**Task Report**

**Python Validation Model**

Contents

[**1.** **Version history** 4](#_Toc101789451)

[**2.** **Introduction** 4](#_Toc101789452)

[2.1. Overview 4](#_Toc101789453)

[2.2. Source code 4](#_Toc101789454)

[**3.** **Contributors** 4](#_Toc101789455)

[3.1. List of contributors 4](#_Toc101789456)

[3.2. Initial task division 4](#_Toc101789457)

[**4.** **Challenges** 4](#_Toc101789458)

[4.1. Solved 4](#_Toc101789459)

[4.1.1. Python version issue 4](#_Toc101789460)

[4.1.2. Modeling upwards vs downwards 5](#_Toc101789461)

[4.1.3. x 5](#_Toc101789462)

[4.2. Unsolved 5](#_Toc101789463)

[4.2.1. x 5](#_Toc101789464)

[4.2.2. x 5](#_Toc101789465)

[**5.** **Sources** 5](#_Toc101789466)

[5.1. Python documentation for base python functions 5](#_Toc101789467)

[5.2. Pandas user guide 5](#_Toc101789468)

[5.3. Stackoverflow threads which help implementation 5](#_Toc101789469)

[5.4. Other useful links 5](#_Toc101789470)

[**6.** **Ideas for future development** 5](#_Toc101789471)

[6.1. Add relations between Test Case and Test Cycle 5](#_Toc101789472)

[6.2. x 6](#_Toc101789473)

[**7.** **Completion report** 6](#_Toc101789474)

[7.1. Validation model application 6](#_Toc101789475)

[7.2. Hangman – Python application to be tested 6](#_Toc101789476)

[7.3. Final task division 6](#_Toc101789477)

[7.4. x 6](#_Toc101789478)

[**8.** **Retrospection** 6](#_Toc101789479)

[8.1. What went well 6](#_Toc101789480)

[8.2. What needs improvement 6](#_Toc101789481)

### **Version history**

|  |  |  |
| --- | --- | --- |
| **Revision** | **Changes** | **Date** |
| 0.1 | Published | 24-04-2022 |
| 0.2 | Footer cleanup, contributors listed | 24-04-2022 |
| 0.3 | Added sources, Validation App completion report, idea for future improvement | 25-04-2022 |

### **Introduction**

### Overview

This document serves as the completion report for two-part task, which consisted of creating:

1. validation model and CLI application for managing test records, with proper documentation
2. Python application, with validation planned using above model and application

### Source code

All files created for the purpose of this task can be found at <https://github.com/dlukaszx/Project>

### **Contributors**

### List of contributors

* Dawid Łukaszewski
* Marta Czaja
* Piotr Skierkowski

### Initial task division

|  |  |  |  |
| --- | --- | --- | --- |
| Task | Dawid | Marta | Piotr |
| Validation model |  | X |  |
| CLI app for the model (all) |  |  | X |
| Model documentation |  | X |  |
| Python app to be tested | X |  |  |
| Testing scope | X | X | X |
| Test execution | X | X | X |
| Completion report |  | X |  |

### **Challenges**

### Solved

### Python version issue

Development of apps was initially split between Python version 3.6 and 3.10. As task clearly listed 3.6 version we opted to downgrade to unify the code and ensure compatibility.

### Modeling upwards vs downwards

One of the things considered was keeping child data in parent record (e.g. all related Test Cases listed in Test Case Description). However, since TC has to be related to single TCD it was more efficient to ask for parent TCD ID upon TC creation and store the value in TC record. Same rule applies to TC-TR and Cycle-TR: they are all one-to-many relations.

### x

### Unsolved

### x

### x

### **Sources**

### Python documentation for base python functions

* <https://docs.python.org/3/tutorial/>

### Pandas user guide

* <https://pandas.pydata.org/docs/user_guide/index.html#user-guide>

### Stackoverflow threads with helpful information

* <https://stackoverflow.com/questions/517970/how-to-clear-the-interpreter-console>
* <https://stackoverflow.com/questions/3207219/how-do-i-list-all-files-of-a-directory>
* <https://stackoverflow.com/questions/31536835/extract-value-from-single-row-of-pandas-dataframe>
* <https://stackoverflow.com/questions/56436024/python-3-count-number-of-rows-in-a-csv>

### Other useful links

* <https://pandas-xlsxwriter-charts.readthedocs.io/chart_pie.html>

### Data sets for Hangman application

* <https://pypi.org/project/english-words/>
* <https://www.wordexample.com/list/proverbs-in-english>

### **Ideas for future development**

### Add relations between Test Case and Test Cycle

Current version of application only have Test Cycle related to Test Result, what means, that Test Case scope for each Test Cycle is the same. If there is new functionality implemented in tested application, all new Test Cases will be included in reports generated for older Test Cycles and marked as ‘NOT RUN’. The same situation will occur for new test Cycles if functionality is removed from tested application.

### More data sets and balance update in Hangman game

Available version of application contains nearly 70000 unique passwords, but most of them are single words. New data sets can be easily added in the future. Game also might need balance update. Six lives may be enough for proverbs, where it is easier to guess a correct letter, but there can be a problem with single words.

### **Completion report**

### Validation model application

* Add new TP, TCD, TC, TR, Cycle – DONE
* Add relations between elements – PARTIALLY DONE
  + Add TCD – TP (many to many) relation – DONE
  + Cycle-TR, TR-TC, TC-TCD (one to many) relations – DONE
  + Cycle-TC – TO DO
* Display all TC for selected TP – TO DO
* Display all TR for selected TP – TO DO
* Display all TR for selected Cycle – TO DO
* Export validation report for specific Cycle as Excel file – DONE

### Hangman – Python application to be tested

* Choose game mode – DONE
* Guess letters and draw hangman – DONE
* End game and restart option – DONE
* Test application – TO DO

### Final task division

|  |  |  |  |
| --- | --- | --- | --- |
| Task | Dawid | Marta | Piotr |
| Validation model |  | X |  |
| CLI app (framework) |  |  | X |
| CLI app (excel integration) |  |  | X |
| CLI app (relations) |  |  | X |
| CLI app (record display) |  | X |  |
| Model documentation |  | X |  |
| Python app to be tested | X |  |  |
| Testing scope | X | X | X |
| Test execution | X | X | X |
| Completion report | X | X | X |

### x

### **Retrospection**

### What went well

* X
* X
* X

### What needs improvement

* X
* X
* X