

Phone Combination Project Report

COSC 4360 Software Engineering

April 30, 2023

Dr. Andrei

Group Members:

Andrew Kalathra, Casey Lieby,

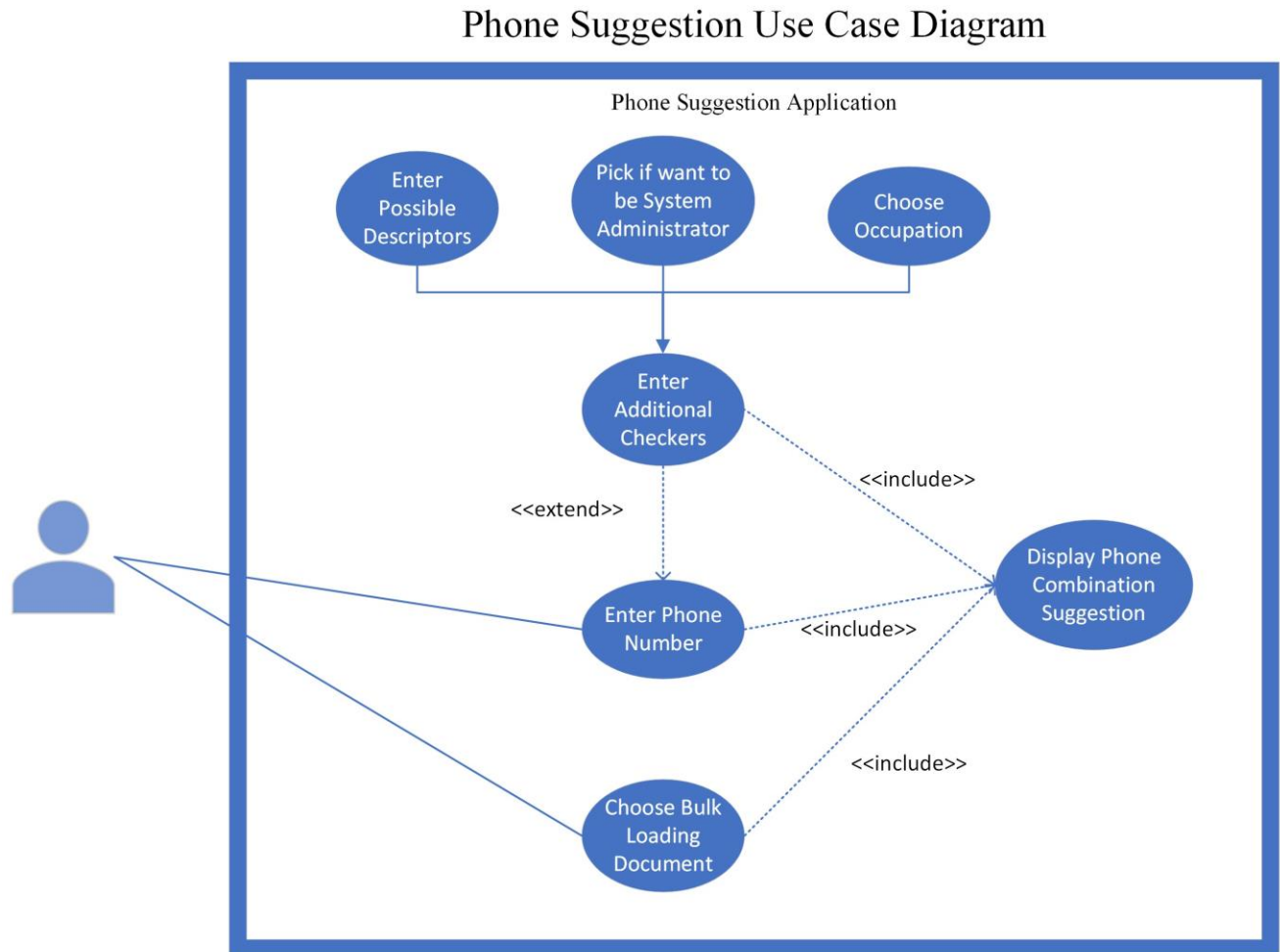
Bralon Holmes, Zachary Essoh

Abstract:

The phone number of a business is just as important to its identity as its name. But finding the right advertising phrases for a business can be a challenge. This project demonstrates one possible software solution that sets out to solve this problem. Using a simple database, possible combinations of numerical symbols and English words are combined to make possible advertising phrases. Filters are then added to narrow or tailor the possible results to the user's business. Through this software, the user would be able to enter their occupation sector or business niche, along with other additional information. This information is then used to check against the possible combinations that were generated. The software goes through a few different states, including a query state, a processing state, a retrieval/filtering state, and finally an output state. The actor, or the user, interacts with the program through a GUI form, which prompts the actor to enter all necessary information. Classes in the program interact to first pass on the

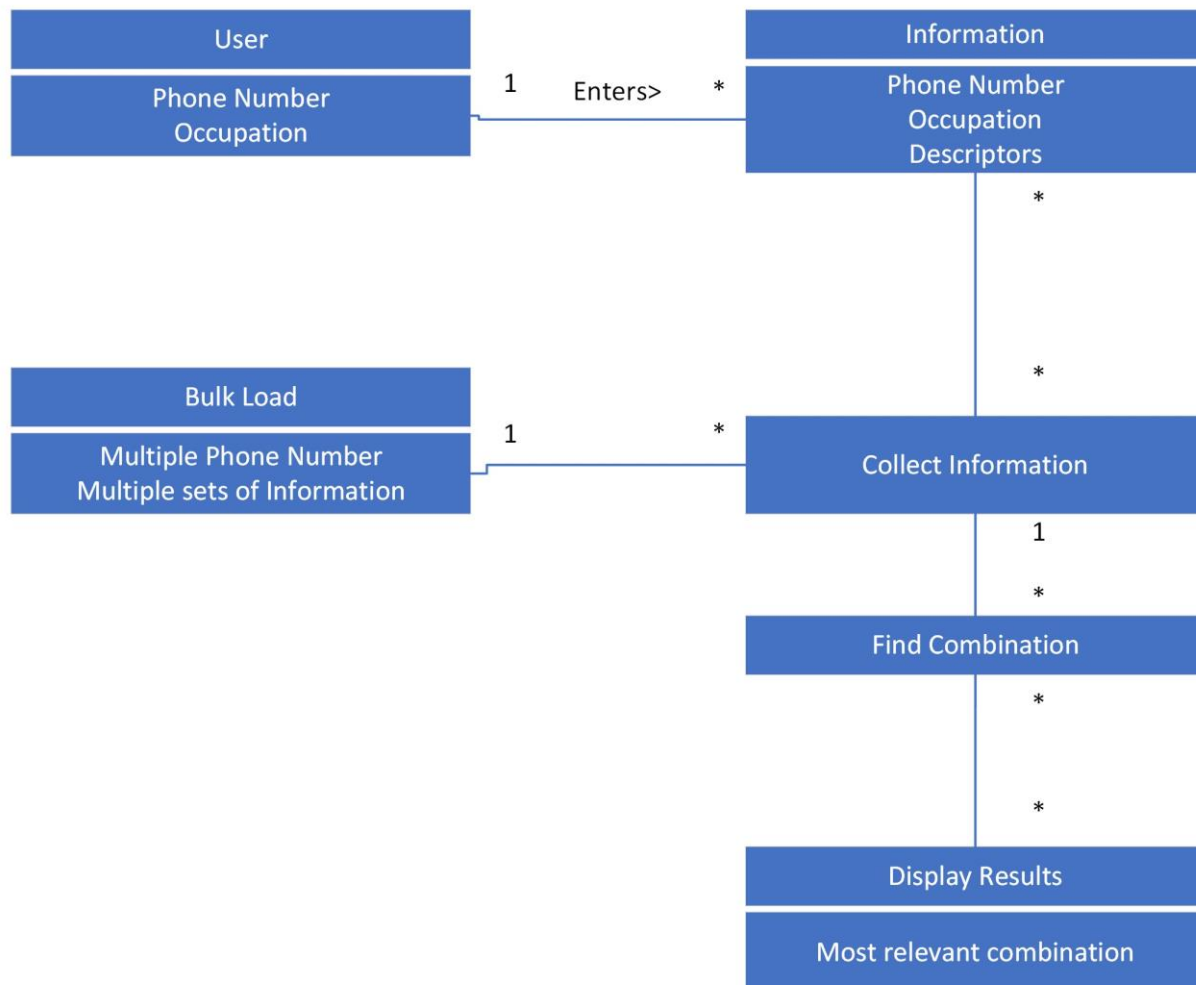
information to process all combinations, then to return the resulting combinations, and finally to filter the results. The actor is then shown the results, giving them their possible choices. Possible businesses that could use this software include the arts, lawyers, education, and home services.

Use Case Diagram:



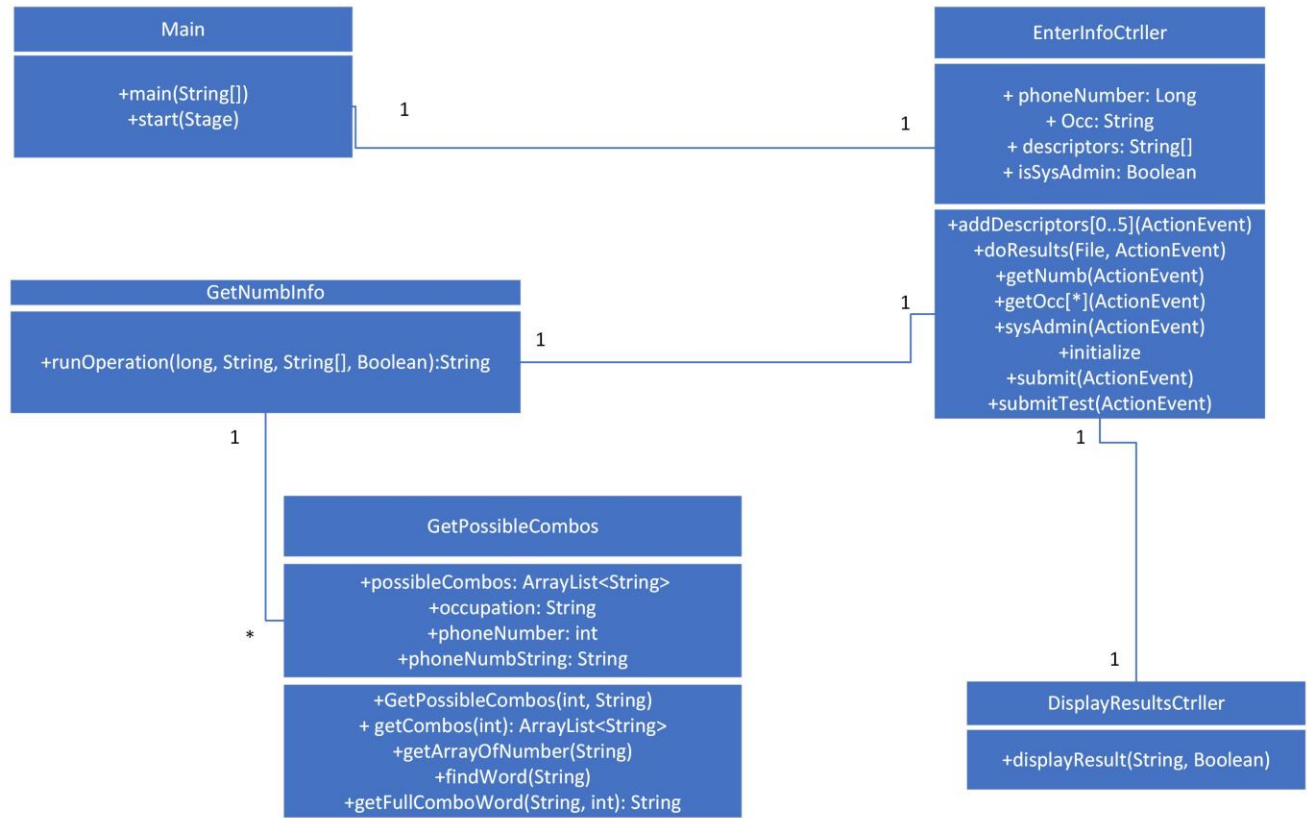
Domain Model:

Phone Suggestion Domain Model



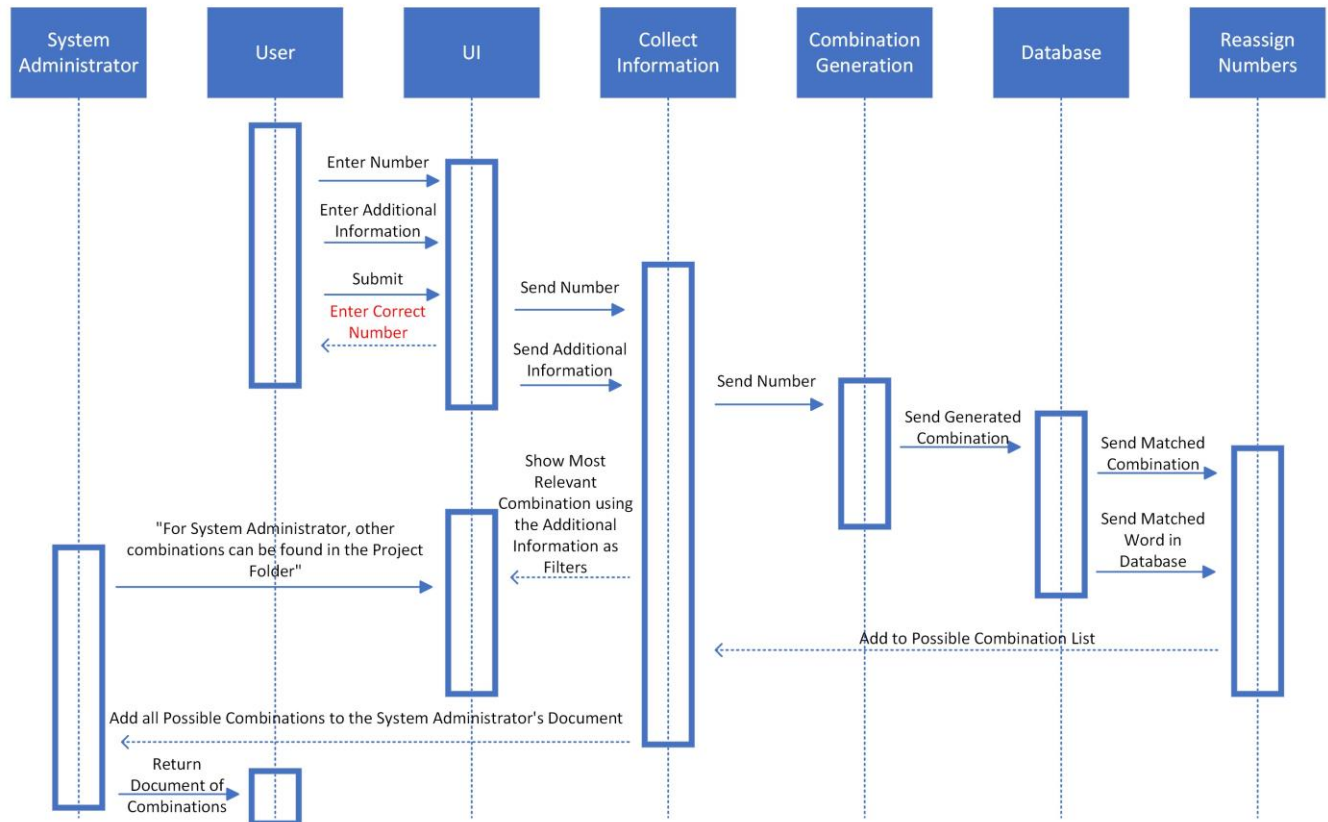
Class Diagram:

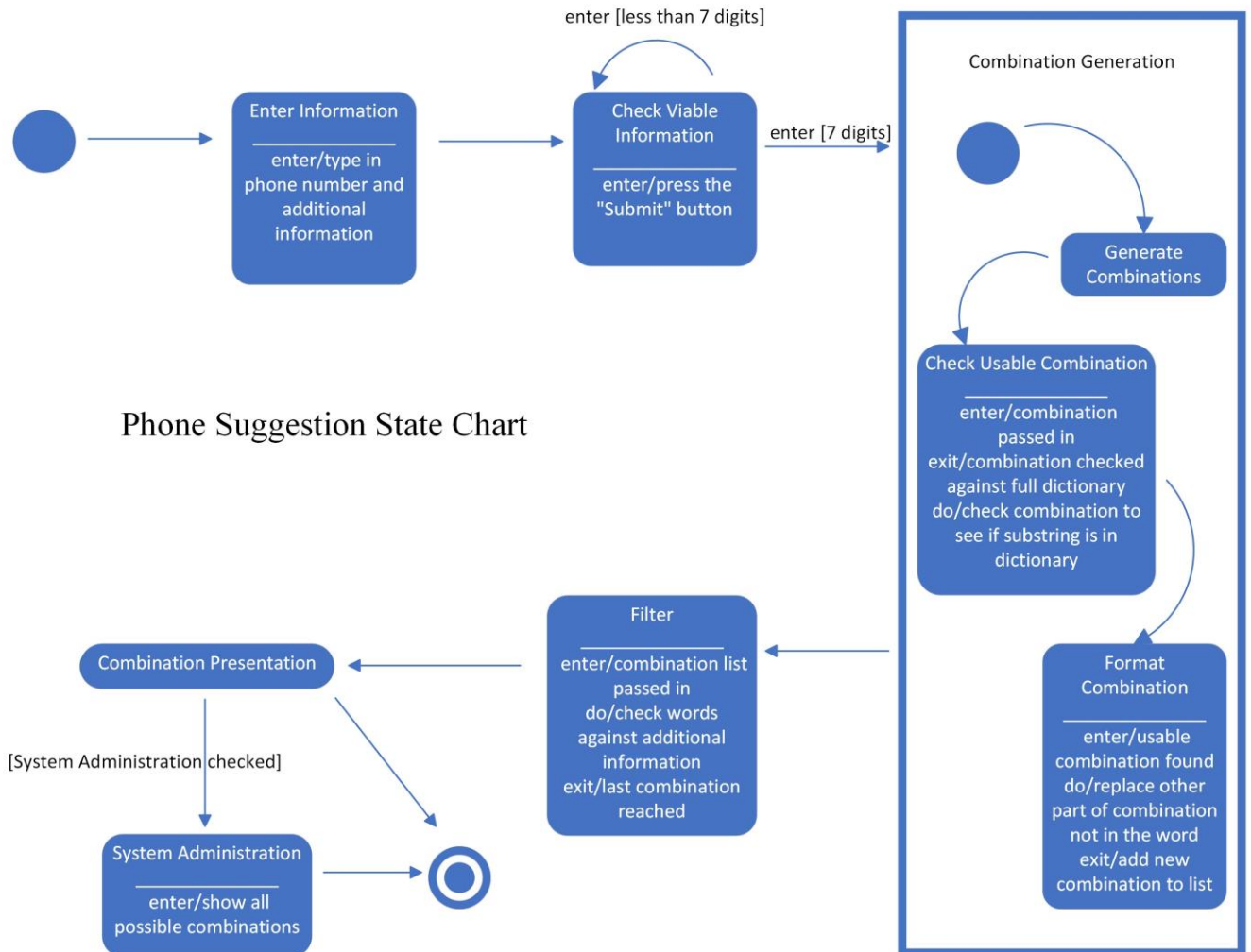
Phone Suggestion Class Diagram



Interaction Diagram:

Phone Suggestion Interaction Diagram



State Chart:

Implementation - UI:

The image shows two side-by-side screenshots of a web application interface. The left screenshot displays a form titled "Enter Phone Number" with a text input field containing "6583437". Below this are five rows, each with a label "Enter Descriptor 1" through "Enter Descriptor 5" and a corresponding text input field. The first descriptor field contains the word "judge". To the right of these fields are two radio buttons, with "System Administrator" selected, and a dropdown menu labeled "Select Occupation Sector" with "Law" selected. At the bottom of the form are two buttons: "Enter" and "Open Test File". The right screenshot shows a result screen with a green header bar. The main content area contains the text "The combination 6judge7 is recommended." and a footer note: "For System Admin, other possible combinations can be found in the project folder".

Enter Phone Number

6583437

Enter Descriptor 1 judge

Enter Descriptor 2 Descriptor 2

Enter Descriptor 3 Descriptor 3

Enter Descriptor 4 Descriptor 4

Enter Descriptor 5 Descriptor 5

System Administrator

Select Occupation Sector

Law

Enter

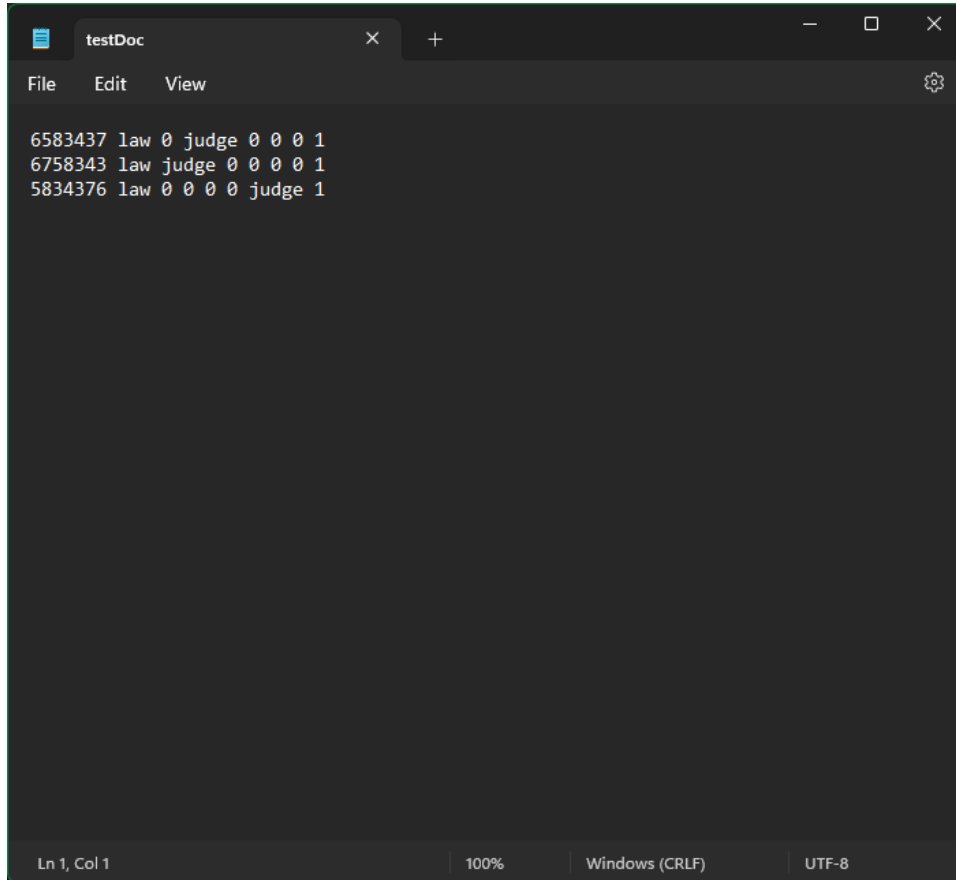
Open Test File

The combination 6judge7 is recommended.

For System Admin, other possible combinations can be found in the project folder

Testing:

The testing document used was a text file, and it looked as such:



```

6583437 law 0 judge 0 0 0 1
6758343 law judge 0 0 0 0 1
5834376 law 0 0 0 0 judge 1
  
```

The screenshot shows a text editor window with a dark theme. The title bar says 'testDoc'. The menu bar has 'File', 'Edit', and 'View'. The status bar at the bottom shows 'Ln 1, Col 1', '100%', 'Windows (CRLF)', and 'UTF-8'. The text content consists of three lines of data, each with a numeric ID, an occupation, and several binary descriptors.

This testing case utilized the bulk loading function to ensure multiple systems worked.

First, let us discuss the format. The format of the bulk loading option is read in as such:

XXXXXX ABCDEF H I J K L *

Where X is a digit (0-9)

A-F is an occupation from the occupation listed (can be of any length as long as it is an occupation)

H-L are individual descriptors (5 in total, “0” entered if none wanted for that location)

and * represents 0 or 1 (1 for being a system administrator, and 0 for not being one)

These numbers were specially chosen because they each have the sequence of numbers of “58343”. This number can be translated into letters to form the word “judge”. This meant that all the phone numbers entered should have given us the same recommended results, no matter where in the number the digits were placed (hence one at the beginning, middle, and end), though the individual ‘System Administrator’ documents would have been different. This number also checks if the occupations folders are read, as the word “judge” cannot be found in the normal dictionary, but rather only in the “Law” occupation dictionary. Alongside this, these test cases verify that the descriptors do not rely on the location they are put in. Finally, the choice of ‘System Administrator’ was selected for all the numbers to check that the individual numbers themselves contained other possible word combinations.

The final scene showed the possible combinations of:

6judge7 , 67judge , and judge67

This confirmed that all our systems worked as intended.

Assumptions and Limitations (& Additional Notes):

- It was assumed that entered numbers would only contain 7 digits. This did not include the area code or other prefixes, as many would change.
- It was assumed that the number 8 could stand for the set of characters “ate” as they sound similar
 - This resulted in words such as “l8” which could be read as “late”
- It was assumed that the number 0 stood for more symbols than are shown on a normal keypad.
- A major limitation in this project is that phone numbers containing the digits 0, 1, and/or 9 had lower results. This was because 0 and 1 only contained symbols, and not many words contained the characters represented by 9.
- Another limitation of this project is the run-time. Though the project gives more accurate results due to the individual dictionaries, the run-time of the process is quite slow. This is due to the way the algorithm for finding if a combination contained a word. A more efficient process could have been obtained but was not able to be implemented due to time constraints.
- An additional note to consider is that the occupations could be expanded simply by adding another text file to the ‘Occupations’ folder as well as adding related words to said file. In the same way, more results could be obtained by the numbers entered by adding to the dictionary file. This, however, would also cause a longer run-time in the current algorithm due to the limitation above.