Technical University of Denmark

Software Engineering

Group 35

Feedback on Group 34's Report 1

Author:	*Student number:*
Lukas Villumsen	s144451
Sebastian Nyholm	s144434

Danmarks Tekniske Universitet

March 19, 2015

Contents

1 Glossary **2**

2 Detailed* Use Case **2**

2.1 #1 - Login 2

2.2 Overall 2

3 Use Case Diagrams **2**

4 User Stories **2**

5 Diagrams **3**

5.1 Class diagram 3

5.2 Sequence diagram 3

5.3 Conflicts and overall assessment 3

6 Discussion **4**

1 Glossary

Nice overview of the terms used. Good restricted amount of terms, not including any unnecessary trivial glosses.

2 Detailed* Use Case

You have made detailed use cases in your report (as you should of course), but named them "Use cases", without the *"detailed"*.

2.1 #1 - Login

What is the user made for? As we can see in the detailed use case, the user has to write his/her employee initials to login. Doesn't that make this user an employee? It causes confusion.

"A user writes his employee initials" - So a user is already an employee. The user has the ability to 'create employee' in which case he does not have to be logged in. This allows outsiders using the computer to create employees at will. We doubt this is what you are trying to achieve. Note that this is not meant as criticism, we just want you to think about how secure your system is without compromising its easy accessibility.
Although this is not your immediate concern, it might help you if you plan on upgrading/expanding the system.

2.2 Overall

The detailed use cases explains very well, how the system is meant to work.

3 Use Case Diagrams

Easy to understand and gives a nice overview of the different user types, however a more clear link between the box "Manage Time" and sub-utilities, explained in the Manage Time use case diagram, would be nice. Same goes for the "Manage Project".

4 User Stories

The user stories are short and concerns mostly only one task, which is good and makes it easy to follow.

5 Diagrams

5.1 Class diagram

Overall it looks fine. You could try to use some arrows instead of implicitly creating objects of the other class. For example in your Employee class, you have a set of Projects. Why not use an arrow to tell it? Might give a better overview. It would be nice, if the class diagram showed some behaviors in form of methods. Some of them does, but not all and it seems like some of those classes, for example Employee, can do more than just be an object.

5.2 Sequence diagram

There is a recurring problem in the sequence diagrams. You consistently draw arrows with solid arrow heads, indicating that a return is to be made. This, however, does not always happen according to your diagrams. Throughout the diagrams this renders the latter executions non-reachable and will cause errors in the implementation phase.

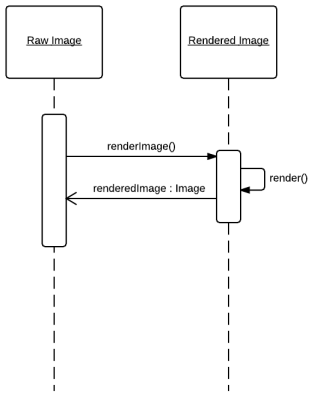

Figure 1: Correct Sequence Diagram

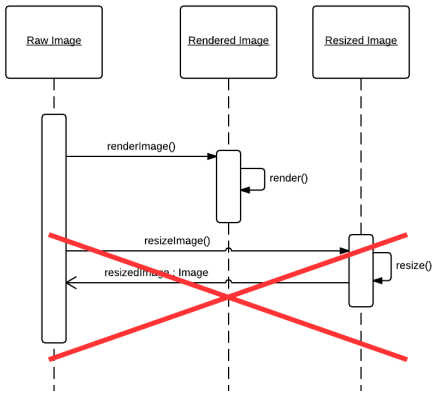

Figure 2: Unreachable Sequence

5.3 Conflicts and overall assessment

It might be difficult to connect the class diagram with the sequence diagrams, when many of the method used in the sequence diagrams are not present in the class diagram. Beware that to ease an implementation, it is a good idea to have the same names- and for the same methods- in each of the diagrams. This way you can shed a light on the method in different ways which makes for a more robust and interconnected implementation.
Other than that it's fairly easy to see your intentions with the system and its architecture. Since this is only a rough estimate of the finalized system, it'll be a fine draft when you start the programming.

6 Discussion

The problem with the assist part is that it might cause some unintended scheduling overlaps and miscommunication. This is due to no way of announcing the occupation of the employee if he/she is helping out another employee on that day.
As an example let's assume that employee A asks for help from employee B:
B lets A know that he is busy but offers his/her help the following day, as he/she will not be working on anything. A project leader is assigning employees to his/her project, and chooses to include B, as he/she is allegedly free to work. Thus B is now supposed to be working two activities at the same time. Consider making it possible for employee A to add B to his project so that he/she will not show up on the `checkAvailability()` list[1]

One more thing...
Does your company only hire male employees? ☺

[1] Assuming list. It is not stated in your class diagram which type this method return. It is however a method in your sequence diagram?