Method Selection and Planning

Cohort 1 Group 8

Roy Asiku, Tianqi Feng, Andrew Jenkins, Nicholas Lambert, Tom Byron

Software engineering methods: The software engineering method our group has decided to go forward with is the SCRUM agile engineering method. It's inclusion makes sense for our project as it emphasises iterative and incremental development, it allows us to focus on delivering a working software in short cycles that are called sprints.

Scrum:

- Setup
- Planning
- Implementation
- Review
- Deployment

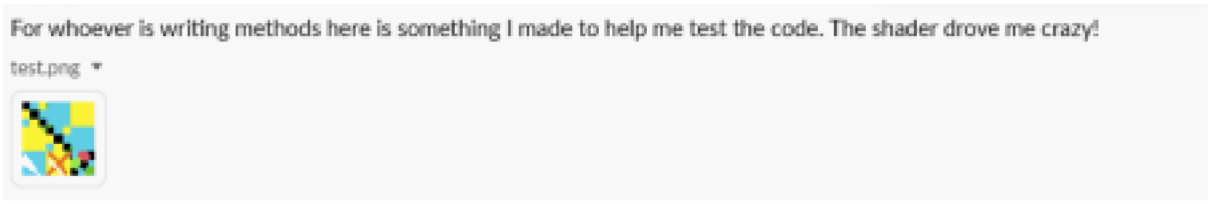
Justification:

- Quick response to feedback: Through continuous feedback loops (sprint reviews) the team is able to adapt features and mechanics based on in house testing
- Self-organising teams: Scrum allows each team to manage their work efficiently allowing each section to worked on without requiring the other to be done usually
- Communication: Sprint reviews which are meetings held weekly allows the communication between team members to be clear

Collaboration tools:

Slack:

- Real-time communication: Allows the team to quickly share information, and allow for quick discussions



For whoever is writing methods here is something I made to help me test the code. The shader drove me crazy!

test.png

Separate channels: By having the ability to add channels for each stage of the development process it allows for everyone to see the process of the team members and updates everyone on where we currently and allows meetings to be planned around it

Discord:

Group 8 have decided to switch from slack to discord for a few main reasons, the first being availability. All members of our team prior to this understood discord and possessed an account finding it much more usable for communication about this project. A second reason for this swap is that discord has easy to use features like screen sharing during calls *allowing us to easily review documents not yet shared or changes and code not yet committed. We found discord to be a useful tool to further our collaboration as a group.*

Google Drive:

- Cloud storage: it provides a central location to allow for storing and accessing project files
- Document sharing and collaboration: It allows for real-time collaborative editing of documents like design specifications, user stories and project plans

Git:

- Version control: It allows for tracking changes and enables us to revert to previous versions if needed and collaboration.
- Branching and merging: It enables parallel development and enables efficient integration of different features

To best take advantage of Git our team used Github. As it provides an easy to use user interface for introducing Git into our development. Moreover, Github Pages allows for free hosting of the required website for this project. This is important as it saves needless group expenses.

Justification for selected tools:

The tools we've chosen are all complemented with the scrum methodology: ● Slack: Supports the daily communication and collaboration needed during sprints. ● Git: Enables effective version controlling and collaboration on the code and is crucial for iterative development

- Google Drive: Allows a central location for files to be stored allows the management of project documents aligned with scrum practices

Alternatives Considered:

Project Management Tools: So while there are other tools like discord and jira. What slacks provides is the ability to have dedicated channels and threads which fosters focused communication and information organisation and allows threads to enable keeping discussion on specific topics within a channel preventing from information overload and maintaining a clear flow of conversation

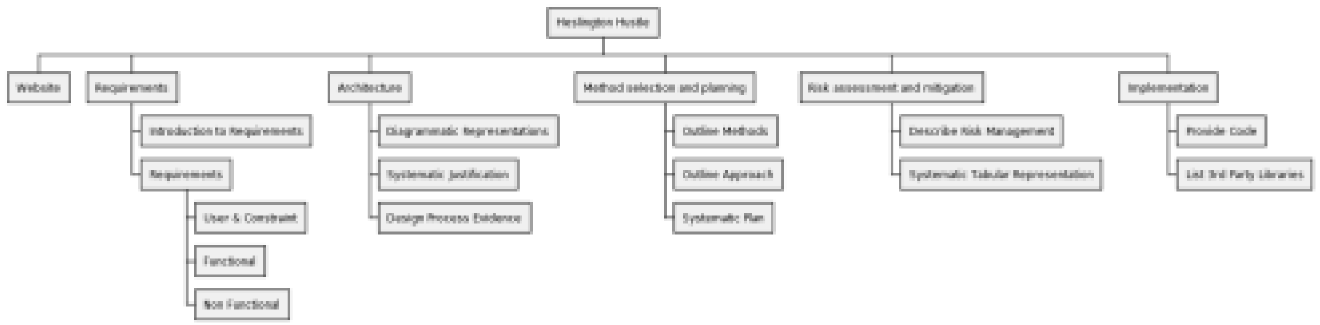
Team Organisation

At the start of the process, we decided to allocate roles to each team member. We felt this would be beneficial as it would allow each member to best utilise their strengths. These were the roles we chose:

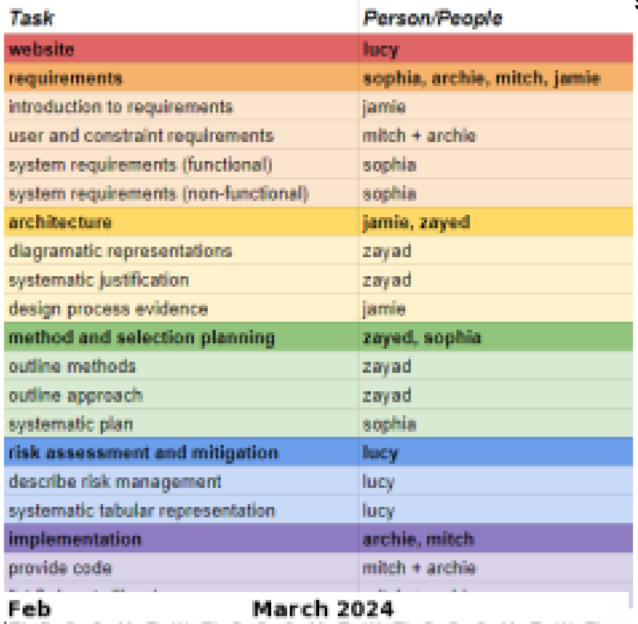
- Secretary
    - Lucy Wood
- Librarian
    - Jamie Creed
- Software Architect
    - Zayed Iqbal
- Scrum Master
    - Sophia Taylor
- Software Developer

- ○ Mitchell Gilbert and Archie Adams

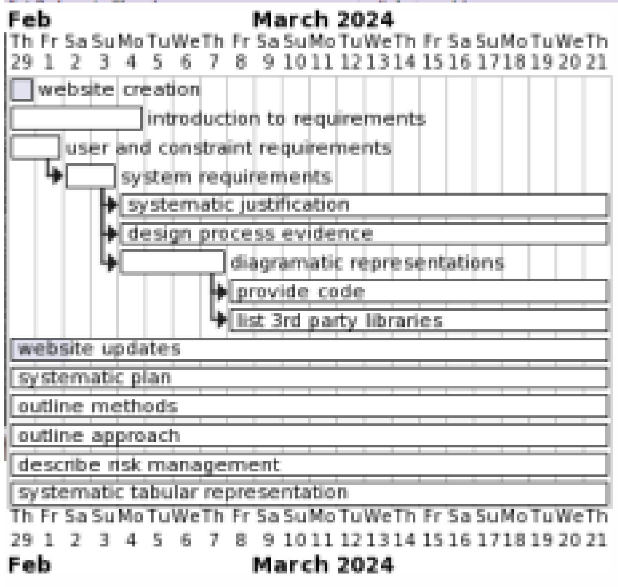
After the first few weeks, we felt like these roles weren't working for us as a team. We chose to move to more fluid roles where people collaborated on tasks. This was a much better fit for us, and we used logbooks for each meeting to allocate tasks to people. We felt the initial roles hindered the development of the project as they were too restrictive and forced people to only focus on one section of the project, rather than collaborating - this was a risk that we identified in the risk register (R1,R2).



Deciphering tasks: We first went through the entirety of the product brief, as well as the assessment and made a checklist of all tasks that needed to be done. From there, we were able to determine the main sections and tasks that make them up. Once this was done, a [illegible] s shown above.



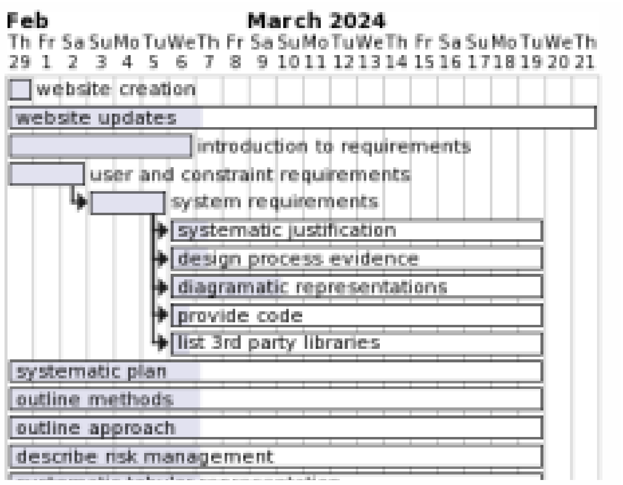
Task	*Person/People*
website	**lucy**
requirements	**sophia, archie, mitch, jamie**
introduction to requirements	jamie
user and constraint requirements	mitch + archie
system requirements (functional)	sophia
system requirements (non-functional)	sophia
architecture	**jamie, zayed**
diagramatic representations	zayad
systematic justification	zayad
design process evidence	jamie
method and selection planning	**zayed, sophia**
outline methods	zayad
outline approach	zayad
systematic plan	sophia
risk assessment and mitigation	**lucy**
describe risk management	lucy
systematic tabular representation	lucy
implementation	**archie, mitch**
provide code	mitch + archie
[illegible]	[illegible]



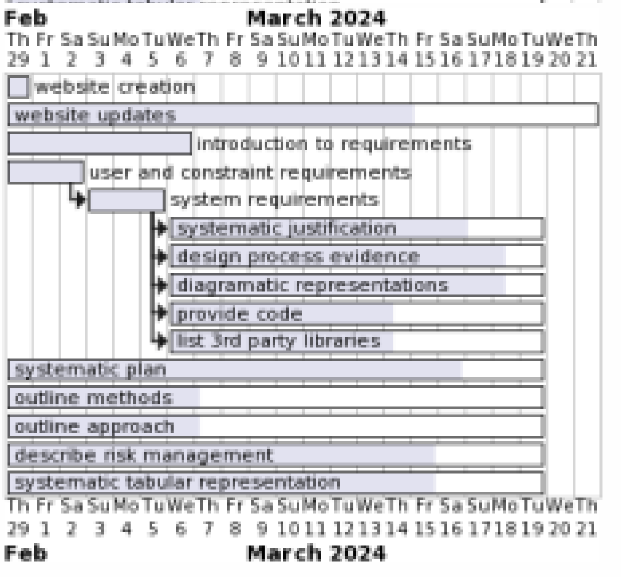
Allocation of tasks: After decomposing the assessment and deciphering the necessary tasks, we then talked between ourselves and allocated ourselves for the rolls we felt most comfortable with. On top of this we tried to make sure that the workload and its specified marks were evenly distributed between everyone.

Now that the tasks were defined and had been allocated to team members, it was time to plan out our time through gantt charts - which are a visual representation of the prioritisation and dependencies of these defined tasks.

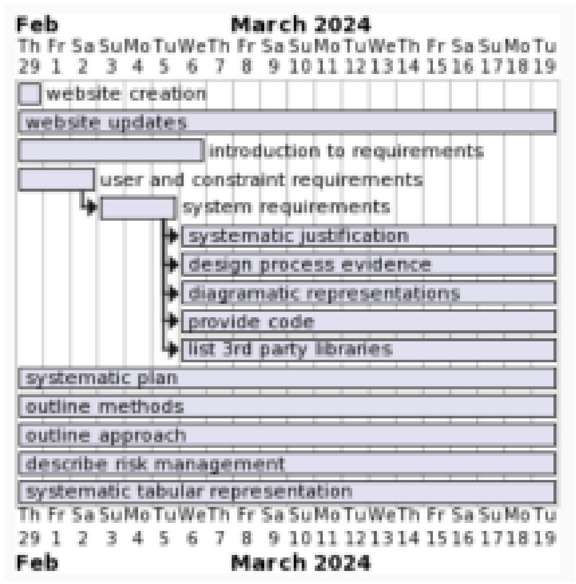
Week 1: During this first week we all sat down and looked through the breakdown diagram and allocation table. Using this paired with the assessment document, we were able to define what the dependencies were and the order in which it seemed best and most efficient to do things.



Week 2: This week we talked about our progress, and re-evaluated the dependencies of our allocated tasks. Through this, we decided that the diagrammatic representations would be better to be made alongside the code in case the code structure were to change, then those team members would talk amongst themselves to make sure they are accurate. The progress of the project as a whole is represented through the shading of the gantt chart.



Week 3: This week was simply a check in for everyone to see where we were at with our allocated tasks, and this progress was represented through the shading on the progress bar. In terms of the outlining methods and approach, now that we have developed further into the project, it should be written with more ease and efficiency. Overall everything looks like it will be finished on target ready to be proofread before submission.



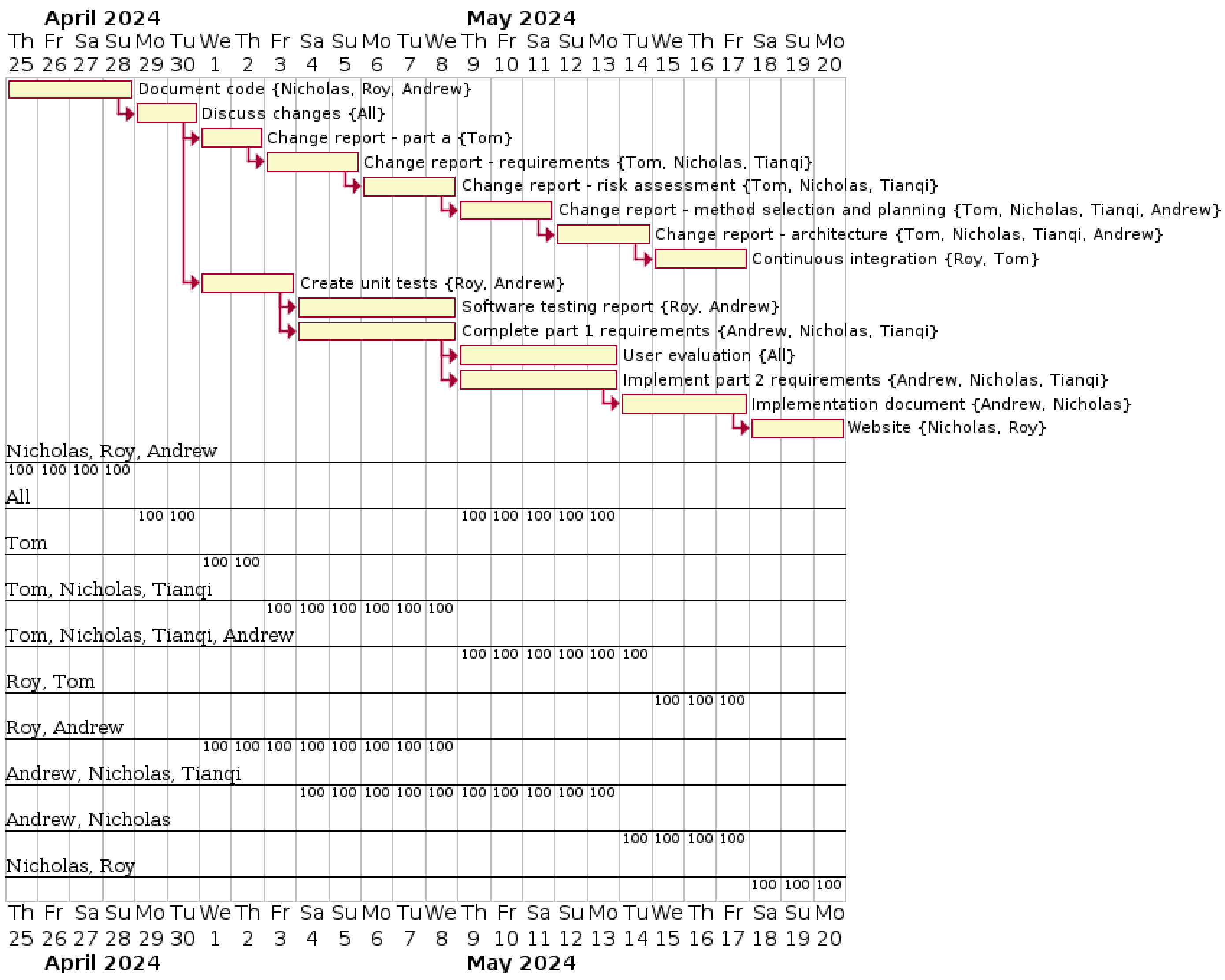
Week 4:. We got all the individual components ready for Tuesday so we could sit down and proofread and potentially submit. The deadline for website updates was also changed to the potential new submission date to keep in line with the rest of the work done.

These gantt charts were made every Thursday for the week except for the last which was made on the Tuesday before the submission deadline date. And the group's overall progress is represented by the steady shading of the separate tasks.

Team organization into the second half of the project

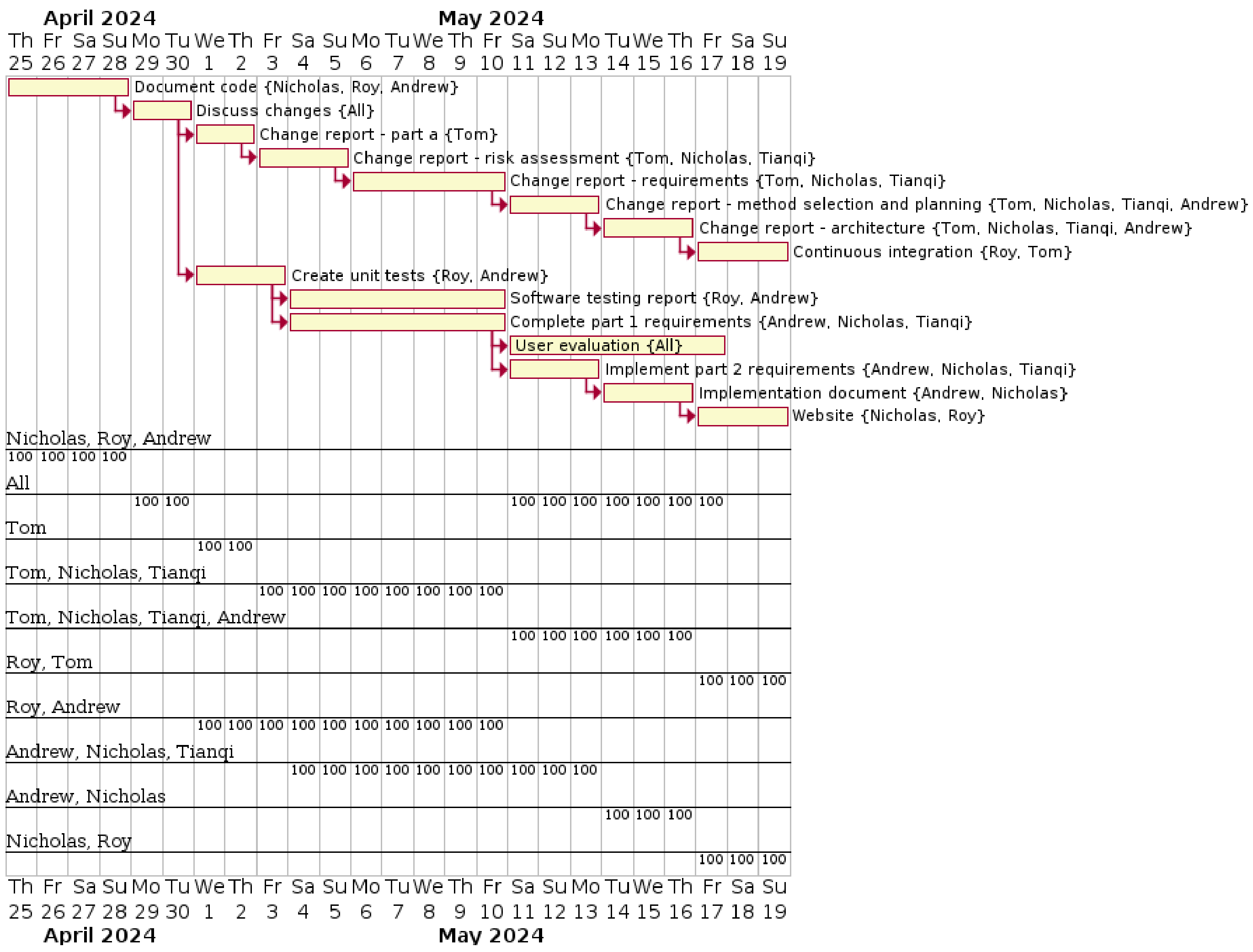
We decided to continue with the use of Gantt charts to schedule the phases of the development of our project, as being uml based it allows for easy creation and editing of a graphical representation of our projects schedule. It allows us to preempt what is coming within the project and prepare accordingly as a team. Furthermore, it allows us to plan out the sharing of our workload between team members across the development time of our project, and means we know what work we still need to complete - there are no surprises.

Plan 1 (25/04)
We felt that the most important thing to do first was to go over the code we inherited from the other team, and write documentation for it, as it had none. We had to ensure that we knew how the code functioned as we moved forward into implementation. Afterwards, we decided our workflow into 2 groups, implementation and documentation. We wanted to start off our implementation work by making unit tests and then follow on with completing the part 1 requirements not implemented by the other team. Then we would move onto part 2 requirements and user evaluation, so that we could implement any changes suggested by users. Simultaneously, we would begin work on updating the other teams' part 1 deliverables, writing up our progress in the change report. Then to finish off we would work on writing up our continuous integration and implementation deliverables, and then uploading all updated files to the website. We put our deadline 3 days ahead of the actual deadline to ensure that we had additional time if we needed it.



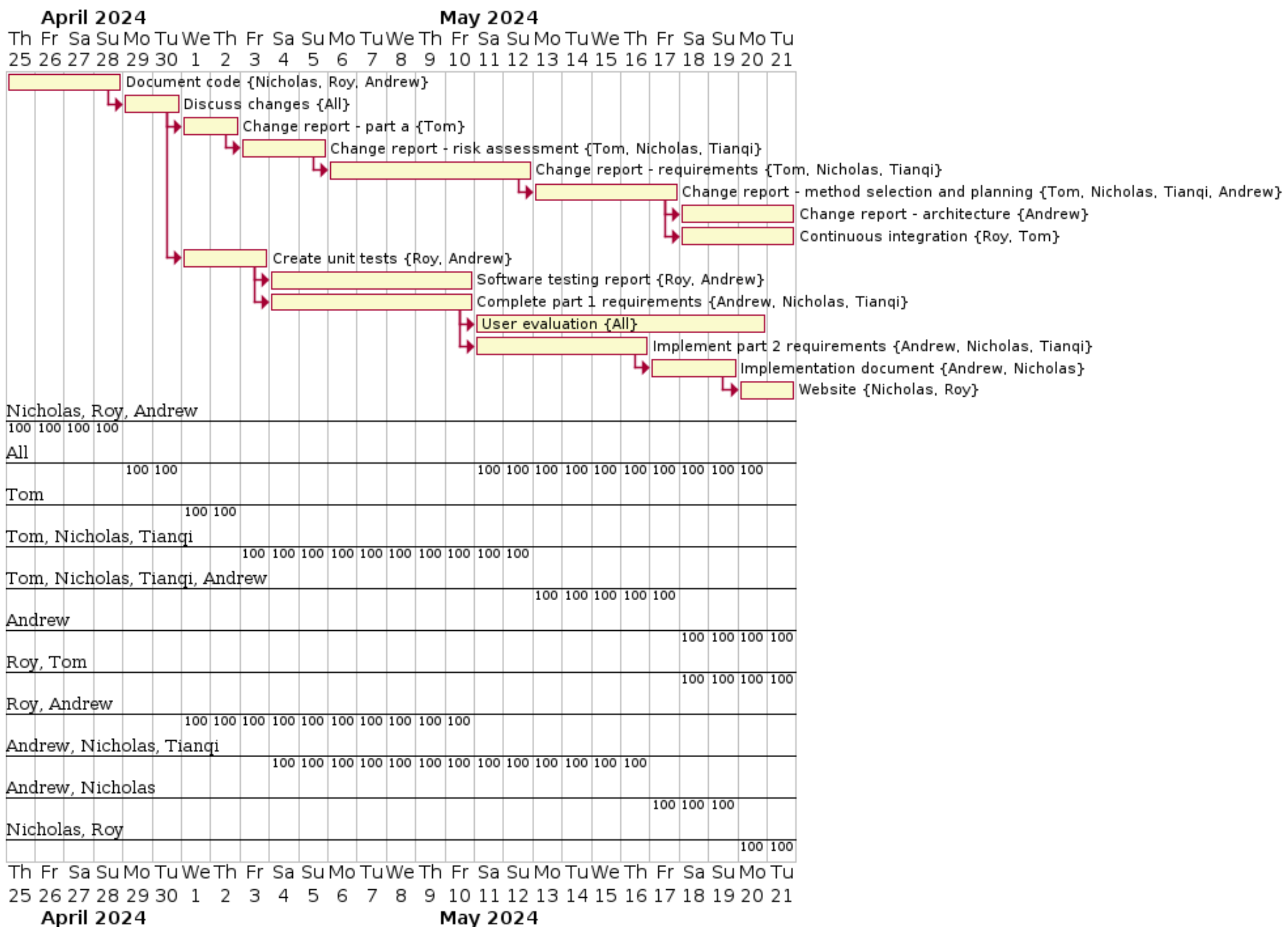
Plan 2 (09/05)

Upon working through the project, we had taken a bit longer than expected on our software testing due to some issues with getting it set up, and we also took longer than expected on part 1 implementation (due to us being busy with software testing), and on parts of the change report. Due to this, and due to re-evaluating how long other aspects of the project would take, we shortened our deadlines on future aspects of the project, such as part 2 implementation and the implementation write up. We had already given ourselves lots of extra time for the change report, so it was easy to just push that back a bit.



Plan 3 (14/05)

As the change report, user evaluation and implementation part 2 was taking longer than expected, mainly due to higher time demands of other uni work, we had to push back our deadline a bit to the 21st, extending times for the sections affected. To compensate, we shortened the time we had to work on the website, as we didn't think it would take that long, and we also decided to start working on the continuous integration write up earlier, simultaneously with the architecture write up, as we had allocated it to different people.


April 2024
May 2024
Document code {Nicholas, Roy, Andrew}
Discuss changes {All}
Change report - part a {Tom}
Change report - risk assessment {Tom, Nicholas, Tianqi}
Change report - requirements {Tom, Nicholas, Tianqi}
Change report - method selection and planning {Tom, Nicholas, Tianqi, Andrew}
Change report - architecture {Andrew}
Continuous integration {Roy, Tom}
Create unit tests {Roy, Andrew}
Software testing report {Roy, Andrew}
Complete part 1 requirements {Andrew, Nicholas, Tianqi}
User evaluation {All}
Implement part 2 requirements {Andrew, Nicholas, Tianqi}
Implementation document {Andrew, Nicholas}
Website {Nicholas, Roy}
Nicholas, Roy, Andrew
All
Tom
Tom, Nicholas, Tianqi
Tom, Nicholas, Tianqi, Andrew
Andrew
Roy, Tom
Roy, Andrew
Andrew, Nicholas, Tianqi
Andrew, Nicholas
Nicholas, Roy