



# 살균기 품질 관리

조민석, 장순재, 김동현, 한형빈

STEP

1

프로젝트 개요

STEP

2

팀 구성 역할

STEP

3

프로젝트 수행

STEP

4

자체  
평가 의견

# 프로젝트 개요

## KB 품질 관리 프로그램

---

- ▶ 살균기는 저온 살균 및 교반하기 때문에 후 공정이 중요
- ▶ 이에 따른 품질분석으로 예견된 불량을 예지하는 프로그램



주제 선정 배경

실제로 사용한 데이터를  
이용하여 실용적인 sw제작



프로젝트 개요

.csv 파일을 mssql에 파싱하여  
사용  
. List 형태로 저장



활용 장비 재료

. MS SQL  
. Visual studio .Net Framework(C#)



프로젝트 결과

. 대용량 제조 데이터 시각화  
. 생산데이터 출력, 저장 삭제

# 제조데이터 소개

분석 목적

살균 공정의 불량 발생 원인 분석

제조 분야

분무건조공법을 이용한 분말유크림 제조

수집 기간

2020년 3월 4일 ~ 2020년 11월 11일

데이터 크기, 수량

6개 칼럼, 210,794개 관측치 (6.11MB)

수집 방식

살균 공정을 설비 2대 (A, B)에 나눠 병렬적으로 진행

측정값

살균기 2대의 온도 (A, B), 불량 여부 (OK or NG)

# 팀 구성원 역할

조민석

한형빈

장순재

김동현

팀장  
전체적인  
SW설계 및 코드  
구조 설정

발표자  
자료수집 및 분석  
데이터 수집

요구사항 정의서  
및 데이터 명세서  
작성

세부적인 기능  
구현 및 코드  
수정

```
public class Pasteurizer
{
    public DateTime STD_DT { get; set; }
    public string MIXA_PASTEUR_STATE { get; set; }
    public string MIXB_PASTEUR_STATE { get; set; }
    public string MIXA_PASTEUR_TEMP { get; set; }
    public string MIXB_PASTEUR_TEMP { get; set; }
    public string INSP { get; set; }
}
```

```
private static SqlConnection conn = new SqlConnection();
public static SqlDataAdapter da;
public static DataSet ds;
public static DataTable dt;
private static string TABLENAME = " pasteurizer ";
private static void connectDB()
{
    string dataSource = "local";
    string db = "pasteurizer";
    string security = "SSPI";

    conn.ConnectionString = $"Data Source={dataSource}; " +
        $"initial Catalog={db}; " +
        $"integrated Security = {security}; " +
        $"Timeout=3";
    conn = new SqlConnection(conn.ConnectionString);
    conn.Open();
}
```

## DataManager – Load()

```
public static List<Pasteurizer> Instance = new List<Pasteurizer>();

static DataManager()
{
    Load();
}
```

```
try
{
    DBHelper.selectQuery();
    Instance.Clear();
    foreach (DataRow item in DBHelper.dt.Rows)
    {
        Pasteurizer i = new Pasteurizer();

        // STD_DT 공백 처리
        string stdDtString = item["STD_DT"].ToString();
        i.STD_DT = !string.IsNullOrEmpty(stdDtString) ?
            DateTime.Parse(stdDtString) : DateTime.MinValue;

        // MIXA_PASTEUR_STATE가 공백이 아닐 때만 값을 읽어옵니다.
        if (!string.IsNullOrEmpty(item["MIXA_PASTEUR_STATE"].ToString()))
        {
            i.MIXA_PASTEUR_STATE = item["MIXA_PASTEUR_STATE"].ToString();
        }

        // MIXB_PASTEUR_STATE가 공백이 아닐 때만 값을 읽어옵니다.
        if (!string.IsNullOrEmpty(item["MIXB_PASTEUR_STATE"].ToString()))
        {
            i.MIXB_PASTEUR_STATE = item["MIXB_PASTEUR_STATE"].ToString();
        }

        // MIXA_PASTEUR_TEMP가 공백이 아닐 때만 값을 읽어옵니다.
        if (!string.IsNullOrEmpty(item["MIXA_PASTEUR_TEMP"].ToString()))
        {
            double mixAPasteurTemp;
            if (double.TryParse(item["MIXA_PASTEUR_TEMP"].ToString(), out
                mixAPasteurTemp) && mixAPasteurTemp < 5000)
            {
                i.MIXA_PASTEUR_TEMP = mixAPasteurTemp.ToString();
            }
        }
    }
}
```

DBHelper의 select문 사용



## DataManager – Load()

```
if (!string.IsNullOrEmpty(item["MIXB_PASTEUR_TEMP"].ToString()))
{
    double mixBPasteurTemp;
    if (double.TryParse(item["MIXB_PASTEUR_TEMP"].ToString(), out
        mixBPasteurTemp) && mixBPasteurTemp < 5000)
    {
        i.MIXB_PASTEUR_TEMP = mixBPasteurTemp.ToString();
    }
}

// INSP 공백 처리
i.INSP = item["INSP"].ToString();

// 모든 속성이 공백이 아니고, 입력된 숫자가 5000 미만인 경우에만 추가합니다.
if (!string.IsNullOrEmpty(i.STD_DT.ToString()) &&
    !string.IsNullOrEmpty(i.MIXA_PASTEUR_STATE) &&
    !string.IsNullOrEmpty(i.MIXB_PASTEUR_STATE) &&
    !string.IsNullOrEmpty(i.MIXA_PASTEUR_TEMP) &&
    !string.IsNullOrEmpty(i.MIXB_PASTEUR_TEMP) &&
    !string.IsNullOrEmpty(i.INSP))
{
    Instance.Add(i);
}
}
}
catch (Exception ex)
{
    System.Windows.Forms.MessageBox.Show(ex.Message);
    System.Windows.Forms.MessageBox.Show(ex.StackTrace);
}
}
```

```
public static void selectQuery()  
{  
    try  
    {  
        connectDB();  
        SqlCommand cmd = new SqlCommand();  
        cmd.Connection = conn;  
        cmd.CommandText = "select * from" + TABLENAME;  
        da = new SqlDataAdapter(cmd);  
        ds = new DataSet();  
        da.Fill(ds, TABLENAME);  
        dt = ds.Tables[0];  
    }  
    catch (Exception ex)  
    {  
        DataManager.printLog("select" + ex.StackTrace);  
    }  
    finally  
    {  
        conn.Close();  
    }  
}
```

## 전체 데이터 조회



```
private void AllPrint_Click(object sender, EventArgs e)
{
    DataManager.Load();
    dataGridView1.DataSource = null;
    dataGridView1.DataSource = DataManager.Instance;

    var allData = DataManager.Instance;
    UpdateChart(allData);
}
```

```
private void UpdateChart(List<Pasteurizer> data)
{
```

```
    chart1.Series.Clear();
    Series series = new Series("Data");
    series.ChartType = SeriesChartType.Pie;
```

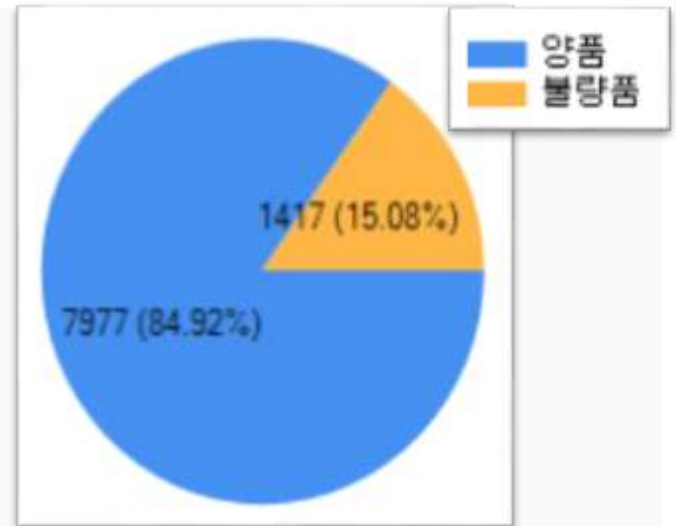
```
    int okCount = data.Count(p => p.INSP == "OK");
    int ngCount = data.Count(p => p.INSP == "NG");
    int totalCount = okCount + ngCount;
```

```
    series.Points.AddXY("양품", okCount);
    series.Points.AddXY("불량품", ngCount);
```

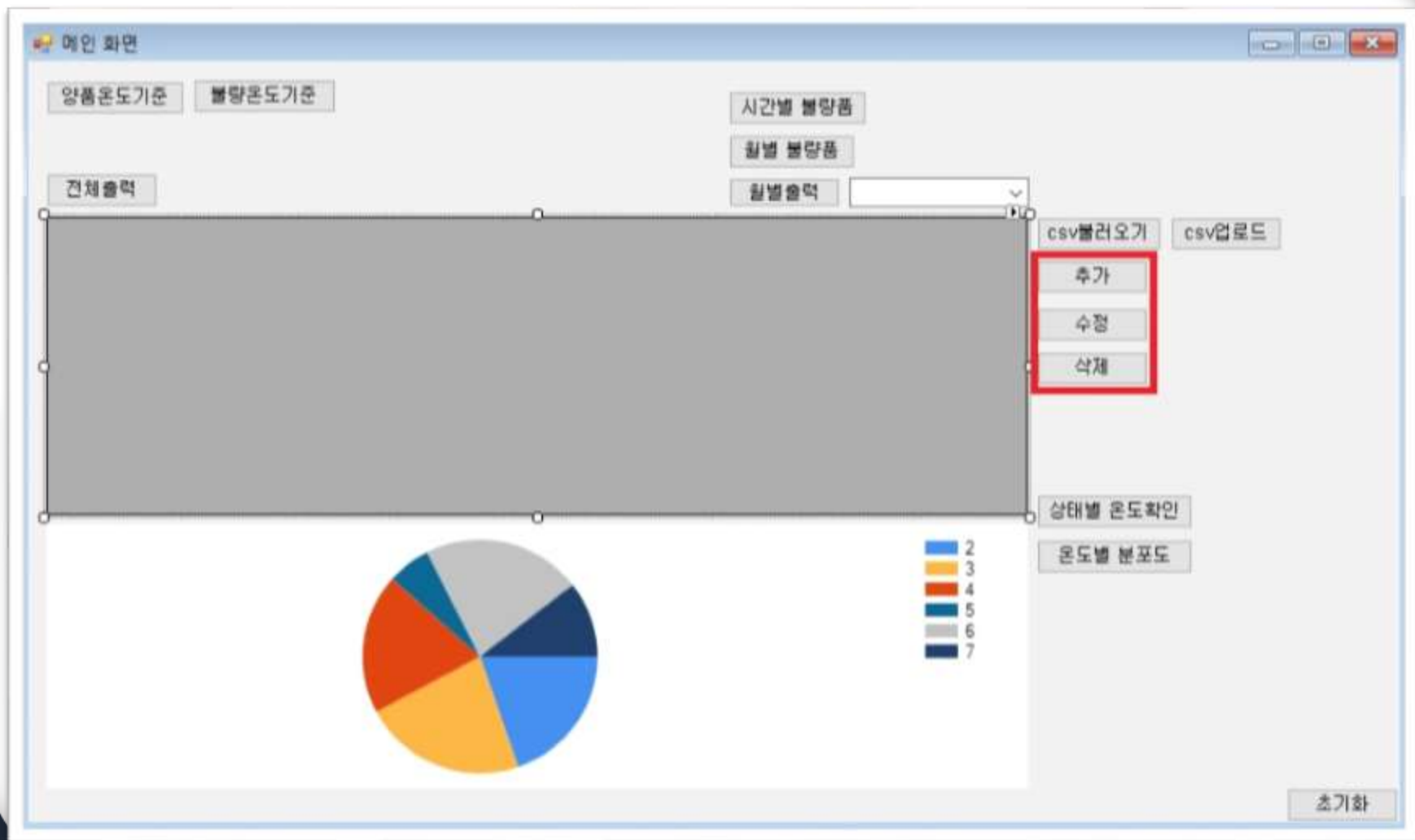
```
    series.Points[0].Label = $"{okCount} ({((double)okCount / totalCount * 100):F2}%)";
    series.Points[1].Label = $"{ngCount} ({((double)ngCount / totalCount * 100):F2}%)";
    series.Points[0].LegendText = "양품";
    series.Points[1].LegendText = "불량품";
```

```
    chart1.Series.Add(series);
```

```
}
```



## 데이터 추가/수정/삭제



## 추가

```
private void insert_Click(object sender, EventArgs e)
{
    Insert form3 = new Insert();
    form3.ShowDialog();
}
```

```
public partial class Insert : Form
{
    public Insert()
    {
        InitializeComponent();
    }

    private void modinsert_Click(object sender, EventArgs e)
    {
        DBHelper.InsertPasteurizerData(textBox1.Text, textBox2.Text, textBox3.Text,
        textBox4.Text, textBox5.Text, comboBox1.Text);
        Close();
    }
}
```

추가

STD_DT	2024-01-03 YYYY-MM-DD HH:mm
MIXA_PASTEUR_STATE	1
MIXB_PASTEUR_STATE	1
MIXA_PASTEUR_TEMP	294
MIXB_PASTEUR_TEMP	352
INSP	OK

추가



## DB Helper – Insert 문

```
public static void InsertPasteurizerData(string param1, string param2 = null, string
param3 = null, string param4 = null, string param5 = null, string param6 = null)
{
    string sqlcmd = "";
    sqlcmd = "INSERT INTO pasteurizer " +
        "(STD_DT, MIXA_PASTEUR_STATE, MIXB_PASTEUR_STATE, MIXA_PASTEUR_TEMP,
MIXB_PASTEUR_TEMP, INSP) " +
        "VALUES (@stdDt, @mixAState, @mixBState, @mixATemp, @mixBTemp, @insp)";
    try
    {
        connectDB();
        SqlCommand command = new SqlCommand();
        command.Connection = conn;
        command.Parameters.AddWithValue("@stdDt", param1);
        command.Parameters.AddWithValue("@mixAState", param2);
        command.Parameters.AddWithValue("@mixBState", param3);
        command.Parameters.AddWithValue("@mixATemp", param4);
        command.Parameters.AddWithValue("@mixBTemp", param5);
        command.Parameters.AddWithValue("@insp", param6);
        command.CommandText = sqlcmd;
        command.ExecuteNonQuery();
    }
    catch (Exception ex)
    {
        DataManager.printLog(ex.StackTrace);
        MessageBox.Show("데이터베이스 업데이트 중 오류 발생: " + ex.Message);
    }
    finally
    {
        conn.Close();
        MessageBox.Show("추가 되었습니다.");
    }
}
```

## 추가 결과

전체 출력

STD_DT
2021-01-01 오전 8:30
2022-01-01 오전 8:30
2024-01-01
2024-01-01

추가

STD\_DT

2024-01-03

YYYY-MM-DD HH:mm

MIXA\_PASTEUR\_STATE

1

MIXB\_PASTEUR\_STATE

1

MIXA\_PASTEUR\_TEMP

294

MIXB\_PASTEUR\_TEMP

352

INSP

OK

추가

2024-01-03

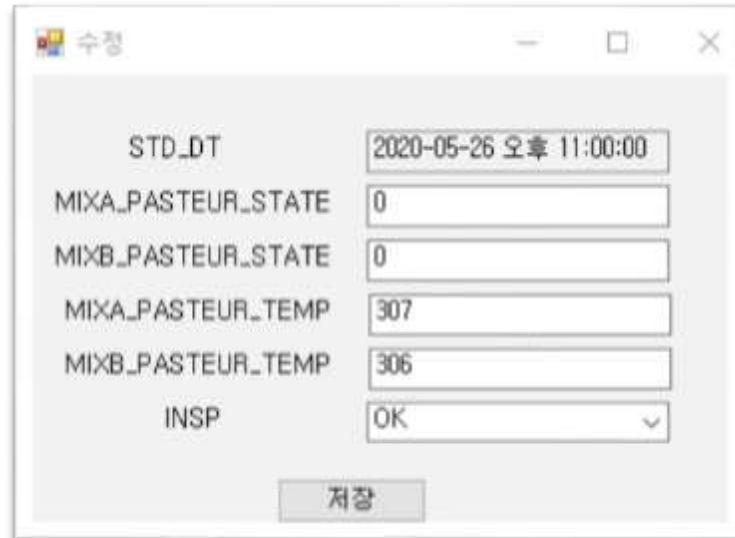
1

1

294



## 수정



STD_DT	2020-05-26 오후 11:00:00
MIXA_PASTEUR_STATE	0
MIXB_PASTEUR_STATE	0
MIXA_PASTEUR_TEMP	307
MIXB_PASTEUR_TEMP	306
INSP	OK

저장

```
private void modify_Click(object sender, EventArgs e)
{
    if (selectedCellValues != null && selectedCellValues.Length >= 5)
    {
        Modify form2 = new Modify(selectedCellValues);
        form2.DataUpdated += Form2_DataUpdated; // 이벤트 핸들러 추가
        form2.ShowDialog(); // 모달 창으로 열기
    }
    else
    {
        MessageBox.Show("먼저 DataGridView에서 셀을 클릭해주세요.");
    }
}
```

```

public partial class Modify : Form
{
    private string[] originalCellValues; // 원본
    public event Action<string[]> DataUpdated;

    public Modify(string[] cellValues)
    {
        InitializeComponent();
        // TextBox에 선택한 셀의 데이터 출력
        if (cellValues.Length >= 6)
        {
            originalCellValues = cellValues.Clone() as string[]; // 원본 데이터 복사
            textBox1.Text = cellValues[0];
            textBox2.Text = cellValues[1];
            textBox3.Text = cellValues[2];
            textBox4.Text = cellValues[3];
            textBox5.Text = cellValues[4];
            comboBox1.Text = cellValues[5]; // 추가된 텍스트박스에 데이터 설정

            textBox1.ReadOnly = true; // 첫 번째 텍스트박스를 읽기 전용으로 설정
        }

        modsave.Click += modsave_Click;
    }
}

```

STD_DT	2020-05-26 오후 11:00:00
MIXA_PASTEUR_STATE	0
MIXB_PASTEUR_STATE	0
MIXA_PASTEUR_TEMP	307
MIXB_PASTEUR_TEMP	306
INSP	OK

저장

## 데이터 그리드 뷰 셀 클릭

	STD_DT	MIXA_PASTEUR_STATE	MIXB_PASTEUR_STATE	MIXA_PASTEUR_TEMP ^
▶	2020-04-22 오전 1:00	0	0	294
	2020-04-22 오전 1:00	0	0	294
	2020-04-22 오전 1:30	0	0	294
	2020-04-22 오전 1:30	0	0	294
	2020-04-22 오전 2:00	0	0	294
	2020-04-22 오전 2:00	0	0	294
	2020-04-22 오전 2:30	0	0	294

```
private void dataGridView1_CellClick(object sender, DataGridViewCellEventArgs e)
{
    if (e.RowIndex >= 0 && e.ColumnIndex >= 0)
    {
        DataGridViewRow selectedRow = dataGridView1.Rows[e.RowIndex];
        selectedCellValues = new string[6];
        selectedCellValues[0] = selectedRow.Cells[0].Value?.ToString();
        selectedCellValues[1] = selectedRow.Cells[1].Value?.ToString();
        selectedCellValues[2] = selectedRow.Cells[2].Value?.ToString();
        selectedCellValues[3] = selectedRow.Cells[3].Value?.ToString();
        selectedCellValues[4] = selectedRow.Cells[4].Value?.ToString();
        selectedCellValues[5] = selectedRow.Cells[5].Value?.ToString();
    }
}
```

```
private void modsave_Click(object sender, EventArgs e)
{
    // 입력 데이터 검증
    if (ValidateInputs())
    {
        // 수정된 데이터를 배열에 저장
        string[] updatedValues = new string[6];
        updatedValues[0] = textBox1.Text;
        updatedValues[1] = textBox2.Text;
        updatedValues[2] = textBox3.Text;
        updatedValues[3] = textBox4.Text;
        updatedValues[4] = textBox5.Text;
        updatedValues[5] = comboBox1.Text;

        // 데이터 업데이트 이벤트 호출
        DataUpdated?.Invoke(updatedValues);

        // 데이터베이스 업데이트
        DBHelper.UpdatePasteurizerData(updatedValues);

        // 폼 닫기
        this.Close();
    }
    else
    {
        MessageBox.Show("모든 필드를 올바르게 입력하세요.", "입력 오류", MessageBoxButtons.OK,
        MessageBoxIcon.Warning);
    }
}
```

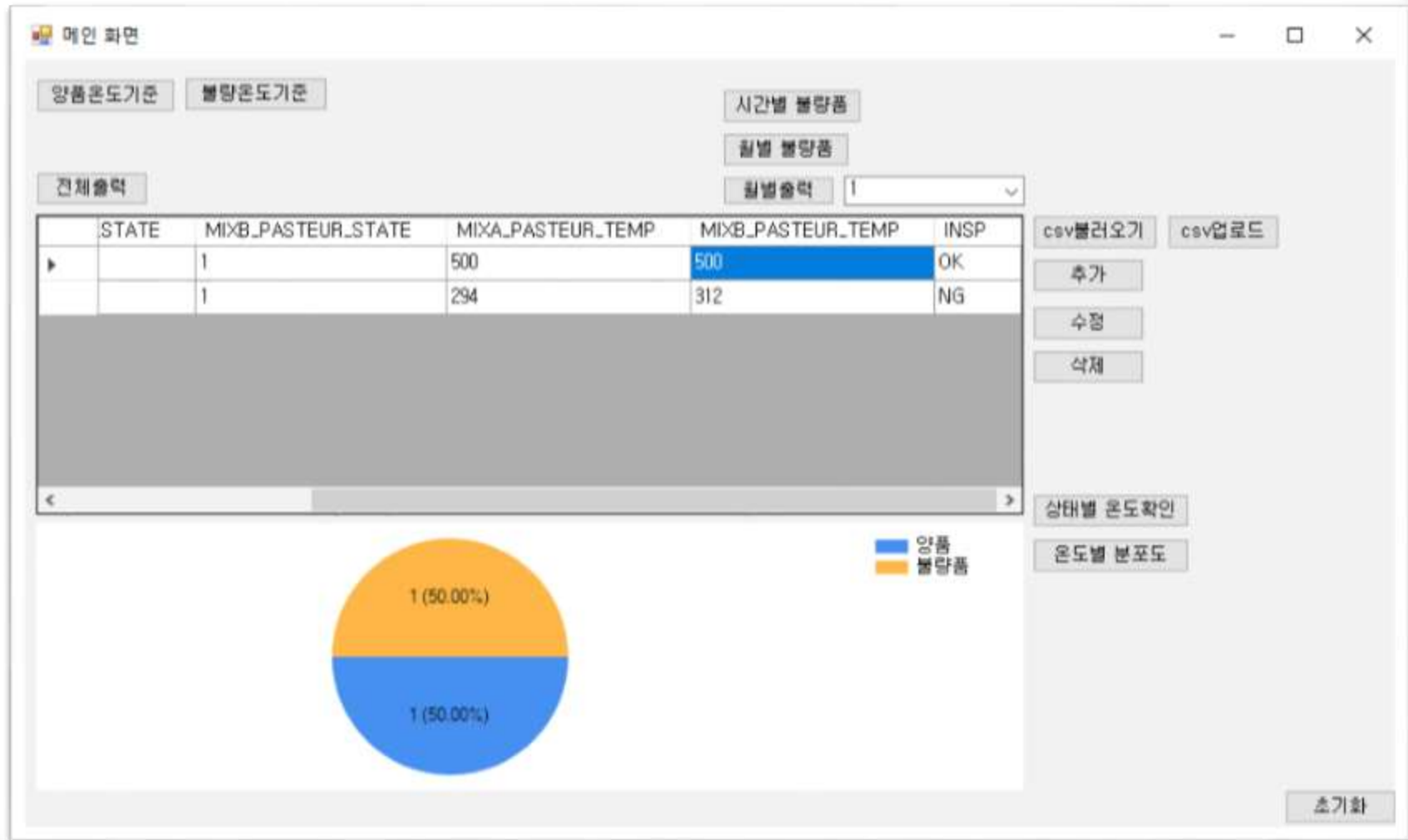
```

public static void UpdatePasteurizerData(string[] updatedValues)
{
    try
    {
        connectDB();
        string query = "UPDATE pasteurizer SET MIXA_PASTEUR_STATE = @mixAState,
MIXB_PASTEUR_STATE = @mixBState, " +
                        "MIXA_PASTEUR_TEMP = @mixATemp, MIXB_PASTEUR_TEMP =
@mixBTemp, " +
                        "INSP = @insp WHERE STD_DT = @stdDt";
        SqlCommand cmd = new SqlCommand(query, conn);
        cmd.Parameters.AddWithValue("@stdDt", DateTime.Parse(updatedValues[0]));
        cmd.Parameters.AddWithValue("@mixAState", updatedValues[1]);
        cmd.Parameters.AddWithValue("@mixBState", updatedValues[2]);
        cmd.Parameters.AddWithValue("@mixATemp", double.Parse(updatedValues[3]));
        cmd.Parameters.AddWithValue("@mixBTemp", double.Parse(updatedValues[4]));
        cmd.Parameters.AddWithValue("@insp", updatedValues[5]);
        cmd.ExecuteNonQuery();
    }
    catch (Exception ex)
    {
        MessageBox.Show("데이터베이스 업데이트 중 오류 발생: " + ex.Message);
    }
    finally
    {
        conn.Close();
    }
}

```



# 수정 결과



```
private void delete_Click(object sender, EventArgs e)
{
    if (selectedCellValues != null && selectedCellValues.Length >= 5)
    {
        // 메시지 박스를 표시하고 사용자의 응답을 확인합니다.
        DialogResult result = MessageBox.Show("정말 삭제하시겠습니까?", "확인",
        MessageBoxButtons.YesNo);

        if (result == DialogResult.Yes)
        {
            DBHelper.DeletePasteurizerData(selectedCellValues[0]);
        }
    }
    else
    {
        MessageBox.Show("먼저 DataGridView에서 셀을 클릭해주세요.");
    }
}
```

```
public static void DeletePasteurizerData(string stdDt)
{
    try
    {
        connectDB();
        string query = "DELETE FROM pasteurizer WHERE STD_DT = @stdDt";
        SqlCommand cmd = new SqlCommand(query, conn);
        cmd.Parameters.AddWithValue("@stdDt", DateTime.Parse(stdDt).ToString("yyyy-
MM-dd HH:mm"));
        cmd.ExecuteNonQuery();
    }
    catch (Exception ex)
    {
        MessageBox.Show("데이터베이스 삭제 중 오류 발생: " + ex.Message);
    }
    finally
    {
        conn.Close();
    }
}
```



# 내용

STD_DT	MIXA_PASTEUR_STATE	MIXB_PASTEUR_STATE	MIXA_PASTEUR_TEMP
▶ 2024-02-01 오전 10:00	1	1	357
2024-02-12 오전 8:30	0	0	294

확인

✕

정말 삭제하시겠습니까?

예(Y)

아니요(N)

건출출력

필출출력

2

STD_DT	MIXA_PASTEUR_STATE	MIXB_PASTEUR_STATE	MIXA_PASTEUR_TEMP
▶ 2024-02-12 오전 8:30	0	0	294

# CSVHandler

```
public static async Task InsertDataFromCsvAsync(string filePath)
{
    //시버 이쁜 변경 필요
    string connectionString = "Data Source=localhost;Initial Catalog=pasteurizer;Integrated Security=True;";
    string[] lines = File.ReadAllLines(filePath);

    using (SqlConnection conn = new SqlConnection(connectionString))
    {
        await conn.OpenAsync();

        foreach (var line in lines)
        {
            var values = line.Split(',');

            try
            {
                string query = "INSERT INTO pasteurizer (STD_DT, MIXA_PASTEUR_STATE, MIXB_PASTEUR_STATE, MIXA_PASTEUR_TEMP, MIXB_PASTEUR_TEMP, INSP) " +
                                "VALUES (@STD_DT, @MIXA_PASTEUR_STATE, @MIXB_PASTEUR_STATE, @MIXA_PASTEUR_TEMP, @MIXB_PASTEUR_TEMP, @INSP)";

                using (SqlCommand cmd = new SqlCommand(query, conn))
                {
                    cmd.Parameters.AddWithValue("@STD_DT", DateTime.Parse(values[0]));
                    cmd.Parameters.AddWithValue("@MIXA_PASTEUR_STATE", values[1]);
                    cmd.Parameters.AddWithValue("@MIXB_PASTEUR_STATE", values[2]);
                    cmd.Parameters.AddWithValue("@MIXA_PASTEUR_TEMP", values[3]);
                    cmd.Parameters.AddWithValue("@MIXB_PASTEUR_TEMP", values[4]);
                    cmd.Parameters.AddWithValue("@INSP", values[5]);

                    await cmd.ExecuteNonQuery();
                }
            }
            catch (Exception ex)
            {
                Console.WriteLine($"Error processing line: {line}. Error: {ex.Message}");
            }
        }
    }
}
```

```
public DataTable GetData(string tableName)
{
    DataTable dataTable = new DataTable();
    try
    {
        using (SqlConnection connection = new SqlConnection(_connectionString))
        {
            connection.Open();
            string query = $"SELECT * FROM {tableName}";
            SqlDataAdapter adapter = new SqlDataAdapter(query, connection);
            adapter.Fill(dataTable);
        }
    }
    catch (Exception ex)
    {
        DataManager.printLog("select" + ex.StackTrace);
        throw;
    }

    return dataTable;
}
```

## CSV 업로드

```
private async void csv_up_Click(object sender, EventArgs e)
{
    OpenFileDialog openFileDialog = new OpenFileDialog();
    openFileDialog.Filter = "CSV Files (*.csv)|*.csv|All Files (*.*)|*.*";
    if (openFileDialog.ShowDialog() == DialogResult.OK)
    {
        string filePath = openFileDialog.FileName;
        try
        {
            await CsvDataHandler.InsertDataFromCsvAsync(filePath);
            MessageBox.Show("업로드가 완료되었습니다.", "성공", MessageBoxButtons.OK,
                MessageBoxIcon.Information);
        }
        catch (Exception ex)
        {
            MessageBox.Show($"파일 업로드 중 오류가 발생했습니다:\n{ex.Message}", "오류",
                MessageBoxButtons.OK, MessageBoxIcon.Error);
        }
    }
}
```

## 결과

### CSV파일

A	B	C	D	E	F
STD_DT	MIXA_PAS	MIXB_PAS	MIXA_PAS	MIXB_PAS	INSP
2024-01-01 0:00	1	1	300	300	NG

### 데이터그리드뷰

2024-01-01	1	1	300
2024-01-01	1	1	300

## 시간별 불량 확인

```
private void hour_Click(object sender, EventArgs e)
{
    Time_Bad form5 = new Time_Bad();
    form5.ShowDialog();
}
```

```
private void LoadChartData()
{
    // DataManager.Instance에서 필요한 데이터를 가져와서 처리
    var data = DataManager.Instance;

    // 시간대별 불량을 계산
    Dictionary<int, double> hourlyDefectRate = new Dictionary<int, double>();

    // 모든 날짜의 해당 시간 합산
    foreach (var record in data)
    {
        int hour = record.STD_DT.Hour;
        double ngCount = record.INSP == "NG" ? 1 : 0;

        if (hourlyDefectRate.ContainsKey(hour))
        {
            hourlyDefectRate[hour] += ngCount;
        }
        else
        {
            hourlyDefectRate[hour] = ngCount;
        }
    }
}
```

## 시간별 불량 확인

```
// 시간대별 평균 불량을 계산
foreach (var hour in hourlyDefectRate.Keys.ToList())
{
    double totalCount = data.Count(p => p.STD_DT.Hour == hour);
    hourlyDefectRate[hour] = (hourlyDefectRate[hour] / totalCount) * 100; // 백분
    //로 변환
}

// 차트 설정
chart1.Series.Clear(); // 기존 Series를 Clear합니다.

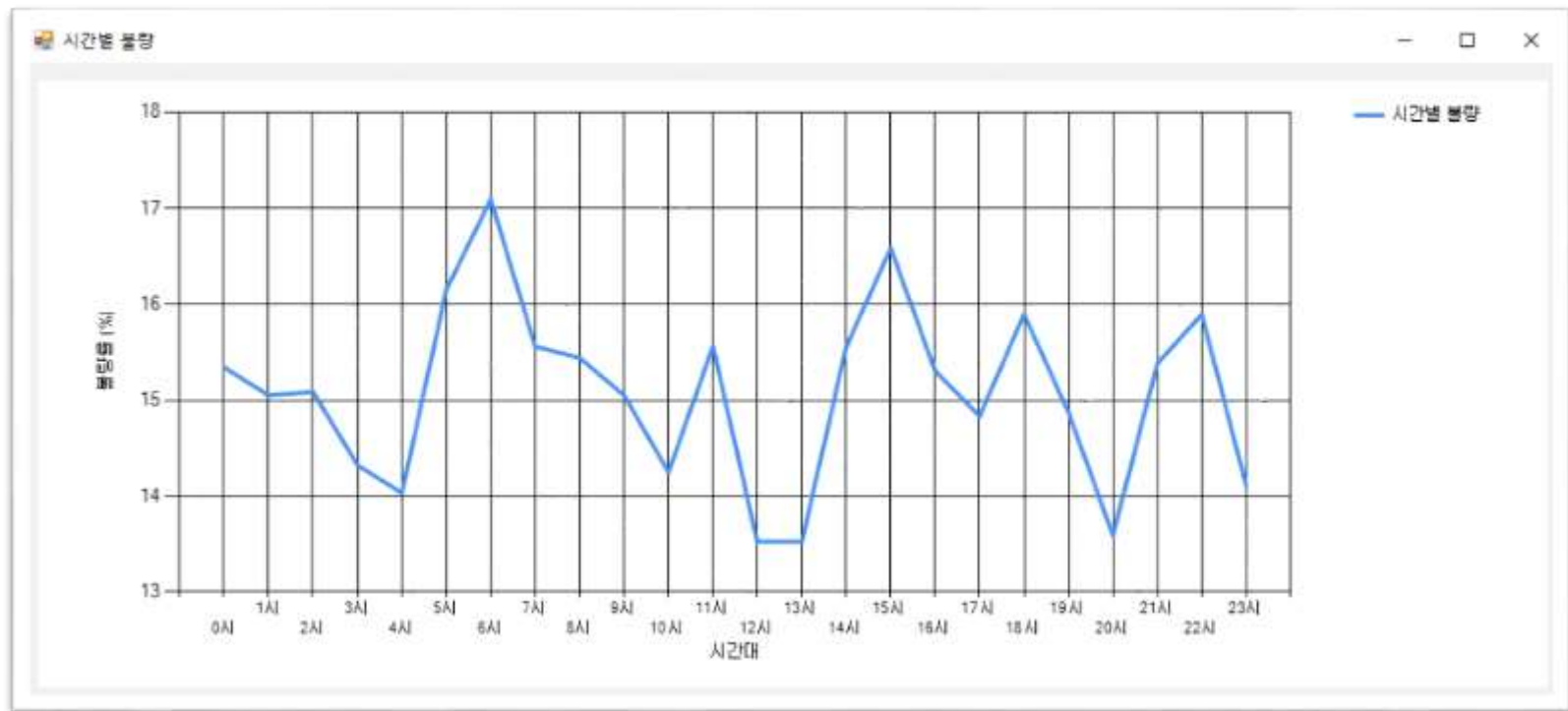
Series series = new Series("시간별 불량")
{
    ChartType = SeriesChartType.Line, // 선 그래프로 설정합니다.
    BorderWidth = 3 // 선의 두께를 설정합니다. 여기서는 3픽셀로 설정했습니다.
};

// 데이터 바인딩
foreach (var item in hourlyDefectRate.OrderBy(kvp => kvp.Key))
{
    series.Points.AddXY($"{item.Key}시", item.Value);
}

chart1.Series.Add(series);
chart1.ChartAreas[0].AxisX.Interval = 1;
chart1.ChartAreas[0].AxisX.Title = "시간대";
chart1.ChartAreas[0].AxisY.Title = "불량률 (%)";
chart1.ChartAreas[0].AxisY.Minimum = 13;
chart1.ChartAreas[0].AxisY.Maximum = 18; // Y축의 최대값을 100으로 설정하여 백분율을
// 표시합니다.

// 차트 다시 그리기
chart1.Invalidate();
}
```

## 결과 화면

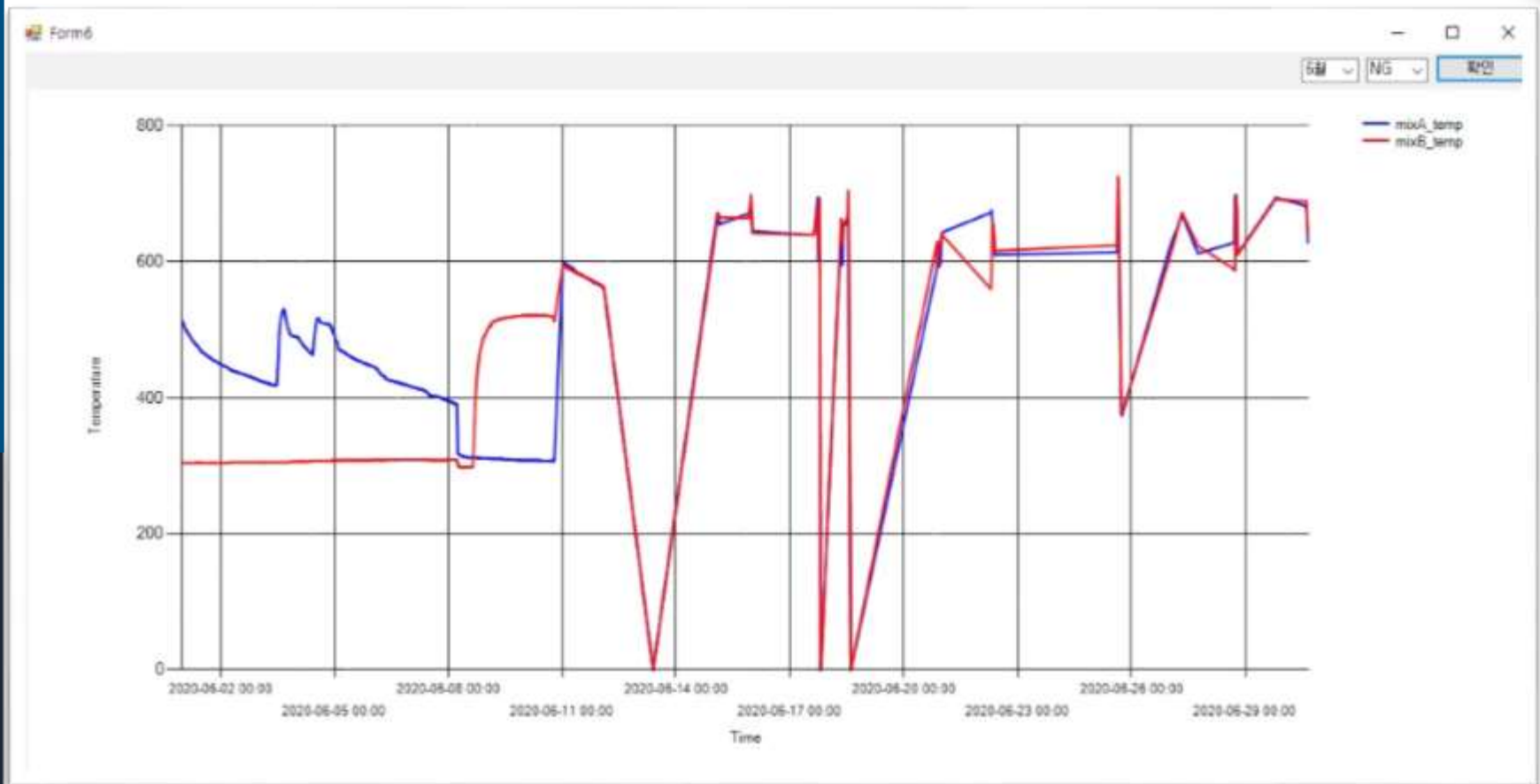




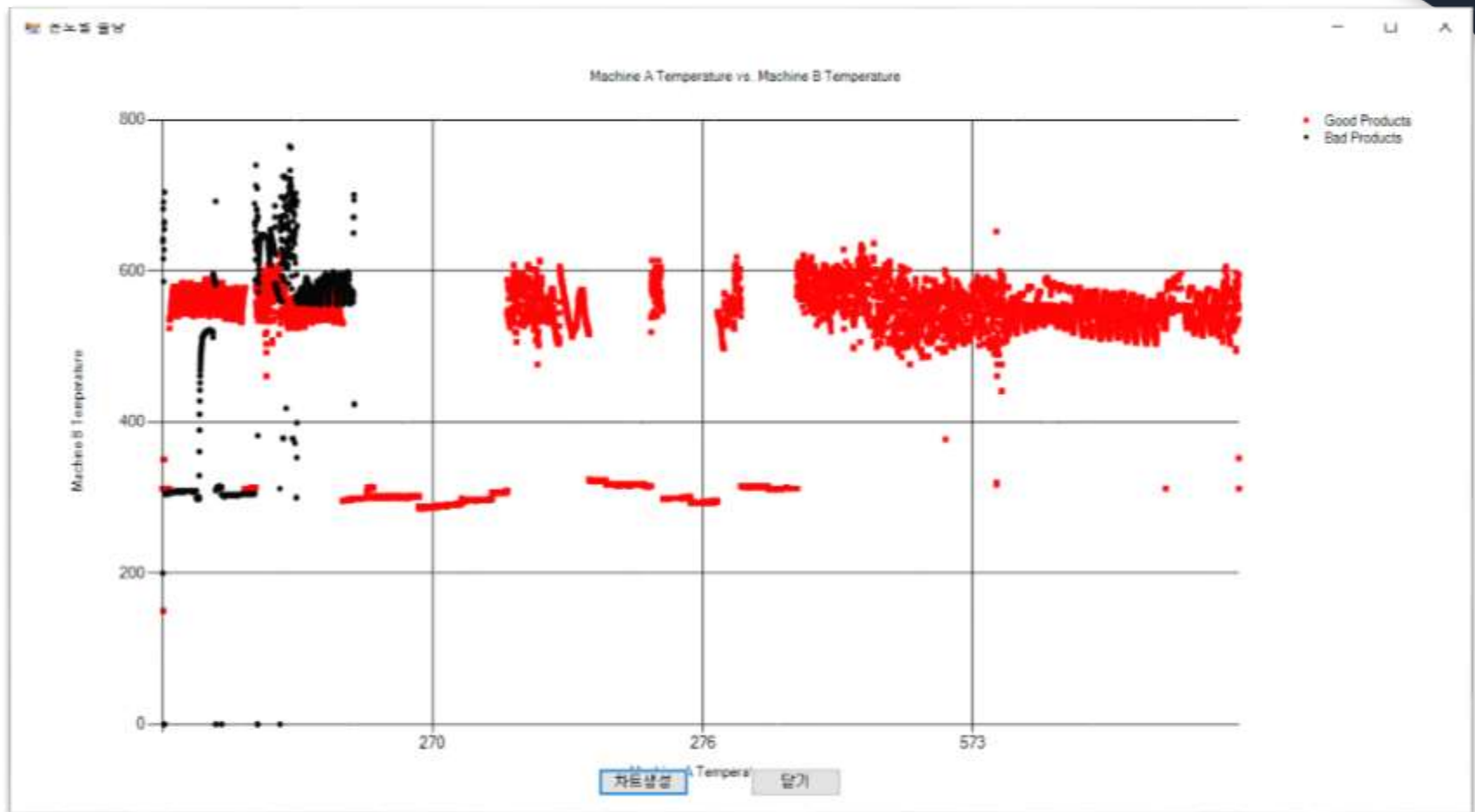
# 양품 온도



# 불량품 온도



## 결과 화면



## 전체 양품/불량 분포도

```
private List<Pasteurizer> _data;  
public temperature(List<Pasteurizer> data)  
{  
    InitializeComponent();  
    _data = data;  
  
    Chart1.Click += showChart_Click;  
}
```

```
private void CreateChart(List<Pasteurizer> data)  
{  
    // 차트 플 생성  
    var chart = new Chart { Dock = DockStyle.Fill };  
  
    // 차트 영역 설정  
    var chartArea = new ChartArea();  
    chart.ChartAreas.Add(chartArea);  
  
    // 양품 데이터 시리즈 생성  
    var goodSeries = new Series  
    {  
        Name = "Good Products",  
        ChartType = SeriesChartType.Point,  
        Color = System.Drawing.Color.Red  
    };  
  
    // 불량 데이터 시리즈 생성  
    var badSeries = new Series  
    {  
        Name = "Bad Products",  
        ChartType = SeriesChartType.Point,  
        Color = System.Drawing.Color.Black  
    };  
}
```

## 전체 양품/불량 분포도

```
// 데이터를 양품과 불량으로 분리하여 추가
foreach (var record in data)
{
    if (record.INSP == "OK")
    {
        goodSeries.Points.AddXY(record.MIXA_PASTEUR_TEMP, record.MIXB_PASTEUR_TEMP);
    }
    else if (record.INSP == "NG")
    {
        badSeries.Points.AddXY(record.MIXA_PASTEUR_TEMP, record.MIXB_PASTEUR_TEMP);
    }
}

chart.Series.Add(goodSeries);
chart.Series.Add(badSeries);

// 차트 제목 설정
chart.Titles.Add("Machine A Temperature vs. Machine B Temperature");

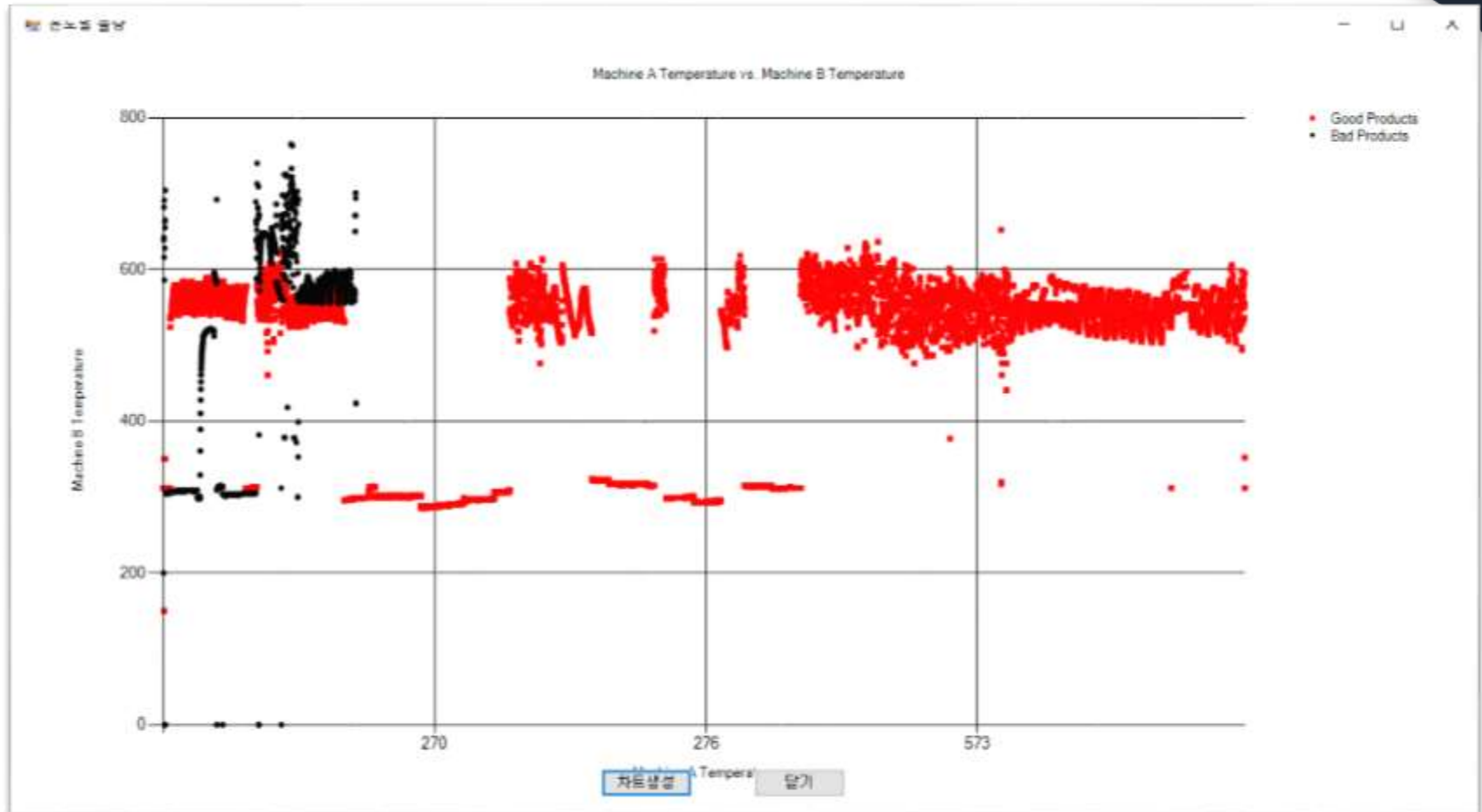
// 축 라벨 설정
chartArea.AxisX.Title = "Machine A Temperature";
chartArea.AxisY.Title = "Machine B Temperature";

// 범례 추가
chart.Legends.Add(new Legend("Legend"));

Controls.Add(chart);
}
```

```
private void showChart_Click(object sender, EventArgs e)
{
    // 차트를 표시하기 전에 기존 차트를 제거하여 중복 생성 방지
    foreach (var control in this.Controls.OfType<Chart>().ToList())
    {
        this.Controls.Remove(control);
        control.Dispose();
    }
    // 새로운 차트 생성
    CreateChart(_data);
}
```

## 결과 화면



살균기 A의 온도가 27도 이상이고 살균기 B온도가 50~60도 사이에 있을 때  
양품이 가장 많이 생산됨을 알 수 있다.

즉 불량률 가장 줄이기 위해서는 위와 같은 온도를 유지하는 것이 좋다.

조민석

팀장으로  
원할한 소통이  
중요하다는  
점을 배웠다.

한형빈

결과를 내는  
것만큼 결과를  
분석하고  
해석하는 것도  
중요하다는 것  
을 알게 된 것  
같습니다.

장순재

새로운 분야라  
처음엔 잘  
몰랐지만  
프로젝트를  
하며 많이  
배웠다.

김동현

프로젝트를 할  
때 힘들었는데  
끝나고 나니  
뿌듯합니다.

감사합니다.