The Ultimate Full Stack Interview Q/A

🔹 Section 1: Frontend Basics

Q1. Difference between React, Angular, and Vue?
Answer: React is a flexible JavaScript library focused on UI, Angular is a full-fledged framework with built-in tools and TypeScript support, and Vue is lightweight, progressive, and easy to integrate. Choice depends on project size, ecosystem, and team expertise.
💡 *Tip: Be ready to justify your choice with a project example.*

Q2. What are React Hooks?
Answer: Hooks let functional components use state and lifecycle features without classes. Common hooks include useState, useEffect, and useContext. They simplify code, improve reusability, and enable custom hook creation.
💡 *Tip: Show that you know at least one real-world use of custom hooks.*

Q3. How does the Virtual DOM work?
Answer: The Virtual DOM is an in-memory representation of the real DOM. React updates the virtual copy first, calculates the differences (diffing), and applies only necessary updates to the actual DOM for better performance.
💡 *Tip: Mention reconciliation to impress the interviewer.*

Q4. State vs Props in React?
Answer: State is local and managed within the component, while props are read-only data passed from parent to child. Together, they help control component behavior and UI rendering.
💡 *Tip: Use a parent-to-child data flow example when explaining.*

Q5. How to optimize frontend performance?
Answer: Techniques include lazy loading, code splitting, memoization

(React.memo), reducing re-renders, and caching static resources. Tools like Lighthouse and Chrome DevTools help identify bottlenecks.
💡 *Tip: Always back your answer with one optimization you've implemented.*

🔹 Section 2: Backend Basics

Q6. REST vs GraphQL?
Answer: REST exposes multiple endpoints with fixed responses, while GraphQL provides a single endpoint where clients can query only the data they need. REST is simpler for small APIs, but GraphQL improves efficiency in complex applications.
💡 *Tip: Mention trade-offs like caching being easier in REST.*

Q7. What is middleware in Node.js?
Answer: Middleware are functions that run between a request and response cycle. They're used for logging, authentication, error handling, and request validation. In Express.js, they're added with app.use().
💡 *Tip: Be prepared to write a simple middleware snippet if asked.*

Q8. Explain event-driven architecture in Node.js.
Answer: Node.js uses an event loop and non-blocking I/O to handle multiple requests asynchronously. Instead of waiting, it registers callbacks or promises, making it efficient for real-time and high-concurrency apps.
💡 *Tip: Use chat or streaming apps as practical examples.*

Q9. SQL vs NoSQL — when to use which?
Answer: SQL databases are relational and enforce structured schemas, making them ideal for transactional systems. NoSQL databases support unstructured or semi-structured data, offering flexibility and scalability for apps like IoT or social media.
💡 *Tip: Give real-world examples — Banking (SQL), IoT (NoSQL).*

Q10. How do you handle authentication & authorization?
Answer: Authentication verifies identity (e.g., with JWT, OAuth), while authorization manages access rights (RBAC, ABAC). Best practices include hashing passwords, role-based policies, and enforcing least privilege.
💡 *Tip: Always distinguish clearly between AuthN and AuthZ.*

🔹 Section 3: Databases & APIs

Q11. What is database indexing?
Answer: Indexing creates a data structure that speeds up query lookups at the cost of additional storage and slower writes. Common implementations use B-trees or hash tables.
💡 *Tip: Know when indexing hurts performance — e.g., frequent inserts.*

Q12. Normalization vs Denormalization?
Answer: Normalization removes redundancy and ensures data integrity, while denormalization introduces redundancy for faster queries. They balance consistency vs performance.
💡 *Tip: Use e-commerce DB examples when answering.*

Q13. What is caching in full stack apps?
Answer: Caching temporarily stores frequently accessed data in memory or edge servers. It reduces load on the database and improves response times for end users.
💡 *Tip: Mention cache invalidation as the hardest challenge.*

Q14. What is API rate limiting?
Answer: Rate limiting restricts the number of API requests allowed per client in a given timeframe. It prevents abuse, protects servers, and ensures fair usage.
💡 *Tip: Know algorithms like token bucket and sliding window.*

Q15. Explain WebSockets.
Answer: WebSockets allow persistent, bi-directional communication between client and server over a single connection. They're used for real-time apps like chats, stock tickers, and live notifications.
💡 *Tip: Differentiate WebSockets from polling and SSE.*

🔹 Section 4: DevOps for Full Stack

Q16. What is CI/CD for full stack apps?
Answer: CI ensures code changes are tested and merged frequently, while CD automates deployment to staging or production. Together, they reduce bugs and accelerate delivery.
💡 *Tip: Name a tool you've personally used like GitHub Actions or Jenkins.*

Q17. How to deploy MERN stack on AWS?
Answer: Use EC2/ECS for backend, S3 + CloudFront for frontend, and MongoDB Atlas or RDS for the database. Docker can containerize services for portability.
💡 *Tip: Always mention scalability with load balancers or auto-scaling.*

Q18. What is Docker in full stack projects?
Answer: Docker packages applications and dependencies into containers that run consistently across environments. It eliminates "works on my machine" issues.
💡 *Tip: Be able to describe a simple Dockerfile you've used.*

Q19. How do you manage environment variables securely?
Answer: Locally use .env files, but in production use secrets managers like AWS Secrets Manager, HashiCorp Vault, or Azure Key Vault. This prevents sensitive data leaks.
💡 *Tip: Stress never committing .env files to Git.*

Q20. What is a reverse proxy (NGINX)?
Answer: A reverse proxy sits in front of servers, handling requests, SSL termination, caching, and load balancing. It improves performance and hides backend details.
💡 *Tip: Mention NGINX and HAProxy as popular examples.*

🔷 Section 5: System Design for Full Stack

Q21. What is load balancing?
Answer: Load balancing distributes traffic across multiple servers to improve availability, scalability, and fault tolerance. Can be at L4 (TCP) or L7 (HTTP).
💡 *Tip: Prepare an example like AWS ELB or NGINX.*

Q22. What is a CDN?
Answer: A Content Delivery Network caches static files at edge locations globally, reducing latency and offloading origin servers. Ideal for images, videos, and scripts.
💡 *Tip: Say it improves both speed and reliability.*

Q23. Monolith vs Microservices?
Answer: Monoliths are built as a single unit, while microservices break apps into independent services. Microservices improve scalability but add complexity in deployment and communication.
💡 *Tip: Show awareness of trade-offs, not just benefits.*

Q24. How to scale a full stack app?
Answer: Vertical scaling adds more resources to a single server, while horizontal scaling adds more servers with load balancers. Use caching, DB replication, and auto-scaling policies.
💡 *Tip: Mention database bottlenecks and solutions.*

Q25. Explain CAP theorem.
Answer: In distributed systems, you can only guarantee two out of

Consistency, Availability, and Partition tolerance. For example, DynamoDB is AP, MongoDB is CP.
💡 *Tip: Real-world mapping impresses interviewers.*

🔹 Section 6: Security & Best Practices

Q26. What is CORS?
Answer: Cross-Origin Resource Sharing is a mechanism that controls which domains can request resources from your API. It prevents unauthorized cross-origin requests.
💡 *Tip: Differentiate preflight (OPTIONS) vs simple requests.*

Q27. How do you prevent SQL Injection?
Answer: Use parameterized queries, ORM frameworks, and input validation. Never concatenate user input into queries. Proper sanitization ensures security.
💡 *Tip: Mention prepared statements explicitly.*

Q28. CSRF vs XSS?
Answer: CSRF tricks authenticated users into executing unwanted actions, while XSS injects malicious scripts into trusted websites. Both compromise security differently.
💡 *Tip: Add mitigation strategies — CSRF tokens, CSP, escaping output.*

Q29. What is JWT?
Answer: JSON Web Tokens are signed tokens used for stateless authentication. They contain claims and are validated on each request without storing session data.
💡 *Tip: Emphasize using short expiry + refresh tokens.*

Q30. OWASP Top 10 vulnerabilities?
Answer: A list of the most critical security risks, including Injection, Broken Authentication, Sensitive Data Exposure, XSS, Security

Misconfigurations, and more.
💡 *Tip: Be able to name at least 4–5 confidently.*