

```
// index.js - V35.6 (商城現金流)
require('dotenv').config();
const line = require('@line/bot-sdk');
const express = require('express');
const { Pool } = require('pg');
const crypto = require('crypto');
const axios = require('axios');
const ImageKit = require("imagekit");
const app = express();
const PORT = process.env.PORT || 3000;

const config = {
  channelAccessToken: process.env.CHANNEL_ACCESS_TOKEN,
  channelSecret: process.env.CHANNEL_SECRET,
};
const client = new line.Client(config);
const pgPool = new Pool({
  connectionString: process.env.DATABASE_URL,
  ssl: { rejectUnauthorized: false }
});
const imagekit = new ImageKit({
  publicKey: process.env.IMAGEKIT_PUBLIC_KEY,
  privateKey: process.env.IMAGEKIT_PRIVATE_KEY,
  urlEndpoint: process.env.IMAGEKIT_URL_ENDPOINT
});
const TEACHER_PASSWORD = process.env.TEACHER_PASSWORD || '9527';
const SELF_URL = process.env.SELF_URL || 'https://你的部署網址/';
const TEACHER_ID = process.env.TEACHER_ID;
const ADMIN_USER_ID = process.env.ADMIN_USER_ID;
const STUDENT_RICH_MENU_ID = process.env.STUDENT_RICH_MENU_ID;
const TEACHER_RICH_MENU_ID = process.env.TEACHER_RICH_MENU_ID;
const ADMIN_RICH_MENU_ID = process.env.ADMIN_RICH_MENU_ID;
const CONSTANTS = {
  TIME: {
    ONE_DAY_IN_MS: 86400000,
    EIGHT_HOURS_IN_MS: 28800000,
    ONE_HOUR_IN_MS: 3600000,
  },
  INTERVALS: {
    PING_INTERVAL_MS: 1000 * 60 * 5,
    CONVERSATION_TIMEOUT_MS: 1000 * 60 * 5,
    NOTIFICATION_CACHE_DURATION_MS: 1000 * 30,
    SESSION_TIMEOUT_MS: 1000 * 60 * 5, // [V28.0 新增] 對話階段超時時間 (5分鐘)
```

```
},
PAGINATION_SIZE: 9,
PURCHASE_PLANS: [
  { points: 5, amount: 500, label: '5 點 (500元)' },
  { points: 10, amount: 1000, label: '10 點 (1000元)' },
  { points: 20, amount: 2000, label: '20 點 (2000元)' },
  { points: 30, amount: 3000, label: '30 點 (3000元)' },
  { points: 50, amount: 5000, label: '50 點 (5000元)' },
],
BANK_INFO: {
  accountName: process.env.BANK_ACCOUNT_NAME,
  bankName: process.env.BANK_NAME,
  accountNumber: process.env.BANK_ACCOUNT_NUMBER,
},
COMMANDS: {
  GENERAL: {
    CANCEL: '❌ 取消操作'
  },
  ADMIN: {
    PANEL: '@管理模式',
    SYSTEM_STATUS: '@系統狀態',
    FAILED_TASK_MANAGEMENT: '@失敗任務管理',
    ADD_TEACHER: '@授權老師',
    REMOVE_TEACHER: '@移除老師',
    SIMULATE_STUDENT: '@模擬學員身份',
    SIMULATE_TEACHER: '@模擬老師身份',
    CONFIRM_ADD_TEACHER: '✅ 確認授權',
    CONFIRM_REMOVE_TEACHER: '✅ 確認移除',
    TOGGLE_NOTIFICATIONS: '@切換推播通知'
  },
  TEACHER: {
    COURSE_MANAGEMENT: '@課程管理',
    ADD_COURSE_SERIES: '@新增課程系列',
    MANAGE_OPEN_COURSES: '@管理已開課程',
    COURSE_INQUIRY: '@課程查詢',
    POINT_MANAGEMENT: '@點數管理',
    PENDING_POINT_ORDERS: '@待確認點數訂單',
    MANUAL_ADJUST_POINTS: '@手動調整點數',
    // [新增]
    VIEW_PURCHASE_HISTORY: '@查詢購點紀錄',
    STUDENT_MANAGEMENT: '@學員管理',
    SEARCH_STUDENT: '@查詢學員',
    VIEW_MESSAGES: '@查看未回覆留言',
  }
}
```

```
// [修改]
MESSAGE_SEARCH: '@查詢歷史留言',
ANNOUNCEMENT_MANAGEMENT: '@公告管理',
ADD_ANNOUNCEMENT: '@頒佈新公告',
DELETE_ANNOUNCEMENT: '@刪除舊公告',
SHOP_MANAGEMENT: '@商城管理',
ADD_PRODUCT: '@上架新商品',
VIEW_PRODUCTS: '@商品管理',
MANAGE_AVAILABLE_PRODUCTS: '@管理販售中商品',
MANAGE_UNAVAILABLE_PRODUCTS: '@管理已下架商品',
SHOP_ORDER_MANAGEMENT: '@訂單管理',
// [新增]
VIEW_SHOP_EXCHANGE_HISTORY: '@查詢兌換紀錄',
REPORT: '@統計報表',
COURSE_REPORT: '@課程報表',
ORDER_REPORT: '@訂單報表',
POINT_REPORT: '@點數報表',
ADD_COURSE: '@新增課程',
CANCEL_COURSE: '@取消課程',
COURSE_LIST: '@課程列表',
PENDING_ORDERS: '@待確認清單',
CONFIRM_MANUAL_ADJUST: '✅ 確認調整',
ADD_POINTS: '+ 加點',
DEDUCT_POINTS: '- 扣點',
// [修改]
MESSAGE_SEARCH: '@查詢歷史留言',
CONFIRM_ADD_ANNOUNCEMENT: '✅ 確認頒佈',
CONFIRM_DELETE_ANNOUNCEMENT: '✅ 確認刪除',
CONFIRM_BATCH_CANCEL: '✅ 確認批次取消',
CONFIRM_SINGLE_CANCEL: '✅ 確認取消單堂'
},
STUDENT: {
BOOK_COURSE: '@預約課程',
MY_COURSES: '@我的課程',
SHOP: '@活動商城',
POINTS: '@點數查詢',
LATEST_ANNOUNCEMENT: '@最新公告',
CONTACT_US: '@聯絡我們',
VIEW_SHOP_PRODUCTS: '@瀏覽商品',
EXCHANGE_HISTORY: '@兌換紀錄',
CHECK_POINTS: '@查看剩餘點數',
BUY_POINTS: '@購買點數',
PURCHASE_HISTORY: '@購點紀錄',
```

```

    ADD_NEW_MESSAGE: '@新增一則留言',
    CANCEL_BOOKING: '@取消預約',
    CANCEL_WAITING: '@取消候補',
    CONFIRM_ADD_COURSE: '確認新增課程',
    CANCEL_ADD_COURSE: '取消新增課程',
    RETURN_POINTS_MENU: '返回點數管理',
    CONFIRM_BUY_POINTS: '✓ 確認購買',
    INPUT_LAST5_CARD_TRIGGER: '@輸入匯款後五碼',
    EDIT_LAST5_CARD_TRIGGER: '@修改匯款後五碼',
    CONFIRM_BOOKING: '✓ 確認預約',
    CONFIRM_CANCEL_BOOKING: '✓ 確認取消預約',
    CONFIRM_CANCEL_WAITING: '✓ 確認取消候補',
  }
}
};
// =====
// [V31.3 新增] 通用快取工具
// =====
const simpleCache = {
  _cache: new Map(),

  /**
   * 設定一筆快取資料
   * @param {string} key - 快取的鍵
   * @param {*} value - 要快取的值
   * @param {number} ttlMs - 快取的存活時間 (毫秒)
   */
  set(key, value, ttlMs) {
    const expires = Date.now() + ttlMs;
    this._cache.set(key, { value, expires });
  },

  /**
   * 讀取一筆快取資料
   * @param {string} key - 快取的鍵
   * @returns {*} - 如果快取存在且未過期, 則回傳其值, 否則回傳 null
   */
  get(key) {
    const entry = this._cache.get(key);
    // 檢查是否存在, 且尚未過期
    if (entry && Date.now() < entry.expires) {
      return entry.value;
    }
  }
}

```

```

// 如果已過期, 可以順便清除它 (可選)
if (entry) {
  this._cache.delete(key);
}
return null;
},

/**
 * 清除一筆指定的快取
 * @param {string} key - 快取的鍵
 */
clear(key) {
  this._cache.delete(key);
}
};

/**
 * 檢查所有必要的環境變數是否已設定。
 * 如果有任何缺少的變數, 將記錄錯誤並終止應用程式。
 */
function checkEnvironmentVariables() {
  const requiredEnvVars = [
    'CHANNEL_ACCESS_TOKEN',
    'CHANNEL_SECRET',
    'DATABASE_URL',
    'ADMIN_USER_ID',
    'TEACHER_ID',
    'STUDENT_RICH_MENU_ID',
    'TEACHER_RICH_MENU_ID',
    'SELF_URL',
    'BANK_NAME',
    'BANK_ACCOUNT_NAME',
    'BANK_ACCOUNT_NUMBER',
    'IMAGEKIT_PUBLIC_KEY',
    'IMAGEKIT_PRIVATE_KEY',
    'IMAGEKIT_URL_ENDPOINT'
  ];
  const missingVars = requiredEnvVars.filter(varName => !process.env[varName]);

  if (missingVars.length > 0) {
    console.error('❌ FATAL ERROR: 缺少必要的環境變數:');
    missingVars.forEach(varName => console.error(` - ${varName}`));
    console.error('請檢查您的 .env 檔案或部署設定。');
  }
}

```

```

    process.exit(1);
}

console.log('✅ 所有必要的環境變數都已設定。');
}
/**
 * 創建一個包含分頁按鈕的 Flex Message 氣泡。
 * @param {string} baseAction - Postback 的基本動作字串，例如 'action=view_courses'。
 * @param {number} currentPage - 當前頁碼。
 * @param {boolean} hasNext - 是否有下一頁。
 * @param {string} [customParams=""] - 要附加到 postback data 的額外參數，例如
'&query=yoga'。
 * @returns {object|null} - Flex Message 氣泡物件，如果不需要分頁則返回 null。
 */
function createPaginationBubble(baseAction, currentPage, hasNext, customParams = "") {
    const buttons = [];
    if (currentPage > 1) {
        buttons.push({
            type: 'button',
            style: 'link',
            height: 'sm',
            action: {
                type: 'postback',
                label: '⬅️ 上一頁',
                data: `${baseAction}&page=${currentPage - 1}${customParams}`
            }
        });
    }

    if (hasNext) {
        buttons.push({
            type: 'button',
            style: 'link',
            height: 'sm',
            action: {
                type: 'postback',
                label: '下一頁 ➡️',
                data: `${baseAction}&page=${currentPage + 1}${customParams}`
            }
        });
    }

    if (buttons.length === 0) return null;

```

```

return {
  type: 'bubble',
  body: {
    type: 'box',
    layout: 'vertical',
    spacing: 'sm',
    contents: buttons,
    justifyContent: 'center',
    alignItems: 'center',
    paddingAll: 'md'
  },
};
}
/**
 * [V29.1 新增] 建立一個通用的、包含分頁功能的 Flex Carousel 訊息。
 * @param {object} options - 設定物件。
 * @param {string} options.altText - Flex Message 的替代文字。
 * @param {string} options.baseAction - Postback 的基本動作字串, 例如 'action=view_history'。
 * @param {number} options.page - 當前頁碼。
 * @param {string} options.dataQuery - 要執行的 SQL 查詢, 必須包含 LIMIT 和 OFFSET 的參數位置 (例如 $2, $3)。
 * @param {Array<any>} options.queryParams - SQL 查詢的參數陣列 (不含 LIMIT 和 OFFSET 的值)。
 * @param {function(object): object} options.mapRowToBubble - 一個將資料庫 row 轉換為 Flex Bubble 物件的函式。
 * @param {string} options.noDataMessage - 當第一頁沒有任何資料時顯示的文字訊息。
 * @param {string} [options.customParams=""] - (可選) 要附加到 postback data 的額外參數。
 * @returns {Promise<object|string>} - Flex Message 物件或無資料時的文字訊息。
 */
async function createPaginatedCarousel(options) {
  const {
    altText,
    baseAction,
    page,
    dataQuery,
    queryParams,
    mapRowToBubble,
    noDataMessage,
    customParams = ""
  } = options;
  const offset = (page - 1) * CONSTANTS.PAGINATION_SIZE;

  return executeDbQuery(async (client) => {

```

```

// 組合查詢參數, 將分頁參數加在最後
const finalQueryParams = [...queryParams, CONSTANTS.PAGINATION_SIZE + 1, offset];
const res = await client.query(dataQuery, finalQueryParams);

const hasNextPage = res.rows.length > CONSTANTS.PAGINATION_SIZE;
const pageRows = hasNextPage ? res.rows.slice(0, CONSTANTS.PAGINATION_SIZE) :
res.rows;

if (pageRows.length === 0 && page === 1) {
  return noDataMessage;
}
if (pageRows.length === 0) {
  return '沒有更多資料了。';
}

const bubbles = pageRows.map(mapRowToBubble);

const paginationBubble = createPaginationBubble(baseAction, page, hasNextPage,
customParams);
if (paginationBubble) {
  bubbles.push(paginationBubble);
}

return {
  type: 'flex',
  altText: altText,
  contents: {
    type: 'carousel',
    contents: bubbles
  }
};
});
}

/**
 * [V30 新增] 執行一個需要資料庫客戶端的操作, 並自動管理連線的開啟與關閉。
 * @param {function(object): Promise<any>} callback - 要執行的函式, 會接收一個 db client 作
為參數。
 * @returns {Promise<any>} - 回傳 callback 函式的執行結果。
 */
async function executeDbQuery(callback) {
  const client = await pgPool.connect();
  try {

```



```

    return await callback(client);
  } finally {
    if (client) client.release();
  }
}
/**
 * [V31.3 重構] 使用通用快取工具來讀取推播設定
 */
async function getNotificationStatus() {
  const cacheKey = 'notifications_enabled';
  const ttl = CONSTANTS.INTERVALS.NOTIFICATION_CACHE_DURATION_MS;
  // 步驟 1: 嘗試從快取中讀取
  const cachedStatus = simpleCache.get(cacheKey);
  if (cachedStatus !== null) {
    // 快取命中, 直接回傳
    return cachedStatus;
  }

  // 步驟 2: 快取未命中, 從資料庫讀取
  try {
    let isEnabled = true;
    // 預設值為 true, 以防資料庫查詢失敗時卡住所有通知
    await executeDbQuery(async (db) => {
      const res = await db.query("SELECT setting_value FROM system_settings WHERE
setting_key = 'notifications_enabled'");
      if (res.rows.length > 0) {
        isEnabled = res.rows[0].setting_value === 'true';
      }
    });
    // 步驟 3: 將從資料庫讀取到的新值寫入快取
    simpleCache.set(cacheKey, isEnabled, ttl);
    return isEnabled;
  } catch (err) {
    console.error('✖ 讀取推播設定失敗:', err);
    // 在發生錯誤時回傳一個安全的預設值
    return true;
  }
}

/**
 * [V24.0 新增] 將一個推播任務加入到資料庫佇列中
 * @param {string} recipientId - 收件人 User ID
 * @param {object|object[]} message - LINE 訊息物件或物件陣列

```

```

* @param {Date} [sendAt=null] - 預計發送時間, 若為 null 則立即發送
*/
async function enqueuePushTask(recipientId, message, sendAt = null) {
  const isSystemRecipient = [TEACHER_ID, ADMIN_USER_ID].includes(recipientId);
  if (isSystemRecipient) {
    const notificationsEnabled = await getNotificationStatus();
    if (!notificationsEnabled) {
      console.log(`[DEV MODE] 系統推播功能已關閉, 已阻擋傳送給 ${recipientId} 的通知。`);
      return;
    }
  }
}

try {
  await executeDbQuery(async (db) => {
    const messagePayload = Array.isArray(message) ? message : [message];
    const validMessages = messagePayload.filter(m => typeof m === 'object' && m !== null &&
m.type);
    if (validMessages.length === 0) {
      console.error(`[enqueuePushTask] 嘗試為 ${recipientId} 加入無效的訊息 payload`,
message);
      return;
    }

    const sendTimestamp = sendAt instanceof Date ? sendAt.toISOString() : new
Date().toISOString();

    await db.query(
      `INSERT INTO tasks (recipient_id, message_payload, send_at) VALUES ($1, $2, $3)`,
      [recipientId, JSON.stringify(validMessages), sendTimestamp]
    );
  });
} catch (err) {
  console.error(`❌ enqueuePushTask 寫入任務失敗 for ${recipientId}:`, err);
}
}
/**
* [V31.1 新增] 將多個推播任務批次加入到資料庫佇列中
* @param {Array<object>} tasks - 任務物件的陣列, 每個物件應包含 { recipientId: string,
message: object|object[] }
*/
async function enqueueBatchPushTasks(tasks) {
  if (!tasks || tasks.length === 0) {
    return;
  }

```

```

}

// 與單一任務函式一樣，檢查系統推播設定
const systemRecipients = [TEACHER_ID, ADMIN_USER_ID];
let tasksToEnqueue = tasks;
// 只有在任務列表中包含系統管理員/老師時，才需要檢查推播開關
if (tasks.some(t => systemRecipients.includes(t.recipientId))) {
  const notificationsEnabled = await getNotificationStatus();
  if (!notificationsEnabled) {
    console.log(`[DEV MODE] 系統推播功能已關閉，已過濾掉傳送給老師/管理員的批次通知。`);
    tasksToEnqueue = tasks.filter(t => !systemRecipients.includes(t.recipientId));
    if (tasksToEnqueue.length === 0) return; // 如果過濾後沒有任務了，就直接返回
  }
}

try {
  const recipientIds = [];
  const messagePayloads = [];
  const sendTimestamps = [];
  const now = new Date().toISOString();
  tasksToEnqueue.forEach(task => {
    const messagePayload = Array.isArray(task.message) ? task.message : [task.message];
    const validMessages = messagePayload.filter(m => typeof m === 'object' && m !== null && m.type);
    if (validMessages.length > 0) {
      recipientIds.push(task.recipientId);
      messagePayloads.push(JSON.stringify(validMessages));
      sendTimestamps.push(now); // 所有批次任務使用相同的時間戳
    } else {
      console.error(`[enqueueBatchPushTasks] 嘗試為 ${task.recipientId} 加入無效的訊息 payload`, task.message);
    }
  });

  if (recipientIds.length === 0) return;

  await executeDbQuery(async (db) => {
    // 使用 unnest 進行高效的批次插入
    await db.query(
      `INSERT INTO tasks (recipient_id, message_payload, send_at)
      SELECT * FROM unnest($1::text[], $2::jsonb[], $3::timestamp[])`,
      [recipientIds, messagePayloads, sendTimestamps]
    );
  });
}

```

```

    );
  });
} catch (err) {
  console.error(`❌ enqueueBatchPushTasks 批次寫入任務失敗:`, err);
}
}
/**
 * [V35.3 新增] 查詢所有老師並發送通知給他們
 * @param {object|object[]} message - 要發送的 LINE 訊息物件或陣列
 */
async function notifyAllTeachers(message) {
  try {
    const teachers = await executeDbQuery(async (db) => {
      const res = await db.query("SELECT id FROM users WHERE role = 'teacher'");
      return res.rows;
    });

    if (teachers.length === 0) {
      console.log('[Notify] 找不到任何老師可以發送通知。');
      return;
    }

    const notificationTasks = teachers.map(teacher => ({
      recipientId: teacher.id,
      message: message
    }));

    await enqueueBatchPushTasks(notificationTasks);
    console.log(`[Notify] 已成功將通知任務加入佇列，準備發送給 ${teachers.length} 位老師。`);

  } catch (err) {
    console.error(`❌ notifyAllTeachers 函式執行失敗:`, err);
  }
}

/**
 * [V24.0] 取消超過 24 小時未付款的訂單
 * [V31.1] 優化為批次處理通知任務
 */
async function cancelExpiredPendingOrders() {
  try {
    await executeDbQuery(async (client) => {

```

```

    const twentyFourHoursAgo = new Date(Date.now() - 24 *
CONSTANTS.TIME.ONE_HOUR_IN_MS);
    const res = await client.query(
      "DELETE FROM orders WHERE status = 'pending_payment' AND timestamp < $1
RETURNING user_id, order_id, user_name",
      [twentyFourHoursAgo]
    );

    if (res.rows.length > 0) {
      console.log(`🧹 已自動取消 ${res.rows.length} 筆逾時訂單。`);

      // 步驟 1: 使用 .map() 準備所有要發送的通知任務
      const notificationTasks = res.rows.map(order => {
        const message = {
          type: 'text',
          text: `訂單取消通知:\n您的訂單 (ID: ...${order.order_id.slice(-6)}) 因超過24小時
未完成付款, 系統已自動為您取消。如有需要請重新購買, 謝謝。`
        };
        return {
          recipientId: order.user_id,
          message: message
        };
      });

      // 步驟 2: 一次性將所有任務加入佇列
      await enqueueBatchPushTasks(notificationTasks).catch(e => {
        console.error(`將批次逾時訂單取消通知加入佇列時失敗`);
      });
    }
  });
} catch (err) {
  console.error("❌ 自動取消逾時訂單時發生錯誤:", err);
}
}

/**
 * [V28.0 新增] 智慧回覆機制: 取得使用者的待辦事項通知
 * @param {object} user - 使用者物件, 包含 id 和 role
 * @returns {Promise<object>} - 一個包含待辦事項計數的物件
 */
async function getPendingNotificationsForUser(user) {
  const notifications = {};
  try {

```

```

    await executeDbQuery(async (client) => {
      if (user.role === 'teacher') {
        const [newMessages, pendingPointOrders, pendingShopOrders] = await Promise.all([
          client.query("SELECT COUNT(*) FROM feedback_messages WHERE status =
'new'"),
          client.query("SELECT COUNT(*) FROM orders WHERE status =
'pending_confirmation'"),
          client.query("SELECT COUNT(*) FROM product_orders WHERE status =
'pending'")
        ]);
        notifications.newMessages = parseInt(newMessages.rows[0].count, 10);
        notifications.pendingPointOrders = parseInt(pendingPointOrders.rows[0].count, 10);
        notifications.pendingShopOrders = parseInt(pendingShopOrders.rows[0].count, 10);
      } else if (user.role === 'admin') {
        const failedTasks = await client.query("SELECT COUNT(*) FROM failed_tasks");
        notifications.failedTasks = parseInt(failedTasks.rows[0].count, 10);
      } else if (user.role === 'student') {
        const unreadReplies = await client.query("SELECT COUNT(*) FROM
feedback_messages WHERE user_id = $1 AND status = 'replied' AND is_student_read = false",
[user.id]);
        notifications.unreadReplies = parseInt(unreadReplies.rows[0].count, 10);
      }
    });
  } catch (error) {
    console.error(`[getPendingNotifications] 查詢使用者 ${user.id} 的通知時發生錯誤:`, error);
  }
  return notifications;
}

```

// --- 資料庫輔助函式 (Database Helper Functions) ---

```

/**
 * [V33.0 新增] 執行一個資料庫查詢，並自動管理連線。
 * 此函式支援傳入一個已存在的 client (用於交易)，或自動建立新連線。
 * @param {function(object): Promise<any>} queryCallback - 要執行的查詢函式，會接收 db
client 作為參數。
 * @param {object} [existingClient=null] - (可選) 一個已經存在的 pg client。
 * @returns {Promise<any>} - 回傳 queryCallback 的執行結果。
 */
async function executeDbQuery(queryCallback, existingClient = null) {
  // 如果沒有傳入現有的 client，則自己建立一個
  const client = existingClient || await pgPool.connect();
  try {

```

```

    // 執行傳入的查詢邏輯
    return await queryCallback(client);
  } finally {
    // 只有在 client 是這個函式自己建立的情況下, 才釋放它
    if (!existingClient && client) {
      client.release();
    }
  }
}

async function generateUniqueCoursePrefix(dbClient) {
  return executeDbQuery(async (client) => {
    let prefix, isUnique = false;
    while (!isUnique) {
      const randomChar1 = String.fromCharCode(65 + Math.floor(Math.random() * 26));
      const randomChar2 = String.fromCharCode(65 + Math.floor(Math.random() * 26));
      prefix = `${randomChar1}${randomChar2}`;
      const res = await client.query('SELECT id FROM courses WHERE id LIKE $1',
[`${prefix}%`]);
      if (res.rows.length === 0) isUnique = true;
    }
    return prefix;
  }, dbClient);
}

async function getUser(userId, dbClient) {
  return executeDbQuery(async (client) => {
    const res = await client.query('SELECT * FROM users WHERE id = $1', [userId]);
    if (res.rows.length === 0) return null;
    const userData = res.rows[0];
    if (userData && typeof userData.history === 'string') {
      try { userData.history = JSON.parse(userData.history); } catch (e) { userData.history = []; }
    }
    return userData;
  }, dbClient);
}

async function saveUser(user, dbClient) {
  return executeDbQuery(async (client) => {
    const historyJson = JSON.stringify(user.history || []);
    await client.query(
      `INSERT INTO users (id, name, points, role, history, last_seen_announcement_id,
picture_url, approved_by) VALUES ($1, $2, $3, $4, $5, $6, $7, $8)

```

```

        ON CONFLICT (id) DO UPDATE SET name = $2, points = $3, role = $4, history = $5,
last_seen_announcement_id = $6, picture_url = $7, approved_by = $8`,
        [user.id, user.name, user.points, user.role, historyJson,
user.last_seen_announcement_id || 0, user.picture_url || null, user.approved_by || null]
    );
}, dbClient);
}

```

```

async function getCourse(courseId, dbClient) {
    return executeDbQuery(async (client) => {
        const res = await client.query('SELECT * FROM courses WHERE id = $1', [courseId]);
        if (res.rows.length === 0) return null;

```

```

        const row = res.rows[0];
        // [V42.1 修正] 確保回傳的課程物件包含 teacher_id
        return {
            id: row.id,
            title: row.title,
            time: row.time.toISOString(),
            capacity: row.capacity,
            points_cost: row.points_cost,
            students: row.students || [],
            waiting: row.waiting || [],
            teacher_id: row.teacher_id
        };

```

```

    }, dbClient);
}

```

```

async function saveCourse(course, dbClient) {
    return executeDbQuery(async (client) => {
        // [V35.0 修改] 新增 teacher_id 欄位
        await client.query(
            `INSERT INTO courses (id, title, time, capacity, points_cost, students, waiting,
teacher_id)
            VALUES ($1, $2, $3, $4, $5, $6, $7, $8)
            ON CONFLICT (id) DO UPDATE SET title = $2, time = $3, capacity = $4, points_cost =
$5, students = $6, waiting = $7, teacher_id = $8`,
            [course.id, course.title, course.time, course.capacity, course.points_cost,
course.students, course.waiting, course.teacher_id]
        );
    }, dbClient);
}

```

```

async function deleteCourse(courseId, dbClient) {

```



```

    return executeDbQuery(async (client) => {
      await client.query('DELETE FROM courses WHERE id = $1', [courseId]);
    }, dbClient);
  }

  async function deleteCoursesByPrefix(prefix, dbClient) {
    return executeDbQuery(async (client) => {
      const coursesToDeleteRes = await client.query('SELECT id, title, time, points_cost,
students, waiting FROM courses WHERE id LIKE $1', [`${prefix}%`]);
      const coursesToDelete = coursesToDeleteRes.rows.map(row => ({
        id: row.id,
        title: row.title,
        time: row.time.toISOString(),
        points_cost: row.points_cost,
        students: row.students || [],
        waiting: row.waiting || []
      }));
      if (coursesToDelete.length > 0) {
        await client.query('DELETE FROM courses WHERE id LIKE $1', [`${prefix}%`]);
      }
      return coursesToDelete;
    }, dbClient);
  }

  async function getProduct(productId, dbClient) {
    return executeDbQuery(async (client) => {
      const res = await client.query('SELECT * FROM products WHERE id = $1', [productId]);
      return res.rows.length > 0 ? res.rows[0] : null;
    }, dbClient);
  }

  async function saveProduct(product, dbClient) {
    return executeDbQuery(async (client) => {
      await client.query(
        `UPDATE products SET name = $1, description = $2, price = $3, image_url = $4,
inventory = $5, status = $6 WHERE id = $7`,
        [product.name, product.description, product.price, product.image_url, product.inventory,
product.status, product.id]
      );
    }, dbClient);
  }

  async function getProductOrder(orderUID, dbClient) {
    return executeDbQuery(async (client) => {

```

```

    const res = await client.query('SELECT * FROM product_orders WHERE order_uid = $1',
[orderUID]);
    return res.rows.length > 0 ? res.rows[0] : null;
  }, dbClient);
}

```

```

async function saveProductOrder(order, dbClient) {
  return executeDbQuery(async (client) => {
    await client.query(
      `UPDATE product_orders SET status = $1, updated_at = $2, teacher_notes = $3
WHERE id = $4`,
      [order.status, order.updated_at, order.teacher_notes, order.id]
    );
  }, dbClient);
}

```

```

async function saveOrder(order, dbClient) {
  return executeDbQuery(async (client) => {
    await client.query(
      `INSERT INTO orders (order_id, user_id, user_name, points, amount, last_5_digits,
status, timestamp) VALUES ($1, $2, $3, $4, $5, $6, $7, $8)
      ON CONFLICT (order_id) DO UPDATE SET user_id = $2, user_name = $3, points =
$4, amount = $5, last_5_digits = $6, status = $7, timestamp = $8`,
      [order.order_id, order.user_id, order.user_name, order.points, order.amount,
order.last_5_digits, order.status, order.timestamp]
    );
  }, dbClient);
}

```

```

async function deleteOrder(orderId, dbClient) {
  return executeDbQuery(async (client) => {
    await client.query('DELETE FROM orders WHERE order_id = $1', [orderId]);
  }, dbClient);
}

```

```

async function cleanCoursesDB() {
  try {
    await executeDbQuery(async (client) => {
      const now = new Date();
      // 刪除一天前的課程
      const pastDate = new Date(now.getTime() - CONSTANTS.TIME.ONE_DAY_IN_MS);

      const result = await client.query(`DELETE FROM courses WHERE time < $1`,

```

```
[pastDate]);
```

```
    if (result.rowCount > 0) {
        console.log(`🧹 定期清理: 已成功移除 ${result.rowCount} 筆過期的課程。`);
    }
});
} catch (err) {
    console.error('❌ 定期清理過期課程時發生錯誤:', err);
}
}

/**
 * [V27.6 新增] 共用的錯誤處理函式
 * @param {Error} error - 捕獲到的錯誤物件
 * @param {string} replyToken - 用於回覆的 token
 * @param {string} context - 錯誤發生的情境, 例如 "查詢我的課程"
 */
async function handleError(error, replyToken, context = '未知操作') {
    console.error('❌ 在執行 [${context}] 時發生錯誤:', error.stack);
    try {
        if (replyToken) {
            await reply(replyToken, `抱歉, 在執行 ${context} 時發生了預期外的錯誤, 請稍後再試。`);
        }
    } catch (replyError) {
        console.error('❌ 連錯誤回覆都失敗了:', replyError.message);
    }
}

/**
 * [V31.2 新增] 將不同格式的內容轉換為 LINE 訊息物件陣列。
 * @param {string|object|Array<string|object>} content - 要發送的內容。
 * @returns {Array<object>} - 標準的 LINE 訊息物件陣列。
 */
function buildMessages(content) {
    const contentArray = Array.isArray(content) ? content : [content];

    return contentArray
        .filter(item => item !== null && item !== undefined) // 過濾掉無效內容
        .map(item => (typeof item === 'string' ? { type: 'text', text: item } : item));
}

/**
 * [V31.2 新增] 將 Quick Reply 選單附加到訊息陣列的最後一則訊息上。

```

```

* @param {Array<object>} messages - 由 buildMessages 產生的訊息陣列。
* @param {Array<object>|null} menu - Quick Reply 的項目陣列。
* @returns {Array<object>} - 附加完 Quick Reply 的訊息陣列。
*/
function attachQuickReply(messages, menu) {
  if (!menu || !Array.isArray(menu) || menu.length === 0 || messages.length === 0) {
    return messages;
  }

  // 驗證並過濾有效的 Quick Reply 項目
  const validMenuItems = menu
    .slice(0, 13) // Quick Reply 最多支援 13 個項目
    .filter(item => item && item.type === 'action' && (item.action.type === 'message' ||
item.action.type === 'postback'));
  if (validMenuItems.length > 0) {
    const lastMessage = messages[messages.length - 1];
    if (!lastMessage.quickReply) {
      lastMessage.quickReply = { items: [] };
    }
    lastMessage.quickReply.items.push(...validMenuItems);
  }

  return messages;
}
/**
* [V31.2 重構] 透過組合輔助函式來回覆訊息，結構更清晰。
*/
async function reply(replyToken, content, menu = null) {
  // 步驟 1: 建立標準的訊息陣列
  let messages = buildMessages(content);
  // 步驟 2: 如果有選單，就附加 Quick Reply
  messages = attachQuickReply(messages, menu);
  // 如果最終沒有任何有效訊息，就直接返回，避免呼叫空的 API
  if (messages.length === 0) {
    console.log('[REPLY-DEBUG] 沒有有效的訊息可以發送，已取消操作。');
    return;
  }

  // 步驟 3: 執行 API 呼叫
  try {
    console.log(`[REPLY-DEBUG] 準備呼叫 client.replyMessage...`);
    const result = await client.replyMessage(replyToken, messages);
    console.log('[REPLY-DEBUG] client.replyMessage 呼叫已完成。');
  }

```

```

// API 錯誤的雙重檢查
if (result && result.response && result.response.status >= 400) {
    console.error('!! API 呼叫回傳了非成功的狀態碼 !!', JSON.stringify(result.response.data,
null, 2));
}

} catch (error) {
    console.error('!! 在 reply 的 CATCH 中捕捉到 API 錯誤 !!');
    if (error.originalError && error.originalError.response && error.originalError.response.data) {
        console.error('【LINE API 回應的詳細錯誤】:',
JSON.stringify(error.originalError.response.data, null, 2));
    } else {
        console.error('【捕獲到的基本錯誤訊息】:', error.message);
    }
    throw error;
}
}

function formatIdForDisplay(id) {
    if (!id || typeof id !== 'string') return id;
    const zws = '\u200B'; // Zero-width space
    return id.match(/.{1,8}/g)?.join(zws) || id;
}

function formatDateTime(isoString) {
    if (!isoString) return '無效時間';
    const date = new Date(isoString);
    const formatter = new Intl.DateTimeFormat('zh-TW', { month: '2-digit', day: '2-digit', weekday:
'short', hour: '2-digit', minute: '2-digit', hour12: false, timeZone: 'Asia/Taipei' });
    const parts = formatter.formatToParts(date);
    const month = parts.find(p => p.type === 'month').value;
    const day = parts.find(p => p.type === 'day').value;
    let weekday = parts.find(p => p.type === 'weekday').value;
    const hour = parts.find(p => p.type === 'hour').value;
    const minute = parts.find(p => p.type === 'minute').value;
    if (weekday.startsWith('週')) weekday = weekday.slice(-1);
    return `${month}-${day}(${weekday}) ${hour}:${minute}`;
}

/**
 * [V23.2 新增] 取得課程主標題, 移除 "- 第 x 堂"
 * @param {string} fullTitle - 完整的課程標題
 * @returns {string} - 主標題
 */

```

```
function getCourseMainTitle(fullTitle) {  
  if (typeof fullTitle !== 'string') return "";  
  return fullTitle.replace(/ - 第 \d+ 堂$/, "");  
}
```

```
function getNextDate(dayOfWeek, timeStr, startDate = new Date()) {  
  const [hours, minutes] = timeStr.split(':').map(Number);  
  const resultDate = new Date(startDate);  
  resultDate.setUTCHours(hours - 8, minutes, 0, 0);  
  let currentDay = resultDate.getUTCDay();  
  let daysToAdd = (dayOfWeek - currentDay + 7) % 7;  
  if (daysToAdd === 0 && resultDate.getTime() <= startDate.getTime()) daysToAdd = 7;  
  else if (resultDate.getTime() < startDate.getTime() && daysToAdd === 0) daysToAdd = 7;  
  resultDate.setUTCDate(resultDate.getUTCDate() + daysToAdd);  
  return resultDate;  
}
```

```
function getDateRange(period) {  
  const now = new Date(new Date().toLocaleString("en-US", { timeZone: "Asia/Taipei" }));  
  let startDate, endDate;  
  switch (period) {  
    case 'week':  
      const dayOfWeek = now.getDay();  
      // Sunday - 0, Monday - 1, ...  
      const diff = now.getDate() - dayOfWeek + (dayOfWeek === 0 ? -6 : 1);  
      // Adjust to make Monday the first day  
      startDate = new Date(now.setDate(diff));  
      endDate = new Date(startDate);  
      endDate.setDate(startDate.getDate() + 6);  
      break;  
    case 'month':  
      startDate = new Date(now.getFullYear(), now.getMonth(), 1);  
      endDate = new Date(now.getFullYear(), now.getMonth() + 1, 0);  
      break;  
    case 'quarter':  
      const quarter = Math.floor(now.getMonth() / 3);  
      startDate = new Date(now.getFullYear(), quarter * 3, 1);  
      endDate = new Date(now.getFullYear(), quarter * 3 + 3, 0);  
      break;  
    case 'year':  
      startDate = new Date(now.getFullYear(), 0, 1);  
      endDate = new Date(now.getFullYear(), 11, 31);  
      break;
```

```

    }

    startDate.setHours(0, 0, 0, 0);
    endDate.setHours(23, 59, 59, 999);
    return {
      start: new Date(startDate.getTime() -
CONSTANTS.TIME.EIGHT_HOURS_IN_MS).toISOString(),
      end: new Date(endDate.getTime() -
CONSTANTS.TIME.EIGHT_HOURS_IN_MS).toISOString()
    };
  }
}

function levenshtein(a, b) {
  const matrix = Array(b.length + 1).fill(null).map(() => Array(a.length + 1).fill(null));
  for (let i = 0; i <= a.length; i += 1) { matrix[0][i] = i;
  }
  for (let j = 0; j <= b.length; j += 1) { matrix[j][0] = j;
  }
  for (let j = 1; j <= b.length; j += 1) {
    for (let i = 1; i <= a.length; i += 1) {
      const indicator = a[i - 1] === b[j - 1] ? 0 : 1;
      matrix[j][i] = Math.min(
        matrix[j][i - 1] + 1,
        matrix[j - 1][i] + 1,
        matrix[j - 1][i - 1] + indicator,
      );
    }
  }
  return matrix[b.length][a.length];
}

function findClosestCommand(userInput, role) {
  const upperCaseRole = role.toUpperCase();
  if (!CONSTANTS.COMMANDS[upperCaseRole]) return null;
  const commandList = Object.values(CONSTANTS.COMMANDS[upperCaseRole]);
  let bestMatch = null;
  let minDistance = Infinity;
  const threshold = Math.floor(userInput.length * 0.4);
  for (const command of commandList) {
    const distance = levenshtein(userInput, command);
    if (distance < minDistance && distance <= threshold) {
      minDistance = distance;
      bestMatch = command;
    }
  }
}

```

```

    }
    return bestMatch;
}
function buildBuyPointsFlex() {
    const plansContent = CONSTANTS.PURCHASE_PLANS.flatMap((plan, index) => {
        const planItems = [
            {
                type: 'box',
                layout: 'horizontal',
                contents: [
                    { type: 'text', text: `${plan.points} 點`, size: 'md', color: '#1A759F', flex: 3, gravity:
'center' },
                    { type: 'text', text: `售價: ${plan.amount} 元`, size: 'md', color: '#666666', align: 'end',
flex: 5, gravity: 'center' }
                ]
            },
            {
                type: 'button',
                action: { type: 'postback', label: '選擇此方案', data:
`action=select_purchase_plan&plan=${plan.points}`, displayText: `我要購買 ${plan.points} 點` },
                style: 'primary',
                height: 'sm',
                color: '#52B69A'
            }
        ];
        if (index < CONSTANTS.PURCHASE_PLANS.length - 1) {
            planItems.push({ type: 'separator', margin: 'md' });
        }
        return planItems;
    });

    return {
        type: 'flex',
        altText: '請選擇要購買的點數方案',
        contents: {
            type: 'bubble',
            header: {
                type: 'box',
                layout: 'vertical',
                contents: [{ type: 'text', text: '✚ 購買點數', weight: 'bold', size: 'lg', color: '#FFFFFF' }],
                backgroundColor: '#34A0A4',
                paddingAll: 'lg'
            },
        },
    },

```



```

      body: {
        type: 'box',
        layout: 'vertical',
        spacing: 'md',
        contents: [
          ...plansContent,
          { type: 'separator', margin: 'xl' },
          { type: 'text', text: '購買後請至「點數查詢」回報匯款資訊', size: 'xs', color: '#aaaaaa',
align: 'center', margin: 'md', wrap: true }
        ]
      }
    }
  };
}

```

// [V35.6 優化] 簡化點數查詢主頁, 移除待處理訂單資訊

```

async function buildPointsMenuFlex(userId) {
  const user = await getUser(userId);
  if (!user) return { type: 'text', text: '無法獲取您的使用者資料.' };

  // 移除查詢 pendingOrder 的邏輯, 直接顯示點數餘額
  const bodyContents = [{
    type: 'box',
    layout: 'vertical',
    margin: 'md',
    alignItems: 'center',
    contents: [
      { type: 'text', text: '目前剩餘點數', size: 'sm', color: '#AAAAAA' },
      { type: 'text', text: `${user.points} 點`, weight: 'bold', size: '3xl', margin: 'sm', color:
'#1A759F' },
    ]
  }];

  return {
    type: 'flex',
    altText: '點數查詢選單',
    contents: {
      type: 'bubble',
      size: 'giga',
      header: {
        type: 'box',
        layout: 'vertical',
        contents: [

```

```

        { type: 'text', text: '💎 點數查詢', color: '#ffffff', weight: 'bold', size: 'lg' }
    ],
    backgroundColor: '#34A0A4',
    paddingBottom: 'lg',
    paddingTop: 'lg'
},
body: {
    type: 'box',
    layout: 'vertical',
    paddingAll: 'xl',
    spacing: 'md',
    contents: bodyContents
},
footer: {
    type: 'box',
    layout: 'vertical',
    spacing: 'sm',
    contents: [
        {
            type: 'button', style: 'secondary', height: 'sm',
            action: { type: 'postback', label: '✚ 購買點數', data:
`action=run_command&text=${encodeURIComponent(CONSTANTS.COMMANDS.STUDENT.B
UY_POINTS)}` }
        },
        {
            type: 'button', style: 'secondary', height: 'sm',
            action: { type: 'postback', label: '📖 查詢購點紀錄', data:
`action=run_command&text=${encodeURIComponent(CONSTANTS.COMMANDS.STUDENT.P
URCHASE_HISTORY)}` }
        }
    ]
}
}
};
}

```

/**

* [V34.1 新增] 建立一個顯示老師個人資訊變更並請求確認的 Flex Message

* @param {string} userId - 使用者的 ID

* @param {object} newData - 一個包含待更新欄位和值的物件，例如 { name: '新名字' }

*/

```

async function buildProfileConfirmationMessage(userId, newData) {
    const fieldMap = { name: '姓名', bio: '簡介', image_url: '照片' };

```

```

const updatedFields = Object.keys(newData).map(key => fieldMap[key] || key).join('、');

const client = await pgPool.connect();
try {
  const res = await client.query('SELECT * FROM teachers WHERE line_user_id = $1',
[userId]);
  const currentProfile = res.rows[0] || { name: '新老師', bio: '尚未填寫簡介', image_url: null };
  const previewProfile = { ...currentProfile, ...newData };
  const placeholder_avatar = 'https://i.imgur.com/8l1Yd2S.png';

  return {
    type: 'flex',
    altText: `確認更新您的${updatedFields}`,
    contents: {
      type: 'bubble',
      header: { type: 'box', layout: 'vertical', contents: [{ type: 'text', text: `⚠ 請確認更新內容`, weight: 'bold', color: '#FFFFFF' }], backgroundColor: '#FFC107' },
      hero: { type: 'image', url: previewProfile.image_url || placeholder_avatar, size: 'full', aspectRatio: '1:1', aspectMode: 'cover' },
      body: {
        type: 'box', layout: 'vertical', paddingAll: 'lg', spacing: 'md',
        contents: [
          { type: 'text', text: previewProfile.name, weight: 'bold', size: 'xl' },
          { type: 'text', text: previewProfile.bio || '尚未填寫簡介', wrap: true, size: 'sm', color: '#666666' }
        ]
      },
      footer: {
        type: 'box', layout: 'vertical', spacing: 'sm', paddingAll: 'lg',
        contents: [
          { type: 'button', style: 'primary', color: '#28a745', action: { type: 'postback', label: '✅ 確認更新${updatedFields}', data: 'action=confirm_teacher_profile_update' } },
          { type: 'button', style: 'secondary', action: { type: 'message', label: '❌ 取消', text: CONSTANTS.COMMANDS.GENERAL.CANCEL } }
        ]
      }
    }
  };
} finally {
  if (client) client.release();
}
}

```

```

const WEEKDAYS = [
  { label: '週日', value: 0 }, { label: '週一', value: 1 }, { label: '週二', value: 2 },
  { label: '週三', value: 3 }, { label: '週四', value: 4 }, { label: '週五', value: 5 },
  { label: '週六', value: 6 },
];
// --- 對話狀態管理 ---
const pendingCourseCreation = {};
const pendingPurchase = {};
const pendingManualAdjust = {};
const sentReminders = {};
const pendingStudentSearchQuery = {};
const pendingBookingConfirmation = {};
const pendingFeedback = {};
const pendingReply = {};
const pendingMessageSearchQuery = {};
const pendingTeacherAddition = {};
const pendingTeacherRemoval = {};
const pendingProductCreation = {};
const pendingCourseCancellation = {};
const pendingTeacherProfileEdit = {};
// [V34.0 新增]
const pendingReportGeneration = {};
const pendingAnnouncementCreation = {};
const pendingAnnouncementDeletion = {};
const repliedTokens = new Set();
const pendingProductEdit = {};
const pendingInventoryAdjust = {};
const pendingManualAdjustSearch = {};
const userProfileCache = new Map();
const userLastInteraction = {}; // [V28.0 新增] 用於智慧回覆機制的 Session 追蹤
const pendingShopPayment = {}; // [V35.5 新增] 處理商城現金支付的對話狀態
// [新增] 查詢歷史紀錄的對話狀態
const pendingPurchaseHistorySearch = {};
const pendingExchangeHistorySearch = {};
const pendingMessageHistorySearch = {};
const cancellableConversationStates = {
  pendingCourseCreation,
  pendingManualAdjust,
  pendingStudentSearchQuery,
  pendingReply,
  pendingFeedback,
  pendingPurchase,
  pendingTeacherAddition,

```

```

    pendingTeacherRemoval,
    pendingAnnouncementCreation,
    pendingAnnouncementDeletion,
    pendingBookingConfirmation,
    pendingCourseCancellation,
    pendingProductCreation,
    pendingProductEdit,
    pendingInventoryAdjust,
    pendingTeacherProfileEdit,
    pendingMessageSearchQuery,
    pendingManualAdjustSearch,
    pendingShopPayment, // [V35.5 新增]
    pendingPurchaseHistorySearch,
    pendingExchangeHistorySearch,
    pendingMessageHistorySearch,
};
/**
 * 清除使用者所有待處理的對話狀態。
 * 用於「智慧取消」機制，當使用者點擊主選單或輸入新指令時，放棄先前的操作。
 * @param {string} userId - 使用者的 ID。
 * @returns {boolean} - 如果清除了任何狀態，則返回 true。
 */
function clearPendingConversations(userId) {
    let cleared = false;
    for (const state of Object.values(cancellableConversationStates)) {
        if (state[userId]) {
            if (state[userId].timeoutId) {
                clearTimeout(state[userId].timeoutId);
            }
            delete state[userId];
            cleared = true;
        }
    }
    return cleared;
}
/**
 * 產生一個包含取消按鈕的快速回覆選單。
 * @returns {Array} - 可用於 reply 函式的 menu 參數。
 */
function getCancelMenu() {
    return [{ type: 'action', action: { type: 'message', label:
CONSTANTS.COMMANDS.GENERAL.CANCEL, text:
CONSTANTS.COMMANDS.GENERAL.CANCEL } }];

```

```


}
function setupConversationTimeout(userId, conversationState, stateName, onTimeout) {
  if (conversationState[userId]?.timeoutId) {
    clearTimeout(conversationState[userId].timeoutId);
  }
  const timeoutId = setTimeout(() => {
    if (conversationState[userId]) {
      delete conversationState[userId];
      onTimeout(userId);
    }
  }, CONSTANTS.INTERVALS.CONVERSATION_TIMEOUT_MS);
  conversationState[userId] = { ...conversationState[userId], timeoutId };
}
async function handlePurchaseFlow(event, userId) {
  const text = event.message.text ? event.message.text.trim() : '';
  const user = await getUser(userId);
  const purchaseState = pendingPurchase[userId];

  if (!purchaseState) return { handled: false };
  let replyContent;

  switch (purchaseState.step) {
    case 'confirm_purchase':
      if (text === CONSTANTS.COMMANDS.STUDENT.CONFIRM_BUY_POINTS) {
        const order_id = `PO${Date.now()}`;
        const order = {
          order_id: order_id,
          user_id: userId,
          user_name: user.name,
          points: purchaseState.data.points,
          amount: purchaseState.data.amount,
          last_5_digits: null,
          status: 'pending_payment',
          timestamp: new Date().toISOString()
        };
        await saveOrder(order);
        delete pendingPurchase[userId];

        // [V35.6 優化] 更新提示文字以符合新流程
        const replyText = `感謝您的購買！訂單已成立 (ID: ${formatIdForDisplay(order_id)})。
        \n\n請匯款至以下帳戶：\n銀行：${CONSTANTS.BANK_INFO.bankName}\n戶名：
        ${CONSTANTS.BANK_INFO.accountName}\n帳號：
        ${CONSTANTS.BANK_INFO.accountNumber}\n金額：${order.amount} 元\n\n匯款完成後，請至

```

「點數查詢」→「查詢購點紀錄」，找到此筆待付款訂單，並點擊按鈕來回報您的後五碼。
 提醒：為確保您的權益，請於24小時內完成匯款與回報，逾時訂單將會自動取消。`;

```
// 因為主頁面已簡化，這裡不再需要回傳 flexMenu
replyContent = replyText;

} else {
  replyContent = '請點擊  確認購買 或  取消操作。';
}
return { handled: true, reply: replyContent };

case 'input_last5':
case 'edit_last5':
  if (/^\d{5}$/.test(text)) {
    const order_id = purchaseState.data.order_id;
    const wasSuccessful = await executeDbQuery(async (client) => {
      const orderRes = await client.query('SELECT * FROM orders WHERE order_id = $1', [order_id]);
      if (orderRes.rows.length > 0) {
        const order = orderRes.rows[0];
        order.last_5_digits = text;
        order.status = 'pending_confirmation';
        order.timestamp = new Date().toISOString();
        await saveOrder(order, client);
        return true;
      }
      return false;
    });
    delete pendingPurchase[userId];
    if (wasSuccessful) {
      const flexMenu = await buildPointsMenuFlex(userId);
      replyContent = [{type: 'text', text: `感謝您！已收到您的匯款後五碼「${text}」。  
我們將盡快為您審核，審核通過後點數將自動加入您的帳戶。`}, flexMenu];

      if (TEACHER_ID) {
        const notifyMessage = { type: 'text', text: `🔔 購點審核通知  
學員 ${user.name} 已提交匯款資訊。  
訂單ID: ${order_id}  
後五碼: ${text}  
請至「點數管理」->「待確認點數訂單」審核。`};
        await notifyAllTeachers(notifyMessage);
      }
    }
  } else {
    replyContent = '找不到您的訂單，請重新操作。';
  }
}
```

```

    } else {
      replyContent = {
        type: 'text',
        text: '格式錯誤, 請輸入5位數字的匯款帳號後五碼。',
        quickReply: { items: getCancelMenu() }
      };
    }
    return { handled: true, reply: replyContent };
  }
  return { handled: false };
}

```

// --- Teacher Command Handlers (V34.0 Refactor) ---

```

async function showCourseManagementMenu(event, user) {
  return {
    type: 'flex',
    altText: '課程與師資管理',
    contents: {
      type: 'bubble',
      size: 'giga',
      header: {
        type: 'box',
        layout: 'vertical',
        contents: [{ type: 'text', text: '📅 課程與師資管理', color: '#ffffff', weight: 'bold', size:
'lg']],
        backgroundColor: '#343A40',
        paddingTop: 'lg',
        paddingBottom: 'lg'
      },
      body: {
        type: 'box',
        layout: 'vertical',
        spacing: 'md',
        paddingAll: 'lg',
        contents: [
          { type: 'text', text: '課程功能', size: 'sm', color: '#888888', weight: 'bold' },
          { type: 'button', style: 'secondary', height: 'sm', action: { type: 'postback', label: '➕
新增課程系列', data:
`action=run_command&text=${encodeURIComponent(CONSTANTS.COMMANDS.TEACHER.ADD_COURSE_SERIES)} ` } },
          { type: 'button', style: 'secondary', height: 'sm', action: { type: 'postback', label: '🔍
課程狀態查詢', data:

```



```

`action=run_command&text=${encodeURIComponent(CONSTANTS.COMMANDS.TEACHER.
COURSE_INQUIRY))` } },
    { type: 'button', style: 'secondary', height: 'sm', action: { type: 'postback', label: '⚙️
管理已開課程', data:
`action=run_command&text=${encodeURIComponent(CONSTANTS.COMMANDS.TEACHER.
MANAGE_OPEN_COURSES))` } },
    { type: 'separator', margin: 'xl' },
    { type: 'text', text: '師資功能', size: 'sm', color: '#888888', weight: 'bold', margin: 'lg' },
    { type: 'button', style: 'secondary', height: 'sm', action: { type: 'postback', label: '📋
師資團隊', data: 'action=list_all_teachers&page=1' } },
    { type: 'button', style: 'secondary', height: 'sm', action: { type: 'postback', label: '👤
個人資訊', data: 'action=manage_personal_profile' } }
  ]
}
};
}

```

```

async function startAddCourseSeries(event, user) {
  const userId = user.id;
  pendingCourseCreation[userId] = { step: 'await_title' };
  setupConversationTimeout(userId, pendingCourseCreation, 'pendingCourseCreation', (u) =>
{
    const timeoutMessage = { type: 'text', text: '新增課程逾時，自動取消。'};
    enqueuePushTask(u, timeoutMessage).catch(e => console.error(e));
  });
  return {
    type: 'text',
    text: '請輸入新課程系列的標題(例如：高階空中瑜伽)，或按「取消」來放棄操作。',
    quickReply: { items: getCancelMenu() }
  };
}

```

```

async function showManageOpenCourses(event, user) {
  return showCourseSeries(1);
}

```

```

async function showCourseInquiry(event, user) {
  return showCourseRosterSummary(1);
}

```

```

async function showPointManagementMenu(event, user) {
  const pendingCount = await executeDbQuery(client =>

```

```

    client.query("SELECT COUNT(*) FROM orders WHERE status = 'pending_confirmation'")
).then(res => parseInt(res.rows[0].count, 10));

// 準備帶有計數的按鈕標籤文字
let pendingPointOrdersLabel = '✅ 待確認點數訂單';
if (pendingCount > 0) {
    pendingPointOrdersLabel = `✅ 待確認點數訂單 (${pendingCount})`;
}

return {
    type: 'flex',
    altText: '點數管理',
    contents: {
        type: 'bubble',
        size: 'giga',
        header: {
            type: 'box',
            layout: 'vertical',
            contents: [{ type: 'text', text: '💎 點數管理', color: '#ffffff', weight: 'bold', size: 'lg' }],
            backgroundColor: '#343A40',
            paddingTop: 'lg',
            paddingBottom: 'lg'
        },
        body: {
            type: 'box',
            layout: 'vertical',
            spacing: 'md',
            paddingAll: 'lg',
            contents: [
                // [V35.6 修正] 在按鈕 label 中使用動態變數
                { type: 'button', style: 'secondary', height: 'sm', action: { type: 'postback', label:
pendingPointOrdersLabel, data: `action=view_pending_orders_page&page=1` } },
                { type: 'button', style: 'secondary', height: 'sm', action: { type: 'postback', label: '✎
手動調整點數', data:
`action=run_command&text=${encodeURIComponent(CONSTANTS.COMMANDS.TEACHER.
MANUAL_ADJUST_POINTS)}` } },
                { type: 'button', style: 'secondary', height: 'sm', action: { type: 'postback', label: '📖
查詢購點紀錄', data: `action=select_purchase_history_view_type` } },
                { type: 'button', style: 'secondary', height: 'sm', action: { type: 'postback', label: '⚙️
查詢手動紀錄', data: `action=select_adjust_history_view_type` } }
            ]
        }
    }
}

```

```
};  
}
```

```
async function showPendingPointOrders(event, user) {  
  return showPendingOrders(1);  
}
```

```
async function showStudentManagementMenu(event, user) {  
  const unreadCount = await executeDbQuery(client =>  
    client.query("SELECT COUNT(*) FROM feedback_messages WHERE status = 'new'")  
  ).then(res => parseInt(res.rows[0].count, 10));  
  let unreadLabel = '查看未回覆留言';  
  if (unreadCount > 0) {  
    unreadLabel += ` (${unreadCount})`;  
  }  
}
```

```
return {  
  type: 'flex',  
  altText: '學員管理',  
  contents: {  
    type: 'bubble',  
    size: 'giga',  
    header: {  
      type: 'box',  
      layout: 'vertical',  
      contents: [{ type: 'text', text: '學員管理', color: '#ffffff', weight: 'bold', size: 'lg' }],  
      backgroundColor: '#343A40',  
      paddingTop: 'lg',  
      paddingBottom: 'lg'  
    },  
    body: {  
      type: 'box',  
      layout: 'vertical',  
      spacing: 'md',  
      paddingAll: 'lg',  
      contents: [  
        { type: 'button', style: 'secondary', height: 'sm', action: { type: 'postback', label: '查詢學員', data: {  
          `action=run_command&text=${encodeURIComponent(CONSTANTS.COMMANDS.TEACHER.SEARCH_STUDENT)}` } },  
        { type: 'button', style: 'secondary', height: 'sm', action: { type: 'postback', label: unreadLabel, data: {  
          `action=run_command&text=${encodeURIComponent(CONSTANTS.COMMANDS.TEACHER.V
```

```

IEW_MESSAGES))` } }},
    { type: 'button', style: 'secondary', height: 'sm', action: { type: 'postback', label: '📄
查詢歷史留言', data: `action=select_message_history_view_type` } }
  ]
}
}
};
}

```

```

async function startStudentSearch(event, user) {
  const userId = user.id;
  pendingStudentSearchQuery[userId] = {};
  setupConversationTimeout(userId, pendingStudentSearchQuery,
'pendingStudentSearchQuery', (u) => {
    if (pendingStudentSearchQuery[u]) {
      delete pendingStudentSearchQuery[u];
      const timeoutMessage = { type: 'text', text: '查詢學員逾時，自動取消。'};
      enqueuePushTask(u, timeoutMessage).catch(e => console.error(e));
    }
  });
  return {
    type: 'text',
    text: '請輸入您想查詢的學員姓名或 User ID:',
    quickReply: { items: getCancelMenu() }
  };
}

```

```

async function showUnreadTeacherMessages(event, user) {
  return showUnreadMessages(1);
}

```

```

async function startMessageSearch(event, user) {
  const userId = user.id;
  pendingMessageSearchQuery[userId] = {};
  setupConversationTimeout(userId, pendingMessageSearchQuery,
'pendingMessageSearchQuery', (u) => {
    if (pendingMessageSearchQuery[u]) {
      delete pendingMessageSearchQuery[u];
      const timeoutMessage = { type: 'text', text: '查詢歷史留言逾時，自動取消。'};
      enqueuePushTask(u, timeoutMessage).catch(e => console.error(e));
    }
  });
  return {

```

```

    type: 'text',
    text: '請輸入您想查詢的學員姓名或留言關鍵字:',
    quickReply: { items: getCancelMenu() }
  };
}

```

```

async function showAnnouncementManagementMenu(event, user) {
  return {
    type: 'flex',
    altText: '公告管理',
    contents: {
      type: 'bubble',
      size: 'giga',
      header: {
        type: 'box',
        layout: 'vertical',
        contents: [{ type: 'text', text: '📢 公告管理', color: '#ffffff', weight: 'bold', size: 'lg' }],
        backgroundColor: '#343A40',
        paddingTop: 'lg',
        paddingBottom: 'lg'
      },
      body: {
        type: 'box',
        layout: 'vertical',
        spacing: 'md',
        paddingAll: 'lg',
        contents: [
          { type: 'button', style: 'secondary', height: 'sm', action: { type: 'postback', label: '➕  
頒佈新公告', data: `action=run_command&text=${encodeURIComponent(CONSTANTS.COMMANDS.TEACHER.ADD_ANNOUNCEMENT)}` } },
          { type: 'button', style: 'secondary', height: 'sm', action: { type: 'postback', label: '🗑️  
刪除舊公告', data: `action=run_command&text=${encodeURIComponent(CONSTANTS.COMMANDS.TEACHER.DELETE_ANNOUNCEMENT)}` } }
        ]
      }
    }
  };
}

async function startAddAnnouncement(event, user) {
  const userId = user.id;
  pendingAnnouncementCreation[userId] = { step: 'await_content' };
}

```

```

    setupConversationTimeout(userId, pendingAnnouncementCreation,
    'pendingAnnouncementCreation', (u) => {
        const timeoutMessage = { type: 'text', text: '頒佈公告操作逾時, 自動取消。'};
        enqueuePushTask(u, timeoutMessage).catch(e => console.error(e));
    });
    return {
        type: 'text',
        text: '請輸入要頒佈的公告內容:',
        quickReply: { items: getCancelMenu() }
    };
}

async function showAnnouncementsForDeletionList(event, user) {
    return showAnnouncementsForDeletion(1);
}

async function showShopManagementMenu(event, user) {
    // [V35.5 修正] 更新查詢條件以計數所有待處理的現金/轉帳訂單
    const pendingShopOrdersCount = await executeDbQuery(client =>
        client.query("SELECT COUNT(*) FROM product_orders WHERE status IN
('pending_payment', 'pending_confirmation')")
    ).then(res => parseInt(res.rows[0].count, 10));

    let pendingShopOrdersLabel = '📋 查看待處理訂單';
    if (pendingShopOrdersCount > 0) {
        pendingShopOrdersLabel += ` (${pendingShopOrdersCount})`;
    }

    return {
        type: 'flex',
        altText: '商城管理',
        contents: {
            type: 'bubble',
            size: 'giga',
            header: {
                type: 'box',
                layout: 'vertical',
                contents: [ { type: 'text', text: '🛒 商城管理', weight: 'bold', size: 'lg', color: '#FFFFFF' }
            ],
            backgroundColor: '#343A40',
            paddingTop: 'lg',
            paddingBottom: 'lg'
        },
    },

```

```

    body: {
      type: 'box',
      layout: 'vertical',
      spacing: 'md',
      paddingAll: 'lg',
      contents: [
        { type: 'button', style: 'secondary', height: 'sm', action: { type: 'postback', label: '➕'
上架新商品', data:
`action=run_command&text=${encodeURIComponent(CONSTANTS.COMMANDS.TEACHER.ADD_PRODUCT)}` } },
        { type: 'button', style: 'secondary', height: 'sm', action: { type: 'postback', label: '🛒'
管理販售中商品', data:
`action=run_command&text=${encodeURIComponent(CONSTANTS.COMMANDS.TEACHER.MANAGE_AVAILABLE_PRODUCTS)}` } },
        { type: 'button', style: 'secondary', height: 'sm', action: { type: 'postback', label: '📦'
管理已下架商品', data:
`action=run_command&text=${encodeURIComponent(CONSTANTS.COMMANDS.TEACHER.MANAGE_UNAVAILABLE_PRODUCTS)}` } },
        { type: 'separator', margin: 'md' },
        { type: 'button', style: 'secondary', height: 'sm', action: { type: 'postback', label:
pendingShopOrdersLabel, data:
`action=run_command&text=${encodeURIComponent(CONSTANTS.COMMANDS.TEACHER.SHOP_ORDER_MANAGEMENT)}` } },
        { type: 'button', style: 'secondary', height: 'sm', action: { type: 'postback', label: '📄'
查詢兌換紀錄', data: `action=select_exchange_history_view_type` } }
      ]
    }
  };
}

```

```

async function startAddProduct(event, user) {
  const userId = user.id;
  pendingProductCreation[userId] = { step: 'await_name' };
  setupConversationTimeout(userId, pendingProductCreation, 'pendingProductCreation', u => {
    const timeoutMessage = { type: 'text', text: '上架商品操作逾時，自動取消。' };
    enqueuePushTask(u, timeoutMessage).catch(e => console.error(e));
  });
  return {
    type: 'text',
    text: '請輸入新商品的名稱:',
    quickReply: { items: getCancelMenu() }
  };
}

```

```
};  
}
```

```
async function showAvailableProductsList(event, user) {  
  return showProductManagementList(1, 'available');  
}
```

```
async function showUnavailableProductsList(event, user) {  
  return showProductManagementList(1, 'unavailable');  
}
```

```
async function showShopOrderManagement(event, user) {  
  return showPendingShopOrders(1);  
}
```

```
async function showReportMenu(event, user) {  
  return {  
    type: 'flex',  
    altText: '統計報表',  
    contents: {  
      type: 'bubble',  
      size: 'giga',  
      header: {  
        type: 'box',  
        layout: 'vertical',  
        contents: [{ type: 'text', text: '📊 統計報表', weight: 'bold', size: 'lg', color: '#FFFFFF' }],  
        backgroundColor: '#343A40',  
        paddingTop: 'lg',  
        paddingBottom: 'lg'  
      }],  
      body: {  
        type: 'box',  
        layout: 'vertical',  
        spacing: 'md',  
        paddingAll: 'lg',  
        contents: [  
          { type: 'button', style: 'secondary', height: 'sm', action: { type: 'postback', label: '📈  
課程報表', data:  
`action=run_command&text=${encodeURIComponent(CONSTANTS.COMMANDS.TEACHER.  
COURSE_REPORT)}` } },  
          { type: 'button', style: 'secondary', height: 'sm', action: { type: 'postback', label: '💰  
訂單報表', data:  
`action=run_command&text=${encodeURIComponent(CONSTANTS.COMMANDS.TEACHER.  
ORDER_REPORT)}` } },
```



```

        { type: 'button', style: 'secondary', height: 'sm', action: { type: 'postback', label: '💎
點數報表', data:
`action=run_command&text=${encodeURIComponent(CONSTANTS.COMMANDS.TEACHER.P
OINT_REPORT)}` } }
    ]
  }
}
};
}

```

```

async function showTimePeriodMenuForReport(event, user) {
  const text = event.message.text.trim();
  const reportType = text === CONSTANTS.COMMANDS.TEACHER.COURSE_REPORT ?
'course' : 'order';
  const title = text === CONSTANTS.COMMANDS.TEACHER.COURSE_REPORT ? '課程報表'
: '訂單報表';
  return {
    type: 'flex',
    altText: '選擇時間週期',
    contents: {
      type: 'bubble',
      header: {
        type: 'box',
        layout: 'vertical',
        contents: [{ type: 'text', text: `📊 ${title}`, weight: 'bold', size: 'lg', color: '#FFFFFF' }],
        backgroundColor: '#52b69a'
      },
      body: {
        type: 'box',
        layout: 'vertical',
        spacing: 'sm',
        contents: [
          { type: 'button', style: 'link', height: 'sm', action: { type: 'postback', label: '本週', data:
`action=generate_report&type=${reportType}&period=week` } },
          { type: 'button', style: 'link', height: 'sm', action: { type: 'postback', label: '本月', data:
`action=generate_report&type=${reportType}&period=month` } },
          { type: 'button', style: 'link', height: 'sm', action: { type: 'postback', label: '本季', data:
`action=generate_report&type=${reportType}&period=quarter` } },
          { type: 'button', style: 'link', height: 'sm', action: { type: 'postback', label: '今年', data:
`action=generate_report&type=${reportType}&period=year` } },
        ]
      },
      footer: {

```

```

      type: 'box',
      layout: 'vertical',
      contents: [{ type: 'text', text: '請選擇要查詢的時間區間', size: 'sm', color: '#AAAAAA',
align: 'center'}]
    }
  }
};
}

```

```

async function generatePointReport(event, user) {
  const userId = user.id;
  const generateReportTask = async () => {
    return executeDbQuery(async (client) => {
      const usersRes = await client.query('SELECT name, points FROM users WHERE role =
'student' ORDER BY points DESC');
      const students = usersRes.rows;
      if (students.length === 0) {
        return '目前沒有任何學員資料可供分析。';
      }
      const totalPoints = students.reduce((sum, s) => sum + s.points, 0);
      const averagePoints = (totalPoints / students.length).toFixed(2);
      const top5 = students.slice(0, 5).map(s => ` - ${s.name}: ${s.points} 點`).join("\n");
      const zeroPointStudents = students.filter(s => s.points === 0).length;
      return `💎 全體學員點數報告 💎\n\n總學員數: ${students.length} 人\n點數總計:
${totalPoints} 點\n平均持有: ${averagePoints} 點/人\n零點學員: ${zeroPointStudents} 人\n\n👑
點數持有 Top 5: \n${top5}`;
    });
  };
};

```

```

const timeoutPromise = new Promise(resolve => setTimeout(() => resolve('timeout'), 8000));
try {
  const result = await Promise.race([generateReportTask(), timeoutPromise]);
  if (result === 'timeout') {
    (async () => {
      const reportText = await generateReportTask();
      await enqueuePushTask(userId, { type: 'text', text: reportText });
    })();
    return '📊 報表生成中，資料量較大，請稍候... 完成後將會推播通知您。';
  } else {
    return result;
  }
} catch (err) {
  console.error('❌ 即時生成點數報表失敗:', err);
}

```

```

    return '✖ 產生報表時發生錯誤，請稍後再試。';
  }
}

```

```

async function startManualAdjust(event, user) {
  const userId = user.id;
  pendingManualAdjust[userId] = { step: 'await_student_search' };
  setupConversationTimeout(userId, pendingManualAdjust, 'pendingManualAdjust', (u) => {
    const timeoutMessage = { type: 'text', text: '手動調整點數逾時，自動取消。'};
    enqueuePushTask(u, timeoutMessage).catch(e => console.error(e));
  });
  return {
    type: 'text',
    text: '請輸入您想調整點數的學員姓名或 User ID:',
    quickReply: { items: getCancelMenu() }
  };
}

```

// --- Teacher Command Map (V34.0 Refactor) ---

```

const teacherCommandMap = {
  [CONSTANTS.COMMANDS.TEACHER.COURSE_MANAGEMENT]:
showCourseManagementMenu,
  [CONSTANTS.COMMANDS.TEACHER.ADD_COURSE_SERIES]: startAddCourseSeries,
  [CONSTANTS.COMMANDS.TEACHER.MANAGE_OPEN_COURSES]:
showManageOpenCourses,
  [CONSTANTS.COMMANDS.TEACHER.COURSE_INQUIRY]: showCourseInquiry,
  [CONSTANTS.COMMANDS.TEACHER.POINT_MANAGEMENT]:
showPointManagementMenu,
  [CONSTANTS.COMMANDS.TEACHER.PENDING_POINT_ORDERS]:
showPendingPointOrders,
  [CONSTANTS.COMMANDS.TEACHER.STUDENT_MANAGEMENT]:
showStudentManagementMenu,
  [CONSTANTS.COMMANDS.TEACHER.SEARCH_STUDENT]: startStudentSearch,
  [CONSTANTS.COMMANDS.TEACHER.VIEW_MESSAGES]:
showUnreadTeacherMessages,
  [CONSTANTS.COMMANDS.TEACHER.MESSAGE_SEARCH]: startMessageSearch,
  [CONSTANTS.COMMANDS.TEACHER.ANNOUNCEMENT_MANAGEMENT]:
showAnnouncementManagementMenu,
  [CONSTANTS.COMMANDS.TEACHER.ADD_ANNOUNCEMENT]: startAddAnnouncement,
  [CONSTANTS.COMMANDS.TEACHER.DELETE_ANNOUNCEMENT]:
showAnnouncementsForDeletionList,
  [CONSTANTS.COMMANDS.TEACHER.SHOP_MANAGEMENT]:
showShopManagementMenu,

```

```

    [CONSTANTS.COMMANDS.TEACHER.ADD_PRODUCT]: startAddProduct,
    [CONSTANTS.COMMANDS.TEACHER.MANAGE_AVAILABLE_PRODUCTS]:
showAvailableProductsList,
    [CONSTANTS.COMMANDS.TEACHER.MANAGE_UNAVAILABLE_PRODUCTS]:
showUnavailableProductsList,
    [CONSTANTS.COMMANDS.TEACHER.SHOP_ORDER_MANAGEMENT]:
showShopOrderManagement,
    [CONSTANTS.COMMANDS.TEACHER.REPORT]: showReportMenu,
    [CONSTANTS.COMMANDS.TEACHER.COURSE_REPORT]:
showTimePeriodMenuForReport,
    [CONSTANTS.COMMANDS.TEACHER.ORDER_REPORT]:
showTimePeriodMenuForReport,
    [CONSTANTS.COMMANDS.TEACHER.POINT_REPORT]: generatePointReport,
    [CONSTANTS.COMMANDS.TEACHER.PENDING_ORDERS]: showPendingPointOrders, //
Alias
    [CONSTANTS.COMMANDS.TEACHER.MANUAL_ADJUST_POINTS]: startManualAdjust,
    // [新增] 購點紀錄與兌換紀錄的指令處理
    [CONSTANTS.COMMANDS.TEACHER.VIEW_PURCHASE_HISTORY]:
showPurchaseHistoryList,
    [CONSTANTS.COMMANDS.TEACHER.VIEW_SHOP_EXCHANGE_HISTORY]:
showExchangeHistoryList,
};
function handleUnknownTeacherCommand(text) {
    let teacherSuggestion = '無法識別您的指令 😞\n請直接使用下方的老師專用選單進行操作。';
    if (text.startsWith('@')) {
        const closestCommand = findClosestCommand(text, 'teacher');
        if (closestCommand) {
            teacherSuggestion = `找不到指令 "${text}"，您是不是想輸入「${closestCommand}」？`;
        } else {
            teacherSuggestion = `哎呀，找不到指令 "${text}"。請檢查一下是不是打錯字了，或直接
使用選單最準確喔！`;
        }
    }
    return teacherSuggestion;
}

// --- Main Command Handlers ---
async function getUserNames(userIds, dbClient) {
    if (!userIds || userIds.length === 0) {
        return new Map();
    }
    const usersRes = await dbClient.query("SELECT id, name FROM users WHERE id =
ANY($1::text[])", [userIds]);

```

```

    return new Map(usersRes.rows.map(u => [u.id, u.name]));
  }
  async function showFailedTasks(page) {
    const offset = (page - 1) * CONSTANTS.PAGINATION_SIZE;
    return executeDbQuery(async (client) => {
      const res = await client.query(
        "SELECT * FROM failed_tasks ORDER BY failed_at DESC LIMIT $1 OFFSET $2",
        [CONSTANTS.PAGINATION_SIZE + 1, offset]
      );

      const hasNextPage = res.rows.length > CONSTANTS.PAGINATION_SIZE;
      const pageTasks = hasNextPage ? res.rows.slice(0, CONSTANTS.PAGINATION_SIZE) :
res.rows;

      if (pageTasks.length === 0 && page === 1) {
        return '✅ 太好了！目前沒有任何失敗的任務。';
      }
      if (pageTasks.length === 0) {
        return '沒有更多失敗的任務了。';
      }

      const userIds = [...new Set(pageTasks.map(task => task.recipient_id))];
      const userNamesMap = await getUserNames(userIds, client);

      const taskBubbles = pageTasks.map(task => {
        const recipientName = userNamesMap.get(task.recipient_id) || '未知用戶';
        const errorMessage = task.last_error || '沒有錯誤訊息。';

        return {
          type: 'bubble',
          size: 'giga',
          header: { type: 'box', layout: 'vertical', contents: [{ type: 'text', text: '💣 任務失敗',
weight: 'bold', color: '#FFFFFF' }], backgroundColor: '#d9534f', paddingAll: 'lg' },
          body: { type: 'box', layout: 'vertical', spacing: 'md', contents: [
            { type: 'box', layout: 'baseline', spacing: 'sm', contents: [ { type: 'text', text: '收件人',
color: '#aaaaaa', size: 'sm', flex: 2 }, { type: 'text', text: `${recipientName}`, color: '#666666', size:
'sm', flex: 5, wrap: true } ] },
            { type: 'box', layout: 'baseline', spacing: 'sm', contents: [ { type: 'text', text: '失敗時間',
color: '#aaaaaa', size: 'sm', flex: 2 }, { type: 'text', text: formatDateTime(task.failed_at), color:
'#666666', size: 'sm', flex: 5, wrap: true } ] },
            { type: 'box', layout: 'vertical', spacing: 'sm', contents: [ { type: 'text', text: '錯誤原因',
color: '#aaaaaa', size: 'sm' }, { type: 'text', text: errorMessage.substring(0, 100), color: '#666666',
size: 'sm', wrap: true, margin: 'md' } ] } ]
        };
      });
    });
  }

```

```

    ]},
    footer: { type: 'box', layout: 'horizontal', spacing: 'sm', contents: [
      { type: 'button', style: 'secondary', flex: 1, height: 'sm', action: { type: 'postback',
label: '🗑️ 刪除', data: `action=delete_failed_task&id=${task.id}` } },
      { type: 'button', style: 'primary', color: '#5cb85c', flex: 1, height: 'sm', action: { type:
'postback', label: '🔄 重試', data: `action=retry_failed_task&id=${task.id}` } }
    ]}
  };
});

```

```

const paginationBubble = createPaginationBubble('action=view_failed_tasks', page,
hasNextPage);
if (paginationBubble) {
  taskBubbles.push(paginationBubble);
}

```

```

return { type: 'flex', altText: '失敗任務列表', contents: { type: 'carousel', contents:
taskBubbles } };
});
}

```

```

async function showSystemStatus() {
  return executeDbQuery(async (db) => {
    const [pendingRes, processingRes, failedRes] = await Promise.all([
      db.query("SELECT COUNT(*) FROM tasks WHERE status = 'pending'"),
      db.query("SELECT COUNT(*) FROM tasks WHERE status = 'processing'"),
      db.query("SELECT COUNT(*) FROM failed_tasks")
    ]);

```

```

const pendingCount = pendingRes.rows[0].count;
const processingCount = processingRes.rows[0].count;
const failedCount = failedRes.rows[0].count;

```

```
const statusText = `
```

⚙️ 背景系統狀態 ⚙️

- 待處理任務: \${pendingCount} 個
- 正在處理中: \${processingCount} 個
- 失敗任務(DLQ): \${failedCount} 個

i 「待處理任務」是系統即將要發送的排程訊息 (如課程提醒)。若「失敗任務」數量持續增加, 請檢查 Worker 紀錄。

```
`.trim();
```

```

    return statusText;
  });
}

async function showTeacherListForRemoval(page) {
  const offset = (page - 1) * CONSTANTS.PAGINATION_SIZE;
  return executeDbQuery(async (client) => {
    const res = await client.query(
      "SELECT id, name, picture_url FROM users WHERE role = 'teacher' ORDER BY name
ASC LIMIT $1 OFFSET $2",
      [CONSTANTS.PAGINATION_SIZE + 1, offset]
    );

    const hasNextPage = res.rows.length > CONSTANTS.PAGINATION_SIZE;
    const pageTeachers = hasNextPage ? res.rows.slice(0, CONSTANTS.PAGINATION_SIZE)
: res.rows;

    if (pageTeachers.length === 0 && page === 1) {
      return '目前沒有任何已授權的老師可供移除。';
    }
    if (pageTeachers.length === 0) {
      return '沒有更多老師了。';
    }

    const placeholder_avatar = 'https://i.imgur.com/8l1Yd2S.png';
    const teacherBubbles = pageTeachers.map(t => ({
      type: 'bubble',
      body: {
        type: 'box',
        layout: 'horizontal',
        spacing: 'md',
        contents: [
          { type: 'image', url: t.picture_url || placeholder_avatar, size: 'md', aspectRatio: '1:1',
aspectMode: 'cover' },
          { type: 'box', layout: 'vertical', flex: 3, justifyContent: 'center',
            contents: [
              { type: 'text', text: t.name, weight: 'bold', size: 'lg', wrap: true },
              { type: 'text', text: `ID: ${formatIdForDisplay(t.id)}`, size: 'xxs', color:
'#AAAAAA', margin: 'sm', wrap: true }
            ]
          }
        ]
      }
    }
  ]
}

```

```

    },
    footer: {
      type: 'box',
      layout: 'vertical',
      contents: [{
        type: 'button',
        style: 'primary',
        color: '#DE5246',
        height: 'sm',
        action: { type: 'postback', label: '選擇此老師', data:
`action=select_teacher_for_removal&targetId=${t.id}&targetName=${encodeURIComponent(t.name)}` }
      }]
    }
  }
  }));
  const paginationBubble = createPaginationBubble('action=list_teachers_for_removal',
page, hasNextPage);
  if (paginationBubble) {
    teacherBubbles.push(paginationBubble);
  }

  return {
    type: 'flex',
    altText: '選擇要移除的老師',
    contents: { type: 'carousel', contents: teacherBubbles }
  };
});
}

```

// [V35.6 重構] 將購點紀錄改為條列式，並整合待處理訂單

async function showPurchaseHistory(userId, page) { // page 參數暫時保留

```

  return executeDbQuery(async (client) => {
    // 抓取最近 20 筆相關紀錄
    const res = await client.query(
      `SELECT * FROM orders WHERE user_id = $1 ORDER BY timestamp DESC LIMIT
20`,
      [userId]
    );

    if (res.rows.length === 0) {
      return '您沒有任何購點紀錄。';
    }
  }

```



```

// 步驟 1: 將訂單分組
const pendingPointOrders = [];
const historyPointOrders = [];

res.rows.forEach(order => {
  if (['pending_payment', 'pending_confirmation', 'rejected'].includes(order.status)) {
    pendingPointOrders.push(order);
  } else {
    historyPointOrders.push(order);
  }
});

const bodyContents = [];
const separator = { type: 'separator', margin: 'md' };

// 步驟 2: 產生「待處理訂單」列表
if (pendingPointOrders.length > 0) {
  bodyContents.push({ type: 'text', text: '待處理訂單', weight: 'bold', size: 'lg', margin: 'md',
    color: '#1A759F' });

  pendingPointOrders.forEach(order => {
    // 這段邏輯是從舊的 buildPointsMenuFlex 搬過來的
    let actionButtonLabel, cardColor, statusText, actionCmd, additionalInfo = '';
    if (order.status === 'pending_confirmation') {
      actionButtonLabel = '修改匯款後五碼'; actionCmd =
CONSTANTS.COMMANDS.STUDENT.EDIT_LAST5_CARD_TRIGGER; cardColor = '#ff9e00';
      statusText = '已提交, 等待老師確認';
    } else if (order.status === 'rejected') {
      actionButtonLabel = '重新提交後五碼'; actionCmd =
CONSTANTS.COMMANDS.STUDENT.EDIT_LAST5_CARD_TRIGGER; cardColor = '#d90429';
      statusText = '訂單被老師退回'; additionalInfo = '請檢查金額或後五碼, 並重新提交。';
    } else { // pending_payment
      actionButtonLabel = '輸入匯款後五碼'; actionCmd =
CONSTANTS.COMMANDS.STUDENT.INPUT_LAST5_CARD_TRIGGER; cardColor =
'#f28482'; statusText = '待付款';
    }

    bodyContents.push({
      type: 'box',
      layout: 'vertical',
      margin: 'lg',
      spacing: 'sm',
      contents: [

```

```

        { type: 'text', text: `${order.points} 點 / ${order.amount} 元`, weight: 'bold', wrap:
true },
        { type: 'text', text: `狀態: ${statusText}`, size: 'sm', color: cardColor, weight: 'bold'
},
        { type: 'text', text: formatDateTime(order.timestamp), size: 'sm', color: '#AAAAAA'
},
        ...(additionalInfo ? [{ type: 'text', text: additionalInfo, size: 'xs', color: '#B00020',
wrap: true, margin: 'sm' }] : []),
        {
            type: 'button', style: 'primary', height: 'sm', margin: 'md', color: cardColor,
            action: { type: 'postback', label: actionButtonLabel, data:
`action=run_command&text=${encodeURIComponent(actionCmd)}` }
        }
    ]
});
bodyContents.push(separator);
});
}

```

// 步驟 3: 產生「歷史紀錄」列表

```

if (historyPointOrders.length > 0) {
    bodyContents.push({ type: 'text', text: '歷史紀錄', weight: 'bold', size: 'lg', margin: 'xl',
color: '#6c757d' });

    historyPointOrders.forEach(order => {
        let typeText, pointsText, pointsColor;
        if (order.amount === 0) { // 手動調整
            if (order.points > 0) { typeText = '✨ 手動加點'; pointsText = `+${order.points}`;
pointsColor = '#1A759F'; }
            else { typeText = '⚠ 手動扣點'; pointsText = `${order.points}`; pointsColor =
'#D9534F'; }
        } else { // 一般購點
            typeText = '✅ 購點成功'; pointsText = `+${order.points}`; pointsColor = '#28A745';
        }

        bodyContents.push({
            type: 'box',
            layout: 'horizontal',
            margin: 'lg',
            contents: [
                {
                    type: 'box', layout: 'vertical', flex: 3,
                    contents: [

```

```

        { type: 'text', text: typeText, weight: 'bold', size: 'sm' },
        { type: 'text', text: formatDateTime(order.timestamp), size: 'xxs', color:
'#AAAAAA' }
    ]
    },
    { type: 'text', text: `${pointsText} 點`, gravity: 'center', align: 'end', flex: 2, weight:
'bold', size: 'sm', color: pointsColor }
    ]
    });
    bodyContents.push(separator);
    });
}

if (bodyContents.length > 0 && bodyContents[bodyContents.length - 1].type ===
'separator') {
    bodyContents.pop();
}

return {
    type: 'flex',
    altText: '購點紀錄',
    contents: {
        type: 'bubble',
        size: 'giga',
        header: {
            type: 'box',
            layout: 'vertical',
            contents: [{ type: 'text', text: '📖 查詢購點紀錄', weight: 'bold', size: 'xl', color:
'#FFFFFF' }],
            backgroundColor: '#343A40',
            paddingAll: 'lg'
        },
        body: {
            type: 'box',
            layout: 'vertical',
            spacing: 'md',
            paddingAll: 'lg',
            contents: bodyContents.length > 0 ? bodyContents : [{type: 'text', text: '目前沒有任
何紀錄。', align: 'center'}]
        }
    }
    };
});

```

```
}
```

```
// [新增] 處理顯示兌換歷史的功能
```

```
async function showExchangeHistoryList(event, user) {
  return {
    type: 'flex',
    altText: '選擇查詢方式',
    contents: {
      type: 'bubble',
      header: {
        type: 'box',
        layout: 'vertical',
        contents: [{ type: 'text', text: '📖 查詢兌換紀錄', weight: 'bold', size: 'lg', color:
'#FFFFFF' }],
        backgroundColor: '#52b69a'
      },
      body: {
        type: 'box',
        layout: 'vertical',
        spacing: 'sm',
        contents: [
          { type: 'button', style: 'link', height: 'sm', action: { type: 'postback', label: '顯示全部紀
錄', data: `action=view_all_exchange_history_as_teacher&page=1` } },
          { type: 'button', style: 'link', height: 'sm', action: { type: 'postback', label: '搜尋特定學
員', data: `action=start_exchange_history_search` } }
        ]
      }
    }
  };
}
```

```
// 處理老師查詢購點歷史的初始選單
```

```
async function showPurchaseHistoryList(event, user) {
  return {
    type: 'flex',
    altText: '選擇查詢方式',
    contents: {
      type: 'bubble',
      header: {
        type: 'box',
        layout: 'vertical',
        contents: [{ type: 'text', text: '📖 查詢購點紀錄', weight: 'bold', size: 'lg', color: '#FFFFFF' }],
        backgroundColor: '#52b69a'
      },
    }
  };
}
```

```

body: {
  type: 'box',
  layout: 'vertical',
  spacing: 'sm',
  contents: [
    { type: 'button', style: 'link', height: 'sm', action: { type: 'postback', label: '顯示全部紀錄',
data: `action=view_all_purchase_history_as_teacher&page=1` } },
    { type: 'button', style: 'link', height: 'sm', action: { type: 'postback', label: '搜尋特定學員',
data: `action=start_purchase_history_search` } }
  ]
}
};
}

```

```

async function handleTeacherCommands(event, userId) {
  const text = event.message.text ? event.message.text.trim().normalize() : "";
  const user = await getUser(userId);
  // 優先處理有延續性的對話 (Pending States)
  if (pendingProductCreation[userId]) {
    const state = pendingProductCreation[userId];
    let proceed = true;
    let errorMessage = "";
    switch (state.step) {
      case 'await_name': state.name = text;
        state.step = 'await_description'; return { type: 'text', text: '請輸入商品描述 (可換行), 或輸入「無」:', quickReply: { items: getCancelMenu() } };
      case 'await_description': state.description = text === '無' ? null : text; state.step = 'await_price';
        return { type: 'text', text: '請輸入商品售價 (元整, 純數字):', quickReply: { items: getCancelMenu() } };
      case 'await_price':
        const price = parseInt(text, 10);
        if (isNaN(price) || price < 0) { proceed = false; errorMessage = '價格格式不正確, 請輸入一個非負整數。';
        }
        else { state.price = price;
          state.step = 'await_inventory'; return { type: 'text', text: '請輸入商品初始庫存 (純數字):', quickReply: { items: getCancelMenu() } };
        }
        break;
      case 'await_inventory':
        const inventory = parseInt(text, 10);

```

```

        if (isNaN(inventory) || inventory < 0) { proceed = false; errorMessage = '庫存格式不正確
, 請輸入一個非負整數。';
        }
        else { state.inventory = inventory;
        state.step = 'await_image_url'; return { type: 'text', text: '請直接上傳一張商品圖片, 或輸
入「無」:', quickReply: { items: getCancelMenu() } };
        }
        break;
    case 'await_image_url':
        let imageUrl = null;
        let proceedToNextStep = true; let imageUrlErrorMessage = "";
        if (event.message.type === 'text' && event.message.text.trim().toLowerCase() === '無') {
imageUrl = null;
        }
        else if (event.message.type === 'image') {
            try {
                const imageResponse = await
axios.get(`https://api-data.line.me/v2/bot/message/${event.message.id}/content`, { headers: {
'Authorization': `Bearer ${process.env.CHANNEL_ACCESS_TOKEN}` }, responseType:
'arraybuffer' });
                const imageBuffer = Buffer.from(imageResponse.data, 'binary');
                const uploadResponse = await imagekit.upload({ file: imageBuffer, fileName:
`product_${Date.now()}.jpg`, useUniqueFileName: true, folder: "yoga_products" });
                imageUrl = uploadResponse.url;
            } catch (err) { console.error("❌ 圖片上傳至 ImageKit.io 失敗:", err);
proceedToNextStep = false; imageUrlErrorMessage = '圖片上傳失敗, 請稍後再試。';
            }
        } else { proceedToNextStep = false;
        imageUrlErrorMessage = '格式錯誤, 請直接上傳一張商品圖片, 或輸入「無」。';
        if (!proceedToNextStep) { return { type: 'text', text: imageUrlErrorMessage, quickReply: {
items: getCancelMenu() } };
        }
        state.image_url = imageUrl; state.step = 'await_confirmation';
        const summaryText = `請確認商品資訊:\n\n名稱: ${state.name}\n描述:
${state.description || '無'}\n價格: ${state.price} 點\n庫存: ${state.inventory}\n圖片:
${state.image_url || '無'}\n\n確認無誤後請點擊「✅ 確認上架」。`;
        return {
            type: 'text',
            text: summaryText,
            quickReply: { items: [ { type: 'action', action: { type: 'postback', label: '✅ 確認上架',
data: 'action=confirm_add_product' } }, { type: 'action', action: { type: 'message', label:
CONSTANTS.COMMANDS.GENERAL.CANCEL, text:
CONSTANTS.COMMANDS.GENERAL.CANCEL } } ] }

```

```

    };
  }
  if (!proceed && state.step !== 'await_image_url') { return { type: 'text', text: errorMessage,
quickReply: { items: getCancelMenu() } };
  }
  } else if (pendingProductEdit[userId]) {
    const state = pendingProductEdit[userId];
    const product = state.product;
    const field = state.field;
    let newValue = text; let isValid = true; let errorMessage = "";
    if (field === 'price' || field === 'inventory') {
      const numValue = parseInt(text, 10);
      if (isNaN(numValue) || numValue < 0) { isValid = false; errorMessage = '請輸入一個非負整
數。'; } else { newValue = numValue;
    }
  } else if (field === 'description' && text.toLowerCase() === '無') { newValue = null;
  } else if (field === 'image_url') {
    if (text.toLowerCase() === '無') { newValue = null;
    }
    else if (!text.startsWith('https://') || !text.match(/\.(jpeg|jpg|gif|png)$/i)) { isValid = false;
errorMessage = '圖片網址格式不正確，必須是 https 開頭的圖片連結。'; }
  }
  if (!isValid) { return { type: 'text', text: errorMessage, quickReply: { items: getCancelMenu() } };
  }
  product[field] = newValue; await saveProduct(product); delete pendingProductEdit[userId];
  const fieldMap = { name: '名稱', description: '描述', price: '價格', image_url: '圖片網址',
inventory: '庫存' };
  return `✅ 已成功將商品「${product.name}」的「${fieldMap[field]}」更新為「${newValue === null
? '無' : newValue}」。`;
  } else if (pendingInventoryAdjust[userId]) {
    const state = pendingInventoryAdjust[userId];
    const product = state.product; const numValue = parseInt(text, 10);
    if(isNaN(numValue)) { return { type: 'text', text: '格式錯誤，請輸入一個整數 (正數為增加，負數
為減少)。', quickReply: { items: getCancelMenu() } };
    }
    const newInventory = product.inventory + numValue;
    if(newInventory < 0) { return { type: 'text', text: `庫存調整失敗，調整後庫存 (${newInventory})
不可小於 0。`, quickReply: { items: getCancelMenu() } };
    }
    product.inventory = newInventory; await saveProduct(product); delete
pendingInventoryAdjust[userId];
    return `✅ 已成功調整商品「${product.name}」的庫存。原庫存: ${state.originalInventory}\n調
整量: ${numValue > 0 ? '+' : ''}${numValue}\n新庫存: ${newInventory}`;
  }

```

```

} else if (pendingAnnouncementCreation[userId]) {
  const state = pendingAnnouncementCreation[userId];
  switch (state.step) {
    case 'await_content':
      state.content = text;
      state.step = 'await_confirmation';
      const confirmMsg = { type: 'flex', altText: '確認公告內容', contents: { type: 'bubble', body: {
type: 'box', layout: 'vertical', spacing: 'lg', contents: [ { type: 'text', text: '請確認公告內容', weight:
'bold', size: 'lg' }, { type: 'separator' }, { type: 'text', text: state.content, wrap: true } ] }, footer: {
type: 'box', layout: 'vertical', spacing: 'sm', contents: [ { type: 'button', style: 'primary', color:
'#52b69a', action: { type: 'message', label:
CONSTANTS.COMMANDS.TEACHER.CONFIRM_ADD_ANNOUNCEMENT, text:
CONSTANTS.COMMANDS.TEACHER.CONFIRM_ADD_ANNOUNCEMENT } }, { type: 'button',
style: 'secondary', action: { type: 'message', label:
CONSTANTS.COMMANDS.GENERAL.CANCEL, text:
CONSTANTS.COMMANDS.GENERAL.CANCEL } } ] } } } };
      return confirmMsg;
    case 'await_confirmation':
      if (text === CONSTANTS.COMMANDS.TEACHER.CONFIRM_ADD_ANNOUNCEMENT) {
        await executeDbQuery(client =>
          client.query( "INSERT INTO announcements (content, creator_id, creator_name)
VALUES ($1, $2, $3)", [state.content, userId, user.name])
        );
        delete pendingAnnouncementCreation[userId];
        return '✅ 公告已成功頒佈！學員可在「最新公告」中查看。';
      } else { return '請點擊「確認頒佈」或「取消操作」。';
      }
    }
  } else if (pendingAnnouncementDeletion[userId]) {
    const state = pendingAnnouncementDeletion[userId];
    if (text === CONSTANTS.COMMANDS.TEACHER.CONFIRM_DELETE_ANNOUNCEMENT)
    {
      await executeDbQuery(client => client.query("DELETE FROM announcements WHERE id
= $1", [state.ann_id]));
      delete pendingAnnouncementDeletion[userId];
      return '✅ 公告已成功刪除。';
    } else { return '請點擊「確認刪除」或「取消操作」。';
    }
  } else if (pendingCourseCancellation[userId]) {
    const state = pendingCourseCancellation[userId];
    switch(state.type) {
      case 'batch':
        if (text === CONSTANTS.COMMANDS.TEACHER.CONFIRM_BATCH_CANCEL) {

```



```

const backgroundState = { ...state };
delete pendingCourseCancellation[userId];
try {
  (async () => {
    await executeDbQuery(async (client) => {
      await client.query('BEGIN');
      try {
        const coursesToCancelRes = await client.query("SELECT * FROM courses
WHERE id LIKE $1 AND time > NOW() FOR UPDATE", [`${backgroundState.prefix}%`]);
        if (coursesToCancelRes.rows.length === 0) {
          const errMsg = { type: 'text', text: `❌ 批次取消失敗: 找不到可取消的「
${backgroundState.prefix}」系列課程。` };
          await enqueuePushTask(userId, errMsg);
          return;
        }
        const coursesToCancel = coursesToCancelRes.rows;
        const affectedUsers = new Map();
        for (const course of coursesToCancel) {
          for (const studentId of course.students) {
            if (!affectedUsers.has(studentId)) affectedUsers.set(studentId, 0);
            affectedUsers.set(studentId, affectedUsers.get(studentId) +
course.points_cost);
          }
        }
        for (const [studentId, refundAmount] of affectedUsers.entries()) {
          if (refundAmount > 0) {
            await client.query("UPDATE users SET points = points + $1 WHERE id = $2",
[refundAmount, studentId]);
          }
        }
        const courseMainTitle = getCourseMainTitle(coursesToCancel[0].title);
        await client.query("DELETE FROM courses WHERE id LIKE $1 AND time >
NOW()", [`${backgroundState.prefix}%`]);
        const batchTasks = Array.from(affectedUsers.entries()).map(([studentId,
refundAmount]) => ({
          recipientId: studentId,
          message: { type: 'text', text: `課程取消通知:\n老師已取消「${courseMainTitle}」系
列所有課程, 已歸還 ${refundAmount} 點至您的帳戶。` }
        }));
        if (batchTasks.length > 0) {
          await enqueueBatchPushTasks(batchTasks);
        }
        await client.query('COMMIT');
      }
    });
  })();
}

```

```
const teacherMsg = { type: 'text', text: `✅ 已成功批次取消「${courseMainTitle}」系列課程，並已退點給所有學員。` };
    await enqueuePushTask(userId, teacherMsg);
  } catch (e) {
    await client.query('ROLLBACK');
    console.error('[批次取消] 背景任務執行失敗:', e);
    const errorMsg = { type: 'text', text: `❌ 批次取消課程時發生嚴重錯誤，操作已復原。請聯繫管理員。\\n錯誤: ${e.message}` };
    await enqueuePushTask(userId, errorMsg);
  }
});
})();
return `✅ 指令已收到，正在為您批次取消課程。\\n完成後將會另行通知，請稍候...`;
} catch (error) {
  console.error(`❌ 啟動批次取消時發生錯誤:`, error);
  return '啟動批次取消任務失敗，請稍後再試。';
}
}
break;
case 'single':
  if (text === CONSTANTS.COMMANDS.TEACHER.CONFIRM_SINGLE_CANCEL) {
    return executeDbQuery(async (client) => {
      await client.query('BEGIN');
      try {
        const courseToCancelRes = await client.query("SELECT * FROM courses WHERE id = $1 FOR UPDATE", [state.course_id]);
        if (courseToCancelRes.rows.length === 0) { delete
pendingCourseCancellation[userId]; return "找不到該課程，可能已被取消。"; }
        const course = courseToCancelRes.rows[0];
        const studentIdsToNotify = [...course.students];
        for (const studentId of studentIdsToNotify) {
          await client.query("UPDATE users SET points = points + $1 WHERE id = $2",
[course.points_cost, studentId]);
        }
        await client.query("DELETE FROM courses WHERE id = $1", [state.course_id]);
        delete pendingCourseCancellation[userId];
        if (studentIdsToNotify.length > 0) {
          const batchTasks = studentIdsToNotify.map(studentId => ({
            recipientId: studentId,
            message: { type: 'text', text: `課程取消通知:\\n老師已取消您預約的課程「${course.title}」，已歸還 ${course.points_cost} 點至您的帳戶。` }
          }));
          await enqueueBatchPushTasks(batchTasks);
        }
      } catch (e) {
        console.error('[單次取消] 背景任務執行失敗:', e);
        const errorMsg = { type: 'text', text: `❌ 單次取消課程時發生嚴重錯誤，操作已復原。請聯繫管理員。\\n錯誤: ${e.message}` };
        await enqueuePushTask(userId, errorMsg);
      }
    });
  }
}
break;
}
```

```

    }
    await client.query('COMMIT');
    return `✅ 已成功取消課程「${course.title}」.`;
  } catch (e) {
    await client.query('ROLLBACK');
    delete pendingCourseCancellation[userId];
    console.error('單堂取消課程失敗:', e);
    return '取消課程時發生錯誤，請稍後再試。';
  }
});
}
break;
}
} else if (pendingCourseCreation[userId]) {
  const state = pendingCourseCreation[userId];
  switch (state.step) {
    case 'await_title':
      state.title = text;
      state.step = 'await_weekday';
      const weekdayButtons = WEEKDAYS.map(day => ({ type: 'action', action: { type:
'postback', label: day.label, data: `action=set_course_weekday&day=${day.value}` } }));
      return { type: 'text', text: `課程標題:「${text}」\n\n請問課程固定在每週的哪一天?`,
quickReply: { items: weekdayButtons } };
    case 'await_time':
      if (!/^\\d{2}:\\d{2}$/.test(text)) { return { type: 'text', text: '時間格式不正確，請輸入四位數時
間，例如: 19:30', quickReply: { items: getCancelMenu() } };
      }
      state.time = text;
      state.step = 'await_sessions';
      return { type: 'text', text: '請問這個系列總共要開設幾堂課? (請輸入數字)', quickReply: {
items: getCancelMenu() } };
    case 'await_sessions':
      const sessions = parseInt(text, 10);
      if (isNaN(sessions) || sessions <= 0) { return { type: 'text', text: '堂數必須是正整數，請重
新輸入。', quickReply: { items: getCancelMenu() } };
      }
      state.sessions = sessions;
      state.step = 'await_capacity';
      return { type: 'text', text: '請問每堂課的名額限制? (請輸入數字)', quickReply: { items:
getCancelMenu() } };
    case 'await_capacity':
      const capacity = parseInt(text, 10);
      if (isNaN(capacity) || capacity <= 0) { return { type: 'text', text: '名額必須是正整數，請重新

```

```

輸入。', quickReply: { items: getCancelMenu() } }];
    }
    state.capacity = capacity;
    state.step = 'await_points';
    return { type: 'text', text: '請問每堂課需要消耗多少點數？(請輸入數字)', quickReply: {
items: getCancelMenu() } }];
    case 'await_points':
        const points = parseInt(text, 10);
        if (isNaN(points) || points < 0) { return { type: 'text', text: '點數必須是正整數或 0, 請重新
輸入。', quickReply: { items: getCancelMenu() } }];
        }
        state.points_cost = points;
        state.step = 'await_teacher';
        return buildTeacherSelectionCarousel();
    case 'await_confirmation':
        if (text === '✅ 確認新增') {
            return executeDbQuery(async (client) => {
                await client.query('BEGIN');
                try {
                    const prefix = await generateUniqueCoursePrefix(client);
                    let currentDate = new Date();
                    for (let i = 0; i < state.sessions; i++) {
                        const courseDate = getNextDate(state.weekday, state.time, currentDate);
                        const course = { id: `${prefix}${String(i + 1).padStart(2, '0')}`, title: state.title,
time: courseDate.toISOString(), capacity: state.capacity, points_cost: state.points_cost,
students: [], waiting: [], teacher_id: state.teacher_id };
                        await saveCourse(course, client);
                        currentDate = new Date(courseDate.getTime() +
CONSTANTS.TIME.ONE_DAY_IN_MS);
                    }
                    await client.query('COMMIT');
                    delete pendingCourseCreation[userId];
                    return `✅ 成功新增「${state.title}」系列共 ${state.sessions} 堂課！`;
                } catch (e) {
                    await client.query('ROLLBACK');
                    console.error("新增課程系列失敗", e);
                    delete pendingCourseCreation[userId];
                    return '新增課程時發生錯誤, 請稍後再試。';
                }
            });
        } else {
            return '請點擊「✅ 確認新增」或「❌ 取消操作」。';
        }
    }
}

```

```

    }
  } else if (pendingManualAdjust[userId]) {
    const state = pendingManualAdjust[userId];
    switch (state.step) {
      case 'await_student_search':
        const res = await executeDbQuery(client =>
          client.query(`SELECT id, name, picture_url FROM users WHERE role = 'student' AND
            (LOWER(name) LIKE $1 OR id = $2) LIMIT 10`, [`${text.toLowerCase()}%`, text])
        );
        if (res.rows.length === 0) { return { type: 'text', text: `找不到學員「${text}」。請重新輸入或取消操作。`, quickReply: { items: getCancelMenu() } }; }
        const placeholder_avatar = 'https://i.imgur.com/8l1Yd2S.png';
        const userBubbles = res.rows.map(u => ({ type: 'bubble', body: { type: 'box', layout:
          'horizontal', spacing: 'md', contents: [ { type: 'image', url: u.picture_url || placeholder_avatar, size:
          'md', aspectRatio: '1:1', aspectMode: 'cover' }, { type: 'box', layout: 'vertical', flex: 3,
          justifyContent: 'center', contents: [ { type: 'text', text: u.name, weight: 'bold', size: 'lg', wrap: true },
          { type: 'text', text: `ID: ${u.id}`, size: 'xss', color: '#AAAAAA', margin: 'sm' } ] } ], footer: { type:
          'box', layout: 'vertical', contents: [ { type: 'button', style: 'primary', color: '#1A759F', height: 'sm',
          action: { type: 'postback', label: '選擇此學員', data:
          `action=select_student_for_adjust&studentId=${u.id}` } } ] } } }));
        return { type: 'flex', altText: '請選擇要調整點數的學員', contents: { type: 'carousel', contents:
          userBubbles } };
      case 'await_operation':
        if (text === CONSTANTS.COMMANDS.TEACHER.ADD_POINTS || text ===
          CONSTANTS.COMMANDS.TEACHER.DEDUCT_POINTS) { state.operation = text ===
          CONSTANTS.COMMANDS.TEACHER.ADD_POINTS ? 'add' : 'deduct'; state.step =
          'await_amount'; return { type: 'text', text: `請輸入要 ${text ===
          CONSTANTS.COMMANDS.TEACHER.ADD_POINTS ? '增加' : '扣除'} 的點數數量 (純數字):`,
          quickReply: { items: getCancelMenu() } }; }
        else { return `請點擊 `+ 加點` 或 `- 扣點` 按鈕。`; }
      case 'await_amount': const amount = parseInt(text, 10);
        if (isNaN(amount) || amount <= 0) { return { type: 'text', text: '點數格式不正確，請輸入一個
          大於 0 的正整數。', quickReply: { items: getCancelMenu() } }; }
        state.amount = amount; state.step = 'await_reason'; return { type: 'text', text: '請輸入調整
          原因 (例如：活動獎勵、課程補償等):', quickReply: { items: getCancelMenu() } };
      case 'await_reason': state.reason = text; state.step = 'await_confirmation'; const opText =
        state.operation === 'add' ? `增加 ${state.amount} 點` : `扣除 ${state.amount} 點`; const
        summary = `請確認調整內容：\n\n對象：${state.targetStudent.name}\n操作：${opText}\n原因：
        ${state.reason}`;
        return { type: 'text', text: summary, quickReply: { items: [ { type: 'action', action: { type:

```

```

'message', label: CONSTANTS.COMMANDS.TEACHER.CONFIRM_MANUAL_ADJUST, text:
CONSTANTS.COMMANDS.TEACHER.CONFIRM_MANUAL_ADJUST } }, { type: 'action',
action: { type: 'message', label: CONSTANTS.COMMANDS.GENERAL.CANCEL, text:
CONSTANTS.COMMANDS.GENERAL.CANCEL } } ] } }];
    case 'await_confirmation':
      if (text === CONSTANTS.COMMANDS.TEACHER.CONFIRM_MANUAL_ADJUST) {
        return executeDbQuery(async (clientDB) => {
          await clientDB.query('BEGIN');
          try {
            const studentRes = await clientDB.query('SELECT * FROM users WHERE id = $1
FOR UPDATE', [state.targetStudent.id]);
            const student = studentRes.rows[0];
            const newPoints = state.operation === 'add' ? student.points + state.amount :
student.points - state.amount;
            if (newPoints < 0) {
              await clientDB.query('ROLLBACK');
              delete pendingManualAdjust[userId];
              return `操作失敗: 學員 ${student.name} 的點數不足以扣除 ${state.amount} 點。`;
            }
            const historyEntry = { action: `手動調整: ${state.operation === 'add' ? '+' :
'-'} ${state.amount}點`, reason: state.reason, time: new Date().toISOString(), operator:
user.name };
            const newHistory = student.history ? [...student.history, historyEntry] : [historyEntry];
            await clientDB.query('UPDATE users SET points = $1, history = $2 WHERE id = $3',
[newPoints, JSON.stringify(newHistory), student.id]);
            // 在 orders 資料表中也新增一筆紀錄
            const orderId = `MA-${Date.now()}`;
            // MA for Manual Adjust
            const pointsChange = state.operation === 'add' ? state.amount : -state.amount;

            // [修改] 將存入 last_5_digits 的值改為一個簡短的代碼
            const reasonForOrder = `手動`;
            await clientDB.query(
              `INSERT INTO orders (order_id, user_id, user_name, points, amount, last_5_digits,
status, timestamp)
              VALUES ($1, $2, $3, $4, $5, $6, $7, $8)`,
              [orderId, student.id, student.name, pointsChange, 0, reasonForOrder, 'completed',
new Date().toISOString()]
            );

            const opTextForStudent = state.operation === 'add' ? `增加了 ${state.amount}` : `扣除
了 ${state.amount}`;
            const notifyMessage = { type: 'text', text: `🔔 點數異動通知\n老師 ${user.name} 為您

```

```

    ${opTextForStudent} 點。原因: ${state.reason}\n您目前的點數為: ${newPoints} 點。`;
    await enqueuePushTask(student.id, notifyMessage);

    await clientDB.query('COMMIT');
    delete pendingManualAdjust[userId];
    return `✅ 已成功為學員 ${student.name} ${state.operation === 'add' ? '增加' : '扣除'}
    ${state.amount} 點。`;
  } catch (e) {
    await clientDB.query('ROLLBACK');
    console.error('手動調整點數失敗:', e);
    delete pendingManualAdjust[userId];
    return `❌ 操作失敗, 資料庫發生錯誤, 請稍後再試。`;
  }
});
}
break;
}
} else if (pendingManualAdjust[userId]) {
  // ... pendingManualAdjust 的程式碼 ...
} else if (pendingManualAdjustSearch[userId]) {
  const searchQuery = text;
  delete pendingManualAdjustSearch[userId];

  const res = await executeDbQuery(client =>
    client.query(`SELECT id, name, picture_url FROM users WHERE role = 'student' AND
    (LOWER(name) LIKE $1 OR id = $2) LIMIT 10`, [`${searchQuery.toLowerCase()}%`,
    searchQuery])
  );
  if (res.rows.length === 0) {
    return { type: 'text', text: `找不到學員「${searchQuery}」。請重新操作。` };
  }

  return showStudentSelectionForAdjustHistory(res.rows, searchQuery);
}

// [新增] 購點歷史查詢的處理
else if (pendingPurchaseHistorySearch[userId]) {
  const searchQuery = text;
  delete pendingPurchaseHistorySearch[userId];

  const res = await executeDbQuery(client =>
    client.query(`SELECT id, name, picture_url FROM users WHERE role = 'student' AND
    (LOWER(name) LIKE $1 OR id = $2) LIMIT 10`, [`${searchQuery.toLowerCase()}%`,

```

```

searchQuery])
);
if (res.rows.length === 0) {
    return { type: 'text', text: `找不到學員「${searchQuery}」。請重新操作。` };
}

return showStudentSelectionForPurchaseHistory(res.rows);
}

// [新增] 兌換歷史查詢的處理
else if (pendingExchangeHistorySearch[userId]) {
    const searchQuery = text;
    delete pendingExchangeHistorySearch[userId];

    const res = await executeDbQuery(client =>
        client.query(`SELECT id, name, picture_url FROM users WHERE role = 'student' AND
(LOWER(name) LIKE $1 OR id = $2) LIMIT 10`, [`${searchQuery.toLowerCase()}%`,
searchQuery])
    );
    if (res.rows.length === 0) {
        return { type: 'text', text: `找不到學員「${searchQuery}」。請重新操作。` };
    }

    return showStudentSelectionForExchangeHistory(res.rows);
}

// [新增] 歷史留言查詢的處理
else if (pendingMessageHistorySearch[userId]) {
    const searchQuery = text;
    delete pendingMessageHistorySearch[userId];

    const res = await executeDbQuery(client =>
        client.query(`SELECT id, name, picture_url FROM users WHERE role = 'student' AND
(LOWER(name) LIKE $1 OR id = $2) LIMIT 10`, [`${searchQuery.toLowerCase()}%`,
searchQuery])
    );
    if (res.rows.length === 0) {
        return { type: 'text', text: `找不到學員「${searchQuery}」。請重新操作。` };
    }

    return showStudentSelectionForMessageHistory(res.rows);
}

else if (pendingStudentSearchQuery[userId]) {

```



```

const searchQuery = text;
delete pendingStudentSearchQuery[userId];
return showStudentSearchResults(searchQuery, 1);
} else if (pendingReply[userId]) {
  const state = pendingReply[userId];
  try {
    await executeDbQuery(client =>
      client.query("UPDATE feedback_messages SET status = 'replied', teacher_reply = $1,
is_student_read = false WHERE id = $2", [text, state.msgId])
    );
    const studentId = state.studentId;
    const originalMessage = state.originalMessage;
    delete pendingReply[userId];
    const notifyMessage = { type: 'text', text: `老師回覆了您的留言:\n\n【您的留言】
\n${originalMessage}\n\n【老師的回覆】\n${text}` };
    await enqueuePushTask(studentId, notifyMessage);
    return '✅ 已成功回覆學員的留言。';
  } catch (err) {
    delete pendingReply[userId];
    throw err;
  }
} else if (pendingMessageSearchQuery[userId]) {
  const searchQuery = text;
  delete pendingMessageSearchQuery[userId];
  return showHistoricalMessages(searchQuery, 1);
} else if (pendingTeacherProfileEdit[userId]) {
  const state = pendingTeacherProfileEdit[userId];
  const step = state.step;
  if (state.type === 'create') {
    switch (step) {
      case 'await_name':
        state.profileData.name = text;
        state.step = 'await_bio';
        setupConversationTimeout(userId, pendingTeacherProfileEdit,
'pendingTeacherProfileEdit', (u) => { enqueuePushTask(u, { type: 'text', text: '建立檔案操作逾時
, 自動取消.' }); });
        return { type: 'text', text: '姓名已收到！\n接下來，請輸入您的個人簡介(例如您的教學
風格、專業認證等)，或輸入「無」表示留空:', quickReply: { items: getCancelMenu() } };
      case 'await_bio':
        state.profileData.bio = text.trim().toLowerCase() === '無' ? null : text;
        state.step = 'await_image';
        setupConversationTimeout(userId, pendingTeacherProfileEdit,
'pendingTeacherProfileEdit', (u) => { enqueuePushTask(u, { type: 'text', text: '建立檔案操作逾時

```

```

, 自動取消。'})); });
    return { type: 'text', text: '簡介已收到！\n最後，請直接上傳一張您想顯示的個人照片，
或輸入「無」使用預設頭像:', quickReply: { items: getCancelMenu() } } };
    case 'await_image':
      let imageUrl = null;
      if (event.message.type === 'image') {
        try {
          const imageResponse = await
axios.get(`https://api-data.line.me/v2/bot/message/${event.message.id}/content`, { headers: {
'Authorization': `Bearer ${process.env.CHANNEL_ACCESS_TOKEN}` }, responseType:
'arraybuffer' });
          const imageBuffer = Buffer.from(imageResponse.data, 'binary');
          const uploadResponse = await imagekit.upload({ file: imageBuffer, fileName:
`teacher_${userId}.jpg`, useUniqueFileName: true, folder: "yoga_teachers" });
          imageUrl = uploadResponse.url;
        } catch (err) {
          console.error('上傳老師照片至 ImageKit 失敗', err);
          delete pendingTeacherProfileEdit[userId];
          return '❌ 圖片上傳失敗，請重新開始建立檔案流程。';
        }
      } else if (event.message.type === 'text' && text.trim().toLowerCase() !== '無') {
        return { type: 'text', text: '格式錯誤，請直接上傳一張照片，或輸入「無」。',
quickReply: { items: getCancelMenu() } } };
      }
      state.profileData.image_url = imageUrl;
      state.step = 'await_confirmation';
      state.newData = state.profileData;
      return buildProfileConfirmationMessage(userId, state.newData);
    }
  }
} else if (state.type === 'edit') {
  const field = step.replace('await_', '');
  let value;
  if (field === 'image_url') {
    if (event.message.type !== 'image') {
      return { type: 'text', text: '格式錯誤，請直接上傳一張照片。', quickReply: { items:
getCancelMenu() } } };
    }
    try {
      const imageResponse = await
axios.get(`https://api-data.line.me/v2/bot/message/${event.message.id}/content`, { headers: {
'Authorization': `Bearer ${process.env.CHANNEL_ACCESS_TOKEN}` }, responseType:
'arraybuffer' });

```

```

        const imageBuffer = Buffer.from(imageResponse.data, 'binary');
        const uploadResponse = await imagekit.upload({ file: imageBuffer, fileName:
`teacher_${userId}.jpg`, useUniqueFileName: true, folder: "yoga_teachers" });
        value = uploadResponse.url;
      } catch (err) {
        console.error('更新老師照片至 ImageKit 失敗', err);
        return '❌ 圖片上傳失敗, 請稍後再試。';
      }
    } else {
      value = text;
    }
    state.newData = { [field]: value };
    state.step = 'await_confirmation';
    return buildProfileConfirmationMessage(userId, state.newData);
  }
}

```

```

// === Refactored Command Handling ===
const commandFunction = teacherCommandMap[text];
if (commandFunction) {
  return commandFunction(event, user);
} else {
  return handleUnknownTeacherCommand(text);
}
}

```

```

async function handleAdminCommands(event, userId) {
  const text = event.message.text ? event.message.text.trim().normalize() : "";
  const user = await getUser(userId);
  if (pendingTeacherAddition[userId]) {
    const state = pendingTeacherAddition[userId];
    switch (state.step) {
      case 'await_student_info':
        const studentSearchRes = await executeDbQuery(client =>
          client.query(`SELECT id, name, role, picture_url FROM users WHERE role = 'student'
AND (LOWER(name) LIKE $1 OR id = $2) LIMIT 25`, [`%${text.toLowerCase()}%`, text])
        );
        if (studentSearchRes.rows.length === 0) {
          return { type: 'text', text: `找不到與「${text}」相關的學員。請重新輸入或取消操作。`,
            quickReply: { items: getCancelMenu() } };
        }
      }
    }
  }
}

```

```

const placeholder_avatar = 'https://i.imgur.com/8l1Yd2S.png';

```

```

const userBubbles = studentSearchRes.rows.map(s => ({
  type: 'bubble',
  body: {
    type: 'box',
    layout: 'horizontal',
    spacing: 'md',
    contents: [
      {
        type: 'image',
        url: s.picture_url || placeholder_avatar,
        size: 'md',
        aspectRatio: '1:1',
        aspectMode: 'cover'
      },
      {
        type: 'box',
        layout: 'vertical',
        flex: 3,
        justifyContent: 'center',
        contents: [
          { type: 'text', text: s.name, weight: 'bold', size: 'lg', wrap: true },
          { type: 'text', text: `ID: ${formatIdForDisplay(s.id)}`, size: 'xxs', color:
'#AAAAAA', margin: 'sm', wrap: true }
        ]
      }
    ]
  },
  footer: {
    type: 'box',
    layout: 'vertical',
    contents: [{
      type: 'button',
      style: 'primary',
      color: '#52B69A',
      height: 'sm',
      action: {
        type: 'postback',
        label: '選擇此學員',
        data:
`action=select_student_for_auth&targetId=${s.id}&targetName=${encodeURIComponent(s.name)}`
      }
    ]
  }
})

```

```

    }
  }));
  delete pendingTeacherAddition[userId];

  return {
    type: 'flex',
    altText: '請選擇要授權的學員',
    contents: {
      type: 'carousel',
      contents: userBubbles
    }
  };
case 'await_confirmation':
  if (text === CONSTANTS.COMMANDS.ADMIN.CONFIRM_ADD_TEACHER) {
    const targetUser = await getUser(state.targetUser.id);
    targetUser.role = 'teacher';
    targetUser.approved_by = userId;
    await saveUser(targetUser);
    delete pendingTeacherAddition[userId];

    const notifyMessage = { type: 'text', text: '恭喜！您的身份已被管理者授權為「老師」。'};
    await enqueuePushTask(targetUser.id, notifyMessage).catch(e => console.error(e));
    if (TEACHER_RICH_MENU_ID) await client.linkRichMenuToUser(targetUser.id,
TEACHER_RICH_MENU_ID);
    return `✅ 已成功授權「${targetUser.name}」為老師。`;
  } else {
    return '請點擊確認或取消按鈕。';
  }
}
} else if (pendingTeacherRemoval[userId]) {
  const state = pendingTeacherRemoval[userId];
  switch (state.step) {
    case 'await_confirmation':
      if (text === CONSTANTS.COMMANDS.ADMIN.CONFIRM_REMOVE_TEACHER) {
        const targetUser = await getUser(state.targetUser.id);
        targetUser.role = 'student';
        targetUser.approved_by = null;
        await saveUser(targetUser);
        delete pendingTeacherRemoval[userId];

        const notifyMessage = { type: 'text', text: '通知：您的「老師」身份已被管理者移除，已切換
為學員身份。'};
        await enqueuePushTask(targetUser.id, notifyMessage).catch(e => console.error(e));

```

```

        if(STUDENT_RICH_MENU_ID) await client.linkRichMenuToUser(targetUser.id,
STUDENT_RICH_MENU_ID);
        return `✅ 已成功將「${targetUser.name}」的身份移除，該用戶已變為學員。`;
    } else {
        return '請點擊確認或取消按鈕。';
    }
}
} else {
    if (text === CONSTANTS.COMMANDS.ADMIN.PANEL) {
        const failedTasksCount = await executeDbQuery(async (client) => {
            const res = await client.query("SELECT COUNT(*) FROM failed_tasks");
            return parseInt(res.rows[0].count, 10);
        });
        let failedTasksLabel = '失敗任務管理';
        if (failedTasksCount > 0) {
            failedTasksLabel += ` (${failedTasksCount})`;
        }

        const adminMenu = [
            { type: 'action', action: { type: 'message', label: '系統狀態',
text:CONSTANTS.COMMANDS.ADMIN.SYSTEM_STATUS } },
            { type: 'action', action: { type: 'message', label: failedTasksLabel, text:
CONSTANTS.COMMANDS.ADMIN.FAILED_TASK_MANAGEMENT } },
            { type: 'action', action: { type: 'message', label: '授權老師', text:
CONSTANTS.COMMANDS.ADMIN.ADD_TEACHER } },
            { type: 'action', action: { type: 'message', label: '移除老師', text:
CONSTANTS.COMMANDS.ADMIN.REMOVE_TEACHER } },
            { type: 'action', action: { type: 'message', label: '模擬學員身份', text:
CONSTANTS.COMMANDS.ADMIN.SIMULATE_STUDENT } },
            { type: 'action', action: { type: 'message', label: '模擬老師身份', text:
CONSTANTS.COMMANDS.ADMIN.SIMULATE_TEACHER } },
            { type: 'action', action: { type: 'message', label: '切換推播通知', text:
CONSTANTS.COMMANDS.ADMIN.TOGGLE_NOTIFICATIONS } }
        ];
        const currentStatus = await getNotificationStatus();
        const statusText = currentStatus ? '【目前為：開啟】' : '【目前為：關閉】';
        return { type: 'text', text: `請選擇管理者功能：\n\n開發者推播通知 ${statusText}`, quickReply:
{ items: adminMenu } };
    }
    else if (text === CONSTANTS.COMMANDS.ADMIN.SYSTEM_STATUS) {
        return showSystemStatus();
    }
    else if (text === CONSTANTS.COMMANDS.ADMIN.FAILED_TASK_MANAGEMENT) {

```

```

    return showFailedTasks(1);
  }
  else if (text === CONSTANTS.COMMANDS.ADMIN.TOGGLE_NOTIFICATIONS) {
    const currentStatus = await getNotificationStatus();
    const newStatus = !currentStatus;
    await executeDbQuery(async (db) => {
      await db.query(
        `INSERT INTO system_settings (setting_key, setting_value, updated_at) VALUES
('notifications_enabled', $1, NOW())
ON CONFLICT (setting_key) DO UPDATE SET setting_value = $1, updated_at =
NOW()`,
        [newStatus.toString()]
      );
    });
    simpleCache.clear('notifications_enabled');
    const statusText = newStatus ? '【開啟】' : '【關閉】';
    return `🟢 開發者推播通知功能已設定為 ${statusText}。\\n此設定只會影響傳送給老師/管理
員的通知。`;
  }
  else if (text === CONSTANTS.COMMANDS.ADMIN.ADD_TEACHER) {
    pendingTeacherAddition[userId] = { step: 'await_student_info' };
    setupConversationTimeout(userId, pendingTeacherAddition, 'pendingTeacherAddition', (u)
=> {
      const timeoutMessage = { type: 'text', text: '授權老師操作逾時，自動取消。'};
      enqueuePushTask(u, timeoutMessage).catch(e => console.error(e));
    });
    return { type: 'text', text: '請輸入您想授權為老師的「學員」姓名或 User ID:', quickReply: {
items: getCancelMenu() } };
  }
  else if (text === CONSTANTS.COMMANDS.ADMIN.REMOVE_TEACHER) {
    return showTeacherListForRemoval(1);
  }
  else if (text === CONSTANTS.COMMANDS.ADMIN.SIMULATE_STUDENT) {
    user.role = 'student';
    await saveUser(user);
    if(STUDENT_RICH_MENU_ID) await client.linkRichMenuToUser(userId,
STUDENT_RICH_MENU_ID);
    return '您已切換為「學員」模擬身份。\\n若要返回，請手動輸入「@管理模式」。';
  }
  else if (text === CONSTANTS.COMMANDS.ADMIN.SIMULATE_TEACHER) {
    user.role = 'teacher';
    await saveUser(user);
    if(TEACHER_RICH_MENU_ID) await client.linkRichMenuToUser(userId,
TEACHER_RICH_MENU_ID);
    return '您已切換為「老師」模擬身份。\\n若要返回，請手動輸入「@管理模式」。';
  }
}
}

```

```
}
```

```
async function handleStudentCommands(event, userId) {
```

```
  const text = event.message.text ?
```

```
  event.message.text.trim().normalize() : '';
```

```
  const user = await getUser(userId);
```

```
  // [V35.5 新增] 處理商品訂單的後五碼回報
```

```
  if (pendingShopPayment[userId]) {
```

```
    const state = pendingShopPayment[userId];
```

```
    if (!/^d{5}$/.test(text)) {
```

```
      return {
```

```
        type: 'text',
```

```
        text: '格式錯誤，請輸入5位數字的匯款帳號後五碼。',
```

```
        quickReply: { items: getCancelMenu() }
```

```
      };
```

```
    }
```

```
    const wasSuccessful = await executeDbQuery(async (client) => {
```

```
      const res = await client.query(
```

```
        "UPDATE product_orders SET last_5_digits = $1, status = 'pending_confirmation',
```

```
        updated_at = NOW() WHERE order_uid = $2 AND user_id = $3 AND status =
```

```
        'pending_payment' RETURNING product_name",
```

```
        [text, state.orderUID, userId]
```

```
      );
```

```
      return res.rowCount > 0 ? res.rows[0].product_name : null;
```

```
    });
```

```
    delete pendingShopPayment[userId];
```

```
    if (wasSuccessful) {
```

```
      const productName = wasSuccessful;
```

```
      const notifyMessage = { type: 'text', text: `🔔 付款回報通知\n學員 ${user.name} 已回報「  
${productName}」訂單的匯款資訊。後五碼: ${text}\n請至「訂單管理」審核。`};
```

```
      await notifyAllTeachers(notifyMessage);
```

```
      return `感謝您！已收到您的匯款後五碼「${text}」。我們將盡快為您審核，審核通過後您  
會收到通知。`;
```

```
    } else {
```

```
      return '找不到您的待付款訂單，或訂單狀態已變更，請重新操作。';
```

```
    }
```

```
  }
```

```
  const purchaseFlowResult = await handlePurchaseFlow(event, userId);
```



```

if (purchaseFlowResult.handled) {
  return purchaseFlowResult.reply;
}

if (pendingBookingConfirmation[userId]) {
  const state = pendingBookingConfirmation[userId];
  const course = await getCourse(state.course_id);
  if (!course && state.type !== 'product_purchase') {
    delete pendingBookingConfirmation[userId];
    return '抱歉，找不到該課程，可能已被老師取消。';
  }

  switch (state.type) {
    case 'cancel_book':
      if (text === CONSTANTS.COMMANDS.STUDENT.CONFIRM_CANCEL_BOOKING) {
        return executeDbQuery(async (client) => {
          await client.query('BEGIN');
          try {
            const userForUpdateRes = await client.query('SELECT points, history FROM
users WHERE id = $1 FOR UPDATE', [userId]);
            const courseForUpdateRes = await client.query('SELECT students, waiting,
points_cost, title FROM courses WHERE id = $1 FOR UPDATE', [state.course_id]);
            if (courseForUpdateRes.rows.length === 0) {
              await client.query('ROLLBACK');
              delete pendingBookingConfirmation[userId];
              return '取消失敗，找不到此課程。';
            }
            const currentCourse = courseForUpdateRes.rows[0];
            const newStudents = [...currentCourse.students];
            const indexToRemove = newStudents.indexOf(userId);

            if (indexToRemove === -1) {
              await client.query('ROLLBACK');
              delete pendingBookingConfirmation[userId];
              return '您尚未預約此課程。';
            }
            newStudents.splice(indexToRemove, 1);
            const newPoints = userForUpdateRes.rows[0].points +
currentCourse.points_cost;
            const historyEntry = { action: `取消預約 (1位):
${getCourseMainTitle(currentCourse.title)}`, pointsChange: +currentCourse.points_cost, time:
new Date().toISOString() };

```

```

    const userHistory = userForUpdateRes.rows[0].history || [];
    const newHistory = [...userHistory, historyEntry];
    await client.query('UPDATE users SET points = $1, history = $2 WHERE id = $3',
[newPoints, JSON.stringify(newHistory), userId]);
    let newWaiting = currentCourse.waiting || [];
    if (newWaiting.length > 0) {
        const promotedUserId = newWaiting.shift();
        newStudents.push(promotedUserId);
        const notifyMessage = { type: 'text', text: `🎉 候補成功通知 🎉\n您候補的課程
「${getCourseMainTitle(currentCourse.title)}」已有空位，已為您自動預約成功！`;
        await enqueuePushTask(promotedUserId, notifyMessage);
    }
    await client.query('UPDATE courses SET students = $1, waiting = $2 WHERE id
= $3', [newStudents, newWaiting, state.course_id]);
    await client.query('COMMIT');
    delete pendingBookingConfirmation[userId];

    const remainingBookings = newStudents.filter(id => id === userId).length;
    let replyMsg = `✅ 已為您取消 1 位「${getCourseMainTitle(currentCourse.title)}」
的預約，並歸還 ${currentCourse.points_cost} 點。`;
    if (remainingBookings > 0) { replyMsg += `\n您在此課程尚有
${remainingBookings} 位預約。`;
    }
    return replyMsg;
} catch (e) {
    await client.query('ROLLBACK');
    console.error('取消預約失敗:', e);
    delete pendingBookingConfirmation[userId];
    return '取消預約時發生錯誤，請稍後再試。';
}
});
} else if (text === CONSTANTS.COMMANDS.GENERAL.CANCEL) {
    delete pendingBookingConfirmation[userId];
    return '已放棄取消操作。';
}
break;
case 'cancel_wait':
    if (text === CONSTANTS.COMMANDS.STUDENT.CONFIRM_CANCEL_WAITING) {
        const newWaitingList = course.waiting.filter(id => id !== userId);
        await saveCourse({ ...course, waiting: newWaitingList });
        delete pendingBookingConfirmation[userId];
        return `✅ 已為您取消「${course.title}」的候補。`;
    } else if (text === CONSTANTS.COMMANDS.GENERAL.CANCEL) {

```

```

        delete pendingBookingConfirmation[userId];
        return '已放棄取消操作。';
    }
    break;
case 'product_purchase':
    if (text === CONSTANTS.COMMANDS.GENERAL.CANCEL) {
        delete pendingBookingConfirmation[userId];
        return '已取消兌換。';
    }
    break;
}
} else if (pendingFeedback[userId]) {
    const feedbackState = pendingFeedback[userId];
    if (feedbackState.step === 'await_message') {
        await executeDbQuery(client =>
            client.query('INSERT INTO feedback_messages (id, user_id, user_name, message,
timestamp) VALUES ($1, $2, $3, $4, NOW())', ['F${Date.now()}', userId, user.name, text])
        );
        delete pendingFeedback[userId];
        if (TEACHER_ID) {
            const notifyMessage = { type: 'text', text: `🔔 新留言通知\n來自: ${user.name}\n內容:
${text}\n\n請至「學員管理」->「查看學員留言」回覆。`};
            await notifyAllTeachers(notifyMessage);
        }
        return '感謝您的留言，我們已收到您的訊息，老師會盡快查看！';
    }
} else {
    // --- 處理一般指令 ---
    if (text === CONSTANTS.COMMANDS.STUDENT.BOOK_COURSE) {
        return showAvailableCourses(userId, 1);
    } else if (text === CONSTANTS.COMMANDS.STUDENT.MY_COURSES) {
        return showMyCourses(userId, 1);
    } else if (text === CONSTANTS.COMMANDS.STUDENT.LATEST_ANNOUNCEMENT) {
        return executeDbQuery(async (client) => {
            const res = await client.query('SELECT * FROM announcements ORDER BY created_at
DESC LIMIT 1');
            if (res.rows.length === 0) { return '目前沒有任何公告。'; }
            const announcement = res.rows[0];
            return { type: 'flex', altText: '最新公告', contents: { type: 'bubble', header: { type: 'box',
layout: 'vertical', backgroundColor: '#de5246', contents: [ { type: 'text', text: '!! 最新公告', color:
'#####', weight: 'bold', size: 'lg' } ] }, body: { type: 'box', layout: 'vertical', contents: [ { type: 'text',
text: announcement.content, wrap: true } ] }, footer: { type: 'box', layout: 'vertical', contents: [ {
type: 'text', text: `由 ${announcement.creator_name} 於`

```

```

    ${formatDateTime(announcement.created_at)} 發布`, size: 'xs', color: '#aaaaaa', align: 'center' }
  ] } } };
  });
  } else if (text === CONSTANTS.COMMANDS.STUDENT.ADD_NEW_MESSAGE) {
    pendingFeedback[userId] = { step: 'await_message' };
    setupConversationTimeout(userId, pendingFeedback, 'pendingFeedback', (u) => {
      const timeoutMessage = { type: 'text', text: '留言逾時，自動取消。'};
      enqueuePushTask(u, timeoutMessage).catch(e => console.error(e));
    });
    return { type: 'text', text: '請輸入您想對老師說的話，或點選「取消」。', quickReply: { items:
getCancelMenu() } } };
  } else if (text === CONSTANTS.COMMANDS.STUDENT.CONTACT_US) {
    const unreadCount = await executeDbQuery(client =>
      client.query("SELECT COUNT(*) FROM feedback_messages WHERE user_id = $1 AND
status = 'replied' AND is_student_read = false", [userId])
    ).then(res => parseInt(res.rows[0].count, 10));
    let historyLabel = '📖 查詢歷史留言';
    if (unreadCount > 0) {
      historyLabel += ` (${unreadCount})`;
    }

    return {
      type: 'flex', altText: '聯絡我們',
      contents: {
        type: 'bubble', size: 'giga',
        header: { type: 'box', layout: 'vertical', contents: [{ type: 'text', text: '☎ 聯絡我們', color:
'#ffffff', weight: 'bold', size: 'lg'}], backgroundColor: '#34A0A4', paddingTop: 'lg', paddingBottom:
'lg' },
        body: { type: 'box', layout: 'vertical', spacing: 'md', paddingAll: 'lg',
          contents: [
            { type: 'button', style: 'secondary', height: 'sm', action: { type: 'postback', label: '📝 新
增留言', data:
`action=run_command&text=${encodeURIComponent(CONSTANTS.COMMANDS.STUDENT.A
DD_NEW_MESSAGE)}` } },
            { type: 'button', style: 'secondary', height: 'sm', action: { type: 'postback', label:
historyLabel, data: `action=view_my_messages&page=1` } }
          ]
        }
      }
    };
  } else if (text === CONSTANTS.COMMANDS.STUDENT.POINTS || text ===
CONSTANTS.COMMANDS.STUDENT.CHECK_POINTS) {
    if (pendingPurchase[userId]?.step !== 'input_last5' && pendingPurchase[userId]?.step !==

```

```

'edit_last5') delete pendingPurchase[userId];
    delete pendingBookingConfirmation[userId];
    return buildPointsMenuFlex(userId);
} else if (text === CONSTANTS.COMMANDS.STUDENT.BUY_POINTS) {
    const hasPendingOrder = await executeDbQuery(async (client) => {
        const existingOrderRes = await client.query(`SELECT 1 FROM orders WHERE user_id
= $1 AND (status = 'pending_payment' OR status = 'pending_confirmation' OR status =
'rejected') LIMIT 1`, [userId]);
        return existingOrderRes.rows.length > 0;
    });
    if (hasPendingOrder) {
        const flexMenu = await buildPointsMenuFlex(userId);
        return [{type: 'text', text: '您目前尚有未完成的訂單，請先處理該筆訂單。'}, flexMenu];
    }

    return buildBuyPointsFlex();
} else if (text === CONSTANTS.COMMANDS.STUDENT.PURCHASE_HISTORY) {
    return showPurchaseHistory(userId, 1);
} else if (text === CONSTANTS.COMMANDS.STUDENT.SHOP) {
    return {
        type: 'flex', altText: '活動商城',
        contents: {
            type: 'bubble', size: 'giga',
            header: { type: 'box', layout: 'vertical', contents: [{ type: 'text', text: '🛒 活動商城', color:
'#ffffff', weight: 'bold', size: 'lg'}], backgroundColor: '#34A0A4', paddingTop: 'lg', paddingBottom:
'lg' },
            body: { type: 'box', layout: 'vertical', spacing: 'md', paddingAll: 'lg',
                contents: [
                    { type: 'button', style: 'secondary', height: 'sm', action: { type: 'postback', label: '🛒
瀏覽商品', data:
`action=run_command&text=${encodeURIComponent(CONSTANTS.COMMANDS.STUDENT.V
IEW_SHOP_PRODUCTS)}` } },
                    { type: 'button', style: 'secondary', height: 'sm', action: { type: 'postback', label: '📄
我的購買紀錄', data:
`action=run_command&text=${encodeURIComponent(CONSTANTS.COMMANDS.STUDENT.E
XCHANGE_HISTORY)}` } }
                ]
            }
        }
    };
} else if (text === CONSTANTS.COMMANDS.STUDENT.VIEW_SHOP_PRODUCTS) {
    return showShopProducts(1);
} else if (text === CONSTANTS.COMMANDS.STUDENT.EXCHANGE_HISTORY) {

```

```

    return showStudentExchangeHistory(userId, 1);
  } else if (text === CONSTANTS.COMMANDS.STUDENT.INPUT_LAST5_CARD_TRIGGER ||
text === CONSTANTS.COMMANDS.STUDENT.EDIT_LAST5_CARD_TRIGGER) {
    const orderId = await executeDbQuery(async (client) => {
      const statusFilter = text ===
CONSTANTS.COMMANDS.STUDENT.EDIT_LAST5_CARD_TRIGGER ?
"pending_confirmation", 'rejected' : "pending_payment";
      const orderRes = await client.query(`SELECT order_id FROM orders WHERE user_id =
$1 AND status IN (${statusFilter}) ORDER BY timestamp DESC LIMIT 1`, [userId]);
      return orderRes.rows.length > 0 ? orderRes.rows[0].order_id : null;
    });

    if (orderId) {
      const step = text ===
CONSTANTS.COMMANDS.STUDENT.INPUT_LAST5_CARD_TRIGGER ? 'input_last5' :
'edit_last5';
      pendingPurchase[userId] = { step: step, data: { order_id: orderId } };
      setupConversationTimeout(userId, pendingPurchase, 'pendingPurchase', (u) => {
        const timeoutMessage = { type: 'text', text: '輸入後五碼逾時，自動取消。'};
        enqueuePushTask(u, timeoutMessage).catch(e => console.error(e));
      });
      return { type: 'text', text: '請輸入您的匯款帳號後五碼 (5位數字):', quickReply: { items:
getCancelMenu() } };
    } else {
      return '您目前沒有需要執行此操作的訂單。';
    }
  } else {
    let studentSuggestion = '我不懂您的意思耶 😞\n您可以試試點擊下方的選單按鈕。';
    if (text.startsWith('@')) {
      const closestCommand = findClosestCommand(text, 'student');
      if (closestCommand) {
        studentSuggestion = `找不到指令 "${text}"，您是不是想輸入「${closestCommand}」？`;
      } else {
        studentSuggestion = `哎呀，找不到指令 "${text}"。請檢查一下是不是打錯字了，或直
接點擊選單按鈕最準確喔！`;
      }
    }
    return studentSuggestion;
  }
}
}
}

```

```

async function showStudentSearchResults(query, page) {

```

```

const offset = (page - 1) * CONSTANTS.PAGINATION_SIZE;
return executeDbQuery(async (client) => {
  const res = await client.query(
    `SELECT id, name, picture_url FROM users
     WHERE role = 'student' AND (LOWER(name) LIKE $1 OR id = $2)
     ORDER BY name ASC LIMIT $3 OFFSET $4`,
    [`%${query.toLowerCase()}%`, query, CONSTANTS.PAGINATION_SIZE + 1, offset]
  );

  const hasNextPage = res.rows.length > CONSTANTS.PAGINATION_SIZE;
  const pageUsers = hasNextPage ? res.rows.slice(0, CONSTANTS.PAGINATION_SIZE) :
res.rows;

  if (pageUsers.length === 0 && page === 1) {
    return `找不到與「${query}」相關的學員。`;
  }
  if (pageUsers.length === 0) {
    return '沒有更多搜尋結果了。';
  }

  const placeholder_avatar = 'https://i.imgur.com/8l1Yd2S.png';
  const userBubbles = pageUsers.map(u => ({
    type: 'bubble',
    body: {
      type: 'box',
      layout: 'horizontal',
      spacing: 'md',
      contents: [
        { type: 'image', url: u.picture_url || placeholder_avatar, size: 'md', aspectRatio: '1:1',
aspectMode: 'cover' },
        {
          type: 'box',
          layout: 'vertical',
          flex: 3,
          justifyContent: 'center',
          contents: [
            { type: 'text', text: u.name, weight: 'bold', size: 'lg', wrap: true },
            { type: 'text', text: `ID: ${formatIdForDisplay(u.id)}`, size: 'xxs', color:
'#AAAAAA', margin: 'sm', wrap: true }
          ]
        }
      ]
    }
  }
  ],
  },

```

```

    footer: {
      type: 'box',
      layout: 'vertical',
      contents: [{
        type: 'button',
        style: 'primary',
        color: '#1A759F',
        height: 'sm',
        action: { type: 'postback', label: '查看詳細資料', data:
`action=view_student_details&studentId=${u.id}` }
      }]
    }
  }
  }));
  const paginationBubble = createPaginationBubble('action=student_search_results', page,
hasNextPage, `&query=${encodeURIComponent(query)}`);
  if (paginationBubble) {
    userBubbles.push(paginationBubble);
  }

  return { type: 'flex', altText: `學員搜尋結果: ${query}`, contents: { type: 'carousel', contents:
userBubbles } };
});
}

```

```

async function showStudentSelectionForAdjustHistory(users, originalQuery) {
  const placeholder_avatar = 'https://i.imgur.com/8l1Yd2S.png';
  const userBubbles = users.map(u => ({
    type: 'bubble',
    body: {
      type: 'box',
      layout: 'horizontal',
      spacing: 'md',
      contents: [
        { type: 'image', url: u.picture_url || placeholder_avatar, size: 'md', aspectRatio: '1:1',
aspectMode: 'cover' },
        {
          type: 'box',
          layout: 'vertical',
          flex: 3,
          justifyContent: 'center',
          contents: [
            { type: 'text', text: u.name, weight: 'bold', size: 'lg', wrap: true },
            { type: 'text', text: `ID: ${formatIdForDisplay(u.id)}`, size: 'xxs', color: '#AAAAAA',

```



```

margin: 'sm' }
    ]
  }
]
},
footer: {
  type: 'box',
  layout: 'vertical',
  contents: [{
    type: 'button',
    style: 'primary',
    color: '#1A759F',
    height: 'sm',
    action: { type: 'postback', label: '查看此學員紀錄', data:
`action=view_manual_adjust_history&user_id=${u.id}&page=1` }
  }]
}
});
return {
  type: 'flex',
  altText: `請選擇要查詢紀錄的學員`,
  contents: { type: 'carousel', contents: userBubbles }
};
}

```

// [新增] 顯示學員選單以查詢購點歷史

```

async function showStudentSelectionForPurchaseHistory(users) {
  const placeholder_avatar = 'https://i.imgur.com/8l1Yd2S.png';
  const userBubbles = users.map(u => ({
    type: 'bubble',
    body: {
      type: 'box',
      layout: 'horizontal',
      spacing: 'md',
      contents: [
        { type: 'image', url: u.picture_url || placeholder_avatar, size: 'md', aspectRatio: '1:1',
aspectMode: 'cover' },
        {
          type: 'box',
          layout: 'vertical',
          flex: 3,
          justifyContent: 'center',
          contents: [

```

```

        { type: 'text', text: u.name, weight: 'bold', size: 'lg', wrap: true },
        { type: 'text', text: `ID: ${formatIdForDisplay(u.id)}`, size: 'xxs', color: '#AAAAAA',
margin: 'sm' }
      ]
    }
  ]
},
footer: {
  type: 'box',
  layout: 'vertical',
  contents: [{
    type: 'button',
    style: 'primary',
    color: '#1A759F',
    height: 'sm',
    action: { type: 'postback', label: '查看此學員紀錄', data:
`action=view_purchase_history_as_teacher&user_id=${u.id}&page=1` }
  }]
}
});
return {
  type: 'flex',
  altText: `請選擇要查詢購點紀錄的學員`,
  contents: { type: 'carousel', contents: userBubbles }
};
}

```

// [新增] 顯示學員選單以查詢兌換歷史

```

async function showStudentSelectionForExchangeHistory(users) {
  const placeholder_avatar = 'https://i.imgur.com/8l1Yd2S.png';
  const userBubbles = users.map(u => ({
    type: 'bubble',
    body: {
      type: 'box',
      layout: 'horizontal',
      spacing: 'md',
      contents: [
        { type: 'image', url: u.picture_url || placeholder_avatar, size: 'md', aspectRatio: '1:1',
aspectMode: 'cover' },
        {
          type: 'box',
          layout: 'vertical',
          flex: 3,

```

```

        justifyContent: 'center',
        contents: [
          { type: 'text', text: u.name, weight: 'bold', size: 'lg', wrap: true },
          { type: 'text', text: `ID: ${formatIdForDisplay(u.id)}`, size: 'xss', color: '#AAAAAA',
margin: 'sm' }
        ]
      }
    ]
  },
  footer: {
    type: 'box',
    layout: 'vertical',
    contents: [{
      type: 'button',
      style: 'primary',
      color: '#1A759F',
      height: 'sm',
      action: { type: 'postback', label: '查看此學員紀錄', data:
`action=view_exchange_history_as_teacher&user_id=${u.id}&page=1` }
    }]
  }
}));
return {
  type: 'flex',
  altText: `請選擇要查詢兌換紀錄的學員`,
  contents: { type: 'carousel', contents: userBubbles }
};
}
// [新增] 顯示學員選單以查詢歷史留言
async function showStudentSelectionForMessageHistory(users) {
  const placeholder_avatar = 'https://i.imgur.com/8l1Yd2S.png';
  const userBubbles = users.map(u => ({
    type: 'bubble',
    body: {
      type: 'box',
      layout: 'horizontal',
      spacing: 'md',
      contents: [
        { type: 'image', url: u.picture_url || placeholder_avatar, size: 'md', aspectRatio: '1:1',
aspectMode: 'cover' },
        {
          type: 'box',
          layout: 'vertical',

```

```

        flex: 3,
        justifyContent: 'center',
        contents: [
            { type: 'text', text: u.name, weight: 'bold', size: 'lg', wrap: true },
            { type: 'text', text: `ID: ${formatIdForDisplay(u.id)}`, size: 'xxs', color: '#AAAAAA',
margin: 'sm' }
        ]
    }
}
],
},
footer: {
    type: 'box',
    layout: 'vertical',
    contents: [{
        type: 'button',
        style: 'primary',
        color: '#1A759F',
        height: 'sm',
        action: { type: 'postback', label: '查看此學員留言', data:
`action=view_historical_messages_as_teacher&user_id=${u.id}&page=1` }
    }]
}
});
return {
    type: 'flex',
    altText: `請選擇要查詢留言的學員`,
    contents: { type: 'carousel', contents: userBubbles }
};
}

```

```

async function showAllTeachersList(page) {
    const offset = (page - 1) * CONSTANTS.PAGINATION_SIZE;
    return executeDbQuery(async (client) => {
        console.log('[DEBUG] 準備查詢所有老師...');
        const res = await client.query(
            "SELECT id, name, bio, image_url FROM teachers ORDER BY name ASC"
        );

        console.log(`[DEBUG] 資料庫回傳了 ${res.rows.length} 筆老師資料。`);
        if (res.rows.length > 0) {
            res.rows.forEach(row => console.log(`[DEBUG] 找到老師: ${row.name} (ID: ${row.id})`));
        }
    });
}

```

```

const allTeachers = res.rows;
const hasNextPage = allTeachers.length > offset + CONSTANTS.PAGINATION_SIZE;
const pageTeachers = allTeachers.slice(offset, offset + CONSTANTS.PAGINATION_SIZE);

if (pageTeachers.length === 0 && page === 1) {
  return '目前尚未建立任何老師的公開資訊檔案。';
}
if (pageTeachers.length === 0) {
  return '沒有更多老師的資訊了。';
}

const placeholder_avatar = 'https://i.imgur.com/8l1Yd2S.png';

const teacherBubbles = pageTeachers.map(t => ({
  type: 'bubble',
  hero: { type: 'image', url: t.image_url || placeholder_avatar, size: 'full', aspectRatio: '1:1',
aspectMode: 'cover' },
  body: {
    type: 'box', layout: 'vertical', paddingAll: 'lg',
    contents: [
      { type: 'text', text: t.name, weight: 'bold', size: 'xl', wrap: true },
      { type: 'text', text: t.bio || '這位老師尚未留下簡介。', wrap: true, size: 'sm', color:
'#666666', margin: 'md' },
    ],
  },
})));
const paginationBubble = createPaginationBubble('action=list_all_teachers', page,
hasNextPage);
if (paginationBubble) {
  teacherBubbles.push(paginationBubble);
}

return {
  type: 'flex',
  altText: '師資列表',
  contents: { type: 'carousel', contents: teacherBubbles }
};
});
}

async function buildTeacherSelectionCarousel() {
  return executeDbQuery(async (client) => {
    const res = await client.query("SELECT id, name, image_url FROM teachers ORDER BY

```

```

name ASC");
    if (res.rows.length === 0) {
        return { type: 'text', text: '錯誤:系統中沒有任何師資檔案, 請先至「個人資訊」建立至少一位老師的檔案.' };
    }

    const placeholder_avatar = 'https://i.imgur.com/8l1Yd2S.png';
    const teacherBubbles = res.rows.map(t => ({
        type: 'bubble',
        hero: {
            type: 'image',
            url: t.image_url || placeholder_avatar,
            size: 'full',
            aspectRatio: '1:1',
            aspectMode: 'cover',
        },
        body: {
            type: 'box',
            layout: 'vertical',
            contents: [
                { type: 'text', text: t.name, weight: 'bold', size: 'lg', align: 'center' }
            ]
        },
        footer: {
            type: 'box',
            layout: 'vertical',
            contents: [{
                type: 'button',
                style: 'primary',
                height: 'sm',
                action: {
                    type: 'postback',
                    label: '選擇此老師',
                    data: `action=select_teacher_for_course&teacher_id=${t.id}`
                }
            }]
        }
    }));
    return {
        type: 'flex',
        altText: '請選擇授課老師',
        contents: { type: 'carousel', contents: teacherBubbles }
    };

```

```
});  
}
```

```
async function showManualAdjustHistory(page, userId = null) {  
  const offset = (page - 1) * CONSTANTS.PAGINATION_SIZE;  
  return executeDbQuery(async (client) => {  
    let query = "SELECT * FROM orders WHERE amount = 0";  
    const queryParams = [];  
    let paramIndex = 1;  
  
    if (userId) {  
      query += ` AND user_id = ${paramIndex++}`;  
      queryParams.push(userId);  
    }  
  
    query += ` ORDER BY timestamp DESC LIMIT ${paramIndex++} OFFSET  
    ${paramIndex++}`;  
    queryParams.push(CONSTANTS.PAGINATION_SIZE + 1, offset);  
  
    const res = await client.query(query, queryParams);  
  
    const hasNextPage = res.rows.length > CONSTANTS.PAGINATION_SIZE;  
    const pageRows = hasNextPage ? res.rows.slice(0, CONSTANTS.PAGINATION_SIZE) :  
    res.rows;  
  
    if (pageRows.length === 0 && page === 1) {  
      return userId ? '這位學員沒有任何手動調整紀錄。' : '目前沒有任何手動調整紀錄。';  
    }  
    if (pageRows.length === 0) {  
      return '沒有更多紀錄了。';  
    }  
  
    const listItems = pageRows.map(record => {  
      const isAddition = record.points > 0;  
      const typeText = isAddition ? ` ✨ 手動加點` : ` ⚠️ 手動扣點`;  
      const pointsText = isAddition ? `+${record.points}` : `-${record.points}`;  
      const pointsColor = isAddition ? '#1A759F' : '#D9534F';  
  
      return {  
        type: 'box',  
        layout: 'horizontal',  
        paddingAll: 'md',  
        contents: [  

```

```

    {
      type: 'box',
      layout: 'vertical',
      flex: 3,
      contents: [
        { type: 'text', text: record.user_name, weight: 'bold', size: 'sm' },
        { type: 'text', text: formatDateTime(record.timestamp), size: 'xxs', color:
'#AAAAAA' }
      ]
    },
    {
      type: 'text',
      text: `${pointsText} 點`,
      gravity: 'center',
      align: 'end',
      flex: 2,
      weight: 'bold',
      size: 'sm',
      color: pointsColor,
    }
  ]
};
});

```

```

const customParams = userId ? `&user_id=${userId}` : "";
const paginationBubble = createPaginationBubble('action=view_manual_adjust_history',
page, hasNextPage, customParams);
const footerContents = paginationBubble ? paginationBubble.body.contents : [];

const headerText = userId ? `${pageRows[0].user_name} 的調整紀錄` : '手動調整紀錄';
return {
  type: 'flex',
  altText: '手動調整紀錄',
  contents: {
    type: 'bubble',
    size: 'giga',
    header: {
      type: 'box',
      layout: 'vertical',
      contents: [{ type: 'text', text: headerText, weight: 'bold', size: 'lg', color: 'FFFFFF' }],
      backgroundColor: '#343A40'
    },
    body: {

```



```

        type: 'box',
        layout: 'vertical',
        paddingAll: 'none',
        contents: listItems.flatMap((item, index) =>
            index === 0 ? [item] : [{ type: 'separator' }, item]
        )
    },
    footer: {
        type: 'box',
        layout: 'vertical',
        contents: footerContents
    }
}
};
});
}

```

// [新增] 老師用來查看購點紀錄的函式

```

async function showPurchaseHistoryAsTeacher(page, userId = null) {
    const offset = (page - 1) * CONSTANTS.PAGINATION_SIZE;
    return executeDbQuery(async (client) => {
        let query = `SELECT * FROM orders WHERE amount > 0 AND status = 'completed'`;
        const queryParams = [];
        let paramIndex = 1;

        if (userId) {
            query += ` AND user_id = ${paramIndex++}`;
            queryParams.push(userId);
        }

        query += ` ORDER BY timestamp DESC LIMIT ${paramIndex++} OFFSET
        ${paramIndex++}`;
        queryParams.push(CONSTANTS.PAGINATION_SIZE + 1, offset);

        const res = await client.query(query, queryParams);

        const hasNextPage = res.rows.length > CONSTANTS.PAGINATION_SIZE;
        const pageRows = hasNextPage ? res.rows.slice(0, CONSTANTS.PAGINATION_SIZE) :
        res.rows;

        if (pageRows.length === 0 && page === 1) {
            return userId ? '這位學員沒有任何購點紀錄。' : '目前沒有任何學員的購點紀錄。';
        }
    });
}

```

```

if (pageRows.length === 0) {
  return '沒有更多紀錄了。';
}

const listItems = pageRows.map(order => ({
  type: 'box',
  layout: 'horizontal',
  paddingAll: 'md',
  contents: [
    {
      type: 'box',
      layout: 'vertical',
      flex: 3,
      contents: [
        { type: 'text', text: order.user_name, weight: 'bold', size: 'sm' },
        { type: 'text', text: `購點: ${order.points} 點`, size: 'sm' },
        { type: 'text', text: formatDateTime(order.timestamp), size: 'xxs', color: '#AAAAAA' }
      ]
    },
    {
      type: 'text',
      text: `$$${order.amount}`,
      gravity: 'center',
      align: 'end',
      flex: 2,
      weight: 'bold',
      size: 'md',
      color: '#28A745',
    }
  ]
}));

const customParams = userId ? `&user_id=${userId}` : '';
const paginationBubble =
createPaginationBubble('action=view_purchase_history_as_teacher', page, hasNextPage,
customParams);
const footerContents = paginationBubble ? paginationBubble.body.contents : [];

const headerText = userId ? `${pageRows[0].user_name} 的購點紀錄` : '所有學員購點紀錄';

return {
  type: 'flex',

```

```

    altText: headerText,
    contents: {
      type: 'bubble',
      size: 'giga',
      header: {
        type: 'box',
        layout: 'vertical',
        contents: [{ type: 'text', text: headerText, weight: 'bold', size: 'lg', color: '#FFFFFF' }],
        backgroundColor: '#343A40'
      },
      body: {
        type: 'box',
        layout: 'vertical',
        paddingAll: 'none',
        contents: listItems.flatMap((item, index) =>
          index === 0 ? [item] : [{ type: 'separator' }, item]
        )
      },
      footer: {
        type: 'box',
        layout: 'vertical',
        contents: footerContents
      }
    }
  };
});
}

```

// [新增] 老師用來查看兌換紀錄的函式

```

async function showExchangeHistoryAsTeacher(page, userId = null) {
  const offset = (page - 1) * CONSTANTS.PAGINATION_SIZE;
  return executeDbQuery(async (client) => {
    let query = `SELECT * FROM product_orders`;
    const queryParams = [];
    let paramIndex = 1;

    if (userId) {
      query += ` WHERE user_id = ${paramIndex++}`;
      queryParams.push(userId);
    }

    query += ` ORDER BY created_at DESC LIMIT ${paramIndex++} OFFSET
    ${paramIndex++}`;
  });
}

```

```

queryParams.push(CONSTANTS.PAGINATION_SIZE + 1, offset);

const res = await client.query(query, queryParams);

const hasNextPage = res.rows.length > CONSTANTS.PAGINATION_SIZE;
const pageRows = hasNextPage ? res.rows.slice(0, CONSTANTS.PAGINATION_SIZE) :
res.rows;

if (pageRows.length === 0 && page === 1) {
  return userId ? '這位學員沒有任何兌換紀錄。' : '目前沒有任何學員的兌換紀錄。';
}
if (pageRows.length === 0) {
  return '沒有更多紀錄了。';
}

const statusMap = {
  'completed': { text: '✅ 已完成', color: '#52b69a' },
  'pending': { text: '🕒 處理中', color: '#ff9e00' },
  'cancelled': { text: '❌ 已取消', color: '#d90429' }
};

const listItems = pageRows.map(order => {
  const statusInfo = statusMap[order.status] || { text: order.status, color: '#6c757d' };
  const titleText = userId ? order.product_name : `${order.user_name} 兌換了
${order.product_name}`;

  return {
    type: 'box',
    layout: 'horizontal',
    paddingAll: 'md',
    contents: [
      {
        type: 'box',
        layout: 'vertical',
        flex: 3,
        contents: [
          { type: 'text', text: titleText, weight: 'bold', size: 'sm', wrap: true },
          { type: 'text', text: statusInfo.text, size: 'xs', color: statusInfo.color, weight: 'bold'
},
          { type: 'text', text: formatDateTime(order.created_at), size: 'xxs', color:
'#AAAAAA' }
        ]
      },
    ]
  },

```

```

    {
      type: 'text',
      text: `-${order.points_spent} 點`,
      gravity: 'center',
      align: 'end',
      flex: 2,
      weight: 'bold',
      size: 'sm',
      color: '#D9534F',
    }
  ]
};
});

```

```

const customParams = userId ? `&user_id=${userId}` : "";
const paginationBubble =
createPaginationBubble('action=view_exchange_history_as_teacher', page, hasNextPage,
customParams);
const footerContents = paginationBubble ? paginationBubble.body.contents : [];

const headerText = userId ? `${pageRows[0].user_name} 的兌換紀錄` : '所有學員兌換紀錄';

return {
  type: 'flex',
  altText: headerText,
  contents: {
    type: 'bubble',
    size: 'giga',
    header: {
      type: 'box',
      layout: 'vertical',
      contents: [{ type: 'text', text: headerText, weight: 'bold', size: 'lg', color: '#FFFFFF' }],
      backgroundColor: '#343A40'
    },
  },
  body: {
    type: 'box',
    layout: 'vertical',
    paddingAll: 'none',
    contents: listItems.flatMap((item, index) =>
      index === 0 ? [item] : [{ type: 'separator' }, item]
    )
  },
  footer: {

```

```

        type: 'box',
        layout: 'vertical',
        contents: footerContents
      }
    }
  };
});
}

```

```

async function showUnreadMessages(page) {
  const offset = (page - 1) * CONSTANTS.PAGINATION_SIZE;
  return executeDbQuery(async (client) => {
    const res = await client.query("SELECT * FROM feedback_messages WHERE status =
'new' ORDER BY timestamp ASC LIMIT $1 OFFSET $2", [CONSTANTS.PAGINATION_SIZE +
1, offset]);

```

```

    const hasNextPage = res.rows.length > CONSTANTS.PAGINATION_SIZE;
    const pageMessages = hasNextPage ? res.rows.slice(0,
CONSTANTS.PAGINATION_SIZE) : res.rows;

```

```

    if (pageMessages.length === 0 && page === 1) {
      return '太棒了！目前沒有未回覆的學員留言。';
    }
    if (pageMessages.length === 0) {
      return '沒有更多未回覆的留言了。';
    }

```

```

    const messageBubbles = pageMessages.map(msg => ({
      type: 'bubble',
      body: {
        type: 'box',
        layout: 'vertical',
        spacing: 'md',
        contents: [
          { type: 'text', text: msg.user_name, weight: 'bold', size: 'lg', wrap: true },
          { type: 'text', text: formatDateTime(msg.timestamp), size: 'xs', color: '#AAAAAA' },
          { type: 'separator', margin: 'lg' },
          { type: 'text', text: msg.message, wrap: true, margin: 'lg', size: 'md' }
        ]
      },
      footer: {
        type: 'box',

```

```

        layout: 'vertical',
        spacing: 'sm',
        contents: [
          { type: 'button', style: 'primary', height: 'sm', action: { type: 'postback', label: '💬 回覆  
此留言', data: `action=reply_feedback&msgId=${msg.id}&userId=${msg.user_id}` } },
          { type: 'button', style: 'secondary', height: 'sm', action: { type: 'postback', label: '標示  
為已讀', data: `action=mark_feedback_read&msgId=${msg.id}` } }
        ]
      }
    }));
    const paginationBubble = createPaginationBubble('action=view_unread_messages', page,
hasNextPage);
    if (paginationBubble) {
      messageBubbles.push(paginationBubble);
    }

    return { type: 'flex', altText: '未回覆的學員留言', contents: { type: 'carousel', contents:
messageBubbles } };
  });
}

```

// [新增] 老師用來查看歷史留言的函式

```

async function showHistoricalMessagesAsTeacher(page, userId = null) {
  const offset = (page - 1) * CONSTANTS.PAGINATION_SIZE;
  return executeDbQuery(async (client) => {
    let query = `SELECT * FROM feedback_messages`;
    const queryParams = [];
    let paramIndex = 1;

    if (userId) {
      query += ` WHERE user_id = ${paramIndex++}`;
      queryParams.push(userId);
    }

    query += ` ORDER BY timestamp DESC LIMIT ${paramIndex++} OFFSET
${paramIndex++}`;
    queryParams.push(CONSTANTS.PAGINATION_SIZE + 1, offset);

    const res = await client.query(query, queryParams);

    const hasNextPage = res.rows.length > CONSTANTS.PAGINATION_SIZE;
    const pageMessages = hasNextPage ? res.rows.slice(0,
CONSTANTS.PAGINATION_SIZE) : res.rows;

```

```

if (pageMessages.length === 0 && page === 1) {
  return userId ? '這位學員沒有任何留言紀錄。': '目前沒有任何學員的留言紀錄。';
}
if (pageMessages.length === 0) {
  return '沒有更多紀錄了。';
}

const statusMap = {
  new: { text: '🟡 新留言', color: '#ffb703' },
  read: { text: '🟡 已讀', color: '#adb5bd' },
  replied: { text: '🟢 已回覆', color: '#2a9d8f' },
};

const listItems = pageMessages.map(msg => {
  const statusInfo = statusMap[msg.status] || { text: msg.status, color: '#6c757d' };
  const replyContent = msg.teacher_reply
    ? [{ type: 'separator' }, { type: 'text', text: `回覆: ${msg.teacher_reply}`, wrap: true, size:
'xs', color: '#495057' }]
    : [];

  return {
    type: 'box',
    layout: 'vertical',
    paddingAll: 'md',
    spacing: 'sm',
    contents: [
      {
        type: 'box',
        layout: 'horizontal',
        contents: [
          { type: 'text', text: msg.user_name, weight: 'bold', size: 'sm', flex: 3 },
          { type: 'text', text: statusInfo.text, size: 'xs', color: statusInfo.color, align: 'end',
flex: 2 }
        ],
      },
      { type: 'text', text: `留言: ${msg.message}`, wrap: true, size: 'sm' },
      ...replyContent,
      { type: 'text', text: formatDateTime(msg.timestamp), size: 'xss', color: '#AAAAAA',
margin: 'md' }
    ],
  };
});

```



```

    const customParams = userId ? `&user_id=${userId}` : "";
    const paginationBubble =
createPaginationBubble('action=view_historical_messages_as_teacher', page, hasNextPage,
customParams);
    const footerContents = paginationBubble ? paginationBubble.body.contents : [];

    const headerText = userId ? `${pageMessages[0].user_name} 的歷史留言` : '所有學員歷史
留言';
    return {
      type: 'flex',
      altText: headerText,
      contents: {
        type: 'bubble',
        size: 'giga',
        header: { type: 'box', layout: 'vertical', contents: [{ type: 'text', text: headerText, weight:
'bold', size: 'lg', color: '#FFFFFF', wrap: true }], backgroundColor: '#343A40' },
        body: { type: 'box', layout: 'vertical', paddingAll: 'none', contents:
listItems.flatMap((item, index) => index === 0 ? [item] : [{ type: 'separator' }, item]) },
        footer: { type: 'box', layout: 'vertical', contents: footerContents }
      }
    };
  });
}

```

```

async function showPendingShopOrders(page) {
  const offset = (page - 1) * CONSTANTS.PAGINATION_SIZE;
  return executeDbQuery(async (client) => {
    // [V35.5 修改] 查詢所有未完成的訂單
    const res = await client.query(
      "SELECT * FROM product_orders WHERE status IN ('pending_payment',
'pending_confirmation') ORDER BY created_at ASC LIMIT $1 OFFSET $2",
      [CONSTANTS.PAGINATION_SIZE + 1, offset]
    );

    const hasNextPage = res.rows.length > CONSTANTS.PAGINATION_SIZE;
    const pageOrders = hasNextPage ? res.rows.slice(0, CONSTANTS.PAGINATION_SIZE) :
res.rows;

    if (pageOrders.length === 0 && page === 1) {
      return '目前沒有待處理的商品訂單。';
    }
    if (pageOrders.length === 0) {
      return '沒有更多待處理的訂單了。';
    }
  });
}

```

```
}
```

```
const listItems = pageOrders.map(order => {
  const bodyContents = [
    { type: 'text', text: order.product_name, weight: 'bold', size: 'md', wrap: true },
    { type: 'text', text: `購買者: ${order.user_name}`, size: 'sm' },
    { type: 'text', text: `金額: ${order.amount} 元`, size: 'sm', color: '#666666' },
    { type: 'text', text: formatDateTime(order.created_at), size: 'xss', color: '#AAAAAA' },
    { type: 'separator', margin: 'md' }
  ];

  let footerContents = [];

  if (order.status === 'pending_payment' && order.payment_method === 'cash') {
    bodyContents.push({ type: 'text', text: '付款方式: 現金面交', margin: 'md', size: 'sm',
weight: 'bold', color: '#1A759F' });
    footerContents.push({ type: 'button', style: 'primary', color: '#28a745', height: 'sm',
action: { type: 'postback', label: '✅ 確認收款', data:
`action=confirm_shop_order&orderUID=${order.order_uid}` } });
    footerContents.push({ type: 'button', style: 'secondary', height: 'sm', action: { type:
'postback', label: '取消訂單', data:
`action=cancel_shop_order_start&orderUID=${order.order_uid}` } });
  } else if (order.status === 'pending_confirmation' && order.payment_method ===
'transfer') {
    bodyContents.push({ type: 'text', text: '付款方式: 轉帳', margin: 'md', size: 'sm', color:
'#34A0A4' });
    bodyContents.push({ type: 'text', text: `後五碼: ${order.last_5_digits}`, size: 'lg',
weight: 'bold', margin: 'sm' });
    footerContents.push({ type: 'button', style: 'primary', color: '#28a745', height: 'sm',
action: { type: 'postback', label: '核准', data:
`action=confirm_shop_order&orderUID=${order.order_uid}` } });
    footerContents.push({ type: 'button', style: 'secondary', height: 'sm', action: { type:
'postback', label: '退回', data: `action=reject_shop_order&orderUID=${order.order_uid}` } });
  } else {
    bodyContents.push({ type: 'text', text: '狀態: 等待學員付款中...', margin: 'md', size:
'sm', color: '#6c757d' });
  }

  return {
    type: 'bubble',
    size: 'giga',
    body: {
      type: 'box',
```

```

        layout: 'vertical',
        paddingAll: 'lg',
        spacing: 'sm',
        contents: bodyContents
    },
    ...(footerContents.length > 0 && {
        footer: {
            type: 'box',
            layout: 'vertical',
            spacing: 'sm',
            contents: footerContents
        }
    })
};
});

const paginationBubble = createPaginationBubble('action=view_pending_shop_orders',
page, hasNextPage);
if (paginationBubble) {
    listItems.push(paginationBubble);
}

return {
    type: 'flex',
    altText: '待處理的商品訂單',
    contents: {
        type: 'carousel',
        contents: listItems
    }
};
});
}

async function showAnnouncementsForDeletion(page) {
    const offset = (page - 1) * CONSTANTS.PAGINATION_SIZE;
    return executeDbQuery(async (client) => {
        const res = await client.query(
            "SELECT * FROM announcements ORDER BY created_at DESC LIMIT $1 OFFSET
$2",
            [CONSTANTS.PAGINATION_SIZE + 1, offset]
        );

        const hasNextPage = res.rows.length > CONSTANTS.PAGINATION_SIZE;

```

```

const pageAnnouncements = hasNextPage ? res.rows.slice(0,
CONSTANTS.PAGINATION_SIZE) : res.rows;

if (pageAnnouncements.length === 0 && page === 1) {
  return '目前沒有任何可刪除的公告。';
}
if (pageAnnouncements.length === 0) {
  return '沒有更多公告了。';
}

const listItems = pageAnnouncements.map(ann => ({
  type: 'box',
  layout: 'horizontal',
  spacing: 'md',
  paddingAll: 'md',
  contents: [
    {
      type: 'box',
      layout: 'vertical',
      flex: 4,
      contents: [
        { type: 'text', text: ann.content, wrap: true, size: 'sm' },
        { type: 'text', text: `由 ${ann.creator_name} 於
${formatDateTime(ann.created_at)} 發布`, size: 'xss', color: '#AAAAAA', margin: 'lg', wrap: true }
      ]
    },
    {
      type: 'box',
      layout: 'vertical',
      flex: 1,
      justifyContent: 'center',
      contents: [
        { type: 'button', style: 'primary', color: '#DE5246', height: 'sm', action: { type:
'postback', label: '刪除', data: `action=select_announcement_for_deletion&ann_id=${ann.id}` } }
      ]
    }
  ]
}));

const paginationBubble =
createPaginationBubble('action=view_announcements_for_deletion', page, hasNextPage);
const footerContents = paginationBubble ? paginationBubble.body.contents : [];
return {
  type: 'flex',

```

```

      altText: '選擇要刪除的公告',
      contents: {
        type: 'bubble',
        size: 'giga',
        header: { type: 'box', layout: 'vertical', contents: [{ type: 'text', text: '刪除舊公告',
weight: 'bold', size: 'lg', color: '#FFFFFF', wrap: true }], backgroundColor: '#343A40' },
        body: { type: 'box', layout: 'vertical', paddingAll: 'none', contents:
listItems.flatMap((item, index) => index === 0 ? [item] : [{ type: 'separator' }, item]) },
        footer: { type: 'box', layout: 'vertical', contents: footerContents }
      }
    };
  });
}

async function showCourseSeries(page) {
  const offset = (page - 1) * CONSTANTS.PAGINATION_SIZE;
  return executeDbQuery(async (client) => {
    const res = await client.query(
      `SELECT DISTINCT ON (LEFT(id, 2)) id, title
      FROM courses
      WHERE time > NOW()
      ORDER BY LEFT(id, 2), time ASC
      LIMIT $1 OFFSET $2`,
      [CONSTANTS.PAGINATION_SIZE + 1, offset]
    );

    const hasNextPage = res.rows.length > CONSTANTS.PAGINATION_SIZE;
    const pageSeries = hasNextPage ? res.rows.slice(0, CONSTANTS.PAGINATION_SIZE) :
res.rows;

    if (pageSeries.length === 0 && page === 1) {
      return '目前沒有任何已開設且未來的課程系列可供管理。';
    }
    if (pageSeries.length === 0) {
      return '沒有更多課程系列了。';
    }

    const seriesBubbles = pageSeries.map(series => {
      const prefix = series.id.substring(0, 2);
      const mainTitle = getCourseMainTitle(series.title);

      return {
        type: 'bubble',
        header: { type: 'box', layout: 'vertical', contents: [ { type: 'text', text: mainTitle, weight:

```

```

'bold', size: 'lg', color: '#FFFFFF', wrap: true } ], backgroundColor: '#343A40', paddingAll: 'lg' },
  body: {
    type: 'box',
    layout: 'vertical',
    spacing: 'md',
    paddingAll: 'lg',
    contents: [
      { type: 'button', style: 'secondary', height: 'sm', action: { type: 'postback', label: '📅
單堂管理與取消', data: `action=manage_course_group&prefix=${prefix}&page=1` } },
      { type: 'button', style: 'secondary', color: '#DE5246', height: 'sm', action: { type:
'postback', label: '🗑️ 批次取消全系列', data:
`action=cancel_course_group_confirm&prefix=${prefix}` } }
    ]
  }
};
});

```

```

const paginationBubble = createPaginationBubble('action=view_course_series', page,
hasNextPage);
if (paginationBubble) {
  seriesBubbles.push(paginationBubble);
}

return {
  type: 'flex',
  altText: '管理已開課程',
  contents: {
    type: 'carousel',
    contents: seriesBubbles
  }
};
});
}

```

```

async function showPendingOrders(page) {
  const mapOrderToBubble = (order) => ({
    type: 'bubble',
    body: {
      type: 'box',
      layout: 'vertical',
      spacing: 'md',
      contents: [
        { type: 'text', text: order.user_name, weight: 'bold', size: 'xl' },

```

```

    { type: 'text', text: `${order.points} 點 / ${order.amount} 元`, size: 'md' },
    { type: 'separator', margin: 'lg' },
    {
      type: 'box',
      layout: 'vertical',
      margin: 'lg',
      spacing: 'sm',
      contents: [
        { type: 'box', layout: 'baseline', spacing: 'sm', contents: [ { type: 'text', text: '後五碼', color: '#aaaaaa', size: 'sm', flex: 2 }, { type: 'text', text: order.last_5_digits, color: '#666666', size: 'sm', flex: 5, wrap: true } ] },
        { type: 'box', layout: 'baseline', spacing: 'sm', contents: [ { type: 'text', text: '時間', color: '#aaaaaa', size: 'sm', flex: 2 }, { type: 'text', text: formatDateTime(order.timestamp), color: '#666666', size: 'sm', flex: 5, wrap: true } ] }
      ]
    }
  ],
  footer: {
    type: 'box',
    layout: 'horizontal',
    spacing: 'sm',
    contents: [
      { type: 'button', style: 'primary', color: '#dc3545', flex: 1, action: { type: 'postback', label: '退回', data: `action=reject_order&order_id=${order.order_id}` } },
      { type: 'button', style: 'primary', color: '#28a745', flex: 1, action: { type: 'postback', label: '核准', data: `action=confirm_order&order_id=${order.order_id}` } }
    ]
  }
});

return createPaginatedCarousel({
  altText: '待確認點數訂單',
  baseAction: 'action=view_pending_orders',
  page: page,
  dataQuery: "SELECT * FROM orders WHERE status = 'pending_confirmation' ORDER BY timestamp ASC LIMIT $1 OFFSET $2",
  queryParams: [],
  mapRowToBubble: mapOrderToBubble,
  noDataMessage: '目前沒有待您確認的點數訂單。'
});
}

async function showAvailableCourses(userId, page) {
  const offset = (page - 1) * CONSTANTS.PAGINATION_SIZE;

```

```

return executeDbQuery(async (client) => {
  const sevenDaysLater = new Date(Date.now() + 7 *
CONSTANTS.TIME.ONE_DAY_IN_MS);
  const coursesRes = await client.query(
    `SELECT
      c.*,
      t.name AS teacher_name,
      t.image_url AS teacher_image_url,
      t.bio AS teacher_bio,
      COALESCE(array_length(c.waiting, 1), 0) AS waiting_count
    FROM courses c
    LEFT JOIN teachers t ON c.teacher_id = t.id
    WHERE c.time > NOW() AND c.time < $1
    ORDER BY c.time ASC LIMIT $2 OFFSET $3`,
    [sevenDaysLater, CONSTANTS.PAGINATION_SIZE + 1, offset]
  );

  const hasNextPage = coursesRes.rows.length > CONSTANTS.PAGINATION_SIZE;
  const pageCourses = hasNextPage ? coursesRes.rows.slice(0,
CONSTANTS.PAGINATION_SIZE) : coursesRes.rows;

  if (pageCourses.length === 0 && page === 1) {
    return '抱歉, 未来 7 天內沒有可預約或候補的課程。';
  }
  if (pageCourses.length === 0) {
    return '沒有更多課程了。';
  }

  const placeholder_avatar = 'https://i.imgur.com/s43t5tQ.jpeg';
  const courseBubbles = pageCourses.map(c => {
    const studentCount = c.students?.length || 0;
    const spotsBookedByUser = (c.students || []).filter(id => id === userId).length;
    const isFull = studentCount >= c.capacity;

    const statusComponents = [];
    if (spotsBookedByUser > 0) {
      statusComponents.push({ type: 'text', text: '✅ 您已預約 ${spotsBookedByUser} 位`,
color: '#28a745', size: 'sm', weight: 'bold', margin: 'md' });
    }

    let courseStatusText;
    let footerButton;

```



```

    if (isFull) {
      courseStatusText = `候補中 (${c.waiting_count}人)`;
      footerButton = { type: 'button', style: 'secondary', height: 'sm', action: { type: 'postback',
label: '加入候補', data: `action=confirm_join_waiting_list_start&course_id=${c.id}` } } };
    } else {
      const remainingSpots = c.capacity - studentCount;
      courseStatusText = `剩餘 ${remainingSpots} 位`;
      footerButton = { type: 'button', style: 'primary', height: 'sm', action: { type: 'postback',
label: '預約此課程', data: `action=select_booking_spots&course_id=${c.id}` }, color: '#52B69A' };
    }

    return {
      type: 'bubble', size: 'giga',
      hero: { type: 'image', url: c.teacher_image_url || placeholder_avatar, size: 'full',
aspectRatio: '1:1', aspectMode: 'cover' },
      body: {
        type: 'box', layout: 'vertical', paddingAll: 'xl', spacing: 'md',
        contents: [
          { type: 'text', text: getCourseMainTitle(c.title), weight: 'bold', size: 'xl', wrap: true },
          ...statusComponents,
          { type: 'separator', margin: 'lg' },
          { type: 'text', text: `授課老師: ${c.teacher_name || '待定'}`, size: 'sm', margin: 'md'
},
          { type: 'text', text: c.teacher_bio || "", wrap: true, size: 'xs', color: '#888888',
margin: 'xs' },
          { type: 'text', text: formatDate(c.time), size: 'sm', margin: 'sm' },
          { type: 'text', text: `${c.points_cost} 點`, size: 'sm' },
          { type: 'text', text: courseStatusText, size: 'sm' },
        ]
      },
      footer: { type: 'box', layout: 'vertical', spacing: 'sm', contents: [footerButton] }
    };
  });

```

```

const paginationBubble = createPaginationBubble('action=view_available_courses', page,
hasNextPage);

```

```

if (paginationBubble) courseBubbles.push(paginationBubble);

```

```

const headerText = `📅 7日內可預約課程`;
const flexMessage = { type: 'flex', altText: headerText, contents: { type: 'carousel', contents:
courseBubbles } };

```

```

return page === 1 ? [{ type: 'text', text: `你好！${headerText}如下，請左右滑動查看：` },
flexMessage ] : flexMessage;

```

```

});
}
async function showMyCourses(userId, page) {
  const offset = (page - 1) * CONSTANTS.PAGINATION_SIZE;
  return executeDbQuery(async (client) => {
    const res = await client.query(
      `SELECT
        c.*,
        t.name AS teacher_name,
        t.image_url AS teacher_image_url
      FROM courses c
      LEFT JOIN teachers t ON c.teacher_id = t.id
      WHERE (
        c.students @> ARRAY[$1]::text[] OR c.waiting @> ARRAY[$1]::text[]
      ) AND c.time > NOW()
      ORDER BY c.time ASC`,
      [userId]
    );

    const allCourseCardsData = res.rows.flatMap(c => {
      const cards = [];
      const spotsBookedByUser = (c.students || []).filter(id => id === userId).length;
      const isUserOnWaitingList = (c.waiting || []).includes(userId);

      if (spotsBookedByUser > 0) cards.push({ course: c, type: 'booked', spots:
spotsBookedByUser });
      if (isUserOnWaitingList) cards.push({ course: c, type: 'waiting' });
      return cards;
    });

    if (allCourseCardsData.length === 0 && page === 1) {
      return '您目前沒有任何已預約或候補中的課程。';
    }

    const hasNextPage = allCourseCardsData.length > offset +
CONSTANTS.PAGINATION_SIZE;
    const pageCardsData = allCourseCardsData.slice(offset, offset +
CONSTANTS.PAGINATION_SIZE);

    if (pageCardsData.length === 0) {
      return '沒有更多課程了。';
    }
  });
}

```

```

const placeholder_avatar = 'https://i.imgur.com/s43t5tQ.jpeg';
const courseBubbles = pageCardsData.map(cardData => {
  const c = cardData.course;
  const statusComponents = [];
  const footerButtons = [];

  if (cardData.type === 'booked') {
    statusComponents.push({ type: 'text', text: `✅ 您已預約 ${cardData.spots} 位`, color:
'#28a745', size: 'sm', weight: 'bold', margin: 'md' });
    footerButtons.push({ type: 'button', style: 'primary', color: '#DE5246', height: 'sm',
action: { type: 'postback', label: `取消 ${cardData.spots > 1 ? '1位' : ''}預約`, data:
`action=confirm_cancel_booking_start&course_id=${c.id}` } });
  }
  if (cardData.type === 'waiting') {
    const waitingPosition = (c.waiting || []).indexOf(userId) + 1;
    statusComponents.push({ type: 'text', text: `🕒 您在候補名單中 (第${waitingPosition}
位)`, color: '#FFA500', size: 'sm', weight: 'bold', margin: 'sm' });
    footerButtons.push({ type: 'button', style: 'secondary', height: 'sm', action: { type:
'postback', label: '取消候補', data: `action=confirm_cancel_waiting_start&course_id=${c.id}` } });
  }

  return {
    type: 'bubble', size: 'giga',
    hero: { type: 'image', url: c.teacher_image_url || placeholder_avatar, size: 'full',
aspectRatio: '1:1', aspectMode: 'cover' },
    body: {
      type: 'box', layout: 'vertical', paddingAll: 'xl', spacing: 'md',
      contents: [
        { type: 'text', text: getCourseMainTitle(c.title), weight: 'bold', size: 'xl', wrap: true },
        ...statusComponents,
        { type: 'separator', margin: 'lg' },
        { type: 'text', text: `授課老師: ${c.teacher_name || '待定'}`, size: 'sm', margin: 'md'
},
        { type: 'text', text: formatDate(c.time), size: 'sm', margin: 'sm' }
      ]
    },
    footer: { type: 'box', layout: 'vertical', spacing: 'sm', contents: footerButtons }
  };
});

const paginationBubble = createPaginationBubble('action=view_my_courses', page,
hasNextPage);
if (paginationBubble) {

```

```

        courseBubbles.push(paginationBubble);
    }

    return { type: 'flex', altText: '我的課程列表', contents: { type: 'carousel', contents:
courseBubbles } };
    });
}

async function showMyMessages(userId, page) {
    const offset = (page - 1) * CONSTANTS.PAGINATION_SIZE;
    return executeDbQuery(async (client) => {
        const res = await client.query(
            `SELECT * FROM feedback_messages WHERE user_id = $1 ORDER BY timestamp
DESC LIMIT $2 OFFSET $3`,
            [userId, CONSTANTS.PAGINATION_SIZE + 1, offset]
        );

        const hasNextPage = res.rows.length > CONSTANTS.PAGINATION_SIZE;
        const pageMessages = hasNextPage ? res.rows.slice(0,
CONSTANTS.PAGINATION_SIZE) : res.rows;

        if (pageMessages.length > 0) {
            await client.query(
                "UPDATE feedback_messages SET is_student_read = true WHERE user_id = $1
AND status = 'replied' AND is_student_read = false",
                [userId]
            );
        }

        if (pageMessages.length === 0 && page === 1) {
            return '您目前沒有任何留言紀錄。';
        }
        if (pageMessages.length === 0) {
            return '沒有更多留言紀錄了。';
        }

        const statusMap = {
            new: { text: '🟡 等待回覆', color: '#ffb703' },
            read: { text: '🟡 老師已讀', color: '#adb5bd' },
            replied: { text: '🟢 老師已回覆', color: '#2a9d8f' },
        };
        const listItems = pageMessages.map(msg => {

```

```

const statusInfo = statusMap[msg.status] || { text: msg.status, color: '#6c757d' };
const replyContent = msg.teacher_reply
  ? [{ type: 'separator', margin: 'sm' }, { type: 'text', text: `老師回覆：
${msg.teacher_reply}`, wrap: true, size: 'xs', color: '#495057' }]
  : [];

return {
  type: 'box',
  layout: 'vertical',
  paddingAll: 'md',
  spacing: 'sm',
  contents: [
    {
      type: 'box',
      layout: 'horizontal',
      contents: [
        { type: 'text', text: '我的留言', weight: 'bold', size: 'sm', flex: 3 },
        { type: 'text', text: statusInfo.text, size: 'xs', color: statusInfo.color, align: 'end',
flex: 2 }
      ],
    },
    { type: 'text', text: msg.message, wrap: true, size: 'sm' },
    ...replyContent,
    { type: 'text', text: formatDateTime(msg.timestamp), size: 'xxs', color: '#AAAAAA',
margin: 'md' }
  ],
};
});

const paginationBubble = createPaginationBubble('action=view_my_messages', page,
hasNextPage);
const footerContents = paginationBubble ? paginationBubble.body.contents : [];
return {
  type: 'flex',
  altText: '您的歷史留言紀錄',
  contents: {
    type: 'bubble',
    size: 'giga',
    header: { type: 'box', layout: 'vertical', contents: [{ type: 'text', text: '我的留言紀錄',
weight: 'bold', size: 'lg', color: 'FFFFFF', wrap: true }], backgroundColor: '#343A40' },
    body: { type: 'box', layout: 'vertical', paddingAll: 'none', contents:
listItems.flatMap((item, index) => index === 0 ? [item] : [{ type: 'separator' }, item]) },
    footer: { type: 'box', layout: 'vertical', contents: footerContents }
  }
}

```

```

    };
  });
}

```

```

async function showSingleCoursesForCancellation(prefix, page) {
  const offset = (page - 1) * CONSTANTS.PAGINATION_SIZE;
  return executeDbQuery(async (client) => {
    const coursesRes = await client.query("SELECT * FROM courses WHERE id LIKE $1 AND
time > NOW() ORDER BY time ASC LIMIT $2 OFFSET $3", [`${prefix}%`,
CONSTANTS.PAGINATION_SIZE + 1, offset]);

    const hasNextPage = coursesRes.rows.length > CONSTANTS.PAGINATION_SIZE;
    const pageCourses = hasNextPage ? coursesRes.rows.slice(0,
CONSTANTS.PAGINATION_SIZE) : coursesRes.rows;

    if (pageCourses.length === 0 && page === 1) {
      return "此系列沒有可取消的未來課程。";
    }
    if (pageCourses.length === 0) {
      return '沒有更多課程了。';
    }

    const listItems = pageCourses.map(c => ({
      type: 'box',
      layout: 'horizontal',
      spacing: 'md',
      paddingAll: 'md',
      contents: [
        {
          type: 'box',
          layout: 'vertical',
          flex: 4,
          contents: [
            { type: 'text', text: c.title, wrap: true, weight: 'bold', size: 'sm' },
            { type: 'text', text: formatDate(c.time), size: 'sm', margin: 'md' }
          ]
        },
        {
          type: 'box',
          layout: 'vertical',
          flex: 2,
          justifyContent: 'center',

```

```

        contents: [
            { type: 'button', style: 'primary', color: '#DE5246', height: 'sm', action: { type:
'postback', label: '取消此堂', data: `action=confirm_single_course_cancel&course_id=${c.id}` } }
        ]
    }
}
]);
const paginationBubble = createPaginationBubble('action=manage_course_group', page,
hasNextPage, `&prefix=${prefix}`);
const footerContents = paginationBubble ? paginationBubble.body.contents : [];
return {
    type: 'flex',
    altText: '請選擇要單次取消的課程',
    contents: {
        type: 'bubble',
        size: 'giga',
        header: { type: 'box', layout: 'vertical', contents: [{ type: 'text', text: '單堂課程管理',
weight: 'bold', size: 'lg', color: '#FFFFFF', wrap: true }], backgroundColor: '#343A40' },
        body: { type: 'box', layout: 'vertical', paddingAll: 'none', contents:
listItems.flatMap((item, index) => index === 0 ? [item] : [{ type: 'separator' }, item]) },
        footer: { type: 'box', layout: 'vertical', contents: footerContents }
    }
};
});
}
async function showShopProducts(page) {
    const offset = (page - 1) * CONSTANTS.PAGINATION_SIZE;
    return executeDbQuery(async (client) => {
        const productsRes = await client.query("SELECT * FROM products WHERE status =
'available' ORDER BY created_at DESC LIMIT $1 OFFSET $2",
[CONSTANTS.PAGINATION_SIZE + 1, offset]);

        const hasNextPage = productsRes.rows.length > CONSTANTS.PAGINATION_SIZE;
        const pageProducts = hasNextPage ? productsRes.rows.slice(0,
CONSTANTS.PAGINATION_SIZE) : productsRes.rows;

        if (pageProducts.length === 0 && page === 1) {
            return '目前商城沒有任何商品，敬請期待！';
        }
        if (pageProducts.length === 0) {
            return '沒有更多商品了。';
        }
    });
}

```

```

const productBubbles = pageProducts.map(p => {
  const isSoldOut = p.inventory <= 0;
  const buttonStyle = isSoldOut ? 'secondary' : 'primary';
  const buttonLabel = isSoldOut ? '已售完' : '我要購買';
  const buttonAction = isSoldOut
    ? { type: 'message', label: buttonLabel, text: '此商品已售完' }
    : { type: 'postback', label: buttonLabel, data:
`action=select_product_quantity&product_id=${p.id}` };
  return {
    type: 'bubble',
    hero: (p.image_url && p.image_url.startsWith('https')) ?
      { type: 'image', url: p.image_url, size: 'full', aspectRatio: '1:1', aspectMode: 'cover' } :
undefined,
    body: {
      type: 'box',
      layout: 'vertical',
      contents: [
        { type: 'text', text: p.name, weight: 'bold', size: 'xl' },
        {
          type: 'box',
          layout: 'horizontal',
          margin: 'md',
          contents: [
            { type: 'text', text: `${p.price} 元整`, size: 'lg', color: '#1A759F', weight: 'bold',
flex: 2 },
            { type: 'text', text: `庫存: ${p.inventory}`, size: 'sm', color: '#666666', align:
'end', flex: 1, gravity: 'bottom' }
          ]
        },
        { type: 'text', text: p.description || ' ', wrap: true, size: 'sm', margin: 'md', color:
'#666666' },
      ]
    },
    footer: {
      type: 'box',
      layout: 'vertical',
      contents: [
        {
          type: 'button',
          style: buttonStyle,
          action: buttonAction,
          color: isSoldOut ? '#AAAAAA' : '#52B69A',
        }
      ]
    }
  }
});

```



```

    ]
  }
};
});

const paginationBubble = createPaginationBubble('action=view_shop_products', page,
hasNextPage);
if (paginationBubble) {
  productBubbles.push(paginationBubble);
}

return { type: 'flex', altText: '活動商城', contents: { type: 'carousel', contents: productBubbles
}};
});
}

async function showProductManagementList(page = 1, filter = null) {
  const offset = (page - 1) * CONSTANTS.PAGINATION_SIZE;
  return executeDbQuery(async (client) => {
    let baseQuery = "SELECT * FROM products";
    const queryParams = [];
    let paramIndex = 1;

    if (filter) {
      baseQuery += ` WHERE status = ${paramIndex++}`;
      queryParams.push(filter);
    }

    baseQuery += ` ORDER BY created_at DESC LIMIT ${paramIndex++} OFFSET
    ${paramIndex++}`;
    queryParams.push(CONSTANTS.PAGINATION_SIZE + 1, offset);

    const productsRes = await client.query(baseQuery, queryParams);

    const hasNextPage = productsRes.rows.length > CONSTANTS.PAGINATION_SIZE;
    const pageProducts = hasNextPage ? productsRes.rows.slice(0,
CONSTANTS.PAGINATION_SIZE) : productsRes.rows;

    if (pageProducts.length === 0 && page === 1) {
      const emptyMessage = filter === 'available'
        ? '目前沒有任何販售中的商品。'
        : (filter === 'unavailable' ? '目前沒有任何已下架的商品。' : '目前沒有任何商品可管理。
');
    }
  });
}

```

```

    return emptyMessage;
  }
  if (pageProducts.length === 0) {
    return '沒有更多商品了。';
  }

  const productBubbles = pageProducts.map(p => {
    const statusColor = p.status === 'available' ? '#52B69A' : '#6A7D8B';
    const toggleLabel = p.status === 'available' ? '下架商品' : '重新上架';
    const toggleAction = `action=toggle_product_status&product_id=${p.id}`;

    return {
      type: 'bubble',
      hero: (p.image_url && p.image_url.startsWith('https')) ? {
        type: 'image',
        url: p.image_url,
        size: 'full',
        aspectRatio: '1:1',
        aspectMode: 'cover',
      } : undefined,
      body: {
        type: 'box',
        layout: 'vertical',
        spacing: 'md',
        contents: [
          { type: 'text', text: p.name, weight: 'bold', size: 'xl', wrap: true },
          { type: 'text', text: p.description || '無描述', wrap: true, size: 'sm', color: '#666666',
margin: 'md' },
          { type: 'separator', margin: 'lg' },
          {
            type: 'box',
            layout: 'horizontal',
            margin: 'md',
            contents: [
              { type: 'text', text: `價格: ${p.price} 點`, size: 'md' },
              { type: 'text', text: `庫存: ${p.inventory}`, size: 'md', align: 'end' }
            ]
          }
        ]
      },
      footer: {
        type: 'box',
        layout: 'vertical',

```

```

        spacing: 'sm',
        contents: [
          { type: 'button', style: 'primary', height: 'sm', color: '#52B69A', action: { type:
'postback', label: '✏️ 編輯資訊', data: `action=manage_product&product_id=${p.id}` } },
          { type: 'button', style: 'secondary', height: 'sm', action: { type: 'postback', label: '📦
調整庫存', data: `action=adjust_inventory_start&product_id=${p.id}` } },
          { type: 'button', style: 'secondary', height: 'sm', color: '#D9534F', action: { type:
'postback', label: toggleLabel, data: toggleAction } }
        ]
      }
    };
  });

```

```

const paginationBubble = createPaginationBubble(
  'action=view_products',
  page,
  hasNextPage,
  filter ? `&filter=${filter}` : ""
);
if (paginationBubble) {
  productBubbles.push(paginationBubble);
}

```

```

    return { type: 'flex', altText: '商品管理列表', contents: { type: 'carousel', contents:
productBubbles } };
  });
}

```

// =====

// 程式碼修改: V35.5 (商品現金購 - Part 2)

// =====

// [V35.6 優化] 將購買紀錄改為條列式，並區分待處理與歷史訂單

async function showStudentExchangeHistory(userId, page = 1) { // page 參數暫時保留，但不再使用

```

  return executeDbQuery(async (client) => {
    // 抓取最近 20 筆訂單以避免訊息過長
    const res = await client.query(`SELECT * FROM product_orders WHERE user_id = $1
ORDER BY created_at DESC LIMIT 20`, [userId]);

```

```

    if (res.rows.length === 0) {
      return '您沒有任何商品購買紀錄。';
    }

```

// 步驟 1: 將訂單分組

```

const pendingOrders = [];
const historyOrders = [];

res.rows.forEach(order => {
  if (['pending_payment', 'pending_confirmation'].includes(order.status)) {
    pendingOrders.push(order);
  } else {
    historyOrders.push(order);
  }
});

const bodyContents = [];
const separator = { type: 'separator', margin: 'md' };

// 步驟 2: 產生「待處理訂單」列表
if (pendingOrders.length > 0) {
  bodyContents.push({ type: 'text', text: '待處理訂單', weight: 'bold', size: 'lg', margin: 'md',
    color: '#1A759F' });

  pendingOrders.forEach(order => {
    let statusText, statusColor, actionButton;
    if (order.status === 'pending_payment' && order.payment_method === 'transfer') {
      statusText = '🚫 待回報匯款';
      statusColor = '#f28482';
      actionButton = {
        type: 'button',
        style: 'primary',
        height: 'sm',
        color: '#f28482',
        action: { type: 'postback', label: '輸入匯款後五碼', data:
          `action=report_shop_last5&orderUID=${order.order_uid}`,
          margin: 'md'
        }
      };
    } else if (order.status === 'pending_payment' && order.payment_method === 'cash') {
      statusText = '💰 待現金付款';
      statusColor = '#1A759F';
    } else { // pending_confirmation
      statusText = '🕒 款項確認中';
      statusColor = '#ff9e00';
    }

    bodyContents.push({
      type: 'box',

```

```

        layout: 'vertical',
        margin: 'lg',
        spacing: 'sm',
        contents: [
          { type: 'text', text: order.product_name, weight: 'bold', wrap: true },
          { type: 'text', text: `金額: ${order.amount} 元`, size: 'sm' },
          {
            type: 'box',
            layout: 'horizontal',
            contents: [
              { type: 'text', text: statusText, size: 'sm', color: statusColor, weight: 'bold' },
              { type: 'text', text: formatDateTime(order.created_at), size: 'sm', color:
'#AAAAAA', align: 'end' }
            ]
          },
          ...(actionButton ? [actionButton] : []) // 如果有按鈕, 就加進來
        ]
      });
      bodyContents.push(separator);
    });
  }

```

```

// 步驟 3: 產生「歷史訂單」列表
if (historyOrders.length > 0) {
  bodyContents.push({ type: 'text', text: '歷史訂單', weight: 'bold', size: 'lg', margin: 'xl',
color: '#6c757d' });

```

```

  historyOrders.forEach(order => {
    let statusText, statusColor;
    if (order.status === 'completed') {
      statusText = '✅ 已完成';
      statusColor = '#28a745';
    } else { // cancelled
      statusText = '❌ 已取消';
      statusColor = '#dc3545';
    }
  }

```

```

  bodyContents.push({
    type: 'box',
    layout: 'vertical',
    margin: 'lg',
    spacing: 'sm',
    contents: [

```

```

        { type: 'text', text: order.product_name, weight: 'bold', wrap: true, color: '#888888'
    },
    { type: 'text', text: `金額:${order.amount} 元`, size: 'sm', color: '#888888' },
    {
        type: 'box',
        layout: 'horizontal',
        contents: [
            { type: 'text', text: statusText, size: 'sm', color: statusColor },
            { type: 'text', text: formatDateTime(order.created_at), size: 'sm', color:
'#AAAAAA', align: 'end' }
        ]
    }
    ]
    });
    bodyContents.push(separator);
    });
}

// 移除最後一個多餘的分隔線
if (bodyContents.length > 0 && bodyContents[bodyContents.length - 1].type ===
'separator') {
    bodyContents.pop();
}

return {
    type: 'flex',
    altText: '我的購買紀錄',
    contents: {
        type: 'bubble',
        size: 'giga',
        header: {
            type: 'box',
            layout: 'vertical',
            contents: [{ type: 'text', text: '📖 我的購買紀錄', weight: 'bold', size: 'xl', color:
'#FFFFFF' }],
            backgroundColor: '#343A40',
            paddingAll: 'lg'
        },
        body: {
            type: 'box',
            layout: 'vertical',
            spacing: 'md',
            paddingAll: 'lg',

```

```

      contents: bodyContents.length > 0 ? bodyContents : [{type: 'text', text: '目前沒有任何紀錄。', align: 'center'}]
    }
  }
};
});
}

```

```

async function showCourseRosterSummary(page) {
  const offset = (page - 1) * CONSTANTS.PAGINATION_SIZE;
  return executeDbQuery(async (client) => {
    const sevenDaysLater = new Date(Date.now() + 7 *
CONSTANTS.TIME.ONE_DAY_IN_MS);
    const res = await client.query(
      `SELECT id, title, time,
        COALESCE(array_length(students, 1), 0) as student_count,
        COALESCE(array_length(waiting, 1), 0) as waiting_count
      FROM courses
      WHERE time > NOW() AND time < $1
      ORDER BY time ASC LIMIT $2 OFFSET $3`,
      [sevenDaysLater, CONSTANTS.PAGINATION_SIZE + 1, offset]
    );

    const hasNextPage = res.rows.length > CONSTANTS.PAGINATION_SIZE;
    const pageCourses = hasNextPage ? res.rows.slice(0, CONSTANTS.PAGINATION_SIZE)
: res.rows;

```

```

    if (pageCourses.length === 0 && page === 1) {
      return '未來 7 天內沒有任何課程。';
    }
    if (pageCourses.length === 0) {
      return '沒有更多課程了。';
    }

```

```

    const listItems = pageCourses.map(c => ({
      type: 'box',
      layout: 'horizontal',
      spacing: 'md',
      paddingAll: 'md',
      contents: [
        {
          type: 'box',
          layout: 'vertical',

```

```

        flex: 4,
        contents: [
          { type: 'text', text: c.title, weight: 'bold', size: 'sm', wrap: true },
          { type: 'text', text: formatDate(c.time), size: 'xs', color: '#666666' },
          { type: 'text', text: `預約: ${c.student_count} 人 / 候補: ${c.waiting_count} 人`,
size: 'xs', margin: 'sm' }
        ]
      },
      {
        type: 'box',
        layout: 'vertical',
        flex: 2,
        justifyContent: 'center',
        contents: [
          { type: 'button', style: 'primary', height: 'sm', action: { type: 'postback', label: '看名
單', data: `action=view_course_roster_details&course_id=${c.id}` } }
        ]
      }
    ]
  }
  }));

```

```

const paginationBubble = createPaginationBubble('action=view_course_roster_summary',
page, hasNextPage);
const footerContents = paginationBubble ? paginationBubble.body.contents : [];

```

```

return {
  type: 'flex',
  altText: '課程狀態查詢',
  contents: {
    type: 'bubble',
    size: 'giga',
    header: { type: 'box', layout: 'vertical', contents: [{ type: 'text', text: '7日內課程狀態查詢
', weight: 'bold', size: 'lg', color: '#FFFFFF' }], backgroundColor: '#343A40' },
    body: { type: 'box', layout: 'vertical', paddingAll: 'none', contents:
listItems.flatMap((item, index) => index === 0 ? [item] : [{ type: 'separator' }, item]) },
    footer: { type: 'box', layout: 'vertical', contents: footerContents }
  }
};
});
}

```

```

async function showCourseRosterDetails(courseId) {

```



```

return executeDbQuery(async (client) => {
  const courseRes = await client.query("SELECT title, time, students, waiting FROM courses
WHERE id = $1", [courseId]);
  if (courseRes.rows.length === 0) {
    return '找不到該課程的資料。';
  }
  const course = courseRes.rows[0];
  const studentIds = course.students || [];
  const waitingIds = course.waiting || [];
  const allUserIds = [...studentIds, ...waitingIds];

  let users = [];
  if (allUserIds.length > 0) {
    const usersRes = await client.query("SELECT id, name, picture_url FROM users
WHERE id = ANY($1::text[])", [allUserIds]);
    users = usersRes.rows;
  }

  const userMap = new Map(users.map(u => [u.id, u]));
  const placeholderAvatar = 'https://i.imgur.com/8l1Yd2S.png';

  const createStudentListComponent = (ids, title) => {
    const studentCounts = ids.reduce((acc, id) => {
      acc[id] = (acc[id] || 0) + 1;
      return acc;
    }, {});

    const uniqueIds = Object.keys(studentCounts);

    const studentBoxes = [];
    if (uniqueIds.length > 0) {
      uniqueIds.forEach(id => {
        const user = userMap.get(id);
        const count = studentCounts[id];
        const displayName = user?.name || '未知用戶';
        const displayText = count > 1 ? `${displayName} (x${count})` : displayName;

        studentBoxes.push({
          type: 'box',
          layout: 'vertical',
          alignItems: 'center',
          spacing: 'sm',
          contents: [

```

```

        {
          type: 'image',
          url: user?.picture_url || placeholderAvatar,
          aspectRatio: '1:1',
          size: 'md',
          flex: 0
        },
        {
          type: 'text',
          text: displayText,
          wrap: true,
          size: 'sm',
          align: 'center'
        }
      ]
    });
  });
}

const listContents = [
  { type: 'text', text: title, weight: 'bold', color: '#1A759F', margin: 'lg', size: 'md', align:
'center' },
];
if (studentBoxes.length === 0) {
  listContents.push({ type: 'text', text: '無', margin: 'md', size: 'sm', color: '#999999',
align: 'center' });
} else {
  const rows = [];
  for (let i = 0; i < studentBoxes.length; i += 4) {
    rows.push({
      type: 'box',
      layout: 'horizontal',
      spacing: 'md',
      margin: 'lg',
      contents: studentBoxes.slice(i, i + 4)
    });
  }
  listContents.push(...rows);
}

return listContents;
};
const bodyContents = [

```

```

    ...createStudentListComponent(studentIds, `✅ 已預約學員 (${studentIds.length})`,
    { type: 'separator', margin: 'xl' },
    ...createStudentListComponent(waitingIds, `🕒 候補中學員 (${waitingIds.length})`)
  ];
  return {
    type: 'flex',
    altText: `課程 ${course.title} 的詳細名單`,
    contents: {
      type: 'bubble',
      size: 'giga',
      header: {
        type: 'box', layout: 'vertical', paddingAll: 'lg',
        contents: [
          { type: 'text', text: course.title, weight: 'bold', size: 'xl', wrap: true },
          { type: 'text', text: formatDate(course.time), size: 'sm', color: '#666666' },
        ]
      },
      margin: 'md' }
    ],
  },
  body: {
    type: 'box',
    layout: 'vertical',
    paddingAll: 'md',
    contents: bodyContents
  }
}
};
});
}

```

```

async function showStudentDetails(studentId) {
  return executeDbQuery(async (client) => {
    const userRes = await client.query('SELECT name, picture_url, points FROM users
WHERE id = $1', [studentId]);
    if (userRes.rows.length === 0) {
      return '找不到該學員的資料。';
    }
    const student = userRes.rows[0];

    const coursesRes = await client.query(
      `SELECT title, time FROM courses WHERE $1 = ANY(students) AND time > NOW()
ORDER BY time ASC LIMIT 3`,
      [studentId]
    );
  });
}

```

```

const ordersRes = await client.query(
  `SELECT points, status, timestamp FROM orders WHERE user_id = $1 ORDER BY
timestamp DESC LIMIT 3`,
  [studentId]
);

const createListItem = (text, size = 'sm', color = '#666666') => ({ type: 'text', text, size, color,
wrap: true, margin: 'sm' });

const coursesContents = [];
if (coursesRes.rows.length > 0) {
  coursesRes.rows.forEach(course => {
    coursesContents.push(createListItem(`- ${getCourseMainTitle(course.title)}
${formatDateTime(course.time)}`));
  });
} else {
  coursesContents.push(createListItem('無', 'sm', '#aaaaaa'));
}

const statusMap = { 'completed': '✅', 'pending_confirmation': '🕒', 'pending_payment': '!',
'rejected': '❌' };
const ordersContents = [];
if (ordersRes.rows.length > 0) {
  ordersRes.rows.forEach(order => {
    const statusIcon = statusMap[order.status] || '?';
    ordersContents.push(createListItem(`${statusIcon} ${order.points}點
${formatDateTime(order.timestamp)}`));
  });
} else {
  ordersContents.push(createListItem('無', 'sm', '#aaaaaa'));
}

return {
  type: 'flex',
  altText: `學員 ${student.name} 的詳細資料`,
  contents: {
    type: 'bubble',
    size: 'giga',
    header: {
      type: 'box',
      layout: 'vertical',
      paddingAll: 'lg',

```

```

        backgroundColor: '#343A40',
        contents: [
          { type: 'text', text: student.name, weight: 'bold', size: 'xl', color: '#FFFFFF', align:
'center' },
          {
            type: 'box', layout: 'baseline', margin: 'md', justifyContent: 'center',
            contents: [
              { type: 'text', text: '剩餘點數', size: 'sm', color: '#FFFFFF' },
              { type: 'text', text: `${student.points}`, weight: 'bold', size: 'xxl', color:
'#52B69A', margin: 'sm' },
              { type: 'text', text: '點', size: 'sm', color: '#FFFFFF' }
            ]
          }
        ]
      },
      body: {
        type: 'box',
        layout: 'vertical',
        paddingTop: 'lg',
        spacing: 'xl',
        contents: [
          {
            type: 'box', layout: 'vertical', margin: 'lg', spacing: 'sm',
            contents: [
              { type: 'text', text: '📅 17 近期預約課程', weight: 'bold', size: 'md', color:
'#333333' },
              ...coursesContents
            ]
          },
          { type: 'separator', margin: 'xl' },
          {
            type: 'box', layout: 'vertical', margin: 'lg', spacing: 'sm',
            contents: [
              { type: 'text', text: '💰 近期購點紀錄', weight: 'bold', size: 'md', color:
'#333333' },
              ...ordersContents
            ]
          }
        ]
      }
    }
  };
});

```

```

}
app.post('/webhook', line.middleware(config), (req, res) => {
  Promise.all(req.body.events.map(handleEvent))
    .then((result) => res.json(result))
    .catch((err) => {
      console.error(err);
      res.status(500).end();
    });
});
app.get('/', (req, res) => res.send('九容瑜伽 LINE Bot 正常運作中。'));

app.listen(PORT, async () => {
  try {
    checkEnvironmentVariables();
    console.log('✅ 資料庫結構已由 Build Command 處理。');

    console.log('✅ 伺服器已啟動, 監聽埠號 ${PORT}');
    console.log(`Bot 版本 V35.6 (商城現金流)`);

    setInterval(() => { if (SELF_URL.startsWith('https')) {axios.get(SELF_URL).catch(err =>
      console.error("Ping self failed:", err.message));}},
      CONSTANTS.INTERVALS.PING_INTERVAL_MS);
    setInterval(cancelExpiredPendingOrders, CONSTANTS.TIME.ONE_HOUR_IN_MS);
    const CLEANUP_INTERVAL_MS = CONSTANTS.TIME.ONE_HOUR_IN_MS * 6;
    setInterval(cleanCoursesDB, CLEANUP_INTERVAL_MS);
    console.log('🧹 已設定定期清理任務, 每 ${CLEANUP_INTERVAL_MS / 3600000} 小時執行
      一次。');
  } catch (error) {
    console.error('❌ 應用程式啟動失敗:', error);
    process.exit(1);
  }
});

async function handlePostback(event, user) {
  const data = new URLSearchParams(event.postback.data);
  const action = data.get('action');
  const userId = user.id;
  const page = parseInt(data.get('page') || '1', 10);
  switch (action) {

    // =====
    // 頁面檢視 (Pagination & Views)
    // =====

```

```

        case 'view_course_series': return showCourseSeries(page);
        case 'view_course_roster_summary': return showCourseRosterSummary(page);
        case 'view_course_roster_details': return showCourseRosterDetails(data.get('course_id'));
        case 'view_student_details': return showStudentDetails(data.get('studentId'));
        case 'list_teachers_for_removal': return showTeacherListForRemoval(page);
        case 'view_pending_orders':
        case 'view_pending_orders_page': return showPendingOrders(page);
        case 'student_search_results': return
showStudentSearchResults(decodeURIComponent(data.get('query') || ""), page);
        case 'view_unread_messages': return showUnreadMessages(page);
        case 'view_announcements_for_deletion': return showAnnouncementsForDeletion(page);
        case 'view_purchase_history': return showPurchaseHistory(userId, page);
        case 'view_available_courses': return showAvailableCourses(userId, page);
        case 'view_my_courses': return showMyCourses(userId, page);
        case 'view_shop_products': return showShopProducts(page);
        case 'view_my_messages': return showMyMessages(userId, page);
        case 'view_products': return showProductManagementList(page, data.get('filter'));
        case 'view_pending_shop_orders': return showPendingShopOrders(page);
        case 'view_exchange_history': return showStudentExchangeHistory(userId, page);
        case 'view_historical_messages': return
showHistoricalMessages(decodeURIComponent(data.get('query') || ""), page);
        case 'view_failed_tasks': return showFailedTasks(page);
        case 'manage_course_group': return showSingleCoursesForCancellation(data.get('prefix'),
page);
        // [修改] 老師手動調整紀錄的 Postback 處理
        case 'view_manual_adjust_history': return showManualAdjustHistory(page,
data.get('user_id'));

        // =====
        // [新增] 老師查詢購點/兌換紀錄、歷史留言
        // =====
        case 'view_all_purchase_history_as_teacher': return
showPurchaseHistoryAsTeacher(page);
        case 'view_purchase_history_as_teacher': return showPurchaseHistoryAsTeacher(page,
data.get('user_id'));
        case 'view_all_exchange_history_as_teacher': return
showExchangeHistoryAsTeacher(page);
        case 'view_exchange_history_as_teacher': return showExchangeHistoryAsTeacher(page,
data.get('user_id'));
        case 'view_all_historical_messages_as_teacher': return
showHistoricalMessagesAsTeacher(page);
        // =====
        // [新增] 老師查詢購點/兌換紀錄、歷史留言

```

```

// =====
// 點數管理
case 'select_purchase_history_view_type': {
    return {
        type: 'text',
        text: '請問您要查詢所有學員的購點紀錄, 還是特定學員?',
        quickReply: {
            items: [
                { type: 'action', action: { type: 'postback', label: '📄 顯示全部紀錄', data:
'action=view_all_purchase_history_as_teacher&page=1' } },
                { type: 'action', action: { type: 'postback', label: '🔍 搜尋特定學員', data:
'action=start_purchase_history_search' } }
            ]
        }
    };
}

// 商城管理
case 'select_exchange_history_view_type': {
    return {
        type: 'text',
        text: '請問您要查詢所有學員的兌換紀錄, 還是特定學員?',
        quickReply: {
            items: [
                { type: 'action', action: { type: 'postback', label: '📄 顯示全部紀錄', data:
'action=view_all_exchange_history_as_teacher&page=1' } },
                { type: 'action', action: { type: 'postback', label: '🔍 搜尋特定學員', data:
'action=start_exchange_history_search' } }
            ]
        }
    };
}

// 學員管理
case 'select_message_history_view_type': {
    return {
        type: 'text',
        text: '請問您要查詢所有學員的留言, 還是特定學員?',
        quickReply: {
            items: [
                { type: 'action', action: { type: 'postback', label: '📄 顯示全部留言', data:
'action=view_all_historical_messages_as_teacher&page=1' } },
                { type: 'action', action: { type: 'postback', label: '🔍 搜尋特定學員', data:
'action=start_message_history_search' } }
            ]
        }
    };
}

```



```

    }
  };
}

case 'view_historical_messages_as_teacher': return
showHistoricalMessagesAsTeacher(page, data.get('user_id'));

// =====
// [V34.0 & V34.1] 師資管理
// =====
case 'list_all_teachers': {
  return showAllTeachersList(page);
}
case 'manage_personal_profile': {
  return executeDbQuery(async (client) => {
    const res = await client.query('SELECT * FROM teachers WHERE line_user_id = $1',
[userId]);
    if (res.rows.length > 0) {
      const profile = res.rows[0];
      const placeholder_avatar = 'https://i.imgur.com/8l1Yd2S.png';
      return {
        type: 'flex', altText: '我的個人資訊',
        contents: {
          type: 'bubble',
          hero: { type: 'image', url: profile.image_url || placeholder_avatar, size: 'full',
aspectRatio: '1:1', aspectMode: 'cover' },
          body: { type: 'box', layout: 'vertical', paddingAll: 'lg', spacing: 'md', contents: [ {
type: 'text', text: profile.name, weight: 'bold', size: 'xl' }, { type: 'text', text: profile.bio || '尚未填寫
簡介', wrap: true, size: 'sm', color: '#666666' } ] },
          footer: { type: 'box', layout: 'vertical', spacing: 'sm', paddingAll: 'lg', contents: [ {
type: 'button', style: 'secondary', height: 'sm', action: { type: 'postback', label: '✎ 編輯姓名', data:
`action=edit_teacher_profile_field&field=name` } }, { type: 'button', style: 'secondary', height:
'sm', action: { type: 'postback', label: '✎ 編輯簡介', data:
`action=edit_teacher_profile_field&field=bio` } }, { type: 'button', style: 'secondary', height: 'sm',
action: { type: 'postback', label: '📷 更換照片', data:
`action=edit_teacher_profile_field&field=image_url` } }, ] }
        }
      };
    } else {
      return { type: 'text', text: '您好！您尚未建立您的公開師資檔案。\\n建立檔案後，您的
資訊將會顯示在「師資查詢」列表中。', quickReply: { items: [{ type: 'action', action: { type:
'postback', label: '✚ 開始建立檔案', data: 'action=create_teacher_profile_start' } } ] } };
    }
  });
}

```

```

    });
  }
  case 'create_teacher_profile_start': {
    pendingTeacherProfileEdit[userId] = { type: 'create', step: 'await_name', profileData: {} };
    setupConversationTimeout(userId, pendingTeacherProfileEdit,
'pendingTeacherProfile-Edit', (u) => {
      enqueuePushTask(u, { type: 'text', text: '建立檔案操作逾時，自動取消。' });
    });
    return { type: 'text', text: '好的，我們開始建立您的師資檔案。\\n\\n首先，請輸入您希望顯示的姓名或暱稱:', quickReply: { items: getCancelMenu() } };
  }
  case 'edit_teacher_profile_field': {
    const field = data.get('field');
    const fieldMap = { name: '姓名/暱稱', bio: '個人簡介', image_url: '新的照片' };
    const promptMap = { name: '請輸入您想更新的姓名或暱稱:', bio: '請輸入您想更新的個人簡介 (可換行):', image_url: '請直接上傳一張您想更換的個人照片:' };
    pendingTeacherProfileEdit[userId] = { type: 'edit', step: `await_${field}` };
    setupConversationTimeout(userId, pendingTeacherProfileEdit,
'pendingTeacherProfileEdit', (u) => {
      enqueuePushTask(u, { type: 'text', text: `編輯${fieldMap[field]}操作逾時，自動取消。`
});
    });
    return { type: 'text', text: promptMap[field], quickReply: { items: getCancelMenu() } };
  }
  case 'confirm_teacher_profile_update': {
    const state = pendingTeacherProfileEdit[userId];
    if (!state || state.step !== 'await_confirmation' || !state.newData) { return '確認操作已逾時或無效，請重新操作。'; }
    const newData = state.newData;
    const isCreating = state.type === 'create';
    delete pendingTeacherProfileEdit[userId];

    await executeDbQuery(async (client) => {
      if (isCreating) {
        await client.query( `INSERT INTO teachers (line_user_id, name, bio, image_url)
VALUES ($1, $2, $3, $4) ON CONFLICT (line_user_id) DO UPDATE SET name =
EXCLUDED.name, bio = EXCLUDED.bio, image_url = EXCLUDED.image_url, updated_at =
NOW()`, [userId, newData.name, newData.bio, newData.image_url] );
      } else {
        const fields = Object.keys(newData);
        const setClauses = fields.map((field, index) => `${field} = ${index + 1}`).join(', ');
        const values = Object.values(newData);
        await client.query( `UPDATE teachers SET ${setClauses}, updated_at = NOW()

```

```

WHERE line_user_id = $$${fields.length + 1}`, [...values, userId] );
    }
  });
  const successMessage = isCreating ? '✅ 恭喜！您的師資檔案已成功建立！': '✅ 您的
個人檔案已成功更新！';
  return successMessage;
}

case 'select_teacher_for_course': {
  const state = pendingCourseCreation[userId];
  const teacher_id = parseInt(data.get('teacher_id'), 10);
  if (!state || state.step !== 'await_teacher' || !teacher_id) { return '操作已逾時或無效，請重
新新增課程。'; }
  state.teacher_id = teacher_id;
  state.step = 'await_confirmation';
  const teacher = await executeDbQuery(client => client.query('SELECT name FROM
teachers WHERE id = $1', [teacher_id])).then(res => res.rows[0]);
  state.teacher_name = teacher?.name || '未知老師';

  const firstDate = getNextDate(state.weekday, state.time);
  const summary = `請確認課程資訊：\n\n` + `標題：${state.title}\n` + `老師：
${state.teacher_name}\n` + `時間：每${state.weekday_label} ${state.time}\n` + `堂數：
${state.sessions} 堂\n` + `名額：${state.capacity} 位\n` + `費用：${state.points_cost} 點/堂\n\n` +
`首堂開課日約為：${firstDate.toLocaleDateString('zh-TW', { timeZone: 'Asia/Taipei' })}`;
  return { type: 'text', text: summary, quickReply: { items: [ { type: 'action', action: { type:
'message', label: '✅ 確認新增', text: '✅ 確認新增' } }, { type: 'action', action: { type: 'message',
label: CONSTANTS.COMMANDS.GENERAL.CANCEL, text:
CONSTANTS.COMMANDS.GENERAL.CANCEL } } ] } };
}

case 'confirm_join_waiting_list_start': {
  const course_id = data.get('course_id');
  const course = await getCourse(course_id);
  if (!course) return '抱歉，找不到該課程，可能已被老師取消。';

  pendingBookingConfirmation[userId] = { type: 'confirm_wait', course_id: course_id };
  setupConversationTimeout(userId, pendingBookingConfirmation,
'pendingBookingConfirmation', (u) => {
    enqueuePushTask(u, { type: 'text', text: '加入候補操作已逾時，自動取消。' });
  });
  const message = `您確定要加入以下課程的候補名單嗎？\n\n課程：
${getCourseMainTitle(course.title)}\n時間：${formatDateTime(course.time)}\n\n候補不需支付點
數，當有名額釋出時，系統將會發送通知給您。`;

```

```

return {
  type: 'text',
  text: message,
  quickReply: {
    items: [
      { type: 'action', action: { type: 'postback', label: '✅ 確認加入候補', data:
`action=execute_join_waiting_list&course_id=${course.id}` } },
      { type: 'action', action: { type: 'message', label: '❌ 取消操作', text:
CONSTANTS.COMMANDS.GENERAL.CANCEL } }
    ]
  }
};
}

case 'execute_join_waiting_list': {
  const course_id = data.get('course_id');
  const result = await executeDbQuery(async (client) => {
    await client.query('BEGIN');
    try {
      const courseRes = await client.query('SELECT * FROM courses WHERE id = $1
FOR UPDATE', [course_id]);
      if (courseRes.rows.length === 0) { await client.query('ROLLBACK'); return '抱歉，找
不到該課程，可能已被老師取消。'; }
      const course = courseRes.rows[0];
      if ((course.students?.length || 0) < course.capacity) { await
client.query('ROLLBACK'); return '好消息！這堂課剛好有名額釋出了，請回到列表直接點擊「預
約課程」按鈕。'; }
      if (course.waiting?.includes(userId)) { await client.query('ROLLBACK'); return '您已
在候補名單中，請耐心等待通知。'; }

      const newWaitingList = [...(course.waiting || []), userId];
      await client.query('UPDATE courses SET waiting = $1 WHERE id = $2',
[newWaitingList, course_id]);
      await client.query('COMMIT');
      return `✅ 已成功將您加入「${getCourseMainTitle(course.title)}」的候補名單！\n當
有名額釋出時，系統將會發送通知給您。`;
    } catch (err) {
      await client.query('ROLLBACK');
      console.error('加入候補失敗 courseId: ${course_id}', err);
      return '加入候補時發生錯誤，請稍後再試。';
    }
  });
  delete pendingBookingConfirmation[userId];
  return result;
}

```

```

    }
    case 'join_waiting_list': {
      const course_id = data.get('course_id');
      return executeDbQuery(async (client) => {
        await client.query('BEGIN');
        try {
          const courseRes = await client.query('SELECT * FROM courses WHERE id = $1
FOR UPDATE', [course_id]);
          if (courseRes.rows.length === 0) { await client.query('ROLLBACK'); return '抱歉，找
不到該課程，可能已被老師取消。'; }
          const course = courseRes.rows[0];
          if ((course.students?.length || 0) < course.capacity) { await
client.query('ROLLBACK'); return '好消息！這堂課剛好有名額釋出了，請回到列表直接點擊「預
約課程」按鈕。'; }
          if (course.waiting?.includes(userId)) { await client.query('ROLLBACK'); return '您已
在候補名單中，請耐心等待通知。'; }

          const newWaitingList = [...(course.waiting || []), userId];
          await client.query('UPDATE courses SET waiting = $1 WHERE id = $2',
[newWaitingList, course_id]);
          await client.query('COMMIT');
          return `✅ 已成功將您加入「${getCourseMainTitle(course.title)}」的候補名單！\n當
有名額釋出時，系統將會發送通知給您。`;
        } catch (err) {
          await client.query('ROLLBACK');
          console.error('加入候補失敗 courseId: ${course_id}', err);
          return '加入候補時發生錯誤，請稍後再試。';
        }
      });
    }
    case 'select_purchase_plan': {
      const points = parseInt(data.get('plan'), 10);
      const plan = CONSTANTS.PURCHASE_PLANS.find(p => p.points === points);
      if (!plan) return '找不到您選擇的購買方案。';
      pendingPurchase[userId] = { step: 'confirm_purchase', data: { points: plan.points,
amount: plan.amount } };
      setupConversationTimeout(userId, pendingPurchase, 'pendingPurchase', (u) => {
        const timeoutMessage = { type: 'text', text: '您的購買確認操作已逾時，請重新點擊「購
買點數」開始。' };
        enqueuePushTask(u, timeoutMessage).catch(e => console.error(e));
      });
      return { type: 'text', text: `您選擇了購買「${plan.label}」。金額為 ${plan.amount} 元。\\n\\n
請確認是否繼續購買？`, quickReply: { items: [ { type: 'action', action: { type: 'message', label:

```

```

CONSTANTS.CONFIRM_BUY_POINTS, text:
CONSTANTS.COMMANDS.STUDENT.CONFIRM_BUY_POINTS } }, { type: 'action', action: {
type: 'message', label: CONSTANTS.COMMANDS.GENERAL.CANCEL, text:
CONSTANTS.COMMANDS.GENERAL.CANCEL } } ]}};
}
case 'confirm_order': {
  const order_id = data.get('order_id');
  return executeDbQuery(async (client) => {
    await client.query('BEGIN');
    try {
      const orderRes = await client.query("SELECT * FROM orders WHERE order_id =
$1 FOR UPDATE", [order_id]);
      if (orderRes.rows.length === 0) { await client.query('ROLLBACK'); return '找不到此
訂單，可能已被其他老師處理。'; }
      const order = orderRes.rows[0];
      if (order.status !== 'pending_confirmation') { await client.query('ROLLBACK'); return
`此訂單狀態為「${order.status}」，無法重複核准。`; }
      await client.query("UPDATE orders SET status = 'completed' WHERE order_id =
$1", [order_id]);
      const userRes = await client.query("SELECT points FROM users WHERE id = $1
FOR UPDATE", [order.user_id]);
      const newPoints = userRes.rows[0].points + order.points;
      await client.query("UPDATE users SET points = $1 WHERE id = $2", [newPoints,
order.user_id]);
      const notifyMessage = { type: 'text', text: `✅ 您的點數購買已核准！\n\n已為您帳戶
加入 ${order.points} 點，您目前的總點數為 ${newPoints} 點。`; };
      await enqueuePushTask(order.user_id, notifyMessage);
      await client.query('COMMIT');
      return `✅ 已核准 ${order.user_name} 的訂單，並已通知對方。`;
    } catch (err) { await client.query('ROLLBACK'); console.error('❌ 核准訂單時發生錯誤:
', err); return '處理訂單時發生錯誤，操作已取消。'; }
  });
}
case 'reject_order': {
  const order_id = data.get('order_id');
  return executeDbQuery(async (client) => {
    const orderRes = await client.query("SELECT * FROM orders WHERE order_id = $1",
[order_id]);
    if (orderRes.rows.length === 0) return '找不到此訂單，可能已被其他老師處理。';
    const order = orderRes.rows[0];
    if (order.status !== 'pending_confirmation') return `此訂單狀態為「${order.status}」，無
法退回。`;
  });
}
}
}

```

```

    await client.query("UPDATE orders SET status = 'rejected' WHERE order_id = $1",
[order_id]);

    const notifyMessage = { type: 'text', text: `❗ 您的點數購買申請被退回。\\n\\n請檢查您的匯款金額或後五碼是否有誤，並至「點數查詢」選單中重新提交資訊。如有疑問請聯絡我們，謝謝。` };

    await enqueuePushTask(order.user_id, notifyMessage).catch(e => console.error(e));
    return `✅ 已退回 ${order.user_name} 的訂單，並已通知對方。`;
  });
}

case 'generate_report': {
  const reportType = data.get('type');
  const period = data.get('period');
  const periodMap = { week: '本週', month: '本月', quarter: '本季', year: '今年' };
  const periodText = periodMap[period] || period;
  const generateReportTask = async () => {
    const { start, end } = getDateRange(period);
    return executeDbQuery(async (client) => {
      if (reportType === 'course') {
        const res = await client.query("SELECT capacity, students FROM courses
WHERE time BETWEEN $1 AND $2", [start, end]);
        if (res.rows.length === 0) return `📊 ${periodText}課程報表 📊\\n\\n此期間內沒有任何課程。`;

        let totalStudents = 0, totalCapacity = 0;
        res.rows.forEach(c => { totalCapacity += c.capacity; totalStudents += (c.students
|| []).length; });
        const attendanceRate = totalCapacity > 0 ? (totalStudents / totalCapacity *
100).toFixed(1) : 0;
        return `📊 ${periodText} 課程報表 📊\\n\\n- 課程總數: ${res.rows.length} 堂\\n- 總計名額: ${totalCapacity} 人\\n- 預約人次: ${totalStudents} 人\\n- **整體出席率: ${attendanceRate}%**.trim();
      } else if (reportType === 'order') {
        const res = await client.query("SELECT COUNT(*), SUM(amount) FROM orders
WHERE status = 'completed' AND timestamp BETWEEN $1 AND $2", [start, end]);
        const count = parseInt(res.rows[0].count, 10) || 0;
        const sum = parseInt(res.rows[0].sum, 10) || 0;
        return `💰 ${periodText} 訂單報表 💰\\n\\n- 已完成訂單: ${count} 筆\\n- **點數總收入: ${sum} 元**.trim();
      }
    });
  };

  const timeoutPromise = new Promise(resolve => setTimeout(() => resolve('timeout'),
8000));
  try {

```

```

const result = await Promise.race([generateReportTask(), timeoutPromise]);
if (result === 'timeout') {
  (async () => {
    try {
      const reportText = await generateReportTask();
      await enqueuePushTask(userId, { type: 'text', text: reportText });
    } catch (bgErr) {
      console.error('✖ 背景生成報表失敗:', bgErr);
      await enqueuePushTask(userId, { type: 'text', text: `抱歉, 產生 ${periodText} 報表時發生錯誤。` });
    }
  })();
  return '📊 報表生成中, 資料量較大, 請稍候... 完成後將會推播通知您。';
} else { return result; }
} catch (err) { console.error('✖ 即時生成 ${reportType} 報表失敗:', err);
return `✖ 產生 ${periodText} 報表時發生錯誤, 請稍後再試。`; }
}
case 'select_adjust_history_view_type': {
  return {
    type: 'text',
    text: '請問您要查詢所有學員的紀錄, 還是特定學員?',
    quickReply: {
      items: [
        { type: 'action', action: { type: 'postback', label: '📄 顯示全部紀錄', data: 'action=view_manual_adjust_history&page=1' } },
        { type: 'action', action: { type: 'postback', label: '🔍 搜尋特定學員', data: 'action=start_manual_adjust_history_search' } }
      ]
    }
  };
}
case 'start_manual_adjust_history_search': {
  pendingManualAdjustSearch[userId] = { step: 'await_student_name' };
  setupConversationTimeout(userId, pendingManualAdjustSearch,
'pendingManualAdjustSearch', (u) => {
    enqueuePushTask(u, { type: 'text', text: '搜尋操作已逾時, 自動取消。' });
  });
  return {
    type: 'text',
    text: '請輸入您想查詢的學員姓名:',
    quickReply: { items: getCancelMenu() }
  };
}
}

```



```

    case 'start_purchase_history_search': {
      pendingPurchaseHistorySearch[userId] = { step: 'await_student_name' };
      setupConversationTimeout(userId, pendingPurchaseHistorySearch,
'pendingPurchaseHistorySearch', u => enqueuePushTask(u, { type: 'text', text: '搜尋購點紀錄操作逾時, 自動取消.' }));
      return {
        type: 'text',
        text: '請輸入您想查詢購點紀錄的學員姓名或 User ID:',
        quickReply: { items: getCancelMenu() }
      };
    }
    case 'start_exchange_history_search': {
      pendingExchangeHistorySearch[userId] = { step: 'await_student_name' };
      setupConversationTimeout(userId, pendingExchangeHistorySearch,
'pendingExchangeHistorySearch', u => enqueuePushTask(u, { type: 'text', text: '搜尋兌換紀錄操作逾時, 自動取消.' }));
      return {
        type: 'text',
        text: '請輸入您想查詢兌換紀錄的學員姓名或 User ID:',
        quickReply: { items: getCancelMenu() }
      };
    }
    case 'start_message_history_search': {
      pendingMessageHistorySearch[userId] = { step: 'await_student_name' };
      setupConversationTimeout(userId, pendingMessageHistorySearch,
'pendingMessageHistorySearch', u => enqueuePushTask(u, { type: 'text', text: '搜尋歷史留言操作逾時, 自動取消.' }));
      return {
        type: 'text',
        text: '請輸入您想查詢歷史留言的學員姓名或 User ID:',
        quickReply: { items: getCancelMenu() }
      };
    }
  }

  case 'confirm_add_product': {
    const state = pendingProductCreation[userId];
    if (!state || state.step !== 'await_confirmation') return '上架流程已逾時或中斷, 請重新操作.';
    await executeDbQuery(client => client.query( `INSERT INTO products (name,
description, price, inventory, image_url, status, creator_id, creator_name) VALUES ($1, $2, $3,
$4, $5, 'available', $6, $7)` , [state.name, state.description, state.price, state.inventory,
state.image_url, userId, user.name] ) );
    delete pendingProductCreation[userId];
  }

```

```

    return '✅ 商品已成功上架！';
  }
  case 'manage_product': {
    const productId = data.get('product_id');
    const product = await getProduct(productId);
    if (!product) return '找不到該商品。';
    const flexMessage = { type: 'flex', altText: '編輯商品資訊', contents: { type: 'bubble',
header: { type: 'box', layout: 'vertical', contents: [{ type: 'text', text: `編輯:${product.name}`,
weight: 'bold', size: 'lg', color: '#FFFFFF', wrap: true }], backgroundColor: '#52B69A' }, body: {
type: 'box', layout: 'vertical', spacing: 'sm', contents: [ { type: 'button', style: 'link', height: 'sm',
action: { type: 'postback', label: '✏️ 編輯名稱', data:
`action=edit_product_field&product_id=${productId}&field=name` } }, { type: 'button', style: 'link',
height: 'sm', action: { type: 'postback', label: '✏️ 編輯描述', data:
`action=edit_product_field&product_id=${productId}&field=description` } }, { type: 'button', style:
'link', height: 'sm', action: { type: 'postback', label: '✏️ 編輯價格', data:
`action=edit_product_field&product_id=${productId}&field=price` } }, { type: 'button', style: 'link',
height: 'sm', action: { type: 'postback', label: '✏️ 編輯圖片網址', data:
`action=edit_product_field&product_id=${productId}&field=image_url` } } ] } } };
    return flexMessage;
  }
  case 'edit_product_field': {
    const productId = data.get('product_id');
    const field = data.get('field');
    const product = await getProduct(productId);
    if (!product) return '找不到該商品。';
    pendingProductEdit[userId] = { product, field };
    setupConversationTimeout(userId, pendingProductEdit, 'pendingProductEdit', u =>
enqueuePushTask(u, { type: 'text', text: '編輯商品操作逾時，自動取消。'}));
    const fieldMap = { name: '名稱', description: '描述', price: '價格 (點數)', image_url: '圖片
網址' };
    return { type: 'text', text: `請輸入新的「${fieldMap[field]}」:\n(目前為:${product[field]} || '無
'}`, quickReply: { items: getCancelMenu() } };
  }
  case 'adjust_inventory_start': {
    const productId = data.get('product_id');
    const product = await getProduct(productId);
    if (!product) return '找不到該商品。';
    pendingInventoryAdjust[userId] = { product, originalInventory: product.inventory };
    setupConversationTimeout(userId, pendingInventoryAdjust, 'pendingInventoryAdjust', u
=> enqueuePushTask(u, { type: 'text', text: '調整庫存操作逾時，自動取消。'}));
    return { type: 'text', text: `正在調整「${product.name}」的庫存 (目前為
${product.inventory}).\n請輸入要調整的數量 (正數為增加，負數為減少):`, quickReply: { items:
getCancelMenu() } };
  }

```

```

    }
    case 'toggle_product_status': {
      const productId = data.get('product_id');
      return executeDbQuery(async (client) => {
        await client.query('BEGIN');
        try {
          const productRes = await client.query('SELECT status, name FROM products
WHERE id = $1 FOR UPDATE', [productId]);
          if (productRes.rows.length === 0) { await client.query('ROLLBACK'); return '找不到
該商品。'; }
          const product = productRes.rows[0];
          const newStatus = product.status === 'available' ? 'unavailable' : 'available';
          await client.query('UPDATE products SET status = $1 WHERE id = $2',
[newStatus, productId]);
          await client.query('COMMIT');
          const statusText = newStatus === 'available' ? '上架' : '下架';
          return `✅ 已成功將商品「${product.name}」設定為「${statusText}」狀態。`;
        } catch(e) { await client.query('ROLLBACK'); console.error("切換商品狀態失敗:", e);
return '操作失敗，請稍後再試。'; }
      });
    }
  }
  // [V35.6 新增] 處理商品購買數量選擇
  case 'select_product_quantity': {
    const productId = data.get('product_id');
    const product = await getProduct(productId);
    if (!product || product.status !== 'available' || product.inventory <= 0) {
      return '抱歉，此商品目前無法購買。';
    }
  }

```

// 最多可以一次購買 5 個，或商品的剩餘庫存量，取較小者

```
const maxQuantity = Math.min(5, product.inventory);
```

// 動態產生數量按鈕

```

const quantityButtons = Array.from({ length: maxQuantity }, (_, i) => {
  const quantity = i + 1;
  const totalAmount = product.price * quantity;
  return {
    type: 'button',
    style: 'secondary',
    height: 'sm',
    margin: 'sm',
    action: {
      type: 'postback',

```

```

        label: `${quantity} 個 (共 ${totalAmount} 元)`,
        // 將選擇的數量 (qty) 傳遞到下一步
        data:
`action=confirm_product_purchase&product_id=${product.id}&qty=${quantity}`
    }
    };
});

return {
  type: 'flex',
  altText: '請選擇購買數量',
  contents: {
    type: 'bubble',
    header: {
      type: 'box',
      layout: 'vertical',
      contents: [{ type: 'text', text: '請選擇購買數量', weight: 'bold', size: 'lg', color:
'#FFFFFF' }],
      backgroundColor: '#52B69A'
    },
    body: {
      type: 'box',
      layout: 'vertical',
      contents: [
        { type: 'text', text: product.name, wrap: true, weight: 'bold', size: 'md' },
        { type: 'text', text: `單價: ${product.price} 元`, size: 'sm', color: '#666666',
margin: 'md' },
        { type: 'text', text: `剩餘庫存: ${product.inventory} 個`, size: 'sm', color:
'#666666' },
        { type: 'separator', margin: 'lg' }
      ]
    },
    footer: {
      type: 'box',
      layout: 'vertical',
      spacing: 'sm',
      contents: quantityButtons
    }
  }
};
}

case 'confirm_product_purchase': {

```

```

const productId = data.get('product_id');
// [V35.6 修改] 從 postback data 讀取數量
const quantity = parseInt(data.get('qty') || '1', 10);

const product = await getProduct(productId);
if (!product || product.status !== 'available') return '找不到此商品，或商品已下架。';
// 檢查庫存是否足夠
if (product.inventory < quantity) return `抱歉，此商品庫存不足！\n您想購買 ${quantity}
個，但僅剩 ${product.inventory} 個。`;

// 計算總金額
const totalAmount = product.price * quantity;

const flexMessage = {
  type: 'flex',
  altText: '請選擇付款方式',
  contents: {
    type: 'bubble',
    header: {
      type: 'box',
      layout: 'vertical',
      contents: [{ type: 'text', text: '請確認訂單並選擇付款方式', weight: 'bold', size: 'lg',
color: '#FFFFFF', wrap: true }],
      backgroundColor: '#52B69A'
    },
    body: {
      type: 'box',
      layout: 'vertical',
      spacing: 'md',
      contents: [
        { type: 'text', text: product.name, weight: 'bold', size: 'md', wrap: true },
        // 顯示單價、數量和總金額
        { type: 'text', text: `單價: ${product.price} 元`, size: 'sm' },
        { type: 'text', text: `數量: ${quantity} 個`, size: 'sm' },
        { type: 'separator', margin: 'sm' },
        { type: 'text', text: `總金額: ${totalAmount} 元`, size: 'lg', weight: 'bold', margin:
'sm' }
      ]
    },
    footer: {
      type: 'box',
      layout: 'vertical',
      spacing: 'sm',

```

```

contents: [
  {
    type: 'button',
    style: 'primary',
    color: '#34A0A4',
    action: {
      type: 'postback',
      label: '💰 轉帳付款',
      // 將數量 (qty) 繼續傳遞到最後一步
      data:
`action=execute_product_purchase&product_id=${product.id}&method=transfer&qty=${quantity}`
    },
  },
  {
    type: 'button',
    style: 'primary',
    color: '#1A759F',
    action: {
      type: 'postback',
      label: '💵 現金面交',
      data:
`action=execute_product_purchase&product_id=${product.id}&method=cash&qty=${quantity}`
    },
  },
  {
    type: 'button',
    style: 'secondary',
    height: 'sm',
    margin: 'md',
    action: {
      type: 'message',
      label: '取消',
      text: CONSTANTS.COMMANDS.GENERAL.CANCEL
    },
  },
]
};
return flexMessage;
}

```

```

case 'execute_product_purchase': {
  const productId = data.get('product_id');
  const paymentMethod = data.get('method');
  const quantity = parseInt(data.get('qty') || '1', 10);

  // 檢查是否已有待付款的相同商品訂單
  const existingOrderRes = await executeDbQuery(client =>
    client.query(
      "SELECT * FROM product_orders WHERE user_id = $1 AND product_id = $2 AND
status IN ('pending_payment', 'pending_confirmation'",
      [userId, productId]
    )
  );

  if (existingOrderRes.rows.length > 0) {
    return '您已經有此商品的待付款訂單，請至「我的購買紀錄」查看或完成付款。';
  }

  // 執行購買流程
  const result = await executeDbQuery(async (client) => {
    await client.query('BEGIN');
    try {
      const productRes = await client.query('SELECT * FROM products WHERE id = $1
FOR UPDATE', [productId]);
      const studentRes = await client.query('SELECT * FROM users WHERE id = $1
FOR UPDATE', [user.id]);

      const product = productRes.rows[0];
      const student = studentRes.rows[0];

      if (!product || product.status !== 'available') {
        await client.query('ROLLBACK');
        return { success: false, message: '購買失敗，找不到此商品或已下架。' };
      }
      if (product.inventory < quantity) {
        await client.query('ROLLBACK');
        return { success: false, message: `抱歉，您慢了一步！商品庫存僅剩
${product.inventory} 個。` };
      }

      const totalAmount = product.price * quantity;
      await client.query('UPDATE products SET inventory = inventory - $1 WHERE id =
$2', [quantity, productId]);
    } catch (error) {
      await client.query('ROLLBACK');
      return { success: false, message: '購買失敗，請稍後再試。' };
    }
  });
  return result;
}

```

```

const orderUID = `PROD-${Date.now()}-${userId.slice(-4)}`;

await client.query(
  `INSERT INTO product_orders (
    order_uid, user_id, user_name, product_id, product_name,
    points_spent, status, amount, payment_method
  ) VALUES ($1, $2, $3, $4, $5, $6, 'pending_payment', $7, $8)`,
  [
    orderUID, userId, student.name, productId, `${product.name} x${quantity}`,
    0, totalAmount, paymentMethod
  ]
);

const notifyMessage = { type: 'text', text: `🔔 商城新訂單通知\n學員
${student.name} 購買了「${product.name} x${quantity}」.\n總金額: ${totalAmount} 元\n付款方式
: ${paymentMethod === 'transfer' ? '轉帳' : '現金'}\n請至「訂單管理」查看.` };
await notifyAllTeachers(notifyMessage);

await client.query('COMMIT');

if (paymentMethod === 'transfer') {
  const replyText = `感謝您的購買！訂單已成立。\\n\\n請匯款至以下帳戶:\\n銀行:
${CONSTANTS.BANK_INFO.bankName}\\n戶名: ${CONSTANTS.BANK_INFO.accountName}\\n
帳號: ${CONSTANTS.BANK_INFO.accountNumber}\\n金額: ${totalAmount} 元\\n\\n匯款完成後,
請至「商城」->「我的購買紀錄」回報後五碼。`;
  return { success: true, message: replyText };
} else {
  const replyText = `✅ 訂單已成立！\\n您購買了「${product.name} x${quantity}」,
總金額 ${totalAmount} 元。\\n您選擇了現金付款, 請直接與老師聯繫並完成支付。`;
  return { success: true, message: replyText };
}

} catch (err) {
  await client.query('ROLLBACK');
  console.error('❌ 商品購買執行失敗:', err);
  return { success: false, message: '抱歉, 購買過程中發生錯誤, 您的訂單未成立, 請
稍後再試。' };
}
});

delete pendingBookingConfirmation[userId];
return result.message;

```



```

}

case 'confirm_shop_order': {
  return executeDbQuery(async (client) => {
    const orderRes = await client.query("SELECT * FROM product_orders WHERE
order_uid = $1", [data.get('orderUID')]);
    if (orderRes.rows.length === 0) return '找不到該筆訂單，可能已被處理。';
    const order = orderRes.rows[0];

    // [V35.5 修正] 讓檢查條件可以接受新的訂單狀態
    if (!['pending_payment', 'pending_confirmation'].includes(order.status)) {
      return `此訂單狀態為「${order.status}」，無法再次確認。`;
    }

    await client.query("UPDATE product_orders SET status = 'completed', updated_at =
NOW() WHERE order_uid = $1", [data.get('orderUID')]);
    const notifyMessage = { type: 'text', text: `📦 訂單更新通知\n您購買的「
${order.product_name}」訂單已確認收款！\n後續請與我們聯繫領取商品，謝謝。` };
    await enqueuePushTask(order.user_id, notifyMessage).catch(e => console.error(e));

    return `✅ 已成功確認訂單 (ID: ...${data.get('orderUID').slice(-6)}). \n系統已發送通知
給學員 ${order.user_name}。`;
  });
}

case 'cancel_shop_order_start': {
  const orderUID = data.get('orderUID');
  const order = await getProductOrder(orderUID);
  if (!order) return '找不到該訂單。';
  return { type: 'text', text: `您確定要取消學員 ${order.user_name} 的訂單「
${order.product_name}」嗎？\n\n⚠️ 此操作將會歸還 ${order.points_spent} 點給學員，並將商品
庫存加回 1。`, quickReply: { items: [ { type: 'action', action: { type: 'postback', label: '✅ 確認取消
', data: `action=cancel_shop_order_execute&orderUID=${orderUID}` } }, { type: 'action', action: {
type: 'message', label: '返回', text:
CONSTANTS.COMMANDS.TEACHER.SHOP_ORDER_MANAGEMENT } } ] } } };
}

case 'reject_shop_order': {
  const orderUID = data.get('orderUID');
  return executeDbQuery(async (client) => {
    const res = await client.query(
      "UPDATE product_orders SET status = 'pending_payment', last_5_digits = NULL,
updated_at = NOW() WHERE order_uid = $1 AND status = 'pending_confirmation'
RETURNING user_id, user_name, product_name",
      [orderUID]
    );
  });
}

```

```

    );

    if (res.rowCount > 0) {
      const order = res.rows[0];
      const notifyMessage = { type: 'text', text: `❗ 訂單退回通知\n您購買「
${order.product_name}」的回報資訊已被退回。請檢查後五碼或金額是否有誤，並重新回報。` };
      await enqueuePushTask(order.user_id, notifyMessage);
      return `✅ 已退回學員 ${order.user_name} 的訂單，並通知對方重新提交資訊。`;
    }
    return '找不到該筆待確認訂單，或已被處理。';
  });
}

case 'cancel_shop_order_execute': {
  const orderUID = data.get('orderUID');
  return executeDbQuery(async (client) => {
    await client.query('BEGIN');
    try {
      const orderRes = await client.query("SELECT * FROM product_orders WHERE
order_uid = $1 FOR UPDATE", [orderUID]);
      if (orderRes.rows.length === 0) { await client.query('ROLLBACK'); return '找不到該
訂單，可能已被處理。'; }
      const order = orderRes.rows[0];
      if (order.status !== 'pending') { await client.query('ROLLBACK'); return `此訂單狀態
為「${order.status}」，無法取消。`; }
      await client.query("UPDATE users SET points = points + $1 WHERE id = $2",
[order.points_spent, order.user_id]);
      await client.query("UPDATE products SET inventory = inventory + 1 WHERE id =
$1", [order.product_id]);
      await client.query("UPDATE product_orders SET status = 'cancelled', updated_at =
NOW() WHERE order_uid = $1", [orderUID]);
      const notifyMessage = { type: 'text', text: `❗ 訂單取消通知\n您兌換的「
${order.product_name}」訂單已被老師取消。已將花費的 ${order.points_spent} 點歸還至您的
帳戶。` };
      await enqueuePushTask(order.user_id, notifyMessage);
      await client.query('COMMIT');
      return `✅ 已成功取消訂單 (ID: ...${orderUID.slice(-6)}) 並歸還點數及庫存。`;
    } catch (err) { await client.query('ROLLBACK'); console.error('❌ 取消商城訂單失敗:',
err);
    return '取消訂單時發生錯誤，操作已復原。'; }
  });
}

case 'set_course_weekday': {
  const state = pendingCourseCreation[userId];

```

```

    if (!state || state.step !== 'await_weekday') return '新增課程流程已逾時或中斷。';
    state.weekday = parseInt(data.get('day'), 10);
    state.weekday_label = WEEKDAYS.find(d => d.value === state.weekday).label;
    state.step = 'await_time';
    return { type: 'text', text: `好的，課程固定在每${state.weekday_label}。\\n\\n請問上課時間是幾點？（請輸入四位數時間，例如：19:30）`, quickReply: { items: getCancelMenu() } };
  }
  case 'cancel_course_group_confirm': {
    const prefix = data.get('prefix');
    const courseTitle = await executeDbQuery(client => client.query("SELECT title FROM courses WHERE id LIKE $1 LIMIT 1", [`${prefix}%`])).then(res => res.rows[0]?.title);
    if (!courseTitle) return '找不到此課程系列。';
    const mainTitle = getCourseMainTitle(courseTitle);
    pendingCourseCancellation[userId] = { type: 'batch', prefix };
    setupConversationTimeout(userId, pendingCourseCancellation,
    'pendingCourseCancellation', u => enqueuePushTask(u, { type: 'text', text: '取消課程操作逾時。'
    }));
    return { type: 'text', text: `⚠ 警告：您確定要批次取消「${mainTitle}」系列的所有未來課程嗎？\\n此操作將會退還點數給所有已預約的學員，且無法復原。`, quickReply: { items: [{type:
    'action', action: { type: 'message', label:
    CONSTANTS.COMMANDS.TEACHER.CONFIRM_BATCH_CANCEL, text:
    CONSTANTS.COMMANDS.TEACHER.CONFIRM_BATCH_CANCEL}}, {type: 'action', action:
    {type: 'message', label: CONSTANTS.COMMANDS.GENERAL.CANCEL, text:
    CONSTANTS.COMMANDS.GENERAL.CANCEL } }]}];
  }
  case 'confirm_single_course_cancel': {
    const courseId = data.get('course_id');
    const course = await getCourse(courseId);
    if (!course) return '找不到此課程。';
    pendingCourseCancellation[userId] = { type: 'single', course_id: courseId };
    setupConversationTimeout(userId, pendingCourseCancellation,
    'pendingCourseCancellation', u => enqueuePushTask(u, { type: 'text', text: '取消課程操作逾時。'
    }));
    return { type: 'text', text: `您確定要取消單堂課程「${course.title}」嗎？\\n此操作將退還點數給已預約的學員。`, quickReply: { items: [{type: 'action', action: { type: 'message', label:
    CONSTANTS.COMMANDS.TEACHER.CONFIRM_SINGLE_CANCEL, text:
    CONSTANTS.COMMANDS.TEACHER.CONFIRM_SINGLE_CANCEL}}, {type: 'action', action:
    {type: 'message', label: CONSTANTS.COMMANDS.GENERAL.CANCEL, text:
    CONSTANTS.COMMANDS.GENERAL.CANCEL } }]}];
  }
  case 'select_booking_spots': {
    const course_id = data.get('course_id');
    const course = await getCourse(course_id);

```

```

    if (!course) return '抱歉，找不到該課程。';
    const remainingSpots = course.capacity - course.students.length;
    if (remainingSpots <= 0) return '抱歉，此課程名額已滿。';
    const maxSpots = Math.min(5, remainingSpots);
    const buttons = Array.from({ length: maxSpots }, (_, i) => ({ type: 'button', style:
'secondary', height: 'sm', margin: 'sm', action: { type: 'postback', label: `${i + 1} 位 (共
${course.points_cost * (i + 1)} 點)`, data:
`action=start_booking_confirmation&course_id=${course.id}&spots=${i + 1}` } }));
    return { type: 'flex', altText: '請選擇預約人數', contents: { type: 'bubble', header: { type:
'box', layout: 'vertical', contents: [{ type: 'text', text: '選擇預約人數', weight: 'bold', size: 'lg', color:
'#FFFFFF' }], backgroundColor: '#52b69a' }, body: { type: 'box', layout: 'vertical', contents: [ {
type: 'text', text: course.title, wrap: true, weight: 'bold', size: 'md' }, { type: 'text', text: `剩餘名額:
${remainingSpots} 位`, size: 'sm', color: '#666666', margin: 'md' }, { type: 'separator', margin: 'lg'
} ] }, footer: { type: 'box', layout: 'vertical', spacing: 'sm', contents: buttons } } } };
  }
  case 'start_booking_confirmation': {
    const course_id = data.get('course_id');
    const spotsToBook = parseInt(data.get('spots'), 10);
    const course = await getCourse(course_id);
    if (!course) return '抱歉，找不到該課程。';
    const totalCost = course.points_cost * spotsToBook;
    const remainingSpots = course.capacity - course.students.length;
    if (spotsToBook > remainingSpots) return `抱歉，課程名額不足！\n目前僅剩
${remainingSpots} 位。`;
    if (user.points < totalCost) return `抱歉，您的點數不足！\n預約 ${spotsToBook} 位需
${totalCost} 點，您目前有 ${user.points} 點。`;
    pendingBookingConfirmation[userId] = { type: 'confirm_book', course_id: course.id,
spots: spotsToBook };
    setupConversationTimeout(userId, pendingBookingConfirmation,
'pendingBookingConfirmation', (u) => {
      enqueuePushTask(u, { type: 'text', text: '預約操作已逾時，自動取消。' });
    });
    const message = `請確認預約資訊：\n\n課程：${course.title}\n時間：
${formatDateTime(course.time)}\n預約：${spotsToBook} 位\n花費：${totalCost} 點\n\n您目前的點
數為：${user.points} 點`;
    return { type: 'text', text: message, quickReply: { items: [ { type: 'action', action: { type:
'postback', label: '✅ 確認預約', data:
`action=execute_booking&course_id=${course.id}&spots=${spotsToBook}` } }, { type: 'action',
action: { type: 'message', label: CONSTANTS.COMMANDS.GENERAL.CANCEL, text:
CONSTANTS.COMMANDS.GENERAL.CANCEL } } ] } };
  }
  case 'execute_booking': {
    const course_id = data.get('course_id');

```

```

const spotsToBook = parseInt(data.get('spots'), 10);
const result = await executeDbQuery(async (clientDB) => {
  await clientDB.query('BEGIN');
  try {
    const userForUpdate = await clientDB.query('SELECT points, history FROM users
WHERE id = $1 FOR UPDATE', [userId]);
    const courseForUpdate = await clientDB.query('SELECT * FROM courses WHERE
id = $1 FOR UPDATE', [course_id]);
    const course = courseForUpdate.rows[0];
    const student = userForUpdate.rows[0];
    if (!course) { await clientDB.query('ROLLBACK'); return '抱歉，找不到該課程，可能
已被老師取消。'; }
    const remainingSpots = course.capacity - course.students.length;
    if (spotsToBook > remainingSpots) { await clientDB.query('ROLLBACK'); return `預約失敗，課程名額不足！\n目前剩餘 ${remainingSpots} 位，您想預約 ${spotsToBook} 位。`; }
    const totalCost = course.points_cost * spotsToBook;
    if (student.points < totalCost) { await clientDB.query('ROLLBACK'); return `預約失敗，您的點數不足！\n需要點數：${totalCost}\n您目前有：${student.points}`; }

    const newStudents = [...course.students, ...Array(spotsToBook).fill(userId)];
    const historyEntry = { action: `預約課程 (共${spotsToBook}位)：${course.title}`,
pointsChange: -totalCost, time: new Date().toISOString() };
    const newHistory = student.history ? [...student.history, historyEntry] : [historyEntry];
    await clientDB.query('UPDATE users SET points = points - $1, history = $2
WHERE id = $3', [totalCost, JSON.stringify(newHistory), userId]);
    await clientDB.query('UPDATE courses SET students = $1 WHERE id = $2',
[newStudents, course_id]);
    const reminderTime = new Date(new Date(course.time).getTime() -
CONSTANTS.TIME.ONE_HOUR_IN_MS);
    if (reminderTime > new Date()) {
      const reminderMessage = { type: 'text', text: `🔔 課程提醒 🔔\n您預約的課程「
${course.title}」即將在約一小時後開始，請準備好上課囉！` };
      await enqueuePushTask(userId, reminderMessage, reminderTime);
    }
    await clientDB.query('COMMIT');
    return `✅ 成功為您預約 ${spotsToBook} 個名額！\n課程：${course.title}\n時間：
${formatDateTime(course.time)}\n\n已為您扣除 ${totalCost} 點，期待課堂上見！`;
  } catch (e) {
    await clientDB.query('ROLLBACK');
    console.error('多人預約課程失敗:', e);
    return '預約時發生錯誤，請稍後再試。';
  }
});

```

```

    delete pendingBookingConfirmation[userId];
    return result;
  }
  case 'confirm_cancel_booking_start':
  case 'confirm_cancel_waiting_start': {
    const course_id = data.get('course_id');
    const course = await getCourse(course_id);
    if (!course) return '找不到該課程，可能已被老師取消或已結束。';
    const isBooking = action === 'confirm_cancel_booking_start';
    pendingBookingConfirmation[userId] = { type: isBooking ? 'cancel_book' : 'cancel_wait',
    course_id: course_id };
    setupConversationTimeout(userId, pendingBookingConfirmation,
    'pendingBookingConfirmation', (u) => enqueuePushTask(u, { type: 'text', text: '取消操作已逾時，
    自動放棄。' }));
    const actionText = isBooking ? '取消預約' : '取消候補';
    const confirmCommand = isBooking ?
    CONSTANTS.COMMANDS.STUDENT.CONFIRM_CANCEL_BOOKING :
    CONSTANTS.COMMANDS.STUDENT.CONFIRM_CANCEL_WAITING;
    return { type: 'text', text: `您確定要「${actionText}」以下課程嗎？\n\n課程：${course.title}\n
    時間：${formatDateTime(course.time)}`, quickReply: { items: [ { type: 'action', action: { type:
    'message', label: `✅ 確認${actionText}`, text: confirmCommand } }, { type: 'action', action: {
    type: 'message', label: CONSTANTS.COMMANDS.GENERAL.CANCEL, text:
    CONSTANTS.COMMANDS.GENERAL.CANCEL } } ] } };
  }
  case 'mark_feedback_read': {
    const msgId = data.get('msgId');
    if (!msgId) return '操作失敗，缺少訊息 ID。';
    await executeDbQuery(client => client.query("UPDATE feedback_messages SET status
    = 'read' WHERE id = $1 AND status = 'new'", [msgId]) );
    return `✅ 已將此留言標示為已讀。`;
  }
  case 'reply_feedback': {
    const msgId = data.get('msgId');
    const studentId = data.get('userId');
    if (!msgId || !studentId) return '操作失敗，缺少必要資訊。';
    const msgRes = await executeDbQuery(client => client.query("SELECT message FROM
    feedback_messages WHERE id = $1", [msgId]) );
    if (msgRes.rows.length === 0) return '找不到這則留言，可能已被其他老師處理。';
    const originalMessage = msgRes.rows[0].message;
    pendingReply[userId] = { msgId, studentId, originalMessage };
    setupConversationTimeout(userId, pendingReply, 'pendingReply', (u) =>
    enqueuePushTask(u, { type: 'text', text: '回覆留言操作逾時，自動取消。' }));
    return { type: 'text', text: `正在回覆學員的留言：\n「${originalMessage.substring(0, 80)}...」`
  }

```

```

\\n\\n請直接輸入您要回覆的內容：`, quickReply: { items: getCancelMenu() } } };
    }
    case 'select_student_for_auth': {
      const targetId = data.get('targetId');
      const targetName = decodeURIComponent(data.get('targetName'));
      if (!targetId || !targetName) return '操作失敗, 缺少目標學員資訊。';
      pendingTeacherAddition[userId] = { step: 'await_confirmation', targetUser: { id: targetId,
name: targetName } };
      setupConversationTimeout(userId, pendingTeacherAddition, 'pendingTeacherAddition', u
=> { enqueuePushTask(u, { type: 'text', text: '授權老師操作逾時。' })).catch(e =>
console.error(e)); });
      return { type: 'text', text: `您確定要授權學員「${targetName}」成為老師嗎？`, quickReply:
{ items: [ { type: 'action', action: { type: 'message', label:
CONSTANTS.COMMANDS.ADMIN.CONFIRM_ADD_TEACHER, text:
CONSTANTS.COMMANDS.ADMIN.CONFIRM_ADD_TEACHER } }, { type: 'action', action: {
type: 'message', label: CONSTANTS.COMMANDS.GENERAL.CANCEL, text:
CONSTANTS.COMMANDS.GENERAL.CANCEL } } ] } };
    }
    case 'select_teacher_for_removal': {
      const targetId = data.get('targetId');
      const targetName = decodeURIComponent(data.get('targetName'));
      if (!targetId || !targetName) return '操作失敗, 缺少目標老師資訊。';
      pendingTeacherRemoval[userId] = { step: 'await_confirmation', targetUser: { id: targetId,
name: targetName } };
      setupConversationTimeout(userId, pendingTeacherRemoval, 'pendingTeacherRemoval',
u => enqueuePushTask(u, { type: 'text', text: '移除老師操作逾時。' }));
      return { type: 'text', text: `您確定要移除老師「${targetName}」的權限嗎？\\n該用戶將會變
回學員身份。`, quickReply: { items: [ { type: 'action', action: { type: 'message', label:
CONSTANTS.COMMANDS.ADMIN.CONFIRM_REMOVE_TEACHER, text:
CONSTANTS.COMMANDS.ADMIN.CONFIRM_REMOVE_TEACHER } }, { type: 'action', action:
{ type: 'message', label: CONSTANTS.COMMANDS.GENERAL.CANCEL, text:
CONSTANTS.COMMANDS.GENERAL.CANCEL } } ] } };
    }
    case 'select_student_for_adjust': {
      const studentId = data.get('studentId');
      const student = await getUser(studentId);
      if (!student) return '找不到該學員的資料。';
      pendingManualAdjust[userId] = { step: 'await_operation', targetStudent: { id: student.id,
name: student.name } };
      setupConversationTimeout(userId, pendingManualAdjust, 'pendingManualAdjust', (u) =>
enqueuePushTask(u, { type: 'text', text: '手動調整點數逾時, 自動取消。' }));
      return { type: 'text', text: `已選擇學員：「${student.name}」。\\n請問您要為他加點或扣點？
`, quickReply: { items: [ { type: 'action', action: { type: 'message', label:

```



```

CONSTANTS.COMMANDS.TEACHER.ADD_POINTS, text:
CONSTANTS.COMMANDS.TEACHER.ADD_POINTS }}, { type: 'action', action: { type:
'message', label: CONSTANTS.COMMANDS.TEACHER.DEDUCT_POINTS, text:
CONSTANTS.COMMANDS.TEACHER.DEDUCT_POINTS }}, { type: 'action', action: { type:
'message', label: CONSTANTS.COMMANDS.GENERAL.CANCEL, text:
CONSTANTS.COMMANDS.GENERAL.CANCEL } } } }];
}
case 'select_announcement_for_deletion': {
  const ann_id = data.get('ann_id');
  const annRes = await executeDbQuery(client => client.query("SELECT content FROM
announcements WHERE id = $1", [ann_id]));
  if(annRes.rows.length === 0) return '找不到該公告。';
  pendingAnnouncementDeletion[userId] = { ann_id };
  setupConversationTimeout(userId, pendingAnnouncementDeletion,
'pendingAnnouncementDeletion', u => enqueuePushTask(u, { type: 'text', text: '刪除公告操作逾
時。' }));
  return { type: 'text', text: `您確定要刪除以下公告嗎？\n\n「
${annRes.rows[0].content.substring(0, 100)}...'`, quickReply: { items: [{type: 'action', action: {
type: 'message', label:
CONSTANTS.COMMANDS.TEACHER.CONFIRM_DELETE_ANNOUNCEMENT, text:
CONSTANTS.COMMANDS.TEACHER.CONFIRM_DELETE_ANNOUNCEMENT}}, {type:
'action', action: {type: 'message', label: CONSTANTS.COMMANDS.GENERAL.CANCEL, text:
CONSTANTS.COMMANDS.GENERAL.CANCEL } } } } }];
}
case 'retry_failed_task':
case 'delete_failed_task': {
  const failedTaskId = data.get('id');
  if (action === 'retry_failed_task') {
    return executeDbQuery(async (db) => {
      await db.query('BEGIN');
      try {
        const failedTaskRes = await db.query('SELECT * FROM failed_tasks WHERE id
= $1 FOR UPDATE', [failedTaskId]);
        if (failedTaskRes.rows.length === 0) { await db.query('ROLLBACK'); return '找不
到該失敗任務，可能已被處理。'; }
        const taskToRetry = failedTaskRes.rows[0];
        await db.query('INSERT INTO tasks (recipient_id, message_payload, status,
retry_count, last_error) VALUES ($1, $2, 'pending', 0, 'Retried from DLQ')',
[taskToRetry.recipient_id, taskToRetry.message_payload]);
        await db.query('DELETE FROM failed_tasks WHERE id = $1', [failedTaskId]);
        await db.query('COMMIT');
        return `✅ 已將任務 #${failedTaskId} 重新加入佇列等待發送。`;
      } catch (err) {

```



```

        await db.query('ROLLBACK');
        console.error(`❌ 重試失敗任務 ${failedTaskId} 失敗:`, err);
        return '處理任務時發生錯誤, 操作已取消。';
    }
    });
    } else { // delete_failed_task
        const result = await executeDbQuery(client => client.query('DELETE FROM
failed_tasks WHERE id = $1', [failedTaskId]) );
        return result.rowCount > 0 ? `✅ 已成功刪除失敗任務 #${failedTaskId}。` : '找不到該
失敗任務, 可能已被刪除。';
    }
}
case 'report_shop_last5': {
    const orderUID = data.get('orderUID');
    if (!orderUID) return '操作無效, 缺少訂單資訊。';

    pendingShopPayment[userId] = { orderUID };
    setupConversationTimeout(userId, pendingShopPayment, 'pendingShopPayment', (u)
=> {
        enqueuePushTask(u, { type: 'text', text: '輸入後五碼操作已逾時, 自動取消。' });
    });
    return {
        type: 'text',
        text: '請輸入您的匯款帳號後五碼 (5位數字):',
        quickReply: { items: getCancelMenu() }
    };
}
case 'run_command': {
    const commandText = decodeURIComponent(data.get('text'));
    if (commandText) {
        const simulatedEvent = { ...event, type: 'message', message: { type: 'text', id:
`simulated_${Date.now()}`, text: commandText } };
        if (user.role === 'admin') return handleAdminCommands(simulatedEvent, userId);
        if (user.role === 'teacher') return handleTeacherCommands(simulatedEvent, userId);
        return handleStudentCommands(simulatedEvent, userId);
    }
    break;
}
default:
    console.log(`[INFO] 未處理的 Postback Action: ${action}`);
    return null;
}
}
}

```

```

async function handleEvent(event) {
  if (event.type === 'unfollow' || event.type === 'leave') {
    console.log(`用戶 ${event.source.userId} 已封鎖或離開`);
    return;
  }
  if (!event.replyToken && event.type !== 'follow') {
    return;
  }
  if (event.type === 'follow') {
    try {
      const profile = await client.getProfile(event.source.userId);
      const user = { id: event.source.userId, name: profile.displayName, points: 0, role:
'student', history: [], picture_url: profile.pictureUrl };
      await saveUser(user);
      userProfileCache.set(event.source.userId, { timestamp: Date.now(), name:
profile.displayName, pictureUrl: profile.pictureUrl });

      const welcomeMessage = { type: 'text', text: `歡迎 ${user.name}！感謝您加入九容瑜伽。
`};

      await enqueuePushTask(event.source.userId, welcomeMessage).catch(err =>
console.error(`發送歡迎詞給新用戶 ${event.source.userId} 失敗:`, err.message));
      if (STUDENT_RICH_MENU_ID) await client.linkRichMenuToUser(event.source.userId,
STUDENT_RICH_MENU_ID);
    } catch (error) {
      console.error(`[Follow Event] 處理新用戶 ${event.source.userId} 時出錯:`,
error.message);
    }
    return;
  }

  const token = event.replyToken;
  if (repliedTokens.has(token)) {
    console.log('🔄 偵測到重複的 Webhook 事件，已忽略。');
    return;
  }
  repliedTokens.add(token);
  setTimeout(() => repliedTokens.delete(token), 60000);

  const userId = event.source.userId;
  let user = await getUser(userId);
  if (!user) {
    try {

```

```

    const profile = await client.getProfile(userId);
    user = { id: userId, name: profile.displayName, points: 0, role: 'student', history: [],
picture_url: profile.pictureUrl };
    await saveUser(user);
    userProfileCache.set(userId, { timestamp: Date.now(), name: profile.displayName,
pictureUrl: profile.pictureUrl });
    const welcomeMessage = { type: 'text', text: `歡迎 ${user.name}！感謝您加入九容瑜伽。
`};
    await enqueuePushTask(userId, welcomeMessage);
    if (STUDENT_RICH_MENU_ID) await client.linkRichMenuToUser(userId,
STUDENT_RICH_MENU_ID);
  } catch (error) { console.error(`創建新用戶時出錯:`, error); return; }
} else {
  const cachedData = userProfileCache.get(userId);
  const now = Date.now();
  if (!cachedData || (now - cachedData.timestamp > 10 * 60 * 1000)) {
    try {
      const profile = await client.getProfile(userId);
      if (profile.displayName !== user.name || (profile.pictureUrl && profile.pictureUrl !==
user.picture_url)) {
        user.name = profile.displayName;
        user.picture_url = profile.pictureUrl; await saveUser(user);
      }
      userProfileCache.set(userId, { timestamp: now, name: profile.displayName, pictureUrl:
profile.pictureUrl });
    } catch (e) { console.error(`[Cache] 更新用戶 ${userId} 資料時出錯:`, e.message); }
  }
}

const now = Date.now();
const lastInteraction = userLastInteraction[userId] || 0;
const isNewSession = (now - lastInteraction) >
CONSTANTS.INTERVALS.SESSION_TIMEOUT_MS;
userLastInteraction[userId] = now;
let notificationMessages = [];
if (isNewSession) {
  const notifications = await getPendingNotificationsForUser(user);
  if (notifications.newMessages > 0) notificationMessages.push({ type: 'text', text: `🔔 老師提醒: 您有 ${notifications.newMessages} 則新留言待回覆喔！` });
  if (notifications.pendingPointOrders > 0) notificationMessages.push({ type: 'text', text: `🔔 老師提醒: 您有 ${notifications.pendingPointOrders} 筆點數訂單待審核。` });
  if (notifications.pendingShopOrders > 0) notificationMessages.push({ type: 'text', text: `🔔 老師提醒: 您有 ${notifications.pendingShopOrders} 筆商城訂單待處理。` });
}

```

```

    if (notifications.failedTasks > 0) notificationMessages.push({ type: 'text', text: `🚨 管理員注意: 系統中有 ${notifications.failedTasks} 個失敗任務, 請至管理模式查看。`});
    if (notifications.unreadReplies > 0) notificationMessages.push({ type: 'text', text: `🔔 學員提醒: 您有 ${notifications.unreadReplies} 則老師的新回覆, 請至「聯絡我們」查看!`});
  }

```

```

let mainReplyContent;
let contextForError = '處理使用者指令';

```

```

try {
  const text = (event.type === 'message' && event.message.type === 'text') ?
event.message.text.trim() : '';

  let shouldClear = true;
  if (event.type === 'postback') {
    const postbackData = new URLSearchParams(event.postback.data);
    const action = postbackData.get('action');
    const continuationActions = [ 'set_course_weekday', 'select_teacher_for_course',
'confirm_add_product', 'edit_product_field', 'start_booking_confirmation', 'execute_booking',
'execute_product_purchase', 'confirm_teacher_profile_update',
'start_purchase_history_search', 'start_exchange_history_search',
'start_message_history_search', 'select_student_for_purchase_history',
'select_student_for_exchange_history', 'select_student_for_message_history'];
    if (continuationActions.includes(action)) {
      shouldClear = false;
    }
  }

  if (shouldClear && (text && text.startsWith('@') || event.type === 'postback')) {
    const wasCleared = clearPendingConversations(userId);
    if (wasCleared) console.log(`使用者 ${userId} 的待辦任務已由新操作自動取消。`);
  }

  if (text === CONSTANTS.COMMANDS.GENERAL.CANCEL) {
    const wasCleared = clearPendingConversations(userId);
    mainReplyContent = wasCleared ? '已取消先前的操作。' : '目前沒有可取消的操作。';
  }
  else if (userId === ADMIN_USER_ID && text ===
CONSTANTS.COMMANDS.ADMIN.PANEL) {
    contextForError = '進入管理模式';
    if (user.role !== 'admin') {
      user.role = 'admin';
      await saveUser(user);
    }
  }
}

```

```

    }
    mainReplyContent = await handleAdminCommands(event, userId);
  }
  else if (event.type === 'message') {
    contextForError = `處理訊息: ${text}`;
    switch(user.role) {
      case 'admin': mainReplyContent = await handleAdminCommands(event, userId);
break;
      case 'teacher': mainReplyContent = await handleTeacherCommands(event, userId);
break;
      default: mainReplyContent = await handleStudentCommands(event, userId); break;
    }
  }
  else if (event.type === 'postback') {
    const action = new URLSearchParams(event.postback.data).get('action');
    contextForError = `處理 Postback: ${action}`;
    mainReplyContent = await handlePostback(event, user);
  }
} catch(err) {
  await handleError(err, event.replyToken, contextForError);
  return;
}

const finalMessages = [...notificationMessages];
if (mainReplyContent) {
  const contentArray = Array.isArray(mainReplyContent) ? mainReplyContent :
[mainReplyContent];
  finalMessages.push(...contentArray);
}

if (finalMessages.length > 0) {
  const formattedMessages = finalMessages
    .filter(Boolean)
    .map(m => (typeof m === 'string' ? { type: 'text', text: m } : m));
  if (formattedMessages.length > 0) {
    try {
      await reply(event.replyToken, formattedMessages);
    } catch (e) {
      console.error(`[FATAL] 在 handleEvent 中捕捉到 reply 函式的嚴重錯誤 for ${userId}:`,
e);
    }
  }
}
}
}

```

