**SRS Check List**

1. **Completeness:**  
   A correct, complete set of requirements is one that correctly and completely states the desires and needs of the sponsor. If the requirements are incorrect, the software may meet the requirements as stated, but will not do what the sponsor wants it to do. If the requirements are incomplete, the software may do only part of what the sponsor hoped it would do.

* Is each characteristic of the final product described?
* Are all the tasks the user wants to perform specified?
* Is each requirement relevant to the problem and its solution?
* Are all the inputs to the system specified including their source, accuracy, range of values, and frequency?
* Are all the outputs from the system specified including their destination, accuracy, range of values, frequency, and format?
* Does each function specify the data used in the function and data resulting from the function?
* Are all aspects of the processing specified for each function?
* Are the requirements complete in the sense that if a product satisfies every requirement, it will be acceptable?
* Do the requirements contain no implied design details?
* Do the requirements contain no implied implementation constraints?
* Do the requirements contain no implied project management constraints?

1. **Consistency:**  
   Consistency is obtained if the requirements do not contradict each other. Inconsistency results when one requirement contradicts another.

* Is each requirement unique? (I.e., no redundancy)
* Are the characteristics of real-world objects consistent?  E.g., If one requirement specifies a report in tabular format but the user interface prototype shows it as text format, there is a conflict.
* Are the logical and temporal characteristics of required actions consistent?  E.g., If one requirement says the database is updated when the file is closed then it would be inconsistent if another requirement said the data is saved when the transaction is completed.
* Is all terminology used consistently? E.g., don't use "prompt" in one requirement and "cue" in another.
* Do all the requirements avoid conflicts with other requirements?

1. **Traceability and Modifiability:**  
   Ultimately every aspect of the finished system should be able to be traced back to the requirements. Therefore, this document should be organized to facilitate tracing the requirements into subsequent work products.  An SRS is modifiable to the extent that requirements changes can be made easily and consistently while retaining its structure and style.

* Does the organization adhere to an accepted standard?
* Is the document organized in a segmented, top-down manner?
* Is every requirement numbered in a manner that facilitates cross referencing and indexing?
* Is each requirement expressed separately, rather than intermixed with other requirements?
* Can each item be traced to its origin in the problem environment?
* Are all possible changes to the requirements specified?

1. **Verifiability**

The requirements must be verifiable in two ways: do the requirements satisfy the sponsor's needs, and does the system satisfy the requirements? In the first case, the requirements must be compared to the sponsor's desires and needs. Do the requirements correctly and completely specify the sponsor's desires and needs? In the second case, once the system has been developed, it must be compared to the requirements. Does the system meet the requirements as they are stated?

* Are all of the requirements feasible? (I.e., possible to implement)
* For each requirement is there a process that can be executed by either a human or a machine to verify the requirement?
* Is each requirement testable or verfiable? Will it be possible for independent testing to determine whether each requirement has been satisfied?
* Does each requirement use concrete terms and measurable quantities?
* Especially for nonfunctional requirements, special attention must be given to stating the requirements in a manner that is objective and quantifiable; there must be some measurable way to assess whether the requirement has been met.
* Does the writing avoid vague terms such as those in Dan Stearns' bad words list?

1. **Clarity**

**It is essential that the requirements be clear to all readers so as to prevent ambiguity and misinterpretation.**

* + Are the requirements written in user language? Do the users think so?
  + Are the requirements written in non-technical language that uses the vocabulary of the client problem domain?
  + Are the requirements stated simply and completely so that they are unambiguous?(NO:1)
  + Are the requirements stated clearly so there is only one interpretation?(NO:2)
  + Do the requirements describe only EXTERNAL behavior, as seen from the user's point of view?
  + Do the requirements avoid stating how the problem is to be solved or what techniques are to be used.
  + Do all data and processes exist in the problem domain, not the solution domain?
  + Is *every* noun a term from the data dictionary?
  + Is *every* noun emphasized in some way in the document (e.g. underlining) and use the *exact* DD term?
  + Are the requirements at a fairly consistent level? Should any requirement be specified in more detail? Should any requirement be specified in less detail?
  + Are the requirements clear enough to be turned over to an independent group for implementation and still be understood?