WARRANTYCONTEXT.CS

public class WarrantyContext : DbContext

{

private readonly IConfiguration \_configuration;

public WarrantyContext()

{

var builder = new ConfigurationBuilder()

.SetBasePath(Directory.GetCurrentDirectory())

.AddJsonFile("appsettings.json");

\_configuration = builder.Build();

}

public DbSet<PurchaseItem> PurchaseItems { get; set; }

protected override void OnConfiguring(DbContextOptionsBuilder optionsBuilder)

{

if (!optionsBuilder.IsConfigured)

{

var connectionString = \_configuration.GetConnectionString("DefaultConnection");

var serverVersion = ServerVersion.AutoDetect(connectionString);

optionsBuilder

.UseMySql(connectionString, serverVersion)

.LogTo(Console.WriteLine)

.EnableSensitiveDataLogging()

.EnableDetailedErrors();

}

}

}

MAIN.CS

using System;

using System.Windows.Forms;

using System.Drawing;

using System.ComponentModel;

using WarrantyTracker.Models;

using WarrantyTracker.Data;

using var context = new WarrantyTrackerContext();

namespace WarrantyTracker.Forms

{

public partial class MainForm : Form

{

// Form controls

private readonly TableLayoutPanel mainLayout = new();

private readonly Panel headerPanel = new();

private readonly Label headerLabel = new();

private readonly TextBox txtItemName = new();

private readonly TextBox txtSerialNumber = new();

private readonly DateTimePicker dtpPurchaseDate = new();

private readonly DateTimePicker dtpWarrantyExpiration = new();

private readonly ComboBox cboWarrantyPlan = new();

private readonly Label lblWarrantyPrice = new();

private readonly TextBox txtPurchasePrice = new();

private readonly TextBox txtMaintenanceNotes = new();

private readonly Button btnSubmit = new();

private readonly Label lblValidation = new();

private readonly ErrorProvider errorProvider = new();

public class WarrantyPlan

{

public required string Duration { get; set; }

public decimal Price { get; set; }

public int Days { get; set; }

public override string ToString() => $"{Duration} (${Price:F2})";

}

public MainForm()

{

InitializeComponent();

SetupValidation();

}

private void InitializeWarrantyPlans()

{

if (cboWarrantyPlan == null) return;

var plans = new[]

{

new WarrantyPlan { Duration = "30 Day Warranty", Price = 7.99M, Days = 30 },

new WarrantyPlan { Duration = "60 Day Warranty", Price = 14.99M, Days = 60 },

new WarrantyPlan { Duration = "90 Day Warranty", Price = 24.99M, Days = 90 }

};

cboWarrantyPlan.DataSource = plans;

cboWarrantyPlan.SelectedIndexChanged += WarrantyPlan\_SelectedIndexChanged;

}

private void WarrantyPlan\_SelectedIndexChanged(object? sender, EventArgs e)

{

if (cboWarrantyPlan == null || dtpPurchaseDate == null ||

dtpWarrantyExpiration == null) return;

var selectedPlan = cboWarrantyPlan.SelectedItem as WarrantyPlan;

if (selectedPlan != null)

{

dtpWarrantyExpiration.Value = dtpPurchaseDate.Value.AddDays(selectedPlan.Days);

if (lblWarrantyPrice != null)

lblWarrantyPrice.Text = $"Warranty Price: ${selectedPlan.Price:F2}";

}

}

private void SetupValidation()

{

if (txtItemName != null)

txtItemName.Validating += (s, e) => ValidateControl(txtItemName, e, "Item name is required");

if (txtSerialNumber != null)

txtSerialNumber.Validating += (s, e) => ValidateControl(txtSerialNumber, e, "Serial number is required");

if (txtPurchasePrice != null)

txtPurchasePrice.Validating += ValidatePurchasePrice;

if (dtpWarrantyExpiration != null)

dtpWarrantyExpiration.ValueChanged += ValidateWarrantyDate;

}

private void ValidateControl(Control? control, CancelEventArgs e, string errorMessage)

{

if (control == null || errorProvider == null || lblValidation == null) return;

if (string.IsNullOrWhiteSpace(control.Text))

{

e.Cancel = true;

errorProvider.SetError(control, errorMessage);

lblValidation.Text = errorMessage;

}

else

{

errorProvider.SetError(control, "");

lblValidation.Text = "";

}

}

private void ValidatePurchasePrice(object? sender, CancelEventArgs e)

{

if (txtPurchasePrice == null || errorProvider == null || lblValidation == null) return;

if (!decimal.TryParse(txtPurchasePrice.Text, out decimal price) || price <= 0)

{

e.Cancel = true;

errorProvider.SetError(txtPurchasePrice, "Please enter a valid price greater than 0");

lblValidation.Text = "Please enter a valid price greater than 0";

}

else

{

errorProvider.SetError(txtPurchasePrice, "");

lblValidation.Text = "";

}

}

private void ValidateWarrantyDate(object? sender, EventArgs e)

{

if (dtpWarrantyExpiration == null || dtpPurchaseDate == null ||

errorProvider == null || lblValidation == null) return;

if (dtpWarrantyExpiration.Value < dtpPurchaseDate.Value)

{

errorProvider.SetError(dtpWarrantyExpiration, "Warranty date must be after purchase date");

lblValidation.Text = "Warranty date must be after purchase date";

}

else

{

errorProvider.SetError(dtpWarrantyExpiration, "");

lblValidation.Text = "";

}

}

private void InitializeComponent()

{

if (mainLayout == null || headerPanel == null || headerLabel == null) return;

// Form settings

Text = "Lamar's Warranty Tracking System";

Size = new Size(800, 600);

MinimumSize = new Size(600, 500);

BackColor = Color.FromArgb(240, 248, 255); // AliceBlue

Font = new Font("Aptos", 10F);

Padding = new Padding(10);

// Configure main layout

mainLayout.Dock = DockStyle.Fill;

mainLayout.ColumnCount = 2;

mainLayout.RowCount = 9; // Increased for new controls

mainLayout.Padding = new Padding(20);

mainLayout.BackColor = Color.White;

// Configure header

headerPanel.Dock = DockStyle.Top;

headerPanel.Height = 60;

headerPanel.BackColor = Color.FromArgb(51, 122, 183);

headerLabel.Text = "Warranty Item Registration";

headerLabel.ForeColor = Color.White;

headerLabel.Font = new Font("Segoe UI", 16F, FontStyle.Bold);

headerLabel.AutoSize = true;

headerLabel.Location = new Point(20, 15);

if (headerPanel.Controls != null)

headerPanel.Controls.Add(headerLabel);

InitializeControls();

AddControlsToLayout();

if (Controls != null)

{

Controls.Add(mainLayout);

Controls.Add(headerPanel);

}

if (mainLayout != null)

mainLayout.Top = headerPanel.Height;

}

private void InitializeControls()

{

if (txtItemName == null || txtSerialNumber == null || dtpPurchaseDate == null ||

dtpWarrantyExpiration == null || txtPurchasePrice == null ||

txtMaintenanceNotes == null || btnSubmit == null || cboWarrantyPlan == null) return;

ConfigureTextBox(txtItemName, "Enter item name");

ConfigureTextBox(txtSerialNumber, "Enter serial number");

ConfigureDatePicker(dtpPurchaseDate);

ConfigureDatePicker(dtpWarrantyExpiration);

// Configure warranty plan dropdown

cboWarrantyPlan.Width = 300;

cboWarrantyPlan.Font = new Font("Segoe UI", 10F);

cboWarrantyPlan.DropDownStyle = ComboBoxStyle.DropDownList;

if (lblWarrantyPrice != null)

{

lblWarrantyPrice.AutoSize = true;

lblWarrantyPrice.Font = new Font("Segoe UI", 10F);

lblWarrantyPrice.ForeColor = Color.FromArgb(51, 122, 183);

}

InitializeWarrantyPlans();

ConfigureTextBox(txtPurchasePrice, "Enter price");

ConfigureTextBox(txtMaintenanceNotes, "Enter maintenance notes");

txtMaintenanceNotes.Multiline = true;

txtMaintenanceNotes.Height = 100;

ConfigureSubmitButton();

}

private void ConfigureTextBox(TextBox textBox, string placeholder)

{

textBox.Width = 200;

textBox.Font = new Font("Segoe UI", 10F);

textBox.BorderStyle = BorderStyle.FixedSingle;

textBox.Padding = new Padding(5);

}

private void ConfigureDatePicker(DateTimePicker picker)

{

picker.Format = DateTimePickerFormat.Short;

picker.Width = 200;

picker.Font = new Font("Segoe UI", 10F);

}

private void ConfigureSubmitButton()

{

if (btnSubmit == null) return;

btnSubmit.Text = "Submit";

btnSubmit.Width = 120;

btnSubmit.Height = 40;

btnSubmit.BackColor = Color.FromArgb(51, 122, 183);

btnSubmit.ForeColor = Color.White;

btnSubmit.Font = new Font("Segoe UI", 10F, FontStyle.Bold);

btnSubmit.FlatStyle = FlatStyle.Flat;

btnSubmit.Click += btnSubmit\_Click;

}

private void AddControlsToLayout()

{

if (mainLayout == null) return;

mainLayout.ColumnStyles.Add(new ColumnStyle(SizeType.Percent, 30F));

mainLayout.ColumnStyles.Add(new ColumnStyle(SizeType.Percent, 70F));

AddFormRow("Item Name:", txtItemName, 0);

AddFormRow("Serial Number:", txtSerialNumber, 1);

AddFormRow("Purchase Date:", dtpPurchaseDate, 2);

AddFormRow("Warranty Expiration:", dtpWarrantyExpiration, 3);

AddFormRow("Warranty Plan:", cboWarrantyPlan, 4);

if (mainLayout.Controls != null && lblWarrantyPrice != null)

{

mainLayout.Controls.Add(lblWarrantyPrice, 1, 5);

}

AddFormRow("Purchase Price:", txtPurchasePrice, 6);

AddFormRow("Maintenance Notes:", txtMaintenanceNotes, 7);

var buttonPanel = new Panel { Dock = DockStyle.Fill };

if (btnSubmit != null && buttonPanel.Controls != null)

buttonPanel.Controls.Add(btnSubmit);

if (mainLayout.Controls != null)

{

mainLayout.Controls.Add(buttonPanel, 1, 8);

if (lblValidation != null)

mainLayout.Controls.Add(lblValidation, 1, 9);

}

}

private void AddFormRow(string labelText, Control? control, int row)

{

if (control == null || mainLayout == null || mainLayout.Controls == null) return;

var label = new Label

{

Text = labelText,

AutoSize = true,

Anchor = AnchorStyles.Left | AnchorStyles.Top,

Padding = new Padding(5)

};

mainLayout.Controls.Add(label, 0, row);

mainLayout.Controls.Add(control, 1, row);

}

private bool ValidateInput()

{

if (txtItemName == null || txtSerialNumber == null || txtPurchasePrice == null ||

dtpWarrantyExpiration == null || dtpPurchaseDate == null) return false;

if (string.IsNullOrWhiteSpace(txtItemName.Text))

{

ShowError("Item name is required.");

return false;

}

if (string.IsNullOrWhiteSpace(txtSerialNumber.Text))

{

ShowError("Serial number is required.");

return false;

}

if (!decimal.TryParse(txtPurchasePrice.Text, out decimal price) || price <= 0)

{

ShowError("Please enter a valid purchase price greater than 0.");

return false;

}

if (dtpWarrantyExpiration.Value < dtpPurchaseDate.Value)

{

ShowError("Warranty expiration date cannot be earlier than purchase date.");

return false;

}

if (dtpPurchaseDate.Value > DateTime.Now)

{

ShowError("Purchase date cannot be in the future.");

return false;

}

if (lblValidation != null)

lblValidation.Text = "";

return true;

}

private async void btnSubmit\_Click(object? sender, EventArgs e)

{

if (ValidateInput())

{

try

{

using var context = new WarrantyContext();

var selectedPlan = cboWarrantyPlan.SelectedItem as WarrantyPlan;

var warrantyPrice = selectedPlan?.Price ?? 0M;

var item = new PurchaseItem

{

Name = txtItemName.Text,

SerialNumber = txtSerialNumber.Text,

PurchaseDate = dtpPurchaseDate.Value,

WarrantyExpirationDate = dtpWarrantyExpiration.Value,

PurchasePrice = decimal.Parse(txtPurchasePrice.Text),

MaintenanceNotes = txtMaintenanceNotes.Text,

WarrantyPrice = warrantyPrice

};

await context.PurchaseItems.AddAsync(item);

var saveResult = await context.SaveChangesAsync();

Console.WriteLine($"Saved {saveResult} records");

MessageBox.Show($"Item successfully saved!", "Success");

ClearForm();

}

catch (Exception ex)

{

MessageBox.Show($"Error saving item: {ex.Message}\n\nStack Trace: {ex.StackTrace}", "Error");

}

}

}

private void ShowError(string message)

{

if (lblValidation != null)

lblValidation.Text = message;

}

private void ClearForm()

{

if (txtItemName == null || txtSerialNumber == null || dtpPurchaseDate == null ||

dtpWarrantyExpiration == null || txtPurchasePrice == null ||

txtMaintenanceNotes == null || lblValidation == null || errorProvider == null) return;

txtItemName.Clear();

txtSerialNumber.Clear();

dtpPurchaseDate.Value = DateTime.Now;

dtpWarrantyExpiration.Value = DateTime.Now;

txtPurchasePrice.Clear();

txtMaintenanceNotes.Clear();

lblValidation.Text = "";

errorProvider.Clear();

}

}

}

using System;

using System.ComponentModel.DataAnnotations;

using System.ComponentModel.DataAnnotations.Schema;

namespace WarrantyTracker.Models

{

public class MaintenanceRequest

{

[Key]

public int RequestID { get; set; }

[Required]

public int ProductID { get; set; }

[Required]

public DateTime RequestDate { get; set; }

[Required]

public string Description { get; set; } = string.Empty;

[Required]

public string Status { get; set; } = "Pending"; // Pending, InProgress, Completed, Cancelled

public string? Resolution { get; set; }

public DateTime? CompletionDate { get; set; }

[Column(TypeName = "decimal(18,2)")]

public decimal? Cost { get; set; }

public string? TechnicianNotes { get; set; }

public DateTime CreatedDate { get; set; } = DateTime.UtcNow;

public DateTime LastModifiedDate { get; set; } = DateTime.UtcNow;

// Navigation property

[ForeignKey("ProductID")]

public virtual Product Product { get; set; } = null!;

}

}

using System;

using System.ComponentModel.DataAnnotations;

using System.ComponentModel.DataAnnotations.Schema;

using System.Collections.Generic;

namespace WarrantyTracker.Models

{

public class Product

{

[Key]

public int ProductID { get; set; }

[Required]

public int ModelID { get; set; }

[Required]

[StringLength(50)]

public string SerialNumber { get; set; } = string.Empty;

[Required]

public DateTime PurchaseDate { get; set; }

[Required]

[Column(TypeName = "decimal(18,2)")]

public decimal PurchasePrice { get; set; }

[StringLength(100)]

public string? RetailerName { get; set; }

public DateTime CreatedDate { get; set; } = DateTime.UtcNow;

public DateTime LastModifiedDate { get; set; } = DateTime.UtcNow;

// Navigation properties

[ForeignKey("ModelID")]

public virtual ProductModel Model { get; set; } = null!;

public virtual ICollection<Warranty> Warranties { get; set; } = new List<Warranty>();

public virtual ICollection<MaintenanceRequest> MaintenanceRequests { get; set; } = new List<MaintenanceRequest>();

}

}

using System;

using System.Collections.Generic;

using System.ComponentModel.DataAnnotations;

using System.ComponentModel.DataAnnotations.Schema;

namespace WarrantyTracker.Models

{

public class ProductModel

{

[Key]

public int ModelID { get; set; }

[Required]

[StringLength(100)]

public string ModelName { get; set; } = string.Empty;

[Required]

[StringLength(100)]

public string Manufacturer { get; set; } = string.Empty;

[Required]

[StringLength(50)]

public string Category { get; set; } = string.Empty;

public string? Specifications { get; set; }

public DateTime CreatedDate { get; set; } = DateTime.UtcNow;

public DateTime LastModifiedDate { get; set; } = DateTime.UtcNow;

// Navigation property

public virtual ICollection<Product> Products { get; set; } = new List<Product>();

}

}

using System.ComponentModel.DataAnnotations;

using System.ComponentModel.DataAnnotations.Schema;

namespace WarrantyTracker.Models

{

[Table("PRODUCT")]

public class PurchaseItem

{

[Key]

[Column("ProductID")]

public int Id { get; set; }

[Required]

public int ModelID { get; set; } // Add this line - it's required in the database

[Required]

[StringLength(50)]

public string SerialNumber { get; set; } = string.Empty;

[Required]

public DateTime PurchaseDate { get; set; }

[Required]

[Column(TypeName = "decimal(18,2)")]

public decimal PurchasePrice { get; set; }

[StringLength(100)]

public string? RetailerName { get; set; }

public DateTime CreatedDate { get; set; } = DateTime.UtcNow;

public DateTime LastModifiedDate { get; set; } = DateTime.UtcNow;

// Your additional properties

public string Name { get; set; } = string.Empty;

public DateTime WarrantyExpirationDate { get; set; }

public decimal WarrantyPrice { get; set; }

public string MaintenanceNotes { get; set; } = string.Empty;

}

}

using System;

using System.ComponentModel.DataAnnotations;

using System.ComponentModel.DataAnnotations.Schema;

namespace WarrantyTracker.Models

{

public class Warranty

{

[Key]

public int WarrantyID { get; set; }

[Required]

public int ProductID { get; set; }

[Required]

[StringLength(50)]

public string WarrantyType { get; set; } = string.Empty;

[Required]

public DateTime StartDate { get; set; }

[Required]

public DateTime EndDate { get; set; }

[Required]

[Column(TypeName = "decimal(18,2)")]

public decimal Price { get; set; }

public string? Terms { get; set; }

public bool IsActive { get; set; } = true;

public DateTime CreatedDate { get; set; } = DateTime.UtcNow;

public DateTime LastModifiedDate { get; set; } = DateTime.UtcNow;

// Navigation property to link back to Product

[ForeignKey("ProductID")]

public virtual Product Product { get; set; } = null!;

}

}