Method Selection and Planning

Group: Group 10

Names:
Haaris Altaf
Casper Co
Will Garavelli
Kamso Osuji
Krishna Rajamannar
Diyar Savda
Sam Storrs

Justification

Justification for website

We decided to use GitHub as it was a commonly used website for software development, and it allowed different versions/backups of the project. GitHub also allows us to raise problems and set milestones, allowing us to communicate and assign problems to our team members, whilst also showing a timeline of the project. We established Google Drive to be where our documents were stored was accessible and familiar to everyone, but decided GitHub would be better overall and provide a learning experience for those of us who weren't familiar.

Justification for UML designs and diagrams

To visualise our code's class hierarchy, we decided to use UML diagrams to show how the classes could be organised. Lucidchart is a simple to use, free website that allows us to design charts and diagrams that we have experience in. It also has tools that allow us to export our diagrams easily. We considered using PlantUML as we were already using it for Gantt charts, but that would require us to spend time learning the syntax when we were already familiar with Lucidchart.

Justification for software engine

For our software engine, we wanted to look for an engine that uses java, is free and popular, as this will allow us to get help from the community on any questions or troubles we may encounter. We also needed an engine that has a licence that allows us to commercially publish our game. We decided to use libGDX as it uses the Apache Software license, which allows people to create and distribute games for any purposes, without concerns for royalties, giving us plentiful freedom for our creative minds. We found online guides on making Maze games on libGDX, which we can use as a baseline guide on our game. The software engine also provides tools for documentation, and allows for cross-platform compatibility, and has a built in physics engine, which will reduce our time spent on programming.

Justification for plantUML for gantt charts

We wanted an open source software to build our gantt charts. We decided to use plantUML as it allows for flexibility in design, is simple to use, and can be easily shared for everyone to access and edit. It also generates a png image of the Gantt chart automatically while running, making documentation of our progress much easier and consistent, and lets us write notes under each task to elaborate or provide additional details. We considered using Excel, but we wanted something that supported the direct use of Gantt charts. We also considered ProjectLibre, but we saw it used AI, which we wanted to avoid using. With PlantUML, we were able to easily export the photos onto our website.

Justification for Whatsapp and Zoom for Communications

Whatsapp was an obvious and immediate choice for us as we were all familiar with it, and it was a professional chatting platform that allowed us to share photos, videos and voice messages. Whatsapp is also a primarily mobile application that's linked to a mobile number, allowing us to have quick and direct communication in case of an emergency. We considered discord, as it allowed for us to screenshare and create multiple channels for different topics, but decided it was a bit unprofessional and unnecessary. Hence, for online meetings where we needed to share our screens, we used Zoom, another application that we were all familiar with, and also allowed us to create breakrooms to separate our conversations if needed.

Team Approach

After going through the assessment paper and assessing how to properly allocate time to each marking point, we decided to implement the waterfall method as it allowed us to have a clear view of our project timeline, and we only needed to implement a few basic gameplay mechanics.

In our requirements analysis phase, we discussed our mandatory features and how to implement them, as well as began planning our timeline during our group meeting as we had a tight deadline. This allowed us to have a visualisation on how to work on the project in our own times, separately. As we had all the information required, we began properly allocating our team to efficiently cover all marking points.

We decided to split our team into two groups: Casper, Sam, William and Krishna would be in the generalist team, whilst Diyar, Haaris and Kamso would be in the hybrid team.

The generalist team was tasked with documenting the Requirements, Method Selection and Planning and Risk Assessment and Mitigation. The team would communicate with each other to manage the project plan and timeline, as well as discuss with the software team ideas to implement into the game and document.

To speed up the documentation and reduce overlap of information, Casper would focus on Method Selection and planning, William and Krishna on Requirements, and Sam on Risk Assessment and Mitigation. We decided to have two people work on Requirements due to the marks being worth double in comparison.

The hybrid team was tasked with software development, as well as documenting the Implementation and Architecture. We decided the software documentation would be easier if done by the software team, as they could highlight specific software design choices made during software development. Having a smaller software team reduced redundancy in tasks, and having specialists work on the coding and design was more efficient.

To speed up the development of the software, Haaris would focus on basic gameplay mechanics, such as movement, collision/wall detection and the hub UI, whilst Diyar would focus on the gimmicks, such as the power-up, obstacle and hidden event. This allowed our software engineers to work on parts of the software that did not rely on or interfere with each other until late stages of the software development. Kamso was tasked to work on the map design by using diagrams, and create multiple maze designs to suggest to the group so we could discuss which designs we preferred, and be implemented by the software engineers at a later stage. Each software team member would contribute to the Architecture and Implementation documentation, in respect to their specific tasks.

Systematic Plan of Project

In terms of the overall project, we aim to finish this part of the assessment by November 3rd. This is a week before the actual submission outlined in the brief (i.e. November 10th). This week acts as a contingency in case something does go wrong during the course of the project. It ensures that we aren't rushing to finish the project.

Below we've outlined the key tasks for the project and the originally decided start/finish dates of each task.
We also provided the weekly snapshots of the plan on our website.

Overall Key Tasks (Including Documentation):

- Initial Research (i.e. Research on LibGDX and programming libraries)
  Due Date: 12th October

- Write documentation for Requirements, Risk Assessment and Method Selection and Planning
  Due Date: 19th October

- Implementation of Game and writing documentation for Architecture
  Due Date: 26th October

- Recheck the documentation written and developed code, make a presentation for other groups
  Due Date: 2nd November

Breakdown of Tasks for Game Design and Implementation :

Priorities:
🟥 Red = Urgent, Required Requests for the Client or Critical to Development
🟨 Yellow = Moderate, Mild Problem if not Completed
⬜ White = Optional, Minor Hindrance if not Completed

Stage	Key Task	Start Date	Finish Date
🟨 Requirements	LibGDX Research	29th September	7th October
🟨 Design	Design Brainstorming	29th September	7th October
🟥 Design	Map Design	7th October	14th October
🟥 Implementation	Player Movement	7th October	14th October
🟥 Implementation	Timer	7th October	14th October

Implementation	Event Creation	7th OCtober	16th October
Implementation	Enemy Creation	7th October	14th October
Implementation	Counter **(Dependent on Event Creation)**	16th October	21th October
Implementation	Map Implementation **(Dependent on Map Design)**	14th October	21st October
Testing	Playtest Game	21st October	24th October
Testing	Fix Bugs	21st October	1st November
Submission	Create Presentation	21st October	10th November
Submission	Recheck Project and Submit	1st November	10th November

The dates above from the overall key tasks and task breakdown for the implementation represent our initial plan on how we could complete the project within the given timeframe. However, whilst working on the project, we realised that we needed to extend the deadlines of a few tasks. In particular, we needed to push back the deadline of writing the documentation for requirements, method selection and risk assessment by another week.

To ensure that we meet our deadlines for our respective tasks, we used milestones on GitHub to keep track of overall documentation related tasks that we needed to complete. We've attached a snapshot of how we did this on our website.

For every task (i.e. writing the requirements deliverable), we would open a new issue on GitHub and we would assign a deadline to that task. We would then assign a team member to work on that issue and they would close the issue once it has been completed.