

Google Summer of Code 2022

Achieve 100% Per-File Branch and Line Coverage for the Frontend and the Backend

Anshuman Maurya

Section 1: About You

What project are you applying for?

Achieve 100% Per-File Branch and Line Coverage for the Frontend and the Backend.

Why are you interested in working with Oppia, and on your chosen project?

I learned about Oppia from one of my college seniors who volunteered at Oppia's codebase. When I gave it some time, I understood the problem Oppia is trying to solve. It made me realize how lucky and privileged I am to attain a quality education and teaching. The best I could do now is to give back to society and help others who lack the resources.

More than 60 million children worldwide lack the resources to attain primary education. I belong to India, where more than 30 million children aged 6-14 cannot attend school, and almost half of the girls of this age are illiterate. About half the habitation lacks primary schooling facilities [\[src\]](#). If we cannot bring these children to school, we can take schools to them in the form of online platforms like Oppia. This could significantly impact the world, and I would consider myself fortunate if I am a small part of it. At Oppia, I am happy to be part of a community that shares an equal enthusiasm and motivation towards the goal we are trying to achieve. Every bug I fix and every feature I add seems to be a small step towards providing quality education to all, which keeps me motivated.

Oppia has a large codebase with Angular JS/Angular 2+ at its frontend and Python at its backend. It is crucial to test the code to avoid any bug or error popping up in production. That's why I chose to work on the project that aims to achieve 100% Per-File Branch and Line Coverage for the Frontend and the Backend. A fully covered codebase would significantly reduce the chances of any regressions. I already have experience in testing, and I have covered 25+ files in the backend and frontend of Oppia's codebase. The project seems to be an excellent opportunity to make myself proficient in testing code.

Prior experience

I started my development journey in the past year and have learned quite a few technologies since then. I am familiar with Typescript and Python and various frontend and backend frameworks like Vue, Angular, and Django.

I have been contributing to Oppia for the past 3 to 4 months and merged about 20+ PRs gaining familiarity with the codebase. I am a part of the Automated QA team and the Learner and Creator Experience team. I am currently working on the User checkpoints project under the mentorship of Sean Lip. The project aims to achieve exploration checkpointing on the learner's side so that no learner may lose his progress if he has to quit an exploration in between. I was responsible for the entire backend changes to be made regarding the storage of checkpoint progress of a logged-in user.

My submitted PRs include:

1. [#15213](#) - User checkpoints: Backend changes for logged in users
2. [#14876](#) - Strip invalid tags and attributes from SVGs instead of showing a warning
3. [#14641](#) - Increased backend test coverage of some files to 100%
4. [#14729](#) - Increased frontend test coverage of some files to 100%
5. [#14506](#) - Changed deprecated python methods

The complete list of my merged PRs to Oppia can be found [here](#).

Along with this, I also worked on many group and personal projects which can be viewed on my [GitHub](#) profile.

Project size

The project size is medium(~175 hours).

Project timeframe

June 13 - September 12

Contact info and timezone(s)

Email: anshumanmaurya111@gmail.com

Phone no: (+91) 8318904105

Preferred mode of communication: Google chat, Gmail, Gmeet, Discord

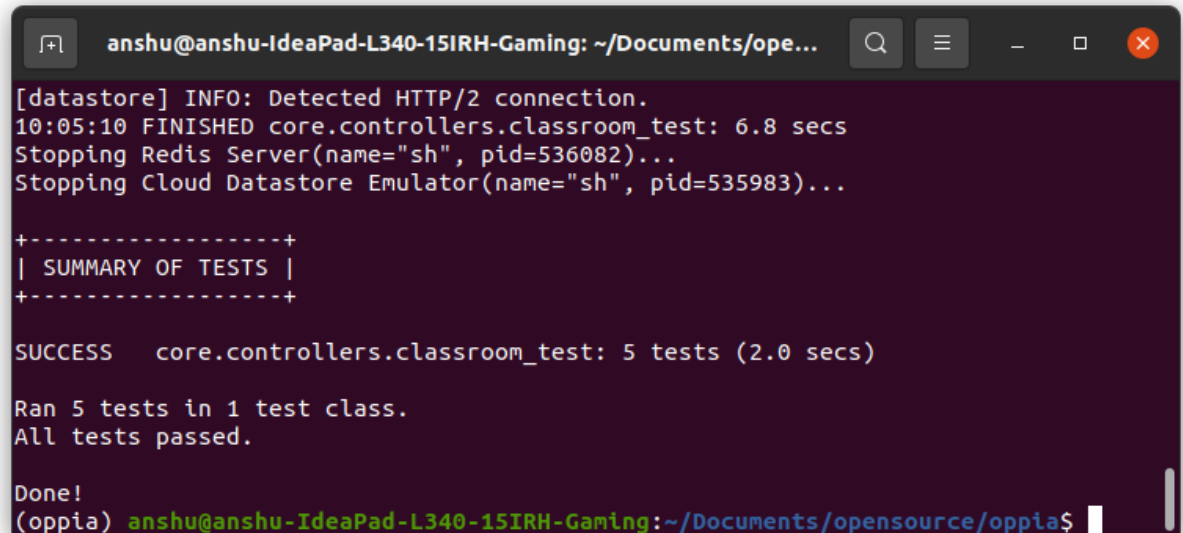
Timezone: Asia/Kolkata (IST) (+5:30 GMT)

Time commitment

I would work for 20hrs per week to complete the project within due time.

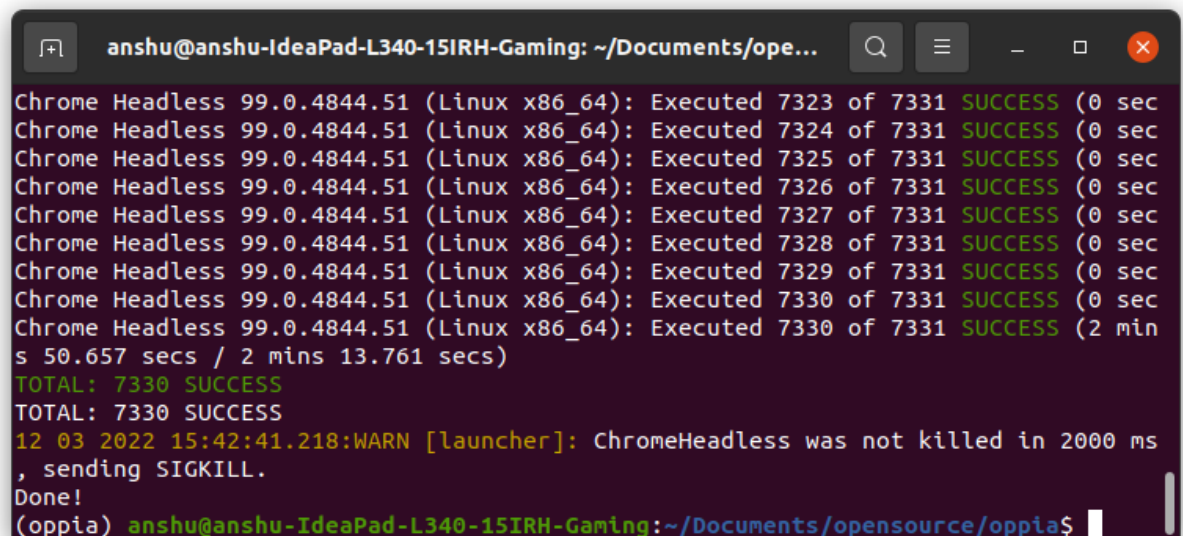
Essential Prerequisites

- I am able to run a single backend test target on my machine.



```
anshu@anshu-IdeaPad-L340-15IRH-Gaming: ~/Documents/ope...  
[datastore] INFO: Detected HTTP/2 connection.  
10:05:10 FINISHED core.controllers.classroom_test: 6.8 secs  
Stopping Redis Server(name="sh", pid=536082)...  
Stopping Cloud Datastore Emulator(name="sh", pid=535983)...  
  
+-----+  
| SUMMARY OF TESTS |  
+-----+  
  
SUCCESS   core.controllers.classroom_test: 5 tests (2.0 secs)  
  
Ran 5 tests in 1 test class.  
All tests passed.  
  
Done!  
(oppia) anshu@anshu-IdeaPad-L340-15IRH-Gaming: ~/Documents/opensource/oppia$
```

- I am able to run all the frontend tests at once on my machine.



```
anshu@anshu-IdeaPad-L340-15IRH-Gaming: ~/Documents/ope...  
Chrome Headless 99.0.4844.51 (Linux x86_64): Executed 7323 of 7331 SUCCESS (0 sec  
Chrome Headless 99.0.4844.51 (Linux x86_64): Executed 7324 of 7331 SUCCESS (0 sec  
Chrome Headless 99.0.4844.51 (Linux x86_64): Executed 7325 of 7331 SUCCESS (0 sec  
Chrome Headless 99.0.4844.51 (Linux x86_64): Executed 7326 of 7331 SUCCESS (0 sec  
Chrome Headless 99.0.4844.51 (Linux x86_64): Executed 7327 of 7331 SUCCESS (0 sec  
Chrome Headless 99.0.4844.51 (Linux x86_64): Executed 7328 of 7331 SUCCESS (0 sec  
Chrome Headless 99.0.4844.51 (Linux x86_64): Executed 7329 of 7331 SUCCESS (0 sec  
Chrome Headless 99.0.4844.51 (Linux x86_64): Executed 7330 of 7331 SUCCESS (0 sec  
Chrome Headless 99.0.4844.51 (Linux x86_64): Executed 7330 of 7331 SUCCESS (2 min  
s 50.657 secs / 2 mins 13.761 secs)  
TOTAL: 7330 SUCCESS  
TOTAL: 7330 SUCCESS  
12 03 2022 15:42:41.218:WARN [launcher]: ChromeHeadless was not killed in 2000 ms  
, sending SIGKILL.  
Done!  
(oppia) anshu@anshu-IdeaPad-L340-15IRH-Gaming: ~/Documents/opensource/oppia$
```

- I am able to run one suite of e2e tests on my machine.

```
anshu@anshu-IdeaPad-L340-15IRH-Gaming: ~/Documents/ope...
Blog dashboard functionality
    ♦♦♦ should check user profile is visible on both blog dashboard and editor, c
create, edit and delete a blog post from blog dashboard
[16:57:22] W/element - more than one element found for locator By(css selector, .
toast-success) - the first result will be used
    ♦♦♦ should create, publish, and delete the published blog post from dashboar
d.
[16:57:38] W/element - more than one element found for locator By(css selector, .
toast-success) - the first result will be used
    ♦♦♦ should create, publish, check for thumbnail uploading error, unpublish a
nd delete the blog post
[16:57:48] W/element - more than one element found for locator By(css selector, .
toast-success) - the first result will be used
[16:58:09] W/element - more than one element found for locator By(css selector, .
toast-success) - the first result will be used
[2022-03-12 16:58:12 +0530] [597798] [INFO] Starting gunicorn 20.1.0
[2022-03-12 16:58:12 +0530] [597798] [INFO] Listening at: http://0.0.0.0:24994 (5
97798)
[2022-03-12 16:58:12 +0530] [597798] [INFO] Using worker: sync
[2022-03-12 16:58:12 +0530] [597802] [INFO] Booting worker with pid: 597802
    ♦♦♦ should create multiple blog posts both published and drafts and check fo
r navigation through list view

4 specs, 0 failures
Finished in 119.501 seconds

Executed 4 of 4 specs SUCCESS in 2 mins.
```

Other summer obligations

I do not have any such obligations this summer and I'll only be applying to Oppia for GSoC.

Communication channels

I intend to meet with my mentor 2 times a week to discuss progress and doubts clarification.
[On Gmeet, Discord, or any other platform].

The meeting time and the number can be flexible as per the mentor's convenience.

Section 2: Proposal Details

Problem Statement

Link to PRD (or N/A if there isn't one)	Achieve 100% Per-File Branch and Line Coverage for the Frontend and the Backend
Target Audience	Oppia developers
Core User Need	The codebase should be completely covered, with 100% <i>per-file</i> branch and line coverage for both frontend and backend to prevent regressions. Here 'per-file' means that every non-test file is fully covered by its associated test file.
What goals do we want the solution to achieve?	<p>The solution aims to achieve following two goals-</p> <ol style="list-style-type: none">1. Achieve 100% Per-File Branch and Line Coverage for the Frontend and the Backend.2. Implement a testing infrastructure to ensure that the 100% per-file branch and line coverage are maintained post GSoC. This could be divided into sub-goals:<ol style="list-style-type: none">a. Modified or newly added files lacking 100% per-file backend line coverage fail the pre-push hook and the GitHub checks.b. Modified or newly added files lacking 100% per-file backend branch coverage fail the pre-push hook and the GitHub checks.c. Incomplete per-file backend line or branch coverage causes the GitHub checks to fail.d. Incomplete overall frontend line or branch coverage causes pre-push hook and the GitHub checks to fail.e. <i>Modified</i> and <i>newly added</i> files lacking 100% per-file frontend line coverage fail the Github checks.f. <i>Modified</i> and <i>newly added</i> files lacking 100% per-file frontend branch coverage fail the Github checks.

Section 2.1: WHAT

Key User Stories and Tasks

#	Title	User Story	Priority ¹	List of tasks needed to	Links to mocks / prototypes,
---	-------	------------	-----------------------	-------------------------	------------------------------

¹ Use the **MoSCoW** system ("Must have", "Should have", "Could have"). You can read more [here](#).

		Description (role, goal, motivation) "As a ..., I need ..., so that"		achieve the goal (this is the "User Journey")	and/or PRD sections that spec out additional requirements.
1	Increase per-file backend Line coverage to be 100%	As a developer in Oppia's team, I want 100% per-file branch and line coverage of the backend, i.e, all non-test files should be covered by its associated test file.	Must have	Cover all remaining backend files increasing line coverage to 100%	After we achieve 100% per-file backend line coverage, if a developer tries to push a file whose all lines are not fully covered, [this] message stops him from pushing the code.
				Remove all file names from scripts/backend_tests_incomplete_coverage.txt . The files listed in backend_tests_incomplete_coverage.txt lack full per-file coverage and once we achieve our goal, we can successfully clear this file.	
2	Increase per-file backend Branch coverage to be 100%		Must have	Use the coverage.py tool to list out the files with incomplete branch coverage in 'backend_test_incomplete_branch_coverage.txt'.	After we achieve 100% per-file backend branch coverage, if a developer tries to push a file whose all branches are not fully covered, [this] message stops him from pushing the code.
				Cover all the branches of the files listed in 'backend_test_incomplete_branch_coverage.txt' and remove the file once the goal of 100% per-file branch coverage is achieved.	
3	Increase per-file frontend line coverage to be 100%.	As a developer in Oppia's team, I want all existing frontend code to be properly tested with all lines and branches covered to avoid any bugs popping out at a later stage.	Must have	Cover all remaining frontend files increasing line coverage to 100%	After we achieve 100% per-file backend line coverage, if a developer tries to push a file whose all lines are not fully covered, [this] message stops him from pushing the code.
				Remove NOT_FULLY_COVERED_FILE NAMES from scripts/check_frontend_test_coverage.py	
4	Increase per-file frontend branch coverage to 100%		Must have	Use this check to identify frontend files which lack 100% branch coverage and store them under a variable FILES_WITH_INCOMPLETE_BRANCH_COVERAGE in scripts/check_frontend_test_coverage.py	After we achieve 100% per-file frontend branch coverage, if a developer tries to push a file whose all branches are not fully covered, [this] message stops him from pushing the code.
				Increase branch coverage of	

				files mentioned in FILES_WITH_INCOMPLETE_BRANCH_COVERAGE to 100% and then remove the variable.	
--	--	--	--	--	--

Technical Requirements






Additions/Changes to Web Server Endpoint Contracts


None

Calls to Web Server Endpoints

None

UI Screens/Components

#	ID	Description of new UI component	i18n required ?	Mock/spec links	A11y requirements
1.	Terminal logs for backend line coverage checks	After we achieve 100% per-file backend line coverage, if a developer tries to push a file whose all lines are not fully covered, a message stops him from pushing the code.	No	 Backend_line....	No
2.	Terminal logs for backend branch coverage checks	After we achieve 100% per-file backend branch coverage, if a developer tries to push a file whose all branches are not fully covered, a message stops him from pushing the code.	No	 Backend_bra...	No
3.	Terminal logs for backend overall coverage check	On running <code>python -m scripts.run_backend_tests --generate_coverage_report</code> , all backend tests are run and the 100% line and branch coverage is reported.	No	 Backend_ove...	No
4.	Terminal logs for frontend line coverage checks	After we achieve 100% per-file frontend line coverage, if a developer tries to push a file whose all lines are not fully covered, a message stops him from pushing the code.	No	 frontend_line....	No
5.	Terminal logs for	After we achieve 100% per-file	No	 frontend_bra...	No

	frontend branch coverage checks	frontend branch coverage, if a developer tries to push a file whose all branches are not fully covered, a message stops him from pushing the code.			
6.	Terminal logs for frontend overall coverage check	On running <code>`python -m scripts.run_frontend_tests --check_coverage`</code> , all frontend tests are run and 100% line and branch coverage is reported.	No	 Frotend_over...	No

Data Handling and Privacy

The project neither collects new user data nor changes the way how user data is collected. This is because the test files simply test the working of existing code and do not participate in the working of the code itself.

Other Requirements

Some frontend files which are still in Angular JS need to be migrated to Angular 2+ first before their tests are written.

The Angular Migration team aims to cover these files before the commencement of GSoC. However, if Angular Migration is running behind schedule, I will migrate these files first before writing tests for them.

Section 2.2: HOW

Existing Status Quo

BACKEND

We currently have ~99% backend branch coverage and ~100% backend line coverage.

FILES LACKING 100% PER-FILE LINE COVERAGE:

As mentioned in [scripts/backend_tests_incomplete_coverage.txt](#), following files are left to be covered.

```
core.constants.py
```


core.controllers.domain_objects_validator.py
core.controllers.feedback.py
core.controllers.reader.py
core.controllers.tasks.py
core.controllers.voice_artist.py
core.domain.app_feedback_report_domain.py
core.domain.auth_services.py
core.domain.blog_domain.py
core.domain.collection_services.py
core.domain.exp_domain.py
core.domain.exp_services.py
core.domain.feedback_domain.py
core.domain.image_validation_services.py
core.domain.interaction_registry.py
core.domain.learner_progress_services.py
core.domain.opportunity_services.py
core.domain.question_fetchers.py
core.domain.rights_manager.py
core.domain.role_services.py
core.domain.state_domain.py
core.domain.stats_services.py
core.domain.story_domain.py
core.domain.subtopic_page_domain.py
core.domain.suggestion_registry.py
core.domain.topic_domain.py
core.domain.user_domain.py
core.domain.user_services.py
core.domain.value_generators_domain.py
core.jobs.job.py_utils.py
core.schema_utils.py
core.storage.app_feedback_report.gae_models.py
core.storage.audit.gae_models.py
core.storage.base_model.gae_models.py
core.storage.classifier.gae_models.py

core.storage.collection.gae_models.py
core.storage.email.gae_models.py
core.storage.exploration.gae_models.py
core.storage.feedback.gae_models.py
core.storage.job.gae_models.py
core.storage.skill.gae_models.py
core.storage.suggestion.gae_models.py
core.storage.topic.gae_models.py
core.storage.user.gae_models.py
core.utils.py
extensions.interactions.base.py
main.py
scripts.docstrings_checker.py
scripts.install_backend_python_libs.py
scripts.install_third_party_libs.py
scripts.linters.js_ts_linter.py
scripts.linters.pylint_extensions.py

FILES LACKING 100% PER-FILE BRANCH COVERAGE:

To list out the files lacking 100% branch coverage, some changes were to be made in the `run_backend_test.py` file.

- The command for the coverage tool to execute, was modified to include the `branch` flag:

```
exc_list = [
    sys.executable, COVERAGE_MODULE_PATH, 'run',
    '--branch', TEST_RUNNER_PATH, test_target_flag
]
```

- Total coverage for a file was calculated from the process output as follows:

```
coverage_result = re.search(
    r'TOTAL\s+(\d+)\s+(\d+)\s+(\d+)\s+(\d+)\s+(\d+)\s+(?P<total>\d+)\s+',
    process.stdout)
```

```
coverage = float(coverage_result.group('total'))
```

- The test_modules are listed below which do not cover all branches of their associated code files.

scripts.pre_push_hook_test
scripts.run_e2e_tests_test
scripts.concurrent_task_utils_test
scripts.common_test
scripts.check_if_pr_is_low_risk_test
scripts.setup_test
scripts.rtl_css_test
scripts.build_test
scripts.install_third_party_test
scripts.pre_commit_hook_test
scripts.check_frontend_test_coverage_test
scripts.clean_test
scripts.create_topological_sort_of_all_services_test
scripts.release_scripts.repo_specific_changes_fetcher_test
scripts.linters.other_files_linter_test
scripts.linters.general_purpose_linter_test
scripts.linters.html_linter_test
core.domain.story_services_test
core.domain.story_fetchers_test
core.domain.classifier_services_test
core.domain.rules_registry_test
core.domain.rte_component_registry_test
core.domain.wipeout_domain_test
core.domain.platform_parameter_registry_test
core.domain.voiceover_services_test
core.domain.html_validation_service_test
core.domain.search_services_test
core.domain.stats_domain_test
core.domain.fs_services_test

core.domain.draft_upgrade_services_test
core.domain.html_cleaner_test
core.domain.beam_job_services_test
core.domain.blog_services_test
core.domain.event_services_test
core.domain.learner_goals_services_test
core.domain.exp_services_test
core.domain.translation_domain_test
core.domain.fs_domain_test
core.domain.skill_domain_test
core.domain.calculation_registry_test
core.domain.skill_services_test
core.domain.topic_services_test
core.domain.learner_playlist_services_test
core.domain.wipeout_service_test
core.domain.question_domain_test
core.domain.config_domain_test
core.domain.param_domain_test
core.domain.summary_services_test
core.domain.visualization_registry_test
core.domain.playthrough_issue_registry_test
core.domain.user_query_services_test
core.domain.action_registry_test
core.domain.recommendations_services_test
core.domain.user_query_domain_test
core.domain.app_feedback_report_services_test
core.domain.question_services_test
core.domain.collection_domain_test
core.domain.feedback_services_test
core.domain.rights_domain_test
core.domain.platform_parameter_domain_test
core.domain.email_manager_test
core.jobs.transforms.validation.base_validation_test
core.jobs.batch_jobs.math_interactions_audit_jobs_test

core.platform.auth.firebase_auth_services_test
core.platform.email.dev_mode_email_services_test
core.platform.taskqueue.cloud_tasks_emulator_test
core.platform.taskqueue.cloud_taskqueue_services_test
core.platform.bulk_email.mailchimp_bulk_email_services_test
core.controllers.contributor_dashboard_admin_test
core.controllers.subtopic_viewer_test
core.controllers.blog_admin_test
core.controllers.contributor_dashboard_test
core.controllers.profile_test
core.controllers.editor_test
core.controllers.blog_homepage_test
core.controllers.topics_and_skills_dashboard_test
core.controllers.classroom_test
core.controllers.learner_playlist_test
core.controllers.suggestion_test
core.controllers.story_editor_test
core.controllers.admin_test
core.controllers.practice_sessions_test
core.controllers.topic_editor_test
core.controllers.questions_list_test
core.controllers.skill_editor_test
core.controllers.library_test
core.controllers.payload_validator_test
core.controllers.acl_decorators_test
core.controllers.base_test
core.controllers.creator_dashboard_test
core.storage.statistics.gae_models_test
core.storage.question.gae_models_test
core.storage.blog.gae_models_test
extensions.objects.models.objects_test
extensions.answer_summarizers.models_test

FRONTEND

We currently have ~80% frontend branch coverage and ~96% frontend line coverage.

FILES LACKING 100% LINE COVERAGE:

The list NOT_FULLY_COVERED_FILENAMES in [scripts/check_frontend_test_coverage.py](#), mentions the files that lack 100% line coverage. They are listed below:

angular-html-bind.directive.ts
App.ts
Base.ts
ck-editor-4-rte.component.ts
ck-editor-4-widgets.initializer.ts
exploration-states.service.ts
expression-interpolation.service.ts
google-analytics.initializer.ts
learner-answer-info.service.ts
mathjax-bind.directive.ts
object-editor.directive.ts
oppia-interactive-music-notes-input.directive.ts
oppia-interactive-pencil-code-editor.component.ts
oppia-root.directive.ts
parameterize-rule-description.filter.ts
python-program.tokenizer.ts
question-update.service.ts
rule-type-selector.directive.ts
select2-dropdown.directive.ts
translation-file-hash-loader-backend-api.service.ts
truncate-input-based-on-interaction-answer-type.filter.ts
unit-test-utils.ajs.ts
voiceover-recording.service.ts

FILES LACKING 100% BRANCH COVERAGE:

We fetch information about frontend coverage from 'oppia/./karma_coverage_reports/lcov.info' file. To list out the files lacking 100% branch coverage, some changes were to be made in the 'check_frontend_test_coverage.py' file.

- Modified the 'LcovStanzaRelevantLines' class to include two new properties (namely total_branches and covered_branches):

```
match = re.search(r'BRF:(\d+)\n', stanza)
if match is None:
    raise Exception(
        'It wasn\'t possible to get the total branches of {} file.'
        'It\'s not possible to diff the test coverage correctly.'
        .format(file_name))
self.total_branches = int(match.group(1))

match = re.search(r'BRH:(\d+)\n', stanza)
if match is None:
    raise Exception(
        'It wasn\'t possible to get the covered branches of {} file.'
        'It\'s not possible to diff the test coverage correctly.'
        .format(file_name))
self.covered_branches = int(match.group(1))
```

- Modified the 'check_coverage_changes()' method to calculate 'total_branches' and 'covered_branches' along with 'total_lines' and 'covered_lines'. If the total branches and covered branches differed in a file, the filename was added in FILES_WITH_INCOMPLETE_BRANCH_COVERAGE list.

```
for stanza in stanzas:
    file_name = stanza.file_name
    total_lines = stanza.total_lines
    covered_lines = stanza.covered_lines
    total_branches = stanza.total_branches
    covered_branches = stanza.covered_branches
    if total_branches != covered_branches:
        FILES_WITH_INCOMPLETE_BRANCH_COVERAGE.append(file_name)
```

- All frontend files with incomplete branch coverage are listed below:

about-foundation-page-root.component.ts

about-foundation-page.component.ts
about-page-root.component.ts
about-page.component.ts
access-validation-backend-api.service.ts
activity-tiles-infinity-grid.component.ts
add-answer-group-modal.controller.ts
add-audio-translation-modal.controller.ts
add-hint-modal.component.ts
add-misconception-modal.component.ts
add-or-update-solution-modal.component.ts
add-worked-example.component.ts
admin-backend-api.service.ts
admin-config-tab.component.ts
admin-data.service.ts
admin-dev-mode-activities-tab.component.ts
admin-features-tab.component.ts
admin-misc-tab.component.ts
admin-navbar.component.ts
admin-page.component.ts
admin-roles-tab.component.ts
alert-message.component.ts
alerts.service.ts
algebraic-expression-editor.component.ts
algebraic-expression-input-rules.service.ts
algebraic-expression-input-validation.service.ts
angular-html-bind-wrapper.directive.ts
angular-html-bind.directive.ts
answer-classification.service.ts
answer-group-editor.component.ts
AnswerGroupObjectFactory.ts
App.ts
assets-backend-api.service.ts
assign-skill-to-topic-modal.component.ts
attribution-guide.component.ts
attribution.service.ts
audio-bar.component.ts
audio-file-uploader.component.ts

audio-player.service.ts
audio-preloader.service.ts
audio-translation-bar.directive.ts
audio-translation-language.service.ts
auth-backend-api.service.ts
auth.service.ts
autogenerated-audio-player.service.ts
autosave-info-modals.service.ts
background-banner.component.ts
base-content.component.ts
Base.ts
beam-jobs-tab.component.ts
blog-admin-backend-api.service.ts
blog-admin-data.service.ts
blog-admin-navbar.component.ts
blog-admin-page.component.ts
blog-card-preview-modal.component.ts
blog-card.component.ts
blog-dashboard-backend-api.service.ts
blog-dashboard-navbar-breadcrumb.component.ts
blog-dashboard-page.component.ts
blog-dashboard-page.service.ts
blog-dashboard-tile.component.ts
blog-post-action-confirmation.component.ts
blog-post-editor-backend-api.service.ts
blog-post-editor-pre-logo-action.component.ts
blog-post-editor.component.ts
boolean-editor.component.ts
bottom-navbar-status.service.ts
browser-checker.service.ts
can-access-splash-page.guard.ts
cancel-beam-job-dialog.component.ts
change-list.service.ts
change-subtopic-assignment-modal.component.ts
chapter-editor-tab.component.ts
ck-editor-4-rte.component.ts
ck-editor-4-widgets.initializer.ts

ck-editor-copy-content.service.ts
ck-editor-copy-toolbar.component.ts
classifier-data-backend-api.service.ts
classroom-backend-api.service.ts
classroom-page-root.component.ts
classroom-page.component.ts
code-repl-rules.service.ts
code-repl-validation.service.ts
code-string-editor.component.ts
codemirror-mergeview.component.ts
codemirror.component.ts
collection-creation-backend-api.service.ts
collection-creation.service.ts
collection-details-editor.component.ts
collection-editor-navbar-breadcrumb.component.ts
collection-editor-navbar.component.ts
collection-editor-page.component.ts
collection-editor-pre-publish-modal.component.ts
collection-editor-routing.service.ts
collection-editor-save-modal.component.ts
collection-editor-state.service.ts
collection-editor-tab.component.ts
collection-footer.component.ts
collection-linearizer.service.ts
collection-local-nav.component.ts
collection-navbar.component.ts
collection-node-creator.component.ts
collection-node-editor.component.ts
collection-permissions-card.component.ts
collection-player-backend-api.service.ts
collection-player-page.component.ts
collection-rights-backend-api.service.ts
collection-summary-tile.component.ts
collection-update.service.ts
community-lessons-tab.component.ts
concept-card-backend-api.service.ts
concept-card.component.ts

ConceptCardObjectFactory.ts
confirm-delete-state-modal.component.ts
confirm-discard-changes-modal.component.ts
confirm-leave-modal.component.ts
confirm-question-exit-modal.component.ts
contact-page-root.component.ts
content-language-selector.component.ts
content-translation-language.service.ts
content-translation-manager.service.ts
context.service.ts
continue-button.component.ts
continue-validation.service.ts
contribution-and-review-backend-api.service.ts
contribution-and-review.service.ts
contribution-opportunities-backend-api.service.ts
contribution-opportunities.service.ts
contributions-and-review.component.ts
contributor-dashboard-admin-backend-api.service.ts
contributor-dashboard-admin-navbar.component.ts
contributor-dashboard-admin-page.component.ts
contributor-dashboard-page.component.ts
conversation-skin.component.ts
coord-two-dim-editor.component.ts
create-activity-button.component.ts
create-activity-modal.component.ts
create-feedback-thread-modal.component.ts
create-new-skill-modal.component.ts
create-new-skill-modal.service.ts
create-new-subtopic-modal.component.ts
create-new-topic-modal.component.ts
creator-dashboard-backend-api.service.ts
creator-dashboard-page.component.ts
current-interaction.service.ts
custom-osk-letters-editor.component.ts
customize-interaction-modal.component.ts
debouncer.service.ts
delete-account-backend-api.service.ts

delete-account-modal.component.ts
delete-account-page-root.component.ts
delete-account-page.component.ts
delete-answer-group-modal.component.ts
delete-audio-translation-modal.component.ts
delete-exploration-modal.component.ts
delete-hint-modal.component.ts
delete-interaction-modal.component.ts
delete-last-hint-modal.component.ts
delete-misconception-modal.component.ts
delete-skill-modal.component.ts
delete-solution-modal.component.ts
delete-state-skill-modal.component.ts
delete-story-modal.component.ts
delete-topic-modal.component.ts
delete-worked-example-modal.component.ts
device-info.service.ts
display-hint-modal.component.ts
display-solution-interstitial-modal.component.ts
display-solution-modal.component.ts
document-attribute-customization.service.ts
donate-page-root.component.ts
donate-page.component.ts
drag-and-drop-sort-input-validation.service.ts
edit-profile-picture-modal.component.ts
edit-thumbnail-modal.component.ts
editable-collection-backend-api.service.ts
editable-exploration-backend-api.service.ts
editable-question-backend-api.service.ts
editable-story-backend-api.service.ts
editable-topic-backend-api.service.ts
editor-first-time-events.service.ts
editor-navbar-breadcrumb.component.ts
editor-navigation.component.ts
editor-reloading-modal.component.ts
email-dashboard-backend-api.service.ts
email-dashboard-data.service.ts

email-dashboard-page.component.ts
email-dashboard-result-backend-api.service.ts
email-dashboard-result.component.ts
end-exploration-backend-api.service.ts
end-exploration-validation.service.ts
error-404-page-root.component.ts
error-404-page.component.ts
error-page-root.component.ts
error-page.component.ts
event-bus.service.ts
exploration-category.service.ts
exploration-correctness-feedback.service.ts
exploration-creation-backend-api.service.ts
exploration-creation.service.ts
exploration-data-backend-api.service.ts
exploration-data.service.ts
exploration-diff.service.ts
exploration-editor-page.component.ts
exploration-editor-suggestion-modal.controller.ts
exploration-editor-tab.component.ts
exploration-embed-button-modal.component.ts
exploration-engine.service.ts
exploration-features-backend-api.service.ts
exploration-footer.component.ts
exploration-html-formatter.service.ts
exploration-id-validation.service.ts
exploration-improvements-backend-api.service.ts
exploration-improvements-task-registry.service.ts
exploration-improvements.service.ts
exploration-init-state-name.service.ts
exploration-language-code.service.ts
exploration-metadata-modal.controller.ts
exploration-objective.service.ts
exploration-param-changes.service.ts
exploration-param-specs.service.ts
exploration-permissions-backend-api.service.ts
exploration-player-page.component.ts

exploration-player-state.service.ts
exploration-property.service.ts
exploration-publish-modal.component.ts
exploration-recommendations-backend-api.service.ts
exploration-recommendations.service.ts
exploration-rights-backend-api.service.ts
exploration-rights.service.ts
exploration-save.service.ts
exploration-states.service.ts
exploration-stats-backend-api.service.ts
exploration-stats.service.ts
exploration-successfully-flagged-modal.component.ts
exploration-summary-backend-api.service.ts
exploration-summary-tile.component.ts
exploration-tags.service.ts
exploration-title.service.ts
ExplorationObjectFactory.ts
expression-evaluator.service.ts
expression-interpolation.service.ts
expression-syntax-tree.service.ts
extension-tag-assembler.service.ts
extract-image-filenames-from-model.service.ts
fatigue-detection.service.ts
favicon.service.ts
feedback-popup-backend-api.service.ts
feedback-popup.component.ts
feedback-tab.component.ts
feedback-thread-summary.model.ts
filtered-choices-field.component.ts
flag-exploration-modal.component.ts
focus-manager.service.ts
focus-on.directive.ts
format-rte-preview.pipe.ts
format-timer.filter.ts
fraction-editor.component.ts
fraction-input-rules.service.ts
fraction-input-validation.service.ts

generate-content-id.service.ts
get-started-page-root.component.ts
goals-tab.component.ts
google-analytics.initializer.ts
graph-data.service.ts
graph-editor.component.ts
graph-input-rules.service.ts
graph-input-validation.service.ts
graph-layout.service.ts
graph-viz.component.ts
guest-collection-progress.service.ts
guppy-configuration.service.ts
guppy-initialization.service.ts
headroom.directive.ts
help-modal.component.ts
hint-and-solution-buttons.component.ts
hint-and-solution-modal.service.ts
hint-editor.component.ts
hints-and-solution-manager.service.ts
history-tab-backend-api.service.ts
history-tab.component.ts
home-tab.component.ts
html-escaper.service.ts
hybrid-router-module-provider.ts
i18n.service.ts
image-click-input-validation.service.ts
image-editor.component.ts
image-local-storage.service.ts
image-preloader.service.ts
image-upload-helper.service.ts
image-uploader.component.ts
image-with-regions-editor.component.ts
image-with-regions-reset-confirmation.component.ts
improvements-tab.component.ts
information-card-modal.component.ts
input-response-pair.component.ts
int-editor.component.ts

interaction-attributes-extractor.service.ts
interaction-rules-registry.service.ts
InteractionObjectFactory.ts
interactive-map-validation.service.ts
internet-connectivity.service.ts
item-selection-input-validation.service.ts
keyboard-shortcut-help-modal.component.ts
keyboard-shortcut.service.ts
language-util.service.ts
layers-demo.model.ts
leaflet-baselayers.directive.ts
leaflet-control-layers-changes.model.ts
leaflet-control-layers.directive.ts
leaflet-control-layers.wrapper.ts
leaflet-layer.directive.ts
leaflet-layers.directive.ts
leaflet.directive.ts
leaflet.util.ts
learner-answer-details-backend-api.service.ts
learner-answer-info-card.component.ts
learner-answer-info.service.ts
learner-dashboard-activity-backend-api.service.ts
learner-dashboard-activity-ids.model.ts
learner-dashboard-backend-api.service.ts
learner-dashboard-icons.component.ts
learner-dashboard-ids-backend-api.service.ts
learner-dashboard-page.component.ts
learner-dashboard-suggestion-modal.component.ts
learner-local-nav-backend-api.service.ts
learner-local-nav.component.ts
learner-playlist-modal.component.ts
learner-story-summary-tile.component.ts
learner-topic-goals-summary-tile.component.ts
learner-topic-summary-tile.component.ts
learner-view-info-backend-api.service.ts
learner-view-info.component.ts
learner-view-rating-backend-api.service.ts

learner-view-rating.service.ts
lesson-information-card-modal.component.ts
library-footer.component.ts
library-page-backend-api.service.ts
library-page-root.component.ts
library-page.component.ts
license-explanation-modal.component.ts
license-page-root.component.ts
list-of-sets-of-translatable-html-content-ids-editor.component.ts
list-of-tabs-editor.component.ts
list-of-unicode-string-editor.component.ts
local-storage.service.ts
login-page-root.component.ts
login-page.component.ts
login-required-message.component.ts
login-required-modal.component.ts
logout-page-root.component.ts
logout-page.component.ts
lost-changes-modal.component.ts
LostChangeObjectFactory.ts
maintenance-page.component.ts
mark-all-audio-and-translations-as-needing-update-modal.component.ts
mark-audio-as-needing-update-modal.component.ts
math-equation-editor.component.ts
math-equation-input-rules.service.ts
math-equation-input-validation.service.ts
math-expression-content-editor.component.ts
math-interactions.service.ts
mathjax-bind.directive.ts
mathjax.directive.ts
merge-skill-modal.component.ts
messenger.service.ts
meta-tag-customization.service.ts
misconception-editor.component.ts
moderator-page-backend-api.service.ts
moderator-page.component.ts

moderator-unpublish-exploration-modal.component.ts
multi-selection-field.component.ts
multiple-choice-input-validation.service.ts
music-notes-input-rules.service.ts
music-notes-input-validation.service.ts
music-phrase-editor.component.ts
navigation.service.ts
nested-directives-recursion-timeout-prevention.service.ts
ng-init.directive.ts
nonnegative-int-editor.component.ts
normalize-whitespace.pipe.ts
normalized-string-editor.component.ts
number-conversion.service.ts
number-with-units-editor.component.ts
number-with-units-rules.service.ts
number-with-units-validation.service.ts
NumberWithUnitsObjectFactory.ts
numeric-expression-editor.component.ts
numeric-expression-input-validation.service.ts
numeric-input-validation.service.ts
object-editor.directive.ts
on-screen-keyboard.component.ts
oppia-angular-root.component.ts
oppia-help-modal-number-with-units.component.ts
oppia-interactive-algebraic-expression-input.component.ts
oppia-interactive-code-repl.component.ts
oppia-interactive-continue.component.ts
oppia-interactive-drag-and-drop-sort-input.component.ts
oppia-interactive-end-exploration.component.ts
oppia-interactive-fraction-input.component.ts
oppia-interactive-graph-input.component.ts
oppia-interactive-image-click-input.component.ts
oppia-interactive-interactive-map.component.ts
oppia-interactive-item-selection-input.component.ts
oppia-interactive-math-equation-input.component.ts
oppia-interactive-multiple-choice-input.component.ts
oppia-interactive-music-notes-input.component.ts

oppia-interactive-number-with-units.component.ts
oppia-interactive-numeric-expression-input.component.ts
oppia-interactive-numeric-input.component.ts
oppia-interactive-pencil-code-editor.component.ts
oppia-interactive-ratio-expression-input.component.ts
oppia-interactive-set-input.component.ts
oppia-interactive-text-input.component.ts
oppia-noninteractive-collapsible.component.ts
oppia-noninteractive-image.component.ts
oppia-noninteractive-link.component.ts
oppia-noninteractive-math.component.ts
oppia-noninteractive-skillreview-concept-card-modal.component.ts
oppia-noninteractive-skillreview.component.ts
oppia-noninteractive-tabs.component.ts
oppia-noninteractive-video.component.ts
oppia-response-algebraic-expression-input.component.ts
oppia-response-code-repl.component.ts
oppia-response-continue.component.ts
oppia-response-drag-and-drop-sort-input.component.ts
oppia-response-fraction-input.component.ts
oppia-response-graph-input.component.ts
oppia-response-image-click-input.component.ts
oppia-response-interactive-map.component.ts
oppia-response-item-selection-input.component.ts
oppia-response-math-equation-input.component.ts
oppia-response-multiple-choice-input.component.ts
oppia-response-music-notes-input.component.ts
oppia-response-number-with-units.component.ts
oppia-response-numeric-expression-input.component.ts
oppia-response-numeric-input.component.ts
oppia-response-pencil-code-editor.component.ts
oppia-response-ratio-expression-input.component.ts
oppia-response-set-input.component.ts
oppia-response-text-input.component.ts
oppia-rte-parser.service.ts
oppia-short-response-algebraic-expression-input.component.ts

oppia-short-response-code-repl.component.ts
oppia-short-response-continue.component.ts
oppia-short-response-drag-and-drop-sort-input.component.ts
oppia-short-response-fraction-input.component.ts
oppia-short-response-graph-input.component.ts
oppia-short-response-image-click-input.component.ts
oppia-short-response-interactive-map.component.ts
oppia-short-response-item-selection-input.component.ts
oppia-short-response-math-equation-input.component.ts
oppia-short-response-multiple-choice-input.component.ts
oppia-short-response-music-notes-input.component.ts
oppia-short-response-number-with-units.component.ts
oppia-short-response-numeric-expression-input.component.ts
oppia-short-response-numeric-input.component.ts
oppia-short-response-pencil-code-editor.component.ts
oppia-short-response-ratio-expression-input.component.ts
oppia-short-response-set-input.component.ts
oppia-short-response-text-input.component.ts
oppia-visualization-click-hexbins.directive.ts
opportunities-list-item.component.ts
opportunities-list.component.ts
outcome-destination-editor.component.ts
outcome-editor.component.ts
outcome-feedback-editor.component.ts
page-head.service.ts
page-title.service.ts
param-changes-editor.component.ts
ParamChangesObjectFactory.ts
parameter-metadata.service.ts
parameterize-rule-description.filter.ts
ParamSpecObjectFactory.ts
ParamSpecsObjectFactory.ts
partnerships-page-root.component.ts
partnerships-page.component.ts
pencil-code-editor-rules.service.ts
pencil-code-editor-validation.service.ts

pencil-code-reset-confirmation.component.ts
pending-account-deletion-page-root.component.ts
pie-chart.component.ts
platform-feature-admin-backend-api.service.ts
platform-feature-backend-api.service.ts
platform-feature-dummy-backend-api.service.ts
platform-feature.service.ts
playbook-page-root.component.ts
playbook.component.ts
player-position.service.ts
player-transcript.service.ts
playthrough-backend-api.service.ts
playthrough-issues-backend-api.service.ts
playthrough-issues.service.ts
playthrough.service.ts
PlaythroughObjectFactory.ts
populate-rule-content-ids.service.ts
positive-int-editor.component.ts
post-publish-modal.component.ts
practice-tab.component.ts
prediction-algorithm-registry.service.ts
preferences-page-root.component.ts
preferences-page.component.ts
preferred-language-selector.component.ts
preferred-languages.component.ts
pretest-question-backend-api.service.ts
prevent-page-unload-event.service.ts
preview-set-parameters-modal.component.ts
preview-summary-tile-modal.component.ts
preview-tab.component.ts
preview-thumbnail.component.ts
privacy-page-root.component.ts
profile-link-image-backend-api.service.ts
profile-link-image.component.ts
profile-page-backend-api.service.ts
profile-page-navbar.component.ts
profile-page-root.component.ts

profile-page.component.ts
progress-nav.component.ts
progress-tab.component.ts
promo-bar-backend-api.service.ts
promo-bar.component.ts
python-program.tokenizer.ts
question-backend-api.service.ts
question-creation.service.ts
question-difficulty-selector.component.ts
question-editor-save-modal.component.ts
question-editor.component.ts
question-misconception-editor.component.ts
question-misconception-selector.component.ts
question-player-engine.service.ts
question-player-state.service.ts
question-player.component.ts
question-suggestion-backend-api.service.ts
question-suggestion-editor-modal.controller.ts
question-suggestion-review-modal.controller.ts
question-update.service.ts
question-validation.service.ts
QuestionObjectFactory.ts
questions-list-select-skill-and-difficulty-modal.component.ts
questions-list.component.ts
questions-list.service.ts
questions-opportunities-select-difficulty-modal.component.ts
rating-display.component.ts
ratings-and-recommendations.component.ts
ratio-expression-editor.component.ts
ratio-expression-input-validation.service.ts
read-only-collection-backend-api.service.ts
read-only-exploration-backend-api.service.ts
real-editor.component.ts
rearrange-skills-in-subtopics-modal.controller.ts
reassign-role-confirmation-modal.component.ts
refresher-exploration-confirmation-modal.component.ts
refresher-exploration-confirmation-modal.service.ts

registration-session-expired-modal.component.ts
release-coordinator-backend-api.service.ts
release-coordinator-navbar.component.ts
release-coordinator-page-root.component.ts
release-coordinator-page.component.ts
remove-activity-modal.component.ts
remove-role-confirmation-modal.component.ts
request-interceptor.service.ts
response-header.component.ts
responses.service.ts
revert-exploration-modal.component.ts
review-material-editor.component.ts
review-test-backend-api.service.ts
roles-and-actions-visualizer.component.ts
router.service.ts
rte-helper-modal.controller.ts
rte-output-display.component.ts
rubrics-editor.component.ts
rule-editor.component.ts
rule-type-selector.directive.ts
RuleObjectFactory.ts
sanitized-url-editor.component.ts
save-pending-changes-modal.component.ts
save-validation-fail-modal.component.ts
save-version-mismatch-modal.component.ts
schema-based-editor.directive.ts
schema-based-float-editor.directive.ts
schema-based-int-editor.directive.ts
schema-based-list-editor.directive.ts
schema-based-unicode-editor.directive.ts
schema-default-value.service.ts
score-ring.component.ts
search-backend-api.service.ts
search-bar.component.ts
search-explorations-backend-api.service.ts
search-results.component.ts
search.service.ts

select-skill-modal.component.ts
select-topics.component.ts
select2-dropdown.directive.ts
server-connection-backend-api.service.ts
set-input-validation.service.ts
set-of-algebraic-identifier-editor.component.ts
set-of-unicode-string-editor.component.ts
setting-tab-backend-api.service.ts
settings-tab.component.ts
sharing-links.component.ts
side-navigation-bar.component.ts
sidebar-status.service.ts
signup-page-backend-api.service.ts
signup-page-root.component.ts
signup-page.component.ts
site-analytics.service.ts
skill-backend-api.service.ts
skill-concept-card-editor.component.ts
skill-creation-backend-api.service.ts
skill-description-editor.component.ts
skill-editor-navbar-breadcrumb.component.ts
skill-editor-navbar.directive.ts
skill-editor-page.component.ts
skill-editor-routing.service.ts
skill-editor-save-modal.component.ts
skill-editor-state.service.ts
skill-mastery-backend-api.service.ts
skill-mastery-modal.controller.ts
skill-mastery.component.ts
skill-misconceptions-editor.component.ts
skill-prerequisite-skills-editor.component.ts
skill-preview-modal.component.ts
skill-preview-tab.component.ts
skill-rights-backend-api.service.ts
skill-rubrics-editor.component.ts
skill-selector-editor.component.ts
skill-selector.component.ts

skill-update.service.ts
SkillObjectFactory.ts
skills-list.component.ts
solution-editor.component.ts
solution-explanation-editor.component.ts
solution-verification.service.ts
SolutionObjectFactory.ts
speech-synthesis-chunker.service.ts
splash-page-root.component.ts
splash-page.component.ts
staleness-detection.service.ts
start-new-beam-job-dialog.component.ts
state-card-is-checkpoint.service.ts
state-classifier-mapping.service.ts
state-content-editor.component.ts
state-content.service.ts
state-customization-args.service.ts
state-diff-modal-backend-api.service.ts
state-diff-modal.component.ts
state-editor.service.ts
state-graph-visualization.directive.ts
state-hints-editor.component.ts
state-hints.service.ts
state-interaction-editor.component.ts
state-interaction-id.service.ts
state-interaction-stats-backend-api.service.ts
state-interaction-stats.service.ts
state-name-editor.component.ts
state-next-content-id-index.service.ts
state-param-changes-editor.component.ts
state-param-changes.service.ts
state-property.service.ts
state-recorded-voiceovers.service.ts
state-responses.component.ts
state-skill-editor.component.ts
state-skill.service.ts
state-solicit-answer-details.service.ts

state-solution-editor.component.ts
state-solution.service.ts
state-top-answers-stats-backend-api.service.ts
state-top-answers-stats.service.ts
state-translation-editor.component.ts
state-translation-status-graph.component.ts
state-translation.component.ts
state-tutorial-first-time.service.ts
state-written-translations.service.ts
StateObjectFactory.ts
StatesObjectFactory.ts
statistics-tab.component.ts
stats-reporting-backend-api.service.ts
stats-reporting.service.ts
story-contents-object.model.ts
story-editor-navbar-breadcrumb.component.ts
story-editor-navbar.component.ts
story-editor-navigation.service.ts
story-editor-page.component.ts
story-editor-save-modal.component.ts
story-editor-state.service.ts
story-editor-unpublish-modal.component.ts
story-editor.directive.ts
story-node-editor.directive.ts
story-preview-tab.component.ts
story-summary-tile.component.ts
story-update.service.ts
story-viewer-backend-api.service.ts
story-viewer-navbar-breadcrumb.component.ts
story-viewer-navbar-pre-logo-action.component.ts
story-viewer-page-root.component.ts
story-viewer-page.component.ts
StoryObjectFactory.ts
subject-interests.component.ts
subtitled-html-editor.component.ts
subtitled-unicode-editor.component.ts
subtopic-editor-tab.component.ts

subtopic-preview-tab.component.ts
subtopic-summary-tile.component.ts
subtopic-validation.service.ts
subtopic-viewer-backend-api.service.ts
subtopic-viewer-navbar-breadcrumb.component.ts
subtopic-viewer-navbar-pre-logo-action.component.ts
subtopic-viewer-page.component.ts
subtopics-list.component.ts
suggestion-modal-for-exploration-editor.service.ts
suggestion-modal-for-learner-dashboard.service.ts
SuggestionThreadObjectFactory.ts
summary-list-header.component.ts
supplemental-card.component.ts
svg-editor.component.ts
svg-file-fetcher-backend-api.service.ts
svg-sanitizer.service.ts
svm-prediction.service.ts
switch-content-language-refresh-required-modal.component.ts
tag-misconception-modal.component.ts
take-break-modal.component.ts
teach-page-root.component.ts
teach-page.component.ts
terms-page-root.component.ts
text-input-prediction.service.ts
text-input-rules.service.ts
text-input-validation.service.ts
text_classifier.ts
thanks-page-root.component.ts
thanks-page.component.ts
thread-data-backend-api.service.ts
thread-table.component.ts
thumbnail-display.component.ts
thumbnail-uploader.component.ts
top-navigation-bar.component.ts
topic-creation-backend-api.service.ts
topic-creation.service.ts

topic-editor-navbar-breadcrumb.component.ts
topic-editor-navbar.component.ts
topic-editor-page.component.ts
topic-editor-routing.service.ts
topic-editor-save-modal.component.ts
topic-editor-send-mail-modal.component.ts
topic-editor-state.service.ts
topic-editor-stories-list.component.ts
topic-editor-tab.directive.ts
topic-landing-page.component.ts
topic-manager-role-editor-modal.component.ts
topic-preview-tab.component.ts
topic-rights-backend-api.service.ts
topic-summary-tile.component.ts
topic-update.service.ts
topic-viewer-backend-api.service.ts
topic-viewer-navbar-breadcrumb.component.ts
topic-viewer-page.component.ts
topic-viewer-stories-list.component.ts
TopicObjectFactory.ts
topics-and-skills-dashboard-backend-api.service.ts
topics-and-skills-dashboard-page.component.ts
topics-list.component.ts
training-data-editor-panel-modal.controller.ts
training-panel.component.ts
transfer-exploration-ownership-modal.component.ts
translatable-html-content-id.component.ts
translatable-set-of-normalized-string-editor.component.ts
translatable-set-of-unicode-string-editor.component.ts
translate-text-backend-api.service.ts
translate-text.service.ts
translation-language-selector.component.ts
translation-language.service.ts
translation-modal.component.ts
translation-opportunities.component.ts
translation-status.service.ts
translation-suggestion-review-modal.component.ts

translation-tab-active-content-id.service.ts
translation-tab-busy-modal.component.ts
translation-tab.component.ts
translation-topic-selector.component.ts
translation-topic.service.ts
translator-overview.component.ts
truncate-input-based-on-interaction-answer-type.filter.ts
truncate.pipe.ts
tutor-card.component.ts
tutorial-events-backend-api.service.ts
unassign-skill-from-topics-modal.component.ts
unicode-string-editor.component.ts
unit-test-utils.ajs.ts
unit-test-utils.ts
upload-activity-modal.component.ts
upload-blog-post-thumbnail-modal.component.ts
upload-blog-post-thumbnail.component.ts
url-interpolation.service.ts
url.service.ts
user-backend-api.service.ts
user-email-preferences-backend-api.service.ts
user-email-preferences.service.ts
user-exploration-permissions.service.ts
user.service.ts
utils.service.ts
validators.service.ts
version-diff-visualization.component.ts
version-tree.service.ts
view-beam-job-output-dialog.component.ts
voiceover-recording.service.ts
volunteer-page-root.component.ts
volunteer-page.component.ts
warnings-and-alerts.component.ts
welcome-modal.component.ts
welcome-translation-modal.component.ts
window-dimensions.service.ts
worked-example-editor.component.ts

wrap-text-with-ellipsis.pipe.ts
WrittenTranslationsObjectFactory.ts

Some of the above stated files are there due to the vague Karma coverage reports which considered commas in constructor definitions as comma operators. I had created a [debugging doc](#) for the same.

Note: Last updated April 12 2022.

The issues [#14187](#) and [#14219](#) are still open and some of the above stated files have been assigned to contributors.

Solution Overview

The entire project could be divided into six major tasks:

1. Write checks to ensure backend per-file line and branch coverage is 100% on the latest PRs.
2. Covering all backend files lacking 100% per-file line coverage.
3. Covering all backend files lacking 100% per-file branch coverage.
4. Write checks to ensure frontend per-file line and branch coverage is 100% on the latest PRs.
5. Covering all frontend files lacking 100% per-file line coverage.
6. Covering all frontend files lacking 100% per-file branch coverage.

Task 1: Write checks to ensure backend per-file line and branch coverage is maintained later on.

- Create a file named 'backend_tests_incomplete_branch_coverage.txt' in the scripts directory. The aim of this file would be to store filenames of all files that lack 100% per-file branch coverage.
- Create a file named 'backend_test_files_exclusion_list.txt' in the scripts folder. The aim of this file would be to store filenames of all files that do not require an associated test file (for example: __init__.py).
- Modify 'run_backend_tests.py':
Following features are to be added in the workflow of run_backend_tests.py:
 - Run branch coverage checks on all files and if the per-file branch coverage is less than 100% and the filename is not present in 'scripts/backend_tests_incomplete_branch_coverage.txt', then fail the coverage checks.
- Modify 'pre_push_hook.py':
 - Use 'collect_files_being_pushed()' method to obtain the list of files that are modified or newly added. Search for the files that end with '.py' extension and do coverage checks on them. These coverage checks are to be done utilizing the

methods defined in `run_backend_tests.py` which fail when the coverage is less than 100%.

- If either per-file line or branch coverage is less than 100% and the file's test_module is not present in 'scripts/backend_tests_incomplete_coverage.txt' or 'scripts/backend_tests_incomplete_branch_coverage.txt', then prevent the dev from pushing the files.
- Check if all the files ending with '.py' extension have an associated test file. If such a file is found and the filename is not present in 'backend_test_files_exclusion_list.txt', then display the error message and prevent the user from pushing the code.
- Also run coverage checks on CI in case the devs bypass the pre-push hook.

Task 2: Covering all backend files lacking 100% per-file line coverage

For a file to attain 100% per-file line coverage, all its lines must be tested by its associated test file. Currently all the files lacking 100% per-file line coverage have their associated test file listed in [scripts/backend_tests_incomplete_coverage.txt](#). Following steps are to be taken to cover the listed files:

1. Remove a test module from **scripts/backend_tests_incomplete_coverage.txt**.
2. Run `python -m scripts.run_backend_test --test_target=<the test module path> --generate_coverage_report` in the terminal.

Example:

On running command `python -m scripts.run_backend_test --test_target=core.domain.user_services_test --generate_coverage_report` following summary is shown:

```
+-----+
| SUMMARY OF TESTS |
+-----+

SUCCESS   core.domain.user_services_test: 120 tests (31.3 secs)
INCOMPLETE COVERAGE (87.0%): core.domain.user_services test
Name                               Stmts  Miss  Cover   Missing
-----
core/domain/user_services.py       660     86    87%    118, 133-137, 169, 593, 675-678, 704
, 754, 1065, 1102-1104, 1201-1203, 1214-1217, 1229-1232, 1275-1281, 1336-1352, 1362-1365,
1517, 1597-1611, 1660, 1663, 1688-1696, 1710, 1728, 1761-1764, 1808-1813, 1891, 1914, 19
26, 1938, 2133-2135, 2179, 2190, 2194, 2213-2214, 2228, 2277-2279
-----
TOTAL                               660     86    87%
```

Ran 120 tests in 1 test class.
All tests passed.

3. Write tests for the missing lines shown in the summary.
4. Follow the above stated steps for all the files listed in **scripts/backend_tests_incomplete_coverage.txt**.

Task 3: Covering all backend files lacking 100% per-file branch coverage

1. Modify the `run_backend_test.py` file to calculate branch coverage by changing:
 - a. Coverage command to be executed:

```
exc_list = [
    sys.executable, COVERAGE_MODULE_PATH, 'run',
    '--branch', TEST_RUNNER_PATH, test_target_flag
]
```

- b. The way coverage is calculated:

```
coverage_result = re.search(
    r'TOTAL\s+(\d+)\s+(\d+)\s+(\d+)\s+(\d+)\s+(\d+)\s+(?P<total>\d+)\s+',
    process.stdout)
coverage = float(coverage_result.group('total'))
```

2. Pick a test module from the `scripts/backend_tests_incomplete_branch_coverage.txt` and run the command `python -m scripts.run_backend_test -test_target=<the test module path> -generate_coverage_report` in the terminal.

Example:

On running command `python -m scripts.run_backend_test -test_target=core.domain.user_services_test -generate_coverage_report` following summary is shown:

```
+-----+
| SUMMARY OF TESTS |
+-----+

INCOMPLETE BRANCH COVERAGE (85.0%): core/domain/user_services.py
Name                               Stmts  Miss Branch BrPart  Cover   Missing
-----
core/domain/user_services.py       660     86    222     22    85%  118, 133-137, 169, 515
->511, 593, 675-678, 704, 754, 1043->1042, 1065, 1102-1104, 1201-1203, 1214-1217, 1229-12
32, 1275-1281, 1336-1352, 1362-1365, 1402->exit, 1415->exit, 1517, 1597-1611, 1660, 1663,
1688-1696, 1710, 1711->exit, 1728, 1761-1764, 1808-1813, 1891, 1914, 1926, 1938, 2133-21
35, 2147->exit, 2179, 2190, 2194, 2213-2214, 2228, 2277-2279
-----
TOTAL                               660     86    222     22    85%
```

3. Write tests to cover missing branch lines as shown in the summary.
4. Follow the above stated steps for all the files lacking 100% per-file branch coverage.

Task 4: Write checks to ensure frontend per-file line and branch coverage is maintained later on.

- Create a list `FILES_WITH_INCOMPLETE_BRANCH_COVERAGE` in `'scripts/check_frontend_test.coverage.py'` file. The aim of this list is to store filenames of all files that lack 100% branch coverage. The process to obtain filenames of such files is described above.
- Modify `'pre_push_hook.py'`:
 - Coverage checks should fail when either of the overall line or branch coverage is less than 100% for the newly added or modified files and the files are not listed in either of `FILES_WITH_INCOMPLETE_BRANCH_COVERAGE` or `NOT_FULLY_COVERED_FILENAMES` list. The code to check if modified and newly added files contain frontend files is already present in `'pre_push_hook.py'` under the method name `'does_diff_include_js_or_ts_files()'`.
- We use Karma to run our tests and calculate coverage reports. However, Karma faces some errors on running multiple files and calculating coverage. For more reference see [Issue #7053](#).

To overcome the situation, we used to merge all our spec files into a single spec file (code is present in ['core/templates/combined-tests.spec.ts'](#)).

However, the coverage report we then obtain is 'overall' and not 'per-file'. To check 'per-file' coverage we would have to manually change outermost 'describe' in the spec file to 'fdescribe' and then run the coverage checks. We would have to manually cover all files which lack 100% per-file coverage currently, but we can write checks to ensure that the files changed in latest PRs have 100% per-file line and branch coverage. We can run these checks on CI so that files without 100% per-file branch and line coverage do not get merged into the codebase. The methods to carry out those checks are described below:

```
# Function to list out all the spec files that need to be checked if
they
# completely cover their associated code file.
def collect_all_spec_files_to_be_checked(diff_files):
    for file_path in diff_files:
        # Check if the modified file is ts or js file.
        if file_path.endswith(b'.ts') or file_path.endswith(b'.js'):
            # Check if the modified file is already a test file.
            if(file_path.endswith(b'.spec.ts')):
                MODIFIED_FRONTEND_TEST_FILES_TO_BE_CHECKED.append(
                    file_path)
            # Check if the modified file has an associate test file.
            else:
                temp_list = file_path.split(".")
```

```

        temp_list.insert(len(temp_list) - 1, 'spec')

        test_file_path = ".".join(temp_list)
        if os.path.isfile(test_file_path):
            MODIFIED_FRONTEND_TEST_FILES_TO_BE_CHECKED.append(
                test_file_path)

# Function that runs frontend tests after changing the outermost
'describe' # to 'fdescribe' in a single spec file.
def run_per_file_frontend_test():
    x = 'describe'
    y = 'fdescribe'

    for file in MODIFIED_FRONTEND_TEST_FILES_TO_BE_CHECKED:
        # Change outermost 'describe' to 'fdescribe' in test files to be
        # checked.
        lines = []
        counter = 0
        f = open(file, "r+")

        for l in fileinput.input(files = file):
            if x in l:
                counter += 1
            if counter == 0:
                l = l.replace(x, y)
            lines.append(l)

        f.writelines(lines)
        f.close()

        # Run frontend tests.
        frontend_test_status = 0
        frontend_test_status = run_script_and_get_returncode(
            FRONTEND_TEST_CMDS)
        if frontend_test_status != 0:
            print('Push aborted due to failing frontend tests.')
            sys.exit(1)

    # Change 'fdescribe' back to 'describe' in modified test files.
    for file in MODIFIED_FRONTEND_TEST_FILES_TO_BE_CHECKED:
        lines = []
        f = open(file, "r+")

        for l in fileinput.input(files = file):
            l = l.replace(y, x)
            lines.append(l)

        f.writelines(lines)
        f.close()

```

The description of the above stated methods is as follows:

- **collect_all_spec_files_to_be_checked**

The method lists out all the spec files that need to be tested.

Pseudo code:

- Loop through all modified files' paths and check if they end with '.ts' or '.js' extension.
- If yes, then check if the file is a spec file.
 - If yes, then add the file path to MODIFIED_FRONTEND_TEST_FILES_TO_BE_CHECKED list.
 - If not, then check if it has an associated test file, then add the path of that spec file to MODIFIED_FRONTEND_TEST_FILES_TO_BE_CHECKED list.

- **run_per_file_frontend_test**

The method changes the outermost 'describe' to 'fdescribe' and then runs the frontend tests of each spec file. The coverage of the associated code files then calculated, is per-file and if it is less than 100%, then CI checks fail.

Pseudo code:

- Loop through all modified files' paths and replace the outermost 'describe' to 'fdescribe'.
- Run frontend tests and calculate coverage of the associated test file.
- If coverage is less than 100%, then fail the checks else pass them.
- At the end, we undo the changes done to the spec file.

Task 5: Covering all frontend files lacking 100% per-file line coverage.

1. Pick a file from NOT_FULLY_COVERED_FILENAMES list from [scripts/check_frontend_test_coverage.py](#). Note that the files listed in this file are those which lack 100% overall line coverage. After completing these files, we would have to manually check and cover all files that lack 100% per-file line coverage.
2. Change the outermost 'describe' of the spec file associated with that particular file to 'fdescribe'. This is done to ensure that the coverage results are per-file and not overall coverage. Now run 'python -m scripts.run_frontend_tests'. A coverage report will be generated at 'oppia../karma_coverage_reports/index.html'.

Example:

On changing outermost 'describe' of 'core/templates/AppSpec.ts' to 'fdescribe' and running frontend tests, coverage report generated looks something like this -

All files core/templates

74.24% Statements 147/198 30.61% Branches 15/49 41.66% Functions 10/24 73.84% Lines 144/195

Press *n* or *j* to go to the next uncovered block, *b*, *p* or *k* for the previous block.

Filter:

File		Statements		Branches		Functions		Lines	
App.ts	<div><div></div></div>	69.92%	86/123	37.84%	14/37	50%	9/18	69.92%	86/123
app.constants.ajs.ts	<div><div></div></div>	100%	48/48	100%	0/0	100%	0/0	100%	48/48
app.constants.ts	<div><div></div></div>	100%	1/1	100%	0/0	100%	0/0	100%	1/1
google-analytics.initializer.ts	<div><div></div></div>	30%	3/10	12.5%	1/8	33.33%	1/3	30%	3/10
hybrid-router-module-provider.ts	<div><div></div></div>	50%	7/14	0%	0/4	0%	0/3	36.36%	4/11
karma.module.ts	<div><div></div></div>	100%	1/1	100%	0/0	100%	0/0	100%	1/1
mathjaxConfig.ts	<div><div></div></div>	100%	1/1	100%	0/0	100%	0/0	100%	1/1

The coverage report of respective file looks something like this:

```
148 // Add an interceptor to convert requests to strings and to log and show
149 // warnings for error responses.
150 10x $httpProvider.interceptors.push([
151     '$exceptionHandler', '$q', '$log', 'AlertsService', 'CsrfTokenService',
152     function($exceptionHandler, $q, $log, AlertsService, CsrfTokenService) {
153 10x     return {
154         request: function(config) {
155             if (config.data) {
156                 return $q(function(resolve, reject) {
157                     // Get CSRF token before sending the request.
158                     CsrfTokenService.getTokenAsync().then(function(token) {
159                         if ((config.data instanceof FormData)) {
160                             var hasPayload = false;
161                             // Check whether the FormData has payload in it.
162                             for (var key of config.data.keys()) {
163                                 if (key === 'payload') {
164                                     hasPayload = true;
165                                     break;
166                                 }
167                             }
168                             // If the payload is not created, create it and append it
169                             // to the request data.
170                             if (!hasPayload) {
171                                 config.data.append(
172                                     'payload', JSON.stringify(config.data));
173                             }
174                             config.data.append('csrf_token', token);
175                             config.data.append('source', document.URL);
176                         } else {
177                             config.data = $.param({
178                                 csrf_token: token,
```

The lines highlighted in red are not tested by the AppSpec.ts file and need to be covered.

3. Write tests for the uncovered lines and on 100% coverage remove the filename from NOT_FULLY_COVERED_FILENAMES list from scripts/check_frontend_test_coverage.py.
4. Follow the above stated steps for all the files lacking 100% per-file line coverage.

Task 6: Covering all frontend files lacking 100% per-file branch coverage.

1. Pick a file from FILES_WITH_INCOMPLETE_BRANCH_COVERAGE list in `check_frontend_test_coverage.py` file.
2. Change the outermost `describe` of the spec file associated with that particular file to `fdescribe` and run `python -m scripts.run_frontend_tests`. A coverage report will be generated at oppia/./karma_coverage_reports/index.html.

Example:

On changing outermost `describe` of

`core/templates/pages/admin-page/roles-tab/admin-roles-tab.component.spec.ts` to




`fdescribe` and running frontend tests, coverage report generated looks something like this -

All files core/templates/pages/admin-page/roles-tab

66.44% Statements 101/152 72.22% Branches 39/54 52.5% Functions 21/40 65.54% Lines 97/148

Press *n* or *j* to go to the next uncovered block, *b*, *p* or *k* for the previous block.

Filter:

File ▲		Statements ▾		Branches ▾		Functions ▾		Lines ▾	
admin-roles-tab.component.ts		100%	90/90	80%	24/30	100%	21/21	100%	88/88
roles-and-actions-visualizer.component.ts		20%	6/30	75%	6/8	0%	0/8	17.24%	5/29
topic-manager-role-editor-modal.component.ts		15.63%	5/32	56.25%	9/16	0%	0/11	12.9%	4/31

The coverage report of respective file looks something like this:

```
62      addWarning(errorMessage: string): void {
63  1x      this.alertsService.addWarning(
64          errorMessage || 'Error communicating with server. ');
65      }
66
67      startEditing(): void {
68  5x      this.roleIsCurrentlyBeingEdited = true;
69  5x      this.adminBackendApiService.viewUsersRoleAsync(
70          this.username
71      ).then((userRoles) => {
72  2x          this.rolesFetched = true;
73  2x          this.userRoles = userRoles.roles;
74  2x          this.managedTopicIds = userRoles.managed_topic_ids;
75  2x          this.userIsBanned = userRoles.banned;
76      },
77      (errorResponse) => {
78  1x          this.roleIsCurrentlyBeingEdited = false;
79  1x          this.setStatusMessage.emit(errorResponse);
80      });
81  }
```

The lines highlighted in yellow are branches that are not tested by the spec file and need to be covered. Apart from these, there can be “E” and “I” indicators for if/else branches that are not covered. Sometimes the html reports can be misleading and do not mark the uncovered branches correctly, in that case, ‘**oppia/./karma_coverage_report/lcov.info**’ file could be viewed. It looks something like this:

```
SF:core/templates/hybrid-router-module-provider.ts
FN:37,(anonymous_2)
FN:40,(anonymous_3)
FN:62,(anonymous_4)
FNF:3
FNH:3
FNDA:2,(anonymous_2)
FNDA:2,(anonymous_3)
FNDA:2,(anonymous_4)
DA:32,1
DA:33,1
DA:37,1
DA:38,2
DA:41,2
DA:59,1
DA:63,2
DA:66,2
DA:68,2
DA:70,1
DA:76,1
LF:11
LH:11
BRDA:66,0,0,2
BRDA:66,0,1,0
BRDA:68,1,0,1
BRDA:68,1,1,1
BRF:4
BRH:3
end_of_record
```

Here SF refers to the filename, LF refers to the total lines of the files, LH refers to total lines which are covered, BRF refers to total branches in the file and BRH refers to the total branches covered. BRDA lists all branches in the file and its first property is the line number at which the branch starts. The last property of BRDA shows how many times a branch is visited. If it is 0, it means that the branch is still left to be covered.

Still if the coverage is not clear, one can view the '**oppia/./karma_coverage_report/coverage-final.json**'. Parsed form of which looks something like this:

```

▼ object {1257}
  ► /home/anshu/Documents/opensource/oppia/assets/constants.ts {7}
  ► /home/anshu/Documents/opensource/oppia/assets/rich_text_components_definitions.ts {7}
  ► /home/anshu/Documents/opensource/oppia/core/templates/App.ts {7}
  ► /home/anshu/Documents/opensource/oppia/core/templates/app.constants.ajs.ts {7}
  ► /home/anshu/Documents/opensource/oppia/core/templates/app.constants.ts {7}
  ► /home/anshu/Documents/opensource/oppia/core/templates/google-analytics.initializer.ts {7}
  ▼ /home/anshu/Documents/opensource/oppia/core/templates/hybrid-router-module-provider.ts {7}
    path : /home/anshu/Documents/opensource/oppia/core/templates/hybrid-router-module-provider.ts
    ► statementMap {14}
    ► fnMap {3}
    ▼ branchMap {2}
      ▼ 0 {3}
        ▼ loc {2}
          ▼ start {2}
            line : 66
            column : 4
          ▼ end {2}
            line : 74
            column : null
          type : if
        ► locations [2]
      ► 1 {3}
    ► s {14}
    ► f {3}
    ► b {2}
  ► /home/anshu/Documents/opensource/oppia/core/templates/karma.module.ts {7}
  ► /home/anshu/Documents/opensource/oppia/core/templates/mathjaxConfig.ts {7}
  ► /home/anshu/Documents/opensource/oppia/core/templates/app-events/app-events.ts {7}
  ► /home/anshu/Documents/opensource/oppia/core/templates/app-events/event-bus.service.ts {7}

```

3. Write tests for the uncovered branches and achieve 100% branch coverage.
4. Follow the above stated steps for all the files lacking 100% per-file branch coverage.

Third-Party Libraries

No new third-party libraries are required.

“Service” Dependencies

This feature adds no new service dependencies.

Impact on Other Oppia Teams

The project is of Dev workflow and the checks it will add will force Oppia developers to first cover all of their files (including both line and branch coverage) before pushing their code to Oppia.

Risks and mitigations

No potential risks.

Implementation Approach

Solution Overview of this proposal breaks the project into six tasks that are to be achieved.

Task 1 deals with writing checks so that backend line and branch coverage does not decrease while I am working to increase it to 100%. The checks would also ensure that coverage is maintained post GSoC too. Their basic workflow has been described above.

Task 2 and **Task 3** deal with writing backend tests that cover uncovered lines and branches and aim to increase the coverage of associated code files to 100%. Basic introduction and brief overview on writing backend tests is given below:

WRITING BACKEND TESTS

Oppia uses Python's [unittest framework](#) to test backend code. Backend test files live alongside the backend code files they test. For example, alongside `core/controllers/base.py` we'll find `core/controllers/base_test.py`. Note that each file is entirely code or entirely tests. No file mixes code and tests.

Inside test files, tests are organized into classes whose names end in `Tests`. Test cases are methods of these classes, and the method names begin with `test_`. Note that only methods beginning with `test_` and inside classes that inherit from `'unittest.TestCase'` are executed as tests. Here is an example of test from `'core/domain/collection_domain_test.py'`:

```

class CollectionChangeTests(test_utils.GenericTestBase):

    def test_collection_change_object_with_missing_cmd(self) -> None:
        with self.assertRaisesRegex( # type: ignore[no-untyped-call]
            utils.ValidationError, 'Missing cmd key in change dict'):
            collection_domain.CollectionChange({'invalid': 'data'})

    def test_collection_change_object_with_invalid_cmd(self) -> None:
        with self.assertRaisesRegex( # type: ignore[no-untyped-call]
            utils.ValidationError, 'Command invalid is not allowed'):
            collection_domain.CollectionChange({'cmd': 'invalid'})

```

Notice that the test class inherits from `test_utils.GenericTestBase` which in turn inherits from `unittest.TestCase`.

To write backend tests, following steps should be followed:

1. First understand the working of the function a test is going to cover. Understand the parameters it receives and how it makes use of them to update something or return something.
2. Try to test the interface, not the implementation. Treat the function as a black box and test its behavior. Don't test that the function uses a particular implementation. This will help to design a better API from an external user's perspective.
3. Give meaningful names to the tests. Following naming syntax has been recommended by Oppia:

`test_{{action}}_with_{{with_condition}}_{{has_expected_outcome}}`

 where `{{action}}`, `{{with_condition}}`, and `{{has_expected_outcome}}` are replaced with appropriate descriptions.
4. Tests should use the following general structure:
 - a. Setup - this is where you prepare any inputs/environment needed for the test.
 - b. Baseline verification - check the values without performing any action. This step is only needed if your action is state-changing (i.e. if the same assert statement would lead to one result at the baseline and a different result at the endline). Check the same values here as you check at the endline.
 - c. Action - perform the action or function call that leads to the expected change.
 - d. Endline verification - check that the values from the baseline verification have changed accordingly.
5. Try to keep the logic of the test simple. Write the test as a series of straightforward, descriptive and meaningful commands (also known in programming circles as "DAMP"). Use comments wherever necessary.
6. To check outputs, following points should be followed:

- a. Test each output as exactly and completely as possible. For example, it's better to compare equality for an entire dict rather than just checking that a particular value has changed.
 - b. Use `assertTrue(value)` or `assertFalse(value)` instead of `assertEqual(value, True)` or `assertEqual(value, False)`.
 - c. Use `assertIsNone(value)` instead of `assertEqual(value, None)`.
7. If you want to test code within a private method/function, test it by instead calling a public function that makes use of that function, or move it to a utility (if it's general-purpose) where it becomes public.
8. You should test each method of a class individually, with one or more test cases for each method. Define a `setUp()` method in the test if functionality or variables are going to be reused between tests. Here is an example of a good `setUp()` method from 'core/domain/exp_fetchers_test.py':

```
class ExplorationRetrievalTests(test_utils.GenericTestBase):
    """Test the exploration retrieval methods."""

    EXP_1_ID = 'exploration_1_id'
    EXP_2_ID = 'exploration_2_id'
    EXP_3_ID = 'exploration_3_id'

    def setUp(self):
        super(ExplorationRetrievalTests, self).setUp()
        self.signup(self.OWNER_EMAIL, self.OWNER_USERNAME)
        self.owner_id = self.get_user_id_from_email(self.OWNER_EMAIL)
        self.exploration_1 = self.save_new_default_exploration(
            self.EXP_1_ID, self.owner_id, title='Aa')
        self.exploration_2 = self.save_new_default_exploration(
            self.EXP_2_ID, self.owner_id, title='Bb')
        self.exploration_3 = self.save_new_default_exploration(
            self.EXP_3_ID, self.owner_id, title='Cc')

    def test_get_exploration_summaries_matching_ids(self):
        summaries = exp_fetchers.get_exploration_summaries_matching_ids([
            self.EXP_1_ID, self.EXP_2_ID, self.EXP_3_ID, 'nonexistent'])
        self.assertEqual(summaries[0].title, self.exploration_1.title)
        self.assertEqual(summaries[1].title, self.exploration_2.title)
        self.assertEqual(summaries[2].title, self.exploration_3.title)
        self.assertIsNone(summaries[3])

    def test_get_exploration_summaries_subscribed_to(self):
        summaries = exp_fetchers.get_exploration_summaries_subscribed_to(
            self.owner_id)
        self.assertEqual(summaries[0].title, self.exploration_1.title)
        self.assertEqual(summaries[1].title, self.exploration_2.title)
        self.assertEqual(summaries[2].title, self.exploration_3.title)
```

Oppia's wiki about writing [backend-tests](#) can be referred to know more about writing good tests.

Task 4 deals with writing checks that ensure that the overall coverage and the per-file coverage does not decrease by the latest PRs. These are mainly checks running on the CI that check the newly added or modified files. Their basic workflow has been described above.

Task 5 and **Task 6** deal with writing frontend tests that cover uncovered lines and branches and aim to increase the coverage of associated code files to 100%. Basic introduction and brief overview on writing frontend tests is given below:

WRITING FRONTEND TESTS

Oppia uses [Jasmine](#) and Karma for frontend testing. Frontend test files live alongside the frontend code files they test.

For example, alongside `core/templates/pages/admin-page/admin-page.component.ts` we'll find `core/templates/pages/admin-page/admin-page.component.spec.ts`. Note that each file is entirely code or entirely tests. No file mixes code and tests.

Inside the test files, there are many test function, some of them are:

- **describe**

The describe function has a string parameter which should contain the name of the component being tested or (when nested within another describe function) should describe the conditions imposed on the specific context pertaining to the tests in that describe block. Here are some examples:

- An outer describe function

```
describe('Component Name', function() {  
  ...  
});
```

- Two describe functions nested inside another describe function:

```
describe('Component Name', function() {  
  describe('when it is available', function() {...});  
  
  describe('when it is not available', function() {...});  
});
```

- **beforeEach**

The beforeEach function is used to set up essential configurations and variables before each test runs. This function is mostly used for three things:

- Injecting the modules to be tested or to be used as helpers inside the test file.
 - Mocking the unit test's external dependencies (only on AngularJS files).

- Providing Angular2+ services in downgrade files when the AngularJS service being tested uses any upgraded (Angular2+) service as a dependency.

Here is a real example from Oppia's codebase:

```
describe('Admin Page component ', () => {
  let component: AdminPageComponent;
  let fixture: ComponentFixture<AdminPageComponent>;

  let adminRouterService: AdminRouterService;
  let mockWindowRef: MockWindowRef;

  beforeEach(() => {
    mockWindowRef = new MockWindowRef();
    TestBed.configureTestingModule({
      imports: [HttpClientTestingModule],
      declarations: [AdminPageComponent],
      providers: [
        AdminRouterService,
        ChangeDetectorRef,
        {
          provide: WindowRef,
          useValue: mockWindowRef
        },
        {
          provide: PlatformFeatureService,
          useClass: MockPlatformFeatureService
        }
      ],
      schemas: [NO_ERRORS_SCHEMA]
    }).compileComponents();

    fixture = TestBed.createComponent(AdminPageComponent);
    component = fixture.componentInstance;
  });
});
```

- it

The 'it' function is where the test happens. Like the describe function, its first parameter is a string which should determine the action to be tested and the expected outcome of the tests. The string should have a clear description of what is going to be tested. Also, the string must start with "should".

Here is a real example from Oppia's codebase:

```
it('should save state content', function() {
  stateEditorService.setActiveStateName('First State');
  expect(explorationStatesService.getState('First State').content).toEqual(
    SubtitledHtml.createFromBackendDict({
      content_id: 'content',
      html: 'First State Content'
    }));
});
```

```

var displayedValue = SubtitledHtml.createFromBackendDict({
  content_id: 'content',
  html: 'First State Content Changed'
});
ctrl.saveStateContent(displayedValue);

expect(explorationStatesService.getState('First State').content).toEqual(
  displayedValue);
expect(ctrl.interactionIsShown).toBe(true);
});

it('should save state interaction id', function() {
  stateEditorService.setActiveStateName('First State');
  stateEditorService.setInteraction(
    explorationStatesService.getState('First State').interaction);

  expect(stateEditorService.interaction.id).toBe('TextInput');

  var newInteractionId = 'Continue';
  ctrl.saveInteractionId(newInteractionId);

  expect(stateEditorService.interaction.id).toBe(newInteractionId);
});

```

- **afterEach**

The `afterEach` function runs after each test, and it is not used often. It's mostly used when we are handling async features such as HTTP and timeout calls (both in AngularJS and Angular 2+).

- **afterAll**

The `afterAll` function runs after all the tests have finished, but it is almost never used in the codebase. There is a specific case which might be very helpful: when a global variable needs to be reassigned during the tests, you need to reset it to the default value after all the assertions are finished.

- **expect**

The `expect` function is used to assert a condition in the test. More can be learned about its methods in the [Jasmine documentation](#).

Here is a real example from Oppia's codebase:

```

it('should initialize the component and set values', fakeAsync(() => {
  componentInstance.ngOnInit();
  expect(componentInstance.explorationId).toEqual(expId);
  expect(componentInstance.expTitle).toEqual(expTitle);
  expect(componentInstance.expDesc).toEqual(expDesc);

  expect(componentInstance.expTitleTranslationKey).toEqual(
    'I18N_EXPLORATION_expId_TITLE');
});

```

```

expect(componentInstance.expDescTranslationKey).toEqual(
  'I18N_EXPLORATION_expId_DESCRIPTION');

// Translation is only displayed if the language is not English
// and its hacky translation is available.
let hackyExpTitleTranslationIsDisplayed = (
  componentInstance.isHackyExpTitleTranslationDisplayed());
expect(hackyExpTitleTranslationIsDisplayed).toBe(true);
let hackyStoryTitleTranslationIsDisplayed = (
  componentInstance.isHackyStoryTitleTranslationDisplayed());
expect(hackyStoryTitleTranslationIsDisplayed).toBe(true);
let hackyExpDescTranslationIsDisplayed = (
  componentInstance.isHackyExpDescTranslationDisplayed());
expect(hackyExpDescTranslationIsDisplayed).toBe(false);
});

```

Other good practise about writing frontend tests and naming conventions can be learned from Oppia's wiki about writing [frontend-tests](#).

Storage Model Layer Changes

None

Domain Objects

No new domain objects to be created and none updated.

User Flows (Controllers and Services)

None

Web frontend changes

None

Documentation changes

1. Documentation about checking and achieving 100% backend branch changes need to be added in [backend-tests wiki](#).
2. Documentation about checking and achieving 100% frontend branch changes need to be added in the [frontend-tests wiki](#).

Testing Plan

Backend-Testing-Framework Testing

All the testing for the backend-testing-framework proposed above would be manual where I will be trying to push files with incomplete coverage and check if the backend coverage checks fail.

Note: *Idle function* in tests below means a function that simply returns a value that is passed to it as a parameter.

#	Test name	Initial setup step	Step	Expectation
1.	Check if files lacking 100% per-file backend line coverage fail backend coverage checks.	<p>Add an idle function (let's say A) in any 1 python code file that has an associated test file and do not write tests for it.</p> <p>Add a function (let's say B) in another python code file code file that calls function A. Write tests for function B in the associated test file. This ensures that overall line coverage is 100% since function A is also tested but the per-file line coverage is less than 100%.</p>	Run backend coverage checks.	Coverage checks fail due to less than 100% per-file backend line coverage.
2.	Check if files lacking 100% per-file backend branch coverage fail backend coverage checks.	<p>Add a modified idle function that returns back the value passed to it only when the parameter has an specific value, i.e, add an if condition. Write tests for the function passing only that specific value to it.</p> <p>This will ensure that the per-file line coverage is 100% but the per-file branch coverage is less than 100% since the condition when If is not matched is never met.</p>	Run backend coverage checks.	Coverage checks fail due to less than 100% per-file backend branch coverage.
3.	Check if files lacking 100% per-file backend line coverage fails pre_push hook.	<p>Add an idle function (let's say A) in any 1 python code file that has an associated test file and do not write tests for it.</p> <p>Add a function (let's say B) in another python code file code file that calls function A. Write tests for function B in the associated test file. This ensures that overall line</p>	Commit the changes and try to push the changes to GitHub repo.	Push fails due to incomplete backend per-file line coverage.

		coverage is 100% since function A is also tested but the per-file coverage is less than 100%.		
4.	Check if files lacking 100% per-file backend branch coverage fail pre_push hook.	Add a modified idle function that returns back the value passed to it only when the parameter has an specific value, i.e, add an <i>if</i> condition. Write tests for the function passing only that specific value to it. This will ensure that the per-file line coverage is 100% but the per-file branch coverage is less than 100% since the condition when <i>If</i> is not matched is never met.	Commit the changes and try to push the changes to GitHub repo.	Push fails due to incomplete backend per-file branch coverage.
5.	Check if files lacking 100% per-file backend line coverage fail GitHub checks.	Add an idle function (let's say A) in any 1 python code file that has an associated test file and do not write tests for it. Add a function (let's say B) in another python code file code file that calls function A. Write tests for function B in the associated test file. This ensures that overall line coverage is 100% since function A is also tested but the per-file coverage is less than 100%.	Commit the changes and push them to GitHub using the 'no verify' flag. Make a PR to a separate branch of the repo.	GitHub checks fail due to incomplete backend per-file line coverage.
6.	Check if files lacking 100% per-file backend branch coverage fail GitHub checks.	Add a modified idle function that returns back the value passed to it only when the parameter has an specific value, i.e, add an <i>if</i> condition. Write tests for the function passing only that specific value to it. This will ensure that the per-file line coverage is 100% but the per-file branch coverage is less than 100% since the condition when <i>If</i> is not matched is never met.	Commit the changes and push them to GitHub using the 'no verify' flag. Make a PR to a separate branch of the repo.	GitHub checks fail due to incomplete backend per-file branch coverage.

The checks to ensure backend overall line coverage does not fall below what it initially was by the modified or newly added files are already done on GitHub.

Frontend-Testing-Framework Testing

The testing for frontend-testing-framework proposed above would involve both manual testing and testing the working of methods added and modified in the ``check_frontend_test_coverage.py`` file.

Backend Test for ``check_frontend_test_coverage.py`` file.

#	Test name	Initial setup step	Step	Expectation
1.	Check if <code>`check_coverage_changes`</code> method fails on incomplete branch coverage.	Change the <code>`self.lcov_items_list`</code> and make the BRV value of a file less than its BRH value. This signifies that covered branches of the file are less than the total branches. Check for the failure of the method using <code>`self.assertRaisesRegex(SystemExit, '1')`</code>	Run backend tests with <code>`check_frontend_test_coverage_test`</code> file as the test target.	The test passes and the method performs as expected, i.e, fails on incomplete branch coverage.

Manual testing of the GitHub checks.

Note: *Idle function* in tests below means a function that simply returns a value that is passed to it as a parameter.

#	Test name	Initial setup step	Step	Expectation
1.	Check if less than 100% overall frontend branch coverage fails the pre_push hook.	Add a modified idle function that returns back the value passed to it only when the parameter has an specific value, i.e, add an if condition. Write tests for the function passing only that specific value to it. This will ensure that the per-file line coverage is 100% but the per-file branch coverage is less than 100% since the condition when If is not matched is never met.	Commit the changes and try to push the changes to GitHub repo.	Push fails due to incomplete frontend per-file branch coverage.
2.	Check if less than 100% overall frontend branch coverage fails the GitHub	Add a modified idle function that returns back the value passed to it only when the parameter has an specific value, i.e, add an if condition. Write tests for the function passing only that specific value to it.	Commit the changes and push them to GitHub using the <code>`no verify`</code> flag. Make a PR to a separate branch of the repo.	GitHub checks fail due to incomplete overall branch coverage.

	checks.	This will ensure that the per-file line coverage is 100% but the per-file branch coverage is less than 100% since the condition when If is not matched is never met.		
3.	Check if files lacking 100% per-file frontend line coverage fail GitHub checks.	<p>Add an idle function (let's say A) in any 1 typescript code file that has an associated test file and do not write tests for it.</p> <p>Add a function (let's say B) in another typescript code file that calls function A. Write tests for function B in the associated test file. This ensures that overall line coverage is 100% since function A is also tested but the per-file line coverage is less than 100%.</p>	Commit the changes and push them to GitHub using the `no verify` flag. Make a PR to a separate branch of the repo.	GitHub checks fail due to incomplete frontend per-file line coverage in the newly added or modified files.
4..	Check if files lacking 100% per-file frontend branch coverage fail GitHub checks.	<p>Add a modified idle function that returns back the value passed to it only when the parameter has an specific value, i.e, add an if condition. Write tests for the function passing only that specific value to it.</p> <p>This will ensure that the per-file line coverage is 100% but the per-file branch coverage is less than 100% since the condition when If is not matched is never met.</p>	Commit the changes and push them to GitHub using the `no verify` flag. Make a PR to a separate branch of the repo.	GitHub checks fail due to incomplete frontend per-file branch coverage in the newly added or modified files.

The code to check if frontend overall line coverage falls below than what it initially was by the modified or newly added files is already present in the `pre_push_hook.py` file. Same checks also run on GitHub in case devs bypass the pre_push checks.

Feature testing

Does this feature include non-trivial user-facing changes?

NO

Implementation Plan

Milestone 1:

Key Objectives:

- Write checks in the pre_push hook.py file to ensure backend per-file line and branch coverage 100% in the newly added or modified files.
- Write checks for the CI to ensure backend per-file line and branch coverage is 100% for the files changed in the submitted PRs.
- Cover all backend files that lack 100% per-file line coverage.
- Cover all backend files that lack 100% per-file branch coverage.

No.	Description of PR / action	Prereq PR numbers	Target date for PR creation	Target date for PR to be merged
1	Checks added in the pre_push hook to maintain 100% per-file line and branch coverage.		13th June	18th June
2	GitHub checks to maintain 100% per-file line and branch coverage in submitted PRs		19th June	24th June
3	Increase backend per-file line coverage for 20 files.		25th June	30th June
4	Increase backend per-file line coverage for 20 files.		31st June	4th July
5	Increase backend per-file line coverage for 10 files.		5th July	8th July
6	Increase backend per-file branch coverage for 30 files.		9th July	14th July
7	Increase backend per-file branch coverage for 30 files.		15th July	20th July
8	Increase backend per-file branch coverage for 36 files.		20th July	25th July

Milestone 2:

Key Objectives:

- Write checks in 'pre_push_hook.py' to ensure frontend overall line and branch coverage is 100%.
- Write checks for the CI to ensure frontend per-file line and branch coverage is 100% for the files changed in the submitted PRs.
- Cover all frontend files that lack 100% line coverage.
- Cover all frontend files that lack 100% branch coverage.

No.	Description of PR / action	Prereq PR numbers	Target date for PR creation	Target date for PR to be merged
1	Checks added in the pre_push hook to maintain 100% overall line and branch coverage.		28th July	2nd August
2	GitHub checks to maintain 100% per-file line and branch coverage in submitted PRs.		3rd August	8th August
3	Increase frontend line coverage for 10 files.		9th August	14th August
4	Increase frontend line coverage for 13 files.		15th August	20th August
5	Increase frontend branch coverage for 30% of the total files.		21st August	26th August
6	Increase frontend branch coverage for 30% of the total files.		27th August	2nd September
7	Increase frontend branch coverage for 40% of the total files.		3rd September	8th September

With the current target date , I can make my final submission on the 8th Sep. The last submission date is 12th Sep. Thus in the worst case I will still have a period of 3-4 days to complete and fix things.

Future Work

- An issue listing backend files which currently do not have test files and should have one will be created. The description will elaborate the procedure to write tests and check coverage with a link pointing to Oppia's wiki about writing [backend-tests](#). It will serve as a beginner issue for new contributors to work upon.
- A new way to run frontend tests without combining all the spec files into one should be searched for. It will solve two problems:
 - Run tests in sequential manner which would help the Automated QA team to debug flakes.
 - Help us calculate 'per-file' coverage checks easily.