**E2415. Reimplement response[s]\_controller.rb**

**Mentor:** [Ameya Vaichalkar](mailto:agvaicha@ncsu.edu)

1. The naming convention for controllers dictates that they be named in the plural, but Expertiza predates that convention and uses the singular for a few controllers. The reimplementation is to follow convention.
2. The current ResponseController is very long, and has many functions besides the basic CRUD methods.
3. The authentication and authorization methods are up to date, and should be kept without change.
4. The edit method contains functionality to sort reviews, which would be better placed in the Response model. Also, it checks whether the reviewer is a team; it would be better to implement this functionality elsewhere, preferably by polymorphism, so it would not have to be checked via an if statement here and elsewhere.,

sort reviews-> in response.helper but not used

1. It uses multiple methods to set parameters (assign\_action\_parameters, set\_content).  Although these methods are much better than duplicating code elsewhere, there may be a more elegant way to assure that client methods have access to the objects that they need.

Set\_content and assign\_action\_parameters -> integrated to Set\_content in the response model

1. new\_feedback is used to create author feedback to send to a reviewer about a review.  It is good for it to be in this controller, since it also creates a Response object.
2. Items (Questions) always need to be displayed in order.  In the controller and helper, there are 4 calls to sort\_questions, whose implementation is only a single line.  Since items only need to be sorted right as they are being displayed, it is not clear why four calls are needed.  BTW, the standard term for components of a rubric is “item”, not “question”, since many of them are not framed as questions.  Please use the term “item” in your code.

4 calls to sort\_questions -> one call to sort\_questions in the ResponseHelper

Change questions to items

1. The method questionnaire\_from\_response\_map decides what kind of rubric (or quiz or survey) is in use.  There is a possibility that the review rubric may [depend on what topic is being reviewed](https://expertiza.csc.ncsu.edu/index.php/CSC/ECE_517_Fall_2021_-_E2147._Role-based_reviewing), and the method needs to handle this case.  Nonetheless, it may be possible to simplify it.

Questionnaire\_from\_response\_map -> get\_questionnaire\_by\_response

1. The controller also contains a method to generate an email from a reviewer to an author.  This is but one of several kinds of emails that are sent pursuant to responses.  Other emails are defined in response.rb.

Emails should be generated in a helper, which can be the same helper that response.rb uses.  Emails should be sent …

* to the team of authors when a reviewer submits a review of the authors’ work,
* to the reviewers when an author submits a new version of their submission,
* to the instructor when two review scores differ by more than a “threshold” amount, and
* from the reviewer to the author whenever the reviewer wants (authors should also be allowed to respond in a double-blind fashion to reviewers).

Notify\_instructor\_on\_difference, notify\_peer\_review\_ready -> moved to the ResponseHelper and created EmailObject, and Mailer

You should re-code the existing methods in a helper, but you need not implement email methods that are currently missing from Response and ResponseController.

**What to code.**  The new ResponsesController should create and edit responses.  As part of creating a response, it needs to create answer blanks (Answer objects) for each of the items on the rubric (or survey or quiz).  When creating the answer blanks, it needs to cycle through the Questions on the Questionnaire in the order specified by their sequence numbers.  I think it will be unnecessary to write an AnswersController, because Answer objects are created only in conjunction with Responses.  After all the Questions have been placed on the view, the “Additional comment” field of the Response object should be rendered.

The various Question objects may represent different kinds of rubric items.  Each kind is implemented in a subclass of Question (read about the subclasses [here](https://docs.google.com/document/d/11TqWr-L5qjd2lHSUwWKmvG4W5jNHy1j2QrZPxnzc7BQ/edit#heading=h.gczmekdk7815); search for “Fields in questions table”).  It is not your job to code these classes, but you do need to be careful to render the right kind of answer blank (checkbox, dropbox, text field, text area, etc.).  These are defined by instance methods in the Question subclasses.  You can assume that the same instance methods will be used as in the current Expertiza code.

The ResponsesController should also reimplement the methods that return information about the response, e.g., questionnaire\_from\_response, questionnaire\_from\_response\_map.  If you do not understand what some method is doing, you can search the repo to see if it is being used.  If it is never called (a “dead” method), then remove it.  If it is called, try to figure out what it is doing, and if in doubt, ask me. Methods that are in use should, in general, be re-written in the new ResponsesController.

Created new actions: show, create, update