**CS673 Software Engineering** 

**Team 1 - Knowitall**

**Tests Report**

| Team Member | Role(s) | Signature | Date |
| --- | --- | --- | --- |
| Daniel Makover | Team Leader | *Daniel Makover* | 12/09/21 |
| Kun Mo | Team Leader | *Kun Mo* | 12/08/21 |
| Gunnar Nichols | QA Leader | *Gunnar Nichols* | 12/05/21 |
| Heli Kolambekar | Requirements Leader | *Heli Kolambekar* | 12/09/21 |
| Haoyi Zhu | Design and Implementation Leader | *Haoyi Zhu* | 12/08/21 |
| Weiye Xu | Configuration Leader | *Weiye Xu* | 12/08/21 |
| Yuwei Wu | Security Leader | *Yuwei Wu* | 12/09/21 |

**Revision history**

| **Version** | **Author** | **Date** | **Change** |
| --- | --- | --- | --- |
| **1.0** | **Gunnar Nichols** | **11/9/21** | Drafted |
| **1.1** | **Gunnar Nichols** | **12/5/21** | Updated to reflect new tests and other changes |

[Introduction](#_87t9hln2vjz0)

[Test Summary](#_sm5odwyvuk3j)

[Tests Reports](#_pqso2mbjyzx4)

[Testing Metrics](#_mtfbusfb0eq3)

[References](#_15tmymhipvdv)

[Glossary](#_8n34lvocupub)

# Introduction

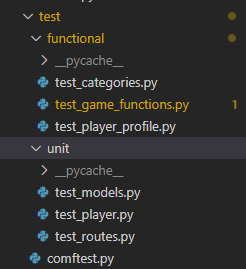
The automated tests for knowitall are based on the pytest-flask plugin, which is an expansion of the capabilities of pytest to be better suited for testing a flask application. We also utilize selenium and manual tests to test game features which were incompatible with pytest due to design decisions.

# Test Summary

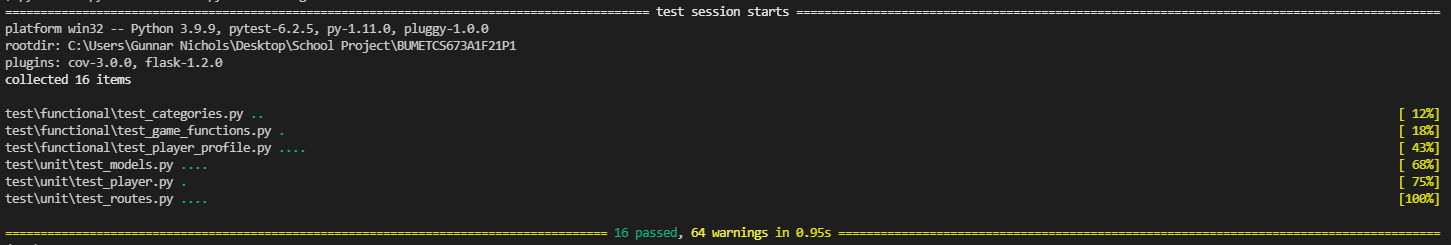
Testing for the knowitall application is accomplished via three distinct methods. Manual testing for features which cannot be automated and do not have the html structure required to properly implement selenium tests, feature testing with selenium for features which require the browsers memory to properly utilize the game as it is programmed, and automated feature testing with pytest for everything else

Scripts for both types of feature testing can be found in the tests folder located in the root directory of our application. When the command ‘$python -m pytest’ is run from the root directory, all of the test functions (beginning with test\_\*) in any of the test files (beginning with test\_\*) will be executed and the summary of the results are provided in the terminal.

The tests are grouped by type within the folder describing the testing activities. Currently functional tests can be found in the ‘functional’ folder, while unit tests can be found in the ‘unit’ folder

Test Directory:

Pytest output



[Warnings relate to deprecated/ soon to be deprecated functions]

# Automated Tests Reports

The full test case table can be found here: Automatic [Test Cases](https://docs.google.com/spreadsheets/d/15znXkVjUvw8YXIBSycquCkoALb_QzPvO2Z4BOEkRmY4/edit#gid=0)

Here are the first few tests/columns

There are a total of 20 automated unit/feature tests

5 of these are part of the automated selenium test

| Test case name | Models (4) | Home Page | Category Page |
| --- | --- | --- | --- |
| New or old: | Old | Old | Old |
| Test Type | Automated Pytest | Automated Pytest | Automated Pytest |
| Test items: | Game, Player, LeaderboardScore and Question Models | Home Page | Category Page |
| Test priority: | medium | Medium | Medium |
| Dependencies: | N/A | App Fixture | App Fixture |
| Preconditions: | N/A | N/A | N/A |
| input data: | Object Model Values | home route | home route |
| Test steps: | Assign Values to Object, make sure they hold | Import app fixture  Navigate to home url  Make sure response has expected content | Import app fixture  Navigate to home url  Make sure response has expected content |
| Postconditions: | N/A | N/A | N/A |
| Expected output: | All asserts true | All asserts true | All asserts true |
| Actual output: | All asserts true | All asserts true | All asserts true |
| Pass or Fail: | Pass | Pass | Pass |
| Bug id/link: | N/A | N/A | N/A |
| Additional notes: | Each model has a test |  |  |

# Manual Test Reports

* + Reset Password: [Manual Testing: “Forgot Password” Feature.docx](https://docs.google.com/document/d/1dE-ePR8s1GtMOcLywFDWU_AaJZi9KBSW/edit?usp=sharing&ouid=114129658996977914799&rtpof=true&sd=true)
  + Reset Questions/Submit Score/Admin Features

[Manual Testing: Question Cycling, Score Posting and Admin .docx](https://docs.google.com/document/d/1jxf7-CoeIL28-etSvhU4EhptTsai_J1_/edit)

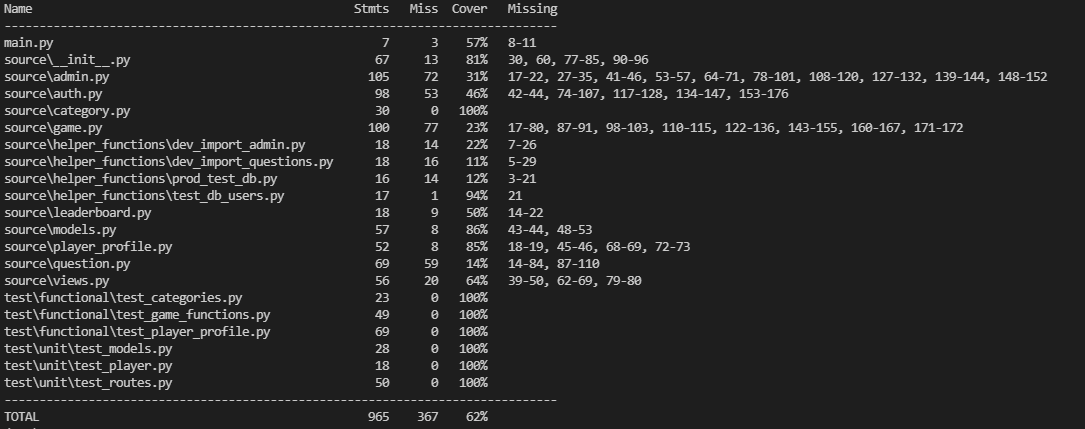
# Testing Metrics

*Metric: What it is. How it is tracked. How it is analyzed and used improve or understand the project*

* + - 1. *Tests Written:* Total number of tests written broken out by Automated and Manual. This is a measurement of how many tests have been written
      2. *Tests Passed:* Test Passed / Tests Written. This is a measurement of how well the current code passes the written tests.
      3. *Pytest Code Coverage*: (Statements Run/ Statements in Code). This is a measurement of how much of the code is covered by all of the tests that impact the coverage report in pytest.
      4. *Selenium Code Coverage*: (Statements Run/ Statements in Code). This is a measurement of how much of the code is covered by all of the tests that are part of the Selenium test which are not covered in the automated Pytest tests.
      5. *Manual Code Coverage*: This is a measurement of how much of the code is covered by Manual tests which are not covered by automated Pytest or Selenium tests.
      6. T*otal Code Coverage*: This is a measurement of how much of the code is covered by any formalized test.

Tests Written: 20 Automated, 4 Manual

Tests Passed: 100%

Pytest Code Coverage: 62%

Selenium Code Coverage: 11%

The Selenium tests cover code present in the game.py file and question.py file.

In the question.py file about half of the statements missed by the automated test are covered 30/965 = 3%

In the game.py file all of the missed statements are covered 77/965 = 8%

Manual Code Coverage: 13%

The Manual tests cover code present in the leaderboard.py and question.py files and covers all of the remaining statements in these files

(9 + 29) / 965 = 4%

The manual tests also cover code in the auth.py and admin.py, covering about 70% of the statements remaining in those files

(0.7 \* 125) / 965 = 9%

Total Code Coverage = 62% + 11% + 13% = 86%

# Disconnects from Production

The testing code as written is only compatible with the development environment for Knowitall. Not-written manual tests were done by Danial Makover to ensure that the production environment is a reasonably faithful representation of the logic automatically tested in the development environment, but the production stack utilized postgreSQL, while the development stack utilizes sql-lite making specific tests incompatible.

# References

Pytest Documentation: https://docs.pytest.org/en/6.2.x/contents.html

Pytest-flask Documentation: <https://pytest-flask.readthedocs.io/en/latest/>

Pytest-flask tutorial: https://testdriven.io/blog/flask-pytest/

# Glossary