Maximum total 100 points. Part 0 is 0 points, parts 1 and 2 are 30 points each, and part 3 is 40 points. Ignore what they put in part 0, even if it is empty or missing (some students may submit this separately and I will also be contacting students to get it from them if its missing, ok if this part is a bit late), and grade parts 1, 2 and 3.

This assumes that each of the IAs grades their own teams. It is intentionally very open-ended grading. If the students don't leave out anything obvious and everything they say sounds reasonable based on what you know about the project, it's fine to give 100. The general rules for grading this assessment are 1. Don't be too picky, 2. Be consistent across students, and 3. Be quick. If possible, the IAs should compare notes to make sure they are being reasonably consistent with each other.

Part 1 (30 points max)

A (21 points max)

There should be at least three user stories (refer to second iteration report, not first iteration or proposal), so 7 points max for the CRC cards corresponding to each story. For teams with more than three stories, consider the "best" three in terms of how much is covered by CRC cards.

For a given user story (7 points max): 0-2 points based on whether there are classes corresponding to most of entities mentioned in the user story, including its conditions of satisfaction (most user stories will involve multiple classes, not just one). 0-3 points based on whether there are responsibilities, associated with one of those classes, corresponding to most of the functionalities mentioned in the user story. 0-2 points based on whether most of the other classes and external entities (e.g., APIs, databases) that each class communicates with, knows about, or keeps a reference to are mentioned as collaborators. Note the same class may be associated with multiple user stories. Also note a class can collaborate with itself, e.g., a member of the TreeNode class might collaborate with other instances of the TreeNode class - its children and possibly its parent. The names of the classes and responsibilities should be self-explanatory given the wording of the user story – or explained in some prose included in the student's answer. If you cannot figure it out, or classes and responsibilities have meaningless names like C1 and R1 without a prose explanation of what these mean wrt the user stories, do not give credit for them.

B (9 points max)

0-9 points based on if it sounds reasonable based on your knowledge of the codebase. If the classes and responsibilities do not obviously match the codebase, then there should be a plausible-sounding discussion of the discrepancies. You do not need to check details.

C. Ignore whether C is present, grade A and B even if it is missing, we will account for later.

Part 2 (30 points max)

A (30 points max)

0-10 points based on the “size” of the part of the codebase that was reviewed, full 10 points for one reasonable-sized class (or equivalent), less if a subset of such a class or a small class when larger classes were available (that were not authored by this student – contact me if some student reviews what you think is their own code or if one student authored essentially the entire codebase, in which case he/she would have no choice but to review their own code).

Look at the code yourself (see B). Do you see any specific problems with identifiers, comments, organization, readability? 0-20 points based on what the student says about those problems you see, full 20 for catching most of them or for pointing out serious problems that you did not notice but agree with once they have been pointed out. Only credit discussions of specific problems with specific pieces of code, not lists of potential problems that might occur in some code somewhere. If you really cannot find any problems, instead 0-20 points if the student finds something plausible to criticize – for example, the student may talk about some code that ought to be there but is missing, which you are probably not going to be able to spot. I’m expecting the report to be a couple paragraphs length. If the code is truly perfect and the student just says that (even though the question said to think of something more to say), ask me to look at it. Maybe they are right!

B. If the student does not state which specific code was reviewed, I don’t see any way to grade A except give 0. Contact me if you have some other idea.

C. Ignore whether C is present, grade A even if C is missing (but B is present), we will account for later.

Part 3 (40 points)

A (20 points max) 0-10 points based on whether it sounds like they really know their codebase (ignoring any login and database components) and the other 0-10 points based on whether they describe what sounds like a plausible skeleton for someone else to start with. You do not need to check details. Note students should “know” code that they did not write fairly well, particularly since they were given the hint that they should look at parts of the code written by other team members (see question 2).

B (20 points max) 0-10 point if it sounds like they really understand the functionality of their project (again ignoring login and database) and the other 0-10 points on whether they describe an assignment that corresponds to a subset of that functionality and sounds plausible. Don’t worry about the realism of whether an arbitrary student next year could really finish it in two weeks.

C. Ignore whether C is present, grade A and B even if it is missing, we will account for later.