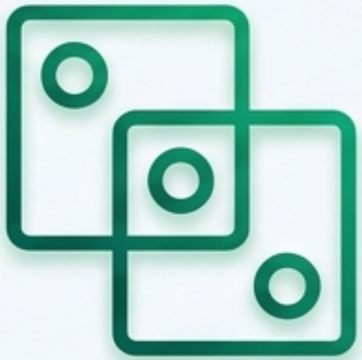
Cracking the Core: Your Ultimate CS Interview Guide

A Crash Course on Databases, OOPs, and Networks



The Three Pillars of the Core CS Interview

MODULE 1: DATA

Mastering Database Management Systems (DBMS)

Understand how to store, manage, and query data reliably and efficiently. The bedrock of any application.

MODULE 2: LOGIC

Thinking in Objects with OOPs

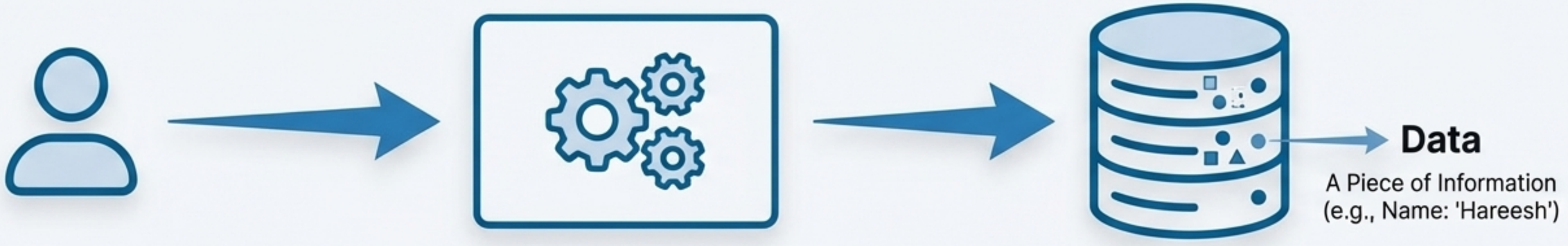
Learn the paradigm for structuring scalable and maintainable code using classes and objects. The blueprint for software architecture.

MODULE 3: CONNECTION

Understanding Computer Networks

Discover how systems communicate, from a single web request to large-scale distributed systems. The link to the outside world.

The Bedrock: Every Great Application is Built on Data



User
Inter

DBMS (Database Management System)
The Software (e.g., MySQL, PostgreSQL, MongoDB)

Database
The Organized Collection

SQL (RDBMS)

Structured data, organized in tables with rows and columns. Stands for Relational Database Management System.

NoSQL

Unstructured or semi-structured data, often stored in document, key-value, or graph formats. Flexible and scalable.

The ACID Test: The Non-Negotiable Contract of Transactions

ACID properties guarantee that database transactions are processed reliably. Think of booking a movie ticket.

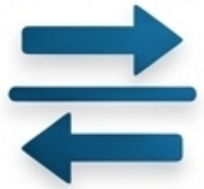
A Atomicity (All or Nothing)

Your entire transaction succeeds or fails as a single unit. You book 10 tickets, and either all 10 are confirmed and paid for, or none are. No partial bookings.

C Consistency (State Remains Valid)

The database remains in a valid, uncorrupted state before and after the transaction. The **number of available seats is correctly updated**, and no rules are violated.

I Isolation (No Interference)



Concurrent transactions do not affect each other. Your process of booking a seat is **isolated** from someone else booking the same seat at the exact same moment.

D Durability (Permanent Changes)

Once a transaction is committed, it is **permanent** and will survive any subsequent system failure. Once your booking is confirmed, it's **saved forever**, even if the server crashes.

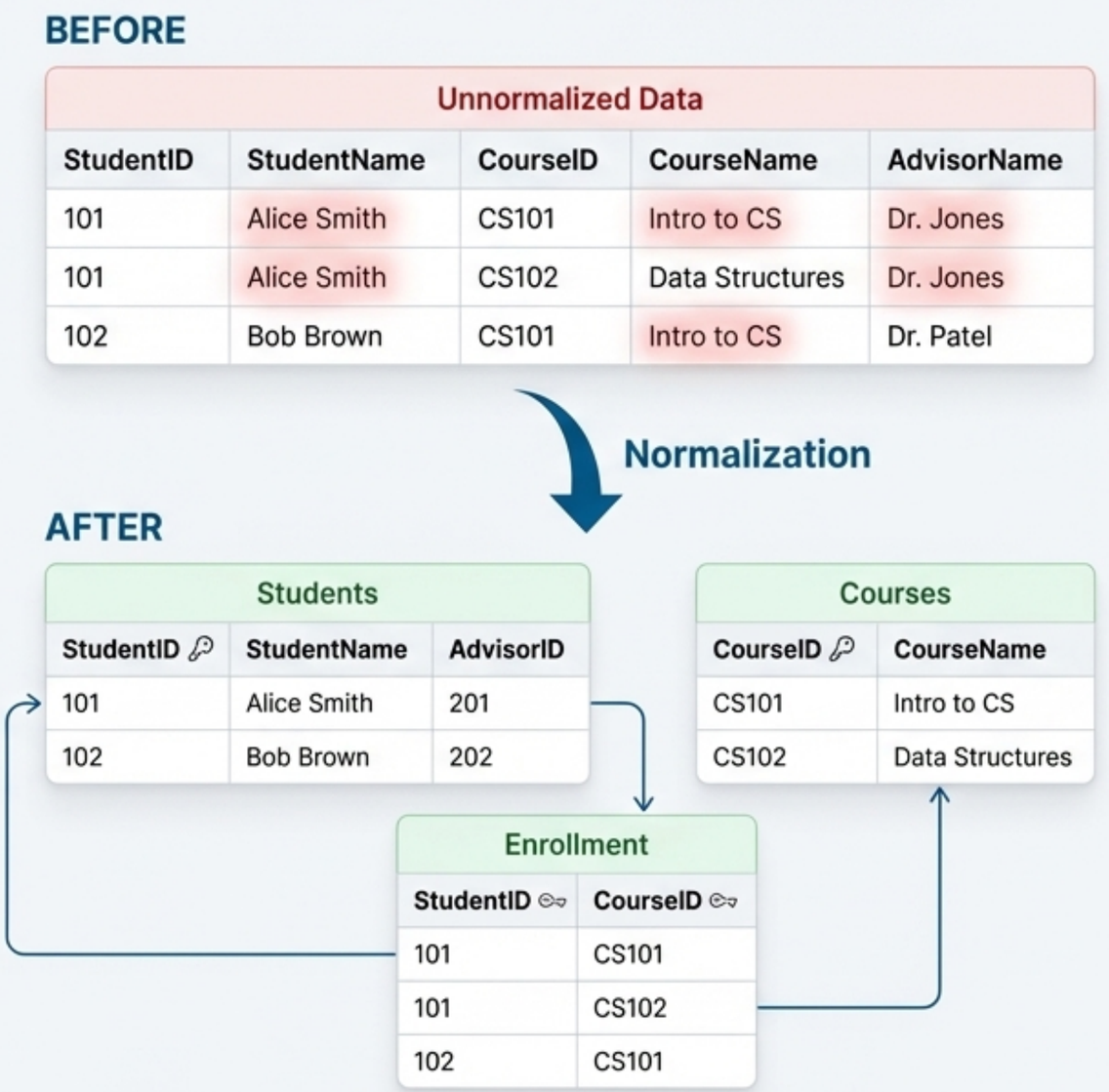
From Chaos to Order: Organizing Your Data

The Keys to Relationships

- **Primary Key:** A unique identifier for each row in a table (e.g., `UserID`). Essential for fast lookups.
- **Foreign Key:** A key that links a row in one table to a row in another, creating a relationship.
- **Candidate Key:** Any column or set of columns that can qualify as a primary key (e.g., `Email`).

The Goal of Normalization

- **Definition:** The process of organizing tables to reduce data redundancy and improve data integrity.
- **The Forms:**
  - **1NF:** Ensures all column values are atomic (indivisible).
  - **2NF:** Removes partial dependencies. All non-key attributes must depend on the *entire* primary key.
  - **3NF:** Removes transitive dependencies. Non-key attributes cannot depend on other non-key attributes.



The Language of Data: SQL Essentials

Core Commands (CRUD Operations)

SELECT: Retrieve data.

```
SELECT username, country
FROM Users
WHERE age > 18;
```

INSERT: Add new data.

```
INSERT INTO Products
(ProductID, ProductName)
VALUES (104, 'Mouse');
```

UPDATE: Modify existing data.

```
UPDATE Products
SET Price = 750
WHERE ProductName = 'Phone';
```

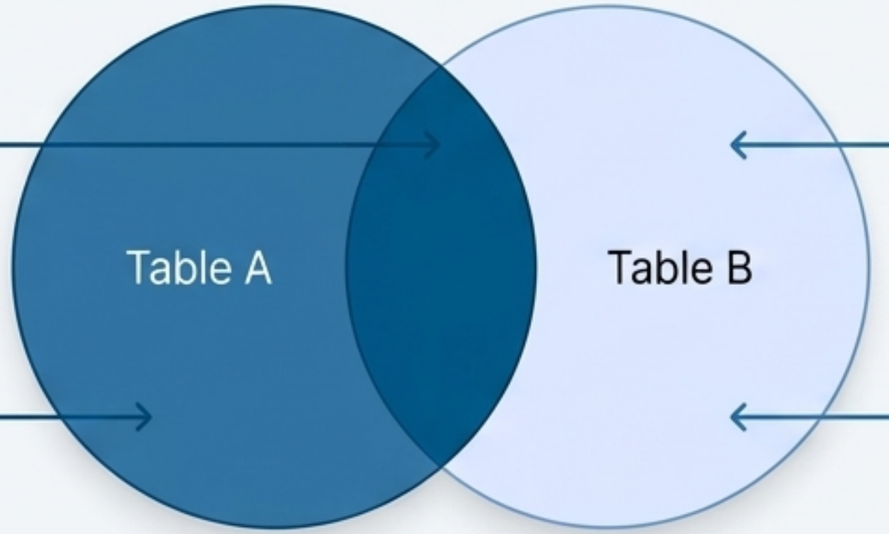
DELETE: Remove data.

```
DELETE FROM Orders
WHERE OrderID = 1003;
```

Connecting Tables with JOINS

INNER JOIN
Returns only the rows where the join condition is met in *both* tables.

LEFT JOIN
Returns *all* rows from the left table, and the matched rows from the right table.



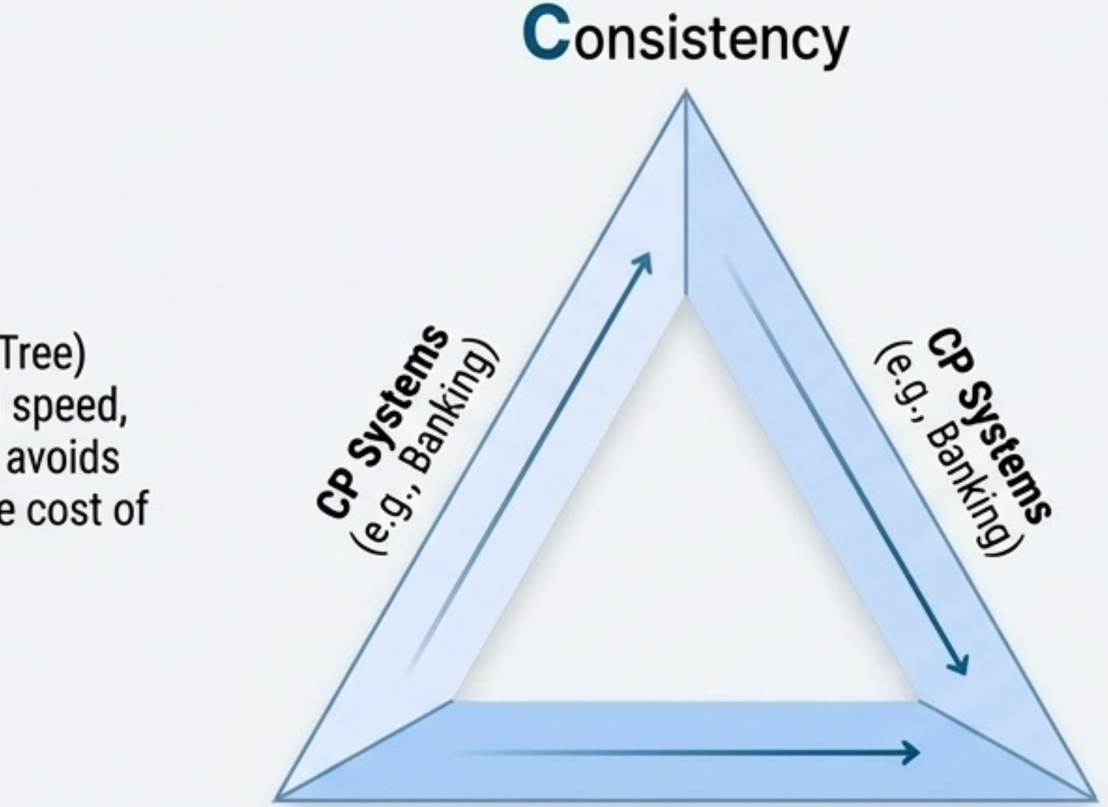
RIGHT JOIN
Returns *all* rows from the right table, and the matched rows from the left table.

FULL OUTER JOIN
Returns all rows when there is a match in either the left or the right table.

Beyond the Basics: Performance and Distributed Systems

Indexing

A data structure (often a B-Tree) that improves data retrieval speed, much like a book's index. It avoids costly full-table scans at the cost of slower writes.



Concurrency Control

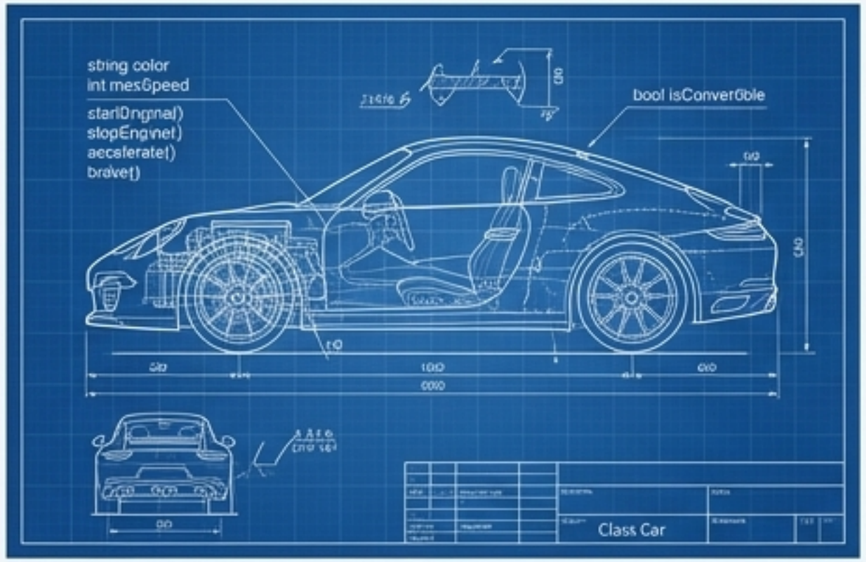
Mechanisms, such as locking (Pessimistic vs. Optimistic), that prevent data inconsistencies when multiple transactions occur simultaneously, avoiding race conditions.

A fundamental principle for distributed systems stating it is impossible to simultaneously provide more than two of these three guarantees.

The Blueprint: Structuring Code with Objects

A programming paradigm that organizes software design around "objects," which bundle together data (attributes) and behavior (methods).

Class (The Blueprint)



A template for creating objects. It defines a set of properties and methods common to all objects of one type. Example: `class Car`

new

Object (The Instance)

A specific instance of a class, created from the blueprint. You can create many unique objects from a single class. Examples: `redCar = new Car()`, `blueCar = new Car()`

The Four Pillars of Object-Oriented Design

Encapsulation (Data Hiding)

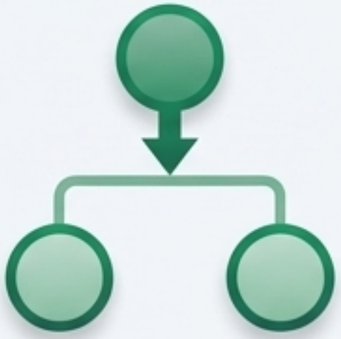
Bundles data and methods into a single unit (class), protecting it from outside interference.

Analogy: A protective phone case. It hides the complex internal circuits while providing simple buttons to interact.

Abstraction (Complexity Hiding)

Exposes only the essential features of an object while hiding the complex implementation details.

Analogy: A TV remote. You use the buttons without needing to know how the underlying electronics work.



Inheritance (Code Reusability)

Allows a new class (child) to acquire attributes and methods from an existing class (parent).

Analogy: Inheriting physical traits like eye color from your parents.

Polymorphism (Many Forms)

The ability of an object or method to take on many forms, allowing a single interface to represent different data types.

Analogy: How you respond differently to "Tell me about your day" when asked by your boss versus your best friend.

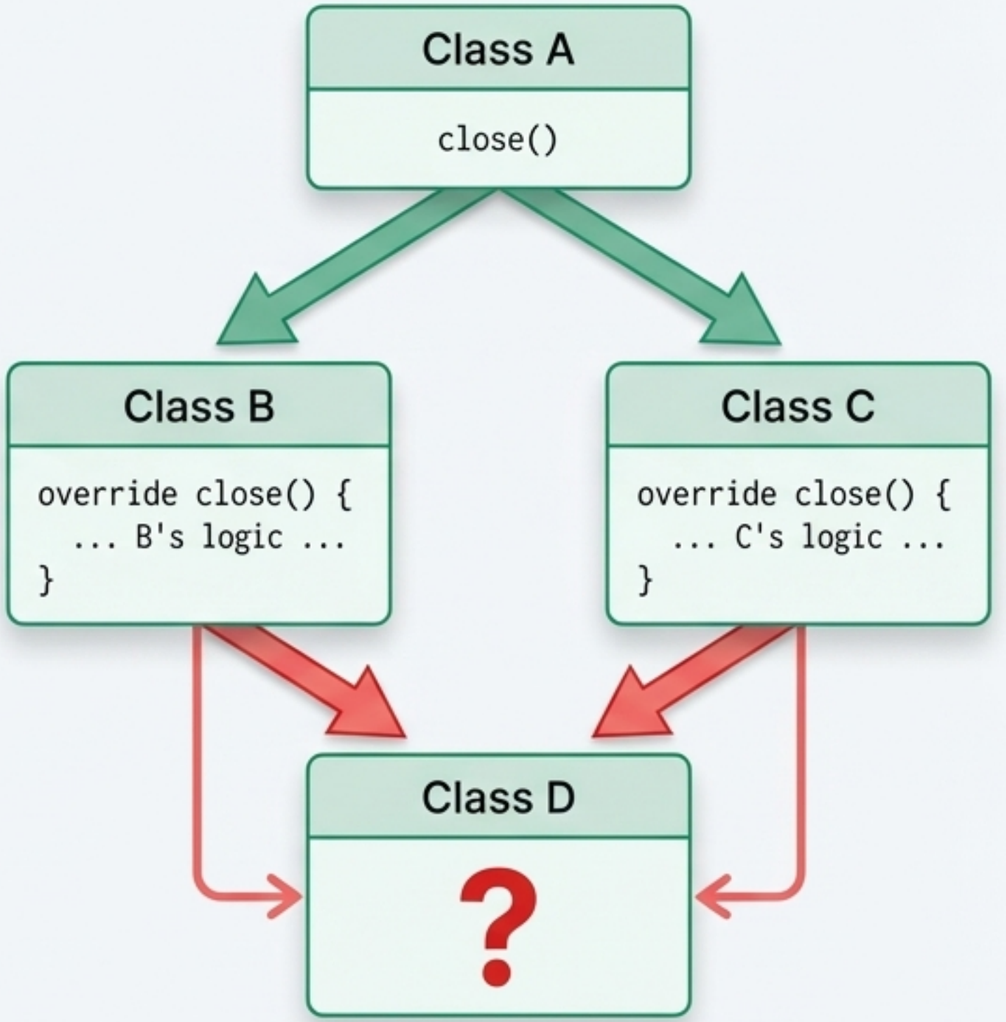
Code Reusability and its Classic Challenge

Inheritance in Practice

- **Single Inheritance:** A class inherits from only one parent class. (e.g., `Dog` extends `Animal`).
- **Multi-level Inheritance:** A class inherits from a child class, creating a "grandchild" relationship (e.g., `Poodle` extends `Dog`).

The Diamond Problem

- **Scenario:** Arises in languages with multiple inheritance. A class `D` inherits from two classes, `B` and `C`, which both inherit from a common ancestor, `A`.
- **The Conflict:** If a method in `A` is overridden by both `B` and `C`, which version should `D` inherit? This creates ambiguity.
- **Common Solution:** Languages like Java prevent this by disallowing multiple inheritance of classes, favoring Interfaces to achieve similar outcomes.



Defining Contracts: Abstract Class vs. Interface

Feature	Abstract Class	Interface
Methods	Can contain both **abstract** (unimplemented) and **concrete** (implemented) methods.	Can only contain **abstract** method signatures. It is a pure contract.
Inheritance Keyword	A class uses **extends** to inherit from an abstract class.	A class uses **implements** to conform to an interface.
Multiple Inheritance	A class can extend only **one** abstract class.	A class can implement **multiple** interfaces.
Purpose	To provide a common base with shared functionality that subclasses can use or override.	To define a contract of capabilities that a class must provide, regardless of its hierarchy.
Analogy	A partially completed blueprint with some sections detailed and others left open.	A strict job description listing all required duties without implementation details.

The Connection: How Systems Communicate

Core Concept:

Computer networking is the set of protocols (rules) and hardware that allow two or more computing devices to exchange data and share resources.

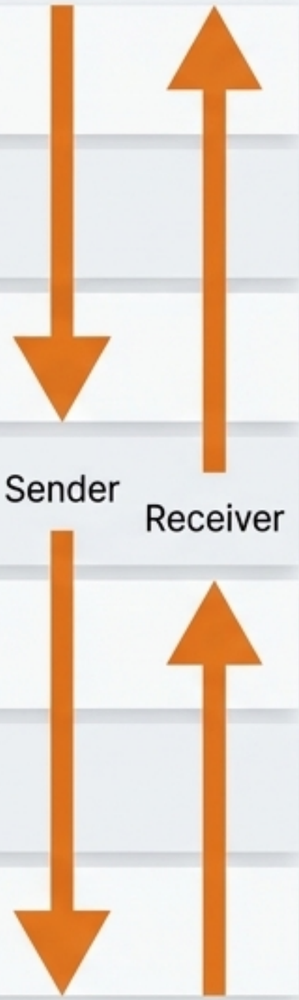
Why It Matters for a Software Engineer:

- It's the foundation of all distributed systems, web applications, and microservices.
- Understanding it helps you diagnose bottlenecks, ensure security, and design resilient applications.
- It's the bridge between the code you write (Logic) and the data it uses (Databases).

The OSI Model: A 7-Layer Framework for Communication

Note: A conceptual framework to understand and standardize network functions. Data is encapsulated as it goes down the stack and de-encapsulated as it goes up.

7	Application	Where network applications like web browsers operate. (Protocols: HTTP, FTP)
6	Presentation	Handles data translation, encryption, and compression.
5	Session	Manages communication sessions between devices.
4	Transport	Provides reliable end-to-end data transfer. (Protocols: TCP, UDP)
3	Network	Handles logical addressing (IP addresses) and routing. (Hardware: Routers)
2	Data Link	Manages physical addressing (MAC addresses) and framing. (Hardware: Switches)
1	Physical	The physical hardware transmitting raw bits over a medium. (Hardware: Cables, Hubs)



The Language of the Internet: Core Protocols

TCP/IP (Transmission Control Protocol / Internet Protocol)

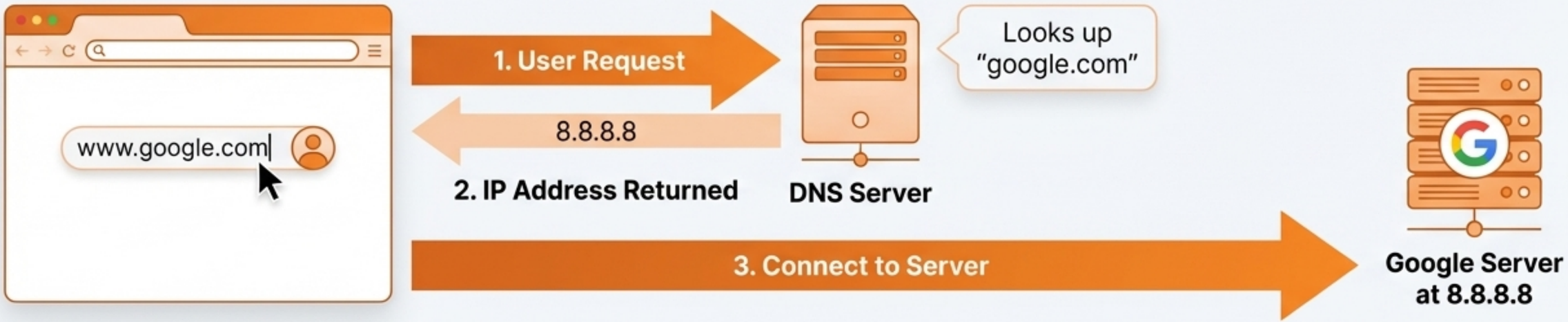
The foundational suite of the internet. **IP** handles addressing and routing packets.
TCP ensures reliable, ordered delivery.

DNS (Domain Name System)

The internet's "phonebook." Translates human-friendly domain names (e.g., `www.google.com`) into machine-readable IP addresses (e.g., `8.8.8.8`).

DHCP (Dynamic Host Configuration Protocol)

Automatically assigns a unique IP address to devices when they connect to a network, preventing manual configuration.



Your Core CS Interview Checklist

Skill: “Explain core concepts with real-world analogies.”
Can you explain ACID properties using the movie ticket example? Or Abstraction with a TV remote?

Skill: “Be ready to write basic SQL queries.”
Can you combine data from a `Users` table and an `Orders` table using an `INNER JOIN`?

Skill: “Design a simple system using OOP principles.”
How would you use classes like `User`, `Product`, and `ShoppingCart`? Where would you use an Interface vs. an Abstract Class?

Skill: “Trace the path of a web request.”
Can you walk through what happens when a user types a URL and hits enter, mentioning DNS, TCP/IP, and HTTP?

You’ve got this. These fundamentals are the toolkit for building great software. Show them you know how to use them. All the best!