Inspection Checklist - Software Architecture Document
From Software Architecture Document Guidelines, by Simon Brown, edited based on project scope.

Inspector: Nicholas May (Senior Software Engineer)	
	Comments/Notes
Functional View	
Is it clear which features/functions/use cases are significant to the architecture?	Explicitly states the significant use cases, as well as breaks them into groups to help clarify why they are significant. Mentions the referenced 'Vision Document' which helps to understand why the use cases were chosen, but one does not have to read that document in order to understand the 'Software Architecture' document.
It is clear that these have these been used to shape and define the architecture?	Yes. While it's hard to see this simply from the Functional View section of the document, it's easy to see the use cases helped influence the overall architecture.
Non-functional View	
Is there a clear understanding of the non-functional requirements that the architecture must satisfy?	Overall, yes. One item that could be missing is a statement regarding the level of transaction rates that must be used while still meeting the non-functional reqs.
Are the non-functional requirements quantifiable and testable?	They are all testable.
Have common non-functional requirements been explicitly marked as out of scope if they are not needed (e.g. "user	Not done in this section, but other sections in the document help answer this question.
Are any of the non-functional requirements unrealistic? (e.g. true 24x7 availability is typically very costly to implement).	Only the fact that peak and sustained transactional rates were not supplied, which could imply this solution works for any level of peak/transactional rates.
Architectural Principles	
Are there any other principles (e.g. other non-functional requirements that have not been explicitly requested) that have	No.
Architectural Constraints	
Are the constraints well documented and comprehensive?	Yes; however mentioning the use of Open Source code might be worth while.
Is it clear how the constraints affect the architecture?	Yes, the document clearly states why the particular architecture and software stack was chosen.
Process View	
Is it clear what the system does from a process perspective?	Good high level overview of the architectually significant portion of the process, but it might be useful to tie the web services section with the Jgroups section.

Are the major flows of information through the system well understood and documented (e.g. using UML activity diagrams)?	No diagrams. A basic view of how data moves through the system and into/out of the cache might be useful…similar to the comment above. (this might be in the Vision document)
Logical View	
Is a logical view of the architecture clearly portrayed?	Yes
Does it show the major components and interfaces?	Yes
Are they described at a high level?	Not described well in this document, but the corresponding Vision document does an excellent job at describing them.
Does the logical view show external systems and any other dependencies at a high level (low level detail about the	NA
Interface View	
Are the key internal (e.g. databases, messaging systems, etc) and external interfaces (e.g. other systems) well specified at a	Exteral interfaces are documented, but not much on interal systems.
What format are the messages (e.g. plain text or XML defined	Not specified, but would be useful.
Who has ownership of the interfaces?	Unknown…not sure if it is applicable here.
Technology Selection	
Is it clear why the selected technologies were chosen?	Covers the significant drivers for choosing the technology, but not too clear on exactly why they were chosen.
If there were options, why were they not chosen?	Doesn't specify any options…it uses a common tech stack.
Do they all fit in with the constraints outlined previously?	Yes
Are all software and hardware tiers covered?	Covers both software and hardware.
Design View	
Is it well understood how the key use cases will be	Yes
How are the chosen technologies used and combined?	Different diagrams and explanations cover this in good detail.
Are there common patterns across the architecture?	Yes
If yes, are these well understood and documented?	Yes, they are well documented. The use of commong patterns among spring and hibernate are an example.
Are the diagrams (e.g. UML class and sequence) up to date	Yes
Are any common wheels being reinvented? If so, why aren't vendor/open source products being used?	No
Is there enough information here to provide the rest of the development team with an overview/the intent of how the	Yes, definitely gives a good overview for others to learn from.

Infrastructure View	
Is there a clear physical architecture?	Instead of just an 'ideal architecture diagram', it would probably help to show a 'actual architecture diagram'"
What hardware does this include across all tiers?	The diagram mentioned above could probably show this information.
Does it cater for redundancy, failover and disaster recovery if	Yes, multiple JVMs on multiple, separate machines (i.e. slices).
Security View	
Is there a clear understanding of how security is handled within the architecture and how any security requirements have been	Yes.
• Authentication.	Discusses current, as well as possible future state, for authentication.
• Authorisation.	Yes.
• Confidentiality of data between components (e.g. during user login, during requests between components, using technologies such as web services or messaging, across	Discusses the cases involving cached data, as that is the focus of the project.
• Different types of users and their roles.	Yes.
• Network separation using firewalls and DMZs (red, amber,	No…probably closer to NA.
• Restricted access to resources.	Sufficient
• Permissioning of data of a per user/role/etc basis and the ability to modify those permissions.	Nothing on modifying permisions, this isn't too important in project scope, but could be mentioned.
Monitoring, Management and Administration View	
Is it clear how the architecture provides the ability for operation/support teams to monitor and manage the system?	Yes.
How is this achieved across all tiers of the architecture (e.g.	Yes.
Data View	
Is there a high level understanding of how much storage will be	No
What are the archiving strategies?	NA
Are there any regulatory requirements for the long term	No
Likewise for log files and audit trails?	No
Justification of Non-Functional Requirements	
For each of the non-functional requirements, is it explicit how	Yes, unless the item has been descoped or not fully implemented.
In the case of performance and scalability targets, are the test cases and results referenced?	No.

Are there any single points of failure?	Database? Not much can be done about that.
What happens if a component fails?	Addresses this from a cache member perspective.
Can you recover from a system failure?	Addresses this from a cache member perspective.
Who is responsible for system recovery and failover?	Addresses this from a cache member perspective. The vision document goes into some of this in a bit more detail.