**Non-Functional Requirements**

* Availability
  + System should ideally be available all of the time for targeted users.
  + Some level of monitoring should be considered for quick detection and fixes of any downtime occurring within the system.
* Maintainability
  + Bugs or any defects in system should be easy to detect and resolve.
  + Code should be easy to read and understand.
  + Code should have comprehensive comments and be written in simplified language
  + Different versions of the system should be easy to maintain.
  + System should be easy to extend or enhance in the future.
  + A test driven development approach should be used for superb low level design and unit tests.
* Interoperability
  + If required the system should be easy to interface with other 3rd party components and to exchange data or services with other systems
  + Different modules within the system should seamlessly work on different operating system platforms, different databases and protocol conditions.
* Performance
  + Web application’s UI should ideally take no longer than 3 seconds to load.
  + Search results should be returned within 2 seconds.
  + Database updates should be done within 2 seconds.
  + Login validation should be done within 3 seconds.
  + Code should be tuned to the maximum with the appropriate use of data structures and algorithms.
  + Databases used should be suitable and well optimized for the type of queries required by the system.
  + Caching should be used to improve performance where applicable.
* Portability
  + Code should be well defined, with portability in mind so that as technology changes in the future, changing platforms can be easily accomplished.
  + Specific dependencies on certain technologies and their features available only in specific forms should be avoided where necessary.
  + The system application should run on any device that has a modern browser with HTML5 and JavaScript support.
* Reliability
  + Data registered in database and/or served to user should be accurate and dependable.
  + Database should be well designed, thereby reducing the chances of having errors with data stored.
  + The system should allow for logged data to be periodically backed up and restored in case of data loss. Logged data backups should be done either hourly, daily, weekly or monthly as specified by the user.
* Reusability
  + Development should follow a modular approach whereby the build is broken down into finer components each having specific and focused responsibilities. Therefore these can then be easily reused where necessary within the system and so increase work efficiency.
* Security
  + System security should be sufficient to prevent unauthorized access to system functions, avoid information loss, ensure that the software is protected from viruses, and protect the privacy of data transactions within the system.
  + User login information should be encrypted.
  + Vulnerabilities such as to SQL injection and Cross-site Scripting should be eliminated.
  + Accounts should have limited permissions and privileges accordingly delegated based on their necessary requirements.
* Scalability
  + System should be flexible enough to easily adapt and meet storage and performance requirements with minimal need of extra infrastructure and continue to be functional as users of the system and load on system resources increase.
* Testability
  + Architecture should incorporate a level of modularity so that instead of being restricted to testing the entire system tests can be conducted on individual singular components.
  + Unit tests should be well designed and used as much as possible for the various components within the system.
  + If time permits, tests should be automated where applicable.
* Usability
  + Application should be user friendly, easy to learn and navigation should be simple.
  + New or infrequent users should encounter minimal difficulty when using the system. Users should be able to perform tasks quickly with little to no errors. If errors do occur the user should be able to easily recover.
  + Users should be able to recover from errors in 10-20 seconds.
  + The system should supply consistent user interface design standards to invoke a sense of familiarity, eliminate confusion and promote learnability for the user.
  + User should take no longer than 2 minutes to learn to use the application interface.