Software Requirement:

- A condition or capability needed by a user to solve a problem or achieve an objective.
- A condition or capability that must be met or processed by a system or system component to satisfy a contract, standard, specification, or other formally imposed documents.
- A documented representation of a condition or capability.

Software requirement analysis is separated into five areas:

a) Problem Recognition:
- Discover and understand the system requirements
- Refine the requirements

b) Evaluation & Synthesis:
- Alternative solutions
- Focus on what solution should be selected or used instead of how to implement a solution

c) Modeling:
- Information and control flow
- Operation, behavior.

d) Specification:
- Functions and performance
- Interfaces between system elements
- System constraints

e) Review:
- Find out problems like: Uncertain stated requirements, Conflicting requirements, Errors in requirements elicitation and analysis, Lack of conformance to quality standards.

Types of Requirements:

1. Functional Requirement:

- Requirements, which are related to functional aspect of software fall into this category.
- They define functions and functionality within and from the software system.
- EXAMPLES:
  - Search option given to user to search from various invoices.
  - User should be able to mail any report to management.
  - Users can be divided into groups and groups can be given separate rights.
  - Should comply business rules and administrative functions.
  - Software is developed keeping downward compatibility intact.

2. Non - Functional Requirement:

- Requirements, which are not related to functional aspect of software, fall into this category. They are implicit or expected characteristics of software, which users make assumption of.
- **Example:**
  - Security, Logging, Storage, Configuration, Performance, Cost, Interoperability, Flexibility, Disaster recovery, Accessibility
- **Requirements are categorized logically as:**
  - **Must have:** Software cannot be said operational without them.
  - **Should have:** Enhancing the functionality of software.
  - **Could have:** Software can still properly function with these requirements.
  - **Wish list:** These requirements do not map to any objectives of software.

    While developing software, 'Must have' must be implemented, 'Should have' is a matter of debate with stakeholders and negation,

whereas 'could have' and 'wish list' can be kept for software updates.

3. Domain Requirement:

- Domain requirements are an important part of any software or hardware or hybrid system.
- A software is widely accepted if it is:
  - Easy to operate
  - Quick in response
  - Effectively handling operational errors
  - Providing simple yet consistent user interface.
- User acceptance majorly depends upon how user can use the software.
- Domain requirements are the only way for users to perceive/observe the system.
- A well performing software system must also be equipped with attractive, clear, consistent and responsive user interface. Otherwise the functionalities of software system cannot be used in convenient way.
- A system is said be good if it provides means to use it efficiently.
- Domain requirements are briefly mentioned below:
  - Content presentation
  - Easy Navigation
  - Simple interface
  - Responsive
  - Consistent UI elements
  - Feedback mechanism
  - Default settings
  - Purposeful layout
  - Strategically use of color and texture
  - Provide help information
  - User centric approach
  - Group based view settings.

Fact Finding Techniques:

- Requirement gathering is carried out to obtain the requirements from system users, customers and stockholders.
- This process is also sometimes called as **requirements gathering technique** or **fact finding technique**.
- Before requirements can be analyzed, modeled or specified they must be gathered through gathering techniques.

1. Interview:

- Interview is fact finding technique where two parties work together; one person acts as an interviewer and other as interviewee.
- The person who collects information is known as system analyst, who plays the role of interviewer.
- System analyst collects fact from individuals who are known as interviewees.
- The interviewees are usually present user of the existing system or possible user of the proposed system.
- **Types of interview:**
  - **Structured Interview:**
    - Structured interviews are those where the interviewee is asked a standard set of questions in a particular order.
    - All interviewees are asked the same set of questions.
  - **Unstructured Interview:**
    - The unstructured interviews are undertaken in a question and answer format.
    - More flexible than structured interview and can be very rightly used to gather general information about the system.
- **Advantages:**
  - Beneficial for those individuals who cannot communicate effectively in writing.

- Allow the analyst to determine are of confusion and indicates to points for resistance to the proposed system.
- Permit the interviewees to replay freely and openly to any type of question.

- **Disadvantages:**
  - This technique fails if suitable environment is not provided for conducting interview.
  - Interview can be subject to bias if the interviewer has a closed mind about the difficulty.

2. Questionnaire:

- Questionnaires gather data from both large and small groups of people.
- This method can be adopted and used only by a skilled analyst.
- Developing of questionnaires requires in depth planning and usually more than one draft is necessary.
- Questionnaires are useful when:
  - The analyst need to gather information from a large number of people. It is not possible to interview each individual.
  - Also if the time is very short, in that case also questionnaires are useful.
- This method is very useful for attaining information from people who are concerned with the usage of the system and who are living in different countries.
- Questions should be short, easy to understand, unbiased, non-threatening and specific.
- **Types of Questionnaire:**
  - **Open Response Based:**
    - Main objective of open response Questionnaire is to gather information and data about the critical design features of the system.

- Respondents must answer the question in their own words. Space is provided under each question for the response. It gives a person the chance to respond in detail.
- Although open ended questions are important, they are time-consuming and should not be over-used.

- **Closed Response Based:**
  - The objective of closed response questionnaire is to collect the factual information of the system.
  - The closed questions can be of various types and the most common ones are listed below:
    a) Multiple Choice questions
    b) Rating scale
    c) Rank ordering
    d) Dichotomous questions

- **Advantages:**
  - Questionnaires gather data from both large and small groups of people.
  - If the time is very short, in that case also questionnaires are useful.
  - Many questions can be asked without utilizing extra time.
- **Disadvantages:**
  - Not useful for the people who are not good at answering in written manner.
  - Customers may answer randomly without reading the question properly.
  - If the customer doesn't answers any question then this becomes an issue while developing the software.

3. Record Review:

- Records and reports are the gathering of information and data collected over the time by the users about the system and its operations.

- For better understanding of any existing system is to review its related records, existing documents, forms or files. This process is known as **record review**.
- It starts at the preliminary stage of the system study or later on in the study for measuring actual operations with what the records indicate.
- Records and reports may have a limited if they are not up to date or if some essential links are missing.
- An analyst always gets facts from documents. An existing system can be better understood by examining existing documents, forms and files.
- **Record review may include:**
  - Written policy
  - Rules and regulation
  - Forms and documents

4. Observation:

- It is also known as on site observation or direct observation, where the system analyst personally goes to the site and discovers the functionalities of the system.
- As an observer, the system analyst can gain direct knowledge of the activities, operations, processes of the system non-site.
- Here the role of the system analyst is of an information seeker.
- This technique is useful when on needs to actually observe how documents are handled, how operations and activities are carried out.
- **Observation can look for:**
  - Operational inefficiencies
  - The usage of files and documents
  - Inter communication channels
  - Interruptions in the normal flow of work
  - Alternative routes and procedures
- **Advantages:**
  - Facts gathered using this technique is highly reliable.

- Inexpensive way of collecting data.
- The system analyst can make out task that have been missing or wrongly illustrated by other fact finding techniques.
- Practical experience

- **Disadvantages:**
  - This technique is time consuming.
  - System analyst should not jump to conclusion or draw assumption from just a minute sample of observation rather the analyst should be more patient in gathering the information.

5. Joint Application Development [JAD]:

- JAD is a methodology that involves the client or end user in the design and development of an application through a succession of collaborative workshops known as JAD sessions or in other words, a group information gathering technique of systems development.
- Joint Application Design was developed by IBM in 1970's. JAD was designed to bring system developers and users of varying backgrounds and opinions together in a productive and creative environment.
- The aim is to get all groups with a stake in the project to work together by getting the team together in meeting rooms with U-shaped or round tables, white boards, overhead projectors and audio-visual tools. This allows everyone in the room to talk and be heard.
- By hearing each other the team is able to produce the appropriate systems requirements.
- Therefore JAD sessions require the right mix of people actively participating in order to achieve the goals of the session.
- **A typical JAD has following roles:**
  - **The Users:** The end user should have good knowledge and experience in the field, they are responsible for input concerning system design and are the only participants with a clear idea of how the system will be used in the work.

- **IT Representatives:** This includes programmers and other developers. This group is present to assess technical feasibility and learn about future plans. For this role, the representative should be able to listen as well communicate ideas and technical information.
- **The Session Leader:** This individual organizes the sessions and keeps the group focused on the task. A good knowledge of the business processes, good interpersonal skills and an outstanding ability to organize and lead group is a must.
- **System Analyst:** The analyst attends to learn from the users and managers to be able to better analyze the entire system.

- **Advantages:**
  - JAD actively involves users and management in the development project.
  - JAD helps to avoid the requirements which cause trouble during implementation and acceptance.
- **Disadvantages:**
  - Slow communication and long feedback time.
  - Can be weak or no support from upper management.

6. Facilitated Application Specification Technique [FAST]

- FAST is a technique for gathering requirements during early stages of analysis and specification.
- The main objective of this technique is to cover the gap between what the developers think they are going to design and what customers believe they will receive from the particulars program.
- This is a team oriented approach for requirement gathering.
- It is a technique where the meeting of software customers and developers happens at a neutral site (no home advantage).
- Any discussion about expectations from proposed software system or cost associated with the system at customer's premises or developer's

premises may lead to strong disagreements on account of home advantage.

- To avoid this, meeting between selected end - users and representatives from developers is conducted at neutral sites, which is facilitated by a third party.
- Such meetings are highly successful because the two parties arrive at agreement very fast because of neutral environment.
- **Goals of FAST:**
  - Problem Identification
  - Solution of Elements
  - Approaches negotiate
  - Requirements specified
  - Such meeting needs lot of preparation in advance in the form of well-defined agenda, points for discussion and so on. Facilitator generally assists in providing necessary infrastructure and documentation. JAD (Joint Application Development) approach frequently uses in this method.
- **Process for FAST:**
  - Initial meetings between the developer and customer occur and basic questions and answers help to establish the scope of the problem and the overall perception of a solution.
  - Out of these initial meetings, the developer and customer write a one or two page "product request." A meeting place, time, and date for FAST are selected and a facilitator is chosen.
  - Attendees from both the development and customer/user organizations are invited to attend.
  - The product request is distributed to all attendees before the meeting date.
- **Basic guidelines:**
  - Meetings are conducted at a neutral site attended by both developers and users.

- The group establishes rules for preparation and participation.
- An agenda is suggested that with enough formality to cover all important points but informal enough to encourage the free flow of ideas.
- A facilitator controls the meeting.
- A definition mechanism (can be work sheets, flip charts, or wall stickers or an electronic bulletin board, chat room or virtual forum) is used.

- **Advantages:**
  - Helpful because meeting is conducted at a neutral site.
  - Team oriented approach. Both developer and user will work together.
- **Disadvantages:**
  - Weak meeting if facilitator not prepared well for the meeting.
  - Ineffective, if there is any communication gap between the user and the developer or system analyst.

7. Rapid Application Development [RAD]:

- This application development methodology goes further than JAD in reducing the time taken to develop an application, it is not always as structured as the JAD and focuses more on software development than JAD.
- RAD is based on the concept that systems can be developed faster and of higher quality by gathering requirements through workshops or focus groups, prototyping, reiterative user testing of designs, use of already existing software components and less formality in reviews and other team communication.
- RAD uses small integrated teams of developers, end users and IT technical resources and short iterative development cycles to optimize its goals of speed, effective informal communication, unity of vision and purpose, and simple project management.

- This RAD Process continues until the system is completely developed and users are satisfied.
- **RAD Phases and Activities:**
  - Requirements planning phase/activity
  - User design phase/activity
  - Construction/Building phase/activity
  - Cutover/Implementation phase/activity
- **RAD Objective:**
  - To cut development time and expense by involving the users in every phase of systems development.
  - Successful RAD team must have IT resources, skills, and management support.
  - Helps a development team design a system that requires a highly interactive or complex user interface
- **Advantages:**
  - Might allow less time to develop quality, consistency, and design standards.
  - Systems can be developed more quickly with significant cost savings.
- **Disadvantages:**
  - Requires highly skilled developers/designers.

8. Sampling of documentation, forms and files:

- Sampling is the process of collecting a representative sample of documents, forms and records.
- Sample can be of the following data:
  - Organizational chart
  - Documents that describe the problem.
  - Standard operating procedure for current system.
  - Sample databases.
  - Flowchart and other system documentation.

Software Requirement Specification [SRS]:

- The specification is a contract between client and developer.
- It must specify what the product must do and the constraints on the product. It is produced when the analysis task is near to complete.
- SRS is a complete description of the system and functionality of the system.
- SRS take two types of requirements:
  - **Behavioral requirements (Functional requirements):**
    - This defines what the system must do that is how the inputs are transformed into output.
  - **Non-Behavioral requirement (Non-Functional requirement)**
    - This defines the attributes of the system as it performs its tasks that is, system's required levels of efficiency, reliability, security, maintainability, portability, visibility, capacity and so on.

Characteristics of SRS:

a) Complete:

- An SRS is complete if, and only if, it include following elements:
  - All significant requirements, whether related to functionality performance, design constraints, attributes, or external interface.
  - Full label and references to all figures, tables, and diagrams in the SRS and definition of all terms and unit of measure.

b) Consistent:

- System functions and performance level must be compatible and the required quality features like reliability, safety, security, etc. must not contradict the utility of the system.

c) Correct:

- An SRS is correct if, and only if, every requirement stated there in is one that the software shall meet. Traceability makes this procedure easier and less prone to error.

d) Modifiable:

- Related concern must be grouped together and unrelated concerns must be separated. Requirements documents must have a logical structure to be modified.

e) Ranked:

- All significant requirements, whether related to functionality performance, design constraints, attributes, or external interface. - Raking specification statements according to stability and importance are established in the requirements document's organization and structure. The larger and more complex the problem addressed by the requirements specification, the more difficult the task is to design a document that aids rather than inhibits understanding.

f) Testable:

- Requirement specification must be stated in such manner that one can test it against pass/fail assessment criteria. All derived from the specification itself referenced information.

g) Traceable:

- Each requirement stated within SRS document must be uniquely identified to achieve traceability.

h) Unambiguous:

- A Statement of a requirement is unambiguous if it can only be interpreted one way. This is more difficult attribute to achieve using natural language.

i) Valid:

- To validate requirement specification all the project participants, manager, engineers and customer representative, must be able to understand, analyze and accept or approve it.

j) Verifiable:

- In order to be verified, requirement specifications at one level of abstraction must be consistent with those at another level of abstraction.

Components of SRS:

1. Functionality:

- Functional requirements identify the expected behavior of the system whose outputs should be produced from the given input.
- They describe the relationship between the input and output of the system.
- For each functional requirement, a detailed explanation of all the data inputs and their source, the unit of measure, and the range of valid must be specified.
- All the operations to be performed on the input data to obtain the output should be specified. This includes specifying the validity checks on the input and output data, parameters affected by the operation, and equation or other logical operation that must be used to transform the inputs into corresponding outputs.
- A significant part of the specification is the system behavior in abnormal situation, like invalid input or error computation. The functional requirement must clearly state what the system should do if such situation occurs.

2. Performance:

- It specifies the performance restrictions on the software system.
- All the requirements related to the performance characteristics of the system must be clearly specified.
- There are two types of performance requirements:
  - **Static:**
    - These requirements are like the number of terminal to be supported, the number of simultaneous users to be supported, the number of files that the system has to process and their sizes.
    - These are also called **capacity requirements** of the system.
  - **Dynamic:**
    - These requirements specify constraints on execution behavior of the system.
    - These typically include response time and throughput on the system. **Response time** is the expected time for the completion of an operation under specify situation. **Throughput** is the expected number of operation that can be performed in a unit time.

3. Design Constraints:

- There are a number of issues in the client's environment that may restrict the choice of designer leading to design constraint. That includes standards that must be followed, resource limit, operating environment and security policy, and other policies that may have impact on the design of the system.
- This includes:
  - **Standards compliance:**
    - This specifies the requirements for the standards the system must follow. The standard may include the report format and accounting procedure.

- **Hardware limitation:**
  - The software may have to operate on some existing or predetermine hardware, thus imposing restrictions on the design. Hardware limitation may include the type of machines to be used, operating system available on the system, language supported and limit on primary or secondary storage.
- **Security:**
  - This factor is becoming more and more essential. These requirements place limitations on the use of the certain commands, control access to data, provide different kind of access requirements for different people, require the use of password and cryptography techniques.

4. External Interfaces:

- These include interface specification part; all the interaction of the software with people, hardware and other software should be clearly specified.
- **For user interface,** the characteristics of each user interface of the software should be specified.
- **For hardware interface,** the SRS should specify the logical characteristics of each interface between the software product and hardware components.