Inspection report for Java-Cli-Table-Builder

Group members:
Serhat Altay - 000398424
Chuangzhang Chen - 000210010
Trieu Huynh Ba Nguyen - 000405980
Annalina Wheeler - 000409268
Yatian Zheng - 000213156

Document history

Version	Date	Creator	Latest updates
T1	20.10.2022	Annalina Wheeler	Document created from template
T2	22.10.2022	Trieu Huynh Ba Ngyuen	Review of environment and test tools was written
T3	23.10.2022	Annalina Wheeler	Review of non-functional and other test cases was written
T4	24.10.2022	Yatian Zheng	Review of functional test cases, written tests and existing tests was written
T5	24.10.2022	Serhat Altay	Testing strategy report was written
T6	26.10.2022	Chuanzhang Chen	Review of bug report was written
T7	26.10.2022	Annalina Wheeler	Conclusion was written
T8	27.10.2022	Trieu Huynh Ba Ngyuen	Points validation was reviewed
T9	27.10.2022	Serhat Altay	Risks report was written

Table of content

Introduction

We are chosen to check a total of 19 pages of test reports that five students had submitted using the Java-Cli-Table-Builder command-line interface application. However, the description of Java-Cli-Table-Builder is too minimal for us to gain sufficient information that would make it easier for us to comprehend the project. Upon the first viewing of the project, in general, the report is outwardly well-structured and described by a combination of graphics and text, so that we can understand the project more clearly. However, the test strategy section's many typefaces and the existing test section's confusing image layout reduce overall legibility and readability. In addition, the report largely covers the components required for the software testing course, demonstrating that the group has a basic understanding of what testing entails and can test software projects successfully and analyze the results. We will analyze each component in detail below and offer some practical suggestions to help them improve. The documented history shows that the group put a lot of time and effort into the whole project. However, the test report lacks a citation section, which reduces the report's persuasiveness.

Points validation

Feature	Validity	Note
We have followed systematic and documented standards in the testing process and completed the reports well.	Partially valid	The report had an inconsistent format and confusing terms. There was a table of contents, but no page numbers.
The software installation process has been carried out as a test case	Valid	
Software installation process has been carried out with 4 different devices	Invalid	There were mentionings of testing environments, but not devices used.
The software has been tested with 4 different environments to test whether the minimum system requirements are solid	Valid	
Software has been tested with 4 different devices to test, for example, whether the user interfaces scale to different sizes	Invalid	There were mentionings of testing environments, but not devices used.
Design 10 functional test cases of your choice (e.g., “User should be able to save a file”) and test them	Partially valid	Many test cases were not specific enough and too general to qualify as test cases. The terms used were methods or categories of tests.
Design 10 non-functional test cases of your choice (e.g., “System response time is fast [What is “fast”? Define it!] ”) and test them	Partially valid	
Code inspection to a part of source code (minimum 100 lines of code)	Invalid	Code inspection was not documented in the report.
Run existing (unit) tests to check the state of software (and the tests)	Partially valid	Some existing tests were also included in functional test cases, leading to unclear formatting of the report.
Found a fault and reported it to developers	Invalid	Reporting to the developers was not documented in the report.
Inspect the documentation; it is up-to-date.	partially valid	Documentation review was insufficient. The documentation was relatively complete, and should not be considered a “fail.”
We checked the pom.xml file and found that the versions of external libraries are outdated and cause the environment and external libraries synchronized by this file cannot support that code run in the correct way.	Invalid	This is not a valid point proposal.

Testing strategy

The testing strategy is written with much unclearness, and the overall testing strategy seems a bit ambiguous due to the confusing terms used to describe testing methods. There are many confusing terms in this section, such as the Value range test, String length test, Character type test, Form test, Cell Border Template Test, CeilTest, ExampleTest, Table builder Test, etc., that cannot be interpreted as special testing methods. However, these terms are described in the testing strategy as different categories of testing, in the same categories as the black-box and regression testing. Additionally, there is a missing description of these terms' goals, such as where to use these testing methods, for what purposes, and why these methods are chosen specifically. The only explanation is "Due to the nature of the project," but the nature of the project part is not described to understand why these testing methods are applied.

A good example of explaining why a testing method is chosen can be seen in the second paragraph, where the reasons for the white-box testing method are justified. To sum up, the testing strategy of the project could have been written with a deeper understanding of the testing methods to avoid confusing terms. The justification could also be more detailed and complete, as it was not enough to comprehensively understand the project's testing strategy.

Environment and test tools

The group being peer-reviewed has provided the environments used during testing. However, only the operating systems (Windows, MacOS, Ubuntu) could be seen; there are no other specifications of each individual machine used for testing. Due to the nature of hardware components, even though the operating system is the same, the results of testing could vary significantly. Furthermore, the authors said that all group members had “Windows 10 or Windows 11 laptops”, so running MacOS was not feasible through ordinary means. It was likely that an installation of MacOS either on non-Apple Inc. hardware or on a virtual machine was conducted. This improvisation could be very unstable and affect the outcomes of testing activities. To make the testing process less susceptible to external factors, the environment (operating system) should be installed as a separate entity on its dedicated platform.

Regarding the tools, the authors listed “Netbean, IntelliJ IDEA Community Edition 2021.3.1, VMware Horizon Client”. These are not software testing tools, since a testing tool must support one or more testing operations, such as planning, requirements, constructing a build, test execution, defect tracking, and test analysis. Netbeans and IntelliJ IDEA are both integrated development environments (IDE), while VMware Horizon Client is a virtualization product that is capable of simulating an operating system. Therefore, the authors have not elaborated on the tools they used during testing at all.

In the operation part, the authors mentioned that they were not able to execute the program during the first trial due to outdated libraries. Then the authors proceeded to “upgrade” using IntelliJ IDEA. However, they clarified neither how this process was carried out nor what was modified. After that, the program was run on Ubuntu via the LUT-VDI remote access service. This virtual machine provided by our university is a highly controlled environment, making testing much less prone to system-related errors. By conducting testing activities in such an environment, the outcomes were once again affected by other factors.

Existing tests

In the existing tests, the authors did not write the test cases they found that the developers had done from GitHub or other sources, but explained their functional test cases. Therefore, the description of this aspect is incorrect. The review of the content of existing tests is presented in the functional tests section.

Test cases

Functional test cases, written tests, and existing tests

In general, the authors did functional tests and used case descriptions mostly correctly. They took different types of tests and described them briefly. However, there is something confusing. First, the authors divide the description of functional tests into two subtopics, existing tests, and written tests. This looks a bit messy and inconsistent. It would be better if they were written below the table like non-functional tests.

In the existing tests, some tests that have been done by the developers or testers should be described, explained, and evaluated in the existing tests, not the authors' test cases. Likewise, the explanations of functional test cases are also written in the written test. But "written test" is confusing. “written test” is not considered a term, and it doesn't make sense. Do you mean to do the test by typing something?

In terms of content, the authors describe the test cases clearly and add pictures to show the results. However, the specific test content should be written in the "Feature being tested" of the table. For example, it would be better to replace "correlation test" with "test the effect of removing an item in a function on the output", since “correlation test” is just a broad term. The authors said black-box and unit tests were done on F-5, F-6, F-8, F-9, and F-10. But unit testing is a white-box technique, not black-box testing. In F-1 in the written tests, the same sentences and pictures appeared twice in the results described by the authors. Also, headings and steps are missing in the F-9 and F-10. These issues should be found before the project was submitted.

Non-functional and other tests

The overall layout of the non-functional test cases is quite clear and generally good. However, some of the tests included in this section seem to be functional tests rather than non-functional. These include the installation test as well as the executability test, as they are directly tied to the necessary behavioral requirements of the software.

The table itself is easy to interpret as each box is filled out clearly and in a relatively uniform manner. On the other hand, the third column (“Feature being tested”) is somewhat confusing due to the broadness of the terms used. For instance, although the purpose, test procedure, and results for each test case are clearly specified in the notes below the table, the terms used in this column, e.g., “compatibility test”, are not test cases, but instead, they are categories or types of testing. This could be changed to a specific test case under the category of compatibility testing, for instance, saying that a specific task in the program was performed using MacOS-, Windows-, and Linux- based devices. The steps taken in the testing process, as well as the results of the test cases, should be reported in more detail by describing what the test case was, what exactly was done to test it, and specifying what kinds of results were obtained.
Similarly, the other tests -test cases were too general to be considered as test cases as, for example, ad-hoc testing is a type of testing rather than an individual test case.

Bug report

We believe they did a great job with their bug report. When the text input was in a different language, bugs appeared. However, there exist two issues. The truth that they omit the bug report from the point proposal is the first problem. The bug report should be mentioned in the point proposal. Second, they said in the point proposal that they discovered a bug and reported it to the developer. However, there was no proof that they informed the developer of the defect. Through email or other channels, they can notify the developer of the bug.

Risks report

Overall, the risks report is written considering the main idea of the software and its functionality. An explanation of how the software works with different languages and characters is specified here. Also, it is emphasized that the program has several bugs while writing with characters other than ASCII, while the program did not crash due to any character. These facts are highly related to the risk report of the program. However, the report could be more detailed by mentioning more specific terms and aspects, such as reliability, ease of usability, and availability. Explaining whether these aspects are fulfilled to meet the need of production of the software makes the risk report more understandable by pinpointing which aspects of the software the tester is talking about. In addition, some testing methods, such as internationalization testing, were mentioned in this section but not in the test cases part.

In the end, the tester mentioned the improvements the software can apply by explaining that the acceptable characters by the software need to be increased. The tester said that automated testing for other characters and languages for the testing part of the software could be taken into consideration. However, some of the reasoning was absent here as the tester didn't mention why acceptable characters need to be increased, what improvements it can bring, and what bugs and output errors it can prevent. Moreover, and most importantly, the tester said in their last sentence that automated testing should be done whenever possible, which is completely opposite to what we learned during the course. Automating testing can be very beneficial. However, it can also be a costly tool. So, the tester needs to consider when to use automated testing, not whenever possible.

Conclusion

It is evident from inspecting the testing report that significant time and effort have been put towards the project as a whole. However, the layout of the document itself is somewhat confusing and quite inconsistent as, for instance, the notes for the functional-, non-functional-, and other test cases do not follow the same logic, but are instead written in completely different parts of the project report. The report includes quite a bit of repetition and redundancies as the contents of some of the sections, such as the introduction and testing strategy, seem to contain the same information twice. Another aspect to consider is that there were some missing sections that were mentioned in the points proposal, but could not be found in the actual report, such as the code review or reporting a fault to the developers. In addition, the report also does not contain in-text citations or a reference list at the end of the report.

More information could be provided with regards to the software itself — what it is, what it can be used for, and potentially some screenshots — in order to give the reader an improved understanding of the value of the aspects under review. However, this report did well in describing the problem found in the bug report, as both the expected and achieved results were clearly visualized in their respective boxes. Furthermore, the execution of existing unit tests was an interesting addition to the project as it contained topical screenshots along with clear descriptions of the steps taken in the process as well as the results achieved from the testing.