Backend Development Interview Q&As

🔹 Section 1: Backend Fundamentals

Q1. What is backend development?
Answer: Backend development manages server-side logic, databases, and APIs that power applications. It ensures smooth data flow between the frontend and storage layers.
💡 **Tip:** Explain using a simple example like user login or payment flow.

Q2. Difference between monolithic and microservices architecture?
Answer: Monolithic is a single deployable unit, easier for small apps but harder to scale. Microservices split apps into independent services, offering flexibility and fault isolation.
💡 **Tip:** Interviewers like when you discuss trade-offs (simplicity vs scalability).

Q3. What is an API, and how does REST differ from GraphQL?
Answer: APIs let systems communicate. REST uses resource-based endpoints with fixed responses, while GraphQL allows flexible queries to fetch only needed data.
💡 **Tip:** Mention REST is widely adopted, GraphQL is efficient for modern apps.

Q4. What is middleware in backend frameworks (e.g., Express.js)?
Answer: Middleware functions process requests/responses before reaching the final handler. They're used for authentication, logging, and error handling.
💡 **Tip:** Always give a practical example like auth middleware in Express.

Q5. What is synchronous vs asynchronous programming?
Answer: Synchronous code executes sequentially and blocks the

process. Asynchronous code executes non-blocking operations, improving scalability and performance.
💡 **Tip:** Relating async to APIs/DB queries makes your answer stronger.

🔹 Section 2: Databases & Caching

Q6. SQL vs NoSQL – when to use what?
Answer: SQL is structured, relational, and ACID-compliant, ideal for transactions. NoSQL is flexible, scalable, and good for large unstructured data.
💡 **Tip:** Use e-commerce example: SQL for orders, NoSQL for product catalog.

Q7. Difference between MongoDB and PostgreSQL?
Answer: MongoDB is a NoSQL document DB with flexible schemas. PostgreSQL is a relational DB with strong consistency and advanced SQL support.
💡 **Tip:** Highlight MongoDB = scalability, PostgreSQL = complex queries.

Q8. What is database indexing?
Answer: Indexes are data structures that speed up queries by reducing the need for full table scans. They improve performance at the cost of extra storage.
💡 **Tip:** Mention "indexes improve reads but slow down writes."

Q9. What is database sharding?
Answer: Sharding splits large datasets across multiple servers to balance load and improve performance. Each shard holds a subset of the data.
💡 **Tip:** Say “sharding helps with scalability but adds complexity.”

Q10. Why use caching (Redis, Memcached)?
Answer: Caching stores frequently accessed data in memory to

reduce DB load and response time. Tools like Redis enable high-speed lookups.
💡 **Tip:** Use “login sessions stored in Redis” as an example.

🔹 Section 3: Authentication & Security

Q11. Difference between authentication and authorization?
Answer: Authentication verifies identity (who you are). Authorization defines permissions (what you can access).
💡 **Tip:** Use "username/password = authN, role-based access = authZ."

Q12. What is JWT (JSON Web Token)?
Answer: JWT is a compact, stateless token format used for authentication. It includes claims and signatures for verification.
💡 **Tip:** Mention JWT is widely used in microservices.

Q13. What is OAuth2.0, and where is it used?
Answer: OAuth2 is a framework for delegated access, allowing apps to access resources without exposing credentials. It’s common in social logins.
💡 **Tip:** Say “OAuth2 = login with Google/Facebook” for clarity.

Q14. What is CORS, and why does it matter?
Answer: CORS is a browser security feature that restricts cross-origin requests. It prevents unauthorized data access from different domains.
💡 **Tip:** Mention “Access-Control-Allow-Origin” header.

Q15. How do you secure sensitive data in backend apps?
Answer: Use encryption, hashing (bcrypt for passwords), environment variables, and secret managers.

💡 **Tip:** Avoid saying "store plain text passwords" — that’s a red flag.

🔹 Section 4: Performance & Scaling

Q16. Vertical vs horizontal scaling?
Answer: Vertical scaling = adding resources to one server; Horizontal scaling = adding more servers.
💡 **Tip:** Always say horizontal is more cloud-native.

Q17. What is load balancing?
Answer: Load balancing distributes traffic across multiple servers to ensure availability and fault tolerance.
💡 **Tip:** Mention both hardware (F5) and software (NGINX, ELB).

Q18. What is rate limiting?
Answer: Rate limiting restricts API calls per user/timeframe to prevent abuse, DDoS, or server overload.
💡 **Tip:** Mention APIs like Twitter/Stripe enforce rate limits.

Q19. Synchronous APIs vs event-driven architecture?
Answer: Sync APIs follow request/response. Event-driven uses message queues for async communication, improving scalability.
💡 **Tip:** Example: payment service → event-driven with Kafka.

Q20. What is database connection pooling?
Answer: Pooling reuses existing DB connections to reduce overhead and improve performance.
💡 **Tip:** Mention popular ORMs (Sequelize, Hibernate) handle this automatically.

🔹 Section 5: System Design Basics

Q21. What is CAP Theorem?
Answer: It states a distributed system can only guarantee two of Consistency, Availability, Partition Tolerance at once.
💡 **Tip:** Mention Cassandra = AP, MongoDB = CP.

Q22. What is eventual consistency?
Answer: Data may not be instantly updated across nodes but will become consistent over time.
💡 **Tip:** Example: Amazon product inventory updates.

Q23. What is a message queue (Kafka, RabbitMQ)?
Answer: Queues enable asynchronous communication between services, ensuring reliability and decoupling.
💡 **Tip:** Always link it to microservices scaling.

Q24. CDN vs reverse proxy?
Answer: CDN caches static content globally, reducing latency. Reverse proxy forwards requests to backend servers, often with load balancing.
💡 **Tip:** Example: Cloudflare = CDN + reverse proxy.

Q25. What is an API Gateway?
Answer: API Gateway acts as a single entry point, handling routing, authentication, rate limiting, and monitoring.
💡 **Tip:** Mention AWS API Gateway or Kong.

🔹 Section 6: Backend Best Practices

Q26. What are 12-factor apps?
Answer: A methodology for building scalable, portable apps with best practices like config in env variables, stateless services, and CI/CD.
💡 **Tip:** Mention “cloud-native apps follow this model.”

Q27. How do you handle logging and monitoring?
Answer: Use centralized logging (ELK, Loki) and monitoring

(Prometheus, Grafana) to track performance and issues.
💡 **Tip:** Don’t forget alerting systems like PagerDuty.

Q28. How do you ensure safe database migrations?
Answer: Use tools (Flyway, Liquibase), run migrations in staging, backup before changes, and roll forward when possible.
💡 **Tip:** Never run migrations directly on production DB.

Q29. Common design patterns in backend?
Answer: Singleton, Factory, Repository, Observer are often used for reusability and maintainability.
💡 **Tip:** Be ready to give code-level examples.

Q30. Common backend interview mistake candidates make?
Answer: Over-focusing on coding while ignoring scalability, security, and best practices.
💡 **Tip:** Always discuss performance trade-offs during interviews.