Introduction

The aim of the task is to design and develop a simple AUV (Autonomous Underwater Vehicle) monitoring application. The AUV is equipped with a collection of sensors that provide information about the current pressure and temperature. Unfortunately, due to mistakes in the hardware design, the sensors provide the readings in different units. Temperature is provided in Celsius or Fahrenheit while pressure readings are provided in kPa or PSI. The purpose of the application is to provide the remote operator with a unified user interface providing readings of both temperature and pressure from all AUV sensors.

Technical Details

The task package contains a .NET Standard library with the AUV sensors API which can be used to obtain the latest sensor readings. The library contains a main factory object which can be called to obtain the list of all sensors installed in the AUV:

```
using Wavefront.AUV.API;

var sensorsList = AUVSensorsFactory.Build();
```

Each sensor in the returned list implements the following interface:

```
using Wavefront.AUV.API.Enums;

namespace Wavefront.AUV.API.Interface
{
    /// <summary>
    /// AUV Sensor Interface
    /// </summary>
    6 references
    public interface IAUVSensor
    {

        /// <summary>
        /// Gets sensor unique ID
        /// </summary>
        6 references
        public int SensorId { get; }

        /// <summary>
        /// Gets the unit used by the sensors when temperature is measured.
        /// </summary>
        4 references
        public eTemperature TemperatureUnit { get; }

        /// <summary>
        /// Gets the unit used by the sensors when pressure is measured.
        /// </summary>
        4 references
        public ePressure PressureUnit { get; }

        /// <summary>
        /// Gets the latest temperature reading
        /// </summary>
        /// <returns>Value of temperature reading in unit defined by TemperatureUnit property.</returns>
        2 references
        public double GetTemperature();

        /// <summary>
        /// Gets the latest pressure reading
        /// </summary>
        /// <returns>Value of pressure reading in unit defined by PressureUnit property.</returns>
        2 references
        public double GetPressure();
    }
}
```

Functional Requirements

The following section provides formal task requirements in the form of simple user stories which need to be implemented by the AUV monitoring application

- As a user I want to use simple AUV sensors monitoring application so I can check the temperature and pressure readings while the AUV is in operation.

- As a user I want to see a list of AUV sensors so I can monitor the temperature and pressure readings from all sensors installed on the AUV.

- As a user I want to see information from all AUV sensors in unified units so that I can directly compare the difference of all the readings on one screen.

- As a user I want to be able to select which unit the temperature and pressure readings from the sensors are displayed in so that I don't have to convert the values.

- As a user I want an automatic update of the temperature and pressure readings on the screen every second so that I can see the latest values provided by the sensors all the time.

- As a user I want the application to be robust so when the error in sensor reading happens the application do not crash.

Solution Review Criteria

The main aim of this task is to demonstrate the ability to understand the requirements, design simple solution and develop basic application providing automatic and recurrent update of the data on the screen. The solution should be developed as a Windows Application using the Sensors AUV library provided. Please focus on the architecture of the solution rather than the look and feel of the UI. It is more important to demonstrate solid software engineering and OOD practices with an understanding of the software architecture rather than demonstrating the ability to design a wonderful user interface.

What we are looking for:

- How the code is structured?
- How the code is implemented and is it easy to understand?
- How requirements were understood and what assumptions were made?
- How the automatic reading updates are managed?
- How the conversion of the units is implemented?
- How the system is handling errors?

Which areas of the solution will **NOT** have an impact on the solution assessment:

- What technology stack was selected to implement the solution (WPF, Forms, Web UI)?
- How the user interface looks as long as it meets the minimum requirements?
- The exact conversion values used to translate the units. Simple values from the internet can be used.