# Classes and Functionality

|  |  |
| --- | --- |
| **Class name** | **Functionality** |
| MainPage | User authentication and query for words to be searched in the Anagram engine. |
| Add | Adds new Anagrams in the database. Depends on whether the word is actually in the database. |
| MyUser | Class for storing user information in database along with the counter for unique anagrams and total words. |
| Words | Model class used to store the information to the data. Contains information such as words, count of alphabets. |

Figure 1| Classes in the application

|  |  |
| --- | --- |
| **Class Name** | **Files** |
| MainPage | main.py and main.html |
| Add | add.py and add.html |
| MyUser, Words | snippets.py |
| Other files | app.yaml(manifest file for Google App Engine) |

Figure 2| information About the Files for each class

# 2. Details information of class methods

**Information about the methods in various classes are listed in the below table. Most classes will have a get and a post method used for communicating with the server which is hosted locally except for the classes that are used to store information in the form of objects.**

**Class MainPage:-**  In this Class we make use of User ID to create an identification for the user and this information is added to the database. This class checks the user ID and then it displays the welcome message biased on the fact that the user has used the application or is using the application for the first time. Along with the welcome message we also provide option to search for Anagrams and subanagrams only when the user has logged in successfully. Different users will have different words stored for them in the database, this will mean complete isolation for each user. All this is done using only the ***get*** method as we are pulling the values of the user from the users class and the Words class. (Generally post method is used to push information in the database or application and get method is used to fetch the information). We have the get, split, merge, sort and substring method to split the words in the query.

* Get Method: The ***get*** method here does the job of rendering the information that we retrieved from the datastore. The user searches for a word and the process of splitting and sorting is done here. The word is then searched in the database. We also include the subanagram search here, which works similar to the Anagram search but includes generation of the words using the alphabets of the searched Sub anagram. These words are then queried in the database.

**Class Add:-** In this class, we have the post that serves the purpose of fetching the information from the datastore. We also have the split, merge and sort method used for splitting the words, merging the alphabets and sorting the elements of the list.

* Post Method: The ***post*** method is used to generate an object for the words and then put that information in the database. Here, we get the word from the user and then split the word and sort it lexographically to be used as a key along with the User ID. This ensures that each user will have their set of words which are isolated from each other. We also have an option to read from lines which will be triggered only when we press the button to add the words. The file is stored locally and is easily modified manually.

**App.Yaml:**  This is the most important file in the whole code as it tells the Google App Engine about the handlers in the project, which runtime to use and what libraries we should be using. It works the same way as a manifest file works in android.

# 3 Models and Data-Structures

In the application the main data-structure that is used is a ***list***. The list is used to pass information to the jinja2 module. We use the MyUsers Model to store information pertaining to the user like the username and the amount of words and unique anagrams in his own isolated area. We also make use of Words class to store all the words in the database and keep track of how many words and entries there are for a particular anagram. We make use of lists to store the strings of the word so that multiple words can be stored for a single entry.

# 4 UI Decisions

The UI consists of simple designs so that all users would feel ease while trying to use the application. We provide the ability to query the anagrams and subanagrams on a single page separated by divisions.