MSc. Big Data Analytics & Management

Cloud Computing - [MSCBD-CC]

Assignment 02 - Documentation

MERT ALTINTAS - 2982775

---

1. Project Overview

In this assignment, the given task was to build an Anagram Engine using the Google Cloud Platform and its components. The project's back-end was realized by Python programming language and Google Cloud's NoSQL database (ndb) was used in order to store the anagram dictionaries and user details such as their dictionaries, and total number of words. On the other hand, the front-end tasks were performed by using HTML, CSS, Bootstrap, and JavaScript. The version control was made via GitHub.

In this paper, firstly, the classes and their methods will be explained. After that section, the detailed information about the data structure and the models are going to be presented. Finally, the methodology behind the UI will be described.

2. Classes & Methods

The functionality of the application is basically realized with 3 Python classes and their methods. The classes used in the project are as follows:

- **MainPage Class**

  This class is mainly responsible for rendering the landing page, user login process, and anagram search function. It has three methods which are orderLetters, get and post.

  To store a word in our database we need a unique key. This key should be generated using the lexicographical order of the given word. Therefore, orderLetters method takes the word and puts its letters in alphabetical order and returns the key. Briefly, this method generates our keys by ordering the letters in the given word.

  Basically, get method sends information to the Google Cloud Platform. MainPage class firstly sets up the 'Context-Type' as HTML and controls if there is any user has already signed in. According to the return value, it directs the user to the landing page by creating a login url from where the user can login or to the main page where the user can search an anagram and log out from the system. The other thing is, this method should also store the users that logged in. Thus, after user authentication, it also controls if the user is placed in the database. If it is the user's first login to the system it creates a user model and stores the user in the database. Finally, get method renders the main page and shows it to the user.

  Post method controls the Enter button. On that page, users can search anagrams related to the input that they entered. The input is restricted only for the alphabetical characters. Whenever a user enters a word to look to its anagrams, post method gets the word/text, splits the words in case there are more than one, and creates appropriate keys related to

each word. After that, it controls if the user has the anagram dictionary/ies under the keys. If the anagrams are found in the user dictionary, the method gives them to the template and presents the anagrams to the user. Otherwise, it just shows no results.

- **AddWord Class**

AddWord class is responsible for adding new words to the user's own dictionary. There are two ways to add a word. A user can add a word either manually or by uploading a text file. In this class, there are 4 methods such as orderLetters, addWord, get, and post in order to realize word addition.

OrderLetter method generates a key from the given word by putting its letters in lexicographical order.

addWord method adds the word into the user dictionary and returns a boolean value to show if the addition is completed or not. It takes 3 arguments. "key" defines the dictionary key which is in user_id:AlphabeticallyOrderedWord format. "myuser" is the user model and "word" parameter indicates the word that the user wants to add to the dictionary.

This method, firstly, tries to get the dictionary in which the word should be added, from the database. If a dictionary doesn't exist it creates a new dictionary instance by setting its own parameters. Secondly, it looks into the dictionary's word list to control if the word exists or not. If it is a new word in the dictionary it adds the word to the list. Finally, it adds the dictionary to the user's dictionary list under user model and returns a boolean value.

Like in the MainPage class, get method controls the user authentication. If there is a user logged in it directs the user to the related page, else it renders the landing page in order to make the user log in the system.

Post method controls two buttons. Add Word button provides the user to add a word manually. Whereas Submit button does the word addition from a text file given by the user. When a user enters a word and clicks Add Word button, post method gets the word, creates the word related key, and calls the addWord method. After that, it renders the page with the new values. In the second case, when the Submit button is pressed, the method takes the file and opens it. For each line, it fetches the word, creates the key and calls the addWord method. After all the lines are processed, it renders the related page.

- **SubAnagram Class**

This class is responsible for finding the sub-anagrams of the given word. It contains 5 methods such as orderLetters, getSubkeys, getAllSubKeys, get, and post.

The orderLetters method does the same thing as in the other classes. It generates a key from the given word.

getSubkeys is a recursive method to find n-1 sized keys of the given n sized key. It takes the "key" parameter from which we'll find the sub-keys. The other parameter "index" indicates the letter position which we start to remove letters and "keys" parameter is a list where we store all the sub-keys. The method basically removes letters starting from the last one and appends the word to the keys list in each removal. For example, if you'll give

the word “live” to the method, first, it will order the letters alphabetically which will be “eilv”. After, it will start to remove letters from the end of this word. Thus, at the end of the first iteration we’ll get “eil”, the second one will give us “eiv”, third will be “elv” and the last will end by finding “ilv”. These 4 words will be kept in the keys list as all possible sub-keys. Finally, the method returns a keys list.

getAllSubKeys method calls the getSubKeys method to find all the sub-keys until it reaches to the size 3. To do that key and keys list must be given. First, it calls getSubkeys and saves the result to a list. Then, the method checks the size of the words, and if it is not 3 it calls the getSubkeys method for each word to generate all the possible sub-keys until the size 3. Finally, it returns the keys list.

get method’s mission is the same as in the other classes. It checks if the user exists and does the authentication for the user.

post method controls the “Enter” button. On that page, the user enters a text to visualize its sub-anagrams. Therefore, this method takes the word that the user entered, generates a key, and finds the related dictionary from the database. If the dictionary exists, it checks if sub-anagram keys exist under the subanagramKeys column. If sub-anagram keys have already generated before, the method directly gets the keys from the column without calling the recursive function. However, if the subanagramKeys column is empty, it calls the getAllSubKeys method to generate all the sub-keys. Then, it puts all the possible keys under subanagramKeys column to get them later easily. Finally, it gets all the related dictionaries by using the keys and presents the sub-anagrams to the user.

3. Models

Two models are used during the realization of the project.

MyUser stands for the user model. Each user has a userDictionary and wordCount attribute that can be reached by user_id. userDictionary holds a string list that consists of dictionary keys that users have. On the other hand, wordCount shows the total number of words that users have in their own dictionaries.

Dictionary model structures each anagram list in the system. Each dictionary has a key which is the letters of a word in lexicographical order, wordList that stores anagrams, wordCount that shows us how many unique words that user have in that dictionary, letterCount that shows how many letters does anagrams have, and subanagramKeys that holds the keys of all possible sub-anagrams related to that key.

4. UI Design

The UI is designed to present a user-friendly environment to the user in order to facilitate the usage of the application. During the development of the interface, CSS, Bootstrap, and JavaScript are used. Thanks to Bootstrap, pages become responsive which means that they change their form automatically according to the window size.

The system is designed in a matrix dashboard style because in the system each user have their own data. There is no shared data between users so, a matrix dashboard style is appropriate for that case. The layout of each page is really simple. On each page, there is a navigation bar on the left size of the page that consists of the application name, and the navigation menu. Log In/Out button is placed on the top-right of the page. The middle part is the body where you can find specific content related to that page. In addition, at the bottom, there is a footer in which you can find the credentials and the navigation links.

The main color scheme for the application is chosen as blue combined with light gray. According to the design perspective, blue is a calming natural color. Plus, shades of blue are perceived to be friendly, authoritative, peaceful, and trustworthy.

The tables, forms, and buttons are designed proportionally so that it presents a proper view to the user. No unnecessary information is used. On the other hand, complexity is avoided while designing the pages. In addition, all the platforms such as Internet Explorer, Google Chrome, Safari, etc. are considered while placing each element on the pages.