Method selection and planning

Cohort 2- Group 1

Members

Ollie Barlow (vrq506)
Jacob Barnes (bdp514)
Zayaan Bawany (fst508)
Lily Boldbaatar (jkk522)
Georgia Briggs (rxm503)
Evelyn Gravett (qnf502)
Hassan Yunus (wfv505)

Software engineering method and collaboration tool

We went with the Agile method for this project as it was preferred in the lecture. GIven the short timeline and relatively small scope of the project, Agile methodology was the ideal fit because it allowed us to work in quick cycles rather than committing to a long-term plan from the start and allowed us to focus on iterative development and regular improvements. The main benefit of the Agile method was its flexibility, letting us adapt and pivot based on our weekly progress. This method made it easier to manage the collaboration aspect, where different team members could contribute to the same things rather than waiting for one part to finish before starting another.

In terms of tools, we relied on Google Drive and GitHub. Google Drive was our go-to for documentation because of its collaborative features. We organised our files in a shared Google Drive folder so that all the team members could access and edit the documents whenever needed. We did not consider alternative tools for our collaboration because Google Drive has a simple interface and everyone on the team was already familiar and comfortable with using it.

When it came to managing our code, we used GitHub. It allowed our team members to work on the code without worrying about overwriting each other's work. For creating the building assets and map we used a pixel art software 'Pixil' that allowed us to efficiently create drawings that were simple but effective and it had a wide range of features giving us full creative control. We decided on pixel art over traditional art after drawing inspiration from other top down games such as Pokemon and Stardew valley. We decided to design our own assets to ensure a consistent art style throughout the game to improve the user experience.

Team organisation approach

Our team consisted of seven members, and we structured our work based on the main deliverables: documentation and implementation. This approach allowed us to divide the workload efficiently, with one group focusing on code development while the other concentrated on writing and structuring the final deliverables. We thought it would be a waste of time if all team members did the codes and the writing deliverables. We did not assign a formal leader, but we made sure responsibilities were clear for each task. We made sure not all crucial tasks solely relied on one person. So that if the task owner cannot contribute to that task again we would have another person who can continue the task. This division of labour helped us stay productive, as those working on implementation could focus on coding without having to juggle writing tasks simultaneously.

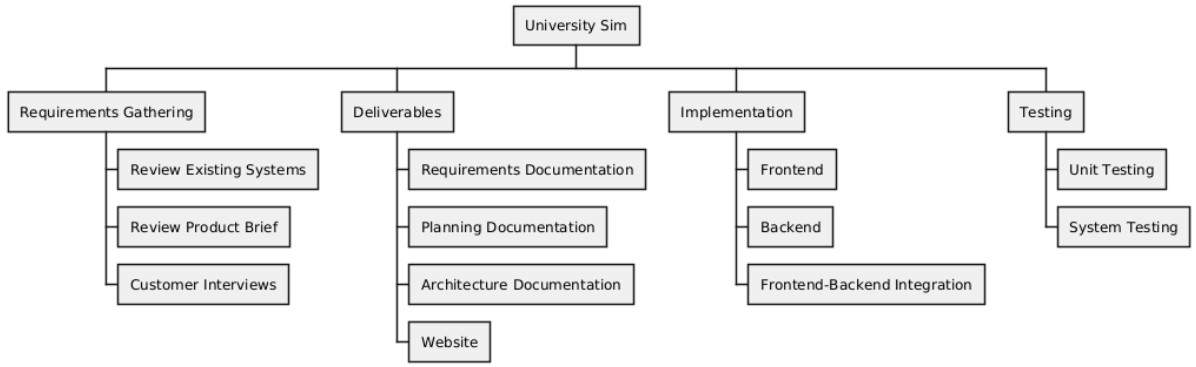
We held regular meetings twice a week. Initially, we only met on Thursday after our practical session to check progress, but we soon realised that this was not enough to keep up with our project's needs. So, we added Monday meetings to our schedule. During monday meetings, we reviewed completed work from the previous week and assigned new tasks, while Thursday meetings served as a progress check-in where we could discuss any challenges or delays. This twice-weekly rhythm helped keep everyone accountable, and it gave us enough time to troubleshoot any issues mid-week instead of waiting until our next meeting.

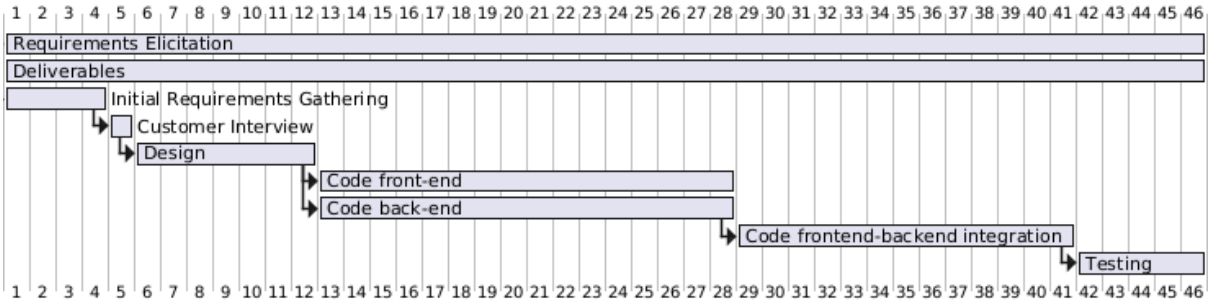
For communication, we started with a group chat on Instagram, but we switched to Discord. Discord offered a more organised platform, with channels that helped keep conversations about specific topics like design, testing, and documentation etc, separate. Discord's file

sharing capabilities also made it easier to share updates and work together outside of our regular meeting times.

Systematic project plan

The following work breakdown structure and gantt chart show a simple overview of what needs to be done in the project and the order in which they need to be completed.





The following work packages expand on the previous diagrams by going into more detail on each of the tasks.

Requirements elicitation (WP1)
In order to make a university simulation match the customer's vision, we need to determine requirements for the university simulation. These will be determined through a customer interview and reviewing existing systems.
T1.1: Review existing systems and product overview. ● To help us determine the requirements for the university simulation, we can look at existing systems that have the same purpose or have features that may relate to the simulation and the given product overview. ● Start: 3/10/2024 ● End: 8/10/24 T1.2: Customer interview. ● To get a clear picture of the customer's vision and requirements, we can ask questions to the customer during an interview. ● Start: 8/10/2024 ● End: 8/10/2024 ● Dependencies: T1.1

Design (WP2)

Before and while doing implementation, we need to design the methods and systems that we are going to be using to make this university simulation.

T2.1: UI design
- Design a user interface that is easy to learn and navigate.
- Start: 8/10/2024
- End: 28/10/2024
- Dependencies: T1.1

T2.2: System design
- Designing the methods and classes that will be used to make the university simulation work and fit the customer's vision.
- Start: 8/10/2024
- End: 28/10/2024
- Dependencies: T1.1, T1.2, T2.1

Implementation (WP3)

Creating the usable program following the design created in the previous work package

T3.1: Backend implementation
- Coding the backend (logic) of the university simulation.
- Start: 15/10/2024
- End: 31/10/2024
- Dependencies: T2.2

T3.2: Frontend implementation
- Coding the frontend (UI) of the university simulation
- Start: 15/10/2024
- End: 31/10/2024
- Dependencies: T2.1, T2.2

T3.3: Frontend-Backend integration
- Coding how the frontend and backend will interact.
- Start: 18/10/2024
- End: 7/10/2024
- Dependencies: T2.2, T3.1, T3.2

Testing (WP4)

Testing the implementation of the project to make sure that it works and won't crash unexpectedly

T4.1: Unit tests
- Testing individual methods of the program
- Start: 15/10/2024
- End: 8/10/2024
- Dependencies: T3.1, T3.2, T3.3

T4.2: General system tests

- Testing the entire project
- Start: 18/10/2024
- End: 10/11/2024
- Dependencies: T3.3

Deliverables

All deliverables are due on the 11/11/2024. All deliverables will be posted and visible on the website. All interim versions of the PDF deliverables will be a google docs file of the same name.

D1: Website
- Produce a website that has links to all other deliverables.

D2: Method selection and planning documentation
- Goes over software engineering methods, team planning and a general project plan.

D3: Risk assessment and mitigation documentation
- Goes over risk identification, risk analysis, risk mitigation and risk monitoring.

D4: Implementation documentation
- Lists any 3rd party libraries or assets we used in the implementation as well as the licences for our project. Also, write about any not fully implemented features using the requirements.
- Relevant tasks: T3.1, T3.2, T3.3

D5: Requirements documentation
- Goes over the user requirements, functional requirements and nonfunctional requirements.
- Relevant tasks: T1.1, T1.2

D6: Architecture documentation
- Gives visual representations of the architecture of the product and goes over the design process of said architecture.
- Relevant tasks: T2.1, T2.2