構建 5G 智慧交通數位神經中樞－

附錄三、《交通數位雙生系統》建置指引手冊

交 通 部 運 輸 研 究 所

中 華 民 國 115 年 10 月

第一章 前言

1.1 編寫目的

本建置指引手冊之編撰目的，在於提供一套針對「交通數位雙生系統」之完整展示、操作與後續擴散的技術與實務參考依據。本平台以3D數位模擬與即時資料整合為核心，藉由 3D 圖台、AI 模型運算結果、及交通動態感測資料的結合，形成交通管理決策可視化的數位環境。

本手冊旨在說明於交通應用情境下的功能設計、資料呈現機制與使用者互動方式，以確保開發與操作人員能理解平台之核心結構；以及作為交通主管機關、地方政府與合作研究單位推動智慧交通數位化治理的重要指引文件，確保數位雙生應用能持續支援國內智慧運輸政策發展與跨域資料整合應用。

1.2 適用對象與範圍

一、適用對象

本手冊的讀者群廣泛，不僅限於技術人員，更涵蓋政策制定與執行的各個層面：政府交通主管機關與決策人員：可藉由本手冊理解數位雙生應用於交通管理決策中的角色與價值，作為推動智慧交通、數位治理與跨域資料整合政策之依據。

(一) 地方政府標案規劃與採購人員：可依據本手冊之內容，掌握系統應用服務流程、硬體需求與導入模式，作為後續在地化部署或採購標案規劃之參考依據。

(一) 系統整合商與軟體開發廠商：可作為開發與維運的參考依據，協助工程人員了解數位雙生平台之功能架構、應用服務與 3D 視覺化設計情境，確保系統可維護性與可擴充性。

(二) 學術與研究單位：可作為後續智慧交通數位雙生研究與延伸應用的參考資料，特別是在人流模擬、交通預測模型、以及資料融合分析等領域。

二、適用範圍

本手冊所定義之「數位雙生平台」，核心在於以「真實交通動態」與「虛擬模擬系統」的雙向連結，支援交通決策、場域管理及事件應變。適用範圍涵蓋下列兩種主要應用模式：

(一) 交通營運監控與即時決策支援：透過平台整合 GVP 即時速率資料、號誌運作資訊及其他交通事件資料，提供即時監測、趨勢分析與異常告警等功能，支援都市交通日常管理。

(二) 大型活動交通模擬與應變預測：以基礎單元網格為分析核心，結合人流預估、場域設施、公共運輸、道路管制等，模擬活動期間人車流動變化，協助事前規劃、事中監控與事後評估作業。

1.3 相關背景依據

本手冊的撰寫，除技術的展現之外，亦為實務經驗與政策目標的結合，其主要背景依據如下：

一. 反應國家智慧運輸發展政策：內容與架構設計，旨在反應國家與地方政府推動「智慧城市」與「智慧運輸系統（ITS）」的政策方向，為提升道路交通管理效率與用路人服務品質，提供具體的應用實踐路徑。

二. 降低導入門檻與風險：各單位無需從零開始進行漫長的技術探索與試錯，可作為未來各地交通單位導入或擴充應用時的參考依據，降低開發與整合風險。

三. 可視化技術之實踐：本手冊所述系統以 3D 圖台為前端核心，結合 AI 模型分析、GVP 即時速率資料、號誌使用率統計與人流變動模型，建構出一套可於 3D 空間中即時展現交通動態的數位雙生平台。透過此技術整合，可實現交通資料「從雲端到場域」的視覺化應用，並提供操作人員以情境模擬回放與多源資訊疊合方式，支援實務決策。

第二章 功能簡介

2.1 系統功能說明

本手冊所構建之「交通數位雙生系統」為一套整合 3D 視覺化引擎、自動化數據處理、時序資料管理與標準化對外服務的端到端智慧化視覺化方案。其總體功能旨在提供高擬真度、高互動性且高度客製化的即時交通 3D 視覺化服務。系統的核心功能包含以下四個層面：

一、3D 即時視覺化能力：本系統採用業界領先的 Cesium 3D 地球引擎，能全自動地完成從交通資料讀取、3D 場景渲染、道路網絡視覺化到互動式資訊展示的完整流程。透過不間斷的即時數據注入與週期性的場景更新，確保系統能持續呈現最新的交通狀態，維持其視覺化的準確性與即時性。

二、時序資料回放能力：本系統不僅展示當前時刻的路況，而是採用「時序資料管理」策略，透過 TemporalManagementService 與 TemporalDataCollection，能夠儲存並管理多個時間點的交通資料快照。使用者可透過時間軸控制器，自由瀏覽過去、現在與未來（預測）的交通狀態，實現交通狀況的時光機式回放功能。

三、高度客製化與可擴充性：為應對不同區域獨特的交通特性，系統的設計核心是支援多種資料來源（GVP Batch API、GVP Predict API、即時交通 API）的彈性整合。此「多資料源融合」的架構能達到最佳的資料覆蓋度與準確性，同時，底層的前端架構（Vue 3 + TypeScript）與視覺化引擎（Cesium）均為可水平擴充的設計，能支撐從數百到數千個路段的大規模視覺化需求。

四、彈性的擴散與整合能力：本系統提供兩種主要的擴散應用模式。政府單位可依據本手冊，在標案中要求廠商建置一套完整的私有化 3D 視覺化系統（自行開發模式）；亦可選擇將開發完成之系統提供一個標準化 API 服務，快速整合至既有的交通控制與資訊發布平台中。

2.2 適用場域與應用情境

本系統的 3D 視覺化能力可根據不同的道路網絡特性進行深度應用，主要分為以下兩大核心場域及其他衍生應用：

一、都市地區交通管理應用情境

在都市地區路況複雜、路網密集的環境下，本系統可以將 3D 視覺化結果整合

至地方政府的交通控制系統，達到以下應用：

(一) 即時路況監控與決策支援：透過 3D 場景即時呈現全市道路網絡的交通速率與壅塞狀況，以直觀的顏色分級（綠色順暢、黃色車多、橙色壅塞、紅色嚴重壅塞）展示，交控中心人員可快速掌握全局路況，並針對壅塞熱點進行即時調度與管制。

(二) 大型活動交通衝擊視覺化：針對演唱會、體育賽事、跨年活動等可預期的事件，可在事前利用本系統的 3D 場景模擬出活動對周邊路網的交通衝擊。藉此，警察局與交通局可制定更完善的交通管制、停車規劃與大眾運輸接駁計畫，並在活動期間根據 3D 視覺化動態調整。

(三) 歷史資料分析與模式發現：透過時序資料回放功能，交通規劃人員可回顧過去特定時段（如尖峰時段、特殊事件期間）的交通狀況，分析壅塞形成的原因與擴散模式，作為未來道路規劃與號誌優化的依據。

二、高速公路應用情境

在高速公路這種路況相對單純、車流連續性強的環境下，本系統的 3D 視覺化能力亦可發揮以下價值：

(一) 全線路況一覽：透過 3D 場景呈現整條高速公路的即時路況，包含主線、匝道、服務區等，讓交控中心能快速掌握全線狀況。

(二) 事故影響範圍視覺化：當發生交通事故時，系統可即時在 3D 場景中標示事故位置，並透過顏色變化呈現事故對下游路段的影響範圍與程度，協助交控中心更精準地規劃替代道路建議與警力疏導範圍。

三、其他衍生應用情境

透過 API 服務，本系統的視覺化數據亦可提供民間業者進行相關商業應用：

(一) 智慧城市儀表板整合：地方政府的智慧城市儀表板可介接本系統的 API，將交通 3D 視覺化作為其中一個模組，與其他城市數據（如空氣品質、公共設施使用率）整合展示。

(二) 交通資訊網站與 App：政府或民間的交通資訊服務可整合本系統的視覺化能力，提供民眾更直觀、更易理解的路況資訊。

第三章 系統功能建置說明

3.1 系統架構概述

本系統旨在建立一個從資料蒐集、時序資料管理、3D 場景渲染到對外服務的完整自動化交通數位雙生系統。系統由五個核心模組構成，本文件將依據系統架構(如圖 1)，詳細說明各模組的功能與運作流程。

3.2 軟體設計／演算法說明

一、資料蒐集與整合（Data Collection & Integration）模組

資料蒐集與整合模組是整個交通數位雙生系統的數據基礎與起點，此模組的核心功能是穩定、持續地從多個外部資料源獲取即時與預測的交通數據，並將這些結構化的資料整合後提供給後續的視覺化模組使用。資料來源的多樣性與高品質是後續 3D 視覺化準確性的根本。

為完成上述功能，本模組的建置與運作包含以下技術細節：

(一) 多資料源整合架構

系統設計支援三種主要資料來源的同時整合：

1. GVP Batch API：提供批次歷史交通資料，用於時序回放功能。
2. GVP Predict API：提供未來時段的交通速率預測資料。
3. 即時交通 API：提供當前時刻的即時交通資料（可選）。

(二) 資料獲取服務層

系統實作三個獨立的服務類別，分別負責與各資料源的通訊：

1. GvpBatchApiService：負責呼叫 GVP Batch API，獲取指定時間範圍內的批次交通資料。
2. GvpPredictApiService：負責呼叫 GVP Predict API，獲取未來時段的預測資料。
3. TrafficDataService：整合上述服務，提供統一的資料存取介面。

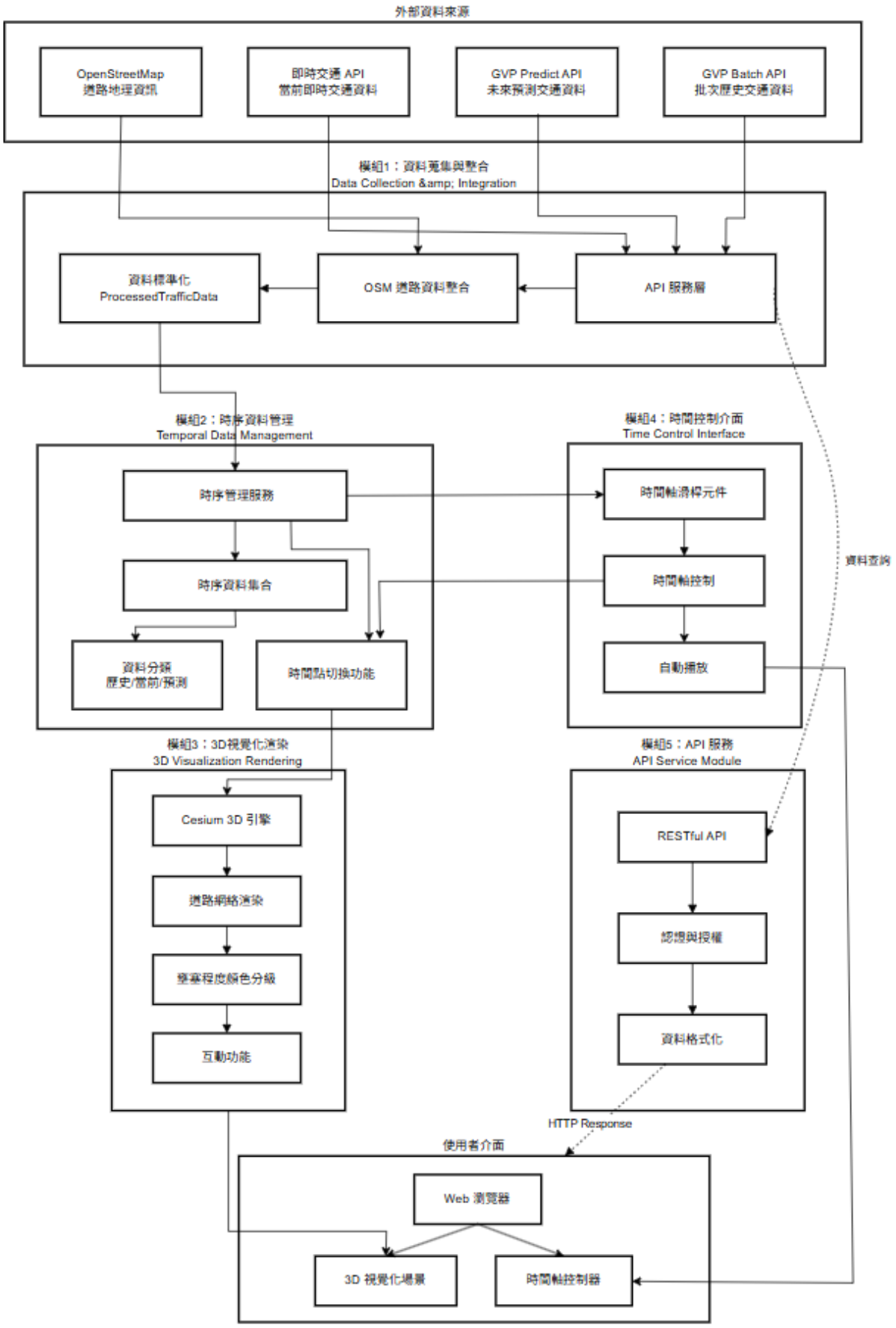


圖 1 交通數位雙生系統架構圖

(三) 資料欄位需求

從 API 獲取之交通資料，其欄位至少需要包含以下核心資訊：

1. 路段識別碼（osm_road_id 或 roadId）：用以唯一識別交通數據所屬的道路。
2. 日期時間（predict_time 或 predictTime）：精確到分鐘的時間戳，記錄該筆數據的時間。
3. 平均速率（avg_speed 或 avgSpeed）：該路段在該時間點的平均車輛行駛速率，是視覺化的基礎數據。
4. 方向（direction）：該路段的行駛方向(如 follow、reverse)。

(四) 資料轉換與標準化

經解析與提取後的資料，會被轉換為系統內部的標準格式 ProcessedTrafficData，包含以下欄位：

1. roadId：路段識別碼
4. roadName：路段名稱（從 OSM 資料庫查詢）
5. avgSpeed：平均速率
6. predictTime：資料時間戳
7. congestionLevel：壅塞程度（smooth、heavy、congested、severe）
8. roadType：道路類型（primary、secondary、tertiary、residential）
9. positions：路段的地理座標陣列（用於 3D 渲染）
10. dataSource：資料來源標記（GVP_PREDICT_API、GVP_BATCH_API、REAL_TIME_API）
11. direction：資料類型（current、predicted、historical）

(五) OSM 道路資料整合

為實現 3D 視覺化，系統需要道路的地理座標資訊。本模組整合 OpenStreetMap(OSM)道路資料庫：

1. 資料來源：使用預先處理的 OSM 道路資料檔案（road_iii.roadinfo.dat）。
2. 資料格式：二進位格式，包含道路 ID、名稱、類型、座標等資訊。
3. 快取機制：系統啟動時載入 OSM 資料至記憶體快取，提供快速查詢。
4. 座標轉換：將 OSM 的經緯度座標轉換為 Cesium 3D 引擎所需的 Cartesian3 座標。

(六) 資料流

1. 輸入：外部 API 提供的交通資料（JSON 格式）。

2. 輸出：標準化的 ProcessedTrafficData 陣列，包含完整的道路資訊與地理座標，供後續模組使用。

二、時序資料管理（Temporal Data Management）模組

時序資料管理模組是系統實現「時光機」功能的核心，其主要功能是儲存、管理與導航多個時間點的交通資料快照。此模組使系統不僅能展示當前路況，更能讓使用者自由瀏覽過去與未來的交通狀態。

本模組的運作流程具體細節說明如下：

(一) 核心服務：TemporalManagementService（時序管理服務）

(二) 每個時間點的資料快照包含以下資訊：

1. data：ProcessedTrafficData 陣列（該時間點的所有道路資料）
2. timestamp：Date 物件（資料的時間戳記）
3. originalTimestamp：原始時間字串（避免時區轉換問題）
4. type：資料類型（DataType.HISTORICAL、DataType.CURRENT、DataType.PREDICTED）
5. confidence：信心度（0-1，用於預測資料）
6. metadata：元資料（資料來源、獲取時間等）

(三) 資料分類與標記

系統將資料分為三種類型，並在視覺化時以不同方式呈現：

1. 歷史資料（HISTORICAL）：過去時間點的實際交通資料。
1. 當前資料（CURRENT）：當前時刻的即時交通資料。
2. 預測資料（PREDICTED）：未來時間點的預測交通資料。

(四) 時間點切換功能實作

系統提供完整的時序導航功能：

1. navigateNext()：前進到下一個時間點。
2. navigatePrevious()：後退到上一個時間點。
3. navigateToIndex(index)：跳轉到指定索引的時間點。
4. navigateToLatest()：跳轉到最新的時間點。

(五) 持久化儲存

為提升使用者體驗，系統支援將時序資料儲存至瀏覽器的 localStorage：

1. 儲存時機：每次新增或更新資料時自動儲存。
2. 資料壓縮：僅儲存必要欄位，減少儲存空間佔用。

3. 版本控制：支援資料格式版本檢查，確保相容性。

4. 容量管理：超過儲存限制時自動壓縮或清理舊資料。

(六) 資料流

1. 輸入：來自資料蒐集模組的 ProcessedTrafficData 陣列。

2. 輸出：TimestampedTrafficData 物件，包含時間戳記與資料類型標記，供視覺化模組使用。

三、3D 視覺化渲染（3D Visualization Rendering）模組

3D 視覺化渲染模組是系統的「視覺呈現層」，負責將交通資料轉換為直觀、互動的 3D 場景。此模組採用 Cesium 3D 地球引擎，提供高效能、高擬真度的地理空間視覺化能力。

(一) 觸發機制（Trigger Mechanism）

本模組為「被動觸發」設計，當時序資料管理模組的活動資料發生變化時，會自動觸發視覺化更新。

(二) Cesium 3D 引擎整合

本系統採用 Cesium 作為核心 3D 渲染引擎，其主要特性包含：

1. 地球模型：提供高精度的 3D 地球模型，支援地形、影像圖層。

2. 向量渲染：支援高效能的向量資料渲染（點、線、面）。

3. 時間軸：內建時間軸控制，支援時序資料動畫。

4. 互動功能：支援滑鼠點擊、懸停、縮放等互動操作。

(三) 道路網絡視覺化

系統將交通資料轉換為 3D 道路網絡視覺化：

1. 道路線條渲染：

(1) 使用 Cesium.PolylineCollection 高效能渲染大量道路線條。

(2) 根據道路類型設定不同的線條寬度（主要道路較寬，次要道路較窄）。

(3) 根據交通速率設定線條顏色（壅塞程度分級）。

2. 壅塞程度顏色分級：

(1) 順暢（smooth）：綠色（#00FF00）- 速率 > 閾值 1。

(2) 車多（heavy）：黃色（#FFFF00）- 速率介於閾值 1 與閾值 2 之間。

(3) 壅塞（congested）：橙色（#FFA500）- 速率介於閾值 2 與閾值 3 之間。

(4) 嚴重壅塞（severe）：紅色（#FF0000）- 速率 ≤ 閾值 3。

(5) 閾值根據道路類型動態調整（如住宅區道路的閾值較低）。

3. 道路資訊標記：

(1) 每條道路線條附加完整的交通資訊（roadId、roadName、avgSpeed、predictTime 等）。

(2) 支援點擊道路顯示詳細資訊彈窗。

(四) 互動功能實作

系統提供豐富的使用者互動功能：

1. 道路點擊事件：

(1) 點擊道路線條時，顯示道路資訊彈窗。

(2) 彈窗內容包含：道路名稱、預測速度、壅塞程度、資料時間、資料來源等。

(3) 支援從彈窗快速縮放至該道路。

2. 道路高亮顯示：

(1) 懸停或點擊道路時，該道路以更亮的顏色高亮顯示。

(2) 支援同時高亮多條道路（如壅塞路段清單）。

3. 視角控制：

(1) 支援滑鼠拖曳旋轉視角。

(2) 支援滾輪縮放。

(3) 支援雙擊快速縮放至特定位置。

(4) 提供「飛行至」功能，平滑動畫飛行至指定道路或區域。

4. 圖層管理：

(1) 支援開關不同類型的道路圖層（主要道路、次要道路等）。

(2) 支援開關其他設施圖層（停車場、YouBike 站點、大眾運輸站點等）。

(五) 效能優化

為確保大規模道路網絡的流暢渲染，系統實作多項效能優化：

1. 批次渲染：使用 Cesium.PolylineCollection 批次渲染所有道路，減少繪製呼叫次數。

2. 視錐剔除：Cesium 自動剔除視野外的物件，減少渲染負擔。

3. LOD（細節層次）：根據縮放層級動態調整道路線條寬度與細節。

4. 資料快取：將 OSM 道路資料快取至記憶體，避免重複查詢。

5. 增量更新：僅更新變化的道路，而非重新渲染整個場景。

(六) 資料流

1. 輸入：來自時序資料管理模組的 TimestampedTrafficData 物件。

2. 輸出：在 Cesium 3D 場景中渲染的道路網絡視覺化，使用者可透過瀏覽器查看。

四、時間控制介面（Time Control Interface）模組

時間控制介面模組提供使用者友善的時間軸控制器，讓使用者能輕鬆瀏覽不同時間點的交通狀態，此模組是時序資料管理模組的使用者介面層。

(一) 觸發機制（Trigger Mechanism）

本模組由使用者操作觸發（點擊時間軸、播放/暫停按鈕等）。

(二) 時間軸控制器元件：PredictTimeSlider（時間軸滑桿元件）

此 Vue 元件提供完整的時間軸控制功能：

1. 時間點顯示：

(1) 以時間軸形式展示所有可用的時間點。

(2) 當前活動時間點以特殊樣式標記（如高亮、放大）。

(3) 支援顯示時間點的資料類型標記（歷史、當前、預測）。

2. 互動操作：

(1) 點擊時間點：跳轉至該時間點。

(2) 拖曳滑桿：連續瀏覽不同時間點。

(3) 播放/暫停按鈕：自動播放時序動畫。

(4) 前進/後退按鈕：單步前進或後退。

(5) 跳至最新按鈕：快速跳轉至最新資料。

3. 時間資訊顯示：

(1) 顯示當前時間點的完整時間資訊（年月日時分秒）。

(2) 顯示當前時間點的資料類型（歷史/當前/預測）。

(3) 顯示當前時間點在時序中的位置（如「3/7」表示第 3 個，共 7 個時間點。

(三) 自動播放功能

系統支援時序資料的自動播放：

1. 播放速度：可設定播放速度（如每秒切換一個時間點）。

2. 循環播放：播放至最後一個時間點後，可選擇循環回到第一個時間點。

3. 暫停與恢復：使用者可隨時暫停與恢復播放。

(四) 狀態同步

時間控制介面與時序資料管理模組保持即時同步：

1. 使用者操作時間軸 → 觸發時序資料管理模組的時間點切換功能 → 更新活動資料 → 觸發 3D 視覺化更新。
2. 時序資料管理模組的狀態變化 → 自動更新時間控制介面的顯示。

(五) 資料流

1. 輸入：使用者的互動操作（點擊、拖曳等）。
2. 輸出：觸發時序資料管理模組的時間點切換功能，間接更新 3D 視覺化。

五、API 服務（API Service Module）模組

API 服務模組是整個數位雙生系統的「數據出口」與「服務層」，扮演著將系統內部的視覺化數據轉化為外部價值的關鍵角色。核心功能是提供一個標準化、安全的 RESTful API 介面，允許經授權的外部應用程式或第三方服務，能夠以程式化的方式輕鬆查詢、介接與整合本系統產出的交通視覺化數據。此模組的穩定性與易用性，是決定整個系統能否成功落地應用的重要一環。

(一) 觸發機制（Trigger Mechanism）

由外部系統的 API 請求觸發，為被動服務。

(二) 運作流程

1. 接收外部系統的 HTTP 請求，並解析請求中的參數（如時間範圍、道路 ID 等）。
2. 根據解析後的參數，向系統內部的資料服務查詢對應的交通資料。
3. 將從資料服務中檢索到的數據，格式化為通用的 JSON 或 CSV 格式。
4. 將格式化後的數據作為 HTTP Response 回傳給請求方。

(三) API 端點設計

系統提供以下主要 API 端點：

1. GET /api/gvp/batch/traffic：獲取批次交通資料。
2. GET /api/gvp/predict/traffic：獲取預測交通資料。
3. GET /api/traffic/current：獲取當前即時交通資料。
4. GET /api/traffic/historical：獲取歷史交通資料。

(四) 認證與授權

所有 API 請求都需要經過認證：

1. 認證方式：API Key 或 JWT Token。
2. 授權機制：基於角色的存取控制（RBAC），不同角色有不同的資料存取權限。

3. 速率限制：防止濫用，限制每個 API Key 的請求頻率（如每分鐘 600 次）。

(五) 資料流

1. 輸入：外部系統發出的、包含認證金鑰與查詢參數的 HTTP API 請求。
2. 輸出：一個標準的 HTTP Response，其 Body 中包含格式化後的 JSON 或 CSV 交通資料。

3.3 軟硬體資源需求

為確保交通數位雙生系統能夠穩定且高效運行，本章將詳細闡述所需的軟體與硬體環境配置。內容涵蓋從開發到正式部署的建議，並針對不同規模的應用情境提供彈性的硬體參考規格。

一、軟體資源需求

本系統所規劃使用軟體，包括前端框架、3D 引擎、程式語言及所有相依套件，均為開源軟體或具有商業友善授權，在標準使用下無需支付高額授權費用。因此系統的建置與擴展在軟體層面成本是相對可控的。

(一) 核心環境

1. 作業系統：Windows 10/11、macOS 10.15+、Ubuntu 20.04+或其他現代作業系統
2. 程式語言：JavaScript / TypeScript
3. Node.js 版本：16.x 或更高版本
4. 套件管理工具：npm 或 yarn

(二) 前端框架與函式庫

1. 核心框架：Vue 3.4+ (Composition API)
2. 狀態管理：Pinia 2.1+
3. 路由管理：Vue Router 4.3+
4. UI 元件庫：Naive UI 2.38+
5. 3D 視覺化引擎
6. Cesium 1.125+：業界領先的開源 3D 地球引擎
7. vue-cesium 3.2+：Cesium 的 Vue 3 整合套件

(三) 開發工具

1. 建置工具：Vite 5.2+
2. TypeScript：5.4+
3. 程式碼編輯器：Visual Studio Code（建議）

4. 版本控制：Git

(四) 資料處理函式庫

1. Papa Parse 5.5+：CSV 檔案解析
2. JSZip 3.10+：ZIP 檔案處理
3. Turf.js 7.2+：地理空間資料處理

(五) 地圖相關函式庫

1. Leaflet 1.9+：2D 地圖函式庫（可選）

(六) 圖表視覺化

1. Chart.js 4.4+：圖表繪製
2. Highcharts 12.1+：進階圖表功能（可選）

(七) 授權與費用

本系統所規劃使用的上述所有軟體元件，均為廣泛使用的開源軟體或具有商業友善授權（如 MIT、Apache 2.0），在標準使用條款下無需支付商業授權費用。唯 Cesium 的某些進階功能（如 Cesium ion 雲端服務）可能需要付費訂閱，但基本的 3D 渲染功能完全免費。

二、硬體資源需求

硬體需求與預計處理的資料量、視覺化複雜度及使用者數量密切相關。以下三種不同情境的建議規格，以供參考。

(一) 情境一：開發與測試環境，此規格適用於開發人員進行功能開發、單元測試或小規模數據驗證，預計處理路段數量在 1000 個以下。

1. 架構：建議採用單機架構，所有服務（前端開發伺服器、資料處理）運行在同一台主機上
2. CPU：現代 x86-64 架構，至少 4 核心處理器（如 Intel Core i5 或 AMD Ryzen 5）
3. 記憶體：至少 8 GB RAM（建議 16 GB）
4. 儲存空間：至少 256 GB SSD，以確保快速的專案建置與模組載入速度
5. 顯示卡：支援 WebGL 2.0 的獨立顯示卡或整合顯示卡（如 Intel UHD Graphics 630 或更高）
6. 網路：標準寬頻網路連線（用於下載套件與存取外部 API）

(二) 情境二：中小型正式環境，此規格適用於正式部署，預計處理數千個路段的即時視覺化，使用者數量適中（同時在線使用者 < 100）。

1. Web 伺服器（前端）：

(1) 架構：建議採用前後端分離架構，前端靜態檔案部署至 Web 伺服器，後端 API 服務部署至應用伺服器

(2) CPU：4 核心以上伺服器級 CPU

(3) 記憶體：至少 8 GB RAM

(4) 儲存空間：100 GB 以上 SSD（用於靜態檔案）

2. 應用伺服器（後端 API）：

(1) 架構：建議採用前後端分離架構，前端靜態檔案部署至 Web 伺服器，後端 API 服務部署至應用伺服器

(2) CPU：8 核心以上伺服器級 CPU

(3) 記憶體：至少 16 GB RAM

(4) 儲存空間：500 GB 以上 SSD

(5) 網路：100 Mbps 以上頻寬

(6) 負載平衡器：建議使用 Nginx 或 HAProxy 進行負載平衡與反向代理

(三) 情境三：中小型正式環境，大型正式環境（大規模部署）此規格為高運算效能的配置，適用於處理數萬個路段的即時視覺化，需應對高併發使用者存取（同時在線使用者 ＞100）與龐大的資料吞吐量。

1. 應用伺服器叢集（後端 API）：

(1) 架構：分散式架構，前端靜態檔案部署至 CDN，後端 API 服務部署至多台應用伺服器叢集

(2) CPU：16 核心以上高效能伺服器級 CPU(如 Intel Xeon 或 AMD EPYC)

(3) 記憶體：至少 32 GB RAM（建議 64 GB）

(4) 儲存空間：1 TB 以上 NVMe SSD

(5) 伺服器數量：至少 3 台，支援水平擴展

2. 資料庫伺服器（若需要）：

(1) 架構：分散式架構，前端靜態檔案部署至 CDN，後端 API 服務部署至多台應用伺服器叢集

(2) CPU：8 核心以上伺服器級 CPU

(3) 記憶體：至少 32 GB RAM

(4) 儲存空間：2 TB 以上 SSD（主資料）+ 10 TB HDD（備份）

(5) 網路：1 Gbps 以上頻寬

(6) 負載平衡器：企業級負載平衡器（如 F5、Citrix ADC）或雲端負載平衡服務（如 AWS ELB、Azure Load Balancer）

表 1 硬體資源需求彙整表

硬體規格	情境一： 開發與測試	情境二： 中小型正式環境	情境三：大型正式環境（大規模部署）
適用情境	功能開發、單元測試、小規模數據驗證 (處理路段數量< 1000)	正式部署，處理數千個路段的視覺化 (同時在線使用者<100)	正式部署，處理數萬個路段的視覺化 (同時在線使用者>100)
架構	單機架構 (所有服務運行於同一主機)	前後端分離架構 (前端 Web 伺服器 + 後端應用伺服器)	分散式架構 (CDN + 應用伺服器叢集 + 資料庫伺服器)
主伺服器 CPU	現代 x86-64 架構，至少 4 核心處理器(如 Intel Core i5 或 AMD Ryzen 5)	Web: 4 核心以上 API: 8 核心以上	API 叢集: 16 核心以上 (Intel Xeon 或 AMD EPYC)
主伺服器記憶體	至少 8 GB RAM（建議 16 GB）	Web: 8 GB API: 16 GB	API 叢集: 32 GB(建議 64 GB) DB: 32 GB
主伺服器儲存空間	至少 256 GB SSD，以確保快速的專案建置與模組載入速度	Web: 100 GB SSD API: 500 GB SSD	API 叢集: 1 TB NVMe SSD DB: 2 TB SSD + 10 TB HDD
顯示卡	支援 WebGL 2.0 的獨立顯示卡或整合顯示卡（如 Intel UHD Graphics 630 或更高）	不需要(伺服器端)	不需要(伺服器端)
網路	標準寬頻網路連線（用於下載套件與存取外部 API）	100 Mbps 以上頻寬	1 Gbps 以上頻寬
其他	-	負載平衡器：建議使用 Nginx 或 HAProxy 進行負載平衡與反向代理	CDN 服務 + 企業級負載平衡器
硬體購置成本	約 2-3 萬元 （開發用電腦）	約 15-25 萬元	約 100-200 萬元 （不含 CDN 年費）

3.4 功能驗證與效果評估方法

一、功能驗證方法

功能驗證旨在確認系統的每一個模組與整體流程，是否都按照設計規格正確無誤地運行。建議採用分層的測試策略，從最小的程式單元到完整的端到端流程進行

驗證。

(一) 單元測試 (Unit Testing)

單元測試主要目的為驗證系統中最小的獨立程式碼單元（如一個函式或一個類別方法）的功能是否如預期般正確。完整的單元測試能確保程式碼品質、降低後續整合難度與維護成本。建議測試對象清單包含：

1. 資料蒐集與整合模組
   (1) API 呼叫函式：驗證 GvpBatchApiService、GvpPredictApiService 能否正確呼叫外部 API 並解析回應
   (2) 資料轉換函式：驗證將 API 回應轉換為 ProcessedTrafficData 格式的邏輯是否正確
   (3) OSM -資料查詢函式：驗證能否正確從 OSM 快取中查詢道路資訊
2. 時序資料管理模組
   (1) 資料新增函式：驗證 addCurrentData、addPredictedData、addHistoricalData 能否正確新增資料至時序集合
   (2) 時間點切換函式：驗證 navigateNext、navigatePrevious、navigateToIndex 能否正確切換活動資料
   (3) 資料清理函式：驗證 removeExpiredPredictions 能否正確移除過期的預測資料
   (4) 持久化函式：驗證 saveToStorage、loadFromStorage 能否正確儲存與載入資料
3. 3D 視覺化渲染模組
   (1) 顏色計算函式：驗證根據速率計算壅塞程度與顏色的邏輯是否正確
   (2) 座標轉換函式：驗證經緯度座標轉換為 Cesium Cartesian3 座標的邏輯是否正確
   (3) 道路渲染函式：驗證能否正確將 ProcessedTrafficData 轉換為 Cesium PolylineCollection
4. 時間控制介面模組
   (1) 時間點計算函式：驗證能否正確從時序資料中提取可用的時間點清單
   (2) 時間格式化函式：驗證時間戳記格式化為顯示字串的邏輯是否正確
5. API 服務模組
   (1) 參數驗證函式：驗證 API 端點能否正確驗證請求參數（如時間範圍、

道路 ID）

(2) 資料格式化函式：驗證能否正確將內部資料格式轉換為 JSON 或 CSV 格式

(3) 認證函式：驗證 API Key 或 JWT Token 的認證邏輯是否正確

(二) 整合測試 (Integration Testing)

整合測試的目的係為驗證不同模組之間介接的正確性，確保資料流在模組間的傳遞無誤。建議測試清單包含：

1. 資料蒐集與時序管理整合測試

(1) 驗證流程：呼叫資料蒐集模組獲取交通資料，然後將資料傳遞給時序管理模組

(2) 檢核點：確認時序管理模組能成功接收並儲存資料，且資料格式正確

2. 時序管理與視覺化渲染整合測試

(1) 驗證流程：在時序管理模組中切換活動資料，觀察視覺化渲染模組是否自動更新

(2) 檢核點：確認視覺化場景中的道路顏色與資訊與活動資料一致

3. 時間控制介面與時序管理整合測試

(1) 驗證流程：操作時間控制介面（如點擊時間點、播放按鈕），觀察時序管理模組的狀態變化

(2) 檢核點：確認時序管理模組的活動資料索引與時間控制介面的顯示同步

4. API 服務與資料蒐集整合測試

(1) 驗證流程：透過 API 端點請求交通資料，驗證回應內容

(2) 檢核點：確認 API 回應的資料與資料蒐集模組獲取的資料一致

(三) 端到端測試 (End-to-End Testing)

端到端測試之目的為模擬真實世界的使用情境，驗證整個系統從起點到終點的完整自動化流程是否順暢。建議測試清單包含：

1. 完整視覺化流程驗證

(1) 前置作業：確認系統已正確配置 API 金鑰與資料來源

(2) 觸發執行：啟動系統，等待資料載入完成

(3) 過程監控：觀察瀏覽器控制台日誌，確認資料蒐集、時序管理、視覺化渲染各模組均正常運作

(4) 結果驗證：確認 3D 場景中顯示道路網絡，且道路顏色與交通資料一致

2. 時序回放功能驗證

(1) 前置作業：確認系統已載入多個時間點的資料

(2) 觸發執行：操作時間控制介面，切換不同時間點

(3) 過程監控：觀察 3D 場景是否隨時間點切換而更新

(4) 結果驗證：確認每個時間點的道路顏色與該時間點的交通資料一致

3. 互動功能驗證

(1) 前置作業：確認系統已正常顯示 3D 場景

(2) 觸發執行：點擊道路、懸停道路、縮放視角等互動操作

(3) 過程監控：觀察系統是否正確回應互動操作（如顯示道路資訊彈窗、高亮道路等）

(4) 結果驗證：確認互動功能符合設計規格

4. API 服務驗證

(1) 前置作業：確認 API 服務已啟動

(2) 觸發執行：使用 API 測試工具（如 Postman、curl）發送請求

(3) 過程監控：觀察 API 回應時間與內容

(4) 結果驗證：確認 API 回應格式正確，且資料與系統內部資料一致

二、效果評估方法

效果評估旨在量化分析系統的表現，主要分為「視覺化品質」與「系統運行效能」兩大維度。

(一) 視覺化品質評估

1. 評估方式：透過使用者測試與專家評審，評估系統視覺化效果是否達到預期。

2. 評估指標：

(1) 視覺清晰度：道路線條是否清晰可辨，顏色分級是否明確

(2) 資訊完整性：道路資訊彈窗是否包含所有必要資訊

(3) 互動流暢度：互動操作（點擊、縮放等）是否流暢無延遲

(4) 使用者體驗：整體使用體驗是否良好，是否易於理解與操作

3. 合格標準：

(1) 視覺清晰度：至少 80% 的測試使用者認為視覺效果清晰

(2) 資訊完整性：道路資訊彈窗包含至少 5 項關鍵資訊（道路名稱、速率、壅塞程度、時間、資料來源）

(3) 互動流暢度：互動操作的回應時間 $< 100ms$

(4) 使用者體驗：使用者滿意度評分 ≥4.0（滿分 5.0）

(二) 系統效能評估

1. 評估方式：透過效能測試工具與資源監控，評估系統在不同負載下的運行效率與穩定性。

2. 評估指標：

(1) 頁面載入時間：首次載入頁面至 3D 場景顯示的時間

(2) 資料更新延遲：從資料來源更新到視覺化更新的時間

(3) 幀率（FPS）：3D 場景的渲染幀率，影響視覺流暢度

(4) 記憶體使用量：系統運行時的記憶體佔用

(5) CPU 使用率：系統運行時的 CPU 佔用

(6) 網路流量：資料傳輸的網路流量

3. 合格標準：

(1) 頁面載入時間：＜5 秒（在標準網路環境下）

(2) 資料更新延遲：＜2 秒

(3) 幀率（FPS）：≥30 FPS（在標準硬體環境下）

(4) 記憶體使用量：＜2 GB（處理 10000 個路段時）

(5) CPU 使用率：平均 ＜50%

(6) 網路流量：初始載入 ＜50 MB，後續更新 ＜5 MB/次

3.5 常見問題與風險因應

任何複雜系統在長期維運中都會面臨各種挑戰。本節旨在透過建置經驗說明交通數位雙生系統在資料、視覺化、效能與服務層面可能遇到的常見問題，並提供具體的技術應對策略。

一、資料來源連線問題

(一) 風險描述：外部 API 可能因網路問題、服務維護或流量限制而暫時無法存取，導致系統無法獲取最新的交通資料。

(二) 潛在影響

1. 視覺化停滯：無法更新 3D 場景，使用者看到的是過時的交通資訊

2. 時序資料中斷：時序資料集合中出現時間缺口

(三) 因應對策

1. 實作重試機制：當 API 呼叫失敗時，自動重試 3 次，每次間隔 5 秒

2. 快取機制：將最近一次成功獲取的資料快取至本地，當 API 無法存取時，使用快取資料
3. 降級策略：當所有資料來源都無法存取時，顯示明確的錯誤訊息給使用者，並提供手動重新整理按鈕
4. 監控與告警：建立 API 連線狀態監控，當連線失敗超過一定次數時，自動發送告警通知給維運人員

二、3D 渲染效能問題

(一) 風險描述：當道路數量過多（如超過 10000 條）或使用者的硬體效能不足時，3D 場景的渲染可能變得緩慢，導致幀率下降、互動延遲。

(二) 潛在影響

1. 視覺卡頓：3D 場景更新不流暢，影響使用者體驗
2. 瀏覽器崩潰：在極端情況下，可能導致瀏覽器記憶體不足而崩潰

(三) 因應對策

1. 批次渲染優化：使用 Cesium.PolylineCollection 批次渲染所有道路，減少繪製呼叫次數
2. 視錐剔除：確保 Cesium 的視錐剔除功能正常運作，自動隱藏視野外的物件
   (1) LOD（細節層次）：根據縮放層級動態調整道路線條寬度，遠距離時使用較細的線條
3. 分層載入：將道路按類型或區域分層，允許使用者選擇性載入需要的圖層
4. 效能監控：在開發者模式下顯示即時 FPS 與記憶體使用量，協助診斷效能問題
5. 硬體檢測：在系統啟動時檢測使用者的硬體效能，若效能不足則自動降低視覺化品質（如減少道路數量、降低線條寬度）

三、時序資料同步問題

(一) 風險描述：當多個資料來源（如 GVP Batch API 與 GVP Predict API）的資料時間戳記不一致時，可能導致時序資料集合中的資料順序混亂。

(二) 潛在影響

1. 時間軸錯亂：時間控制介面顯示的時間點順序不正確
2. 視覺化跳躍：切換時間點時，3D 場景出現不連續的跳躍

(三) 因應對策

1. 時間戳記標準化：統一使用 ISO 8601 格式的時間戳記，並在資料轉換時進行標準化處理

2. 時區處理：明確定義系統使用的時區（建議使用 UTC 或本地時區），並在所有資料處理流程中保持一致
3. 資料排序：在時序資料集合中始終按時間戳記排序資料，確保時間軸的正確性
4. 重複資料檢測：在新增資料至時序集合前，檢查是否已存在相同時間戳記的資料，避免重複

四、瀏覽器相容性問題

(一) 風險描述：不同瀏覽器對 WebGL、CSS、JavaScript 的支援程度不同，可能導致系統在某些瀏覽器上無法正常運作。

(二) 潛在影響

1. 視覺化無法顯示：在不支援 WebGL 2.0 的瀏覽器上，Cesium 無法初始化
2. 樣式錯亂：在某些瀏覽器上，CSS 樣式顯示不正確
3. 功能異常：某些 JavaScript 功能在舊版瀏覽器上無法運作

(三) 因應對策

1. 瀏覽器檢測：在系統啟動時檢測使用者的瀏覽器版本與 WebGL 支援情況
2. 降級提示：若瀏覽器不支援必要功能，顯示明確的升級建議訊息
3. Polyfill：使用 Polyfill 函式庫(如 core-js)提供舊版瀏覽器缺少的 JavaScript 功能
4. 跨瀏覽器測試：在開發階段進行跨瀏覽器測試，確保系統在主流瀏覽器上均能正常運作

五、API 服務過載

(一) 風險描述：當同時存取系統的使用者數量過多時，API 服務可能因請求過載而回應緩慢或無法回應。

(二) 潛在影響

1. 回應延遲：使用者等待時間過長，影響使用體驗
2. 服務中斷：在極端情況下，API 服務可能完全無法回應

(三) 因應對策

1. 負載平衡：使用負載平衡器(如 Nginx、HAProxy)將請求分散至多台應用伺服器
2. 快取機制：對於不常變動的資料（如 OSM 道路資訊），使用快取減少資料庫查詢
3. 速率限制：實作 API 速率限制，防止單一使用者或惡意攻擊者佔用過多資源
4. 水平擴展：根據負載情況，動態增加應用伺服器數量
5. CDN 加速：將前端靜態檔案部署至 CDN，減少伺服器負擔

第四章 整合與落地應用作法

4.1 整合應用模式

為使本數位雙生系統的產出能無縫融入交通管理單位的日常運作，建議在系統建置完成後，可採用以下整合模式，並結合實際情境進行應用。

一、前端嵌入模式

此模式是彈性最高、整合最簡單且最符合現代 Web 應用架構的整合方式。由交通管理單位的既有資訊系統（如交控平台）透過 iframe 或 Web Component 的方式，直接嵌入本數位雙生系統的前端頁面。

(一) 整合架構：交控中心應用系統→ iframe/Web Component →數位雙生前端頁面

(二) 優勢

1. 技術解耦：兩邊系統可獨立開發與升級，只要介面規格不變，彼此不互相影響
2. 快速整合：無需修改既有系統的後端程式碼，僅需在前端頁面中嵌入程式碼
3. 視覺一致：可透過 CSS 調整嵌入頁面的樣式，使其與既有系統的視覺風格一致

(三) 嵌入方式

1. 使用 iframe 標籤嵌入完整頁面
2. 使用 Web Component 嵌入特定元件（需系統支援）
3. 跨域通訊：若需要在嵌入頁面與父頁面之間傳遞資料，可使用 postMessage API 進行跨域通訊
4. 認證整合：可透過 URL 參數或 postMessage 傳遞認證 Token，實現單一登入（SSO）

二、API 介接模式

此模式適用於需要將數位雙生系統的資料整合至既有系統的後端，或需要在既有系統中自行實作視覺化介面的情況。

(一) 整合架構：交控中心應用系統 → API Gateway → 數位雙生 API 服務

(二) 優勢：

1. 資料整合：可將數位雙生系統的資料與既有系統的其他資料整合，進行更深入的分析
2. 客製化視覺化：可根據既有系統的需求，自行實作視覺化介面

(三) 認證與加密：

1. 認證機制：建議採用 API Key、JWT (JSON Web Token) 或 OAuth2 等標準機

制進行客戶端身份驗證

2. 傳輸加密：所有 API 通訊皆應強制使用 HTTPS/TLS 進行全程加密，確保資料傳輸安全

(四) 資料查詢與回傳格式：

1. API 呼叫範例：使用 GET 方法，並透過 Query String 傳遞參數

2. 回傳格式範例 (JSON Schema)：回傳的 JSON 應具備清楚且固定的結構

(五) 流量管理與錯誤處理：

1. 速率限制 (Rate Limit)：應在 API Gateway 或服務本身設置速率限制（如每分鐘 600 次請求），避免因客戶端程式錯誤或惡意請求導致服務過載

2. 錯誤碼設計：定義標準的 HTTP 錯誤碼，便於客戶端進行錯誤處理

三、混合模式

結合前端嵌入與 API 介接的優勢，既在既有系統中嵌入數位雙生的視覺化頁面，又透過 API 獲取資料進行深度整合。

(一) 整合架構：交控中心應用系統 → iframe（視覺化） + API（資料整合）

(二) 優勢：

1. 最佳體驗：使用者可直接在既有系統中查看 3D 視覺化，同時系統後端可獲取資料進行分析

2. 彈性最高：可根據不同需求選擇最適合的整合方式

4.2 系統維運建議

為確保本系統在建置完成後能長期穩定地提供高品質服務，需建立一套涵蓋資料、功能、效能到設備的完整維運計畫。

一、資料來源監控

建立自動化的資料來源監控模組，定期檢測外部 API 的可用性與回應時間。若發現 API 無法存取或回應過慢，系統應立即透過 Email 或即時通訊軟體通報維運人員處理。

二、系統功能檢測

設置自動化的系統健康檢查（Health Check）排程，每日在離峰時段執行。此排程可執行一個簡化版的端到端測試，包括以下項目，若健康檢查失敗，應立即通報系統維運人員，進行問題排查。

(一) 自動載入系統

(二) 檢查 3D 場景是否正常顯示

(三) 驗證時間控制功能是否正常運作

三、效能監控

建立效能監控機制，持續追蹤系統的關鍵效能指標（如頁面載入時間、幀率、記憶體使用量）。若效能指標持續下滑並觸發警報閾值，則應通知相關人員評估是否需要進行效能優化或硬體升級。

四、設備保養檢查

為確保系統硬體設施的長期穩定與可靠，應建立一套兼具自動化健康監控與週期性實體巡檢的雙軌保養計畫。自動化監控應透過部署監控代理程式，對伺服器的關鍵指標進行全天候追蹤，包括以下項目，並設定警報閾值以即時通報異常。

(一) 硬碟健康度

(二) 磁碟空間使用率

(三) CPU 溫度

(四) 電源狀態

五、異常即時通報

將上述所有監控項目（資料、功能、效能、硬體）的警報，整合至一個統一的告警平台。設定明確的觸發條件，包括以下項目，一旦觸發，系統應能立即透過 Email 或即時通訊軟體發出警報給對應的權責人員。

(一) 「API 連線失敗超過 3 次」

(二) 「頁面載入時間超過 10 秒」

(三) 「CPU 使用率持續高於 80%」

六、版本更新管理

為確保系統在功能迭代與安全升級過程中的穩定性，應建立嚴謹的版本更新管理流程。此流程應以 Git Flow 等標準化分支模型為基礎，並導入持續整合與持續交付（CI/CD）的實踐。版本更新流程如下：

(一) 所有程式碼變更在合併前需經審查

(二) 透過自動化管線在預備（Staging）環境中完成所有單元、整合及端到端測試

(三) 唯有通過完整驗證的版本，才能被批准部署至正式（Production）環境