專案任務：為 eccal 專案建立「行銷企劃 AI 資料庫」功能

1. 專案目標 **(Project Goal)**

我們要為現有的 `eccal` 專案後台 (`/bdmin`) 新增一個名為「行銷資料庫」的模組。此功能的目標是讓管理員能上傳行銷企劃的 PDF 檔案，透過 AI 自動分析檔案內容，將其中的策略摘要出來，並根據「預熱期」、「活動期」、「回購期 」三個階段進行分類，這三個階段應該與 /campaign-planner 規劃出來的定義相同。分析完成的結果需存入資料庫，並提供後台介面讓管理員進行審核與編輯。

2. 核心流程與需求 **(Core Flow & Requirements)**

1. 檔案上傳: 管理員在後台上傳一個 PDF 檔案。
2. 後端接收: 伺服器接收到檔案後，將其儲存至臨時目錄。
3. 立即回應與背景處理: API 應立即回傳「處理中」的狀態給前端，並開始在背景非同步地執行以下任務：
    - 解析 PDF 檔案以提取純文字內容。
    - 呼叫 AI 模型，傳送提取出的文字和一個精確的指令 (Prompt)，要求其分析並回傳結構化的 JSON 資料。
    - 將 AI 回傳的各項策略存入資料庫。
4. 檔案清理: 無論分析成功或失敗，任務完成後，伺服器上的臨時 PDF 檔案必須被刪除。
5. 審核與編輯: 管理員可以在後台介面查看分析結果，並手動修改每個策略項目所屬的階段（例如，從「活動期」修正為「預熱期」），以及對每個項目進行「核准」。

3. 後端開發任務 **(Backend Tasks)**

3.1. 安裝新套件

請執行以下指令來安裝必要的套件：

```
npm install multer pdf-parse
npm install --save-dev @types/multer
```

3.2. 資料庫 **Schema** 擴充 **(`shared/schema.ts`)**

請新增以下兩個 Drizzle ORM 資料表定義：

```
// 檔案上傳任務與行銷企劃的記錄表
export const marketingPlans = pgTable("marketing_plans", {
  id: text("id").primaryKey().$defaultFn(() => crypto.randomUUID()),
  fileName: varchar("file_name").notNull(),
  fileSize: integer("file_size"),
  fileType: varchar("file_type"),
  status: varchar("status", { enum: ["processing", "completed", "failed"]
}).default("processing"),
  errorMessage: text("error_message"),
```

```
  uploadedBy: varchar("uploaded_by").references(() => users.id),
  createdAt: timestamp("created_at").defaultNow(),
  completedAt: timestamp("completed_at"),
});

// AI 分析出的策略項目表
export const planAnalysisItems = pgTable("plan_analysis_items", {
  id: text("id").primaryKey().$defaultFn(() => crypto.randomUUID()),
  planId: text("plan_id").notNull().references(() => marketingPlans.id, { onDelete: 'cascade' }),
  phase: varchar("phase", { enum: ["pre_heat", "campaign", "repurchase"] }).notNull(),
  strategySummary: text("strategy_summary").notNull(),
  isApproved: boolean("is_approved").default(false),
});
```

成後，請執行 `npm run db:push` 將變更同步至資料庫。

3.3. 更新資料存取層 **(`server/storage.ts`)**

請在 `IStorage` 介面和 `DatabaseStorage` 類別中，加入以下用於操作新資料表的方法：

```
// 在 IStorage 介面中新增
createMarketingPlan(plan: Omit<typeof marketingPlans.$inferInsert, 'id' | 'createdAt'>):
Promise<typeof marketingPlans.$inferSelect>;
updateMarketingPlanStatus(planId: string, status: 'completed' | 'failed', errorMessage?:
string): Promise<void>;
saveAnalysisItems(planId: string, items: Array<Omit<typeof planAnalysisItems.$inferInsert,
'id' | 'planId'>>): Promise<void>;
getMarketingPlans(): Promise<Array<typeof marketingPlans.$inferSelect>>;
getAnalysisItemsForPlan(planId: string): Promise<Array<typeof
planAnalysisItems.$inferSelect>>;
updateAnalysisItemPhase(itemId: string, newPhase: 'pre_heat' | 'campaign' | 'repurchase'):
Promise<void>;
approveAnalysisItem(itemId: string, isApproved: boolean): Promise<void>;

// 在 DatabaseStorage 類別中實作上述方法
async createMarketingPlan(plan) { /* ... */ }
async updateMarketingPlanStatus(planId, status, errorMessage) { /* ... */ }
async saveAnalysisItems(planId, items) { /* ... */ }
async getMarketingPlans() { /* ... */ }
async getAnalysisItemsForPlan(planId) { /* ... */ }
async updateAnalysisItemPhase(itemId, newPhase) { /* ... */ }
async approveAnalysisItem(itemId, isApproved) { /* ... */ }
```

3.4. 建立後端 **API** 端點 **(`server/routes.ts`)**

請建立以下 API 端點來驅動整個功能：

1. **`POST /api/bdmin/marketing-plans`**:
    - 使用 `multer` 處理 `multipart/form-data` 格式的檔案上傳，檔案欄位名為 `file`。
    - 流程：
        1. 驗證是否為管理員 (`requireAdmin`)。
        2. 驗證檔案是否存在。
        3. 將檔案儲存至伺服器上的 `tmp/uploads/` 目錄。
        4. 在 `marketing_plans` 資料表中建立一筆狀態為 `processing` 的紀錄。
        5. 立即回傳 `202 Accepted` 和這筆紀錄的 JSON 給前端。
        6. 非同步地執行後續的檔案解析和 AI 分析。
        7. 使用 `pdf-parse` 解析 PDF 內容。

呼叫 AI 模型（請先用模擬資料），並傳送以下指令：
你是一位專業的電商行銷策略分析師。請分析以下這份行銷企劃的文字內容，並將其策略摘要出來。將所有策略嚴格地歸類到 'pre_heat', 'campaign', 或 'repurchase' 這三個階段中。請嚴格以 JSON 格式輸出，格式為 { "pre_heat": ["策略一", "策略二"], "campaign": ["策略三"], "repurchase": [] }，不要包含任何 JSON 格式以外的解釋文字。

        8.
        9. 將 AI 回傳的 JSON 資料整理後，存入 `plan_analysis_items` 表。
        10. 更新 `marketing_plans` 表中的對應紀錄，狀態改為 `completed`。
        11. 在整個流程中使用 `try...catch...finally` 結構，若發生錯誤，需更新狀態為 `failed` 並記錄錯誤訊息，且無論成功或失敗，最後都必須刪除臨時檔案。
2. **`GET /api/bdmin/marketing-plans`**: 獲取所有已上傳的企劃列表，按時間倒序排列。
3. **`GET /api/bdmin/marketing-plans/:id`**: 獲取指定企劃的所有分析項目 (`plan_analysis_items`)。
4. **`PUT /api/bdmin/analysis-items/:id`**: 更新單一分析項目，可接收 `phase` 或 `isApproved` 欄位的更新。

4. 前端開發任務 **(Frontend Tasks)**

4.1. 在管理後台 **(`client/src/pages/admin-dashboard.tsx`)** 實作介面

1. 新增一個名為「行銷資料庫」的 `TabsTrigger` 和 `TabsContent`。
2. 檔案上傳區塊：
    - 建立一個包含 `<input type="file" />` 和「上傳」按鈕的卡片。
    - 實作檔案選取和上傳的邏輯，點擊按鈕後會呼叫 `POST /api/bdmin/marketing-plans`。
3. 企劃列表區塊：
    - 使用 `useQuery` 呼叫 `GET /api/bdmin/marketing-plans` 來獲取列表。
    - 用表格呈現所有企劃，欄位包含：檔案名稱、上傳時間、狀態 (用 Badge 元件顯示 處理中、已完成、失敗)。

    - 表格應支援點擊某一列來選取該企劃，並載入其詳細分析結果。
    - 實作輪詢機制，每 5 秒自動重新獲取一次列表，直到所有 `processing` 狀態的任務都完成為止。
4. 分析結果顯示與編輯區塊：
    - 當使用者點擊某個「已完成」的企劃時，呼叫 `GET /api/bdmin/marketing-plans/:id` 獲取其分析項目。
    - 將結果以「預熱期」、「活動期」、「回購期」三個群組卡片來呈現。
    - 每一項策略摘要旁都應有：
        - 一個 `Switch` 元件，用來切換 `isApproved` 狀態（呼叫 `PUT /api/bdmin/analysis-items/:id`）。
        - 一個 `Select` (下拉選單)，選項為「預熱期」、「活動期」、「回購期」，用來修正該項目的 `phase`（同樣呼叫上述 API）。

5. 完成的定義 **(Definition of Done)**

- [ ] 資料庫 `marketing_plans` 和 `plan_analysis_items` 兩個資料表已成功建立。
- [ ] 後端 API 皆已實作並能正確運作，包含檔案的臨時儲存與刪除。
- [ ] 前端後台已新增「行銷資料庫」頁面。
- [ ] 管理員可以成功上傳 PDF 檔案，並在列表中看到任務狀態從「處理中」變為「已完成」。
- [ ] 點擊已完成的企劃，可以正確顯示 AI 分析出的三階段策略。
- [ ] 管理員可以透過介面上的 UI 元件（下拉選單和開關）成功修改策略的階段和審核狀態，並將變動同步到資料庫。