```
using System;
using System.Collections.Generic;
using System.ComponentModel;
using System.Data;
using System.Drawing;
using System.IO;
using System.Linq;
using System.Text;
using System.Threading.Tasks;
using System.Windows.Forms;

namespace ProductPerformance
{
    public partial class frmObservations : Form
    {
        private string _selectedPerson = string.Empty;
        private string _selectedProductType = string.Empty;
        private string _selectedProduct = string.Empty;
        private string _selectedDate = string.Empty;
        private string _selectedContinuous = "False";
        private string _selectedSize = "250ml";
        private string _selectedObservation1 = "Very poor";
        private string _selectedObservation2 = "None";
        private string _selectedObservation3 = "None";

        private Observation _currentObservation = new Observation();

        private List<string> _dataState = new List<string>();
        private bool _dataGood = false;


        public frmObservations()
        {
            InitializeComponent();

            //-- Populate the combo box with Persons
            List<string> people = PersonDB.GetAllPersonShortData();
            foreach (var value in people)
            {
                cboPerson.Items.Add(value);
            }

            //-- Populate the ProductType combo
            List<string> types = Lists.GetProductTypes();
            foreach (var value in types)
            {
                cboProductType.Items.Add(value);
            }

            //-- Populate the product sizes list
            List<string> sizes = Lists.ProductSizes();
            foreach (var value in sizes)
            {
                cboSize.Items.Add(value);
            }

            //-- Populate the Observation1 options
            List<string> obs1List = Lists.ObservationOneOptions();
            foreach (var value in obs1List)
            {
                cboObservation1.Items.Add(value);
            }

            //-- Populate Observations 2 and 3
            List<string> obsOther = Lists.ObservationOtherOptions();
            foreach (var value in obsOther)
            {
                cboObservation2.Items.Add(value);
                cboObservation3.Items.Add(value);
            }
```

```
            //-- Date
            _selectedDate = this.dtpFirstUse.Value.ToString("dd/MM/yyyy");

            //-- Continuous use
            _selectedContinuous = "Not continuous use";

            //-- Set control values to default
            ResetFormControls();

            LoadObservationsListBox();
        }

        private void ResetFormControls()
        {
            cboPerson.SelectedIndex = -1;
            cboProductType.SelectedIndex = -1;
            cboProduct.SelectedIndex = -1;
            cboSize.SelectedIndex = -1;
            lblOtherSize.Visible = false;
            txtOtherSize.Text = String.Empty;
            txtOtherSize.Visible = false;
            dtpFirstUse.Value = DateTime.Now;
            chkContinuousUse.Checked = false;
            cboObservation1.SelectedIndex = -1;
            txtOtherObservation1.Text = String.Empty;
            txtOtherObservation1.Visible = false;
            cboObservation2.SelectedIndex = -1;
            txtOtherObservation2.Text = String.Empty;
            txtOtherObservation2.Visible = false;
            cboObservation3.SelectedIndex = -1;
            txtOtherObservation3.Text = String.Empty;
            txtOtherObservation3.Visible = false;
        }



        private void exitToolStripMenuItem_Click(object sender, EventArgs e)
        {
            Utility.CloseApplication();
        }

        private void Clear_Click(object sender, EventArgs e)
        {
            ResetFormControls();
        }


        private void ClearObservationsListBox()
        {
            this.lbObservations.Items.Clear();
        }

        private void LoadObservationsListBox()
        {
            using (StreamReader reader = new StreamReader(@"D:\observations.txt"))
            {
                while (true)
                {
                    string line = reader.ReadLine();
                    if (line == null)
                    {
                        break;
                    }

                    string[] fields = line.Split(',');
                    Observation observation = new Observation()
                    {
                        Person = fields[0],
                        ProductType = fields[1],
                        Product = fields[2],
                        ProductSize = fields[3]
```

```
                };

                string listItem = observation.ObservationDataShort();

                this.lbObservations.Items.Add(listItem);

            }
        }
    }

    private void btnSummary_Click(object sender, EventArgs e)
    {
        var formSummary = new frmSummary();
        formSummary.Show();
        this.Hide();
    }

    // -- Select products and fill combo based on ProductType
    private void cboProductType_DropDownClosed(object sender, EventArgs e)
    {
        //-- Track product type selected
        string productType = cboProductType.GetItemText(this.cboProductType.
        SelectedItem);
        _selectedProductType = productType.Trim();

        cboProduct.SelectedIndex = -1;
        cboProduct.Items.Clear();

        List<string> products = ProductCatalogueDB.ProductsByType(
        _selectedProductType);

        foreach (var value in products)
        {
            cboProduct.Items.Add(value);
        }
    }

    // -- Get 'Other' value selected and show text box
    private void cboSize_DropDownClosed(object sender, EventArgs e)
    {
        string size = cboSize.GetItemText(cboSize.SelectedItem);
        _selectedSize = size.Trim();

        if (size == "Other")
        {
            txtOtherSize.Visible = true;
            lblOtherSize.Visible = true;
        }
        else
        {
            lblOtherSize.Visible = false;
            txtOtherSize.Visible = false;
        }
    }

    private void cboPerson_DropDownClosed(object sender, EventArgs e)
    {
        var person = cboPerson.GetItemText(cboPerson.SelectedItem);
        _selectedPerson = person.Trim();
    }

    private void cboProduct_DropDownClosed(object sender, EventArgs e)
    {
        string product = cboProduct.GetItemText(this.cboProduct.SelectedItem);
        _selectedProduct = product.Trim();
    }

    // -- Update module wide variable if selection changed
    private void dtpFirstUse_ValueChanged(object sender, EventArgs e)
    {
```

```
        _selectedDate = this.dtpFirstUse.Value.ToString("dd/MM/yyyy");
    }

    private void chkContinuousUse_CheckStateChanged(object sender, EventArgs e)
    {
        if (chkContinuousUse.Checked)
        {
            _selectedContinuous = "In continuous Use";
        }
        else
        {
            _selectedContinuous = "Not continuous use";

        }
    }



    // -- Observation combos - deal with other being selected item

    private void cboObservation1_DropDownClosed(object sender, EventArgs e)
    {
        string observationOne = cboObservation1.GetItemText(this.cboObservation1.
        SelectedItem);
        _selectedObservation1 = observationOne.Trim();

        if (_selectedObservation1 == "Other")
        {
            txtOtherObservation1.Visible = true;
        }
        else
        {
            txtOtherObservation1.Visible = false;
        }
    }


    private void cboObservation2_DropDownClosed_1(object sender, EventArgs e)
    {
        string observationTwo = cboObservation2.GetItemText(this.cboObservation2.
        SelectedItem);
        _selectedObservation2 = observationTwo.Trim();

        if (_selectedObservation2 == "Other")
        {
            txtOtherObservation2.Visible = true;
        }
        else
        {
            txtOtherObservation2.Visible = false;
        }
    }

    private void cboObservation3_DropDownClosed_1(object sender, EventArgs e)
    {
        string observationThree = cboObservation3.GetItemText(this.
        cboObservation3.SelectedItem);
        _selectedObservation3 = observationThree.Trim();

        if (_selectedObservation3 == "Other")
        {
            txtOtherObservation3.Visible = true;
        }
        else
        {
            txtOtherObservation3.Visible = false;
        }
    }


    // -- Method to harvest all observation data
```

```
        // -- If 'Other is and option, must get values from the control
        // -- else get from module wide variables previously set when
        // -- options were changed
        private void HarvestObservationData()
        {
            _currentObservation = new Observation();

            _dataState.Clear();

            //-- Person
            if (_selectedPerson == string.Empty)
            {
                //MessageBox.Show("You must select a person");
                _dataState.Add("Bad");
            }
            else
            {
                _currentObservation.Person = _selectedPerson;
                _dataState.Add("Good");
            }

            //-- ProductType
            if (_selectedProductType == String.Empty)
            {
                //MessageBox.Show("You must select a type");
                _dataState.Add("Bad");
            }
            else
            {
                _currentObservation.ProductType = _selectedProductType;
                _dataState.Add("Good");
            }

            //-- Product
            if (_selectedProduct == String.Empty)
            {
                _dataState.Add("Bad");
            }
            else
            {
                _currentObservation.Product = _selectedProduct;
                _dataState.Add("Good");
                ;
            }

            //-- Size
            if (_selectedSize == "Other")
            {
                if (txtOtherSize.Text != "")
                {
                    _selectedSize = txtOtherSize.Text;
                }
            }

            _currentObservation.ProductSize = _selectedSize;

            //-- First use. Should always have a value
            _currentObservation.FirstUse = _selectedDate;

            //-- Continuous
            _currentObservation.Continuous = _selectedContinuous;

            //-- Observation1
            if (_selectedObservation1 == "Other")
            {
                _currentObservation.Observation1 = txtOtherObservation1.Text;
            }
            else
            {
                _currentObservation.Observation1 = _selectedObservation1;
            }
```

```
            //-- Observation2
            if (_selectedObservation2 == "Other")
            {
                _currentObservation.Observation2 = txtOtherObservation2.Text;
            }
            else
            {
                _currentObservation.Observation2 = _selectedObservation2;
            }

            //-- Observation3
            if (_selectedObservation3 == "Other")
            {
                _currentObservation.Observation3 = txtOtherObservation3.Text;
            }
            else
            {
                _currentObservation.Observation3 = _selectedObservation3;
            }
        }


        // --- Write the observation to the database
        private void btnAddObservation_Click(object sender, EventArgs e)
        {
            //-- Populate class with form data
            HarvestObservationData();
            //-- Try the write
            if (ObservationDB.WriteObservation(_currentObservation))
            {

                MessageBox.Show("Observation written to file");
            }

            ClearObservationsListBox();
            LoadObservationsListBox();

            //_currentObservation = null;
            ResetFormControls();
        }
    }
}
```