Contents

[Analysis 1](#_Toc93476646)

[The problem 1](#_Toc93476647)

[Stakeholders 1](#_Toc93476648)

[Research 1](#_Toc93476649)

[Essential feature 1](#_Toc93476650)

[Requirements 1](#_Toc93476651)

[Success criteria 1](#_Toc93476652)

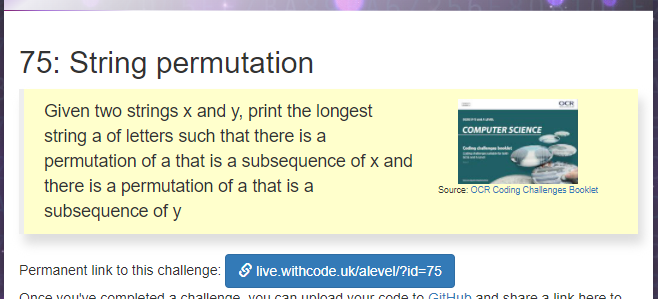
[Design 1](#_Toc93476653)

[Implementation 1](#_Toc93476654)

[Evaluation 1](#_Toc93476655)

# Analysis

## The problem



## Stakeholders

## Research

## Essential feature

## Requirements

## Success criteria

|  |  |  |
| --- | --- | --- |
| Number | Criteria | Justification |
| 1 | It must have a graphical user interface | The user needs to be able to control the program with a mouse and see the responses on the screen |
| 2 | User must be able to type in string x |  |
| 3 | User must be able to type in string y |  |
| 4 | The program must display the longest sequence of letters that is shared by both strings |  |
| 5 | If either string is left empty, an error message asks the user to enter a string | The stakeholder wants the program to be easy to use and it should be impossible to crash the program by entering invalid data |
| 6 | If no letters or sequence of letters are shared between x and y, a message is displayed to that effect |  |
| 7 | It must work on a Windows 10 computer with a screen size of 1920x1080 or larger |  |
| 8 | Must have a title at the top |  |
| 9 | Must have a help button to show hints on how to use the program |  |
| 10 | User input must not be more than 1024 characters |  |
| 11 | The user input must work with any valid ASCII character |  |

# Design

## Decomposition

## Structure

Spiral iterative approach.

Iteration 1

Prototype 1.1: user interface with no functionality

Prototype 1.2: user interface with user input validated

Iteration 2

Prototype 2.1 Calculation for normal data

Prototype 2.2 Calculation for all data

Prototype 2.3 Fully working solution with help screen

## Algorithms

Function ValidationResult ValidateString(string userInput) {

// presence check

// length check

}

## Usability

String Overlap Calculator

String X

String Y

Calculate

Result

Help

Justification:

## Variables and data structures

|  |
| --- |
| ValidationResult |
| +ValidationSuccess:bool  +Message |
| +ValidateUserInput(string):bool |

## Test data

Prototype 1.1

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Test number | Description | Test data | Expected result | Success criteria |
| 1.1a | User interface | Start the program | User should be able to enter in two strings (X and Y) and press a button (calculate).  Nothing needs to happen yet but the title “String comparer” should be at the top of the form | 1, 8 |

Prototype 1.2

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Test number | Description | Test data | Expected result | Success criteria |
| 1.2a | Validate string X  Presence check | User enters “” for string x | Program should not crash but displays “Please enter a valid string” as an error message | 2, 5 |
| 1.2b | Validate string X  Length check | User enters 1024 zeros for string x | String x is accepted as valid | 2 |
| 1.2c | Validate string X  Length check | User enters 1025 zeros for string x | “String x is too long” message appears | 2 |
| 1.2d | Validate string X pattern check | User enters “£3.50” | “Non ASCII character detected in string x” message appears | 2 |
| 1.2e | Validate string Y  Presence check | User enters “” for string y | Program should not crash but displays “Please enter a valid string” as an error message | 3, 5 |
| 1.2f | Validate string Y  Length check | User enters 1024 zeros for string y | String y is accepted as valid | 3 |
| 1.2g | Validate string Y  Length check | User enters 1025 zeros for string y | “String y is too long” message appears | 3 |
| 1.2h | Validate string Y pattern check | User enters “£3.50” | “Non ASCII character detected in string y” message appears | 3 |

## Post development testing

# Implementation

# Evaluation