這是一個非常好的問題！如果你的 **商家管理系統** 是整個 **線上點餐平台** 的一部分，那麼整體架構設計就需要考慮到 **模組化、權限分離、資料一致性與可擴充性**。以下是我建議的設計方式：

**🧱 整體系統架構設計（模組化）**

📦 Online Ordering Platform

├── 🌐 Public Frontend（客戶端）

│ ├── Web App（點餐、瀏覽菜單、下單）

│ └── Mobile App（可選）

│

├── 🧾 POS / Kiosk 前端

│ ├── POS 系統（店員操作）

│ └── 自助點餐機（Kiosk）

│

├── 🛠️ Merchant Admin（商家管理系統）

│ └── Web Components SPA（你目前正在開發的部分）

│

├── 🔐 Auth Service（登入、權限、JWT）

│

├── 📦 Backend API（RESTful 或 GraphQL）

│ ├── 商家管理 API

│ ├── 商品與分類 API

│ ├── 訂單與付款 API

│ ├── 設備與通知 API

│ └── 使用者與權限 API

│

├── 🗃️ Database（你已設計完成的 schema）

│

└── 📡 Realtime Service（WebSocket / SSE）

└── 訂單即時更新、設備狀態、列印通知等

**🔐 權限與角色分離建議**

| **角色** | **可存取模組** | **說明** |
| --- | --- | --- |
| admin | 商家管理系統全部模組 | 商家老闆或主管 |
| pos | POS 系統、部分商品與訂單管理 | 店員 |
| kiosk | 自助點餐機 | 顧客操作 |
| customer | 公開前台（點餐、查詢訂單） | 顧客 |
| guest | 公開前台（瀏覽菜單） | 未登入 |

**🔄 API 設計建議**

* 使用 RESTful API 或 GraphQL，並統一使用 JWT 驗證
* 每個 API endpoint 應該根據 merchant\_id 做資料隔離
* 商家管理系統的 API 可設計為 /api/merchant/\*
* 訂單與付款模組應支援 webhook（例如 Stripe、Line Pay）

**🔔 即時功能建議**

* 使用 WebSocket 或 Server-Sent Events（SSE）來：
  + 即時接收新訂單
  + 顯示設備連線狀態
  + 自動列印通知

**🧩 商家管理系統與主系統整合方式**

1. **登入後取得 JWT Token**
   * 商家登入後，取得 token 並儲存在 localStorage
   * 所有 API 請求都帶上 Authorization: Bearer <token>
2. **API Gateway 或 Proxy**
   * 可使用 Nginx 或 API Gateway 將 /admin 導向商家 SPA
   * 其他路徑導向前台或 POS 系統
3. **模組共用**
   * 可將共用元件（如商品卡片、訂單詳情）抽成獨立 Web Components 或 NPM 套件

如果你想，我可以幫你畫出整體架構圖，或是幫你設計 API 文件、JWT 權限驗證流程、甚至是 WebSocket 訂單通知模組。你想先深入哪一塊呢？

當你的線上點餐平台是一個完整的系統時，除了商家管理系統外，還會包含前台點餐、POS、自助機、金流、通知等模組。以下是我為你細緻化的整體模組設計：

**🧱 線上點餐平台模組總覽**

📦 線上點餐平台

├── 🌐 前台點餐系統（Web / Mobile）

├── 🧾 POS 系統（店員用）

├── 🖥️ 自助點餐機（Kiosk）

├── 🛠️ 商家管理系統（Merchant Admin）

├── 🔐 使用者與權限系統（Auth）

├── 📡 即時通訊模組（WebSocket / SSE）

├── 💳 金流模組（Payment Gateway）

├── 🖨️ 自動列印模組（Printer Service）

└── 📦 後端 API 與資料庫（Backend + DB）

**🔍 各模組細節設計**

**1. 🌐 前台點餐系統（Web / Mobile）**

**使用者**：顧客  
**功能**：

* 商家選擇與搜尋
* 菜單瀏覽（分類、標籤、圖片）
* 商品選項與套餐選擇
* 加入購物車、下單
* 支付（Line Pay、信用卡、Apple Pay 等）
* 訂單追蹤與通知
* 多語系支援（中、英）

**2. 🧾 POS 系統（店員用）**

**使用者**：店員（POS 角色）  
**功能**：

* 快速點餐與結帳
* 掃碼取單、叫號
* 訂單狀態更新（製作中、完成）
* 支援現金與行動支付
* 設備連線狀態顯示
* 自動列印整合

**3. 🖥️ 自助點餐機（Kiosk）**

**使用者**：顧客  
**功能**：

* 全螢幕觸控操作
* 商品選擇與自訂
* 多語系與無障礙設計
* 支付整合（掃碼、信用卡）
* 自動列印取餐單
* 設備狀態監控與遠端更新

**4. 🛠️ 商家管理系統（Merchant Admin）**

**使用者**：商家管理員（admin）  
**功能**：

* 儀表板（營運摘要）
* 商品與分類管理
* 套餐與選項設定
* 訂單與金流查詢
* 設備管理（POS、Kiosk、Printer）
* 商家設定（營業時間、自動接單等）
* 員工帳號與權限管理

**5. 🔐 使用者與權限系統（Auth）**

**功能**：

* 登入 / 註冊 / 忘記密碼
* JWT 驗證與刷新
* 角色與權限控管（admin, pos, kiosk, customer, guest）
* OAuth 第三方登入（可選）

**6. 📡 即時通訊模組（WebSocket / SSE）**

**功能**：

* 即時訂單通知（POS、Admin）
* 設備狀態更新（Printer、Kiosk）
* 訂單狀態同步（前台、後台、POS）

**7. 💳 金流模組（Payment Gateway）**

**功能**：

* 整合第三方支付（Line Pay、TapPay、Stripe、Apple Pay）
* 支付狀態回傳與 webhook
* 退款處理
* 金流對帳與報表

**8. 🖨️ 自動列印模組（Printer Service）**

**功能**：

* 接收訂單列印請求
* 支援多種印表機（USB、網路、藍牙）
* 列印格式設定（小票、廚房單）
* 列印失敗重試與通知

**9. 📦 後端 API 與資料庫**

**功能**：

* RESTful API 或 GraphQL
* 資料隔離（依 merchant\_id）
* 資料一致性與交易處理
* 日誌與錯誤追蹤
* 可擴充的模組化架構

如果你需要，我可以幫你：

* 畫出整體系統架構圖（前後端 + 模組）
* 設計 API 文件（Swagger 格式）
* 建立模組間的資料流與事件流程圖
* 實作其中一個模組的前端或後端原型

你想先深入哪一個模組或要我幫你畫出整體架構圖？