**CHAPTER 3**

**SYSTEM REQUIREMENT SPECIFICATION**

**3.1 System Requirements**

While working on a project as our problem statement is all in all what we need to do is we need to automize the whole processes. There are so many companies which are still using their technologies manually . people may get so many new technologies but it is very much necessary to save the time and cost. And most of all backup. Now imagine there isn a worldwide company which is working on various projects and in a team one person is sitting in country A and other fellow is sitting in country B. now sending the code to the other person through mail or other medium is just not feasible all the time. Now we want some solution to this problem. Here we get is the concept of continuous integration and continuous deployment. So here in our project we will be using GIT. Github is a remote repository. Example if admin has assigned some work to the developer , then in this case what we require is that all the codes done by all the developers should be present somewhere that it should be accessible to all the team members irrespective of the work locations. Now imagine there is a worldwide company which is working on various projects and in a team one person is sitting in country A and other fellow is sitting in country B. now sending the code to the other person through mail or other medium is just not feasible all the time. Now we want some solution to this problem.

**3.2 Functional Requirements**

Functional requirements specify the key functions to be performed by the proposed system. These describe the functionality or services that the system is expected to provide.

**Functional requirements** are baked into the code that developers deliver (interpreted or compiled).   Events from input devices (network, keyboard, devices) trigger functions to convert input into output -- all functions have the form: 

[http://accelerateddevelopment.ca/blog/wp-content/uploads/2013/12/Function.png](http://accelerateddevelopment.ca/blog/wp-content/uploads/2013/12/Function.png)

**3.3 Non-functional requirements**

  Non-functional requirements involve everything that surrounds a functional code unit.  Non-functional requirements concern things that involve time, memory, access, and location:

* Availability
* Capacity
* Continuity
* Security

Non-functional requirements are slightly different between desktop applications and services;

**3.3.1 Availability**

Availability is about making sure that a service is available when it is supposed to be available. Availability is about a **Configuration Item** (CI) in the environment of the operations centre that specifies how the code is accessed.  Availability is decided independently of the code and is at best part of the **Service Design Package** (SDP) that is delivered to the operations department, at worst it is simply code dumped on the operations personnel.   
  
 Developer's need to be aware of the difficulty of creating the **CI** for the operations personnel.  If a **CI** is manually created then there will always be a potential for an error when the service is installed or updated. The requirement to create a **CI** is a non-functional requirement and the ability to minimize errors is another non-functional requirement.   
  
 Developer's need to be aware of single-points of failure (i.e. services hard-coded to a specific IP) which causes fits in operations that are not running virtual machines (VM) that can have virtual IPs . The requirement to create code that is not reliant on static IPs or specific machines is a non-functional requirement.  Availability is simplified in operations if the code is resilient enough to allow itself to easily move (or be replicated) among servers.

Availability non-functional requirements include:

* Ability to easily make the **CI**
* Automatic installation of **CI** or mechanisms
* Ability to detect and prevent manual errors for a **CI**
* Ability to easily move code between servers

**3.3.2 Capacity**

Capacity is about delivering enough functionality when required.  If you ask a web service to supply 1,000 requests a second when that server is only capable of 100 requests a second then some requests will get dropped.  This may look like an availability issue, but

it is caused because you can't handle the capacity requested.

Internet services almost always can't provide enough capacity with a single machine and operations personnel need to be able to run multiple servers with the same software to meet capacity requirements.  The ability to run multiple servers without conflicts is a non-functional requirement. The ability to take a failing node and restart it on another machine

or **VM** is a non-functional requirement.   
Capacity non-functional requirements include:

* Ability to run multiple instances of code easily
* Ability to easily move a running code instance to another server

**3.3.3 Continuity**

Continuity involves being able to be robust against major interruptions to a service, these include power outages, floods or fires in an operational center, or any other disaster that

can disrupt the network or physical machines.

Where availability and capacity often involve redundancy inside a single operation center, continuity involves geographic and network redundancy.  Continuity at best involves having multiple servers that can work in geographically distributed operation centers.  At worst, you need to be able to have a master-slave fail over model with the ability to journal transactions and eventually bring the master back up.

**3.3.4 Security**

Security non-functional requirements concern who has access to functions and

preventing the integrity of data from being corrupted.

Where access is concerned, how difficult will it be for operations personnel or help desks to set up security for users?  Developer's build in different levels of access into their applications without considering how difficult it will be for a 3rd party (help desk or operations) to set-up end users.  The ease of setting up security is a non-functional requirement.   
 Data integrity is another non-functional requirement.  Developer's need to consider how their applications will behave if the program encounters corrupted data due to machine or network failures.

**3.4 Hardware requirements**

* Processor : Intel Core
* RAM : 4GB
* Hard disk space : 100GB
* Input : Keyboard and Mouse
* Output : LCD Monitor

**3.5 Software Requirements**

* Operating System : Windows 7 and Above
* Browser : Internet Explorer 11, Chrome
* Database : MySQL
* Build : Maven, MS Build
* Integration : Scripting (Puppet, Bash Shell)
* Version Control : GIT
* Project Management : JIRA
* Environment : Eclipse, Intellij