Requirements Peer Review
Checklist

SRS Document Prepared by:	**SRS Document Reviewed by:**
Hala Ali Khan, Mahrukh Ali Khan	*Manahil Ashfaq, Nimra Waheed*
Dated:	May 26, 2022

Requirements Peer Review Checklist

The Requirements Peer Review Checklist defines the criteria to be used during a peer review of a software requirements specification. For a detailed explanation of how peer reviews are conducted, and how they differ from formal reviews, inspections, and walkthroughs, please refer to "Inspections, Peer Reviews, and Walkthroughs," PAL #3.2.3.

This checklist may be used for all software or system requirements specifications, both new and revised.

For each checklist item below, place a check (✓) in the box if the checklist item is satisfied. Otherwise, list any problem areas or exceptions under "Issues and Comments."

		✓	Issues and Comments
1.	**Completeness of Specifications** — Does the requirements specification document address all known requirements? GUIDANCE: A requirements specification should address such elements as control flow, data transformations, design constraints, and user interface.	✓	
2.	**Clarity** — Are the requirements clear enough to be turned over to an independent group for implementation?	✓	The requirements are quite detailed and well explained
3.	**Consistency** — Are the specifications consistent in notation, terminology, and level of functionality?	✓	The requirements are all consistent and well specified
4.	**External Interfaces** — Have external interfaces been adequately defined? GUIDANCE: Interface requirements are frequently documented in a separate Interface Requirements		Hardware has not been defined at all. Even if it is a software still it would require some hardware to operate on or to communicate with its users.

	Document (IRD) or Interface Control Document (ICD		
5.	**Testability** — Are the requirements testable? Will the testers be able to determine whether each requirement has been satisfied?	✓	Different Modules make testing easier
	GUIDANCE: The requirements specification should stat how every requirement will be tested. This helps to reduce ambiguity, increase clarity, and show testability:		
6.	**Design-Neutrality** — Does the requirements specification state what actions are to be performed, rather than how these actions will be performed? GUIDANCE: In other words, the requirements should concentrate on what the software needs to do rather than how it will do it. GUIDANCE: In the case where asystem or subsystem iS being configured from a product line, design neutrality does not apply. Instead, one should show that requirements are consistent with the selected product line architecture.	✓	It describes both 'hows' and 'whys'
7.	**Readability** — Does the requirements specification use the language of the intended testers and users of the system, not software personnel?	✓	
8.	**Level of Detail** — Are the requirements at a fairly consistent level of detail? Should any particular requirement be specified in more detail? In less detail? GUIDANCE: At GSFC, there are at least three levels of requirements. Level 1 is for Mission-level or Project-levé/ requirements, Level 2 for requirements at the software system level, and Level 3 for subsystem-level requirements. Frequently there is also a Level 4, whic contains internal, or all-software, requirements. There is normally a separate requirements specification for each level of requirements. It is important that each requirement be stated at an appropriate level of detail, and that all the requirements in a given requirements specification be at the same level of detail;	✓	All modules are in favorable detail.

9.	**Definition of Inputs and Outputs** — Have the internal interfaces, i.e., the required inputs to and outputs from the software system, been fully defined? Have the required data transformations been adequately specified? GUIDANCE: Note that use of correct units IS a commonly occurring issue for data interfaces and transformations	✓	Yes, all the interfaces have been defined in detail.
10.	**Scope** — Does the requirements specification adequately define boundaries for the scope of the target software system? Are any essential requirements missing?	✓	Section 1.4
11.	**Design Constraints** — Are all stated design and performance constraints realistic and justifiable? GUIDANCE: An example of an unrealistic constraint might be 100% availability of the system, or 1 nanosecond response to the user. Actually, a 1 nanosecond respons time might seem unrealistic, but could also be necessary.	✓	Yes, the constraints are realistic and justifiable
12.	**Traceability** — Dose each requirement can be traceable to its backward or forward requirement artifacts?	✓	