.NET Core + React.js Mid-Level Cheat Sheet 🧠💥

🖕🏼 .NET Core

- **Middleware**: Process HTTP request/response pipeline. Can log, authenticate, etc. Use `next.Invoke()` to pass control.
- **DI Lifetimes**:
- `Singleton`: One app-wide instance
- `Scoped`: Per request
- `Transient`: New per use
- **Global Exception Handling**:
- `app.UseExceptionHandler()` + JSON error output
- **Role-based Authorization**:
- Use `[Authorize(Roles = "Admin")]`
- Configure roles/claims in Identity
- **EF Core Migrations**:
- `Add-Migration` – generate schema
- `Update-Database` – apply changes

👉🏾React.js

- **Hooks**:
- `useState`, `useEffect`, `useMemo`, `useCallback`
- **Controlled vs Uncontrolled**:
- Controlled: state-driven (`value`, `onChange`)
- Uncontrolled: access via `ref`
- **Context API vs Redux**:
- Context: Simple global state (theme, auth)
- Redux: Complex app state, async flows
- **Memoization**:
- `useMemo`, `React.memo`, `useCallback` optimize renders
- **Routing**:
- `useParams`, `useNavigate`, `<Routes><Route/></Routes>`

👈🏼Integration (React + .NET Core)

- **API Calls & CORS**:
- Use `fetch/axios` in React
- Enable CORS: `services.AddCors()` + `app.UseCors()`
- **JWT Auth**:
- Backend: Issue token
- Frontend: Store in `localStorage`, use `Authorization` header

- Use `[Authorize]` for protection
- **Hosting React in .NET**:
- `npm run build` → Copy to `wwwroot`
- Use `app.UseSpa()` in `Startup.cs`

🚀Tip: Be ready to whiteboard how data flows from React to .NET Core API to DB and back.

Need a printable PDF or a slick slide deck version? Say the word, anthamma!