**Notes on Key Decisions**

* **Functionality**: Reused the user stories and task breakdowns from the previous response, ensuring no deleted content (e.g., ingredient details, contact form) was included. Adjusted Must Have #20 to “manage product categories” as per your latest list.
* **Content**: Assumed developer or client will write text content; images are licensed or client-provided to avoid copyright issues.
* **Technical Stack**: Chose WordPress/WooCommerce for rapid development, with a custom child theme and plugin to showcase skills.
* **Schedule**: Spread 177 hours over 8 weeks, accounting for planning, development, and testing phases.
* **Accessibility**: Included basic WCAG compliance (e.g., semantic HTML, alt tags) despite removing screen reader story, as it’s a standard requirement.
* **GitHub**: Placeholder link provided; you’ll need to create the repository.

**Recommendations and Next Steps**

1. **Review Document**:
   * Share the artifact with your faculty supervisor to confirm scope and timeline.
   * Adjust hourly estimates or schedule if needed based on feedback.
2. **Create Wireframes**:
   * Use Figma or Sketch to develop wireframes for Homepage, Shop, Product, Cart, and Checkout pages.
   * Validate layouts with stakeholders before coding.
3. **Source Content**:
   * Draft text for Homepage, About page, and product descriptions.
   * License cake images from stock sites or coordinate with client for photos.
   * Generate privacy policy using Termly.io.
4. **Set Up Development Environment**:
   * Install LocalWP for local WordPress development.
   * Create GitHub repository and initialize with child theme and plugin folders.
   * Install WordPress 6.6 and WooCommerce 9.2.
5. **Begin Development**:
   * Start with child theme setup and WooCommerce configuration (Week 2).
   * Prioritize Must Have user stories (#1–17) for Shopper flow to build core functionality.
6. **Test and Deploy**:
   * Test responsiveness on mobile devices and browsers.
   * Deploy to SiteGround (or chosen host) in Week 8.
   * Document any issues and resolutions.
7. **Finalize Documentation**:
   * Add wireframes and content sources to the Content Evaluation section.
   * Update GitHub link in Development Specifications.
   * Submit final Requirements Document and project deliverables to faculty.

**說明**

Capstone 專題作業是 Web Development 課程的最終項目。目的是創建一個應用程式，展示你在能力、才華、見解、工作態度和知識方面的綜合表現。因此，該應用應具備良好的設計、易於使用、精緻且專業。同時別忘了它也應該對殘障人士友好，並具備跨裝置的響應式設計。這是一項需要大量努力的任務。為了幫助你專注於創作出最佳應用，我們限制了可選擇的應用類型。這不是為了壓制你的創意，而是為了給你一個更明確的工作目標。每個應用都配有詳細的使用者故事，有助於你進行開發。

**第一步**

你應該先做以下幾件事：

1. 選擇一個應用類型，並取得老師的同意。
2. 為你的應用選擇一個主題。例如，如果你選擇的是「預約系統」，那是用於理髮店還是家教服務？
3. 閱讀所有的使用者故事，確保你理解它們，也了解你想開發的應用的複雜程度。
4. 檢視該使用者故事清單是否缺少對你的主題來說很重要的內容。例如，針對家教預約系統，可能需要針對家長和學生分別定義不同的使用者故事。
5. 注意：你的內容也可能需要未被列出的使用者故事。
6. 根據你對 MVP（最小可行產品）的理解，評估所有使用者故事。分類如下：
   * **必須有（Must have）**：這些是核心功能。
   * **應該有（Should have）**：這些是非核心但有價值的功能。
   * **可以有（Nice to have）**：這些是提升應用完整性與質感的額外功能。
7. 根據以上分類，整理出使用者故事的清單。
8. 對於所有「必須有」的使用者故事，進行評估並確定開發所需的功能，這將成為需求文件的一部分。

**使用者故事**

使用者故事描述的是使用者能執行的操作。作為開發者，你需要閱讀這些故事，並拆解出為了實現它所需的具體任務，這相當於待辦事項清單。針對「必須有」與「應該有」的故事，你都需要這樣做，並估算完成每項任務所需的時間。以下是兩個範例：

**範例一**

**使用者故事**：作為一個使用者，我想登入網站。

所需任務與時間估計如下：

| **任務** | **時間（小時）** |
| --- | --- |
| 頁面文字 | 0.5 |
| 登入表單 - 使用者名稱 | 0.5 |
| 登入表單 - 密碼 | 0.5 |
| 提交按鈕 | 0.5 |
| 資料庫功能 | 15 |
| 驗證機制 | 1.5 |
| 忘記密碼連結 | 0.5 |
| 保持登入選項 | 2 |
| 驗證碼（Captcha） | 7 |
| **總計** | **28** |

忘記密碼不是這個使用者故事的一部分，因此你應建立一個連結到另一個故事：「作為使用者，我希望能找回忘記的密碼」。

**範例二**

**使用者故事**：作為一位管理者，我希望能編輯頁面內容。

這屬於 CRUD 操作中的更新（Update）功能。

| **任務** | **時間（小時）** |
| --- | --- |
| 存取頁面 | 4 |
| 顯示欲編輯頁面內容 | 1.5 |
| 提交按鈕 | 0.5 |
| 取消按鈕 | 0.5 |
| 資料庫功能 | 12 |
| **總計** | **18.5** |

注意：這個故事與另一個故事有關：「作為管理者，我能創建新的頁面內容」。這兩者可共用相同的表單與驗證邏輯，展現出良好的程式碼重用能力。

**我需要多少個使用者故事？**

這沒有一定的答案，因為每個應用的複雜度不同。你應與指導老師討論應用的規模與難度。我們不希望你給自己太大壓力。一般來說，請以以下數量為目標：

* **25 個**「必須有」的使用者故事
* **5 個**「應該有」的使用者故事
* **3 個**「可以有」的使用者故事

也請注意這些故事之間的順序與互動，這樣你可以設計出有邏輯連貫性的流程。

**需求文件中還包含哪些內容？**

請注意：這些應用預設是沒有內容的，內容的產出由你負責。內容可能包含：文字、圖片、外部 API 資料等。你需要說明這些內容的來源與取得方式。同時還要撰寫應用的整體理念，包括其目標、預期效果與動機。

以下是需求文件的建議目錄：

1. **簡介與開發理念**
2. **內容規劃**
   * 導覽架構
   * 版面配置（初步線框圖）
   * 內容說明與來源
3. **功能規劃（使用者故事）**
   * 必須有
   * 應該有
   * 可以有
   * 時間估算
4. **技術規格**
   * 使用的技術組合（技術棧）
   * 主機與部署資訊
   * 第三方軟體工具（提供版本與連結）
5. **開發規格**
   * 資料夾結構
   * 檔名命名規則
   * 無障礙設計標準
   * 響應式需求
   * GitHub 專案連結
6. **開發時間表**
7. **結語**

**評分標準（Rubric）**

| **分數** | **說明** |
| --- | --- |
| 10 分 | 完整包含所有目錄內容，有約 30 個使用者故事，導航與線框圖清楚，內容規劃明確。 |
| 6 分 | 包含大部分內容，但使用者故事、線框圖、內容等有缺漏。 |
| 4 分 | 內容缺漏嚴重，整體文件不完整。 |
| 1 分 | 幾乎沒有投入心力，文件非常不完整。 |

**最後提醒**

這份文件的目的，是為了在你動手寫程式前，妥善規劃你的應用。透過這種方式整理使用者故事，你就擁有了一份詳細的待辦清單。當所有的使用者故事都開發完成並成功整合進應用中時，你的項目也就完成了。這不是一件很棒的事嗎？

**從提案走向需求文件**

恭喜你，項目構想已經草擬並獲得核准。下一步是撰寫 **需求文件（Requirements Document）**，詳細記錄你所有的開發規格。你必須寫得非常詳盡，理論上這份文件可以交給另一位開發者，他就能只靠它完成整個應用。而客戶也會根據這份文件來判定你是否達成承諾。

聽起來很多？確實如此。不過以下是能幫助你起步並避免過度焦慮的策略：

**使用者故事步驟建議**

1. 先從 **Must Have（最小可行產品）** 的功能或場景出發，轉化成使用者故事。
   * 你可能會發現需要不同層級的使用者存取權限。為了更清楚表達，使用首字母大寫來區分不同使用者角色（例如 Teacher、Parent）。若能將資料結構也一併命名清楚（如 Room vs. meeting），能幫助溝通整體架構。
2. 完成 Must Have 的使用者故事後，繼續擴展 Should Have，再來是 Nice-to-have。
3. 根據這些故事，整理你需要的其他資料與內容：資料來源、網站架構、資料流程圖、介面設計等。
4. 清單完成後，開始填寫需求文件中對應的詳細資料。
5. 一切都會是反覆修正的過程，不必苛責自己。當你變更某一個元件，記得確認其他相關元件也要同步更新。

**範例**  
這是一份關於網頁開發待辦清單（To Do List）提案的節錄：

**功能**

「使用者可以新增任務到自己的任務清單，為任務選擇一個類別，並設定到期日。」

**從此功能延伸出的使用者故事**

* 開發人員可以查看所有分配給他們的任務。
* 開發人員可以為自己建立新的任務。
* 開發人員可以為任務選擇一個類別。
* 開發人員可以為任務設定到期日。
* 開發人員可以將任務標記為已完成。
* 開發人員可以編輯或刪除自己的任務。

**資料來源（資料表）**

從使用者故事可以看出，我們需要為「開發人員」這個使用者角色提供完整的 CRUD 功能（建立、讀取、更新、刪除）。再來，看「開發人員可以為自己建立任務」這句，我們就知道必須要有一個 **開發人員（Developers）** 資料表，因為任務只能由對應的開發人員建立。此外，我們還需要一個 **任務（Tasks）** 資料表。如果市面上有可用的 API 來提供開發人員或任務的資料，我們也可以使用它。但在這個案例中，我們會自行建立資料來源。  
從其他使用者故事來看，我們還應該設計出任務與類別的資料表。  
目前來說，一個任務需要包含以下欄位：所屬的開發人員、任務類別、到期日，以及是否已完成。

**資料來源：資料庫資料表**

* **Developers（開發人員）**：
* **Tasks（任務）**：Developer | Category | Date | Completed
* **Categories（任務類別）**：

**從使用者故事中，我們可以推導出資料流程圖（Data Flow Diagram）與畫面線框圖（Wireframes）……**

**網站架構與資料流程**

* 開發人員登入
* 開發人員查看任務
* 開發人員建立新任務
* 開發人員編輯或刪除既有任務
* 開發人員標記任務為已完成

**介面設計（畫面線框）**

* 首頁
* 登入畫面
* 查看所有任務（列表檢視）
* 建立／編輯任務（詳細檢視）

請先從這樣的條列式開始，接著再一步步細化各個元件，深入設計細節。這會幫助你從整體掌握應用程式的運作方式，進而讓你後續的設計決策更加順利。

**保持一致性非常重要！**  
資料流程中不應包含沒有對應畫面操作的使用者步驟；  
介面畫面中也不應包含沒有儲存在資料庫中的輸入項目；  
資料庫則不應有任何在使用者故事中未提及的欄位。

在上述的 To Do List 範例中，若我們後來想為「主導開發者（Lead Developer）」這個管理者角色建立新的使用者故事，例如「任務只有在被主導開發者審核過後才算完成」，那我們就需要做以下調整：

* 在 **Developers** 資料表中加入 **Role（角色）** 欄位；
* 在 **Tasks** 資料表中加入 **Status（狀態）** 欄位；
* 在資料流程圖中加入一個「審核步驟」；
* 在主導開發者的任務介面中加入「核准（Approve）」按鈕；
* 在開發人員的介面中加入「核准完成」的圖示；

你還需要做什麼？**持續反覆迭代與更新！**