# Requirements Analysis in SDLC and Algorithms

## Introduction

Requirements Analysis is a critical phase in the Software Development Life Cycle (SDLC) and algorithm development, as it lays the groundwork for the entire project. This document provides an overview of its role and importance in both contexts.

## Requirements Analysis in SDLC

Definition: This phase involves gathering, analyzing, and documenting the functional and non-functional requirements of a system from stakeholders and end-users. The goal is to ensure that the software being developed meets the intended purpose and user needs.

### Key Steps in Requirements Analysis:

* Gathering Requirements: Conduct stakeholder interviews, perform surveys and questionnaires, analyze existing systems or workflows, and organize brainstorming sessions.
* Analyzing Requirements: Assess feasibility, risks, and constraints; resolve ambiguities and conflicts; prioritize requirements based on business value.
* Documenting Requirements: Use formal documentation methods like Software Requirement Specifications (SRS), employ use cases, user stories, and diagrams (e.g., flowcharts, UML diagrams).
* Validating Requirements: Ensure requirements are clear, complete, and testable; gain formal approval from stakeholders.

### Output:

* Functional Requirements: Define specific behaviors or functions (e.g., 'The system must allow users to log in').
* Non-functional Requirements: Define system qualities (e.g., performance, security, scalability).

### Significance in SDLC:

• Prevents scope creep by clearly defining what the software will do.  
• Aligns stakeholder expectations with the development team's understanding.  
• Provides a blueprint for design and implementation.

## Requirements Analysis in Algorithms

Definition: In algorithm development, requirements analysis focuses on understanding the problem domain and the specific goals the algorithm must achieve.

### Key Steps in Algorithm Requirements Analysis:

* Problem Definition: Clearly state the problem to be solved. Identify inputs, outputs, and constraints.
* Understanding Constraints: Consider time complexity (speed of execution), space complexity (memory usage), scalability, and adaptability to edge cases.
* Specifying Goals: Define success criteria (e.g., accuracy, efficiency). Determine whether the algorithm needs to be exact, approximate, or heuristic.
* Validating Feasibility: Analyze if the problem is solvable with the given constraints. Consider trade-offs between competing requirements (e.g., speed vs. accuracy).

### Output:

• Problem statement with clearly defined inputs, outputs, and constraints.  
• A roadmap for designing an efficient algorithm.

### Significance in Algorithm Development:

• Ensures the algorithm aligns with the intended use case.  
• Helps identify optimal approaches (e.g., greedy algorithms, dynamic programming, etc.).  
• Reduces development time by avoiding rework due to unclear goals.

## Relationship Between Requirements Analysis in SDLC and Algorithms

In SDLC, requirements analysis ensures the software addresses user needs, which influences the design of algorithms used within the software. For algorithms, requirements analysis ensures that the computational processes meet system requirements and constraints, feeding into the SDLC design and implementation phases.

In essence, effective requirements analysis bridges the gap between what stakeholders need and how developers and systems deliver it.