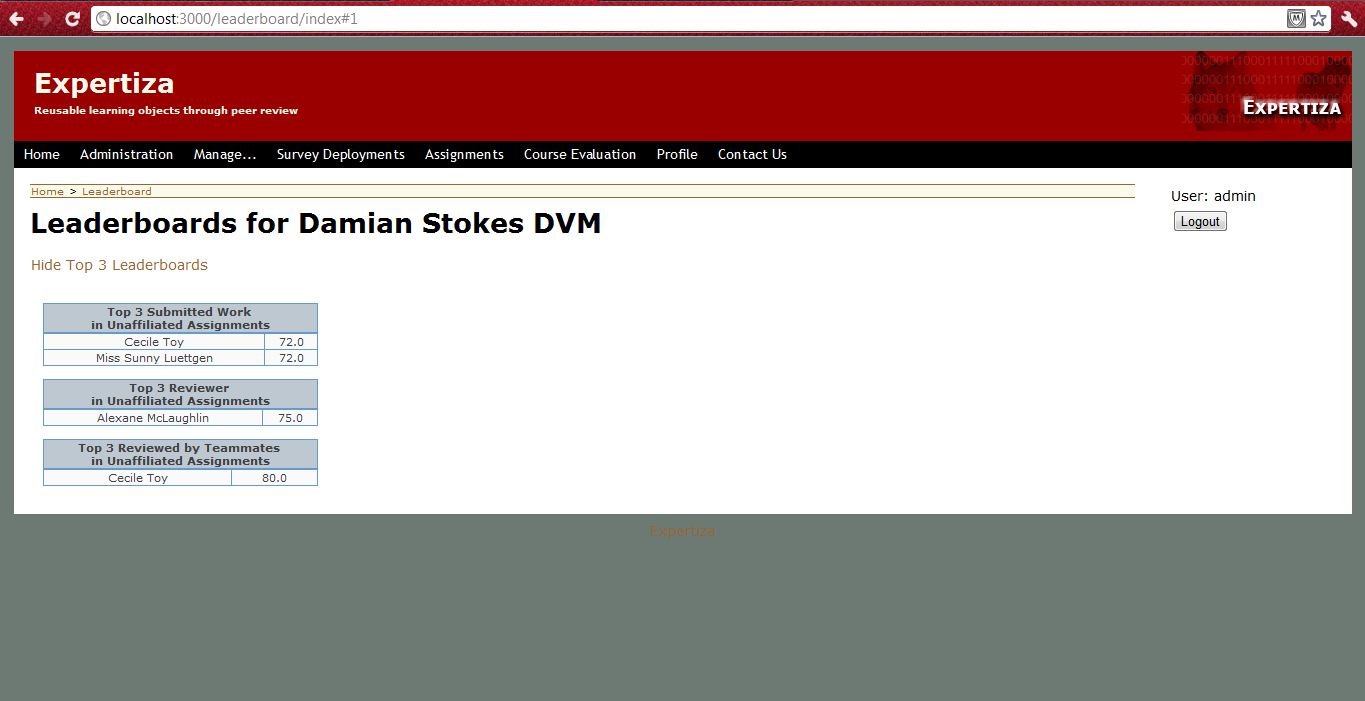
**Project E218, Refactor leaderboard.rb and score\_cache.rb**

Project Explanation:

The project was related to leaderboard and score\_cache. Below is brief explanation of what leaderboard is.

Expertiza is a Web-based system for peer-reviewing student work. After students submit their work by uploading a file or writing on a wiki, other students are assigned to review their work. Author and reviewer communicate in double-blind fashion using rubrics. The reviewer fills out a rubric that includes several questions about the author’s work. Each rubric question can be given a numeric score, and there is space for a prose comment on each question. The system computes a composite score based on the scores for the individual rubric questions. If we keep track of these scores for a whole class, we can display them—with the student’s permission—to others via a leaderboard. A leaderboard is simply a list of the leaders in a particular category, such as the students receiving the highest scores on reviews of their work, or the students whose feedback is rated most highly by the authors they have reviewed. Students will only appear on the leaderboard if they give permission (by editing their profile) for their scores to appear.



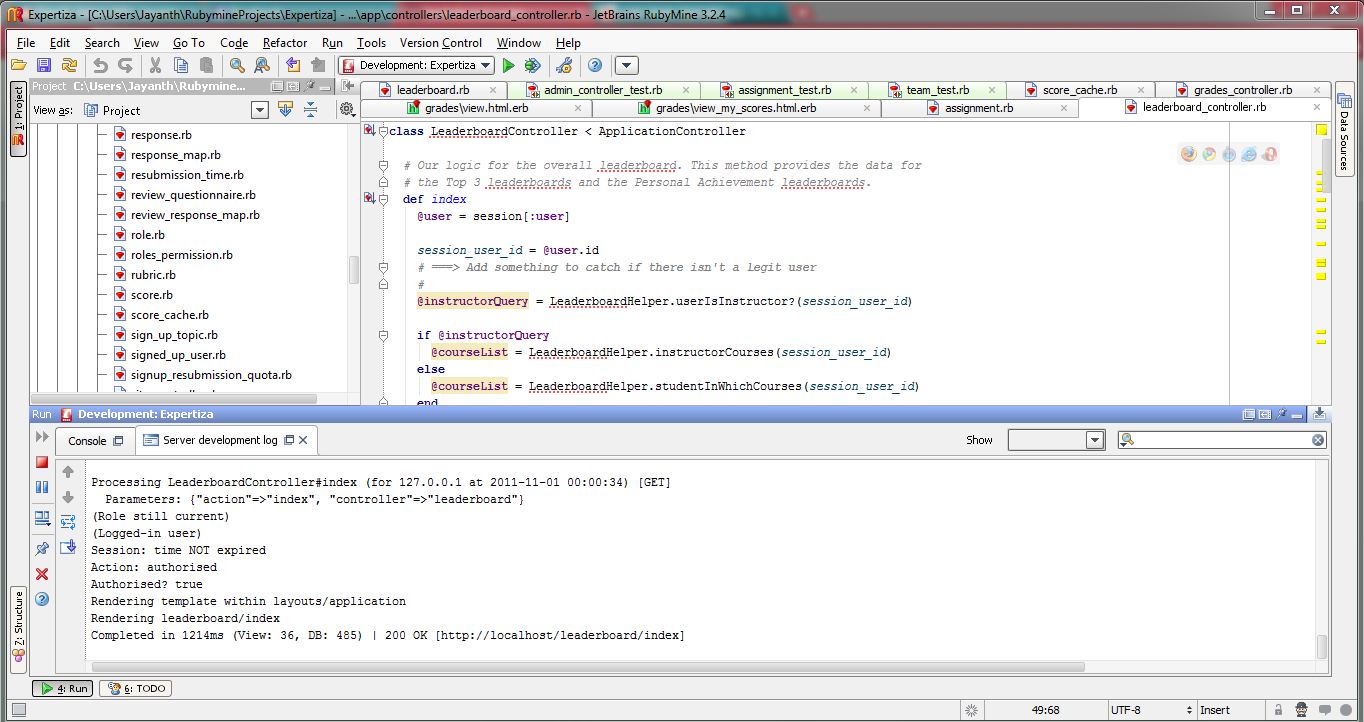
**Fig 1:** Showing the Leaderboard.

Project:

The project asked to do the timing studies of the leaderboard.rb and score\_cache.rb files. To reduce the number of accesses to the database, the results are cached into a separate table called score\_cache.rb file after each reviewer grades an assignment (could be individual or team).

We did the timing studies with score\_cache and without score\_cache to check the effectiveness of it being there.

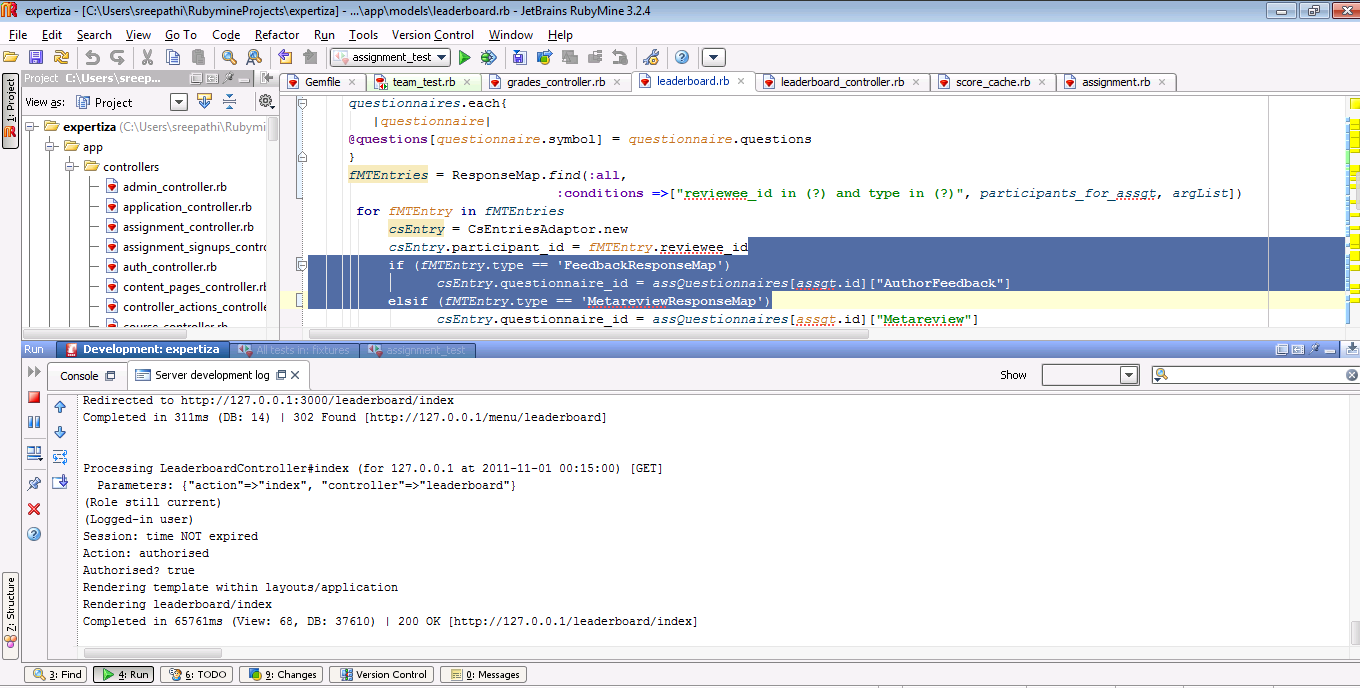
With score\_cache for 300 odd students, the leaderboard display took not more than 3 seconds. We ported some of the code from score\_cache to leaderboard.rb(the code can be found in GITHUB) to calculate the score on the fly for each student participated in the assignment. With this the performance reduced considerably and it took 40-50 sec to display for 300 odd students.



**Fig 2:** Showing the running time with score\_cache on.

With the above observation we concluded that the performance without score\_cache drastically reduces when the number of students increases. The reason is due to the fact that, for each student who took part in the assignment, the score has to be calculated on the fly whereas with score\_cache, the result is already calculated and we just need to take the precomputed data for processing.

The Figures 2 and 3 shows the running time taken during processing.



**Fig 3:** Showing the running time with score\_cache off.

The other part of the project asked to write testcases for leaderboard functions. The testcases can be found in the Github.

Also the project asked to move some the static calls to the leaderboard\_helper.rb file. But we found that moving these calls to the leaderboard\_helper.rb does not make much sense. Either we have to move all the calls or not. We felt that the functions should stay in leaderboard.rb. Though we have refactored some the names and rewrote the comments.