**Testing**

Testing was fundamental in ensuring that JustHealth was able to develop fully functioning web and mobile applications as well as a comprehensive back-end API. JustHealth adopted a test driven development approach and therefore, ensured that test cases were written at the start of every iteration for each piece of functionality that was scheduled to be implemented within that iteration. Following this, development would begin and each of the test cases was run once implementation was complete.

Initially, for the first two iterations JustHealth had been writing their test cases within a word document template that was designed at the start of the project APPENDIX. However, this became unmanageable for the team especially when it came to re-running tests that had failed. Moreover, the formatting of the document would change unexpectedly which would subsequently be time consuming to fix. In order to overcome this, the team looked at alternatives to this method, researching online tools and finding out what else was available. We found that these online tools were either made for a larger project or they were costly.

That said, the team decided to build a testing portal in python using a backend PostgreSQL database. The team used the testing portal to record, search and run tests, which worked very effectively. Details of how the testing portal was and can be used are detailed in the latest version of the JustHealth Test Plan.

JustHealth tested components implemented at each iteration using both functional and automated testing. The whole application was tested again before the project fair to ensure that no functionality had broken since last tested. The automated testing adopted the use of an automated unit-testing package available in python. These were just used to test the API and enabled the team to ensure that errors were being caught, some that would not be able to happen on the website or mobile application. The automated testing was also very useful as it was much quicker to run than the functional tests, which meant that they could be run more often to ensure that nothing had broken when branches were merged for example.