Method selection and planning

4a

The software engineering methodology we used was agile scrum. We used agile due to its popularity as it has extensive use in professional teams and its ability to handle complexity and variability especially since our team members all have different constraints such as jobs, studies and other extracurricular activities. This meant it was very important to select a methodology that could take this into account and as a result agile was a very natural fit. We used scrum because we felt like it integrated structure and control with the flexibility of agile. By using Scrum it forced us to divide our development into sprints which helped to focus the project since the prior planning meetings helped to guide the team on what key features we needed to implement. The 3 stand-ups we had in a week helped everyone on the team as it ensured no one had too much or little work to complete and if ever a problem came up, everyone's insight were taken into consideration to come up with the best solution. Another reason we picked scrum was because of the sprint reviews, which enabled us to improve our efficiency by reviewing mistakes or inconsistencies present in the previous sprint.

The developmental tools we used were Github and the Intellij IDEA. We used github to host our repository due to the additional services it provides such as documentation, tracking changes and pages - that we used for the website. Most members of the team were already familiar with Github, which meant that it was quick and easy to begin using it. We used Intellij as our IDE for everyone because it helped reduce setup time, increase productivity and standardise the development process. Eclipse was considered as well due to its customizability but Intellij had more features and in our opinion a better debugging tool after testing both of them. The customizability also felt overwhelming as none of us had any experience with it. The IDE also lended itself well to development with the LibGDX library - it was easy to keep track of scenes, screens and the classes that made up the game.

The Collaboration tools we used were Discord, Google drive and Kanbanflow. Discord was primarily used for our weekly meetings when everyone would join the call and do their standup. All of the team members had used Discord before so no time would be wasted in explaining how it worked. One advantage of discord is its integration across devices, we could call and message from our mobile phones for convenience, but also join calls on our computers so that we could share our screens to explain code, solve problems and easily share and upload files. Google Drive was used to store documents that we used to work on for the project and that team members may need to refer back to at some point. This included the Gantt chart, team meeting minutes, section sheets (Requirements, Architecture, etc) and other documentation. By having all our documents in one shared drive it helped ensure the security and protection of files while allowing everyone to access it whenever they needed to. The ability to share documents also meant that we were all able to edit and contribute to the deliverables, given that we all have a unique insight on the development process, it allows our deliverables to reflect this. Google Teams was also considered as an additional collaboration tool but we felt by having too many lines of communication (Discord, Drive or Teams) we risked losing focus by having to look for messages or files as we may not have remembered which service we used to send it. We also used an online tool called Kanbanflow in order to create dynamic to do lists. We were all able to create tasks and mark them as “To-do”, “Doing” and “Done”. This was immensely helpful in helping us to stick to our plan and keep track of what was still to be done. This helped us to prioritise and distribute tasks, improving our efficiency. Additionally, we were able to mark tasks as implementation

or documentation, which further aided organisation. By referencing this board and the gantt chart, creating plans was easy and intuitive. An alternative to using Kanbanflow would have been another online project management tool such as Trello, which - like Kanbanflow - allows users to create and categorise different tasks. It includes more organisational and communication features, however we decided that these were unnecessary for our purposes, and could potentially be distracting, given that we already have systems in place for these purposes. The simplicity of Kanbanflow was a big advantage to us.

4b

The team initially worked together to plan out the project, decide which tools we would use and the overarching methodology that would guide us throughout. As the project progressed we split into two groups: one for the development while the other handled the report, however this was a very flexible arrangement as both groups contributed to each other's sections whenever needed. The technical team followed an agile methodology using scrum to organise their works into sprints. The report based team each took up writing different parts of the report like Architecture, Requirements, etc.

This approach fitted the wishes of the team as some members wanted to focus more on developing the game itself while others felt more comfortable writing instead of coding. It was also a great fit for the project as it meant we did not get carried away with coding and were able complete the different sections of the report.

4c

Key Tasks

Task	Start date	Dependencies
Requirements Elicitation	17/11/2022	N/A
Initial Architecture	23/1/2023	Requirements elicitation
Risk Assessment	6/12/2022	N/A
Implementation	23/1/2023	Requirements, Architecture
Architecture Evaluation	23/1/2023	Implementation, Initial architecture

The planning of the project firstly focused on outlining the key tasks we needed to complete in order to be successful. We followed the requirements of the report when making our own plan so we split tasks under the Website, Requirements, Architecture, Method selection and planning, Risk assessment and mitigation and Implementation deliverables.

To properly illustrate the schedule of the project we used a Gantt chart (which can be found on our website) that allowed us to show the dependencies between activities and the current schedule status so we would not fall behind. The chart was structured based on the sections of the report and then split into key tasks we needed to complete for that section. Each task was given a start and end date. However, many tasks required more or less time than we had initially planned - this slack is known as the float time in a gantt chart. The team was aware of this risk from the start so we started work quickly to ensure we had time left over at the end to account for any tasks. We uploaded screenshots of the gantt chart at the end of every week to show our progress and how the plan evolved week to week.

1. Project Conception and Initiation
    1.1 Tools setup

- 1.1.1 Github Setup
- 1.1.2 Discord Setup
- 1.1.3 Google Drive Setup
- 1.1.4 Gantt Chart overview
- 1.2 Client Meeting Preparation
  - 1.2.1 Go over Script for meeting
- 1.3 Actual Client Meeting

2. Website Deliverable
   - 2.1 Hosting discussion - Github Pages or Firebase?
   - 2.2 Github pages setup
   - 2.3 Weekly snapshot upload
3. Method Selection and Planning Deliverable
   - 3.1 Discussion for IDE
   - 3.2 Discussion for Game engine
   - 3.3 Discussion for work flow
   - 3.4 How to use GitHub effectively
   - 3.5 Formal Write Up
4. Risk Management
   - 4.1 Discussion on the management process
   - 4.2 Come up with ideas on potential risks
   - 4.3 Possible ways to mitigate risk
   - 4.4 Formal Write Up
5. Requirements
   - 5.1 Research into requirements specification
   - 5.2 Requirements discussion for elicitation and presentation
   - 5.3 Formal Write Up
6. Architecture
   - 6.1 UML
7. Implementation
   - 7.1 Game design
     - 7.1.1
   - 7.2 Report write up
     - 7.2.1