國立臺北商業大學

資 訊 管 理 系

113 資訊系統專案設計

**系統手冊**

一張含有 藝術, 圖畫, 寫生, 黑與白 的圖片

自動產生的描述

**組 別：第113202組**

**題 目：趣書園**

**指導老師：林宏仁老師**

**組 長：11336005 陳平**

**組 員：11236020 李佳祐 11336013 張彤**

**11336008 林嘉珮**

**中華民國113年10月16日**

**目錄**

1. **背景與動機**----------------------------------------------------------------------------------6
   1. 背景介紹--------------------------------------------------------------------------------------6
   2. 動機--------------------------------------------------------------------------------------------6
   3. 系統目的與目標-----------------------------------------------------------------------------6
   4. 預期成果--------------------------------------------------------------------------------------7

**第二章 營運計畫**-------------------------------------------------------------------------------------8

2-1 可行性分析-------------------------------------------------------------------------------------8

2-2 商業模式－Business model------------------------------------------------------------------9

2-3 市場分析—STP--------------------------------------------------------------------------------9

2-4 競爭力分析SWOT-TOWS-----------------------------------------------------------------10

**第三章 系統規格**-----------------------------------------------------------------------------------11

3-1 系統架構---------------------------------------------------------------------------------------11

3-2 系統軟、硬體需求及技術平台------------------------------------------------------------11

3-3 開發標準與使用工具------------------------------------------------------------------------12

**第四章 專案時程與組織分工**--------------------------------------------------------------------13

4-1 專案時程---------------------------------------------------------------------------------------13

4-2 專案組織與分工------------------------------------------------------------------------------14

**第五章** **需求模型**-----------------------------------------------------------------------------------16

5-1 使用者需求------------------------------------------------------------------------------------16

5-2 使用個案圖------------------------------------------------------------------------------------17

5-3 使用個案描述---------------------------------------------------------------------------------17

5-4 分析類別圖------------------------------------------------------------------------------------21

**第六章** **設計模型**-----------------------------------------------------------------------------------22

6-1 循序圖------------------------------------------------------------------------------------------22

6-2 設計類別圖------------------------------------------------------------------------------------25

**第七章 實作模型**-----------------------------------------------------------------------------------26

7-1 佈署圖------------------------------------------------------------------------------------------26

7-2 套件圖------------------------------------------------------------------------------------------26

7-2 元件圖------------------------------------------------------------------------------------------27

7-2 狀態機------------------------------------------------------------------------------------------27

**第八章 資料庫設計**--------------------------------------------------------------------------------28

8-1 資料庫關聯圖---------------------------------------------------------------------------------28

8-2 表格及其Meta data--------------------------------------------------------------------------28

**第九章 程式**-----------------------------------------------------------------------------------------28

9-1 元件清單及其規格描述---------------------------------------------------------------------28

9-2 其他附屬之各種元件------------------------------------------------------------------------28

**第十章 測試模型**-----------------------------------------------------------------------------------28

10-1 測試計畫-------------------------------------------------------------------------------------28

10-2 測試個案與測試結果資料----------------------------------------------------------------28

**第十一章 操作手冊**--------------------------------------------------------------------------------28

系統之元件及其安裝及系統管理--------------------------------------------------------------28

**第十二章 使用手冊**--------------------------------------------------------------------------------28

各畫面、操作之移轉-----------------------------------------------------------------------------28

**第十三章 感想**--------------------------------------------------------------------------------------28

感想與建議-----------------------------------------------------------------------------------------28

**第十四章 參考資料**--------------------------------------------------------------------------------28

參考資料--------------------------------------------------------------------------------------------28

**附錄** --------------------------------------------------------------------------------------------------28

審查評審意見之修正情形-----------------------------------------------------------------------28

圖目錄

圖2-2-1 商業模式分析-----------------------------------------------------------------------------9

圖3-1-1 系統架構----------------------------------------------------------------------------------11

圖5-2-1 使用個案圖-------------------------------------------------------------------------------17圖5-3-1 使用者登入活動圖----------------------------------------------------------------------17

圖5-3-2 新增、編輯故事活動圖----------------------------------------------------------------18

圖5-3-3 搜尋故事活動圖-------------------------------------------------------------------------18

圖5-3-4 閱讀故事活動圖-------------------------------------------------------------------------19

圖5-3-5 設定資料活動圖-------------------------------------------------------------------------19

圖5-3-6 追蹤創作者活動圖----------------------------------------------------------------------20

圖5-4-1 分析類別圖-------------------------------------------------------------------------------21

圖6-1-1 註冊登入循序圖-------------------------------------------------------------------------22

圖6-1-2 新增創作循序圖-------------------------------------------------------------------------22

圖6-1-3 編輯歷史創作循序圖-------------------------------------------------------------------22

圖6-1-4 閱讀繪本循序圖-------------------------------------------------------------------------23

圖6-1-5 搜尋故事循序圖-------------------------------------------------------------------------23

圖6-1-6 查看其他公開創作的循序圖----------------------------------------------------------23

圖6-1-7 收藏故事循序圖-------------------------------------------------------------------------24

圖6-1-8 追蹤創作者循序圖----------------------------------------------------------------------24

圖6-1-9 設定個人檔案循序圖-------------------------------------------------------------------24

圖6-2-1 設計類別圖-------------------------------------------------------------------------------25

圖7-1-1 佈署圖-------------------------------------------------------------------------------------26

圖7-2-1 套件圖-------------------------------------------------------------------------------------26

圖7-3-1 元件圖-------------------------------------------------------------------------------------27

圖7-4-1 狀態機-------------------------------------------------------------------------------------27

圖8-1-1 資料庫關聯圖----------------------------------------------------------------------------28

表目錄

表2-4-1 SWOT分析--------------------------------------------------------------------------------10

表2-4-2 TOWS分析--------------------------------------------------------------------------------10

表3-2-1 系統軟、硬體需求及技術平台-------------------------------------------------------11

表3-3-1 開發標準與使用工具-------------------------------------------------------------------12

表4-1-1 專案時程表-------------------------------------------------------------------------------13

表4-2-1 專案組織與分工表----------------------------------------------------------------------14

表4-2-2 專題成果工作內容與貢獻度表-------------------------------------------------------15

表5-1-1 功能性需求清單-------------------------------------------------------------------------16

表8-2-1、T01使用者資訊------------------------------------------------------------------------28

表8-2-2、T02追蹤記錄---------------------------------------------------------------------------29

表8-2-3、T03故事---------------------------------------------------------------------------------29

表8-2-4、T04故事類型---------------------------------------------------------------------------30

表8-2-5、T05故事分頁---------------------------------------------------------------------------30

表8-2-6、T06故事收藏---------------------------------------------------------------------------31

1. **背景與動機**
   1. 背景介紹

家長的陪伴及關愛對孩子的人生極具影響力，是促使孩子快樂成長十分重要的一環，而藉由說故事的方式，不但可以深化家長與孩子之間的感情，也能讓孩童更容易理解我們所想表達的事物，但在實際生活中，普遍家長並沒有足夠的時間去準備故事讀本以陪伴、教育他們的孩子。為了有效的改善這樣的情況，我們認為透過時下流行的「生成式AI」輔助家長快速生成故事，可以有效的製作家長們所期望的繪本，更能讓孩子們擁有豐富美好的童年。

* 1. 動機

陪伴是兒童在成長過程中不可或缺的一部分，我們希望在近幾年快速發展的生成式AI的輔助下，儘管是再忙碌的家長都能將想表達的事物透過故事好好地傳達給他們親愛的孩子們。

另外，AI能夠提供豐富的素材和故事，進而激發孩子的好奇心和創造潛能，鼓勵他們大膽創新，自由探索，全面提升他們的創造力和綜合素養。

* 1. 系統目的與目標

本平台的目的是希望在AI圖文生成的輔助下，讓家長能輕易地將想表達給孩子的事物轉換成簡單易懂的故事。

我們希望透過下列幾項功能，以達成上述的目標：

◆ 故事生成：使用者在輸入簡單的故事大綱後能夠生成出適合兒童閱讀的短篇故事。

◆ 故事閱覽：使用者在閱讀故事時能得到趨近於翻閱實體書的優良體驗。

◆ 故事收藏：使用者可以欣賞其他創作者公開的故事並加入收藏，甚至基於其他創作者的作品進行再創作。

◆ 創作者追蹤：使用者可以追蹤自己喜愛的創作者，並在其發表最新作品時獲得第一手消息。

* 1. 預期成果

◆ 親子互動：家長可以透過此平台加深與孩子的感情。

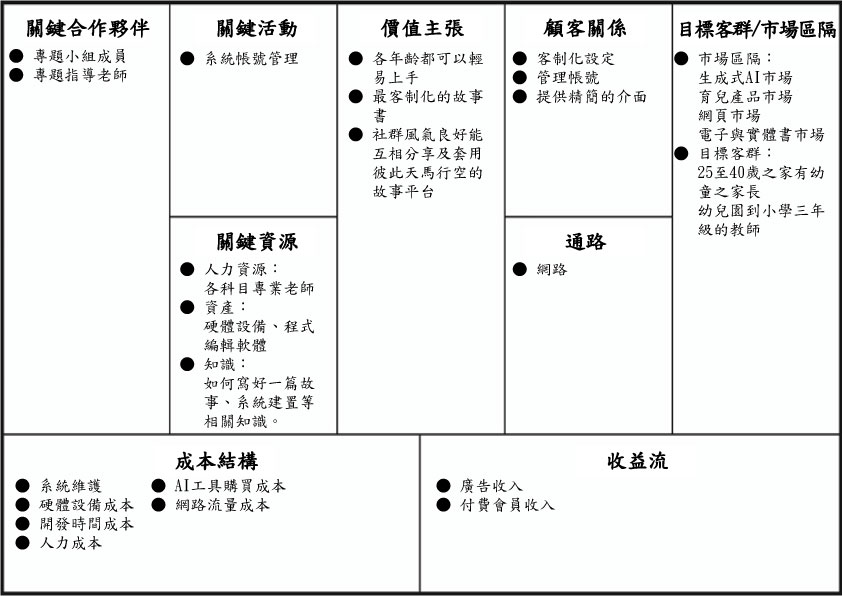
◆ 操作流暢：使用者能流暢生成完整且滿意的繪本。

**第二章 營運計畫**

2-1 可行性分析

* 1. 營運可行性：近年來雙薪家庭比例的高升，導致父母育兒的時間減少，隨著3C產品的高普及性及生成式AI的爆紅，大家對於AI產品接受度有所提升， 而故事書在育兒產品當中仍占有一席之地，但實體書籍卻有著需要付費及攜帶的不便性，我們的系統正好能補足這個缺口。
  2. 經濟可行性：系統上架初期為了快速提升使用者的數量，及收集用戶數據，將先採取免費註冊使用，往後隨著使用者數量的累積我們將採取付費升級制，如:GPT-4，每月有付費的用戶可使用更加客制化的AI及新開發的功能，再加上供業者投放廣告，相信有這兩個資金流，讓我們能購買更為高階的AI工具及放膽地開發新功能、優化系統，使本系統提升穩定性與降低生成的響應時間，最佳化使用者的用戶體驗，形成良性循環。
  3. 技術可行性：本專題涉及多種技術來實現系統開發，組員都具撰寫網頁程式碼的基本能力，也對於資料庫的建置與使用有著較高的理解，加上修習過系上的系統分析、UML等課程，使得我們能畫出更完善的系統架構圖及更富有邏輯的規畫來建制本系統，本系統所需的AI工具也都有進行購買，而在開發時遇到困難也可詢問專題老師，或運用網路上的相關資源、AI工具、及最為重要的自學能力來排除問題，最後因有五專時期專題的開發經驗，使得我們對於這次專題的時程能有更近一步的掌握。

2-2 商業模式－Business model



▲圖2-2-1 商業模式分析

2-3 市場分析－STP

市場區隔(Segmenting)：主要鎖定幼教市場、家庭。

目標市場(Targeting)：25至40歲之家有幼童之家長及幼稚園至小學二、三年級之教師。

產品定位(Positioning)： 本平台主打用簡單的描述創作出幼童易懂的繪本故事。

2-4 競爭力分析SWOT-TOWS

▼表2-4-1 SWOT分析

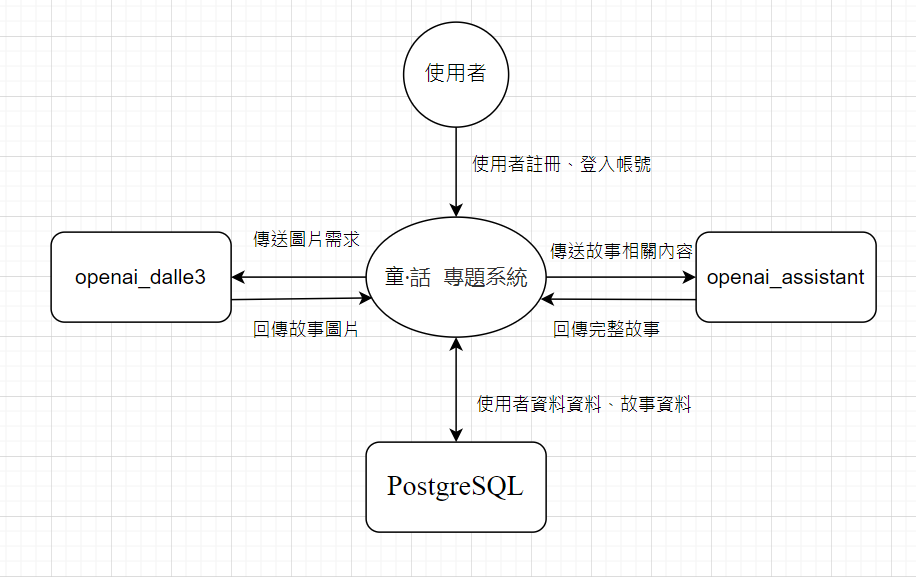
|  |  |
| --- | --- |
| 優勢 | 劣勢 |
| * 較類似產品功能完善 * 操作步驟簡短，學習成本低 * 系統互動性、編輯性較高 * 內建AI繪本內文語音 | * 生成圖片時程長 * 無法生成完全一樣的圖片 |
| 機會 | 威脅 |
| * 巧連智停刊 * AI生成熱潮 * 親子互動時間少 * 3C產品普及家家戶戶 | * GPTs商店內有類似功能的應用 * 少子化 * 實體書不易取代 |

▼表2-4-2 TOWS分析

|  |  |
| --- | --- |
| 優勢/機會 | 優勢/威脅 |
| * 將生成故事書的流程最簡單化 * 完善個平台的兼容性 * 增加自行訓練AI語音的功能 | * 提高故事的客製化 * 獨家的故事看板角色 |
| 劣勢/機會 | 劣勢/威脅 |
| * 縮短生成圖片的時間 | * 訓練出獨有的圖片風格 * 生成出適合成年人閱讀的繪本 |

**第三章 系統規格**

3-1 系統架構



▲圖3-1-1 系統架構

3-2 系統軟、硬體需求及技術平台

▼表3-2-1 系統軟、硬體需求及技術平台

|  |  |  |
| --- | --- | --- |
| 手機硬體需求 | | |
| 作業系統 | https://lh3.googleusercontent.com/LRO6Sdzt7hZ5KA_50y2mWl_iDruuEu-fyMoUZWwK33muX9V-sOY0NLNIfdxfLTEgfeHeF37MxwMIXGulX4f21ZBNPOx8Hnh4ctzn6_CVYMldM0296vTFzHVa8N6NzjANTzox6r2-kKKDChwO_Ahttps://lh6.googleusercontent.com/DiqTxFDQ4DICh1npxrOsCc1d-nWpF0PtFvl3wqPckhfEh6Oh0TNiA-XdKQRljF0wwaaRth62sGdAO9drW-MuWYgF4kQ9Ao1J17ut46paP6Z074GrPjaGVpOCZjdD9pOlberUp6_QLypLooS1vw | Android/iOS |
| 電腦硬體需求 | | |
| 作業系統 |  | Windows 10(含)以上版本之電腦 |
| 手機軟體需求 | | |
| 系統需求 | https://lh6.googleusercontent.com/EnI1PRQxGXZ--KWbyytYfVoo56r4VPse3LVMCpNW10zwZCXIeAhzcZNFPC4SKD6siZT3j4jxMzMh0WnRZRW5_Kx169UurDzVLRG34E6ToSanDEoXZi-XEizo525rHXnm62Fs_PQ5pPWZpDUP1w | 瀏覽器 |
| 網路需求 | | |
| 網路 | https://lh4.googleusercontent.com/0DRlSbDXYTfEQ8u20vQAqY8mHuF0NE8n1IEjzq-mKWl93dihN9C4xinprv6Tqd0KNgTBZf9Gn5n5zelOMQwCBfqUQG8IEw_LWQb-V35rPXaxepnyo_yG_-qSYHykWOIt31vVT2-f62QZ7u9hKwhttps://lh6.googleusercontent.com/-1GmrjIFgwIkagVpvkv5PYm4QopjAPS8yr2Eb3YeXGzpT6Ym8z5Gdp-8SCY7V_TvmUYUmUzPPQtVZo6tujbUjeQMCANJXexP3zwl_uMdqN0rHBkXaeEuu8SPtwh0HdDLItf7DNP_NXOOJ_N6wQhttps://lh3.googleusercontent.com/PErd9-kDQoVCinmzjNdfSjmo-IVehmintiYJjYKs-HOpI_cJNMrJ2vup68SuxTbOT9SeEa5XFBqnvK9CjYObFL1P21wFIoWChY_uspwtWDLpSyxalAYbL033Chwd8V4M0JblbwVOKF9BN5cOGwhttps://lh4.googleusercontent.com/n6uAxRbR5YQQqzu2r6S2KdvEVDEtdBTkKVMTahobjAePsb8K-vCFvXUVQf-OCd47fHspNwgQyz3vx7r_mhiY1ODzHSYKwU0F57NVNjJSpa1-Ka2OwQfBzvjYzXBeVbPfoiJj5LJc3wiCBNmPGA | Wi-Fi/乙太網路/4G/5G |

3-3 開發標準與使用工具

▼表3-3-1 開發標準與使用工具

|  |  |  |
| --- | --- | --- |
| 系統開發環境 | | |
| 作業系統 | https://lh5.googleusercontent.com/fO35PI2iDtHZ-BKydeuW64C_gOD3ilAqFDfYR85T7OKeWYcDRt94T5GzuLCTiNYXXOw33ibDKpdoCkv-u8i40n7vaqJY1wG4SDi7u_7eS2M-bmHei7PjoDe6GPj5Zl_bwuU5nbCSVQ_osk41Nwhttps://lh5.googleusercontent.com/Io5R6mXFSoO0F3bA1BkFxJkygZyp92pslaqRjxFTBkpEZI1ij3-HVnmnOycUek-B_Y6Ar3CAHDpnV7g94toZsywW84tAshHB_I287-8oFjfG7rAuBrZp2Pajj_ufSdufHAwqSeREitsrnbvimw | Windows/MacOS |
| 撰寫工具 |  | Visual Studio Code |
| 程式語言 | https://lh6.googleusercontent.com/sSSGrprxAFQFDkAIJjNid-21jaROCId7AOYLmK5F4kjp4LfIcC5rlmvuKGxngigXkCXqj6ogCAvK7R2Ykq4y07wJNbet6I2-sro6RJSplFGUSTmKyc5TVt8OMiiLuL5vdFa9zAQFJKW1sDn-1w | Python |
| 程式開發工具 | | |
| 前端 | https://lh5.googleusercontent.com/xSPtF7BK7slK8lk8jY17zvvbXqmYfjt9cDZADzU4nGRR6A0jIq43zTIH6uQ41yaHzlttzfjMRRR73nN6IArkckodIumuBHIPVaBWRou4CPQxyXO1b9xf12zAiGYK6_uxXch_ZxSzBUKwURVSDghttps://lh5.googleusercontent.com/uDslkAwoxYJgU1Rl1G1NBBt4Sw4HmzXVxEfFnh8arWW-JtS6-K9S3iu1x-1Ue4yPlxqDt71gbKkatyYlIPGyyibOnwOd-4ZEvLrWlef5kRt6ePSuo3gggPd690Hj98BhWEJNOFxj8J8XYiB_NQhttps://lh5.googleusercontent.com/zCGOw9F7w8rhIBcKx1Q1i7Py1MBE7X2CS9K4WD2-SMPA1GWaTJT3fMaCalwwUOBp7d7RkwEuPove2CAk21oS9l94qkO_zy6srnoc9x2WfqT349GdkqSjRA6TBw4WAseO1qZqEXCzqFg9sGq8ww | Html / CSS/ JavaScript |
| 後端 |  | Flask(Python) |
| 資料庫管理 |  | DBeaver |
| 資料庫 | https://lh6.googleusercontent.com/bMTJvVGGfcX_xSRKi8uODactXXi3MsbzNdwvNLPee8nuHxx-kETE5ryR11c9mEVdGX4zeAjCEp8o2w29uoxDfQ7qNCf7Q5SKtT_ouqRpENAWkuP-i8Z2YGW4PelEJZ9ffnQ0hGmiDOuD9MykqA | PostgreSQL |
| 介面及插圖繪製工具 | | |
| 插圖與logo | https://lh3.googleusercontent.com/Rn551RZXQbPTr4x7UcCtda8EOgoGj2xnAZSJIZO03bipqyoS8rQpCAiOPtB_VdimFP7skWNP4OZVcnzbZ-lk2uP81FwL_GpOIMJrEfLQeWm9L2QXrBUwFByhLEvNbHiNtwhd-b-lwag4XLPTRAhttps://lh5.googleusercontent.com/_-Wo0GI_3kNwC9YUY1zQkFraooBZuyJ2VfypPvOo3aXwi0C9JD5tpbJ8MKjRXG9A9jdeCE5EaqNQtnGMWb43o5GnCB1o7W2PD_49ZSstWqXgTBf7oZacyU6po-i3vbKcF_OkBk7c3VUIVhVqag | Adobe Illustrator / Photoshop |
| 文件及美化工具 | | |
| 文件 | https://lh3.googleusercontent.com/shvOof2jIp6XQY9tKzg-KPQREnc4GKrPlj_T78V4RgBOYEqQssFx40JJffhh3OxMukx7z4CiJrd9GZEYsn2jpPBcMYdo7-9rRNkOGWv86YAmY6ygWj-9XZdgq1hcNmqi3o3dTkMlHfmFCshMRw | Microsoft Word |
| 圖表 | https://lh3.googleusercontent.com/td-PGR-E_Sue47ez0uT6L0uL9JpLoc-7jVp0r4-q7BBxVsHa4UFW9YEIuQPvAy3L7Q8_kF3QooZXGJ3oFhQsfLFHX_5XOpWLM7oLNrixrnQrDr2nIbzbnki-2PgdXHbxiarh16upQZk_n57acg | Draw.io |
| 簡報 | https://lh3.googleusercontent.com/Wy_cg0w4ZX9g0lRKTT19hSeonSdpRuo8ZPswHzriIAeNZMa71F32nzfBCbiEtFeq9ifh6-WdwoV62gX4pxrl7gao5Wjdgdy-QOw5c3rWr4KH5EYVTKLqINMZ9TIUwerTVFFZaRDVEiu7moX-1Q | PowerPoint、Canva |
| 專案管理及版本控制工具 | | |
| 應用程式 |  | fork |
| 版本控制 |  | git |

**第四章 專案時程與組織分工**

4-1 專案時程

▼表4-1-1 專案時程表

|  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
|  | 2024年 | | | | | | | | | | | | | |
| 任務名稱/時間 | 2月 | | 3月 | 4月 | 5月 | 6月 | | 7月 | 8月 | 9月 | | 10月 | 11月 | 12月 |
| 主題構想  功能構想 |  | |  |  |  |  | |  |  |  | |  |  |  |
|  | |  |  |  |  | |  |  |  | |  |  |  |
| Logo設計 |  | |  |  |  |  | |  |  |  | |  |  |  |
|  | |  |  |  |  | |  |  |  | |  |  |  |
| 使用者  介面設計 |  | |  |  |  |  | |  |  |  | |  |  |  |
|  | |  |  |  |  | |  |  |  | |  |  |  |
| 資料庫建置 |  | |  |  |  |  | |  |  |  | |  |  |  |
|  | |  |  |  |  | |  |  |  | |  |  |  |
| 伺服器架設 |  | |  |  |  |  | |  |  |  | |  |  |  |
|  | |  |  |  |  | |  |  |  | |  |  |  |
| 資料庫調整與連接 |  | |  |  |  |  | |  |  |  | |  |  |  |
|  | |  |  |  |  | |  |  |  | |  |  |  |
| 前端程式撰寫 |  | |  |  |  |  | |  |  |  | |  |  |  |
|  | |  |  |  |  | |  |  |  | |  |  |  |
| 後端程式撰寫 |  | |  |  |  |  | |  |  |  | |  |  |  |
|  | |  |  |  |  | |  |  |  | |  |  |  |
| 前端  整合與測試 |  | |  |  |  |  | |  |  |  | |  |  |  |
|  | |  |  |  |  | |  |  |  | |  |  |  |
| 後端  整合與測試 |  | |  |  |  |  | |  |  |  | |  |  |  |
|  | |  |  |  |  | |  |  |  | |  |  |  |
| 系統  整合與測試 |  | |  |  |  |  | |  |  |  | |  |  |  |
|  | |  |  |  |  | |  |  |  | |  |  |  |
| 文件製作 |  | |  |  |  |  | |  |  |  | |  |  |  |
|  | |  |  |  |  | |  |  |  | |  |  |  |
| 簡報製作 |  | |  |  |  |  | |  |  |  | |  |  |  |
|  | |  |  |  |  | |  |  |  | |  |  |  |
| 預計完成： | |  | | | | |  | | | |  | | | |
| 實際完成： | |  | | | | |  | | | |  | | | |

4-2 專案組織與分工

▼表4-2-1 專案組織與分工表

●主要負責人 〇次要負責人

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| 項目/組員 | | 10856005  陳平 | 10856007  張彤 | 10856043  林嘉珮 | 11236020  李佳祐 |
| 後端開發 | 資料庫建置 | 〇 |  | ● | 〇 |
| 資料庫連接 |  | 〇 | ● | 〇 |
| 伺服器架設 | 〇 |  | 〇 | ● |
| 程式撰寫 |  | 〇 | ● | 〇 |
| 整合與測試 | 〇 |  | 〇 | ● |
| 前端開發 | 功能構想 |  | ● | 〇 | 〇 |
| 程式撰寫 | ● | 〇 | 〇 |  |
| 網站架設 | ● | 〇 |  | 〇 |
| 整合與測試 | 〇 | ● | 〇 |  |
| 美術設計 | UI/ UX | ● | 〇 |  | 〇 |
| Web/APP介面設計 | 〇 | ● | 〇 |  |
| 色彩設計 | 〇 | 〇 |  | ● |
| Logo設計 | 〇 | ● | 〇 |  |
| 素材設計 | ● | 〇 |  | 〇 |
| 文件撰寫 | 統整 | ● | 〇 | 〇 |  |
| 第1章 前言 | ● | 〇 |  | 〇 |
| 第2章 營運計畫 | 〇 |  | 〇 | ● |
| 第3章 系統規格 |  | 〇 | 〇 | ● |
| 第4章 專題時程與組織分工 |  | 〇 | ● | 〇 |
| 第5章 需求模型 | 〇 | ● | 〇 |  |
| 第6章 程序或設計模型 | 〇 | ● |  | 〇 |
| 第7章 資料或實作模型 | ● |  | 〇 | 〇 |
| 第8章 資料庫設計 |  | 〇 | ● | 〇 |
| 第9章 程式 | 〇 |  | ● | 〇 |
| 第10章 測試模型 |  | 〇 | 〇 | ● |
| 第11章 操作手冊 | 〇 | ● |  | 〇 |
| 第12章 使用手冊 | 〇 |  | 〇 | ● |
| 報告 | 簡報製作 | 〇 | 〇 | ● |  |

▼表4-2-2 專案組織與分工表

|  |  |  |  |
| --- | --- | --- | --- |
| 序號 | 姓名 | 工作內容 | 貢獻度 |
| 1 | 組長  陳 平 | 文件製作(UML、整合)、繪本瀏覽頁、網頁模板、繪本語音 | 25 % |
| 2 | 組員  張 彤 | 文件製作(UML)、網頁設計、UI/UX | 25 % |
| 3 | 組員  林嘉珮 | 文件製作(分工、資料庫)、資料庫、整體串連、登入登出 | 30 % |
| 4 | 組員  李佳祐 | 文件製作(分析、規格說明)、AIGC研究、伺服器架設 | 20 % |
|  | | | 總計:100% |

**第五章 需求模型**

* 1. 使用者需求
  2. 功能性需求

▼表5-1-1 功能性需求清單

|  |  |
| --- | --- |
| 使用案例 | 說明 |
| 使用者註冊&登入 | * 使用者註冊並登入使用網頁 |
| 新增及編輯故事 | * 新增創作 * 編輯歷史創作 |
| 搜尋故事 | * 搜尋其他使用者創作的故事 * 查看其他使用者公開的創作 |
| 閱讀故事 | * 閱讀包含文字及圖片內容 |
| 收藏故事 | * 收藏其他使用者創作的故事 |
| 追蹤創作者 | * 追蹤其他使用者, 並可以查看他們創作的最新故事 |
| 設定個人資料 | * 修改使用者的暱稱及頭像 |

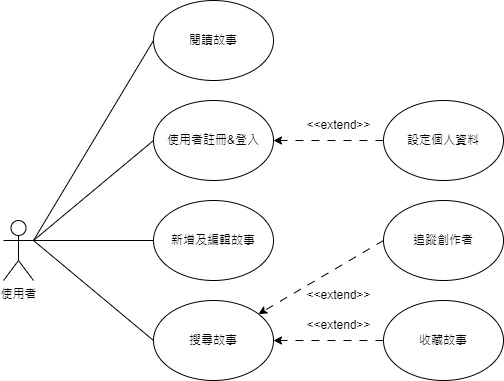
* 1. 非功能性需求

a.需要登入後才能使用系統功能

b.使用者需有Wi-Fi或行動網路才能瀏覽創作區與盡情上傳或下載

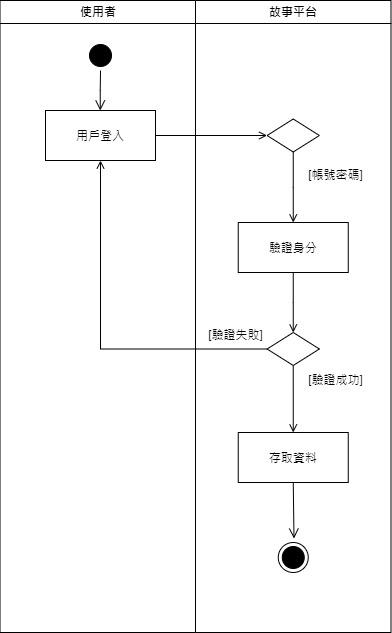
c. 使用者的個人資料受到隱私保護

5-2 使用個案圖

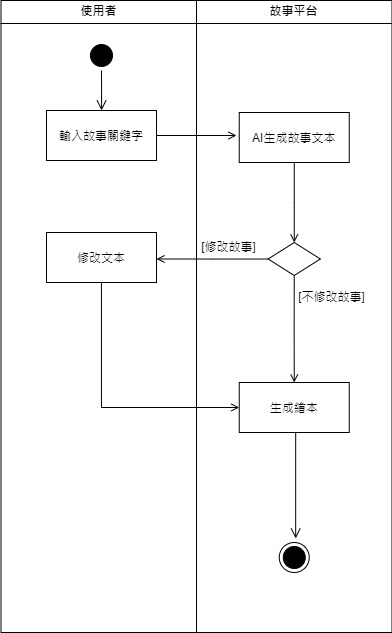


▲圖5-2-1 使用個案圖

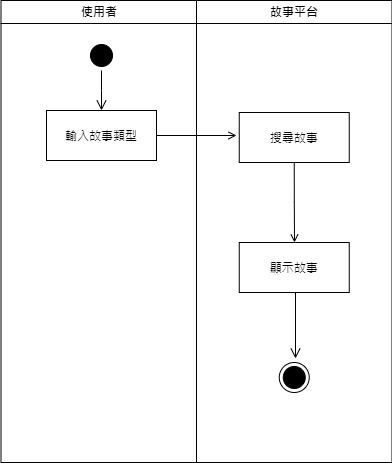
5-3 使用個案描述



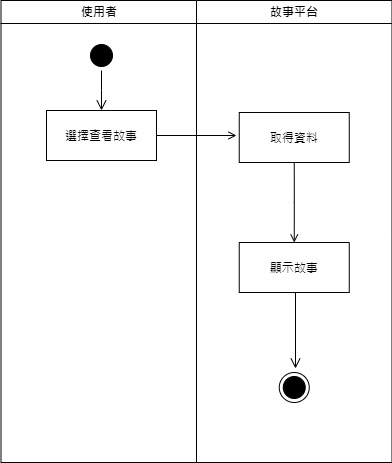
▲圖5-3-1 使用者登入活動圖



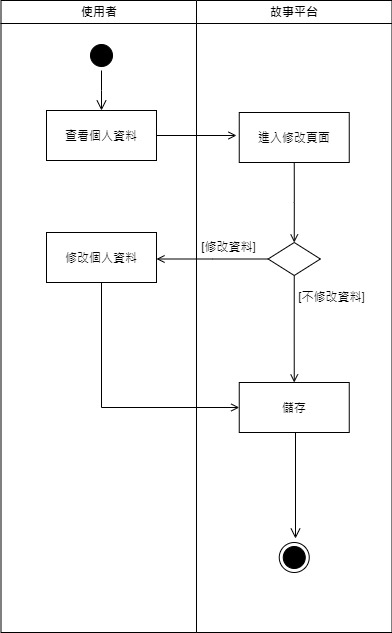
▲圖5-3-2 新增、編輯故事活動圖



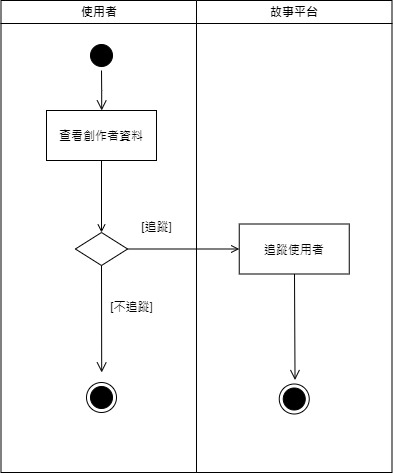
▲圖5-3-3 搜尋故事活動圖



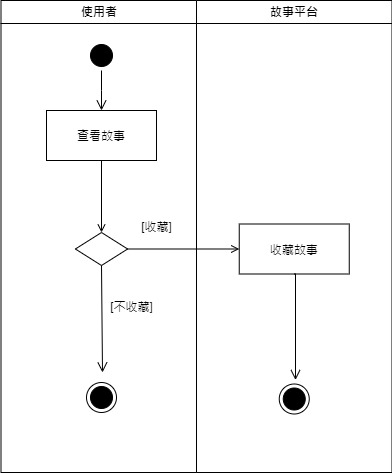
▲圖5-3-4 閱讀故事活動圖



▲圖5-3-5 設定資料活動圖

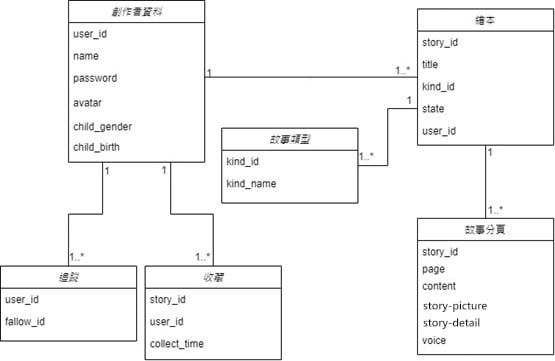


▲圖5-3-6 追蹤創作者活動圖



▲圖5-3-7 收藏故事活動圖

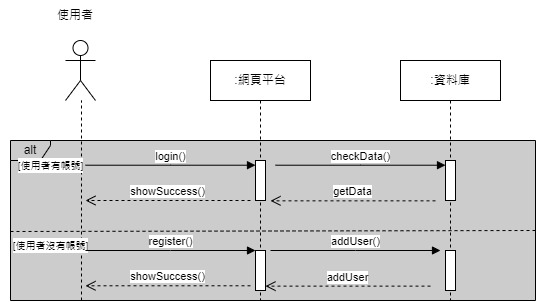
5-4 分析類別圖



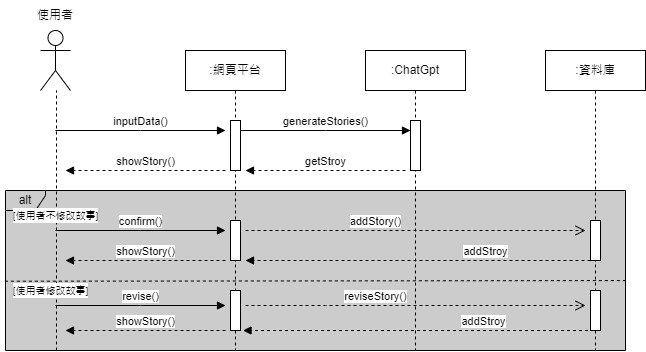
▲圖5-4-1 分析類別圖

**第六章 設計模型**

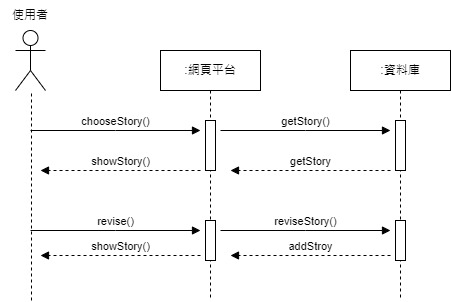
6-1 循序圖



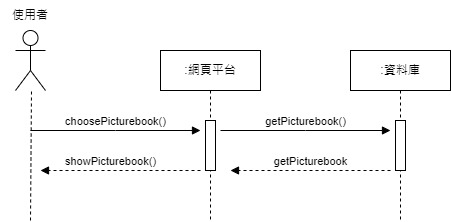
▲圖6-1-1 註冊登入循序圖



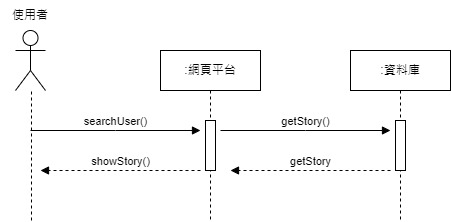
▲圖6-1-2 新增創作循序圖



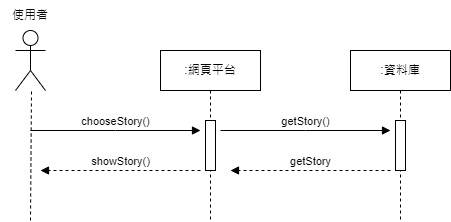
▲圖6-1-3 編輯歷史創作循序圖



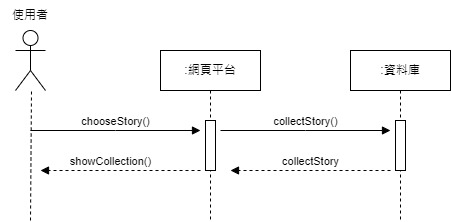
▲圖6-1-4 閱讀繪本循序圖



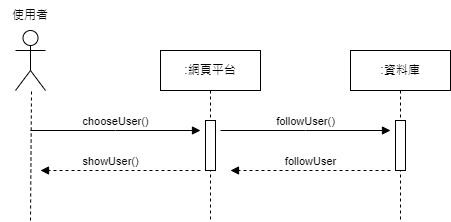
▲圖6-1-5 搜尋故事循序圖



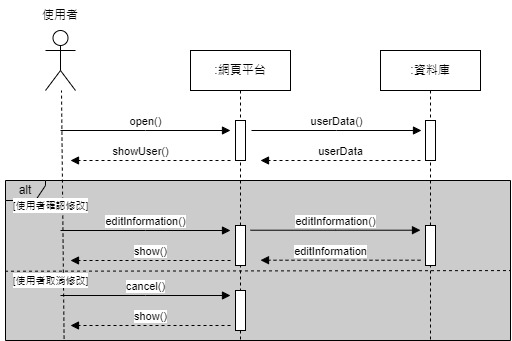
▲圖6-1-6 查看其他公開創作的循序圖



▲圖6-1-7 收藏故事循序圖

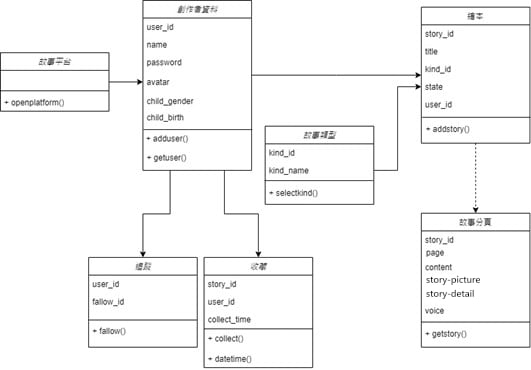


▲圖6-1-8 追蹤創作者循序圖



▲圖6-1-9 設定個人檔案循序圖

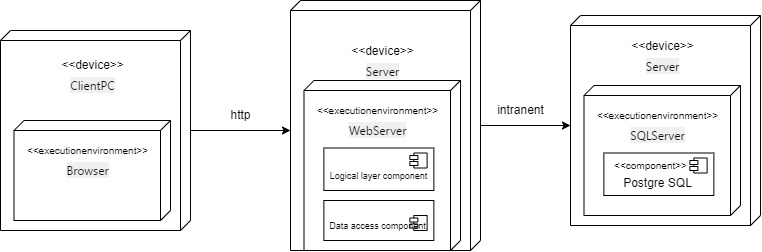
6-2 設計類別圖



▲圖6-2-1 分析類別圖

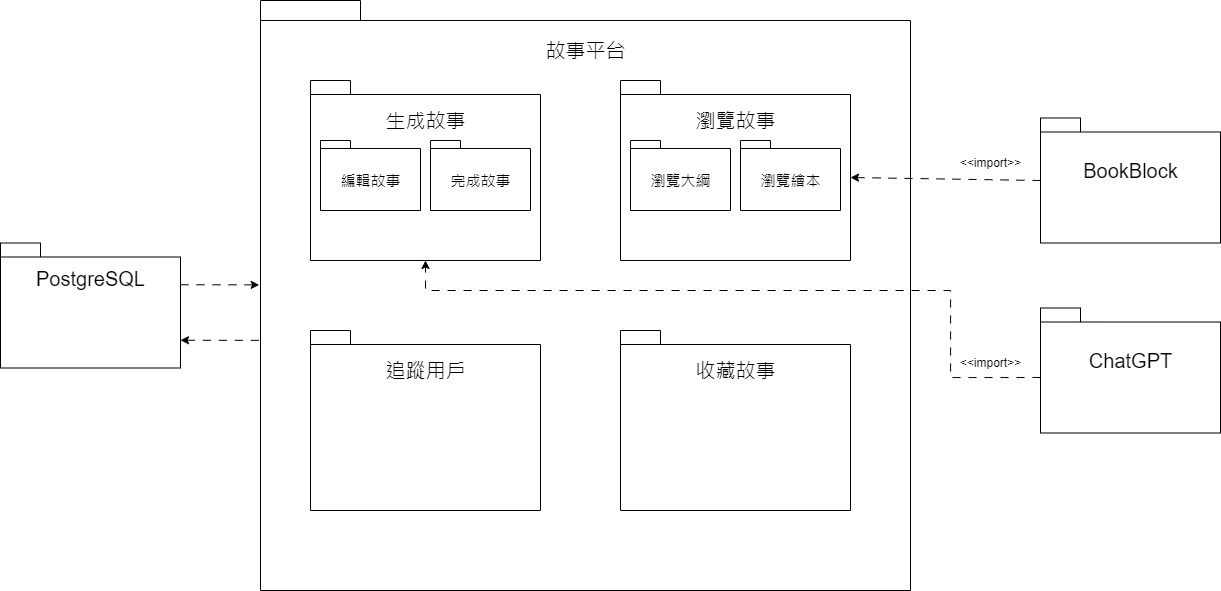
**第七章 實作模型**

7-1 佈署圖



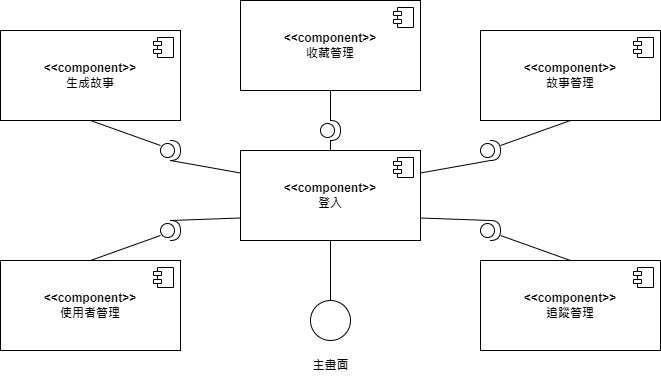
▲圖7-1-1 佈署圖

7-2 套件圖



▲圖7-2-1 套件圖

7-3 元件圖



▲圖7-3-1 元件圖

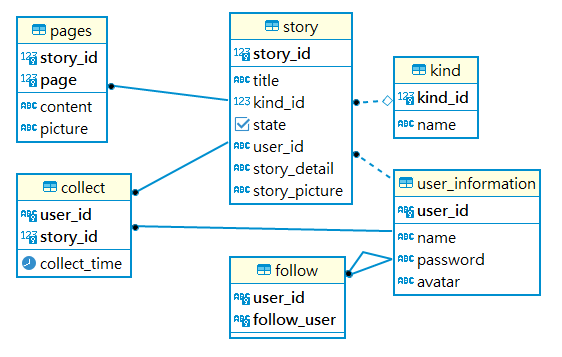
7-4 狀態機



▲圖7-4-1 狀態機

**第八章 資料庫設計**

8-1 資料庫關聯圖



▲圖8-1-1 資料庫關聯圖

8-2 表格及其Meta data

▼表8-2-1、T01使用者資訊

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| 中文名稱 | 使用者資訊 | | 資料表編號 | T01 | |
| 英文名稱 | user\_information | | 索引 | user\_id(PK) | |
| 資料表陳述 | 用於儲存使用者的基本資訊 | | | | |
| 欄位名稱 | 資料型態 | 長度 | 唯一性 | 允許空值 | 中文名稱 |
| user\_id | varchar | 30 | 是 | 否 | 使用者編號 |
| name | varchar | 20 | 是 | 否 | 使用者名稱 |
| password | text | － | 否 | 是 | 密碼 |
| avatar | text | － | 否 | 是 | 頭像 |

▼表8-2-2、T02追蹤記錄

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| 中文名稱 | 追蹤記錄 | | 資料表編號 | T02 | |
| 英文名稱 | follow | | 索引 | user\_id(PK1,FK)  follow\_id(PK2,FK) | |
| 資料表陳述 | 用於記錄使用者追蹤其他使用者 | | | | |
| 欄位名稱 | 資料型態 | 長度 | 唯一性 | 允許空值 | 中文名稱 |
| user\_id | varchar | 30 | 否 | 否 | 使用者編號 |
| follow\_\_id | varchar | 30 | 否 | 否 | 追蹤編號 |

▼表8-2-3、T03故事

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| 中文名稱 | 故事 | | 資料表編號 | T03 | |
| 英文名稱 | story | | 索引 | story\_id(PK)  kind\_id(FK)  user\_id(FK) | |
| 資料表陳述 | 用於儲存使用者創作的故事 | | | | |
| 欄位名稱 | 資料型態 | 長度 | 唯一性 | 允許空值 | 中文名稱 |
| story\_id | serial | 4 | 是 | 否 | 故事編號 |
| title | varchar | 30 | 否 | 是 | 故事名稱 |
| kind\_id | int | 4 | 否 | 是 | 故事種類 |
| state | bool | ­－ | 否 | 是 | 故事狀態 |
| user\_id | varchar | 30 | 否 | 否 | 使用者編號 |
| story\_detail | Text | － | 否 | 是 | 故事簡介 |
| story\_picture | text | － | 否 | 是 | 故事封面 |

▼表8-2-4、T04故事類型

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| 中文名稱 | 故事類型 | | 資料表編號 | T04 | |
| 英文名稱 | kind | | 索引 | kind\_id(PK) | |
| 資料表陳述 | 用於儲存故事類型 | | | | |
| 欄位名稱 | 資料型態 | 長度 | 唯一性 | 允許空值 | 中文名稱 |
| kind\_id | serial | 4 | 是 | 否 | 類型編號 |
| name | varchar | 10 | 否 | 是 | 類型名稱 |

▼表8-2-5、T05故事分頁

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| 中文名稱 | 故事分頁 | | 資料表編號 | T05 | |
| 英文名稱 | page | | 索引 | story\_id(PK1, FK)  page(PK2) | |
| 資料表陳述 | 用於儲存故事每一頁的內容 | | | | |
| 欄位名稱 | 資料型態 | 長度 | 唯一性 | 允許空值 | 中文名稱 |
| story\_id | int | 4 | 否 | 否 | 故事編號 |
| page | int | 4 | 否 | 否 | 分頁編號 |
| content | text | － | 否 | 是 | 故事內容 |
| picture | text | － | 否 | 是 | 故事圖片 |

▼表8-2-6、T06故事收藏

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| 中文名稱 | 故事收藏 | | 資料表編號 | T07 | |
| 英文名稱 | collect | | 索引 | user\_id (PK1, FK)  story\_id (PK2, FK) | |
| 資料表陳述 | 用於記錄使用者收藏的故事 | | | | |
| 欄位名稱 | 資料型態 | 長度 | 唯一性 | 允許空值 | 中文名稱 |
| user\_id | varchar | 30 | 否 | 否 | 使用者編號 |
| story\_id | int | 4 | 否 | 否 | 故事編號 |
| collect\_time | date | － | 否 | 是 | 收藏時間 |

**第九章 程式**

9-1元件清單及其規格描述

▼表9-1-1 程式清單

|  |  |  |
| --- | --- | --- |
| 編號 | 檔案名稱 | 功能名稱 |
| 1 | app.py | 專案主程式 |
| 2 | home.html | 網站首頁 |
| 3 | login.html | 網站登入頁 |
| 4 | register.html | 網站註冊頁 |
| 5 | index.html | 網站導覽頁 |
| 6 | creation.html | 繪本創作頁 |
| 7 | edit.html | 繪本編輯頁 |
| 8 | edit2.html | 繪本二次編輯頁 |
| 9 | follow.html | 被追蹤用戶之歷史創作 |
| 10 | followStory.html | 繪本收藏 |
| 11 | myStory.html | 用戶歷史創作 |
| 12 | story-details.html | 繪本簡介 |
| 14 | user-profile.html | 用戶資料 |
| 15 | test2.html | 繪本閱讀 |
| 16 | forget.html | 忘記密碼 |
| 17 | collection.html | 繪本收藏瀏覽頁 |

▼表9-1-2 規格功能描述-app.py

|  |  |
| --- | --- |
| 程式名稱 | app.py |
| 目的 | 負責建立路由，並串聯每一個網頁，對資料進行驗證，並從資料庫抓取所需資料 |
| 部分程式碼 | |
| # 匯入模組  from flask import (Flask, render\_template, request, Response, jsonify, redirect, url\_for, session, send\_from\_directory)  from utils import db # 自定義模組，處理資料庫相關功能  from datetime import datetime  import requests # 用於處理 HTTP 請求  import bcrypt # 用於密碼加密和驗證  import os # 用於操作系統相關操作，例如路徑管理  import time # 用於時間相關操作  import re # 用於正則表達式處理  import shutil # 用於文件和目錄操作  import logging # 用於記錄日誌  import hashlib # 用於雜湊函數處理  from werkzeug.utils import secure\_filename # 用於安全地處理文件名  from openai import OpenAI # 用於與 OpenAI API 交互  import subprocess # 用於執行外部命令  # 產生Flask app物件 print (\_\_name\_\_) # 輸出當前模組的名稱  app = Flask(\_\_name\_\_) # 創建一個 Flask 應用程式的實例  # 設定一個祕鑰  app.config['SECRET\_KEY'] = 'itismysecretkey'  # 初入網頁、進入登入畫面、進入註冊畫面、進入忘記密碼畫面  @app.route('/')  def first():  return render\_template('index.html')  # 註冊動作  UPLOAD\_FOLDER = 'static/avatar' # 設定文件上傳目錄  ALLOWED\_EXTENSIONS = {'jpg', 'png', 'jpeg'} # 允許上傳的文件類型  app.config['UPLOAD\_FOLDER'] = UPLOAD\_FOLDER # 配置 Flask 的文件上傳目錄  def allowed\_file(filename):  # 檢查文件是否為允許的類型  return '.' in filename and filename.rsplit('.', 1)[1].lower() in ALLOWED\_EXTENSIONS  @app.route('/do\_register', methods=['POST'])  def do\_register():  # 處理註冊請求  try:  # 取得表單參數  title = request.form.get('title') # 取得用戶輸入的帳號  name = request.form.get('name') # 取得用戶輸入的名稱  password = request.form.get('password0') # 取得用戶輸入的密碼  avatar = request.files.get('avatar') # 取得上傳的頭像文件  # 驗證表單參數是否齊全  if not (title and name and password):  return render\_template('register.html', error\_message="所有字段都是必填的") # 若有任何欄位未填，返回錯誤信息  # 檢查電子郵件格式  email\_regex = r'^[a-zA-Z0-9.\_%+-]+@[a-zA-Z0-9.-]+\.[a-zA-Z]{2,}$'  try:  # 驗證帳號格式是否正確，且長度超過10  if not re.match(email\_regex, title) or len(title) <= 10:  return render\_template('register.html', error\_message="註冊失敗，帳號輸入格式有誤")  # 驗證帳號與使用者名稱是否相同  if title == name:  return render\_template('register.html', error\_message="註冊失敗，帳號與使用者名稱請不要使用同一個")  # 檢查和處理上傳的頭像文件  if avatar and allowed\_file(avatar.filename):  file\_extension = avatar.filename.rsplit('.', 1)[1].lower() # 獲取文件擴展名  filename = f"{title}.{file\_extension}" # 使用帳號作為文件名  avatar\_path = os.path.join(app.config['UPLOAD\_FOLDER'], filename) # 構建文件存儲路徑    # 確保上傳目錄存在  os.makedirs(os.path.dirname(avatar\_path), exist\_ok=True)    avatar.save(avatar\_path) # 保存文件到指定路徑  print(f"File saved as {filename}")  else:  filename = None  return render\_template('register.html', error\_message="註冊失敗，頭像上傳有誤") # 上傳頭像失敗  # 加密密碼  hash\_password = bcrypt.hashpw(password.encode('utf-8'), bcrypt.gensalt()).decode('utf-8')  # 將用戶資料插入資料庫  conn = db.get\_connection() # 獲取資料庫連接  cursor = conn.cursor()  cursor.execute(  "INSERT INTO user\_information (user\_id, name, password, avatar) VALUES (%s, %s, %s, %s)",  (title, name, hash\_password, filename) # 插入用戶資料  )  conn.commit() # 提交事務  conn.close() # 關閉連接  # 渲染註冊成功頁面  return render\_template('login.html', success\_message="註冊成功")  except ValueError:  return render\_template('register.html', error\_message="註冊失敗") # 處理驗證錯誤  except Exception as e:  logging.error(f"An error occurred: {e}") # 記錄錯誤信息  return render\_template('register.html', error\_message="註冊失敗") # 處理其他異常情況  # 變更密碼動作  @app.route('/change', methods=['POST'])  def change():  # 取得表單參數  title = request.form.get('title') # 取得用戶輸入的帳號  name = request.form.get('name') # 取得用戶輸入的名稱  password\_new = request.form.get('password1') # 取得用戶輸入的新密碼  password\_check = request.form.get('password2') # 取得用戶輸入的新密碼確認  # 使用正則表達式檢查 email 格式  email\_regex = r'^[a-zA-Z0-9.\_%+-]+@[a-zA-Z0-9.-]+\.[a-zA-Z]{2,}$'  if not re.match(email\_regex, title) or len(title) <= 10:  return render\_template('forget.html', error\_message="更新失敗，帳號輸入格式有誤") # 若帳號格式錯誤，返回錯誤信息  try:  # 取得資料庫連線  conn = db.get\_connection() # 獲取資料庫連接  cursor = conn.cursor()  # 查詢資料表以確認用戶存在  cursor.execute("SELECT \* FROM user\_information WHERE user\_id = %s AND name = %s", (title, name))  user = cursor.fetchone() # 獲取查詢結果  if user:  # 如果用戶存在，檢查新密碼和確認密碼是否一致  if password\_new == password\_check:  hash\_password = bcrypt.hashpw(password\_new.encode('utf-8'), bcrypt.gensalt()).decode('utf-8') # 加密新密碼  # 更新密碼  cursor.execute("UPDATE user\_information SET password=%s WHERE user\_id=%s", (hash\_password, title))  conn.commit() # 提交變更  return render\_template('login.html', success\_message="密碼已修改，請登入") # 渲染成功頁面  else:  return render\_template('forget.html', error\_message="密碼修改失敗，請確認新密碼是否輸入正確") # 密碼不一致，返回錯誤信息  else:  return render\_template('forget.html', error\_message="密碼修改失敗，查無此使用者") # 用戶不存在，返回錯誤信息  except Exception as e:  # 紀錄異常  print(f"An error occurred: {e}") # 輸出異常信息  return render\_template('forget.html', error\_message="密碼修改失敗，請稍後再試") # 返回錯誤信息  finally:  # 確保資料庫連線被關閉  conn.close() # 關閉資料庫連接  # 登入動作  @app.route('/home', methods=['POST'])  def home():  conn = None # 初始化資料庫連接變數  try:  # 取得參數  title = request.form.get('title') # 取得用戶輸入的帳號  password = request.form.get('password') # 取得用戶輸入的密碼  # 驗證電子郵件格式  email\_regex = r'^[a-zA-Z0-9.\_%+-]+@[a-zA-Z0-9.-]+\.[a-zA-Z]{2,}$'  if (not re.match(email\_regex, title)) or len(title) <= 10:  return render\_template('login.html', error\_message="登入失敗，帳號輸入格式有誤") # 帳號格式不正確，返回錯誤信息  # 取得資料庫連線  conn = db.get\_connection() # 獲取資料庫連接  cursor = conn.cursor()    # 查詢資料表以確認用戶存在  cursor.execute("SELECT \* FROM user\_information WHERE user\_id = %s", (title,))  user = cursor.fetchone() # 獲取查詢結果  # 檢查用戶是否存在且密碼是否正確  if user and bcrypt.checkpw(password.encode('utf-8'), user[2].encode('utf-8')):  user\_id = user[0] # 獲取用戶ID  username = user[1] # 獲取用戶名稱  avatar = "static/avatar/" + user[3] # 獲取用戶頭像路徑    # 儲存用戶信息到會話  session['user\_id'] = user\_id  session['username'] = username  session['lookavatar'] = avatar  # 渲染成功畫面，重定向到登入後的主頁  return redirect(url\_for('afterlogin\_home'))  else:  # 渲染登入失敗畫面  return render\_template('login.html', error\_message="登入失敗") # 登入失敗，返回錯誤信息  except Exception as e:  print(f"Error: {e}") # 輸出異常信息  # 渲染登入失敗畫面  return render\_template('login.html', error\_message="登入失敗") # 返回錯誤信息  finally:  if conn:  conn.close() # 確保資料庫連接被關閉  # 登入後畫面轉換  @app.route('/afterlogin\_home')  def afterlogin\_home():  # 從會話中獲取用戶信息  username = session['username'] # 取得用戶名稱  user\_id = session['user\_id'] # 取得用戶ID  lookavatar = session['lookavatar'] # 取得用戶頭像路徑    try:  # 取得資料庫連線  conn = db.get\_connection() # 獲取資料庫連接  cursor = conn.cursor()  # 批量查詢所有故事  cursor.execute("SELECT \* FROM story") # 查詢故事資料  stories = cursor.fetchall() # 獲取所有故事資料  # 取得所有kind資料並轉為字典以提高查找效率  cursor.execute("SELECT kind\_id, name FROM kind") # 查詢故事類型資料  kind\_dict = {row[0]: row[1] for row in cursor.fetchall()} # 轉換為字典  # 批量查詢所有使用者資料並轉為字典  cursor.execute("SELECT user\_id, name, avatar FROM user\_information") # 查詢用戶資料  user\_dict = {str(row[0]): (row[1], row[2]) for row in cursor.fetchall()} # 轉換為字典  # 批量查詢所有收藏資料  cursor.execute("SELECT story\_id, COUNT(\*) FROM collect GROUP BY story\_id") # 查詢收藏資料  collect\_dict = {str(row[0]): row[1] for row in cursor.fetchall()} # 轉換為字典  # 批量查詢所有追蹤資料  cursor.execute("SELECT follow\_user FROM follow WHERE user\_id = %s", (user\_id,)) # 查詢用戶的追蹤資料  follow\_dict = {str(row[0]): '已追蹤' for row in cursor.fetchall()} # 轉換為字典  except Exception as e:  # 在發生錯誤時，確保資料庫連線關閉  if conn:  conn.close() # 關閉資料庫連接  # 返回錯誤頁面或訊息  return jsonify({'success': False, 'error': str(e)}), 500 # 返回錯誤訊息  # 構建故事列表  stories\_list = [] # 初始化故事列表  for story in stories:  if story[4] != user\_id: # 過濾掉自己的故事  story\_kind = kind\_dict.get(story[2], "未知類型") # 從字典中查詢類型名稱  user = user\_dict.get(str(story[4]), ("未知作者", "default\_avatar.png")) # 從字典中查詢作者信息  story\_detail = story[5][:40] + '...' if len(story[5]) > 40 else story[5] # 縮短故事內容  num\_collects = collect\_dict.get(str(story[0]), 0) # 從字典中查詢收藏數量  follow\_state = follow\_dict.get(str(story[4]), '未追蹤') # 從字典中查詢追蹤狀態  if story[3] == True: # 確保故事狀態為有效  # 將故事資訊添加到列表  stories\_list.append({  "story\_id": story[0], # 故事ID  "title": story[1], # 故事標題  "detail": story\_detail, # 故事內容簡介  "image": "static/image/story\_cover/" + story[6], # 故事封面圖片路徑  "author\_image": "static/avatar/" + user[1], # 作者頭像路徑  "author\_name": user[0], # 作者名稱  "style": story\_kind, # 故事類型  "favorites": num\_collects, # 收藏數量  "follow": follow\_state # 追蹤狀態  })  conn.close() # 結束資料庫連線  # 渲染主頁模板，傳遞用戶資訊和故事列表  return render\_template('home.html', username=username, lookavatar=lookavatar, stories=stories\_list)  # 搜尋故事動作  @app.route('/searchstory', methods=['GET'])  def searchstory():  # 獲取搜尋故事的關鍵字  search\_story = request.args.get('search\_story') # 取得查詢參數  # 從會話中獲取用戶信息  user\_id = session['user\_id'] # 取得用戶ID  username = session['username'] # 取得用戶名稱  lookavatar = session['lookavatar'] # 取得用戶頭像路徑  stories\_list = [] # 初始化故事列表  # 找到與搜尋條件匹配的故事  conn = db.get\_connection() # 獲取資料庫連接  cursor = conn.cursor()    # 查詢與搜尋條件匹配的故事  cursor.execute("SELECT \* FROM story WHERE title LIKE %s", (f"%{search\_story}%",))  stories\_information = cursor.fetchall() # 獲取符合條件的故事資料  # 取得所有kind資料並轉為字典以提高查找效率  cursor.execute("SELECT kind\_id, name FROM kind") # 查詢故事類型資料  kind\_dict = {row[0]: row[1] for row in cursor.fetchall()} # 轉換為字典  # 批量查詢所有使用者資料並轉為字典  cursor.execute("SELECT user\_id, name, avatar FROM user\_information") # 查詢用戶資料  user\_dict = {str(row[0]): (row[1], row[2]) for row in cursor.fetchall()} # 轉換為字典  # 批量查詢所有收藏資料  cursor.execute("SELECT story\_id, COUNT(\*) FROM collect GROUP BY story\_id") # 查詢收藏資料  collect\_dict = {str(row[0]): row[1] for row in cursor.fetchall()} # 轉換為字典  # 批量查詢所有追蹤資料  cursor.execute("SELECT follow\_user FROM follow WHERE user\_id = %s", (user\_id,)) # 查詢用戶的追蹤資料  follow\_dict = {str(row[0]): '已追蹤' for row in cursor.fetchall()} # 轉換為字典  conn.close() # 結束資料庫連線  # 遍歷所有符合搜尋條件的故事  for story in stories\_information:  if story[4] != user\_id: # 過濾掉自己的故事  story\_kind = kind\_dict.get(story[2], "未知類型") # 從字典中查詢類型名稱  user = user\_dict.get(str(story[4]), ("未知作者", "default\_avatar.png")) # 從字典中查詢作者信息  story\_detail = story[5][:40] + '...' if len(story[5]) > 40 else story[5] # 縮短故事內容  num\_collects = collect\_dict.get(str(story[0]), 0) # 從字典中查詢收藏數量  follow\_state = follow\_dict.get(str(story[4]), '未追蹤') # 從字典中查詢追蹤狀態  if story[3] == True: # 確保故事狀態為有效  # 將故事資訊添加到列表  stories\_list.append({  "story\_id": story[0], # 故事ID  "title": story[1], # 故事標題  "detail": story\_detail, # 故事內容簡介  "image": "static/image/story\_cover/" + story[6], # 故事封面圖片路徑  "author\_image": "static/avatar/" + user[1], # 作者頭像路徑  "author\_name": user[0], # 作者名稱  "style": story\_kind, # 故事類型  "favorites": num\_collects, # 收藏數量  "follow": follow\_state # 追蹤狀態  })  # 渲染主頁模板，傳遞用戶資訊和故事列表  return render\_template('home.html', username=username, lookavatar=lookavatar, stories=stories\_list)  # 進入細節畫面  @app.route('/story\_details', methods=['GET'])  def story\_details():  # 從會話中獲取用戶信息  user\_id = session['user\_id'] # 取得用戶ID  username = session['username'] # 取得用戶名稱  story\_id = request.args.get('story\_id') # 獲取故事ID  lookavatar = session['lookavatar'] # 取得用戶頭像路徑  # 獲取資料庫連接  conn = db.get\_connection()  cursor = conn.cursor()  # 查詢故事詳情，包括故事、類型、作者等信息  query = "SELECT s.story\_id, s.title, s.story\_detail, s.story\_picture, s.user\_id AS author\_id, k.name AS kind\_name, p.picture AS page\_picture, s.state FROM story s JOIN kind k ON s.kind\_id = k.kind\_id LEFT JOIN pages p ON s.story\_id = p.story\_id WHERE s.story\_id = %s"  cursor.execute(query, (story\_id,))  story\_info = cursor.fetchall() # 獲取故事資訊  if not story\_info:  return "Story not found", 404 # 如果故事不存在，返回404錯誤  story\_info = story\_info[0] # 獲取第一條故事資料  # 獲取作者名稱  cursor.execute("SELECT name FROM user\_information WHERE user\_id = %s", (story\_info[4],))  author\_name = cursor.fetchone()[0]  # 獲取收藏數量  cursor.execute("SELECT COUNT(\*) FROM collect WHERE story\_id = %s", (story\_id,))  total\_collections = cursor.fetchone()[0]  # 確認追蹤狀態  follow\_state = '未追蹤'  cursor.execute("SELECT 1 FROM follow WHERE user\_id = %s AND follow\_user = %s", (user\_id, story\_info[4]))  if cursor.fetchone():  follow\_state = '已追蹤' # 如果已追蹤，則狀態改為已追蹤  # 獲取作者的其他故事數量  cursor.execute("SELECT COUNT(\*) FROM story WHERE user\_id = %s", (story\_info[4],))  total\_stories = cursor.fetchone()[0]  # 獲取追蹤此作者的用戶數量  cursor.execute("SELECT COUNT(\*) FROM follow WHERE follow\_user = %s", (story\_info[4],))  total\_follow\_users = cursor.fetchone()[0]  # 獲取該故事的所有頁面圖片  cursor.execute("SELECT picture FROM pages WHERE story\_id = %s ORDER BY page", (story\_id,))  page\_pictures = cursor.fetchall() # 獲取頁面圖片資料  # 將圖片路徑轉換為完整路徑  page\_pictures = ["static/image/page/" + pic[0] for pic in page\_pictures]  # 確認收藏狀態  collected\_state = '未收藏'  cursor.execute("SELECT 1 FROM collect WHERE user\_id = %s and story\_id = %s", (user\_id, story\_id,))  if cursor.fetchone():  collected\_state = '已收藏' # 如果已收藏，則狀態改為已收藏  # 確認故事的公開狀態  openstate = '未公開'  if story\_info[7] is True:  openstate = '已公開' # 如果故事已公開，則狀態改為已公開  # 構建故事詳細信息字典  detail\_list = [{  "openstate": openstate,  "story\_id": story\_id,  "title": story\_info[1],  "detail": story\_info[2],  "kind": story\_info[5],  "user": author\_name,  "follow": follow\_state,  "tot\_follow": total\_follow\_users,  "tot\_stories": total\_stories,  "collect": total\_collections,  "picture\_cover": "static/image/story\_cover/" + story\_info[3],  "pictures": page\_pictures,  "link": '/test2?story\_id=' + story\_id,  "collected": collected\_state  }]  # 渲染故事詳情頁面  return render\_template('story-details.html', username=username, lookavatar=lookavatar, details=detail\_list[0])  # 新增追蹤動作、取消追蹤動作  @app.route('/add\_follow', methods=['POST'])  def add\_follow():  user\_id = session['user\_id'] # 從會話中獲取當前用戶ID  data = request.get\_json() # 從請求中取得 JSON 數據  author\_name = data.get('author\_name') # 獲取要追蹤的作者名稱  try:  # 取得資料庫連線  conn = db.get\_connection()  cursor = conn.cursor()  # 查詢 user\_information 表以取得對應的 user\_id  cursor.execute("SELECT user\_id FROM user\_information WHERE name = %s", (author\_name,))  follow\_user\_id = cursor.fetchone() # 獲取作者的用戶ID  if follow\_user\_id is None:  return jsonify({'success': False, 'error': 'Author not found'}), 404 # 找不到作者，返回404錯誤  # 插入追蹤記錄，避免重複追蹤  cursor.execute(  "INSERT INTO follow (user\_id, follow\_user) "  "SELECT %s, %s WHERE NOT EXISTS (SELECT 1 FROM follow WHERE user\_id = %s AND follow\_user = %s)",  (user\_id, follow\_user\_id[0], user\_id, follow\_user\_id[0])  )  conn.commit() # 提交事務  return jsonify({'success': True}), 200 # 返回成功狀態碼  except Exception as e:  conn.rollback() # 發生錯誤時回滾事務  return jsonify({'success': False, 'error': str(e)}), 500 # 返回錯誤訊息  finally:  conn.close() # 確保無論是否發生錯誤，都會關閉資料庫連線  # 收藏故事動作、取消收藏故事動作  @app.route('/collect\_story', methods=['POST'])  def collect\_story():  data = request.get\_json() # 從請求中取得 JSON 數據  story\_id = data.get('story\_id') # 獲取故事ID  user\_id = session["user\_id"] # 從會話中獲取當前用戶的ID  # 取得資料庫連線  conn = db.get\_connection()  cursor = conn.cursor()  try:  # 將收藏記錄插入 collect 表  cursor.execute("INSERT INTO collect (user\_id, story\_id, collect\_time) VALUES (%s, %s, %s)",  (user\_id, str(story\_id), datetime.now()))  # 提交所有資料庫操作  conn.commit() # 提交事務  return jsonify(success=True) # 返回成功狀態  except Exception as e:  conn.rollback() # 如果發生錯誤，則回滾事務  return jsonify(success=False, error=str(e)) # 返回錯誤訊息  finally:  conn.close() # 確保無論是否發生錯誤，都會關閉資料庫連線  # 閱讀故事  @app.route('/test2')  def test2():  story\_id = request.args.get('story\_id') # 獲取 URL 中的故事ID  # 取得資料庫連線，使用連線池來提高效能  with db.get\_connection() as conn:  with conn.cursor() as cursor:  # 使用JOIN合併兩個查詢，減少多次查詢的開銷  cursor.execute(  "SELECT s.title, s.story\_picture, p.content, p.picture FROM story s JOIN pages p ON s.story\_id = p.story\_id WHERE s.story\_id = %s ORDER BY p.page",  (story\_id,)  )  # 抓取所有結果  results = cursor.fetchall()  if results:  # 從查詢結果中提取標題和封面  title = results[0][0] # 故事標題  cover = results[0][1] # 故事封面  # 拆分成配對列表，包含內容和頁面圖片  paired\_list = [(row[2], row[3]) for row in results]  story\_list = [{  "title": title,  "cover": cover,  "pages": paired\_list, # 這裡使用配對的列表  }]  else:  story\_list = [] # 如果沒有找到結果，返回空列表  return render\_template('test2.html', story\_list=story\_list) # 渲染模板並傳遞故事列表  # 歷史創作按鈕、搜尋故事動作  @app.route('/myStory')  def myStory():  username = session['username'] # 獲取當前用戶的名稱  user\_id = session['user\_id'] # 獲取當前用戶的ID  lookavatar = session['lookavatar'] # 獲取當前用戶的頭像  # 取得資料庫連線  conn = db.get\_connection()  cursor = conn.cursor()  # 查詢該使用者的所有故事  cursor.execute("SELECT \* FROM story WHERE user\_id = %s", (user\_id,))  stories = cursor.fetchall() # 獲取所有故事資料  conn.close() # 結束資料庫連線    stories\_list = []  for story in stories:  # 縮短故事內容  story\_detail = story[5][:40] + '...' if len(story[5]) > 40 else story[5]  stories\_list.append({  "image": "static/image/story\_cover/" + story[6], # 故事封面圖片路徑  "title": story[1], # 故事標題  "user": username, # 當前用戶名稱  "description": story\_detail, # 縮短的故事內容  "link": '/story\_details?story\_id=' + str(story[0]) # 連結到故事詳情頁面  })  htmltitle = '我的創作' # 頁面標題  return render\_template('myStory.html', username=username, lookavatar=lookavatar, htmltitle=htmltitle, stories=stories\_list)  # 新增公開動作、取消公開動作  @app.route('/add\_open', methods=['POST'])  def add\_open():  data = request.get\_json() # 從請求中取得 story\_ID 參數  story\_ID = data.get('story\_ID')  try:  # 取得資料庫連線  conn = db.get\_connection()  cursor = conn.cursor()  # 更新故事的狀態為公開  cursor.execute("UPDATE story SET state=%s WHERE story\_id=%s", ('true', story\_ID))  conn.commit() # 提交事務  return jsonify({'success': True}), 200 # 返回成功狀態碼  except Exception as e:  conn.rollback() # 發生錯誤時回滾事務  return jsonify({'success': False, 'error': str(e)}), 500 # 返回錯誤訊息  finally:  conn.close() # 確保無論是否發生錯誤，都會關閉資料庫連線  # 故事收藏按鈕、搜尋收藏故事動作  @app.route('/collection')  def collection():  username = session['username']  user\_id = session['user\_id']  lookavatar = session['lookavatar']  # 取得資料庫連線  conn = db.get\_connection()  cursor = conn.cursor()  # 合併查詢，獲取收藏的故事以及相關的故事資訊和用戶名稱  query = "SELECT c.story\_id, c.collect\_time, s.title, s.story\_detail, s.story\_picture, u.name AS user\_name FROM collect c JOIN story s ON c.story\_id = s.story\_id JOIN user\_information u ON s.user\_id = u.user\_id WHERE c.user\_id = %s"  cursor.execute(query, (user\_id,))  stories = cursor.fetchall()  stories\_list = []  for story in stories:  story\_id = str(story[0])  summary = story[3]  if len(summary) > 40:  summary = summary[:40] + '...'  stories\_list.append({  "image": "static/image/story\_cover/" + story[4],  "title": story[2],  "date": story[1],  "user": story[5],  "summary": summary,  "link": '/story\_details?story\_id=' + story\_id  })  return render\_template('collection.html', username=username, lookavatar=lookavatar, stories=stories\_list)  # 進入追蹤者畫面、搜尋追蹤者畫面  @app.route('/follow')  def follow():  user\_id = session['user\_id'] # 獲取當前使用者的ID  username = session['username'] # 獲取當前使用者的名稱  lookavatar = session['lookavatar'] # 獲取當前使用者的頭像  # 建立資料庫連接  conn = db.get\_connection()  cursor = conn.cursor()  # 查詢用戶的 follow\_user  cursor.execute("SELECT follow\_user FROM follow WHERE user\_id = %s", (user\_id,))  follow\_users = cursor.fetchall() # 獲取該用戶所追蹤的所有用戶ID    # 提取所有 follow\_user 的 user\_id  follow\_user\_ids = [user[0] for user in follow\_users] # 將查詢結果轉換為列表    # 查詢所有跟隨用戶的信息  if follow\_user\_ids:  format\_strings = ','.join(['%s'] \* len(follow\_user\_ids)) # 建立占位符串以便於SQL查詢  query = f"SELECT user\_id, name, avatar FROM user\_information WHERE user\_id IN ({format\_strings})"  cursor.execute(query, tuple(follow\_user\_ids)) # 執行查詢以獲取用戶信息  users\_information = cursor.fetchall() # 獲取所有相關用戶的信息  else:  users\_information = [] # 如果沒有跟隨用戶，設置為空列表  # 關閉資料庫連接  conn.close()  # 整理用戶信息  users\_info\_list = [{'name': user[1], 'avatar': user[2], 'link': '/user\_detail?follow\_user\_name=' + user[1]} for user in users\_information]    return render\_template('follow.html', username=username, lookavatar=lookavatar, users\_information=users\_info\_list) # 返回渲染的模板  # 追蹤者的歷史創作瀏覽、搜尋追蹤者的故事動作  @app.route('/user\_detail', methods=['GET'])  def user\_detail():  follow\_user\_name = request.args.get('follow\_user\_name') # 從請求中獲取要查看的用戶名  username = session.get('username', '未知用户') # 獲取會話中的用戶名，默認值為'未知用户'  lookavatar = session.get('lookavatar', '未知用戶') # 獲取會話中的用戶頭像，默認值為'未知用戶'  # 初始化 htmltitle 以確保它在函數的所有路徑中都被賦值  htmltitle = f"{follow\_user\_name}的創作" if follow\_user\_name else "用户的創作" # 設置頁面標題  search\_followname = follow\_user\_name # 儲存搜尋的用戶名  try:  # 取得資料庫連線  with db.get\_connection() as conn: # 使用連線池獲取資料庫連線  with conn.cursor() as cursor: # 創建游標以執行SQL查詢  # 查詢用戶ID  cursor.execute("SELECT user\_id FROM user\_information WHERE name = %s", (follow\_user\_name,)) # 查詢用戶信息表獲取ID  follow\_user\_id = cursor.fetchone() # 獲取查詢結果  if follow\_user\_id: # 如果找到用戶ID  follow\_user\_id = follow\_user\_id[0] # 提取ID值  # 查詢該用戶的所有故事  cursor.execute("SELECT \* FROM story WHERE user\_id = %s AND state = true", (follow\_user\_id,)) # 查詢該用戶的公開故事  stories = cursor.fetchall() # 獲取所有符合條件的故事  stories\_list = [] # 初始化故事列表  for story in stories: # 遍歷所有故事  # 縮短故事內容  story\_detail = story[5][:40] + '...' if len(story[5]) > 40 else story[5] # 若故事內容超過40字，縮短並添加省略號  stories\_list.append({  "image": "static/image/story\_cover/" + story[6], # 故事封面圖片路徑  "title": story[1], # 故事標題  "user": follow\_user\_name, # 用戶名  "description": story\_detail, # 縮短的故事描述  "link": '/story\_details?story\_id=' + str(story[0]) # 故事詳細頁面連結  })  else:  stories\_list = [] # 如果未找到用戶ID，返回空故事列表  except Exception as e:  print(f"发生错误: {e}") # 打印錯誤信息  stories\_list = [] # 出現錯誤時返回空故事列表  # 渲染模板，傳遞所需的參數  return render\_template('followStory.html', username=username, search\_followname=search\_followname, lookavatar=lookavatar, htmltitle=htmltitle, stories=stories\_list)  # 進入故事創作畫面  @app.route('/creation\_story')  def creation\_story():  # 獲取當前用戶的名稱和頭像，這些將用於渲染模板  username = session['username'] # 從會話中獲取用戶名稱  lookavatar = session['lookavatar'] # 從會話中獲取用戶頭像  # 渲染創建故事的模板，並將用戶名稱和頭像傳遞給模板  return render\_template('creation.html', username=username, lookavatar=lookavatar)  # openai\_api\_key  client=OpenAI(api\_key=os.environ.get("OPENAI\_API\_KEY"," OPENAI\_API\_KEY "))  # 隨機生成故事名稱、隨機生成故事大綱、建立故事並進入故事編輯頁  @app.route('/create\_name', methods=['POST'])  def create\_name():  # 獲取從請求中發送的 JSON 數據  data = request.get\_json()  kind = data.get('kind') # 獲取風格類型  question = f"請根據風格 '{kind}' 生成一個故事標題。" # 構建請求問題  # 創建一個新的線程並發送請求消息  thread = client.beta.threads.create() # 創建線程  client.beta.threads.messages.create(  thread\_id=thread.id,  role="user",  content=question, # 發送問題內容  )  # 執行線程運行  run = client.beta.threads.runs.create(  thread\_id=thread.id,  assistant\_id=' assistant\_id ', # 助理名稱：story\_name  )  # 高效等待運行完成的函數  def wait\_on\_run(run):  while run.status in ["queued", "in\_progress"]: # 當運行狀態為排隊或進行中時  time.sleep(1) # 休眠以避免達到 API 限流  run = client.beta.threads.runs.retrieve(  thread\_id=thread.id,  run\_id=run.id, # 獲取運行狀態  )  return run # 返回運行對象  run = wait\_on\_run(run) # 等待運行完成  # 獲取並處理消息  messages = client.beta.threads.messages.list(thread\_id=thread.id) # 獲取線程中的所有消息  response\_text = ""  for message in reversed(messages.data): # 反向遍歷消息列表  # 檢查消息是否來自助理  if message.role == "assistant":  response\_text += message.content[0].text.value # 獲取助理的回應內容  # 清理和格式化響應文本  response\_text = response\_text.strip() # 去除首尾空白  print(response\_text) # 輸出生成的故事標題到控制台  # 返回 JSON 格式的生成標題  return jsonify({'generated\_title': response\_text})  # 故事2次編輯  @app.route('/edit2\_story')  def edit2\_story():  story\_id = request.args.get('story\_id') # 獲取請求中的故事ID  lookavatar = session['lookavatar'] # 獲取用戶的頭像  # 取得資料庫連線  conn = db.get\_connection() # 獲取資料庫連接  cursor = conn.cursor() # 創建游標以執行SQL查詢  # 獲取故事細節  cursor.execute("SELECT \* FROM story WHERE story\_id = %s", (story\_id,)) # 查詢故事細節  story\_detail = cursor.fetchone() # 獲取查詢結果  # 獲取故事類型  cursor.execute("SELECT name FROM kind WHERE kind\_id = %s", (story\_detail[2],)) # 查詢故事類型  story\_kind = cursor.fetchone() # 獲取故事類型名稱  # 獲取故事內容  cursor.execute("SELECT \* FROM pages WHERE story\_id = %s order by page", (story\_id,)) # 查詢故事的所有頁面  story\_pages = cursor.fetchall() # 獲取所有故事頁面  storylist\_detail = [] # 初始化故事詳情列表  # 將故事詳細信息添加到列表中  storylist\_detail.append({  'story\_id': story\_id, # 故事ID  'title': story\_detail[1], # 故事標題  'kind': story\_kind[0], # 故事類型  'state': story\_detail[3], # 故事狀態  'story\_detail': story\_detail[5], # 故事大綱  'story\_cover': "static/image/story\_cover/" + story\_detail[6], # 故事封面圖片路徑  'story\_page1': story\_pages[0][2], # 第一頁的內容  'story\_picture1': "static/image/page/" + story\_pages[0][3], # 第一頁的圖片路徑  'story\_page2': story\_pages[1][2], # 第二頁的內容  'story\_picture2': "static/image/page/" + story\_pages[1][3], # 第二頁的圖片路徑  'story\_page3': story\_pages[2][2], # 第三頁的內容  'story\_picture3': "static/image/page/" + story\_pages[2][3], # 第三頁的圖片路徑  'story\_page4': story\_pages[3][2], # 第四頁的內容  'story\_picture4': "static/image/page/" + story\_pages[3][3], # 第四頁的圖片路徑  'story\_page5': story\_pages[4][2], # 第五頁的內容  'story\_picture5': "static/image/page/" + story\_pages[4][3], # 第五頁的圖片路徑  })  # 渲染模板，傳遞用戶的頭像和故事頁面信息  return render\_template('edit2.html', lookavatar=lookavatar, storypage=storylist\_detail[0])  # 生成圖片  save\_folder = 'static/image/story\_select' # 設定圖片儲存資料夾，若不存在則創建  os.makedirs(save\_folder, exist\_ok=True) # 創建資料夾，若已存在則不進行任何操作  # 設定日誌記錄，日誌級別設為 INFO  logging.basicConfig(level=logging.INFO)  def hash\_image\_content(content):  # 使用 SHA-256 對圖片內容進行哈希運算  sha256\_hash = hashlib.sha256() # 創建 SHA-256 哈希對象  sha256\_hash.update(content) # 更新哈希對象，加入圖片內容  return sha256\_hash.hexdigest() # 返回哈希值的十六進位字符串表示  def generate\_single\_image(prompt, i):  # 生成單張圖片並保存到指定資料夾  try:  # 生成圖片請求，使用 DALL-E 3 模型  response = client.images.generate(  prompt=prompt, # 輸入的提示語  model="dall-e-3", # 使用的模型名稱  n=1, # 生成一張圖片  size="1792x1024" # 設定圖片大小  )  image\_url = response.data[0].url # 獲取生成圖片的 URL  image\_response = requests.get(image\_url) # 下載圖片  if image\_response.status\_code == 200: # 檢查請求是否成功  content = image\_response.content # 獲取圖片內容  hashed\_name = hash\_image\_content(content) # 計算圖片內容的哈希值  image\_path = os.path.join(save\_folder, f"{hashed\_name}.png") # 生成儲存路徑  with open(image\_path, 'wb') as f: # 以二進位寫入方式打開文件  f.write(content) # 寫入圖片內容  logging.info(f"第{i+1}張圖片已保存至 {image\_path}") # 記錄圖片保存日誌  # 返回相對路徑  relative\_path = os.path.relpath(image\_path, start='static') # 生成相對於 'static' 的路徑  return relative\_path  else:  # 若下載失敗，記錄錯誤日誌  logging.error(f"下載第{i+1}張圖片失敗: {image\_response.status\_code}")  return None  except Exception as e:  # 處理異常，記錄錯誤信息  logging.error(f"生成第{i+1}張圖片時出現錯誤: {str(e)}", exc\_info=True)  return None  @app.route('/ai-image', methods=['POST'])  def generate\_images():  try:  data = request.get\_json() # 獲取請求中的 JSON 數據  # 確保請求包含消息  if not data or 'message' not in data:  return jsonify({"message": "未提供消息"}), 400 # 返回 400 錯誤碼  message = data.get('message') # 獲取消息內容  story\_kind = session.get('story\_kind', 'default') # 獲取故事類型，默認為 'default'  print(message) # 輸出消息到控制台  prompt = f"Create child-friendly {story\_kind} scene based on: {message}. No text, letters, or words, just a visual background for a children's storybook."  total\_images = 1 # 設定生成圖片的數量  image\_paths = [] # 用於儲存生成的圖片路徑  for i in range(total\_images): # 遍歷生成每張圖片  image\_path = generate\_single\_image(prompt, i) # 調用生成圖片的函數  if image\_path:  # 只返回圖片檔案名稱，不再加上靜態文件夾的前綴  image\_paths.append(f'static/image/story\_select/{os.path.basename(image\_path)}')  else:  return jsonify({"message": f"生成第{i+1}張圖片失敗"}), 500 # 返回 500 錯誤碼  return jsonify({"message": "success", "image\_paths": image\_paths}) # 返回成功和圖片路徑  except Exception as e:  # 捕捉異常並記錄錯誤  logging.error(f"處理請求時出現錯誤: {str(e)}")  return jsonify({"message": "內部錯誤"}), 500 # 返回 500 錯誤碼  # 儲存故事動作、更新故事動作  @app.route('/save-story', methods=['POST'])  def save\_story():  try:  lookavatar = session['lookavatar'] # 從會話中獲取用戶的頭像  # 取得參數  openstate = request.form.get('isPublic', 'false') # 獲取故事是否公開的狀態  message1 = request.form.get('message1') # 獲取故事的五個段落內容  image\_filename1 = request.form.get('image-filenames1')  message2 = request.form.get('message2')  image\_filename2 = request.form.get('image-filenames2')  message3 = request.form.get('message3')  image\_filename3 = request.form.get('image-filenames3')  message4 = request.form.get('message4')  image\_filename4 = request.form.get('image-filenames4')  message5 = request.form.get('message5')  image\_filename5 = request.form.get('image-filenames5')  image\_filename\_cover = request.form.get('image-filenames0') # 獲取封面圖片的檔名    # 檢查是否有任何值為空  if not all([message1, image\_filename1, message2, image\_filename2,  message3, image\_filename3, message4, image\_filename4,  message5, image\_filename5, image\_filename\_cover]):  # 如果有空值，則返回編輯頁面並顯示錯誤訊息  return render\_template('edit.html',  lookavatar=lookavatar,  error\_message="儲存失敗，請檢查所有故事內容是否填寫完整或圖片是否皆生成完成。",  story\_title=session.get('story\_title'),  story\_kind=session.get('story\_kind'),  story\_detail=session.get('story\_detail'),  array\_story=[['1', message1], ['2', message2], ['3', message3], ['4', message4], ['5', message5]])    # 取得資料庫連線  conn = db.get\_connection()  cursor = conn.cursor()  # 獲取 kind\_id  cursor.execute("SELECT kind\_id FROM kind WHERE name = %s", (session['story\_kind'],))  kind\_id = cursor.fetchone()[0] # 獲取故事類型的 ID    # 移動封面圖片  source = "static/image/story\_select" # 圖片來源資料夾  destination\_cover = "static/image/story\_cover" # 封面圖片目的資料夾  destination\_page = "static/image/page" # 故事內容圖片目的資料夾  os.makedirs(destination\_cover, exist\_ok=True) # 確保資料夾存在  os.makedirs(destination\_page, exist\_ok=True)  # 移動封面圖片到 story\_cover 資料夾  if image\_filename\_cover:  source\_path\_cover = os.path.join(source, image\_filename\_cover)  destination\_path\_cover = os.path.join(destination\_cover, image\_filename\_cover)  shutil.move(source\_path\_cover, destination\_path\_cover) # 移動檔案  # 移動其他圖片到 page 資料夾  files = {  '1': image\_filename1,  '2': image\_filename2,  '3': image\_filename3,  '4': image\_filename4,  '5': image\_filename5  }  for page, filename in files.items():  if filename: # 確保檔名不為空  source\_path = os.path.join(source, filename)  destination\_path = os.path.join(destination\_page, filename)  shutil.move(source\_path, destination\_path) # 移動檔案  # 插入故事資料到資料庫  cursor.execute("INSERT INTO story (title, kind\_id, state, user\_id, story\_detail, story\_picture) VALUES (%s, %s, %s, %s, %s, %s) RETURNING story\_id",  (session['story\_title'], kind\_id, openstate, session['user\_id'], session['story\_detail'], image\_filename\_cover))  story\_id = str(cursor.fetchone()[0]) # 獲取插入後的故事 ID  # 插入每頁的內容到資料庫  pages = [  (1, message1, image\_filename1),  (2, message2, image\_filename2),  (3, message3, image\_filename3),  (4, message4, image\_filename4),  (5, message5, image\_filename5)  ]  cursor.executemany("INSERT INTO pages (story\_id, page, content, picture) VALUES (%s, %s, %s, %s)",  [(int(story\_id), page, message, filename) for page, message, filename in pages]) # 批量插入每頁資料  # 提交所有資料庫操作  conn.commit() # 提交事務  conn.close() # 關閉資料庫連線  return render\_template('creation.html',  lookavatar=lookavatar,  success\_message="儲存成功，可到歷史創作瀏覽故事") # 返回成功頁面  except Exception as e:  print(f"Error: {e}") # 輸出錯誤訊息  return render\_template('edit.html',  lookavatar=session.get('lookavatar'),  error\_message="儲存失敗，請檢查所有故事內容是否填寫完整或圖片是否皆生成完成。",  array\_story=[['1', message1], ['2', message2], ['3', message3], ['4', message4], ['5', message5]])  # 進入使用者更新資訊畫面  @app.route('/user\_profile')  def user\_profile():  lookavatar = session['lookavatar'] # 從會話中獲取用戶的頭像  return render\_template('user-profile.html', lookavatar=lookavatar) # 渲染用戶資料頁面，並傳遞頭像資料  # 更新使用者資訊動作  @app.route('/update\_userinformation', methods=['POST'])  def update\_userinformation():  try:  # 取得參數  name = request.form.get('name') # 從表單中獲取用戶名  avatar = request.files.get('avatar') # 從上傳的文件中獲取頭像  user\_id = session.get('user\_id') # 從會話中獲取用戶 ID  # 檢查用戶是否登入  if not user\_id:  return render\_template('user-profile.html', lookavatar=session.get('lookavatar'), error\_message="用戶未登入")  # 取得資料庫連線  conn = db.get\_connection()  cursor = conn.cursor()  # 取得用戶資訊  cursor.execute("SELECT \* FROM user\_information WHERE user\_id = %s", (user\_id,))  user = cursor.fetchone() # 獲取用戶資料  # 檢查用戶是否存在  if not user:  conn.close()  return render\_template('user-profile.html', lookavatar=session.get('lookavatar'), error\_message="用戶不存在")  user\_name = user[1] # 原用戶名稱  user\_avatar = user[3] # 原用戶頭像  # 更新用戶名  if name:  cursor.execute("SELECT \* FROM user\_information WHERE name = %s", (name,)) # 檢查新名稱是否已存在  existing\_user = cursor.fetchone()  if existing\_user and existing\_user[0] != user\_id: # 如果名稱已存在且不是當前用戶  conn.close()  return render\_template('user-profile.html', lookavatar=session.get('lookavatar'), error\_message="更新失敗，暱稱可能已被使用")  user\_name = name # 更新用戶名  # 更新頭像  if avatar and allowed\_file(avatar.filename): # 確保頭像文件格式合法  file\_extension = avatar.filename.rsplit('.', 1)[1].lower() # 獲取文件擴展名  filename = f"{user\_id}.{file\_extension}" # 使用用戶 ID 作為文件名  avatar\_path = os.path.join(app.config['UPLOAD\_FOLDER'], filename) # 設置頭像保存路徑  avatar.save(avatar\_path) # 保存頭像  user\_avatar = filename # 更新用戶頭像名稱  # 更新資料庫中的用戶資訊  cursor.execute("UPDATE user\_information SET name=%s, avatar=%s WHERE user\_id=%s", (user\_name, user\_avatar, user\_id))  conn.commit() # 提交變更  conn.close() # 關閉連接  avatar = "static/avatar/" + user\_avatar # 設定新的頭像路徑  session['username'] = user\_name # 更新會話中的用戶名  session['lookavatar'] = avatar # 更新會話中的頭像路徑  return render\_template('user-profile.html', lookavatar=avatar, success\_message="更新成功")  except Exception as e:  print(f"An error occurred: {e}") # 打印錯誤訊息  return render\_template('user-profile.html', lookavatar=session.get('lookavatar'), error\_message="更新失敗")  # 登出  @app.route('/logout')  def logout():  session.clear() # 清除會話中的所有資料  return render\_template('login.html') # 返回登錄頁面  # 執行app if \_\_name\_\_ == '\_\_main\_\_':  # 先執行 node server.js  subprocess.Popen(['node', 'server.js'])  # 然後啟動 Flask 應用  app.run('140.131.114.151', port=1024, debug=True) | |

▼表9-2-1 其他附屬程式清單

|  |  |  |
| --- | --- | --- |
| 編號 | 檔案名稱 | 功能名稱 |
| 1 | static/css/style.css | 基本頁面設計 |
| 2 | static/css/demo2.css | 繪本瀏覽頁面設計 |
| 3 | utils/db.py | 資料庫連線 |
| 4 | gold-hold.json | Google text-to-speech憑證 |
| 5 | server.ls | TTS 伺服器 |
| 6 | web.config | FastCGI 設定 |
| 7 | wfastcgi.py | WSGI 處理器 |

**第十章 測試模型**

10-1測試計畫

1. 使用者註冊&登入
   * + 註冊&登入：使用者能夠成功註冊並登入使用網頁
2. 新增及編輯故事
   * + 新增創作：使用者能夠成功新增創作
     + 編輯繪本：使用者能夠成功編輯歷史創作
3. 搜尋功能
   * + 搜尋使用者：使用者能夠成功搜尋其他使用者並查看其創作的故事
     + 搜尋繪本：使用者能夠成功查看其他使用者公開的創作
4. 閱讀繪本
   * + 閱讀繪本：故事內容文字及圖片能夠正常顯示，語音亦能正常撥放
5. 收藏繪本
   * + 收藏繪本：使用者能夠成功收藏其他使用者創作的故事
6. 追蹤創作者
   * + 追蹤創作者：追蹤其他使用者, 並可以查看他們創作的最新故事
7. 設定個人資料
   * + 資料設定：使用者能夠成功修改自己的暱稱及頭像

10-2測試個案與測試結果資料

▼表10-2-1 使用者註冊&登入測試個案與測試結果資料

|  |  |
| --- | --- |
| 測試編號 | 1 |
| 功能名稱 | 註冊&登入 |
| 測試目的 | 使用者能夠成功註冊並登入使用網頁 |
| 測試流程 | 填寫註冊資料完成註冊後使用帳號、密碼登入 |
| 預期結果 | 註冊、登入成功 |
| 測試成果 | 正常 |

▼表10-2-2 新增創作測試個案與測試結果資料

|  |  |
| --- | --- |
| 測試編號 | 2 |
| 功能名稱 | 新增創作 |
| 測試目的 | 使用者能夠成功新增創作 |
| 測試流程 | 選擇繪本風格後經由AI輔助生成繪本大綱、內文及圖片 |
| 預期結果 | 繪本創作成功 |
| 測試成果 | 正常 |

▼表10-2-3 編輯繪本測試個案與測試結果資料

|  |  |
| --- | --- |
| 測試編號 | 3 |
| 功能名稱 | 編輯繪本 |
| 測試目的 | 使用者能夠成功編輯歷史創作 |
| 測試流程 | 對先前創作完成的繪本再次進行編輯 |
| 預期結果 | 編輯成功 |
| 測試成果 | 正常 |

▼表10-2-4 搜尋使用者測試個案與測試結果資料

|  |  |
| --- | --- |
| 測試編號 | 4 |
| 功能名稱 | 搜尋使用者 |
| 測試目的 | 使用者能夠成功搜尋其他使用者並查看其創作的故事 |
| 測試流程 | 在搜尋框中輸入使用者名稱並搜尋 |
| 預期結果 | 搜尋成功 |
| 測試成果 | 正常 |

▼表10-2-5 搜尋繪本測試個案與測試結果資料

|  |  |
| --- | --- |
| 測試編號 | 5 |
| 功能名稱 | 搜尋繪本 |
| 測試目的 | 使用者能夠成功查看其他使用者公開的創作 |
| 測試流程 | 在繪本搜尋框中輸入繪本名稱並搜尋 |
| 預期結果 | 搜尋成功 |
| 測試成果 | 正常 |

▼表10-2-6 閱讀繪本測試個案與測試結果資料

|  |  |
| --- | --- |
| 測試編號 | 6 |
| 功能名稱 | 閱讀繪本 |
| 測試目的 | 故事內容文字及圖片能夠正常顯示，語音亦能正常撥放 |
| 測試流程 | 選擇繪本後開啟閱讀介面 |
| 預期結果 | 顯示、撥放成功 |
| 測試成果 | 正常 |

▼表10-2-7 收藏繪本測試個案與測試結果資料

|  |  |
| --- | --- |
| 測試編號 | 7 |
| 功能名稱 | 收藏繪本 |
| 測試目的 | 使用者能夠成功收藏其他使用者創作的故事 |
| 測試流程 | 選擇一繪本並收藏 |
| 預期結果 | 收藏成功 |
| 測試成果 | 正常 |

▼表10-2-8 追蹤創作者測試個案與測試結果資料

|  |  |
| --- | --- |
| 測試編號 | 8 |
| 功能名稱 | 追蹤創作者 |
| 測試目的 | 追蹤其他使用者, 並可以查看他們創作的最新故事 |
| 測試流程 | 選擇一創作者並追蹤 |
| 預期結果 | 追蹤成功 |
| 測試成果 | 正常 |

▼表10-2-9 資料設定測試個案與測試結果資料

|  |  |
| --- | --- |
| 測試編號 | 9 |
| 功能名稱 | 資料設定 |
| 測試目的 | 使用者能夠成功修改自己的暱稱及頭像 |
| 測試流程 | 開啟設定頁更新個人資料 |
| 預期結果 | 更新成功 |
| 測試成果 | 正常 |

**第十一章 操作手冊**

系統之元件及其安裝及系統管理

▼表11-1-1 系統安裝元件資訊

|  |  |
| --- | --- |
| 系統安裝元件資訊 | |
| 元件名稱 | 趣書園 |
| 版本 | 1.0 |
| 支援語言 | 繁體中文 |
| 價格 | 免費 |
| 環境需求 | 網路連線 |

▼表11-1-2 系統安裝步驟

|  |
| --- |
| 系統安裝步驟 |
| 於瀏覽器輸入網址””即可使用 |

**第十二章 操作手冊**

各畫面、操作之移轉

▼表12-1 導覽頁面

|  |
| --- |
| 趣書園之導覽頁面 |
|  |

▼表12-2 登入頁面

|  |
| --- |
| 趣書園之登入頁面 |
|  |

▼表12-3 註冊頁面

|  |
| --- |
| 趣書園之帳號註冊頁面 |
|  |

▼表12-3 忘記密碼

|  |
| --- |
| 趣書園之忘記密碼頁面 |
|  |

▼表12-4 故事總覽

|  |
| --- |
| 趣書園之首頁頁面 |
|  |

▼表12-5 繪本創作-書名及大綱

|  |
| --- |
| 趣書園之創作頁面 |
|  |

▼表12-6 繪本創作-內文及圖片

|  |
| --- |
| 趣書園之編輯頁面 |
|  |

▼表12-7 歷史創作

|  |
| --- |
| 趣書園之歷史創作頁面 |
|  |

▼表12-8 繪本簡介

|  |
| --- |
| 趣書園之繪本簡介頁面 |
|  |

▼表12-9 閱讀介面

|  |
| --- |
| 趣書園之繪本閱讀頁面 |
|  |

▼表12-10 用戶歷史創作

|  |
| --- |
| 趣書園之被追蹤用戶之歷史創作頁面 |
|  |

▼表12-11 我的追蹤

|  |
| --- |
| 趣書園之追蹤瀏覽頁面 |
|  |

▼表12-12 我的收藏

|  |
| --- |
| 趣書園之收藏頁面 |
|  |

▼表12-13 資料設定

|  |
| --- |
| 趣書園之資料設定頁面 |
|  |

**第十三章 感想**

感想與建議

11336005 陳平

這是我在臺北商業大學參與的第二次專題，相較於上一次的慌亂與迷茫，有了經驗的加持，讓我在同時執行六年一貫計畫的壓力下依舊順利的完成這次專題，這除了要歸功於自己的堅持，更多的是同伴們的配合與努力，以及指導老師的用心。

我常常覺得未知的事物是會令人恐懼的，比如在專案中要去接觸新的技術，在真正動手開始研究之前我都會倍感焦慮，但往往在開始之後可以感受到獲取新知、掌握新技能的滿足感，利用裝備的新技能去完成一個作品，我認為這就是專題的意義之所在。

11336008 林嘉珮

這次的專題製作對我來說是一個既有趣又充滿挑戰的過程。從主題構想到最終成果呈現，每一個步驟都要求在美觀與實用性之間找到平衡，特別是在視覺設計和介面操作方面。我們的目標是讓使用者能輕鬆上手，同時激發創意和親子互動。在過程中，我們運用了過去學到的知識與技能，也投入了大量時間進行資料查找與新技術學習，以解決專題中的各種問題。

其中，使用 OpenAI 和 Google 語音技術是我收穫最多的部分。這些技術雖然有提供現成的開放程式碼，但要真正應用到我們的專題中，仍需要經過多次測試與調整，直到找到最符合需求的解決方案。這次專題不僅提升了我的技術掌握能力，也讓我更深入體會到，靈活運用現有資源和知識才能有效解決實際問題。

11336013 張彤

在這次專題中，我學到了幾個關鍵點。首先是自主學習的能力，網頁設計的過程中充滿了許多細節和技術挑戰，很多時候需要自己查閱資料、學習新知識。在這個過程中，我逐漸培養了善於利用網絡資源進行自主學習的習慣，提升了我的問題解決能力和獨立思考的能力。

其次，團隊溝通也顯得尤為重要。通過這次專題，我學會了在與老師或團隊成員討論問題前，先對現有問題進行整理和歸納，然後帶著明確的問題去討論。這種做法不僅能夠使會議更加高效，還能夠幫助大家快速聚焦在解決問題的核心部分，避免浪費時間。

這次專題不僅讓我提升了網頁設計的技術能力，還讓我更加體會到了自主學習和團隊合作的重要性，十分感謝同學以及老師此次專題的合作以及教導。

11236020 李佳祐

這次的專題比起五專時技能、心態都更加提升了，在專題中我們持續的面對到未知的問題、挫折，但比起五專時的煩躁、無奈，現在的我能夠靜下心來花時間專心找尋問題的關鍵點、查詢相關資料與使用工具來解決問題，專題期間，在組內的討論我們仍舊會遇到不少分岐，討論不順、小吵小鬧的時候，但小組內成員都非常的成熟，在專題時都整理好自我情緒，客觀的來面對問題並設法解決，而在時程的規劃所有組員都有遵循相關排程來進行，讓開發非常順利，最後也要特別感謝林宏仁老師，老師非常耐心的協助我們，不只教導我們技術、給予相關資源，專題會議時還能一針見血的點出關鍵問題，並給予意見。

**第十四章 參考資料**

**附錄**

審查評審意見之修正情形

▼表15 審查評審意見之修正情形

|  |  |
| --- | --- |
| 評審建議事項 | 修正情形及回覆 |
| 1. 閱讀器是閱讀創作生成的故事嗎？  閱讀器沒有情緒，關於父母親、老師的閱讀有什麼規劃 | 閱讀器是指可閱讀任何公開繪本的閱讀網頁，其功能包括：閱讀、播放音檔、模擬翻書效果  我們在原本的語音基礎上引入Goole的TTS，讓語音除了語速外更附帶抑揚頓挫。 |
| 2. 以廣告為收益，對於SEO方面有規劃嗎？ | SEO搜尋引擎最佳化的部分，我們在原本有些大眾化的名稱中改成較有辨識度的名稱，也預計在首頁的入口網站植入老師、家長查閱此類網站時會運用到的關鍵字。 |
| 3. 關於AIGC，是想要用什麼樣技術 呢？  Midjourney 或 Stable Diffusion 自己訓練，還是串接 AI 繪圖軟體或網站呢？ | AIGC我們使用了兩項，一是ChatGPT-Assistant做繪本的創作生成，二是ChatGPT-Dell來做繪本圖片的生成。  兩者都是藉由設定關鍵字做運作，並沒有直接的訓練餵資料…… |
| 4. 在 UX 方面，使用者是如何創作故事的，語音還是文字? 關係到語音辨識、操作複雜度..等面相 | UX方面，使用者只需要指定故事標題，選擇繪本的風格類型，最後再以文字輸入簡單的大綱就可以完成繪本，若沒有特別的想法也可以進行隨機生成。 |
| 5. 如何行銷給目標客戶？ | 行銷給目標客戶的方面除了上面說到的SEO，我們認為可能也可以以社群的形式在同類型的使用者中做擴散。 |
| 6. 內容，如何夠專業? |  |
| 7. AI工具越來越多、功能越來越強，如何有競爭力？ |  |
| 8. 一鍵生成的技術門檻 | 目前已完成。 |