**MAJOR PROJECT SUMMARY**

###### ***Submitted by***

###### ***AGAM CHAUDHRY(171B019)***

**Under the guidance of : Dr. Babita Tiwari**

****

**July 2020**

***Submitted in partial fulfillment for the award of the degree of***

##### BACHELOR OF TECHNOLOGY

IN

Department of Computer Science & Engineering

JAYPEE UNIVERSITY OF ENGINEERING & TECHNOLOGY, AB ROAD, RAGHOGARH, DT. GUNA-473226 MP, INDIA.

**INTRODUCTION TO PROJECT**

**Executive Summary**

To-do Lists or "Checklists" are the best way of tracking tasks. To-do List is software in the category of Task Management, Project Management, Productivity, “Getting Things Done” (GTD), Scheduling, and Collaboration.We have a lot of choices to help us keep track of daily obligations. A simple list on paper of things “To Do” is enough for some people. To-do List is general-purpose, Web-based software, which can be used for simple “honey do” home lists or to manage complex multi-user projects for business. In addition to tracking the status of tasks

**What does the “To Do List” report?**

The "To Do List" report displays the list of tasks with start and finish dates for a specified resource.

**What Technology Used in Project.**

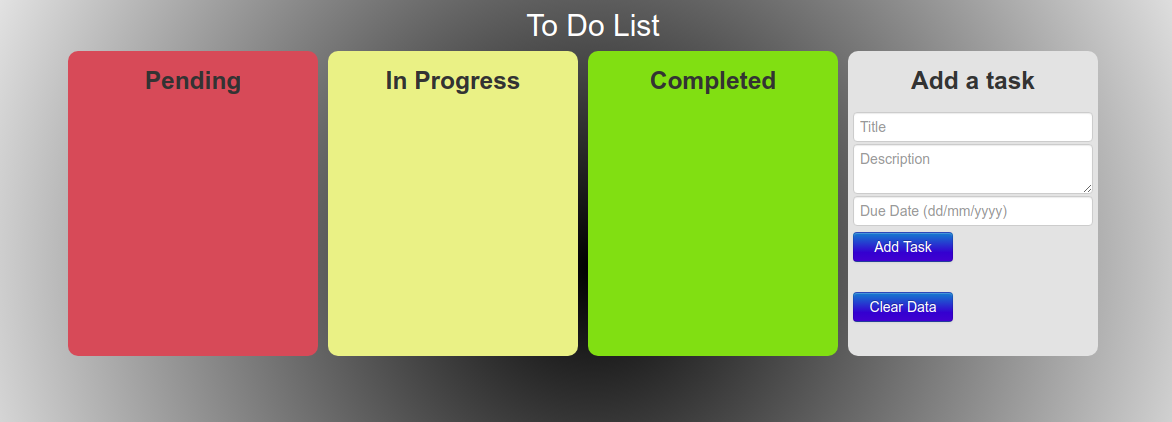
Below Language use to make to-do list.

1. Java Script (bootstrap, jquery)
2. CSS
3. HTML

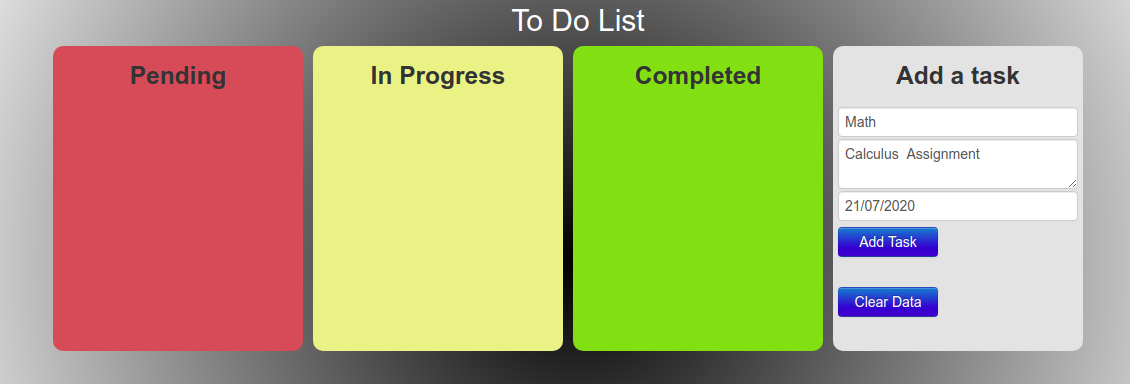
**Step by step on how to create To Do List :**

1. Open the index.html file.

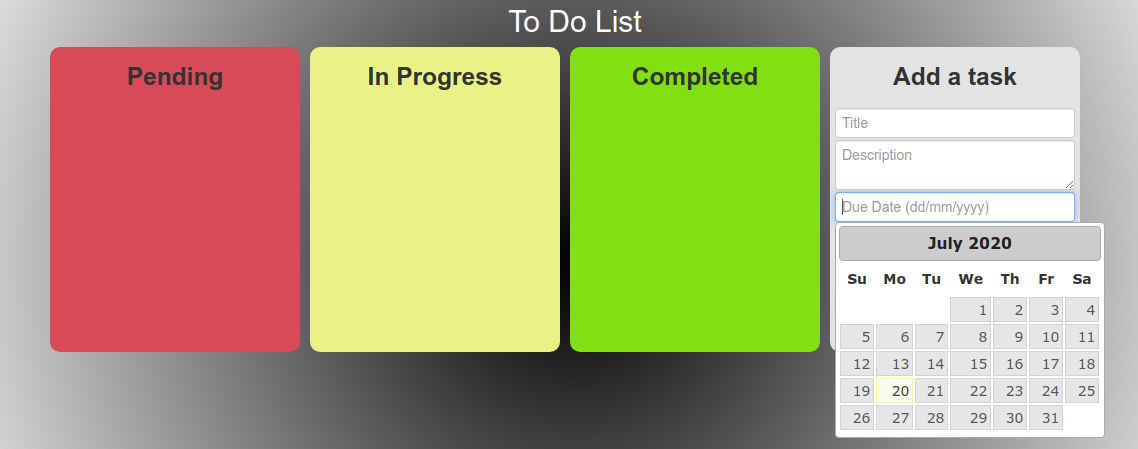
2. Go to Add Task box, Fill your details and click on Add Task Button.



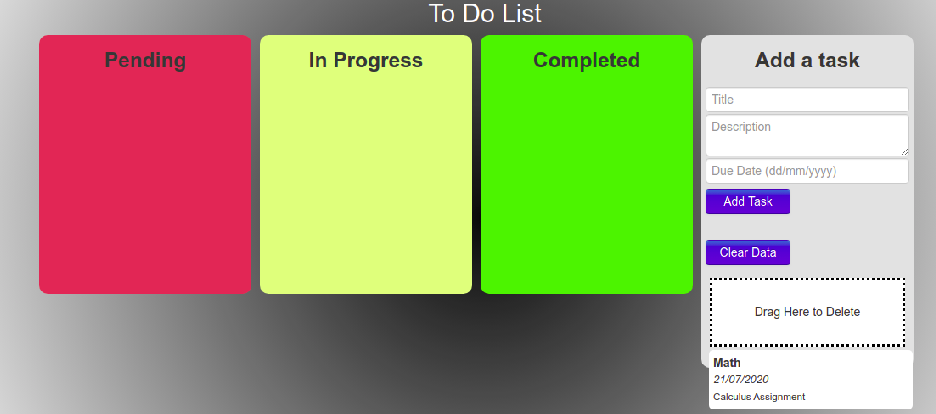
1. Fill your details and click on Add Task Button.



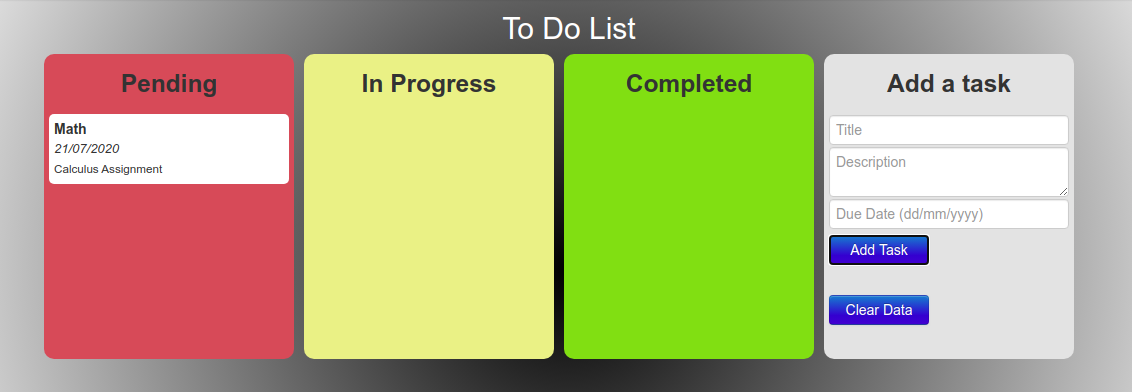
1. Select Date from date box



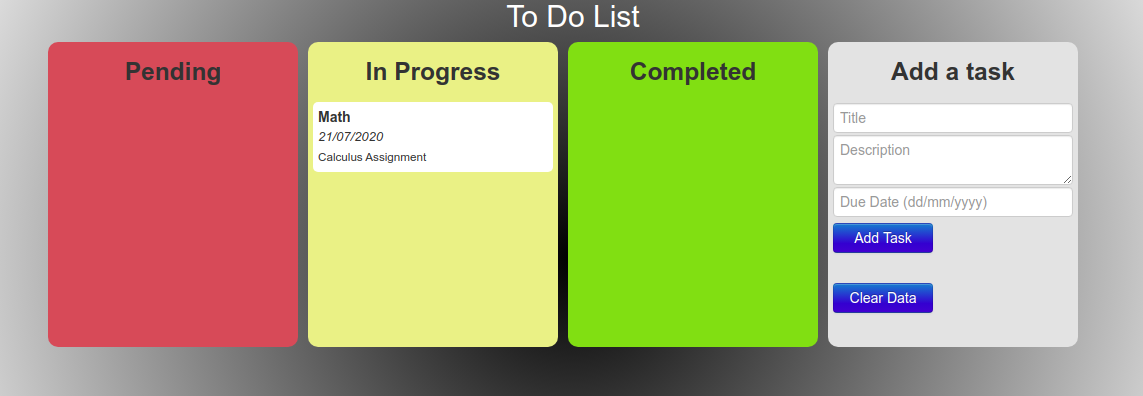
1. For Delete your single task need to drag and drop your task like below screenshot



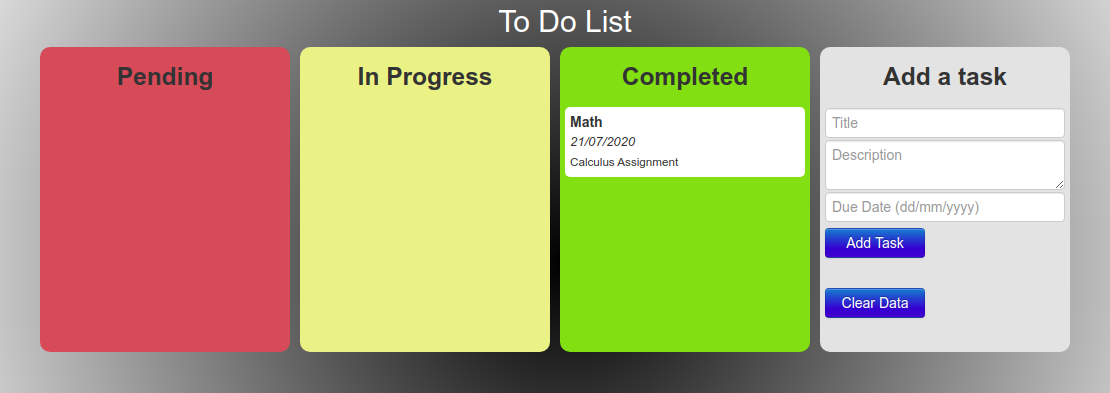
1. There are three level of task pending In Progress and Complete.
2. Pending (You can drag and drop your task on the basis of in progress or complete)



1. In Progress (You can drag and drop your task on the basis of pending or complete)



1. Complete (You can drag and drop your task on the basis of pending or in progress)



**Data**

To gather the data many options are possible. We will build a program to collect automatically a corpus of tweets based on two classes, “positive” and “negative”, by querying Twitter with two type of emoticons: ● Happy emoticons, such as “:)”, “:P”, “:­)” etc. ● Sad emoticons, such as “:(“, “:’(”, “=(“. Others make their own dataset of tweets my collecting and annotating them manually which very long and fastidious. Additionally to find a way of getting a corpus of tweets, we need to take of having a balanced data set, meaning we should have an equal number of positive and negative tweets, but it needs also to be large enough. Indeed, more the data we have, more we can train our classifier and more the accuracy will be.

**Difficulties in data pee-processing**

The presence of acronyms​"bf" or more complicated "APL". Does it means apple ? Apple (the company) ? In this context we have "friend" after so we could think that he refers to his smartphone and so Apple, but what about if the word "friend" was not here ? ● The presence of sequences of repeated characters​such as "Juuuuuuuuuuuuuuuuussssst", "hmmmm". In general when we repeat several characters in a word, it is to emphasize it, to increase its impact. ● The presence of emoticons​, ":O", "T\_T", ":­|" and much more, give insights about user's moods. ● Spelling mistakes​and “urban grammar​” like "imgunna" or "mi". ● The presence of nouns​such as "TV", "New Moon". Furthermore, we can also add, ● People also indicate their moods, emotions, states, between two such as, \*\cries\*, \*hummin\*, \*sigh\*. ● The negation, “can't”, “cannot”, “don't”, “haven't” that we need to handle like: “I don’t like chocolate”, “like” in this case is negative.

**Resources**

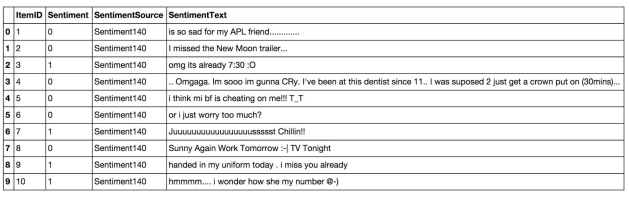
In order to facilitate the pre­processing part of the data, we introduce five resources which are, ● An emoticon dictionary​regrouping 132 of the most used emoticons in western with their sentiment, negative or positive. ● An acronym dictionary​of 5465 acronyms with their translation. ● A stop word dictionary​corresponding to words which are filtered out before or after processing of natural language data because they are not useful in our case. ● A positive and negative word dictionaries g​iven the polarity (sentiment out­of­context) of words. ● A negative contractions and auxiliaries dictionary​which will be used to detect negation in a given tweet such as “don’t”, “can’t”, “cannot”, etc. The introduction of these resources will allow to uniform tweets and remove some of their complexities with the acronym dictionary for instance because a lot of acronyms are used in tweets. The positive and negative word dictionaries could be useful to increase (or not) the accuracy score of the classifier. The emoticon dictionary has been built from wikipedia with each emoticon annotated manually. The stop word dictionary contains 635 words such as “the”, “of”, “without”. Normally they should not be useful for classifying tweets according to their sentiment but it is possible that they are.

**Pre-processing**

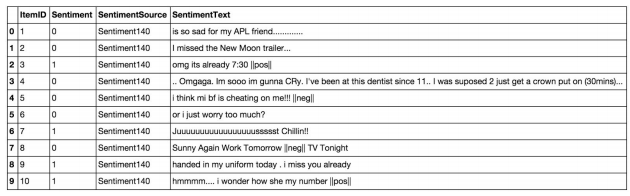
Now that we have the corpus of tweets and all the resources that could be useful, we can pre­process the tweets. It is a very important since all the modifications that we are going to during this process will directly impact the classifier’s performance. The pre­processing includes cleaning, normalization, transformation, feature extraction and selection, etc. The result of pre­processing will be consistent and uniform data that are workable to maximize the classifier's performance. All of the tweets are pre­processed by passing through the following steps in the same order.

**Emoticons**

We replace all emoticons by their sentiment polarity ||pos|| ​and ||neg||​using the emoticon dictionary. To do the replacement, we pass through each tweet and by using a regex we find out if it contains emoticons, if yes they are replaced by their corresponding polarity.



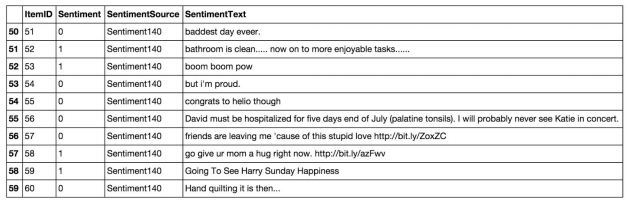




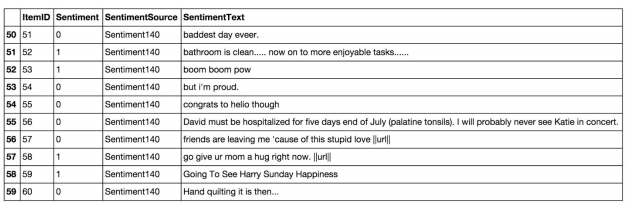


**URLs**

We replace all URLs with the tag ||url||​. There is about 73824 urls in the data set and we proceed as the same way we did for the emoticons.



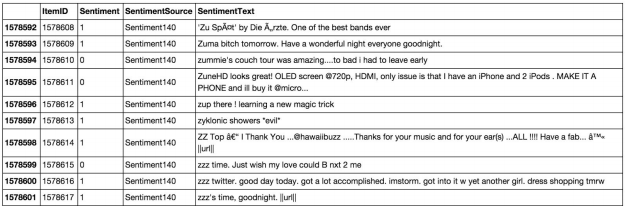




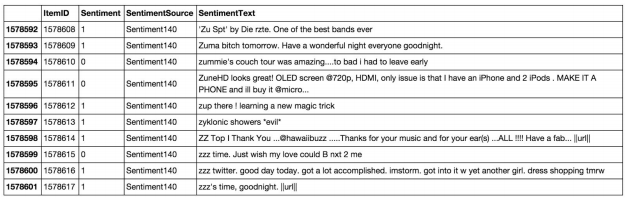


**Unicode**

For simplicity and because the ASCII table should be sufficient, we choose to remove any unicode character that could be misleading for the classifier.







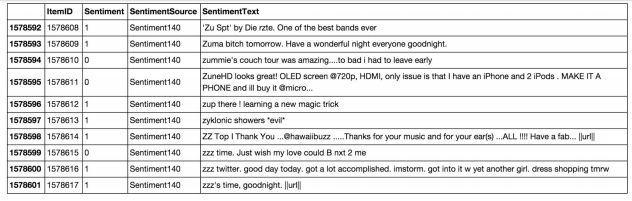


**HTML entities**

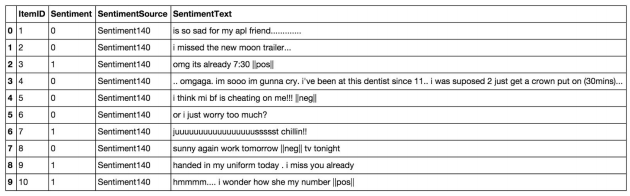
HTML entities are characters reserved in HTML. We need to decode them in order to have characters entities to make them understandable.

**Case**

The case is something that can appears useless but in fact it is really important for distinguish proper noun and other kind of words. Indeed: “General Motor” is the same thing that “general motor”, or “MSc” and “msc”. So reduce all letters to lowercase should be normally done wisely. In this project, for simplicity we will not take care of that since we assume that it should not impact too much the classifier’s performance.









**Targets**

The target correspond to usernames in twitter preceded by “@” symbol. It is used to address a tweet to someone or just grab the attention. We replace all usernames/targets by the tag ||target||.

**Acronyms**

We replace all acronyms with their translation. An acronym is an abbreviation formed from the initial components in a phrase or a word. Usually these components are individual letters (as in NATO or laser) or parts of words or names (as in Benelux). Many acronyms are used in our data set of tweets as you can see in the following bar chart. At this point, tweets are going to be tokenized by getting rid of the punctuation and using split in order to do the process really fast. We could use nltktokenizer but it is definitely much much slower (also much more accurate).

**Negation**

We replace all negation words such as “not”, “no”, “never” by the tag ||not||​using the negation dictionary in order to take more or less of sentences like "I don't like it". Here like should not be considered as positive because of the "don't" before. To do so we will replace "don't" by ||not|| and the word like will not be counted as positive. We should say that each time a negation is encountered, the words followed by the negation word contained in the positive and negative word dictionaries will be reversed, positive becomes negative, negative becomes positive, we will do this when we will try to find positive and negative words.

**Sequence of repeated characters**

Now, we replace all sequences of repeated characters by two characters (e.g: "helloooo" = "helloo") to keep the emphasized usage of the word.

**Conclusion**

Nowadays, sentiment analysis or opinion mining is a hot topic in machine learning. We are still far to detect the sentiments of s corpus of texts very accurately because of the complexity in the English language and even more if we consider other languages such as Chinese. In this project we tried to show the basic way of classifying tweets into positive or negative category using Naive Bayes as baseline and how language models are related to the Naive Bayes and can produce better results. We could further improve our classifier by trying to extract more features from the tweets, trying different kinds of features, tuning the parameters of the naïve Bayes classifier, or trying another classifier all together