**Software Requirements Specification**

**For**

**Course Forum**

**Version 1.0 approved**

Prepared by Kevin(鄭政文)、Ian(沈育安)、Dragon(丁襄龍)

Bernie(劉官瑜)、Cheng(許哲晟)、Jerry(吳俊傑)

2023-04-19

**目錄**

[1. 簡介 3](#_Toc132771757)

[1.1. 目的 3](#_Toc132771758)

[1.2. 文件規範 3](#_Toc132771759)

[1.3. 目標使用者與閱讀建議 3](#_Toc132771760)

[1.4. 產品範圍 3](#_Toc132771761)

[1.5. 參考 3](#_Toc132771762)

[2. 系統整體性描述 4](#_Toc132771763)

[2.1. 產品敘述 4](#_Toc132771764)

[2.2. 產品功能 4](#_Toc132771765)

[2.3. 使用者類型與客製化 4](#_Toc132771766)

[2.4. 操作環境 4](#_Toc132771767)

[2.5. 設計和實作限制 4](#_Toc132771768)

[2.6. 假設與依賴性 4](#_Toc132771769)

[3. 需求詳細說明 5](#_Toc132771770)

[3.1. 外部介面需求 5](#_Toc132771772)

[3.1.1. 使用者介面 5](#_Toc132771773)

[3.1.2. 軟體介面 7](#_Toc132771774)

[3.1.3. 溝通介面 9](#_Toc132771775)

[3.2. 功能性需求(Functional requirement) 10](#_Toc132771776)

[3.2.1. 學生 10](#_Toc132771777)

[3.2.2. 管理員 10](#_Toc132771778)

[3.2.3. 遊客 10](#_Toc132771779)

[3.3. 非功能性需求(Non-functional requirement) 10](#_Toc132771780)

[3.3.1. 可維護性、可用性、安全性、可靠性與效能 10](#_Toc132771781)

[3.3.2. 編碼風格 Coding Style 11](#_Toc132771782)

[3.3.3. 版本控制 Version Control 11](#_Toc132771783)

[3.3.4. 自動化測試與自動化部屬 CI/CD 11](#_Toc132771784)

[3.3.5. 測試 Testing 11](#_Toc132771785)

[3.3.6. 容器 Container 12](#_Toc132771786)

[3.4. 測試案例 12](#_Toc132771787)

[3.4.1. Client 測試 12](#_Toc132771788)

[3.4.2. Gateway 測試 13](#_Toc132771789)

[3.4.3. Auth Service 測試 13](#_Toc132771790)

[3.4.4. Forum Service 測試 14](#_Toc132771791)

[3.5. 效能需求 14](#_Toc132771792)

[3.6. 設計限制 14](#_Toc132771793)

[4. 軟體成本估計 15](#_Toc132771794)

[4.1. Complexity factor: Fi 15](#_Toc132771796)

[4.2. Function point 15](#_Toc132771797)

[4.3. Constructive Cost Model 16](#_Toc132771798)

[附錄1. 詞彙表 18](#_Toc132771799)

[附錄2. 分析模型 19](#_Toc132771800)

[2.1. Class Diagram 19](#_Toc132771801)

[2.1.1. Gateway 19](#_Toc132771802)

[2.1.2. Auth Service 20](#_Toc132771803)

[2.1.3. Forum Service 20](#_Toc132771804)

[2.2. Activity Diagram 21](#_Toc132771805)

[2.2.1. Client 21](#_Toc132771806)

[2.2.2. App 22](#_Toc132771807)

[2.2.3. Gateway 23](#_Toc132771808)

[2.2.4. Auth Service 23](#_Toc132771809)

[2.2.5. Forum Service 24](#_Toc132771810)

[附錄3. 待定清單 25](#_Toc132771811)

[附錄4. 組員分工 26](#_Toc132771812)

1. 簡**介**
   1. 目的

　　本文件詳細說明 「Course Forum」的軟體需求，包含產品概述、軟體功能、軟體內部架構、使用者介面、使用者流程。開發團隊需遵從本需求文件進行開發。

* 1. 文件規範

　　各項大標題文字以粗體18點字表示，各段大標題之副標題則以粗體14點字，副標內之細項以12點字，內文為11點字。文件中使用到本文件數與表中的詞彙則以字型以斜體表示。

* 1. 目標使用者與閱讀建議

工程師須遵照文件時做相關程式。請目標客戶參閱產品概序與產品功能，並確認使用者手冊之正確性。

* 1. 產品範圍

　　「Course Forum」可以讓使用者們在該論壇上分享對於課程的評價跟心得，幫助學生們在選課時可以做參考。

* 1. 參考

1. Golang <https://go.dev/>
2. Gin <https://gin-gonic.com/>
3. Gorm <https://gorm.io/>
4. Go-redis <https://redis.uptrace.dev/>
5. Swag <https://github.com/swaggo/swag>
6. Logrus <https://github.com/sirupsen/logrus>
7. Validator <https://github.com/go-playground/validator>
8. Python Flask <https://flask.palletsprojects.com/en/2.2.x/>
9. JSON Web Tokens <https://jwt.io/>
10. Vue3 <https://vuejs.org/>
11. Vuetifly <https://vuetifyjs.com/en/>
12. Flutter <https://docs.flutter.dev/>
13. 系統整體性描述
    1. 產品敘述

　　「Course Forum」是一個線上平台，旨在促進學生和教師之間的互動和討論。該論壇可用於課程內容討論、意見交流、社區互動等，以幫助學生更好地學習和理解課程內容，且這樣的平台可以讓學生們更深入地了解不同教授的教學風格，並且從其他學生的建議和經驗中獲得選課方面的建議，同時也為教師提供一個與學生交流的平台。

* 1. 產品功能

　　「Course Forum」可以讓使用者們在該論壇上分享對於課程的評價跟心得，幫助學生們在選課時可以做參考。

* 1. 使用者類型與客製化

　　本產品主要是設計給高科大學生作為使用。

* 1. 操作環境

使用者客戶端：一般可連網的windows電腦上，使用滑鼠和鍵盤進行操作，也可使用手機APP。

* 1. 設計和實作限制

電腦客戶端：請使用Chorme60以上的瀏覽器，並以滑鼠操作。其他瀏覽器或許可以執行，但不保證能正常運作。

手機客戶端：請使用本專案的APP去啟用，並以觸控螢幕操作、其他使用手機端瀏覽器不保證能正常運作。

* 1. 假設與依賴性

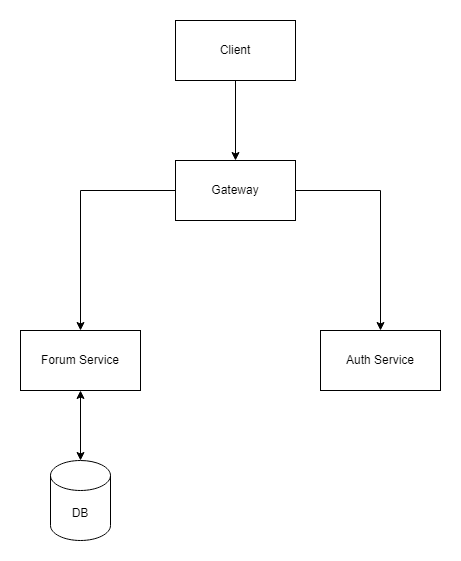
　　瀏覽器需支援JavaScript。

1. 需求詳細說明
2. 1. 外部介面需求
      1. 使用者介面

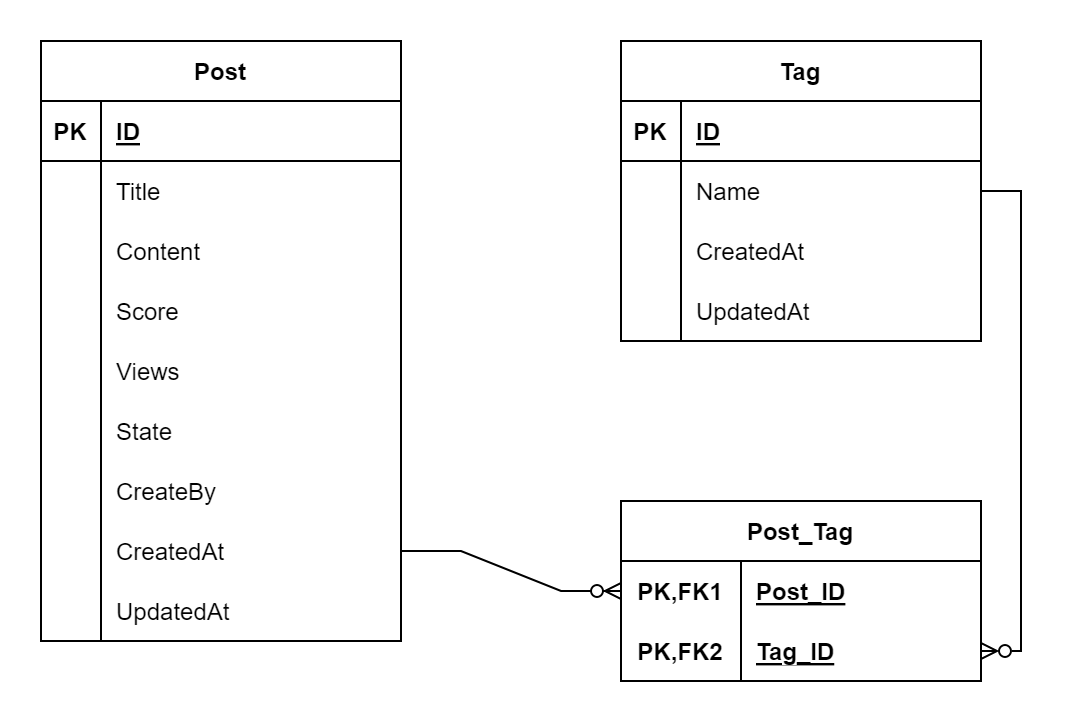
****

****

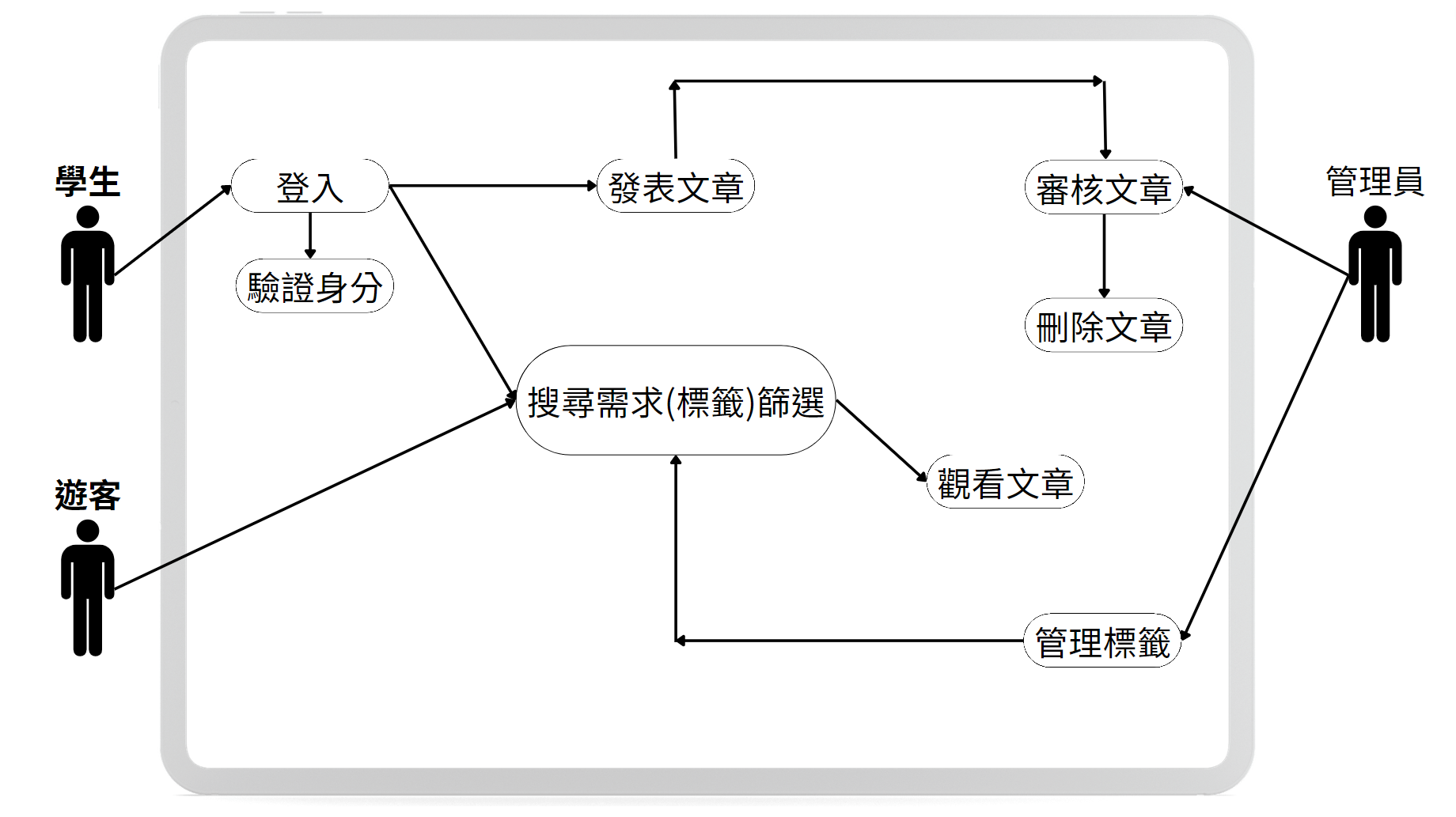
* + 1. 軟體介面
       1. 架構圖



* + - 1. ERD

****

* + - 1. Use case

****

* + - 1. Relational Schema

****

* + - 1. Client 系統

　　手機APP端，使用Flutter框架的Dart語言開發此APP，由於APP介面通常由多個Widget組成，而Flutter提供豐富的Widget庫，能夠快速開發高質量且美觀的APP。

網頁端，使用vue.js搭配vuetify進行架設，因為有vuetify框架提供的套件，讓前端的設計與開發變得簡單與快速，同時也不失質感。

1. 文章功能模組

2. 身分驗證模組

* + - 1. Gateway 系統

　　使用Laravel框架來實現Gateway系統。Gateway是一種介於客戶端和後端服務器之間的中間層，它負責管理API請求和響應，並提供一個統一的接口來處理多個後端服務器的請求。部分功能有：

1. 文章功能模組

2. 身分驗證模組

* + - 1. Auth Service 系統

　　使用Python撰寫、Flask框架來實現Auth Service系統。Flask提供了基本的Web應用程式功能，如路由、請求和響應處理、會話管理、模板引擎等，它將提供身分驗證的RESTful的API，對學校網站嘗試登入，並使用JWT身分驗證來記錄帳戶資訊。

1.身分驗證模組

* + - 1. Forum Service 系統  
         　　使用Golang撰寫、Gin框架來構建Forum Service系統。Forum Service擁有Redis快取，可以應付大流量、高併發的情況。並提供一個RESTful的API供操作文章功能。

1.文章功能模組

* + 1. 溝通介面

　　本系統微服務採用RESTful方式進行溝通。

* 1. 功能性需求(Functional requirement)
     1. 學生

|  |  |  |
| --- | --- | --- |
| **功能ID** | **功能名稱** | **功能描述** |
| R101 | 驗證身分 | 以學校系統帳密進行身分驗證及登入。 |
| R102 | 發表文章 | 使用者在登入後皆可發表文章，等待管理員審核後即可發表成功，且在文章標上星級評分與文章標籤。 |
| R103 | 搜尋需求(標籤)篩選 | 使用者可選擇自己有興趣的標籤，以在文章列表中篩選出符合標籤的文章。 |

* + 1. 管理員

|  |  |  |
| --- | --- | --- |
| **功能ID** | **功能名稱** | **功能描述** |
| R201 | 審核文章 | 可閱覽發表文章的申請，並決定是否要讓該文章發表成功。 |
| R202 | 刪除文章 | 可刪除任何一則已發表的文章。 |

* + 1. 遊客

|  |  |  |
| --- | --- | --- |
| **功能ID** | **功能名稱** | **功能描述** |
| R301 | 觀看文章 | 可閱覽文章列表以及點進文章觀看內容。 |

* 1. 非功能性需求(Non-functional requirement)
     1. 可維護性、可用性、安全性、可靠性與效能

1. 維護性

我們專案會要求以業界的方式來撰寫程式碼，以保持程式碼成品的維護性以及可讀性。

1. 可用性

該論壇平台具備良好的可用性，使學生能夠快速地找到所需要的資訊。這包括易於使用的界面、搜索功能、快速的響應速度等等。

1. 安全性

透過微服務分離各個功能，在使用RESTful進行溝通，伺服器並不保留有關使用者的任何資訊，一切都是無狀態的形式，並且Auth Service以及Forum Service有透過CORS來保證只有Gateway能夠呼叫API，以及透過Nginx進行反向代理，不會透漏出不必要資訊。

1. 可靠性

Forum Service 採用 golang 撰寫，在高併發的情況下，能夠維持服務。

1. 效能

Forum Service 採用Redis 快取文章資訊，並且使用golang 撰寫，已達成在高併發的狀況下能夠維持提供服務，並且資料庫採用PostgreSQL在高併發的情況下相較於MySQL有不錯的表現。

* + 1. 編碼風格 Coding Style

GO <https://google.github.io/styleguide/go/>

Python <https://peps.python.org/pep-0008/>

HTML <https://altcampus.school/posts/writing-semantically-correct-and-clean-html>

CSS <https://www.turing.com/kb/what-goes-behind-writing-a-flawless-css-code>

Javascript <https://blog.bitsrc.io/5-ways-to-write-clean-javascript-code-19aa6338fe00>

PHP <https://metabox.io/php-technique-write-clean-readable-code/>

Dart <https://dart.dev/guides/language/effective-dart/style>/

* + 1. 版本控制 Version Control

　　我們使用Git當作版本控制軟體，並且將程式碼都放到GitHub上。

* + 1. 自動化測試與自動化部屬 CI/CD

好處1：

維護軟體品質有接觸過程式開發的人都知道，若能及早發現程式碼的 bug，就能避免問題隨著時間越滾越大，形成難以修復的狀況。透過CI/CD工具，就像是建立一個標準化、自動化的工廠流水線，當每位開發人員完成一項任務後就交由工具檢驗程式碼是否正確且符合標準，確認無誤後再發布，便能大幅減少錯誤發生的機率，提升整體軟體的品質。

好處2：

減少各項時間成本由於發佈時間的間隔縮短，CI/CD工具一次需要驗證的內容較少，此時經由CI/CD工具便能夠迅速檢測錯誤的根源，以利開發人員盡快修復。由此可知，CI/CD工具不僅加速了每一次的檢驗、發布流程，更可以預防小錯誤演變成難以處理的複雜問題，節省日後大量修復的隱藏時間成本。

　　我們使用github action來處理我們的自動化測試跟部屬，在每次commit過後部屬環境也能自動部署，增加效率。

* + 1. 測試 Testing

　　在軟體測試中，自動化測試指的是使用獨立於待測軟體的其他軟體來自動執行測試、比較實際結果與預期並生成測試報告這一過程。在測試流程已經確定後，測試自動化可以自動執行的一些重複但必要測試工作。也可以完成手動測試幾乎不可能完成的測試。對於持續交付和持續集成的開發方式而言，測試自動化是至關重要的。我們要求每個人都要在程式底下撰寫自動化測試，有助於維護整體程式品質。

* + 1. 容器 Container

　　我們要求每個服務的專案都需要使用docker當作容器來建構環境，使用容器的優點是環境設定只需要做一次，寫成設定檔後，人人都可以直接執行docker自動化建置環境。

容器化有什麼好處？

1. 可攜性

　　軟體開發人員使用容器化在多個環境中部署應用程式，而無需重新撰寫程式碼。他們一次建置一個應用程式，並將其部署在多個作業系統上。例如，它們在Linux和Windows作業系統上執行相同的容器。開發人員也會使用容器進行部署，將舊版應用程式程式碼升級為新版本。

1. 可擴展性

　　容器是可有效執行的輕量型軟體元件。例如，虛擬機器可以更快地啟動容器化應用程式，因為它不需要啟動作業系統。因此，軟體開發人員可以輕鬆地在單一機器上為不同應用程式新增多個容器。容器叢集會使用來自相同共用作業系統的運算資源，但其中一個容器不會干擾其他容器的操作。

1. 容錯能力

　　軟體開發團隊使用容器建置容錯應用程式。他們使用多個容器在雲端上執行微服務。由於容器化微服務在隔離的使用者空間中運作，因此單個故障容器不會影響其他容器。這會增加應用程式的彈性和可用性。

1. 敏捷性

　　容器化應用程式可在隔離的運算環境中執行。軟體開發人員可以對應用程式碼進行故障排除和變更，而不會干擾作業系統、硬體或其他應用程式服務。它們可以縮短軟體發行週期，並使用容器模型快速進行更新。

* 1. 測試案例
     1. Client 測試

|  |  |  |  |
| --- | --- | --- | --- |
| **編號** | **名稱** | **輸入** | **預期輸出** |
| T101 | 點擊登入按鈕 | 模擬輸入學號及密碼，並點擊Login按鈕。 | 返回使用者資訊，並將資料傳至Gateway，如登入成功回傳HTTP 200 OK，失敗則回傳400。 |
| T102 | 點擊發文按鈕 | 模擬輸入文章及選擇標籤，並點擊Post按鈕。 | 返回文章內容及標籤，從Gateway拿取文章資料，並且成功回傳HTTP 200 OK，失敗回傳400。 |
| T103 | 點擊文章觀看 |  | 頁面跳轉至顯示文章內容頁面 |
| T104 | 點擊文章返回鍵 |  | 頁面跳轉至前一個頁面 |

* + 1. Gateway 測試

|  |  |  |  |
| --- | --- | --- | --- |
| **編號** | **名稱** | **輸入** | **預期輸出** |
| T201 | 獲取所有文章 | 呼叫Forum Service 使用GET api/posts | 從Forum Service拿取所有文章資料，並且返回 HTTP 200 OK |
| T202 | 獲取文章 | 呼叫Forum Service 使用GET api/posts/:id | 從Forum Service拿取文章資料，並且返回 HTTP 200 OK |
| T203 | 新增文章 | 呼叫Auth Service使用 POST /login 身分認證再呼叫Forum Service POST /posts | POST /login登入失敗回傳 400 Bad Request；成功登入回傳HTTP 200 OK |
| T204 | 更新文章 | 呼叫Auth Service使用 POST /login身分認證再呼叫Forum Service POST /posts:id | POST /login登入失敗回傳 400 Bad Request；成功登入回傳HTTP 200 OK |
| T205 | 刪除文章 | 呼叫Auth Service使用 POST /login身分認證再呼叫Forum Service POST /posts:id | POST /login登入失敗回傳 400 Bad Request；成功登入回傳HTTP 200 OK |
| T206 | 登入 | 呼叫Auth Service使用 POST /login身分認證 | 辨別身分為何者  1:學生  2:管理員 |

* + 1. Auth Service 測試

|  |  |  |  |
| --- | --- | --- | --- |
| **編號** | **名稱** | **輸入** | **預期輸出** |
| T301 | 登入 | POST /login | 透過爬蟲獲取的驗證碼，配合使用者傳輸的學校帳密進行登入，成功就回傳JWT，返回 HTTP 200 OK  登入失敗回傳 400 Bad Request |
| T302 | 驗證 JWT | POST /jwt-tokens/verify | 驗證JWT Token 成功，返回 HTTP 200 OK。失敗返回 400 Bad Request |
| T303 | 獲取驗證碼圖片 | 爬蟲抓取學校系統登入頁面驗證碼 | 爬到驗證碼圖片 |
| T304 | 解析驗證碼 | 透過ddddocr套件進行圖片辨識 | 正確的驗證碼 |

* + 1. Forum Service 測試

|  |  |  |  |
| --- | --- | --- | --- |
| **編號** | **名稱** | **輸入** | **預期輸出** |
| T401 | 獲取所有文章 | GET /posts | 從資料庫拿取所有文章資料，並且返回 HTTP 200 OK |
| T402 | 獲取所有文章(快取) | GET /posts | Redis hit!從記憶體拿取所有文章資料，並且返回 HTTP 200 OK |
| T403 | 獲取文章 | GET /posts/:id | 從資料庫拿取文章資料，並且返回 HTTP 200 OK |
| T404 | 獲取文章(快取) | GET /posts/:id | Redis hit!從記憶體拿取文章資料，並且返回 HTTP 200 OK |
| T405 | 新增文章 | POST /posts | 新增文章資料到資料庫，並且刪除快取，並且返回 HTTP 201 Created |
| T406 | 更新文章 | PATCH /posts/:id | 更新文章資料到資料庫，並且刪除快取，並且返回 HTTP 200 OK |
| T407 | 刪除文章 | DELETE /posts/:id | 從資料庫刪除文章，並且刪除快取，並且返回HTTP 204 No Content |

* 1. 效能需求

　　使用 JMeter 測試每秒鐘1萬次 API 呼叫，並且能夠穩定維持服務。

* 1. 設計限制

　　符合前述規範。

1. 軟體成本估計
2. 1. Complexity factor: Fi

|  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- |
| **Complexity factor: Fi** | **value=0** | **value=1** | **value=2** | **value=3** | **value=4** | **value=5** | **Fi** |
| Backup and recovery | 0 | 0 | 0 | 0 | 1 | 0 | 4 |
| Data communication | 0 | 0 | 0 | 1 | 0 | 0 | 3 |
| Distributed processing functions | 0 | 0 | 1 | 0 | 0 | 0 | 2 |
| Is performance critical? | 0 | 1 | 0 | 0 | 0 | 0 | 1 |
| Existing operating environment | 0 | 0 | 1 | 0 | 0 | 0 | 2 |
| on-line data entry | 0 | 0 | 0 | 0 | 1 | 0 | 4 |
| Input transaction built over multiple screens | 0 | 0 | 0 | 0 | 1 | 0 | 4 |
| Master files updated on-line | 0 | 0 | 1 | 0 | 0 | 0 | 2 |
| Complexity of inputs, outputs, files, inquiries | 0 | 1 | 0 | 0 | 0 | 0 | 1 |
| Complexity of processing | 0 | 0 | 0 | 0 | 1 | 0 | 4 |
| Code design for re-use | 0 | 0 | 0 | 0 | 0 | 1 | 5 |
| Are conversion/installation included in design? | 0 | 0 | 0 | 1 | 0 | 0 | 3 |
| Multiple installations | 0 | 0 | 0 | 0 | 0 | 1 | 5 |
| Application designed to facilitate change by the user | 0 | 0 | 0 | 0 | 0 | 1 | 5 |
|  | | | | | | Sigma(F) | 45 |
| Complexity adjustment factor | 0.65 + 0.01 \* Sigma（F）= | | | | | 1.1 | |

* 1. Function point

|  |  |  |  |
| --- | --- | --- | --- |
| **Measurement Parameter** | **Simple** | **Average** | **Complex** |
| External Inputs | 11\*3=33 | 9\*4=36 | 2\*6=12 |
| External Outputs | 20\*4=80 | 5\*5=25 | 0\*7=0 |
| External Inquiries | 2\*3=6 | 5\*4=20 | 1\*6=6 |
| Internal Logical Files | 6\*7=42 | 4\*10=40 | 6\*15=90 |
| External Logical Interfaces | 0\*5=0 | 4\*7=28 | 3\*10=30 |
| Unadjusted Function Points | 448 | | |
| Value Adjustment Factor | 1.1 | | |
| Adjusted Function Points | 492.8 | | |
| Loc | 10713 | | |

* 1. Constructive Cost Model

|  |  |  |
| --- | --- | --- |
| **Scale Factor Estimation（SFE）** | | |
| Process maturity | VERY LOW | 0.05 |
| Precedentedness | LOW | 0.04 |
| Flexibility | Norminal | 0.03 |
| Team interaction process | HIGH | 0.02 |
| Significant risks eliminated | VERY HIGH | 0.01 |
|  | Add | 1.01 |
| Total | 1.16 |

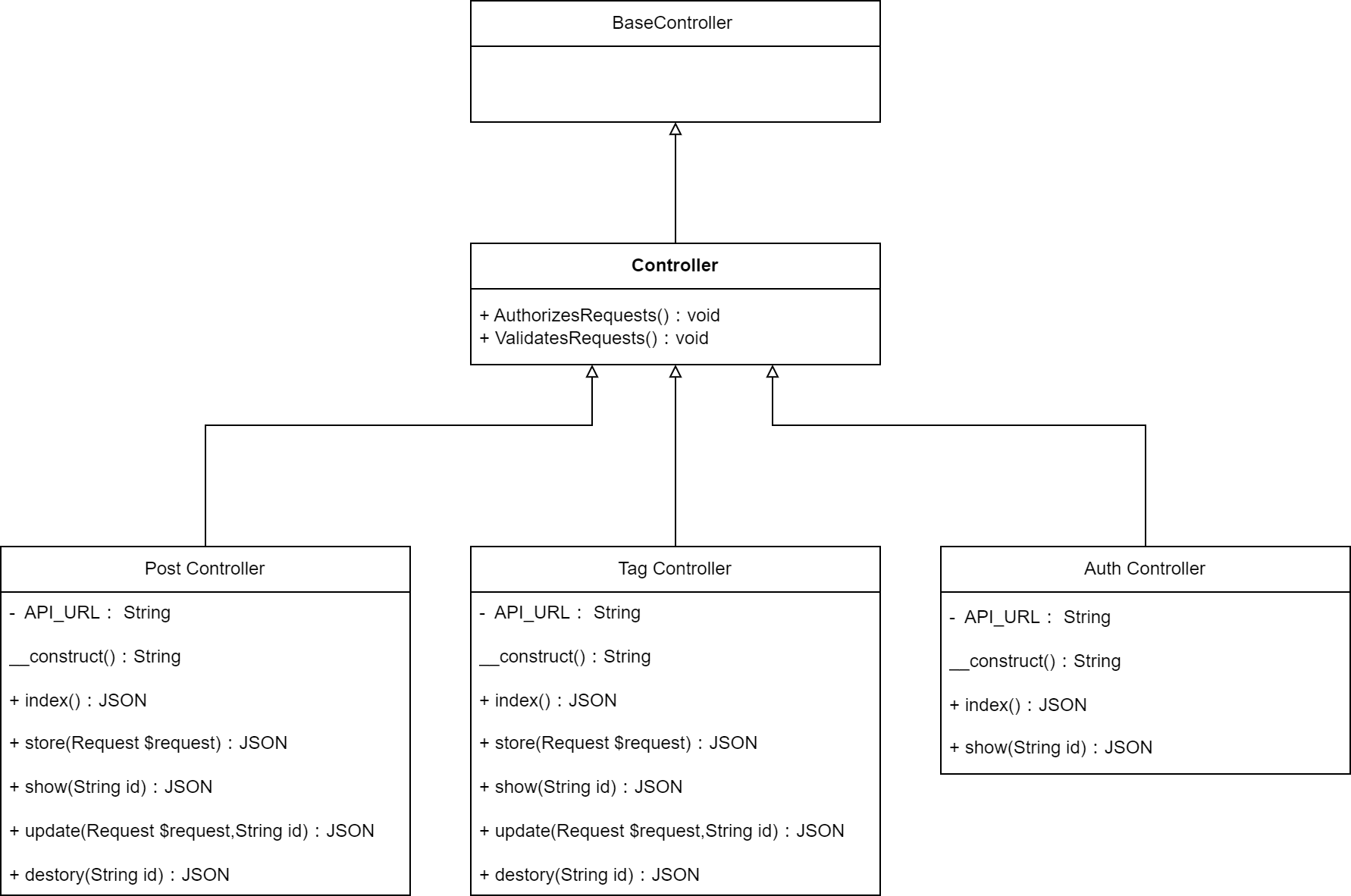
|  |  |
| --- | --- |
| **Effort Adjustment Factors (EAF)** | |
| **Product Attributes** | |
| Required Reliability | 1.40 (XH) |
| Database Size | 0.94 (L) |
| Product Complexity | 1.15 (H) |
| **Computer Attributes** | |
| Execution Time Constraint | 1.00 (VL) |
| Main Storage Constraint | 1.00 (VL) |
| Platform Volatility | 1.30 (XH) |
| Computer Turnaround Time | 0.87 (VL) |
| **Personnel Attributes** | |
| Analyst Capability | 0.71 (XH) |
| Applications Experience | 0.91 (H) |
| Programmer Capability | 0.86 (H) |
| Platform Experience | 1.10 (L) |
| Programming Language and Tool Experience | 0.95 (H) |
| **Project Attributes** | |
| Modern Programming Practices | 0.82 (XH) |
| Use of Software Tools | 0.91 (H) |
| Required Development Schedule | 1.00 (N) |
| **New (Values are probably wrong)** | |
| Required reusability | 1.05 (H) |
| Documentation match to life-cycle needs | 1.10 (H) |
| Personnel continuity | 1.00 (N) |
| Multisite development | 1.00 (N) |
| Product | 0.8566 |

**Effort=2.5⨯(10.713)1.16⨯0.8566=33.528  person-months**

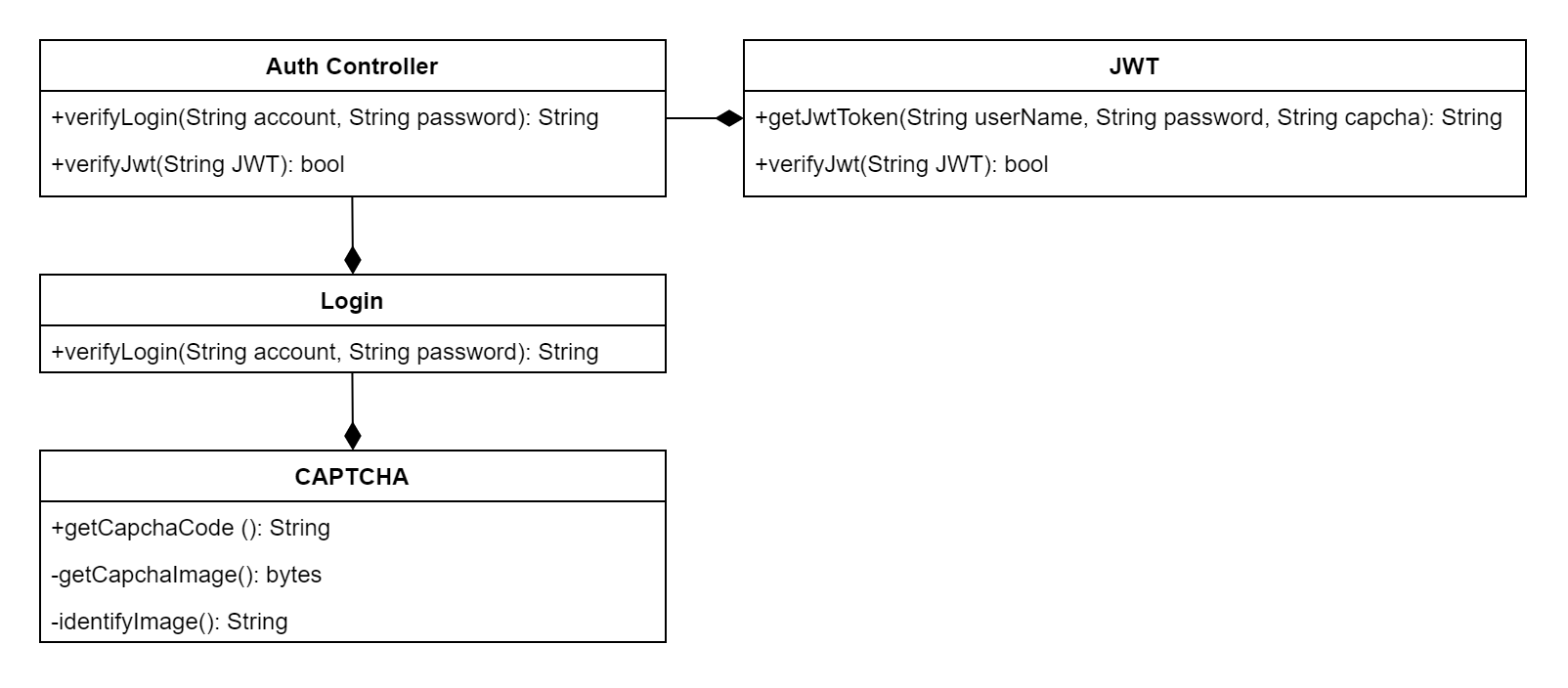
1. 詞彙表

* 高併發：在一個特定的時間，會有大量的請求湧入。
* 負載均衡（Load Balance）：把所有的請求分攤到各個子機器（服務）中。
* RESTful：一種軟體架構風格（並非標準），目的是幫助在世界各地不同軟體、程式在網際網路中能夠互相傳遞訊息。
* Redis：使用ANSI C編寫的開源、支援網路、基於記憶體、分散式、可選永續性的鍵值對儲存資料庫。
* 持續整合 CI（Continuous Integration）：顧名思義，就是當開發人員完成一個階段性的程式碼後就經由自動化工具測試、驗證，協助偵測程式碼問題，並建置出即將部署的版本（Build）。
* 持續部署 CD（Continuous Deployment）：可以說是CI的下一階段，經過CI測試後所構建的程式碼可以透過CD工具部署至伺服器，減少人工部署的時間。
* 版本控制（Version control）：是維護工程藍圖的標準做法，能追蹤工程藍圖從誕生一直到定案的過程。此外，版本控制也是一種軟體工程技巧，藉此能在軟體開發的過程中，確保由不同人所編輯的同一程式檔案都得到同步。
* JWT（JSON Web Token）：是一種基於 JSON 的開放標準(RFC 7519)，它定義了一種簡潔(compact)且自包含(self-contained)的方式，用於在雙方之間安全地將訊息作為 JSON 物件傳輸。而這個訊息是經過數位簽章(Digital Signature)，因此可以被驗證及信任。可以使用 密碼(經過 HMAC 演算法) 或用一對公鑰/私鑰(經過 RSA 或 ECDSA 演算法) 來對 JWT 進行簽章。
* JSON（JavaScript Object Notation）：是由道格拉斯·克羅克福特構想和設計的一種輕量級資料交換格式。其內容由屬性和值所組成，因此也有易於閱讀和處理的優勢。JSON是獨立於程式語言的資料格式，其不僅是JavaScript的子集，也採用了C語言家族的習慣用法，目前也有許多程式語言都能夠將其解析和字串化，其廣泛使用的程度也使其成為通用的資料格式。
* Nginx：免費的開源軟體，一個非同步框架的 web server，不過它的功用遠不僅止於 web server，它更多的用途是作為反向代理、Http Cache、負載平衡器。它在近幾年變得十分火熱，與較老牌的 Apache 成為 web server 的兩大巨頭，並且在較新的軟體專案中，使用率有超越 Apache 的趨勢。
* Docker：是一個開放原始碼的開放平臺軟體，用於開發應用、交付（shipping）應用和執行應用。Docker允許使用者將基礎設施（Infrastructure）中的應用單獨分割出來，形成更小的顆粒（容器），從而提高交付軟體的速度。Docker容器與虛擬機器類似，但二者在原理上不同。容器是將作業系統層虛擬化，虛擬機器則是虛擬化硬體，因此容器更具有可攜式性、更能高效地利用伺服器。 容器更多的用於表示軟體的一個標準化單元。由於容器的標準化，因此它可以無視基礎設施（Infrastructure）的差異，部署到任何一個地方。另外，Docker也為容器提供更強的業界的隔離相容。
* 測試（testing），描述一種用來促進鑑定軟體的正確性、完整性、安全性和品質的過程。依照可計算理論（計算機科學的一個支派）一個簡單的數學證明推斷出下列結果：不可能完全解決所謂「當機」，指任意電腦程式是否會進入無窮迴圈，或者罷工並產生輸出問題。換句話說，軟體測試是一種實際輸出與預期輸出間的稽核或者比較過程。

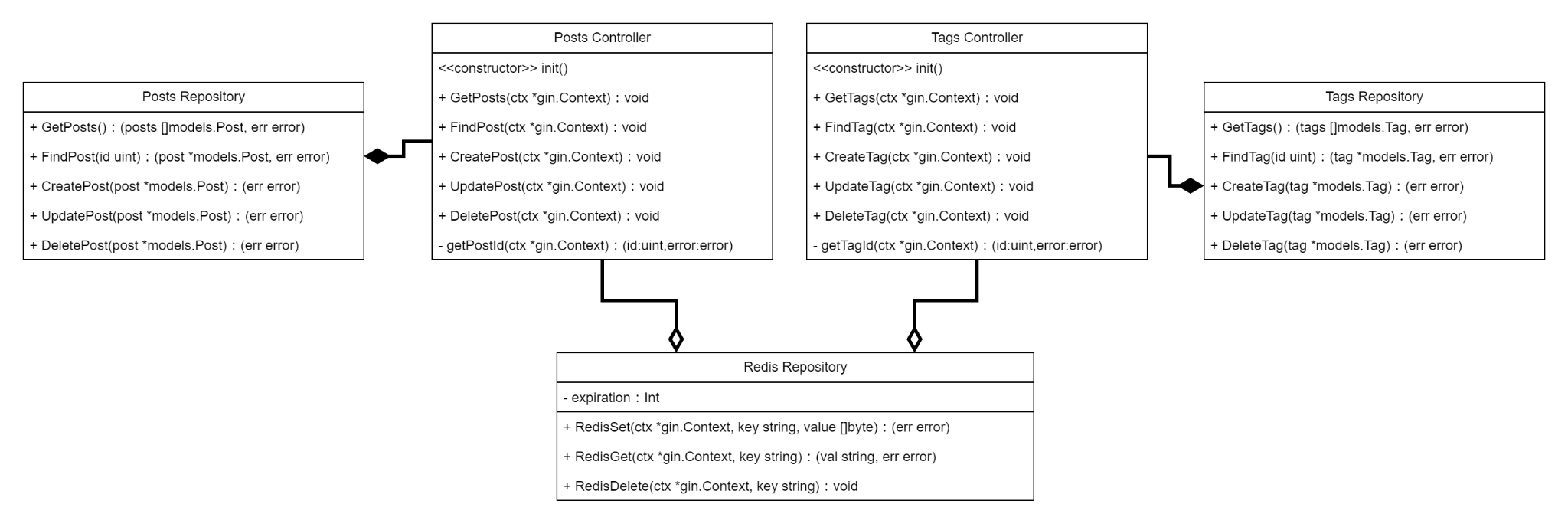
1. 分析模型
   1. Class Diagram
      1. Gateway

****

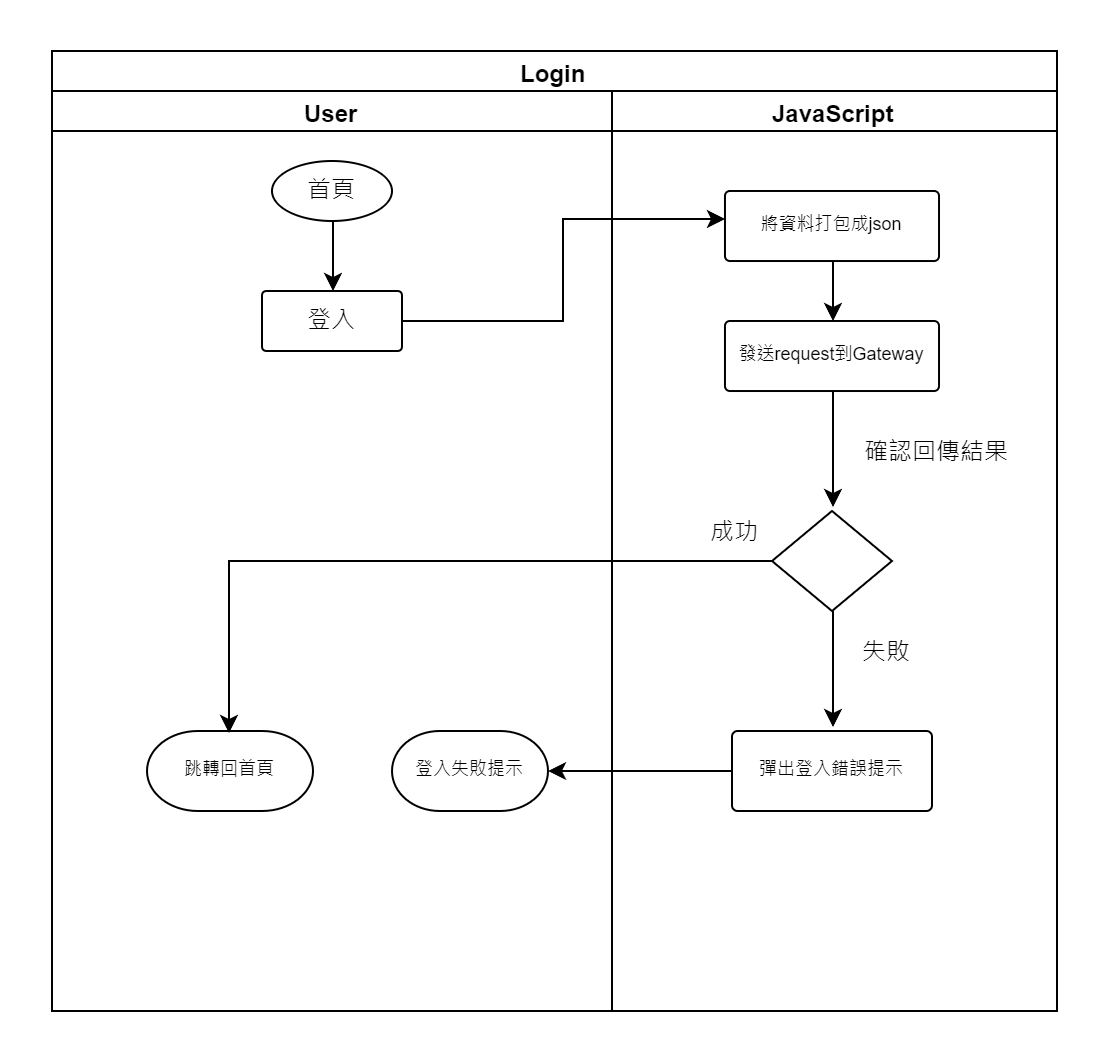
* + 1. Auth Service

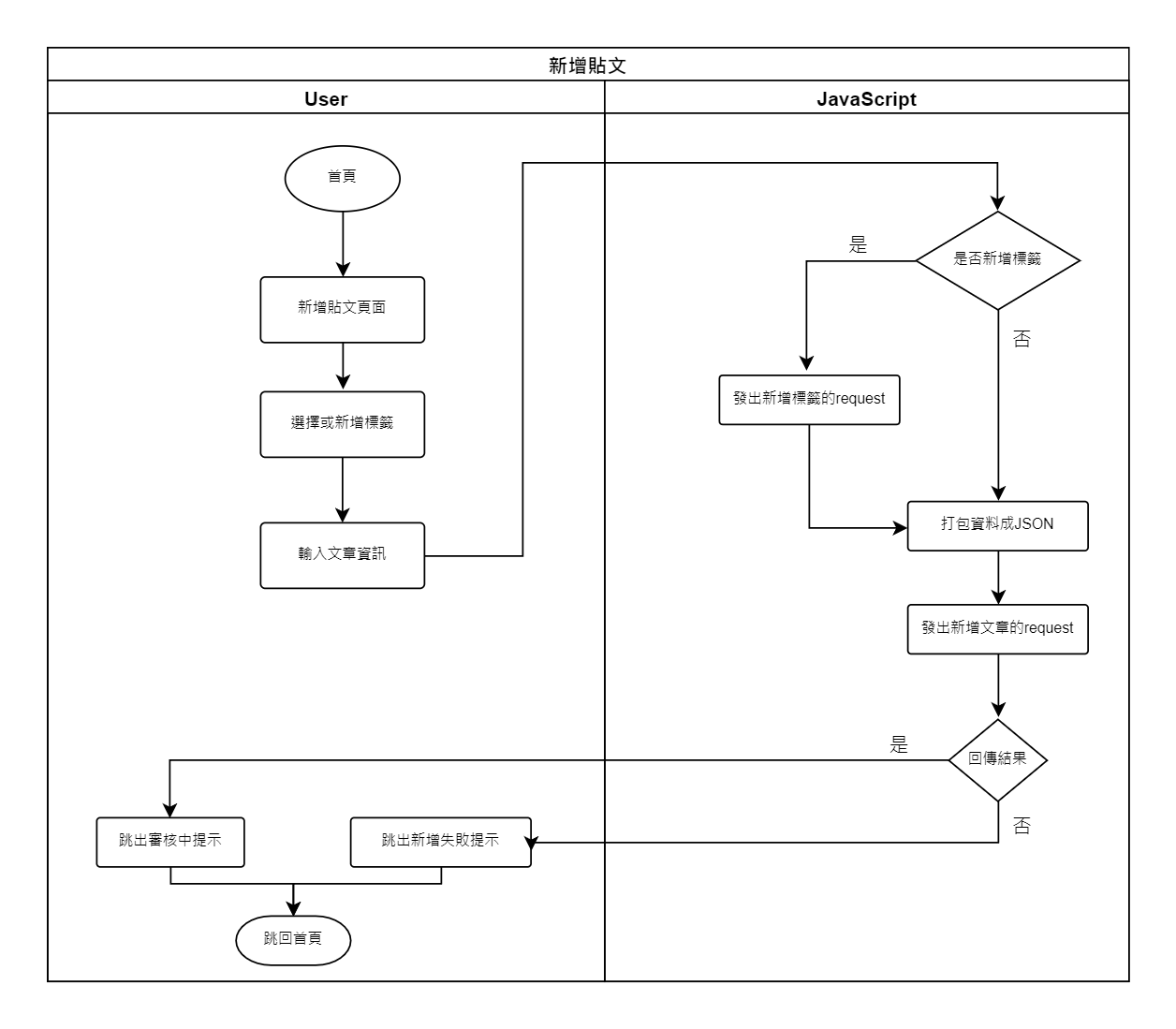
****

* + 1. Forum Service

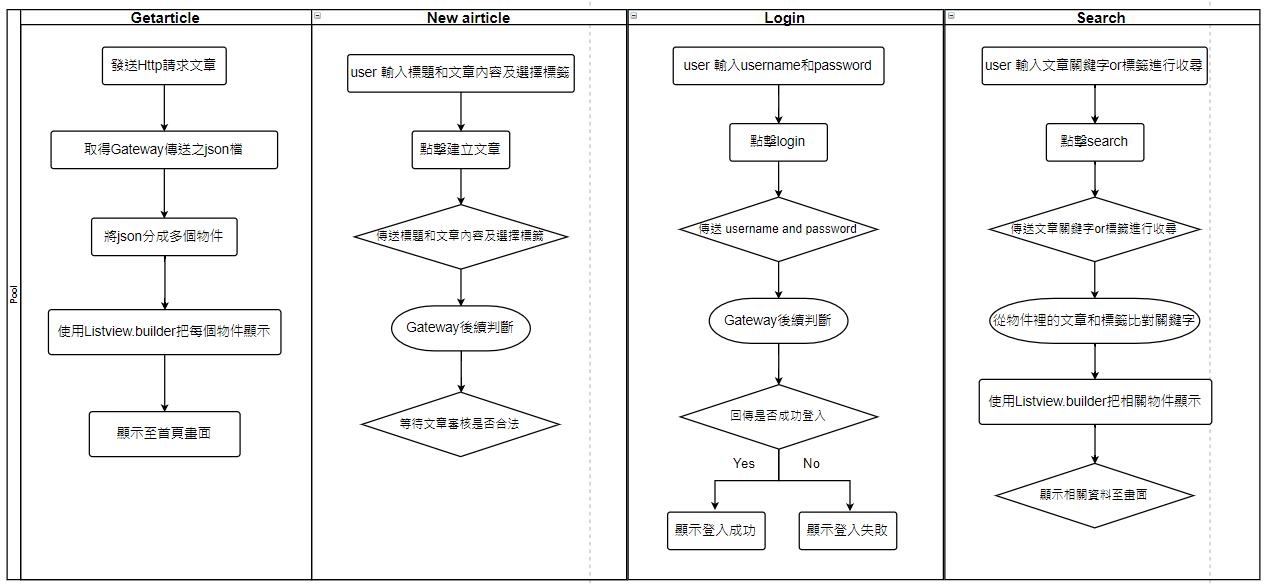
**

* 1. Activity Diagram
     1. Client

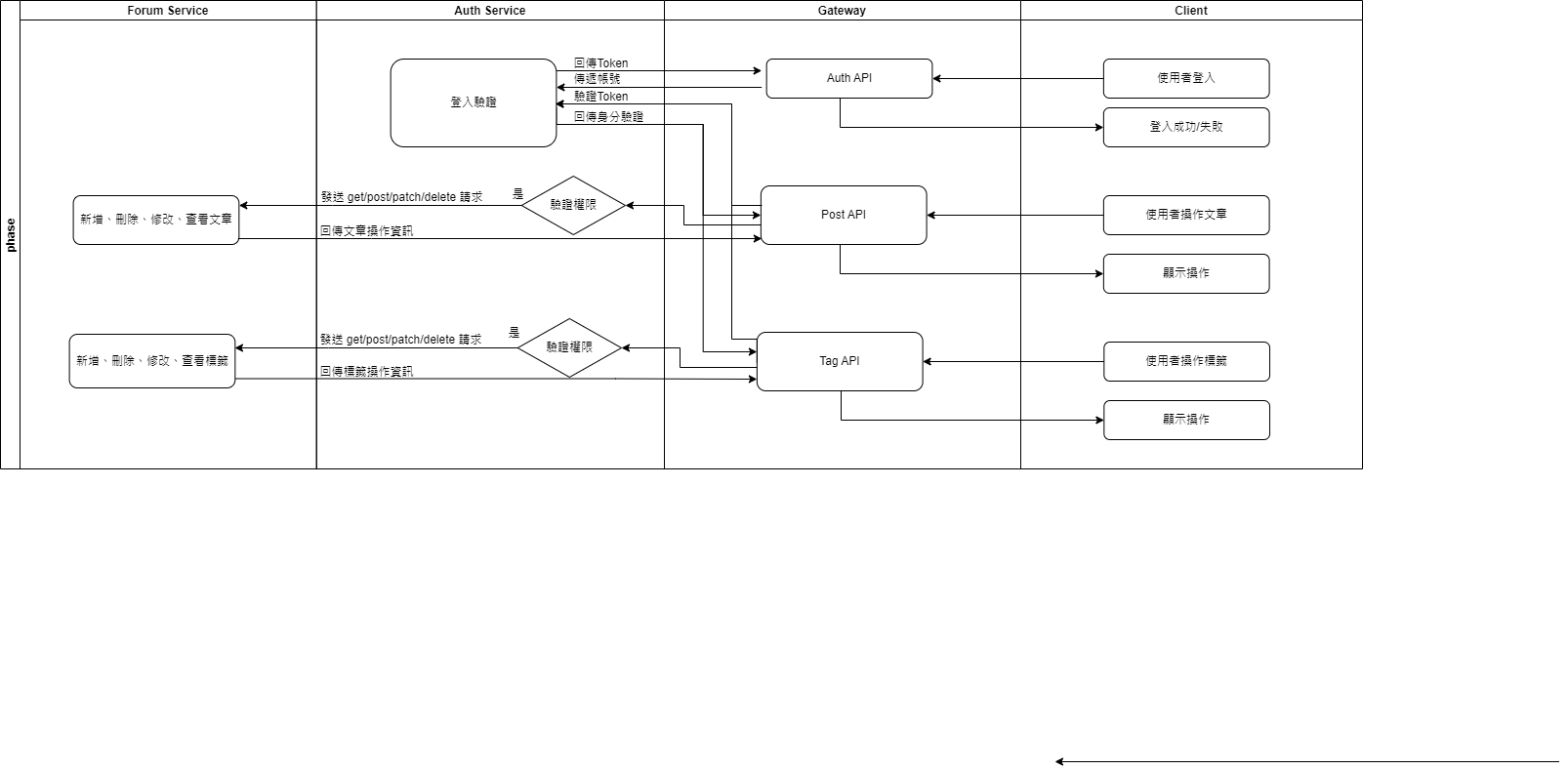
****

****

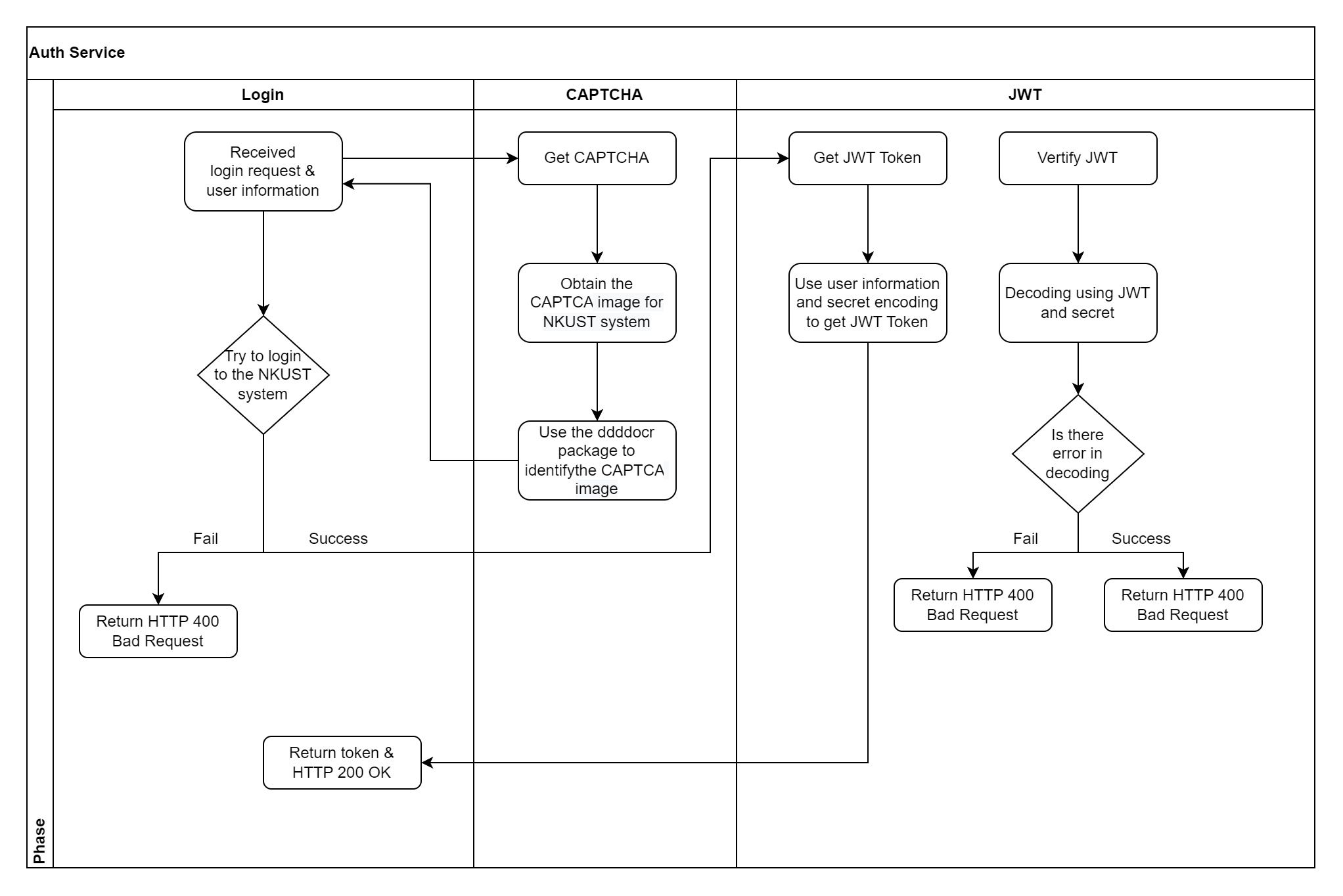
* + 1. App

****

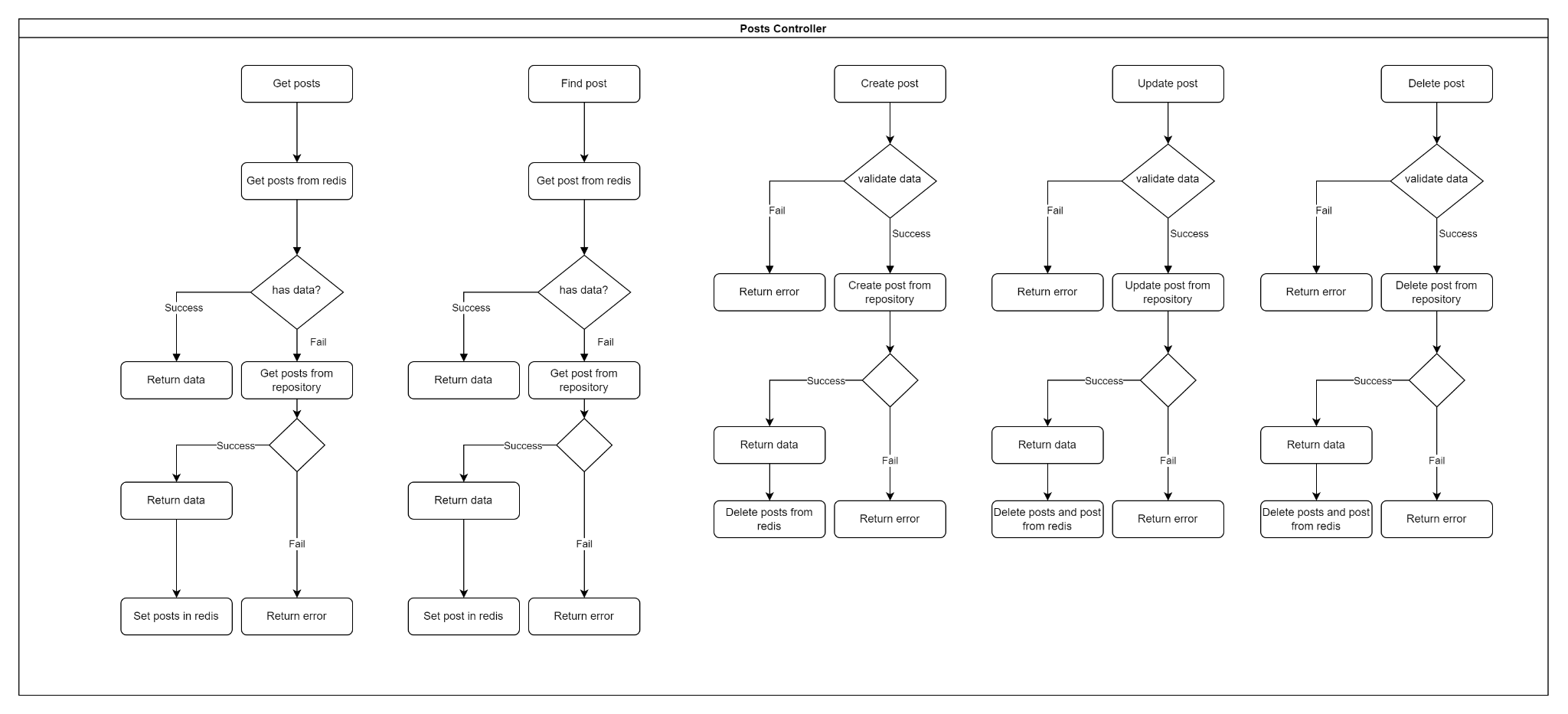
* + 1. Gateway

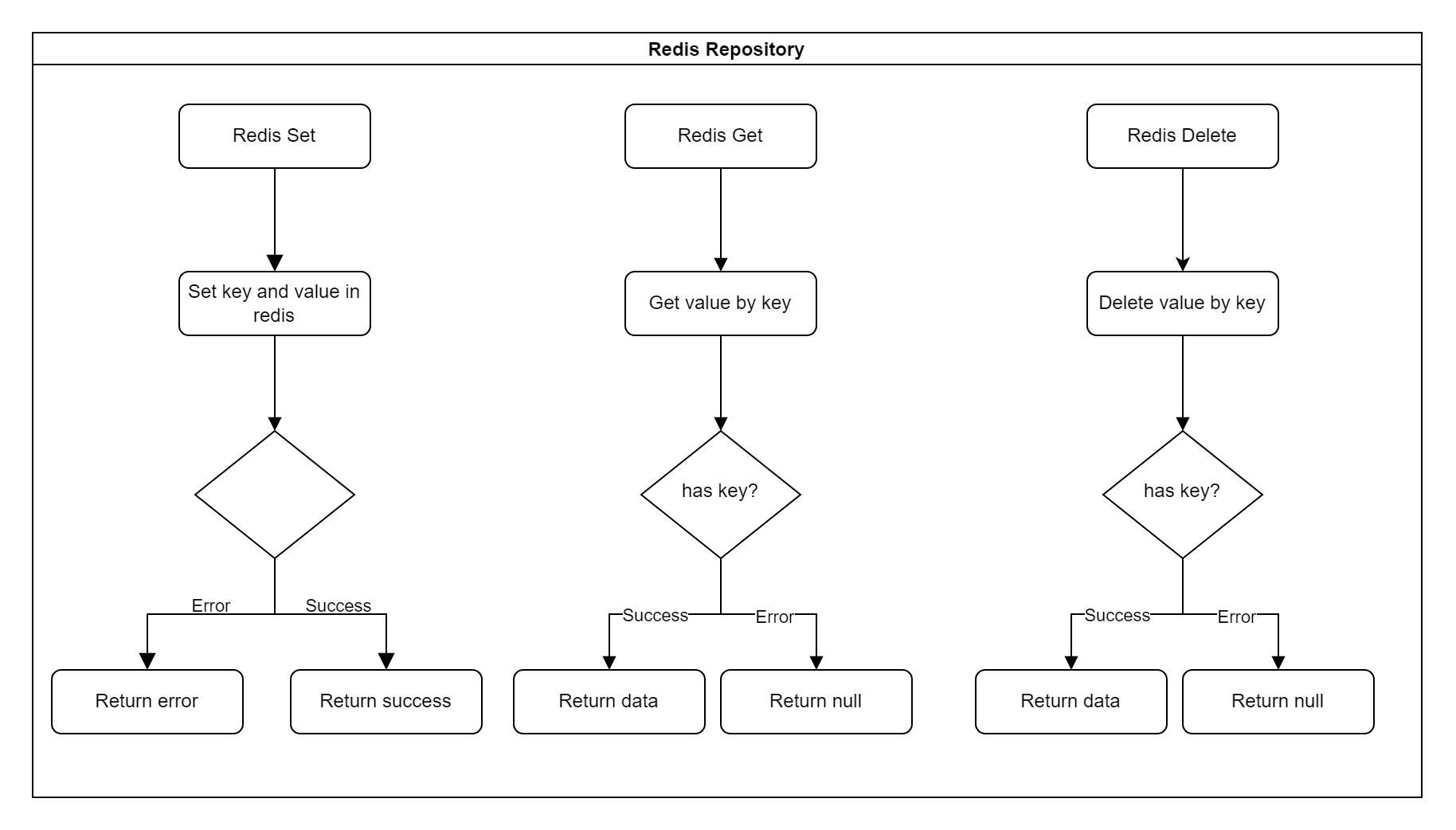
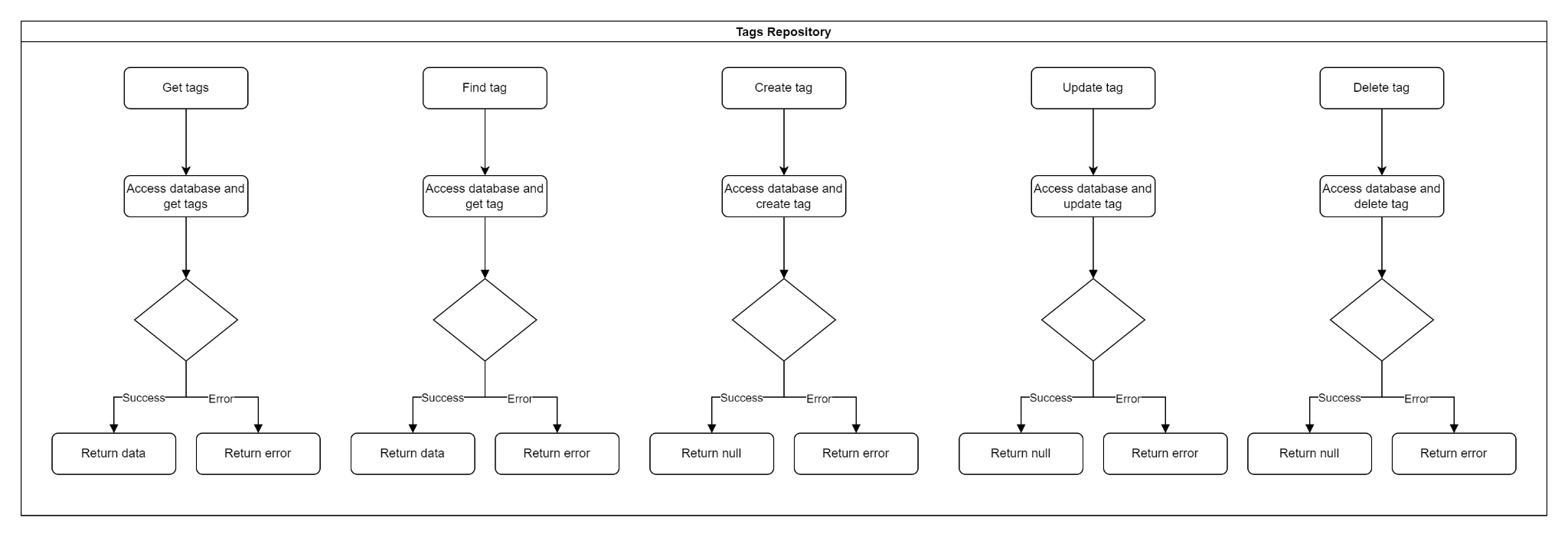
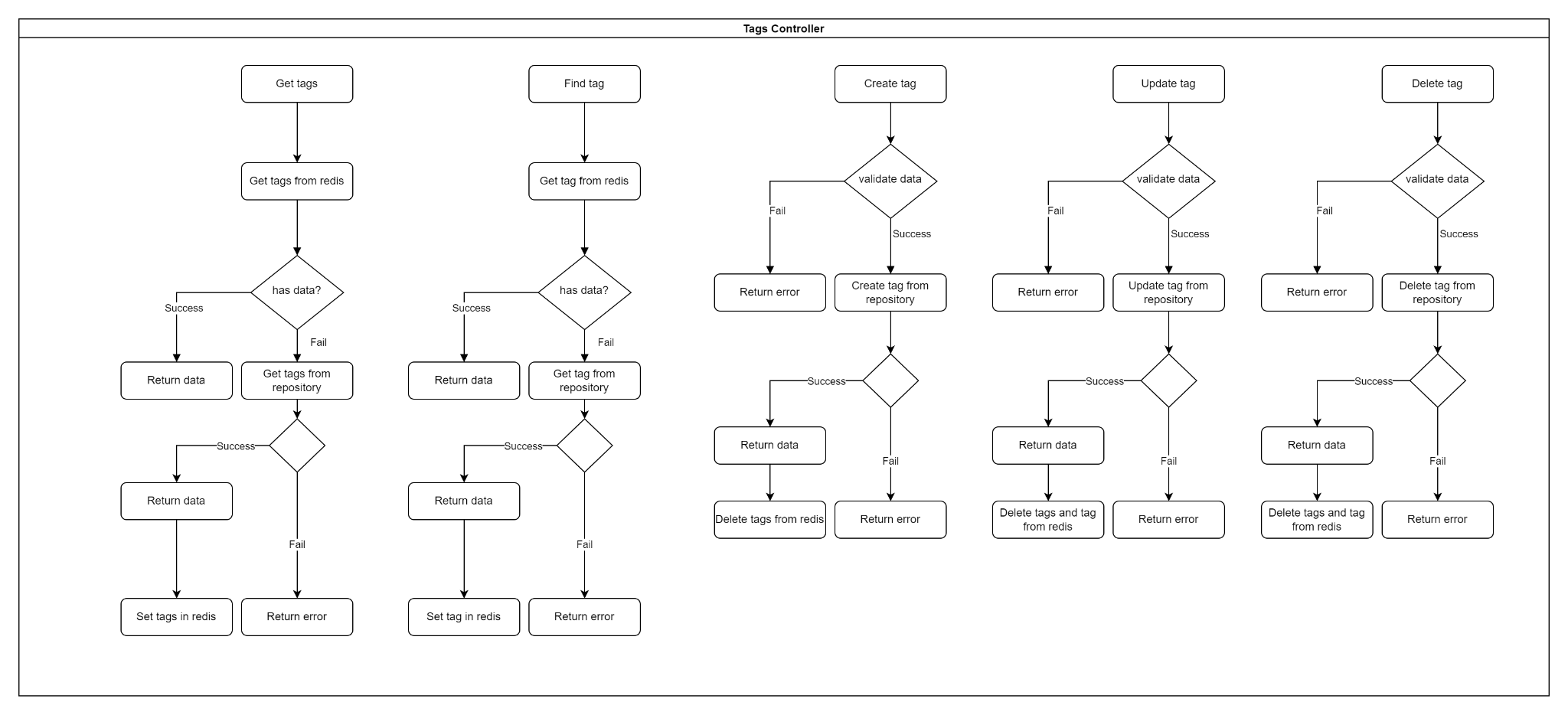
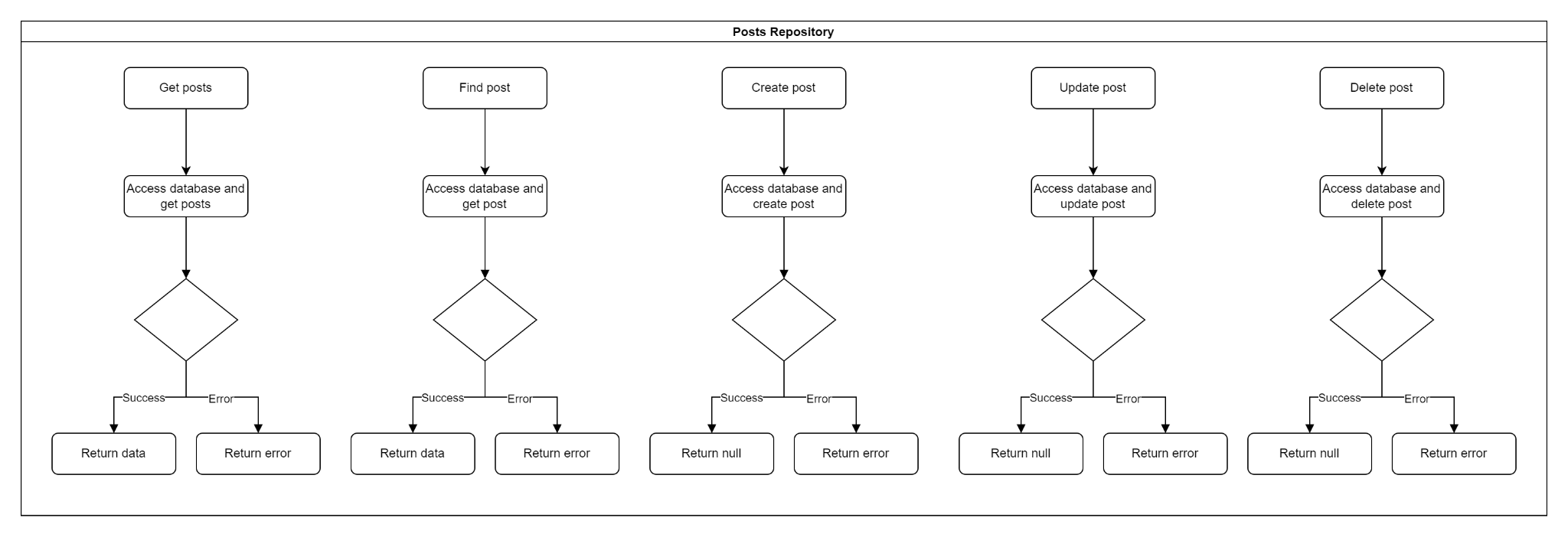
****

* + 1. Auth Service

****

* + 1. Forum Service

****

****

1. 待定清單
2. 引入文本情感分析AI模型，進行AI自動化文章審核功能 。
3. 學生發文進度查詢。
4. 組員分工

|  |  |
| --- | --- |
| **組員名稱** | **工作** |
| 鄭政文 C109151104 | Forum Servies、Design System、Database Design |
| 沈育安 C109151110 | Server Manage、Website Frontend |
| 劉官瑜 C109151102 | Website Frontend |
| 吳俊傑 C109151150 | Authentication Servies |
| 丁襄龍 C109151163 | Gateway and APP Frontend |
| 許哲晟 C109151162 | APP Frontend |