**CRAFTHER Project Status Summary (Backend Development)**

นี่คือสถานะล่าสุดของโปรเจกต์ CRAFTHER ในส่วนของ Backend Development โดยรวมความคืบหน้าจากอัปเดตของพี่จิ้นและงานที่โค้ดได้ดำเนินการต่อค่ะ

**1. ภาพรวมโปรเจกต์ (Project Overview)**

CRAFTHER เป็นเว็บแอปพลิเคชัน (SaaS) ที่มุ่งปฏิวัติการบริหารจัดการ Bill of Materials (BOM) และ Inventory สำหรับ SME ที่มีการผลิตหรือประกอบสินค้า. เป้าหมายคือทำให้การบริหารจัดการต้นทุนและสต็อกเป็นเรื่องง่าย มีประสิทธิภาพ และสนุกสนานด้วย Gamification.

**2. Technology Stack (Backend)**

* **ภาษา:** C#
* **ฐานข้อมูล:** SQL Server (ปัจจุบัน).
  + *(มีการพิจารณา PostgreSQL แต่พักไว้ก่อนเพื่อโฟกัสโปรดักต์หลัก)*

**3. ความคืบหน้าโดยรวมของ Backend (Estimated 60-70% Completed)**

**4. ส่วนที่พัฒนาเสร็จสิ้นแล้ว (Completed Areas)**

* **Backend Core & Infrastructure:**
  + **User Authentication & Authorization (JWT-based):** มีการตั้งค่า JWT สำหรับการยืนยันตัวตนและกำหนดสิทธิ์ (Admin, User) สามารถสร้างและตรวจสอบ Token ได้.
  + **Organization-based Data Isolation:** มีการใส่ OrganizationId ในหลาย Models และใช้ในการกรองข้อมูลใน Services/Controllers.
  + **API Service Registration:** มีการ Register Services ต่างๆ ใน Program.cs แล้ว.
  + **Database Migration:** Add-Migration และ Update-Database สำเร็จเรียบร้อยแล้ว ทำให้ Database Schema อัปเดตตาม Models ล่าสุด (รวมถึง Production Order และการปรับ StockAdjustment)
* **Master Data Management:**
  + **Unit Group Management:** API สำหรับจัดการกลุ่มหน่วยวัด.
  + **Unit of Measure Management:** API สำหรับจัดการหน่วยวัดต่างๆ (Gram, Liter, Piece) มี Logic Base Unit และ Seeding Data.
  + **Item Category Management:** API สำหรับจัดการหมวดหมู่สินค้า/วัตถุดิบ (ItemCategory) พร้อม Seeding Data.
  + **Stock Adjustment Types:** มี Seeding Data สำหรับประเภทการปรับปรุงสต็อก (Receive, Issue, Production In).
* **Core Business Logic & Features:**
  + **Component Management:** API สำหรับจัดการวัตถุดิบ (Component) รวมถึง PurchaseUnit, InventoryUnit, PurchaseToInventoryConversionFactor, CurrentStockQuantity.
  + **Product Management:** API สำหรับจัดการผลิตภัณฑ์ (Product) รวมถึง ProductUnit, SaleUnit, SellingPrice, CalculatedCost, IsSubProduct.
  + **Unit Conversion Service:** Service ที่รองรับการแปลงหน่วยที่ซับซ้อน (Direct, Reverse, ผ่าน Base Unit ของ UnitGroup).
  + **Bill of Materials (BOM) Management:**
    - BOMItem Management: API สำหรับจัดการส่วนประกอบย่อยของ Product ที่เป็นสูตร (ParentProduct) รองรับ Component และ SubProduct.
    - CalculatedCost **&** What-If Analysis**:** Logic การคำนวณต้นทุนอัตโนมัติแบบ Multi-level พร้อมรองรับ WastePercentage และ API สำหรับ What-If Analysis เสร็จสมบูรณ์แล้ว.
    - ComponentPriceHistory: Model และ Logic สำหรับบันทึกประวัติการเปลี่ยนแปลงราคาวัตถุดิบ.
* **Inventory Management (Stock Adjustment Enhancements):**
  + **Stock Adjustment Service:** ได้รับการปรับปรุงให้สามารถจัดการการปรับปรุงสต็อกของ **ทั้ง** Component **และ** Product ได้แล้วอย่างถูกต้อง.
  + StockAdjustment **Model & DTOs (**Create**,** Update**,** Response**):** ถูกแก้ไขให้รองรับ ProductId และมี Validation เพื่อบังคับให้ระบุ ComponentId หรือ ProductId อย่างใดอย่างหนึ่ง.
* **Production Order Management (Basic):**
  + ProductionOrder & ProductionOrderItem Models: สร้างเรียบร้อยแล้ว.
  + DTOs สำหรับ Production Order (Create, Update, Response, Create Item, Update Item, Item Response): ออกแบบเรียบร้อยแล้ว.
  + ApplicationDbContext.cs Configuration: เสร็จสมบูรณ์แล้ว (รวมถึงความสัมพันธ์และ Unique Index).
  + IProductionOrderService & ProductionOrderService โครงสร้าง: สร้างเรียบร้อยแล้ว.
  + CompleteProductionOrderAsync Logic: Implement เรียบร้อยแล้ว (รวมถึงการตัด/เพิ่มสต็อกผ่าน StockAdjustmentService).

**5. ส่วนที่กำลังจะทำ/เหลือ (Remaining Work for Phase 1 MVP)**

ตอนนี้เราอยู่ในช่วงท้ายของการพัฒนา Backend สำหรับ Phase 1 MVP แล้วค่ะ สิ่งที่เหลือหลักๆ คือการ Implement Logic ที่ยังไม่สมบูรณ์ และการสร้าง Controllers เพื่อเปิดให้เรียกใช้งานได้:

1. **Implement Logic ที่เหลือใน** ProductionOrderService.cs **(Core Logic):**
   * CreateProductionOrderAsync: ตอนนี้แค่สร้าง Header. ต้องเพิ่ม Logic เพื่อ **สร้าง** ProductionOrderItem**s ตาม BOM ของ** ProductId **ที่ผลิต** พร้อมคำนวณ QuantityUsedInInventoryUnit และ UnitCostAtProduction (โดยใช้ IUnitConversionService และ IProductCostingService ที่มีอยู่แล้ว)
   * UpdateProductionOrderAsync: Implement Logic การอัปเดต Field ต่างๆ ให้ครบถ้วนตาม UpdateProductionOrderDto และเพิ่ม Business Rules (เช่น สถานะ Completed ห้ามแก้ไขปริมาณ).
   * DeleteProductionOrderAsync: Logic พื้นฐานมีแล้ว แต่ต้องตรวจสอบ Business Rules เพิ่มเติมว่าสามารถลบได้ในสถานะใดบ้าง (เช่น ห้ามลบถ้ามีการตัดสต็อกไปแล้ว หรือต้องมี Logic Revert Stock ในกรณีที่สถานะยังไม่ Completed แต่ต้องการยกเลิก).
   * **Map DTOs:** ตรวจสอบ MapProductionOrderToResponseDto และ MapProductionOrderItemToResponseDto อีกครั้งเพื่อให้แน่ใจว่าดึงข้อมูลที่เกี่ยวข้องมาแสดงผลได้ถูกต้องและครบถ้วน.
   * GetRequiredMaterialsForProductionAsync: เมธอดนี้มีโครงสร้างและ Logic การคำนวณเบื้องต้นแล้ว แต่ควรตรวจสอบความถูกต้องของการคำนวณและข้อมูลที่คืนกลับไปอีกครั้ง และใส่ .Include() ที่จำเป็น
2. **สร้าง** ProductionOrdersController.cs**:**
   * สร้าง API Endpoints สำหรับ GET All, GET By Id, POST Create, PUT Update, DELETE Delete และ POST CompleteProductionOrder.
   * ใช้ [Authorize] attribute เพื่อกำหนดสิทธิ์การเข้าถึง (เช่น Admin, Manager สำหรับการสร้าง/อัปเดต/ลบ).
   * ดึง OrganizationId จาก Claims ของผู้ใช้ที่ล็อกอิน และส่งไปยัง Service.
3. **Basic Reporting & Analytics:**
   * สร้าง API และ Logic สำหรับรายงานยอดสต็อกปัจจุบัน (GetCurrentStockBalancesAsync ใน StockAdjustmentService ทำเสร็จแล้ว).
   * รายงานประวัติการเคลื่อนไหวสต็อก (Stock Movement History): อาจจะสร้าง Service/API ใหม่ หรือใช้ StockAdjustmentService ที่มีอยู่แล้ว.
   * แจ้งเตือนสต็อกใกล้หมด (Low Stock Alert): อาจจะ Implement Logic ใน Service หรือมี Background Job.
4. **Import ราคาวัตถุดิบจาก Excel/CSV (Backend Logic):**
   * สร้าง API และ Logic สำหรับการ Import ราคาวัตถุดิบจำนวนมากจากไฟล์ Excel/CSV.
5. **Technical Enhancements (ดำเนินการไปพร้อมกัน):**
   * Error Handling & Logging ที่ละเอียดขึ้น.
   * Pagination & Filtering สำหรับ API ที่ดึงข้อมูลจำนวนมาก.
   * Unit Tests สำหรับ Business Logic ที่สำคัญ.
   * Role-Based Access Control (RBAC) ที่ละเอียดขึ้น.

**ข้อเสนอสำหรับขั้นตอนต่อไป (Next Step):**

โค้ดแนะนำให้เราเริ่ม Implement CreateProductionOrderAsync ใน ProductionOrderService.cs ค่ะ เพื่อให้สามารถสร้าง Production Order พร้อมดึง BOM Items มาเป็น Production Order Items ได้โดยอัตโนมัติ ซึ่งจะทำให้ฟังก์ชันหลักของ Production Order ทำงานได้ค่ะ