The site performs the way it does because of multiple reasons. First of all, it is important to point out that I imported the AVL map from the textbook as mentioned in the instructions. After I did that, I realized that the AVL tree class file imported from the binary search tree class file, the binary search tree class file imported from the linked binary tree class file and map base class file, and so on and so forth. I ended up in a rabbit hole of searching for the files that each file I needed referred to, and ended up adding eight files in total from the textbook to my project folder.

The relationship between these files are based on the concept of inheritance. The AVL tree class inherits from the binary search tree class, which inherits from the linked binary tree class and the map base class, etc. If a child class has a method with the same name as the parent class, the child class's method overwrites the parent class's method. If the parent class has a method that is never overwritten in a child class, the child class would just inherit it and the method could be called when using the child class. I wanted to find some methods to call for the purposes of my project, so when I didn't find a suitable method in the AVL tree class, I would search in its parent, the binary search tree class.

Since I couldn't find a reliable append or insertion method, I experimented with __setitem__ method from the binary search tree class, and it worked when I used it to add words from professor's dictionary to an instance of the AVL tree map that I created to store the dictionary. Due to the dunders surrounding the method name, I could tell that this is a magic method, so instead of calling the method name directly, I could use an array-like syntax when adding each word to the tree map, and the method would be called automatically.

To check if words that the user inputs into the textbox are misspelled, I used the __getitem__ function. The original __getitem__ method from the binary search tree class raises KeyErrors whenever you are trying to get the value of a key/word that is not stored in the tree map. In my project, it isn't helpful to be raising KeyErrors, so I actually copied the __getitem__ method to the AVL tree class, and edited it to suit the project needs. I changed it so that the function returns None whenever a key/word does not exist in the tree map dictionary, which is when a word is considered misspelled. Misspelled words are then appended to a list one by one and displayed on the screen when the user clicks submit. In this case, the __getitem__ function from the AVL tree class overrides the __getitem__ function of the binary search tree class since the AVL tree class is its child.

An AVL tree map is much more efficient than a built-in python dictionary because searching for a key in a built-in python dictionary has a runtime of O(n), while searching for a key in an AVL tree map only has a runtime of roughly O(log(n)). This is because the depth of a balanced binary tree (like an AVL tree) is O(log(n)), so when a search is conducted on it, the run time is O(log(n)).