Dungeons & Dragons Game State Manager

Testing Policy Document
Developed for Retro Rabbit by Optimize Prime

Team:
James Hertzog
Thomas Scholtz
Jason Davidson
Ruben Denner

1. Testing Process

Definition of Testing:

Testing is done to effectively and efficiently provide timely, accurate, and useful information of the current state of the application. This is to ensure the software fulfills all its requirements while remaining in a healthy condition (bug free, etc).

Description of the test process:

Unit tests:

Goals:

- Detect defective code in units
- Reduce risk of unit failure in Production

Unit tests must comply with all coding standards.

Integration:

Goals:

- Detect defects in unit interfaces
- Reduce risk of dataflow and workflow failures in Production

After every major feature commit a test should be written to validate that the feature is working and giving the expected result. Theses tests must be added to automatically run on our continuous integration service (Travis CI).

Quality Level to be achieved:

Tests should result in no outstanding high severity faults prior to version releases. All server side tests must succeed for server commits to be deemed ready for merging into develop. Current running instances of the development server can only be updated if all unit tests pass.

Approach to Test Process Improvement

Tests should be reviewed alongside code reviews after a sprint. Tests should be on the same quality standard as the production code released to end-users.

2. Testing tools

We use Travis CI for automated testing of the front end and the back end. Since the front end and backend are in different GitHub repositories, testing on Travis is also done separate. Each repository has a .travis.yml file that tells travis what to install and what commands to run to execute the tests.

We chose Travis CI, because it executes tests each time new code gets pushed to GitHub or when merging occurs. This means that when developers do not run tests locally before pushing or merging, the tests will run online. Travis testing results are visible for all team members which helps the entire team know the current situation of the project.

Taken from https://travis-ci.org/COS301-OptimizePrime/COS301-DnD (Front end)

```
The command "./flutter/bin/flutter -v build apk" exited with 0.
$ ./flutter/bin/flutter test
00:10 +6: - Character Preview Test
Start: 400.0
End: 400.0
00:10 +10: All tests passed!


The command "./flutter/bin/flutter test" exited with 0.

Done. Your build exited with 0.
```

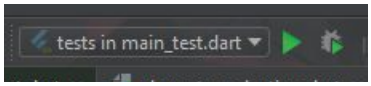
Taken from https://travis-ci.org/COS301-OptimizePrime/COS301_DND_Backend

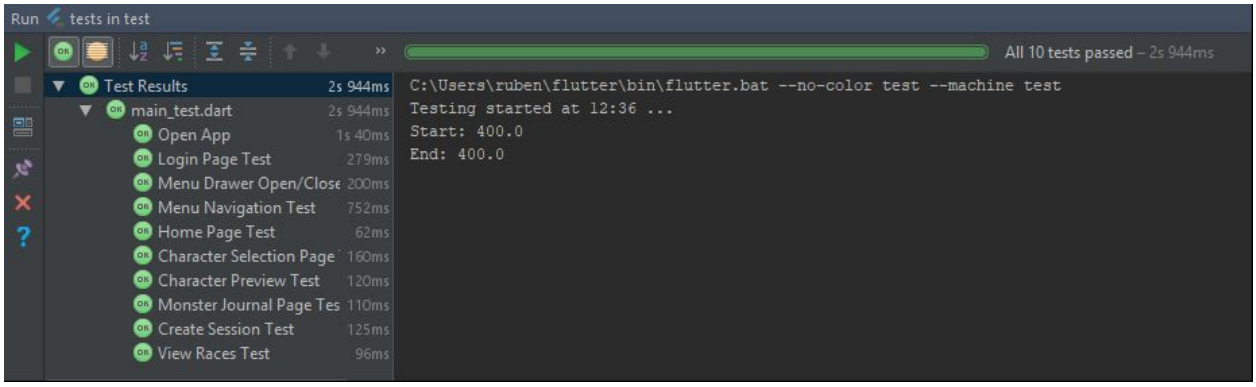
```
========================== 7 passed in 3.27 seconds ==========================
===Runing dart tests!===
00:00 +0: test_create_rpc_good_login
2018-07-19 19:42:45,047 - DEBUG - session - ipv4:127.0.0.1:54740
2018-07-19 19:42:45,047 - INFO - session - Create new session called! Name:mySession
Client received: SUCCESS
00:00 +1: All tests passed!


The command "make test" exited with 0.

Done. Your build exited with 0.
/home/travis/.travis/job_stages: line 177:  7059 Terminated              make run
```

Front end:

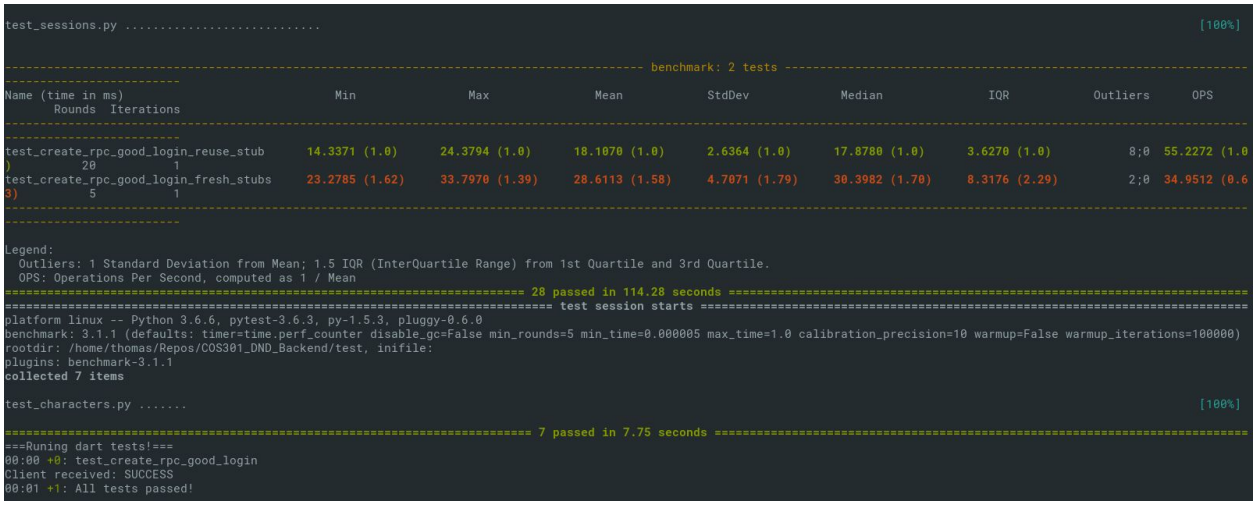
The front end is developed in Android studio that provides functionality to easily run all written tests located in a folder tests. All tests are written in Dart and Android studio calls Flutter to execute the tests. We write the tests in Dart, because the source code for the project is written in Dart.

Taken from Android Studio:





Back end:



3. Test cases

Front end:

Test cases are located in the test folder of the repository.
(https://github.com/COS301-OptimizePrime/COS301-DnD/tree/master/test)

These tests include:

1) Login Page Test
    a) Opens the Login Page to see if it loads correctly.
2) Menu Drawer Open/Close Test:
    a) Tests if the menu drawer opens and closes correctly.
3) Menu Navigation Test:
    a) Test the menu navigation functionality by tapping on all the icons in the drawer and testing if the app correctly navigates to all pages.
4) Home Page Test:

    a) Tests if the Home Page loads correctly
5) Light Character Test:
    a) Creates a Light Character and tests its functionality
6) Character Light View Test:
    a) Creates a CharacterLightView widget and tests if it is loaded correctly
7) Character Item Test:
    a) Creates a ChacterItem widget and tests if it is loaded correctly
8) Character Selection Page Test:
    a) Tests if the Character Selection Page loads correctly.
    b) Tests tap on Character functionality that shows more details about the Character.
9) Gender Icon Test:
    a) Creates a GenderIcon widget and tests if it is loaded correctly
10) Race Preview Test:
    a) Creates a RacePreview widget and tests if it is loaded correctly
11) Class Preview Test:
    a) Creates a ClassPreview widget and tests if it is loaded correctly
12) Class Icon Test:
    a) Creates a ClassIcon widget and tests if it is loaded correctly
13) Stat Icons Test:
    a) Creates all the Stat widgets and tests if they are loaded correctly
14) Character Details Page Test:
    a) Tests if the Character Details Page loads correctly.
    b) Checks if character information shown is correct.
15) Character Creation Page Test:
    a) Tests navigation to character creation.
    b) Tests if page loads correctly.
    c) Creates a character and tests functions of page.
16) Character Preview Test:
    a) Tests if a preview of the Character pops up when a swipe is made on a Character and if preview is correct.
17) Monster Journal Page Test:
    a) Tests if the Monster Journal Page loads correctly.
    b) Tests tap on Monster functionality that shows more details about the Monster.
18) Create Session Test:
    a) Tests functionality when a user taps on the ”Create Session” button.
19) View Races Page Test:
    a) Tests if the View Races Page loads correctly.
    b) Tests tap on Race functionality that shows more details about the Race.

Backend:

Test cases are located in the test folder of the repository:
(https://github.com/COS301-OptimizePrime/COS301_DND_Backend/tree/master/test)

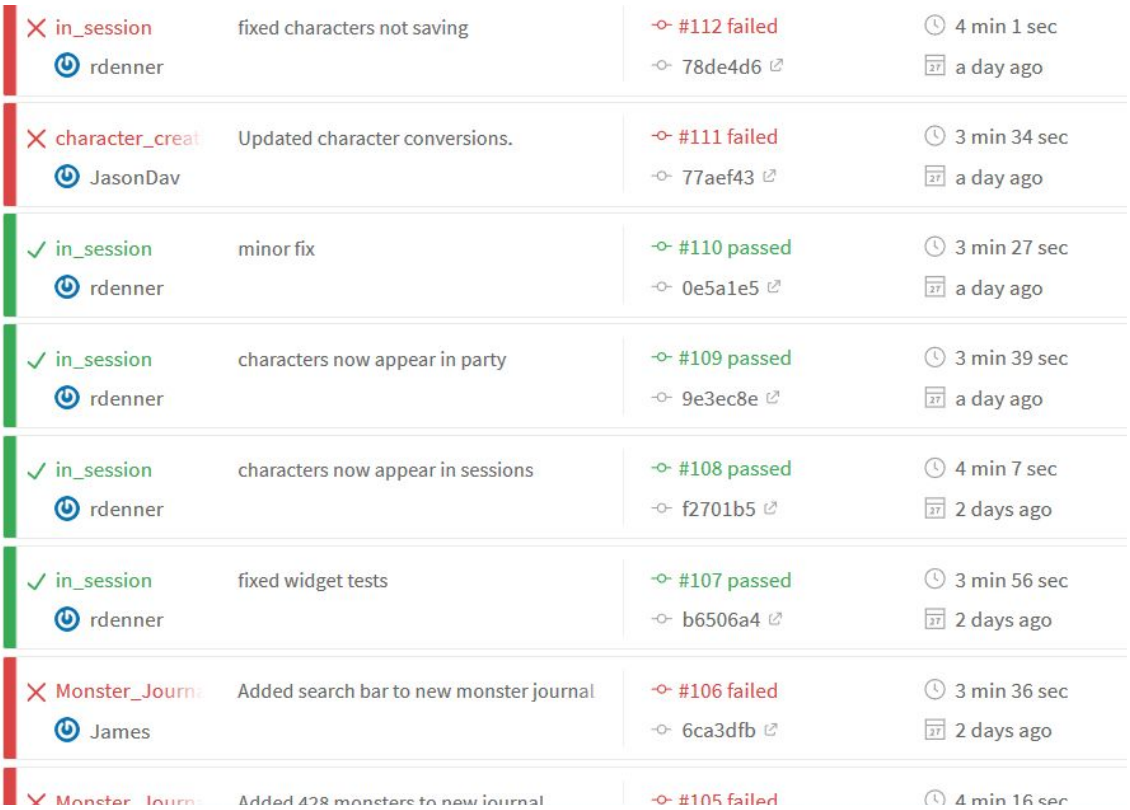
The backend currently has:

- 2 Benchmark tests (Speed at which characters and sessions can be created)
- 28 session related tests (Create, Delete, Join, Leave, Bad Join, Bad Create, etc)
- 7 character related tests (Create, Delete, Update, etc) In addition these tests do full object comparisons to make sure that nothing has changed erroneously.

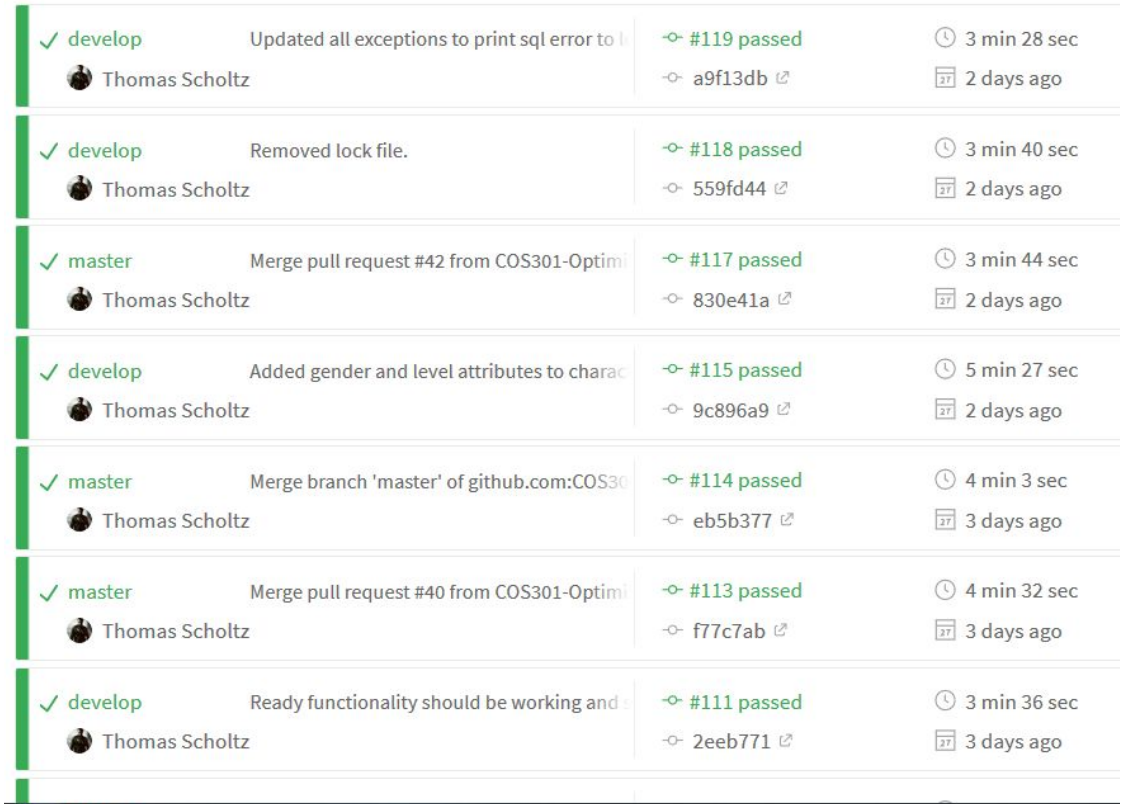
Most backend tests are full integration tests. The backend emulates a small client which calls all the real outward facing functions and compares the results.

4. History

The Travis builds for front end test can be found at https://travis-ci.org/COS301-OptimizePrime/COS301-DnD/builds.



For backend Travis builds go to https://travis-ci.org/COS301-OptimizePrime/COS301_DND_Backend/builds.


develop
Updated all exceptions to print sql error to
Thomas Scholtz
#119 passed
a9f13db
3 min 28 sec
2 days ago
develop
Removed lock file.
Thomas Scholtz
#118 passed
559fd44
3 min 40 sec
2 days ago
master
Merge pull request #42 from COS301-Optimi
Thomas Scholtz
#117 passed
830e41a
3 min 44 sec
2 days ago
develop
Added gender and level attributes to charac
Thomas Scholtz
#115 passed
9c896a9
5 min 27 sec
2 days ago
master
Merge branch 'master' of github.com:COS30
Thomas Scholtz
#114 passed
eb5b377
4 min 3 sec
3 days ago
master
Merge pull request #40 from COS301-Optimi
Thomas Scholtz
#113 passed
f77c7ab
4 min 32 sec
3 days ago
develop
Ready functionality should be working and
Thomas Scholtz
#111 passed
2eeb771
3 min 36 sec
3 days ago