**BibTeX**

**Final Report**

**Author: Jakub Hutny**

List of contents

[1. Test scenarios. 3](#_Toc438413389)

[2. Testing Methodology. 6](#_Toc438413390)

[3. Testing Results. 6](#_Toc438413391)

[4. Conclusions 6](#_Toc438413392)

Revision History

|  |  |  |  |
| --- | --- | --- | --- |
| **Date** | **Version** | **Description** | **Author** |
| 20-12-2015 | 0.1 | First draft. | Jakub Hutny |
| 20-12-2015 | 0.2 | Added content. | Jakub Hutny |
|  |  |  |  |

# Test scenarios.

* Importing files.

|  |  |  |
| --- | --- | --- |
| **Pre-conditions** | **Steps To Reproduce** | **Expected Result** |
| Proper .bib file exists. | 1. Choose ‘Open’ from ‘File’ menu.  2. Choose proper .bib file and load it. | 1. File explorer should be opened.  2. Dialog box with ‘File opened’ text should appear and entries should be filled with data. |

* Exporting files.

|  |  |  |
| --- | --- | --- |
| **Pre-conditions** | **Steps To Reproduce** | **Expected Result** |
| Data is loaded in the program. | 1. Choose ‘Save’ from ‘File’ menu.  2. Put the name of new file and click ‘Save’. | 1. File explorer should be opened.  2. New .bib file with data from program should appear. |

* Making backup of the imported file.

|  |  |  |
| --- | --- | --- |
| **Pre-conditions** | **Steps To Reproduce** | **Expected Result** |
| Proper .bib file exists. | 1. Choose ‘Open’ from ‘File’ menu and load the .bib file. | 1. After loading the .bib file, there should be backup .bib file in the folder containing the original file.  . |

* Logging the actions.

|  |  |  |
| --- | --- | --- |
| **Pre-conditions** | **Steps To Reproduce** | **Expected Result** |
| Proper .bib file exists. | 1. Choose ‘Open’ from ‘File’ menu and load the .bib file.  2. Perform some actions, i.e. change some Styles.  3. Click ‘Undo’ button. | 1. Dialog box with ‘File opened’ text should appear and entries should be filled with data.  2. Actions should be logged.  3. Actions should be undone. |

* Management of non-unique identifiers.

|  |  |  |
| --- | --- | --- |
| **Pre-conditions** | **Steps To Reproduce** | **Expected Result** |
| Two .bib files with the same identifier exist. | 1. Choose ‘Open’ from ‘File’ menu and load the .bib file.  2. Change the identifier to be unique. | 1. Window should appear that allows to change the identifier to be unique.  2. Identifier should be unique. |

* Choosing from multiple styles.

|  |  |  |
| --- | --- | --- |
| **Pre-conditions** | **Steps To Reproduce** | **Expected Result** |
| Proper .bib file exists. | 1. Choose ‘Open’ from ‘File’ menu and load the .bib file.  2. Choose one entry.  3. Open Styles list. | 1. Dialog box with ‘File opened’ text should appear and entries should be filled with data.  2. Parameters should be loaded.  3. Multiple styles should be available. |

* Ignoring selected fields of entry.

|  |  |  |
| --- | --- | --- |
| **Pre-conditions** | **Steps To Reproduce** | **Expected Result** |
| Proper .bib file exists. | 1. Choose ‘Open’ from ‘File’ menu and load the .bib file.  2. Choose one entry.  3. Untick ‘Enabled’ checkbox.  4. Export the file. | 1. Dialog box with ‘File opened’ text should appear and entries should be filled with data.  2. Parameters should be loaded.  4. Disabled parameter should not be put in the exported file. |

* Management of abreviations.

|  |  |  |
| --- | --- | --- |
| **Pre-conditions** | **Steps To Reproduce** | **Expected Result** |
| Proper .bib file exists. | 1. Choose ‘Open’ from ‘File’ menu and load the .bib file.  2. Choose one entry.  3. Click ‘Abbreviations’.  4. Add new abbreviation.  5. Edit existing abbreviation.  6. Delete existing abbreviation. | 1. Dialog box with ‘File opened’ text should appear and entries should be filled with data.  2. Parameters should be loaded.  3. Abbreviations managements panel should be opened.  4. Abbreviation should be added.  5. Edited data should be saved.  6. Abbreviation should be deleted. |

# Testing Methodology.

Testing was performed at two levels: integration and acceptance. Both of them were done manually.

Integration tests were provided after each new functionality implementation.

Acceptance tests were provided at the end of the project to verify correctness of the solution.

# Testing Results.

|  |  |  |
| --- | --- | --- |
| **No** | **Test Name** | **Outcome** |
| 1 | Importing files. | Passed |
| 2 | Exporting files. | Passed |
| 3 | Making backup of the imported file. | Failed |
| 4 | Logging the actions. | Failed |
| 5 | Management of non-unique identifiers. | Failed |
| 6 | Choosing from multiple styles. | Passed |
| 7 | Ignoring selected fields of entry. | Failed |
| 8 | Management of abreviations. | Failed |

# Conclusions

Basing on the acceptance tests, we can see that we did not get a goal of the project. Most of the functionalities are not implemented at all or just do not work properly.

Analysing the whole project, we found following sources of our failure:

* lack of programming knowledge in case of the challenges of the project
* bad project organization
  + rather poor task description and division
  + no official flow of task – no taskboard
  + too less people in the team
  + no live meeting – basing on the Skype meetings, which were not enough

All those reasons together resulted in our failure. It is painful, but valuable lesson for us that good organization is base for all the other project job.