𝙅𝒂𝙫𝒂 𝑰𝙣𝒕𝙚𝒓𝙫𝒊𝙚𝒘 𝑷𝙧𝒆𝙥𝒂𝙧𝒂𝙩𝒊𝙤𝒏 - 2  
  
🚀 Master most commonly used Stream Operations in Java 8+:   
filter(), map(), and flatMap() with Real-Project Examples! 🚀   
  
It is a Part 3 of video series on Stream APIs. The attached document 📄 and below video 📷 will provide in-depth knowledge of the topic:  
🔗 <https://lnkd.in/djaD32WD>  
  
📌 Demo Scenarios / Examples Covered:  
✅ Get employees whose name contains a given sub-string (filter, collect)  
✅ Get employees living in a given city (filter, collect)  
✅ Get employee DTOs having a given skill (filter, map, collect)  
✅ Get employee DTOs in a given department (filter, map, collect)  
✅ Get skill-capacity for a given department (filter, flatMap, distinct, collect)  
  
📌 What you'll learn:  
✅ How to efficiently filter data using filter()  
✅ Transform data like a pro with map()  
✅ Flatten complex structures using flatMap()  
✅ Real-world coding examples that every Java developer should know!  
  
Do not forget to watch these videos on streams:  
🎉 What are Streams? - Part 1: <https://lnkd.in/dK2ZGJAP>  
  
🎉 Creating Streams in Java 8+ - Part 2: <https://lnkd.in/grFBjVt7>  
  
  
𝐏𝐥𝐚𝐲𝐥𝐢𝐬𝐭 𝐨𝐧 𝐈𝐧𝐭𝐞𝐫𝐯𝐢𝐞𝐰 𝐏𝐫𝐞𝐩𝐚𝐫𝐚𝐭𝐢𝐨𝐧: <https://lnkd.in/dP6a44bS>  
  
𝑼𝒏𝒅𝒆𝒓𝒔𝒕𝒂𝒏𝒅𝒊𝒏𝒈 𝒍𝒂𝒎𝒃𝒅𝒂 𝒆𝒙𝒑𝒓𝒆𝒔𝒔𝒊𝒐𝒏𝒔 𝒂𝒏𝒅 𝒇𝒖𝒏𝒄𝒕𝒊𝒐𝒏𝒂𝒍 𝒊𝒏𝒕𝒆𝒓𝒇𝒂𝒄𝒆𝒔 𝒊𝒔 𝒌𝒆𝒚 𝒇𝒐𝒓 𝒖𝒔𝒊𝒏𝒈 Streams 𝒆𝒇𝒇𝒆𝒄𝒕𝒊𝒗𝒆𝒍𝒚.  
  
If you want to improve your knowledge of lambda expressions and functional interfaces, watch the videos below:  
  
🎉 𝐏𝐚𝐫𝐭 𝟏 - <https://lnkd.in/d35-sQCc>  
  
🎉 𝐏𝐚𝐫𝐭 𝟐 - <https://lnkd.in/dJvEQ_h7>  
  
🎉 𝐏𝐚𝐫𝐭 𝟑 - <https://lnkd.in/dpFdPagT>  
  
If you're working with Java Streams or preparing for interviews, you don’t want to miss this! Let me know your thoughts in the comments. 😊

<https://www.linkedin.com/feed/update/urn:li:activity:7292933694397243393?utm_source=share&utm_medium=member_desktop>

<https://www.linkedin.com/posts/naveen-chandrawanshi-86b976220_dsa-terms-ugcPost-7293096117070655490-Srjr?utm_source=share&utm_medium=member_desktop>

Tech Terms That Confuse Almost Everyone 🤯 (Explained Simply)  
  
Ever found yourself stuck between subarrays and subsequences? Or wondering if a DAG is some cryptic coding jargon? You’re not alone!  
  
Here are some of the most confusing terms in Data Structures & Algorithms—explained in plain English. 👇  
  
1️⃣ Strings: Substring vs. Subsequence  
🔹 Substring – A contiguous part of a string (must be in order).  
Example: "abc" is a substring of "abcdef" but "ace" is not.  
🔹 Subsequence – A sequence derived from the string by deleting characters without changing order.  
Example: "ace" is a subsequence of "abcdef", but "cab" is not.  
  
2️⃣ Arrays: Subarray vs. Subsequence vs. Subset  
  
🔹 Subarray – A contiguous portion of an array.  
Example: [2, 3] is a subarray of [1, 2, 3, 4], but [1, 3] is not.  
  
🔹 Subsequence – Elements in the same order but not necessarily contiguous.  
Example: [1, 3] is a subsequence of [1, 2, 3, 4].  
  
🔹 Subset – Any possible selection of elements (order doesn’t matter).  
Example: {2, 4} is a subset of {1, 2, 3, 4}, but {5} is not.  
  
3️⃣ Linked List: Singly vs. Doubly vs. Circular  
  
🔹 Singly Linked List – Each node points to the next node only.  
🔹 Doubly Linked List – Nodes have next and previous pointers.  
🔹 Circular Linked List – The last node points back to the first node, forming a loop.  
  
4️⃣ Trees: Binary Tree vs. BST vs. Balanced Tree  
🔹 Binary Tree – Each node has at most 2 children.  
🔹 Binary Search Tree (BST) – A binary tree where left < root < right.  
🔹 Balanced Tree – A tree where height is O(log n) (AVL, Red-Black Trees).  
  
5️⃣ Graphs: Directed vs. Undirected vs. DAG vs. Bidirectional  
🔹 Directed Graph (Digraph) – Edges have a direction (A ➝ B).  
🔹 Undirected Graph – Edges don’t have direction (A — B).  
🔹 DAG (Directed Acyclic Graph) – A directed graph with no cycles (used in scheduling & dependencies).  
🔹 Bidirectional Graph – A directed graph where edges go both ways (A ⇄ B).  
  
Pro Tip:  
If a term confuses you, break it down with examples it makes learning 10x easier!   
  
What’s a tech term that confused you at first? Let me know in the comments!   
  
repost this with your network ♻️   
Follow Naveen chandrawanshi ✅

𝗨𝗻𝗹𝗼𝗰𝗸 𝗛𝗶𝗴𝗵-𝗣𝗲𝗿𝗳𝗼𝗿𝗺𝗮𝗻𝗰𝗲 𝗖𝗮𝗰𝗵𝗶𝗻𝗴 𝘄𝗶𝘁𝗵 𝗝𝗲𝗱𝗶𝘀 𝗶𝗻 𝗦𝗽𝗿𝗶𝗻𝗴 𝗕𝗼𝗼𝘁 🚀  
  
Caching plays a critical role in building high-performance, scalable applications. Among the many tools available, 𝗥𝗲𝗱𝗶𝘀, paired with the Java client 𝗝𝗲𝗱𝗶𝘀, stands out as a powerful solution for caching in Spring Boot applications. But why should you consider using Jedis, and how can it elevate your caching strategy?  
  
𝗪𝗵𝘆 𝗝𝗲𝗱𝗶𝘀 𝗳𝗼𝗿 𝗥𝗲𝗱𝗶𝘀 𝗖𝗮𝗰𝗵𝗶𝗻𝗴?  
Jedis is a lightweight and fast Java client for Redis, offering low-level access to Redis commands. When integrated with Spring Boot's caching abstraction, Jedis provides:  
- 𝗛𝗶𝗴𝗵 𝗣𝗲𝗿𝗳𝗼𝗿𝗺𝗮𝗻𝗰𝗲: Optimized for speed, making it ideal for real-time applications.  
- 𝗔𝗱𝘃𝗮𝗻𝗰𝗲𝗱 𝗙𝗲𝗮𝘁𝘂𝗿𝗲𝘀: Supports pipelining, clustering, and pub/sub messaging.  
- 𝗧𝗵𝗿𝗲𝗮𝗱 𝗦𝗮𝗳𝗲𝘁𝘆: Ensures robust handling of concurrent operations.  
- 𝗘𝗮𝘀𝗲 𝗼𝗳 𝗨𝘀𝗲: Seamlessly integrates with Spring Data Redis.  
  
𝗕𝗲𝘀𝘁 𝗣𝗿𝗮𝗰𝘁𝗶𝗰𝗲𝘀 𝗳𝗼𝗿 𝗨𝘀𝗶𝗻𝗴 𝗝𝗲𝗱𝗶𝘀 𝗶𝗻 𝗦𝗽𝗿𝗶𝗻𝗴 𝗕𝗼𝗼𝘁  
1. 𝗘𝗻𝗮𝗯𝗹𝗲 𝗖𝗮𝗰𝗵𝗶𝗻𝗴: Use `@EnableCaching` in your Spring Boot application to activate cache management.  
2. 𝗖𝗼𝗻𝗳𝗶𝗴𝘂𝗿𝗲 𝗥𝗲𝗱𝗶𝘀 𝗖𝗼𝗻𝗻𝗲𝗰𝘁𝗶𝗼𝗻: Define a `JedisConnectionFactory` bean to connect to your Redis instance.  
3. 𝗨𝘀𝗲 𝗖𝗮𝗰𝗵𝗲 𝗔𝗻𝗻𝗼𝘁𝗮𝘁𝗶𝗼𝗻𝘀: Leverage `@Cacheable`, `@CacheEvict`, and `@CachePut` to manage caching behavior effortlessly.  
4. 𝗢𝗽𝘁𝗶𝗺𝗶𝘇𝗲 𝗧𝗧𝗟 (𝗧𝗶𝗺𝗲-𝘁𝗼-𝗟𝗶𝘃𝗲): Set appropriate expiration times for cached data to avoid stale entries.  
5. 𝗠𝗼𝗻𝗶𝘁𝗼𝗿 𝗣𝗲𝗿𝗳𝗼𝗿𝗺𝗮𝗻𝗰𝗲: Use tools like Redis CLI or monitoring dashboards to track cache hits/misses and optimize accordingly.  
  
𝗘𝗻𝗴𝗮𝗴𝗲 𝘄𝗶𝘁𝗵 𝘁𝗵𝗲 𝗖𝗼𝗺𝗺𝘂𝗻𝗶𝘁𝘆  
What has been your experience with Redis and Jedis? Are there specific challenges you've faced or optimizations you've discovered? Share your thoughts and let's discuss how we can make our applications faster and more efficient!  
  
[hashtag#Java](https://www.linkedin.com/search/results/all/?keywords=%23java&origin=HASH_TAG_FROM_FEED) [hashtag#SpringBoot](https://www.linkedin.com/search/results/all/?keywords=%23springboot&origin=HASH_TAG_FROM_FEED) [hashtag#Redis](https://www.linkedin.com/search/results/all/?keywords=%23redis&origin=HASH_TAG_FROM_FEED) [hashtag#Jedis](https://www.linkedin.com/search/results/all/?keywords=%23jedis&origin=HASH_TAG_FROM_FEED) [hashtag#Caching](https://www.linkedin.com/search/results/all/?keywords=%23caching&origin=HASH_TAG_FROM_FEED) [hashtag#PerformanceOptimization](https://www.linkedin.com/search/results/all/?keywords=%23performanceoptimization&origin=HASH_TAG_FROM_FEED) [hashtag#BackendDevelopment](https://www.linkedin.com/search/results/all/?keywords=%23backenddevelopment&origin=HASH_TAG_FROM_FEED)

𝐄𝐱𝐜𝐥𝐮𝐬𝐢𝐯𝐞 𝐂𝐨𝐫𝐞 𝐉𝐚𝐯𝐚 𝐒𝐭𝐮𝐝𝐲 𝐌𝐚𝐭𝐞𝐫𝐢𝐚𝐥 𝐅𝐨𝐫 𝐁𝐞𝐠𝐢𝐧𝐧𝐞𝐫 𝐒𝐭𝐮𝐝𝐞𝐧𝐭𝐬 & 𝐏𝐫𝐨𝐟𝐞𝐬𝐬𝐢𝐨𝐧𝐚𝐥𝐬   
•••••••••••••••••••••••••••••••••••••••••••••••••••••  
Java is a programming language and computing platform that is used to create a variety of applications, including games, mobile apps, enterprise software, and cloud applications.  
  
Knowing fundamental of any programming language is Important so here's pdf to understand basic concept in java  
  
🕸️🕸️ [Allauddin Khan Pathan](https://www.linkedin.com/in/allauddin-khan-pathan/) 🕸️🕸️  
••••••••••••••••••••••••••••••••••••••••••••••••••••••  
👆🏻 𝑭𝒐𝒓 𝑴𝒐𝒓𝒆 𝑺𝒖𝒄𝒉 𝑪𝒐𝒏𝒕𝒆𝒏𝒕 𝑭𝒐𝒍𝒍𝒐𝒘 𝑴𝒆 👆🏻  
  
🔸🔸🔸🔸🔸𝐃𝐈𝐒𝐂𝐋𝐀𝐈𝐌𝐄𝐑 🔸🔸🔸🔸  
This Document is not my work or I'm not trying to frame it as Mine just sharing this to help Students or professional who are preparing for interview or exam.  
  
👨🏻‍🏫 Credit : 𝐃𝐮𝐫𝐠𝐚 𝐒𝐢𝐫 (Durga\_Soft\_Hyd)  
   
[hashtag#allauddin](https://www.linkedin.com/search/results/all/?keywords=%23allauddin&origin=HASH_TAG_FROM_FEED) [hashtag#java](https://www.linkedin.com/search/results/all/?keywords=%23java&origin=HASH_TAG_FROM_FEED) [hashtag#oops](https://www.linkedin.com/search/results/all/?keywords=%23oops&origin=HASH_TAG_FROM_FEED) [hashtag#corejava](https://www.linkedin.com/search/results/all/?keywords=%23corejava&origin=HASH_TAG_FROM_FEED) [hashtag#scala](https://www.linkedin.com/search/results/all/?keywords=%23scala&origin=HASH_TAG_FROM_FEED)   
•••••••••••••••••••••••••••••••••••••••••••••••••••••  
Java is a popular programming language for many reasons, including:  
  
Platform independence:  
Java code can run on any operating system without being rewritten. This makes Java a good choice for developing applications that can run on different servers and devices.   
   
Security:  
Java has security features that allow users to run untrusted code in a secure environment.   
   
Large community:  
Java has a large community of users and developers who can help each other with coding challenges.   
   
Learning resources:  
There are many resources available to help new programmers learn Java, including documentation, books, and courses.   
   
Built-in functions and libraries:  
Java provides a large library of functions and classes that developers can use to build applications.   
   
Development tools:  
Java offers tools for automated editing, debugging, testing, deployment, and change management.   
   
Multithreading:  
Java's multithreading feature allows programs to perform multiple tasks simultaneously. This is useful for heavy applications like games and animation.   
   
Object-oriented:  
Java is an object-oriented language that uses classes to build objects

<https://www.linkedin.com/feed/update/urn:li:activity:7292906024070463488?utm_source=share&utm_medium=member_desktop>

Git vs GitHub  
  
Git and GitHub are popular tools for version control. They work together and complement each other to provide effective source control management.  
  
On a high level, Git is focused on version control and code sharing, whereas GitHub is focused on centralized source code hosting for sharing with other developers.  
  
However, they have some key differences  
  
 1. Definition  
  
 • [hashtag#Git](https://www.linkedin.com/search/results/all/?keywords=%23git&origin=HASH_TAG_FROM_FEED): A distributed version control system (DVCS) used to track code changes.  
 • [hashtag#GitHub](https://www.linkedin.com/search/results/all/?keywords=%23github&origin=HASH_TAG_FROM_FEED): A cloud-based platform that hosts Git repositories and facilitates collaboration.  
  
 2. Purpose  
  
 • [hashtag#Git](https://www.linkedin.com/search/results/all/?keywords=%23git&origin=HASH_TAG_FROM_FEED): Helps manage code versions locally and enables branching, merging, and rollback.  
 • [hashtag#GitHub](https://www.linkedin.com/search/results/all/?keywords=%23github&origin=HASH_TAG_FROM_FEED): Provides remote storage, team collaboration, pull requests, and project management tools.  
  
 3. Installation & Access  
  
 • [hashtag#Git](https://www.linkedin.com/search/results/all/?keywords=%23git&origin=HASH_TAG_FROM_FEED): Needs to be installed on a local machine. Works via command-line or GUI.  
 • [hashtag#GitHub](https://www.linkedin.com/search/results/all/?keywords=%23github&origin=HASH_TAG_FROM_FEED): Web-based, accessible through a browser, Git CLI, or GitHub Desktop.  
  
 4. Repository Storage  
  
 • [hashtag#Git](https://www.linkedin.com/search/results/all/?keywords=%23git&origin=HASH_TAG_FROM_FEED): Repositories are stored locally on a computer.  
 • [hashtag#GitHub](https://www.linkedin.com/search/results/all/?keywords=%23github&origin=HASH_TAG_FROM_FEED): Repositories are stored in the cloud, allowing remote access.  
  
 5. Collaboration  
  
 • [hashtag#Git](https://www.linkedin.com/search/results/all/?keywords=%23git&origin=HASH_TAG_FROM_FEED): Collaboration requires manual setup of remote repositories.  
 • [hashtag#GitHub](https://www.linkedin.com/search/results/all/?keywords=%23github&origin=HASH_TAG_FROM_FEED): Provides built-in tools like forks, pull requests, and code reviews.  
  
 6. Authentication  
  
 • [hashtag#Git](https://www.linkedin.com/search/results/all/?keywords=%23git&origin=HASH_TAG_FROM_FEED): No authentication is needed for local usage.  
 • [hashtag#GitHub](https://www.linkedin.com/search/results/all/?keywords=%23github&origin=HASH_TAG_FROM_FEED): Requires authentication (via SSH keys, OAuth, or tokens) for private repositories.  
  
 7. Features  
  
 • [hashtag#Git](https://www.linkedin.com/search/results/all/?keywords=%23git&origin=HASH_TAG_FROM_FEED): Version control, branching, merging, commit history.  
 • [hashtag#GitHub](https://www.linkedin.com/search/results/all/?keywords=%23github&origin=HASH_TAG_FROM_FEED): Repository hosting, issue tracking, discussions, CI/CD, project boards.  
  
 8. Alternatives  
  
 • [hashtag#Git](https://www.linkedin.com/search/results/all/?keywords=%23git&origin=HASH_TAG_FROM_FEED): No direct alternatives since it’s a version control system.  
 • [hashtag#GitHub](https://www.linkedin.com/search/results/all/?keywords=%23github&origin=HASH_TAG_FROM_FEED): Alternatives include GitLab, Bitbucket, and Azure DevOps.

https://www.linkedin.com/feed/update/urn:li:activity:7292843647106965504?utm\_source=share&utm\_medium=member\_desktop

Understand the 𝗜𝗻𝘁𝗲𝗿𝗮𝗰𝘁𝗶𝗼𝗻 𝗙𝗹𝗼𝘄 of a Springboot application  
  
1: 𝗖𝗹𝗶𝗲𝗻𝘁 (e.g., browser, Postman, or another service) sends an HTTP request to the Controller.  
  
2: The 𝗖𝗼𝗻𝘁𝗿𝗼𝗹𝗹𝗲𝗿 receives the request, validates it, and calls the appropriate Service method.  
  
3: The 𝗦𝗲𝗿𝘃𝗶𝗰𝗲 contains the business logic and interacts with the Repository to fetch or save data.  
  
4: The 𝗥𝗲𝗽𝗼𝘀𝗶𝘁𝗼𝗿𝘆 communicates with the database to perform CRUD operations.  
  
🌻 𝗧𝗵𝗲 𝗿𝗲𝘀𝗽𝗼𝗻𝘀𝗲 𝗳𝗹𝗼𝘄𝘀 𝗯𝗮𝗰𝗸 𝗳𝗿𝗼𝗺 𝘁𝗵𝗲 𝗥𝗲𝗽𝗼𝘀𝗶𝘁𝗼𝗿𝘆 → 𝗦𝗲𝗿𝘃𝗶𝗰𝗲 → 𝗖𝗼𝗻𝘁𝗿𝗼𝗹𝗹𝗲𝗿 → 𝗖𝗹𝗶𝗲𝗻𝘁.  
  
src/  
├── main/  
│ ├── java/  
│ │ └── com/  
│ │ └── example/  
│ │ └── userapp/  
│ │ ├── [Application.java](http://application.java/)  
│ │ ├── controller/  
│ │ │ └── [UserController.java](http://usercontroller.java/)  
│ │ ├── service/  
│ │ │ └── [UserService.java](http://userservice.java/)  
│ │ ├── repository/  
│ │ │ └── [UserRepository.java](http://userrepository.java/)  
│ │ └── model/  
│ │ └── [User.java](http://user.java/)  
│ └── resources/  
│ └── [application.properties](http://application.properties/)  
└── test/  
 └── java/  
 └── com/  
 └── example/  
 └── userapp/  
 └── [UserAppTests.java](http://userapptests.java/)  
  
🌻 Follow [Hamza Ullah](https://www.linkedin.com/in/hamza-ullah-a04b9727a/) for daily getting better in your engineering journey  
  
[hashtag#Programming](https://www.linkedin.com/search/results/all/?keywords=%23programming&origin=HASH_TAG_FROM_FEED) [hashtag#SoftwareEngineering](https://www.linkedin.com/search/results/all/?keywords=%23softwareengineering&origin=HASH_TAG_FROM_FEED) [hashtag#CleanCode](https://www.linkedin.com/search/results/all/?keywords=%23cleancode&origin=HASH_TAG_FROM_FEED) [hashtag#EducationInTech](https://www.linkedin.com/search/results/all/?keywords=%23educationintech&origin=HASH_TAG_FROM_FEED)

<https://www.linkedin.com/feed/update/urn:li:activity:7292843612722085891?utm_source=share&utm_medium=member_desktop>

<https://www.linkedin.com/feed/update/urn:li:activity:7292843598222426113?utm_source=share&utm_medium=member_desktop>

Spring Boot and Microservices

𝗦𝗶𝗺𝗽𝗹𝗶𝗳𝘆 𝗬𝗼𝘂𝗿 𝗦𝗽𝗿𝗶𝗻𝗴 𝗕𝗼𝗼𝘁 𝗝𝗼𝘂𝗿𝗻𝗲𝘆 𝘄𝗶𝘁𝗵 𝗦𝗽𝗿𝗶𝗻𝗴 𝗜𝗻𝗶𝘁𝗶𝗮𝗹𝗶𝘇𝗿! 🌱  
  
Have you ever felt overwhelmed by the initial setup of a Spring Boot project? Selecting dependencies, configuring build tools, and structuring the project can be time-consuming. Enter 𝗦𝗽𝗿𝗶𝗻𝗴 𝗜𝗻𝗶𝘁𝗶𝗮𝗹𝗶𝘇𝗿 - your ultimate shortcut to kickstart your Spring Boot applications!  
  
Spring Initializr is more than just a tool; it’s a productivity booster. With just a few clicks, you can:  
- Choose your preferred build tool (Maven or Gradle).  
- Select the language (Java, Kotlin, or Groovy) and Spring Boot version.  
- Add essential dependencies like Spring Web, Security, or JPA.  
- Download a fully structured project ready to run in your favorite IDE.  
  
𝗪𝗵𝘆 𝘀𝗵𝗼𝘂𝗹𝗱 𝘆𝗼𝘂 𝘂𝘀𝗲 𝗶𝘁?  
1. 𝗧𝗶𝗺𝗲 𝗘𝗳𝗳𝗶𝗰𝗶𝗲𝗻𝗰𝘆: Spend less time on setup and more on coding.  
2. 𝗘𝗿𝗿𝗼𝗿 𝗥𝗲𝗱𝘂𝗰𝘁𝗶𝗼𝗻: Avoid dependency misconfigurations with preconfigured setups.  
3. 𝗖𝗼𝗻𝘀𝗶𝘀𝘁𝗲𝗻𝗰𝘆: Generate projects with a standard structure that’s easy to share across teams.  
4. 𝗜𝗻𝘁𝗲𝗴𝗿𝗮𝘁𝗶𝗼𝗻-𝗙𝗿𝗶𝗲𝗻𝗱𝗹𝘆: Works seamlessly with IDEs like IntelliJ IDEA, Eclipse (via STS), and VSCode.  
  
Whether you're a beginner exploring Spring Boot or an experienced developer looking to save time, Spring Initializr is a game-changer.  
  
🔗 Try it now at [[start.spring.io](http://start.spring.io/)] and experience the difference!  
  
💬 What do you think about Spring Initializr? Do you use it in your projects? Let’s discuss in the comments below!

𝗦𝗶𝗺𝗽𝗹𝗶𝗳𝘆 𝗬𝗼𝘂𝗿 𝗦𝗽𝗿𝗶𝗻𝗴 𝗕𝗼𝗼𝘁 𝗝𝗼𝘂𝗿𝗻𝗲𝘆 𝘄𝗶𝘁𝗵 𝗦𝗽𝗿𝗶𝗻𝗴 𝗜𝗻𝗶𝘁𝗶𝗮𝗹𝗶𝘇𝗿! 🌱  
  
Have you ever felt overwhelmed by the initial setup of a Spring Boot project? Selecting dependencies, configuring build tools, and structuring the project can be time-consuming. Enter 𝗦𝗽𝗿𝗶𝗻𝗴 𝗜𝗻𝗶𝘁𝗶𝗮𝗹𝗶𝘇𝗿 - your ultimate shortcut to kickstart your Spring Boot applications!  
  
Spring Initializr is more than just a tool; it’s a productivity booster. With just a few clicks, you can:  
- Choose your preferred build tool (Maven or Gradle).  
- Select the language (Java, Kotlin, or Groovy) and Spring Boot version.  
- Add essential dependencies like Spring Web, Security, or JPA.  
- Download a fully structured project ready to run in your favorite IDE.  
  
𝗪𝗵𝘆 𝘀𝗵𝗼𝘂𝗹𝗱 𝘆𝗼𝘂 𝘂𝘀𝗲 𝗶𝘁?  
1. 𝗧𝗶𝗺𝗲 𝗘𝗳𝗳𝗶𝗰𝗶𝗲𝗻𝗰𝘆: Spend less time on setup and more on coding.  
2. 𝗘𝗿𝗿𝗼𝗿 𝗥𝗲𝗱𝘂𝗰𝘁𝗶𝗼𝗻: Avoid dependency misconfigurations with preconfigured setups.  
3. 𝗖𝗼𝗻𝘀𝗶𝘀𝘁𝗲𝗻𝗰𝘆: Generate projects with a standard structure that’s easy to share across teams.  
4. 𝗜𝗻𝘁𝗲𝗴𝗿𝗮𝘁𝗶𝗼𝗻-𝗙𝗿𝗶𝗲𝗻𝗱𝗹𝘆: Works seamlessly with IDEs like IntelliJ IDEA, Eclipse (via STS), and VSCode.  
  
Whether you're a beginner exploring Spring Boot or an experienced developer looking to save time, Spring Initializr is a game-changer.  
  
🔗 Try it now at [[start.spring.io](http://start.spring.io/)] and experience the difference!  
  
💬 What do you think about Spring Initializr? Do you use it in your projects? Let’s discuss in the comments below!

Day 6: Difference Between Primitive Types and Wrapper Classes in Java  
In Java, wrapper classes are used to convert primitive data types into objects. But what exactly makes them different? Let’s break it down:  
📌 Key Takeaways:  
✔ Use primitives when performance matters.  
✔ Use wrapper classes when working with Collections, Streams, or needing null values.  
✔ Autoboxing & Unboxing handle conversion between them automatically.  
💬 Which one do you prefer using in your projects? Drop your thoughts below! 👇  
📢 Follow me for more Java tips and tricks  
💬 Follow me for daily updates: <https://lnkd.in/gsF-Uxaq>  
Join the discussion in the Java Tech Talk Group: <https://lnkd.in/gYvTGTTH>

𝗨𝗻𝗹𝗼𝗰𝗸𝗶𝗻𝗴 𝘁𝗵𝗲 𝗣𝗼𝘄𝗲𝗿 𝗼𝗳 𝗕𝗮𝘁𝗰𝗵 𝗣𝗿𝗼𝗰𝗲𝘀𝘀𝗶𝗻𝗴 𝘄𝗶𝘁𝗵 𝗦𝗽𝗿𝗶𝗻𝗴 𝗕𝗼𝗼𝘁: 𝗔 𝗗𝗲𝗲𝗽 𝗗𝗶𝘃𝗲 𝗶𝗻𝘁𝗼 𝗦𝘁𝗲𝗽𝘀  
  
In the world of Java development, particularly when using Spring Boot, batch processing is an essential technique that can significantly enhance performance and efficiency. One of the core components of Spring Batch is the concept of 𝘀𝘁𝗲𝗽𝘀, which allows developers to break down complex batch jobs into manageable, discrete tasks.  
  
Each step in a Spring Batch job represents a phase in the processing pipeline. This modular approach not only simplifies error handling and debugging but also provides flexibility in job design. By defining multiple steps, you can easily orchestrate various processing activities such as reading data, transforming it, and writing the results to a destination.  
  
What makes steps particularly powerful is their ability to be reused and composed. You can create a library of steps that can be shared across different jobs, promoting code reusability and consistency. Additionally, Spring Batch offers robust support for transaction management, ensuring data integrity throughout the processing lifecycle.  
  
As we continue to embrace microservices architecture, the role of batch processing becomes even more critical. Efficiently managing large volumes of data without compromising performance is vital for modern applications. By leveraging Spring Batch and its step-oriented architecture, developers can build scalable solutions that meet the demands of today's data-driven landscape.  
  
I’d love to hear your thoughts! How have you utilized batch processing in your projects? What challenges have you faced when implementing steps in Spring Batch? Let’s start a conversation!  
  
[hashtag#SpringBoot](https://www.linkedin.com/search/results/all/?keywords=%23springboot&origin=HASH_TAG_FROM_FEED) [hashtag#Java](https://www.linkedin.com/search/results/all/?keywords=%23java&origin=HASH_TAG_FROM_FEED) [hashtag#BatchProcessing](https://www.linkedin.com/search/results/all/?keywords=%23batchprocessing&origin=HASH_TAG_FROM_FEED) [hashtag#SoftwareDevelopment](https://www.linkedin.com/search/results/all/?keywords=%23softwaredevelopment&origin=HASH_TAG_FROM_FEED) [hashtag#Microservices](https://www.linkedin.com/search/results/all/?keywords=%23microservices&origin=HASH_TAG_FROM_FEED) [hashtag#DataManagement](https://www.linkedin.com/search/results/all/?keywords=%23datamanagement&origin=HASH_TAG_FROM_FEED) [hashtag#CodingCommunity](https://www.linkedin.com/search/results/all/?keywords=%23codingcommunity&origin=HASH_TAG_FROM_FEED) [hashtag#TechTalk](https://www.linkedin.com/search/results/all/?keywords=%23techtalk&origin=HASH_TAG_FROM_FEED)

𝐓𝐡𝐞 𝐒𝐞𝐜𝐫𝐞𝐭 𝐒𝐚𝐮𝐜𝐞 𝐛𝐞𝐡𝐢𝐧𝐝 𝐒𝐩𝐫𝐢𝐧𝐠 𝐌𝐕𝐂

𝐒𝐩𝐫𝐢𝐧𝐠 𝐌𝐕𝐂 is a module of the Spring framework that implements the Model-View-Controller (MVC) design pattern. MVC is a common pattern for web development that separates the concerns of the application into three components:

- 𝐌𝐨𝐝𝐞𝐥: The data and business logic of the application.

- 𝐕𝐢𝐞𝐰: The presentation layer that displays the data to the user.

- 𝐂𝐨𝐧𝐭𝐫𝐨𝐥𝐥𝐞𝐫: The component that handles the user requests and orchestrates the interaction between the model and the view.

By using MVC, you can achieve a clear separation of concerns, high cohesion, low coupling, and easy testability in your web application.

But Spring MVC is not just a plain implementation of MVC. It adds some extra features and benefits that make it stand out from other web frameworks. Here are some of them:

- 𝐃𝐢𝐬𝐩𝐚𝐭𝐜𝐡𝐞𝐫𝐒𝐞𝐫𝐯𝐥𝐞𝐭: This is the core component of Spring MVC that acts as a front controller. It receives all the incoming requests and delegates them to the appropriate controller based on the URL mapping. It also handles the view resolution, exception handling, locale resolution, and other common tasks.

- 𝐀𝐧𝐧𝐨𝐭𝐚𝐭𝐢𝐨𝐧-𝐛𝐚𝐬𝐞𝐝 𝐜𝐨𝐧𝐟𝐢𝐠𝐮𝐫𝐚𝐭𝐢𝐨𝐧: Spring MVC supports annotation-based configuration that allows you to define your controllers, request mappings, request parameters, model attributes, validation rules, and more with simple annotations. This reduces the need for XML configuration and makes your code more concise and readable.

- 𝐅𝐥𝐞𝐱𝐢𝐛𝐥𝐞 𝐯𝐢𝐞𝐰 𝐭𝐞𝐜𝐡𝐧𝐨𝐥𝐨𝐠𝐢𝐞𝐬: Spring MVC does not force you to use any specific view technology. You can choose from various options such as JSP, Thymeleaf, etc. You can also integrate with other view frameworks such as Angular or React.

- 𝐃𝐚𝐭𝐚 𝐛𝐢𝐧𝐝𝐢𝐧𝐠 𝐚𝐧𝐝 𝐯𝐚𝐥𝐢𝐝𝐚𝐭𝐢𝐨𝐧: Spring MVC provides powerful data binding and validation features that allow you to easily convert and validate the data from the user input to your model objects. You can use annotations such as @Valid, @NotNull, @Size, etc. to define your validation rules and get error messages automatically.

- 𝐂𝐨𝐧𝐭𝐞𝐧𝐭 𝐧𝐞𝐠𝐨𝐭𝐢𝐚𝐭𝐢𝐨𝐧: Spring MVC supports content negotiation that allows you to serve different types of content based on the request headers or parameters. For example, you can return JSON or XML depending on the Accept header of the request.

- 𝐓𝐞𝐬𝐭𝐢𝐧𝐠 𝐬𝐮𝐩𝐩𝐨𝐫𝐭: Spring MVC provides excellent testing support that allows you to write unit tests and integration tests for your controllers using mock objects and test frameworks such as JUnit or TestNG.

These are just some of the features that make Spring MVC a great choice for web development. If you want to learn more about Spring MVC, please follow me Dilane FOGUE KAMGA and hit the 🔔 on my profile to get a notification for all my new posts.

hashtag#programming

hashtag#softwaredevelopment

hashtag#java

hashtag#‍springmvc‍‍‍‍‍‍‍‍‍

Activate to view larger image,

A diagram of a computer program

AI-generated content may be incorrect.

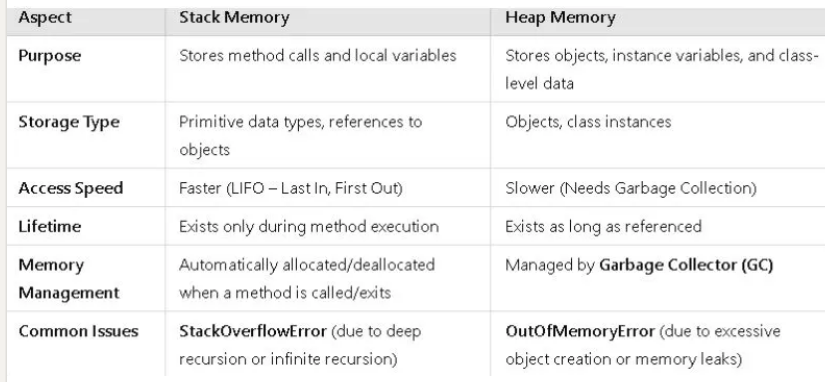
<https://www.linkedin.com/feed/update/urn:li:activity:7292500101845929984?utm_source=share&utm_medium=member_desktop>

<https://www.linkedin.com/feed/update/urn:li:activity:7292500089560793088?utm_source=share&utm_medium=member_desktop>

<https://www.linkedin.com/feed/update/urn:li:activity:7292500067834359808?utm_source=share&utm_medium=member_desktop>

🚀 Day 5: Java Memory Management – Understanding Heap vs. Stack!  
Java efficiently manages memory using two key areas: Stack Memory and Heap Memory. Knowing how they work helps in writing optimized and memory-efficient applications.🔥 How Java Manages Memory Efficiently?  
✅ Garbage Collection (GC): Automatically removes unreferenced objects in Heap Memory to free up space.  
✅ Escape Analysis: Java optimizes object allocation by deciding whether to store an object in Heap or Stack.  
✅ Memory Leaks Prevention: Use WeakReference, SoftReference, and finalize() wisely to avoid holding unnecessary objects in memory.  
💡 Best Practices for Memory Optimization  
✔ Use primitive types (int, double) instead of wrapper classes (Integer, Double) when possible.  
✔ Avoid creating unnecessary objects inside loops.  
✔ Use StringBuilder instead of String concatenation (+ operator creates multiple objects in Heap!).  
✔ Monitor memory usage with profiling tools like JVisualVM, YourKit, or Eclipse MAT.  
Efficient memory usage can boost performance and prevent memory leaks in large-scale Java applications. 🚀  
💬 What’s the biggest memory-related challenge you’ve faced in Java? Let’s discuss! 👇  
📢 Follow me for more Java tips and tricks  
💬 Follow me for daily updates: <https://lnkd.in/gsF-Uxaq>  
Join the discussion in the Java Tech Talk Group: <https://lnkd.in/gYvTGTTH>  
[hashtag#Java](https://www.linkedin.com/search/results/all/?keywords=%23java&origin=HASH_TAG_FROM_FEED) [hashtag#MemoryManagement](https://www.linkedin.com/search/results/all/?keywords=%23memorymanagement&origin=HASH_TAG_FROM_FEED) [hashtag#HeapVsStack](https://www.linkedin.com/search/results/all/?keywords=%23heapvsstack&origin=HASH_TAG_FROM_FEED) [hashtag#GarbageCollection](https://www.linkedin.com/search/results/all/?keywords=%23garbagecollection&origin=HASH_TAG_FROM_FEED) [hashtag#JavaDifferenceSeries](https://www.linkedin.com/search/results/all/?keywords=%23javadifferenceseries&origin=HASH_TAG_FROM_FEED) [hashtag#Coding](https://www.linkedin.com/search/results/all/?keywords=%23coding&origin=HASH_TAG_FROM_FEED) [hashtag#SoftwareDevelopment](https://www.linkedin.com/search/results/all/?keywords=%23softwaredevelopment&origin=HASH_TAG_FROM_FEED)

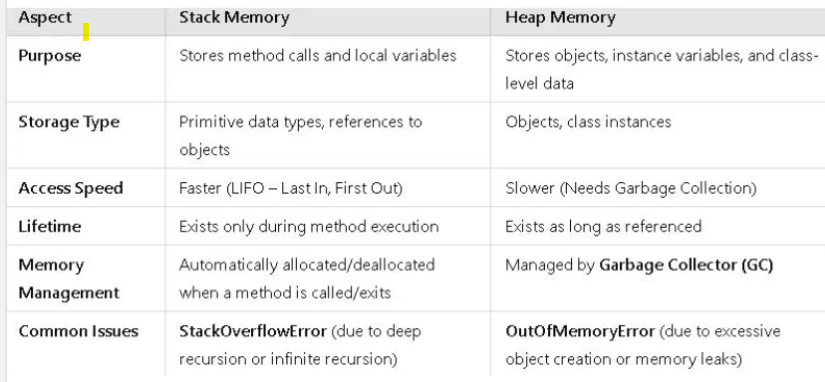
Activate to view larger image,



𝐃𝐚𝐭𝐚𝐛𝐚𝐬𝐞 𝐑𝐞𝐥𝐚𝐭𝐢𝐨𝐧𝐬𝐡𝐢𝐩𝐬 𝐢𝐧 𝐒𝐩𝐫𝐢𝐧𝐠  
  
𝐒𝐩𝐫𝐢𝐧𝐠 𝐃𝐚𝐭𝐚 𝐉𝐏𝐀 is a powerful abstraction layer that simplifies the use of JPA in Spring applications. It provides a consistent and easy-to-use API for accessing and manipulating data from various sources. It also supports various features such as CRUD operations, pagination, sorting, auditing, caching, and more.  
  
One of the main benefits of using Spring Data JPA is that it allows you to define your database relationships using annotations on your entity classes. You don’t need to write any SQL queries or boilerplate code to handle the associations between your entities. Spring Data JPA will take care of generating and executing the appropriate queries for you.  
  
There are three common types of database relationships that you can define with Spring Data JPA:  
  
- 𝐎𝐧𝐞-𝐭𝐨-𝐎𝐧𝐞: This is a relationship where one entity is associated with another entity, and both entities share the same primary key. For example, a library and an address can have a one-to-one relationship, where each library has one address and each address belongs to one library.  
  
- 𝐎𝐧𝐞-𝐭𝐨-𝐌𝐚𝐧𝐲: This is a relationship where one entity is associated with multiple entities of another type, but each entity of the other type can only be associated with one entity of the first type. For example, a customer and an order can have a one-to-many relationship, where each customer can have many orders, but each order belongs to one customer.  
  
- 𝐌𝐚𝐧𝐲-𝐭𝐨-𝐌𝐚𝐧𝐲: This is a relationship where multiple entities of one type are associated with multiple entities of another type, and vice versa. For example, a student and a course can have a many-to-many relationship, where each student can enroll in many courses, and each course can have many students.  
  
These features are used extensively in Spring Boot application development. If you want to learn more about Spring Data JPA, please follow me [Dilane FOGUE KAMGA](https://www.linkedin.com/in/dilane-fogue-kamga/) and hit the 🔔 on my profile to get a notification for all my new posts.  
  
[hashtag#programming](https://www.linkedin.com/search/results/all/?keywords=%23programming&origin=HASH_TAG_FROM_FEED)  
[hashtag#softwaredevelopment](https://www.linkedin.com/search/results/all/?keywords=%23softwaredevelopment&origin=HASH_TAG_FROM_FEED)  
[hashtag#java](https://www.linkedin.com/search/results/all/?keywords=%23java&origin=HASH_TAG_FROM_FEED)  
[hashtag#springframework](https://www.linkedin.com/search/results/all/?keywords=%23springframework&origin=HASH_TAG_FROM_FEED)  
[hashtag#springboot](https://www.linkedin.com/search/results/all/?keywords=%23springboot&origin=HASH_TAG_FROM_FEED) ‍  
[hashtag#databases](https://www.linkedin.com/search/results/all/?keywords=%23databases&origin=HASH_TAG_FROM_FEED) ‍‍‍‍  
[hashtag#springdata](https://www.linkedin.com/search/results/all/?keywords=%23springdata&origin=HASH_TAG_FROM_FEED)

=================

1. Use HTTPS  
     
    Always enforce HTTPS to protect data in transit from MITM (Man-in-the-Middle) attacks. Redirect HTTP traffic to HTTPS and use TLS 1.2+ for security.  
     
   2. Use OAuth2  
     
    OAuth 2.0 (along with OpenID Connect for identity) is the gold standard for API authentication and authorization. Avoid using API keys for authentication where possible.  
     
   3. Use WebAuthn  
     
    WebAuthn provides phishing-resistant authentication using hardware tokens or biometrics. It strengthens authentication beyond just passwords.  
     
   4. Use Leveled API Keys  
     
    Not all API keys should have the same level of access. Assign different permissions based on the client’s needs (e.g., read-only vs. full-access keys). Rotate and expire keys periodically.  
     
   5. Authorization (Beyond Authentication)  
     
    Implement role-based access control (RBAC) or attribute-based access control (ABAC). Ensure that even authenticated users only access what they’re allowed.  
     
   6. Rate Limiting  
     
    Prevent abuse and DoS attacks by setting rate limits. Use techniques like token buckets, request quotas, or IP-based throttling.  
     
   7. API Versioning  
     
    Use versioning (e.g., /v1/resource) to prevent breaking changes and ensure backward compatibility.  
     
   8. Whitelisting  
     
    Restrict API access to known, trusted IPs or clients whenever possible. This minimizes exposure to unknown threats.  
     
   9. Check OWASP API Security Risks  
     
    Stay updated with OWASP API Security Top 10 to address emerging API vulnerabilities like Broken Object Level Authorization (BOLA) and excessive data exposure.  
     
   10. Use an API Gateway  
     
    An API Gateway acts as a security layer for authentication, request validation, rate limiting, and logging. It also abstracts backend services, reducing exposure.  
     
   11. Error Handling  
     
    Avoid exposing sensitive details in error messages. Use generic error responses (400 Bad Request, 403 Forbidden) and log full error details internally.  
     
   12. Input Validation  
     
    Sanitize and validate all input to prevent SQL injection, XSS, and deserialization attacks. Use allowlists for expected values and reject everything else.  
     
    If you’re working on API security in your projects, I’d also recommend adding logging and monitoring—you should always know who’s accessing your APIs and detect anomalies early.



<https://www.linkedin.com/posts/abd-alrhman-alkraien-83a93a1b1_java-garbage-collection-activity-7293161547252535297-z3tG?utm_source=share&utm_medium=member_desktop>

hings Every Developer Should Know — CI/CD Pipeline.  
  
A CI/CD pipeline is an automated workflow that facilitates continuous integration (CI) and continuous delivery or deployment (CD) by managing code building, testing, and release processes.  
  
It integrates the various stages of the software development lifecycle (SDLC) into a seamless, repeatable process.  
  
These stages include source code management, automated testing, artifact creation, and deployment orchestration.  
  
Continuous ‘delivery’ and ‘deployment’ are sometimes used synonymously.  
  
But there is a clear and important distinction between the two.  
  
Delivery is about ensuring the software can be released at any time.  
  
It requires manual intervention to deploy to production.  
  
Deployment, on the other hand, does the release through automated workflows.  
  
Learn more here: <https://lnkd.in/gYEu2wnr>  
  
~~  
Thank you to our partner Kickresume who keeps our content free to the community.  
  
𝗔𝗿𝗲 𝘆𝗼𝘂 𝘀𝘂𝗿𝗲 𝘆𝗼𝘂𝗿 𝗿𝗲𝘀𝘂𝗺𝗲 𝗽𝗮𝘀𝘀𝗲𝘀 𝗔𝗧𝗦 𝘀𝗰𝗮𝗻𝘀? Check by using Kickresume’s AI ATS Resume checker.  
  
Check it out: <https://lnkd.in/gpbPZV4e>

Java Intermediate Interview Questions Series - Day 6  
  
🔹 Topic: JVM & Performance Optimization  
  
1️⃣ What is the Java Virtual Machine (JVM), and how does it work?  
2️⃣ Explain the different memory areas in JVM (Heap, Stack, Metaspace, etc.).  
3️⃣ What are the different types of garbage collectors in Java?  
4️⃣ How does the Garbage Collector (GC) work in Java?  
5️⃣ What are memory leaks in Java, and how do you prevent them?  
6️⃣ How does the Just-In-Time (JIT) compiler improve Java performance?  
7️⃣ What is the difference between strong, weak, soft, and phantom references?  
8️⃣ How do you monitor and optimize JVM performance?  
9️⃣ What are the best practices for writing high-performance Java applications?  
🔟 How does the -Xms and -Xmx JVM options affect memory management?

12 𝐦𝐨𝐬𝐭 𝐜𝐨𝐦𝐦𝐨𝐧 𝐆𝐢𝐭 𝐜𝐨𝐦𝐦𝐚𝐧𝐝𝐬  
  
1. git init  
  
Purpose: Initializes a new Git repository in your current directory.  
Usage: Run git init to start tracking your project with Git  
  
2. git clone  
Purpose: Creates a copy of an existing repository on your local machine.  
Usage: Run git clone <repository\_url> to download a repository.  
  
3. git add  
Purpose: Stages changes in your working directory for the next commit.  
Usage: Use git add <file> or git add . to add all changes.  
  
4. git commit  
Purpose: Commits staged changes to the repository with a descriptive message.  
Usage: Run git commit -m "Your message" to save your changes.  
  
5. git status  
Purpose: Shows the status of changes in your working directory.  
Usage: git status lets you see which files are modified, staged, or untracked.  
  
6. git push  
Purpose: Pushes your commits to a remote repository, like GitHub.  
Usage: Use git push origin <branch\_name> to upload changes.  
  
7. git pull  
Purpose: Fetches and merges updates from a remote repository.  
Usage: Run git pull origin <branch\_name> to sync your local branch.  
  
8. git branch  
Purpose: Manages branches within your repository.  
Usage: Use git branch to list branches or git branch <branch\_name> to create a new one.  
  
9. git checkout  
Purpose: Switches to a different branch or restores files.  
Usage: Run git checkout <branch\_name> to switch branches or git checkout <file> to revert a file.  
  
10. git merge  
Purpose: Merges one branch into another.  
Usage: Use git merge <branch\_name> to integrate changes from one branch into the current branch.  
  
11. git log  
Purpose: Displays the commit history.  
Usage: Run git log to see a list of past commits, along with their messages and details.  
  
12. git diff  
Purpose: Shows the differences between commits, branches, or your working directory and the last commit.  
Usage: Use git diff to compare changes and git diff <commit1> <commit2> to see differences between two commits.  
  
Here’s an extended list of useful Git commands beyond the common ones :  
  
 🚀𝐠𝐢𝐭 𝐟𝐞𝐭𝐜𝐡: Retrieve updates from a remote repository without merging.  
 🚀𝐠𝐢𝐭 𝐫𝐞𝐛𝐚𝐬𝐞: Apply commits on top of another base commit.  
 🚀𝐠𝐢𝐭 𝐫𝐞𝐬𝐞𝐭: Undo changes in the working directory or commit history.  
 🚀𝐠𝐢𝐭 𝐬𝐭𝐚𝐬𝐡: Temporarily save changes for later use.  
 🚀𝐠𝐢𝐭 𝐭𝐚𝐠: Mark specific points in history with a tag.  
 🚀𝐠𝐢𝐭 𝐫𝐞𝐦𝐨𝐭𝐞e: Manage remote repository connections.  
 🚀𝐠𝐢𝐭 𝐬𝐡𝐨𝐰: Show detailed information about a commit.  
 🚀𝐠𝐢𝐭 𝐫𝐞𝐯𝐞𝐫𝐭: Create a new commit that undoes a previous commit.  
 🚀𝐠𝐢𝐭 𝐛𝐥𝐚𝐦𝐞: Show who modified each line in a file.  
 🚀𝐠𝐢𝐭 𝐜𝐡𝐞𝐫𝐫𝐲-𝐩𝐢𝐜𝐤: Apply specific commits from another branch.  
 🚀𝐠𝐢𝐭 𝐫𝐦: Remove files from the working directory and staging area.

Authentication Mechanisms - Microservices   
  
[1.] 𝐀𝐏𝐈 𝐊𝐞𝐲𝐬  
◾ Simple, unique identifiers assigned to each client or service.  
◾ Sent as a header or query parameter with each request.  
◾ Best suited for internal services, less sensitive APIs, or for granting access to specific features.   
◾ Easy to implement and manage.  
◾ Not as secure as token-based methods. Keys can be easily leaked or stolen.  
  
[2.] 𝐁𝐚𝐬𝐢𝐜 𝐀𝐮𝐭𝐡𝐞𝐧𝐭𝐢𝐜𝐚𝐭𝐢𝐨𝐧  
◾ Username and password are sent in the Authorization header as base64 encoded string.  
◾ Simple to implement but requires HTTPS to be secure.  
◾ Suitable for simple scenarios with low-security requirements.  
◾ Widely supported and easy to understand.  
◾ Vulnerable to man-in-the-middle attacks if not used with HTTPS.  
◾ Passwords are sent in cleartext (even when encoded).  
  
[3.] 𝐉𝐒𝐎𝐍 𝐖𝐞𝐛 𝐓𝐨𝐤𝐞𝐧𝐬 (𝐉𝐖𝐓)  
◾ Self-contained tokens that carry user information and claims in a JSON payload.  
◾ Issued by an authentication server after successful login, then sent by the client in the Authorization header.  
◾ Widely used for stateless authentication in microservices, single sign-on (SSO) and authorization.  
◾ Stateless, secure, compact and can contain additional claims.  
◾ Requires proper key management for signing and verification.  
  
[4.] 𝐎𝐀𝐮𝐭𝐡 2.0  
◾ An authorization framework allowing third-party applications to obtain limited access to resources on behalf of the resource owner (user) without sharing credentials.  
◾ Uses various grant types (authorization code, implicit, client credentials, etc.) to obtain access tokens and refresh tokens.  
◾ Widely used for user authorization and delegated access to APIs.  
◾ Provides a standardized way to secure access to resources without sharing credentials.  
◾ Can be complex to implement and requires careful consideration of security vulnerabilities.  
  
[5.] 𝐎𝐩𝐞𝐧𝐈𝐃 𝐂𝐨𝐧𝐧𝐞𝐜𝐭 (𝐎𝐈𝐃𝐂)  
◾ An identity layer on top of OAuth 2.0 that provides user authentication and profile information.  
◾ Uses an ID token along with the access token to provide user identity information.  
◾ Used for authentication in conjunction with OAuth 2.0 for authorization.  
◾ Simplifies authentication by providing a standardized way to obtain user information.  
◾ Requires integration with an OIDC provider (e.g., Google, Okta).  
  
[6.] 𝐌𝐮𝐭𝐮𝐚𝐥 𝐓𝐋𝐒 (𝐦𝐓𝐋𝐒)  
◾ Both client and server authenticate each other using X.509 certificates.  
◾ Requires a certificate authority (CA) to issue and manage certificates.  
◾ Best suited for securing communication between internal services or highly sensitive APIs.  
◾ Strong security due to mutual authentication and encryption.  
◾ More complex to set up and manage compared to other mechanisms.   
  
  
\*\* These Authentication Mechanisms/Types are not limited just to microservices.

System Design was HARD  
  
Learning these concepts made it easier for me:  
  
➤ 𝗕𝗮𝘀𝗶𝗰𝘀 𝗼𝗳 𝗦𝘆𝘀𝘁𝗲𝗺 𝗗𝗲𝘀𝗶𝗴𝗻  
  
 • What is System Design?  
 • Functional vs Non Functional Requirements  
 • What are the components of System Design?  
 • System Design Life Cycle | SDLC (Design)  
 • Structured Analysis and Structured Design  
 • System Design Strategy  
 • Database Sharding - Concept  
 • Horizontal and Vertical Scaling  
 • Load Balancer in System Design  
 • Routing requests through Load Balancer  
 • Latency and Throughput in System Design  
 • Object-Oriented Analysis and Design  
 • Difference between Structured and Object-Oriented Analysis  
  
➤ 𝗟𝗼𝘄 𝗟𝗲𝘃𝗲𝗹 𝗗𝗲𝘀𝗶𝗴𝗻 (𝗟𝗟𝗗)  
  
 • What is Low Level Design or LLD  
 • Data Structures and Algorithms for System Design  
 • Event-Driven Architecture  
 • Difference between Authentication and Authorization  
 • What is API Gateway  
 • What is Data Encryption?  
 • Design Patterns  
 • Code Optimization Techniques  
 • Unit Testing  
 • Integration Testing  
 • CI/CD: Continuous Integration and Continuous Delivery  
 • Introduction to Modularity and Interfaces In System Design  
 • Data Partitioning Techniques  
 • Class Diagrams | UML  
  
➤ 𝗛𝗶𝗴𝗵 𝗟𝗲𝘃𝗲𝗹 𝗗𝗲𝘀𝗶𝗴𝗻 (𝗛𝗟𝗗)  
  
 • What is High Level Design  
 • Availability in System Design  
 • Consistency in System Design  
 • Reliability in System Design  
 • CAP Theorem  
 • Difference between Process and Thread  
 • Difference between Concurrency and Parallelism  
 • Load Balancer  
 • Consistent Hashing  
 • Content Delivery Network (CDN) in System Design  
 • Caching in System Design  
 • Cache Eviction Policies  
 • Message Queues  
 • Communication Protocols  
 • Network Protocols and Proxies in System Design  
 • Unified Modeling Language (UML)  
  
➤ 𝗗𝗼 𝗧𝗵𝗲𝘀𝗲 𝗤𝘂𝗲𝘀𝘁𝗶𝗼𝗻𝘀  
  
 • Design URL Shortening Service  
 • Design Dropbox  
 • Design Twitter  
 • System Design Netflix - A Complete Architecture  
 • System Design of Uber App - Uber System Architecture  
 • Design Book My Show  
 • Designing Facebook Messenger  
 • Designing WhatsApp Messenger  
 • Designing Instagram  
  
➤ To learn all these concepts, check out this detailed post of mine on LinkedIn. It contains a comprehensive Medium article that covers each of these topics: <https://lnkd.in/dcrYuz3a>  
<https://lnkd.in/dqnEe-xT>  
  
➤ 𝐋𝐞𝐚𝐫𝐧 𝐑𝐞𝐚𝐥-𝐖𝐨𝐫𝐥𝐝 𝐒𝐲𝐬𝐭𝐞𝐦 𝐃𝐞𝐬𝐢𝐠𝐧:  
  
 - [Arpit Bhayani](https://www.linkedin.com/in/arpitbhayani/)   
 - [Shrayansh Jain](https://www.linkedin.com/in/jainshrayansh/)   
 - [Gaurav Sen](https://www.linkedin.com/in/gkcs/)   
 - [Saurabh Dashora](https://www.linkedin.com/in/saurabh-dashora/)   
 - [Rocky Bhatia](https://www.linkedin.com/in/rocky-bhatia-a4801010/)  
 - [Alexandre Zajac](https://www.linkedin.com/in/alexandre-zajac/)

𝗝𝗮𝘃𝗮 𝗕𝗮𝘀𝗶𝗰𝘀 𝗮𝗻𝗱 𝗖𝗼𝗿𝗲 𝗖𝗼𝗻𝗰𝗲𝗽𝘁𝘀

• Differentiate between Volatile and Transient Variable in Java.

• How is Collection different from Collections in Java?

• What is the importance of the hashCode() and equals() contract?

• How is the classpath variable different from the path variable?

• Is it necessary to declare all immutable objects as final?

• Can we use HashMap in a multi-threaded environment?

• What is the Properties class?

• What are weak references in Java?

• How does Java handle stack and heap memory?

• What will happen if you run 1.0/0.0?

𝗖𝗼𝗹𝗹𝗲𝗰𝘁𝗶𝗼𝗻𝘀 𝗮𝗻𝗱 𝗗𝗮𝘁𝗮 𝗦𝘁𝗿𝘂𝗰𝘁𝘂𝗿𝗲𝘀

• Differentiate between the Vector and ArrayList collections in Java.

• Distinguish between ArrayList and Vector in Java.

• Differentiate between Enumeration and Iterator.

• Describe EnumSet.

• IdentityHashMap: Discuss its functionality.

𝗠𝘂𝗹𝘁𝗶𝘁𝗵𝗿𝗲𝗮𝗱𝗶𝗻𝗴 𝗮𝗻𝗱 𝗖𝗼𝗻𝗰𝘂𝗿𝗿𝗲𝗻𝗰𝘆

• Can we use HashMap in a multi-threaded environment?

• How would you help a colleague with less Java experience who has trouble serializing a class?

• Write some important features of Java 8.

• How can you prevent Deadlock in Java?

• What are ThreadLocal variables?

• Mention 5 best practices used with threads.

• What is the best possible way to call the wait() method – using the if construct or the loop construct?

𝗗𝗲𝘀𝗶𝗴𝗻 𝗣𝗮𝘁𝘁𝗲𝗿𝗻𝘀

• Briefly discuss the Factory Design pattern.

• What do you know about the Factory Design Pattern in Java?

• Explain the Template Method design pattern.

• Discuss Dependency Injection and its significance in Object-Oriented Programming.

• When should one use the Flyweight pattern?

• In the Singleton Pattern, what is the significance of double-checked locking?

𝗝𝗮𝘃𝗮 𝗠𝗲𝗺𝗼𝗿𝘆 𝗠𝗮𝗻𝗮𝗴𝗲𝗺𝗲𝗻𝘁

• How does Garbage Collection work in Java?

• What are Java Memory Leaks?

• Explain SerialVersionUID.

𝗦𝘁𝗿𝗶𝗻𝗴 𝗛𝗮𝗻𝗱𝗹𝗶𝗻𝗴

• Distinguish between String creation using the new() operator and String literal.

• Is there any difference in defining or creating a String by using a String literal and using the new() operator?

𝗘𝘅𝗰𝗲𝗽𝘁𝗶𝗼𝗻 𝗛𝗮𝗻𝗱𝗹𝗶𝗻𝗴

• Is it possible to override a method to throw a RuntimeException from throwing a NullPointerException in the parent class?

𝗔𝗱𝘃𝗮𝗻𝗰𝗲𝗱 𝗧𝗼𝗽𝗶𝗰𝘀

• Differentiate between fail-safe and fail-fast.

• What is the difference between SAX and DOM parsers?

• Is it possible to overload the main method?

• What is the significance of String creation using the new() operator vs String literals?

• Mention the benefits and limitations of Spring AOP.

• How would you help a colleague with less Java experience who has trouble serializing a class?

JWT 101: Key to Stateless Authentication  
  
JWT or JSON Web Tokens is an open standard for securely transmitting information between two parties. They are widely used for authentication and authorization.  
  
A JWT consists of three main components:  
  
1 - Header  
Every JWT carries a header specifying the algorithms for signing the JWT. It’s written in JSON format.  
  
2 - Payload  
The payload consists of the claims and the user data. There are different types of claims such as registered, public, and private claims.  
  
3 - Signature  
The signature is what makes the JWT secure. It is created by taking the encoded header, encoded payload, secret key, and the algorithm and signing it.   
  
JWTs can be signed in two different ways:  
  
1 - Symmetric Signatures  
It uses a single secret key for both signing the token and verifying it. The same key must be shared between the server that signs the JWT and the system that verifies it.  
  
2 - Asymmetric Signatures  
In this case, a private key is used to sign the token, and a public key to verify it. The private key is kept secure on the server, while the public key can be distributed to anyone who needs to verify the token.  
  
Over to you: Do you use JWTs for authentication?=

Spring Boot Annotations Overview: Mastering Your Development  
  
Ready to dive into the world of Spring Boot? Here’s a quick overview of key annotations that can supercharge your development process.  
  
→ @SpringBootApplication combines essential configurations for your app to run smoothly.  
  
→ @EnableAutoConfiguration automates settings based on your classpath and beans.  
  
→ @ComponentScan specifies which packages the Spring Framework should scan for components.  
  
→ @RestController simplifies REST API development by combining @Controller and @ResponseBody.  
  
→ @RequestMapping maps HTTP requests to specific controller methods for seamless navigation.  
  
→ @Autowired allows Spring to manage dependencies automatically—no more manual wiring!  
  
→ @Qualifier helps when multiple candidates exist for dependency injection, ensuring the right one is chosen.  
  
→ @Bean indicates that a method produces a bean to be managed by the Spring container.  
  
→ @ConfigurationProperties binds external configurations to your application, improving flexibility.  
  
→ @Scheduled enables you to run methods at specific intervals, perfect for background tasks.  
  
What annotation are you most excited to use in your next project? Let's discuss in the comments!

SQL Must Know Differences:  
  
🔰 RANK vs DENSE\_RANK:  
RANK: Provides a ranking with gaps if there are ties.  
DENSE\_RANK: Provides a ranking without gaps, even in the case of ties.  
  
🔰 HAVING vs WHERE Clause:  
WHERE: Filters rows before grouping.  
HAVING: Filters groups after the GROUP BY clause.  
  
🔰 UNION vs UNION ALL:  
UNION: Removes duplicates and combines results.  
UNION ALL: Combines results without removing duplicates.  
  
🔰 JOIN vs UNION:  
JOIN: Combines columns from multiple tables.  
UNION: Combines rows from multiple tables with similar structure.  
  
🔰 DELETE vs DROP vs TRUNCATE:  
DELETE: Removes rows, with the option to filter.  
DROP: Removes the entire table or database.  
TRUNCATE: Deletes all rows but keeps the table structure.  
  
🔰 CTE vs TEMP TABLE:  
CTE: Temporary result set used within a single query.  
TEMP TABLE: Physical temporary table that persists for the session.  
  
🔰 SUBQUERIES vs CTE:  
SUBQUERIES: Nested queries inside the main query.  
CTE: Can be more readable and used multiple times in a query.  
  
🔰 ISNULL vs COALESCE:  
ISNULL: Replaces NULL with a specified value, accepts two parameters.  
COALESCE: Returns the first non-NULL value from a list of expressions, accepting multiple parameters.  
  
🔰 INTERSECT vs INNER JOIN:  
INTERSECT: Returns common rows from two queries.  
INNER JOIN: Combines matching rows from two tables based on a condition.  
  
🔰 EXCEPT vs NOT IN:  
EXCEPT: Returns rows in the first query but not in the second.  
NOT IN: Filters rows where a column's value is not in a given list.

Stop searching for jobs only on Linkedin or Naukri!  
  
Here are 20 Sites to find Remote jobs that pay in USD:  
  
1) Flexjobs: <https://www.flexjobs.com/>  
  
- #1 job site to find remote jobs - no ads, scams, or junk.  
  
2) We work Remotely: <https://lnkd.in/gyhYQ4AN>  
  
-Home to the largest remote work community in the world with 4.5M visitors.  
  
3) Remote Workmate: <https://lnkd.in/g78sRb_6>  
  
-Achieve more for less with a reliable offshore specialist  
  
4) Pangian: <https://pangian.com/>  
  
-The fastest-growing remote community worldwide  
  
5) JustRemote: <https://justremote.co/>  
  
-Fully and partially remote jobs from the greatest remote working companies  
  
6) Remotive - Remote Jobs: <https://remotive.com/>  
  
-Easily access active and fully remote job opportunities from vetted tech companies.  
  
7) Indeed(dot)com: <https://in.indeed.com/>  
  
-needs no introductions.  
  
8) Upwork: <https://www.upwork.com/>  
  
-You can have the best people. Right now. Right here.  
  
Hey students! 👋 Searching for Free Courses?  
  
Access FREE Courses straight from the Google Platform as you gear up for opportunities this fall.  
  
Here are 13 courses you can’t afford to miss:  
  
✅ Like  
✅ Comment  
✅ Share  
  
↣ 7000+ Course Free Access : <https://lnkd.in/drN4mT6t>  
  
<>. Google Data Analytics:  
  
▶ <https://lnkd.in/dGN74Ctb>  
  
↣ IBM Data Science Professional Certificate  
<https://lnkd.in/dP2UpAju>  
  
▶ Python  
<https://lnkd.in/dBFMhgvK>  
  
↣ R  
<https://lnkd.in/dQAgfGAG>  
  
▶ PowerBI  
<https://lnkd.in/dsFqHgtU>  
  
↣ Mathematics  
<https://lnkd.in/dQr5Qqat>  
  
▶ Tableau  
<https://lnkd.in/dx5Rb-5x>  
  
↣ Excel and PowerBI  
<https://lnkd.in/dFyens7K>  
  
▶ Probability  
<https://lnkd.in/dxNkhZBp>  
  
↣ Statistics  
<https://lnkd.in/dWkiYKPN>  
  
▶ Linear Algebra  
<https://lnkd.in/dJg6Gx5j>  
  
↣ Machine Learning  
<https://lnkd.in/dxiBYBtb>  
  
▶ Deep Learning  
<https://lnkd.in/dj2kBYgv>  
  
↣ Data Analysis  
<https://lnkd.in/dnNqGvNE>  
  
▶ Data Visualization  
<https://lnkd.in/dXtQX6Zk>  
  
↣ SQL  
<https://lnkd.in/dEEzThMn>  
  
Happy Learning   
  
Pdf credit -Respective Owner   
  
Follow [Anjali .](https://www.linkedin.com/in/anjali-984058306/) for more .. 🎯

How Digital Signatures Work?  
  
A digital signature is a specific kind of electronic signature to sign and secure electronically transmitted documents.  
  
Digital signatures are similar to physical signatures since they are unique to every person. They identify the identity of the signer.  
  
Here’s an example of the working process of a digital signature with Alice as the sender and John as the recipient:  
  
1 - Alice generates a cryptographic key pair consisting of a private key and a corresponding public key. The private key remains confidential and is known only to the signer, while the public key can be shared openly.  
2 - The signer (Alice) uses a hash function to create a unique fixed-length string of numbers and letters, called a hash, from the document. This hash value represents the contents of the document.  
3 - Alice uses their private key to encrypt the hash value of the message. This hash value is known as the digital signature.  
4 - The digital signature is attached to the original document, creating a digitally signed document. It is transmitted over the network to the recipient.  
5 - The recipient (John) extracts both the digital signature and the original hash value from the document.   
6 - The recipient uses Alice’s public key to decrypt the digital signature. This produces a hash value that was originally encrypted with the private key.  
7 - The recipient calculates a new hash value for the received message using the same hashing algorithm as the signer. They then compare this recalculated hash with the decrypted hash value obtained from the digital signature.  
8 - If the hash values are equal, the digital signature is valid, and it is determined that the document has not been tampered with or altered.  
  
Over to you: Have you used digital signatures?

16 API Terms You Must Know  
  
→ Resource: The fundamental concept in REST, representing data or service.  
  
→ Request: A call made to a server to access a resource.  
  
→ Response: The data sent back from the server to the client.  
  
→ Response Code: Indicates the status of a HTTP request, like 404 not found.  
  
→ Payload: Data sent within a request or response.  
  
→ Pagination: The process of dividing response data into discrete pages.  
  
→ Method: The HTTP actions such as GET, POST, PUT, DELETE.  
  
→ Query Parameters: Data appended to the URL to refine searches.  
  
→ Authentication: The verification of a user's identity.  
  
→ Rate Limiting: Restricting the number of requests a user can make.  
  
→ API Integration: Connecting various services using APIs.  
  
→ API Gateway: A service that provides a single entry point for APIs.  
  
→ API Lifecycle: The phases of API development and retirement.  
  
→ CRUD: An acronym for create, read, update, delete.  
  
→ Cache: Temporary storage to speed up data retrieval.  
  
→ Client: The device or program that requests data from a server.  
  
What API term surprised you the most?

 Unlock the Power of AI with Spring Boot Azure AI Text Analytics  
! 🚀 <https://lnkd.in/gJNwWJjR>  
Dive into Sentiment Analysis, Entity Recognition, and Language Detection in your Spring Boot applications using Azure AI Text Analytics. 🌐💡  
📖 Guide Includes:  
Seamless integration of Azure AI  
Detailed steps for sentiment analysis, entity recognition, and language detection  
Build smarter applications in no time!  
🔗 Read Full Guide: [<https://lnkd.in/gJNwWJjR>]  
[hashtag#AI](https://www.linkedin.com/search/results/all/?keywords=%23ai&origin=HASH_TAG_FROM_FEED) [hashtag#SpringBoot](https://www.linkedin.com/search/results/all/?keywords=%23springboot&origin=HASH_TAG_FROM_FEED) [hashtag#AzureAI](https://www.linkedin.com/search/results/all/?keywords=%23azureai&origin=HASH_TAG_FROM_FEED) [hashtag#SentimentAnalysis](https://www.linkedin.com/search/results/all/?keywords=%23sentimentanalysis&origin=HASH_TAG_FROM_FEED) [hashtag#EntityRecognition](https://www.linkedin.com/search/results/all/?keywords=%23entityrecognition&origin=HASH_TAG_FROM_FEED) [hashtag#LanguageDetection](https://www.linkedin.com/search/results/all/?keywords=%23languagedetection&origin=HASH_TAG_FROM_FEED) [hashtag#MachineLearning](https://www.linkedin.com/search/results/all/?keywords=%23machinelearning&origin=HASH_TAG_FROM_FEED) [hashtag#AIIntegration](https://www.linkedin.com/search/results/all/?keywords=%23aiintegration&origin=HASH_TAG_FROM_FEED) [hashtag#TechInnovation](https://www.linkedin.com/search/results/all/?keywords=%23techinnovation&origin=HASH_TAG_FROM_FEED) [hashtag#JavaDevelopers](https://www.linkedin.com/search/results/all/?keywords=%23javadevelopers&origin=HASH_TAG_FROM_FEED) [hashtag#CloudComputing](https://www.linkedin.com/search/results/all/?keywords=%23cloudcomputing&origin=HASH_TAG_FROM_FEED) [hashtag#Azure](https://www.linkedin.com/search/results/all/?keywords=%23azure&origin=HASH_TAG_FROM_FEED) [hashtag#TechGuide](https://www.linkedin.com/search/results/all/?keywords=%23techguide&origin=HASH_TAG_FROM_FEED)

𝗧𝗼𝗽 𝗰𝗼𝗱𝗶𝗻𝗴 𝗮𝘀𝘀𝗶𝘀𝘁𝗮𝗻𝘁 𝘁𝗼𝗼𝗹𝘀 𝗲𝘃𝗲𝗿𝘆 𝗱𝗲𝘃𝗲𝗹𝗼𝗽𝗲𝗿 𝘀𝗵𝗼𝘂𝗹𝗱 𝗲𝘅𝗽𝗹𝗼𝗿𝗲 📌   
  
⏹️ GitHub Copilot  
GitHub Copilot is a code completion and automatic programming tool developed by GitHub and OpenAI  
  
⏹️ Replit  
Replit is an AI-powered software development & deployment platform for building, sharing, and shipping software fast  
  
⏹️ ChatGPT  
AI-driven conversational assistant, helps with code and more. Limited context handling, requires manual code execution.  
  
⏹️ Coderabbit  
CodeRabbit is an AI-driven code review partner which enhances code quality and minimizes manual effort by providing context-aware, line-by-line feedback, highlighting critical changes, enabling bot interactions  
  
⏹️ CodePro  
Advanced AI Assitant For Lighting-Fast Answer  
  
⏹️ IntelliCode  
The Visual Studio IntelliCode extension provides AI-assisted development features for Python, TypeScript/JavaScript and Java developers   
  
⏹️ Codiga  
Codiga offers a birds-eye view of your code quality. The Codiga dashboard reports all important metrics about your code quality  
  
⏹️ CodeWP  
It's a fantastic tool for generating WordPress code snippets  
  
⏹️ Autocode  
Build complex software with AI. An innovative, terminal-based AI coding engine  
  
⏹️ AIXcoder  
This provides you with method-level code generation from natural language to code, and intelligent code completion for whole or multiple lines  
  
⏹️ Hugging Face  
Offers advanced AI models for coding, multi-language support.  
  
⏹️ CodiumAI  
Offers quality-first AI tools for developers to write, test, and review code directly within their IDE and Git.  
  
⏹️ Tabnine  
AI code assistant that accelerates and simplifies software development while keeping your code private, secure, and compliant.  
  
⏹️ Cursor AI  
Built to make you extraordinarily productive, Cursor is the best way to code with AI.  
  
⏹️ Amazon Q Developer  
The most capable generative AI–powered assistant for software development

Spring Boot simplified in this 3-phase roadmap👇🏻  
  
PHASE 1 - Basics  
  
• Creating a spring boot project using spring initializr  
• Maven and gradle build tools  
• Annotations  
• Profiles and environment-specific configurations  
• @getmapping, @postmapping, @putmapping, @deletemapping  
• Handling path variables and request parameters  
• Setting up database connection (h2, mysql, postgresql)  
• Using jparepository and crudrepository  
• Introduction to spring boot devtools  
• Enabling hot reloading  
• Spring batch, scheduling and cron expressions  
  
  
PHASE 2 - Intermediate  
  
• Using @controlleradvice and @exceptionhandler  
• Custom error responses and exception classes  
• Global exception handling  
• Basic authentication  
• Configuring security for apis  
• Implementing jwt (json web tokens) for stateless authentication  
• Introduction to hateoas  
• Versioning rest apis (uri, parameter, headers)  
• Unit testing with junit and mockito  
• Writing integration tests with spring boot test  
• Testing restful services with mockmvc  
• Exploring actuator endpoints  
• Creating custom health indicators  
  
  
PHASE 3 - Advanced  
  
• Using @profile annotation  
• Configuring environment-specific beans  
• Switching profiles for different environments  
• Setting up a spring cloud project  
• Key components of spring cloud  
• Setting up eureka server  
• Registering microservices with eureka  
• Service discovery in action  
• Introduction to api gateway  
• Setting up spring cloud gateway  
• Configuring routes and filters  
• Setting up spring cloud config server  
• Managing configuration in a centralized repository  
• Configuring spring boot applications to use config server  
  
This roadmap covers everything for Springboot but unfortunately, this isn’t enough → You have to build high-quality projects and learn to explain these concepts💯  
  
You can learn it easily from experienced SDEs through 1:1 live sessions, get your doubts solved, give mocks and become 100% interview-ready!  
  
Book your FREE session - <https://lnkd.in/dfmDV_QD>

Clear Next Java Developer Interview  
  
𝗧𝗼𝗽𝗶𝗰 𝟭: 𝗣𝗿𝗼𝗷𝗲𝗰𝘁 𝗳𝗹𝗼𝘄 𝗮𝗻𝗱 𝗮𝗿𝗰𝗵𝗶𝘁𝗲𝗰𝘁𝘂𝗿𝗲   
 - Please tell me about your project and its architecture, Challenges faced?  
 - What was your role in the project? Tech Stack of project? why this stack?  
 - Problem you solved during the project? How collaboration within the team?  
 - What lessons did you learn from working on this project?  
 - If you could go back, what would you do differently in this project?  
  
𝗧𝗼𝗽𝗶𝗰 𝟮: 𝗖𝗼𝗿𝗲 𝗝𝗮𝘃𝗮  
 - String Concepts/Hashcode- Equal Methods  
 - Immutability  
 - OOPS concepts  
 - Serialization  
 - Collection Framework  
 - Exception Handling  
 - Multithreading  
 - Java Memory Model  
 - Garbage collection  
  
𝗧𝗼𝗽𝗶𝗰 𝟯: 𝗝𝗮𝘃𝗮-𝟴/𝗝𝗮𝘃𝗮-𝟭𝟭/𝗝𝗮𝘃𝗮𝟭𝟳  
 - Java 8 features  
 - Default/Static methods  
 - Lambda expression  
 - Functional interfaces  
 - Optional API  
 - Stream API  
 - Pattern matching  
 - Text block  
 - Modules  
  
𝗧𝗼𝗽𝗶𝗰 𝟰: 𝗦𝗽𝗿𝗶𝗻𝗴 𝗙𝗿𝗮𝗺𝗲𝘄𝗼𝗿𝗸, 𝗦𝗽𝗿𝗶𝗻𝗴-𝗕𝗼𝗼𝘁, 𝗠𝗶𝗰𝗿𝗼𝘀𝗲𝗿𝘃𝗶𝗰𝗲, 𝗮𝗻𝗱 𝗥𝗲𝘀𝘁 𝗔𝗣𝗜  
 - Dependency Injection/IOC, Spring MVC  
 - Configuration, Annotations, CRUD  
 - Bean, Scopes, Profiles, Bean lifecycle  
 - App context/Bean context  
 - AOP, Exception Handler, Control Advice  
 - Security (JWT, Oauth)  
 - Actuators  
 - WebFlux and Mono Framework  
 - HTTP methods  
 - JPA  
 - Microservice concepts  
 - Spring Cloud  
  
𝗧𝗼𝗽𝗶𝗰 𝟱: 𝗛𝗶𝗯𝗲𝗿𝗻𝗮𝘁𝗲/𝗦𝗽𝗿𝗶𝗻𝗴-𝗱𝗮𝘁𝗮 𝗝𝗽𝗮/𝗗𝗮𝘁𝗮𝗯𝗮𝘀𝗲 (𝗦𝗤𝗟 𝗼𝗿 𝗡𝗼𝗦𝗤𝗟)  
 - JPA Repositories  
 - Relationship with Entities  
 - SQL queries on Employee department  
 - Queries, Highest Nth salary queries   
 - Relational and No-Relational DB concepts  
 - CRUD operations in DB  
 - Joins, indexing, procs, function  
  
𝗧𝗼𝗽𝗶𝗰 𝟲: 𝗖𝗼𝗱𝗶𝗻𝗴  
 - DSA Related Questions  
 - Sorting and searching using Java API.  
 - Stream API coding Questions  
  
𝗧𝗼𝗽𝗶𝗰 𝟳: 𝗗𝗲𝘃𝗼𝗽𝘀 𝗾𝘂𝗲𝘀𝘁𝗶𝗼𝗻𝘀 𝗼𝗻 𝗱𝗲𝗽𝗹𝗼𝘆𝗺𝗲𝗻𝘁 𝗧𝗼𝗼𝗹𝘀  
 - These types of topics are mostly asked by managers or leads who are heavily working on it, That's why they may grill you on DevOps/deployment-related tools, You should have an understanding of common tools like Jenkins, Kubernetes, Kafka, Cloud, and all.  
  
𝗧𝗼𝗽𝗶𝗰𝘀 𝟴: 𝗕𝗲𝘀𝘁 𝗽𝗿𝗮𝗰𝘁𝗶𝗰𝗲  
 - The interviewer always wanted to ask about some design patterns, it maybe Normal design patterns like singleton, factory, or observer patterns to know that you can use these in coding.  
\

Clear all DSA rounds  
  
By mastering these 22 DSA patterns  
  
1. Fast and Slow Pointer  
- Cycle detection method  
- O(1) space efficiency  
- Linked list problems  
  
2. Merge Intervals  
- Sort and merge  
- O(n log n) complexity  
- Overlapping interval handling  
  
3. Sliding Window  
- Fixed/variable window  
- O(n) time optimization  
- Subarray/substring problems  
  
4. Islands (Matrix Traversal)  
- DFS/BFS traversal  
- Connected component detection  
- 2D grid problems  
  
5. Two Pointers  
- Dual pointer strategy  
- Linear time complexity  
- Array/list problems  
  
6. Cyclic Sort  
- Sorting in cycles  
- O(n) time complexity  
- Constant space usage  
  
7. In-place Reversal of Linked List  
- Reverse without extra space  
- O(n) time efficiency  
- Pointer manipulation technique  
  
8. Breadth First Search  
- Level-by-level traversal  
- Uses queue structure  
- Shortest path problems  
  
9. Depth First Search  
- Recursive/backtracking approach  
- Uses stack (or recursion)  
- Tree/graph traversal  
  
10. Two Heaps  
- Max and min heaps  
- Median tracking efficiently  
- O(log n) insertions  
  
11. Subsets  
- Generate all subsets  
- Recursive or iterative  
- Backtracking or bitmasking  
  
12. Modified Binary Search  
- Search in variations  
- O(log n) time  
- Rotated/specialized arrays  
  
13. Bitwise XOR  
- Toggle bits operation  
- O(1) space complexity  
- Efficient for pairing  
  
14. Top 'K' elements  
- Use heap/quickselect  
- O(n log k) time  
- Efficient selection problem  
  
15. K-way Merge  
- Merge sorted lists  
- Min-heap based approach  
- O(n log k) complexity  
  
16. 0/1 Knapsack (Dynamic Programming)  
- Choose or skip items  
- O(n \* W) complexity  
- Maximize value selection  
  
17. Unbounded Knapsack (Dynamic Programming)  
- Unlimited item choices  
- O(n \* W) complexity  
- Multiple item selection  
  
18. Topological Sort (Graphs)  
- Directed acyclic graph  
- Order dependency resolution  
- Uses DFS or BFS  
  
19. Monotonic Stack  
- Maintain increasing/decreasing stack  
- Optimized for range queries  
- O(n) time complexity  
  
20. Backtracking  
- Recursive decision-making  
- Explore all possibilities  
- Pruning with constraints  
  
21. Union Find  
- Track and merge connected components  
- Used for disjoint sets  
- Great for network connectivity  
  
22. Greedy Algorithm  
- Make locally optimal choices  
- Efficient for problems with optimal substructure  
- Covers tasks like activity selection, minimum coins  
  
𝗝𝗼𝗶𝗻 𝗺𝘆 𝗪𝗵𝗮𝘁𝘀𝗔𝗽𝗽 𝗖𝗵𝗮𝗻𝗻𝗲𝗹 - <https://lnkd.in/d4Ht9Ggj>  
  
Stay curious, keep learning, and keep sharing!

Give me 2 minutes, I will tell you  
  
How to clear coding rounds.  
  
Every Company First round is a coding round. So cracking this round is really important if you want to go to the next round. But we solve multiple problems but still we are not able to crack this round.  
  
Here are a few steps you can follow which will help you to crack this interview round:  
  
1. Understand the requirements and write down important keywords  
2. Reframe the question to make sure you understand it  
3. Ask edge cases in the input  
4. Explain the brute force solution in a minute  
5. Explain the optimal solution and discuss its asymptotic complexity quickly  
6. Ask if the optimal solution is acceptable  
7. Write code in the language you're comfortable with  
8. Dry run the code line by line to find mistakes  
9. Rewrite code to cover edge or base cases if necessary  
10. Review the solution's asymptotic complexity  
11. Ask if tests or concurrency cases are necessary  
12. Ask for the interviewer's feedback  
  
Think before you code, Learn patterns, not problems.  
  
𝟭. 𝗙𝗼𝗰𝘂𝘀 𝗼𝗻 𝗗𝗲𝗽𝘁𝗵 𝗼𝘃𝗲𝗿 𝗕𝗿𝗲𝗮𝗱𝘁𝗵:   
 - Don't solve 500 coding problems aimlessly. Master around 100 core problems deeply instead.  
 - 40 Problems on Array, Strings, LinkedList, Stack & Queue, Binary search, Trees, Graph, Sorting and Searching: <https://lnkd.in/djnaPkeD>  
 - 40 Problems on Dynamic Programming (DP), Backtracking, Hashing, Heap, Tries, and Greedy Algorithms: <https://lnkd.in/dF3h-Khk>  
  
𝟮. 𝗖𝗿𝗲𝗮𝘁𝗲 𝗮 𝗹𝗶𝘀𝘁 𝗼𝗳 𝗸𝗲𝘆 𝗾𝘂𝗲𝘀𝘁𝗶𝗼𝗻𝘀:   
 - Use resources like "Strivers A2Z DSA Sheet" by Raj Vikramaditya to curate around 100 core problems.  
 - <https://lnkd.in/dQMGy9zF> (Strivers)  
  
𝟯. 𝗠𝗮𝘀𝘁𝗲𝗿 𝗲𝗮𝗰𝗵 𝗱𝗮𝘁𝗮 𝘀𝘁𝗿𝘂𝗰𝘁𝘂𝗿𝗲:   
 - Understand and implement them by hand. Know how they work internally to ace interview questions.  
 - Fundamentals, Intermediate, Advance DSA topics: <https://lnkd.in/d4ws9xfr>  
  
𝟰. 𝗣𝗿𝗮𝗰𝘁𝗶𝗰𝗲 𝘄𝗶𝘁𝗵 𝗦𝗽𝗮𝗰𝗲𝗱 𝗥𝗲𝗽𝗲𝘁𝗶𝘁𝗶𝗼𝗻:   
 - Revisit problems after 3 days, a week, and 15 days. Break down solutions instead of rote memorization.  
 - 3:7:15 Rule for DSA: <https://lnkd.in/dW6a8wcg>  
  
𝟱. 𝗜𝗱𝗲𝗻𝘁𝗶𝗳𝘆 𝗿𝗲𝘂𝘀𝗮𝗯𝗹𝗲 𝘁𝗲𝗰𝗵𝗻𝗶𝗾𝘂𝗲𝘀 𝗮𝗻𝗱 𝗰𝗼𝗱𝗲 𝗯𝗹𝗼𝗰𝗸𝘀:   
 - Isolate common patterns like Binary Search or Depth First Search for focused practice.  
 - 20 DSA patterns: <https://lnkd.in/d9GCezMm>  
 - 14 problem solving patterns: <https://lnkd.in/daysVFSz>  
 - DSA questions patterns: <https://lnkd.in/d3rRHTfE>  
  
𝟲. 𝗘𝘅𝗽𝗮𝗻𝗱 𝗶𝗻𝘁𝗼 𝗕𝗿𝗲𝗮𝗱𝘁𝗵:   
 - Once you've mastered core problems and techniques, tackle a wider range of questions. Keep it realistic and relevant to interview scenarios.  
 - 16 Important algorithms problems: <https://lnkd.in/dfjm8ked>  
 - Tips to solve any DSA question by understanding patterns: <https://lnkd.in/d9GVyfBY>

Software Engineering is less about DSA and more about OOP and system design in terms of coding.  
  
PS: Checkout the first comment  
  
DSA helps with problem-solving, but OOPS and System Design make a software engineer stand out. These skills are key to building strong and scalable applications. They are also important for technical interviews and real-world projects. If you're preparing for interviews, here are some useful resources to help you:  
  
Top 16 OOP interview questions:  
 - <https://lnkd.in/gAiyTgv9>  
  
Topics you can't skip while interview preparation:  
 - <https://lnkd.in/gpS9P9_z>  
  
List of highly useful LeetCode posts on System Design interviews:  
 - <https://lnkd.in/gZHquQu3>  
  
System Design Cheatsheet:  
 - <https://lnkd.in/g-B8UTpK>  
  
How to approach system design interview:  
 - <https://lnkd.in/gQSVnzBF>  
  
23 System Design Principles:  
 - <https://lnkd.in/gYiimpx9>

Spring Boot Roadmap  
  
PHASE 1 - Basics  
  
• Creating a spring boot project using spring initializr  
• Maven and gradle build tools  
• Annotations  
• Profiles and environment-specific configurations  
• @getmapping, @postmapping, @putmapping, @deletemapping  
• Handling path variables and request parameters  
• Setting up database connection (h2, mysql, postgresql)  
• Using jparepository and crudrepository  
• Introduction to spring boot devtools  
• Enabling hot reloading  
• Spring batch, scheduling and cron expressions  
  
  
PHASE 2 - Intermediate  
  
• Using @controlleradvice and @exceptionhandler  
• Custom error responses and exception classes  
• Global exception handling  
• Basic authentication  
• Configuring security for apis  
• Implementing jwt (json web tokens) for stateless authentication  
• Introduction to hateoas  
• Versioning rest apis (uri, parameter, headers)  
• Unit testing with junit and mockito  
• Writing integration tests with spring boot test  
• Testing restful services with mockmvc  
• Exploring actuator endpoints  
• Creating custom health indicators  
  
  
PHASE 3 - Advanced  
  
• Using @profile annotation  
• Configuring environment-specific beans  
• Switching profiles for different environments  
• Setting up a spring cloud project  
• Key components of spring cloud  
• Setting up eureka server  
• Registering microservices with eureka  
• Service discovery in action  
• Introduction to api gateway  
• Setting up spring cloud gateway  
• Configuring routes and filters  
• Setting up spring cloud config server  
• Managing configuration in a centralized repository  
• Configuring spring boot applications to use config server  
  
  
PHASE 4 - Microservices  
  
• Introduction to inter-service communication  
• Using resttemplate for synchronous communication  
• Using feign client for simplified service calls  
• Setting up resilience4j  
• Configuring circuit breakers, retry in microservices  
• Introduction to distributed tracing  
  
This roadmap guide you through the essential skills and concepts.  
  
➤ Watch Resources : [Shrayansh Jain](https://www.linkedin.com/in/jainshrayansh/)   
  
1. Advantage of Spring boot over traditional Spring MVC:   
 - <https://lnkd.in/gbHYBdFJ>  
2. What is Layered Architecture:  
 - <https://lnkd.in/gdstJnet>  
3. Maven Lifecycle:  
 - <https://lnkd.in/gjYygKX7>  
4. Spring boot REST Api Annotations:  
 - <https://lnkd.in/gGYpinzs>  
5. Bean Lifecycle and loC:  
 - <https://lnkd.in/g6PCbj5U>  
6. Dependency Injection:  
 - <https://lnkd.in/gMBUFDxC>  
7. Bean Scope:  
 - <https://lnkd.in/gita68gc>  
8. How Profiling works in Spring boot:  
 - <https://lnkd.in/gNJ69SuD>  
9. How you will initialize bean conditionally:  
 - <https://lnkd.in/gB-_M5sN>  
10. Dynamic Initialization of Bean:  
 - <https://lnkd.in/gaerePEh>  
  
Here are some repositories that have helped me dive deeper into Spring Boot:  
 - <https://lnkd.in/dxFBncus>  
 - <https://lnkd.in/dB2i8Yxc>  
 - <https://lnkd.in/dnNCXVxy>  
 - <https://lnkd.in/diqd44Vr>  
 - <https://lnkd.in/dc2ADrZA>

Consistency  
  
• Studying daily for 1 hour is far better than studying for 5 hours on weekend  
  
• Learning new technology on a daily basis is far better than learning all at the time of need  
  
• Solving 1 problem daily on Leetcode is far better than solving 6 problems on Saturday and Sunday  
  
• Going to gym for 1 hour a day is far better than doing gym for 6 hours on weekends  
  
Overall, if you want something in your life then do that thing with consistency. You need to conquer 4 days in a week and thats enough to push you more towards your goal.

-===============

But, GRAPHS are TOUGH🙃  
  
Say no more.  
  
Graphs were always my fav topic, sharing some sheets for your ease.  
  
Go through these 5 LeetCode discussion. Graphs sorted:  
  
1. Graph algorithms + problems to practice:   
 - <https://lnkd.in/dCgzmMEr>  
  
2. Graph For Beginners (Problems, Pattern, Sample Solutions):   
 - <https://lnkd.in/dw78hTsC>  
  
3. Graph Common Question Patterns CheatSheet:   
 - <https://lnkd.in/d-ZYvZaq>  
  
4. Become Master In Graph:   
 - <https://lnkd.in/dWnfYDTu>  
  
5. Important Graph Questions:  
 - <https://lnkd.in/dJ6sYcMz>

Dynamic Programming (DP) was HARD  
  
until I found this method   
  
After solving 90+ problems in DP I have noticed that there are few patterns that can be found in different problems.   
  
I have concentrated my attention on Dynamic Programming cause it's one of the hardest topics in an interview prep. So I did some research on that and found the following patterns:  
   
𝟭. 𝗠𝗶𝗻𝗶𝗺𝘂𝗺 (𝗠𝗮𝘅𝗶𝗺𝘂𝗺) 𝗣𝗮𝘁𝗵 𝘁𝗼 𝗥𝗲𝗮𝗰𝗵 𝗮 𝗧𝗮𝗿𝗴𝗲𝘁:  
   
 - Given a target find minimum (maximum) cost / path / sum to reach the target.  
 - Approach: Choose minimum (maximum) path among all possible paths before the current state, then add value for the current state.  
 - Generate optimal solutions for all values in the target and return the value for the target (Top-down, Bottom-up).  
 - Problem List: <https://lnkd.in/dqbRTeNp>  
  
𝟮. 𝗗𝗶𝘀𝘁𝗶𝗻𝗰𝘁 𝗪𝗮𝘆𝘀  
  
 - Given a target find a number of distinct ways to reach the target.  
 - Approach: Sum all possible ways to reach the current state.  
 - Generate sum for all values in the target and return the value for the target (Top-down, Bottom-up).  
 - Problem List: <https://lnkd.in/dpjEQvap>  
  
𝟯. 𝗠𝗲𝗿𝗴𝗶𝗻𝗴 𝗜𝗻𝘁𝗲𝗿𝘃𝗮𝗹𝘀  
   
 - Given a set of numbers find an optimal solution for a problem considering the current number and the best you can get from the left and right sides.  
 - Approach: Find all optimal solutions for every interval and return the best possible answer.  
 - Get the best from the left and right sides and add a solution for the current position.  
 - Problem List: <https://lnkd.in/d-RqjvB9>  
  
𝟰. 𝗗𝗣 𝗼𝗻 𝗦𝘁𝗿𝗶𝗻𝗴𝘀  
  
 - General problem statement for this pattern can vary but most of the time you are given two strings where lengths of those strings are not big  
 - Approach: Most of the problems on this pattern require a solution that can be accepted in O(n^2) complexity.  
 - If you are given one string s the approach may vary  
 - Problem List: <https://lnkd.in/dedE5ezi>  
  
𝟱. 𝗗𝗲𝗰𝗶𝘀𝗶𝗼𝗻 𝗠𝗮𝗸𝗶𝗻𝗴  
  
 - The general problem statement for this pattern is the forgiven situation decides whether to use or not to use the current state. So, the problem requires you to make a decision at a current state.  
 - Approach: If you decide to choose the current value use the previous result where the value was ignored; vice-versa, if you decide to ignore the current value use the previous result where the value was used.  
 - Problem List: <https://lnkd.in/dedE5ezi>

ips to solve any DSA question by understanding patterns  
  
If the input array is sorted then  
 - Binary search  
 - Two pointers  
  
If asked for all permutations/subsets then   
 - Backtracking  
  
If given a tree then   
 - DFS  
 - BFS  
  
If given a graph then  
 - DFS  
 - BFS  
  
If given a linked list then   
 - Two pointers  
  
If recursion is banned then  
 - Stack  
  
If must solve in-place then  
 - Swap corresponding values  
 - Store one or more different values in the same pointer  
  
If asked for maximum/minimum subarray/ subset/options then   
 - Dynamic programming  
  
If asked for top/least K items then  
 - Heap  
 - QuickSelect  
  
If asked for common strings then  
 - Map  
 - Trie  
  
Else  
 - Map/Set for O(1) time & O(n) space  
 - Sort input for O(nlogn) time and O(1) space

Don’t pay for 𝗥𝗲𝗮𝗰𝘁 courses just because it is in demand.  
  
You can learn here for free  
  
I am right now giving interviews, trust me 50% of the questions are from React.  
  
𝐏𝐫𝐨𝐯𝐢𝐝𝐢𝐧𝐠 𝐭𝐡𝐞 𝐫𝐞𝐬𝐨𝐮𝐫𝐜𝐞𝐬 𝐭𝐨 𝐥𝐞𝐚𝐫𝐧 𝐑𝐞𝐚𝐜𝐭 𝐟𝐨𝐫 𝐟𝐫𝐞𝐞:  
  
YouTube channels to master basic and advanced concepts:  
  
1. [Hitesh Choudhary](https://www.linkedin.com/in/hiteshchoudhary/) : <https://lnkd.in/dVfsfQGh>  
2. [Piyush Agarwal](https://www.linkedin.com/in/piyush-eon/) : <https://lnkd.in/d-zFpJpf>  
3. [Tanish Garg](https://www.linkedin.com/in/tnshgarg/) : <https://lnkd.in/dtzUjJVs>  
4. 𝗖𝗼𝗱𝗲 𝗪𝗶𝘁𝗵 𝗛𝗮𝗿𝗿𝘆: <https://lnkd.in/dJXB6Cau>  
  
Free Courses to Learn React:  
  
1. React Fundamentals: <https://lnkd.in/dtmcUCSR>  
2. Frontend Dev Using React: <https://lnkd.in/daaMgaU5>  
3. React and Redux: <https://lnkd.in/d-sGMqKK>  
4. ReactJS Tutorials: <https://lnkd.in/dPbJavEv>  
5. Learn React: <https://lnkd.in/dp_AzGM3>  
  
Top Repositories to Master React:  
  
1. React in Patterns: <https://lnkd.in/dcPmuS2K>  
2. Real world react apps: <https://lnkd.in/dkSSQNFj>  
3. Beautiful react hooks: <https://lnkd.in/dnzVAxam>  
4. Awesome React: <https://lnkd.in/dQ5CnVeW>  
5. 30 Days of React: <https://lnkd.in/dJEmtZ6u>  
6. React coding challenge: <https://lnkd.in/djbZs9Yv>  
7. React Developer Roadmap: <https://lnkd.in/dH7aVHa8>  
  
Must-Follow YouTube Channels for Learning React  
  
1. [RoadsideCoder.com](https://www.linkedin.com/company/roadsidecoder/) : <https://lnkd.in/dtBth5PF>  
2. 𝗧𝗿𝗮𝘃𝗲𝗿𝘀𝘆 𝗠𝗲𝗱𝗶𝗮: <https://lnkd.in/dTwyd5ri>  
3. [JavaScript Mastery](https://www.linkedin.com/company/javascriptmastery/) : <https://lnkd.in/dsaTSadP>  
4. [Academind](https://www.linkedin.com/company/academind-pro/) : <https://lnkd.in/dCpsr92T>  
5. [Sonny Sangha](https://www.linkedin.com/in/saajansangha/) : <https://lnkd.in/dUz4jFr9>  
6. [freeCodeCamp](https://www.linkedin.com/company/free-code-camp/) : <https://lnkd.in/dru-qEDJ>  
7. Web Dev Simplified: <https://lnkd.in/d5pCtB6r>  
8. 𝗰𝗼𝗱𝗲𝘃𝗼𝗹𝘂𝘁𝗶𝗼𝗻: <https://lnkd.in/dyJVvjWa>  
  
React Cheat Sheets:  
  
1. <https://devhints.io/react>  
2. <https://lnkd.in/duVuBr-c>  
3. <https://lnkd.in/dT-eZqEH>  
4. <https://lnkd.in/dgN_2xiU>  
  
React Project you must do:   
   
1. <https://lnkd.in/dF--w_xU> (GitHub Repo)  
  
Finally React interview questions:  
  
1. <https://lnkd.in/dkAysuBi>  
2. <https://lnkd.in/d3CgksPV>  
3. <https://lnkd.in/dVwNkMRx>  
  
Follow this path and you will master the React and crack the interview.  
  
𝗝𝗼𝗶𝗻 𝗺𝘆 𝗪𝗵𝗮𝘁𝘀𝗔𝗽𝗽 𝗖𝗵𝗮𝗻𝗻𝗲𝗹 - <https://lnkd.in/d4Ht9Ggj>

Trust me, nobody will give you a job  
  
you have to deserve it and earn it.   
  
This ULTIMATE list of resources has already helped over 5,000 job-seekers land interview calls and secure jobs  
  
𝟱 𝗣𝗹𝗮𝘁𝗳𝗼𝗿𝗺𝘀 𝘁𝗼 𝗴𝗲𝘁 𝘆𝗼𝘂 𝗶𝗻𝘁𝗲𝗿𝘃𝗶𝗲𝘄 𝗰𝗮𝗹𝗹𝘀:  
  
1. [LinkedIn](https://www.linkedin.com/company/linkedin/) - <https://lnkd.in/gknBqMgE>  
2. [Naukri.com](https://www.linkedin.com/company/naukri.com/) - <https://www.naukri.com/>  
3. [foundit](https://www.linkedin.com/company/foundit-jobs/) - <https://lnkd.in/dTRYCQJE>  
4. [Indeed](https://www.linkedin.com/company/indeed-com/) - [www.indeed.com](http://www.indeed.com/)  
5. [Built In](https://www.linkedin.com/company/built-in/) - <https://builtin.com/>  
6. [ZipRecruiter](https://www.linkedin.com/company/ziprecruiter/) - [www.ziprecruiter.com](http://www.ziprecruiter.com/)  
  
𝟮𝟱 𝘄𝗲𝗯𝘀𝗶𝘁𝗲𝘀 𝗳𝗼𝗿 𝘆𝗼𝘂𝗿 𝗿𝗲𝗺𝗼𝘁𝗲 𝗝𝗢𝗕:  
  
[We Work Remotely](https://www.linkedin.com/company/we-work-remotely/)  
[Remote.co](https://www.linkedin.com/company/remote-co/)  
[FlexJobs](https://www.linkedin.com/company/flexjobs-com/)  
[Freelancer.com](https://www.linkedin.com/company/freelancer-com/)  
[AngelList](https://www.linkedin.com/company/angellist/)  
[Sorce](https://www.linkedin.com/company/sorcejobs/)  
[JustRemote](https://www.linkedin.com/company/justremote/)  
[RemoteOK](https://www.linkedin.com/company/remoteok/)  
[Upwork](https://www.linkedin.com/company/upwork/)  
[Remotive](https://www.linkedin.com/company/remotive.io/)  
[Virtual Vocations, Inc.](https://www.linkedin.com/company/virtual-vocations/)  
[Github Jobs](https://www.linkedin.com/company/github-jobs/)  
[Jobspresso](https://www.linkedin.com/company/jobspresso/)  
[PowerToFly](https://www.linkedin.com/company/powertofly/)  
[SimplyHired](https://www.linkedin.com/company/simply-hired/)  
[Working Nomads](https://www.linkedin.com/company/working-nomads/)  
[Pangian](https://www.linkedin.com/company/pangian/)  
[Crossover](https://www.linkedin.com/company/crossover/)   
[SkipTheDrive](https://www.linkedin.com/company/skipthedrive/)   
[Outsourcely](https://www.linkedin.com/company/outsourcely/)  
[Jobbatical](https://www.linkedin.com/company/jobbatical/)  
[Remotive](https://www.linkedin.com/company/remotive.io/)  
  
𝟵 𝘄𝗲𝗯𝘀𝗶𝘁𝗲𝘀 𝘁𝗼 𝗺𝗮𝗸𝗲 𝘆𝗼𝘂𝗿 𝗿𝗲𝘀𝘂𝗺𝗲 𝗯𝗲𝘁𝘁𝗲𝗿:  
  
1. Jobscan - [jobscan.co](http://jobscan.co/)  
2. Resumeworded - <https://resumeworded.com/>  
3. Kickresume - <https://lnkd.in/dZxmaKPb>  
4. Beamjobs- <https://www.beamjobs.com/>  
5. Overleaf - <https://www.overleaf.com/>  
6. Wonsulting - <https://lnkd.in/d_wvDvXE>  
7. Zety - <https://zety.com/>  
8. Teal - [www.tealhq.com](http://www.tealhq.com/)  
9. Rezi - [www.rezi.ai](http://www.rezi.ai/)  
  
𝗥𝗲𝘀𝗼𝘂𝗿𝗰𝗲𝘀 𝘁𝗼 𝗵𝗲𝗹𝗽 𝘆𝗼𝘂 𝗻𝗲𝘁𝘄𝗼𝗿𝗸 𝗮𝗻𝗱 𝗴𝗲𝘁 𝗿𝗲𝗳𝗲𝗿𝗿𝗮𝗹𝘀:  
  
1. Luma - <https://lu.ma/>  
2. Meetup - [www.meetup.com](http://www.meetup.com/)  
3. Fishbowl - [www.fishbowlapp.com](http://www.fishbowlapp.com/)  
4. Blind - [www.teamblind.com](http://www.teamblind.com/)

Dont confuse to learn NodeJs  
  
Learn this concept to be proficient in NodeJs  
  
➤ Phase 1 - JavaScript Basics  
  
 𝗝𝗮𝘃𝗮𝗦𝗰𝗿𝗶𝗽𝘁 𝗖𝗼𝗻𝗰𝗲𝗽𝘁𝘀:  
 - Lexical Structure  
 - Expressions  
 - Data Types  
 - Classes  
 - Variables  
 - Functions  
 - this Keyword  
 - Arrow Functions  
 - Loops  
 - Scopes  
 - Arrays  
 - Template Literals  
  
 𝗔𝘀𝘆𝗻𝗰𝗵𝗿𝗼𝗻𝗼𝘂𝘀 𝗣𝗿𝗼𝗴𝗿𝗮𝗺𝗺𝗶𝗻𝗴:  
 - Callbacks  
 - Timers  
 - Promises  
 - Async & Await  
  
 𝗖𝗼𝗿𝗲 𝗖𝗼𝗻𝗰𝗲𝗽𝘁𝘀:  
 - Closures  
 - The Event Loop  
  
  
➤ Phase 2 - Core NodeJS Concepts  
  
 𝗡𝗼𝗱𝗲𝗝𝗦 𝗙𝘂𝗻𝗱𝗮𝗺𝗲𝗻𝘁𝗮𝗹𝘀:  
 - Global Objects in NodeJS (\_\_dirname, \_\_filename, module)  
 - Modules (CommonJS Modules and ES6 Modules import/export)  
 - NPM (Node Package Manager)  
  
 𝗙𝗶𝗹𝗲 𝗦𝘆𝘀𝘁𝗲𝗺 (𝗳𝘀) 𝗠𝗼𝗱𝘂𝗹𝗲:  
 - Reading/Writing Files  
 - File Streams  
 - Asynchronous vs. Synchronous File Operations  
  
 𝗘𝘃𝗲𝗻𝘁𝘀 & 𝗧𝗵𝗲 𝗘𝘃𝗲𝗻𝘁 𝗘𝗺𝗶𝘁𝘁𝗲𝗿:  
 - Custom events  
 - Handling asynchronous events  
  
 𝗛𝗧𝗧𝗣 𝗠𝗼𝗱𝘂𝗹𝗲:  
 - Creating an HTTP server (Handling req and res)  
  
 𝗕𝘂𝗳𝗳𝗲𝗿 & 𝗦𝘁𝗿𝗲𝗮𝗺𝘀:  
 - Buffer: Handling binary data  
 - Streams: Readable, Writable, Duplex, and Transform streams  
 - Stream handling (piping, chunking)  
  
  
➤ Phase 3 - Advanced NodeJS Concepts  
  
 𝗘𝘅𝗽𝗿𝗲𝘀𝘀 𝗙𝗿𝗮𝗺𝗲𝘄𝗼𝗿𝗸:  
 - Routing  
 - Middleware  
 - Handling Req, Res & Error   
 - Serving Static Files  
 - Template Engines (e.g., Pug, EJS)  
 - RESTful APIs  
  
 𝗗𝗮𝘁𝗮𝗯𝗮𝘀𝗲 𝗜𝗻𝘁𝗲𝗴𝗿𝗮𝘁𝗶𝗼𝗻:  
 - MongoDB with Mongoose (CRUD operations, Schemas & Models)  
 - SQL (MySQL, PostgreSQL)  
 - Using Sequelize  
 - Redis  
  
  
➤ Phase 4 - Real-time Applications  
  
 𝗪𝗲𝗯𝗦𝗼𝗰𝗸𝗲𝘁𝘀 𝘄𝗶𝘁𝗵 𝗦𝗼𝗰𝗸𝗲𝘁.𝗶𝗼:  
 - Real-time data communication  
  
 𝗘𝘃𝗲𝗻𝘁-𝗗𝗿𝗶𝘃𝗲𝗻 𝗔𝗿𝗰𝗵𝗶𝘁𝗲𝗰𝘁𝘂𝗿𝗲:  
 - Understanding event-driven development  
  
  
➤ Phase 5 - Security Practices  
  
 - Environment Variables & Configuration (.env)  
 - Input Validation (Joi)  
 - Preventing Common Vulnerabilities (XSS, CSRF, SQL Injection)  
 - Authentication & Authorization  
 - JWT  
 - OAuth (Session-based authentication)  
 - Data Encryption  
 - Helmet for securing HTTP headers  
  
  
Here are some repositories that have helped me dive deeper into NodeJS:  
  
- <https://lnkd.in/dgwQ3Yxa>  
- <https://lnkd.in/daGdthwe>  
- <https://lnkd.in/d3FhvTPj>

Harsh Truth but True..  
  
During Interview: DSA, System Design, JAVA, Spring boot, Javascript, SQL, Kafka, etc..  
  
Actual Job: using IF/ELSE  
  
Still we have to keep up-skill because competition is growing and now in-depth knowledge is in demand. Sharing below resources which can help you build solid understanding of below topics.  
  
Namaste JavaScript by [Akshay Saini 🚀](https://www.linkedin.com/in/akshaymarch7/):  
 - <https://lnkd.in/gYVUUp25>  
  
LLD by [Shrayansh Jain](https://www.linkedin.com/in/jainshrayansh/):  
 - <https://lnkd.in/gFxtSxJU>  
  
HLD by [Shrayansh Jain](https://www.linkedin.com/in/jainshrayansh/):  
 - <https://lnkd.in/gtgR2FEQ>  
  
System Design by [Gaurav Sen](https://www.linkedin.com/in/gkcs/):  
 - <https://lnkd.in/gF8Q4mef>  
  
Designing Microservices by [Arpit Bhayani](https://www.linkedin.com/in/arpitbhayani/):  
 - <https://lnkd.in/gebTXbDC>  
  
Frontend Machine Coding by [Piyush Agarwal](https://www.linkedin.com/in/piyush-eon/):  
 - <https://lnkd.in/gexXg8eG>  
  
Frontend System Design by [Piyush Agarwal](https://www.linkedin.com/in/piyush-eon/):  
 - <https://lnkd.in/g3C6C3eM>  
  
SQL by [Riti Kumari](https://www.linkedin.com/in/riti2409/):  
 - <https://lnkd.in/gA85-Zev>  
  
Java by [Shrayansh Jain](https://www.linkedin.com/in/jainshrayansh/):  
 - <https://lnkd.in/g2qhipNK>  
  
Spring boot by [Shrayansh Jain](https://www.linkedin.com/in/jainshrayansh/):  
 - <https://lnkd.in/gaDXrjTc>  
  
A2Z DSA sheet by [Raj Vikramaditya](https://www.linkedin.com/in/rajstriver/):  
 - <https://lnkd.in/dQMGy9zF>  
  
Kafka resources   
 - <https://lnkd.in/d_Y2BcSf>

Kafka is Complicated,  
  
Until you go through these resources,  
  
𝗙𝗿𝗲𝗲 𝗞𝗮𝗳𝗸𝗮 𝗥𝗲𝘀𝗼𝘂𝗿𝗰𝗲𝘀:   
  
1. Apache Kafka crash course with Spring Boot 3.0.x by Java Techie  
 - <https://lnkd.in/dP3wziT6>  
2. Apache Kafka with Spring Boot crash course by Daily Code Buffer  
 - <https://lnkd.in/dhz-kJB5>  
3. All about Kafka by Learning Tutorial  
 - <https://lnkd.in/drZ_hbq2>  
4. Kafka Tutorial by Intellipaat  
 - <https://lnkd.in/dSr77EcE>  
5. Kafka crash course by Hussein Nasser   
 - <https://lnkd.in/dDND2e6z>  
  
𝗛𝗼𝘄 𝗞𝗮𝗳𝗸𝗮 𝗶𝗺𝗽𝗹𝗲𝗺𝗲𝗻𝘁𝗲𝗱 𝗮𝘁 𝗧𝗲𝗰𝗵 𝗚𝗶𝗮𝗻𝘁𝘀:  
  
1. LinkedIn's Kafka Journey   
 - <https://lnkd.in/d_QijMDk>  
2. Pinterest's Kafka at Scale  
 - <https://lnkd.in/dvV-8hun>  
3. Why Trello Chose Kafka  
 - <https://lnkd.in/gJhvX-dg>  
4. Salesforce's Kafka-Inspired Architecture  
 - <https://lnkd.in/gBH3bwGq>  
5. NYT's Publishing with Kafka  
 - <https://lnkd.in/gqcwF_zP>  
6. Yelp's Billions of Messages  
 - <https://lnkd.in/g7_fcfB7>  
7. Criteo's Kafka Upgrade   
 - <https://lnkd.in/gwGx8wvq>  
8. Shopify's Kafka on Kubernetes  
 - <https://lnkd.in/gSdHqzb4>  
9. Yelp's Zero-Downtime Zookeeper Migration  
 - <https://lnkd.in/gHdphiQY>  
10. Uber's Kafka Reprocessing  
 - <https://lnkd.in/gti2xZuR>  
 - <https://lnkd.in/gDF2S-vX>  
11. Dropbox's Kafka Throughput Limits  
 - <https://lnkd.in/gqpwjHzv>  
12. Walmart's Cost Orchestration  
 - <https://lnkd.in/gdtc5Az9>  
13. PayPal's Kafka Scaling  
 - <https://lnkd.in/gSxAVa89>  
  
𝗞𝗮𝗳𝗸𝗮 𝗜𝗻𝘁𝗲𝗿𝘃𝗶𝗲𝘄 𝗤𝘂𝗲𝘀𝘁𝗶𝗼𝗻𝘀:  
  
Basic Level:  
1. What is Apache Kafka, and what are its core components?  
2. Explain the difference between a topic, partition, and segment.  
3. How does Kafka ensure message ordering?  
4. What is a consumer group in Kafka?  
  
Intermediate Level:  
5. How does Kafka achieve fault tolerance?   
6. Explain Kafka's partitioning strategy and how it impacts performance.  
7. Describe Kafka's consumer offset management.  
  
Advanced Level:  
8. Explain the concept of exactly-once semantics (EOS) in Kafka.  
9. How would you monitor and optimize Kafka performance in a production environment?   
10. How would you design a Kafka-based system to guarantee data consistency in the event of node failures?  
  
Over the years, I've explored Kafka's powerful capabilities in various projects, and I know firsthand how crucial the right resources are when implementing it from scratch.   
  
I’ve gathered some excellent free resources and insightful case studies that showcase how tech giants are using Kafka at scale. Plus, I’m shared a few key interview questions that helped me prepare and excel.

16 Programming Project Ideas That Will Make Your Resume Stand Out in 2025  
  
  
1. Airline Reservation System:   
 - <https://lnkd.in/dRFK2vPh>  
  
2. Data Visualization Software:   
 - <https://lnkd.in/dVWVU8xn>  
  
3. Electricity Billing System:  
 - <https://lnkd.in/dekycNXQ>  
  
4. E-Healthcare Management System:   
 - <https://lnkd.in/dEkE2raN>  
  
5. Email Client Software:   
 - <https://lnkd.in/d_qz7U9E>  
  
6. Library Management System:   
 - <https://lnkd.in/dY7bDjFn>  
  
7. Network Packet Sniffer:   
 - <https://lnkd.in/dXPtyzz4>  
  
8. Online Bank Management System:   
 - <https://lnkd.in/d4Qzy8fN>  
  
9. Online Medical Management System:  
 - <https://lnkd.in/dHciHGGz>  
  
10. Online Quiz Management System:   
 - <https://lnkd.in/djKs3DJq>  
  
11. Online Survey System:   
 - <https://lnkd.in/dw9Cmhix>  
  
12. RSS Feed Reader:   
 - <https://lnkd.in/dupDQPnG>  
  
13. Smart City Project:   
 - <https://lnkd.in/d3YT36aJ>  
  
14. Stock Management System:   
 - <https://lnkd.in/dTb3hikj>  
  
15. Supply Chain Management System:   
 - <https://lnkd.in/dAzJthMQ>  
  
16. Virtual Private Network:   
 - <https://lnkd.in/dyEcgrFC>  
  
Save for Yourself & Share to help others

DSA on Sunday!  
  
Save this post, you will need it during interviews.  
  
40 most asked DSA questions to clear your next interview:  
  
1. Reverse a linked list.  
2. Find the middle element of a linked list.  
3. Implement a stack using arrays/linked list.  
4. Implement a queue using arrays/linked list.  
5. Find the factorial of a number using recursion.  
6. Implement binary search in an array.  
7. Find the largest/smallest element in an array.  
8. Implement merge sort.  
9. Implement quick sort.  
10. Detect a cycle in a linked list.  
11. Find the intersection point of two linked lists.  
12. Check if a binary tree is a binary search tree (BST).  
13. Print all leaf nodes of a binary tree.  
14. Reverse a binary tree.  
15. Find the height of a binary tree.  
16. Implement depth-first search (DFS) on a graph.  
17. Implement breadth-first search (BFS) on a graph.  
18. Check if a graph is connected.  
19. Implement Dijkstra's algorithm for shortest path.  
20. Implement Prim's algorithm for minimum spanning tree.  
21. Implement Kruskal's algorithm for minimum spanning tree.  
22. Find the longest common subsequence of two strings.  
23. Find the longest increasing subsequence of an array.  
24. Implement the Knuth–Morris–Pratt (KMP) algorithm for string matching.  
25. Implement the Rabin-Karp algorithm for string matching.  
26. Check if a string is a palindrome.  
27. Check if two strings are anagrams of each other.  
28. Find the next greater element in an array.  
29. Find the kth smallest/largest element in an array.  
30. Find the median of two sorted arrays.  
31. Implement a trie (prefix tree).  
32. Find all subsets of a set.  
33. Find all permutations of a string.  
34. Implement the Josephus Problem.  
35. Implement an LRU (Least Recently Used) Cache.  
36. Find the longest palindrome substring in a string.  
37. Implement a priority queue.  
38. Implement a hashmap (dictionary).  
39. Count the number of inversions in an array.  
40. Find the shortest path in a maze.

Java Ultimate Resources Sheet  
  
1. Java Basic:   
 - Data types, keywords, variables, operator, loops: <https://lnkd.in/dr8fYigm>  
 - String: <https://lnkd.in/dA2nn79A>  
 - Array: <https://lnkd.in/dMDTpPm2>  
  
2. Java OOPs:   
 - abstraction, encapsulation, inheritance, polymorphism: <https://lnkd.in/djNeVRex>  
 - Constructor: <https://lnkd.in/d_W6Dd8D>  
  
3. Java Collections:  
 - Map, Queue, List, Set: <https://lnkd.in/dJPdQhXK>  
 - <https://lnkd.in/dtdQsFgj>  
  
4. JVM architecture:  
 - <https://lnkd.in/dEEAasFa>  
  
5. Java Memory Model:  
 - <https://lnkd.in/dV_WAEHr>  
 - <https://lnkd.in/dADTTYJG>  
 - <https://lnkd.in/dr8XmbrD>  
 - <https://lnkd.in/d7nkrKii>  
  
6. Garbage Collections:  
 - <https://lnkd.in/d--bgKvK>  
  
7. Exception Handling:  
 - <https://lnkd.in/d7pgMRCJ>  
 - <https://lnkd.in/dE7MCH8j>  
  
8. Generics:   
 - <https://lnkd.in/dwD7Bzss>  
 - <https://lnkd.in/d9Bb9fb7>  
  
9. Serialization:  
 - <https://lnkd.in/dGm2mjwy>  
 - <https://lnkd.in/dFNepNW9>  
  
10. Reflection API:  
 - <https://lnkd.in/dBXdHSD7>  
 - <https://lnkd.in/dXsfPWUp>  
  
11. File Handling:  
 - <https://lnkd.in/d8g8f45b>  
 - <https://lnkd.in/dyCwzkhp>  
  
12. Java Functional Programming:   
 - <https://lnkd.in/dPk5vszt>  
 - <https://lnkd.in/dVzpUvXn>  
 - <https://lnkd.in/d_7pzZuh>  
 - <https://lnkd.in/dS9x8SnA>  
  
13. Java multi-threading:   
 - <https://lnkd.in/ddKr9rgz>  
 - <https://lnkd.in/d6uFXkce>  
 - <https://lnkd.in/dK32XAuV>  
 - <https://lnkd.in/dP39ZYgT>  
  
14. Java Regex:   
 - <https://lnkd.in/dRJf5nX4>  
  
15. Java 8 Features:  
 - <https://lnkd.in/dMqx4dve>  
 - <https://lnkd.in/dfFkbkmc>  
  
16. Java All Remaining Features:   
 - <https://lnkd.in/dSc6MqfA>  
 - <https://lnkd.in/drMWeGKw>