cors
因為在 javascript 中，若要使用 fetch API 或是 XMLHttpRequest 等其他方式來向其他網頁、伺服器發送請求時，基於安全性的考量(指的是不允許未知、未授權的伺服器可以透過 API 來刪除、更新資料庫，或是存取儲存在 cookies 上的 token…等)，需要遵守同源政策（同源政策必須滿足三個條件:1.發送請求和接收請求的網站需要使用相同的通訊協定(protocol),ex:http/https. 2. 兩者須在相同的網域(domain)3.兩者需使用相同的 port），若遵守同源政策，即可不受限制地發起任何 request，但若不遵守同源政策，就必須透過設定 CORS 的方法，來讓 request 成功執行。

在 CORS 的規範中，跨來源請求分成兩種：簡單請求、非簡單請求

所謂的「簡單」請求，必須符合下面兩個條件：

只能是 HTTP GET, POST or HEAD 方法
自訂的 request header 只能是 Accept、Accept-Language、Content-Language 或 Content-Type（值只能是 application/x-www-form-urlencoded、multipart/form-data 或 text/plain）
不符合以上任一條件的請求就是非簡單請求

簡單請求要使用 CORS 的方法如下：
在 response 裡加上 Access-Control-Allow-Origin header, EX:
Access-Control-Allow-Origin: https://yourserver
如果 server 允許任何來源的跨來源請求，那可以直接回 *
Access-Control-Allow-Origin: *

而非簡單跨來源請求作法如下：

瀏覽器在發送請求之前會先發送一個 「preflight request（預檢請求）」，其作用在於先問伺服器：你是否允許這樣的請求？真的允許的話，才會把請求完整地送過去。
這麼做的原因是因為，通常在此類請求中，請求訊息會帶有機密、不公開的資料，如果說 CORS 的設定是在保護持有 API 的 server，而預檢請求便是在保護發送請求的端點，（保護 server 端的行為我覺得是在預檢請求過後的確認 header 才產生）。

server 端設定預檢請求的方法：
Access-Control-Allow-Methods: POST
Access-Control-Allow-Headers: X-MY-CUSTOM-HEADER, Content-Type

若通過後會再檢查 origin:
Access-Control-Allow-Origin:

跨來源請求的 Cookie:
一般的 http request 會帶有該網域底下的 cookie；然而，跨來源請求預設是不能帶 cookie 的。

如果請求攜帶的 cookie 是 session token，那這個請求可以以你的身份做很多機敏的事情，像是存取你的隱私資料、從你的銀行帳戶轉帳等。

所以瀏覽器端針對跨來源請求的 cookie 也做了規範。

首先，請求必須要明確地標示「我要存取跨域 cookie」。使用 fetch API 和 XMLHttpRequest 的設定方法如下：

透過 fetch API 發送跨來源請求，需要設定 credentials: 'include
如此一來跨來源請求就會攜帶 cookie 。

Server 端也需要額外的設定：如果是信任的來源，回應要帶有 Access-Control-Allow-Credentials header：
Access-Control-Allow-Credentials: true

**如果是允許使用 cookie 的情況，Access-Control-Allow-Origin 不能用 *，必須明確標示哪些來源允許存取。

我自己在寫 API 開放其他網域去 fetch api 的方法是 require cors;
app.use(cors); 預設的行爲就如同：Access-Control-Allow-Origin:*