# Scalability and Performance

* + **Real time access -** combining system with Cloud computing
  + **Data Partitioning -** Spread data into multiple DB
  + **Cache Engine (Dynamic Cache) -** instead of redo the same execution for same input parameters, we can remember the previous execution's result
  + **Resources Pool -** DBSession and TCP connection are expensive to create, so reuse them across multiple requests
  + **Asynchronous Processing – continue with other processes, whilst waiting on the response of the other**

# Maintainability

* + Design for maintainability from the outset
  + Iterative development and regular reviews improve quality, e.g. Using the agile approach
  + Code readable that is easy to understand
  + Provide relevant documentation helps developers understand the software for further maintenance
  + Make use of automated builds make the code easy to compile
  + Make use of automated tests make it easy to validate changes

# Reliability

* + Use good architectural infrastructure
  + Build management information into the application
  + Use redundancy for reliability, that is have multiple copies or services of the same type running at the same time
  + Use quality development tools, this includes HTML and the DOM, JavaScript debugging, profiling and auditing and so on
  + Use consistent error handling, and provide meaningful error messages

# Availability

* + Elimination of single points of failure, by having multiple copies or services of the same type running at the same time, as specified before. So that when one service cannot be provided, then others or possibly the same services can be provided
  + Provide reliable crossover, enabling uptime to remain as high as possible even during maintenance
  + Detection of failures as they occur

# Security

* + Input Validation
  + Authentication
  + Authorization
  + Sensitive Data - Confidential information disclosure and data tampering.
  + Session Management
  + Cryptography
  + Parameter Manipulation
  + Exception Management
  + Auditing and Logging

# Usability

* + [Look-and-feel](http://www.motive.co.nz/glossary/looknfeel.php) - includes making [navigation](http://www.motive.co.nz/glossary/navigation.php) easy, useful [interface cues](http://www.motive.co.nz/glossary/interface.php), good color choice, for easy reading and scanning
  + [Navigation](http://www.motive.co.nz/glossary/navigation.php) - to tell the user where they are, and enable the user to go somewhere else
  + [Interface design](http://www.motive.co.nz/glossary/interface.php) -inform users of the task the interface can be used to complete and provide feedback to let users now what has been done
  + [Information architecture](http://www.motive.co.nz/glossary/information.php) – organize or structure contact pleasant to read manner, and make use of short phrases as much as possible

# Testability

* + Isolate the Ugly Stuff -"ugly stuff" is any kind of code or infrastructure that is complicated or laborious or just plain inconvenient to get into a test harness, or that makes tests run very slowly.
  + Using Fakes to Establish Boundary Conditions - For instance instead of doing the data directly delegate to some other services
  + Separate Deciding from Doing - an action and deciding to take an action treated as two separate responsibilities.
  + Small Tests before Big Tests - Small test often point you direct to point of failure or error, where else Big test have a lot of factors to consider, which makes debugging hard, so more small test and leading to easy debugging for Big test