### **Front-End:**

**Difference between useState and useReducer in React. When to choose which?**

* **useState:** A React hook for managing simple state variables (numbers, strings, booleans, objects). It is easy to use when dealing with small, isolated state changes (e.g., toggling a modal, updating a form field).
* **useReducer:** A more advanced hook used for managing complex state logic where multiple state values are interdependent, or where the state update logic is more involved (e.g., when the state depends on previous state or multiple actions). Similar to Redux-style reducer functions.

**When to choose:**

* Use useState for **simple state** with straightforward updates.
* Use useReducer for **complex state management**, especially when actions trigger state changes, or when multiple state variables are involved (such as form validation, shopping cart, or state machines).

### 

### 

### 

### 

### 

### 

### 

### 

### 

### **Back-End:**

**How to handle CORS in Node.js/Express?**

You can use the cors middleware package in Express to handle Cross-Origin Resource Sharing (CORS):

| const express = require('express'); const cors = require('cors');  const app = express();  // Enable CORS for all routes app.use(cors());  // Optionally configure CORS: app.use(cors({  origin: 'https://your-frontend-domain.com',  methods: ['GET', 'POST', 'PUT', 'DELETE'],  credentials: true }));  app.listen(3000, () => {  console.log('Server running on port 3000'); }); |
| --- |

### 

### 

### 

### 

### 

### 

### 

### 

### 

### **Database:**

**Difference between SQL and NoSQL databases. Example use cases.**

* **SQL (Relational Database):** Structured data stored in tables with predefined schema (columns, data types). Supports ACID transactions and complex joins.  
   **Example:** PostgreSQL, MySQL.  
    
   **Use case:** Financial applications, inventory management, systems that require strict data consistency.
* **NoSQL (Non-relational Database):** Flexible schema. Stores data as key-value pairs, documents, graphs, or wide-columns. Scales horizontally easily.  
   **Example:** MongoDB, Cassandra, Redis.  
    
   **Use case:** Real-time analytics, user profile stores, high-speed caching.

### 

### 

### 

### 

### 

### 

### 

### 

### 

### 

### 

### 

### 

### **General:**

**How to debug a performance issue in a full-stack application.**

1. **Identify the bottleneck:** Use browser DevTools (Network tab, Performance tab) to measure frontend load times. Use backend APM tools (New Relic, Datadog) to profile backend API performance.
2. **Check network latency:** Inspect API call latency and network issues (DNS, SSL, CDN delays).
3. **Analyze backend performance:** Use profiling to check for slow database queries, inefficient loops, memory leaks, blocking operations (e.g., sync file reads).
4. **Check database queries:** Use query profiling (EXPLAIN ANALYZE in SQL) to optimize slow queries, add indexes where needed.
5. **Monitor resource usage:** Use monitoring tools to observe CPU, memory, I/O usage on servers.
6. **Optimize:** Minify assets, lazy-load components, use caching (Redis), optimize database queries, reduce API payload size, leverage CDN.