Julia Abbott

ZKanji Testing Document

Table of Contents

Software Version and Summary: page 1

Deliverables and Bug Logs: page 2

Test review Overview: page 3

Code Review Overview: page 3

Exploratory testing Overview: page 3

Exploratory testing Log: pages 3-4

Security Testing Overview: page 4

**Software**

Version: 0.731

Summary: Acts both as a study guide for the Japanese language, and also as a kanji dictionary. Features include, but are not limited to:

* Hiragana and Katakana reading and writing practice
* Kanji lookup through meaning, hiragana, and radicals

Objective for Testing: Ensuring there are no fatal defects such as security flaws, the program crashing, and a reasonable amount of correctness

Note: Software is only supported on Windows platforms

**System 1**

Windows 10.1

**System 2**

Windows 2000, using the program Virtual Box to create a virtual machine. (Not yet implemented)

Link to Deliverables:

<Deliverables.xlsx>

Link to Bug Log:

[Bug log.xlsx](Bug%20log.xlsx)

**Code Reviews**:

Code reviews will be used in two ways. First, they will be used to understand the processes behind the interface. Secondly, they will be used to identify possible weak spots within the source code itself. Needless to say, both instances will lead to more test cases, and, hopefully, errors.

As I am working by myself, I will take on all of the roles during the code review. This includes, but is not limited to, reading the code, looking for possible errors in the code, coming up with test cases, and documenting my findings. Also, since there are no scheduling conflicts, it is sufficed to say that the first code review will take place in November, and any concurrent reviews after this.

**Test Case Reviews:**

I do not foresee test case reviews being very helpful as I am working by myself, instead of in a group. Because of this, I will not have to worry about duplicates in test cases, as I will be writing and executing all of them myself. I will also not have another pair of eyes/ears to bounce ideas off of, and to work with to find cases I might have missed. That being said, test case reviews will be useful when I am trying to evaluate my coverage of sections within the program, and will thus be utilized as such.

**Exploratory Testing:**

Exploratory Testing has been used extensively throughout the testing process. In order to determine which parts of the user interface the user was able to get to, and most likely to use, I tried to use the program as if I were a regular user. The results of this was that I created a series of tests to make sure that the reading test was working as it should, as it is probably one of the most used features of this program. Exploratory testing was also used when searching for random kanji, as it is impossible to search for each and every kanji and ensure that they each have the correct meaning and pronunciation.

Nov 18 Session

-Looked at the Kata Practice Feature, specifically the Reading practice.

-Tested various settings, from only the あ and ん series, to all of the hiragana series at once. Also tested only all of the Katakana series, and every available series.

-The mistakes counter acted oddly, as when I generated a new test after making any mistakes, the counter was not reset to zero

-The じゃ series acted oddly as well, rejecting the input “jya” as incorrect, and only accepting “ja”.

-Also looked at the writing feature

- The lines that are generated after the left mouse button is selected and held down follows the pointer well, and is able to generate curves and loops

-Stroke order is important – the kata cannot be found if the strokes are made out of order

-It can be difficult to distinguish between large, stand-alone letters, and letters that are meant to denote glides.

**Security Testing Overview:**

For the application Zkanji, I have determined that security will most likely not a.) be testable in the limited timeframe I have been given, and b.) be a problem. To begin with, Zkanji only connects to the internet when the user asks the program to check for updates. If an update is found, it is then summarily installed. Because of this, there is not a high risk of the program being attacked over any network. Furthermore, one would have to be extremely malicious to create a security problem.

If I had the time and the expertise, I would write a program that would intercept the request to check for updates, and redirect it to a new location through various means. If I were able to complete this task, theoretically I would be able to tell it that there is a new update waiting to be installed, and then proceed to have it download my own, masked, malicious package; thus creating a security issue. However, as stated previously, this would be a very unlikely case, and thus not worth researching and testing for in the immediate timeframe I have been given.