Db.json(DataBase 資料庫)

* users：用戶資訊的集合，key 是用戶的身份證字號
  + password：用戶密碼
  + blocked：布林值，表示該用戶是否被封鎖
* tickets：用戶的訂票資料集合，同樣用身份證字號作為 key
  + 每個用戶對應一個陣列，裡面是該用戶購買過的車票

**舉例**

* 用戶 A184804821 密碼是 "skysea950304"，沒被封鎖，有一張訂票紀錄：
  + 從「台北」到「新竹」
  + 日期是 2025 年 6 月 7 日
  + 時間是 17:30 → 18:11
  + 價格 99 元
  + 付款方式是「Line Pay」
  + 座位等級是「一般座」

**功能**

這個 JSON是專案的後端資料庫，用來：

* 紀錄用戶帳號和密碼
* 紀錄用戶是否被封鎖
* 紀錄用戶購買的車票明細

票務系統的帳號和訂票資料存儲格式。

EecsDB.js(資料核心)

操作 db.json的「資料存取層」，用來讀取、寫入、管理用戶與訂票資料的函式集合，採用非同步 async/await 處理檔案操作。

## 邏輯說明

### 1. 私有函式：\_readDB 與 \_writeDB

* \_readDB()：非同步讀取 db.json，回傳解析後的物件。
  + 失敗（檔案不存在或格式錯誤）時，回傳預設結構避免程式崩潰。
* \_writeDB(data)：把傳入的物件寫入 db.json，且格式化成漂亮的 JSON 字串。

### 2. 使用者管理函式

* existsUser(id)：檢查用戶 ID 是否存在於資料庫中，回傳 true 或 false。
* createUser(id, password)：
  + 新增用戶，若用戶已存在，拋出錯誤。
  + 新增時同時建立空的訂票陣列。
* getUser(id)：取得用戶資料，找不到回傳 null。
* blockUser(id)：將用戶帳號設定為封鎖，找不到用戶拋錯。
* unblockUser(id)：解除封鎖，找不到用戶拋錯。

### 3. 訂票管理函式

* getTickets(userId)：取得該用戶的訂票陣列，若無資料回傳空陣列。
* addTicket(userId, ticketObj)：
  + 新增一筆訂票到指定用戶的訂票清單。
  + 若用戶不存在，拋出 'NOT\_FOUND\_USER'。
* removeTicket(userId, index)：
  + 刪除用戶訂票清單中指定 index 的訂票。
  + 若用戶不存在或 index 不合法（超出範圍），拋錯。
* getAllTickets()：
  + 回傳所有使用者和訂票資料，通常管理員用來查看全資料。
* 封裝對 db.json 讀寫操作的函式，將檔案作為簡單資料庫使用。
* 支援新增、讀取、刪除使用者和訂票資料，並管理用戶封鎖狀態。
* 票務系統的核心資料層，讓其他程式碼可以透過這些介面操作資料，不需直接處理檔案。

node\_modules (Node.js套件)

**Node.js** 是一個基於 **Chrome V8 JavaScript 引擎執行環境**，讓開發者可以用 JavaScript 來撰寫後端程式，也就是不只在瀏覽器，還可以在伺服器上執行 JS 程式碼。

### Node.js 的特點：

* **非同步 I/O（Asynchronous I/O）**：使用事件驅動架構，處理大量並發請求效率很高。
* **單執行緒但高效能**：適合 I/O 密集型應用，如 API 伺服器或聊天室。
* **有豐富的 npm 生態系**：可用 npm install 快速引入數十萬個套件（如 express、dotenv 等）。
* **跨平台**：可以在 Windows、macOS、Linux 執行。

App.js(前端函式核心)

### 1. getAuthHeader()

* **作用**：每次呼叫 API 時，從 localStorage (當地存取資料) 拿出目前登入的 userId，然後把它放在 HTTP 請求的自訂標頭 "X-User-Id" 裡。
* **用途**：用來讓後端知道是哪個使用者發出的請求。

### 2. validateTaiwanID(id)

* **作用**：驗證輸入的身分證字號格式是否符合中華民國身分證規則（字母開頭＋特定格式數字）。
* **用途**：確保使用者輸入的身分證字號是合法有效的。

### 3. register()

* **作用**：讓使用者註冊新帳號。會檢查身分證和密碼格式，通過後呼叫後端 /api/register 來建立帳號。
* **用途**：註冊新使用者帳號。

### 4. login()

* **作用**：讓使用者登入。檢查輸入格式，呼叫後端 /api/login。登入成功時，把使用者身分證存入 localStorage，切換頁面 UI，並根據是否為管理員顯示不同介面。
* **用途**：讓使用者登入系統，並顯示適合的功能頁面。

### 5. setupDateTimeInputs(dateInput, timeInput)

* **作用**：設定「日期只能從今天開始」以及「同一天不可選擇過去時間」的輸入限制。並監聽使用者改變日期和時間時的行為，確保時間有效。
* **用途**：防止使用者選擇過去的日期和時間(但是白癡蘋果會繞過前端時間判定，因為瀏覽器對時間的設定不成熟，所以我後面有加一個後端監聽的功能，哭了都)。

### 6. logout()

* **作用**：登出使用者，清除 localStorage 的登入資料，恢復到登入畫面，隱藏所有使用者和管理員的功能介面。
* **用途**：讓使用者安全退出系統。

### 7. querySchedule()

* **作用**：根據使用者輸入的起站、訖站、日期和時間，計算並顯示可搭乘的列車班次清單。裡面包含列車時刻和票價選項。
* **用途**：提供查詢火車班次的功能，純前端運算，但會先向後端驗證日期時間是否合法(對，就是上面說的那樣，有被搞到)。

### 8. bookTicket(from, to, date, time, classSelectId, priceSpanId, paySelectId)

* **作用**：當使用者按下「訂票」按鈕時，把訂票資訊（起訖站、日期、時間、座位等）送給後端 /api/tickets 儲存。
* **用途**：讓使用者成功訂票。

### 9. cancelTicket()

* **作用**：向後端 /api/tickets 取得該使用者的所有訂票資料，然後在前端顯示票券清單，並提供取消與取票按鈕。
* **用途**：讓使用者管理自己的訂票，包括取消和取票。

### 10. removeTicket(index)

* **作用**：使用者按取消某張票時，呼叫後端 /api/tickets/:index 刪除該票。
* **用途**：讓使用者取消已訂的票。

### 11. updatePrice(priceSpanId, basePrice, classSelectId)

* **作用**：當使用者改變座位等級時，更新顯示的票價（依不同座位加價）。
* **用途**：動態顯示票價變動。

### 12. showAdminPanel()

* **作用**：顯示管理者介面，包含查詢所有訂票、封鎖使用者、刪除車票等功能。
* **用途**：讓管理員管理系統和使用者。

### 13. getAllTickets()

* **作用**：管理員呼叫後端 /api/admin/tickets 取得所有使用者的訂票資料並顯示。
* **用途**：管理員查看所有訂票。

### 14. blockUser(userId)

* **作用**：管理員封鎖指定使用者，呼叫 /api/admin/block/${userId}。
* **用途**：防止惡意或違規使用者使用系統。

### 15. removeUserTicket(userId, index)

* **作用**：管理員刪除某使用者特定的一張票
* **用途**：管理員手動移除使用者的訂票。

### 16.deleteSingleTicket(userId, ticketIndex)

* **用途**：管理員想刪除「別人」某張票的時候用。
* **做法說明**：
  + 後端 API 僅允許使用者刪除自己帳號底下的票券，沒有直接給管理員刪別人票的接口。
  + 這裡用一個「假裝自己是該使用者」的變通方法（workaround）：
    1. 先呼叫 /api/blockUser 但用 unblock 動作，目的是把 HTTP 請求標頭 X-User-Id 改成目標使用者的 userId，讓後端誤以為這個請求是由該使用者本人發出。
    2. 再用同樣的 X-User-Id 呼叫 DELETE /api/tickets/:ticketIndex，來刪除該使用者的票。
  + 注意：這只是範例示範用，實務上應該要在後端新增「管理員專用路由」來直接刪除任意使用者票券，會比較安全乾淨。
  + (你們可以拿這個當未來展望)
* **功能重點**：模擬用戶身份，以便完成跨帳號刪票的需求。

### 17.toggleBlockUser(id, currentlyBlocked)

* **用途**：管理員用來封鎖或解鎖指定的使用者。
* **細節**：
  + 根據目前使用者是否已被封鎖，決定是要封鎖（block）還是解鎖（unblock）。
  + 呼叫後端 /api/blockUser，將指定的使用者 id 與動作（block/unblock）傳給後端處理。
  + 成功時顯示提示，並刷新管理員介面。
* **功能重點**：快速封鎖或解鎖使用者帳號。

### 18.updatePrice(priceSpanId, basePrice, classSelectId)

* **用途**：根據使用者選擇的座位等級，動態更新票價顯示。
* **做法**：
  + 從指定的下拉選單取得座位等級（如「商務座」、「貴賓座」）。
  + 根據不同等級，將基本票價乘以特定倍數（1.3 或 1.7）做加價。
  + 將計算結果寫入，更新畫面顯示。
* **功能重點**：票價根據座位種類即時更新。

### 19.scrollToTop()

* **用途**：平滑地將網頁視窗捲動到頁面頂端。
* **功能重點**：用於使用者體驗，讓頁面快速回到頂部。

### 20.window.addEventListener('DOMContentLoaded', ...)

* **用途**：頁面載入完成時，檢查是否有已登入的使用者。
* **行為**：
  + 從 localStorage 拿出 currentUser。
  + 若有登入使用者，顯示歡迎訊息，並切換 UI 到操作介面。
  + 如果是管理員帳號，直接進入管理員介面。
  + 否則顯示普通使用者介面。
* **功能重點**：實現自動登入狀態維持與界面初始化。

### 21.toggleMenu()

* **用途**：控制手機版選單的開關切換。
* **做法**：
  + 取得選單元素，切換它的 hidden class，達到顯示或隱藏的效果。
* **功能重點**：實作手機響應式選單的開合。

### 22.eecs\_admin(req)

* **用途**：驗證傳入請求是否包含正確的管理員 Token。
* **做法**：
  + 從請求參數取得 token，與環境變數 ADMIN\_TOKEN 比對。
  + 若相符，回傳 true，表示該請求為管理員權限。
* **功能重點**：簡易管理員權限認證，(用來配合下載資料庫以用於本地更新啦)。

index.html 網頁前端

## <!DOCTYPE html>&<html lang="zh-TW">

* 宣告文件是 HTML5 格式。
* lang="zh-TW" 表示網頁主要語言為繁體中文（台灣）。

## <head> 區塊

* <meta charset="UTF-8">：設定文件使用 UTF-8 編碼，支援中文與其他文字。
* <meta name="viewport" content="width=device-width, initial-scale=1.0">：讓頁面在手機等裝置上自適應寬度。
* <title>：設定瀏覽器標籤標題為「訂票系統\_EECS出品」。
* <link rel="stylesheet" href="style.css">：載入外部 CSS 樣式檔 style.css，負責頁面美化。

## <body> 主體結構

### 1. <header> 頭部區塊

* 網頁標題 <h1>：「我沒有醉」鐵路有限公司。
* 導覽列 <nav> 包含四個連結：
  + 開發團隊網站
  + 隨機生成（隨機帳號）
  + 免費票券（實際是著名惡搞影片連結(never gonna give u up)）
  + 資料來源（一些引用照片和台鐵網站）
* 手機版選單切換按鈕 .menu-toggle，點擊時呼叫 toggleMenu() 函式控制手機選單顯示。
* 手機版選單 <nav id="mobileMenu" class="mobile-menu hidden">，初始隱藏，包含和上面同樣的連結，方便手機用戶操作。

### 2. <main> 主要內容區

#### (a) 登入區 <section class="login">

* 歡迎標語和簡短說明，用分階段的動畫效果。
* 登入表單 <form>，防止表單直接提交：
  + 帳號輸入框
  + 密碼輸入框
  + 兩個按鈕：註冊與登入，分別觸發 register() 和 login() 函式，點擊後會呼叫 scrollToTop() 回到頁面頂部。
* 一個空的 <div class="particles">，用來當景特效的容器。

#### (b) 操作區 <section class="operation hidden">

* 初始為隱藏，登入成功後會顯示。
* 使用者面板：
  + 顯示歡迎文字與使用者名稱。
  + 票務選擇區域：
    - 出發站與目的地下拉選單（幾個常見台灣車站）
    - 日期與起始時間選擇器
  + 三個按鈕：
    - 查詢車次 querySchedule()
    - 我的車票 cancelTicket()
    - 登出 logout()
* #result：顯示查詢結果或其他訊息
* #adminPanel：管理員專用面板，預設隱藏
* <canvas id="ticketCanvas">：用來繪製電子票券，預設隱藏
* 另一個 .particles 容器，用於背景動畫或效果。
* 在 <main> 內載入了 QR Code 生成函式庫 qrcode.min.js，用於生成票券的 QR Code(但最後沒用到 因為搞不出來)。

### 3. <section class="slogan">

* 引用鐵路局官方網站上的企業宗旨作為網頁標語。

### 4. <footer>

* 簡單註明此網頁為中原大學電資學院學士班學習用途。

## 頁面結尾腳本

* 載入 app.js：主要的 JavaScript 功能實作

### CSS(美化版面語法)

### (有點長，其實不太重要，因為這部分只有最一開始的模板是我做的，後面八成是GPT做的，我只有調整參數，可以理解為網頁修圖軟體hehe)

### /\* Reset \*/

{

padding: 0;

margin: 0;

}

**說明：**  
「重置」的 CSS 規則，將所有 HTML 元素的內距（padding）和外距（margin）都設為 0。這樣做是為了避免不同瀏覽器的預設樣式差異，確保頁面在各瀏覽器間的呈現更一致。

### /\* Header 開始----\*/

header {

background-color: rgba(14, 4, 28, 0.94);

height: 80px;

width: 100%;

position: fixed;

top: 0;

left: 0;

z-index: 9999;

display: flex;

align-items: center;

padding: 0 20px;

box-shadow: 0 2px 10px rgba(0, 0, 0, 0.3);

}

* **background-color:** 使用半透明深色（紫黑色系），讓 Header 有點透明效果。
* **height:** 固定高度 80px。
* **width:** 寬度佔滿整個視窗寬度。
* **position: fixed:** 固定在頁面頂部，滾動時不會移動。
* **top, left:** 頂端和左側對齊。
* **z-index: 9999:** 置於非常前面，確保不被其他元素蓋住。
* **display: flex; align-items: center;** 使用 Flexbox 水平排列，垂直置中內容。
* **padding:** 左右內距 20px。
* **box-shadow:** 下方陰影，增加立體感。

header ul {

position: absolute;

right: 5vw;

top: 0;

line-height: 80px;

}

* 導航選單使用絕對定位，放在右側距離螢幕寬度 5% 的位置。
* line-height 設為 80px，讓選單項目垂直置中。

header li {

display: inline;

margin-right: 5vw;

}

* 清單項目橫向排列（inline），每個項目右邊留出 5% 的寬度空白。

header a {

color: white;

text-decoration: none;

}

header a:hover {

color: rgb(246, 255, 84);

text-decoration: underline white;

}

* 連結文字預設為白色，且取消底線。
* 滑鼠懸停時改變顏色為亮黃色，並加上白色底線，增強互動感。

h1 {

color: white;

position: absolute;

left: 120px;

top: 0;

line-height: 80px;

}

* 頁首標題文字白色，絕對定位在距離左側 120px，頂端對齊，並垂直置中（利用 line-height 80px）。

### booking-row（訂票列區塊）

.booking-row {

display: flex;

align-items: center;

justify-content: space-between;

gap: 10px;

margin: 10px 0;

flex-wrap: nowrap;

}

* 用 Flexbox 排版，一排橫向排列項目。
* align-items: center; 垂直置中對齊。
* justify-content: space-between; 左右兩端對齊，中間均分空間。
* gap: 10px; 項目間距 10px。
* flex-wrap: nowrap; 不換行，保持在同一排。

### booking-row h4

.booking-row h4 {

margin: 10px;

font-size: 16px;

flex: 2;

white-space: nowrap;

overflow: hidden;

text-overflow: ellipsis;

}

* 標題文字大小 16px，四邊外距 10px。
* flex: 2; 佔兩倍比例空間。
* white-space: nowrap; 不換行。
* overflow: hidden; 超出文字隱藏。
* text-overflow: ellipsis; 超出部分用省略號顯示。

### booking-row select（下拉選單）

.booking-row select {

padding: 4px 8px;

font-size: 14px;

}

* 內邊距上下 4px、左右 8px。
* 字體大小 14px。

### booking-row .booking（訂票按鈕）

.booking-row .booking {

font-size: 14px;

padding: 6px 14px;

border-radius: 6px;

border: 1.5px solid #ccc;

background-color: transparent;

color: #f0f0f0;

cursor: pointer;

transition: all 0.25s ease-in-out;

backdrop-filter: blur(2px);

}

* 14px 字體，內邊距適中，圓角 6px。
* 邊框淺灰色，背景透明，文字白色偏灰。
* 鼠標變成手指指標。
* 0.25秒過渡動畫。
* 背景模糊效果，半透明風格。

**懸停與點擊時效果：**

.booking-row .booking:hover {

border-color: #ffe082;

background-color: rgba(255, 255, 255, 0.05);

color: #ffe082;

box-shadow: 0 0 8px rgba(255, 224, 130, 0.5);

}

.booking-row .booking:active {

transform: scale(0.96);

background-color: rgba(255, 255, 255, 0.08);

}

* 滑鼠停留時邊框變成淡黃，背景微亮，文字變黃，並有光暈。
* 按下去時縮小 4%，背景稍微變亮。

### choose-group 與 .choose（選擇組與選擇項目）

.choose-group {

display: flex;

flex-wrap: wrap;

justify-content: center;

gap: 12px;

margin-bottom: 20px;

}

* Flex 排列，可換行。
* 項目置中對齊，間距 12px，底部留白 20px。

.choose {

display: flex;

flex-direction: column;

font-size: 16px;

color: #f0f0f0;

background-color: rgba(255, 255, 255, 0.05);

padding: 12px 18px;

border-radius: 16px;

box-shadow: 0 4px 8px rgba(0, 0, 0, 0.25);

transition: transform 0.2s ease, background-color 0.3s ease;

backdrop-filter: blur(6px);

min-width: 160px;

}

* 單個選項內部垂直排列。
* 字體大小 16px，文字淺灰白。
* 半透明白色背景，有模糊效果，帶陰影。
* 圓角 16px，帶動畫過渡。
* 每個項目最小寬度 160px。

.choose select,

.choose input {

margin-top: 8px;

padding: 10px;

border-radius: 10px;

border: none;

font-size: 15px;

background-color: #1c1c1c;

color: white;

}

* 下拉選單和輸入框有上方空隙、內邊距、圓角和深色背景，字體白色。

### hidden（隱藏）

.hidden {

display: none;

}

* 用於隱藏元素。

### no-train-box（無車次提示盒）

.no-train-box {

background-color: #1e2a35;

padding: 20px;

border-radius: 12px;

border: 1px solid #2f3b46;

box-shadow: 0 4px 10px rgba(0, 0, 0, 0.3);

color: #fff;

max-width: 600px;

margin-top: 20px;

animation: fadeIn 0.6s ease;

text-align: center;

margin-left: auto;

margin-right: auto;

}

* 深藍灰背景，內距 20px，圓角 12px。
* 邊框淡藍，陰影效果。
* 文字白色，最大寬度 600px，置中。
* 進場時有淡入動畫。

### my-ticks（我的票券清單）

.my-ticks {

list-style: none;

padding: 12px 18px;

margin: 12px 0;

background-color: rgba(255, 255, 255, 0.05);

color: #fff;

border: 1px solid rgba(255, 255, 255, 0.1);

border-radius: 12px;

box-shadow: 0 4px 10px rgba(0, 0, 0, 0.2);

font-size: 1rem;

display: flex;

justify-content: space-between;

align-items: center;

flex-wrap: wrap;

gap: 8px;

transition: background-color 0.3s ease;

}

* 清單去掉預設點。
* 半透明淺白背景與邊框，圓角、陰影。
* 票券文字白色，字體 16px 左右。
* Flex 佈局，左右分散，內容換行，間距 8px。
* 滑鼠移入時背景會稍亮。

刪除按鈕樣式：

.my-ticks button {

padding: 6px 12px;

background-color: #ff5c5c;

border: none;

color: white;

border-radius: 6px;

cursor: pointer;

font-size: 0.9rem;

transition: background-color 0.2s ease;

}

.my-ticks button:hover {

background-color: #ff1f1f;

}

* 按鈕紅色，圓角，字體稍小，懸停顏色加深。

### admin-control（管理介面）

.admin-control {

padding: 20px;

margin-top: 20px;

color: #fff;

font-size: 16px;

box-shadow: 0 4px 12px rgba(0, 0, 0, 0.25);

}

* 白字，適當內距和陰影，字體稍大。

### 響應式設計

### Css中比較有意義的部分（螢幕最大寬度 768px）

針對手機等窄螢幕做多項調整：

* Header 高度降低，字體縮小。
* 隱藏大螢幕的選單。
* 表單字體縮小，按鈕寬度拉滿。
* 多個彈性區塊改成垂直排列、寬度調整。
* 文字大小減小，間距變小，讓介面更緊湊易操作。

### menu-toggle 與 .mobile-menu（手機選單切換）

.menu-toggle {

position: absolute;

top: 15px;

left: 15px;

font-size: 24px;

cursor: pointer;

z-index: 999;

color: white;

}

.mobile-menu {

display: flex;

flex-direction: column;

position: absolute;

top: 60px;

left: 0;

right: 0;

background-color: #ffffff;

border-top: 1px solid #ccc;

z-index: 998;

}

.mobile-menu a {

padding: 10px 20px;

text-decoration: none;

color: #333;

}

.mobile-menu.hidden {

display: none;

}

* .menu-toggle 是漢堡選單按鈕，固定在左上角，字體大且白色。
* .mobile-menu 是手機版選單，白底縱向排列。
* .mobile-menu.hidden 用來控制選單的顯示或隱藏。

Package.js(可以理解為簽名)

### "name": "eecs-ticketing-system"

* 這是你的專案名稱。
* 通常會用小寫字母和連字號，符合 npm 規範。

### "version": "1.0.0"

* 版本號，符合 [Semantic Versioning](https://semver.org/) 規範。
* 1.0.0 表示第一個正式版本。

### "description": ""專案描述(我懶得寫ㄏ)。

### "main": "server.js"

* 入口檔案，Node.js 預設從這個檔案開始執行程式。
* 這裡指定的是 server.js。

### "scripts"

* 這裡定義可用的指令腳本， npm run 執行。

"start": "node server.js"

* npm start 會執行 node server.js，啟動你的伺服器。

"test": "echo \"Error: no test specified\" && exit 1"

* npm test 預設會印出錯誤訊息，因為沒寫測試程式(我沒在本機測試， 全部直接丟到網上測試)

### "keywords": []

* 關鍵字陣列，用來描述專案，方便在 npm 搜尋找到。
* 現在是空的，可以填如 "ticketing", "express" 等。

### "author": ""

* 作者欄位，通常填寫開發者名字或團隊名稱(我沒寫)

### 8. "license": "ISC"

* 授權條款，ISC 是一種開源授權，允許自由使用、修改與散布。

### 9. "dependencies"

* 這是專案執行時需要的外部套件。

"body-parser": "^2.2.0",

"dotenv": "^16.5.0",

"express": "^5.1.0"

* body-parser：用來解析 HTTP 請求的請求體（例如 JSON 或表單資料）。
* dotenv：從 .env 檔案載入環境變數，方便管理密鑰與設定。(但我放在render上了)
* express：Node.js 最流行的 Web 框架，用於建立伺服器與路由。

^ 符號表示安裝的版本可以自動升級到主版本號不變的最新版本（例如 2.x.x）。

Server.js

(前後端連接最重要的東西)

**1.引入套件與模組：**

dotenv：讀取 .env 環境變數設定（例如 PORT、管理員 TOKEN）。

express：Web 框架，負責建立 HTTP 伺服器和路由。

body-parser：用來解析 JSON 格式的 HTTP 請求體。

path、fs：Node.js 內建，用於路徑處理與檔案操作。

db：自訂的資料庫操作模組，放在 ./data/eecsDB.js。

**2. 建立 Express 應用。**

const app = express();

const PORT = process.env.PORT || 3000;

從環境變數讀取 PORT，如果沒設定就用 3000。

app.use(bodyParser.json());

app.use(express.static(path.join(\_\_dirname, 'public')));

使用 bodyParser.json() 解析請求中的 JSON。

**3.使用者註冊 API : api/register**

註冊新用戶。

需要 id 和 password。

呼叫 db.createUser 建立用戶。

若帳號已存在，回傳 409 衝突錯誤。

成功回傳 { success: true }。

**4.使用者登入 API : api/login**

登入驗證

驗證帳號是否存在、是否被封鎖、密碼是否正確。

如果帳號是指定帳號，視為管理員，回傳 isAdmin: true。

**5.權限驗證中介 authMiddleware**

讀取客戶端在 HTTP header x-user-id 傳來的用戶 ID。

檢查用戶存在且沒被封鎖。

通過後，將 userId 存到 req 物件上，方便後續路由使用。

若不通過，回傳對應錯誤碼。

**6.查詢使用者票券 API : api/tickets**

需要驗證身份（用 authMiddleware）。

回傳該用戶的票券資料。

**7.新增票券**

app.post('/api/tickets', authMiddleware, async (req, res)

需身份驗證。

新增票券，資料欄位必填。

限制每人最多 5 張票。

成功回傳 { success: true }。

**8.刪除票券**

app.delete('/api/tickets/:index', authMiddleware, async (req, res)

刪除指定索引的票券。

驗證身份。

傳入的索引需為數字，且必須有效。

**9.管理員查看所有票券**

app.get('/api/allTickets', authMiddleware, async (req, res)

只有管理員可用。

回傳所有使用者與票券資料。

**10.管理員封鎖或解封使用者**

app.post('/api/blockUser', authMiddleware, async (req, res)

管理員可封鎖或解封使用者。

不允許封鎖自己。

**11.管理員下載資料檔案（db.json）**

function eecs\_admin(req)

需要 URL 帶有管理員的 ADMIN\_TOKEN，才能下載 db.json。

將 db.json 以檔案附件方式下載。

**12.表單日期時間驗證**

app.post('/submit', express.json(), (req, res)

用於前端表單提交驗證日期和時間。

不允許選擇過去的時間。

**13.啟動伺服器**

app.listen(PORT)

監聽指定的 PORT，啟動伺服器。

啟動成功會印出成功訊息。

Render(佈署平台)

### Render 部署 Node.js 專案流程簡介：

1. **專案準備好**
   * server.js、package.json，以及前端靜態檔放在 public/ 等目錄中。
2. **Push 到 GitHub**
   * 把專案上傳到 GitHub 儲存庫。
3. **登入 Render 並建立 Web Service**
   * 進入 Render → 選擇 **"New Web Service"**
   * 連接 GitHub 帳號，選取專案。
4. **設定部署細節**
   * **Build Command**：通常為 npm install
   * **Start Command**：對應 package.json 裡的 start，例如 node server.js
   * **Environment**：加上 .env 需要的變數（如 PORT、ADMIN\_TOKEN 等(我就把金鑰放在這裡)）
5. **部署完成！**
   * Render 會自動安裝、建構並啟動伺服器。
   * 成功後，你會獲得一個公開網址（例如 https://your-app.onrender.com）