Design Document Revision - MMSG1 to Watchdog

Summary

Overall, this team did a solid job on their design document. However, there were some parts that were not too clear and may need a little more of an explanation.

Checklist

Content

Are all the interfaces to the system specified in detail?

*Interface to the system is detailed but should be realized using more class diagrams. More specifically widgets and components included in the SDK and how they will be implemented a step above actually programming the GUI.*

Are acceptable solution alternatives and their trade-offs specified?

*Within the threshold algorithms for image processing, yes. Although no other major components algorithms pro vs. con specified.*

Are you satisfied with all parts of the document?

*Take a look at your use-cases from the requirements document, each component or object in your design document should be implemented in more specific detail within UMLs in this design document. Reading components used for the SDK will aid in this and help clarify the details within these diagrams. Also major components are described at a very high-level in this document. Again, break the components down in a top-down fashion and clarify each step a step above pseudocode with class-diagrams and other UMLs.*

Do you believe all parts are possible to implement?

*Within the time-allotted I believe having an in-detailed description of the system is good. This calls for a deeper understanding come time to code. Although some parts of the project (minor details within the system) are a ambiguous to finish in 8 weeks.*

Are all parts of the document in agreement with the product requirements?
No, not all of the use-case components are brought into depth here. This document is supposed to be an in detailed description of the requirements. The details should be at a lower level here.

Is there risk associated with the proposed design? Is it discussed in the document?

No, should include these “risks”.

Will the goal for each type of testing be met with the testing that is described?

Yes, but should focus more-so in on SDK testing. Read the SDK manuals for graphical interface testing and other JUnit testing simulators.

Are the testing activities scheduled at the appropriate times?

Yes, although this will be implemented towards the end of the semester if given time.

Is each part of the document in agreement with all other parts?

For this document yes, but try to incorporate more of your design within the requirements document. This document should be a more detailed version of that document.

Completeness

Have alternative designs been considered and documented?

None, some should be though.

Are the limitations of the specified implementation sufficiently documented?

No, include some more detailed pros and cons to the approach in the document.

Are dependencies and assumptions thoroughly documented?

No, what is needed should already be realized by the group.

Where information isn't available before review, are the areas of incompleteness specified?

Class diagrams and UMLs in relation to use-cases in the requirements document.

Are all the testing stages covered?
No, the testing plan is a bit ambiguous. Should bring it down to lower level SDK tests and JUnit tests.

Is the regression test strategy covered?

Should go more into detail here.

Are there high-level validation tests necessary and covered (performance, security, etc.)?

Should go more into detail here as well.

Clarity

Is the solution at a fairly consistent and appropriate level of detail?

No, more detail should be implemented in this document.

Is the solution clear enough to be turned over to an independent group for implementation and still be understood?

No, still at a very high level.

Is the control flow and the data flow clear?

Still at very high level. Should be implemented by class diagrams and UMLs.

Is each requirement (or feature list item) measurable (will it be possible for independent testing to determine whether each requirement has been satisfied)?

Should break down testing major component by component.

Is the test strategy for each type of testing clearly described?

Again, breaking down each component will make the testing plan easier.

Are the test report requirements clearly described?

No.

Are all items clear and not ambiguous? (Minor document readability issues should be handled off-line, not in the review.)
**Comments**

An issue that came up with two members from our team was that they had difficulty going over how the layers interacted with each other. This part needs a little bit of more clarification and perhaps a bit of expansion on that description.

So far, it seems pretty general of a flow chart on how the different layers talk to each other.

A suggestion for describing what specific software is being used came up. For example, What server client are you guys installing it on? What specific tools are you guys using for your environment?

In your SQL section you guys talk about the tables being used to store the image data. I don’t think you guys actually included a column to store the actual image file like you stated in the description.

On the layer diagram, perhaps you might want to have a bit more of an interaction between the Service Layer and the Data abstraction layer, since they both seemed to be managed by the server.

Good job on the pseudo code on the design details section.

We recommend doing more class diagrams of main components to show oh what a certain class access to provide a specific functionality.

On your testing idea, I think you guys should also use other techniques for testing, and not solely rely on the automated testing program hence automation can have its faults if it doesn’t catch a specific bug.

Your distribution of the time on your weekly milestones seems reasonable.