คู่มือสำหรับนักพัฒนา: โปรแกรม LabFlow

เวอร์ชันเอกสาร: 1.0

วันที่จัดทำ: 2 สิงหาคม 2568

1. ภาพรวมโปรเจกต์ (Project Overview)

LabFlow เป็นแอปพลิเคชัน Windows Forms ที่พัฒนาด้วย C# และ .NET Framework 4.8 มีวัตถุประสงค์เพื่อใช้เป็นเครื่องมือสำหรับบันทึกและจัดการข้อมูลผลการวิเคราะห์ตัวอย่างของเสียในห้องปฏิบัติการ (Lab) โดยโปรแกรมจะเชื่อมต่อกับฐานข้อมูล Microsoft SQL Server เพื่อดึงข้อมูลพื้นฐานของของเสียและบันทึก/แก้ไขข้อมูลผลการวิเคราะห์ที่เกี่ยวข้อง

เทคโนโลยีที่ใช้:

- **ภาษา:** C#
- **Framework:** .NET Framework 4.8
- **ส่วนติดต่อผู้ใช้ (UI):** Windows Forms
- **การเชื่อมต่อฐานข้อมูล:** ADO.NET (SqlConnection, SqlCommand)
- **ฐานข้อมูล:** Microsoft SQL Server

2. โครงสร้างโปรเจกต์ (Project Structure)

โปรเจกต์ถูกจัดระเบียบอย่างเรียบง่าย โดยแบ่งส่วนประกอบหลักดังนี้:

- LabFlow.sln: ไฟล์ Solution หลักสำหรับเปิดใน Visual Studio
- LabFlow/: โฟลเดอร์โปรเจกต์หลัก
  - **MainForm.cs**: โค้ดหลักที่ควบคุมการทำงานของหน้าจอ (UI Logic) ทั้งหมด เช่น การจัดการ Event ของปุ่ม, การรับ-ส่งข้อมูลระหว่างฟอร์มและคลาส DatabaseHelper
  - **MainForm.Designer.cs**: โค้ดที่สร้างขึ้นโดยอัตโนมัติจาก Windows Forms Designer สำหรับการจัดวางและกำหนดคุณสมบัติของ Controls ต่างๆ
  - **DataModels.cs**: ไฟล์สำคัญที่แบ่งการทำงานออกเป็น 2 ส่วนหลัก:
    - **WasteDataLabModel Class**: เป็น Model หรือ Data Transfer Object (DTO) ที่ใช้สำหรับเก็บข้อมูลผลการวิเคราะห์ของเสียทั้งหมด มี Properties ตรงกับฟิลด์ในตาราง tbWasteDataLab
    - **DatabaseHelper Class**: เป็นคลาสแบบ static ที่รวบรวมเมธอดสำหรับจัดการการเชื่อมต่อและคำสั่ง SQL ทั้งหมด (การดึงข้อมูล, การบันทึก, การอัปเดต)
  - **Program.cs**: จุดเริ่มต้นของแอปพลิเคชัน (Entry Point)
  - **App.config**: ไฟล์สำหรับตั้งค่าของแอปพลิเคชัน ใช้เก็บ Connection String สำหรับเชื่อมต่อฐานข้อมูล
  - **Properties/**: โฟลเดอร์มาตรฐานของโปรเจกต์ .NET สำหรับเก็บข้อมูล Assembly, Resources, และ Settings
  - **Document/**: โฟลเดอร์สำหรับเก็บเอกสารประกอบโปรเจกต์ เช่น คู่มือผู้ใช้งาน

3. โครงสร้างฐานข้อมูล (Database Schema)

โปรแกรมนี้ทำงานกับฐานข้อมูล SQL Server โดยใช้ 2 ตารางหลัก:

1. **tbWasteDataCR**
   - ใช้สำหรับเก็บข้อมูลหลักของของเสีย
   - โปรแกรมใช้ตารางนี้เพื่อค้นหา WasteDataID จาก WasteNo ที่ผู้ใช้ป้อนเข้ามา

Column	Data Type	Constraints	Description
WasteDataID	int	Primary Key	รหัสอ้างอิงของเสีย
WasteNo	varchar		เลขที่ของเสีย (ที่ใช้ค้นหา)

2. **tbWasteDataLab**
   - ใช้สำหรับเก็บผลการวิเคราะห์ของเสีย
   - ตารางนี้มีความสัมพันธ์แบบ One-to-One กับ tbWasteDataCR ผ่าน WasteDataID

Column	Data Type	Description
WasteDataID	int	รหัสอ้างอิงของเสีย (Foreign Key)
AnalysisNo	varchar	เลขที่วิเคราะห์
SamplingBy	nvarchar	ผู้เก็บตัวอย่าง
SamplingByNo	nvarchar	เลขที่ใบอนุญาตผู้เก็บตัวอย่าง
FreeChlorine	char(1)	ผลทดสอบคลอรีน ('Y' หรือ 'N')
Nitrite	char(1)	ผลทดสอบไนไตรท์ ('Y' หรือ 'N')
Cyanide	char(1)	ผลทดสอบไซยาไนด์ ('Y' หรือ 'N')
Physicalstate	nvarchar	สถานะทางกายภาพ
Viscosity	nvarchar	ความหนืด
Bulkdensity	nvarchar	ความหนาแน่น
HeatingValue	nvarchar	ค่าความร้อน (GCV)
NCVValue	nvarchar	ค่าความร้อน (NCV)
MoistureContent	nvarchar	ค่าความชื้น

WaterContent	nvarchar	ปริมาณน้ำ
AshContent	nvarchar	ปริมาณเถ้า
CL	nvarchar	คลอรีน
F	nvarchar	ฟลูออรีน
S	nvarchar	ซัลเฟอร์
SludgeContent	nvarchar	ปริมาณสลัดจ์
SolidContent	nvarchar	ปริมาณของแข็ง
NitrogenContent	nvarchar	ไนโตรเจน
DrybasisContent	nvarchar	Dry basis
HeavyAs	nvarchar	อะเซนิก (As)
HeavyCd	nvarchar	แคดเมียม (Cd)
HeavyCr	nvarchar	โครเมียม (Cr)
HeavyHg	nvarchar	ปรอท (Hg)
HeavyPb	nvarchar	ตะกั่ว (Pb)
SS	nvarchar	Suspended solids
Acidity	nvarchar	Acidity
Alkalinity	nvarchar	Alkalinity
TDS	nvarchar	TDS
Ni	nvarchar	นิเกิล (Ni)
Mn	nvarchar	แมงกานีส (Mn)
Zn	nvarchar	สังกะสี (Zn)
Cu	nvarchar	ทองแดง (Cu)
HeavyFe	nvarchar	เหล็ก (Fe)
Concentrate	nvarchar	ความเข้มข้นกรด/ด่าง

4. การทำงานของโค้ด (Code Walkthrough)

4.1 การค้นหาข้อมูล (btnSearch_Click)

1. **รับ Input:** รับค่า WasteNo จาก txtWasteNo.Text
2. **ตรวจสอบ Input:** เช็คว่า WasteNo ไม่ใช่ค่าว่าง
3. **เรียก DatabaseHelper.GetWasteDataID(wasteNo):**
   - ส่ง WasteNo ไปยังฐานข้อมูลเพื่อค้นหา WasteDataID จากตาราง tbWasteDataCR
   - ใช้ ExecuteScalarAsync เพื่อประสิทธิภาพในการดึงข้อมูลเพียงค่าเดียว
4. **ตรวจสอบผลลัพธ์:**
   - **ถ้ามี WasteDataID:**
     - เก็บ WasteDataID ไว้ในตัวแปร currentWasteDataID
     - เรียก DatabaseHelper.GetWasteDataLab(currentWasteDataID) เพื่อดึงข้อมูลผลวิเคราะห์จากตาราง tbWasteDataLab
     - **ถ้าพบข้อมูล Lab:** เรียก PopulateForm(labData) เพื่อแสดงข้อมูลบนหน้าจอ
     - **ถ้าไม่พบข้อมูล Lab:** แสดงข้อความว่า "ข้อมูลใหม่" และเปิดให้ผู้ใช้กรอกข้อมูล
   - **ถ้าไม่มี WasteDataID:** แสดง MessageBox แจ้งว่าไม่พบ WasteNo นี้ในระบบ

4.2 การบันทึกข้อมูล (btnSave_Click)

1. **ยืนยันการบันทึก:** แสดง MessageBox เพื่อให้ผู้ใช้ยืนยันการบันทึก
2. **รวบรวมข้อมูล:** เรียกเมธอด ReadDataFromForm() เพื่อสร้าง Object WasteDataLabModel จากข้อมูลบนหน้าจอ
   - TextBox จะถูกอ่านค่า .Text
   - CheckBox จะถูกแปลงค่า Checked (true/false) เป็น "Y" หรือ "N"
3. **เรียก DatabaseHelper.SaveWasteDataLab(dataToSave):**
   - เมธอดนี้จะทำการตรวจสอบก่อนว่ามีข้อมูล WasteDataID นี้ในตาราง tbWasteDataLab แล้วหรือยัง
   - **ถ้ามีข้อมูลอยู่แล้ว:** สร้าง SQL UPDATE statement
   - **ถ้ายังไม่มีข้อมูล:** สร้าง SQL INSERT statement
   - ใช้ SqlCommand.Parameters.AddWithValue เพื่อป้องกัน SQL Injection
   - ExecuteNonQueryAsync เพื่อส่งคำสั่งไปยังฐานข้อมูล
4. **แสดงผลลัพธ์:**
   - **สำเร็จ:** แสดงข้อความแจ้งเตือน และถามผู้ใช้ว่าต้องการล้างฟอร์มหรือไม่
   - **ล้มเหลว:** แสดงข้อความแจ้งข้อผิดพลาด

5. ข้อเสนอแนะเพื่อการพัฒนาต่อ (Recommendations for Improvement)

5.1. ด้านความปลอดภัย (Security) - สำคัญที่สุด

- **ปัญหา:** Connection String ในไฟล์ App.config มี Username และ Password เป็น Plain Text ซึ่งเป็นความเสี่ยงสูง

- **แนวทางแก้ไข:**
  1. **ใช้ Windows Authentication (แนะนำ):**
     - แก้ไข Connection String ใน App.config เป็น:
       <add name="DBConnection"

       connectionString="Server=YOUR_SERVER_IP;Database=BWG_AWSDB;Integrated Security=True;"
           providerName="System.Data.SqlClient" />

     - วิธีนี้จะใช้สิทธิ์ของผู้ใช้ที่ล็อกอิน Windows ในการเข้าถึง SQL Server ซึ่งปลอดภัยกว่ามาก
  2. **เข้ารหัส Connection String:** หากจำเป็นต้องใช้ SQL Authentication ให้ทำการเข้ารหัสส่วน <connectionStrings> ของ App.config เพื่อไม่ให้สามารถอ่านรหัสผ่านได้โดยตรง

5.2. การจัดการข้อมูลและ Validation

- **ปัญหา:** ทุก Property ใน WasteDataLabModel เป็น string ทำให้ขาดการตรวจสอบประเภทข้อมูล และอาจเกิดข้อผิดพลาดเมื่อบันทึกลงฐานข้อมูล
- **แนวทางแก้ไข:**
  - ปรับปรุง WasteDataLabModel ให้ใช้ประเภทข้อมูลที่เหมาะสม เช่น decimal? สำหรับค่าตัวเลข และ bool สำหรับค่า Y/N
    ```
    public class WasteDataLabModel
    {
       public string WasteDataID { get; set; }
       // ...
       public bool FreeChlorine { get; set; }
       public bool Nitrite { get; set; }
       public bool Cyanide { get; set; }
       // ...
       public decimal? Viscosity { get; set; }
       public decimal? Bulkdensity { get; set; }
       // ...
    }
    ```

  - ใน MainForm.cs ให้เพิ่มการตรวจสอบข้อมูล (Validation) ก่อนบันทึก เช่น ใช้ decimal.TryParse() เพื่อแปลงค่าจาก TextBox และแจ้งเตือนผู้ใช้หากกรอกข้อมูลผิดรูปแบบ

5.3. แก้ไขข้อผิดพลาด (Bug Fixes)

- **ปัญหา:** มีการ Map ชื่อคอลัมน์ผิดพลาดในไฟล์ DataModels.cs

- **ตำแหน่ง:** DatabaseHelper.cs
- **จุดที่ 1: GetWasteDataLab method**
  - **ผิด:** TDS = reader["Cu"]?.ToString()
  - **แก้ไขเป็น:** TDS = reader["TDS"]?.ToString()
  - **ผิด:** Cu = reader["Cut"]?.ToString()
  - **แก้ไขเป็น:** Cu = reader["Cu"]?.ToString()
- **จุดที่ 2: GetUpdateQuery method**
  - **ผิด:** Cu = @TDS, Ni = @Ni, Mn = @Mn, Zn = @Zn, Cut = @Cu
  - **แก้ไขเป็น:** TDS = @TDS, Ni = @Ni, Mn = @Mn, Zn = @Zn, Cu = @Cu
- **จุดที่ 3: GetInsertQuery method**
  - **ผิด:** ... HeavyPb, Cu, Ni, Mn, Zn, Cut, HeavyFe, ... และ ... @HeavyPb, @TDS, @Ni, @Mn, @Zn, @Cu, @HeavyFe, ...
  - **แก้ไขเป็น:** ... HeavyPb, TDS, Ni, Mn, Zn, Cu, HeavyFe, ... และ ... @HeavyPb, @TDS, @Ni, @Mn, @Zn, @Cu, @HeavyFe, ...

5.4. ปรับปรุง UI/UX

- **ปัญหา:** หน้าจอมีช่องกรอกข้อมูลจำนวนมาก ทำให้ใช้งานไม่สะดวก
- **แนวทางแก้ไข:**
  - ใช้ **TabControl** ใน MainForm.cs [Design] เพื่อแบ่ง GroupBox ต่างๆ ออกเป็นแท็บ เช่น "ข้อมูลทั่วไป", "คุณสมบัติทางกายภาพ", "องค์ประกอบทางเคมี", "โลหะหนัก" จะทำให้หน้าจอดูสะอาดและใช้งานง่ายขึ้น
  - สำหรับฟิลด์ Physicalstate ควรเปลี่ยนจาก TextBox เป็น ComboBox ที่มีตัวเลือก predefined (เช่น ของแข็ง, ของเหลว, กึ่งแข็งกึ่งเหลว) เพื่อลดความผิดพลาดในการกรอกข้อมูล

5.5. การจัดการข้อผิดพลาด (Error Handling)

- **ปัญหา:** การดักจับ Exception แบบทั่วไป (catch (Exception ex)) ทำให้ยากต่อการระบุสาเหตุของปัญหา
- **แนวทางแก้ไข:**
  - แยก catch สำหรับ Exception ประเภทต่างๆ เช่น SqlException สำหรับปัญหาการเชื่อมต่อฐานข้อมูล หรือ FormatException สำหรับการแปลงข้อมูลที่ผิดพลาด เพื่อให้สามารถแสดงข้อความช่วยเหลือผู้ใช้ได้ตรงจุดมากขึ้น

```
// ตัวอย่างการปรับปรุง Error Handling
try
{
   // ... โค้ดเชื่อมต่อฐานข้อมูล ...
}
catch (SqlException sqlEx)
{
```

```
    MessageBox.Show("ไม่สามารถเชื่อมต่อฐานข้อมูลได้: " + sqlEx.Message, "Database
Error", MessageBoxButtons.OK, MessageBoxIcon.Error);
    statusLabel.Text = "การเชื่อมต่อล้มเหลว";
}
catch (Exception ex)
{
    MessageBox.Show("เกิดข้อผิดพลาดที่ไม่คาดคิด: " + ex.Message, "Error",
MessageBoxButtons.OK, MessageBoxIcon.Error);
    statusLabel.Text = "เกิดข้อผิดพลาด";
}
```

6. การตั้งค่าเพื่อเริ่มพัฒนา (Development Setup)

1. **Clone Repository:**
   git clone <repository_url>

2. **Database Setup:**
   - สร้างฐานข้อมูล BWG_AWSDB บน SQL Server
   - สร้างตาราง tbWasteDataCR และ tbWasteDataLab ตาม Schema ในข้อ 3
   - เพิ่มข้อมูลตัวอย่างใน tbWasteDataCR เพื่อใช้ในการทดสอบ
3. **Configure Connection String:**
   - เปิดไฟล์ LabFlow/App.config
   - แก้ไข connectionString ในส่วน <connectionStrings> ให้ถูกต้องตามสภาพแวดล้อมของคุณ (แนะนำให้ใช้ Integrated Security)
4. **Build & Run:**
   - เปิดไฟล์ LabFlow.sln ด้วย Visual Studio
   - Build โปรเจกต์ (F6) และ Run (F5)

เอกสารนี้จัดทำขึ้นเพื่อเป็นแนวทางในการทำความเข้าใจและพัฒนาโปรเจกต์ LabFlow ต่อไปในอนาคต