海大教室借用平台

設計文件

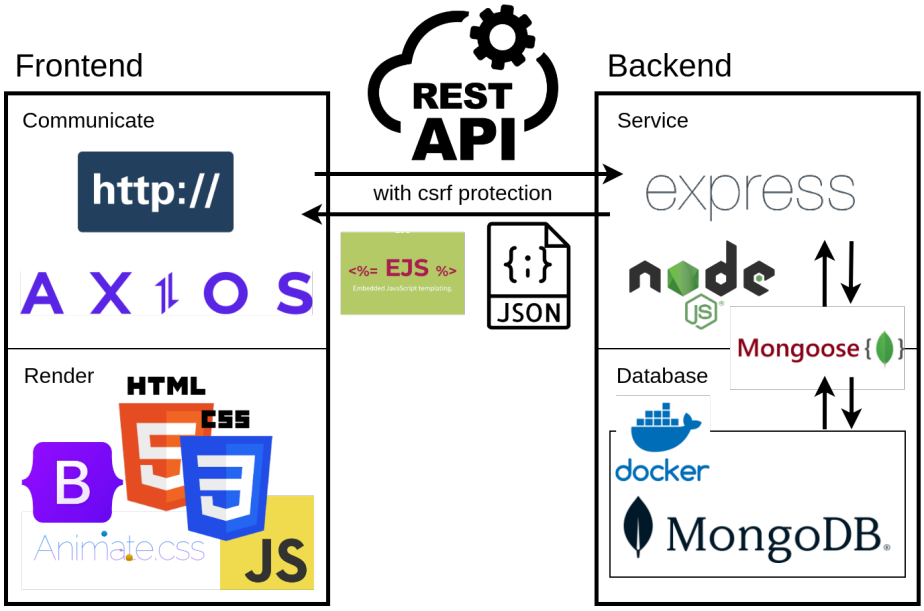
專案名稱	海大教室借用平台
撰寫日期	November 30, 2022
發展者	曾昱翔、林暐傑、陳鈺翔、張銀軒、黃見弘

版次變更紀錄

版次	變更項目	變更日期
0.1	初版	2022/10/04
0.1.1	新增系統模型與架構	2022/11/28
0.1.2	新增介面需求與設計	2022/11/29
0.1.3	新增資料設計	2022/11/30
0.1.4	新增類別圖設計	2022/11/30
0.1.5	新增實作方案和設計議題	2022/11/30

目錄

1 系統模型與架構 (System Model/System Architecture)



前後端透過瀏覽器的 HTTP request 和 axios 提供的 request 方法與後端溝通，後端透過 express 提供的路由來處理前端的要求。

實際傳輸的方式包含了 RESTful API 和直接用 ejs render 後的結果當成網頁內容。

每個頁面透過瀏覽器發送的 request 來得到後端 ejs render 完的 HTML 內容，頁面中需要純前端處理的部分則以 axios 來傳送 request ，接收後端 render 完的 response 後直接操作 DOM 來改變頁面內容。

1.1 後端架構

後端架構主要分為兩個部分：後端伺服器和資料庫。

1.1.1 後端伺服器

伺服器為 Node.js express 搭配 ejs 來實作，主要負責處理前端的 request 並回傳給前端。

後端架構遵循 express-generator 產生出的 MVC 架構，將後端大致分為三個部分：routes、views 和 models 。

1.1.2 資料庫

透過 docker 來管理本地端的資料庫，使用的是 mongodb ，並在 express app 中以 mongoose ODM 來操作資料庫。

資料庫連線分為 /test 和 /NTOUClassroomBorrowing ，分別用來測試和正式使用。

1.2 前端架構

前端透過 Bootstrap 5 和基本的 HTML、CSS、JavaScript 來實作，主要負責處理使用者的操作並將結果傳送給後端。

部份動畫透過 animate.css 提供，純前端處理 request 的部份則以 axios 與後端溝通，收取 response 來更新頁面內容，或是彈出提示訊息。

2 介面需求與設計 (Interface Requirements and Design)

2.1 帳號管理

介面名稱 (Interface Name)	介面提供者 (Interface Provider)	介面使用者 (Interface Consumer)
登入/註冊畫面	User 模組	登入/註冊頁面
連結方式 (Connection Type)	輸入資料 (Input Data)	輸出資料 (Output Data)
`HTTP GET`	`None`	`Plain HTML`
URL		
`/users/session`		
介面描述 (Interface Description)		
回傳包含登入和註冊表單的頁面		

介面名稱 (Interface Name)	介面提供者 (Interface Provider)	介面使用者 (Interface Consumer)
使用者註冊	User 模組	註冊表單
連結方式 (Connection Type)	輸入資料 (Input Data)	輸出資料 (Output Data)

`HTTP POST`	`{`
	`"id": "10987654",`
	`"password": "password",`
	`"username": "name",`
	`"email": "email@mail.host",`
	`"phone": "0912345678"`
	`}`
		302 `/users/session`
		500 Internal Server Error

URL		
`/users/register`		
介面描述 (Interface Description)		
接收註冊表單，並將使用者資料存入資料庫		

介面名稱 (Interface Name)	介面提供者 (Interface Provider)	介面使用者 (Interface Consumer)
使用者登入	User 模組	登入表單
連結方式 (Connection Type)	輸入資料 (Input Data)	輸出資料 (Output Data)

`HTTP POST`	`{`
	`"id": "10987654",`
	`"password": "password",`
	`}`
		302 `/home`
		302 `/users/sesion`

URL		
`/users/login`		
介面描述 (Interface Description)		
接收登入表單，並將使用者資料存入 session		

介面名稱 (Interface Name)	介面提供者 (Interface Provider)	介面使用者 (Interface Consumer)
使用者登出	User 模組	登出按鈕
連結方式 (Connection Type)	輸入資料 (Input Data)	輸出資料 (Output Data)
HTTP POST	{}	302 /users/sesion
URL		
/users/logout		
介面描述 (Interface Description)		
刪除 session 中之使用者資訊，並回到登入畫面		

介面名稱 (Interface Name)	介面提供者 (Interface Provider)	介面使用者 (Interface Consumer)
待審使用者列表	Admin 模組	Admin/Account 頁面
連結方式 (Connection Type)	輸入資料 (Input Data)	輸出資料 (Output Data)
HTTP GET	None	Plain HTML
URL		
/admin/account		
介面描述 (Interface Description)		
列出尚未審核之帳號		

介面名稱 (Interface Name)	介面提供者 (Interface Provider)	介面使用者 (Interface Consumer)
查詢使用者資訊	Admin 模組	Admin/Account 頁面
連結方式 (Connection Type)	輸入資料 (Input Data)	輸出資料 (Output Data)
HTTP GET	None	Plain HTML
URL		
/admin/account/:id		
介面描述 (Interface Description)		
查詢指定使用者之個人資訊		

介面名稱 (Interface Name)	介面提供者 (Interface Provider)	介面使用者 (Interface Consumer)
審核使用者	Admin 模組	Admin/Account 頁面
連結方式 (Connection Type)	輸入資料 (Input Data)	輸出資料 (Output Data)

HTTP POST	{
	verified: true
	}
		200 OK
		500 Internal Server Error

URL		
/admin/account/:id		
介面描述 (Interface Description)		
修改指定使用者審核狀態		

2.2 大樓、樓層與教室編輯

介面名稱 (Interface Name)	介面提供者 (Interface Provider)	介面使用者 (Interface Consumer)
編輯頁面	Admin 模組	Admin/Classroom 頁面
連結方式 (Connection Type)	輸入資料 (Input Data)	輸出資料 (Output Data)
HTTP GET	None	Plain HTML
URL		
/admin/classroom		
介面描述 (Interface Description)		
對大樓、樓層與教室進行編輯之主要操作頁面		

介面名稱 (Interface Name)	介面提供者 (Interface Provider)	介面使用者 (Interface Consumer)
新增大樓	Admin 模組	Admin/Classroom 頁面
連結方式 (Connection Type)	輸入資料 (Input Data)	輸出資料 (Output Data)
HTTP POST	{ "name": "電機二館" }	200 OK
URL		
/admin/classroom/building		
介面描述 (Interface Description)		
在資料庫中新增大樓		

介面名稱 (Interface Name)	介面提供者 (Interface Provider)	介面使用者 (Interface Consumer)
刪除大樓	Admin 模組	Admin/Classroom 頁面
連結方式 (Connection Type)	輸入資料 (Input Data)	輸出資料 (Output Data)
HTTP DELETE	None	200 OK
URL		
/admin/classroom/building/:id		
介面描述 (Interface Description)		
在資料庫中刪除大樓		

介面名稱 (Interface Name)	介面提供者 (Interface Provider)	介面使用者 (Interface Consumer)
新增樓層	Admin 模組	Admin/Classroom 頁面
連結方式 (Connection Type)	輸入資料 (Input Data)	輸出資料 (Output Data)

HTTP POST	<pre>{
	"building": "電機二館",
	"floor": "一樓",
	}</pre>
		200 OK

URL		
/admin/classroom/floor		
介面描述 (Interface Description)		
在指定的大樓中新增樓層		

介面名稱 (Interface Name)	介面提供者 (Interface Provider)	介面使用者 (Interface Consumer)
刪除樓層	Admin 模組	Admin/Classroom 頁面
連結方式 (Connection Type)	輸入資料 (Input Data)	輸出資料 (Output Data)
HTTP DELETE	None	200 OK
URL		
/admin/classroom/building/:id/floor/:fid		
介面描述 (Interface Description)		
刪除大樓中指定的樓層		

介面名稱 (Interface Name)	介面提供者 (Interface Provider)	介面使用者 (Interface Consumer)
空教室表單	Admin 模組	Admin/Classroom 頁面
連結方式 (Connection Type)	輸入資料 (Input Data)	輸出資料 (Output Data)
HTTP GET	None	Plain HTML
URL		
/admin/classroom/empty		
介面描述 (Interface Description)		
準備建立新教室時所需的表單		

介面名稱 (Interface Name)	介面提供者 (Interface Provider)	介面使用者 (Interface Consumer)
教室資訊	Admin 模組	Admin/Classroom 頁面
連結方式 (Connection Type)	輸入資料 (Input Data)	輸出資料 (Output Data)
HTTP GET	None	Plain HTML
URL		
/admin/classroom/building/:id/floor/:fid/classroom/:cid		
介面描述 (Interface Description)		
查詢指定教室的資訊		

介面名稱 (Interface Name)	介面提供者 (Interface Provider)	介面使用者 (Interface Consumer)
建立新教室	Admin 模組	Admin/Classroom 頁面
連結方式 (Connection Type)	輸入資料 (Input Data)	輸出資料 (Output Data)

HTTP POST	<pre>{
	"building": "電機二館",
	"floor": "一樓",
	"name": "INS 103",
	"capacity": 30,
	"schedule":
	[
	[null, lesson2, ...],
	[lesson1, null, ...], ...
	],
	"options": ["computer"]
	}</pre>
		200 OK

URL		
/admin/classroom		
介面描述 (Interface Description)		
在指定的樓層中建立新教室		

介面名稱 (Interface Name)	介面提供者 (Interface Provider)	介面使用者 (Interface Consumer)
更新教室	Admin 模組	Admin/Classroom 頁面
連結方式 (Connection Type)	輸入資料 (Input Data)	輸出資料 (Output Data)

HTTP PUT	<pre>{
	"building": "電機二館",
	"floor": "一樓",
	"name": "INS 103",
	"capacity": 30,
	"schedule":
	[
	[null, lesson2, ...],
	[lesson1, null, ...], ...
	],
	"options": ["computer"]
	}</pre>
		200 OK

URL		
/admin/classroom/building/:id/floor/:fid/classroom/:cid		
介面描述 (Interface Description)		
更新指定教室的資訊		

介面名稱 (Interface Name)	介面提供者 (Interface Provider)	介面使用者 (Interface Consumer)
刪除教室	Admin 模組	Admin/Classroom 頁面
連結方式 (Connection Type)	輸入資料 (Input Data)	輸出資料 (Output Data)
HTTP DELETE	None	200 OK
URL		
/admin/classroom/building/:id/floor/:fid/classroom/:cid		
介面描述 (Interface Description)		
刪除指定教室		

2.3 課程管理

介面名稱 (Interface Name)	介面提供者 (Interface Provider)	介面使用者 (Interface Consumer)
課程列表	Admin 模組	Admin/Lesson 頁面
連結方式 (Connection Type)	輸入資料 (Input Data)	輸出資料 (Output Data)
HTTP GET	None	Plain HTML
URL		
/admin/lesson		
介面描述 (Interface Description)		
列出所有課程		

介面名稱 (Interface Name)	介面提供者 (Interface Provider)	介面使用者 (Interface Consumer)
新增課程	Admin 模組	Admin/Lesson 頁面
連結方式 (Connection Type)	輸入資料 (Input Data)	輸出資料 (Output Data)
HTTP POST	{ "name": "課程A", "teacher": "王大明", "description": "課程A的說明" }	200 OK
URL		
/admin/lesson		
介面描述 (Interface Description)		
新增一個課程於資料庫中		

介面名稱 (Interface Name)	介面提供者 (Interface Provider)	介面使用者 (Interface Consumer)
編輯課程	Admin 模組	Admin/Lesson 頁面
連結方式 (Connection Type)	輸入資料 (Input Data)	輸出資料 (Output Data)
HTTP PUT	{ "name": "課程A", "teacher": "王大明", "description": "課程A的說明" }	200 OK
URL		
/admin/lesson/:id		
介面描述 (Interface Description)		
編輯指定課程的資訊		

介面名稱 (Interface Name)	介面提供者 (Interface Provider)	介面使用者 (Interface Consumer)
查詢課程	Admin 模組	Admin/Lesson 頁面
連結方式 (Connection Type)	輸入資料 (Input Data)	輸出資料 (Output Data)
HTTP GET	None	Plain HTML
URL		
/admin/lesson/:id		
介面描述 (Interface Description)		
查詢指定課程的資訊		

介面名稱 (Interface Name)	介面提供者 (Interface Provider)	介面使用者 (Interface Consumer)
刪除課程	Admin 模組	Admin/Lesson 頁面
連結方式 (Connection Type)	輸入資料 (Input Data)	輸出資料 (Output Data)
HTTP DELETE	None	200 OK
URL		
/admin/lesson/:id		
介面描述 (Interface Description)		
刪除指定課程		

介面名稱 (Interface Name)	介面提供者 (Interface Provider)	介面使用者 (Interface Consumer)
列出課表	Home 模組	Home 頁面
連結方式 (Connection Type)	輸入資料 (Input Data)	輸出資料 (Output Data)
HTTP GET	None	Plain HTML
URL		
/home/curriculum		
介面描述 (Interface Description)		
列出當天課表		

介面名稱 (Interface Name)	介面提供者 (Interface Provider)	介面使用者 (Interface Consumer)
列出課表	Home 模組	Home 頁面
連結方式 (Connection Type)	輸入資料 (Input Data)	輸出資料 (Output Data)
HTTP GET	None	Plain HTML
URL		
/home/curriculum/:date		
介面描述 (Interface Description)		
列出指定日期的課表		

介面名稱 (Interface Name)	介面提供者 (Interface Provider)	介面使用者 (Interface Consumer)
課程資訊	Home 模組	Home 頁面
連結方式 (Connection Type)	輸入資料 (Input Data)	輸出資料 (Output Data)
HTTP GET	None	Plain HTML
URL		
/home/lesson/:id		
介面描述 (Interface Description)		
列出指定課程的資訊		

3　流程設計 (Process Design)

4　使用者畫面設計 (User Interface Design)

5 資料設計 (Data Design)

5.1 檔案結構

- app.js: 主程式
- package.json: 套件設定檔
- public: 靜態檔案
- routes: 路由
- views: 樣板
- models: 資料庫模組
- middlewares: 中介軟體
- config: 設定檔
- bin: 執行檔
- node_modules: 套件

5.2 資料庫設計 (Database Design)

Building

```
const buildingSchema = new mongoose.Schema({
  name: {
    type: String,
    required: true,
    unique: true,
    trim: true,
    minlength: 3,
  },
  floors: [
    {
      type: mongoose.Schema.Types.ObjectId,
      ref: "Floor",
    },
  ],
});
```

Floor

```
const floorSchema = new mongoose.Schema({
  name: {
    type: String,
    required: true,
  },
  building: {
    type: mongoose.Schema.Types.ObjectId,
    ref: "Building",
    required: true,
  },
  classrooms: [
    {
      type: mongoose.Schema.Types.ObjectId,
      ref: "Classroom",
    },
  ],
});
```

Classroom

```
const classroomSchema = new mongoose.Schema({
  name: {
    type: String,
    required: true,
  },
  schedule: [
    [
      {
        type: mongoose.Schema.Types.ObjectId,
        ref: "Lesson",
      },
    ],
  ],
  capacity: {
    type: Number,
    required: true,
  },
  floor: {
    type: mongoose.Schema.Types.ObjectId,
    ref: "Floor",
    required: true,
  },
  options: [
    {
      type: String,
    },
  ],
});
```

Lesson

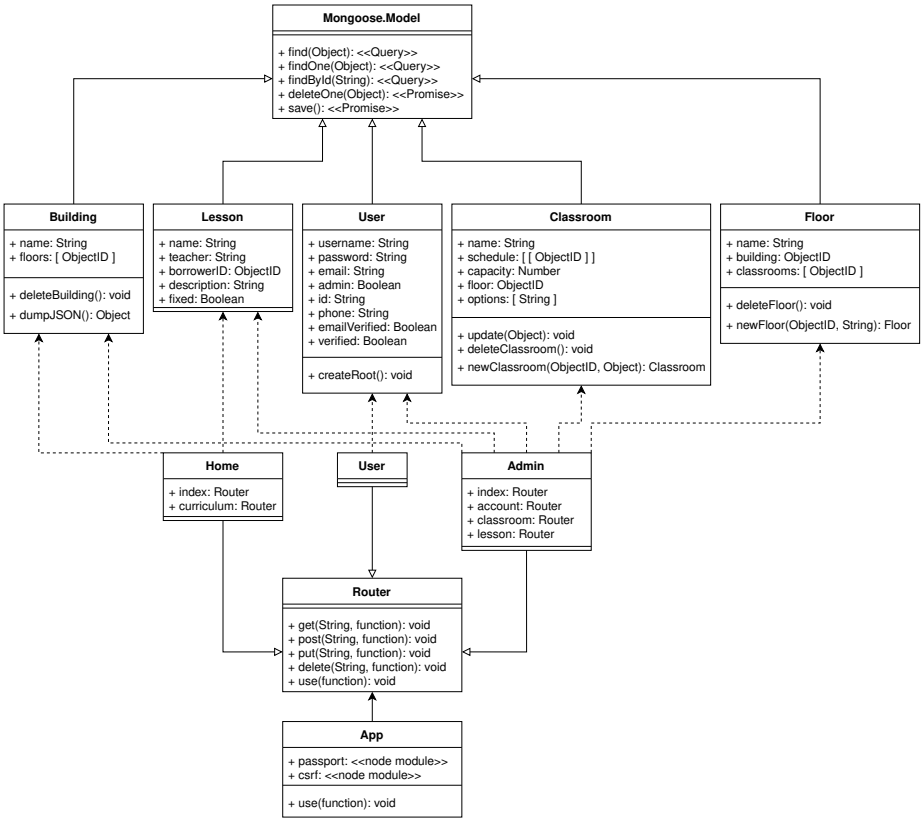
```
const lessonSchema = new mongoose.Schema({
  name: {
    type: String,
    required: true,
  },
  teacher: {
    type: String,
    required: true,
  },
  borrowerID: {
    type: mongoose.Schema.Types.ObjectId,
    ref: "User",
  },
  description: {
    type: String,
  },
  fixed: {
    type: Boolean,
    default: false,
    required: true,
  },
});
```

User

```
const userSchema = new mongoose.Schema({
  username: {
    type: String,
    required: true,
    unique: true,
    trim: true,
    minlength: 3,
  },
  password: { type: String, required: true, trim: true, minlength: 3 },
  email: {
    type: String,
    required: true,
    unique: true,
    trim: true,
    minlength: 3,
  },
  admin: { type: Boolean, required: true },
  id: { type: String, required: true, unique: true, trim: true, minlength: 3 },
  phone: {
    type: String,
    required: true,
    unique: true,
    trim: true,
    minlength: 3,
  },
  emailVerified: { type: Boolean, required: true },
  verified: { type: Boolean, required: true },
});
```

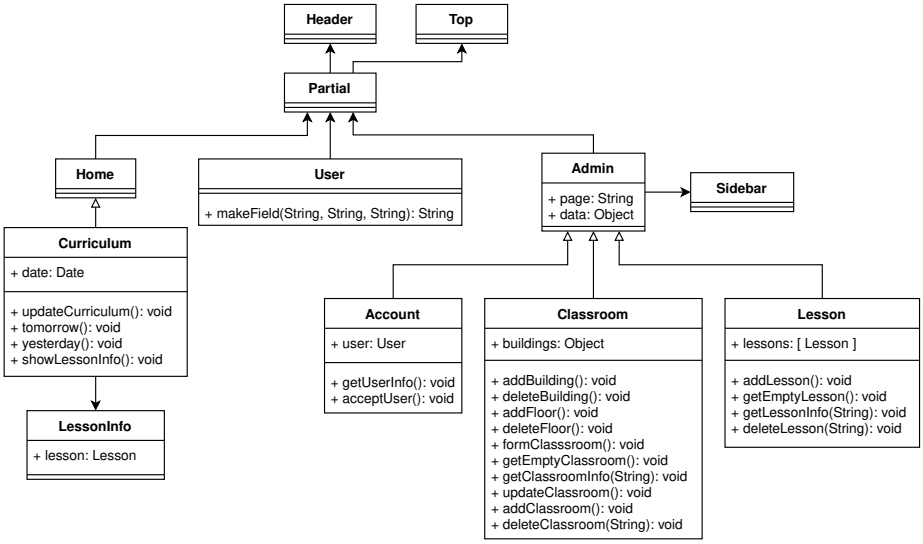
6 類別圖設計 (Class Diagram Design)

6.1 後端與資料庫 ODM



說明：程式主體為 App ，App 會使用 Express 建立伺服器，並使用 Mongoose 連接資料庫。App 透過各個 Router 來處理使用者的請求，並使用 Middleware(passport) 來管理使用者登入狀態。各個 Router 會依據需要使用相對應的 Model 來操作資料庫，並將結果傳回給使用者。

6.2 前端互動及資料傳輸



說明：每個頁面皆包含 Header 和 Topbar ，分別為 HTML 的雜項設定和功能列。其中多數函式皆在內部以純 JavaScript 實作（例如抓取元素資訊等），使用 Axios 來傳送請求至後端，並直接操作 DOM 來處理畫面的更新，達成前端的 jQuery Free 。

7 實作方案 (Implementation Language and Platform)

7.1 後端與資料庫

後端採用 Node.js 的 Express 框架，並使用 Mongoose 來操作 MongoDB 資料庫。使用者登入的部份由 Node Package 的 Passport 來建立 Session 。CSRF Protection 也是使用 Node Package 提供的 csurf 套件來實作。預計執行在 x86_64 架構的 Linux 作業系統上，並使用 Docker 來建立 MongoDB 資料庫。後續可能會使用 Docker Compose 來建立整個環境。

7.2 前端

前端採用 Bootstrap 5 框架配合基本的 JavaScript 和 animate.css ，並使用 Axios 來傳送請求至後端，以直接操作 DOM 的方式來處理畫面的更新。

8 設計議題 (Design Issue)

8.1 後端框架選擇

此專案為較常見的 CRUD 應用，因此選擇 Express 作為後端框架，並使用 Mongoose 來操作 MongoDB 資料庫。但有些實作上的細節需要注意：

- 安全性：例如 CSRF Protection 的部份，因為 Express 本身並沒有提供 CSRF Protection 的功能，因此需要使用 Node Package 的 csurf 套件來實作。但像 Ruby on Rails 就預設提供了這樣的功能，雖然 Express 學習成本較低，但新手反而會在無意間犯下安全性上的錯誤。
- 維護成本：例如在 Flask 中有的 Helper ，像 url_for 可以直接產生對應的 URL ，但 Express 中並沒有提供類似的功能，因此需要自己實作，並且每當 URL 改變時，都需要同步修改程式碼，這樣的維護成本較高。
- 程式碼品質：例如在 Ruby on Rails 中有 `rails generate resource` 這樣的命令可以快速產生 RESTful API 的 Controller ，但 Express 全部要自己寫，以新手而言也不知道如何設計 URL 是最佳的，因此需要花費較多的時間來設計，也可能會有設計上的缺陷。

綜觀上述問題可以發現，選擇專案的前提如果是「新手」、「快速」、「稍微有點規模」的專案，那麼 Express 不是一個很好的選擇。但畢竟發現的時候也寫到一半了，因此就繼續使用 Express 來實作。

8.2 資料庫選擇

觀察前面章節所述的內容可以發現，資料表中像是「大樓」、「樓層」、「教室」及「課程」等是環環相扣的，並且理論上來說資料量都不會到巨量，而當中的關係還有 Nested 跟 Array of 的，用 Mongoose 的 `populate` 相當繁瑣且容易出錯。因此本專案更適合的資料庫應為標準的 SQL ，但如前一個議題，因為已經寫到一半了，因此就繼續使用 MongoDB 來實作。

8.3 模板引擎選擇或使用前端框架

Express 本身並沒有提供 Template Engine ，因此需要自己選擇或是自己實作。但 Express 本身提供了 Middleware 的功能，因此可以使用 Frontend Framework 來實作，例如 Vue 或 React 等。考慮到專案的工作量跟我的肝指數，所以目前沒有使用前端框架，而是使用 EJS 直接在後端 render 完後配合 Bootstrap 實作簡單的互動介面傳回前端。