Guidance:

1. Purpose - this is just what the document is for, you don't need to modify this section in any way.

2. Goals and Philosophy - this is just what are the primary NFRs that the architecture addresses. For example, if reliability is a key concern you would say that the architecture addresses reliability issues. Its not really a 'philosophy', it just what you are trying to achieve with the architecture.

3. Assumptions and Dependencies - this is where you state some of your key assumptions - things like that the warehouse has a connection to the Internet, that the warehouse has a wireless network to connect the cranes - that the cranes have an independent processing node on them and aren't just a remote screen (for example). That the WMS is a separate system. Dependencies are things like the SOAP specification, the WAF specs, access to a TCP/IP network - external factors that you can't control and need to support/implement, or depend on.

4. Architecturally significant requirements - requirements that affect the architecture - like supporting access as a web service, the fact that the crane drivers consoles are mobile, if you have any reliability recoverability or performance requirements.

Note : sections 2 through 4 are 'forward looking' - ie they spell out requirements that your architecture has to address. The following sections are 'backward looking' - ie they document things that you decide and do in the other parts (B,C, D) of the assignment. You should NOT get stuck trying to figure these later sections out before you have actually decided on what your architecture is going to be, and what software patterns you are going to use. Come back and fill these sections in **after** you have worked out your architecture and detailed design.

5. Decisions, constraints, and justifications. Here is where you note down and explain/justify decisions you have **made** (note past tense).

6. Architectural mechanisms. Mechanism is sort of a synonym for 'pattern'. I relate it back to architectural 'tactics'. Tactics are a specific approach you are going to take to address a particular NFR. For example, we may have a reliability NFR, and we decide we are going to address that NFR using a 'hot spare'. 'Hot spare' is an 'architectural mechanism' we are going to use to address the reliability NFR. Refer to the lecture on 'Principles, Patterns, Tactics, and Views' for some idea of what tactics are. Otherwise, get hold of 'Software Architecture in Practice' (SWAiP) - or google 'software architecture tactics' and look at the articles published by the Carnegie SEI. I have also uploaded Chapter 5 from SWAiP into the Asg2 Resources section of the website.

My take is - an 'architectural mechanism' is some software pattern by which we are going to implement some architectural tactic. IE a mechanism implements a tactic.

Just note down what tactic you are going to employ to address some NFR, and then what pattern/arrangement of components you are going to use to implement that tactic. Once again this is best done **after** you have decided how you are going to address those issues.

7. Key Abstractions - for abstraction, read 'representation' or 'concept'. This section wants to know about the key concepts that are represented in the application - ie your domain model or entity classes. Also, if there is some really dominant pattern that helps explain how your whole system is going to function - like 'event driven interaction between components', or 'data access object encapsulate persistence system', then mention that here.

8. Architectural Framework - this where you typify your application by calling it 'basically a client-server model' (or not) - or 'the architecture uses a layered style, separating user interfaces from business logic and persistence mechanisms. Also that it is partitioned into your functional components. Basically, here is where you say how you divided the system up into functional and technical components, and whether there is some overarching 'architectural style' that you can relate it to.

9. Architectural Views. OK there is a mismatch here between the template and what I really want - but you don't have to include anything here anyway - you only have to refer to the diagrams that are requested in other parts of the assignment.

What we are producing as 'architectural views' corresponds to Kruchten's 4+1 Views as described in the Kruchten article in the Asg2 resources, and also that FCGSS\_US\_WP pdf - which tells you what UML 2 diagrams correspond with what views. Essentially its like this:

Logical view - Class diagram - Asg2 Part B Section a. The logical view basically shows the class collaboration necessary to support the CCRD use case according to the architecture that you have designed. Its a bit like the summary analysis class diagram, except that it shows **all** the classes involved in supporting the use case, including the user interface, controller, and other 'infrastructure classes. The logical diagram shows everything you would need to **implement** that single CCRD use case according to your architecture - so it shows what would be necessary to support the NFRs, as well as just the functional requirements - so if there is some need to store things in a database, the logical diagram would show the classes necessary to support that. This is where any DAObjects start to show up.

Process View - Sequence diagram - Asg2 Part B Section b. This is very similar to the sequence diagram we drew for Asg1, but it shows the details of how the control class is going to support the user interface you come up with for the crane driver in Part C, and as in the logical view, it shows how the NFRs are addressed as well as the functional requirements. So - as above , if you are going to store things in a database, we want to see when those DAObjects are told to update and save as part of the message sequence that supports the use case. Basically, the message sequence we drew in Asg1 (that is provided to you for 'manual store pallet' in the Asg1 Solutions) forms a skeleton, which we now flesh out with specific information need to support the crane driver interface, and also to conform with and support the NFRs addressed by the architecture. The message sequence has to be consistent with the architecture.

Implementation View - Component Diagram Asg2 Part A Section b. This is a 'high level' view that shows your functional and technical components. The other thing it has to do is identify any 3rd party components your application is going to rely on - like a database implementation, or a web server for example. The component diagram is for 'implementation' so it assumes you have actually identified the platform/language you are going to implement in. In terms of development this is an important diagram because it does things like identify what versions of what libraries you are going to need - and that the developers have to stick to.

Deployment View - Deployment Diagram Asg2 Part A Section c. This is a complementary high level view that shows how your high level components are laid out on the actual hardware of the deployment environment - its a pretty straight derivative of the component diagram with the addition of 'hardware nodes'. The trick is that your components have to respect hardware boundaries, and any 'inter-node' communication has to incorporate some sort of network communication - so there's another technical component you are going to need - probably on every node in the system.

Use case View - not required for this assignment, but basically its job is to provide a context for the other views.

Anyway the point here is that you don't have to duplicate all these diagrams in the architecture notebook - you just have to identify them, and reference the other parts of the assignment