CHAPTER 4

SYSTEM REQUIREMENT SPECIFICATIONS

A **software requirements specification** (SRS) is a detailed description of a software system to be developed with its functional and non-functional requirements. The SRS is developed based the agreement between customer and contractors. It may include the use cases of how user is going to interact with software system. The software requirement specification document consistent of all necessary requirements required for project development. To develop the software system we should have clear understanding of Software system. To achieve this we need to continuous communication with customers to gather all requirements.

A good SRS defines the how Software System will interact with all internal modules, hardware, communication with other programs and human user interactions with wide range of real life scenarios. Using the Software requirements specification (SRS) document on QA lead, managers creates test plan. It is very important that testers must be cleared with every detail specified in this document in order to avoid faults in test cases and its expected results.

It is highly recommended to review or test SRS documents before start writing test cases and making any plan for testing. Let's see how to test SRS and the important point to keep in mind while testing it.

1. Correctness of SRS should be checked. Since the whole testing phase is dependent on SRS, it is very important to check its correctness. There are some standards with which we can compare and verify.

2. Ambiguity should be avoided. Sometimes in SRS, some words have more than one meaning and this might confuse tester's making it difficult to get the exact reference. It is advisable to check for such ambiguous words and make the meaning clear for better understanding.

3. Requirements should be complete. When tester writes test cases, what exactly is required from the application, is the first thing which needs to be clear. For e.g. if application needs to send the specific data of some specific size then it should be clearly mentioned in SRS that how much data and what is the size limit to send.

4. Consistent requirements. The SRS should be consistent within itself and consistent to its reference documents. If you call an input "Start and Stop" in one place, don't call it "Start/Stop" in another. This sets the standard and should be followed throughout the testing phase.

5. Verification of expected result: SRS should not have statements like "Work as expected", it should be clearly stated that what is expected since different testers would have different thinking aspects and may draw different results from this statement.

6. Testing environment: some applications need specific conditions to test and also a particular environment for accurate result. SRS should have clear documentation on what type of environment is needed to set up.

7. Pre-conditions defined clearly: one of the most important part of test cases is pre-conditions. If they are not met properly then actual result will always be different expected result. Verify that in SRS, all the pre-conditions are mentioned clearly.

8. Requirements ID: these are the base of test case template. Based on requirement Ids, test case ids are written. Also, requirements ids make it easy to categorize modules so just by looking at them, tester will know which module to refer. SRS must have them such as id defines a particular module.

9. Security and Performance criteria: security is priority when a software is tested especially when it is built in such a way that it contains some crucial information when leaked can cause harm to business. Tester should check that all the security related requirements are properly defined and are clear to him. Also, when we talk about performance of a software, it plays a very important role in business so all the requirements related to performance must be clear to the tester and he must also know when and how much stress or load testing should be done to test the performance.

10. Assumption should be avoided: sometimes when requirement is not cleared to tester, he tends to make some assumptions related to it, which is not a right way to do testing as assumptions could go wrong and hence, test results may vary. It is better to avoid assumptions and ask clients about all the "missing requirements" to have a better understanding of expected results.

11. Deletion of irrelevant requirements: there are more than one team who work on SRS so it might be possible that some irrelevant requirements are included in SRS. Based on the understanding of the software, tester can find out which are these requirements and remove them to avoid confusions and reduce work load.

12. Freeze requirements: when an ambiguous or incomplete requirement is sent to client to analyse and tester gets a reply, that requirement result will be updated in the next SRS version and client will freeze that requirement. Freezing here means that result will not change again until and unless some major addition or modification is introduced in the software.

Most of the defects which we find during testing are because of either incomplete requirements or ambiguity in SRS. To avoid such defect it is very important to test software requirements specification before writing the test cases. Keep the latest version of SRS with you for reference and keep yourself updated with the latest change made to the SRS. Best practice is to go through the document very carefully and note down all the confusions, assumptions and incomplete requirements and then have a meeting with the client to get them clear before development phase starts as it becomes costly to fix the bugs after the software is developed. After all the requirements are cleared to a tester, it becomes easy for him to write effective test cases and accurate expected results.

4.1 Functional Requirements

A function of software system is defined in functional requirement and the behavior of the system is evaluated when presented with specific inputs or conditions which may include calculations, data manipulation and processing and other specific functionality. The functional requirements of the project are one of the most important aspects in terms of entire mechanism of modules.

Image acquisition and preprocessing

The model should be able to preprocess the image so as to remove the noise from the images, eliminate the spurious pixels and harshness in the image and also remove defected pixels and resized the all the images to some certain size ratio.

Feature extraction

The model should be able to effectively extract all the features from the images in the form of feature vector in order to build and train a CNN classification model.

Classification model

The model should be able to identify the species of a bird image given as an input by the user.

User friendly Interface

The developed model should have a user-friendly interface.

4.2 Non-Functional Requirements

Nonfunctional requirements describe how a system must behave and establish constraints of its functionality this type of requirements is also known as the system's quality attributes. Attributes such as performance, security, usability, compatibility are not the feature of the system, they are a required characteristic. They are "developing" properties that emerge from the whole arrangement and hence we can't compose a particular line of code to execute them. Any attributes required by the customer are described by the specification. We must include only those requirements that are appropriate for our project.

Reliability

The structure must be reliable and strong in giving the functionalities. The movements must be made unmistakable by the structure when a customer has revealed a couple of enhancements. The progressions made by the Programmer must be Project pioneer and in addition the Test designer.

Maintainability

The system watching and upkeep should be fundamental and focus in its approach. There should not be an excess of occupations running on diverse machines such that it gets hard to screen whether the employments are running without lapses.

Scalability

The framework should be sufficiently adaptable to include new functionalities at a later stage. There should be a run of the mill channel, which can oblige the new functionalities.

Performance

The framework will be utilized by numerous representatives all the while. Since the system will be encouraged on a single web server with a lone database server outside of anyone's ability to see, execution transforms into a significant concern. The structure should not capitulate when various customers would use everything the while. It should allow brisk accessibility to each and every piece of its customers. For instance, if two test specialists are all the while attempting to report the vicinity of a bug, then there ought not to be any irregularity at the same time.

Portability

The framework should to be effectively versatile to another framework. This is obliged when the web server, which s facilitating the framework gets adhered because of a few issues, which requires the framework to be taken to another framework.

Flexibility

Flexibility is the capacity of a framework to adjust to changing situations and circumstances, and to adapt to changes to business approaches and rules. An adaptable framework is one that is anything but difficult to reconfigure or adjust because of diverse client and framework prerequisites. The deliberate division of concerns between the trough and motor parts helps adaptability as just a little bit of the framework is influenced when strategies or principles change.