좋아 👍
이제 **"카카오 로그인 + 기본 로그인" 전체 흐름을**
진짜 정확하게 정리해서 보여줄게.
(개발자 면접 답변 수준으로 완벽히 이해될 거야 👇)

🧩 전체 구조 요약

구분		보안 정책 영향	쿠키 저장 여부
기본 로그인 (Proxy)	axios (동일 출처처럼 동작)	CORS 적용 ❌	저장 ✅
기본 로그인 (CORS 설정)	axios (Cross-Origin)	CORS 적용 ✅	저장 ✅ (조건 충족 시)
카카오 로그인 (OAuth2 Redirect)	브라우저 리다이렉트	CORS 적용 ❌	저장 ✅ (`SameSite=Lax` 덕분)

🔷 1️⃣ 기본 로그인 (Proxy 방식)

✅ 구성

React → `"proxy": "http://localhost:8080"`
요청 코드:

```js
axios.post("/auth/login", { userId, password });
```


✅ 동작 원리

- 브라우저는 `localhost:3000` 요청이라고 인식함
- CRA 개발 서버가 내부적으로 `8080`으로 전달 (프록시 중계)
- 결과적으로 **같은 출처 요청처럼 동작**
- 그래서 **CORS 정책이 적용되지 않음**
- 서버가 `Set-Cookie` 내려주면 그대로 저장됨 ✅

📌 `{ withCredentials: true }` 불필요
📌 개발환경(로컬 테스트)에서 가장 간단

◆ 2 기본 로그인 (CORS 정식 설정 방식)

✅ 요청 코드

```js
axios.post("http://localhost:8080/auth/login",
  { userId, password },
  { withCredentials: tr
);
```

ChatGPT에게 묻기

✅ 서버 설정 (Spring)

```java
registry.addMapping("/**")
        .allowedOrigins("http://localhost:3000")
        .allowedMethods("GET", "POST", "PUT", "DELETE", "OPTIONS")
        .allowCredentials(true);
```

✅ 동작 원리

1 브라우저가 CORS preflight(OPTIONS) 요청 전송
2 서버가 아래 헤더로 응답 👇

```yaml
Access-Control-Allow-Origin: http://localhost:3000
Access-Control-Allow-Credentials: true
```

3 브라우저가 "이 서버는 쿠키 허용했네?" 하고
→ 응답의 `Set-Cookie`를 저장함 ✅
4 이후 `withCredentials:true`인 요청마다 자동으로 쿠키 포함됨

📌 이건 **운영 서버(실 배포 환경)** 에서 반드시 써야 하는 방식
📌 프록시 없이도 작동 가능하지만, 두 조건(`withCredentials`, `allowCredentials`)이 세트로 있어야 함

◆ 3 카카오 로그인 (OAuth2 Redirect)

✅ 요청 코드

```js
window.location.href = "http://localhost:8080/oauth2/authorization/kakao";
```

✅ 동작 순서

1 브라우저가 **React(3000) → Spring(8080)** 으로 리다이렉트 이동

- 이건 `axios` 요청이 아니라 **브라우저 네이티브 리다이렉트**
- 따라서 **CORS 정책이**

2 Spring Security가 카카오 로그인 인증 처리 후

- `CustomOAuth2SuccessHandler` 실행
- 여기서 `Set-Cookie` 내려줌 👇

```java
ResponseCookie cookie = ResponseCookie.from("refresh_token", token)
    .httpOnly(true)
    .sameSite("Lax")
    .path("/")
    .build();
response.setHeader(HttpHeaders.SET_COOKIE, cookie.toString());
```

3 브라우저는 이걸 **"사용자가 직접 이동한 리다이렉트 요청"**으로 판단
→ `SameSite=Lax` 정책에 따라 쿠키 저장 허용 ✅

4 이후 서버가 `response.sendRedirect("http://localhost:3000/oauth/success?...")`
→ React로 돌아감 (AccessToken은 URL 파라미터로 전달)

5 React는 `/oauth/success` 페이지에서 AccessToken 저장,
이후 요청 시 자동으로 `refresh_token`이 쿠키에 포함됨 👇

```makefile
Cookie: refresh_token=xxxxx
```

🔷 4️⃣ SameSite=Lax란?

설정	설명
Strict	외부 사이트에서 온 모든 요청에서 쿠키 차단
Lax	사용자가 직접 클릭하거나 리다이렉트한 요청은 쿠키 허용 ✅
None	모든 출처 허용 (단 HTTPS + secure=true 필요)

👉 즉,

> `Lax`는 "직접 이동은 신뢰할게, 자동 호출은 막을게" 정책.

OAuth 로그인처럼 브라우저 리다이렉트 기반 요청에는 딱 맞음.

🧠 전체 흐름 비교 요약

항목	기본 로그인 (Proxy)	기본 로그인 (CORS)	카카오 로그인 (OAuth2 Redirect)
요청 방식	axios	axios	window.location.href
출처	동일 (프록시로 위장)	교차 출처	교차 출처
브라우저 정책	CORS 미적용	CORS 적용	CORS 미적용
쿠키 저장 방식	Set-Cookie 자동 반영	`withCredentials` + `allowCredentials` 필요	리다이렉트 + `Lax` 허용
특징	개발 편의용	배포용 정식 방식	소셜 로그인 전용 방식

✅ 한 줄 요약

- **기본 로그인:**
  - 개발용 → 프록시
  - 운영용 → `{ withCredentials: true }` + `.allowCredentials(true)`
- **카카오 로그인:**
  - 브라우저 리다이렉트라 CORS 영향 없음
  - `.sameSite("Lax")`로 설정해 쿠키 저장 허용
- **Lax:**
  - "사용자가 직접 이동한 요청만 쿠키 허용"