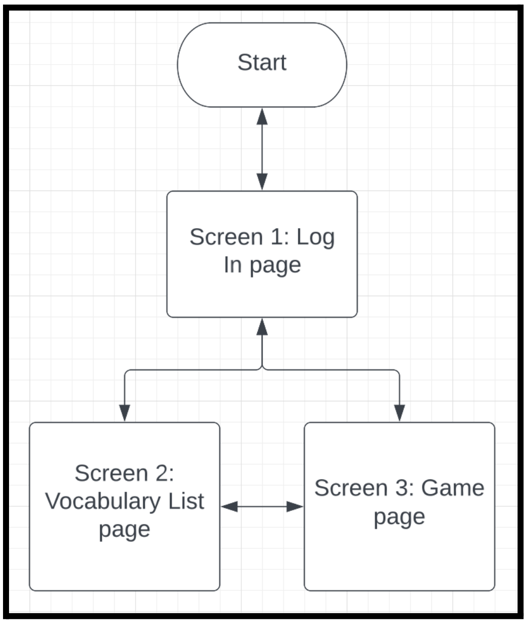
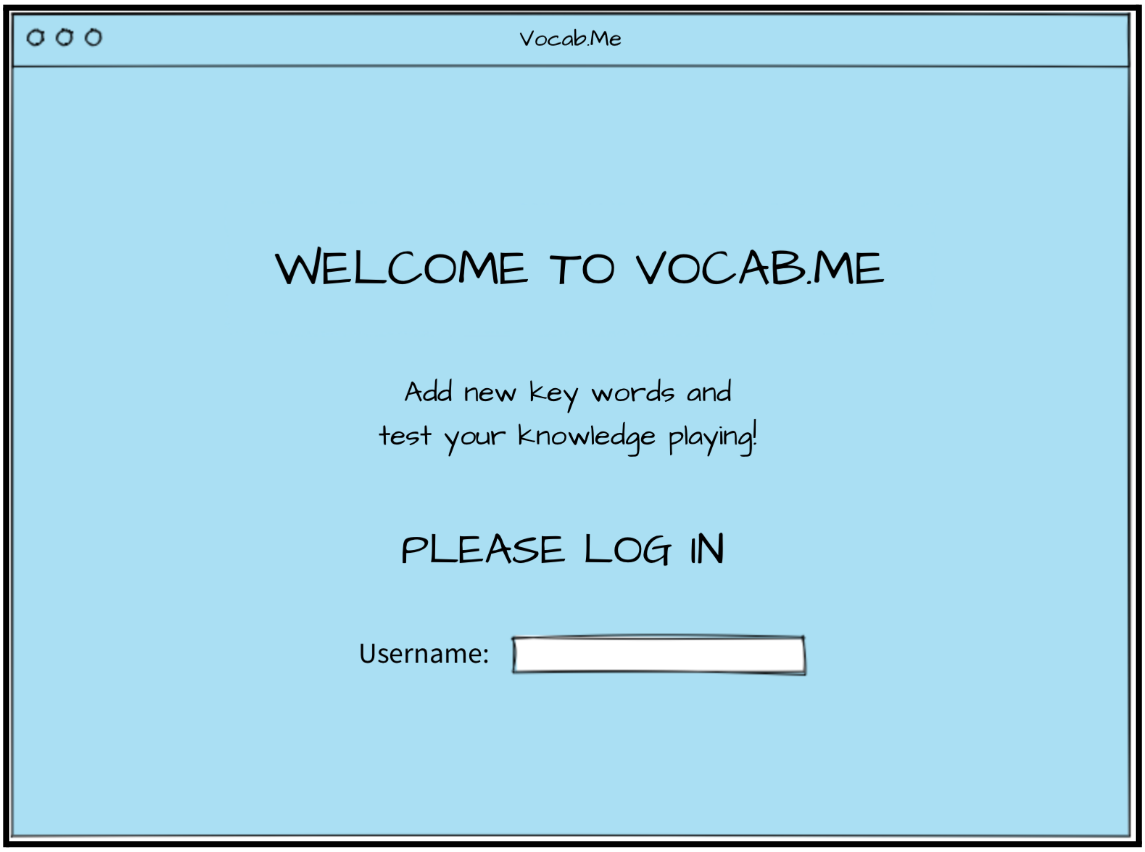
Criterion B: Design

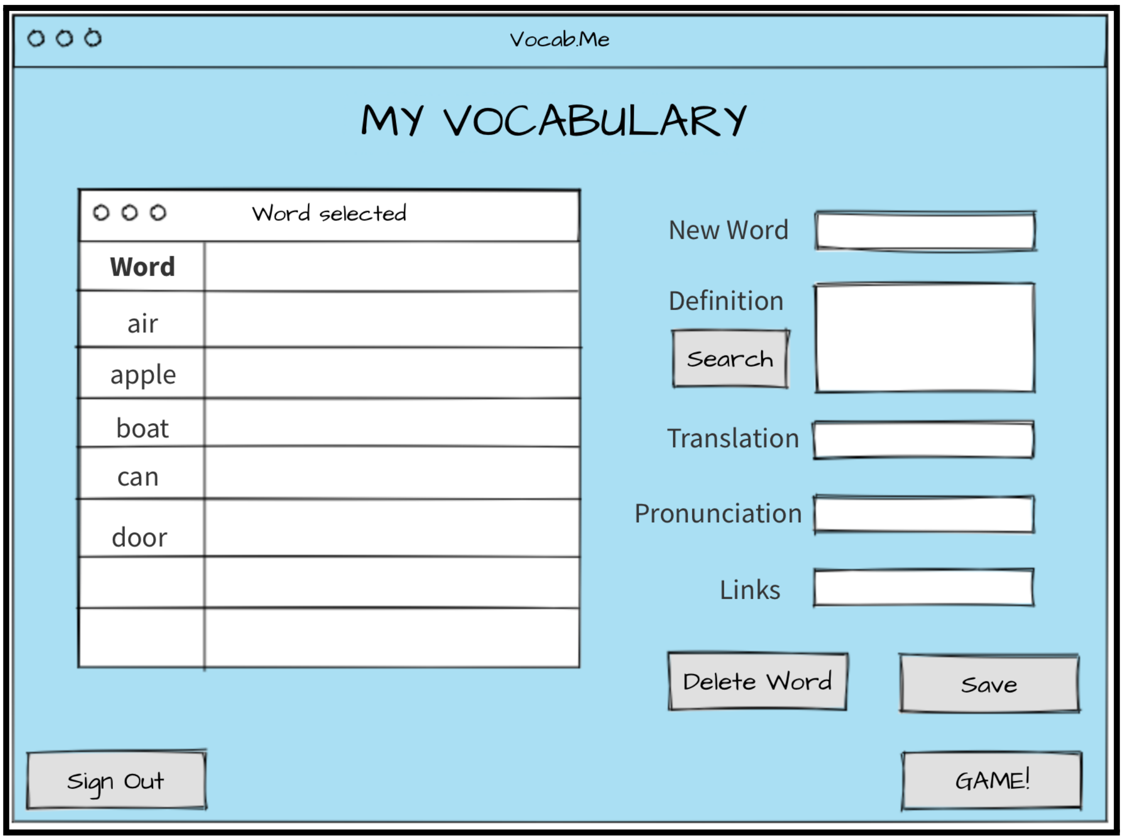
Program flow:



Screen designs:

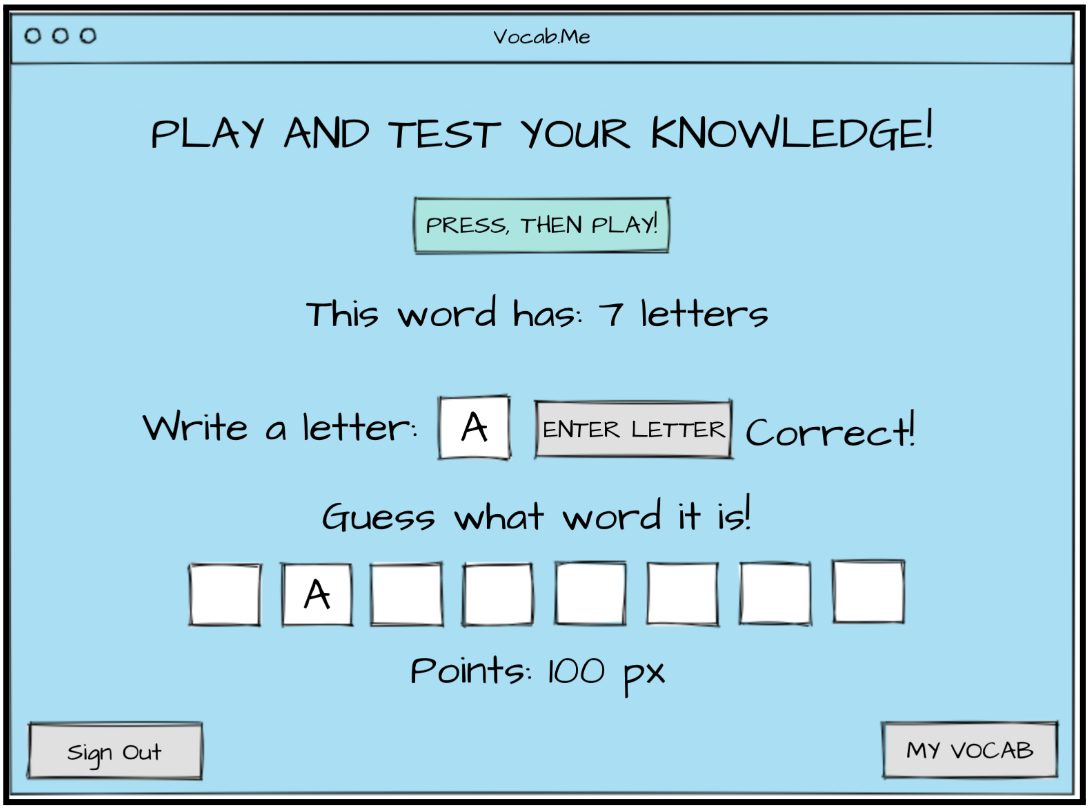


Screen 1: The first page that appears to the user when the program is opened is the login page. The user is supposed to input their username to access the rest of the program.



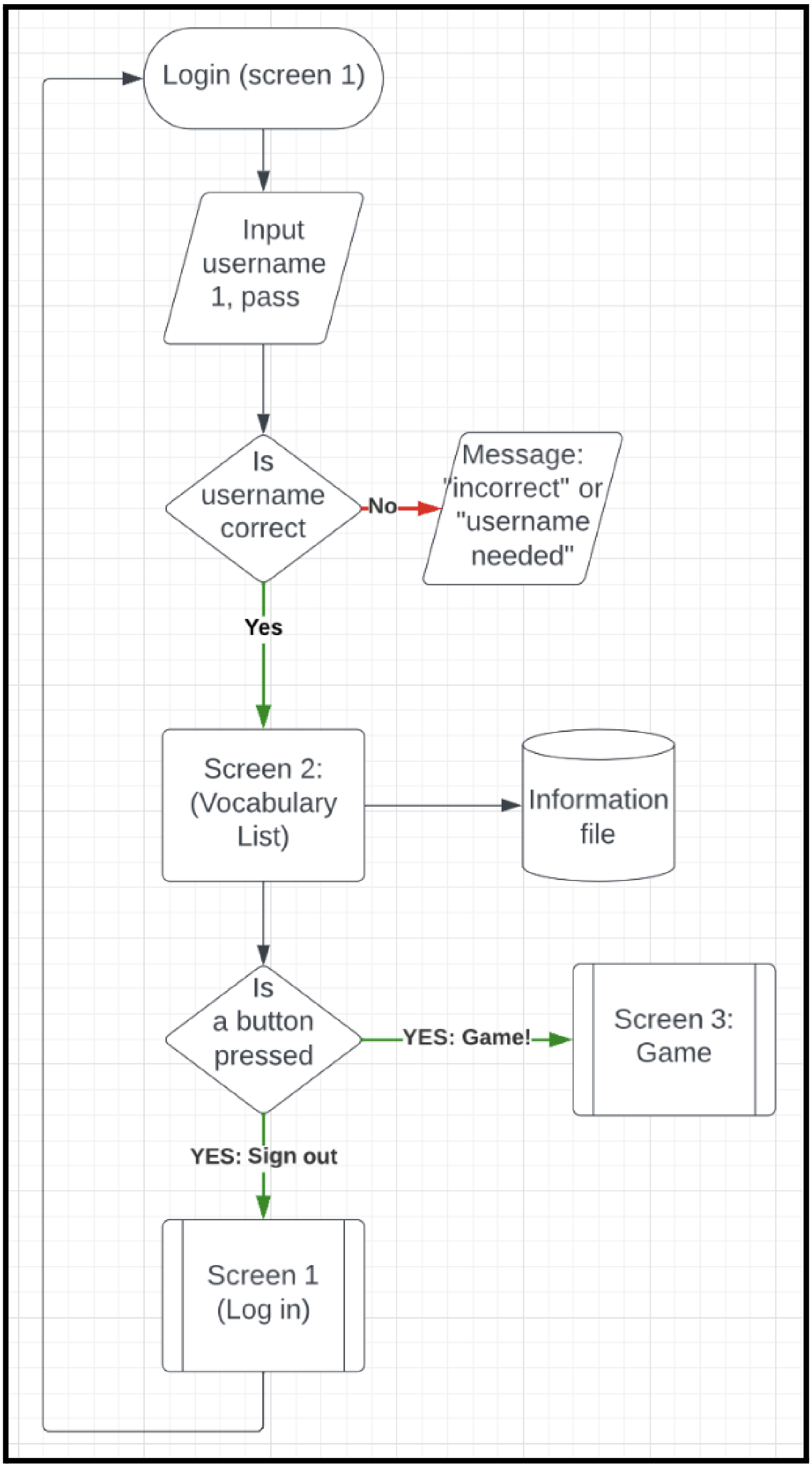
Word	
air	
apple	
boat	
can	
door	

Screen 2: In the vocabulary list page, the user can input words that can be saved/deleted in the JSON file according to the button they press. They can search for a word's definition, sign out of the program, or go to the game by clicking the buttons.

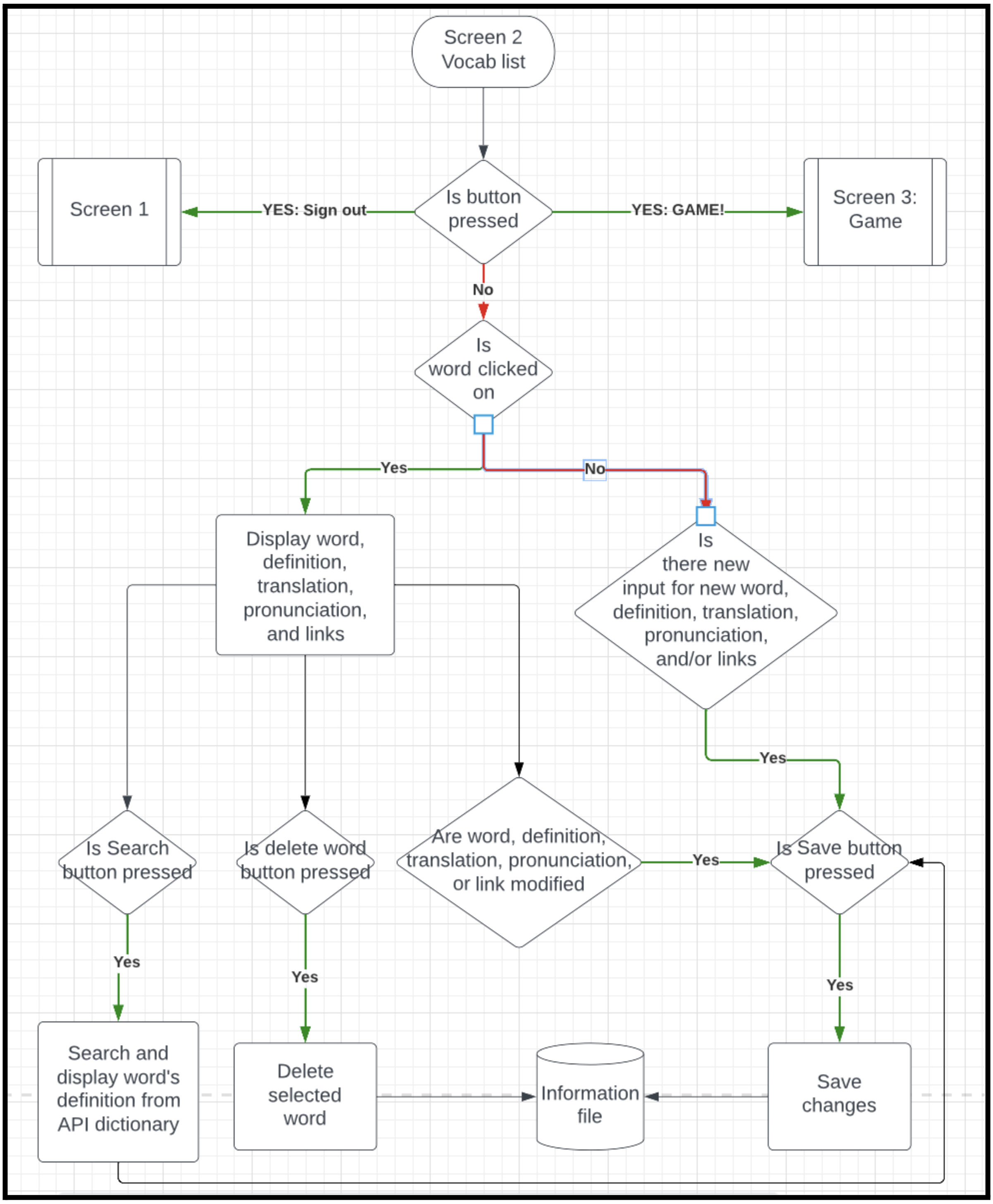


Screen 3: The game grabs a random word from the user's vocabulary list. The user needs to guess what the word is by inputting a letter at a time, and points will be awarded or subtracted accordingly. The program will say if the guess was correct or not.

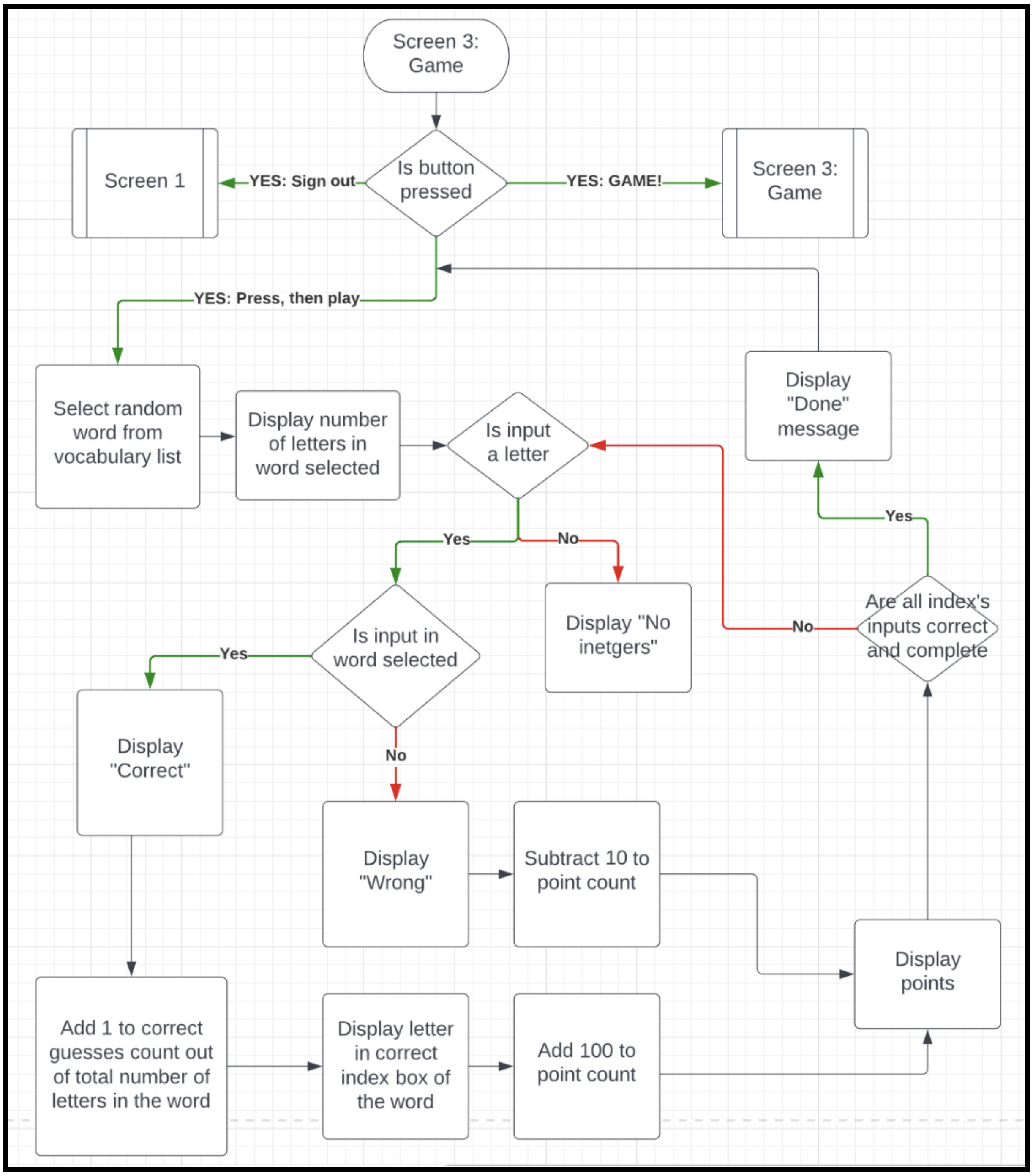
Flow Charts:



Flowchart 1: If the user inputs the correct username, the program redirects them to screen 2, if not, it asks for the correct username or a username in the case there was no input. The user can log out from the second screen or go to the game screen.


Screen 2
Vocab list
Is button pressed
YES: Sign out
Screen 1
YES: GAME!
Screen 3: Game
No
Is word clicked on
Yes
No
Display word, definition, translation, pronunciation, and links
Is there new input for new word, definition, translation, pronunciation, and/or links
Yes
Is Search button pressed
Is delete word button pressed
Are word, definition, translation, pronunciation, or link modified
Yes
Is Save button pressed
Yes
Yes
Yes
Search and display word's definition from API dictionary
Delete selected word
Information file
Save changes

Flowchart 2: If the user clicks the “Save” or “Delete” button, the textboxes’ inputs will be saved or deleted in the JSON file. If the “Search” button is clicked, the program gets a word’s definition from the API and displays it.

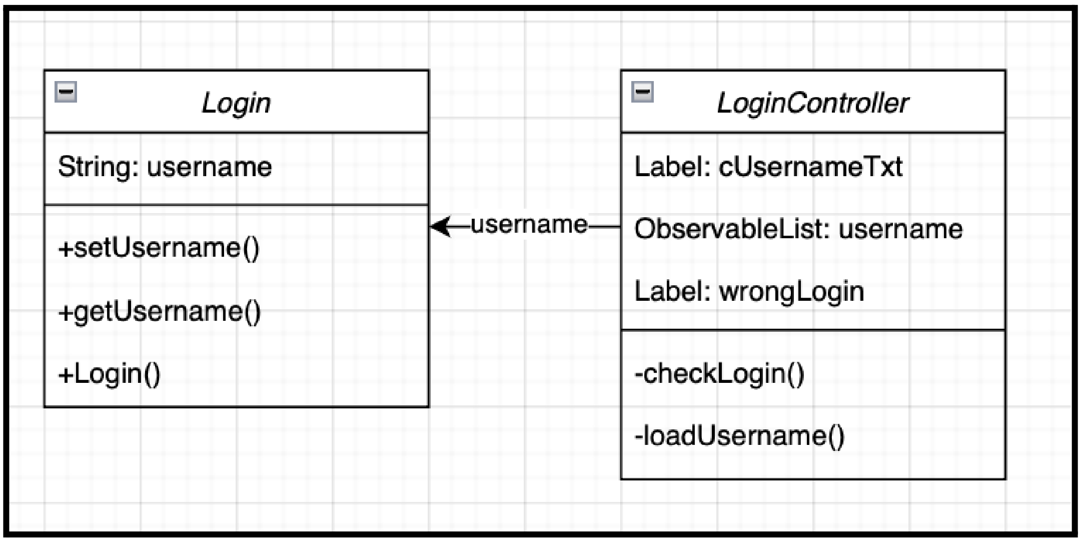

Screen 3: Game
Is button pressed
YES: Sign out
Screen 1
YES: GAME!
Screen 3: Game
YES: Press, then play
Select random word from vocabulary list
Display number of letters in word selected
Is input a letter
Display "Done" message
Yes
No
Yes
Display "No inetgers"
Are all index's inputs correct and complete
No
Is input in word selected
Yes
Display "Correct"
No
Display "Wrong"
Subtract 10 to point count
Display points
Add 1 to correct guesses count out of total number of letters in the word
Display letter in correct index box of the word
Add 100 to point count

Flowchart 3: The program randomly gets a word from the user's vocabulary list. A message will appear indicating if the guess was right or not, or if it's not valid (e.g integer) and points will be added or subtracted accordingly.

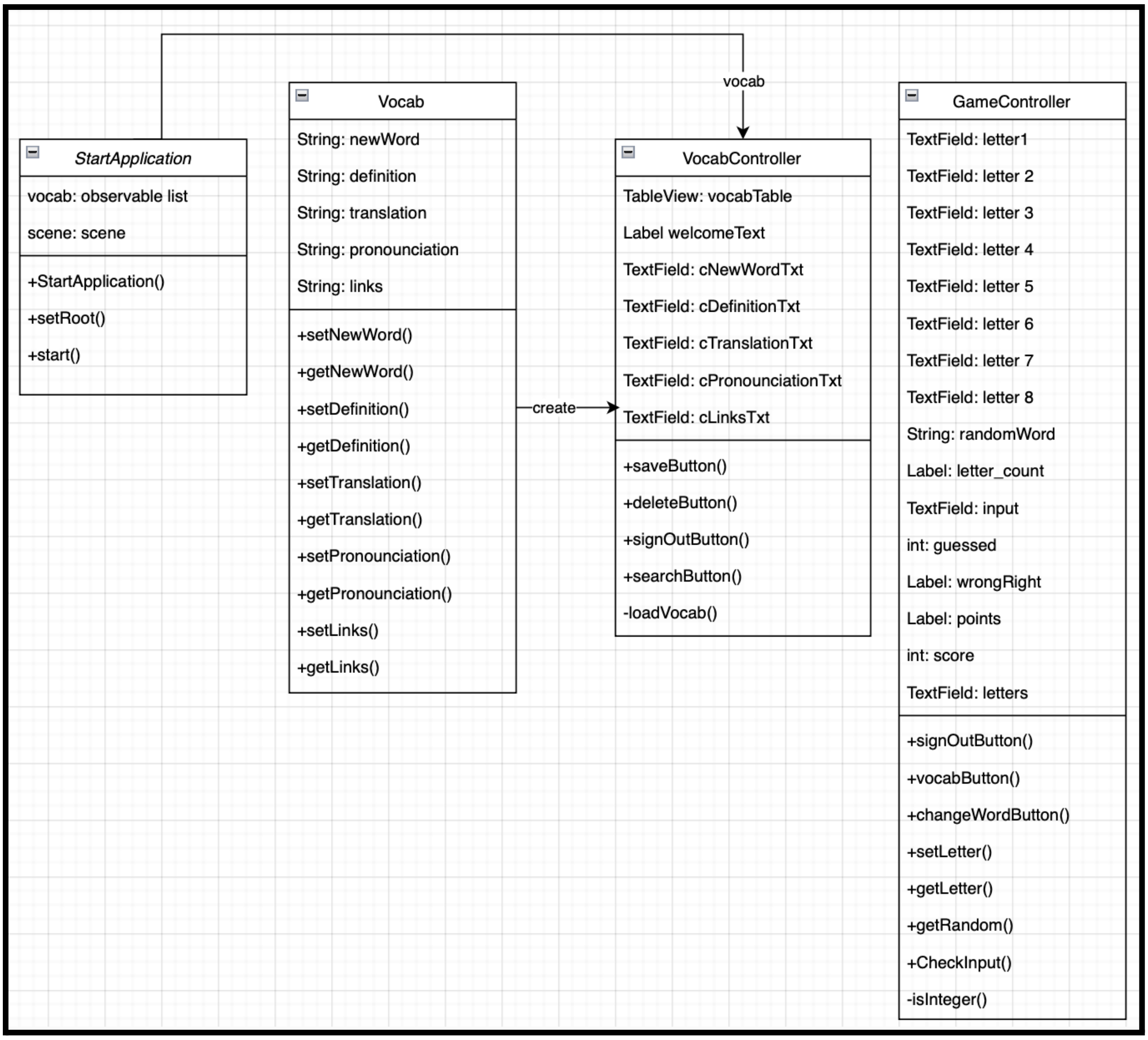
Pseudocode:

```
Unset
      //GAME PSEUDOCODE
      if word.contains(input) then
            output "Right!"
            score += 100
      else if isIneteger(input)then
            output "No integers"
      else
            output "Wrong"
            score +- 10
      end if
```

UML diagrams:



The program allows to set and get the username string from the Login class. The checkLogin() method in the LoginController class checks if the user’s input matches the username. The label displays a message indicating if it’s right or not.



The Vocab glass allows to set and get a word's attributes. The VocabController displays them in text boxes. The GameController has methods to check the user's input, display if it was right or not, and update the score.

JSON Data Structure:

```
Unset
{
   "newWord": "static",
   "definition": "lacking in movement",
   "translatioin": estatico,
   "pronounciation": "static",
   "links": "at rest"
}
```

Test Plan:

Colour code: Normal, abnormal and extreme data

Success Criteria	How I will test it	What I expect
The client can input new words, its definition, translation, pronunciation, and “links” to the vocabulary list.	Ask the client to input a vocabulary word, its definition, translation, pronunciation, and “links”.	Text can be written in the textboxes and is displayed.
The client’s inputs (vocabulary words) can be saved in the program.	Ask the client to save the data, quit the program, and open it again to see if the inputs were saved and can be displayed.	Display of the previously saved inputs.
The client is able to remove old vocabulary words. When removed, the word, definition, translation, pronunciation, and “links” will be deleted from the file.	Clicking the remove button and see if the inputs are deleted.	The old inputs are no longer displayed and are erased from the vocabulary list.
The vocabulary words are saved in an array, and sorted and displayed in alphabetical order for the client to find a word in an easier and faster way to improve time efficiency.	View the vocabulary list’s order of words.	The words are sorted in alphabetical order, displayed from top to bottom in a list.
A message appears in the controller if a new input/word already exists in the array.	Try to input an existing vocabulary word and see the controller’s output.	An “already exists” message appears in the controller if the user tries to save an existing word in the vocabulary list
When the client clicks on a word, the vocabulary words and its attributes are displayed.	Double click a word and see if its definition, translation, pronunciation, and links are displayed in their textboxes.	When a word in the list is double clicked, the word, its definition, translation, pronunciation, and links are displayed in their textboxes.

When the user clicks "search", an API dictionary's definition is displayed.	Input a new word and click the "search" button next to the definition text box and see if a definition is displayed.	When the "search" button is clicked, the word's definition from the dictionary API is displayed.
Have a game to test the clients memory on their vocabulary words. The goal is to guess the letters of the vocabulary words until it's completed.	Ask client to play the game: 1) The client inputs a correct letter in the word. 2) The client does not input a correct letter in the word. 3) The client inputs an integer. 4) The client finishes guessing all the letters in the word.	1) Notify "Right!" if the letter guessed is correct. 2) Notify "Wrong" if the letter guessed is not correct. 3) Notify "No integers" if the input is an integer. 4) Notify "Done" when all the letters in the word were guessed correctly.
The scores from the game are correctly and successfully added or subtracted after each input and displayed. 100 points are added for correct guesses, 10 points are subtracted for incorrect guesses.	Ask the client to input different letters to guess a word: 1) The letter guessed is in the word that the user tries to guess. 2) The letter guessed is not in the word that the user tries to guess.	1) 100 points are added to the user's score. 2) 10 points are subtracted from the user's score.
The buttons/text that is viewed by the user should have simple/short vocabulary, easy for a beginner English student to understand.	Ask the client to read the words in the buttons/textboxes and ask if they are simple and short.	The client easily understands what each button or textbox says.
The program should work on the client's laptop, it should be designed with an interface for a laptop.	Ask the client to test and run the program on their laptop and try to use it.	The program works correctly on a common laptop.

The program includes methods, if-else statements, and loops that are annotated with comments for further development.	The code in the IDE has an appropriate amount of annotations in each class next to each method, if-else statement, and loop.	In each class, next to each method, if-else statement, and loop, there is annotation of what the code does and its purpose.