Use Case Scenario

- Upload File
- Validate File name
- Read File
- View Help Info
- Remove Punctuation
- Remove possession (apostrophe's)
- Report history of all files processed
- Report averages across all files

User

Text Files Analyzer

- Upload File
- Validate File Name
- Read File
- View Help Info
- Remove Punctuation
- Remove Possession (Apostrophes)
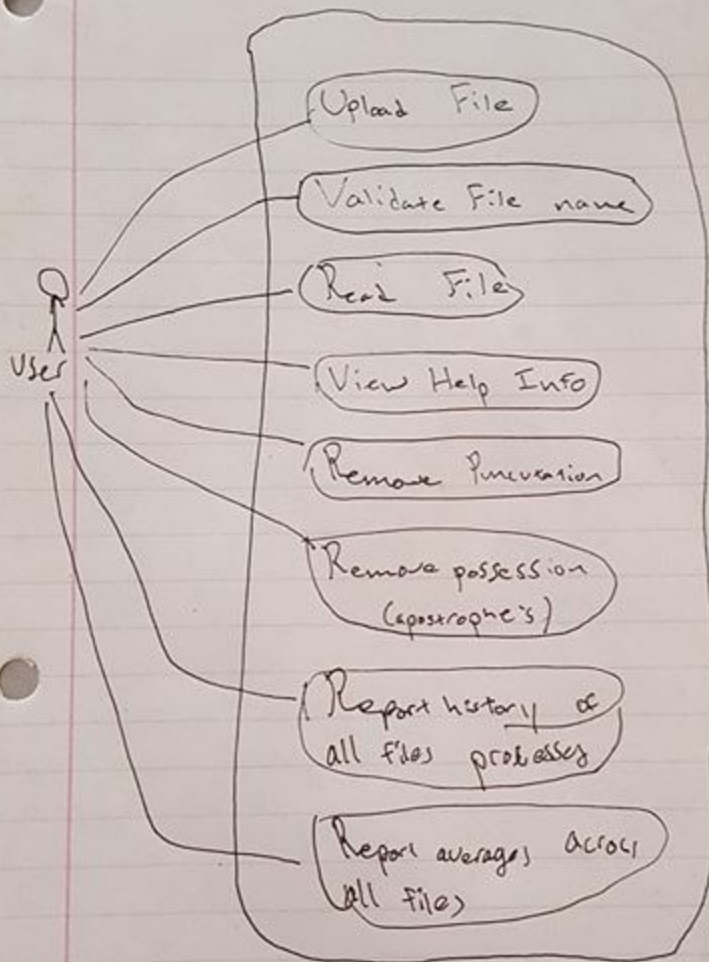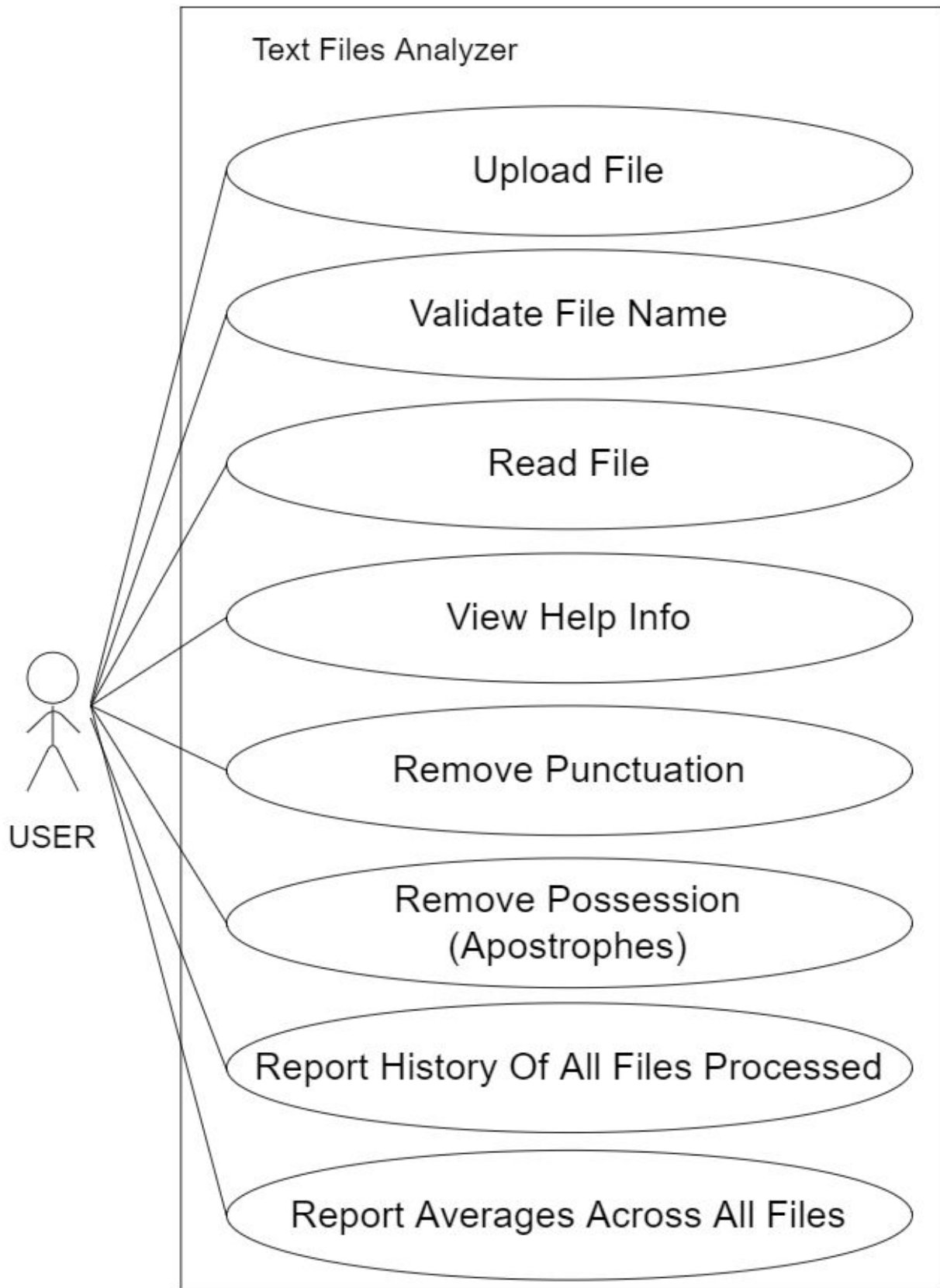- Report History Of All Files Processed
- Report Averages Across All Files

USER

use case scenarios, test plan, plan for deliverable 1

Users: seems like there is only 1

Use Case:
Upload a File
Validate File
Read the File
View Help Information

Show Reporting HIstory of All Files Processed

Test plan:
if the file is compatible (i.e. file type)
Corrupted file?
what if a file is empty?
what if a file is... a bunch of scribbles?
Are we reading the correct number of lines?
Example:
Def testCorrectNumberOfLines(unit.test):
newFile = read(newFile.txt)
newFile.text =
" Hi, this is a test file!,
I should only have two lines!"
assert(LinesInTheFile(newFile.text),2)
#This means it will fail if the line count isn't = 2
Are we getting the correct number of blank lines?
Example:
Def testCorrectNumberOfBlankLines(unit.test):
newFile = read(newFile.txt)
newFile.text =
" HI I GOT A BLANK LINE,

IF YOU READ ME RIGHT, SHOULD HAVE 2 BLANK"

        assert(BlankLinesInTheFile(newFile.text), 2)

Are we getting the Correct number of words?

    Example:

```
Def testCorrectNumberOfWords(unit.test):
    NewFile = read(newFile.txt)
    newFile.text =
        "Hi I really do have 7 words"
    assert(NumberOfWordsInTheFile(newFile.text), 7)
```

Are we getting the correct number of spaces in the file?

```
Def testCorrectNumberOfSpaces(unit.test):
    NewFile = read(newFile.txt)
    newFile.text =
        "Hi I really do have 6 spaces"
    assert(NumberOfSpacesInTheFile(newFile.text), 6)
```

Are we calculating the average characters per line in the file correctly?

```
Def testAverageCharactersPerLineInTheFile(unit.test):
    NewFile1 = read(newFile1.txt)
    NewFile2 = read(newFile2.txt)
    # Spaces count as characters
    newFile1.text =
        "Hi
         I am
         An Average
         Of 8.2 Characters
         Per Line"
    newFile2.text =
        "Also
         I am another cool file
         And my average
         Char per line
         is  12"
    assert(AverageCharactersPerLine(newFile1.text), 8.2)
    assert(AverageCharactersPerLine(newFile2.text), 12)
```

Are we calculating the average word length correctly in the file?

```
Def testAverageWordLengthInTheFile(unit.test):
    NewFile = read(newFile.txt)
    newFile.text =
        "Hi I should Have an Average
         Word length of 3.7"
```

```
                    assert(NumberOfSpacesInTheFile(newFile.text), 3.7)
        Are we calculating the most common words correctly in the file?
                    Def testMostCommonWords(unit.test):
                        NewFile = read(newFile.txt)
                        newFile.text =
                            "Hi My Most Common Word is Cloud Cloud Cloud"
                        assert(MostCommonWordInFile(newFile.text), "Cloud")
```

Reading text from file and reporting
a.   # lines
b.   # blank lines
c.   # spaces
d.   # words
e.   Average chars per line
f.   Average word length
g.   Most common words

Plan for deliverable 1 :
        Start with Unit Tests first (TDD)
                Write the unit tests that we laid out in Pseudocode on this doc.
        Then Write the Functions needed on the tests
        Function that Opens a file (with a simple validation, that will be improved on in del. 2.)
        Design the GUI Simultaneously (A very simple GUI)
                Will just have a file uploader, and then an output of all the methods needed:
                        a.   # lines
                    b.   # blank lines
                    c.   # spaces
                    d.   # words
                    e.   Average chars per line
                    f.   Average word length
                    g.   Most common words
                Will only ask to upload one file for deliverable 1
        Have a function that will close the file at end.

Cody (classes) : out of town thursday afternoon through saturday for all remaining weekends of
october - available to google hangout any time on saturday (must plan in advance), evenings /
afternoons preferred

Available all day after noon on tuesdays and thursdays

Javier Fernandez Valles (From October 20th - Nov 27th)
        All Saturdays -> from 2pm - 10 pm
        All Sundays after 5pm
        Available Tuesday Nights (Tentative)
        Available Mondays from 3:00 to when 360 starts, then after 360 to 7:30 when we got
recitation

Adrian (GUI)
        Mondays before 360 and after 360 to when recitation starts
        Wednesday before and after 360
        Tuesdays all day up to 6:30 and Thursday from 2-6:30
        Fridays, Saturdays, Sunday after 2 pm

Avi (testing):
        tues/thurs 1:30 pm onwards is free. On campus at this time.
        Can be available before 4:35 PM on monday and wednesday. Would just need to plan it.
        Nov 4 (saturday) and nov 11 (saturday) is no good for me.
        But in general, saturday and sundays can work for me all day