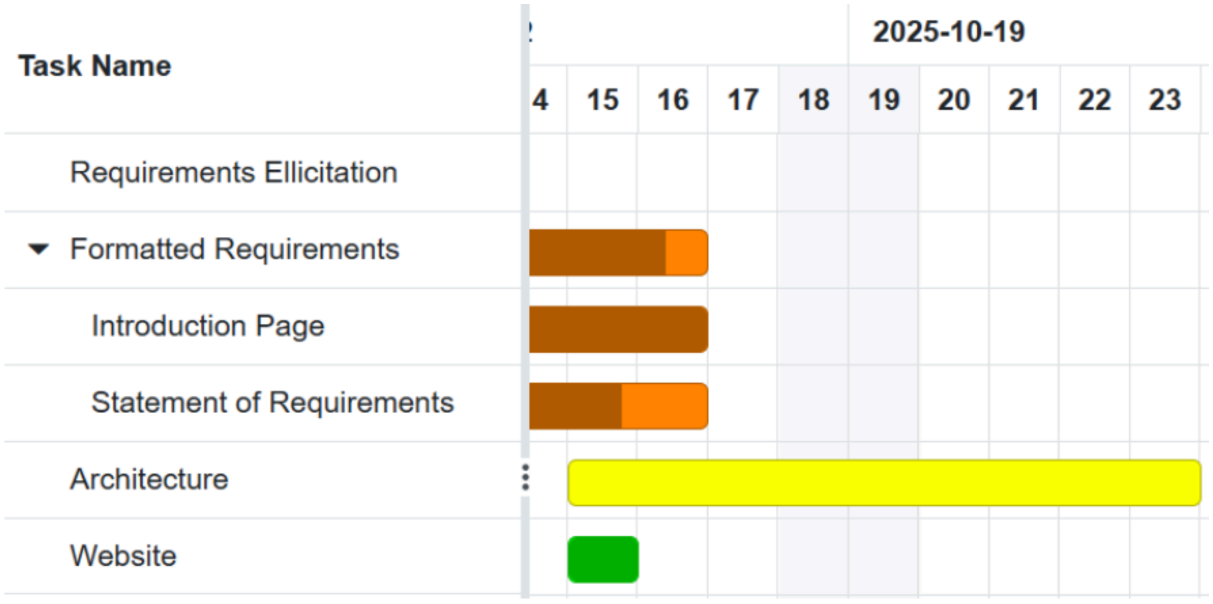
- Task: Requirements Elicitation (Customer Meeting 7/10/25 - 7/10/25)
- Priority: High
- Dependencies: None
- Task: Formatted User & System Requirements (8/10/25 - 17/10/25)
- Priority: High
- Dependencies: Requirements Elicitation
- Task: Requirements Introduction Page (8/10/25 -15/10/25)
- Priority: Low
- Dependencies: Formatted User & System Requirements



Week 3

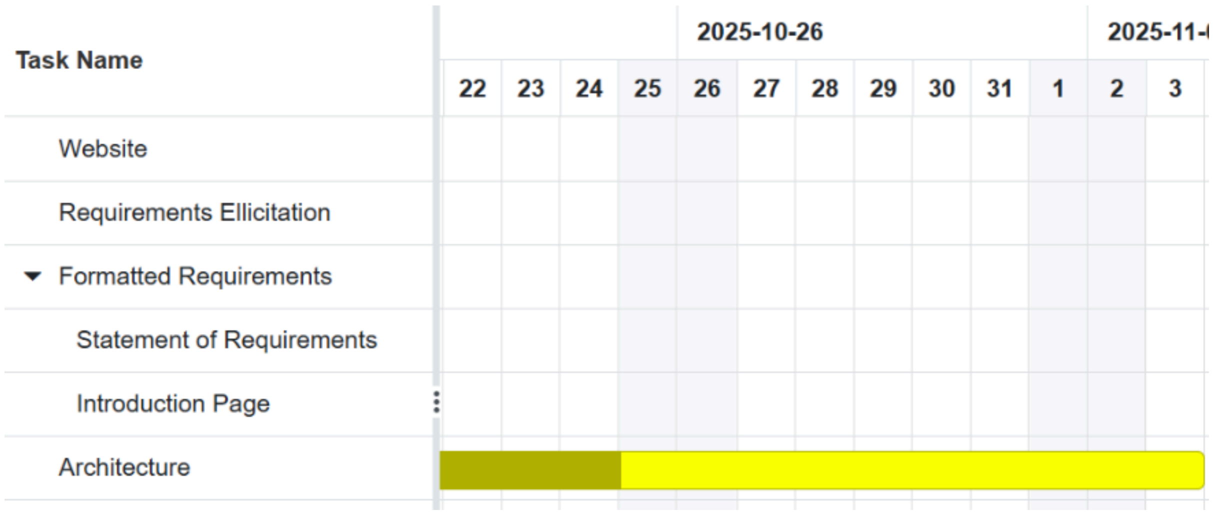
- Task: Method Selection & Planning (15/10/25 - 5/11/25)
- Priority: Low
- Dependencies: None (To be done alongside requirements)
- Task: Architecture (15/10/25 - 3/11/25)
- Priority: Medium
- Dependencies: Formatted User & System Requirements
- Task: Website (15/10/25 - 5/11/25)
- Priority: Low

- Dependencies: Method Selection & Planning



Week 4

- No change in plan



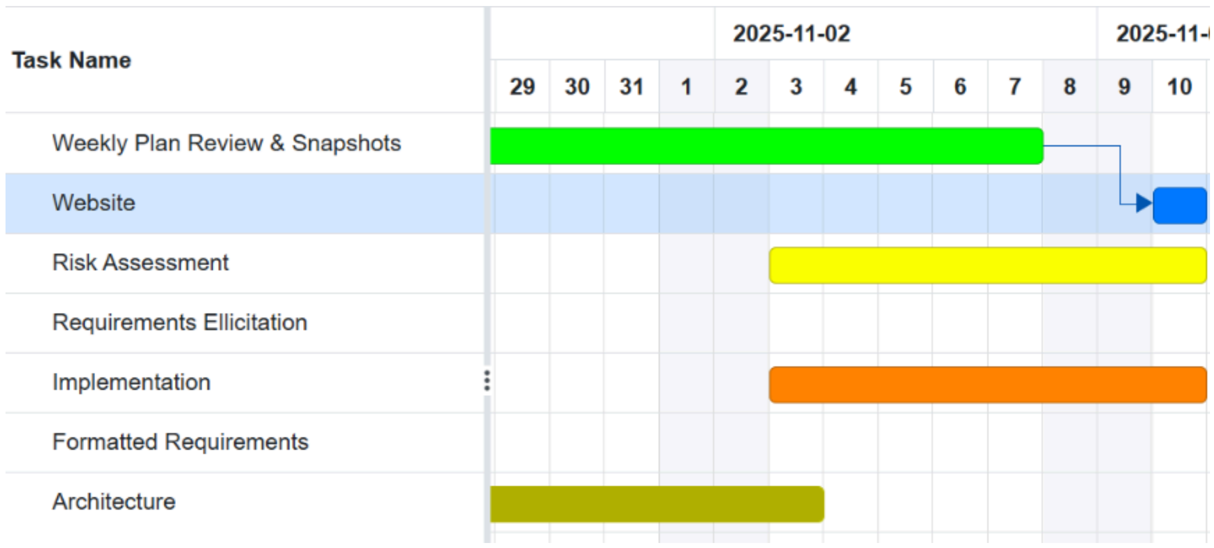
Week 5

- No change in plan

Week 6

- Task: Risk Assessment & Mitigation (3/11/25 - 7/11/25)
- Priority: Medium
- Dependencies: Formatted User & System Requirements, Initial Architecture Design
- Task: Implementation (1/11/25 - 9/11/25)
- Priority: High

- Dependencies: Formatted User & System Requirements, Architecture, Method Selection & Planning, Risk Assessment & Mitigation



Project Plan - Discussion:

Initially, we based our plan off of a deliberate attempt to tackle the tasks with the most dependencies first. Naturally, this meant that we would all work together on the Statement of User and System Requirements as to not bottleneck our productivity as we would know exactly what requirements we were working towards. After completing the customer meeting and requirements, the team had to adapt and allocate members individually to each remaining task. As shown in our week 3 snapshot, we started working on Architecture, after allocating a dedicated sub team to it, who would inform the implementation team about how the program should be structured.
Week 4's plan was a continuation of week 3's, we decided to proactively assign work to whichever task needed it.
The most significant evolution of the plan happened near the end of consolidation week, this week was used to assess our progress and plan the implementation. We dedicated members to work on implementation as well as the Risk Assessment. The plan had now shifted solely towards us working in parallel, rather than linearly, which gave us the most productivity.

Team organisation
The main way we keep our team organised is by making sure everyone can communicate with each other via a WhatsApp group. We also meet at least once a week in the engineering practical session to work and discuss the project in person. Each part of the project is given a person or multiple people to work on in their own time including having separate meetings with just the people working on that section. Any work that is done is put into a google drive which is shared with every member of the team and any documents that are used are updated in real time for everyone else in the group. For the implementation part of the project GitHub is used to make sure that everyone has the same version of the implementation.

A WhatsApp group is appropriate to keep the team organized because there are only 8 people in the team so there are not too many messages that people cannot keep up with.

Having meetings at least once a week is also appropriate because we are students at the same university so it is easy to have a timetabled meeting. Splitting up the project so each person has a section of it is appropriate for the team because having everyone do everything would just make the project disorganized. Splitting up is appropriate for the project because the assessment document already splits up the project into separate requirements so it is easier to give these to people. Having a google drive is appropriate for the team because it means that everyone can work on their own device in their own time and what they have done will still be seen by everyone else. The use of GitHub is appropriate because multiple people will be coding the project and GitHub makes sure that the changes people do will not conflict with each other.