QPED-O4 – Rubric Tool – *Manual*

Steffen Dick, Christoph Bockisch (Philipps-Universität Marburg)

1 General

Using the Rubric tool involves two jobs:

1. For each task (i.e., an assignment graded separately for each student solution), the html page "new task.html" must be used once to specify the meta-information of this task. By this, a json-file containing the meta-information is generated.
2. Second, the page "Rubric.html" is used to enter assessments for all solutions submitted to a specific task. For this, import the json-file from step one. For each student solution fill in the rubric. After all solutions are assessed, all results can be downloaded as CSV-file. Additionally, a feedback text can be generated for each student solution.

Where possible, the tool stores information on the current "assessment session" in the local store of the Web Browser. So usually, it is possible to continue a session after the window has been closed. **ATTENTION**: This is meant to be a safety mechanism. Instead of relying on this, the data should be exported at the end of the assessment session. The **local store is cleared** when the browser history is cleared. And the **local stored is tied to the URL** of the page; so, when the HTML page is moved or changed, the local store is likely lost.

The implementation of the Rubric Tool can be found on GitHub:
https://github.com/Alucard2112/QPED-O4

The tool is fully self-contained and only uses standard-features of HTML 5 browsers. Therefore, no installation is needed. Just download everything from Github (if you don't want to clone the repository, go to the Github web page, click the "Code" button and select "Download ZIP") to a convenient location on your local disk. The tool works **fully offline**.

Compatibility: The tool is **tested with Google Chrome**.

2 Defining new tasks

What exactly is the granularity of a "task" can be determined be the lecturer. At the coarsest granularity, there should be one task per (weekly) exercise sheet. But a task could also be a sub-assignment, so that there are multiple tasks per exercise sheet.

For a task specify the following things in the HTML page:

1. **Course** – Select the course to which the assignment belongs. A dropdown list provides all courses that we have decided to use in QPED IO4.
2. **Name of task** – This should be a unique identifier for an assignment within your course. It will be used to associate results to the corresponding assignment in the evaluations of

pilot studies. **Attention**: The task name will become part of the file name when data is exported. Therefore, **only use characters that are legal in file names**.

3. **Week of task** – The week of the course in which students have to solve the assignment. This is supposed to be an indicator of how advanced students are within the course.
4. **Max points** – Specify the maximum number of points that students can get for this assignment. It is an indicator of the weight of the assignment within your course. This is also used to automatically compute a score for a solution.
5. **Differentiation of background (TU/e)** – This is included for the TU/e course with three different branches and corresponding variants of assignments (challenging, regular, extra support). All other courses should ignore this and keep the default selection "regular".
6. **QPED deliverables** – Select all QPED deliverables that students have used for this assignment. Either because they were used to teach the topics that are practiced by this assignment, the assignment itself uses a deliverable or students may use a deliverable to solve the assignment. (Multiple selections can be made by pressing the Control or Command key.)
7. **Additional comments** – If you have some additional notes, you can to attach to the assignment specification.
8. **Include features**
    a. Select the check box for each rubric-feature according to which the solutions of the assignment should be assessed.
    b. For each feature, you can also specify a weight. This weight will be used by the automatic computation of a grade for a solution. The weights are all relative, i.e., they do not have to sum up to a specific value.
    c. The buttons "Toggle …" provide a shortcut to enable or disable multiple features at once.
9. After specifying all this information, press the "**Generate task**" button at the very end of the page. This will trigger a download of a json-file that contains this specification.

3 Filling in Rubric

If multiple persons (e.g., tutors) are involved in assessing the solutions, you can distribute the json-file to all graders.

3.1 Start of the grading/assessment session

1. **Upload new task** – To assess student solutions, open "Rubric.html" and load the json-file that was generated with "new task.html" (see above). To do so, use the "Upload new task" facility at the top right of the page.
2. **Select task** – If you continue an assessment session and already uploaded the corresponding json-file, it should **not be uploaded again**. In that case, use the "Select task" dropdown box at the top left. Only if you cleared your browser data in between, upload the task definition again.

3. **Grader** – You must enter the name of the person who is filling in the rubric.
   This name is stored in the local store and therefore it should only be necessary to fill this in once. This will become **part of the export file name**, so only use legal characters.
4. **Course year** – Enter the year in which the course started using 4 digits.
   This will be stored in the local store and does, thus, only have to be entered once. Make sure to update this, when you grade a different course.
5. **Course run** – Enter the course run, if the course runs multiple times per year. If there is only one run, enter '1', which is pre-filled as default.
   This will be stored in the local store and does, thus, only have to be entered once. Make sure to update this, when you grade a different course.
6. **Show Tooltips** – If this checkbox is activated, you can move the mouse over a short description of an example (see below), a long description will be shown. When you are familiar with the meaning of the examples, it is recommended to deactivate for a clearer view.

3.2 Grading/assessing a student solution

For each student solution of the same task, do the following:

1. **For each feature**, the rubric must be filled in.
   a. If (some of the) examples of fulfilling or failing the feature are applicable to the solution, select the corresponding check box.
   b. A score for the feature is automatically computed based on the number of selected (positive and negative) examples.
   c. The computed score can always be overridden by changing the value of the radio button.
2. **Maximum points** – The maximally available points for this assignment as specified for the task (see above). This is only shown here as a reminder for the grader.
3. **Calculated points** – This box shows two things. The average weighted score (a value between 1 and 4), i.e., based on the selected scores for all features under consideration of the weights specified for each feature for this task. And the computed points for this solution. This is computed by extrapolating the average score to the maximum points. This is, thus, a value between 0 and "maximum points" and rounded to the next half point.
4. **Achieved Points** – The points awarded to the solution by the grader. The grade may be influenced by additional criteria and not only the rubric features. If the calculated points should be used as the achieved points, they can be filled in by pressing the "Apply Computed" button.
5. **Additional comment** – You can enter extra, individual text here, which will be added to the generated feedback for students.
6. **Generate feedback** – When this button is pressed, a summary text is generated in the text field "Generated Feedback" in the bottom right. The feedback lists the selected positive and negative examples, the individual and average scores for the rubric, the achieved points, and (if present) additional comments. It is intended that this can be provided to students as feedback. When you click into the text field "Generated

Feedback" it is automatically copied to the clipboard, then you can directly paste it into a message the student whose solution has just been assessed.

7. **Next student** – **IMPORTANT**: You must press this button to save the assessment of the student solution that you have just filled in. Also, pressing this button resets all checkboxes, radio buttons and text fields.

3.3 End of a grading session

At the end of a grading or assessment session of a task, you should export all the data gathered for the solutions of this task.

1. **Export all feedback for chosen task** – This will provide a file for download that contains the assessments for all student solutions of the currently selected task. With the dropdown list, you can select whether a Json- or CSV-file is to be generated.

The Json-file is more structured and suitable for further programmatic evaluation of the assessment results. The CSV file, on the other hand, allows easy evaluation with a spreadsheet tool like Excel. The CSV file is generated such that each row has the same columns in the same order. Therefore, data from multiple different tasks or from different graders can simply be merged by copying the data in a row.

In addition to the information entered by users, each record contains a timestamp.

3.4 Unsaved changes and "Next Student", "Generate Feedback", "Export All Feedback"

The button "Next Student" only stored the changes and adds a new student solution, if data has already been entered. Otherwise, a message is shown that the tool did not go to the next student. It is considered a data entry if an example or a score has been selected, achieved points of a comment have been entered.

The button "Export All Feedback" exports the feedback of student solutions which have been stored previously by the "Next Student" button. If for the currently assessed student solution input has already been made, it also exports the current feedback. Additionally, the current feedback is persisted in the local store, but the tool does not advance to the next student solution.

When pressing "Generate Feedback", not only the feedback text field is filled in. This also explicitly persists the feedback for the current student solution, disregarding whether data was entered or not. Thus, by pressing this button, it is possible to explicitly store feedback information for the current task which only consists of the pre-filled information.

4 Data collection

To facilitate the joint evaluation of the data, you should provide the following at the end of a course run:

- All Json-files defining a task. I.e., the files generated with the "new task.html" page.
- All CSV files containing the feedback data, which have been generated with the export-functionality of the "Rubric.html" page. Please remember that each grader must create

an export for each task, which will each result in a separate file. All files from all graders must but provided.
- Optionally, you can also provide the corresponding Json-files for the feedback data. The same instructions apply as for CSV files.

I have created a folder on the Research Drive of QPED: “IO4 Validation/Rubric_Data”. All files should be uploaded there.

5 Adapting the tool or contributing

You can easily adapt the tool to your needs. And if a change is relevant for the other project partners, you can even contribute it. If you intend to make changes to the tool, you should fork the Github project. You can then apply changes in your own fork. For a documentation on forking, see here:
https://docs.github.com/en/get-started/quickstart/fork-a-repo

If other QPED partners could benefit from the change, you can contribute it by creating a pull request for our main repository:
https://docs.github.com/en/pull-requests/collaborating-with-pull-requests/proposing-changes-to-your-work-with-pull-requests/creating-a-pull-request

When you make changes, you should make sure that the format of the exported data does not change. Otherwise, it will not be possible to (easily) jointly evaluate data from all partners in the pilot study!

The file “Javascript/features.js” defines the features of the rubric, including the positive and negative examples. You can translate some of the text here, if you wish to generate feedback in your local language. You should **not change the values of any fields named “key”** for the examples or features. You must also **not remove any features or change their order**. If necessary, you can append features. But it will not be (easily) possible to consider the additional features in the joint evaluation.

It may also be interesting to tailor the following functions to your needs:

- main.js: function handleFeedbackButtonClick()
  This generates the feedback text.
- main.js: function computePointsPerFeature(feature)
  This computes the score based on the selected examples.