## 3.

### a.risks

As the interface is not connected to the rest of the system there’s a risk that the clients won’t be able to interact with it.

It is very unclear how the system will be deployed.

Parts of the architecture is unclear, incoherent and sometimes contradicting.

### b.non-Risks

It is hard to point out safe non-risks with the architecture provided as it leaves a lot unclear or open to interpretation. Security should be helped by the database access component taking care of all the queries.

### c.sensitivity points

Connection to database as everything is connected through that component.

### d.tradeoff points

## 4.

### a. Information hiding

With the ui not really connected to the rest of the parts information is very much hidden.

### b. Minimize coupling

Coupling is pretty low based on the component diagram. UI is disconnected from everything. Unless reading the explanation in 10 where one might think each component on each layer can access averything on previous layer (not sure which way is “previous”) which would imply a lot of coupling.

### c. Coherence

The various use cases from 3 are not covered when assigning area of functionality. These in turn doesn’t match the components in 7. 6 doesn’t mention what the different components are supposed to be responsible for and number 5 does nothing along the lines of identifying subsystems or components. Use case diagrams all seem to follow different ideas of how the system works. Sequence diagrams are not coherent with eachother and some parts mentioned there are not seen anywhere else.

### d. Divide and conquer

What is there seems to be split up well. But some parts seem to be missing, shouldn’t the employees be more than a “Warehouse DB Access” for instance?

### e. Separation of concerns

Parts seem separated from things they don’t have much to do with, but with a lot missing it’s hard to tell for sure.

### f. Keep it simple

Considering how incoherent most of the architecture is and that you have to look through everything else too to get an idea on what things are doing I wouldn’t call it simple. If the incoherency is reduced and the things asked for added it would likely become simple to read as well. The image in 7 is really hard to read because of compression.

### g. No circular dependencies

Depends on the components “user account”, “shopping cart” and “DB access”. It is not mentioned how they are connected, just that they are.

### h. Layering

Architecture is layered, but I wouldn’t agree with the way “Business logic layer” is set up with advertisement all alone when it is likely very similar to what most other users will be using the system for (Query the database and do something with the information). Payment on another hand that will use external services would be a candidate for its own layer.

### i. Modularity

Looks decently modular, but it is hard to tell if that’s because a lot of things are missing or not.

**5.**

2/10, A lot of the architecture from the questions seems to simply be missing, misplaced or replaced with something else. The Architecture doesn’t really look like a model view controller structure. While the MVC architecture describes what is happening a client server structure might better describe the nature of a webshop and users connecting to it.