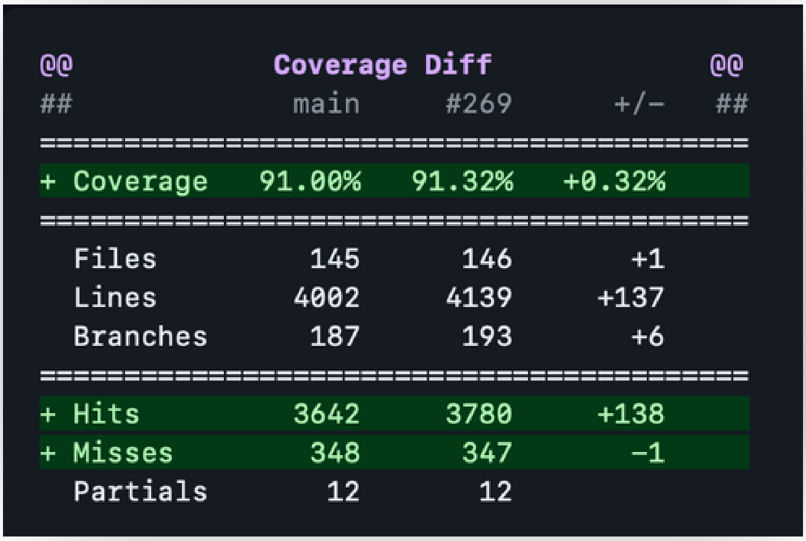
# Domain Pulse

Software Engineering Excellence

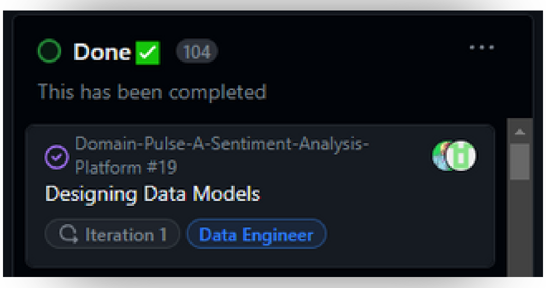
During the planning and development of Domain Pulse, strong software engineering principles and standards were put in place to ensure a smooth, consistent development and functional team environment, always in accordance with the Agile development methodology.

The first of these principles used was in regards to how code was developed within our system. Coding standards were followed rigidly, such as following a snake case naming convention for variables and methods in the backend, and within the frontend, a camel case naming convention was followed for angular code, and a kebab case naming convention for html and css. Furthermore all variables and methods had meaningful and descriptive names, using adequate prefixing and suffixing such as ‘is’ within ‘isActive’. As well as this our files and directories were structured and named so as to keep an organised and easily navigable codebase. Backend code was separated by service, with further subdirectories for each application within the service, and frontend development code was stored within an src folder, with an assets folder storing all needed assets, and an app folder with respective subdirectories for each component.

To accompany these coding standards, unit and integration tests were required to be created for code developed and committed, for the branch they were developed in to be merged to the development branch (as per our CI/CD pipeline and automated testing), ensuring that all code developed is tested and reviewed by the developer to the standard required, and such that no erroneous or incomplete code may appear within the development branch. Furthermore all changes made to our master branch require review from other developers within our team, to ensure correct standards and code is in place



```
@@             Coverage Diff           @@
##             main     #269     +/-   ##
==========================================
+ Coverage   91.00%   91.32%   +0.32%
==========================================
  Files         145      146       +1
  Lines        4002     4139     +137
  Branches      187      193       +6
==========================================
+ Hits         3642     3780     +138
+ Misses        348      347       -1
  Partials       12       12
```



Done ✅ 104

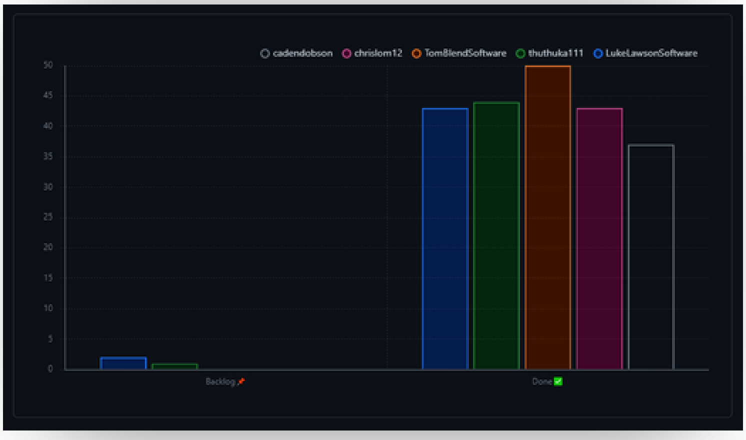
This has been completed

Domain-Pulse-A-Sentiment-Analysis-Platform #19

Designing Data Models

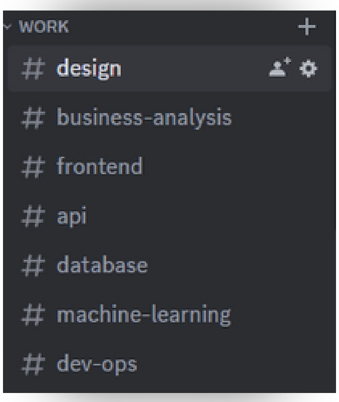
Iteration 1 Data Engineer

The next important principle followed by the team is in regards to how we planned and monitored the required tasks and development for each member. For the assignment of tasks, GitHubs Project's were used extensively, whereby descriptive tasks can be assigned to members and tracked through the section they are placed in such as 'in progress' or 'Done'. To help aid with the understanding of which members have done sufficient work we often used a visualisation of these tasks with a bar graph, showing the number of tasks categorised by status of the tasks, with coloured columns for each member.

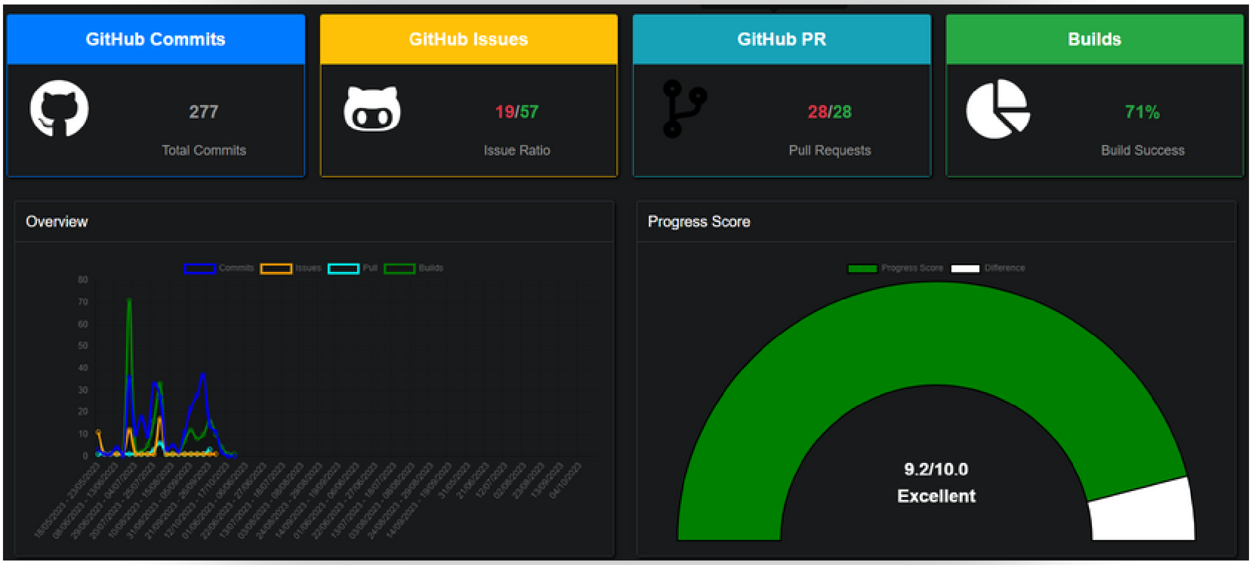


During the project, we had several communication channels, each with specific purposes that were used extensively, with easily understandable distinctions between the purpose of each channel. Our main channel of communication between team members was our private discord server, which had well defined text channels regarding each aspect of the project such as UI, API's, business analysis, etc...

For communication with our clients, we used a private slack channel, allowing for easy textual communication and meetings. Finally, due to sensitive information often being needed to be shared, such as API Keys, or environment variables, a private Signal group was created for secure sharing of this sensitive information.



To further assist in the planning of our project, we hosted weekly team meetings on our main communication channel, the Ctrl Alt Defeat Discord Server, discussing what work had been completed by each member, what work needs to be completed by the group, and which tasks will be completed or worked on by each member for the following week. This was accommodated by a kanban board and burndown chart, to help the tracking and completion of issues and tasks. Within our weekly meetings, we also utilised the university provided system, HyperPerform, to track the amount of commits, pull requests, and tasks completed for each member, so as to ensure a balanced workload for all members, that is being completed as needed. Furthermore, we often hosted sprint reviews with our clients so as provide validation and verification of what features we plan to develop, and how they intended to be developed, such that we are always in accordance to the clients needs



GitHub Commits
277
Total Commits

GitHub Issues
19/57
Issue Ratio

GitHub PR
28/28
Pull Requests

Builds
71%
Build Success

Overview

Progress Score

9.2/10.0
Excellent

Finally, in regards to the structure of our team, we followed the approach of an egoless, flat structure, whereby each member had fairly equal responsibilities within the project, however designated to specific roles such as UI Engineer, Services Engineer, etc… Although we had this flat team structure, our project manager was still present and necessary for the structure of our team, having broader knowledge of the state of the project, than other members with focuses on specific tasks. To accompany this flat structure, within our project we promoted and enforced a light-hearted but professional team culture, whereby all members would firstly and primarily discuss topics regarding the project itself, but often discuss other topics secondarily, allowing for a friendly and conducive working environment.