Non-functional Requirements

In addition to the obvious features and functions that you will provide in your system, there are other requirements that don't actually DO anything, but are important characteristics nevertheless. These are called "non-functional requirements" or sometimes "Quality Attributes." For example, attributes such as performance, security, usability, compatibility. aren't a "feature" of the system, but are a required characteristic. You can't write a specific line of code to implement them, rather they are "emergent" properties that arise from the entire solution. The specification needs to describe any such attributes the customer requires. You must decide the kind of requirements that apply to your project and include those that are appropriate.

Each requirement is simply stated in english. Each requirement must be objective and quantifiable; there must be some measurable way to assess whether the requirement has been met.

Often deciding on quality attributes requires making tradeoffs, e.g., between performance and maintainability. In the APPENDIX you must include an engineering analysis of any significant decisions regarding tradeoffs between competing attributes.

Here are some examples of non-functional requirements:

Performance requirements
Requirements about resources required, response time, transaction rates, throughput, benchmark specifications or anything else having to do with performance.

Operating constraints
List any run-time constraints. This could include system resources, people, needed software, ...

Platform constraints
Discuss the target platform. Be as specific or general as the user requires. If the user doesn't care, there are still platform constraints.

Accuracy and Precision
Requirements about the accuracy and precision of the data. (Do you know the difference?) Beware of 100% requirements; they often cost too much.

Modifiability
Requirements about the effort required to make changes in the software. Often, the measurement is personnel effort (person- months).

Portability
The effort required to move the software to a different target platform. The measurement is most commonly person-months or % of modules that need changing.

Reliability
Requirements about how often the software fails. The measurement is often expressed in MTBF (mean time between failures). The definition of a failure must be clear. Also, don't confuse reliability with availability which is quite a different kind of requirement. Be sure to specify the consequences of software failure, how to protect from failure, a strategy for error detection, and a strategy for correction.

Security
One or more requirements about protection of your system and its data. The measurement can be expressed in a variety of ways (effort, skill level, time, ...) to break into the system. Do not discuss solutions (e.g. passwords) in a requirements document.

Usability
Requirements about how difficult it will be to learn and operate the system. The requirements are often

expressed in learning time or similar metrics.

Legal
There may be legal issues involving privacy of information, intellectual property rights, export of restricted technologies, etc.

Others ... there are many others. Consult your resources.

There are examples of good and bad quality attributes, written by the fall 98 CSC 205 classes for your viewing pleasure.
Here are some more examples.
