**Experiment No.: 9**

**Mapping of non-functional components with system requirements.**

**Learning Objective:** Student should be able to understand Mapping of non-functional components with system requirements.

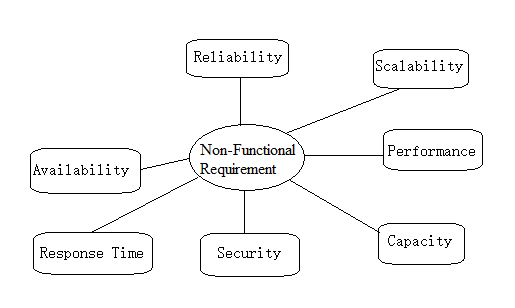
**Theory:   
What is Non-Functional Requirement?**

**NON-FUNCTIONAL REQUIREMENT** (NFR) specifies the quality attribute of a software system. They judge the software system based on Responsiveness, Usability, Security, Portability and other non-functional standards that are critical to the success of the software system.

NFRs define the system properties and specify the behavioral pattern under various operating conditions. The various estimation methods help in sizing the application based on the functional requirements. However most of these methods have overlooked the influence of non-functional requirements.

The key NFRs that can be attributed to an application and their mapping as follows.

1. Reliability Operation Ease
2. Response Time No mapping given
3. Performance Performance, Online Update, Online Date Entry
4. Security No mapping given
5. Availability No mapping given
6. Scalability Transaction rate
7. Capacity No mapping given



Middleware is a more effective program that acts as bridge in between various applications and other databases otherwise tools. It is placed in between [operating system](https://digitalthinkerhelp.com/what-is-operating-system-and-its-types-uses-tutorial/) and other applications which run on it. Middleware allows making better communication, application services, messaging, authentication, API management and management of data between different kinds of applications which help to exchange data.

The connectors sit between the two APIs or you can say and the ends of the connectors are APIs. The connectors receive data from one app/solution and process it to make it understandable and accessible in the other app/solution, regardless of whether any direct form of integration was available in the two apps.

**Mapping of NFRs :**

**Operation Ease to Reliability** :

An application or the software system once installed and configured on a given platform should require no manual intervention, except for starting and shutting down. The system should be able to maintain a specified level of performance in case of software faults. It should also be able to re-establish its level of performance and to recover all the data directly affected in case of a failure in the minimum time and effort. This is mapped on to the reliability NFR. It may be defined as “a system which is capable of reestablishing its level of performance and recovering the data directly affected in case of a failure and on the time and effort needed for it. The design criteria for reliability can be defined as self-contained the system should have all the features necessary for all its operations including recovering it by itself; completeness- it should be complete in itself and not dependent on anything else; robustness/integrity- it should not easily breakdown; error tolerance- it should be able to tolerate errors and rectify them and continue in its operation. There are “numerous metrics for determining reliability: mean time to failure, defect reports and counts, resource consumption, stability, uptime percentage and even customer perception.”

**Performance :**

Real time systems have strict performance parameters like performing at the same level even during peak user times, producing high throughput, serving a huge user base, etc. The DI varies from no special performance requirements to response time being critical during all business hours and till performance analysis tools being used in the design. System should meet the desired performance expectation. Also if online update has to take place then the performance expectations to be met are very high – fast response, low processing time and high throughput rates. The performance NFR is also based on the Online Data Entry requirements of an application. The present day trend is to have interactive and real-time data entry. The GUI development requires a lot of effort as help has to be provided, validation to be implemented, reference information for faster data entry operations, etc. Performance when related to this can be defined as “attributes of software that bear on response and processing times and on throughput rates in performing its function.

**Transaction Rate to Scalability :**

In many business applications the transaction rate increases to high peak levels once in a day or once in a week with the requirement remaining so that there has to be no dramatic increase in transaction time. This issue has to be looked into in the design, development and/or installation phases of a project. This GSC is mapped on to the scalability NFR. The term scalability implies “the ability to scale up to peak transaction loads. In order to achieve this the application has to be designed in such a way so that it should cater to the highest possible figures thus wasting resources when the transaction rate is low. The architecture should be designed in a multi-layered manner in complex algorithm based applications to scale up to peak transaction rates. In today‟s systems, this GSC does not contribute much to the DI as present day hardware and operating systems provide builtin features such as high bandwidth network, high speed storage disks with high-speed disk access timings and CPUs with high MHZ processing speed which when combined leads to built in high transaction rates.

**Result and Discussion:**

## Performance Requirements

Performance requirements are specifications that define the expected behavior of a software system in terms of its speed, response time, and resource utilization. In the case of a disease prediction system using Machine Learning, the following performance requirements can be included in the SRS:

1. Response Time: The system should provide a quick response time to user requests, such as predicting a disease or generating a report. The maximum response time should be defined, for example, 5 seconds.
2. Accuracy: The system should provide accurate predictions and results with a high degree of confidence. The accuracy of the system should be specified, for example, with a minimum accuracy of 90%.
3. Scalability: The system should be able to handle many users and data inputs without compromising its performance. The system should be designed to scale horizontally or vertically, depending on the requirements.
4. Availability: The system should be always available for use, except for planned maintenance periods. The system's uptime should be defined, for example, 99.9%.

## Safety Requirements

* Users must enter their data accurately or the system give an incorrect prediction.
* Users should not use the application while operating machinery, while driving, or in any other circumstance that requires them to pay close attention.
* The developers of this software application product disclaim all responsibility for any advice, course of treatment, diagnosis, or other information offered through the system, including any consequences that may follow.
* This system is not intended for emergency medical use. In case of an emergency, users should contact 108 or their primary doctor.

## Security Requirements

* The system is safe enough to prevent unauthorized or inappropriate disclosure of personal health information.
* System Application will have a secure "passcode" which the user has to enter each time they log in to prevent unauthorized access.
* Developers will maintain the confidentiality of all user data knowledge.
* The user's personal device won't be harmed by this application or any other installed software, and neither will the device's internal parts be harmed.

## Business Rules

This application will be mainly used by two sets of people:

* Patients: People who have symptoms of some illness and want to find out what it is can use this application. They can input their symptoms and the system would predict what disease they have. Also, they can search for doctors and consult them online using this application.
* Doctors: Doctors can use this application to interact with patients.

Because software connectors provide a uniform interface to other connectors and components within an architecture, architects need not be concerned with the properties of different middleware technologies as long as the technology can be encapsulated within a software connector.

**Learning Outcomes:** Students should have be able to

LO1: Define non functional Requirement.

LO2: Identify different component of non functional Requirement.

LO3: Explain mapping.

**Course Outcomes:** Upon completion of the course students will be able to understand mapping of non-functional components with system requirements.

**Conclusion:** The disease prediction system using Machine Learning requires performance, safety, security, and business rules. These include response time, accuracy, scalability, and availability. Safety measures emphasize responsible user behavior, data protection, and device safety. Adhering to these requirements ensures a trustworthy healthcare application that aligns with ethical standards.

For Faculty Use

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Correction Parameters** | **Formative Assessment [40%]** | **Timely completion of Practical [ 40%]** | **Attendance / Learning Attitude [20%]** |  |
|  |  |  |