Team G Iteration 1

High-level design

Good. 80 / 100

Components, package, classes

The overview in layer_description.txt is useful. However it doesn't go into each layer. For example on the client side, what are the components (html files, java script classes etc. On the server side, it seems that much is already handled by the JAX-RS framework, although there is still some application logic to be dealt with, especially around ensuring that users have permission.

Specification of internal interfaces (APIs, WebAPIs, sequence diagrams)

The descriptions of the various data representation classes (JavaClasses.txt and javascriptObjects.txt) is clear. But I would expect some other classes

- Are there server side classes interacting with the database?
- Are there client side classes (or modules) for housing the controllers?

The web interface is reasonably clear. But I was left wondering about error situations. E.g. what if a get is sent when the resource does not exist, or if the user attempts to do something for which they don't have permission.

I assume that the user is identified to the server via a cookie, but this could be made clearer.

There doesn't seem to be any API to deal with compilation and execution of projects files.

Database Schemata

Good.

Other comments

Since I'm not very familiar with JAX-RS, it was hard to see what's missing from the documentation; i.e. what is already done by the framework and what is not.

Source code (optional)

Notes from Chen

Database-schema:

What is the relationship between each tables? to-one or to-many?

In account table and role table, there is accountid and userid, what is the different between them? why account table doesn't have "account_id"?

The file table needs "fileid". the assignment table needs "assignmentid" the course table needs "courseid" since the the assignment contains submissions, the submission contains submissionfiles, the submission files don;t need to connect to the assignment directly.

Class Diagrams:

It's better to provide the structure of main classes, for example: define a course class, what parameters are required in course class, what methods are required in course class? the method like: how to add instructors or students and how to add assignments.

Web-API,

It is better to describe HTTP Verb, Parameters and behaviors separately to minimize the mis-understanding for client-side users. for example: in your API:

POST /grade/{courseid}/{studentuser_name}/{assignment} :posts a grade and any comments relevent to a submission using

I think the "grade" should be a parameter, since this API is used to post a grade to server.

The UI design is good.

Databased-schema needs to be improved. Miss the class diagrams. Web API are not clear.