**User Scenarios:** long description for the use of the product. Can include multiple requirements.

Daniel:

1. Implement a prompt that asks for the name of an input and the name of the dictionary. Both of these files should be in the same directory as the program. Make sure only text files work. Once this is done, test to make sure that the files can be read and it can be stored. Make sure that the stored data can also be retrieved.

3.) Using the functionality from (1), find a way to convert the words in these text files so that only non-special characters are read. After converting them to normal characters, force them to be all lowercase and then check to see if the words compare to the dictionary correctly.

Allie:

4) The dictionary must be stored internally in an efficient manner. Use hash tables as the data structure and make sure storage method has the capacity to manage input information. Use optimal methods to ensure efficiency, and test multiple cases for the most efficient result. The user will not have any interaction of methods, but will be ensured efficiency.

5) Duplicate Errors should not be displayed. Branching from task 4, as a benefit of using hash tables, duplicates are automatically not allowed. So, it is of integral importance to ensure efficiency of hash tables for specific requirements. While the program is running, if the input or dictionary files happen to have duplicates, then they will be eliminated before information is displayed on the GUI allowing the user to easily access the dictionary without having to worry about duplicate words being added.

Petra:

7)The user should have the option of storing all text and words before opening another file. The option will be available to the user in the form of a button to the right of the central GUI component labeled “Store”. In case the user forgets to click the necessary button to store the words, just before the user decides to open another file, a popup box will appear to remind the user to store the words or lose the data. This information will also be included in the help GUI. The button needs to include the necessary functionality to store added words to the end of the linked list or in alphabetical order.

8) There must be a GUI with a help screen. The help screen will display the step-by-step procedure to make sure that the user is aware of how to use the program. The screen will also include anticipated troubleshooting question and answers that the user can peruse through to address problems they encounter.

David:

2.) Create an algorithm that displays words in given input file(s) that are not in given dictionary file(s). The algorithm will have two arguments, a hash set that holds all the words from the all the selected input files, and another hash set that has all words from the dictionary files selected.

6.) Implement an input file system that can handle multiple files. The system would keep track of a number of selected files and sequentially open each file. Upon opening each file, the contents of the file would be added to the corresponding hash set.

**Selection and Rationale for Internal Data Structures:**

LinkedList? Hash Set?

Hash Set:

* Cost Efficient
* Does not allow duplicates
* More efficient than an Array
* Dynamic Structure

**Mockup GUI:**

