Method Selection and Planning

Cohort 1, Group 5
Team Name:
JAzZ MoLeS

Group Members:
Sophia Taylor, Lucy Wood, Mitchell Gilbert
Jamie Creed, Archie Adams, Zayed Iqbal

Selection and Justification of Software engineering methodology and development tools

Our team has adopted an Agile development methodology, notably practises similar to Scrum, which includes “regular meetings, continuous communication with clients, transparency in progress, and consistent retrospective meetings” [1, pp. 148-165]. The choice of this methodology enhances the project's flexibility and strengthens the capability to swiftly respond to changing requirements. Additionally, it fosters cooperation and communication within the team and supports a continuous improvement process. We felt that active participation and continuous communication was vital for our teams success for this project.

We utilise GitHub for code version control and task visualisation, Google Drive for the dissemination of documents and deliverables, slack and whatsapp for real-time communication, and zoom and google meet for conducting retrospective meetings to deliberate on project progression and areas for improvement. These tools align with our work processes and the Agile development methodology, particularly in augmenting transparency and fostering efficient teamwork and collaboration

Transparency, in our project’s framework, is not about public visibility but rather about the clear visibility of project changes, decisions, and progress within the team. Git enables this internal transparency by providing a comprehensive history of code changes, who made them, and when they were made. This historical record is invaluable for understanding the evolution of the project, diagnosing issues, and facilitating effective collaboration among team members who are working on different aspects of the project simultaneously.

Additionally, the adaptation of Git supports our Agile methodology by enabling iterative development and frequent integration.

For our use of this website, we decided it must be accessible for the duration of the project and free to use. There are many options available at our disposal for this, however, we have decided to use GitHub Pages. During team discussion, we discovered that a member of our team has had experience with this software before. With this added experience it makes it easier to estimate the duration of time it would need to be allocated to it, as well as identifying any risks associated with it, so we decided it would be the most effective and efficient choice for the team's website. Additionally, GitHub Pages is easily built into the GitHub Project, allowing us to work on both the game and website simultaneously.

Our selection of primarily complimentary tools was driven not solely by budget considerations but also by the need for efficiency and transparency. Upon reviewing various alternatives, these tools were adjudged to best fulfil our team's requirements.

From initial discussions there were 2 clear options for the IDE we would use to create our project: VS Code and IntelliJ.

Despite us all having previous experience with VS Code, it was determined that we would use IntelliJ as this option specialise in Java development. Additionally, one of the main group

members working on the implementation of the project has more experience with this specific IDE. As such, it was justified that we would choose IntelliJ as our IDE of choice. We selected primarily available free tools, not merely due to budget constraints but for efficiency and transparency. Other tools and services were considered; however, these were chosen as they best fit the team's needs. GitHub and messaging applications, in particular, were determined to provide the greatest benefit for both the development process and team communication.

With the Agile development methodology and our chosen tools, we are propelling the project forward effectively and efficiently, deeply ingraining these choices into our workflow and Agile practices, thereby contributing to the fulfilment of project objectives, enhancing team efficiency, and elevating the quality of the final product.

Approach to team organisation

Our approach to team organisation has evolved throughout the project, as we became more familiar with it and each other, and learnt what worked and what didn't work. We had to find the balance between too much organisation, which would be cumbersome, and too little, which would result in a lack of coordination [1, pp. 148–165]. We also had to consider that we would not all be working 9-5 in the same office, so couldn't rely on simply observing what other people were doing to stay up to date. Conversely, spending huge amounts of time on documentation and needless processes would be wasteful, as the team only consists of six members, and the project is not safety critical.

At the beginning of the project, we initially tried allocating roles. However, we felt that these roles weren't necessarily working for us as individuals, or in terms of working collaboratively and efficiently. From then on, we chose to move to more fluid roles and work more collaboratively on tasks. This was a much better fit for us personally.

We decided collaboratively, on whatsapp and/or slack, when we felt we needed to meet and discuss progress of what we had all done thus far. These meetings consisted of in-person and online meetings when working remotely from the university. We felt that doing in person meetings when we were able would be the most productive and engaging way of holding them, as we found that people were more likely to share any ideas they had in person than online. Additionally these can be seen as a way of team bonding, and this feeling is backed up by textbooks [1, pp. 148–149].

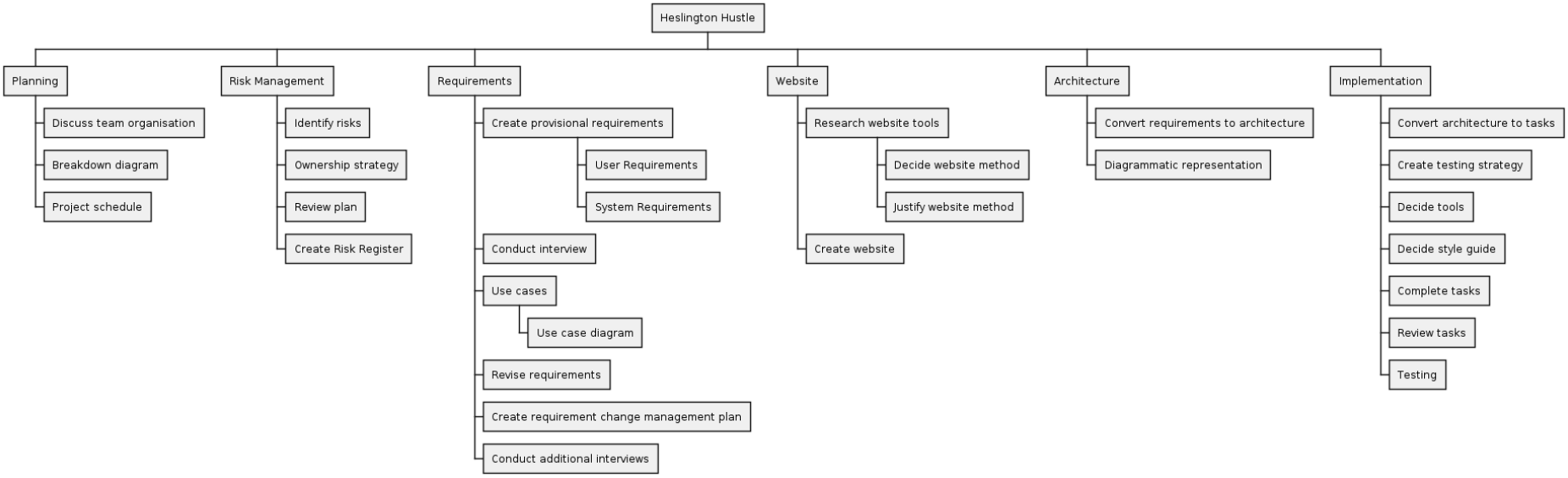
We created a logbook on a shared Google Docs document in order to keep track of our current work. During our meetings, we identified what work needed to be done, broke it down into specific tasks, delegated people to said tasks. These tasks as well as a summary of the points made in the meeting were written to the logbook. The logbook had a list of tasks at the end, noted with their status (aka completed, in progress) and the owner of the task. The information in relation to its status is then updated as the tasks are completed. This was useful as it allowed everyone to see, in one place, the work that needed to be done between meetings and practicals. Writing names next to the tasks was helpful in terms of making it clear what everyone is responsible for. Overall, breaking down what needed doing into specific tasks made it clear what needed to be done, and allowed for the work to be more evenly and easily distributed.

In order to make sure the quality of our work was held to a good standard, toward the end of the project, before submission, we held a meeting to discuss the work that everyone had done. In this meeting, we sat down, read through everyone's section on the report and made adjustments if we felt that they were necessary. Having everyone review and read through the work, we can identify potential mistakes or suggest improvements. Additionally, it ensures that more than one team member is familiar with each aspect of the project, which is important in the event that a member becomes unavailable.

Throughout the duration of the project, communication was upheld frequently between all of the group members via the methods mentioned above, that way if anyone needed any help with anything, or clarifications about deadlines etc. it was clear to everyone. (need to rewrite)

Systematic plan for the project

At the start of our project, we created a work breakdown chart using PlantUML. We did this by reading the assessment brief and product brief and breaking the project down into smaller, more manageable tasks. The purpose of a breakdown chart isn't to provide a timeline to complete the tasks, but rather to understand the key activities. This allows us to have an understanding of what the main hurdles and roadblocks will be before we create our projected timeline for completion of our project. We were more confident in our breakdowns of the sections that we would undertake towards the start of the project compared to the ones towards the end.

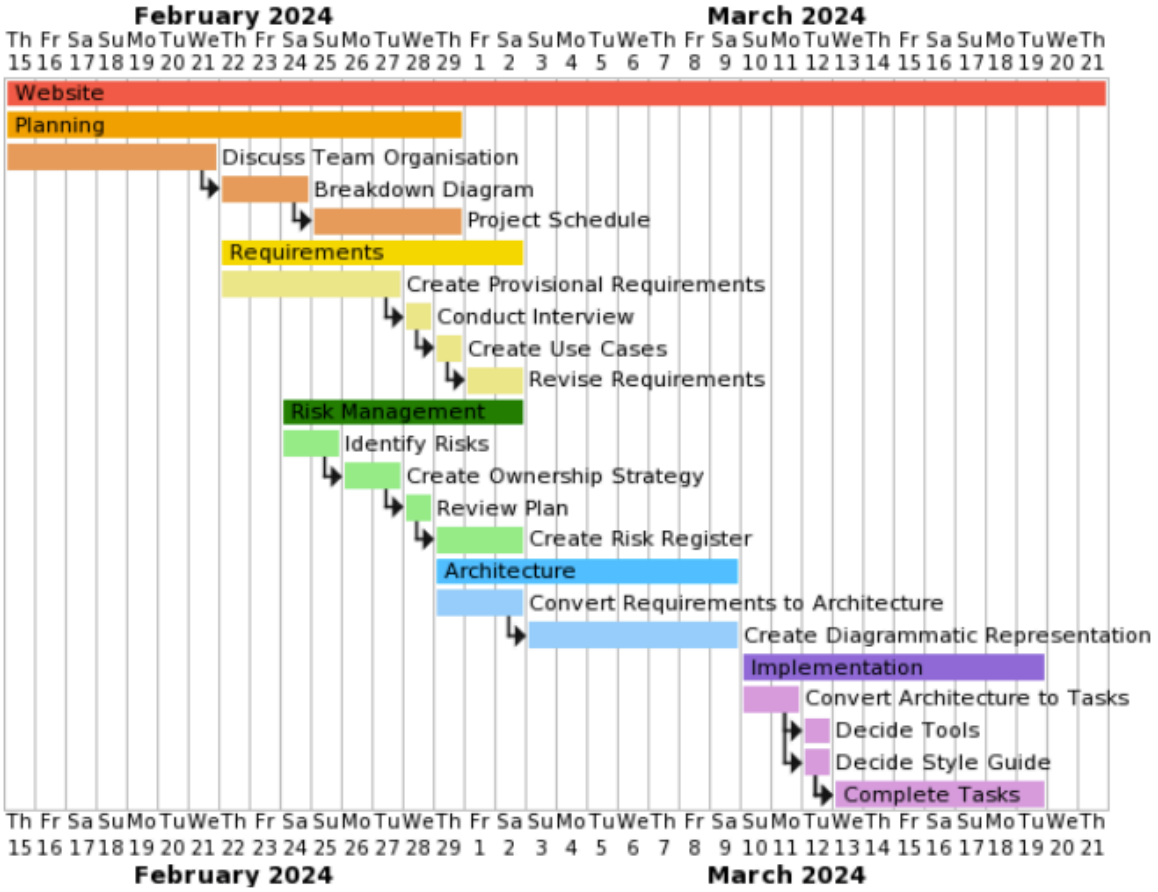


After completing our work breakdown chart, we then started work on the Gantt chart, also using PlantUML. The main purpose of this was to 'visually represent a project schedule'[2] from the start of our project to the end date. As it was our first plan, we were not expecting to be strictly sticking with the dates outlined on the Gantt chart, but rather use it as a guide and edit the chart using UML when problems arose. The darker colours chosen show the broad sections, signified as headers in our breakdown chart, with the smaller, more manageable tasks signified with the paler colours.

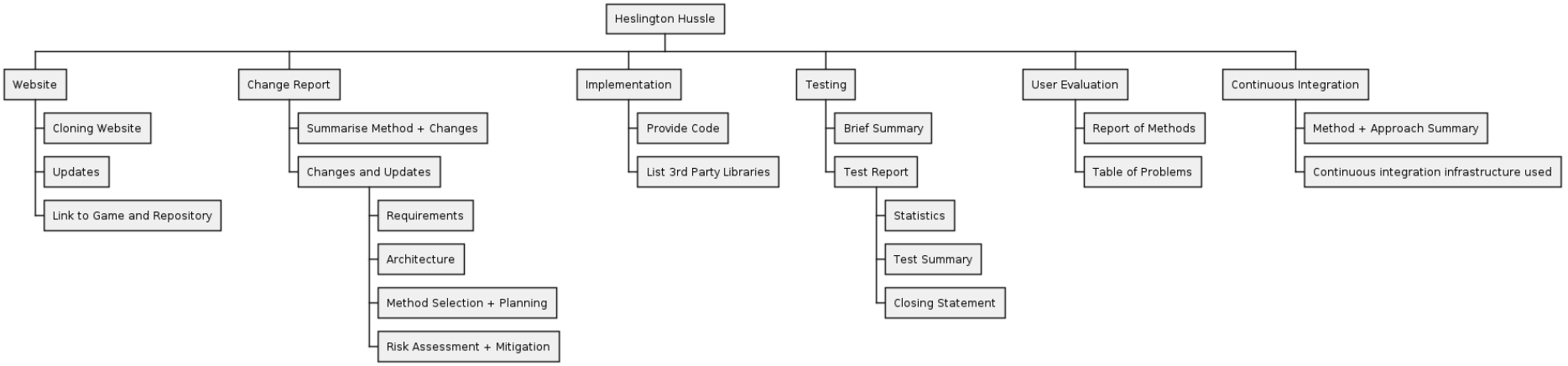
In our initial version of the chart, we used the start date of the first in-person practical, Thursday the 15th of February and the projected end date of the submission deadline, Thursday the 21st of March. Overall, this gave us 5 weeks to complete this section of the project. We predicted that the most time consuming section would be the implementation, and that we would not be able to begin work on it until all the other sections had been completed. The cascading design loosely shows that some work was dependent on other work, although much of the planning and risk management could be worked on alongside other areas.

As the project progressed, we found ourselves mostly sticking to the plan outlined in the original version of the Gantt chart. This was the case until we started to work on the Architecture, and found that figuring out the game logic and class diagrams would take longer than first anticipated. As a result of this, we pushed back the beginning of our implementation by 3 days and removed the dedicated testing portion of the plan. Furthermore, we realised that after completing the implementation portion of the project, we would need at least a day to compile our work onto the website, proofread and make any changes necessary.

This brings us to the final edition of the Gantt chart, where the plan for the project has stayed mostly the same. The final date has been brought forward to Wednesday the 20th of March. But overall, the breakdown plan and initial Gantt chart created in week 1 proved to be very helpful for us as a group to delegate smaller tasks and give appropriate deadlines for that task.



On taking over this project, to begin with, we created a work breakdown chart using PlantUML. We did this using the second part of the assessment documentation, and the new product brief with the additional requirements. Using this, we highlighted the key activities that will be allocated between group members based on the quantity of work and marks.



The diagram above could be considered the second half of the breakdown diagram created two pages prior. The sections under ‘Change Report’, specifically ‘Changes and Updates’ refer to the sections specified in the original diagram. The additional sections represented are taken from the second assessment in the project and with additional requirements, is a continuation of the project overall.

After completing the work breakdown diagram, we felt that going forward, it would be more efficient and beneficial for us to know what tasks we would be responsible for. Hence we created a task allocation table in order for us to clearly visualise the tasks for us to do for the rest of the duration of the project. This table is located below:

Task	Person
Website [3 Marks]	Lucy
Change Report [23 marks]	Jamie, Sophia, Lucy
Part a (3 marks) - Brief Summary	Jamie
Part b (20 marks)	Jamie, Sophia, Lucy
Part b i {5 marks} - Requirements	Jamie
Part b ii {5 marks} - Architecture	Jamie
Part b iii {5 marks} - Method selection and planning	Sophia
Part b iv {5 marks} - Risk assessment and mitigation	Lucy
Implementation [23 marks]	Archie, Mitch
Part a (20 marks) - Provide Code	Archie, Mitch
Part b (3 marks) - List 3rd Party Libraries	Archie, Mitch
Testing [18 marks]	Zayed, Lucy
Part a (4 Marks) - Brief Summary	Zayed
Part b (10 Marks) - Test Report	Zayed
Part c (4 Marks) - Provide URLs for testing material	Lucy
User Evaluation [10 marks]	Jamie, Sophia
Part a (5 marks) - Report of the methods	Sophia
Part b (5 marks) - Table of Problems	Jamie
Continuous Integration [8 marks]	Sophia, Lucy
Part a (4 marks) - Summary	Sophia
Part b (4 marks) - Continuous integration infrastructure used	Lucy

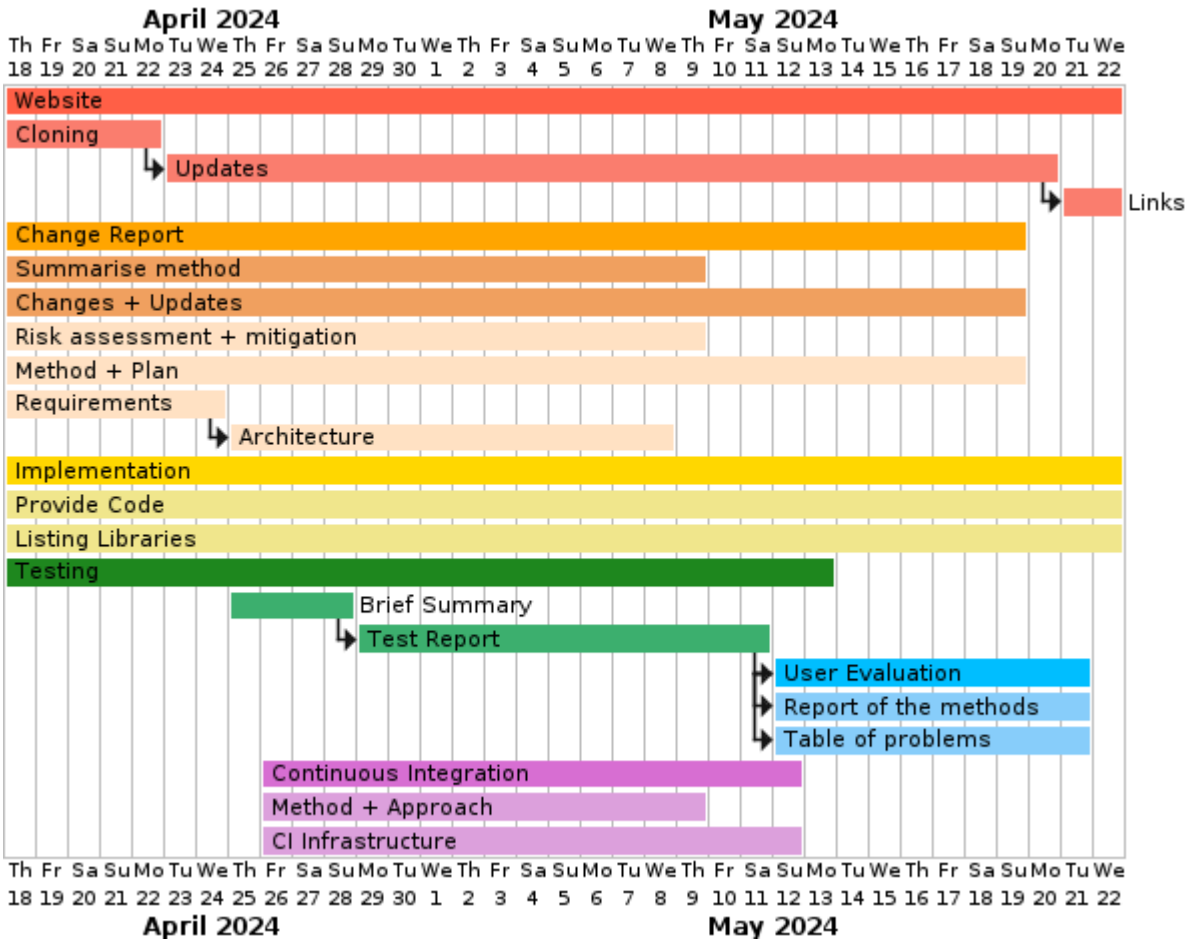
Once it was clear on what tasks we would be doing, it was time to schedule them for the second half of the assessment. The method used is the same to that prior in relation to making gantt charts. These gantt charts are made in PlantUML and act as a great way to ‘visually represent a project schedule’[2].

Once we had made the initial plan via gantt chart, we then met up at least once a week, as well as communicated over messages about our progress with our tasks. Once this had been completed, the Gantt chart was adjusted accordingly, whether that meant keeping it the same as the original, or allocating more time, or moving back a task to start.

As the project progressed, we found ourselves mainly following the first gantt chart made. The only changes that had to be made were pushing back the start of the continuous integration process/task. This was because we felt it was better to first become familiar with the code, and make any necessary adjustments in relation to its code structure before adding code to it in order to satisfy the new requirements.

As it turns out, to begin with, our personal deadlines were met earlier than originally anticipated for some of the sections. With some dependent tasks being able to be done earlier also, which was beneficial in order for us to have more time to proofread and make sure the quality of our work is the best that we can do. As the assessment went on, we became behind with some deadlines and had to adjust the timings in which we did things. But this was okay as the initial deadline for getting all of the work done was just under a week before the official deadline for the project and its documents to be handed in

This brings us to the final edition of the Gantt chart of the second assessment, where the plan for the project has stayed mostly the same. All in all, the first version of the gantt chart proved itself to be extremely useful in terms of time management and a visual representation of task delegation. Below is the final gantt chart created for this project.


April 2024
May 2024
Th Fr Sa Su Mo Tu We Th Fr Sa Su Mo Tu We Th Fr Sa Su Mo Tu We Th Fr Sa Su Mo Tu We Th Fr Sa Su Mo Tu We
18 19 20 21 22 23 24 25 26 27 28 29 30 1 2 3 4 5 6 7 8 9 10 11 12 13 14 15 16 17 18 19 20 21 22
Website
Cloning
Updates
Links
Change Report
Summarise method
Changes + Updates
Risk assessment + mitigation
Method + Plan
Requirements
Architecture
Implementation
Provide Code
Listing Libraries
Testing
Brief Summary
Test Report
User Evaluation
Report of the methods
Table of problems
Continuous Integration
Method + Approach
CI Infrastructure
Th Fr Sa Su Mo Tu We Th Fr Sa Su Mo Tu We Th Fr Sa Su Mo Tu We Th Fr Sa Su Mo Tu We Th Fr Sa Su Mo Tu We
18 19 20 21 22 23 24 25 26 27 28 29 30 1 2 3 4 5 6 7 8 9 10 11 12 13 14 15 16 17 18 19 20 21 22
April 2024
May 2024

Reference list

[1] A. Cockburn, Agile Software Development. Harlow, England: Addison Wesley, 2002.

[2] PlantUML Gantt Chart
https://plantuml.com/gantt-diagram