**2. SOFTWARE REQUIREMENT SPECIFICATION**

**2.1 Introduction**

It specifies for the software requirement for the search the particular documents and various activities to be performed.

**2.1.1 Purpose and Scope of Document**

The purpose of the document to enlist the various software frameworks is required to build the system. The document covers the following points:

* Responsibility of the develop
* System Architecture
* Use cases scenario

**2.1.2 Overview of responsibilities of Developer**

* Firstly developer has to design a system according to requirement and specification
* The developer has to code for different algorithms
* To create different scenarios and to generate output
* To test the system under above stated scenarios and do necessary changes if required.

**2.2 Hardware Resources required**

1. System: Intel I5 2.4 GHz.

2. Hard Disk: 40 GB

3. Monitor: 15 VGA Color

**2.3 Software Interfaces:**

1. Operating system: Windows

2. Coding Language: Jdk 1.7

3. Database: MYSQL 5

5. IDE: Eclipse Luna.

**2.4 System Architecture Proposed Algorithm.**

**2.4.1 Design Goals**

****

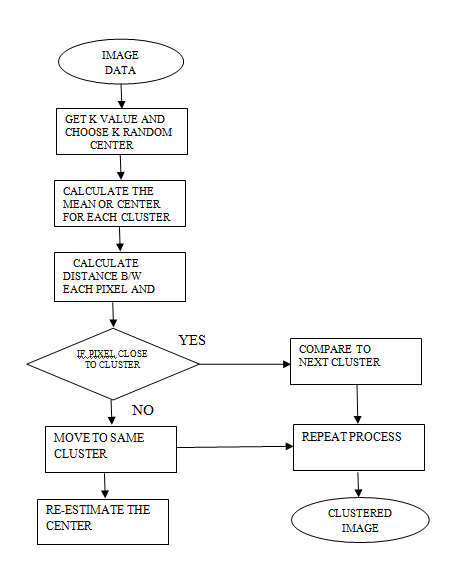
**Figure: Block diagram of System.**

**2.5 PROPOSED ALGORITHM.**

**A. K-means clustering**

K-Means is the one of the unsupervised learning algorithm for clusters. Clustering the image is grouping the pixels according to the some characteristics. In the k-means algorithm initially we have to define the number of clusters k. Then k-cluster center are chosen randomly. The distance between the each pixel to each cluster centers are calculated. The distance may be of simple Euclidean function. Single pixel is compared to all cluster centers using the distance formula. The pixel is moved to particular cluster which has shortest distance among all. Then the centroid is re-estimated. Again each pixel is compared to all centroids. The process continuous until the center converges.

**Flowchart of k-means algorithm**



**Fig: K-means flow chart**

**B. Fuzzy C-Means Algorithm**

The Fuzzy C-means is an unsupervised clustering algorithm whish can be applied to several problems involving feature analysis, clustering, medical diagnosis and image segmentation. Fuzzy C-means clustering (FCM) algorithm was proposed by Bezdek et.al in which each data point belongs to a cluster to a degree specified by a membership grade. The FCM algorithm minimizes the objective function for the partition of data set, x = [x1, x2, ..., xd]T

**Step 1:** Initialization

* Scan the image line by line to construct the vector X containing all the gray level of the image
* Randomly initialize the centers of the classes vector V (0).

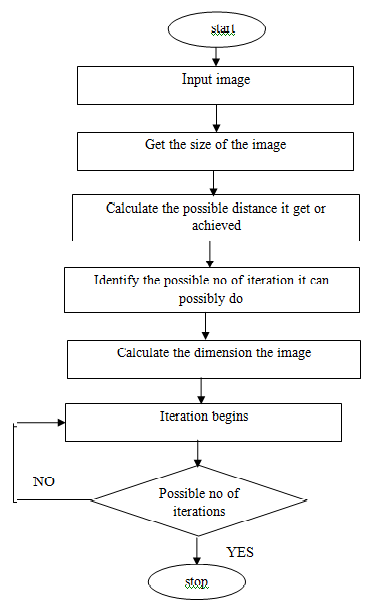
From the iteration t=1 to the end of the algorithm:

**Step 2:** Calculate the membership matrix U(t) of element uik .

**Step 3:** Calculate the vector V(t) = [v1, v2, ..., vc].

**Step 4:** Convergence test: if ||V (t) − V (t−1) || >ε, then increment the iteration t, and return the Step 2, otherwise, stop the algorithm. ε is a chosen positive threshold.

**Flowchart of C-means algorithm**



**Fig: FCM flow chart**

**2.5.1 RESULT ANALYSIS OF THE PROPOSED SYSTEM:**

**Set-up**

Here we used MRI scan brain images for detection if tumor for future treatment.

**2.7 Software Quality Attribute.**

**2.7.1 Reliability**

Software Reliability is the probability of failure-free software operation for a specified period of time in a specified environment. Software Reliability is also an important factor affecting system reliability. It differs from hardware reliability in that it reflects the design perfection, rather than manufacturing perfection. The high complexity of software is the major contributing factor of Software Reliability problems. Software Reliability is not a function of time – although researchers have come up with models relating the two. The modeling technique for Software

Reliability is reaching its prosperity, but before using the technique, we must carefully select the appropriate model that can best suit our case. Measurement in software is still in its infancy.

No good quantitative methods have been developed to represent Software Reliability without excessive limitations. Various approaches can be used to improve the reliability of software, however, it is hard to balance development time and budget with software reliability.

**2.7.2 Availability**

* Over-engineering, which is designing systems to specifications better than minimum requirements.
* Duplication, which is extensive use of redundant systems and components.
* Recoverability, which is the use of fault-tolerant engineering methods.
* Automatic updating, which is keeps OSs and applications current without user intervention.
* Data backup, which prevents catastrophic loss of critical information.
* Data archiving, which keeps extensive records of data in case of audits or other recovery needs?
* Power-on replacement, which is the ability to hot swap components or peripherals.
* The use of virtual machine s, which minimizes the impact of OS or software faults.
* Use of surge suppressor s, which minimizes risk of component damage resulting from
* Power-line anomalies.
* Continuous power, which is the use of an un interruptible power supply keeps systems operational while switching from commercial power to backup or auxiliary power.
* Backup power sources, which includes batteries and generators to keep systems operational during extended interruptions in commercial power.

**2.7.3 Security**

When the security functionality in a proposed product does not satisfy specific security requirements then the risk introduced must be evaluated and additional controls must be reconsidered prior to purchasing the product. Where additional functionality is supplied and causes a security risk, this must be disabled or the proposed control structure must be reviewed to determine if advantage can be taken of the available enhanced functionality.

Design reviews must be conducted at periodic intervals during the development process to assure that the proposed design will satisfy the functional and security requirements specified by the owner.

Applying the security requirements to the project and allocating financial, technical and human resources as required for meeting the security requirements of the project ensuring that the security controls are tested and validated during acceptance test phase maintaining the security controls throughout the life cycle of the product or the application product or service specifications must include the requirements for security controls.

Contracts with the Providers must also address the identified security requirements.

**2.7.4 Maintainability**

The following steps should be undertaken to assess maintainability statically:

A list of maintainability factors to be included in the assessment should be devised e.g. structure, complexity.

Each factor (or group of factors) should be assigned a weighting to indicate its importance to the overall maintainability of the system. Each factor will have a maximum score of 10. The higher the score the less maintainable the system.

During the assessment a score is awarded against each factor on the list. For example, a relatively old system may be awarded a score of 8 out of 10 to indicate that due to its age the system will relatively difficult to maintain.

The scores for each of the factors assessed are then multiplied by the appropriate weighting and the resultant products are then summed to give an overall score which forms the maintainability Measure of the system (the lower the score, the better the maintainability of the software system).

Example factors which can be used in a maintainability assessment are given below; the list is not exhaustive and should be modified to suit an individual organization (although it is helpful if the same list is used throughout the organization so comparisons between systems can be made):

**2.7.5 Other requirements.**

**Functional Requirement**

* System must be fast and efficient
* User friendly GUI
* Reusability
* Performance
* System Validation input
* Proper output

**Non Functional Requirement**

* Accessibility
* Capacity, current and forecast
* Compliance
* Documentation
* Disaster recovery
* Efficiency
* Effectiveness
* Extensibility
* Fault tolerance
* Interoperability
* Maintainability
* Privacy
* Portability
* Quality
* Reliability
* Resilience
* Response time
* Robustness
* Scalability
* Security
* Stability
* Supportability
* Testability

**UML Diagrams:**

**Use Case Diagram:**

****

**Activity Diagram:**

**Doctor Activity**

****

**Patient Activity**

****

**Sequence Diagram:**



**Class Diagram:**



**Component Diagram**

****

**Deployment Diagram**



**DFD 0**

****

**DFD 1**

****

**DFD 2**

****