

About You

My name is James John James and I am a final year student of Mechanical and Aerospace Engineering in the prestigious University of Uyo, Uyo.

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

I want to work with Oppia mainly because of the learning process. I like that there are very talented and experienced software engineers in the Oppia team and I believe I will be able to learn a lot by working with the team.

I chose the Oppiabot project because I like working on developer tools stuff. GitHub Actions is a new and fun thing that I would like to learn a lot about and building a bot that will be used by the Oppia community is a fun way to learn about GitHub Actions.

Prior experience

Github actions can be created using YAML files, and shell or javascript scripts.

I have been learning/using javascript for building web applications for 3 years now. I worked on GSoC 2019 which I wrote quite an amount of javascript for [my project\(static serving\)](#).

I've also been making some contributions to the Oppia codebase in the speed team where we try to evaluate the causes of the current slow speed experienced on the platform and try to fix them.

Once the project ideas for GSoC 2020 got released, I started learning about GitHub Actions from the Github lab. So far, I have built 2 projects where I've used GitHub Actions to do some things similar to what can be found in the Oppiabot requirement spreadsheet. Here are the links to the projects.

1. <https://github.com/jameesjohn/github-actions-for-ci>
2. <https://github.com/jameesjohn/writing-javascript-actions>
3. <https://github.com/jameesjohn/comment-on-pr/>

PRs Made:

<https://github.com/oppia/oppia/pull/8137>

<https://github.com/oppia/oppia/pull/8078>

<https://github.com/oppia/oppia/pull/7702>

Contact info and timezone(s)

My primary email address is jamesjay4199@gmail.com. I can also be reached at jameesjohn@outlook.com.

The time zone I will be working from is GMT+1.

Time commitment

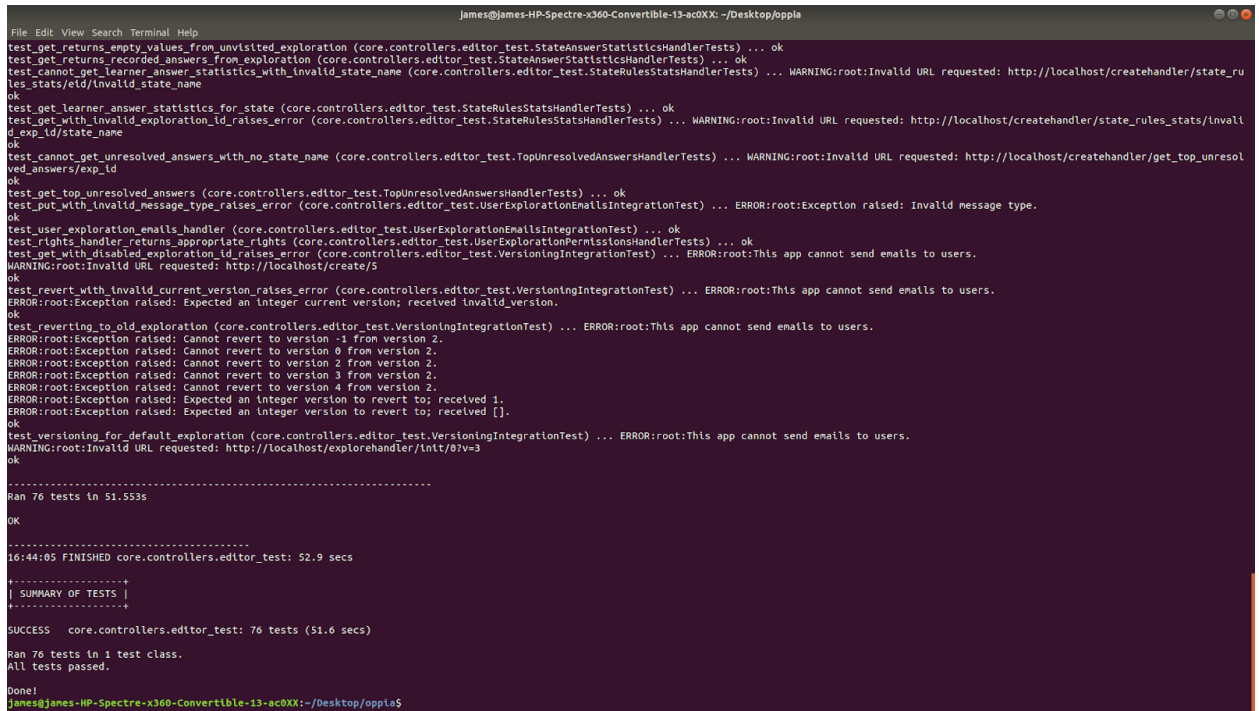
I will be able to put in about 6-7 hours per day for 6 days per week.

I'm not sure of how much time I will be able to commit on Sunday since my availability on Sunday is relative. So on average, 36 - 42 hours per week.

Essential Prerequisites

Answer the following questions:

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



```
File Edit View Search Terminal Help
James@james-HP-Spectre-x360-Convertible-13-ac0XX: ~/Desktop/oppla
test_get_returns_empty_values_from_unvisited_exploration (core.controllers.editor_test.StateAnswerStatisticsHandlerTests) ... ok
test_get_returns_recorded_answers_from_exploration (core.controllers.editor_test.StateAnswerStatisticsHandlerTests) ... ok
test_cannot_get_learner_answer_statistics_with_invalid_state_name (core.controllers.editor_test.StateRulesStatsHandlerTests) ... WARNING:root:Invalid URL requested: http://localhost/createhandler/state_rules_stats/eld/invalid_state_name
ok
test_get_learner_answer_statistics_for_state (core.controllers.editor_test.StateRulesStatsHandlerTests) ... ok
test_get_with_invalid_exploration_id_raises_error (core.controllers.editor_test.StateRulesStatsHandlerTests) ... WARNING:root:Invalid URL requested: http://localhost/createhandler/state_rules_stats/invalid_exp_id/state_name
ok
test_cannot_get_unresolved_answers_with_no_state_name (core.controllers.editor_test.TopUnresolvedAnswersHandlerTests) ... WARNING:root:Invalid URL requested: http://localhost/createhandler/get_top_unresolved_answers/exp_id
ok
test_get_top_unresolved_answers (core.controllers.editor_test.TopUnresolvedAnswersHandlerTests) ... ok
test_put_with_invalid_message_type_raises_error (core.controllers.editor_test.UserExplorationEmailsIntegrationTest) ... ERROR:root:Exception raised: Invalid message type.
ok
test_user_exploration_emails_handler (core.controllers.editor_test.UserExplorationEmailsIntegrationTest) ... ok
test_rights_handler_returns_appropriate_rights (core.controllers.editor_test.UserExplorationPermissionsHandlerTests) ... ok
test_get_with_disabled_exploration_id_raises_error (core.controllers.editor_test.VersioningIntegrationTest) ... ERROR:root:This app cannot send emails to users.
WARNING:root:Invalid URL requested: http://localhost/create/s
ok
test_revert_with_invalid_current_version_raises_error (core.controllers.editor_test.VersioningIntegrationTest) ... ERROR:root:This app cannot send emails to users.
ERROR:root:Exception raised: Expected an integer current version; received invalid_version.
ok
test_reverting_to_old_exploration (core.controllers.editor_test.VersioningIntegrationTest) ... ERROR:root:This app cannot send emails to users.
ERROR:root:Exception raised: Cannot revert to version -1 from version 2.
ERROR:root:Exception raised: Cannot revert to version 0 from version 2.
ERROR:root:Exception raised: Cannot revert to version 2 from version 2.
ERROR:root:Exception raised: Cannot revert to version 3 from version 2.
ERROR:root:Exception raised: Cannot revert to version 4 from version 2.
ERROR:root:Exception raised: Expected an integer version to revert to; received 1.
ERROR:root:Exception raised: Expected an integer version to revert to; received [].
ok
test_versioning_for_default_exploration (core.controllers.editor_test.VersioningIntegrationTest) ... ERROR:root:This app cannot send emails to users.
WARNING:root:Invalid URL requested: http://localhost/explorehandler/int/0?v=3
ok
-----
Ran 76 tests in 51.553s
OK
-----
16:44:05 FINISHED core.controllers.editor_test: 52.9 secs
+-----+
| SUMMARY OF TESTS |
+-----+
SUCCESS core.controllers.editor_test: 76 tests (51.6 secs)
Ran 76 tests in 1 test class.
All tests passed.
Done!
James@james-HP-Spectre-x360-Convertible-13-ac0XX: ~/Desktop/oppla$
```

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

```

File Edit View Search Terminal Help
James@James-HP-Spectre-x360-Convertible-13-ac0XX: ~/Desktop/oppla
HeadlessChrome 80.0.3987 (Linux 0.0.0): Executed 1965 of 2013 SUCCESS (0 secs / 15.839 secs)
ERROR: 'Error communicating with server.'
HeadlessChrome 80.0.3987 (Linux 0.0.0): Executed 1968 of 2013 SUCCESS (0 secs / 15.863 secs)
ERROR: 'Error communicating with server.'
HeadlessChrome 80.0.3987 (Linux 0.0.0): Executed 1968 of 2013 SUCCESS (0 secs / 15.863 secs)
ERROR: 'Error communicating with server.'
HeadlessChrome 80.0.3987 (Linux 0.0.0): Executed 1968 of 2013 SUCCESS (0 secs / 15.863 secs)
ERROR: 'Error communicating with server.'
HeadlessChrome 80.0.3987 (Linux 0.0.0): Executed 1969 of 2013 SUCCESS (0 secs / 15.873 secs)
ERROR: 'Error communicating with server.'
HeadlessChrome 80.0.3987 (Linux 0.0.0): Executed 1969 of 2013 SUCCESS (0 secs / 15.873 secs)
ERROR: 'Error communicating with server.'
HeadlessChrome 80.0.3987 (Linux 0.0.0): Executed 1969 of 2013 SUCCESS (0 secs / 15.873 secs)
ERROR: 'Error communicating with server.'
HeadlessChrome 80.0.3987 (Linux 0.0.0): Executed 1970 of 2013 SUCCESS (0 secs / 15.885 secs)
ERROR: 'Error communicating with server.'
HeadlessChrome 80.0.3987 (Linux 0.0.0): Executed 1972 of 2013 SUCCESS (0 secs / 15.904 secs)
ERROR: 'Error communicating with server.'
HeadlessChrome 80.0.3987 (Linux 0.0.0): Executed 1972 of 2013 SUCCESS (0 secs / 15.904 secs)
ERROR: 'Error communicating with server.'
HeadlessChrome 80.0.3987 (Linux 0.0.0): Executed 1972 of 2013 SUCCESS (0 secs / 15.904 secs)
ERROR: 'Error communicating with server.'
HeadlessChrome 80.0.3987 (Linux 0.0.0): Executed 1973 of 2013 SUCCESS (0 secs / 15.913 secs)
ERROR: 'Error communicating with server.'
HeadlessChrome 80.0.3987 (Linux 0.0.0): Executed 1974 of 2013 SUCCESS (0 secs / 15.927 secs)
ERROR: 'Sorry, you have been logged out [probably in another window]. Please log in again. You will be redirected to main page in a while!'
HeadlessChrome 80.0.3987 (Linux 0.0.0): Executed 1976 of 2013 SUCCESS (0 secs / 15.963 secs)
ERROR: 'Error communicating with server.'
HeadlessChrome 80.0.3987 (Linux 0.0.0): Executed 1987 of 2013 SUCCESS (0 secs / 16.024 secs)
ERROR: 'Error communicating with server.'
HeadlessChrome 80.0.3987 (Linux 0.0.0): Executed 1988 of 2013 SUCCESS (0 secs / 16.032 secs)
ERROR: 'Error communicating with server.'
HeadlessChrome 80.0.3987 (Linux 0.0.0): Executed 1989 of 2013 SUCCESS (0 secs / 16.039 secs)
ERROR: 'Error communicating with server.'
HeadlessChrome 80.0.3987 (Linux 0.0.0): Executed 1990 of 2013 SUCCESS (0 secs / 16.051 secs)
ERROR: 'Error communicating with server.'
HeadlessChrome 80.0.3987 (Linux 0.0.0): Executed 1991 of 2013 SUCCESS (0 secs / 16.058 secs)
ERROR: 'Error communicating with server.'
HeadlessChrome 80.0.3987 (Linux 0.0.0): Executed 1991 of 2013 SUCCESS (0 secs / 16.058 secs)
ERROR: 'Error communicating with server.'
HeadlessChrome 80.0.3987 (Linux 0.0.0): Executed 1991 of 2013 SUCCESS (0 secs / 16.058 secs)
ERROR: 'Error communicating with server.'
HeadlessChrome 80.0.3987 (Linux 0.0.0): Executed 1991 of 2013 SUCCESS (0 secs / 16.058 secs)
ERROR: 'Error communicating with server.'
HeadlessChrome 80.0.3987 (Linux 0.0.0): Executed 2013 of 2013 SUCCESS (19.167 secs / 16.125 secs)
TOTAL: 2013 SUCCESS
26.03.2020 17:49:15.856 WARN [launcher]: ChromeHeadless was not killed in 2000 ms, sending SIGKILL.
Done!
James@James-HP-Spectre-x360-Convertible-13-ac0XX:~/Desktop/oppla$

```

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

```

File Edit View Search Terminal Help
James@James-HP-Spectre-x360-Convertible-13-ac0XX: ~/Desktop/oppla
Entrypoint HtmlWebpackPlugin_41 = __child-HtmlWebpackPlugin_40
Entrypoint HtmlWebpackPlugin_42 = __child-HtmlWebpackPlugin_41
[./node_modules/html-webpack-plugin/lib/loader.js!./core/templates/pages/about-page/about-page.mainpage.html] 1.14 KiB [HtmlWebpackPlugin_0] [built]
[./node_modules/html-webpack-plugin/lib/loader.js!./core/templates/pages/admin-page/admin-page.mainpage.html] 854 bytes [HtmlWebpackPlugin_1] [built]
[./node_modules/html-webpack-plugin/lib/loader.js!./core/templates/pages/classroom-page/classroom-page.mainpage.html] 930 bytes [HtmlWebpackPlugin_2] [built]
[./node_modules/html-webpack-plugin/lib/loader.js!./core/templates/pages/collection-editor-page/collection-editor-page.mainpage.html] 1.2 KiB [HtmlWebpackPlugin_3] [built]
[./node_modules/html-webpack-plugin/lib/loader.js!./core/templates/pages/collection-player-page/collection-player-page.mainpage.html] 3.33 KiB [HtmlWebpackPlugin_4] [built]
[./node_modules/html-webpack-plugin/lib/loader.js!./core/templates/pages/community-dashboard-page/community-dashboard-page.mainpage.html] 1.21 KiB [HtmlWebpackPlugin_5] [built]
[./node_modules/html-webpack-plugin/lib/loader.js!./core/templates/pages/contact-page/contact-page.mainpage.html] 3.89 KiB [HtmlWebpackPlugin_6] [built]
[./node_modules/html-webpack-plugin/lib/loader.js!./core/templates/pages/creator-dashboard-page/creator-dashboard-page.mainpage.html] 1.43 KiB [HtmlWebpackPlugin_7] [built]
[./node_modules/html-webpack-plugin/lib/loader.js!./core/templates/pages/delete-account-page/delete-account-page.mainpage.html] 1.07 KiB [HtmlWebpackPlugin_8] [built]
[./node_modules/html-webpack-plugin/lib/loader.js!./core/templates/pages/donate-page/donate-page.mainpage.html] 1.14 KiB [HtmlWebpackPlugin_9] [built]
[./node_modules/html-webpack-plugin/lib/loader.js!./core/templates/pages/enroll-dashboard-pages/enroll-dashboard-page.mainpage.html] 1.35 KiB [HtmlWebpackPlugin_10] [built]
[./node_modules/html-webpack-plugin/lib/loader.js!./core/templates/pages/enroll-dashboard-pages/enroll-dashboard-result.mainpage.html] 1.24 KiB [HtmlWebpackPlugin_11] [built]
[./node_modules/html-webpack-plugin/lib/loader.js!./core/templates/pages/error-pages/error-page.mainpage.html] 979 bytes [HtmlWebpackPlugin_12] [built]
[./node_modules/html-webpack-plugin/lib/loader.js!./core/templates/pages/error-pages/error-iframe.mainpage.html] 2.3 KiB [HtmlWebpackPlugin_13] [built]
[./node_modules/html-webpack-plugin/lib/loader.js!./core/templates/pages/exploration-editor-page/exploration-editor-page.mainpage.html] 6.41 KiB [HtmlWebpackPlugin_14] [built]
+ 41 hidden modules
curl -o /home/james/Desktop/oppla/node_modules/webdriver-manager/downloads/chromedriver.xln https://chromedriver.storage.googleapis.com/
curl https://api.github.com/rate_limit -H "User-Agent: angular/webdriver-manager"
curl -o /home/james/Desktop/oppla/node_modules/webdriver-manager/downloads/selenium-server.xml https://selenium-release.storage.googleapis.com/
curl -o /home/james/Desktop/oppla/node_modules/webdriver-manager/downloads/geckodriver.json https://api.github.com/repos/mozilla/geckodriver/releases -H "User-Agent: angular/webdriver-manager"
curl -o /home/james/Desktop/oppla/node_modules/webdriver-manager/downloads/chromedriver_Linux64.zip https://chromedriver.storage.googleapis.com/2.41/chromedriver_Linux64.zip
curl -o /home/james/Desktop/oppla/node_modules/webdriver-manager/downloads/selenium-server-standalone-4.0.0-alpha-1.jar https://selenium-release.storage.googleapis.com/4.0/selenium-server-standalone-4.0.0-alpha-1.jar
curl -o /home/james/Desktop/oppla/node_modules/webdriver-manager/downloads/geckodriver-v0.26.0-linux64.tar.gz https://github.com/mozilla/geckodriver/releases/download/v0.26.0/geckodriver-v0.26.0-linux64.tar.gz
java -Dwebdriver.chrome.driver=/home/james/Desktop/oppla/node_modules/webdriver-manager/downloads/chromedriver_2.41 -Dwebdriver.gecko.driver=/home/james/Desktop/oppla/node_modules/webdriver-manager/downloads/geckodriver-v0.26.0-linux64.jar -jar /home/james/Desktop/oppla/node_modules/webdriver-manager/downloads/selenium-server-standalone-4.0.0-alpha-1.jar -role node -servlet org.openqa.grid.web.servlet.LifecycleServlet -registercycle 0 -port 4444
selenium process id: 1767
[08:09:15] I/launcher - Running 1 instances of WebDriver
[08:09:15] I/hosted - Using the selenium server at http://localhost:4444/wd/hub
Started
Dasmine started

Embedding
  ✓ should display and play embedded explorations
  ✓ should use the exploration language as site language.

2 specs, 0 failures
Finished in 183.557 seconds

Executed 2 of 2 specs SUCCESS in 3 mins 4 secs.
[08:12:00] I/launcher - 0 instance(s) of WebDriver still running
[08:12:00] I/launcher - chrome #01 passed
Killing /usr/bin/python /home/james/Desktop/oppla/./oppla_tools/google_appengine_1.9.67/google_appengine/dev_appserver.py --host 0.0.0.0 --port 9001 --clear_datastore=yes --dev_appserver_log_level=critical --log_level=critical --skip_sdk_update_check=true app_dev.yaml ...
Killing java -Dwebdriver.chrome.driver=/home/james/Desktop/oppla/node_modules/webdriver-manager/downloads/chromedriver_2.41 -Dwebdriver.gecko.driver=/home/james/Desktop/oppla/node_modules/webdriver-manager/downloads/geckodriver-v0.26.0-linux64.jar -jar /home/james/Desktop/oppla/node_modules/webdriver-manager/downloads/selenium-server-standalone-4.0.0-alpha-1.jar -role node -servlet org.openqa.grid.web.servlet.LifecycleServlet -registercycle 0 -port 4444 ...
James@James-HP-Spectre-x360-Convertible-13-ac0XX:~/Desktop/oppla$

```

Other summer obligations

I have no summer job or classes.

Communication channels

I predominantly use Whatsapp for communication. I also use Hangouts and normal email messaging.

Product Design

The primary users of Oppiabot are the developers on the Oppia team.

Oppiabot is needed for quite a number of functions in the workflow of Oppia developers, especially commenting on pull requests with recommended actions based on the status of the pull request, and when an action is needed from the author of the pull request.

Oppia developers primarily work on 2 areas: issues and pull requests.

Checks Related To Issues

Labelling Issues

User Behaviour: User adds a label to an issue.

Types of Labels

1. Project Labels

The following are project labels

- Backend
- Frontend

It should be noted that project labels can only be used for issues.

2. PR Labels:

The following are PR labels and should not be used on issues:

- Dependencies
- Critical
- stale
- Changelog labels(labels containing **PR Changelog**)
- Labels starting with "**PR:** " like *PR: LGTM*, *PR: don't merge - needs CLA*

3. Issue Labels:

All other labels not covered as a PR label is an issue label like *code-health*, *good-first-issue* etc.

Oppiabot Response:

Oppiabot checks the label that just got added and responds based on the type of label that was added.

- **Good first issue label:** If a *good-first-issue* label gets added by a user, Oppiabot will check if the user is allowed(white-listed) to add the label. This whitelist will be provided

by onboarding team lead and can be updated by them if need be, If the user is not allowed, Oppiabot removes the label with an appropriate message and pings the onboarding team lead to decide if the issue is a good first issue or not.[\[23\]](#). Oppiabot also assigns the onboarding team lead to ensure that an action is taken from him.

The screenshot shows a vertical timeline of GitHub issue activity. At the top, a label 'good-first-issue' is added by user 'jameesjohn' 9 days ago. Below this, a comment from the 'github-actions' bot is shown, stating: 'Hi @jameesjohn, thank you for proposing this as a good first issue. Looping in @Showtim3 to confirm, removing the tag until he does so.' At the bottom, the 'github-actions' bot is shown removing the 'good-first-issue' label 9 days ago.

- **PR label:** Oppiabot comments on the issue telling the user that the added label is not allowed on issues, providing a link to show users what labels are allowed for issues and also automatically removes the label.[\[17\]](#)

The screenshot shows a vertical timeline of GitHub issue activity. At the top, a label 'PR CHANGELOG: code health -- @jameesjohn' is added by user 'jameesjohn' 9 days ago. Below this, a comment from the 'github-actions' bot is shown, stating: 'Hi @jameesjohn, Changelog labels should not be used on issues, I'm removing the label. You can learn more about oppia labels and how they should be used here: <https://github.com/oppia/oppia/wiki/labels>.' At the bottom, the 'github-actions' bot is shown removing the 'PR CHANGELOG: code health -- @jameesjohn' label 9 days ago.

Periodic Checks [\[25\]](#)

Oppiabot will go through the issues periodically (24 hours) and add the **TODO: Triage** label to issues that have not been added to a project. Oppiabot will also ping the core maintainers to add the issue to a project.

Getting Assigned to Issues

User Behaviour: User gets assigned to an issue.

Oppiabot Response: Oppiabot checks if the user has signed the CLA.[23]

- **If the user has signed CLA:** Oppiabot does nothing
- **If the user has not signed the CLA:** Oppiabot comments on the issue telling the user to sign the CLA and unassigns the user from the issue[20].



github-actions bot commented 4 days ago



Hi @jameesjohn, you need to sign the CLA before you can get assigned to issues. Learn more about contributing to Oppia at <https://github.com/oppia/oppia/wiki/Contributing-code-to-Oppia#setting-things-up>



github-actions bot unassigned jameesjohn yesterday

Note: This feature does not apply to subtasks created within an issue.

New contributors can get assigned to an issue if they have interacted (commented) on the issue. <https://github.blog/2019-06-25-assign-issues-to-issue-commenters/>.

Checks Related To Pull Requests

Creating a Pull Request

User Behaviour: User creates a Pull Request.

Oppiabot Response: When a user creates a pull request, Oppiabot carries out the following checks:

1. New Contributor Check[14][2]

a. **New contributor that hasn't signed the CLA:**

Oppiabot will check if the PR author is a new contributor not in the CLA sheet, Oppiabot will mention the onboarding team lead in a comment informing them of the new user.

Oppiabot will also **close** the PR and provide a link to the wiki showing the next action for the user..



oppiabot bot commented 4 days ago



Hi @jameesjohn,(cc@Showtim3) you need to sign the CLA before you can send in PRs. Learn more about contributing to Oppia by following the guides at <https://github.com/oppia/oppia/wiki/Contributing-code-to-Oppia#setting-things-up>



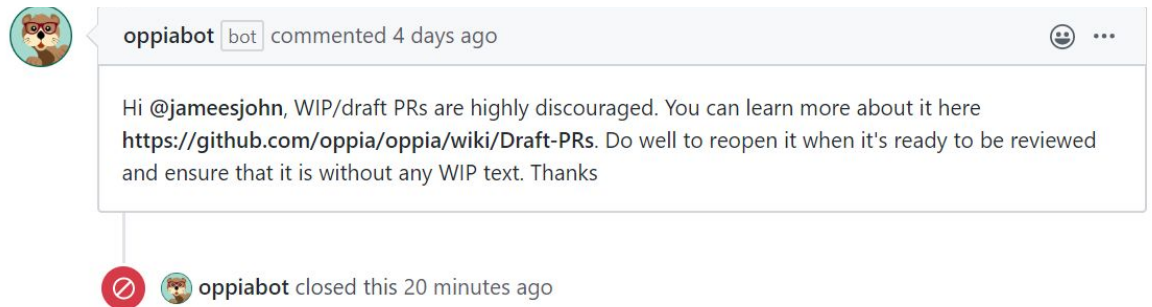
oppiabot closed this 11 minutes ago

b. New contributor that has signed the CLA:

Oppiabot will not do anything if the user has signed the CLA.

2. WIP / Draft Check[11]

Here, Oppiabot checks that the title/explanation(first comment) of the PR does not contain **WIP**. If they contain **WIP**, Oppiabot will comment telling the user that WIP PRs are not allowed, and close the PR. **This check will also be done when the user updates the title of the pull request.**



Oppiabot will also respond with a similar response if the PR is a draft PR.

3. New Code owner check[26]

Check that the PR does not add a new codeowner. Oppiabot will ensure that no new code owner who isn't in a whitelist (provided by project leads) gets added to the .CODEOWNER files.

If one gets added, Oppiabot will mark the PR as requested changes (review the PR) and comment on the PR (cc'ing oppia-core-maintainers) telling the user that he is not allowed to add new code owners.



This check will also be run when the author **pushes a commit** to the PR to ensure that the new codeowner doesn't get added subsequently.

4. PR Template Check [21]

Check that the PR follows the PR template. Specifically, Oppiabot will be checking that the overview section is available and contains the issue number which the PR fixes. If it is a PR that doesn't fix an issue, Oppiabot will check that an explanation is provided in the overview section.

Oppiabot will also check that the checklist section is available and the checklist items are checked.

If any section is missing, Oppiabot comments on the PR telling the user to update the template, providing the link to the recommended description template and assigns the PR to the author.

This check will be redone once the user edits the PR description to ensure compliance.



oppiabot [bot] commented 24 days ago



Hi @jameesjohn, this PR's description does not follow the recommended template. Please update it to follow template in order for it to get reviewed. The recommended template can be found here: <https://github.com/oppia/oppia/wiki/PR-Template>.



oppiabot [bot] assigned jameesjohn 24 days ago

Note: The PR template isn't currently in the wiki, so a page will be created for the template.

The recommended PR template is shown below.

Overview

1. This PR fixes or fixes part of #[fill_in_number_here].
2. This PR does the following: [Explain here what your PR does and why]

Essential Checklist

- ☐ The PR title starts with "Fix #bugnum: ", followed by a short, clear summary of the changes. (If this PR fixes part of an issue, prefix the title with "Fix part of #bugnum: ...".)
- ☐ The linter/Karma presubmit checks have passed locally on your machine.
- ☐ "Allow edits from maintainers" is checked. (See [here](#) for instructions on how to enable it.)
 - This lets reviewers restart your CircleCI tests for you.
- ☐ The PR is made from a branch that's **not** called "develop".

PR Pointers

- Oppiabot will notify you when you don't add a PR_CHANGELOG label. If you are unable to do so, please @-mention a code owner (who will be in the Reviewers list), or ask on [Gitter](#).
- For what code owners will expect, see the [Code Owner's wiki page](#).
- Make sure your PR follows conventions in the [style guide](#), otherwise this will lead to review delays
- Never force push. If you do, your PR will be closed.

5. PR branch check[[16](#)]

If the branch from which the PR is created is *develop* or starts with *release-* or *test-*, Oppiabot will comment on the PR, telling the user to not make PRs from branch names that start with the prefixes listed above and automatically close the PR.



oppiabot bot commented 24 days ago



Hi @jameesjohn, PR's made from the develop/release/test branch is not allowed. Learn more here: <https://github.com/oppia/oppia/wiki/PR-branch>. In the mean time, I'll be closing this. Please make your changes in another branch and send in a PR



oppiabot bot closed this 19 days ago

6. New Job Check [[22](#)]

Here, Oppiabot will check if the PR created, adds a new job (modifies the `job_registry.py` file, `jobs.py` or modifies a file with the name pattern `<name>_jobs_<anothername>.py`). If the PR creates a new job, Oppiabot will mention the server jobs admin in a comment and assign the PR to them. Oppiabot will also remind the PR author to fill the jobs form.



oppiabot bot commented 2 days ago



Hi @seanlip, PTAL at this PR. It adds/modifies a server job. Also, @jameesjohn, please endeavour to fill the server jobs form: <https://goo.gl/forms/XIj00RJ2h5L55XzU2> for the new job to get tested on the test server. Thanks.

7. Critical PR Check[15]

Oppiabot will check the files edited by the PRs. If it contains a file in the **core/storage** folder (models), Oppiabot will add the **critical** label and mention the release coordinator in a comment informing them of the models that got affected. The current release team lead can be found out by checking the **PR CHANGELOG: Release Team** label. The label text will contain the name of the current release team lead. Oppiabot will assign the Release Team lead to ensure that it doesn't get unnoticed.



oppiabot bot commented 2 days ago



Hi @release_coordinator, PTAL at this PR. It modifies the following models: model A, model B.



oppiabot bot assigned release_coordinator 2 days ago

Labelling a Pull Request

User Behaviour: User adds a label to a Pull Request.

Oppiabot Response: Oppiabot responds differently depending on the label that got added.

1. Critical label [15]

Oppiabot will check when this label gets removed from a PR. Oppiabot will check if the label got removed by a whitelisted user (the whitelist will be provided by the release coordinator). If it is removed by someone who isn't in the whitelist, Oppiabot will add it back automatically with a comment stating that the critical label can only be removed by the release coordinator.

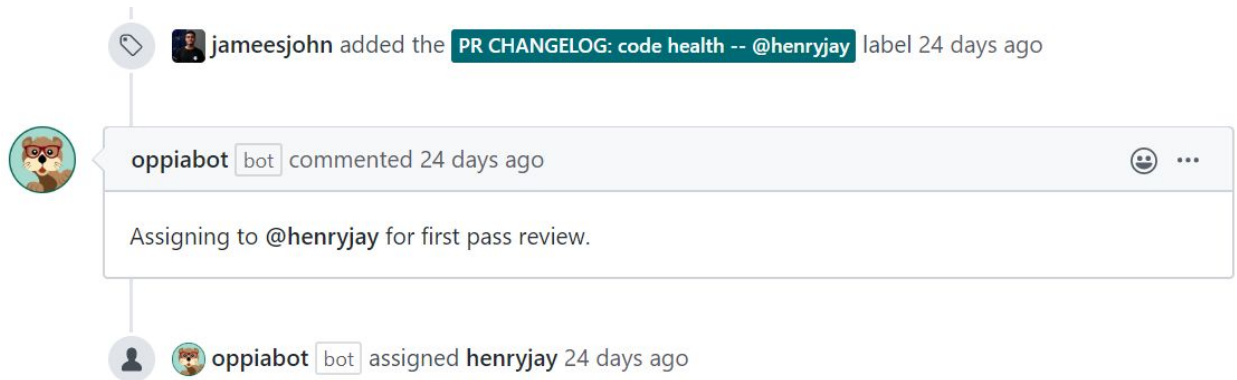
2. Changelog label [5]

• Label present

When a changelog label gets applied, Oppiabot will respond based on the following cases:

- Case1: PR author is the project owner
Oppiabot Response: Nothing
- Case2: PR has review comments (PR has been reviewed initially)
Oppiabot Response: Nothing

- Case3: All other cases
Oppiobot Response: Assign the project owner to PR.

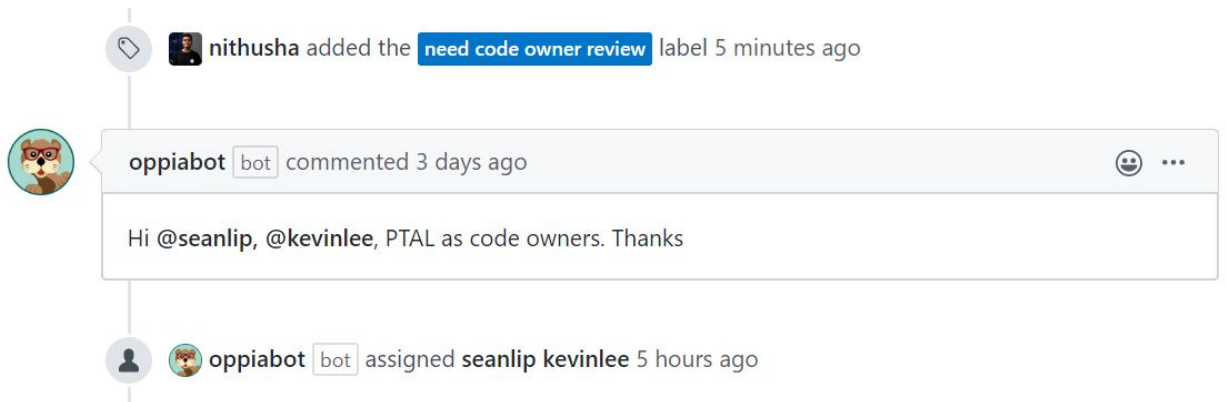


- Label Absent
Oppiobot will check if the author is a new contributor and mention the onboarding team to help with the changelog assignment. If the author is not a new contributor, Oppiobot will assign the author to the PR and mention them to add the appropriate changelog label.

3. Need code owner approval label [6]

A need code owner approval label will be created so users (project owners) will be able to add them to PRs after reviewing.

When a **Need code owners approval** label is added, Oppiobot will go through the list of code owners and check if there are any who haven't given a review. If there are, Oppiobot will assign them to the PR.



4. Other labels [9]

If a label that is not listed above gets added, Oppiobot will check if the label is an issue label.

PR labels have been listed above and can be found [here](#).

If the label is an issue label, Oppiabot will remove the label with a message telling the user that the label should not be added to a PR. The interaction for this will be similar to that found in the [issue case](#)

Force Pushing Updates to a PR [8]

User Behaviour: User force pushes to a PR

Oppiabot Response: Oppiabot will automatically close the PR with a comment stating that it is not allowed to force push to a PR.



oppiabot bot commented 24 days ago



Hi @jameesjohn, force pushing is not allowed as it makes code reviews hard. Learn more here: <https://github.com/oppia/oppia/wiki/Contributing-code-to-Oppia#setting-things-up>. I'll be closing this, please make a new PR with the required changes.



oppiabot bot closed this 19 days ago

Review / Responding to Reviews

User behaviour: User's PR gets reviewed

Oppiabot Response:

When a reviewer requests for changes, Oppiabot will automatically assign the PR to the author for further actions [19].

seanlip requested changes on Feb 7 [View changes](#)

seanlip left a comment Member 😊 ...

Hi @jameesjohn, I took a first pass. PTAL

oppiabot bot assigned jameesjohn 24 days ago

jameesjohn commented 24 days ago 😊 ...

Changes applied. PTAL@seanlip.

oppiabot bot assigned seanlip 24 days ago

User Behaviour: After responding to the reviews, the PR author can mention the reviewer(s) to take a look at the new changes.

Oppiabot Response: Oppiabot noticing this call to the reviewer will automatically assign all the reviewers to the PR[12].

Periodic Checks

Oppiabot will periodically (3 days) check PRs to ensure the following:

1. There are still activities going on in a PR. Activities here include comments, reviews, request for reviews and branch pushes.
 - a. If there has not been any activity from the PR author in 7 days, Oppiabot will comment on the PR notifying the PR author that there has been no activity in the PR for 7 days and that the PR will be closed if there hasn't been any new activity from them after 4 more days[3].
 - b. If there are assigned reviewers who are yet to review and have not responded within 24 hours of their assignment, Oppiabot will ping them to review the PR and also ping oppia-core-maintainers.
 - c. If there are unassigned reviewers who are yet to review, Oppiabot will assign them to the PR.
2. Check that the PR is assigned to at least one person. If not, Oppiabot will assign it to the author of the PR. If the PR author is a new contributor, Oppiabot will comment on the PR mentioning the members of the onboarding team to assign the appropriate reviewer[10].

Merging another PR

Behaviour: Another PR gets merged.

Oppiabot Response:

- In case of a merge conflict, Oppiabot will comment on the affected PRs to inform the author of the conflict and automatically assign the PR author [\[4\]](#).
- If there is no merge conflict, Oppiabot will comment on the PR telling the PR authors to update their branches with the latest update from develop [\[13\]](#).

CI Test Failure

User Behaviour: User's PR fails tests.

Oppiabot Response:

Oppiabot will notify the PR author with the information of the test failures, provide a link to the failed test and assign the author for more actions from them [\[7\]](#).

If the failed checks are frontend tests or lint tests, Oppiabot will advise the user by commenting on the PR to run the tests locally before pushing [\[24\]](#).

Technical Design

Problems with using Github Actions for Oppiabot

All functionalities of Oppiabot through GitHub action depends on being able to react to events and create comments, assign users to issues, etc.

However, if a PR is created from a forked repository (which is the recommended way for contributing to Oppia), the [GitHub token](#) provided has [read-only access](#). Hence, can't be used for modification of PRs(commenting, assigning, etc).

I contacted the Github support and got a response after some back and forth emails:



Yingluo (GitHub Developer Support)

Mar 26, 10:25 PM UTC

Hi James,

Sorry for the misunderstanding on my part. You're right that secrets cannot be passed to the runner when the workflow is triggered from a forked repository.

Unfortunately, there isn't a way to natively get this working. What you could do as an extra step is to create another branch (or ask the owner to do so if you don't have permission) in the parent repository. For example, `dev`. From there, you could open a pull request from your fork into `dev` (note no workflow will run). Once that's done, you can create a second pull request merging `dev` into `master`. This will trigger successfully as the pull request will now be made from the parent repository.

We understand this is not ideal, and we will pass your feedback to the Actions Product team.

In addition, we recommend asking in our GitHub Community Forum as other GitHub users may have found other workarounds:

<https://github.community/t5/GitHub-Actions/bd-p/actions>

Due to this constraint, I have resolved to use probot for the implementation of the new oppiabot PR features. Github actions can, however, be used for Issues checks since it does not have the **forked repository** token constraint.

Architectural Overview

Oppia Bot Using Probot

Oppiabot is structured in a feature module system. Hence for every new feature, a new module is created.

Entry Point:

The entry point to the app is a javascript file, **index.js**. This script is mostly referred to in this section as the dispatcher. It handles the function of listening to events and reacting to these events by calling the handlers provided.

Handlers

The handlers are functions which handle the actions to be carried out when events are triggered.

They are modules with some functions exported to be called by the dispatcher. They may or may not have private functions (depending on the degree of complexity of the feature).

These handlers might in some cases call other handlers (in the case of shared functionality).

Folder Structure

The folder structure after the implementation will be as shown below.

```
oppiabot/  
  lib/  
    someFeatureModule1.js  
    someFeatureModule2.js  
  spec/  
    someFeatureModuleSpec1.js  
    someFeatureModuleSpec2.js  
  node_modules/  
  index.js
```

Oppia bot Using Github actions

Building Oppiabot using GitHub actions will mean reacting to events. A workflow file which is written using YAML syntax is used to specify actions that will be taken when certain events are triggered.

The workflow file will call the actions on Oppiabot.

Oppiabot will have an action.yml file which is similar to the workflow file but has the primary task of providing the entry point to the entire application.

The action.yml is also written using YAML syntax and specifies what will be used in running Oppiabot.

Javascript will be used in writing the actions, so **node** will be specified in action.yml as the runtime. A particular javascript file (main.js) will serve as the entry point and will be called in response to the events

Entry Point

The entry point to the app will be a javascript file, **main.js**.

This file will be in charge of the bootstrapping of the entire app. Bootstrapping the app involves calling the dispatcher with the event that got triggered.

Dispatcher

The dispatcher is a script that works as a router. It will receive the event and call different handlers based on the event that was sent.

Handlers

The handlers are functions which handle reacting to events. The handlers are methods of different classes.

For example, if an issue gets created, the **issues.handleCreate** function will be called to perform the actions.

Folder Structure

```
oppia
├── .github/
│   └── actions/
│       ├── oppiabot/
│       │   ├── src/
│       │   │   ├── issues/
│       │   │   │   ├── featureModule1.js
│       │   │   │   └── featureModule2.js
│       │   └── spec/
│       │       ├── issues/
│       │       │   ├── featureModuleSpec1.js
│       │       │   └── featureModuleSpec2.js
│       └── dispatcher.js
├── main.js
├── action.yml
├── workflows/
│   └── oppiabot.yml
└── index.js
```

Implementation Approach

Issue Checks:

1. Name: Issue Labelled

Status: New feature to be created using GitHub Actions

Implementation

1. Listen on event

The workflow file, `oppiabot.yml` will be modified to listen to the **issues.labelled** events.

2. Dispatch to handler

The dispatcher will be modified to dispatch events which the event name is **"issues"** and the payload action is **labelled** to the **checkIssueLabels.checkLabels** handler.

3. Create Handler

The **checkIssueLabels** module will be created and will contain different functions. The primary function **checkLabels** will get exported to be used in the dispatcher (`dispatcher.js`).

checkLabels:

Gets called with an argument (context) which contains references to the payload (context.payload) and an octokit reference can be created using [@actions/github](#) and the repo token which will be gotten from the workflow via [@actions/core](#).

The label that got added to the issue will be gotten from the payload (context.payload.label), from which the label's name can be gotten via context.payload.label.name.

The check for the label type will be done on the name of the label.

- **Case good-first-issue**

If the label's name is **good-first-issue**, we get the username of the user that added the label from the payload via context.payload.sender.login and check if the username can be found in the `whitelist.goodFirstIssue` array.

If it can't be found, we create a comment via.

Note: The whitelist is a JSON file which is of the structure:

```
{
  "goodFirstIssue" : []
}
```

- **Others**

Check if the label's name matches the PR labels (is dependencies, critical, starts with PR Changelog or PR:).

If it is a PR label, a comment will be created via [octokit.issues.createComment](#).

4. Payload Structure:

```
{
  "action": "labeled"
  "issue": { ... }
  "label": {
    "id": 1913153623
    "node_id": "MDU6TGFiZWwzOTZzMTUzNjIz"
    "url": "https://api.github.com/repos/user/repo/labels/documentation"
    "name": "documentation"
  }
}
```

```

    "color": "0075ca"
    "default": true
    "description": "Improvements or additions to documentation"
  }
  "repository": {...}
  "sender": {
    "login": "username"
    "id": userid
    "node_id": "MDQ6VXNlcjMxMDUyNDg5"
    "avatar_url": "https://avatars1.githubusercontent.com/u/31052489?v=4"
    ...
  }
  "installation": {...}
}

```

Code Snippets

- **Dispatcher**

```

switch (event) {
  case 'issue':
    switch (action) {
      case 'labelled':
        await checkIssuesLabels.checkLabels(context);
        break;

      default:
        break;
    }
    break;

  default:
    break;
}

```

- **Handler**

```

module.exports.checkLabels = async function(context) {
  const token = core.getInput('repo-token');
  const label = context.payload.label;
  const octokit = new GitHub(token);

  if(label.name == 'good-first-issue' &&
  !whitelist.goodFirstIssue.includes(context.payload.sender.login)) {
    octokit.issue.createComment(commentParams)
  } else if(prLabels.includes(label.name) || label.name.startsWith('PR'))
  {
    octokit.issue.createComment(commentParams);
  }
}

```

```

    octokit.issue.deleteLabel(labelParams)
  }
}

```

Tests

This feature can be tested by mocking the context with a sample object that has payload and issue as properties. createComment, deleteLabel will be jasmine spies which will help in providing the return data.

The test will assert that octokit.issue.createComment gets called based on the parameters passed in the payload.

Response Messages

Response message text	Under what circumstances is this response triggered?	Is this a periodic check or a check triggered by some activity? If it is triggered by an activity, what is the expected response time?
Hi @user, thanks for proposing this as a good first issue. Looping in @showtim3 to confirm, removing the tad until he does so.	A good-first-issue label is added to an issue	Triggered by activity. Response time is immediate.
Hi @user, changelog labels should not be used on issues, I'm removing the label. You can learn more about labels here: <link to wiki page on labels>	A changelog label is added to an issue	Triggered by activity. Response time is immediate

APIs

No external API is required here.

Structure

```

oppiabot/
  src/
    issues/
      checkIssuesLabel.js
  spec/
    issues/
      checkIssuesLabelSpec.js

```

2. Name: Assigned to Issues

Status: New feature to be implemented using Github Actions

Implementation:

1. Listen on event

The workflow file, `oppiabot.yml` will be modified to listen to the **issues.assigned** events.

2. Dispatch to handler

The dispatcher will be modified to dispatch events which the event name is **"issues"** and the payload action is **assigned** to the **checkIssueAssignees.checkAssignees** handler.

3. Create Handler

The **checkIssueAssignees** module will be created and the **checkAssignees** function will be exported from it.

checkAssignees will receive the context argument which contains the payload from which the data of the issue affected can be retrieved.

The user that got assigned be gotten from the payload, and using the google sheets API, will check if the user is found in the sheet.

Looping through the sheet, we check if the user's GitHub username can be found in the sheet.

If the user isn't found, a comment will be created in the issue via [octokit.issues.createComment](#) and the user will be [unassigned](#).

4. Payload Structure

```
{
  "action": "assigned",
  "issue": {...},
  "assignee": {
    "login": "jameesjohn",
    "id": 31052489,
    ...
  },
  "repository": {...},
  "sender": {...},
  "installation": {...}
}
```

Code Snippets

• Dispatcher

```
robot.on('issues.assigned', async context => {
  await checkIssueAssignees.checkAssignees(context);
});
```



```
});
```

- **Handler**

```
module.exports.checkAssignees = async function(context) {
  const issue = context.payload.issue;
  const token = core.getInput('repo-token');
  const octokit = new GitHub(token);

  const assignee = context.payload.assignee.login;
  //get rows from sheet api
  for (var row in rows) {
    var rowUserName = rows[row][0];
    if (rowUserName.toLowerCase() === assignee.toLowerCase()) {
      hasUserSignedCla = true;
      break;
    }
  }
  if(!hasUserSignedCla) {
    octokit.issue.createComment(commentParams)
    octokit.issue.removeAssignees(params)
  }
}
```

Tests

To test this feature, mock the context as in the previous case, assert that calls to the sheet apiForSheetModule has been made, and assert that octokit.issue.createComment and octokit.issue.removeAssignees got called.

Response Messages

Response message text	Under what circumstances is this response triggered?	Is this a periodic check or a check triggered by some activity? If it is triggered by an activity, what is the expected response time?
Hi @user, you need to sign the CLA before you can get assigned to issues. Follow this link to sign the CLA <link to CLA>.	A User gets assigned to an issue when they haven't signed the CLA.	Triggered by activity. Response time is immediate.

APIs

Google APIs will be used. The set up for this is already done as the API is currently in use.

Structure

```
oppiabot/  
  src/  
    issues/  
      checkIssueAssignees.js  
  spec/  
    issues/  
      checkIssueAssigneesSpec.js
```

PR Checks:

1. Name: PR Created

Status: New feature to be implemented using Probot

Implementation:

1. Listen on event/dispatch to handler

Currently, there is an entry in the dispatcher which handles the **pull_request.opened** and **pull_request.reopened** event. More handlers will be added to the already called handlers.

The handlers to be added include:

- `checkWIPPullRequests.checkWIP`
- `checkPullRequestTemplate.checkTemplate`
- `checkPullRequestBranch.checkBranch`
- `checkPullRequestJobs.checkForNewJob`
- `checkCriticalPullRequest.checkIfCritical`

2. Create handlers

- The **checkWIPPullRequests** module will be created which exports the **checkWIP** function. The **checkWIP** function receives the context which contains the github payload as an argument, from which we can get the `pull_request` title and the comments on the pull request.

The PR title will be checked for the occurrence of *WIP*.

If found, a comment will be created on the issue via

[context.github.issues.createComment](#). We are using `issues.createComment` here because using `pulls.createComment` expects the commit ID and the file which is triggering the comment (as in a code review comment) but that is not what we want here.

If WIP wasn't found in the title, a request will be made to get a list of comments on the PR via [context.github.issues.listComments](#). The comments are ordered according to how they are created, so the first comment on the list will be the description comment. This comment will also be searched for the presence of WIP. And if found, a comment will be created and PR closed via [context.github.pulls.update](#) setting the state to **closed**.

- The **checkPullRequestTemplate** module will be created and will export the **checkTempate** method. This method will get the pull_request data from the payload and load the first comment using a similar manner as discussed above. The comment body(string) will be split into 2 arrays based on the 2 required sections (Overview and checklist). And the two arrays will be searched for the occurrence of the required texts.
- The **checkPullRequestBranch** module will be created and will export the **checkBranch** function. This function will get the branch from which the PR was made from the payload via payload.head.ref. The branch can now be compared if it starts with **release-** or **test-** or is **develop**
- The **checkPullRequestJobs** module will be created and will export the **checkForNewJob** function. The exported function as expected will get called providing the context as an argument. From the context, we can access octokit (context.github) and get the list of files modified in the PR via **context.github.pulls.listFiles**. Looping through the files modified in the PR, we'll check if the job registry (jobs_registry.py) any job file is modified. Job files have names looking like <name>_jobs_<anothername>.py. Using this, we can compare with all the files modified to find a match. Once a match is found, we carry out the appropriate action: create a comment to ping the server jobs admins and assign them to the PR.
- The **checkCriticalPullRequest** module will be created and will export the **checkIfCritical** method. This method using the payload that gets sent to it will get the files modified in the PR and check if any file from the core/storage folder got modified. If that is the case, oppiabot will check if the critical label has already been added. If not, will add the [critical label](#) to the PR and create a comment pinging the release team lead. The Release team lead can be gotten by getting the release team changelog label.

3. Payload Structure

```
{  
  "action": "opened",  
  "number": 2,  
  "pull_request": {
```

```

        "url":
"https://api.github.com/repos/Codertocat/Hello-World/pulls/2",
        "id": 279147437,
        "node_id": "MDExOlB1bGxSZXF1ZXN0Mjc5MTQ3NDM3",
        "html_url": "https://github.com/Codertocat/Hello-World/pull/2",
        "diff_url":
"https://github.com/Codertocat/Hello-World/pull/2.diff",
        "number": 2,
        "state": "open",
        "locked": false,
        "title": "Update the README with new information.",
        "user": {
            // user data
        },
        "body": "This is a pretty simple change that we need to pull
into master.",
        "assignees": [

        ]
    }
}

```

Code Snippets

- **Dispatcher**

```

robot.on('pull_request.opened, async context => {
    await checkWIPPullRequests.checkWIP(context);
});

```

- **Handler**

```

module.exports.checkWIP = async function(context) {
    var pullRequest = context.payload.pull_request;
    var pullRequestNumber = pullRequest.number;

    if(pullRequest.title.toUpperCase().includes('WIP') ||
    (pullRequest.body.toUpperCase().includes('WIP'))){
        context.issues.createComment(commentParams)
        const updateParams = {
            ...,
            state: closed
        }
        context.pullRequest.update(updateParams);
    }
}

```

Tests

To test this feature, mock the context as in the previous case and assert that `context.issue.createComment` and `context.pullRequest.update` got called.

Response Messages

Response message text	Under what circumstances is this response triggered?	Is this a periodic check or a check triggered by some activity? If it is triggered by an activity, what is the expected response time?
Hi @user, WIP PRs are highly discouraged. Learn more here: <link to wiki>. In the meantime, this PR will be closed.	PR title contains WIP	Triggered by activity. Response time is immediate.
Hi @user, you are not allowed to add new code owners. Cc @oppia-core-maintainers	New code owner added who isn't whitelisted.	Triggered by activity. Response time is immediate.
Hi @user, this PR does not follow the recommended template. Please update it as required. Closing this PR until the template is followed.	PR doesn't follow the recommended template	Triggered by activity. Response time is immediate.
Hi @user, PR's made from the develop/release/test branch is not allowed. I'll be closing this. Please make your changes in another branch and send in a PR.	PR is made from the develop/release/test branch.	Triggered by activity. Response time is immediate.
Hi @server_job_admin, PTAL at this PR. It modifies the server jobs	PR modifies jobs file.	Triggered by activity. Response time is immediate.
Hi @release_coordinator, PTAL at this PR. It modifies the following models: model A, model B.	PR modifies models	Triggered by activity. Response time is immediate.

APIs

No external APIs would be used here.

Structure

All modules will be kept in the **lib/** folder and their respective tests in the **spec/** folder.

```
lib/  
  checkWIPPullRequests.js  
  checkPullRequestTemplate.js  
  checkPullRequestBranch.js  
  checkPullRequestJobs.js  
  checkCriticalPullRequest.js  
  
spec/  
  checkWIPPullRequestsSpec.js  
  checkPullRequestTemplateSpec.js  
  checkPullRequestBranchSpec.js  
  checkPullRequestJobsSpec.js  
  checkCriticalPullRequestSpec.js
```

2. Name: PR Labelled

Status: New feature to be implemented using Probot

Implementation:

1. Listen on event/dispatch to handler

Currently, oppiabot listens on the **pull_request.labelled** event but dispatches to one handler. That will be updated to listen on **pull_request.labelled** and **pull_request.unlabelled** events. More handlers will be added to increase the checks to be carried out.

2. Create Handler

There is an already existing **checkPullRequestsLabels** module. This module will be updated to add new checks. The following functions will be created:

- **checkCriticalLabel**

Check that a PR which modifies a file in core/storage always has the critical label. Hence, when any label gets added or removed, the critical pull request check will be carried out and if the PR should have the critical label, but doesn't have it, will check the user that performed the action. If the user is not the release coordinator or in the release coordinator's whitelist, the label will be added back.

The names for the whitelist will be gotten from

<https://github.com/oppia/oppia/wiki/Release-Schedule#release-coordinator-s-and-qa-coordinators-for-upcoming-releases> and will be stored in a whitelist.json file under the property name: **release**.

- **checkIssuesLabel**

Check that a PR does not contain labels that should be used on PRs. It will go through the list of labels in a PR and ensure that only PR labels can be found.

The newly added functions will be exported so they can be added to the dispatcher.

3. Payload structure

```
{
  "action": "labeled",
  "number": 8,
  "pull_request": {...},
  "label": {
    "id": 1913153622,
    "name": "bug",
    ...
  },
  "repository": {...},
  "sender": {...},
}
```

Code Snippets

```
module.exports.checkLabels = async function(context) {
  var pullRequest = context.payload.pullRequest;
  var label = context.payload.label;

  if (label.name === 'critical' &&
    !whitelist.critical.includes(context.payload.sender.login)) {
    context.issue.createComment(commentParams);
    context.issue.addLabel(labelParams);
  } else if (!(prLabels.includes(label.name) || label.name.startsWith('PR'))) {
    context.issue.createComment(commentParams);
    context.issue.deleteLabel(labelParams);
  } else if (label.name.toLowerCase().includes('changelog')) {
    let index = label.name.indexOf('@');
    let projectOwner = label.name.substring(index + 1);
    if (pullRequest.user.login !== projectOwner && pullRequest.reviews.length === 0)
    {
      context.github.issues.addAssigneesToPullRequest(assigneeParams);
      context.issue.createComment(commentParams);
    }
  }
};
```

Tests

To test this feature, mock the context as in the previous case and assert that the mocked functions got called.

Response Messages

Response message text	Under what circumstances is this response triggered?	Is this a periodic check or a check triggered by some activity? If it is triggered by an activity, what is the expected response time?
Hi @user, critical labels are not allowed to be removed. Learn more about labels at <link to wiki>	User not whitelisted removes the critical label	Triggered by activity. Response time is immediate.
Assigning to project owner @projectOwner	User adds changelog label	Triggered by activity. Response time is immediate
Hi @user, the <label name> labels should not be used on PRs, I'm removing the label. You can learn more about labels here: <link to wiki page on labels>	User adds a PR label	Triggered by activity. Response time is immediate

APIs

No external API used.

Structure

```
lib/  
  checkPullRequestLabels.js  
spec/  
  checkPullRequestLabelsSpec.js
```

3. Name: Force Push

Status: New feature to be added using probot

Implementation:

1. Listen on event/dispatch to handler

The event to be listened to here is **push**. The dispatcher will be modified to listen to the push event and call the **handleBranchPush.checkForcePush** function to react to the event.

2. Create Handler

The **handleBranchPush** module will be created and will export the **checkForcePush** function. The payload that will be sent to the

checkForcePush function contains a **forced** property which is boolean and shows if the push was a force push or not. **checkForcePush** will check the forced property and if it is **true**, then the branch was force pushed and will close the PR. The associated PR can also be gotten from the payload. The PR will be updated by setting its state to **closed** and then a comment will be created on the PR as required.

3. Payload structure

Code Snippets

```
module.exports.checkForcePush = async function(context) {  
  if(context.payload.forced === true) {  
    context.issue.createComment(commentParams);  
    context.pullRequest.update(closePRParams);  
  }  
};
```

Tests

To test this feature, mock the context as in the previous case and assert that the `context.issue.createComment` and `context.pullRequest.update` functions got called.

Response Messages

Response message text	Under what circumstances is this response triggered?	Is this a periodic check or a check triggered by some activity? If it is triggered by an activity, what is the expected response time?
Hi @user, force pushing is not allowed as it makes code reviews hard. Learn more about this here <link to wiki>. I'll be closing this. Please make a new PR with the required changes.	User force pushed to a branch	Triggered by activity. Response time is immediate.

API

No external API was used.

Structure

```
lib/  
  handleBranchPush.js  
spec/  
  handleBranchPushSpec.js
```

4. Name: PR Reviewed

Status: New feature to be added using probot

Implementation:

1. Listen on event/dispatch to handler

When a PR gets reviewed, the **pull_request_reviewed.submitted** event gets triggered. The dispatcher(index.js) will be modified to listen on the **pull_request_review.submitted** event and call the **handlePRReview** module.

2. Create handler

The **handlePRReview** module will be created and will export the **prReviewed** function. The function will check the payload and get the state of the review (commented, approved or request changes). If the state is **requested changes**, the PR author will get assigned to the PR.

3. Structure Payload

```
{
  "action": "submitted",
  "review": {
    "id": 237895671,
    "node_id": "MDE3OlB1bGxSZXF1ZXN0UmV2aWV3MjM3ODk1Njc=",
    "user": {
      // user data
    },
    "body": null,
    "commit_id": "ec26c3e57ca3a959ca5aad62de7213c562f8c821",
    "submitted_at": "2019-05-15T15:20:38Z",
    "state": "commented",
    "html_url": ""
  }
  "pull_request": {
    // PR data
  },
}
```

Code Snippets

```
module.exports.prReviewed = async function(context) {
  const review = context.payload.review;
  const pullRequest = context.payload.pull_request;
  if(review.state === 'requested changes') {
    context.github.issue.createComment(commentParam);
    context.github.pullRequest.addAssignees(pullRequest.user.login);
  }
};
```

Tests

To test this feature, mock the context as in the previous case and assert that the `context.issue.createComment` and `context.pullRequest.addAssignees` functions got called.

Response Messages

Response message text	Under what circumstances is this response triggered?	Is this a periodic check or a check triggered by some activity? If it is triggered by an activity, what is the expected response time?
Assigning @user to respond to reviews.	User's PR got a "changes requested" review.	Triggered by activity. Response time is immediate.

API

No external API was used.

Structure

```
lib/  
  handlePRReview.js  
spec/  
  handlePRReviewSpec.js
```

5. Name: Respond to Review

Status: New feature to be added using probot

Implementation:

1. Listen on event/dispatch to handler

The event that gets triggered here is **issues_comment.created**. The dispatcher will get updated to listen for the **issues_comment.created** event and the **handlePRReview.assignReviewer** handler will get called.

2. Create Handler

The **assignReviewer** function will be called with the payload, and we can get the body of the comment that got created. If the comment contains the string "**PTAL @**", the function will send a request to assign the user with the username that was gotten from the comment after the **@**.

This action is carried out in the **handlePRReview** module because it is related to responding to reviews.

3. Payload Structure

```
{
  "action": "created",
  "issue": {
    "url":
"https://api.github.com/repos/Codertocat/Hello-World/issues/1",
    "repository_url":
"https://api.github.com/repos/Codertocat/Hello-World",
    "id": 444500041,
    "node_id": "MDU6SXNzdWU0NDQ1MDAwNDE=",
    "number": 1,
    "title": "Spelling error in the README file",
    "user": {
      // user data
    },
    "labels": [
    ]
  },
  "comment": {
    ...
    "body": "You are totally right! I'll get this fixed right
away."
  }
}
```

Code Snippets

```
module.exports.assignReviewer = async function(context) {
  const comment = context.payload.comment.body;
  const pullRequest = context.payload.issue;
  if(comment.toLowerCase().includes('ptal @')) {
    let index = comment.indexOf('@');
    let reviewer = comment.substring(index + 1);
    context.github.issue.createComment(commentParam);
    context.github.pullRequest.addAssignees(reviewer);
  }
};
```

Tests

To test this feature, mock the context as in the previous case and assert that the `context.issue.createComment` and `context.pullRequest.addAssignees` functions got called.

Response Messages

Response message text	Under what circumstances is this response triggered?	Is this a periodic check or a check triggered by some activity? If it is triggered by an activity, what is the expected response time?
Assigning @reviewer to review.	User (pr author) asks (comment) the reviewer for a review.	Triggered by activity. Response time is immediate.

API

No external API was used.

Structure

```
lib/
  handlePRReview.js
spec/
  handlePRReviewSpec.js
```

6. Name: Handling Merging of PRs

Status: Existing feature to be improved

Implementation:

1. Listen on event/dispatch to handler

The event triggered here is `pull_request.synchronize`. The dispatcher calls the **`checkMergeConflictsModule.checkMergeConflictsInPullRequest`** (existing functionality).

2. Create Handler

The handler to be used here has already been created and will be modified to add new functionality. Currently, oppiabot checks for merge conflicts and pings the user to fix the merge conflict. The function will be updated to also assign the user to the PR so the user knows that an action is needed from him.

A new function (`informUserOfChanges`) will be created to handle informing the user of the changes in develop (in cases where there are no merge conflicts).

This new function will only be called if there is no merge conflict and will ping the user(PR author) of the new change in the base branch, and asking them to update their branch.

3. Payload Structure

No payload will be gotten here since this is a periodic check

Code Snippets

```
module.exports.informUserOfChanges = async function(context, pullRequest) {  
  context.github.issue.createComment(commentParam);  
  context.github.pullRequest.addAssignees(pullRequest.user.login);  
};
```

Tests

To test this feature, mock the context as in the previous case and assert that the **context.issue.createComment** and **context.pullRequest.addAssignees** functions got called.

Response Messages

Response message text	Under what circumstances is this response triggered?	Is this a periodic check or a check triggered by some activity? If it is triggered by an activity, what is the expected response time?
Hi @user, please update your branch to get the latest changes from develop	A PR gets merged to the develop branch	Periodic check. Response time is immediate.

API

No external API was used.

Structure

```
lib/  
  checkMergeConflictsModule.js  
spec/  
  checkMergeConflictsModuleSpec.js
```


7. Name: Handling CI Tests

Status: New feature to be created using Probot

Implementation:

1. Listen on event/dispatch to handler

The event to be listened on here is **check_suite.completed**. The dispatcher will call the **ciChecks.handleFailure** function passing the context when the **check_suite.completed** event occurs.

2. Create Handler

The ciChecks module will be created and the **handleFailure** function will be exported. The **handleFailure** method will use the payload to get the **check_suite.conclusion** value which can either be success, failure, neutral, cancelled, timed_out, action_required or stale. If the value is failure, cancelled or timed_out, handleFailure will create a comment on the PR informing the user (PR author) of the failure and also add a link to the failed test which is also gotten from the payload.

3. Payload Structure

```
{
  "action": "completed",
  "check_suite": {
    "id": 118578147,
    "node_id": "MDEwOkNoZWNRU3VpdGUxMTg1NzgxNDc=",
    "status": "completed",
    "conclusion": "success",
    "url":
    "https://api.github.com/repos/Codertocat/Hello-World/check-suites/118578147",
    "before": "6113728f27ae82c7b1a177c8d03f9e96e0adf246",
    "after": "ec26c3e57ca3a959ca5aad62de7213c562f8c821",
    "pull_requests": [
      // PR data
    ],
  },
}
```

Code Snippets

```
module.exports.handleFailure = async function(context) {
  const checkSuite = context.payload.check_suite;
  if (checkSuite.conclusion === 'failure') {
    context.github.issues.createComment(commentParams);
    context.github.pullRequest.addAssignees(checkSuite.pull_requests[0].user.login)
  }
};
```

Tests

To test this feature, mock the context as in the previous case and assert that the **context.issue.createComment** and **context.pullRequest.addAssignees** functions got called.

Response Messages

Response message text	Under what circumstances is this response triggered?	Is this a periodic check or a check triggered by some activity? If it is triggered by an activity, what is the expected response time?
Hi @user, there are some failed checks in your latest push. PTAL <link to check>	PR fails test	Triggered by activity. Response time is immediate.

API

No external API was used.

Structure

```
lib/  
  ciChecks.js  
spec/  
  ciChecksSpec.js
```

8. Name: Periodic Checks

Status: New feature to be created using Probot

Implementation:

1. Create Scheduler

Using [probot-scheduler](#), we can create a scheduler to be run every 3 days.

2. Listen on event/dispatch to handler

The dispatcher will be updated to listen to the **schedule.repository** event. And call the **periodicChecks.assignReviewers** function.

3. Create Handler

The **periodicChecks** module will be created and the **assignReviewers** function will be created and exported. Using **context.github** (octokit), all the PRs will be fetched and their comments, reviews alongside.

Going through the list of reviewers all users who are to review the PR but have left no review comment will be pinged to review and assigned.

Oppiabot will also go through the assignees and if there are any reviewers who haven't given a review, Oppiabot will ping them in a comment, and also ping the oppia-core-maintainers.

4. Payload Structure

No payload since it is a periodic check.

Code Snippets

```
module.exports.assignReviewers = async function(context) {
  var pullRequestsPromiseObj = await context.github.pullRequests.getAll(
    context.repo({per_page: 60}));

  var openPullRequests = pullRequestsPromiseObj.data;
  openPullRequests.forEach(PR => {
    PR.reviewers.forEach(reviewer => {
      const reviews = PR.reviews.filter(reviewer => review.user.login ===
reviewer.login);
      if (reviews.length === 0) {
        context.github.pullRequest.addAssignees(reviewer.login);
        context.issues.createComment(commentParams);
      }
    });

    PR.assignees.forEach(assignee => {
      if (PR.reviewers.include(assignee)) {
        const reviews = PR.reviews.filter(reviewer => review.user.login ===
assignee.login);
        if (reviews.length === 0) {
          context.issues.createComment(commentParams);
        }
      }
    });
  });
};
```

Tests

To test this feature, mock the context as in the previous case and assert that the **context.issue.createComment** and **context.pullRequest.addAssignees** functions got called.

Response Messages

Response message text	Under what circumstances is this response triggered?	Is this a periodic check or a check triggered by some activity? If it is triggered by an activity, what is the expected response time?
Hi @user, there has been no update on this PR. It will be marked as stale if there is no update in 4 days time.	No PR update from the author after 7 days	Periodic check. Response time is immediate.
Hi @reviewer, PTAL. cc @oppia-core-maintainers	Assigned reviewer is yet to review a PR	Periodic check. Response time is immediate.

API

No external API was used.

Structure

```
lib/
  periodicChecks.js
spec/
  periodicChecksSpec.js
```

Testing Strategy

The following will be required for testing each feature.

1. Sample Payload. This will be a JSON file, and will be stored in the oppiabot/fixtures directory.
2. Mocked octokit data. This is an object which will contain mocks of the octokit methods that were used in the feature. For example, if a feature creates a comment via **octokit.issues.createComment**, then the octokit data would look like:

```
{
  issues: {
    createComment: jasmine.createSpy('createComment').and.returnValue({
      params: {
        // comment data
      }
    }),
  },
}
```

3. Mock the probot implementation by providing the auth function which will resolve with the mock octokit data.

```
robot.auth = () => Promise.resolve(octokitData);
```

4. Receive the sample payload which can be gotten from oppiabot/fixtures via
`robot.receive(samplePayload);`
5. Assert that our mocked implementation from the spies that got created in step 2 got called.
`expect(octokitData.createComment).toHaveBeenCalled();`

Milestones

Milestone 1

All issue-related checks and the following "PR creation" checks are fully operational:

- WIP PR checks
- PR branch name check
- PR job check

No.	Description of PR	Prereq PR numbers	Target date for PR submission	Target date for PR to be merged
1.1	Issue labelled checks		June 3rd	June 7th
1.2	Issue assigned checks		June 8th	June 12th
1.3	WIP PR checks		June 13th	June 17th
1.4	PR branch check		June 18th	June 24th
1.5	PR job check		June 24st	June 28th

Milestone 2

Key Objective: All Oppiabot checks that should run on PR creation, PR labelling, and force pushes are fully operational.

No.	Description of PR	Prereq PR numbers	Target date for PR submission	Target date for PR to be merged
2.1	PR template check		July 3rd	July 8th

2.2	New code owner check		July 7th	July 12th
2.3	Critical PR check		July 13th	July 16th
2.4	PR labelled checks		July 17th	July 21st
2.5	Force push check		July 21st	July 26th
...				

Milestone 3

Key Objective: All Oppiabot checks that should run on PR reviews, PR review comments, merge conflicts, and CI result publication are fully operational. In addition, the following periodic checks are implemented: PR's aren't stale, issues are associated with a project.

No.	Description of PR	Prereq PR numbers	Target date for PR submission	Target date for PR to be merged
3.1	PR reviewed		July 30th	August 4th
3.2	Respond to review		August 5th	August 9th
3.3	Handle Merge conflict		August 10th	August 14th
3.4	CI checks		August 18th	August 24th
3.5	Periodic checks		August 24th	August 29th
...	Clean up any remaining tasks		August 29th	August 31st

Optional Sections

Future Work

Adding CI tests to be a part of the workflow. Currently, tests in Oppiabot repository don't run on every pull request created. This increases the possibility of bugs on the code. So adding this will help new contributors make changes without breaking the codebase.

Additional Project-Specific Considerations

Labels to be created

- Need code owners approval

- PR: don't merge - NEW CODE OWNER

Documentation Changes

- Update the wiki to show how labels should be used.
- Update the wiki to include notes about WIP and Draft PRs.