Testing Policy Document

Testing Policy Document

Ctrl Alt Defeat

Contents

1 Quality Requirements Testing

1.1 Usability

1.2 Security

1.3 Performance

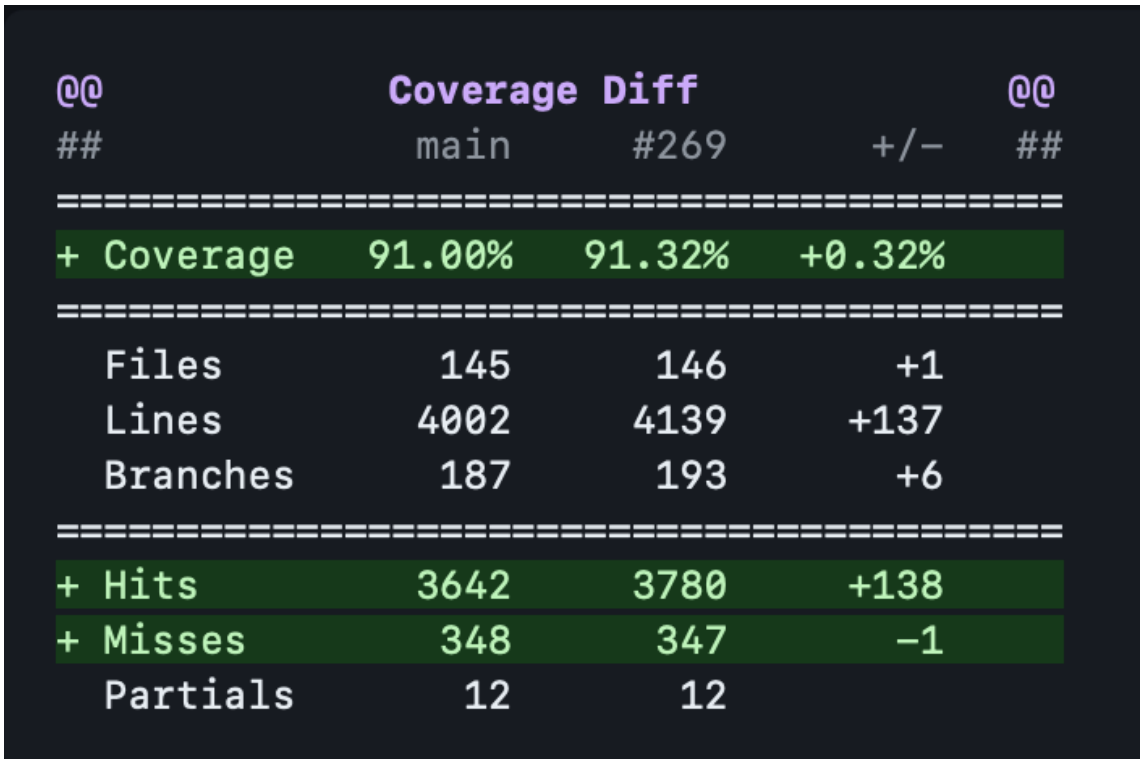
1.4 Scalability

1.5 Modifiability

2 Code Coverage

Any commit made to a branch causes automated tests to be run on the codebase of that branch, thereafter the code coverage of said branch is calculated. Any branch being merged via pull request into the development branch (dev) needs to have the coverage of changes to the codebase to match or better the coverage of the development branch. Furthermore, the coverage of the newly commited code (ie: patch) must match or exceed the coverage percentage of the project. This ensures that the coverage of the codebase is never decreasing, and that sufficient testing is being done on the codebase. Furthermore any time the development branch is merged into the master branch, the coverage of the development branch must match or be higher than that of the master branch, ensuring an increasing coverage and sufficient testing.

- A code coverage report is included in the below image:
- A link to our code cov profile can be found HERE.



@@	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		

3 Types of testing

- **Unit testing** - Unit testing

4 Choice of testing tools/frameworks

4.1 Frontend Testing

For our frontend testing frameworks and tools we decided to use the following:

- **Karma and Jasmine** - Jasmine is the testing framework that is used to write actual tests and are typed in Javascript, Karma is the test runner that executes the tests. Karma is run from a CLI(Command Line Interface) and it will open up a browser window and run the tests in that browser. Karma will then report the results of the tests back to the CLI and can be used to generate a coverage report. Karma and Jasmine are recommended by Angular which is what our frontend is primarily built upon and they are the most popular testing frameworks for Angular applications. The advantages of using Karma and Jasmine over other testing frameworks is that they are easy to set up and use, they are well documented and they are popular amongst Angular developers which means there is a extensive amount of resources available online for help if need be.

- **Cypress** - Cypress is a testing framework that is used to write end to end tests which are tests that try and simulate a user using the application. End to end tests are needed to ensure that the application is working exactly as expected from the user interface level all the way through the application to the database level and checks all the integration between these componets work as expected. Cypress runs in a browser which makes it easy to setup and follow the tests as they excute in the browser. Cypress is documented well with a thriving community which allows for easy access to information if any problems arise.

4.2 Backend Testing

For our backend testing framework and tools we decided to use the following:

- **Django built in testing module** - Django has a built in testing module which allows for the testing of Django applications, this is yet another reason why we decided to use django as it has amazing functionality out of the box. The django testing module allows for extensive testing of the application. The advantages of using django testing framework over an external framework is that one is already used to the syntax of django since our backend is primarily built on django and hence saves valuable time trying to learn syntax of another backend testing framework. Django allows for fast pace development which is much needed in certain situations such as in ours when following an agile development strategy. Django testing framework is also well documented and has a large community which allows for easy access to information if any problems arise and help is needed.

4.3 Usability Testing

Process of Usability Testing:

- **Planning** - The first step in usability testing is to plan what you want to test and how you are going to test it.
- **Recruiting** - The next step is to recruit participants to test the application. The participants should be representative of the target audience of the application but should also be diverse enough to get a wide range of feedback.
- **Testing** - The next step is the actual testing of the application, the way in which we facilitated the testing of the application was as follows:
  - We sent all participants of our usability testing a pdf explaining all aspects of navigating to the app and a list of tasks in which they could try. The pdf can be found HERE.
  - The participant will access the application in their own time in the environment of their liking and use the application to perform tasks that were suggested on the pdf.
  - Once the user has completed the testing of the app, a questionaire was then to be filled out by all participants the questionaire can be found HERE.
- **Feedback implementation** - We collect all data from the questionaire that the testing participants filled out and use this to modify and fix issues accordingly.

It is understood that the environment in which the users completed the testing in were all up to the users choice which allows for skewed results in terms of the environment variable not being kept constant and things like noise pollution etc can all have effects on how one would use the app but we belive that this bias of an inconstant environment is neutralised by the adaptability of the app to be used in any environment.

4.4 Github Actions and workflows

Github actions were used to aid in the automatic testing and generating of code coverage reports for our application. Github actions are workflows that are activated by certain events such as a merge, push or pull to certain branches of the repository. The workflows are defined in a yaml file and are run on a virtual machine hosted by github, Github actions ensure that all tests pass before a branch is merged into main/master. The workflows that we have defined are as follows:

- backend build
- frontend build
- coverage report
- automatic deployment
- automatic testing