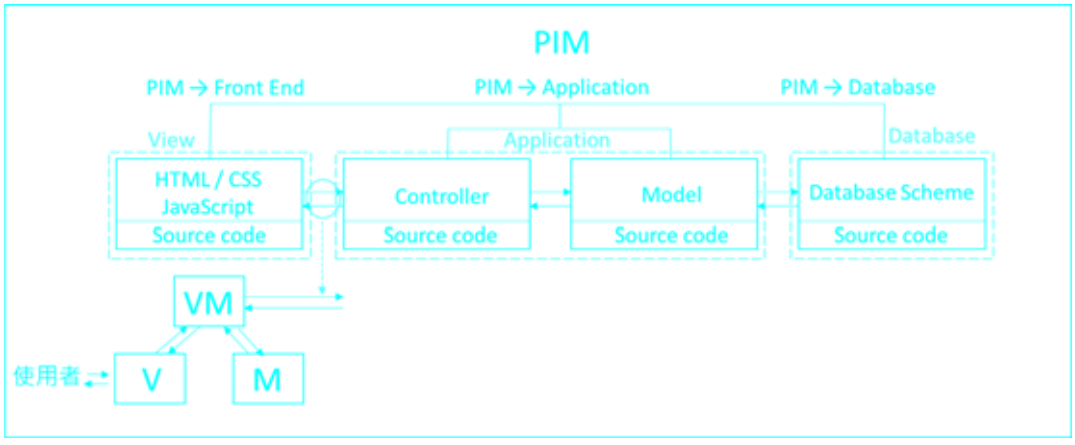
11.2.3.2 應用 GPT-4 轉換之案例實作

以下以客戶登入使用個案為例，以GPT-4 執行 PIM 轉 Web 使用者介面與程式碼。其中，前端網頁使用 MVVM (Model-View-ViewModel)架構（如圖 11-9 所示）之 React 框架與 HTML、CSS 及 Java Script 程式碼。此外，Model（模型）：通常在 React 中通常用{useState}來管理儲存應用程序的數據；View（視圖）：是指網頁應用程式的使用者介面，通常是由 HTML、 CSS 和 JavaScript 組成，在 React 框架下再加上 React 組件來實現，負責將資料呈現給使用者，並且可接收使用者的輸入。ViewModel（視圖模型）：在 React 中，可以使用 Hooks 來管理與 UI 相關的狀態和邏輯。

Figure
圖 11-9 PIM 轉 MVVM 架構與 Code



依上述方法介紹實作步驟。實作步驟包括：步驟一，登入 GPT-4。步驟二，依據上述規劃及認知，輸入明確的特徵與指令，例如輸入必要之 PIM 圖表檔（介面藍圖、介面詞彙、循序圖）、指令（必要之 PIM 圖表名、功能、格式／程式語言，例如 React 框架與 HTML/CSS 程式碼等），並要求 GPT-4 產出相應的 Web 使用者介面與程式碼等。步驟三，針對 GPT-4 產出之成果進行編修（步驟二、步驟三反覆進行，直到成果滿意）。步驟四，複製成果（程式碼），至程式設計編輯器進行介面設計成果預覽等，此處使用 CodeSandbox 程式設計平台為例；同時，亦可於程式設計平台進行內容的調整。

PIM 轉 Web 使用者介面之系統畫面及其程式碼，主要實作步驟結合使用者登入案例介紹如下。

步驟一，登入 GPT-4。若要登入須先註冊帳號才能使用，註冊方式只需輸入

電子郵件信箱以及使用手機號碼來驗證即可，亦或是使用 Google、Microsoft、Apple 帳號也可進行連動、快速登入（如圖 11-10 所示）。本案例使用 GPT-4，在介面上與免費的 3.5 版本相比，增加了上傳檔案的功能（如圖 11-11 所示）。

Figure 圖 11-10 GPT-4 登入畫面

Figure 圖 11-11 指令輸入 (Prompt)介面

步驟二，依據上述規劃及認知，輸入明確指令，持續進行互動，包含：(1)角色：程式設計人員；(2)執行什麼任務：系統開發；(3)想像目標產出：使用者介面、框架、程式語言；(4) 抽象化特徵：必要之 PIM 圖表檔（例如以圖 11-8 下方之使用者登入介面藍圖、介面詞彙、循序圖為例）與依圖 11-8 下方之 PIM 圖表檔內容，撰寫其功能或格式／程式語言等完整指令（如表 11-3 所示）(5) 輸入特徵（登入頁面 UI 為較簡易的介面可一次輸入特徵，若遇到較為複雜的頁面則分批次執行較佳）。

步驟三，針對 GPT-4 產出成果進行編修，步驟二、步驟三可反覆進行，直到成果滿意。以下產出符合 React 框架的 HTML 程式碼、CSS 程式碼、JavaScript 程式碼。表 11-4 之第 5 行到第 23 行程式碼，為輸入循序圖及指令所產出之登入驗證 JavaScript 部分樣板程式碼，使用者（程式人員）後續須將必要之程式碼補齊，6 至 9 行同時也相當於 Hooks 來管理與 UI 相關的狀態和邏輯作為 MVVM 中的 ViewModel。6、7 行，使用 useState 來管理儲存帳號、密碼之數據，作為 MVVM 中的 Model。表 11-4 第 25 至 51 行程式碼為輸入介面藍圖、介面詞彙及指令所產出之 HTML 程式碼，讓網頁具有結構性，例如標題、段落、列表等，並且能夠嵌入圖片等媒體。表 11-5 為輸入介面藍圖產出之 CSS 樣式程式碼，CSS 影響網頁元素中的布局、顏色、字體多方面的視覺效果（如表 11-4、表 11-5 所示）。上述 Login.js 檔、Login.css 檔中的 HTML、CSS 及 JavaScript 構成 MVVM 中的 View。

Table 表 11-3 完整指令撰寫參考

指令（必要之 PIM 圖表名、功能、格式 / 程式語言等）
請依照介面藍圖生成登入頁面 UI，使用 React 前端網頁框架使用 React 前端網頁框架（格式）之HTML與CSS（格式／程式語言），產出程式碼，例如須有登入系統名稱「便當王網路訂購系統」。依照介面詞彙命名元件名稱，例如便當王網路訂購系統、帳號、密碼、登入，元件類型，例如 Label，並與資料庫資料表欄位對應產出程式碼，例如 name = “Customer_ID”、name = “Customer_Pswd”，依照循序圖上的訊息與操作，例如登入驗證，產出JavaScript程式碼。此外，請生成上述所需的 index.js 作為入口檔程式。

Table 表 11-4 Chat GPT 產出之 Login.js 主檔程式碼

1.	/* Login.js */
2.	import React, { useState } from 'react';
3.	import './Login.css';
4.	
5.	function Login() {//相當於 Hooks 來管理與 UI 相關的狀態和邏輯
6.	const [customerID, setCustomerID] = useState('');
7.	const [customerPswd, setCustomerPswd] = useState('');
8.	//使用 useState 來管理儲存帳號、密碼之數據
9.	const handleLogin = async (e) => {
10.	e.preventDefault();
11.	
12.	// Here you would usually make a request to your server
13.	// For example:
14.	// const response = await fetch('/api/login', { method: 'POST', body: JSON.stringify({ customerID, customerPswd }), headers: { 'Content-Type': 'application/json' } });
15.	// const data = await response.json();
16.	
17.	// If login is successful:
18.	// if (data.success) {
19.	// // Redirect to the dashboard or do something else
20.	// }
21.	
22.	console.log('Login attempt with:', customerID, customerPswd);
23.	};
24.	
25.	return (

Table 表 11-4 Chat GPT 產出之 Login.js 主檔程式碼（續）

```
26.     <div className="login-container">
27.       <h1>便當王網路訂購系統</h1>
28.       <form onSubmit={handleLogin}>
29.         <label htmlFor="customerID">帳號:</label>
30.         <input
31.           type="text"
32.           id="customerID"
33.           name="Customer_ID"
34.           value={customerID}
35.           onChange={(e) => setCustomerID(e.target.value)}
36.         />
37.
38.         <label htmlFor="customerPswd">密碼:</label>
39.         <input
40.           type="password"
41.           id="customerPswd"
42.           name="Customer_Pswd"
43.           value={customerPswd}
44.           onChange={(e) => setCustomerPswd(e.target.value)}
45.         />
46.         <button type="submit">登入</button>
47.       </form>
48.     </div>
49.   );
50. }
51. export default Login;
```

Table 表 11-5 Chat GPT 產出之 Login.css 檔程式碼

```
1.  /* Login.css */
2.  .login-container {
3.    width: 100%;
4.    max-width: 400px;
5.    margin: auto;
6.    padding: 20px;
7.    text-align: center;
8.  }
```

Table
表 11-5 Chat GPT 產出之 Login.css 檔程式碼（續）

9.	
10.	h1 {
11.	margin-bottom: 20px;
12.	}
13.	
14.	label {
15.	display: block;
16.	margin-top: 10px;
17.	}
18.	
19.	input[type="text"],
20.	input[type="password"] {
21.	width: 100%;
22.	padding: 10px;
23.	margin-top: 5px;
24.	}
25.	
26.	button {
27.	margin-top: 20px;
28.	padding: 10px 20px;
29.	cursor: pointer;
30.	}

此外，GPT-4 同時協助生成 React 框架中作為入口的 index.js 檔程式碼（如表 11-6 所示）。

步驟四，複製成果（程式碼），至程式設計編輯器進行介面設計成果預覽等（如圖 11-12 所示）。同時，亦可於程式設計平台進行內容的調整。此處使用 CodeSandbox 程式設計平台為例，點選+Sandbox 創立 React templates 專案，使用者（程式人員）加入 Login.js 檔、Login.css 檔及 index.js 檔，進行預覽（如表 11-6 所示）。

基於上述的實作案例顯示，應用 GPT-4 將 PIM 轉 Web 使用者介面程式碼已能產出使用者靜態頁面與部分互動的程式碼，例如 Login.js 檔、Login.css 檔及 index.js 檔。然而，GPT-4 根據上述所整理之輸入，Login.js 主檔中的 Function 僅能產出部分參考程式碼或框架。因此，使用 GPT-4 將 PIM 轉 Web 使用者介面程式碼之任務，大約可轉出 80%的程式碼。

Table
表 11-6 index.js 程式碼

1.	import React from "react";
2.	import ReactDOM from "react-dom";
3.	import Login from "./Login";
4.	
5.	ReactDOM.render(
6.	<React.StrictMode>
7.	<Login />
8.	</React.StrictMode>,
9.	document.getElementById("root")
10.	);

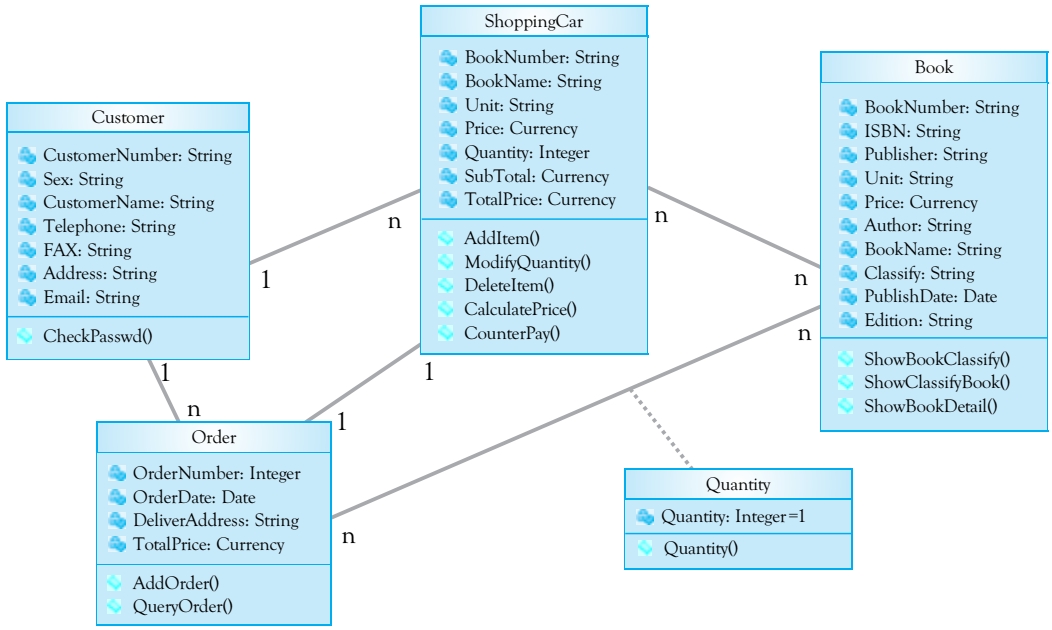
Figure
圖 11-12 於程式設計平台進行內容的預覽

11.3 MDA 轉換案例——PIM 轉 PSM 轉 Code

本節以一個網路購書之系統為例，說明上述 PIM 轉 PSM 方法論之應用，並說明如何將 PSM 轉換至程式碼。當然，PIM 至 PSM 與程式碼之轉換是可以被自動化的，也就是可以藉由 CASE Tool 直接來完成（請參考本章附錄之案例），但瞭解自動化背後的轉換原理或方法論，有助於瞭解 PIM 之建構。此網路購書系統

最終之類別圖可表達成實體類別圖以及介面與控制類別圖；實體類別圖共有 5 個實體類別（含一個關聯類別），類別間之關係均是關聯關係，如圖 11-13 (Wu et al., 2005; 2007)；介面與控制類別圖共有 5 個控制類別與 11 個介面類別，類別間均是相依關係，如圖 11-14。

Figure
圖 11-13 網路購書系統之實體類別圖



假設該網路購書系統由多種技術開發：介面是 Web-based 模式、應用程式是 EJB(Enterprise Java Bean) 模式、資料庫是 Oracle 9i 的關聯式(Object-Relational) 資料庫模式，則轉換工作將可分為 PIM 轉 Oracle 9i PSM、PIM 轉 EJB PSM ，以及 PIM 轉 Web PSM 轉系統畫面及其程式碼。

11.3.1 PIM 轉 Oracle 9i PSM 與 DDL 語法

當 PIM 階段之類別圖產出後，接下來須按照關聯式資料庫（例如 Oracle 9i）平台的特性，應用 CASE Tool 將此類別圖轉成符合該平台資料型態的資料表，接著進行資料庫設計。本案例使用的 CASE Tool 是 Relational Rose（以下簡稱為 Rose），其 PIM 轉關聯式 PSM 包括資料型態對應、實體類別轉換與實體類別的關係轉換，其操作順序是先刪除類別圖中之暫存類別，將調整後之類別圖轉成資料模型(Data Model Diagram, DMD)，接下來再將 DMD 做必要的調整以產生 DDL 語法，並建立關聯表，介紹如下。