# CMSC 447

# Fall 2022

# Wordle Coach Testing Report

Wordle Coach

Testing Report

#### Table of Contents

Page

1. Introduction 3

1.1 Purpose of This Document 3

1.2 References 3

1. Testing Process 4

2.1 Description 4

2.2 Testing Sessions 5

2.3 Impressions of the Process 5

1. Test Results 7

Appendix A - Peer Review Sign-off 13

Appendix B – Document Contributions 14

### 1. **Introduction**

* 1. Purpose of This Document

The purpose of this document is to outline several test cases for the Wordle Coach that are to be implemented in our application. It also will describe our testing process, testing sessions, impressions of the process, and all results of our tests. This document is intended for developers and testers who are to perform the required tests and report any errors that arise as a result.

* 1. References

Team Odin. *Wordle Coach System Requirements Specification*. UMBC CMSC 447, 2022. Accessed 1 Dec. 2022.

Team Odin. *Wordle Coach System Design Document*. UMBC CMSC 447, 2022. Accessed 1 Dec. 2022.

Team Odin. *Wordle User Interface Design Document*. UMBC CMSC 447, 2022. Accessed 1 Dec. 2022.

2. **Testing Process**

* 1. Description

The ideal process would be that everyone can just work on the project at least a few hours every day, and test as we code so that we can get ahead of schedule and then dedicate any remaining time we can to meet Jamie's needs in terms of aesthetics and features and it can provide flexibility for everyone to maintain their other priorities. Jamie being the customer would just message Zan anything that he would be concerned about and then Zan would distribute that information to the remaining members of the team so that we could split the tasks evenly. We were also going to meet every day in order to be very up to date with the project. We ended up splitting the use cases evenly based on everyone’s schedule in terms of classes and work since everyone has vastly different schedules throughout the timespan of the project. We would have team meetings on our discord at least twice a week in order to share status updates in terms of progress on the project along with voicing out issues that needed to be fixed. Plus, it wouldn’t be redundant if there was no progress made. Also, Jamie came to every team meeting in order to check on the progress and mention anything that he wanted for the app to have rather than just mentioning it to Zan so that everyone will know his concerns, needs, input, etc. at the same time.

Zan would start off the meeting by asking our progress and then comparing it to the schedule. Everyone one by one would describe their progress in terms of the use cases. If there was any issue that needed to be addressed in the meeting, we would hear them out and take time in the meeting to help debug the issue and solve it. Everyone would come together and help each other out if we were stuck on a particular issue. For the most part, open help menu and scroll through words was smooth since there weren’t many cases to consider in the first place but insert word and undo guess were tricky. Even though undo guess didn’t have a lot of cases to test for, the interaction of the classes in the android environment made it difficult to run even though conceptually the function worked but it physically didn’t due to the class interactions from the android app environment. Insert word was difficult due to the keyboard initially not displaying so the word wasn’t showing on the UI. Give probability had a small issue in terms of the probability because it was off by two decimal places but that was fixed relatively quickly. Reset wordle board was simple after reflecting on the experience from undo guess.

Our main unit testing strategy was to test “as we go”. After any change was made, the app build process was run to test that there were no compilation errors. When the app build was successful, a test of the changes or new implemented feature was conducted. Additionally, a cursory system test was performed as a pseudo regression test of the preexisting features of the app. The details of these tests are enumerated in the testing suite. Prior to the demonstration, a full in-depth system test was performed

2.2 Testing Sessions

For each testing session, specify the date, location, time started, time ended, who performed the tests, and which use case(s) was covered.

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| Date | Location | Time Started | Time Ended | Performer | Use Case |
| 10/05/2022 | Team meeting discord | 7:21 PM | 10:06 PM | Zan Wills | Insert Word |
| 10/14/2022 | Team meeting discord | 2:49 PM | 4:45 PM | Parth Patel | Open Help Menu |
| 10/19/2022 | Team meeting discord | 7:02 PM | 8:32 PM | Jamie Kirk | Give Probability |
| 10/30/2022 | Team meeting discord | 8:11 PM | 9:47 PM | Dennis Mayher | Scroll through words |
| 11/7/2022 | Team meeting discord | 6:39 PM | 7:58 PM | Nathan Hoernlein | Undo guess and Reset Wordle bard |
| 12/4/2022 | Team discord meeting | 3:30 PM | 6:47 PM | Zan Wills | System test |

2.3 Impressions of the Process

The testing process that the team performed seemed to be effective at finding bugs. Testing the application on different occasions and testing different functionalities helped to determine any specific bugs associated with specific functionality. Another important step in the process was doing a rigorous full system test. Being able to test different functionalities in the same session helped to understand if the system worked as a whole and if any components were interfering with each other, or if any interfaces between classes and functions were misbehaving.

The best modular units in the program were the modules implementing the input of the text and color as well as non-actionable popup screens. There is little to go wrong with these sections of code since they are just typing in text and changing the color which can only take three values. These modules consist mostly of preexisting code provided in Android Studio and java libraries which gives it a lower likelihood of failure. Another thing to mention is that most of the popup screens are as simple as pressing the button which triggers the listener, and it comes up with the screen that just says exit (non-actionable). There is only one route that this functionality can take, and it has little wiggle room for failure.

The worst modular units in this program in terms of higher likelihood of failure rely on the algorithm backend portion and how the back end is called. This high likelihood of failure is due to the fact that there is a lot more going on in the algorithm than the user interface and more variables to cause issues. The back-end calls store a lot of information and performs mathematical analysis. This mathematical portion could potentially cause problems, as it is the one of the modules comprised entirely of custom code. Another thing to mention is how the data from the back end is taken and stored on the UI. There is a risk that the number of views or possible data taken could cause memory issues or invalid data issues. Although thoroughly tested and analyzed there is always a small risk involved.

After thoroughly testing out app most of this holds true, however the data being displayed has seemed to cause little to no issues at all. The method of creating new views and adding them to the scroll section has not caused any memory issues even when running on a physical Android device.

3. **Test Results**

Insert word edge/boundary cases: Nathan Hoerlein

* No letters are entered (and colors may or may not be entered)
* No letters are entered, and colors are entered
* One or more letters are missing from the entry
* One or more letters are missing from the entry, and a color is entered under a missing letter
* The letters do not form a word in the word list
* The user selects 4 green colors and 1 yellow color for the letters (this is impossible)
* The user tries to enter a 7th guess

Tests for Insert word use case:

* NOTE: for input data: 0 = GRAY, 1 = YELLOW, 2 = GREEN

|  |  |  |  |
| --- | --- | --- | --- |
| Purpose/Description | Input Data | Prior Inputs | Expected Output |
| Each letter is filled and all the letters form a word, the user is not on their last guess and no colors are selected | “CRANE”  0 0 0 0 0 | None | Accepted, no error message. This will get placed in the next appropriate guess box. All suggested words will not contain any of the letters the user entered |
| Each letter is filled and all the letters form a word. At least one letter is chosen to be green. | “CRANE”  2 0 0 0 0 | None | Accepted, all suggested words should contain only words that have the green letter(s) in that spot |
| Each letter is filled and all the letters form a word. At least one letter is chosen to be yellow. | “CRANE”  1 1 2 2 2 | None | Accepted, all suggested words should contain only words that have the yellow letter(s) NOT in that spot |
| One or more letters are missing from the entry | “\_RANE”  “A\_A\_A”  “\_ \_ \_ \_ \_” | None | Rejected, an error message will pop up saying the word is incomplete |
| All letters are entered, but they do not form a word in the Wordle word list | “XYZKA” | None | Rejected, an error message will pop up saying the word is not in the word list |
| The user selects 4 green colors and 1 yellow color for the letters. | “CRANE”  1 2 2 2 2 | None | Rejected, no words would be able to be suggested. |
| The user enters their guess and all their letters are green | “CRANE”  2 2 2 2 2 | None | Accepted, a message would tell them that they got the Wordle |
| The user enters their last guess and not all the letters are green | “CRANE” 1 1 1 1 2 | None | Accepted, a message would tell the user that they failed to get the word |
| The user attempts to enter a 7th guess | “CRANE”  2 2 2 2 2 | Guesses 1-6 | Rejected, there would be no place to put guess 6 and the user would be asked to clear their guesses or undo a guess |

Give probability edge/boundary cases: Parth Patel

* If the user enters a guess with all green letters, only the guess they entered should show up with 100% probability
* If no words can be predicted using all the guesses the user has entered, the output should display this
* If the user enters a guess (and words are recommended), the probability of each word should be calculated by taking the percent frequency of a letter appearing in each spot in the word

Tests for give probability

* NOTE: for input data: 0 = GRAY, 1 = YELLOW, 2 = GREEN

|  |  |  |  |
| --- | --- | --- | --- |
| Purpose/Description | Input Data | Prior Inputs | Expected Output |
| The user enters a guess with all green letters | “CRANE”  2 2 2 2 2 | None | The dialog box with the words and the probabilities should only display the current guess with 100% probability |
| The user enters a guess, but no words can be recommended (after several guesses) and no probabilities can be generated | “PARES”  1 0 0 1 2  “TEMPS”  2 1 0 0 1 | None | There should be no words in the dialog box, a message should show up saying “There are no word recommendations. Perhaps you made a mistake? Press ‘undo guess’ or ‘start over’.” |
| The user enters a guess, and there are recommended words | “RATES”  0 1 0 1 0 | None | the probability of each recommended word is determined by taking the percent frequency of a letter appearing in each spot in the word |

Open help menu edge/boundary cases: Jamie Kirk

* If the user clicks on the help button, it will display a popup that shows/tells the user how to use the app

Test for open help menu

|  |  |  |  |
| --- | --- | --- | --- |
| Purpose/Description | Input data | Prior Inputs | Expected Output |
| To show the user the help option and display popup on how to use the app | User presses the help menu button | N/A | popup on how to use the app is displayed |

Scroll through words edge/boundary cases: Dennis Mayher

* If the best possible guesses are lower than 25 words, then display that number of words (ex: user inputted 2 guess and reduced the best possible words list from 25 words to 7, the scroll best words function will display the 7 best words)
* If the best possible guesses are higher than 25 words, display the best 25 words based on percent of frequency.

|  |  |  |  |
| --- | --- | --- | --- |
| Purpose/Description | Input data | Prior Inputs | Expected Output |
| To display the best amount of possible word choices for the user to solve the wordle after the user enters at least one guess | The user enters a guess | none | display that number of words (ex: user inputted 2 guess and reduced the best possible words list from 25 words to 7, the scroll best words function will display the 7 best words) |
| To display the best amount of possible word choices for the user to solve the wordle | The user enters a guess | none | display the best 25 words based on percent of frequency |

Undo guess edge/boundary cases: Zan Wills

* If the user doesn’t have a guess already attempted and tries to click undo guess, it won’t do anything
* If the user has attempted a guess and clicks undo guess, then it will prompt the user to confirm the action. If they confirm, it gets rid of the last guess entered and goes back to that guess attempt. If they don’t confirm, nothing happens.

|  |  |  |  |
| --- | --- | --- | --- |
| Purpose/Description | Input data | Prior Inputs | Expected Output |
| To help the user go back to their last entered guess for strategic purposes. The user will click on the button after at least one guess was made on the board and they confirm undoing the last entered guess. | The user clicks on the button | One guess was entered on the board at least | The last entered guesses is erased and it goes back to that attempt ex: user has entered 3 guesses and they click undo and confirm prompt, it erases the 3rd guess and only 2 guesses are displayed |
| To help the user go back to their last entered guess for strategic purposes. The user will click on the button after at least one guess has been made on the board and they deny the prompt of confirmation. | The user clicks on the button | One guess was entered on the board at least | The board will just remain as it was and it will act like nothing happened |

Reset Wordle board edge/boundary cases: Parth Patel

* If the user doesn’t have a guess already attempted and tries to click reset wordle button, it won’t do anything
* If the user has attempted a guess and clicks reset wordle button, then it will prompt the user to confirm the action. If they confirm, it resets the entire board and the board will be empty and then the user can make their first guess. If they don’t confirm, nothing happens

|  |  |  |  |
| --- | --- | --- | --- |
| Purpose/ Description | Input data | Prior Inputs | Expected Output |
| The user wants to clear the entire board. They click on the reset wordle board button, confirmation prompt appears, they accept. | The user clicks on the button | One guess was entered on the board at least | It will prompt the user to confirm the action. If they confirm, it resets the entire board and the board will be empty and then the user can make their first guess |
| The user wants to clear the entire board. They click on the reset wordle board button, confirmation prompt appears, they deny | The user clicks on the button | One guess was entered on the board at least | The board will just remain as it was and it will act like nothing happened |

Test Results

Zan –Undo Guess use case. Undo guess didn’t go exactly as planned since the class interaction with the function and the app environment seemed to be more complicated than our initial impression. Basically, the previous guess that the user would have entered would be gone but it didn’t allow for them to reenter that guess. Apparently, after undoing the guess, it somehow was erased from the word list after undoing the guess so there had to be constraints placed in terms of not jeopardizing the word list after undoing the guess. The dialogue builder never loaded when first testing and that was because the class for the prompt was its own class and thus could not make changes within the MainActivity class. To have the Dialogue class act on member variables of the MainActivity class, the Dialogue class had to declare a listener, then MainActivity had to implement the listener. This issue was also found in the start over Dialogue and had the same solution.

Nathan – Scroll through words – at first the user could not scroll through the list even though all the words were able to be displayed. However, we found a piece of code online that allowed us to scroll very well with our on click listener whenever the user swiped through our box class that kept the list of words.

Dennis- Reset Wordle Board- This was the first piece of functionality implemented in which the app had to take an action based on user input from a dialogue prompt. Initially this functionality did not work because the dialogue box executed asynchronously with relation to the main reset board (“Start over”) function. This was resolved after declaring a listener within the dialogue class, and having the main activity implement the start over behavior. The behavior itself was very simple as it was essentially just rerunning the same code onCreate() runs when the app launches.

Parth- Open Help menu- The prompt never loaded when first testing and that was because the class for the prompt was its own class and not inherited from the environment class. Apparently, we had to have the on clickers from the app environment to interact with the application in order to display the popup without altering the dimensions of the app.

Jamie- Give Probability – the percentages were calculating correctly with the algorithm by adding up the probabilities that a letter would occur in each spot, the only issue was that they were displaying two decimal places too far in (for example, 0.4567% instead of 45.67%). To get around this, we multiplied each probability by 100 to get the percentage of each recommended word after a user entered a guess. One issue to note as well that was not fixed was that the way the words were sorted by probability, lead to words with duplicate letters, with high probabilities, being placed low on the list with low probability words. Ultimately, this turned out to be more of a display bug than a functionally bug because choosing words with repeated letters is statistically almost always a bad choice in wordle if it has not been confirmed that the word has repeated letters. The correction would be to display these words with a percentage that more accurately reflected their usefulness.

**Appendix A – Team Review Sign-off**

This sign-off agreement confirms that all members of the Wordle Coach development team have both reviewed and agree with the provided content. Each member who has signed this has validated the testing specifications and agrees that this work will be submitted as a team effort.

Signature: Date:

Zan Wills 12/11/2022

Jamie Kirk 12/11/2022

Dennis Mayher 12/11/2022

Parth Patel 12/10/2022

Nathan Hoernlein 12/10/2022

**Appendix B – Document Contributions**

|  |  |
| --- | --- |
| Name | Role |
| Zan Wills | Insert Word/Undo guess testing suite |
| Dennis Mayher | Testing impressions (transcribed from group members) |
| Nathan Hoernlein | Give Probability/reset wordle board testing suite |
| Parth Patel | Testing Sessions/ open help menu testing suite |
| Jamie Kirk | Testing Description |