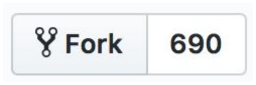
OSS Projects on Expertiza for Fall 2020

Expertiza repository on Github: https://github.com/expertiza/expertiza
Create yourself a Github account and fork the Expertiza repo. **Fork the project.** Click on the button that looks like this:



After git clone your forked repository, you can execute the command **`run cd expertiza && bash ./setup.sh`** to install dependencies.

Please view the Expertiza YouTube channel before beginning your project. The videos on this channel will show you many areas of the system that you do not see as a student, but which are important in setting up and managing courses and assignments in Expertiza.

Details of the Expertiza project, including links to documentation and development may be found at: http://research.csc.ncsu.edu/efg/expertiza**.** Especially note the Documentation on Database Tables, which, though not completely up to date, explains almost everything you would want to know about Expertiza db tables.

Setting up Expertiza environment (see Google doc)

On all projects

1. Establish your project Github Repository and Accompanying Github Project Board.
    a. All groups are required to plan and track project tasks using Project Boards. This will also be used as a mechanism to monitor teammate contributions and management of your project.
2. **Create a pull request as soon as you make your first commit so that you have a much higher chance to get a good score** and the staff can track your progress and offer help where needed.
3. Make sure to sync with the beta branch frequently.
4. If your project is focused on fixing multiple issues, it will help the Expertiza team if you create separate pull requests, one for each issue you have fixed.
5. You may find that some functions are already broken before you do your refactoring. It is *not acceptable* if you take the broken code, refactor it so that it breaks in the same way as before. Instead you should try to fix the code to make it work first. You are also welcome to talk to the Expertiza Support team.
6. For each project, you **must** write related tests.
7. When you finish your project, synchronize it with the master branch and resolve conflicts. Make sure that **all tests succeed**, as evidenced by the Travis CI build passing.
8. Expertiza DB document

9. When you create the test cases, keep in mind that the testing DB does not have the production data in it. If you need some pre-existing entities like courses, assignments or students, you can find them in factories folder.
10. Common mistakes checklist (we will check these common mistakes during grading.)
11. Some good RSpec tutorials. RSpec tutorial slides, video (until 9:32), http://rspec.info and http://www.betterspecs.org.

What to submit

- On Monday, Oct. 28, the project is due. You should submit to Expertiza …
    - your pull request (which should have been created when you started the project),
    - a wiki page talking about what you have already done
        - Here are examples of good writeups for the OSS project: **E1553**, **E1566**,**E1571**, **M1504**, **M1506**
    - for testing projects, record a video of tests running and submit a link
    - for other projects, deploy your application on Heroku or VCL and make sure the server is up for 3 weeks
    - A link to your Github Repo, which should also grant access to your Github Project Board.

E2050. OSS Project Juniper: Bookmark enhancements

Mentor: Saurabh Shingte (svshsingt@ncsu.edu)

Files involved: app/models/bookmark.rb
app/models/bookmark_rating.rb
app/models/bookmark_rating_questionnaire.rb
app/models/bookmark_rating_response_map.rb
app/controllers/bookmarks_controller.rb
and possibly
app/models/assignment_participant.rb

The issue: “Bookmarks” in Expertiza can be attached to a topic to help the authors who choose the project to do a good job. Let’s say there are five topics that I’m interested in and would like to contribute to, but I can only choose one. Well, for the other topics, I’m allowed to submit hyperlinks to pages that I think would help the author do the work. On each line of the signup sheet are two icons, one for adding a bookmark to the topic, and another for viewing bookmarks on the topic.

What needs to be done: Authors are permitted to rate bookmarks by filling out a dropdown or a rubric that asks how helpful the bookmark was. I am not sure whether this is completely implemented in the current Expertiza, but it is easy to find the related code in the repo by searching the repo for the term “bookmark”. If that is partially implemented, please discuss it with me and get it working.

When you go to Manage > Questionnaires, Bookmark ratings should be a type of rubric, but it's not shown. Add it. Confer with Ajay Pendyala (apendya), since he has recently worked on this code. You may have to merge his changes before continuing.

But not only the author can tell whether bookmarks were useful; the reviewer can be asked to weigh in too. As part of the review questionnaire, it would be helpful to ask the reviewer whether the bookmarks appear to have been used by the author.

This would require some design effort: we would like to ask the reviewer for each specific bookmark submitted for this work, whether it appears that the author has made use of the bookmark.

If the author has made use of the bookmark, the participant who submitted the bookmark should get some recognition or credit for submitting it. What form this credit should take is a topic for discussion (should there be a badge for this? Should it be factored into the submitter's score?). You might consider some of these changes, but it is enough to add a way for the reviewer to rate the bookmarks (and of course to view those ratings). We can think later about how the ratings should be used.

For results of the second-last bookmarking project, see

- https://github.com/rahuliyer95/expertiza
- https://github.com/expertiza/expertiza/pull/1266
- https://youtu.be/tPpowgNRINk
- https://youtu.be/F95FA9Iam0s
- http://wiki.expertiza.ncsu.edu/index.php/E1830_OSS_Project_Juniper:_Bookmark_enhancements

Here are our comments on that project.

It is not clear why so much code about bookmark rating is added to the bookmarks_controller. Shouldn't it be in the bookmark_rating model or controller? Tests seem to be pretty extensive. Videos show them running.

Several reviewers say build did not pass. Most say it was because one of their tests failed. Should be resolved before code is merged. Also, many issues were raised by Code Climate.

You committed `.vscode/` folder; please remove it.

You are requiring `rspec` gem or different helper methods in RSpec tests. There have already been included, you do not need to require them again. Please remove them.

Last bookmarking project:

http://wiki.expertiza.ncsu.edu/index.php/CSC/ECE_517_Fall_2019_-_E1939._OSS_Project_Juniper:_Bookmark_enhancements

https://github.com/expertiza/expertiza/pull/1540

Issues identified with this project:

"Design

• I think new_bookmark_review may not be an appropriate method for the bookmarks_controller class; shouldn't this be created by the response_controller?

• Ditto for the code in bookmark_rating.rb that calculates an average score: seems to duplicate on_the_fly_calc.rb.

• Also, there is a magic constant 5.0; what is that? Does it ASSUME that the scale is 1 to 5? This is controlled by min_question_score and max_question_score for the questionnaire, and these variables should be used.

• In questionaires.rb, there is a literal ""Bookmark RatingQuestionnaire"" in addition to ""BookmarkRatingQuestionnaire""; there should be only one such rubric..

• According to the code, a teammate of the person who uploaded a bookmark could rate it; is this right?

• bookmarks/list.html.erb is pretty long; it would be better to use one or more partials. Magic constants 0 and 5!

Rough edges

• Bookmarks can be added, and can be viewed by the team whose topic they relate to. But the team has no way of knowing, without clicking on the ""View bookmarks"" icon, that any bookmarks exist for their topic. A UI change is needed here, even though the project spec did not say to change the UI.On the screen for rating bookmarks, there needs to be more horizontal space between ""Edit"" and ""Delete"".

Missing functionality

• The table of bookmarks has an ""Avg. rating"" and a ""Your rating"" column. Only ""Avg. rating"" is populated; ""Your rating"" is blank, at least for the bookmark that I created and rated.

Apparent bugs

• The ""Back"" link from bookmark listing ""OSS project and documentation"" takes me to the signup sheet for Wikipedia contribution … though the assignment number (827) in the URL is for the ""OSS project and documentation"" assignment. The id=30777, and that evidently takes precedence over the assignment number.

• When using a dropdown to rate bookmarks, I still need to specify a bookrmark-rating rubric, or otherwise the assignment cannot be saved. And when I rate a bookmark, I get this rubric, not the dropdown."

E2051. Refactor stage deadlines in assignment.rb

Mentor: Srujana Rachakonda (srachak@ncsu.edu)

Background: An assignment in Expertiza typically has many deadlines: deadlines for submission and review in each round, and potentially deadlines for signing up for topics, dropping topics, submitting metareviews, etc. Depending on what the next deadline is, an Expertiza assignment is said to be in a particular kind of stage. For example, if the next deadline (after the current time) is a review deadline (a `DueDate` object), then the assignment is in a “review phase.” Similarly, an assignment may be in a “submission phase” or a “signup phase.”

Depending on which stage an assignment is in, certain activities may be permissible or impermissible. For example, the default for a submission stage is to allow submission but not review, and in a review stage, the default is to allow review but not submission. These defaults may be changed by checking the relevant boxes on the Due Dates tab of assignment creation or editing. In any case, though, the `DueDate` object says what is and is not permissible, and the system has to know whether to gray out the submission or “Others’ work” link based on what is permissible at the current time.

Class `assignment.rb` used to have several methods for checking what kind of stage an assignment is in:

- `current_stage_name(topic_id = nil)`
- `find_current_stage(topic_id = nil)`
- `get_current_stage(topic_id = nil)`
- `link_for_current_stage(topic_id = nil)`
- `stage_deadline(topic_id = nil)`

Just from their names, it was hard to tell the methods apart. What’s the difference between `find_current_stage` and `get_current_stage`? The comments are not much help; you will need to see how the methods are actually used. Project E1906 reduced the five methods to two, but the tests failed, evidently because of changes they had made.

Note that both of the methods have a parameter that is a topic ID. This is because different topics may have different due dates. To understand how this works, read about staggered-deadline assignments on the Expertiza wiki (you can just search for “staggered” and it will bring up several pages). Again, the goal is to make the handling of stage deadlines easy to understand.

Here is the previous submission of the project.

- https://github.com/expertiza/expertiza/pull/1391
- http://wiki.expertiza.ncsu.edu/index.php/CSC/ECE_517_Spring_2019_-_Project_E1906._Refactor_stage_deadlines_in_Assignment.rb
- http://oss.flazzle.com/

Here are our comments on it:

The team did quite a nice job of refactoring 5 methods into 2 and making the code more readable. However, many reviewers have reported that the build has failed, and build failures are due to their changes, so I would not feel comfortable merging this code.

The project was assigned again, but this team didn't get the build to pass either. However, they made some additional refactorings, which appear to be useful. Try to merge their refactorings with E1906's.

- https://github.com/AnfGod/expertiza/tree/fresh-start
- https://github.com/expertiza/expertiza/pull/1709
- https://expertiza.csc.ncsu.edu/index.php/CSC/ECE_517_Spring_2020_Refactor_assignment.rb

Also, write tests as needed so that the changes are comprehensively tested.

E2052. Remove multiple topics at a time (Issue #1409)

Mentor: Nisarg Chokshi (nmchoks2@ncsu.edu)

Background: Expertiza has Assignment objects, which represent an assignment that is done by some number of users. For some assignments, students need to select a topic before submitting work.

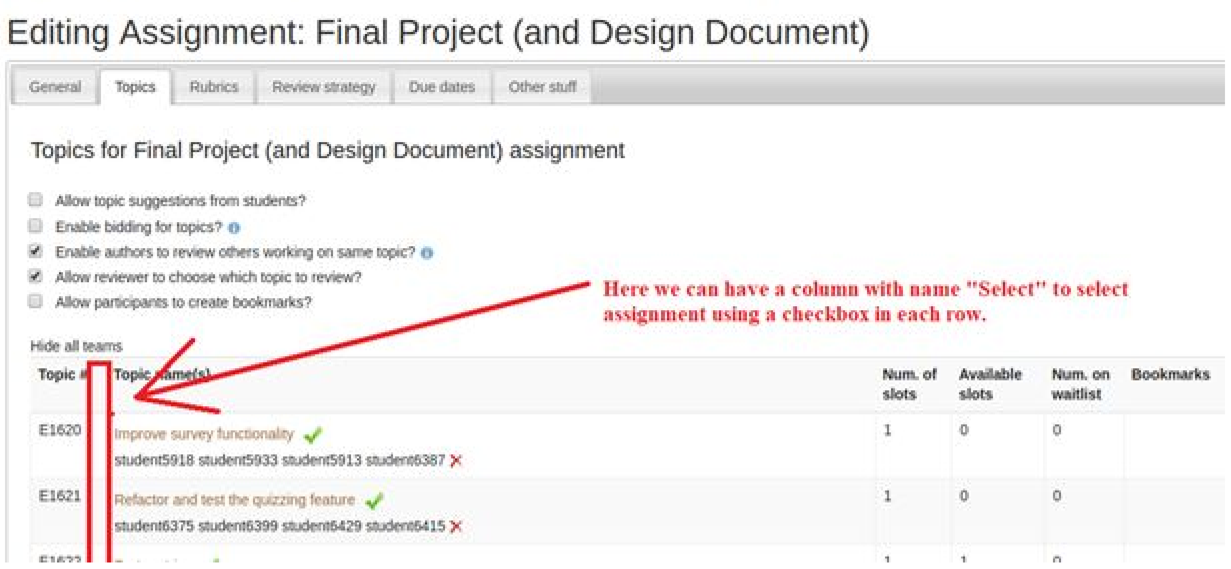
Topics associated with an assignment be reached as follows:
1. Log in as an instructor or a TA.
2. Select Manage->Assignments, which will bring up a list of assignments.
3. Click on "edit logo" under the "Action" column of an assignment.
4. Click on "Topics" tab in edit assignment page.

Issue1409
What is wrong:

If an instructor or a TA wants to delete topics he has to delete one topic at a time and has to wait for the page to refresh and then (s)he can proceed to delete the next topic, topics can only be deleted one by one.

What needs to be done:

A good fix would be to delete selected topics.
1. There should be a checkbox column, along with other columns in "Topics" tab, where a user can select the topics (s)he wants to delete.
2. There should be a delete button/link at the end of the topic table with the name "delete selected topics" to delete the selected topics after a confirmation, prompted post clicking the button/link.
3. There should be a button/link alongside "delete selected topics" by the name "Select all" so that a user can select all and delete them in one go after clicking on "delete selected topics".


Editing Assignment: Final Project (and Design Document)

Topics for Final Project (and Design Document) assignment

☐ Allow topic suggestions from students?
☐ Enable bidding for topics?
☑ Enable authors to review others working on same topic?
☑ Allow reviewer to choose which topic to review?
☐ Allow participants to create bookmarks?

Here we can have a column with name "Select" to select assignment using a checkbox in each row.

Hide all teams

Topic #	Topic name(s)	Num. of slots	Available slots	Num. on waitlist	Bookmarks
E1620	Improve survey functionality ✓ student5918 student5933 student5913 student6387 ✗	1	0	0	
E1621	Refactor and test the quizzing feature ✓ student6375 student6399 student6429 student6415 ✗	1	0	0	
E1622		1	1	0	

This project was done on E1951, but the code did not work, so it was not merged. Here is what was submitted.

- http://wiki.expertiza.ncsu.edu/index.php/CSC/ECE_517_Fall_2019_-_E1951._Remove_multiple_topics_at_a_time
- https://github.com/expertiza/expertiza/pull/1520
- https://github.com/nbPeterWang/expertiza/tree/beta

Here are our comments on the project.

1.The code is readable, the naming is good with good comments .

2.Not working on staggered assignments (If I check the staggered assignment checkbox).

3.Shallow test cases.

4.Before it is merged, it needs a thorough testing on all types of assignments.

E2053. Tagging report for students

Mentor: Yunkai “Kai” Xiao (yxiao28@ncsu.edu)

Background:
Students might want to keep track of the number of review tags they have entered, and check whether all their review comments have been tagged. This might help them avoid missing tags they want to assign..

What needs to be done:
Possibly this could be implemented in two different ways, depending on whether the student was

currently tagging a review.

- If the student is tagging a review, then a Javascript counter would count the number of tags assigned on this page.
- If the student is off the page but wants to inquire how many of his/her review comments have already been tagged, the code would do a query for the number of answer_tags with user_id = the current user's id, and answer_id's response_id field pointing to a response_map whose reviewed_object_id field specifies the current assignment.

The total count of tags should be shown in the heatgrid, probably at the top or the bottom (e.g., left header: # of comments tagged, in each cell, something like "20/32".This is best put into the heatgrid code, which should be usable by both instructor and student views.

Note: Write RSpec unit tests for the code affected or added.

References:

- https://github.com/expertiza/expertiza/issues/1468

This project was done in E1953, and the following submission was made:
http://wiki.expertiza.ncsu.edu/index.php/CSC/ECE_517_Fall_2019_-_E1953._Tagging_report_for_student
https://github.com/expertiza/expertiza/pull/1572
https://github.com/sameeradh/expertiza

The following issues were identified with their changes:

Travis CI checks are failing.
The functionality has been implemented correctly ans does give a correct count and the count increases on tagging more review comments.
Code changes breaking existing functionality makes it not mergeable.
The number of tags should be depicted as "7 out of 25".
The number of tags for each assignment should be placed in the HeatGrid.

E2054. Auto-generate submission directory names based on assignment names

Mentor: Sanket Pai (sgpai@ncsu.edu)

Background:
Each assignment should have its own submission directory.

Files involved: submitted_content_controller.rb

What needs to be done:

- The directory name should be auto-generated from the assignment name.
- It should be done by changing spaces in the names to underscores. E.g., the directory for Program 1 is by default "Program_1".
- A check should be added to prevent two assignments in the same course from having the same name.
- Verify or add if not present - a check to stop two assignments from sharing the same directory.

References:

- https://github.com/expertiza/expertiza/issues/1201

This project was done in E1954, and the following submission was made:

https://drive.google.com/open?id=1Xcr7tfVQU9xFmeEcR3iuIWUzuzjVsUNK
https://drive.google.com/open?id=1O81Er0QrVceDiVhHVIC57BDToVFuIgDN

http://wiki.expertiza.ncsu.edu/index.php/CSC/ECE_517_Fall_2019_-_E1954._Auto-generate_submission_directory_names_based_on_assignment_names#References
https://github.com/anujak30/expertiza-1
https://github.com/expertiza/expertiza/pull/1574

The following issues were identified with their changes:

The code has no comments at all.
The code breaks on changing the name of the assignment while creating it. It throws NoMethodError in AssignmentsController#create

Note: Write RSpec unit tests for the code affected or added.

E2055. Write unit tests for student_task.rb

Mentor: Sanket Pai (sgpai2@ncsu.edu)

Files involved: student_task.rb + student_task_spec.rb

What it does: `student_task.rb` is used to prepare data for student tasks page.

What's wrong with it: There are insufficient unit tests for `student_task.rb`. Path coverage is only 43.0% with 43 lines covered and 57 lines missed.

What needs to be done:
- Write RSpec unit tests to make the path coverage above 90%.
- Cover as many edge cases as you can.
- Achieve as high branch coverage as you can. We will use the mutant-rspec gem to measure test thoroughness and fault-finding capability of your tests.

Note:
- You can run RSpec tests by executing the command

`rspec spec/models/student_task_spec.rb`

- After that, you can see the detailed coverage information by opening `coverage/index.html` file in your Expertiza folder.

This project was done in E1955, and the following submission was made:

https://github.com/tusharkundra/expertiza
http://wiki.expertiza.ncsu.edu/index.php/CSC/ECE_517_Fall_2019_-_E1955.Write_unit_tests_for_student_task.rb
https://drive.google.com/file/d/1O-1k3mqo1yRMp0yxgTIuM3P5cegB5DCy/view
https://github.com/expertiza/expertiza/pull/1549

The following issues were identified with their changes:

It is very difficult to determine if the tests are deep or shallow without any comments about what they are supposed to be testing. In general, tests that something is not nil are shallow tests.

E2056. Refactor and improve account_request_controller.rb

Mentor: Ed Gehringer, efg@ncsu.edu

Background: Prospective users are allowed to request Expertiza accounts over the web. When they are requested, the super-admin receives an email, and can go and approve or reject the request. The requester is then emailed as to whether the account has been approved, and if so, receives login credentials.

What's wrong with it: The implementation is tedious to use, since all account requests since the beginning of time appear on the request page, and the super-admin needs to scroll to the bottom to find the current request. Also, the method names are quite nonstandard, when a lot of the methods have standard CRUD functionality.

What needs to be done:

- `action_allowed?` needs to be adapted to use authorization utilities from Project E1915.
- It is not clear what `foreign` does. It needs a method comment.
- In general, the method names could be more CRUD-like. For example, `request_new` could be just `new` (since this is the `account_request_controller`, and it would be a new account request).
- `list_pending_requested` lists all of the requests in chronological order. This means that the current request shows up at the bottom. Since there have been hundreds of requests, this is very annoying, as one always has to scroll down to the bottom of the list to approve or reject the request. It would be better to …
    - show the requests in reverse chronological order;
    - by default, not display the requests that have already been accepted or rejected (though it should be possible for the user to override this); and

    - paginate the listing. Look through the repo for occurrences of “paginate”, and try to reuse code (perhaps create a common helper method that can be called by multiple classes).
    - Also, it would be better to have checkboxes next to each pending request, and then let the super-admin check the requests to be approved or rejected, and then hit an “Approve selected requests” or “Reject selected requests” button.
- `create_requested_user_record` is a fairly long method, and seems to go through several steps. Would it make sense to break it up into multiple methods? Or at least write comments on what each section of the method is doing.

E2057. Time travel Not Allowed..!!! Restrict TAs’ ability to change their own grade + limit file-size upload

Mentor: Yunkai “Kai” Xiao (yxiao28@ncsu.edu)

Issue #1412:

Issue:
If a person is listed as a TA in one course and as a student in another course, then if they navigate to the "Your scores" page of one of the assignments in which they are participating as a student, they can see a TA's view of that page - effectively allowing them to assign their own grade!

What’s wrong with it:
TAs should not be able to change their grades from the course that they participated as a student.

Files Involved:
app/controllers/course_controller.rb, app/controllers/grades_controller.rb, app/models/assignment.rb, app/models/ta_mapping.rb Edit the appropriate action_allowed methods.

What needs to be done:

Change the access control for TAs such that they can only modify the contents of the course they are added as TA

Issue #1351:

Issue:
Part b: A student can upload files with their submission. In some cases, students upload long videos that might not be necessary for the submission.

What’s wrong with it:

As there is no restriction on the files being uploaded, this is a security issue in Expertiza. Large files should be restricted. A student may also upload malware into the system affecting expertiza

Files Involved:
app/views/submitted_content/_submitted_files.html.erb,
app/controllers/submitted_content_controller.rb,

What needs to be done:

Limit the type (PDF, PNG, JPEG, JPG, ZIP, TAR, 7z, ODT, DOCX) and size (5MB) of the file to be uploaded. Design decision should be made very *carefully*. In the future, we would need this feature to be specific to assignment. That would mean instructors can specify type of file that can be submitted per assignment. The design students would develop needs to reviewed thoroughly before implementation.

References: https://github.com/expertiza/expertiza/issues/1412

https://github.com/expertiza/expertiza/issues/1351

Note: Write RSpec unit tests for the code affected or added.

This project was done in E1957, and the following submission was made:

https://github.com/blsk22/expertiza
http://wiki.expertiza.ncsu.edu/index.php/CSC/ECE_517_Fall_2019_-_E1957._Time_travel_Not_Allowed..!!!_Restrict_TAs%E2%80%99_ability_to_change_their_own_grade_%2B_limit_file-size_upload
https://github.com/expertiza/expertiza/pull/1602

The following issues were identified with their changes:

- Comments are added but is not easy to understand
- Poor choice of variable names
- Access check is only added to Instructor but not to "Teaching Assistant"

File Upload
- Harcoded 5 MB size limit in both front end and back end, could have added an attribute to assignment policy (but was optional)
- Too much javascript utilization to verify file type in Front end. Could have used ruby's inbuilt functionality to achieve this

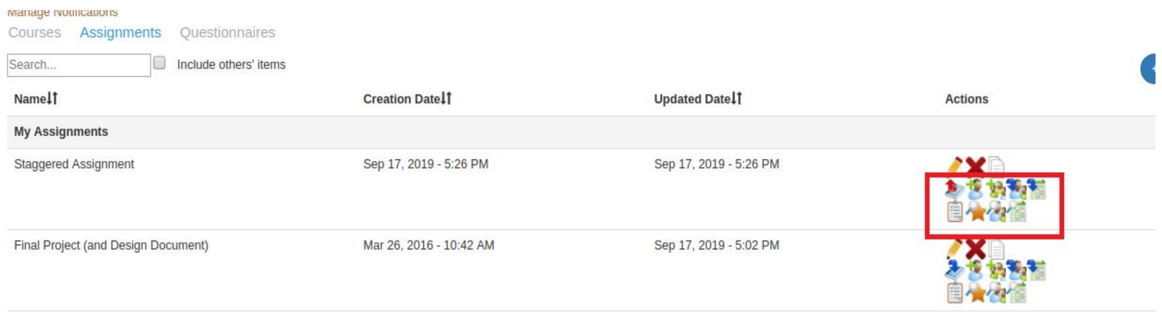
Only one test case is added and is not extensive

E2058. Two issues related to assignment management

Mentor: Yunkai “Kai” Xiao (yxiao28@ncsu.edu)

Background: Expertiza has Assignment objects, which represent an assignment that is done by some number of users. An instructor or a TA can perform several kinds of operations on assignments, such as, ”Add participants”, “Create teams”, and “View scores”. These operations can be seen:

First, on the homepage, under the “Actions” column in the assignment list when a user (instructor or TA or admin) logs in and navigates to Manage -> Assignments.

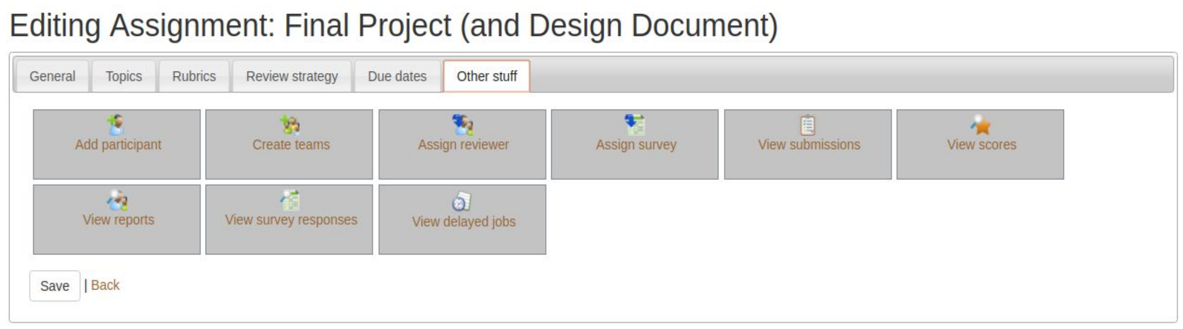


Courses Assignments Questionnaires

Search... Include others' items

Name	Creation Date	Updated Date	Actions
My Assignments			
Staggered Assignment	Sep 17, 2019 - 5:26 PM	Sep 17, 2019 - 5:26 PM	
Final Project (and Design Document)	Mar 26, 2016 - 10:42 AM	Sep 17, 2019 - 5:02 PM	

Second, when editing an assignment (by clicking on the edit logo above), on the tab called “Other stuff”.



Editing Assignment: Final Project (and Design Document)

General | Topics | Rubrics | Review strategy | Due dates | Other stuff

Add participant | Create teams | Assign reviewer | Assign survey | View submissions | View scores

View reports | View survey responses | View delayed jobs

Save | Back

Issue 1384

What needs to be done:

Implement a setting in the user’s (instructor/TA) profile, from where the user can choose whether to see these actions on the homepage or in a tab associated with each assignment.

Issue 1430

Under the “General” tab of the assignment edit page, an instructor or a TA can move an assignment to a different course.

What is wrong:

1. A TA or an instructor can assign an assignment to any course even when they don't have access to the course.
2. TAs can unassign an assignment from the course, and if they do so, they lose access to the assignment.

What needs to be done:

1. Only those courses should be shown in the dropdown list of courses, the assignment is part of and the instructor or TA has access to.
2. Instructors, but not TAs, would then be allowed to change an assignment to be part of no course.

Also, change the name of the tab from "Other stuff" to "Etc.".

This project was done in E1958, and the following submission was made:
http://wiki.expertiza.ncsu.edu/index.php/CSC/ECE_517_Fall_2019_-_E1958._Two_issues_related_to_assignment_management
https://github.com/expertiza/expertiza/pull/1594
https://github.com/expertiza/expertiza/pull/1603
https://github.com/Anmoldesai1997/expertiza

The following issues were identified with their changes:

Created two pull request for the separate issue.
Issue 1430 Pull Request 1603: It has small code change but the Code coverage decreased by 13.7%, has changes that fixes the issue.
Issue 1384 Pull Request 1594 : Does not resolve the issue 1384, this PR is messy, has code changes for both the issues.

E2059. Email notification to reviewers and instructors

Mentor: Ed Gehringer (efg@ncsu.edu)

What's wrong: Expertiza is supposed to email authors each time a review of their work is submitted. It is supposed to email reviewers each time an author that they have reviewed submits new work. Those emails are enabled by default; however, any user can turn off by unchecking boxes in their profile.

However, these features have never worked reliably. Two Gitub issues (87, 1330) document the failures. In addition to fixing any bugs, two interventions should be undertaken to make them more reliable.

- Appropriate tests should be written to check the functionalities that are likely to fail (i.e., not shallow tests).
    - Check to see that the Rails message that queues the email to be sent, is actually

being sent.
    - Check to see that the recipient of this message is the correct recipient.
    - Check to see that the body of the message has the correct content.
- The instructor should be able to get copies of emails that are sent by the system. In fact, there is currently a bit in the instructor's profile that is supposed to control this, but this feature too has proved unreliable. Project E1834 was the best previous attempt at addressing this issue; links are below.
    - https://github.com/pratik-abhyankar/expertiza
    - https://github.com/expertiza/expertiza/compare/master...pratik-abhyankar:master
    - https://github.com/expertiza/expertiza/pull/1253
    - http://wiki.expertiza.ncsu.edu/index.php/CSC/ECE_517_Fall_2018/E1834_Improve_email_notifications
    - Here is feedback on that project
    - Ruby idiom: "if !" --> "unless". Method comments are needed on the new methods. is_in_final_round uses magic constant 2 for a deadline_type_id. Not clear at all! Moreover, some code is duplicated! A lot of apparent changes were just indentation, which makes the diff hard to read.
    - Several reviewers say the build did not pass. One says, "I think User experience and code readability are as imp as producing a working code. So based on how urgent is the requirement for this functionality, I would prioritize refactoring the Notices and Display messages to more appropriately indicate what action was taken when we try to add a participant to the assignment or import using a CSV file. Secondly improving the comments as explained with an example in the first question."
    - You do not need to delete Gemfile.lock.
    - You are including debug code in your pull request, please remove it.
    - There are code changes, but no corresponding tests.
    - Please include tests if this PR introduces any modifications in behavior.
    - You should commit changes to the DB schema (`db/schema.rb`) only if you have created new DB migrations.
    - Please double check your code. If you did not aim to change the DB, please revert the DB schema changes.

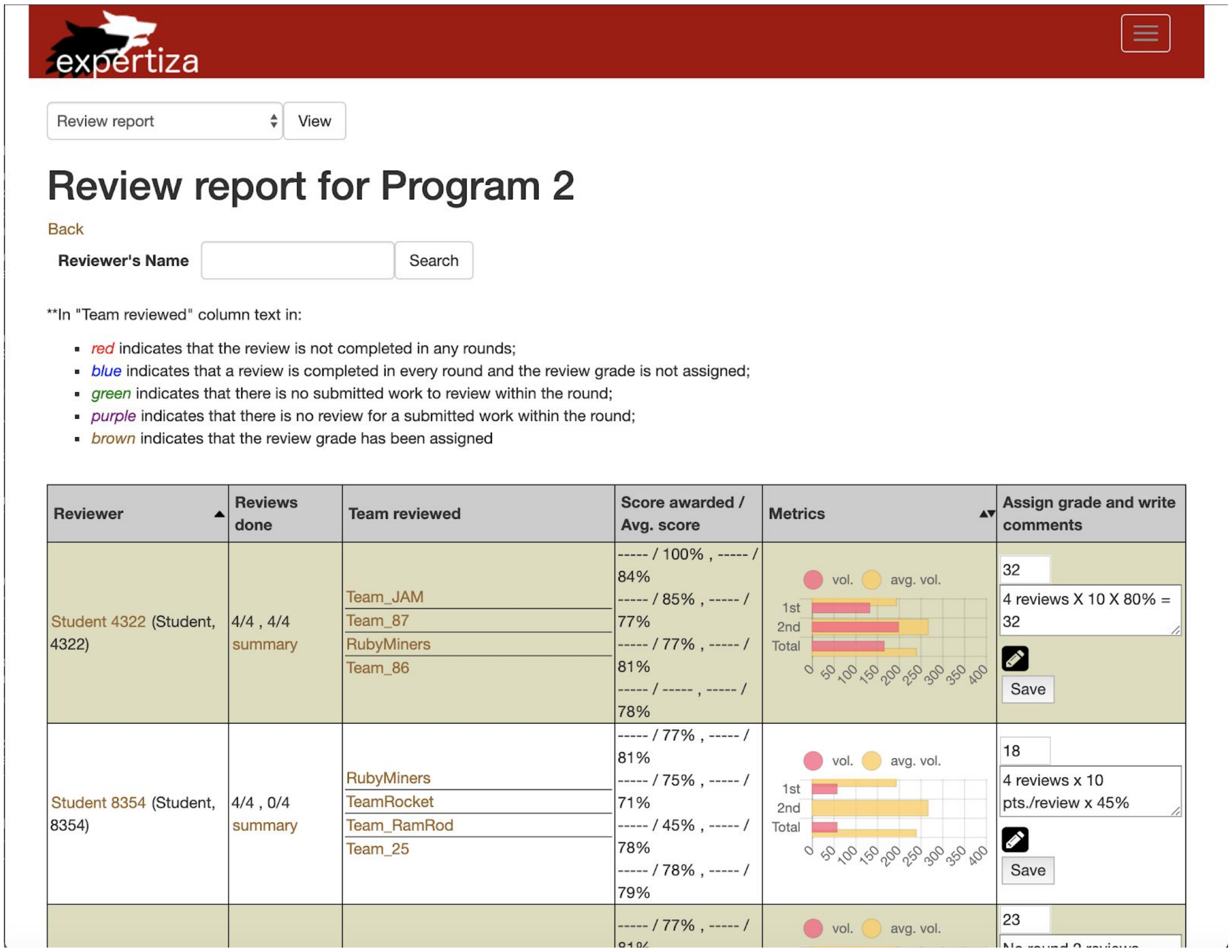
  So tests should be written for this feature too, to make sure that the instructor does get emailed.

E2060. Review report should link to the usual view for reviews

Mentor: Nisarg Chokshi (nmchoks2@ncsu.edu)

Background: Currently Review report uses its own code to display reviews. This a pretty basic view, and it does not interpret HTML codes. It should be changed so that it calls the usual code that is used for displaying reviews, that gives the circle with the score inside.
Issue #1478

Currently, if you pull up a review report,



Review report | View

Review report for Program 2

Back

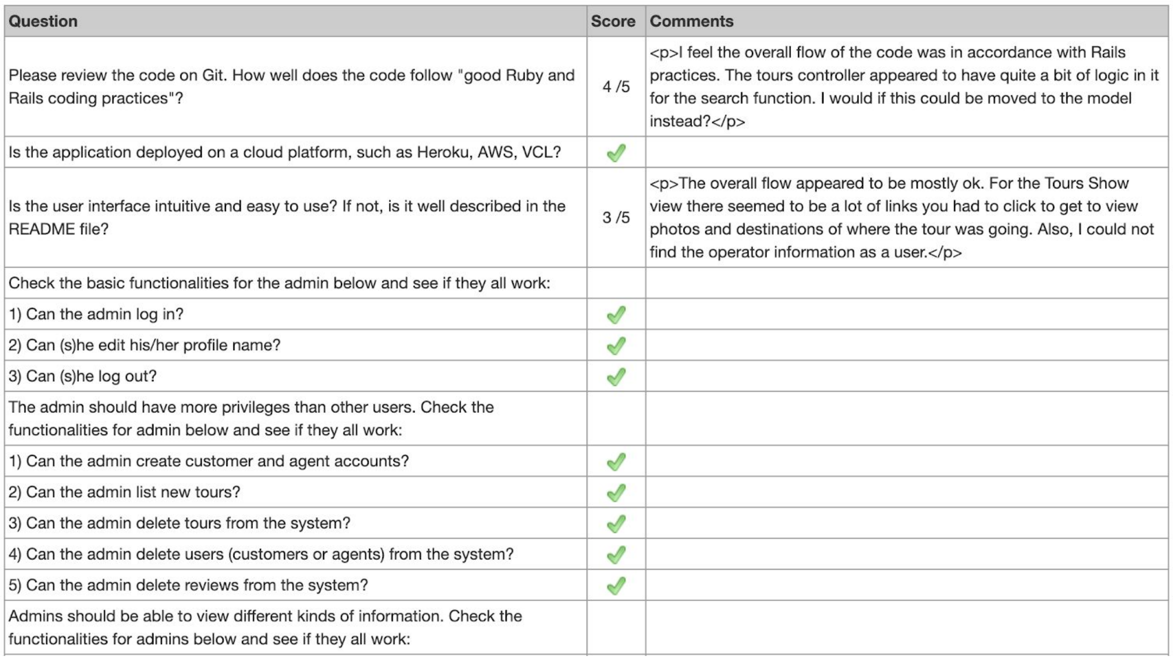
Reviewer's Name | Search

**In "Team reviewed" column text in:

- *red* indicates that the review is not completed in any rounds;
- *blue* indicates that a review is completed in every round and the review grade is not assigned;
- *green* indicates that there is no submitted work to review within the round;
- *purple* indicates that there is no review for a submitted work within the round;
- *brown* indicates that the review grade has been assigned

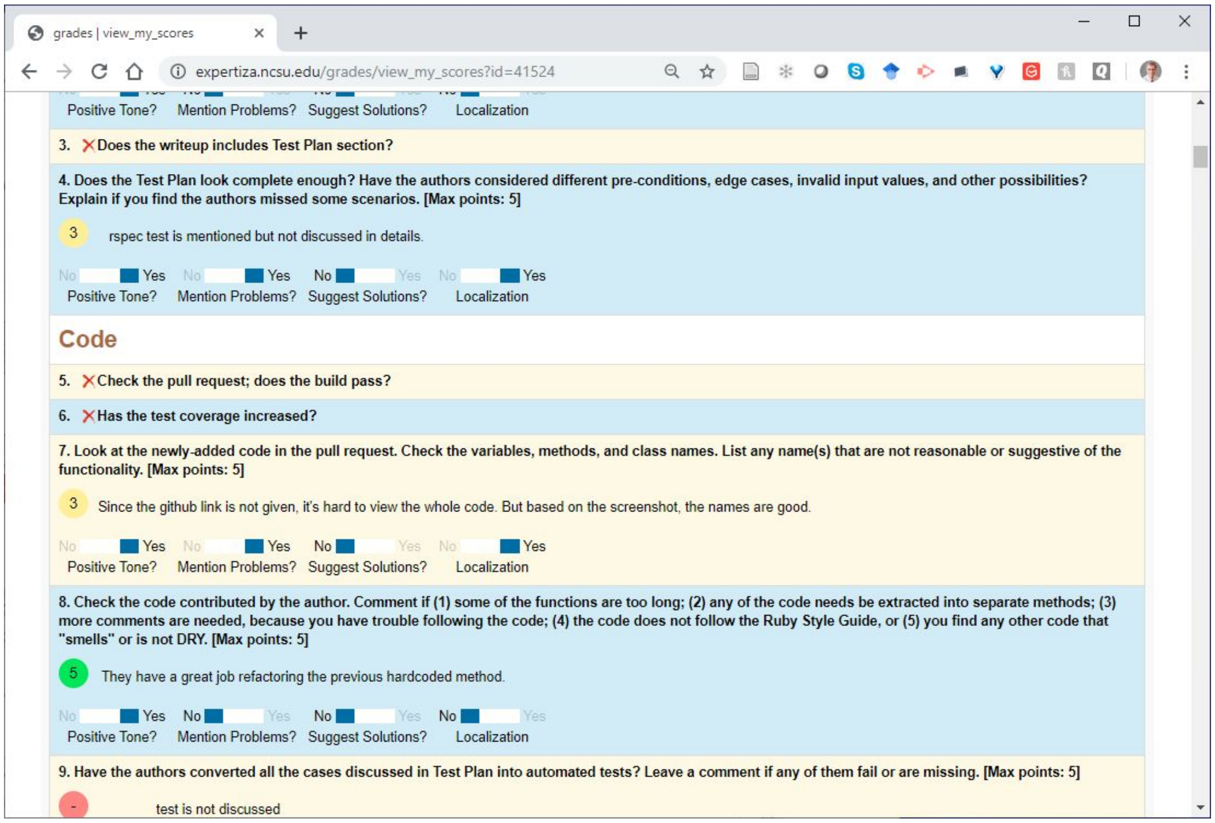
Reviewer	Reviews done	Team reviewed	Score awarded / Avg. score	Metrics	Assign grade and write comments
Student 4322 (Student, 4322)	4/4 , 4/4 summary	Team_JAM Team_87 RubyMiners Team_86	----- / 100% , ----- / 84% ----- / 85% , ----- / 77% ----- / 77% , ----- / 81% ----- / ----- , ----- / 78%	● vol. ● avg. vol.	32 4 reviews X 10 X 80% = 32 Save
Student 8354 (Student, 8354)	4/4 , 0/4 summary	RubyMiners TeamRocket Team_RamRod Team_25	----- / 77% , ----- / 81% ----- / 75% , ----- / 71% ----- / 45% , ----- / 78% ----- / 78% , ----- / 79%	● vol. ● avg. vol.	18 4 reviews x 10 pts./review x 45% Save
			----- / 77% , ----- /	● vol. ● avg. vol.	23

And then click on one of the teams reviewed, e.g., the first one, you get a report that looks like this:



Question	Score	Comments
Please review the code on Git. How well does the code follow "good Ruby and Rails coding practices"?	4 /5	<p>I feel the overall flow of the code was in accordance with Rails practices. The tours controller appeared to have quite a bit of logic in it for the search function. I would if this could be moved to the model instead?</p>
Is the application deployed on a cloud platform, such as Heroku, AWS, VCL?	✓	
Is the user interface intuitive and easy to use? If not, is it well described in the README file?	3 /5	<p>The overall flow appeared to be mostly ok. For the Tours Show view there seemed to be a lot of links you had to click to get to view photos and destinations of where the tour was going. Also, I could not find the operator information as a user.</p>
Check the basic functionalities for the admin below and see if they all work:		
1) Can the admin log in?	✓	
2) Can (s)he edit his/her profile name?	✓	
3) Can (s)he log out?	✓	
The admin should have more privileges than other users. Check the functionalities for admin below and see if they all work:		
1) Can the admin create customer and agent accounts?	✓	
2) Can the admin list new tours?	✓	
3) Can the admin delete tours from the system?	✓	
4) Can the admin delete users (customers or agents) from the system?	✓	
5) Can the admin delete reviews from the system?	✓	
Admins should be able to view different kinds of information. Check the functionalities for admins below and see if they all work:		

We need to link the views to existing templates which are being used here :



Positive Tone? Mention Problems? Suggest Solutions? Localization

3. ✗ Does the writeup includes Test Plan section?

4. Does the Test Plan look complete enough? Have the authors considered different pre-conditions, edge cases, invalid input values, and other possibilities? Explain if you find the authors missed some scenarios. [Max points: 5]

3 rspec test is mentioned but not discussed in details.

No Yes No Yes No Yes No Yes
Positive Tone? Mention Problems? Suggest Solutions? Localization

Code

5. ✗ Check the pull request; does the build pass?

6. ✗ Has the test coverage increased?

7. Look at the newly-added code in the pull request. Check the variables, methods, and class names. List any name(s) that are not reasonable or suggestive of the functionality. [Max points: 5]

3 Since the github link is not given, it's hard to view the whole code. But based on the screenshot, the names are good.

No Yes No Yes No Yes No Yes
Positive Tone? Mention Problems? Suggest Solutions? Localization

8. Check the code contributed by the author. Comment if (1) some of the functions are too long; (2) any of the code needs be extracted into separate methods; (3) more comments are needed, because you have trouble following the code; (4) the code does not follow the Ruby Style Guide, or (5) you find any other code that "smells" or is not DRY. [Max points: 5]

5 They have a great job refactoring the previous hardcoded method.

No Yes No Yes No Yes No Yes
Positive Tone? Mention Problems? Suggest Solutions? Localization

9. Have the authors converted all the cases discussed in Test Plan into automated tests? Leave a comment if any of them fail or are missing. [Max points: 5]

- test is not discussed

This project was done in E1958, and the following submission was made:
http://wiki.expertiza.ncsu.edu/index.php/CSC/ECE_517_Fall_2019_-_Project_E1965._Review_r

eport_should_link_to_the_usual_view_for_reviews#Team_members
https://github.com/expertiza/expertiza/pull/1532
https://github.com/ljxxcaijing/expertiza/tree/beta
https://youtu.be/Kd6iaHIT3Y8

The following issues were identified with their changes:

1) Code for displaying a review is essentially copied instead of parametrized and reused, violating the DRY principle.
2) No new tests added although they do have a well detailed manual test plan.
3) HTML codes (<b>) are left visible in the view.

E2061. Refactor questionnaires_controller.rb

Mentor: Nisarg Chokshi (nmchoks2@ncsu.edu)

Background: In Expertiza, `Questionnaire` is the superclass for all kinds of questionnaires and rubrics—rubrics for evaluating submissions and teammate contributions, and taking quizzes and surveys. All of these are subclasses of `Questionnaire`. questionnaires_controller.rb is charged with creating, displaying, and managing `Questionnaires`. Because it is used so widely, malfunctions could cause bugs in many parts of the system; hence it is especially important that the code be clear.

In recent years, questionnaires_controller.rb has been refactored repeatedly. It is a lot clearer than it used to be. But rough edges still remain.

- There is a method called create_questionnaire used. Is it used? There are no immediately apparent calls to it. The name is wrong (should be just `create`, but there is another, different create method). If not, remove `create_questionnaire` and the tests that test it. [It is not used, but it assigns a creator ID (instructor of the class) whereas create does not.]
- The create method itself is 49 lines long. This is much too long to be viewable at a glance. Break it up into other methods whose names are clear enough to be self-documenting.
- There are three checks for whether a questionnaire is a `QuizQuestionnaire`. Testing the class of an object is always suspect, though in a controller, which has to control a whole hierarchy of objects, it is not necessarily wrong. However, they probably aren't necessary.
    - One of the checks is in `create_questionnaire`.
    - One of them deals with weights for a question, which is a kludge or hack, since managing weights should really be done in the model (superclass `Questionnaire` or subclass `QuizQuestionnaire`).

    - The third check has something to do with `TreeNodes`, which are Expertiza's way of displaying objects (such as courses and assignments) that need to be shown in a hierarchy. It's not clear what's going on here, but I doubt that `QuizQuestionnaire` really has to be special.
- Like the weights referred to above, there are other values that are hardwired into the code, especially in `add_new_questions`. A 0–5 scale for rubric criteria should not be wired in, nor should min and max labels. These variables (`alternatives, min_label`, `max_label`) should be, and probably are, set somewhere in the UI. Textfield sizes are OK to default to the sizes given, but they should be defined constants, not literal values in an assignment statement.
- Several methods have a `questionnaire_id` as a parameter. It's not clear that it's necessary, since it seems just to be `params[:id]` for the questionnaire in use.
- BTW, you can check on the purpose of any field in almost any table of the db here.

Other issues come from Code Climate.

Method save_all_questions has a Cognitive Complexity of 8 (exceeds 5 allowed). Consider refactoring.

New

```
 def save_all_questions
   questionnaire_id = params[:id]
   begin
     if params[:save]
       params[:question].each_pair do |k, v|
```

Severity: Minor
Found in app/controllers/questionnaires_controller.rb - About 45 mins to fix

Assignment Branch Condition size for save_all_questions is too high. [26.78/15]

Severity: Minor
Found in app/controllers/questionnaires_controller.rb by rubocop

Assignment Branch Condition size for copy is too high. [20.64/15]

New

```
 def copy
   begin
     instructor_id = session[:user].instructor_id
     @questionnaire = Questionnaire.copy_questionnaire_details(params, instructor_id)
     p_folder = TreeFolder.find_by(name: @questionnaire.display_type)
```

Severity: Minor
Found in app/controllers/questionnaires_controller.rb by rubocop

This project was done in E2001, and the following submission was made, but the fix did not work as intended:
https://github.com/Sattlert2/expertiza
https://github.com/expertiza/expertiza/pull/1702
https://expertiza.csc.ncsu.edu/index.php/CSC/ECE_517_Spring_2020_/_E2001_Refactor_Questionnaires_Controller

E2062. Add test cases to review_mapping_helper.rb

Mentor: Ed Gehringer (efg@ncsu.edu)

Background: Project E1948 from last fall, Refactor review_mapping_helper.rb, was done well, but the team did not write any tests, so we cannot consider merging it. The file contains about 25 methods for helping assign reviews and calculate scores. For many of the methods, it should be easy to set up test scenarios and test whether the method returns the right value or does the right thing. Also change `get_review_metrics` to `get_review_volume`.

Be sure to start with the code in the pull request below, not the beta branch.

- E1948. Refactor review_mapping_helper.rb
- https://github.com/expertiza/expertiza/pull/1526
- https://bit.ly/2NiWlgl
- https://github.com/ArshdeepSinghSyal/expertiza/tree/beta

Here is the feedback we gave Project E1948

Pros:

1) Good refactoring of code

2) Added comments in existing code which makes it easier to understand the code.

3) Resolved code climate issue – completed their task

4) Created methods doing only 1 functionality – good

Cons:

1) Asked the team to create test cases for all the new method created. I can't see those.

2) Even though overall method is tested by the original test cases, it is always mandatory to write test cases for the new methods as well.

Reviews:

1) Appropriate variable and method name except naming convention not followed in naming variable teamID. It should be team_id

2) Few existing methods name are too long, this could have been changed by the team if they were working on that file - get_each_review_and_feedback_response_map, get_css_style_for_calibration_report

3) Test coverage decreased because they created new methods without writing test cases for them

Video:

1)Video showed the result of code climate which passed all the earlier failed test cases.

2) They could have showed working of the project as well in the video.

Tests:

1) No new test cases written.

2) Team should have written test cases for the new methods created

There has been an attempt to write the test cases for review_mapping_helper.rb in project E2007, but their build failed :

http://wiki.expertiza.ncsu.edu/index.php/CSC/ECE_517_Spring_2020_-_E2007_Add_test_cases_to_review_mapping_helper#References
https://github.com/expertiza/expertiza/pull/1689
https://github.com/jrgachie/expertiza
https://www.youtube.com/watch?v=jouVxnODyJk

Use this as reference to write your own tests and make sure the build passes.

E2063. Refactor tree-display.js and tree_display_controller.rb

Mentor: Saurabh Shingte (svshingt@ncsu.edu)

Background: The tree-display.js and its tree_display_controller.rb files are designed to allow Expertiza users to view their Assignments, Courses and Questionnaires at one place. This is the primary control page, as well as the home page for instructors on Expertiza which allows them to create, modify, delete and view Assignments.

The primary problem with this is that both the files, due to their bulky and unoptimized methods, slow the rendering of UI on screen. The methods in these files can be studied and refactored to improve the overall performance of this controller and its corresponding UI. Moreover, any obsolete or unused methods can be removed and DRY principle should be implemented. This project mostly revolves around these 2 files, and would involve refactoring JavaScript more than Ruby on Rails. Knowledge of JavaScript is a prerequisite for this project.

This project was done in E2013, but their changes did not focus on improving performance. The following submission was made:

http://wiki.expertiza.ncsu.edu/index.php/CSC/ECE_517_Spring_2020_-_E2013._Refactor_tree-display.js_a

nd_tree_display_controller.rb
https://github.com/bzhong2/expertiza
https://github.com/expertiza/expertiza/pull/1699

E2064. Refactor reputation_web_service_controller.rb

Mentor: Saurabh Shingte (svshingt@ncsu.edu)

Background: Expertiza allows student work to be peer-reviewed, since peers can provide more feedback than the instructor can. However, if we want to assure that all students receive competent feedback, or even use peer-assigned grades, we need a way to judge which peer reviewers are most credible. The solution is the reputation system. Reputation systems have been deployed as web services, peer-review researchers will be able to use them to calculate scores on assignments, both past and present (past data can be used to tune the algorithms).

For this project, our team's job is to refactor the file: reputation_web_service_controller.rb. This file is the controller to calculate the reputation scores. A "reputation" measures how close a reviewer's scores are to other reviewers' scores. This controller implements the calculation of reputation scores.

Issues to be fixed:

`reputation_web_service_controller.rb` is a fairly complex file. It contains a lot of methods that are long and hard to understand (for e.g. send_post_request). These methods need to be broken down into simpler and more specific methods that are easier to read/understand. Also, the few instances of code duplication that exist should be removed.

- There is a lot of unused/commented code, which should be removed.
- Figure out what the code is doing and write appropriate comments for it.
- Rename bad method names such as (`db_query`, `json_generator`). Method names should be verbs, and should say what the method does. Method names should not be so general that they could apply to many different methods..
  - In the case of `db_query`, the name should say what it queries for.
    - Also, this method not only queries, but calculates sums. Since each method should do only one thing, the code for calculating sums should be in another method.
    - And there should be comments in the code!
  - `json_generator` should be `generate_json`
    - There needs to be a method comment saying what the parameters are.
- In `send_post_request`, there are references to specific assignments, such as 724, 735, and 756. They were put in to gather data for a paper published in 2015. They are no longer relevant and should be removed.
  - `send_post_request` is 91 lines long, far too long.
- There is a password for a private key in the code (and the code is open-sourced!) It should be in the db instead.
- Fix spelling of "dimention"

- `client` is a bad method name; why is stuff being copied from class variables to instance variables?

E2065. Fix view in student_task/list page

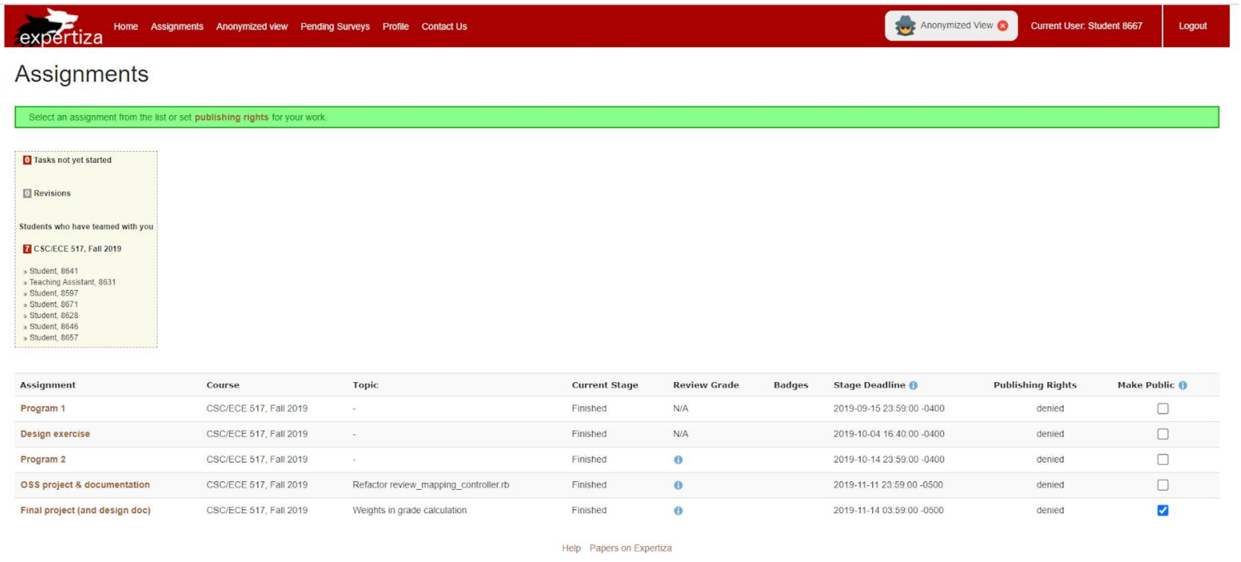
Mentor: Saurabh Shingte (svshingt@ncsu.edu)

Background: The `student_task/list` page is the page displayed to a student after logging in into expertiza. It has mainly two div(s), one to show the upcoming tasks along with some important information, another div is to display all the assignments and their information in which the student is participating. You should modify the view to remove extraneous information and make it clearer and more concise.

- You should consider the cases where a student might be enrolled in assignments from more than one course. In this case, assignments should be grouped by course, with the course name given before the listing of the first assignment. (This allows for the column for course to be removed, making the display more compact.
- In general, no column needs to be shown if there are no values in it. This would apply to "Badges" if no assignments in the list contain badges.)
- Also, it doesn't make sense to have a column for Review Grade, and not for the grade for submitted work ("Submission Grade").

You're also expected to write tests (maybe using capybara) to check the consistency of the page.

Current View:



Assignments

Select an assignment from the list or set **publishing rights** for your work.

0 Tasks not yet started

0 Revisions

Students who have teamed with you

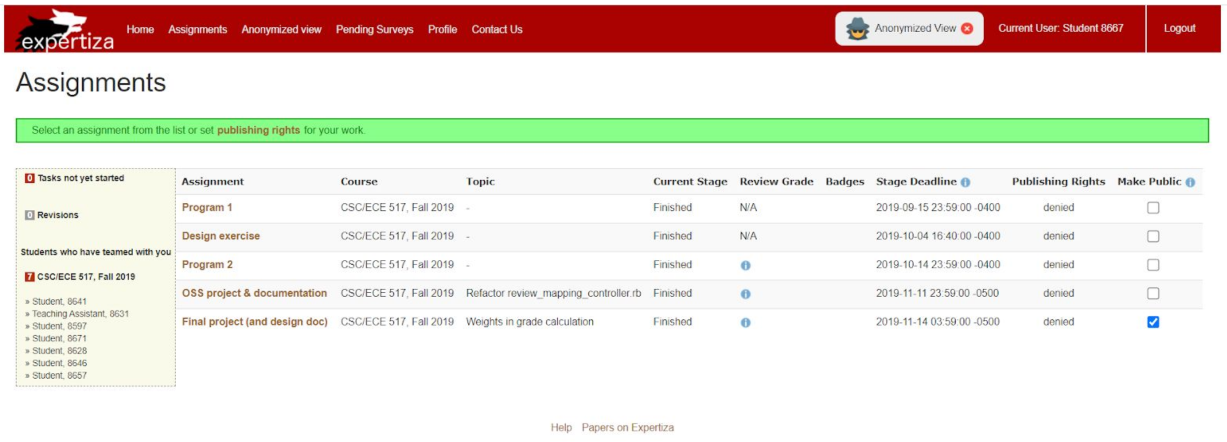
7 CSC/ECE 517, Fall 2019

» Student, 8641
» Teaching Assistant, 8631
» Student, 8597
» Student, 8671
» Student, 8628
» Student, 8646
» Student, 8657

Assignment	Course	Topic	Current Stage	Review Grade	Badges	Stage Deadline	Publishing Rights	Make Public
Program 1	CSC/ECE 517, Fall 2019	-	Finished	N/A		2019-09-15 23:59:00 -0400	denied	☐
Design exercise	CSC/ECE 517, Fall 2019	-	Finished	N/A		2019-10-04 16:40:00 -0400	denied	☐
Program 2	CSC/ECE 517, Fall 2019	-	Finished	ⓘ		2019-10-14 23:59:00 -0400	denied	☐
OSS project & documentation	CSC/ECE 517, Fall 2019	Refactor review_mapping_controller.rb	Finished	ⓘ		2019-11-11 23:59:00 -0500	denied	☐
Final project (and design doc)	CSC/ECE 517, Fall 2019	Weights in grade calculation	Finished	ⓘ		2019-11-14 03:59:00 -0500	denied	☑

Help Papers on Expertiza

Expected View (Rough estimation):



Assignments

Select an assignment from the list or set **publishing rights** for your work.

0 Tasks not yet started

0 Revisions

Students who have teamed with you

7 CSC/ECE 517, Fall 2019

» Student, 8641
» Teaching Assistant, 8631
» Student, 8597
» Student, 8671
» Student, 8628
» Student, 8646
» Student, 8657

Assignment	Course	Topic	Current Stage	Review Grade	Badges	Stage Deadline	Publishing Rights	Make Public
Program 1	CSC/ECE 517, Fall 2019	-	Finished	N/A		2019-09-15 23:59:00 -0400	denied	☐
Design exercise	CSC/ECE 517, Fall 2019	-	Finished	N/A		2019-10-04 16:40:00 -0400	denied	☐
Program 2	CSC/ECE 517, Fall 2019	-	Finished	ⓘ		2019-10-14 23:59:00 -0400	denied	☐
OSS project & documentation	CSC/ECE 517, Fall 2019	Refactor review_mapping_controller.rb	Finished	ⓘ		2019-11-11 23:59:00 -0500	denied	☐
Final project (and design doc)	CSC/ECE 517, Fall 2019	Weights in grade calculation	Finished	ⓘ		2019-11-14 03:59:00 -0500	denied	☑

Help Papers on Expertiza

E2066. Refactor lottery_controller.rb

Mentor: Aditya Govil (agovil@ncsu.edu)

Background: The lottery controller assigns teams to topics based on the priorities the team gave to each signup topic during the bidding process.

In more detail, each student starts off on a team of size 1. They can bid on signup topics by dragging and dropping them in priority order. Any team member can invite other users to join their team. If student2 from Team B joins Team A, which includes student1, then student2 will lose his/her bids and take on the bids of Team A and student1.

When the lottery controller is called to run its method `run_intelligent_assignment`, a web service is sent the bidding data from each team and returns a new list of teams, each of which is close to the maximum team size specified for the assignment. The web service coalesces teams that have similar bid data and then assigns those coalesced teams to topics, giving each team their top bid on a topic that hasn't been assigned yet. Teams with larger team sizes and more bids are assigned their topics first.

Issues to be fixed:

The controller has some methods like `create_new_team_for_bidding_response` and `remove_user_from_previous_team` that manipulate the team. They are more suited to being model methods. So, a new model can be made (named bid.rb) to which these methods can be moved.

As much as possible, the controller should contain only the standard CRUD methods. Anything else is a candidate for moving to a model class.

Unlike most of Expertiza, this class contains very good comments. When code is moved to a model class, the comments should be adapted as necessary and moved too. There is a comment that contains the word "bidded", which should of course be changed to "bid".

E2067. Refactor student_teams_controller.rb

Mentor: Aditya Govil (agovil@ncsu.edu)

Background: The `student_teams_controller.rb` controller used in Expertiza to manipulate teams that are created for assignments. Primary functions that this controller provides is create a new team, update team name and delete members from a team.

Issues to be fixed:

- The `action_allowed method` determines whether an action is permissible, depending on the privileges of the user who is performing it. In the `view` method, there is a clause that says, `return unless current_user_id? student.user_id`. This needs to be moved to the `when 'view'` clause in the `action_allowed method`.
- The code beginning at line 44 deals with `due_dates`, which are part of the "business logic" and should be moved to an appropriate model method. Thus, `@student.assignment.due_dates.each do |due_date|` should invoke an appropriate method in `due_date.rb`. It may or may not be necessary to add a method to the `DueDates` class.
- The variable `current_team` seems unnecessary; it only saves one character vs. `@student.team`. Consider removing it.
- The code on Line 53 (`@users_on_waiting_list ...`) is not clear at all. Needs at least a comment, and also the condition should not be so complicated.
- In line 55, the `@teammate_review_allowed`: condition is way too complex, uses magic constants, and belongs in code in another model class (maybe `due_date.rb`); no way should it be in the controller!
- Line 59: `existing_assignments` is evidently a team! So why is the variable name `existing_assignments`? It should perhaps be changed to `existing_teams`.
- The `update` method needs comments about what it is doing.
- Line 95: Why isn't `(matching_teams[0].name <=> team.name).zero?` just `(matching_teams[0].name == team.name)`?
- Line 108: `advertise_for_partners` needs a method comment. Since its body is only 1 line, does it make sense to have a separate method for this? It makes sense only if having a separate method improves readability. Ditto for the `remove_advertisement` method at line 112.
- The name `sign_up` needs to be `signup`; it is being used as a noun (`sign_up` would suggest a verb, the action of signing up).
- `remove_participant` is generally good code, but handling of waitlists is again model activity, and should be moved to a model class, perhaps `waitlist.rb`.
- Line 169: The `review` method needs a method comment; its purpose is not clear.

E2068. Refactor quiz_questionnaires_controller.rb

Mentor: Sanket Pai (sgpai@ncsu.edu)

Background: quiz_questionnaires_controller.rb is used in Expertiza to handle all functionality related to quizzes. A quiz is a type of questionnaire that allows reviewees to interact with their reviewers and making sure they read the submissions before reviewing. The student creating a quiz is supposed to ask questions related to their work, which, ideally, a reviewer should be able to answer. (If a reviewer cannot answer the questions about the reviewed work, then we might doubt the quality of that reviewer's review.) This controller needs some changes as detailed below.

Issues to be fixed:

- Change the way `min_question_score` and `max_question_score` are set for `@questionnaire` on lines 39-40, as well as on lines 53-54.
  - These statements set the max and min scores to 0 and 1 regardless of what the user enters, which is not intended.
  - Change it so that the values are set according to what the user enters from the UI.
  - Refer to the code for setting similar values in questionnaires_controller.rb, see if you can implement a helper method that can be used by both the controllers.
  - Also, both the pairs of statements are the same, and hence violate the DRY principle. Find the redundant part, and factor it out.
- Change the error message on line 78:
  - "some other students" → "one or more students".
  - change comma to semicolon.
- Consider lines 259-265, different methods are called with the same parameters (`question`, `choice_key`, `q_choices`) to create different types of questions, depending on `q_type`.
  - See if this code can be refactored to use polymorphism.
- Make appropriate changes to tests so that they pass after the above changes.

E2069. Refactor suggestion_controller.rb

Mentor: Sanket Pai (sgpai@ncsu.edu)

Background: The suggestion_controller.rb is used in Expertiza to control all functionality related to suggestion submission and approval for a new project topic that a student/team can submit and then work on if approved. This controller violates the Single responsibility principle by accessing/modifying members of other Model classes. Such functionality needs to be moved to appropriate classes.

Issues to be fixed:

- Move the method `create_new_team` (on line 94) to a more appropriate class.
  - Team creation should be handled by classes responsible for teams, for instance, Team.rb or AssignmentTeam.rb
- Move the method `approve` to SignupTopic.rb, since it is modifying fields belonging to SignupTopic directly, which should be done by a call to this model class.
- There are two methods named similarly: `approve` and `approve_suggestion`. Their functionality is not clear at first glance. Rename them so that they more accurately reflect their purpose. Add appropriate comments to explain the changes.
- Move the `send_email` method to the `Mailer` class in `app/mailers/mailer.rb.`
- In views/suggestion/show.html.erb and views/suggestion/student_view.html.erb, there seems to be a DRY violation which needs to be fixed.
  - Merge both files into a single view file in order to fix the DRY problem.
- Make appropriate changes to tests so that they pass after the above changes.

E2070. Refactor response_controller.rb

Mentor: Ed Gehringer, efg@ncsu.edu

Background: A *response* is the object that is created when someone fills out a review rubric, such as when one writes a review, gives feedback to a reviewer, or fills out a survey. Responses to the individual rubric items are kept in `Answer` objects; each `Answer` object has a `response_id` to say what `Response` it is part of. Since response_controller needs to work with many kinds of responses, its code is pretty general. It is not the worst controller in the system, but it would be much clearer if its method names were more descriptive of what they do.

Issues to be fixed:

- `def assign_instance_vars` This name could be more specific.
- `def scores` This should be a model method (in response.rb)! It is all calculation.
- `def new` This method contains a complicated condition that determines whether the submission has been updated since the last time it was reviewed. If it has not, then the reviewer can *edit* his/her previous review. If there has been an update, then the reviewer gets a new review form to "update" the review. It would make sense to have a model method that tests whether there has been a review since the last file or link was submitted. Then the code here would just call that function. It would be a lot clearer what the new method is doing. `set_content(new_response = false)` also plays a role in this calculation. Perhaps this method should also be included in the refactoring, for the sake of clarity of the resulting code.
- `def set_questionnaire` Badly named; what kind of questionnaire and why? The name should be a lot clearer.

- `def set_questionnaire_for_new_response` Badly named, no comments, not at all clear. The logic is not like `set_questionnaire`. Rename, refactor for clarity, and add comments as appropriate.
- `def show_calibration_results_for_student` This method makes about five db accesses. Can it be broken into 2 methods, with the business logic moved to `response.rb`?

E2071. Improve assessment360_controller.rb

Mentor: Ed Gehringer (efg@ncsu.edu)

Background: This controller provides functionality for displaying course-level data, specifically, review averages, and for each student who is a course participant, a list of the projects they worked on and the grades that they received. The reason it is called "assessment 360" is because that's a term that relates to evaluating performance by combining a lot of different perspectives into a single view. Its externally callable methods are `all_students_all_reviews` and `course_student_grade_summary`. I believe that all other methods are simply helper methods for these methods, but please check, because I'm not sure of it.

Issues to be fixed:

- `all_students_all_reviews` looks at metarviews and teammate reviews. It should be possible to view just one or the other. So the method should take a parameter specifying which kind(s) of review scores are to be displayed, and columns relating to reviews that are not being displayed should be suppressed, so they don't appear in the listing. [Could be expanded to list peer-review averages as well.]
- `@teamed_count` is a cryptic name for a variable; rename.
    - If it is not used, then simply remove it.
- `# If a student has not taken an assignment or if they have not received any grade for the same,`
  `# assign it as 0 instead of leaving it blank. This helps in easier calculation of overall grade`
    - Sure it's easier, but it's not accurate! Averages should only include assignments that were attempted.
- `course_student_grade_summary` should show average peer-review scores, as well as instructor-assigned scores.
- Missing/skipped assignments should be signified by a dash, not a hyphen (which is too hard to see).
- We would like to have the following columns potentially shown, depending on whether the corresponding box is checked
    - Instructor-assigned scores
    - Peer grades
    - Teammate review scores

    - Topics
    - Meta-review scores (only show if the set of deadlines has a meta-review deadline). `find_due_dates(metareview)` see the deadline_types table for the parameter name
    - Number of teammates [cumulative]

E2072. Refactor assignment_participant.rb

Mentor: Ed Gehringer, efg@ncsu.edu

Background: In addition to `User` objects, `Participant` objects are needed in Expertiza. This is because the system needs to be able to show the user a list of assignments that (s)he is participating in. If there were only `User` objects, the system would have no way of limiting the list of assignments shown to a user to just the assignments that the user is participating in.

Each `Participant` object specifies a `user_id` that indicates which user is participating, and a `parent_id` that refers to the `Course` or `Assignment` that the user is participating in. `Participant` has two subclasses, `CourseParticipant` and `AssignmentParticipant`, which indicate that a user is participating in a course or assignment, respectively. Of course, common functionality for the two classes should be implemented in the superclass `Participant`.

Issues to be fixed: Probably the biggest challenge is making score calculation more transparent.

- ~~Line 20: `validates` could be changed to use whitelisting~~
- There are variables that refer to review "volume"—the number of distinct words in a review. These are `:avg_vol_in_round_1`, `:avg_vol_in_round_2`, `:avg_vol_in_round_3`. This builds in an expectation that there can be no more than 3 rounds of review. In fact, Expertiza permits any number of rounds of review (even though no one has ever created an assignment with more than three). These variables should be changed to an array, both here and in `app/helpers/review_mapping_helper.rb`, where they are actually used. Of course, this means that the code that manipulates them has to be changed to a loop.
- `assign_quiz` really seems misplaced. Can't it be in quiz_questionnaire.rb?
- `review_score` should be able to compute the score for a particular round of review. For example, if there are two rounds of review, someone's score should not depend on what their reviewers wrote in Round 1.
- The `scores` method contains a lot of commented-out code. It is complicated, and calls several methods that purport to compute different kinds of scores. Score calculation should be rewritten to be more transparent.
- `compute_assignment_score`: We need better documentation of the methods that compute scores.
- `merge_scores`, `topic_total_scores`, `calculate_scores` also need method comments.

    - Looks like fixing score calculation could be a project in itself!
- `copy` is a misleading name for a method that copies a participant to a course.
- It seems unlikely that `reviews_by_reviewer` is still needed. It refers to reviews of this assignment_participant. But for the last 5 years or so, all reviews have been of assignment_teams, not assignment_participants.
    - Ditto for `get_logged_in_reviewer_id` and `files`.
    - (In general, method names beginning with `get_` are not Ruby-like, and should be changed.)
- The name `self.grant_publishing_rights` needs to be changed to `assign_copyright`. If it is set to true, that means that the student is allowing the instructor to use the work in some other project (e.g., in an open-source project like Expertiza).

E2073. Refactor course_controller.rb

Mentor: Srujana Rachakonda

Background: The `CourseController` handles creating, modifying, and deleting courses.

Issues to be fixed:

- The class should be named `courses_controller.rb`, to follow the current Rails convention that controllers should be named in the plural..
- Line 77: "`The copy is currently associated with an existing location from the original course.`" Does this mean that the copy is using the same submission directory as the original course? Then it should say that! The existing phrasing is confusing.
    - To explain, when students submit files (rather than links) to Expertiza, the files must be stored somewhere. The instructor is supposed to specify the path, and then Expertiza stores the files there. But suppose the instructor specifies a path that has been used for another assignment, or forgets to specify a path, causing a null path to be used. In either case, the new course will use the same directory as an existing course, which can cause files submitted for one assignment to show up in another assignment. So we absolutely have to enforce that the submission directory for this course is different.from the submission directory for all other courses.
        - I think we've already adopted that rule for assignments; check assignment_controller.rb and make sure that courses_controller.rb is consistent with it.
- `create` and `update` duplicate code. Duplicate parts should be moved to a partial or a private method.
- `create_course_node` should be a method of `course_node.rb`. It does not belong in a controller.