線上家教應用程式

Online Tutor Application

辛錫進、王彥文、邱茂瑜、宋文斌

國立聯合大學 資訊工程學系

苗栗市南勢里聯大 2 號

flhuang@nuu.edu.tw

一、專題介紹與專題定位

本專題的目標是開發一款線上家教應用程式，提供學生與老師在網路上進行遠距教學的便利平台，無需面對面即可完成教學互動。應用程式內建多種教學輔助工具，例如整合本地化的語言模型（Local LLM），幫助學生和老師解答疑惑，提升學習效率。此外，系統還包含群組管理功能與線上測驗模組，方便老師管理學生並進行即時的學習評估。

然而，我們的應用程式並非家教媒合平台。學生與老師需要透過其他方式進行配對，然後使用我們的應用程式進行教學活動。由於我們仍是學生，尚無能力處理涉及金錢交易及用戶權益相關的複雜事宜，因此選擇專注於教學工具與功能的開發，以提供高效且易用的教學輔助平台。

二、專題構想與架構設計

我們想要設計並開發一款具有完整軟體架構的家教應用程式，著重於穩定性、擴充性以及測試性，以確保系統能長期穩定運行並適應未來需求。為實現這一目標，我們將系統依職責劃分為三個層次：

UI 層：負責呈現 GUI 顯示界面及處理用戶互動。

邏輯層：處理核心功能邏輯，作為應用程式的運算與決策中心。

資料層：管理底層的資料結構以及所有資料來源。

此外我們在 UI 層使用 MVI（Model-View-Intent） 設計模式，細分如下：

Model：定義應用程式的狀態，確保數據與狀態的一致性。

View：負責渲染應用程式的狀態並回饋用戶操作。

Intent：封裝用戶的操作，統一傳遞至邏輯層進行處理。

在整體架構中，我們引入了 BLoC（Business Logic Component） 作為邏輯層的入口，負責將資料層及邏輯層與 UI 層解耦且當作其之間的橋樑。BLoC 通過資料流式處理實現狀態的單向數據流，確保邏輯的清晰與穩定，並提供良好的測試性與可維護性。

架構的優勢

1. 明確的責任分工：UI 層、邏輯層、資料層各司其職，提升了代碼的可讀性與可維護性。

2. 高度的解耦設計：通過 MVI 和 BLoC 的結合，減少了模組間的依賴，提升系統的彈性。

3. 易於測試與擴展：清晰的架構使單元測試與功能測試變得簡單，為未來的功能迭代提供了堅實基礎。

相較於盲目地追求功能疊加，我們更注重系統性與結構化的開發流程，致力於打造一個高效、穩定且可持續維護的應用程式。這樣的設計不僅滿足了當前需求，也為長期發展奠定了堅實的基礎。

三、專題開發前的規範與預計採用技術

考量到我們的用戶，包括學生與老師，可能會使用電腦或移動裝置來操作應用程式，我們選擇了 Flutter 作為開發框架。

相比於其他跨平台框架如 React Native，我們認為 Flutter 更具以下優勢：

1. 跨平台支援：Flutter 能以單一程式碼構建 Android、iOS、Web 及桌面應用程式，極大地降低了開發與維護成本。

2. 優異的性能：Flutter 使用自有的繪圖引擎，直接渲染 UI，無需依賴橋接技術（Bridge），相較於 React Native，能提供更流暢的用戶體驗。

3. 完整的開發體驗：Flutter 提供豐富且統一的 UI 元件庫，避免依賴第三方套件導致的不一致性與潛在問題。

此外，Dart 作為 Flutter 的底層開發語言，也相較 JavaScript 提供了顯著優勢：

1. 靜態型別安全：Dart 在編譯階段檢查型別錯誤，降低了運行時的潛在風險。

2. Null Safety：Dart 內建 Null Safety 機制，有效防止空值相關的錯誤，提升系統穩定性。

3. 語法簡潔直觀：Dart 的學習曲線平緩，比 JavaScript 更易於上手，特別適合需要快速開發與高效維護的專案。

基於 Flutter 的跨平台特性與 Dart 提供的安全性和易用性，我們得以設計出穩定、高效且便於維護的應用程式架構，滿足多設備用戶的需求，並為未來的功能擴展打下堅實基礎。

在正式開發前，我們制訂了明確的開發規範，像是統一變數名稱的格式與風格提升可讀性與一致性，明確劃分檔案的功能與用途，並採用清晰的資料夾結構，讓團隊成員能快速定位與管理檔案。

這些規範的制定不僅確保了開發過程的高效性與條理性，還讓每位成員能專注於各自的任務分工。

為了實現應用程式的實際運營，我們在後端架構上選擇了 Firebase 作為後端伺服器。Firebase 不僅由 Google 提供運營，還依靠其強大的基礎設施為用戶帶來快速且穩定的體驗。以下是 Firebase 為我們系統提供的核心優勢與功能：

1. OAuth 用戶驗證整合：簡化了多平台的第三方登入流程，提升用戶體驗。

2. Firestore 可視化 NoSQL 資料庫：提供高效的資料存取與更新，滿足應用程式的即時性需求。

3. Cloud Messaging：支援推播通知功能，實現 App 外的訊息服務，提升用戶互動性。

這些功能經過完整測試，讓我們可以專注於應用程式的開發，無需擔心基礎設施的穩定性與安全性。

視訊與畫面分享技術

為了解決視訊通話與畫面分享需求，我們採用了 WebRTC。WebRTC 提供了一套簡單的 API 與高彈性的框架，讓我們能輕鬆實現以下功能：

1. 端對端加密的 P2P 連線：不僅確保用戶的通訊安全，還免除了手動實作加密的負擔。

2. 減少中繼伺服器流量負擔：透過直接的 P2P 連線，大幅降低伺服器的

運算壓力與成本。

本地化 LLM（Local LLM）

在人工智慧功能方面，我們選用了 MediaPipe 框架。MediaPipe 採用了輕量化的 TFLite 模型，允許我們在移動裝置上運行 AI 模型，帶來以下優勢：

1. 省錢：用戶無需購買 ChatGPT 或 Gemini 等服務，便可在本地進行詢問與互動。

2. 隱私性與即時性：所有 AI 處理均在裝置本地完成，避免了資料上傳的延遲與隱私問題。

群組與測驗功能

在群組管理與線上測驗的開發中，我們同樣依靠 Firestore NoSQL 資料庫來協助建立和管理測驗資料。Firestore 提供了即時同步與高效查詢能力，讓老師能快速創建、發布並管理學生的測驗內容。

透過 Firebase、WebRTC、MediaPipe 與 Firestore 等技術的整合，我們成功打造了一款具備穩定性、高效能且具現代化功能的家教應用程式。這些技術選擇不僅確保了系統的實用性，還為用戶提供了良好的使用體驗，同時降低了開發與運營的技術負擔與成本。

四、主要功能說明與技術實現

1. 用戶登入

功能說明：用戶可透過 Gmail 快速登入應用程式，無需額外註冊，提供便捷且安全的登入體驗。

技術實現：我們採用了 Firebase Auth 與 Google OAuth 完成用戶驗證，並實時監控用戶的登入狀態。若用戶登出，系統會立即將其重定向至登入頁面，確保帳號安全性。同時，將用戶驗證交由 Google 處理，不僅減輕了我們的資料維護負擔，也為用戶提供了無需額外帳號的便利性。

2. 即時聊天室與狀態顯示

功能說明：用戶可以透過內建的即時聊天室進行交流，並可查看對方是否在線或正在使用聊天室。該功能解決了用戶隱私問題，避免用戶必須分享私人聯絡方式（如 Line 或電話號碼），同時提供了方便的溝通途徑。

技術實現：我們使用 Firestore 儲存聊天訊息，並利用 Firestore 的資料變更監聽功能，實現即時更新聊天室狀態，確保聊天流暢和準確。

3. 好友功能

功能說明：用戶可以透過搜尋名稱添加好友，成為好友後即可進行聊天和線上教學互動。

技術實現：與即時聊天室相同，我們使用 Firestore 儲存好友資料並進行狀態更新。

4. 即時通知

功能說明：用戶可接收到即時通知，例如新消息或系統提醒，確保資訊傳遞的及時性。

技術實現：與聊天室功能類似，我們利用 Firestore 實現通知的即時更新和推送。

5. 線上教室

功能說明：用戶可以透過好友頁面的教室功能，進行視訊教學或螢幕分享，提供便利的線上互動教學體驗。

技術實現：此功能基於 WebRTC 技術，利用其高效的 API 與 P2P 加密，實現低延遲的音視頻傳輸及螢幕共享。

6. 群組與測驗功能

功能說明：用戶可以創建群組進行學術討論，或建立測驗方便老師管理學生學習進度。

技術實現：我們利用 Firestore 儲存群組與測驗資料，並提供即時更新功能，確保多用戶間的協同流暢。

7. 本地語言模型（Local LLM）

功能說明：用戶可在應用程式內，透過本地模型獲取學術問題的答案，無需依賴雲端服務，節省成本且保護用戶隱私。

技術實現：採用了 MediaPipe 與 TFLite 模型，允許用戶在移動裝置上直接執行 AI 模型，實現離線的語言模型功能。

五、開發過程中的難題與解決

開發過程中的難題與解決方案

1. Google OAuth Platform Error 10

我們在實現 OAuth 登入功能時遇到了第一個重大問題，即 Platform Error 10。這是一個平台設置錯誤，而非程式碼問題。無論我們如何重新配置設定或參考其他專案的範例，問題依然存在。經過一周的調查後，我們發現問題的根本原因在於，Google OAuth 註冊的套件名稱與專案實際使用的套件名稱不一致。修正套件名稱後，問題順利解決，讓我們成功導入 Google OAuth 登入功能。

2. 對封裝與抽象化的過度執著

起初我們對專案設計追求完美，特別是資料層的封裝與抽象化。我們試圖將後端 API 和第三方套件完全封裝成介面，實現最小暴露與可測試性。然而這需要詳細閱讀原始碼，並進行高難度的封裝，尤其是 Dart 語言的限制使得某些功能無法完全封裝。後來我們妥協，採用了不完全封裝的方式，或使用 Mocktail 套件模擬資料來源，實現更高效的測試與開發流程。

3. WebRTC 連線問題

WebRTC 需要透過 NAT 穿透技術交換 Peer 的公共 IP 和 Port。然而在非對稱式 NAT 架構下，P2P 連線可能失敗。我們嘗試使用實驗室架設的 Coturn 伺服器，但因學校的 IP 地址頻繁變更而無法穩定運作。最終雖然只能使用傳統 P2P 連線，我們學習到 WebRTC 的另一種方法：使用中繼伺服器交換影音資料，儘管未能實現此方案，但為未來的設計積累了寶貴經驗。

4. WebRTC Signaling

WebRTC 連線前需要有個中繼站交換連線資訊與 SDP(影音編碼資訊)。由於無法自行搭建伺服器，我們使用 Firestore 作為 Signaling Server，整體運作良好。

然而在交換 SDP 的過程中，表現出抽象狀態（如等待、處理、送出）的難度極高。最終我們透過三個旗標模擬狀態，解決了在 Firestore 上實現 SDP 交換的問題。

5. WebRTC Race Condition

WebRTC 的高度彈性使得開發者需要自行實作多數邏輯，例如 SDP 交換過程中的同步問題。當雙方同時交換 SDP 時，可能導致 Race Condition。我們的解決方案是檢測雙方是否同時有交換意圖，若發生則取消本次交換，儘管可能需要使用者重試操作，但降低了問題的複雜性。此外，音訊與影像的狀態同步問題，我們改為等待資料傳輸完成後才通知對方，最終解決了這一系列 Race Condition 問題。

6. Flutter Master Channel 的問題

為使用 MediaPipe 套件，我們將 Flutter 從穩定版本升級至 Master 版本。然而此舉導致許多原本正常的功能出現嚴重錯誤，例如 Android 前台服務權限的問題。最終我們找到解決方案，通過重複啟動兩次應用程式以成功獲取權限，雖是權宜之計，但確保了功能的正常使用。

7. MediaPipe 套件的不穩定性

我們使用的 MediaPipe 套件版本為 0.0.1，且長期未更新，導致多個問題，包括無法在 Release 模式下編譯 APK（僅支持 Debug 模式）。此外，該套件在 Android 平台上完全無法使用。為解決此問題，我們計劃通過 Flutter 平台方法調用 Android 原生 MediaPipe 庫，目前尚在實現中。

8. TFLite 模型太大跟不太聰明

由於 MediaPipe 僅支援 TFLite 模型，我們選擇了 Gemma 模型，因為其他模型需要手動轉換格式。然而，Gemma 模型容量非常大（通常超過 1GB），直接將其放入應用的 assets 資料夾中幾乎不可能。最後，我們決定改用線上下載模型到應用的 application documents directory，這個方法運作良好。此外
Gemma 模型在智能化方面表現有限，尤其是對中文的理解非常弱，因此用戶只能使用英文進行交互。這個問題短期內無法解決，只能期待未來有更強大的模型來改善。

9. Gemma 上下文理解問題

Gemma 模型是無狀態的，因此無法記住之前的對話，導致回答時可能缺乏上下文理解。我們的解決方案是記錄每次對話的內容，並在發送新問題時將這些歷史內容作為提示一起傳遞給模型。然而，這仍有待進一步優化，目前大部分情況下模型表現正常，但有時候會出現跳針或無法理解的回應。我們正在分析問題來源，可能是模型本身的限制或提示設計不夠完善。

六、未來方向與發展潛力

1. 家教配對功能

未來，我們希望進一步擴展應用，將現有技術整合進家教配對平台。例如，透過社群功能，讓教師能設立個人專頁，展示專長科目、授課時間和費用。學生則可透過 Visa 或 Paypal 完成付費流程，快速獲取教學資源。此外，我們計劃加入討論區功能，促進學生間的學科交流，進一步提升應用的互動性與實用性。

2. 突破 P2P 一對一限制

因現有的實驗室網路條件，無法完整搭建 Coturn 影像中繼服務，導致目前僅支持 P2P 一對一連線。未來若具備資金支持，我們可考慮將 Coturn 部署至雲端，甚至利用 Kubernetes 管理多個 Coturn 容器，實現穩定的大規模連線。如此一來，應用將可支援一對多教學，擴展至遠距課堂的場景，例如 Google Meet 或 Microsoft Teams 的模式。

3. 課堂回放功能

WebRTC 內建的錄像功能，讓課堂內容的錄製與回放成為可能。未來，我們可加入此功能，讓學生在課後能隨時重溫教學內容，提升學習效果。此功能因專題時程限制未能實現，但具有極大的發展潛力。

4. 語音字幕與翻譯

應用語音辨識技術，為課堂內容生成實時字幕，並利用翻譯功能將其轉換為其他語言，是一項令人期待的進階功能。然而，目前的第三方語音辨識套件辨識能力有限，僅能處理短時間的語音片段，未來若技術成熟，將能大幅提升應用的多語言教學價值，拓展至跨國教育領域。

5. Gemma 助教化應用

未來，若 Gemma 模型與 MediaPipe 更新更強大的功能，應用將能進一步發展，例如透過語音辨識記錄課堂內容，並結合影像分析技術，讓 Gemma 充當虛擬助教，協助生成課堂摘要或講義，提升教學效率。

6. iOS 平台及上架計畫

目前，受限於缺乏 iOS 實機測試條件，應用的功能尚未能支援 iOS 平台。若未來具備相關設備，將可進一步開發並測試 iOS 版本。此外，基於我們現有使用 Firebase 的架構，應用具有快速上架至 App Store 及 Google Play 的潛力，未來將致力於提升功能完整度以實現此目標。

七、結論

本專題開發了一款專注於線上教學的家教應用程式，透過結構化的軟體架構及整合多項現代技術，如 Firebase、WebRTC 和本地化 LLM，提供穩定、高效且使用者友好的教學平台。系統功能涵蓋用戶登入、即時通訊、線上教室、好友管理、通知提醒及測驗輔助等，旨在滿足學生與教師的遠距互動需求。

專案開發過程中，我們注重系統的穩定性、擴展性及測試性，並採用高度解耦的設計模式如 MVI 和 BLoC，確保長期維護的便利性。選擇 Flutter 為開發框架進一步提升了跨平台支援與性能表現，使應用程式能在多設備環境中無縫運行。

整體而言，本應用程式實現了專題目標，為遠距教學提供了實用的解決方案，並為後續功能擴展及技術創新奠定了良好的基礎。