國立中央大學資訊管理學系

111（一）系 統 分 析 與 設 計

系統軟體分析規格書

第二組

資管三 B 109403537 林俊霆

資管三 B 109403540 許珀維

資管三 B 109403055 王柏勛

資管三 B 109403534 賴思妤

資管三 A 109401027 陳佳妤

資管三 A 109403009 蘇湘婷

指導教授：葉羅堯 教授

中 華 民 國 111 年 11 月 30 日

目錄

表目錄

圖目錄

第 1 章 簡介

軟體分析規格書（software analysis description，SAD）係依據軟體產品、專案之主要使用者之需求規格文件（software requirements specification，SRS），主要用於描述邏輯之軟體架構與系統範圍之文件。藉由本文件得以分析軟體系架構之目的，並作為軟體設計階段之依據。

本專案之文件採用統一塑模語言（Unified Modeling Language，UML）說明與建構本系統之方法與架構，包含：使用案例圖（Use Case Diagram）、泳道圖（Swim-lane）與循序圖（Sequence Diagram）等。

1.1 文件目的

本文件之目的用於提供軟體系統開發人員分析之規範與藍圖，透過軟體分析規格書，開發人員可以明確了解軟體系統之邏輯與運作方式，並得以此為據遵照共同訂定之規格設計軟體系統。

本文件針對系統之分析為邏輯階段（logical phase）而非是實際設計階段（physical phase）之內容，分析模型與系統設計與實作環境無關之邏輯結構（logical Structure），得以使用邊界、控制和實體物件呈現系統資訊（information）、行為（behavior）和展示（presentation）三個層面。

1.2 目標對象

本系統範圍用於電子商務（學校教科書書城），其中主要包含會員、圖書資訊、訂購圖書、圖書結帳、訂單管理、圖書管理、會員管理與類別管理八個模組，並且能進行相關新增、查閱與維護工作。

1.3 參考文件

1. 系統分析與設計—需求（Requirement）

1.4 文件架構

本文件共分為五個章節，用以闡述本專案之分析相關內容：

1. 第 1 章 針對本文件進行簡介，說明本文件重要之處。
2. 第 2 章 依據本專案前份文件之使用者案例依序進行分析，於本章節依照使用者案例將產出所需活動圖與強韌圖。
3. 第 3 章 分析本專案所需之資料庫架構與資料表內容。

4. 第 4 章 則是陳列出本專案所需之類別、屬性與方法的類別圖。
5. 第 5 章 說明本專案所需之系統開發環境，其中包含系統架構圖、MVC 架構之說明、JAVA 三層架構說明。

第 2 章 系統動態分析圖

在本章節中，將透過在前一份文件中所分析之使用者案例（use case）逐一進行詳細之系統動態分析。首先須先將使用者案例之主要流程轉換成活動圖，再者依照所分析之活動圖產生強韌圖以找出分析之類別。

2.1 使用案例圖

依據第一份文件—系統軟體需求規格書（Software Requirement Specification），本教科書商城系統預計共有 3 位動作者與 40 個使用案例，並依照不同之模組區分成不同子系統共計七個子系統，其中包含以下：1. 會員子系統、2. 圖書資訊子系統、3. 訂購圖書子系統、4. 結帳圖書子系統、5. 訂單管理子系統、6. 圖書管理子系統、7. 類別管理。

下圖（圖 1）為本系統之使用案例圖：

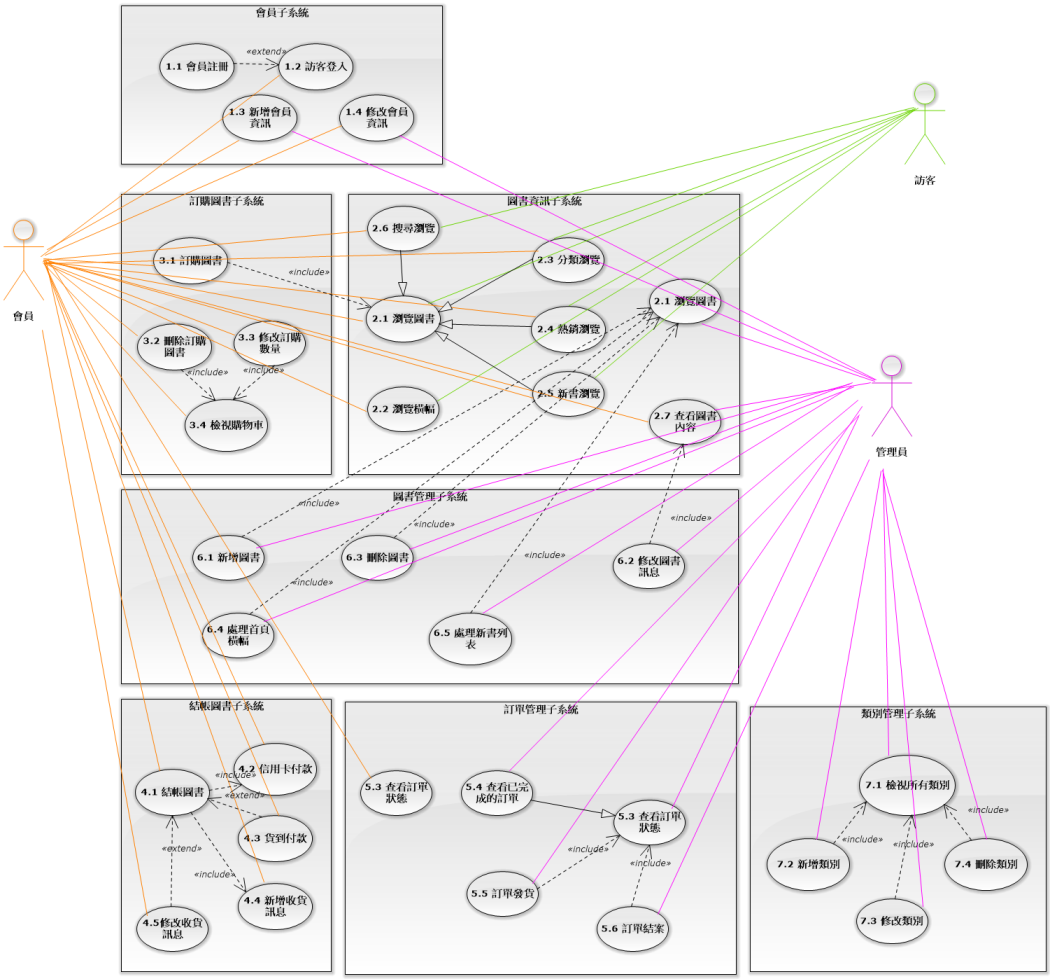


圖 1: 使用案例圖

根據上圖所分析之使用案例規格，需先逐一將每個使用案例轉換成活動圖與強韌圖，並逐一詳細闡述，並且進行分析。再者依據就前揭所述之活動圖、強韌圖產生所需之通訊圖與狀態機圖，進行統整並繪製出分析階段之循序圖，依照本系統之架構劃分而繪製而成。

2.2 使用案例 1.0：會員模組

本使用案例包含「1.1 會員註冊」至「1.4 會員更改資訊」四個使用案例，主要描述一般訪客註冊會員、登入、修改會員資料。對於管理員來說則可以修改所有會員的資料。其餘管理員對於會員的管理與操作放在 7.0 會員管理模組。以下並根據此些使用案例進行必要圖形之分析。

模組	說明	功能名稱	簡單說明
1.0 模組：會員	提供訪客註冊和登入功能，且會員可修改收貨訊息、安全訊息。 管理者可對所有會員進行操作，部分功能放在會員管理模組。	1.1 會員註冊	訪客可註冊帳號
		1.2 訪客登入	訪客可以登入系統
		1.3 新增會員資訊	會員可以新增自己的收貨訊息 管理員可以填入所有人的收貨訊息
		1.4 修改會員資訊	會員可以修改自己的收貨訊息、安全訊息 管理員可以修改所有指定會員收貨訊息、安全訊息

表格 1: 流程編號 1.0 會員模組

2.2.1 使用案例 1.2：會員登入

2.2.1.1 活動圖

將會員登入主要流程轉換為活動圖如下：

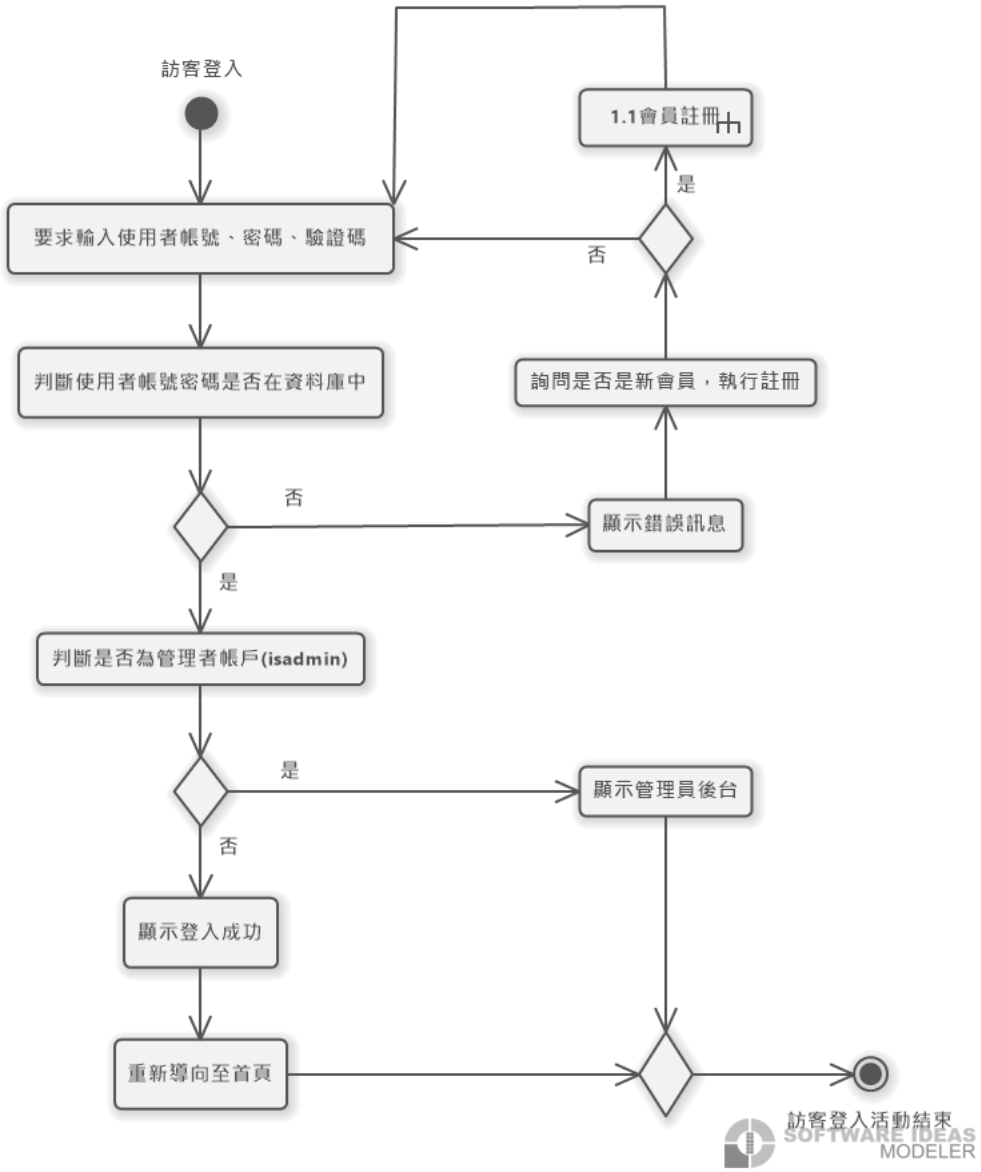


圖 2: 使用案例 1.2 會員登入活動圖

2.2.1.2 強韌圖

依據該使用案例之活動圖，可建立強韌圖以找出分析之類別，如下圖所示：

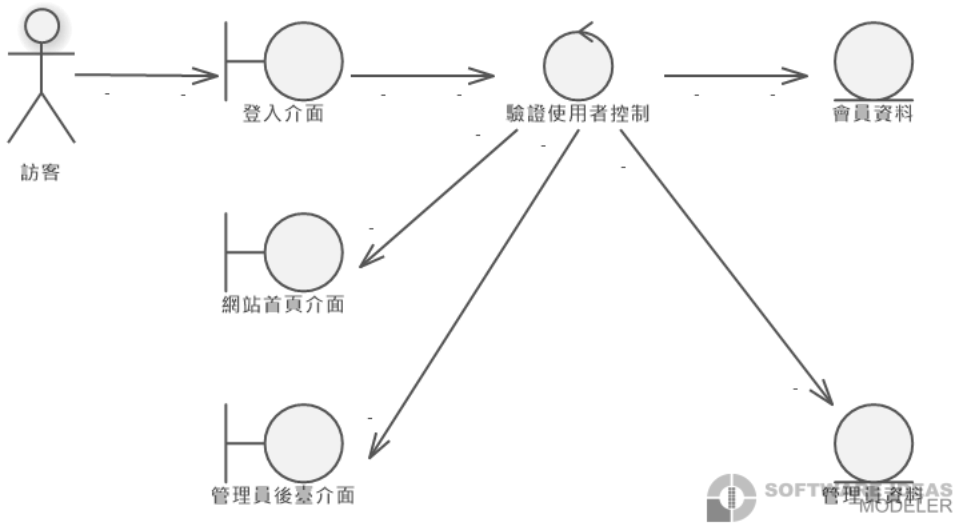


圖 3: 使用案例 1.2 會員登入強韌圖

2.3 使用案例 4.0 結帳圖書

結帳圖書包含「4.1 結帳圖書」至「4.5 修改收貨資訊」五個使用案例，主要用於描述使用者在確認購物車的內容後，執行結帳書本，結帳時必須確認收貨資訊有無錯誤，並可選擇付款方式。

模組	說明	功能名稱	簡單說明
4.0 模組：結帳圖書	會員可以結帳購物車內的圖書產生新訂單，並可以選擇信用卡付款或貨到付款	4.1 結帳圖書	會員選擇結帳圖書送出訂單
		4.2 信用卡付款	會員可使用信用卡付款
		4.3 貨到付款	會員可使用貨到付款
		4.4 新增收貨訊息	假如沒有收貨訊息，會員必須新增收貨訊息，才能產生訂單
		4.5 修改收貨訊息	會員可以修改收貨訊息，再產生訂單

表格 2: 流程編號 4.0 圖書結帳

2.3.1 使用案例 4.0 結帳圖書

2.3.1.1 活動圖

將結帳圖書主要流程轉換為活動圖如下：

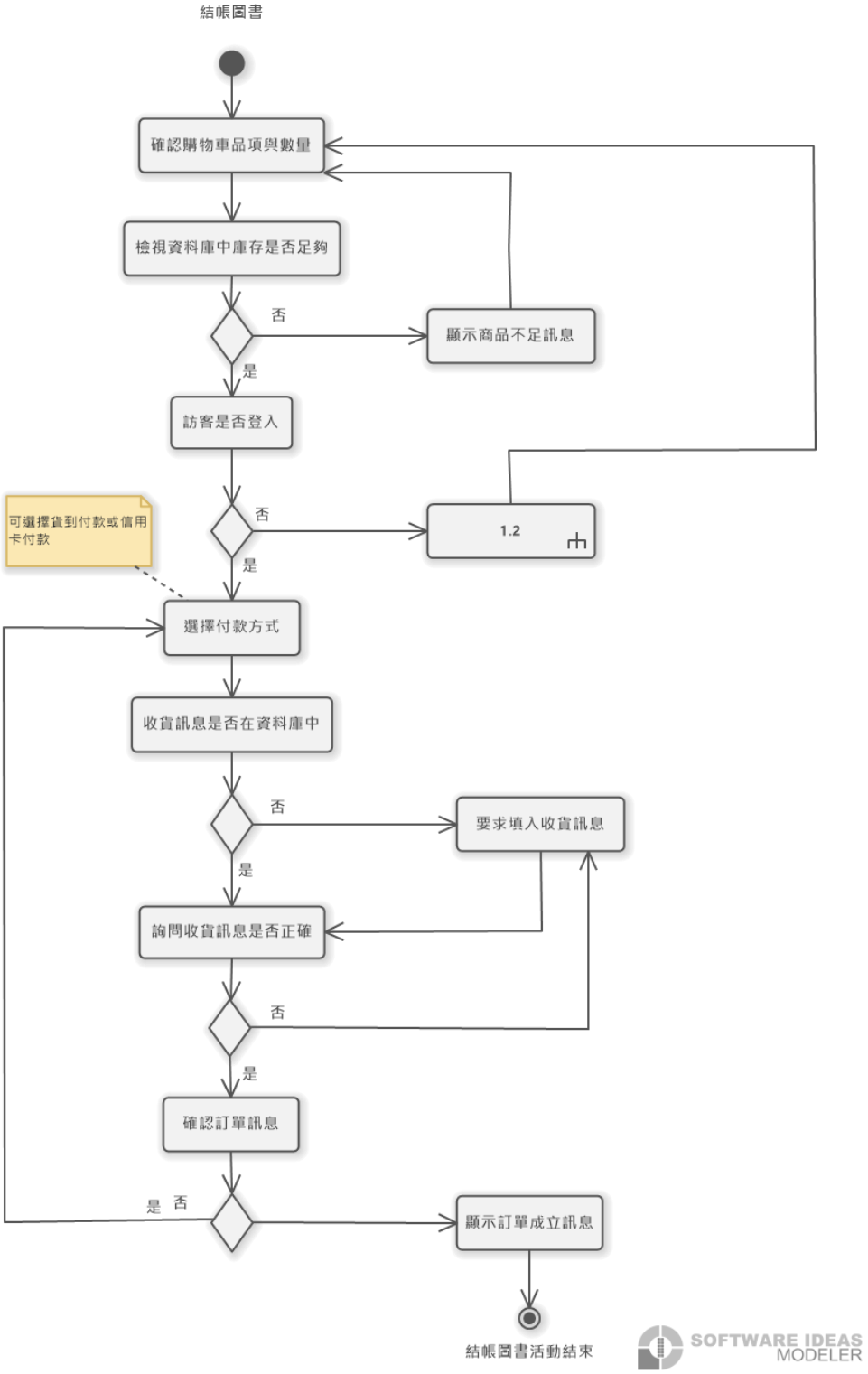


圖 4: 使用案例 4.0 圖書結帳活動圖

2.3.1.2 強韌圖

依據該使用案例之活動圖，可建立強韌圖以找出分析之類別，如下圖所示：

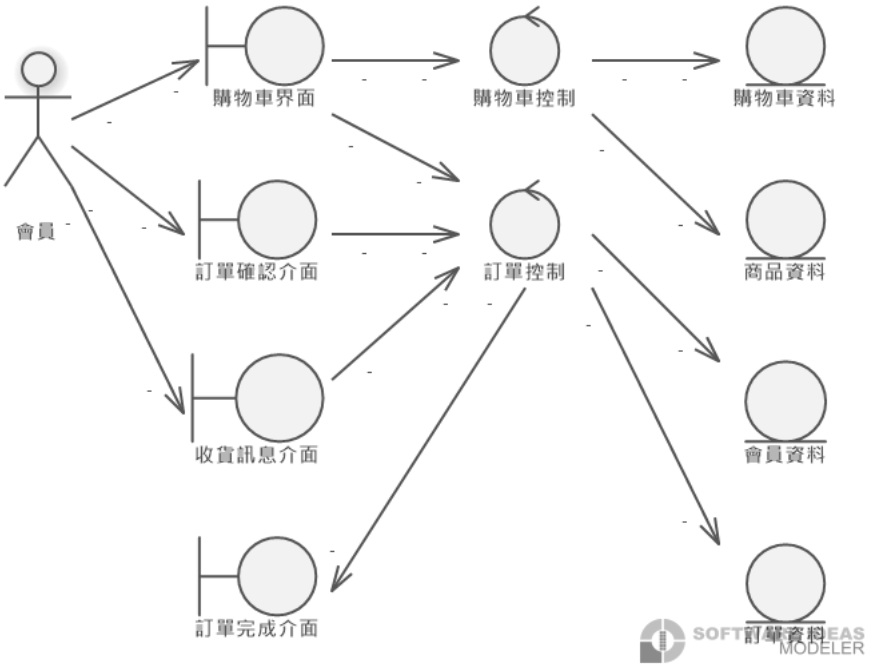


圖 5: 使用案例 4.0 圖書結帳強韌圖

第 3 章 資料庫設計

本專案之教科書商城系統提供使用者簡單與便利之線上購物、訂單管理與會員資料維護之服務，對於管理者來說，亦能以簡便方式進行會員管理、商品維護等後台作業，希冀不僅能提供最新與詳盡之商品說明，更能提供良好之購物體驗。

以下分析階段之資料庫設計採用實體關係圖（Entity-Relation Diagram）表示，並根據管理者與使用者之需求進行歸納與整理初步之系統條件。

以下詳述系統之資料庫需求，並將其整理成下圖之實體關係圖，共計包含6個實體 (Entity)、5 個關係(Relationship)：

1. 一般訪客可以註冊成為會員且必須以電子郵件或用戶名作為登入之帳號使用， 同時系統會自動給予每位會員編號。

2. 一般訪客與會員皆可將商品加入購物車，但伺服器不儲存該資料，而存於使用者之本地端。

3. 會員可以將購物車中內的商品進行結帳，如果尚未填寫地址、電話等個人資訊，須完成個人資訊，並確認過訂單內容後，才可完成訂單。

4. 管理者可以管理商品之異動與會員資料，並且進行維護作業。

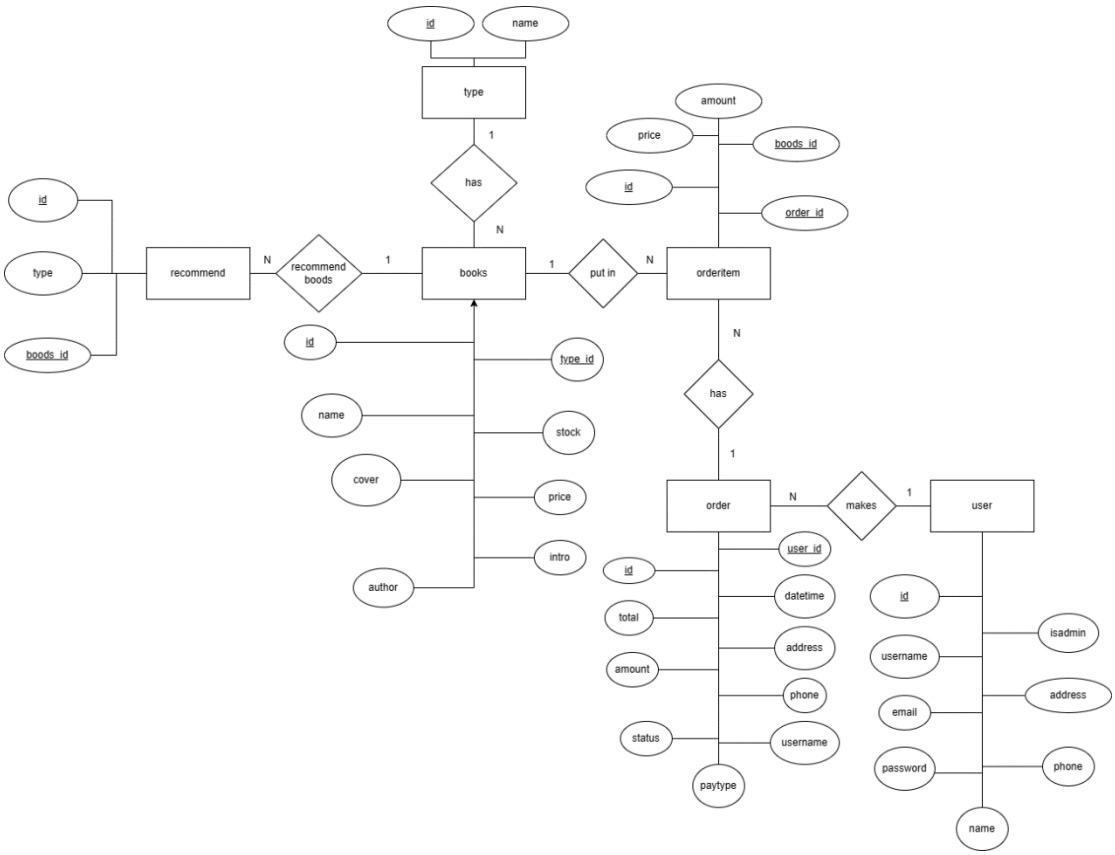


圖 6：實體關係圖

根據實體關係圖分析本專案所需之資料庫架構，以下將針對每張資料表進行描述，由於本範例僅實作後台管理者會員模組，因此資料表僅就會員與商品進行呈現，實際上仍需將所有資料表之分析呈現於此：

1. 會員資料表(user)

名稱	類型
id	int
username	varchar
email	varchar
password	varchar
name	varchar
phone	varchar
address	varchar
isadmin	bit

2. 商品資料表(books)

名稱	類型
id	int
name	varchar
cover	varchar
author	varchar
price	float
intro	varchar
stock	int
type_id	int

3. 訂單資料表(order)

名稱	類型
id	int
total	float
amount	int
status	tinyint
paytype	tinyint
username	varchar
phone	varchar
address	Varchar
datetime	datetime
user_id	int

4. 訂購品項資料表(orderitem)

名稱	類型
id	int
price	float
amount	int
goods_id	int
order_id	int

5. 類別資料表(type)

名稱	類型
id	int
name	varchar

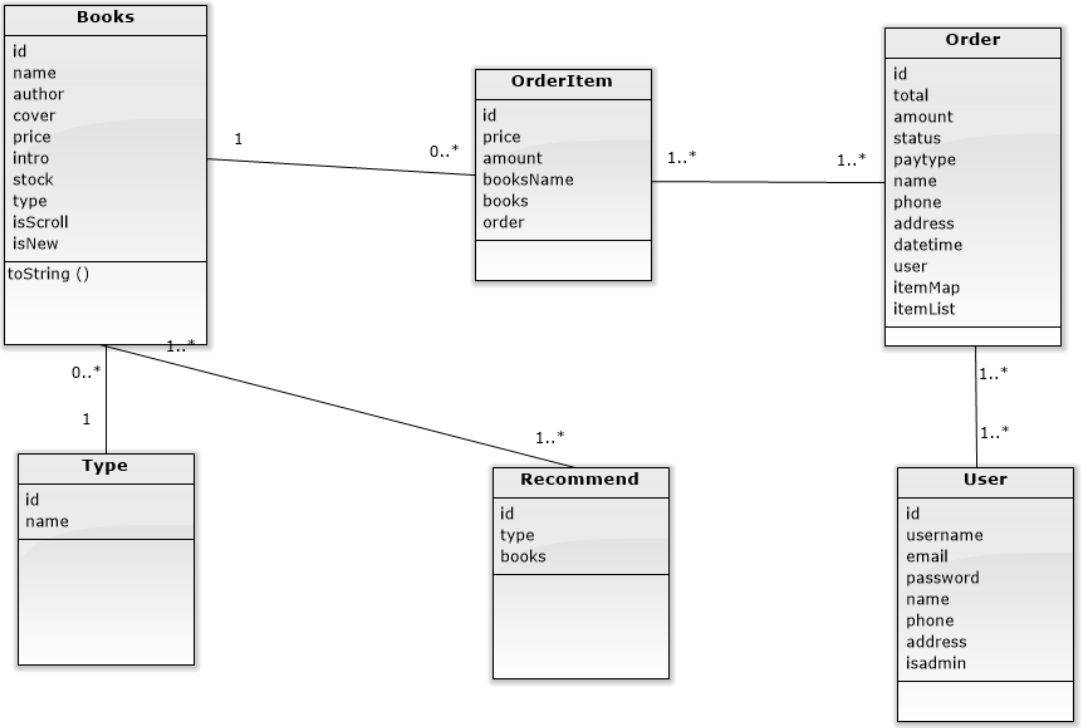
6. 推薦資料表(recommend)

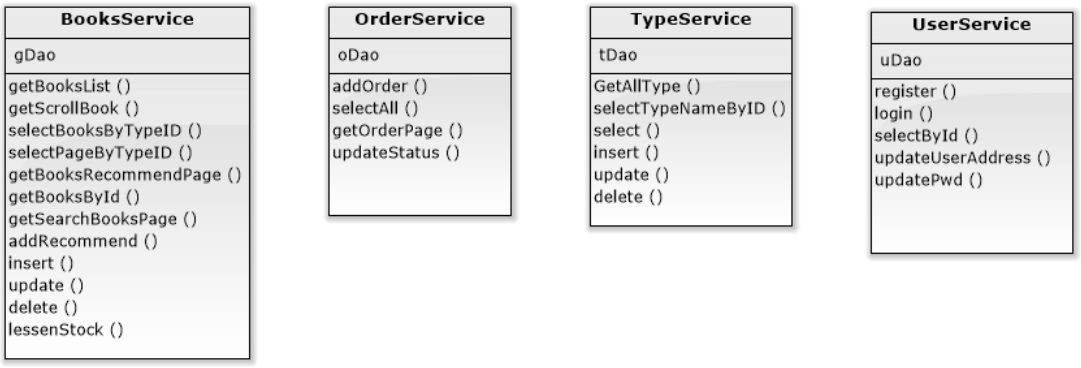
名稱	類型
id	int
type	tinyint
goods_id	int

第 4 章 類別圖

分析階段之類別圖（class diagram）依據第一份文件所述之使用案例找出並分析類別，同時也參照前章節（第 3 章 資料庫設計）以建立本專案之書城系統分析模型之類別圖。

該階段之類別圖僅列出控制（controller）和實體之類別，其內部之詳細屬性與方法僅大略進行定義，詳細之設計細節與使用之參數屬性與方法於第三份文件—設計（design）詳細描述。





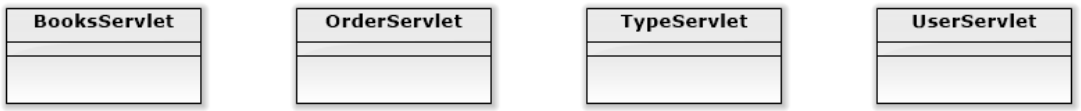


圖 7：分析階段之類別圖

第 5 章 系統開發環境

5.1 系統架構圖

本專案之整體架構如下圖所示，主要採用 Java 語言所撰寫之書城網站之應用程式，並預期採用 Java 平台技術之 Servlet 框架建構 Web 應用程式：

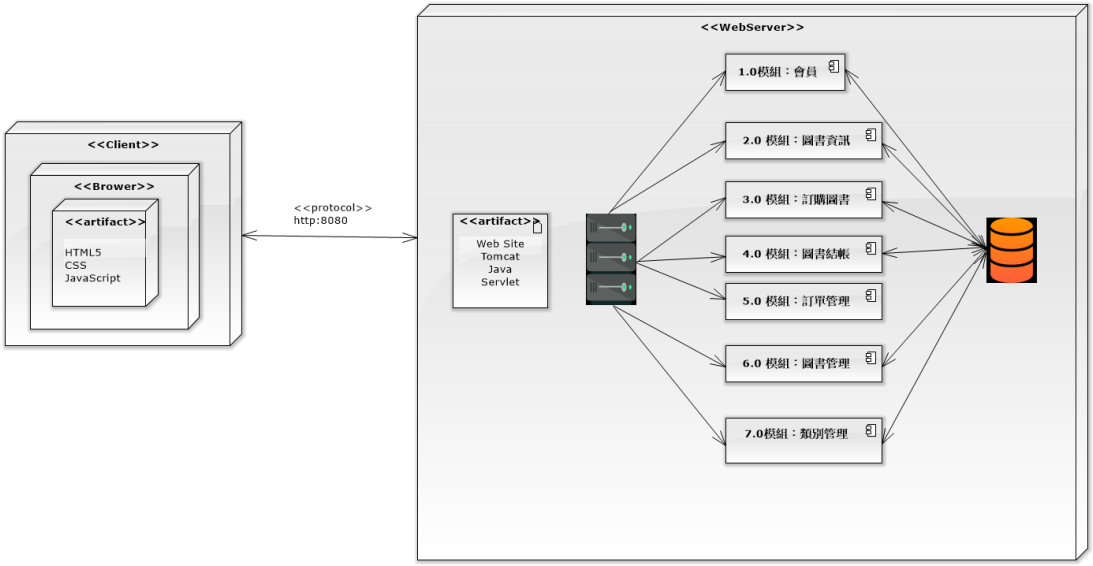


圖 8：分析階段之系統架構圖

1. 由於本專案之撰寫語言為 Java 因此需要採用 Apache Tomcat 作為伺服器軟體，預設 http 之埠號（port number）為 8080。
2. 資料庫採用關聯性資料庫 Oracle MySQL 進行使用，開發階段使用 community 版本即可。
3. 本專案依照 use case 共計有八個模組，每個模組在下份文件中必須進行細部之設計與說明。
4. 使用者之裝置僅須透過瀏覽器即可瀏覽本書城網站。

5.2 架構

本專案預計採用 JAVA 三層架構的程式設計思維，並使用 MVC 的軟體設計模式

5.2.1 Java 三層架構

1. Dao 層
   - ✓ Dao 全稱 Data Access Object（數據訪問對象）主要負責訪問資料庫，

對數據的 CRUD 操作。

- ✓ 獲取結果集返回給 Service 層，不會涉及事務操作。

2. Service 層
   - ✓ Service 層主要負責業務邏輯的實現，實際就是對 Dao 層的增刪改查操作的進一步封裝，涉及到事務的操作。
   - ✓ 獲取資料庫連接，關閉資料庫連接，事務回滾或提交(rollback、commit)或者一些複雜的邏輯業務處理。

3. Controller 層
   - ✓ Controller 層主要用於對業務邏輯進行控制，控制用戶輸入，接收來自前端的請求。
   - ✓ 將需要執行的操作交給 Service 層進行處理，再將處理後的結果返回給前端。

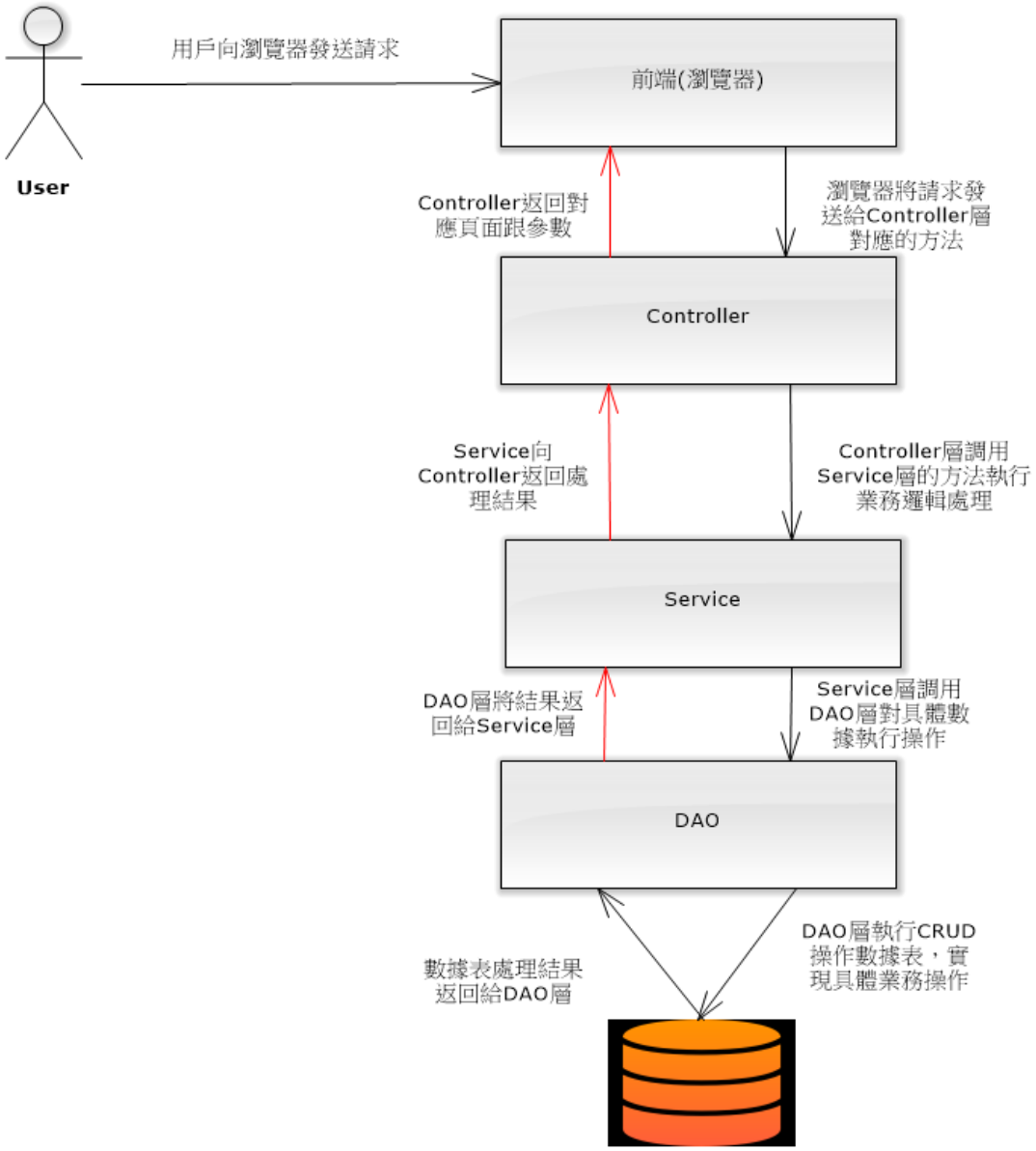


圖 9: JAVA 三層架構

幾乎所有的業務邏輯，實際上最後就是對資料庫表的操作，Dao 層面向資料庫中的表，Service 進行事務、業務邏輯的具體實現，Controller 層對接收到的請求進行控制，然後負責調用 Service 層進行業務邏輯實現，Service 層將邏輯處理中需要對資料庫表的操作交給 Dao 層進行數據操作，最後將處理結果逐層返回給前端，用戶就可以看到最後的處理結果。

5.2.2 MVC 設計模式

MVC 是一種軟體設計模式，將軟體程序分為 3 個核心模組：模型（Model）、視圖（View）、控制器（Controller）

1. 模型（Model）
   - ✓ 管理資料庫，用於數據的封裝和傳輸，實現具體業務功能（事務處理、算法等）。
   - ✓ 在 MVC 的 Model 元件是實作如何儲存應用程式的資料，包含資料和驗證規則。
   - ✓ 以 Web 應用程式來說，Model 元件負責 Web 應用程式的資料存取和處理，即存取和處理儲存在資料庫、文字檔案或 XML 檔案的資料。

2. 視圖（View）
   - ✓ 負責與用戶交互，從模型中獲取數據通過網頁向用戶展示，將用戶請求傳遞給控制器進行處理。
   - ✓ 實作展示邏輯（Presentation Logic）的物件。
   - ✓ Web 應用程式是建立使用者在瀏覽器看到的 HTTP 回應訊息，通常就是 HTML 網頁。
   - ✓ 使用 Model 物件儲存的資料來產生輸出結果，所以 View 元件可以透過 Model 元件取得資料庫的資料，然後將資料庫的資料轉換成有用的資訊來呈現給使用者檢視。

3. 控制器（Controller）
   - ✓ 接收用戶請求，對請求進行處理和轉發，用於業務流程控制，並向模型發送數據。
   - ✓ 整個應用程式的中心，連接 View 和 Model 元件來協調和控制應用程式的執行。
   - ✓ Web 應用程式的 Controller 元件是控制資料處理流程的控制器，負責接收使用者從瀏覽器送出的 HTTP 請求，依請求執行所需操作，即下達指令給 Model 取出所需的資料，然後送至 View 元件來產生顯示結果的 HTML 網頁。

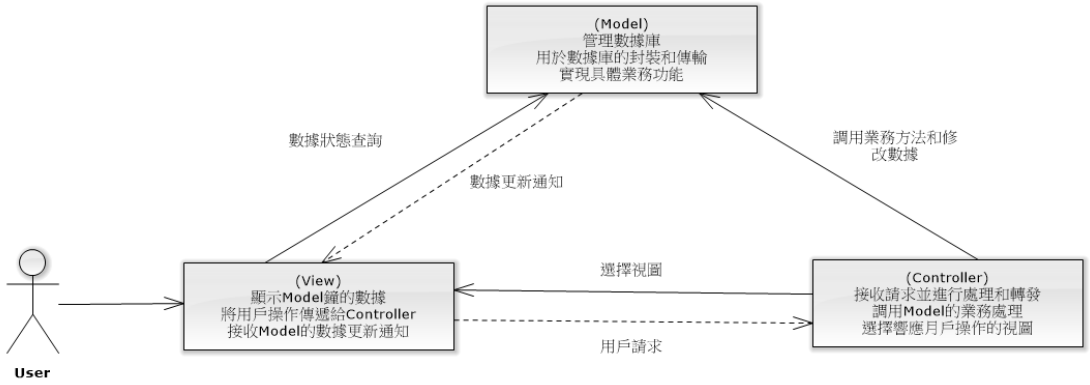


圖 10: MVC 三者關係

用戶通過 View 操作軟體，View 將請求傳遞給 Controller，Model 負責數據的管理，View 負責與用戶交互，Controller 負責對用戶的請求進行響應，同時，當數據更新時會傳給 View 然後更新頁面。

5.2.3 三層體系與 MVC 之間的關係

三層體系和 MVC 之間並不矛盾，三層體系是一種編程思想，目的是為了降低類別之間的耦合，更好的處理業務邏輯；MVC 是一種軟體設計模式，按照功能對軟體進行的模組化的劃分，目的是為了更好的實現軟體開發。二者之間的關係如下：

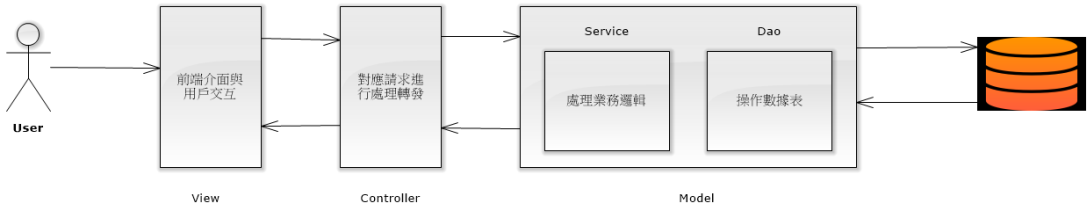


圖 11 三層體系與 MVC 之間的關係(一)

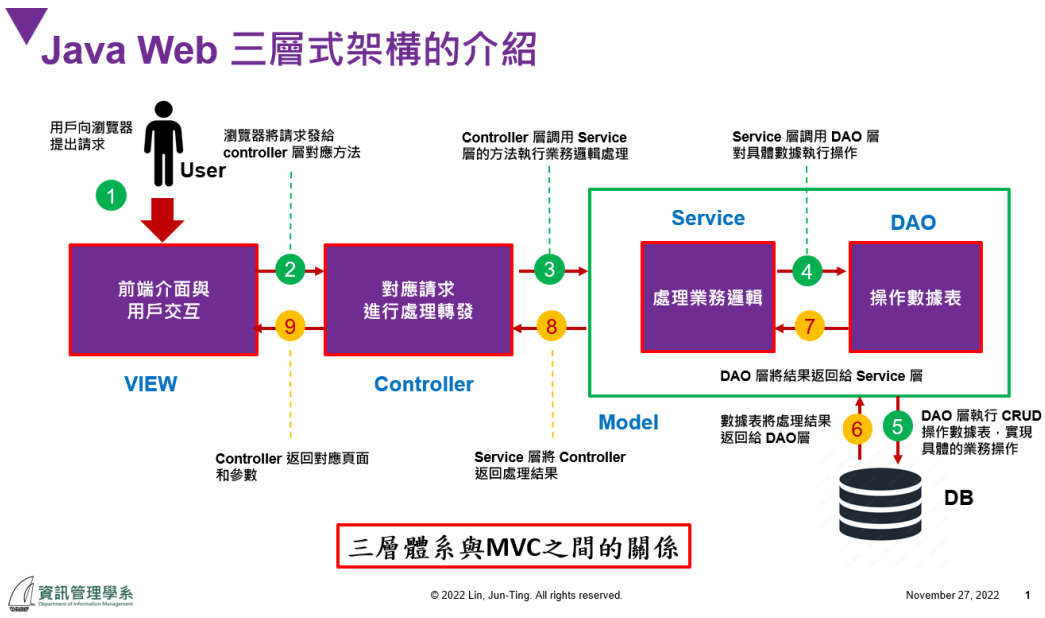


圖 12 三層體系與 MVC 之間的關係(二)