Project work

# Requirements

## Challenge

Within the team, we don't like clocks that display as numbers. In fact, we like clocks that present the current time in a more "Human Friendly" way.

## Objectives

|  |  |
| --- | --- |
| 1 | Write a command-line program that returns the current time using the "Human Friendly Text" demonstrated in the example below.  Numeric Time Human Friendly Text  • 1:00 One o'clock  • 2:00 Two o'clock  • 13:00 One o'clock  • 13:05 Five past one  • 13:10 Ten past one  • 13:25 Twenty five past one  • 13:30 Half past one  • 13:35 Twenty five to two  • 13:55 Five to tw  For example, if we execute this program at 16:30, it should output "Half past four"  C:\ ..some command..  Half past four |
| 2 | Allow the command to take an arbitrary Numeric Time parameter as input and return the "Human Friendly Text" equivalent.  For example:  C:\ ..some command.. 15:00  Three o'clock |
| 3 | * This is to promote conversation and to see how you approach coding challenges and problems. Please supply clean *production ready* code / scripts. * You must use C#. You are able to use any library or framework to complete the objectives. * Do not spend too long on it – its not supposed to be onerous   It doesn’t matter how you approach the details of the problem – this is a technical competency not an English test – so use your judgement on edge cases or any assumptions you need to make. |

# Design (Technical)

Design goes here.

|  |  |
| --- | --- |
| 1 | Prepare class and interface diagrams |
| 2 | Design namespace, interface, classes, properties, methods and variables |
| 3 | Design code analysis rules set   1. CodeAnalsyis.Csharp 2. CodeAnalsyis.Csharp.Features 3. CodeAnalsyis.Features |
| 4 | Design algorithm |
| 5 | Design unit test cases |

# Test Strategy

The Test strategy goes here

|  |  |
| --- | --- |
| 1 | Prepare use cases from requirements |
| 2 | Code analysis rule set |
| 3 | Prepare unit test cases as per the requirement use cases |

# Coding

Plan for coding goes here

|  |  |
| --- | --- |
| 1 | Setup empty project in repository Git |
| 2 | Create a branch, and pull the branch to local git repository |
| 3 | Create new solution in visual studio (2019) using C# |
| 4 | Add Class library project “HumanarizeClockTime” |
| 5 | Obtain Humanarize library from Nuget.  Add reference to the project |
| 6 | Add console project “ReadableClockTime” to the solution |
| 7 | Add Unit Test Case project to the solution |
| 8 | Compile and test in Debug mode |
| 9 | Compile in Release mode |
| 10 | Push the changes to remote branch in Git |
| 11 | Merge the code |
| 12 | Tag the branch |
| 13 | Release binaries |

# Quality Control

## Code Analysis

|  |  |
| --- | --- |
| 1 | Create code analysis rule sets |
| 2 | Execute them |

## Unit Testing

|  |  |
| --- | --- |
| 1 | Write unit test cases to handle use case scenarios |
| 2 | Write test data with positive input data in json format in xml file |
| 3 | Write test data with negative input data in json format in xml file |
| 4 | Execute unit test cases |
| 5 | Resolve any bugs |

# Software Package

|  |  |
| --- | --- |
| 1 | Compile solution in release mode |
| 2 | Package the binaries and publish to package folder |

# Release

|  |  |
| --- | --- |
| 1 | Release the package via folder copy or ftp |
| 2 | Push the code binaries package to the repository (Git) |