Tithonus

The Code Review Quality Checker

User Manual (v1.0.0)

Team: Ziya Mukhtarov, Mokhlaroyim Raupova, Javid Baghirov, Mannan Abdul, Osama Tanveer, Haluk Altunel, Eray Tüzün

GitHub App: https://github.com/apps/tithonus-bilkent

April 17, 2022

Table of Contents

1. Installation

The following steps can be used to install Tithonus so that you too, can start improving the quality of your code reviews!

a. Sign into the GitHub account that you plan to use with Tithonus and then, in a separate tab, open the following link: https://github.com/apps/tithonus-bilkent
b. Once you have clicked the link, you should see a similar screen to the one below

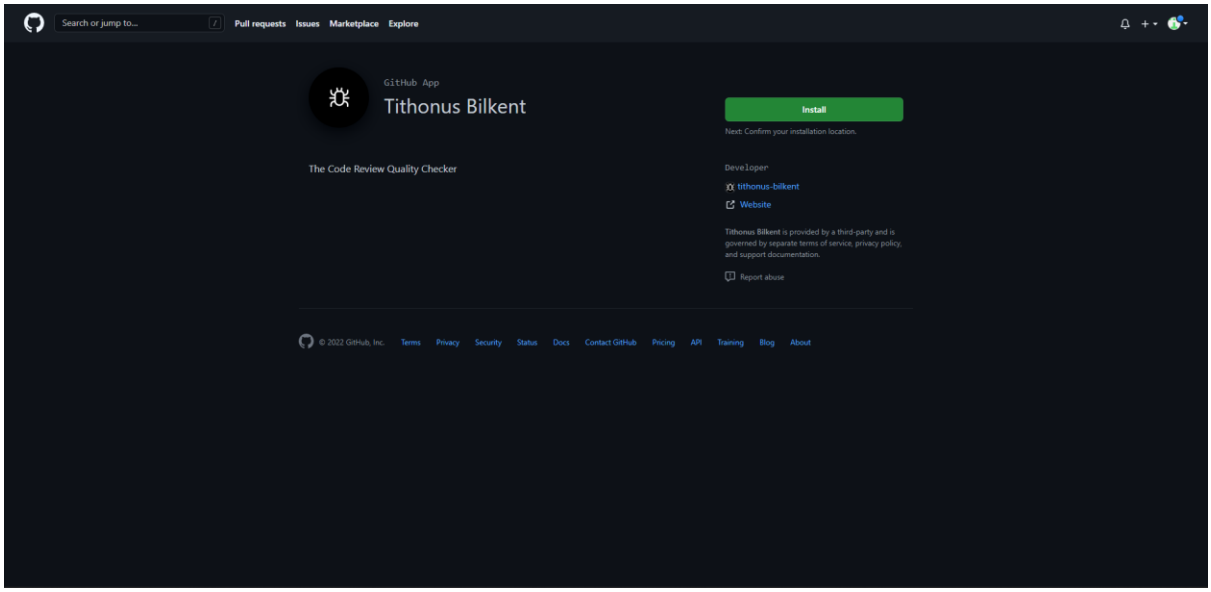

Figure 1 - install Tithonus Bilkent.

c. Click on the install button to access the following screen:

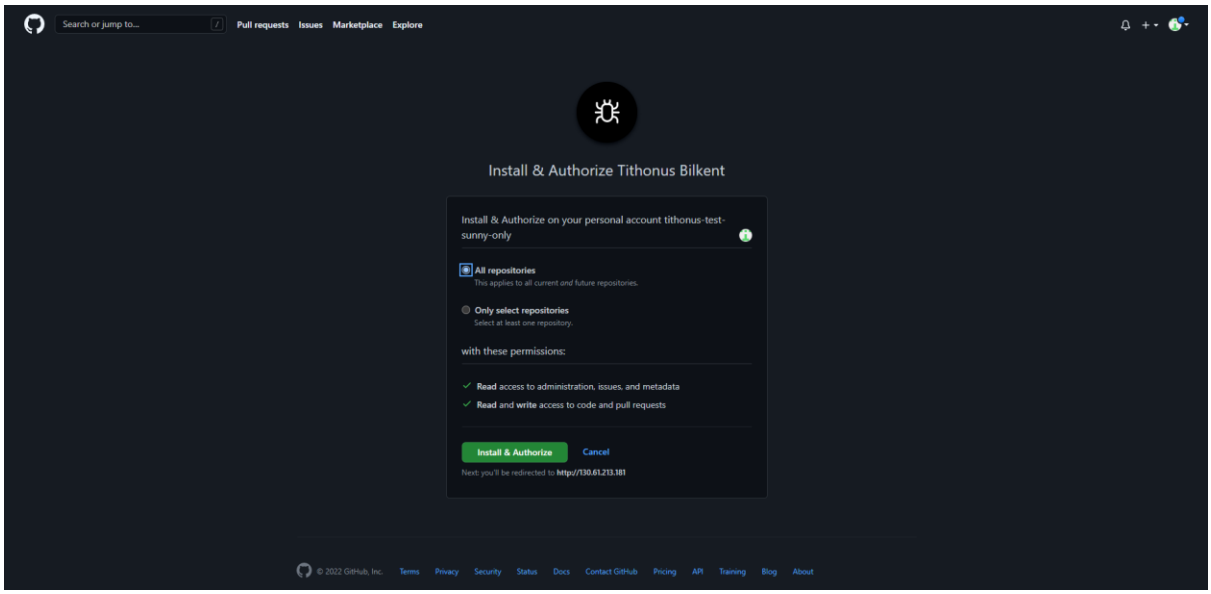

Figure 2 - Choose where to install Tithonus Bilkent.

d. Here, you can choose to install the application in all your current and future repositories or a select repository. You may select the option that suits your use case.
e. Once you click the install & authorize button, you will be taken to the Tithonus application and will be able to see your repository list

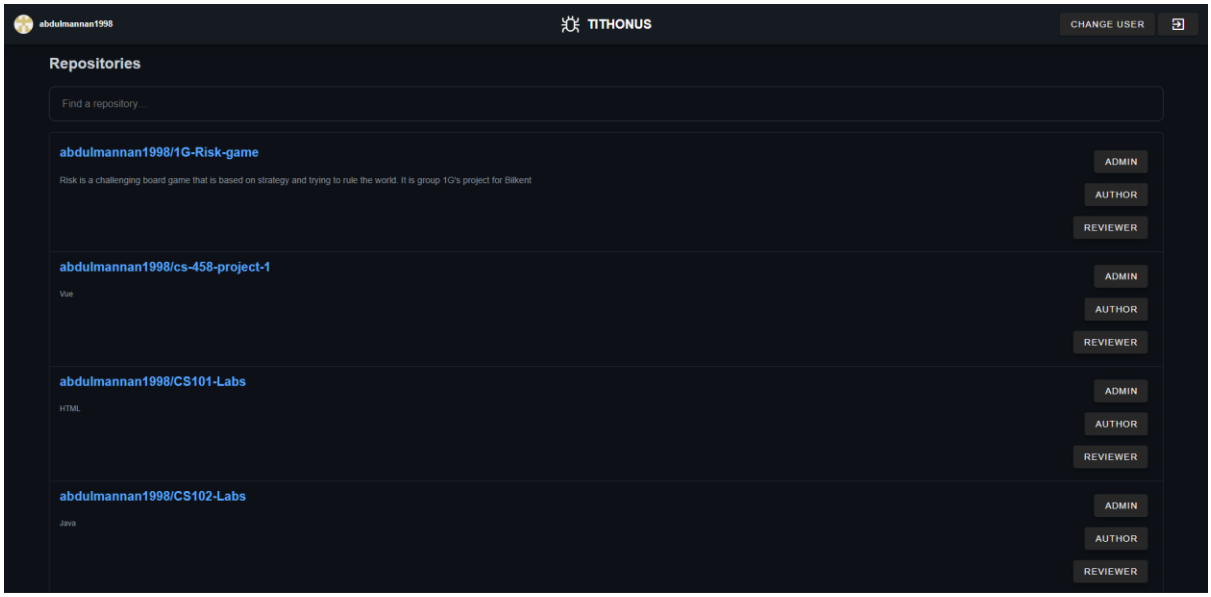

Figure 3 - Repository List after App Installation

2. Overview

2.1 Authorization

a. When starting the application, click on the Login with GitHub button to get a GitHub token.

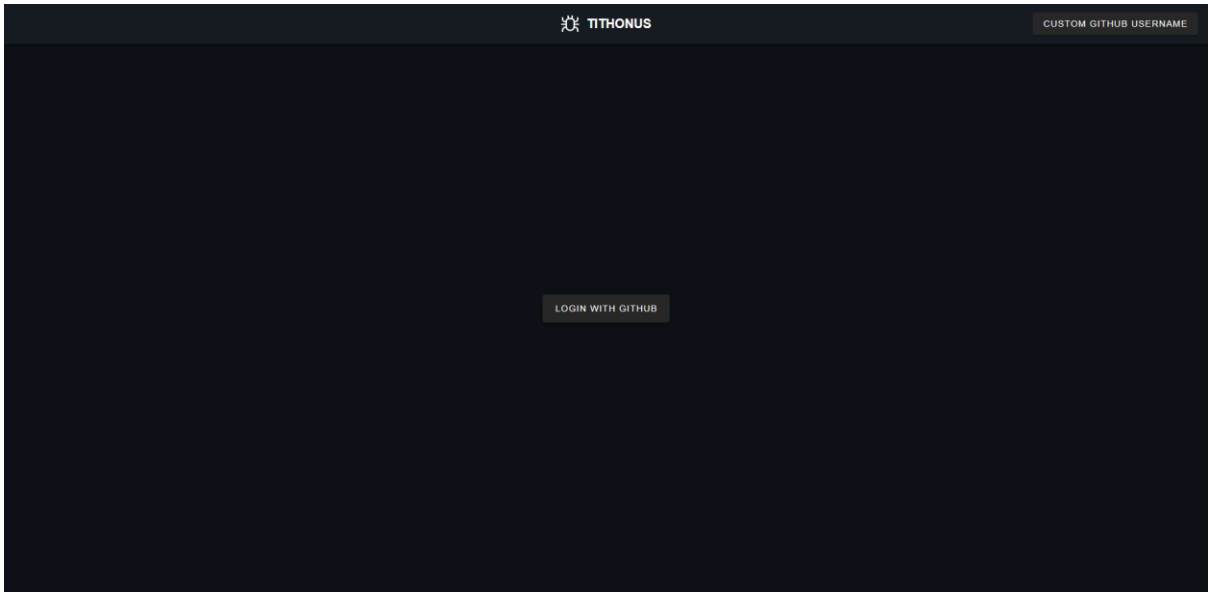

Figure 4 – Login with GitHub

b. The previous step will take you to the Repository List page, where you will be able to see the list of all your repositories. You can enter any of your repositories as an Author, a Reviewer, or an

Admin by clicking the appropriate button beside the repository listed, as long as you have the proper permission.

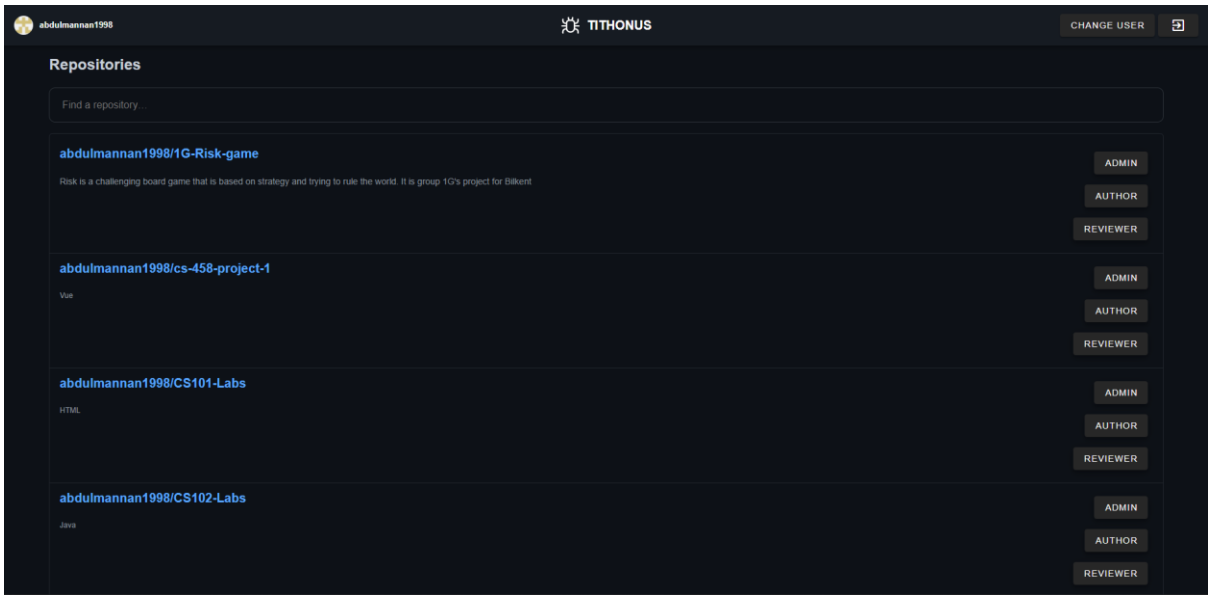

Figure 5 - Repository List page

c. Regardless of the role you enter, you will be taken to the Pull Requests page.

Figure 6 - Pull Requests page

2.2 Change User

a. A user may choose to access Tithonus with a different account. This can be done using the change user button in the top right corner of the Repository List page.

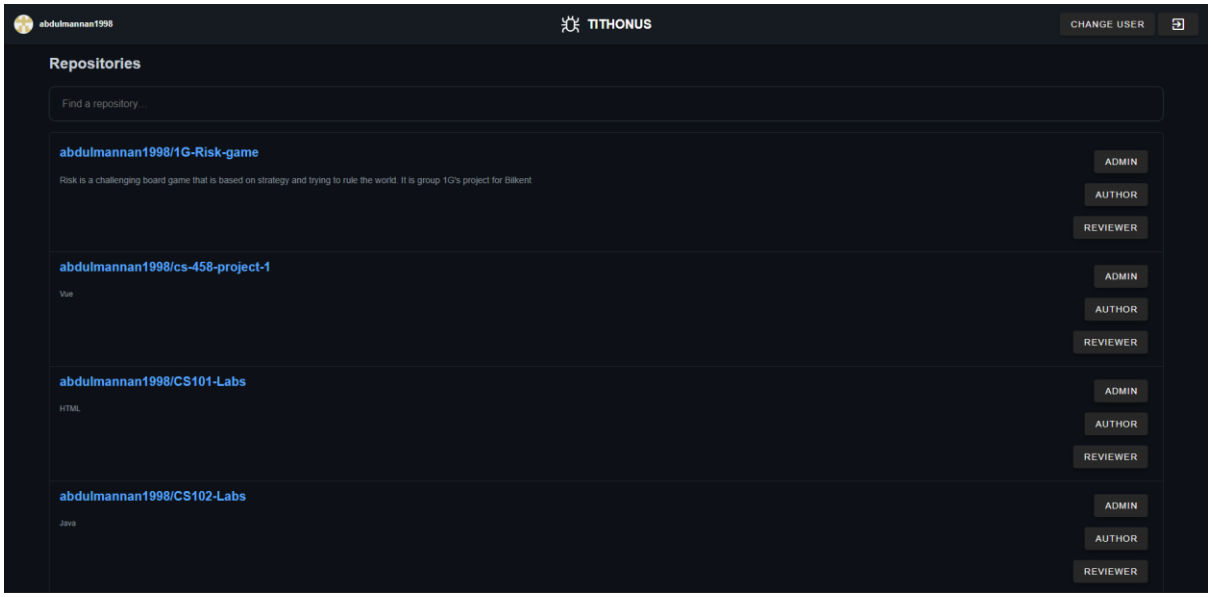

Figure 7 - Change User button on Repository List page

b. The user can click this button and enter their GitHub username in the text field that appears.

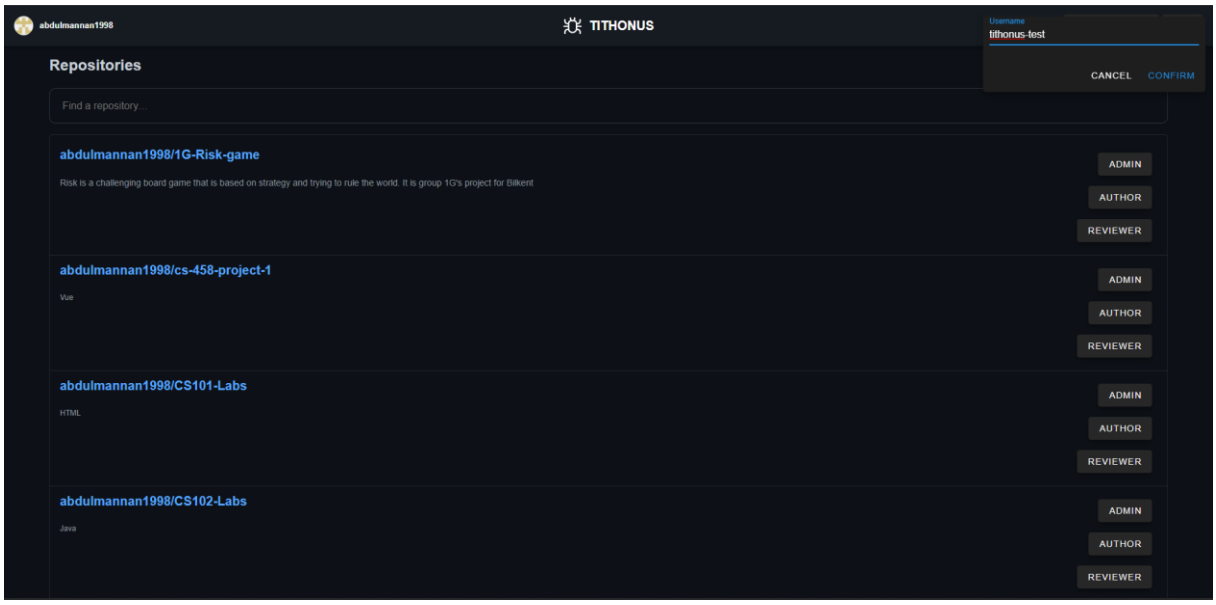

Figure 8 - GitHub username to change account

c. Clicking the confirm button will lead to the Authorization page if the account that you are changing to is not logged in already. Here you will have to click the sign in as *username* button in the top bar highlighted blue.

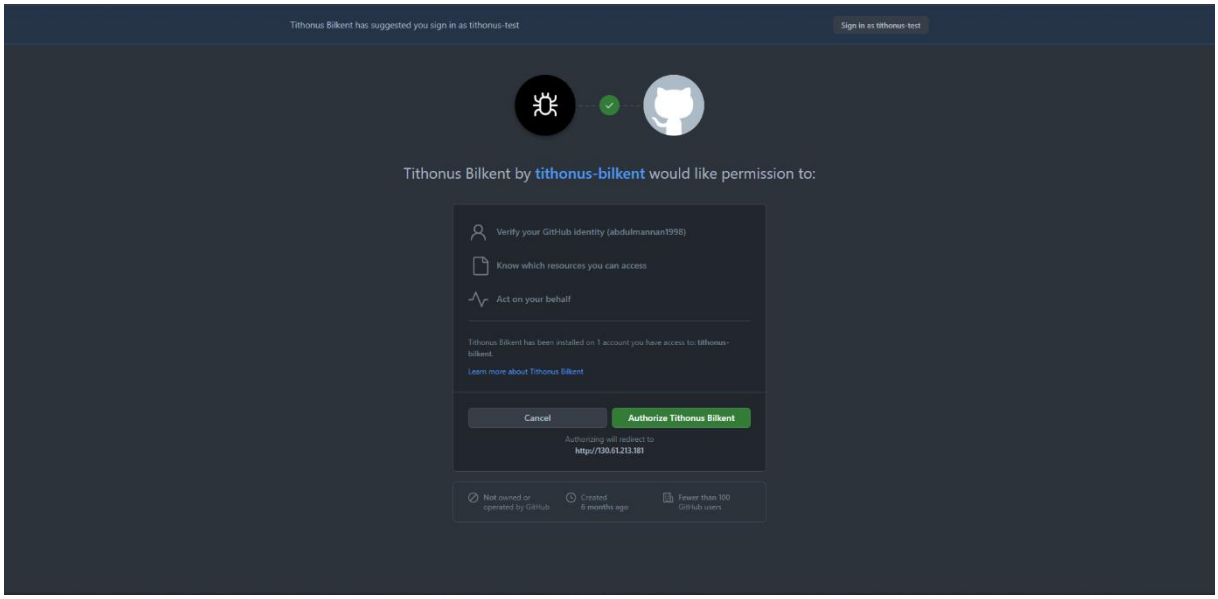

Figure 9 - Authorize account to change to

d. Once you sign in as this new user, you will be rerouted to the Repository List page for the new user account you just logged into.

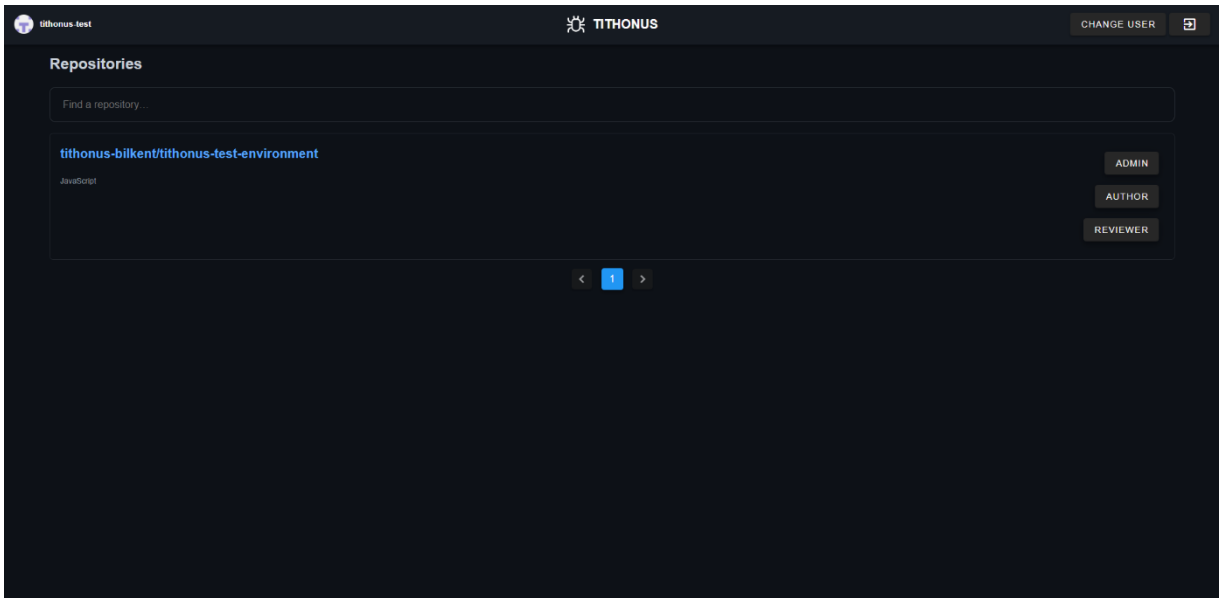

Figure 10 - Repository List of New User

2.3 View Pull Request Details

2.3.1 Pull Request Details Tab

a. In this tab, you can view the timeline for a pull request. On any code diff block shown in the timeline, you can reply to comments left in it. Any comments left can be edited using the edit icon in the top right corner of the comment. It can be deleted using the trash icon also.

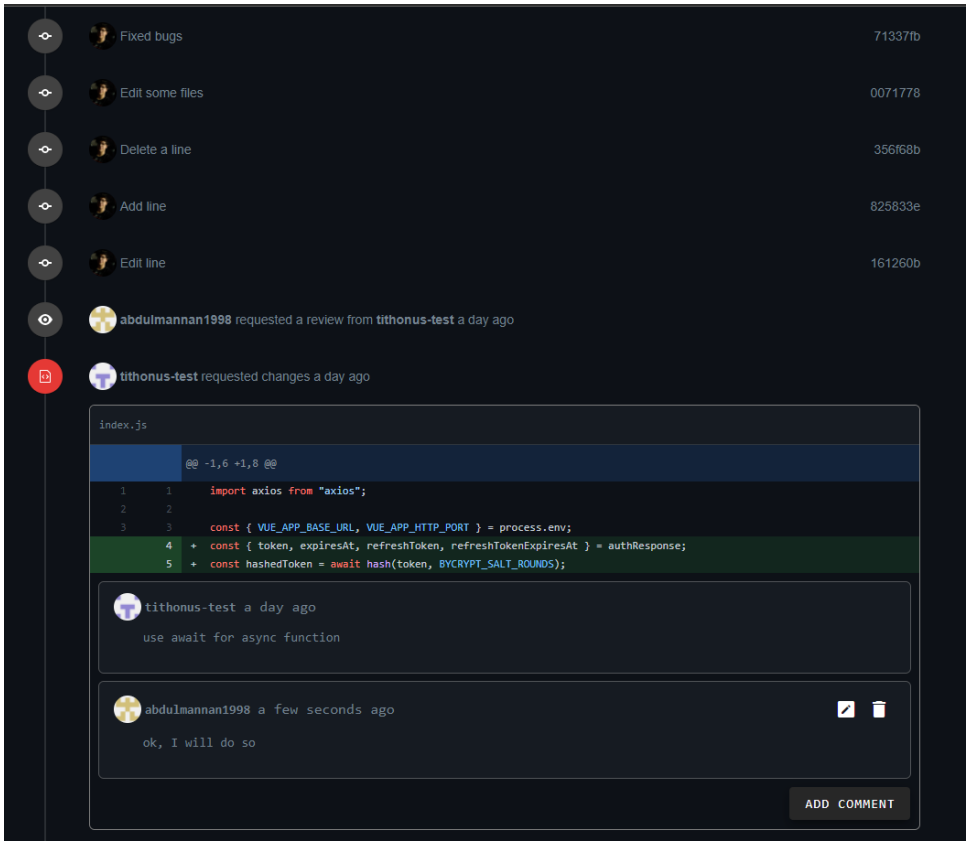

Figure 11 - Pull Request Timeline

b. You can also view the pull request's status and, if applicable, merge the pull request.

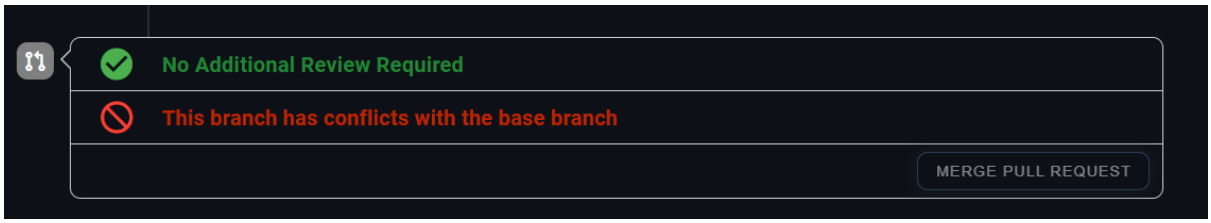

Figure 12 - Pull Request Status

c. Comments can also be left on the pull request itself. All comments support Markdown syntax.

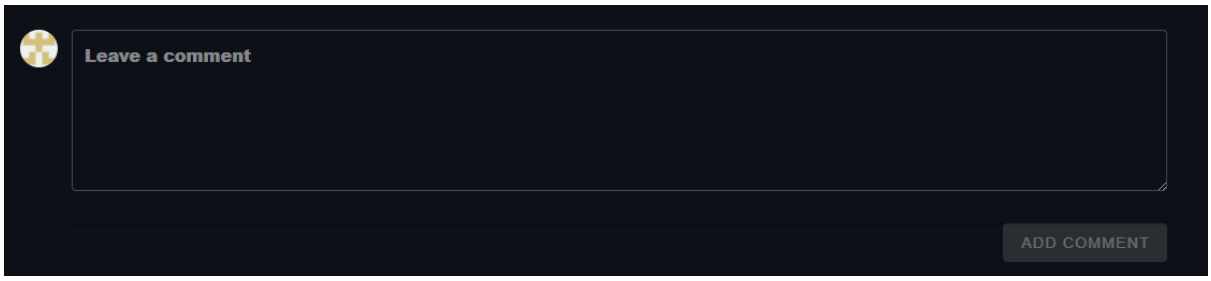

Figure 13 - Pull Request Comment

2.3.2 Files Changed Tab

a. The files changed tab shows you the code diff for the pull request.

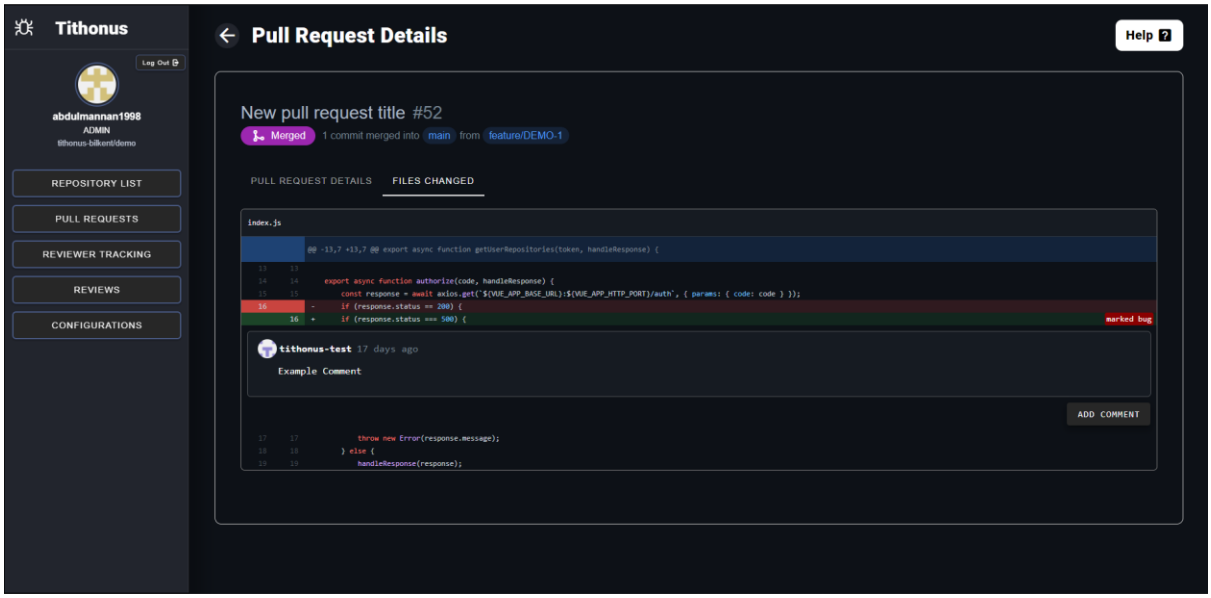

Figure 14 - Files Changed tab

b. Comments left on any line in the latest review can also be replied to from here, as seen in the figure above.

2.4 Help and Tooltips

To guide the user and make the whole experience of using Tithonus easier to learn, we have included a help button on most pages. Hovering on the help button will reveal tips on navigating the page and interacting with it. It is located in the top-right corner.

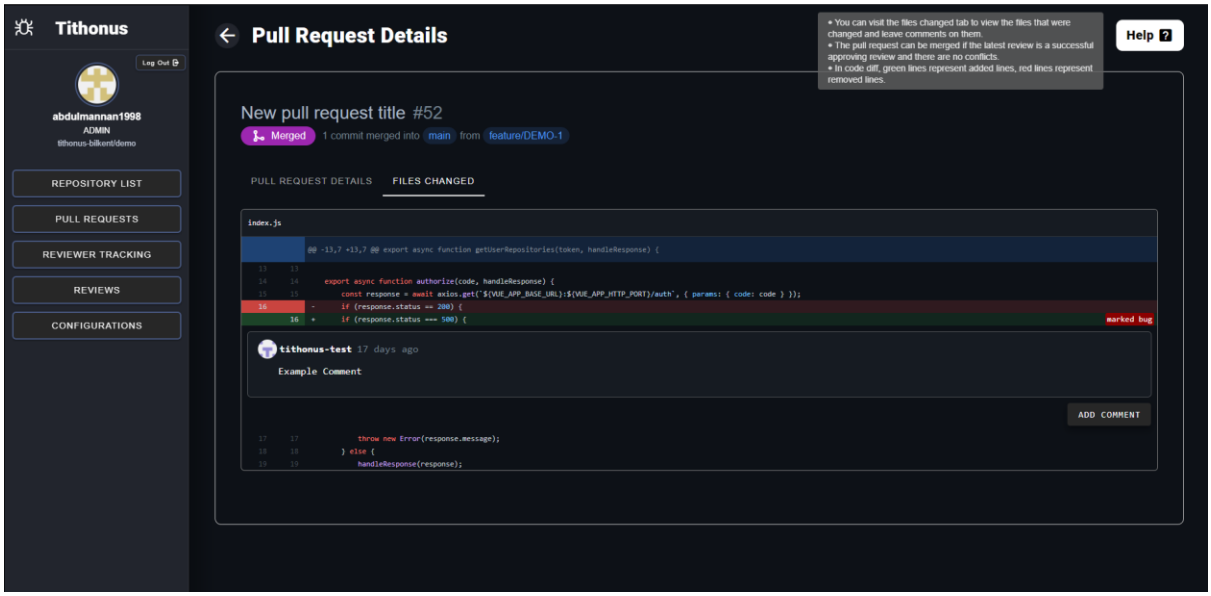

Figure 15 - Help Button on Pull Request Details page

Tooltips are also included on different pages. These tooltips may be on fields, buttons, or data table columns that need more information about what purpose they serve.

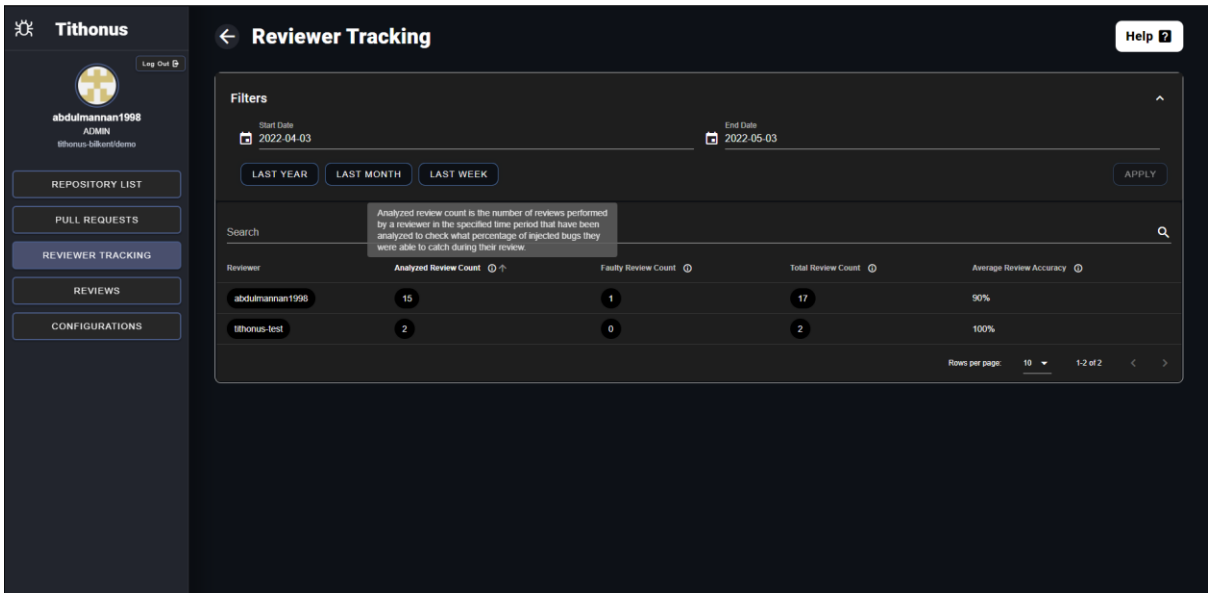

Figure 16 - Tooltip in Reviewer Tracking data table

2.5 Logging Out

All users can use the Log Out button in the sidebar to log out from the application.

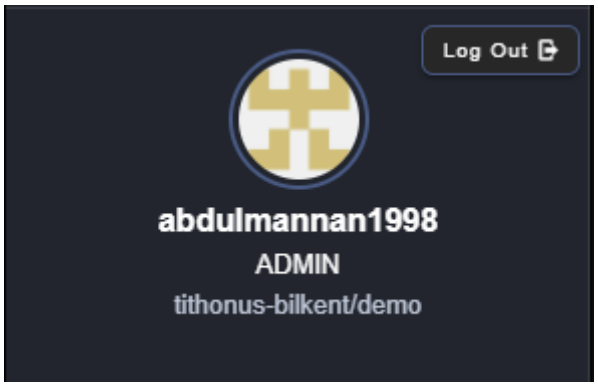

Figure 17 - Logout Button in Sidebar

3. Definitions

Analyzed Review

An Analyzed review is a review that has been checked to see whether it is successful or faulty.

Base Branch

The base branch is the branch where we are trying to incorporate the changes made in the compare branch.

Compare Branch

The compare branch is the branch from which we want to copy changes to incorporate into another branch.

Diff Content

The diff content is the series of changes made to the code in a pull request. It is shown as the lines/code removed (highlighted red) and the lines/code added (highlighted green).

Faulty Review

An admin can set a threshold for minimum review accuracy score in a repository. If a review has a review accuracy score that is lower than this threshold, the review is considered faulty.

Injected Bug

Injected bugs are those that the author injects while creating a new pull request. These bugs are usually small mutations in the code that would change the code's behavior.

Marked Bug

A marked bug is some line of code in a pull request that the reviewer thinks contains a bug and has consequently marked as buggy.

Pull Request Timeline

The pull request timeline displays information such as commits made, reviews requested, reviews canceled, changes requested, or review approved. It is arranged in a chronological order and serves to show the order in which different events on a timeline occurred.

Review Accuracy Score

The review accuracy score is measured as follows:

$$Review\ Accuracy\ (\%) = \frac{Number\ of\ Injected\ Bugs\ Marked\ by\ Reviewer}{Total\ Number\ of\ Bugs\ Injected\ by\ Author} * 100\%$$

For an injected bug to be counted as marked, the reviewer has to mark the exact line that the author injected the bug in.

Successful Review

An admin can set a threshold for minimum review accuracy score in a repository. If a review has a review accuracy score that is higher than this threshold, the review is considered successful.

User Role

A user role is decided based on the GitHub permissions a certain user has for a repository. An author is someone who has triage and pull permissions. A reviewer is someone who has push permissions. And an admin is someone who has maintain or admin permissions.

4. Manual for Each User Role

4.1 Author

4.1.1 Create a New Pull Request

a. From the Pull Requests page, click the New Pull Request button.

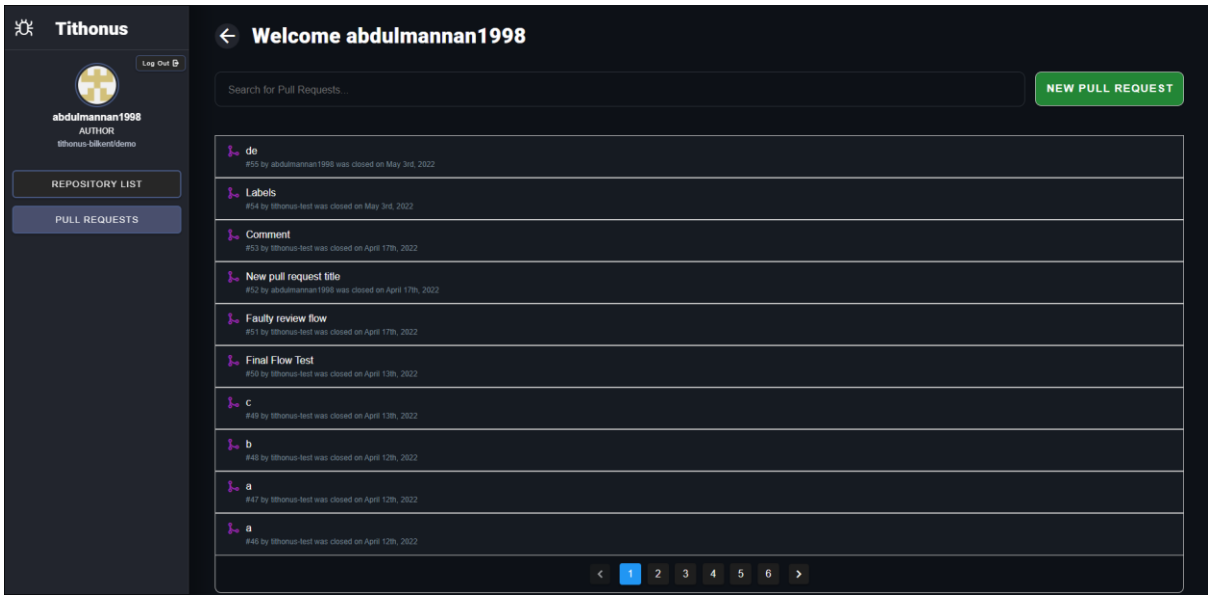

Figure 18 - New Pull Request Button on Pull Requests page

b. Choose the base branch and the compare branch from the dropdown menus. Once you select the branches for the pull request, you will also be able to view the commits on the compare branch.

You also need to provide a title for your pull request for the Create Pull Request button to be enabled. Once it is enabled, you may press it to proceed.

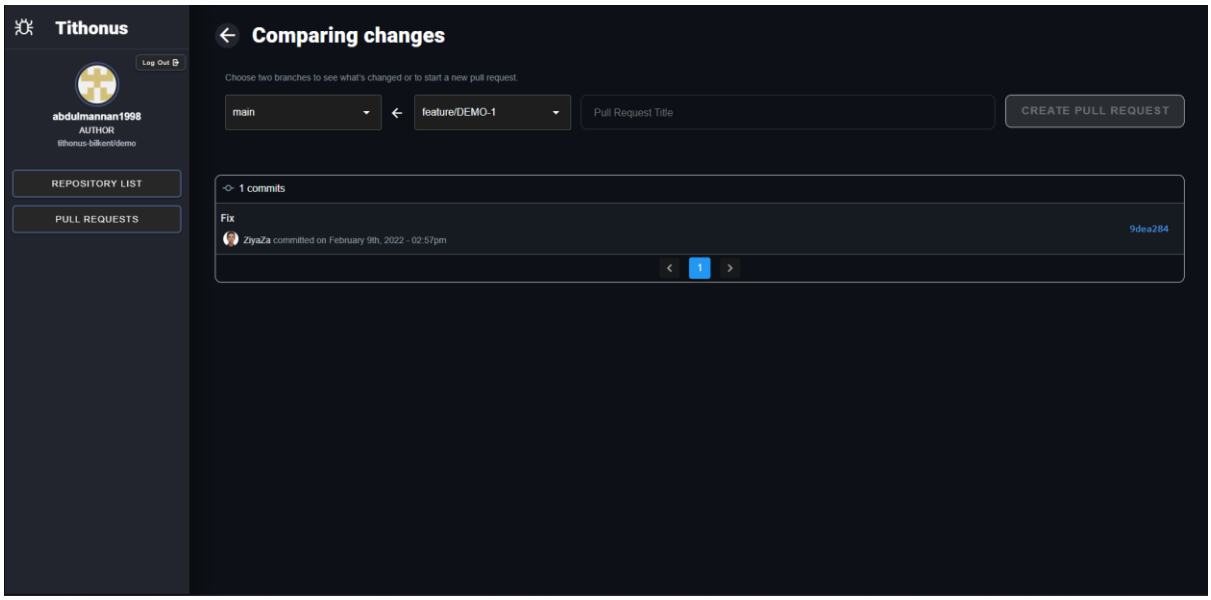

Figure 19 - Comparing Changes page

c. On the following pull request page, you will be presented with the content of the pull request, mainly the differences between the compare and base branches. You may click on the text in any line highlighted green to edit it and inject a bug into your pull request code. Once you are done injecting a bug in a particular line, you will need to click on the tick button in the same line to persist that bug. To assign a reviewer, you must click the cog button beside the Reviewers text. This will give you a dropdown from which you may choose the reviewer for this pull request. Once you are done, click the Confirm button to finalize the process.

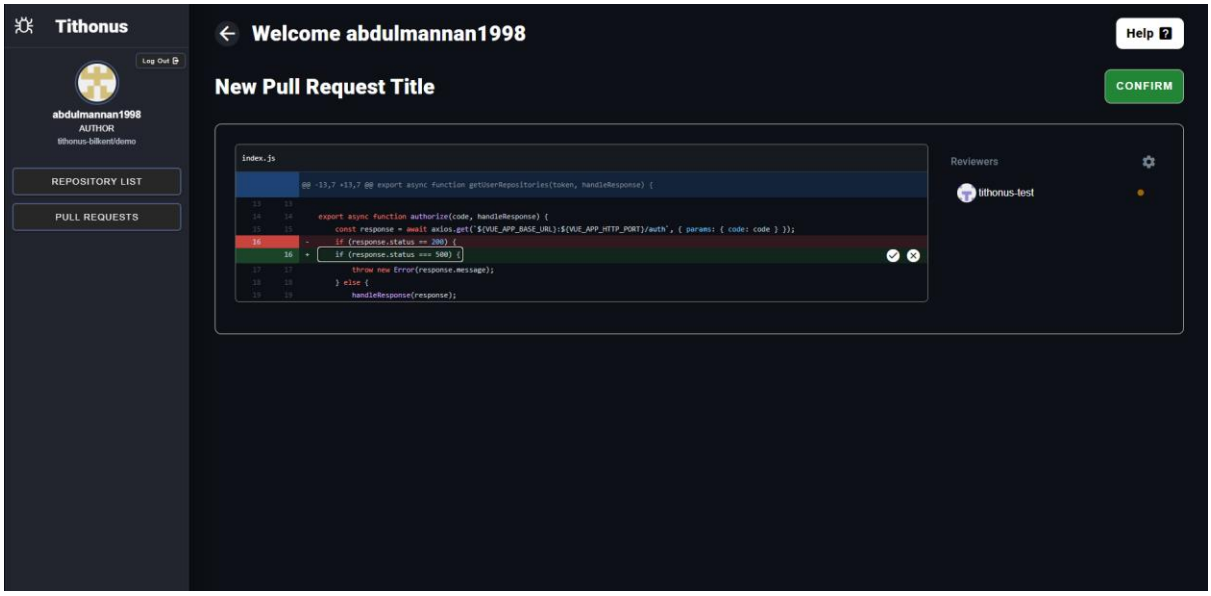

Figure 20 - Injecting Bugs in New Pull Request

d. When the pull request is confirmed, you will be taken to the Pull Request Details page, where you can check the details of the created pull request. In addition to the functionality specified in the

Overview section, an author can also change the reviewer on this page. In case of a faulty review, an author can also reassign a reviewer from this page.

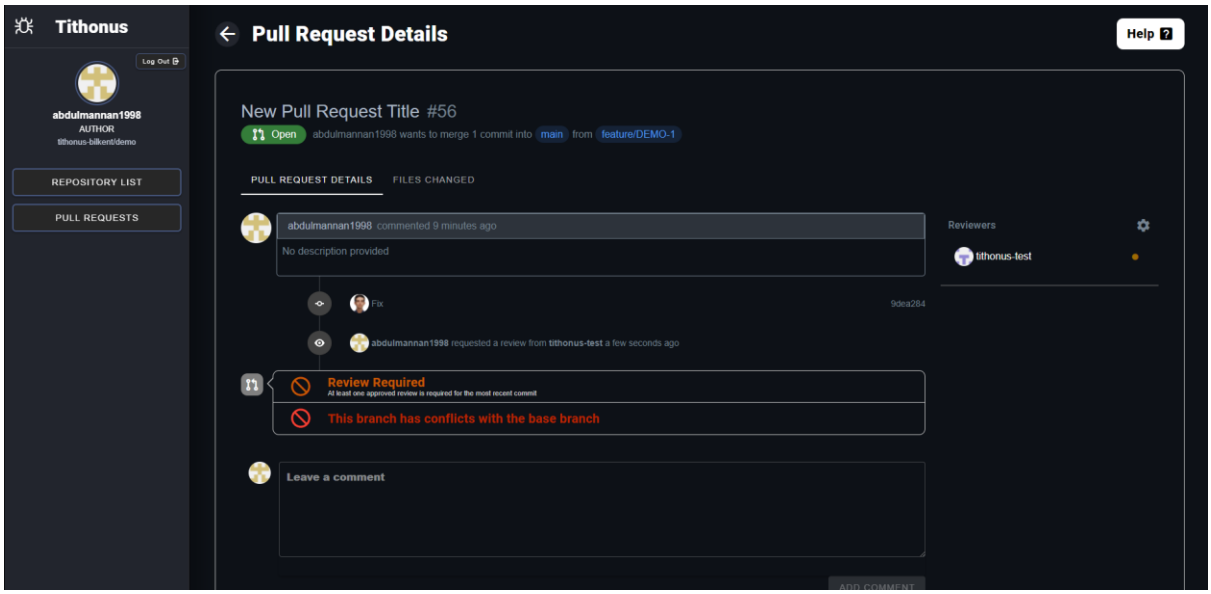

Figure 21 - Pull Request Details of newly created Pull Request

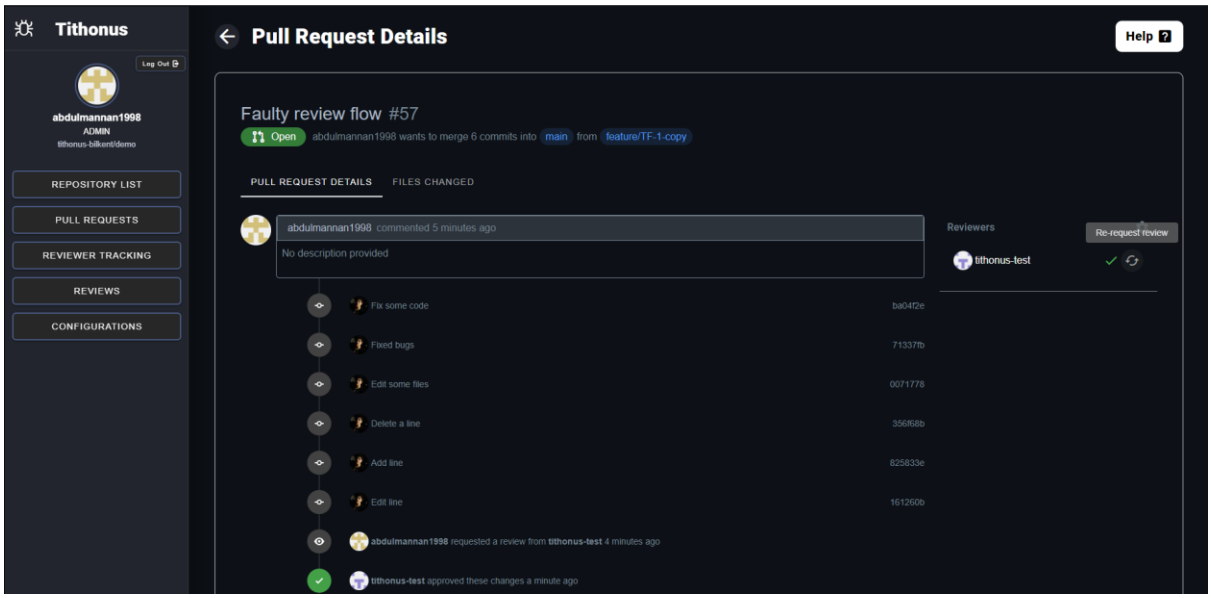

Figure 22 – Re-request Review in case review was faulty

4.2 Reviewer

4.2.1 Reviewing a Pull Request

a. From the Pull Requests Page, click on the pull request you want to review from the list.

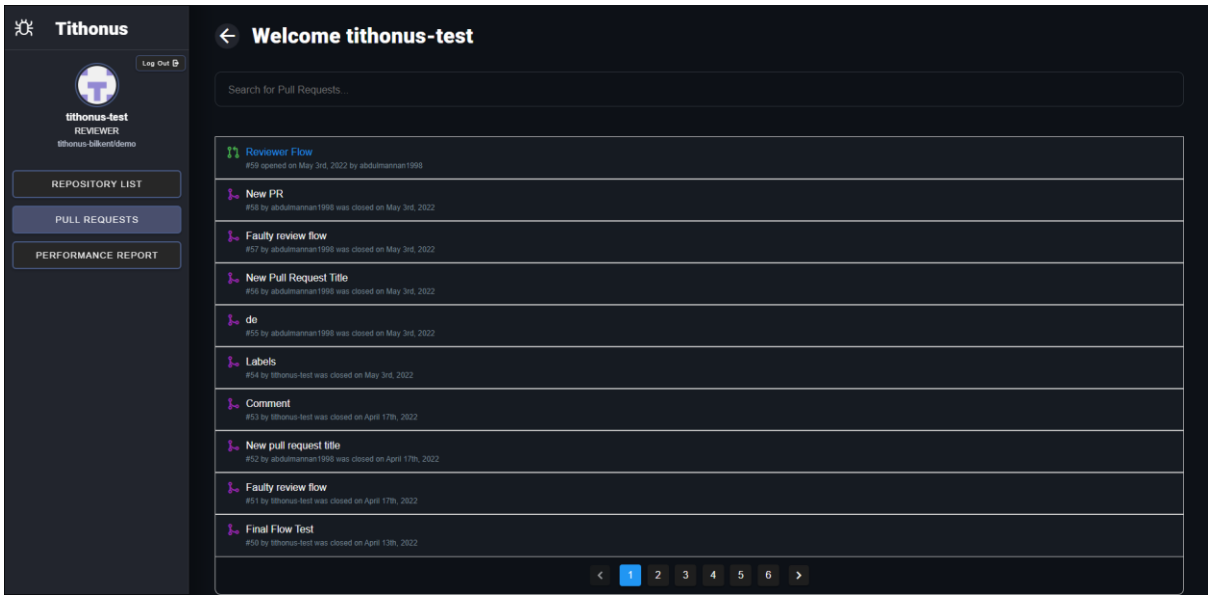

Figure 23 - Pull Requests list on Pull Requests Page

b. Once the pull request is clicked, you will be taken to the Pull Request Details page for that particular pull request. To start your review, click on the Add your review button. This button will only appear if you are the reviewer assigned to this pull request. Clicking the button will take you to the New Review page.

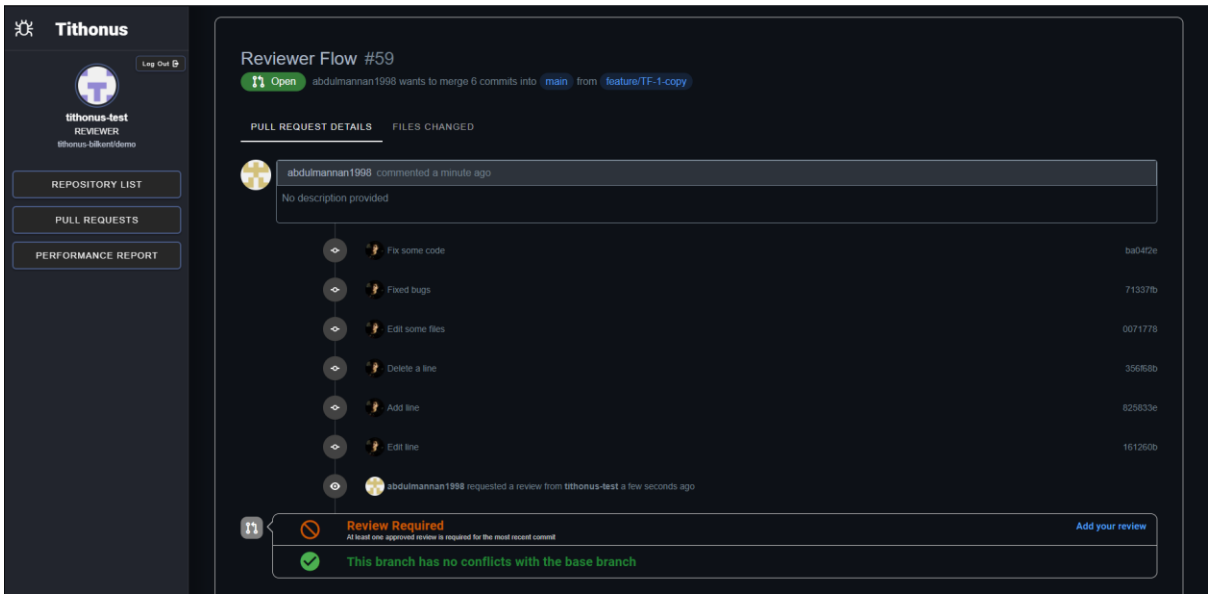

Figure 24 - Add your review Button on the Pull Request Details page

c. On the New Review page, you will see the diff content of the pull request. Hovering on any line that is highlighted green will give you two options. You may either leave a comment on a particular line or mark the line as buggy if you feel that there is some bug in that line. Once you mark a line as buggy, it will be highlighted purple. If a mistake was made when marking a line, it can also be unmarked by clicking the bug icon. Details of how the review accuracy score is calculated is

provided in the Glossary. A comment added to a line will be shown beneath it. Comments also support Markdown syntax. Any comments that you add may be edited or deleted if you wish. This can be done by clicking the appropriate icon in the top right corner of the comment box.

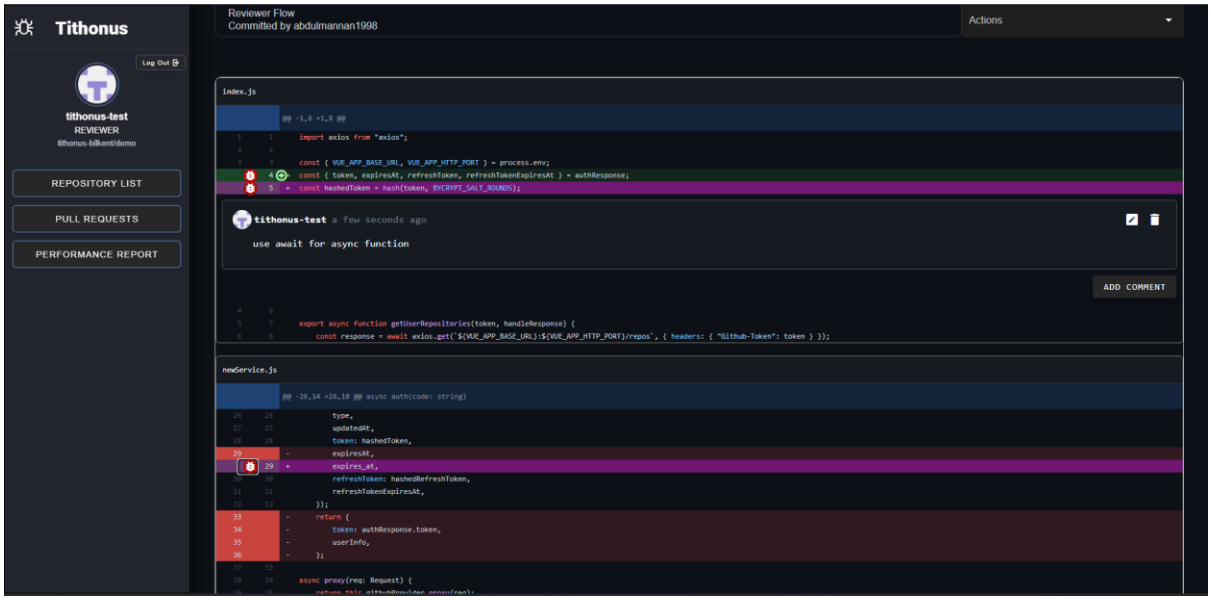

Figure 25 – Line marked as buggy, and comment left on it

d. Once you have done your due process on the review, you can choose one of two options. If there were no bugs in the review, you might select the Approve option from the Actions dropdown, and you will be taken back to the Pull Request Details page. If you marked some bugs in the review, you might choose the Request For Change option from the Actions dropdown, after which you will be taken to the Review Action Confirmation page.

e. On the Review Action Confirmation page, you will be shown the pull request diff again. However, this time the diff will be annotated with labels. A line can have the marked bug or injected bug label. The purpose of showing such a page is to help you confirm whether you still want to request changes or approve the pull request. This may happen because you marked a line as buggy, but that bug turned out to be an injected bug. In such a case, you may want to approve the review. If you still feel that there are bugs that are different from the injected ones, you may proceed to request changes from the Actions dropdown again. It should be noted that reviewers may not go

back to the New Review page from here but will instead be rerouted to the Pull Requests Details Page, and their review will be automatically approved.

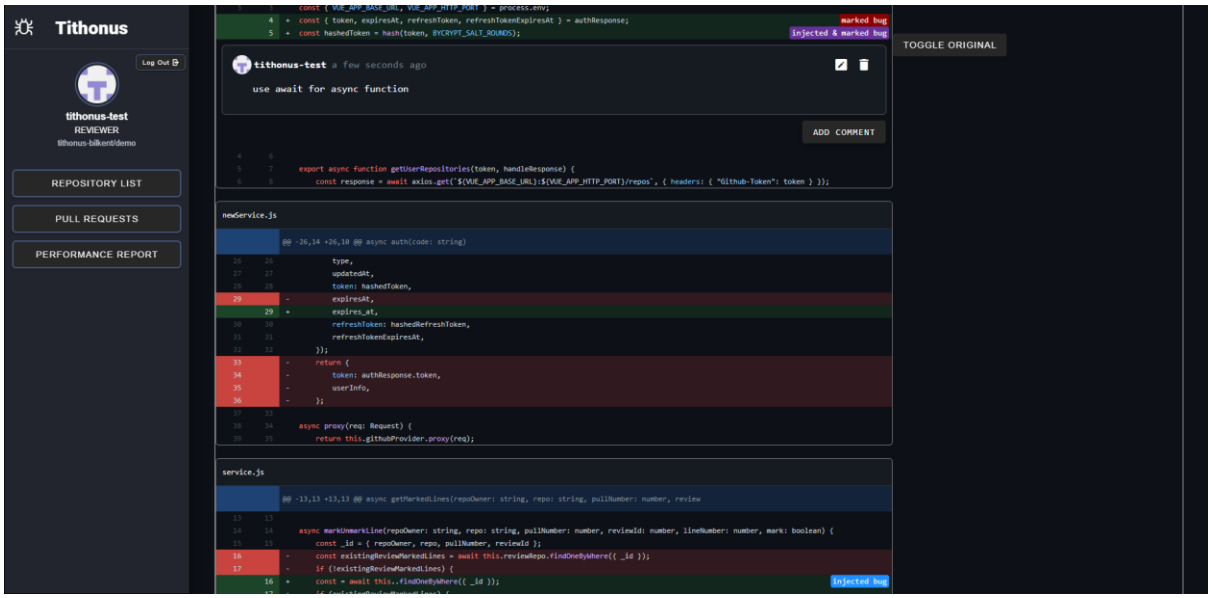

Figure 26 - Review Action Confirmation page

f. Once a review is approved and if the reviewer does not have a successful review on this pull request already, the review accuracy score is calculated. Details are given in the Glossary. If the review accuracy score is below the threshold specified by the repository admin, then the review is considered faulty. Otherwise, it is considered to be successful. If the review is faulty, the admin must reassign a reviewer to the pull request again.

4.2.2 View Performance Report

a. This process can only be done if an admin has allowed reviewers to be able to view their performance.

b. From the Pull Requests page, click the Performance Report button in the sidebar on the left.

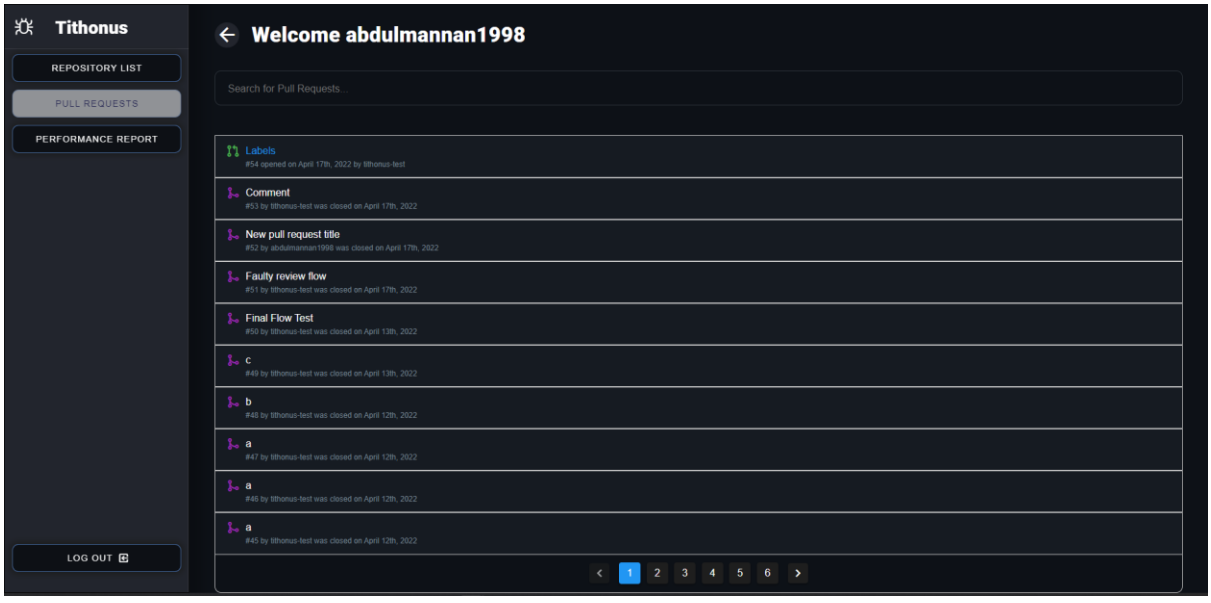

Figure 27 - Performance Report Button in sidebar

c. Completing the previous step will take you to your Performance Report page. Here, you can view numerical and graphical statistics about your review performance. There is a filter bar present that can be used to show data with a specific start and end date. You can also tweak the data interval to see the graphs by that interval.

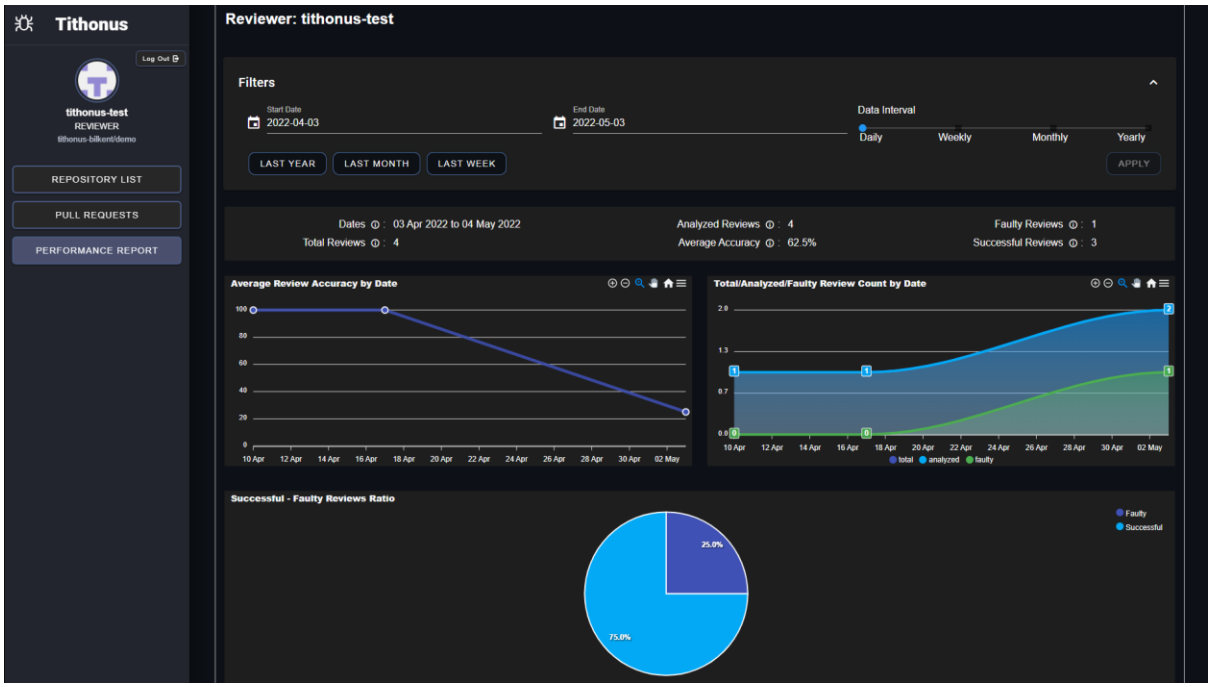

Figure 28 - Performance Report Page

4.3 Admin

4.3.1 Handling Faulty Reviews

a. As an admin, you can access the Reviews page and set the review types filter to faulty reviews. This will make it so the reviews shown in the data table are the ones that were faulty. Any of these reviews can be clicked to view the pull request details for it.

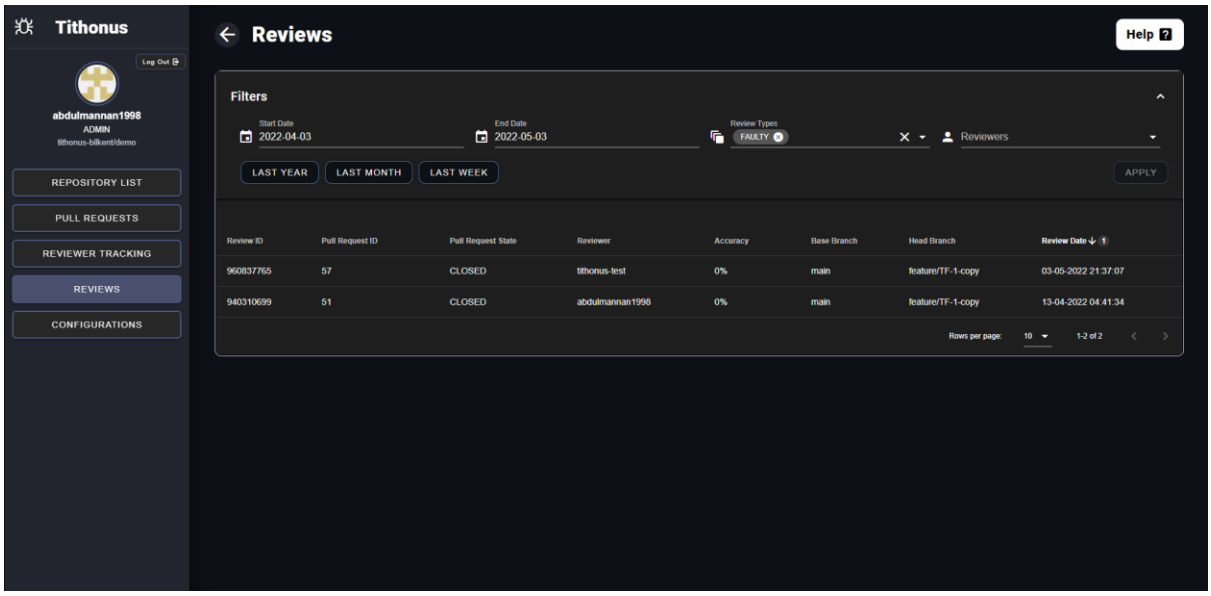

Figure 29 - Reviews filtered by faulty review type

b. You can also look for the faulty review in the list of pull requests on the Pull Requests page.

c. Once at the Pull Request Details page for the pull request, you can see in the pull request status bar that the review was faulty and that an admin action is required. This action can be assigning the pull request to another reviewer or reassigning it to the same reviewer who did the faulty review in the first place.

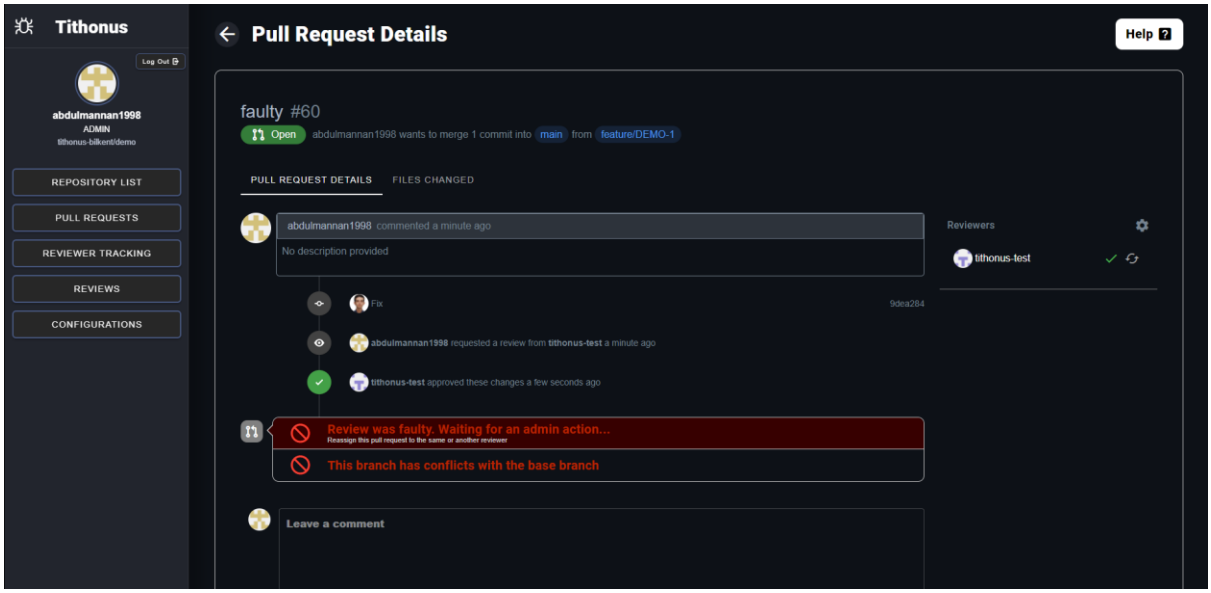

Figure 30 - Faulty Pull Request Review Status

d. There are different actions available to you as an admin. You can leave a warning on a pull request using a comment or you can use the files changed tab to view review details. These are details such as viewing comments and labels.

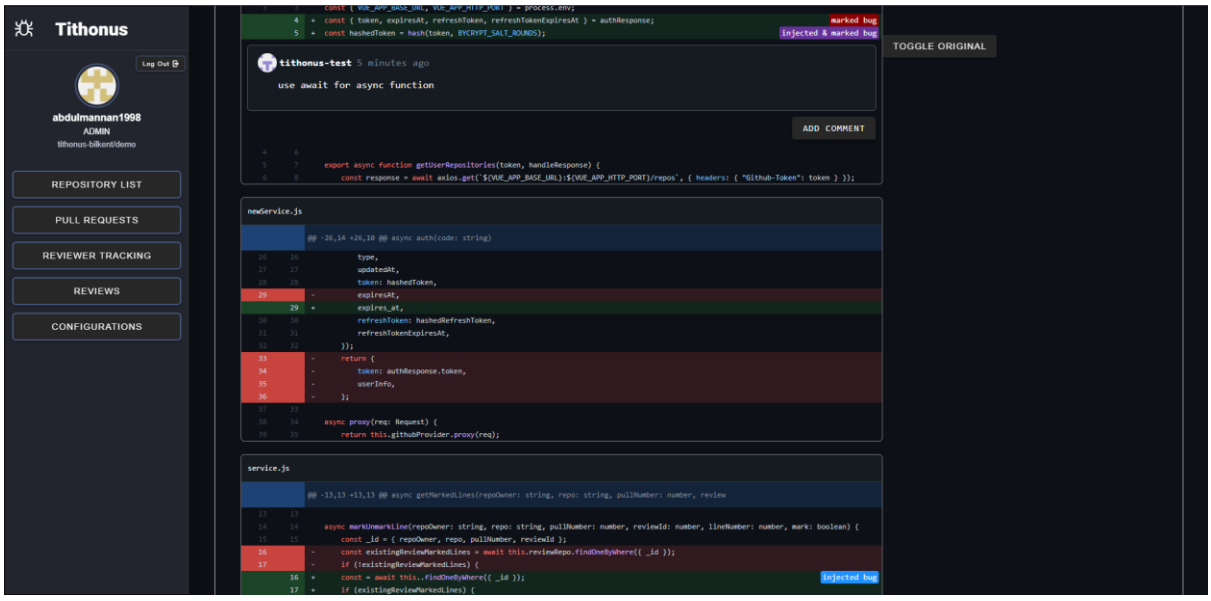

Figure 31- Comments and Labels in Files Changes tab

4.3.2 Reviewer Tracking

a. Click on the Reviewer Tracking button in the sidebar to access the Reviewer Tracking page.

b. On the Reviewer Tracking page, you can set date filters to see reviews only between the start and end dates specified in the filters. This is set to an interval of one month by default.

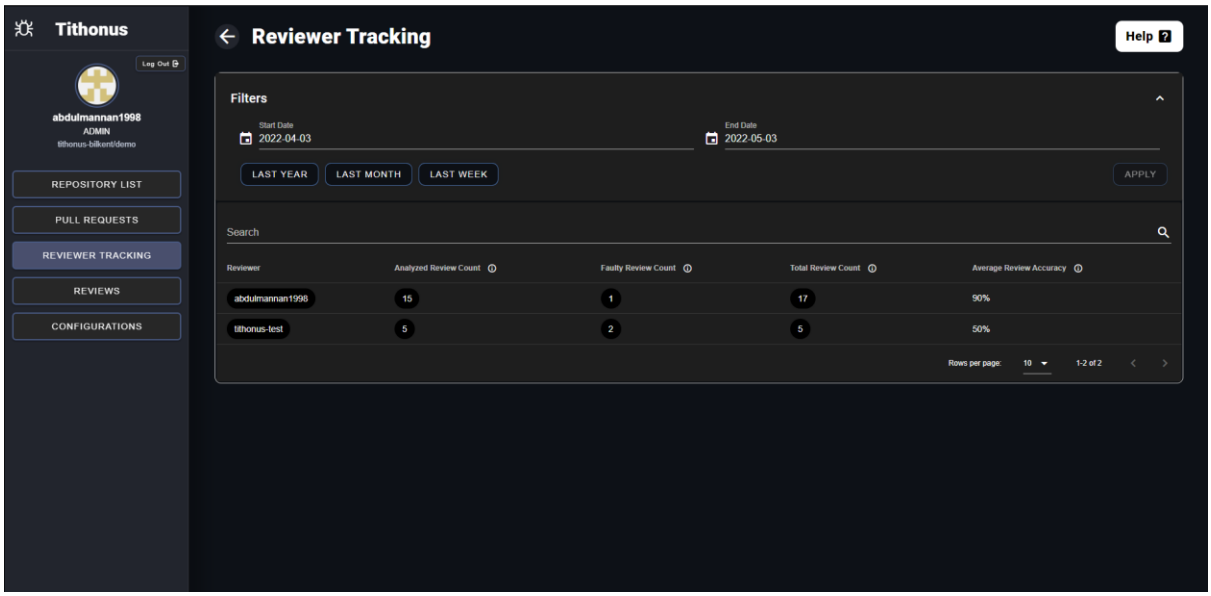

Figure 32 - Reviewer Tracking page

c. For each row, you can access further details about a reviewer by clicking on the following items:
   - Click on a reviewer's name to see their Performance Report.
   - Click on a reviewer's analyzed review count to see the list of reviews they did that have been analyzed. This information will be shown on the Reviews page with the appropriate filters.
   - Click on a reviewer's faulty review count to see the list of reviews they did that have been analyzed and were faulty. This information will be shown on the Reviews page with the appropriate filters.
   - Click on a reviewer's total review count to see the list of reviews they did. This information will be shown on the Reviews page with the appropriate filters.

4.3.3 Reviews

a. Click on the Reviews button in the sidebar to access the Reviews page.

b. You can see the list of all reviews here. By default, reviews done in the last month are shown.

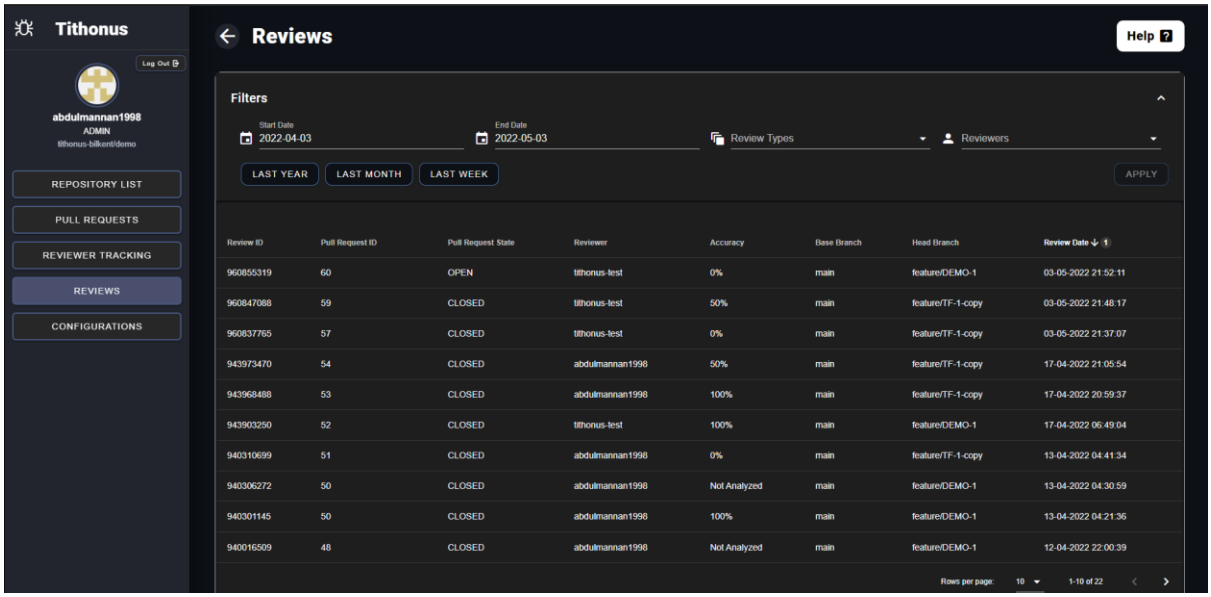

Figure 33 - Reviews page

c. The filter bar can be used to adjust the start and end dates for the reviews shown in that interval.
d. Reviews can also be filtered by review types.
e. Reviews can also be filtered by reviewers so that reviews from only certain reviewers are shown.

4.3.4 Configurations

a. Click on the Configurations button in the sidebar to access the Configurations page.

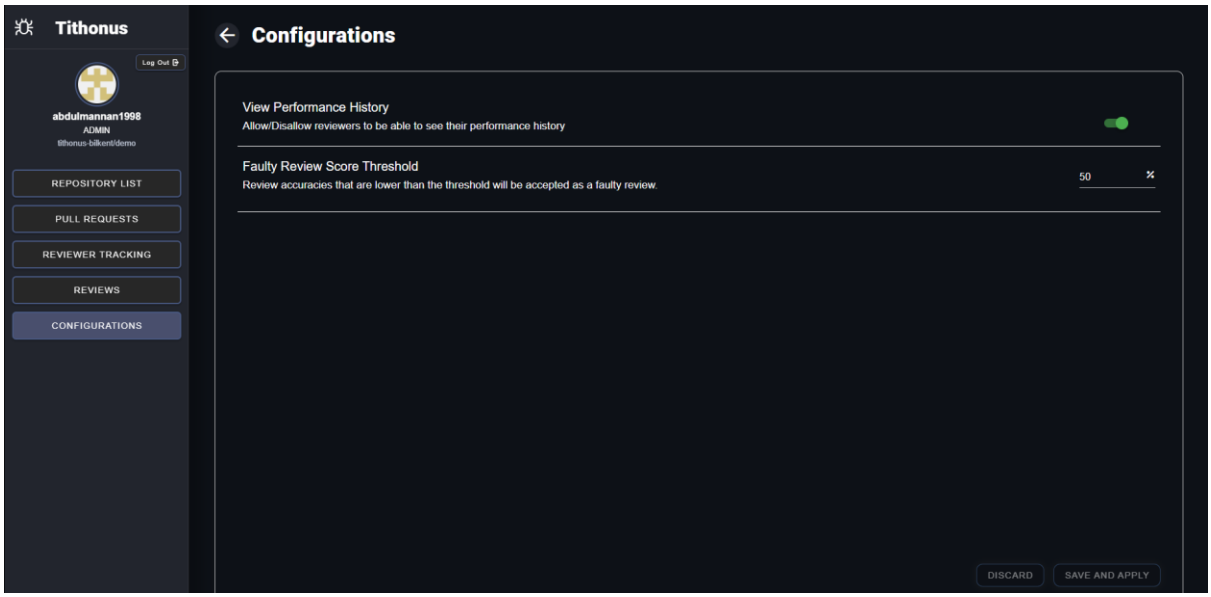

Figure 34 - Configurations page

b. There are two options on this page.
   - You can allow/disallow reviewers to see their performance history.
   - You can set the threshold for review accuracy score. If a review has a review accuracy score less than this threshold, the review would be considered faulty.

5. Data Accessed and Stored by Tithonus

5.1 User Data

When you first access Tithonus, you will need to authorize the application. To authenticate you as a user, GitHub will show you the following screen:

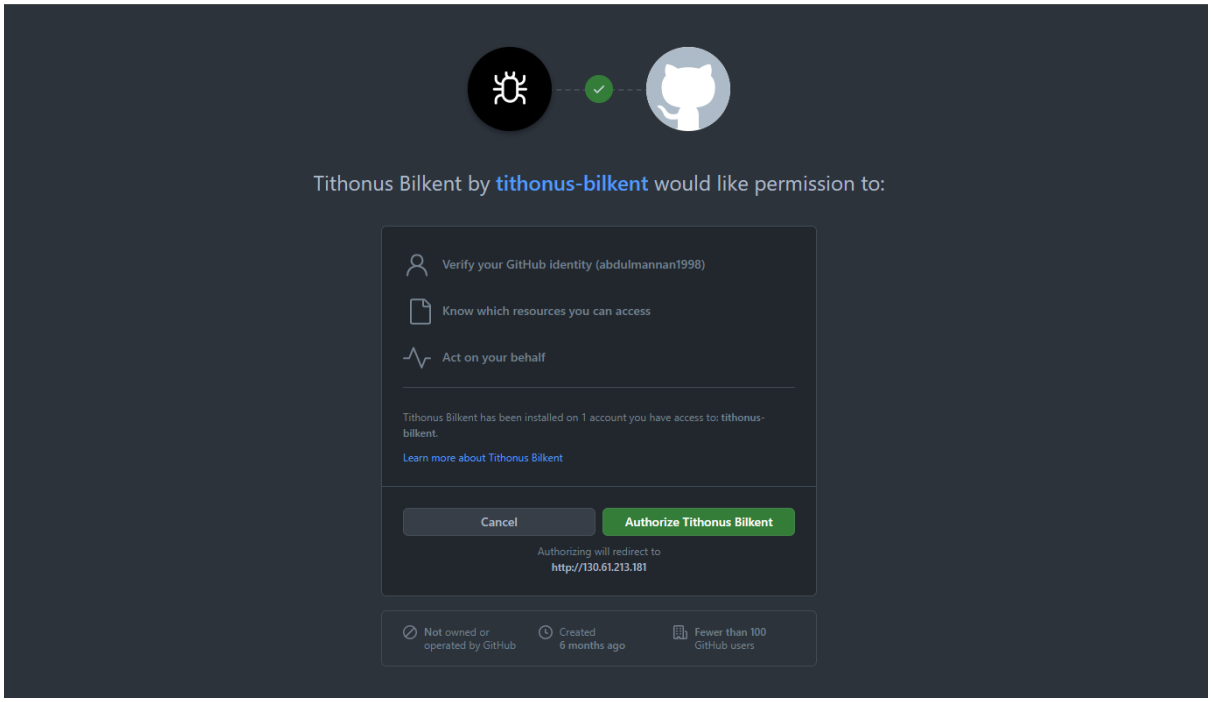

Figure 35 - GitHub User Authentication

The application is able to verify your GitHub identity. If Tithonus is not installed in any of your repositories or your account, then Tithonus will only be able view your public repositories after authorization. Lastly, Tithonus is also able to act on your behalf. This is only possible when Tithonus is installed in your GitHub account or a repository. View the Installation section to review these details.

5.2 Installation Data

If the Tithonus Bilkent GitHub application is installed in your account or repository, it will need to act on your behalf to function correctly. To this end, Tithonus will need a user access token that it gets when you click the Login with GitHub button in the application startup. This user access token is then used to interact with the GitHub API. Additionally, it will store some of this data on our server. This is so that it does not repeatedly poll GitHub servers for information we have previously gotten. The GitHub API has a rate limit that would be exhausted quickly if we did not perform caching for the data.

The GitHub data that we access is as follows:

- Repository Permissions
    - Administration (Read-only access): Repository creation, deletion, settings, teams, and collaborators.
    - Contents (Read and Write access): Repository contents, commits, branches, downloads, releases, and merges.
    - Issues (Read-only access): Issues and related comments, assignees, labels, and milestones.
    - Metadata (Read-only access): Search repositories, list collaborators, and access repository metadata.

    - Pull requests (Read and Write access): Pull requests and related comments, assignees, labels, milestones, and merges.
- Events Subscribed to
    - Commit comment: Commit or diff commented on.
    - Pull request review: Pull request review submitted, edited, or dismissed.
    - Pull Request: Pull request opened, closed, reopened, edited, assigned, unassigned, review requested, review request removed, labeled, unlabeled, synchronized, ready for review, converted to draft, locked, unlocked, auto merge enabled, auto merge disabled, milestoned, or demilestoned.

Events that we subscribe to will automatically send any new information about an action performed to our server. For example, when a new pull request is made, GitHub will automatically send the details about that pull request to our server.

The GitHub data that we store on our server is as follows:

- For each pull request: base and head branch name, commit sha, and current state.
- For each review: review submission date and review username.
- For each injected bug: modified (buggy) version of a line.