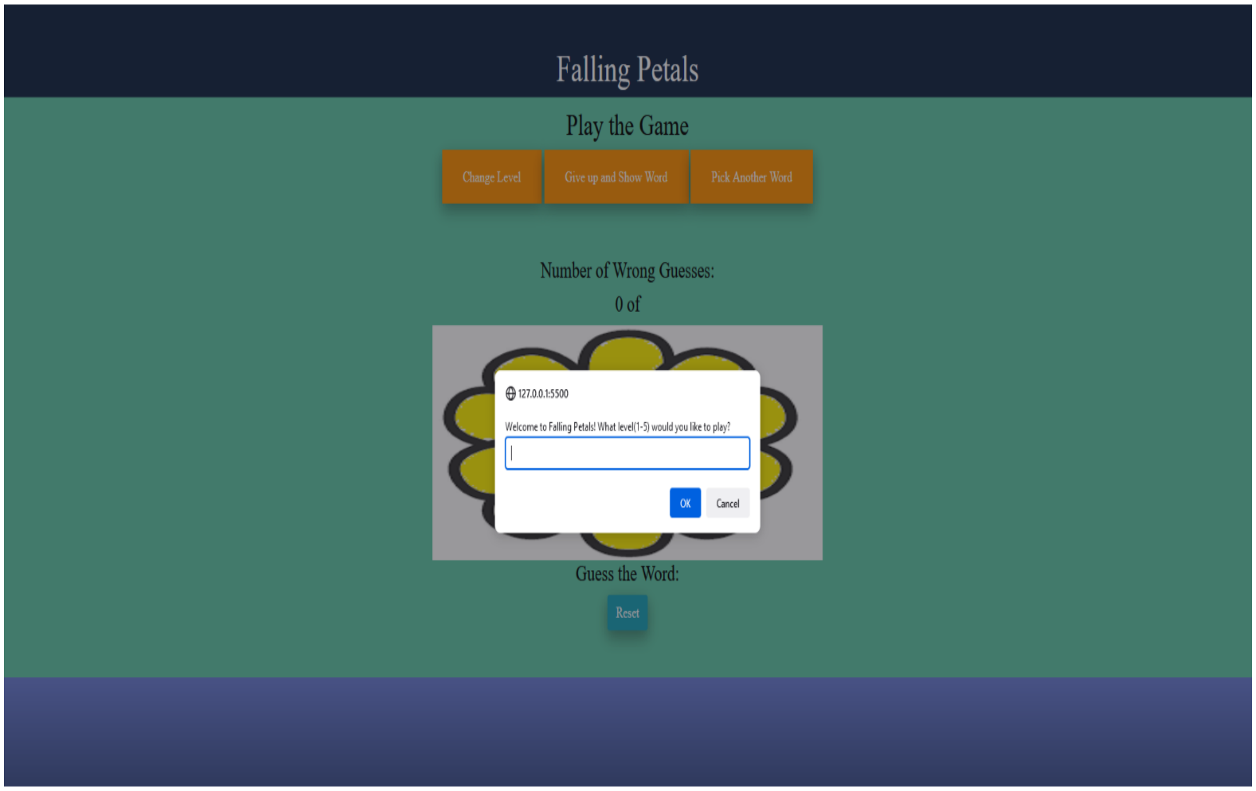
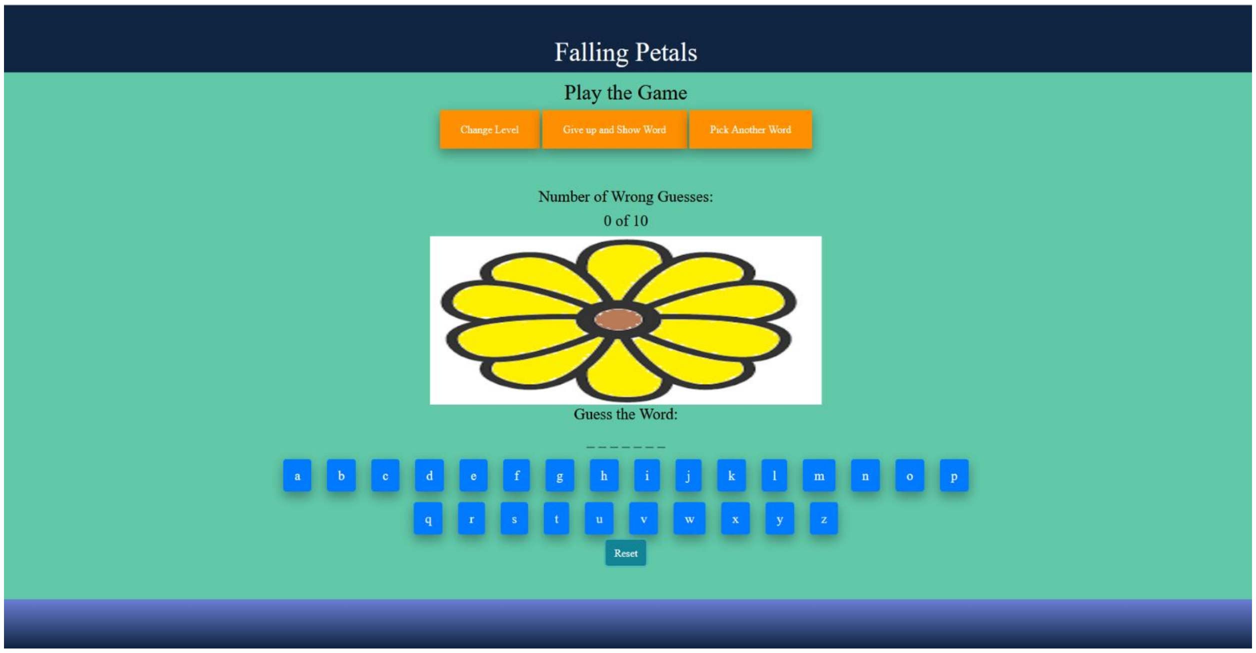
1. Scope AND Purpose

To make a kid friendly version of the game "Hangman". This is aimed to be created as an educational tool for teachers to use in a noncompetitive, classroom setting. In addition, to give players more opportunity to correctly guess the word in comparison to a regular game of Hangman.

Screenshots of the Program in Use:

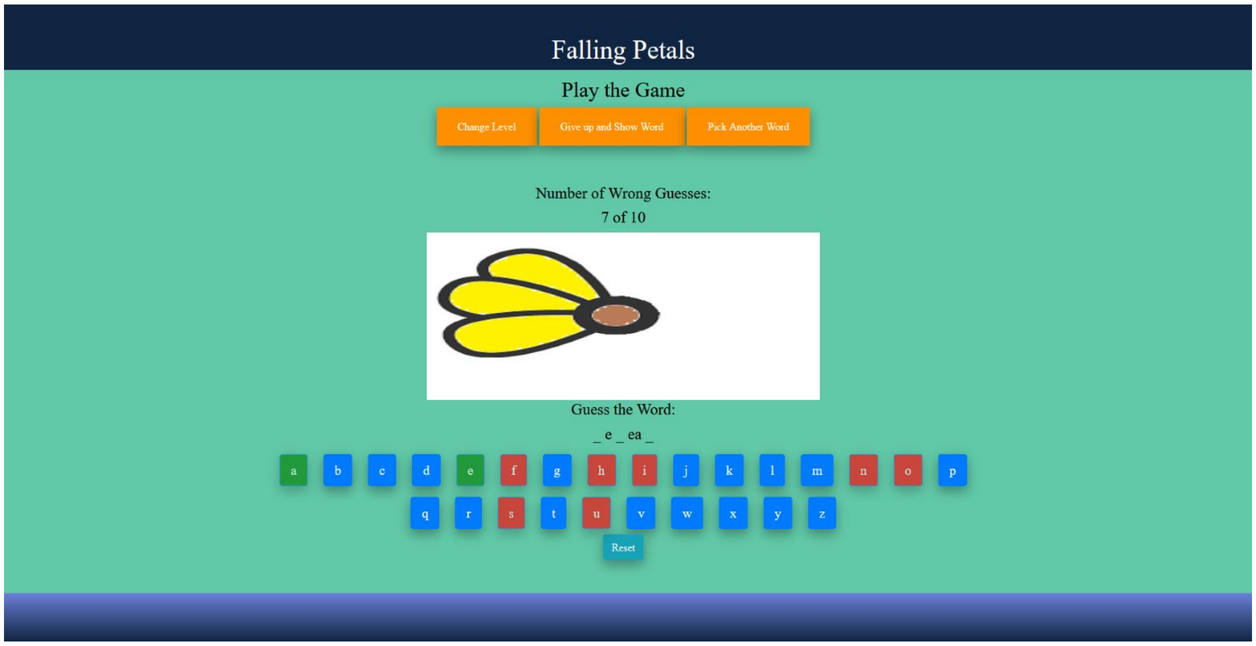


Welcome to Falling Petals! What level(1-5) would you like to play?



Number of Wrong Guesses:
0 of 10

Guess the Word:
_ _ _ _ _ _ _

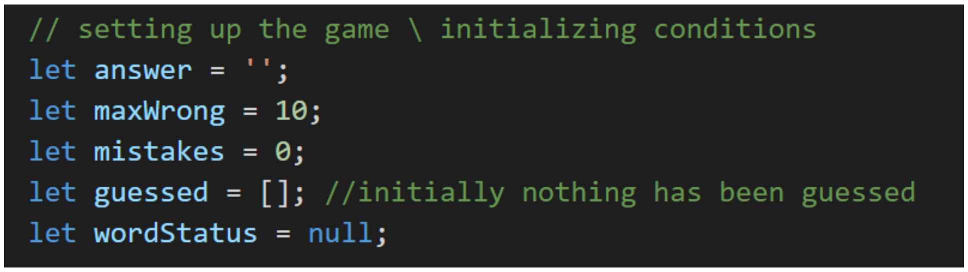


Number of Wrong Guesses:
7 of 10

Guess the Word:
_ e _ ea _

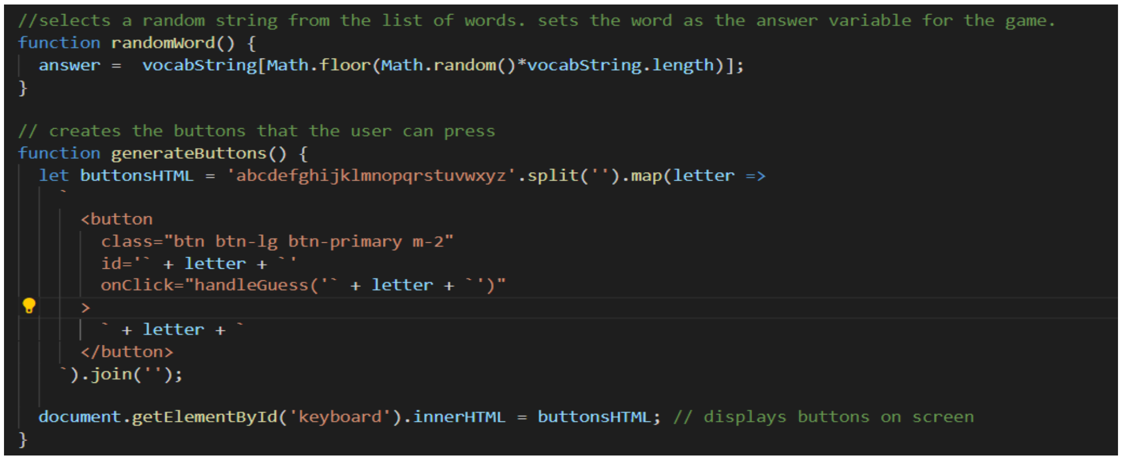
2. Detailed Description of the Application and its features

The game starts off with a prompt for the user to input a number 1-5 by the user to pick what level they would like to play. With the information input by the user the game then uses an array structure to hold a number of string elements. These string elements represent the levels/string of vocabulary words that are held in the word bank of each level. Then the program uses Math. random() to randomize the word that was selected from the string. The initial conditions are setup for the game



```
// setting up the game \ initializing conditions
let answer = '';
let maxWrong = 10;
let mistakes = 0;
let guessed = []; //initially nothing has been guessed
let wordStatus = null;
```

and the randomly selected word from the string array is held in a variable called answer. There is a generated keyboard of buttons that are displayed on the screen for the user to make their letter guesses.

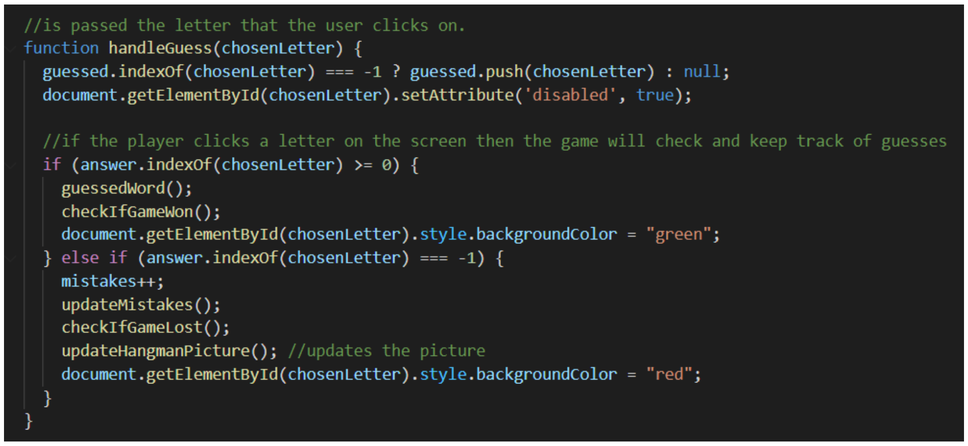


```
//selects a random string from the list of words. sets the word as the answer variable for the game.
function randomWord() {
  answer =  vocabString[Math.floor(Math.random()*vocabString.length)];
}

// creates the buttons that the user can press
function generateButtons() {
  let buttonsHTML = 'abcdefghijklmnopqrstuvwxyz'.split('').map(letter =>
    `
      <button
        class="btn btn-lg btn-primary m-2"
        id='` + letter + `'
        onClick="handleGuess('` + letter + `')"
      >
        ` + letter + `
      </button>
    `).join('');

  document.getElementById('keyboard').innerHTML = buttonsHTML; // displays buttons on screen
}
```

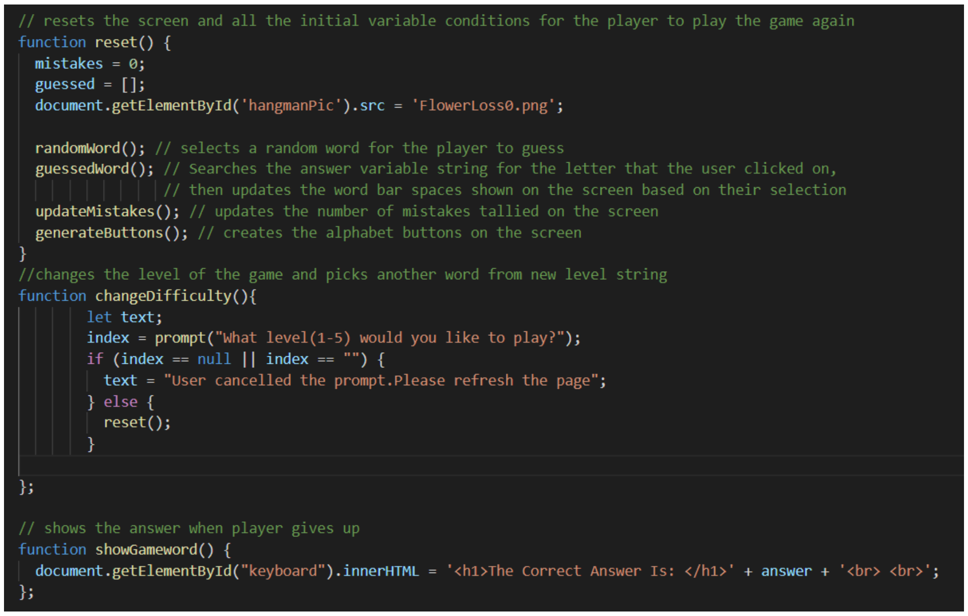
The user can attempt to click on letters that they think that the answer variable may be. Each letter that is clicked causes a call to a method that searches the word in the answer variable for the letter that is clicked. If it is a hit, then that button will turn green and if it is a miss then that button will turn red. The number of wrong guesses tallied on the screen will also be updated. If a correct letter is guessed, then the letter will appear in the word spaces displayed on the screen.



```
//is passed the letter that the user clicks on.
function handleGuess(chosenLetter) {
  guessed.indexOf(chosenLetter) === -1 ? guessed.push(chosenLetter) : null;
  document.getElementById(chosenLetter).setAttribute('disabled', true);

  //if the player clicks a letter on the screen then the game will check and keep track of guesses
  if (answer.indexOf(chosenLetter) >= 0) {
    guessedWord();
    checkIfGameWon();
    document.getElementById(chosenLetter).style.backgroundColor = "green";
  } else if (answer.indexOf(chosenLetter) === -1) {
    mistakes++;
    updateMistakes();
    checkIfGameLost();
    updateHangmanPicture(); //updates the picture
    document.getElementById(chosenLetter).style.backgroundColor = "red";
  }
}
```

There are 3 play menu buttons on the game top of the game screen. One button allows you to change the level of words that the game randomly pulls from. Another button allows you to give up and show the word. The third button on top of the screen, allows you to skip the current word and attempt another word from the same level. There is also a fourth button that allows you to reset the game located at the bottom of the screen.



```
// resets the screen and all the initial variable conditions for the player to play the game again
function reset() {
  mistakes = 0;
  guessed = [];
  document.getElementById('hangmanPic').src = 'FlowerLoss0.png';

  randomWord(); // selects a random word for the player to guess
  guessedWord(); // Searches the answer variable string for the letter that the user clicked on,
                 // then updates the word bar spaces shown on the screen based on their selection
  updateMistakes(); // updates the number of mistakes tallied on the screen
  generateButtons(); // creates the alphabet buttons on the screen
}
//changes the level of the game and picks another word from new level string
function changeDifficulty(){
        let text;
        index = prompt("What level(1-5) would you like to play?");
        if (index == null || index == "") {
          text = "User cancelled the prompt.Please refresh the page";
        } else {
          reset();
        }

};

// shows the answer when player gives up
function showGameword() {
  document.getElementById("keyboard").innerHTML = '<h1>The Correct Answer Is: </h1>' + answer + '<br> <br>';
};
```

The game uses a 10-petal flower and gives the player 10 incorrect guesses before the game terminates. Each new wrong guess makes 1 flower petal disappear from the picture. This continues until the flower runs out of petals or the user correctly guesses the word.

I used JavaScript, HTML, CSS, and Bootstrap as the application stack. I used Visual studio code as my IDE. And I used Github as the version control platform

3. Future Work

- Adding the ability to change the flower to other images
- Letting players create a user profile to play on their own outside of the classroom
- Implementing the ability to select a theme with your level of difficulty catered towards a specific class I.e., Science, English, etc.
- Letting the user input a specific word instead of picking from a string.