Word Hunt Test Report

Baibhav Barwal
Union College
barwalb@union.edu

Neil Daterao
Union College
dateraon@union.edu

Simon Gray
Union College
grays2@union.edu

Jacob Schuckman
Union College
schuckm@union.edu

March 2024

We abide by the honor code principles!

1 Introduction

This document presents the testing procedures, methodologies, and results for the word hunt game. Our objective was to ensure the reliability, performance, and user-friendliness of the application through comprehensive backend and user interface (UI) testing.

2 Backend Tests

We have JUnit tested all three files in the model. Each of the test details is explained in the upcoming table 1 to 3.

3 UI Tests

Over the following sections(figure 1 to 5), we will describe a few User Interface testings along with a diagram explaining different design options.

Test Name	Test Description
testGenerateRandomBoard	For a random board with inputted size, we test the number of rows and columns
testIsValidMove	Tests by using the isValidMove such that the return should/not be true
testGetFoundWords	Test when isFoundWord is/is not empty
testGetFoundBonusWords	Test when isFoundBonusWord is/is not empty
testIsValidWord	Testing with the random input word and what flag number it returns
testAddFoundWord	Manually testing for WORD in the possible FoundWord from the board
testGetPossibleWords	Testing if the possible word is/is not null
testTearDown	Testing if the board is empty/not empty after isEmpty
testFoundWordBonusFlag	Testing by inputting a 5-letter word
testGetEmoji	Testing by inputting an emoji with a description of the emoji

Table 1: Backend Test Descriptions for WordHuntGameTest.java

Test Name	Test Description
testGetFoundWords	Testing, if get found word, is created and is initially empty. Next, added a word into the FoundWord manually and tested if it was contained in the database
testGetNumFoundWords	Adding one word and testing if the size of the data structure is 1
testTearDown	Testing if there are zero found words after the function called

Table 2: Backend Test Descriptions for WordHuntScoreTest.java

Test Name	Test Description
testIsValidWord	Passing a non-word and testing if it returns 0 or not
testGetPossibleWords	Testing the initial condition of possibleWords if it's empty or not
testGetFoundWords	Testing if the foundWords data structure is null or not
testGetNumFoundWords	Testing if the number of foundWords
testGetFoundBonusWords	Tests if the foundBonusWords initially is empty or size 0 or not
testAddFoundWord	After adding a word to the foundWord data structure, testing if it has size incremented and .contains(word) returns true or not
testFoundWordBonusFlag	Testing by inputting a 5 letter word
testTearDown	Testing if there is zero found words after the function called

Table 3: Backend Test Descriptions for WordHuntWordTest.java

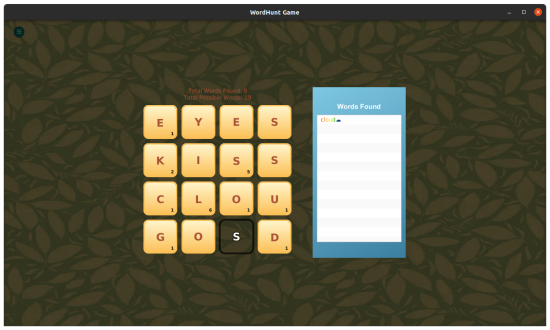

Figure 1: The above diagram shows the emoji being popped for the cloud word.

Figure 2: The above diagram shows the front loading screen of the diagram. All three tabs *NewGame*, *LoadGame*, *Quittodesktop* are tested for UI purposes.

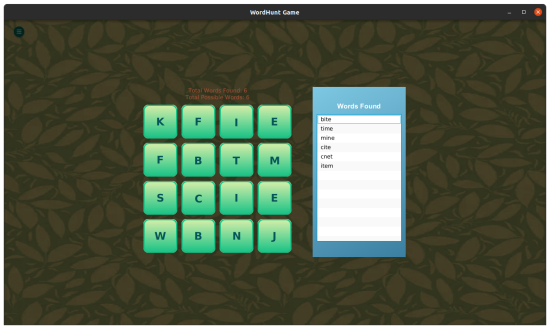

Figure 3: When the game is over, the state of all tiles changes to green.

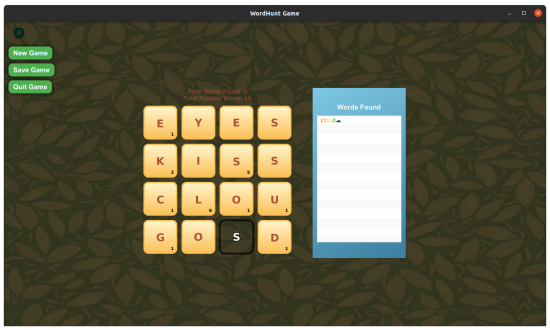

Figure 4: The in-game interactive menu has three options, new game, load game, and quit and all of which are functional.

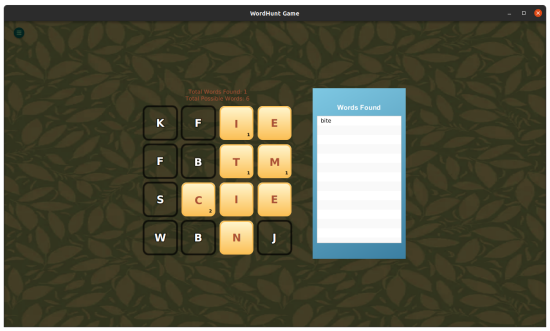

Figure 5: Tiles that will never be used in the game are greyed out as shown in the diagram above