Test Plan

Proof Checker Tool - SE691

Rakhfa Amin, Thoman Andrews, Kersley Jatto, Colton Shoenberger

Server-Side Test Plan

We are utilizing the Django framework for our project, which comes with a robust suite of built-in testing tools (https://docs.djangoproject.com/en/3.2/topics/testing/). Any filename that matches pattern `test*.py` is recognized by Django as a test file. Django `TestCase`'s subclass the Python built-in `unittest` module, which is capable of performing both unit and integration testing. Any function in a test file that matches the pattern `test*` is recognized by Django as a test method, and is run when the command `python manage.py test` is run in the terminal. Django test methods support a wide variety of assertions (`assertTrue`, `assertEqual`, etc.) that can be utilized to validate that the system is behaving as expected.

We are complementing our use of Django tests with the Coverage.py tool, which provides code coverage reports when our tests are run. This allows us to identify exactly what lines of code are being executed (and perhaps more importantly, which lines are *not* being executed) when our tests are performed. It also provides us with percentages of lines executed in individual files and the overall system. On the next page, you will find a recent report of our overall code coverage. Our test suite currently includes 99 test cases with over 300 assertions. We are proud to report our code coverage is currently at 95% of server-side lines of code.

Code Coverage Report

Name	Stmts	Miss	Cover
accounts__init__.py	0	0	100%
accounts\admin.py	0	0	100%
accounts\apps.py	4	0	100%
accounts\forms.py	29	0	100%
accounts\migrations__init__.py	0	0	100%
accounts\models.py	0	0	100%
accounts\tests__init__.py	0	0	100%
accounts\tests\test_forms.py	12	0	100%
accounts\tests\test_views.py	21	0	100%
accounts\urls.py	3	0	100%
accounts\views.py	28	0	100%
proofchecker__init__.py	0	0	100%
proofchecker\admin.py	14	0	100%
proofchecker\apps.py	4	0	100%
proofchecker\forms.py	30	0	100%
proofchecker\migrations\0001_initial.py	11	0	100%
proofchecker\migrations__init__.py	0	0	100%
proofchecker\models.py	70	5	93%
proofchecker\proof.py	718	91	87%
proofchecker\tests__init__.py	0	0	100%
proofchecker\tests\test_models.py	37	0	100%
proofchecker\tests\test_proof.py	856	0	100%
proofchecker\tests\test_utils.py	175	2	99%
proofchecker\tests\test_views.py	119	0	100%
proofchecker\urls.py	5	0	100%
proofchecker\utils\binarytree.py	104	8	92%
proofchecker\utils\constants.py	8	0	100%
proofchecker\utils\numlex.py	16	1	94%
proofchecker\utils\numparse.py	9	0	100%
proofchecker\utils\syntax.py	96	5	95%
proofchecker\utils\tfllex.py	23	1	96%
proofchecker\utils\tflparse.py	35	0	100%
proofchecker\views.py	133	4	97%
prooftool__init__.py	0	0	100%
prooftool\settings.py	27	0	100%
prooftool\urls.py	5	0	100%
TOTAL	**2592**	**117**	**95%**

Client-Side Test Plan

While Python accounts for over 75% of the code in our system, JavaScript accounts for about 16.5% (according to GitHub). While we have been utilizing some rudimentary "DIY" testing measures, there is a clear need for us to integrate an automated testing suite with code coverage reporting for our client-side code (especially since we hope to expand our client-side proof validation capabilities). This will be an early objective as we enter next term. We are currently exploring several options. Django actually has built-in support for the QUnit testing framework, which we are researching further (https://qunitjs.com/).