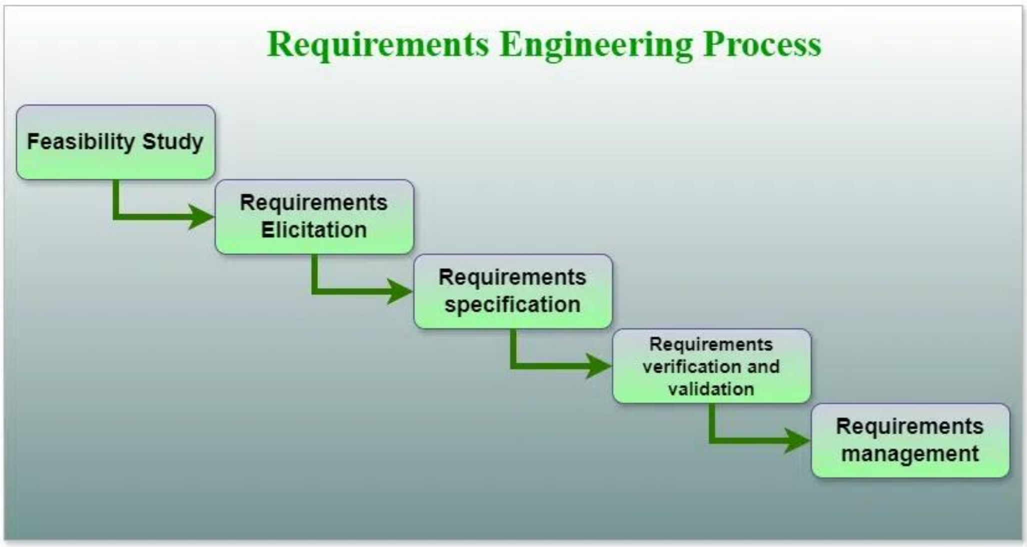
Requirement Engineering

Requirement Engineering is formally defined as the process of systematically identifying, documenting, and managing the requirements of a software system. It involves understanding what the stakeholders need from the system, ensuring that these needs are accurately captured and clearly specified, and then maintaining these requirements throughout the software development life-cycle.

Process or Phases or Activities of Requirement Engineering:

The RE process encompasses four distinct phases, each critical to the project's success:


Requirements Engineering Process
Feasibility Study
Requirements Elicitation
Requirements specification
Requirements verification and validation
Requirements management

1) Feasibility study:

This initial phase assesses whether the project should proceed. The feasibility study examines the proposed system's technical, economic, legal, operational, and schedule feasibility. It helps the stakeholders understand the project's practical aspects, including potential benefits and limitations. Understanding the feasibility of the project can help set the budget and reduce unnecessary expenses. This helps make the process more cost efficient.

Types of Feasibility:

(i) Technical Feasibility: It checks whether the required technology, tools, hardware, software, and technical expertise are available to build and run the system.
(ii) Operational Feasibility: It checks whether the new system will actually work in the real environment and be accepted by users
(iii) Economic Feasibility: Economic feasibility decides whether the necessary software can generate financial profits for an organization and is it budget friendly.
(iv) Legal Feasibility: It checks whether the proposed system or project follows all the laws, rules, and regulations.
(v) Schedule Feasibility: It checks whether the project can be completed within the required timeframe (deadline).

2) Requirement elicitation and analysis:

In requirement elicitation, the Requirements Engineering Practitioner gathers detailed information about what the stakeholders need from the proposed system using techniques like interviews, focus groups, workshops, shadowing, survey etc. Once gathered, the requirements undergo thorough analysis to ensure they are clear, actionable, achievable, and verifiable. This stage is critical for clarifying the stakeholders' actual needs and ensuring that these needs are well understood and agreed upon. This helps bring clarity to the project and its purpose, making sure that there is a plan at play to satisfy the stakeholders' needs.

3) Requirement Specification:

The Software Requirement Specification (SRS) document is a comprehensive description of the behaviour of the system to be developed. It includes detailed descriptions of the system's functions, capabilities, interfaces, and interactions with users. This document should be clear enough for the development team to implement the system and for the test team to validate the implemented system. It acts as a contract between the development team and the customer.

4) Requirement Validation:

This phase checks the SRS for errors and ensures that it accurately reflects the client's requirements. Validation techniques include requirement reviews, feedback, ratings, prototype testing, simulation, and model validation. The goal is to make sure that the requirements specified in the Software Requirement Specification are complete, consistent, correct, relevant, and testable. This helps mitigate unnecessary steps from the process making it simpler.

5) Requirement Management:

Requirement management is the process of analyzing, documenting, tracking, prioritizing, and agreeing on the requirement and controlling the communication with relevant stakeholders. This stage takes care of the changing nature of requirements. It should be ensured that the SRS is as modifiable as possible to incorporate changes in requirements specified by the end users at later stages too. Modifying the software as per requirements in a systematic and controlled manner is an extremely important part of the requirements engineering process.

Functional Requirements

Functional requirements describe the specific behaviors, functions, or operations that a system must perform. These are the features and capabilities that the system must have to fulfill its intended purpose. They answer the question, "What should the system do?"

Examples:

i. User Authentication: The system must allow users to log in using a username and password.The system must provide functionality for users to reset their passwords.

ii. Data Management: The system must allow users to create, read, update, and delete (CRUD) records in a database. The system must generate reports based on user-selected criteria.

iii. Transaction Processing: The system must process payments and issue receipts. The system must validate transactions before completion.

Characteristics:

(i) Specific and detailed: Clearly defines what the system should do.

(ii) User-focused: Directly impacts the user's interaction with the system.

Non-Functional Requirements

These define the system's quality attributes, constraints, and conditions under which the system must operate. They outline how the system should perform. They are often referred to as quality attributes of the system, addressing aspects like performance, security, usability, and reliability. Non-functional requirements may be more critical than functional requirements. If these are not met, the system may be useless.

Examples:

i. Performance: The system must support 1000 concurrent users without significant performance degradation. Response time for any user action should not exceed 2 seconds.
ii. Security: The system must encrypt all sensitive data both at rest and in transit. The system must require multi-factor authentication for access to critical functions.
iii. Usability: The system must be accessible to users with disabilities, adhering to the WCAG 2.1 standards. The system's interface must be intuitive, with users able to complete key tasks with minimal training.
iv. Reliability: The system must have an uptime of 99.9% during business hours. The system must recover from crashes within 5 minutes without data loss.

Constraints:

Constraints are the **limitations or restrictions** that a software system must operate under while being developed or used. Examples:

I. The project must be completed **within 3 months**.
II. The software must comply with **data privacy laws**.
III. The hospital management system must have **99.9% uptime**.
IV. All user passwords must be stored using **encryption**.