Quality Assurance
Document for AudioAppraiser web application

Review of user stories

Before the start of a sprint, there is a careful look at the user stories written down and are edited to be up to date in case any decisions during a previous sprint have changed the development process in the future. More complex user stories also have checklists on them to give a more detailed look at the progress of a story, so that one may see which core points of a user story are complete.

Commenting

Any methods and code snippets that are not easy to explain are commented and explained to be short and concise.

The basics rules of commenting are simple:

- Make them brief
- Keep them relevant
- Use them liberally, but not to excessive

Code Quality Assurance

All code in the application, both front and back-end are supplied with code quality plugins that ensure that the developer applies sufficient code quality during the coding process.

For the presentation layer code quality is ensured with ESLint. It is a JavaScript linting utility that checks through changed files while coding and sends error logs of any incorrect code syntax.

A specific instance concerning ESLint should be mentioned here - in multiple of the source files a line can be seen at the top configuring ESLint to ignore verification for react hooks. This is due to a known bug with ESLint that incorrectly throws an error for verification for a react hook as it considers it as an incorrectly made react state. As such, errors like these are ignored in any pages that use react hooks.

For the back-end, which is done in Spring Boot, I use CheckStyle. It is very similar to ESLint except that it sends a report through its own tab instead of errors.

Frontend testing

To ensure that the presentation layer of the application stays functional with every new implementation, there are unit tests created for each page using the website. The package @testinglibrary/react is used to write tests for the pages, from a simple render of a static page to feeding mock data for any frontend functions to become visualised correctly. The tests are run after every new push to the repository to ensure everything is functional before the upload.

Unit testing

Unit testing is done for every core function of the application, such as, but not limited to - fetching charts, searching for projects that contain a phrase as either the name or artist. Basic integration tests are done as well for the CRUD to feed tests to the git workflow to ensure everything is stable during pushing.