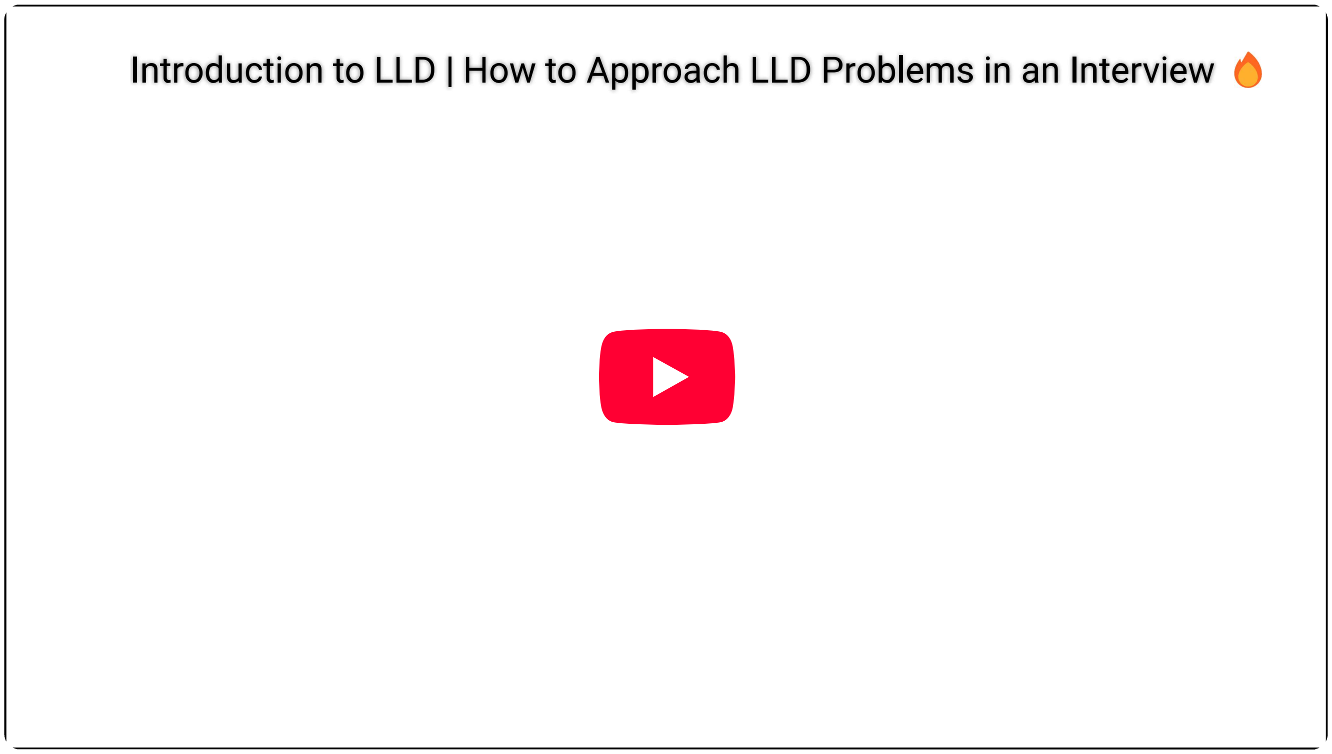
What is Low level System Design ?


Introduction to LLD | How to Approach LLD Problems in an Interview 🔥

Topic Tags:

system design LLD

Low-Level Design (LLD) is a detailed phase in the software development process that focuses on designing the individual components outlined in the **High-Level Design (HLD)**. LLD delves into the specifics, defining how modules, classes, functions, and data structures interact to achieve the desired functionality. 🛠️

For example, in a movie ticket booking system, LLD would detail how components like seat selection, payment processing, and ticket generation are implemented and interact with one another. 🎥 🎟️ 💳

Below is a class diagram illustrating the essential components of a movie ticket booking system, which we will reference to understand how LLD principles are applied in practice. 📊

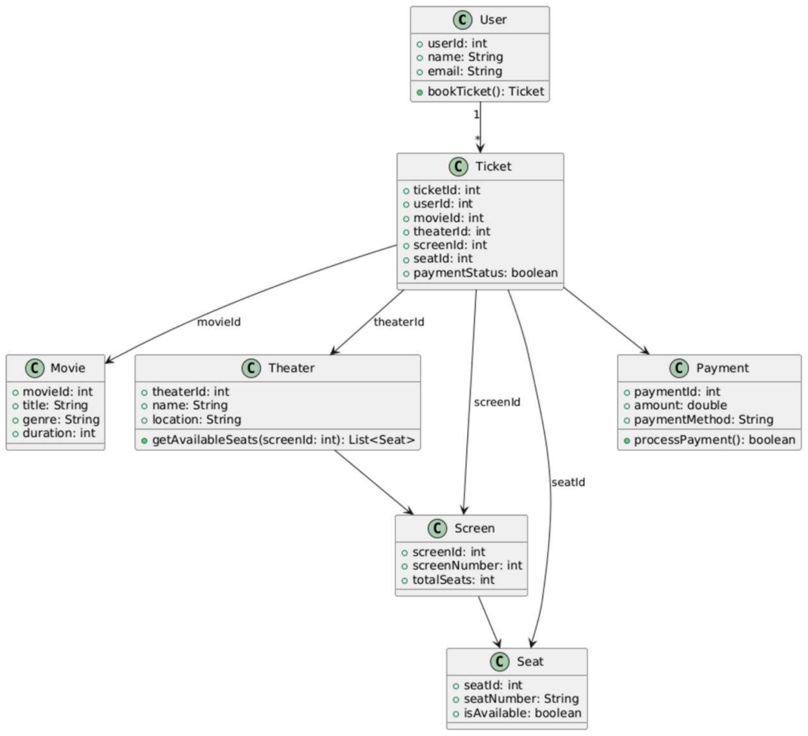


fig 1: Class Diagram

🤔 How is LLD Different From HLD?

Low-Level Design (LLD) and High-Level Design (HLD) are two key stages in software development, focusing on different levels of detail. Let's take an example of a movie ticket booking system to understand their roles. 🎥 🎟️

fig 2: Movie Ticket Booking System

High-Level Design (HLD) focuses on the overall architecture of the system. For instance, in a movie ticket booking system, HLD would outline the main components like the user interface (where users select movies and seats), the backend services (handling booking

requests, seat availability, and notifications), and the database (storing movie schedules, user data, and bookings). 🖥️ 🎟️
It would also define how these components interact—like the flow of data between the user interface, backend, and third-party payment gateways. 🔄 💳

Below is a high-level diagram (HLD) showcasing the overall architecture of a movie ticket booking system, which we will reference to understand how HLD principles are applied in designing scalable and robust systems. 📊

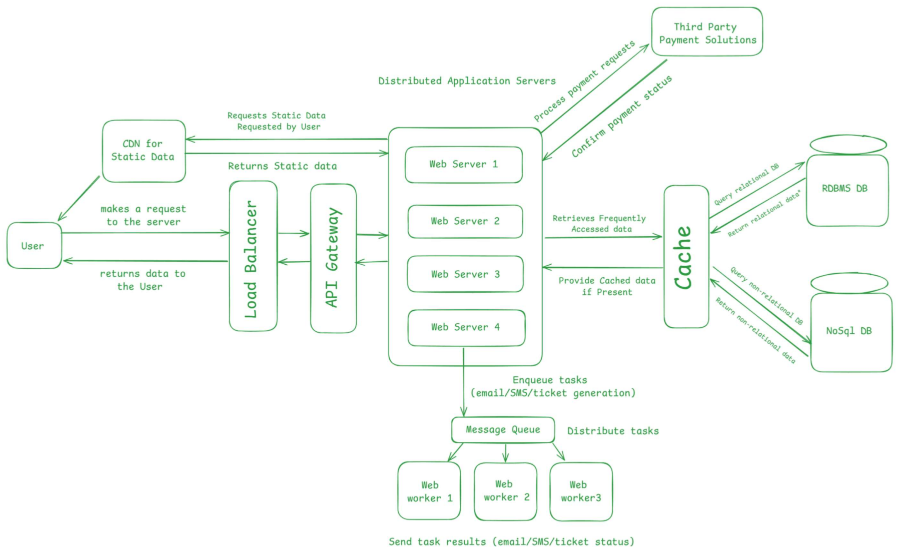


fig 3: High Level Design Diagram
(Not the detailed diagram, just an overview to differentiate between LLD and HLD.)

Low-Level Design (LLD), on the other hand, dives into the specifics of implementing individual features. For example, it defines how the booking process works—detailing the step-by-step flow from when a user selects a movie and showtime to when a ticket is successfully booked. 🎬 📅

It specifies how data is validated (e.g., ensuring selected seats are available and payment details are correct), algorithms for locking seats (to prevent double booking), and how the booking information is stored in the database (schema). 💾 🔒

LLD also describes the flow of data, such as how a booking confirmation is generated and sent to the user via email or SMS. 📧 📱 It's like creating blueprints for each transaction in the system, covering the smallest details to ensure reliability and precision. 🎯

While HLD sets the vision for the system, LLD ensures every feature, like booking tickets, is implemented accurately and aligns with the high-level plan. 🧩
Below is an activity diagram depicting the workflow of a movie ticket booking system, which we will reference to understand how the process flows and how individual actions contribute to achieving the desired functionality of making a movie booking. 📊

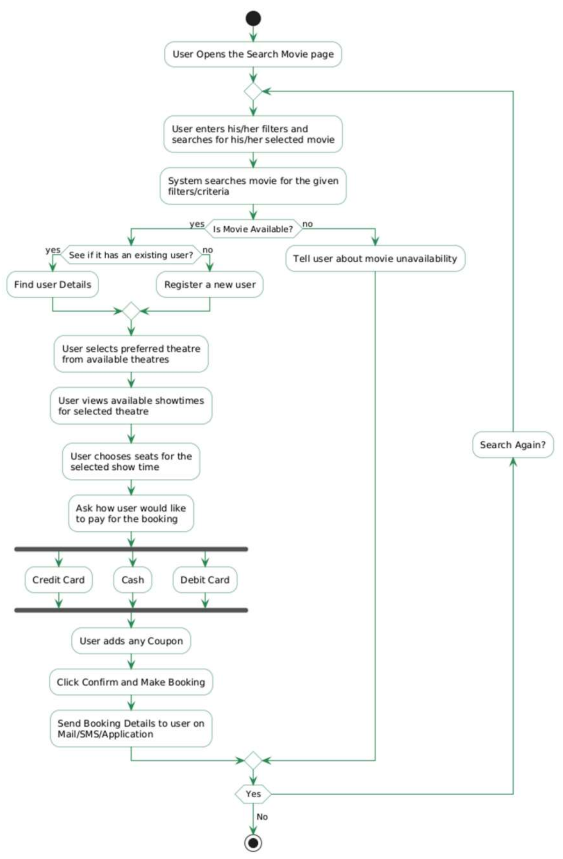

User Opens the Search Movie page
User enters his/her filters and searches for his/her selected movie
System searches movie for the given filters/criteria
yes
Is Movie Available?
no
yes
See if it has an existing user?
no
Tell user about movie unavailability
Find user Details
Register a new user
User selects preferred theatre from available theatres
User views available showtimes for selected theatre
User chooses seats for the selected show time
Ask how user would like to pay for the booking
Credit Card
Cash
Debit Card
User adds any Coupon
Click Confirm and Make Booking
Send Booking Details to user on Mail/SMS/Application
Search Again?
Yes
No

fig 4: Movie Ticket Booking Activity Diagram
(Not the detailed diagram, just an overview to differentiate between LLD and HLD.)

🧩 Building Blocks of LLD

1. Requirement Gathering:

Understand the detailed requirements of the system. This includes identifying the functionality, constraints, and edge cases to ensure the design meets user needs. For example, in a movie ticket booking system, this would include functionality such as: 🎥 🎟️

- ○ Seat selection 🪑
- ○ Payment processing 💳
- ○ Ticket generation 🧾
- ○ Simultaneous bookings for the same seat 🔄

2. Laying Down Use Cases:

Define specific scenarios that the system will handle, outlining inputs, actions, and expected outputs. Use cases help in clarifying the scope and guiding the design process. For a movie ticket booking system, use cases would include: 🎬

- ○ Booking tickets for a single user 👤
- ○ Handling group bookings with seat proximity 👥
- ○ Canceling a booking and refunding payment 💵
- ○ Sending notifications (SMS/email) for booking confirmation 📧 📱

3. UML Diagrams:
Create diagrams to visually represent the structure, behavior, and interactions of system components. In a movie ticket booking system, these could include: 📊
○ **Class Diagram:** Representing entities like User, Ticket, Movie, Payment, and Theater 🎟️ 🎬 💳 🏛️
○ **Sequence Diagram:** Showing the flow of actions for booking a ticket, including seat selection, payment processing, and confirmation 🔄 💳
○ **Activity Diagram:** Mapping the workflow for booking and canceling tickets 🔁 📅

Below is a use case diagram representing the key interactions within a movie ticket booking system, which we will reference to understand how various actors and use cases collaborate to fulfill the system's objectives. 🎥 🎬

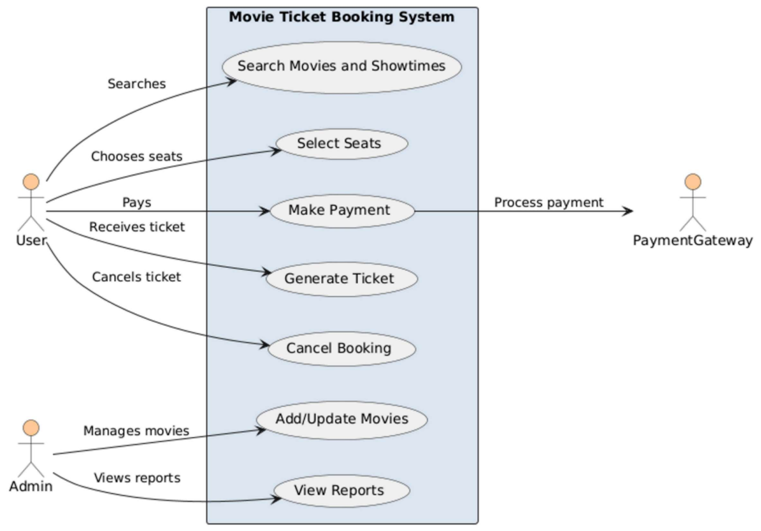


fig 5: Use-case Diagram

🛠️ Model Problems:

This phase focuses on solving specific problems identified during the use case analysis. It involves breaking down the system into smaller components and addressing individual design challenges. 💡

Design patterns play a crucial role in this phase, offering reusable solutions to common design problems. Some of the most famous Design Patterns include:
• Factory Design Pattern 🏭
• Strategy Design Pattern 🧠
• Observer Design Pattern 👀
• Singleton Design Pattern 🔑

💻 Implement Code:

Translate the design into clean, modular, and efficient code, adhering to coding standards and principles such as SOLID and DRY to ensure maintainability and scalability. ⚙️ 📏

🎯 Conclusion:

In conclusion, Low-Level Design (LLD) is an important phase in the software development lifecycle that translates high-level architectural concepts into detailed specifications for individual components. By focusing on modularity, clear interface definitions, and following the design principles, LLD ensures that software systems are robust, maintainable, and scalable. 🚀 📈