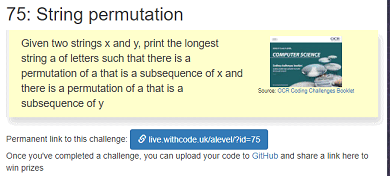
# Analysis

## 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 |  |  |
| 6 |  |  |
| 7 |  |  |
| 8 |  |  |
| 9 |  |  |
| 10 |  |  |
| 11 |  |  |

# 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 ValidString(string userinput){

// presence check

// length check

}

## Usability

Help

String X

String Y

Result

Calculate

String Overlap Calculator

Justification:

## Variables and data structures

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

## Test data

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| 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 the a button (Calculate). Nothing needs to happen yet but the title should be at the top of the form | 1, 8 |
| 1.2a | Validate string X presence check | User enters “” for string x | Program should not crash but display |  |
|  |  |  |  |  |

## Post development testing

# Implementation

# Evaluation