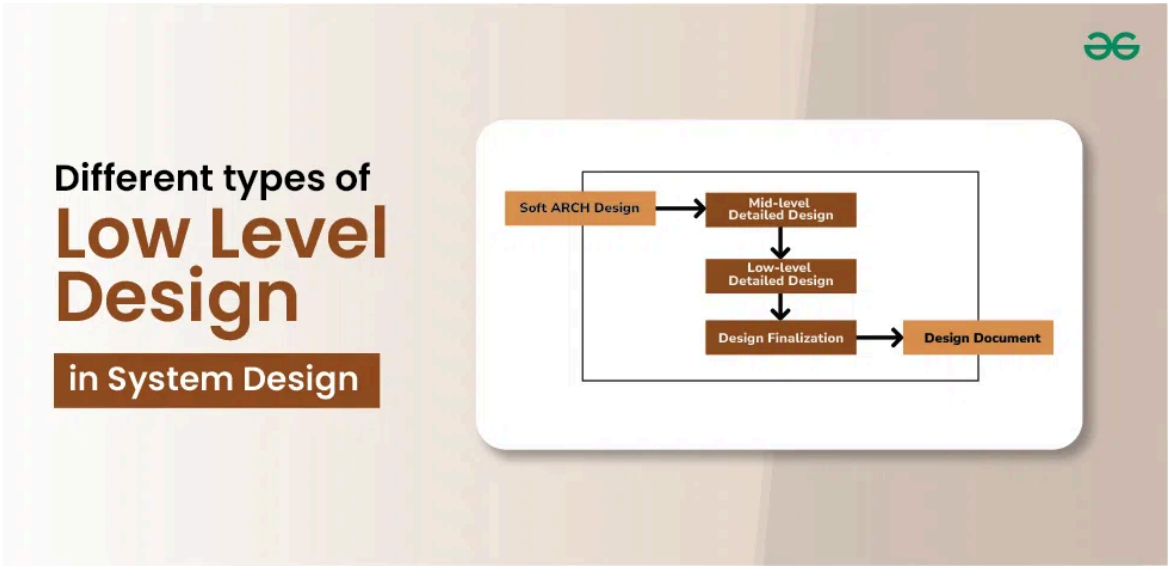
Different types of Low Level Design in System Design

Last Updated : 01 Aug, 2024

Low-level design refers to the process of defining the detailed, functional design of a software system or component. It involves specifying the individual modules, data structures, algorithms, interfaces, and inputs/outputs of a system. The purpose of low-level design is to provide a clear and precise description of how the system should behave and how its different components will interact with each other.

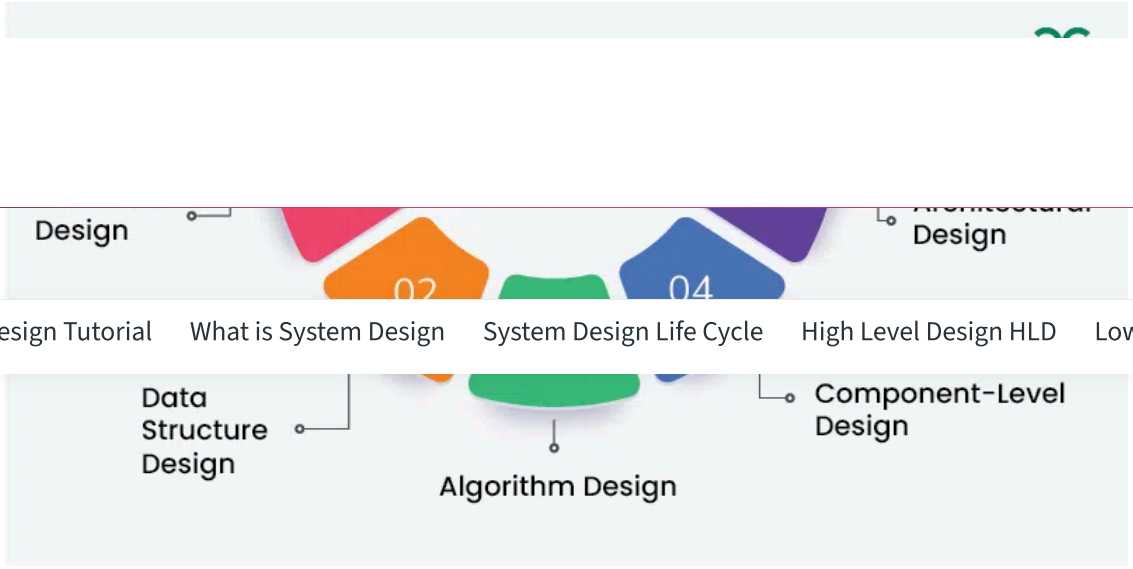
Data Flow Design is a software engineering technique to represent the data flow within a system. It is an important aspect of low-level design, which provides a detailed blueprint for the implementation of a software system.



Different types of Low Level Design in System Design

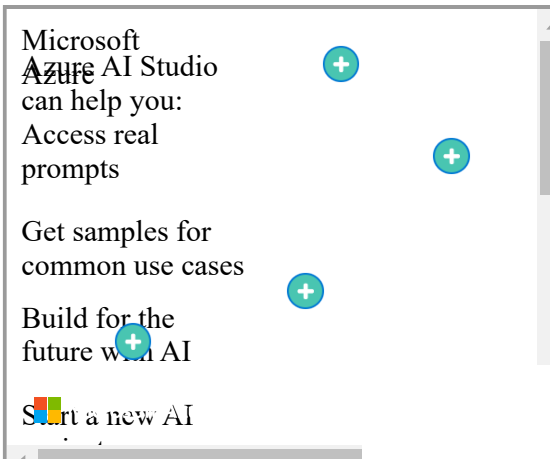
Types of Low-Level Design

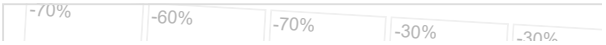
Each type of low-level design is an important part of the software development process and helps to ensure that a software system is well-designed, reliable, and efficient. There are several types of low-level design,



1. Data Flow Design

Data Flow Design (DFD) is a technique used in low-level design to represent the flow of data and information through a system or application. The purpose of DFD is to identify the different components of the system and how they interact with each other, in order to optimize the design and improve its overall functionality. Each diagram represents a different level of detail, starting with a high-level overview and gradually zooming in on the individual components of the system.





The diagrams consist of these basic components:

- **Data sources and sinks:** These are the sources and destinations of data within the system. A data source could be an external entity that provides

- **Processes**: These are the individual components of the system that manipulate or transform the data. Processes are represented as boxes on the DFD diagram, and each process is labeled with a name and a description of its functionality.
- **Data flows**: These are the pathways through which data moves from one component of the system to another. Data flows are represented as arrows on the DFD diagram, and each arrow is labeled with a description of the data being transferred.
- **Data stores**: These are the locations within the system where data is stored for later use. Data stores are represented as rectangles on the DFD diagram, and each data store is labeled with a description of the data it contains.

DFD is useful in low level design because it provides a clear and concise representation of how data flows through the system. This makes it easier to identify potential bottlenecks or areas where the design could be improved.

2. Data Structure Design

Data structure design is a key component of software engineering that involves the selection, implementation, and optimization of data structures to meet the requirements of a particular application or system. Data structures are a way of organizing and storing data in memory so that it can be accessed and manipulated efficiently.

There are many different types of data structures, each with its own strengths and weaknesses. Some common data structures include:

- **Array**s: A simple data structure that stores elements in contiguous memory locations.
- **Linked lists**: A data structure that consists of a series of nodes, each containing a data element and a pointer to the next node in the list.
- **Trees**: A hierarchical data structure that consists of nodes connected by edges, with a single root node at the top.
- **Graphs**: A data structure that consists of nodes connected by edges, where

When designing data structures, there are several key factors to consider. These include:

- **Performance**: The data structure should be optimized for the specific operations that will be performed on it. For example, if the data structure will be frequently searched, a hash table or binary search tree may be more appropriate than an array.
- **Memory usage:** The data structure should be designed to use as little memory as possible, while still meeting the application's requirements.
- **Ease of use**: The data structure should be easy to understand and use, with clear documentation and intuitive interfaces.
- **Extensibility**: The data structure should be designed to allow for future modifications or extensions, as the requirements of the application may change over time.

Data structure design is an important part of software engineering, as the choice of data structure can have a significant impact on the performance and scalability of a system. By carefully selecting and optimizing data structures, developers can create systems that are efficient, easy to use, and able to meet the needs of their users.

3. Algorithm Design

Algorithm design is the process of creating a step-by-step procedure for solving a problem or achieving a specific goal. In computer science, algorithm

to perform a wide range of tasks, from sorting data to searching for information and making decisions.

The process of algorithm design typically involves the following steps:

- **Problem definition:** The first step in algorithm design is to clearly define the problem that needs to be solved or the goal that needs to be achieved. This involves understanding the input data, any constraints on the problem, and the desired output.
- **Input/output analysis**: Once the problem is defined, the next step is to analyze the input data and desired output to determine the most efficient and effective way to achieve the desired result.
- **Algorithm design**: Based on the input/output analysis, the algorithm designer creates a step-by-step procedure for solving the problem or achieving the goal. This typically involves a combination of mathematical and logical operations, as well as the use of data structures and algorithms.
- **Algorithm optimization**: Once the initial algorithm is designed, the algorithm designer may need to optimize the algorithm to improve performance, reduce memory usage, or achieve other goals.
- **Algorithm testing**: The final step in algorithm design is to test the algorithm to ensure that it produces the desired output for a variety of input data.

There are many different types of algorithms, including sorting algorithms, searching algorithms, and graph algorithms, among others. The choice of algorithm will depend on the specific problem being solved and the requirements of the system.

4. Component-Level Design

Component-level design is the process of designing the individual components of a software system in detail. It involves defining the structure and behavior of each component, as well as specifying the interfaces between components and any interactions or dependencies between them. The goal of component-level design is to create components that are well-designed, reusable, and easy to

- **Component identification**: The first step in component-level design is to identify the individual components that make up the system. This may involve breaking down larger system components into smaller, more manageable pieces.
- **Component specification**: Once the components have been identified, the next step is to specify the behavior and structure of each component in detail. This may involve creating detailed diagrams or models that show how the component interacts with other components, as well as any input and output data.
- **Interface design:** Components typically interact with one another through well-defined interfaces, so the next step in component-level design is to design the interfaces between components. This may involve specifying the types of data that are passed between components, as well as the format and structure of that data.
- **Component testing**: Once the components have been designed, the next step is to test them to ensure that they meet the requirements of the system. This may involve unit testing individual components in isolation, as well as integration testing to ensure that the components work together properly.

The benefits of good component-level design include improved system maintainability, increased code reuse, and better overall system performance.

5. Architectural Design

Architectural design is the process of defining the overall structure and organization of a software system. It involves identifying the key components of the system, their relationships, and the patterns of interaction between them. The goal of architectural design is to create a software system that meets its functional and non-functional requirements, is easy to understand, and is scalable and maintainable over time.

The process of architectural design typically involves the following steps:

stakeholders to identify the key functional and non-functional requirements of the system, as well as any constraints that must be considered.

- **Step 2: System decomposition**: Once the requirements have been analyzed, the next step is to decompose the system into its key components. This may involve breaking down the system into modules, subsystems, and other functional components.
- **Step 3: Component interaction:** With the system decomposed, the next step is to define the patterns of interaction between the components. This may involve specifying the types of data that are passed between components, as well as the protocols and interfaces that are used to facilitate that communication.
- **Step 4: System deployment**: Once the components and their interactions have been defined, the next step is to determine how the system will be deployed in the real world. This may involve considerations such as hardware and software platforms, network topology, and system performance requirements.
- **Step 5: Evaluation**: Finally, the architectural design must be evaluated to ensure that it meets the requirements of the system. This may involve using formal methods to analyze the design for correctness, as well as testing and simulation to verify that the system behaves as expected.

The benefits of good architectural design include improved system performance, scalability, and maintainability. By defining the overall structure and organization of the system, developers can create software that is easier to understand and modify over time, reducing the cost and effort required for maintenance and updates.

FAQs for Different types of Low Level Design in System Design

Q 1. Why is Low Level Design important in system design?

optimizing performance, and facilitating easier system maintenance and scalability.

Q 2. Why is the choice of data structure important?

The choice of data structure impacts the performance, memory usage, and ease of use of the system, making it critical to select the appropriate structure for specific operations.

Q 3. How does algorithm design affect system performance?

Efficient algorithm design can significantly improve system performance and scalability by optimizing the way tasks are executed.

Q 4. Why is architectural design important?

Architectural design ensures that the system is scalable, maintainable, and efficient, meeting all functional and non-functional requirements.

Q 5. How do you integrate different components in LLD?

Integration involves defining clear interfaces and interaction patterns between components, ensuring that they work together seamlessly.

Want to be a Software Architect or grow as a working professional? Knowing both Low and High Level System Design is highly necessary. As such, our

Real-World Examples. Learn how to come up with systems that are scalable, efficient, and robust—ones that impress. Ready to elevate your tech skills? Enrol now and build the future!

Comment More info

Next Article

Similar Reads

Different types of Low Level Design in System Design

Low-level design refers to the process of defining the detailed, functional design of a software system or component. It involves specifying the individual modules, data structures, algorithms, interfaces, and...

9 min read

What is Low Level Design or LLD - Learn System Design

Low-Level Design (LLD) is the detailed design process in the software development process that focuses on the implementation of individual components described in the High-Level Design. It provides a blueprint for...

5 min read

What is High Level Design – Learn System Design

In Developing scalable applications, proper planning, and organization play a significant role. High-level design plays an important role in this process by serving as the blueprint of the system's architecture. It...

8 min read

Design Principles in System Design

Design Principles in System Design are a set of considerations that form the basis of any good System. But the question now arises why use Design Principles in System Design? Design Principles help teams with...

5 min read

5 Tips to Crack Low-Level System Design Interviews

Cracking low-level system design interviews can be challenging, but with the right approach, you can master them. This article provides five essential tips to help you succeed. These tips will guide you through the...

6 min read

What is Systems Design - Learn System Design

System Design vs. Software Design

System Design and Software Design are two important concepts in the creation of robust and effective technological solutions. While often used interchangeably, they represent distinct disciplines with unique...

8 min read

Design Principles for System Design in Go

In this article, we will discover essential design principles for efficient system architecture in Go programming. Learn how to optimize concurrency, leverage interfaces, and manage errors effectively, ensuring robust and...

8 min read

System Analysis and Design Interview Topics for Freshers

System Analysis is the "what" before the "how" in system design. It provides the essential roadmap for crafting a system that is both effective and efficient in solving the intended problem. "Imagine you're building...

15+ min read

Guide to System Design Interview for Senior Engineers

Preparing for a system design interview as a senior engineer requires a deep understanding of architectural principles, scalability, and complex problem-solving skills. This guide aims to equip you with the essential...

12 min read

Elevator System Low-Level Design (LLD)

Elevator System is one of the famous Low-Level Design(LLD) problems being asked in interviews. We are here going to discuss this problem along with its requirements, use-cases diagram, class diagram and...

8 min read

Why OOPS is Important in System Design?

Object-oriented programming (OOP) plays a crucial role in system design due to its ability to organize complex systems into manageable units. This approach promotes modularity, reusability, and encapsulation,...

2 min read

High Latency vs Low Latency | System Design

In system design, latency refers to the time it takes for data to travel from one point in the system to another and back, essentially measuring the delay or lag within a system. It's a crucial metric for evaluating the...

4 min read

What is Low-Level Design Document?

Types of System Design

System design is a crucial process that involves defining the architecture, components, modules, interfaces, and data for a system to satisfy specified requirements. It plays a pivotal role in developing complex softwar...

7 min read

System Design Life Cycle | SDLC (Design)

System Design Life Cycle is defined as the complete journey of a System from planning to deployment. The System Design Life Cycle is divided into 7 Phases or Stages, which are: 1. Planning Stage 2. Feasibility Study...

6 min read

Object-Oriented Design (OOD) - System Design

A crucial method for system design is object-oriented design (OOD), which places an intense focus on scalability, modularity, and reusability. OOD resembles real-world systems by encapsulating data and...

12 min read

Difference between High Level Design(HLD) and Low Level Design(LLD)

System design involves creating both a High-Level Design (HLD), which is like a roadmap showing the overall plan, and a Low-Level Design (LLD), which is a detailed guide for programmers on how to build each part. It...

4 min read

Analysis of Monolithic and Distributed Systems - Learn System Design

System analysis is the process of gathering the requirements of the system prior to the designing system in order to study the design of our system better so as to decompose the components to work efficiently so tha...

10 min read

Article Tags : System Design