FACULTY OF INFORMATION AND COMMUNICATION TECHNOLOGY

SEMESTER 1 SESSION 2022/2023

BITI 3413 NATURAL LANGUAGE PROCESSING

TITLE REPORT:

WHAT DO STUDENTS THINK ABOUT YOU? (SENTIMENT ANALYSIS)

PREPARED BY:

NO	NAME	MATRIC NO	SECTION / GROUP
1.	NUR SYUHADA BINTI AZHAR	B032010378	3 BITI S1G1
2.	NURSYAZA NISA BINTI ARFARIZAL	B032010244	3 BITI S1G1
3.	SYAKIRAH HANIM BINTI ZULKERNAIN	B032010116	3 BITI S1G1
4.	THIVEYA A/P MAHENDRAN	B032010296	3 BITI S1G1

TABLE OF CONTENT

1.0 INTRODUCTION

The relationship between students and lecturers in higher education can promote and create an environment in which students can learn alot from their lecturers. Nevertheless there are students who wish their lecturer had a better teaching style and lecturers who look forward to improving their teaching style and method to make students understand the subject better. ***What do students think about you?*** is a web-based system to allow lecturers to analyze feedback from their students depicted in the form of word clouds and pie chart using sentiment analysis. Sentiment analysis is a natural language processing (NLP) technique used to determine whether data is positive, negative or neutral. Sentiment analysis is often performed on textual data to help monitor brand and product sentiment in customer feedback, and understand customer needs. We have implemented sentiment analysis in getting feedback from students regarding their lecturers. Target users for this application are lecturer and student. There are 4 subjects available to observe graphs and conclusion for lecturer and drop opinion for students, which are Natural Language Processing (NLP), Artificial Intelligence in Robotic and Automation (AIRA), Artificial Intelligence Project Management (AIPM) and English for Professional Interaction (EPI). For the dataset, we have collected from the students from our course which is BITI, consisting of different classes, 3 BITI S1G1, 3 BITI S1G2, and 3 BITI S2G1 to process data for this application. This application can give advantages for students and lecturers. For students, they can write their opinion of the selected subject without entering their name. For lecturers, this application can help them to improve their teaching method for the upcoming future students.

2.0 ANALYSIS FOR THE DEVELOP SYSTEM

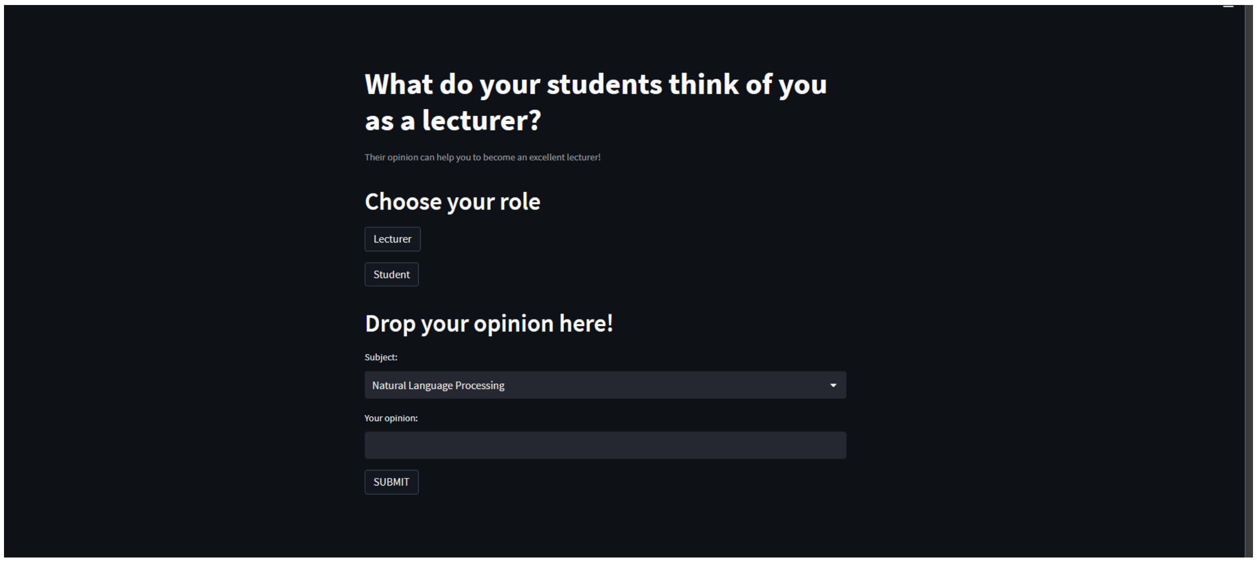
The analysis of the system is technically the process of collecting and interpreting facts, identifying the problems, and decomposition of a system into its components. So for this project, we collected the data by conducting a survey through Google forms for subjects such as BITI3413 Natural Language Processing, BITI3523 Artificial Intelligence in Robotics and Automation, BITI3533 Artificial Intelligence Project Management and BLLW 3162 English for Professional Interaction. The questionnaires were distributed to students from respective classes taking the stated subjects and were required to fill up the form with their opinion about the subjects and how the lecturers convey teaching methods. From this survey, we have collected 100 over feedback which is our data for the system. Then sentiment analysis, feature extraction and data pre-processing was carried out with the data, labeling them into positive and negative feedbacks. The system uses sentiment analysis to separate the negative and positive feedback from students about the lecturer and their teaching method. For example, students' feedbacks' used as a helping tool to improvise the lecturer's teaching methods and lecture notes and this can be displayed on the web page called Streamlit using pie charts and word clouds to show the feedback percentages and frequently used words from the feedback.

Feature extraction was performed to remove and convert the words and phrases into its basic form like removing stopwords, lowercase conversion and stemming. Techniques such as TF-IDF were implemented for the feedback pre-processing part. TF-IDF stands for term frequency-inverse document frequency and it is a measure that can quantify the importance or relevance of string representations like words, phrases, lemmas and etc in a document amongst a collection of documents., in this case the feedback. TF-IDF model contains information on the more important words and the less important ones as well although Bag of Words vectors are easier to interpret. In this case, TF-IDF weighting scheme gives more weight to words which appear in fewer feedback and less weight to words which appear in many feedbacks.

3.0 DESIGN OF THE DEVELOP SYSTEM

Figure below shows the main user interface of *What do students think about you?* sentiment analysis system. System's model and user interface has been designed using Python code and Streamlit web-app.

Student main page



What do your students think of you as a lecturer?

Their opinion can help you to become an excellent lecturer!

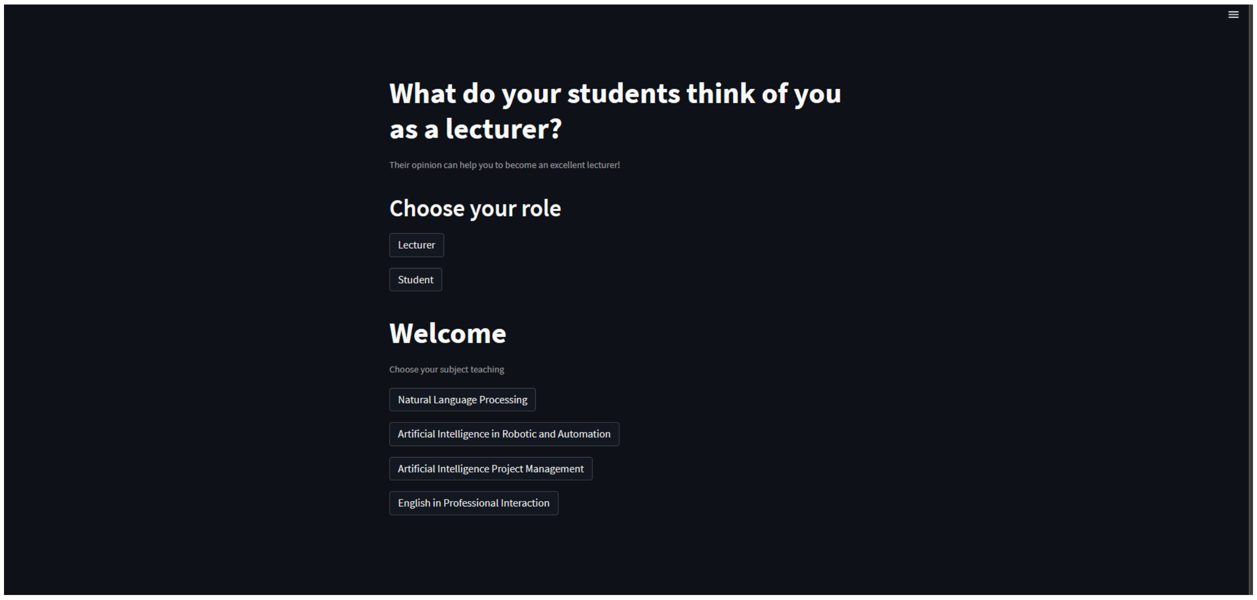
Choose your role

Drop your opinion here!

Subject: Natural Language Processing

Your opinion:

Lecturer main-page:



What do your students think of you as a lecturer?

Their opinion can help you to become an excellent lecturer!

Choose your role

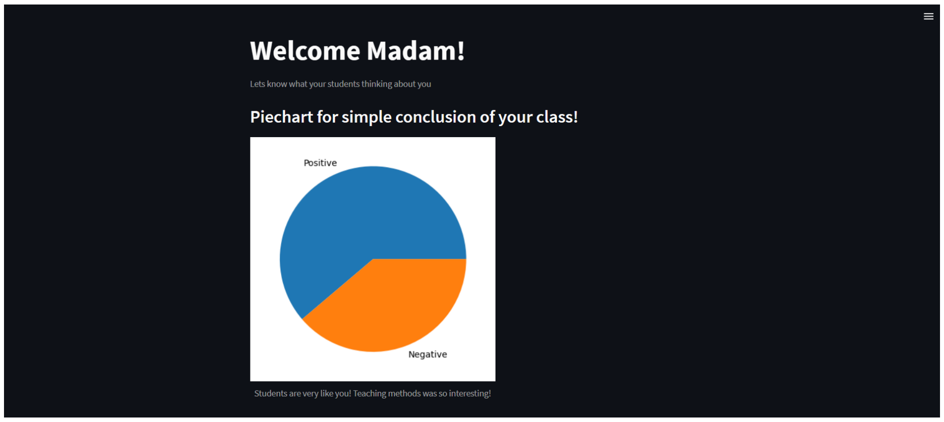
Welcome

Choose your subject teaching

Natural Language Processing
Artificial Intelligence in Robotic and Automation
Artificial Intelligence Project Management
English in Professional Interaction

Lecturer sub-page: Natural Language Processing (sample one of the subject in our application)

Piechart



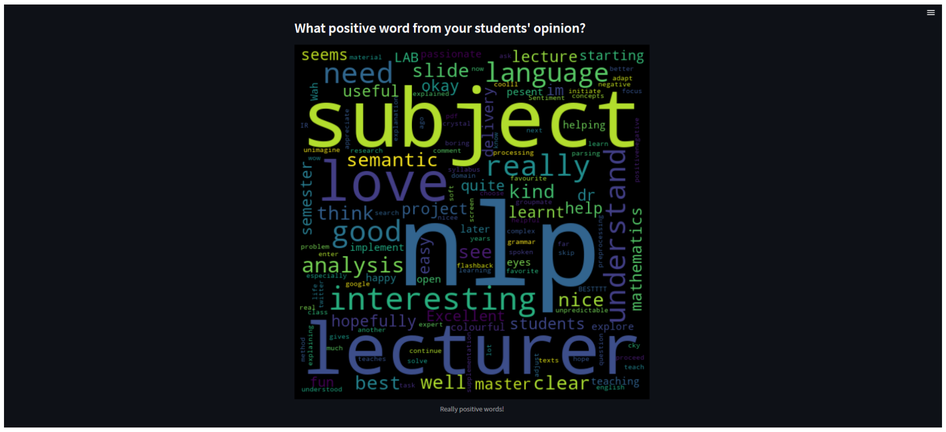
Welcome Madam!

Lets know what your students thinking about you

Piechart for simple conclusion of your class!

Students are very like you! Teaching methods was so interesting!

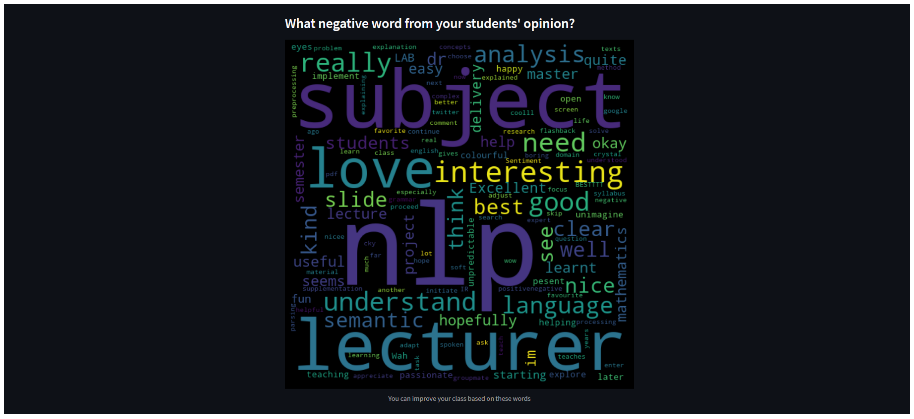
Positive cloud words



What positive word from your students' opinion?

Really positive words!

Negative cloud words

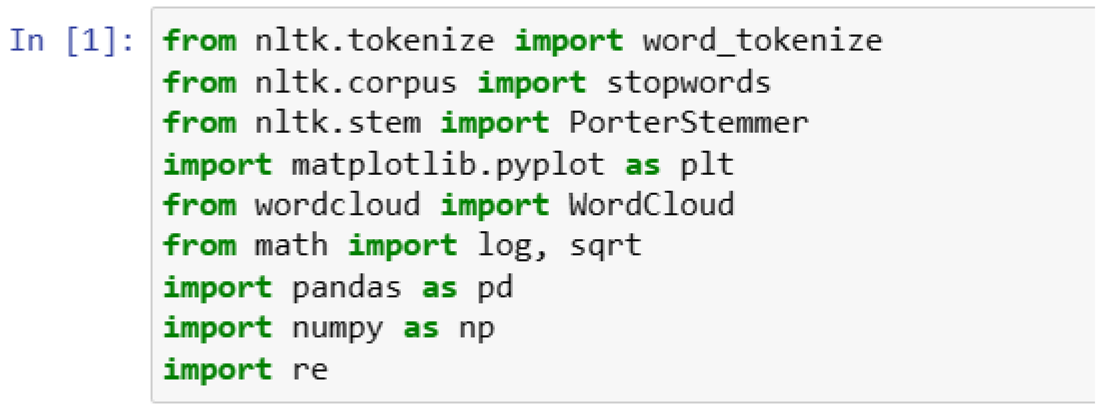


What negative word from your students' opinion?

You can improve your class based on these words

4.0 IMPLEMENTATION

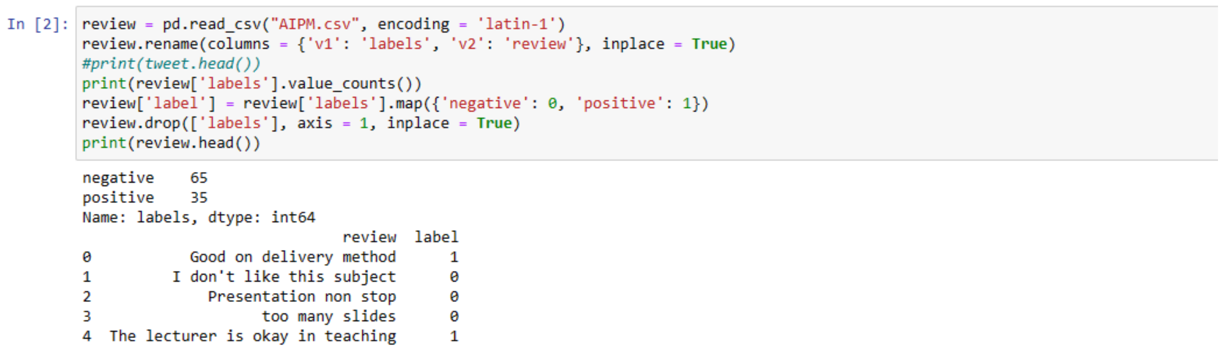
We are going to import our data that we have collected from the google forms into Jupyter Notebook. But first, let’s import the important package.



```
In [1]: from nltk.tokenize import word_tokenize
        from nltk.corpus import stopwords
        from nltk.stem import PorterStemmer
        import matplotlib.pyplot as plt
        from wordcloud import WordCloud
        from math import log, sqrt
        import pandas as pd
        import numpy as np
        import re
```

Figure 1: Import packages in Jupyter Notebook

Next we will import the excel data for the first subject which is Artificial Intelligence in Project Management (AIPM). Jupyter Notebook will execute and process the following codes.



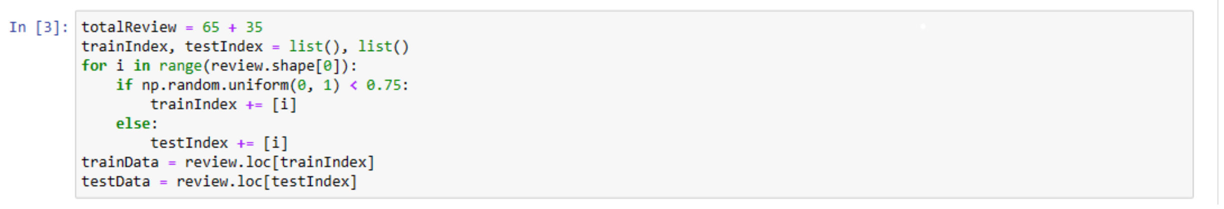
```
In [2]: review = pd.read_csv("AIPM.csv", encoding = 'latin-1')
        review.rename(columns = {'v1': 'labels', 'v2': 'review'}, inplace = True)
        #print(tweet.head())
        print(review['labels'].value_counts())
        review['label'] = review['labels'].map({'negative': 0, 'positive': 1})
        review.drop(['labels'], axis = 1, inplace = True)
        print(review.head())
```

```
negative    65
positive    35
Name: labels, dtype: int64
                             review  label
0            Good on delivery method      1
1          I don't like this subject      0
2             Presentation non stop      0
3                    too many slides      0
4   The lecturer is okay in teaching      1
```

Figure 2: Import data for AIPM subject

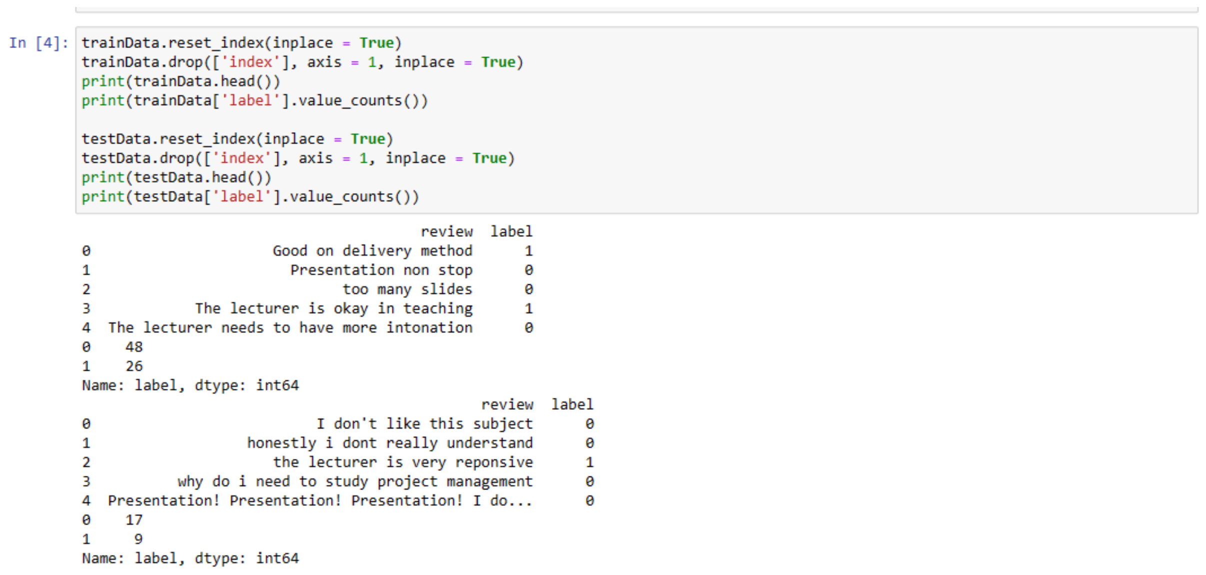
Here, we can see that the first column that is named as ‘v1’ has been changed to labels and the second column ‘v2’ has been changed to message. This is to ease the process of detecting the positive and negative comments from the data. Next the program will create a new column in order to give every comment a label which is either ‘1’ for positive comments or ‘0’ for negative comments.

The next part is, we will split the data into training and testing data. After the data had been splitted, the program will show five comments picked at random for each training and training data.



```
In [3]: totalReview = 65 + 35
        trainIndex, testIndex = list(), list()
        for i in range(review.shape[0]):
            if np.random.uniform(0, 1) < 0.75:
                trainIndex += [i]
            else:
                testIndex += [i]
        trainData = review.loc[trainIndex]
        testData = review.loc[testIndex]
```

Figure 3: Divide the data into training and testing

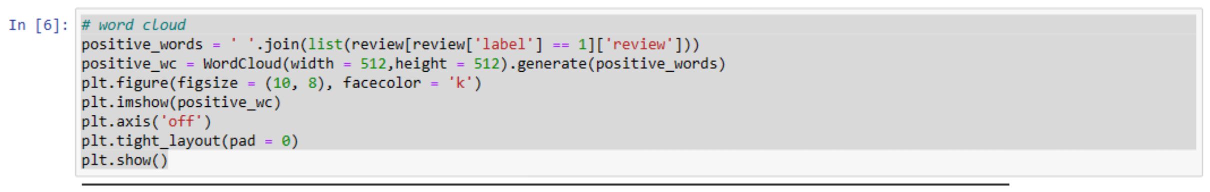


```
In [4]: trainData.reset_index(inplace = True)
        trainData.drop(['index'], axis = 1, inplace = True)
        print(trainData.head())
        print(trainData['label'].value_counts())

        testData.reset_index(inplace = True)
        testData.drop(['index'], axis = 1, inplace = True)
        print(testData.head())
        print(testData['label'].value_counts())
```

```
                                       review  label
0                      Good on delivery method      1
1                        Presentation non stop      0
2                              too many slides      0
3              The lecturer is okay in teaching      1
4  The lecturer needs to have more intonation      0
0    48
1    26
Name: label, dtype: int64
                                               review  label
0                             I don't like this subject      0
1                    honestly i dont really understand      0
2                       the lecturer is very reponsive      1
3            why do i need to study project management      0
4  Presentation! Presentation! Presentation! I do...      0
0    17
1     9
Name: label, dtype: int64
```

Figure 4: Display five comments for each category

After done with the training and testing data, we will proceed with creating a word cloud for both positive and negative comments. This is to see the compilation of words that are used in positive and negative comments. The same process is also done for the negative comments.



```
In [6]: # word cloud
        positive_words = ' '.join(list(review[review['label'] == 1]['review']))
        positive_wc = WordCloud(width = 512,height = 512).generate(positive_words)
        plt.figure(figsize = (10, 8), facecolor = 'k')
        plt.imshow(positive_wc)
        plt.axis('off')
        plt.tight_layout(pad = 0)
        plt.show()
```

Figure 5: Word cloud for positive comments

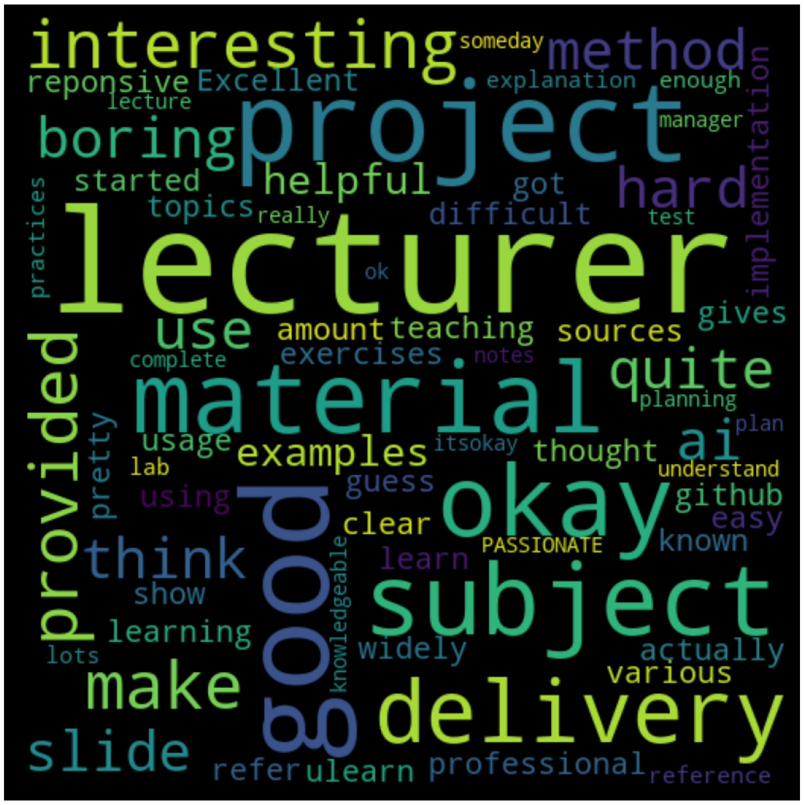
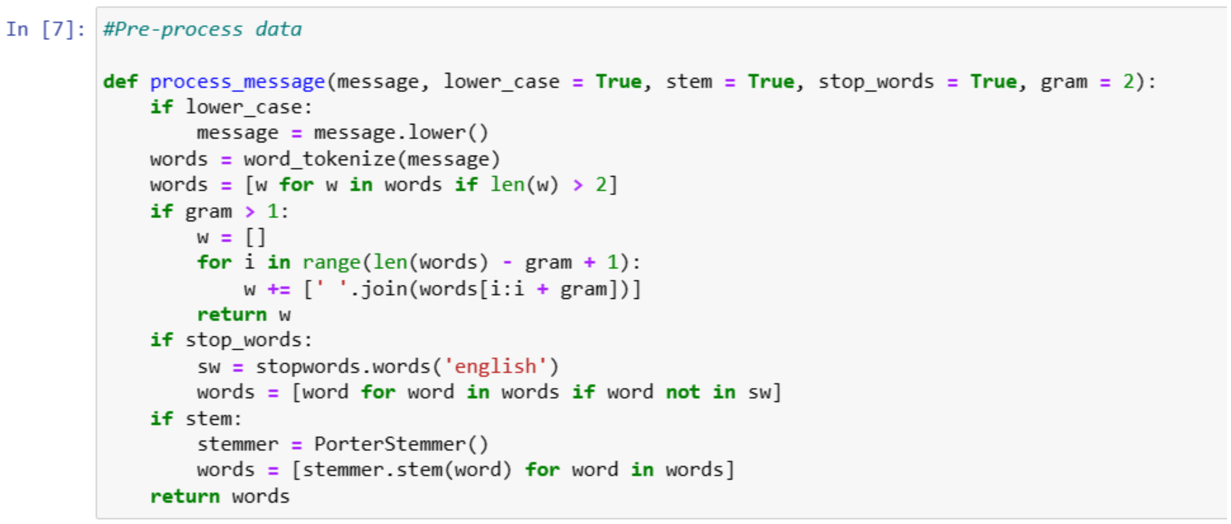


Figure 6: Word cloud for positive comments for AIPM

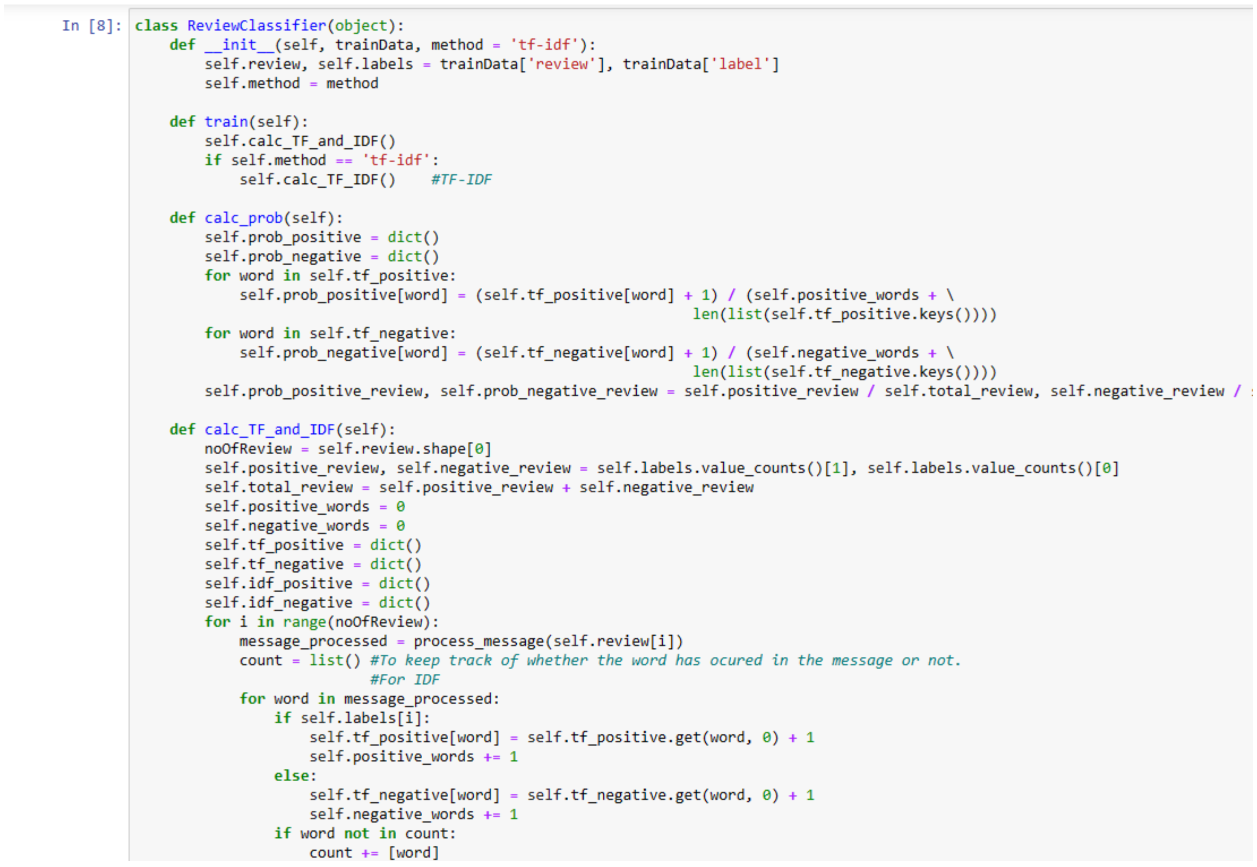
After that, we will pre process the data. First, the data will all be converted into lower case. Why do we do this process? This is because we need to do a stemming process for that data, so converting the data to lowercase will help the process become easier. We will also remove all the stopwords in the comment in order to not process the unnecessary word from the comments.



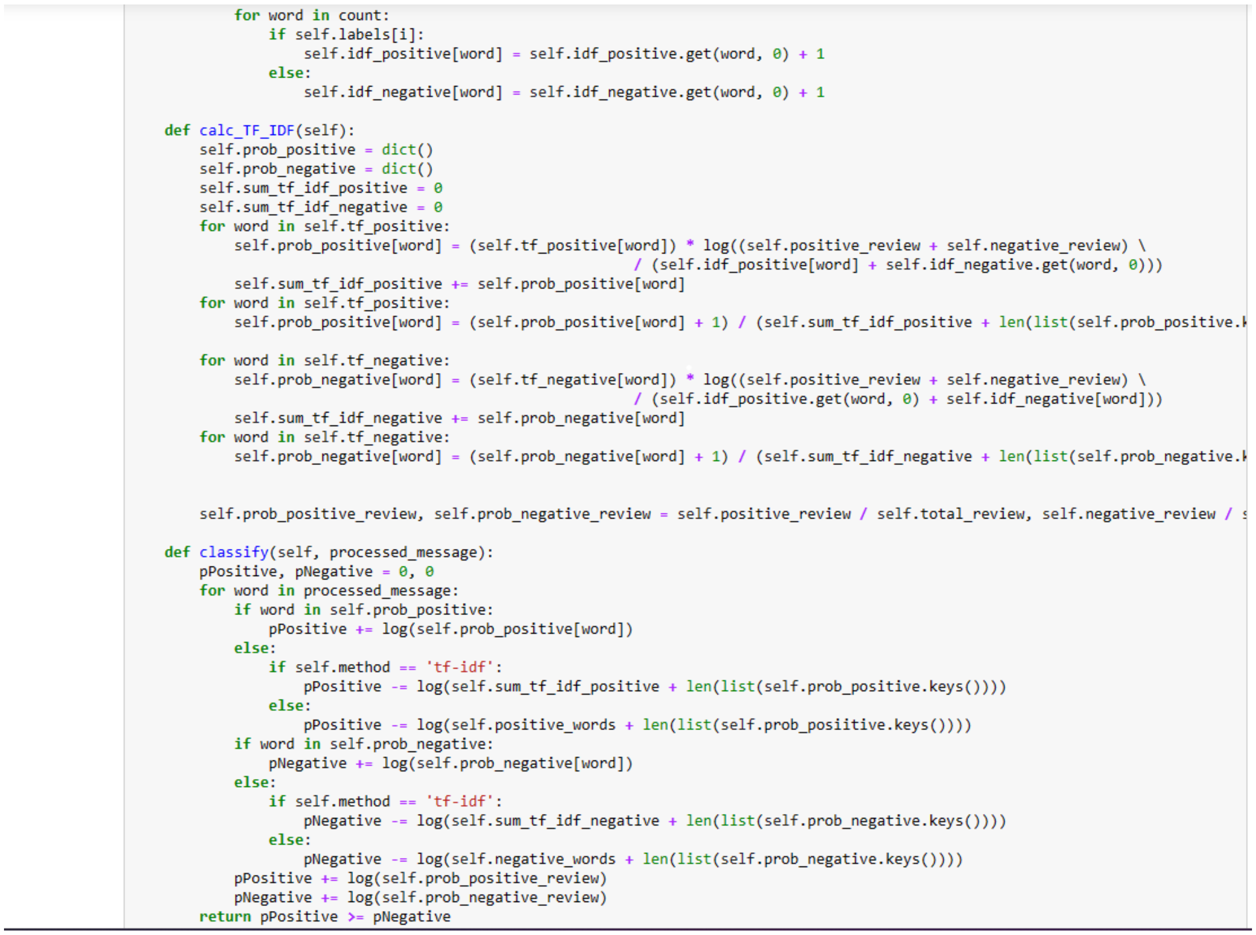
```
In [7]: #Pre-process data

        def process_message(message, lower_case = True, stem = True, stop_words = True, gram = 2):
            if lower_case:
                message = message.lower()
            words = word_tokenize(message)
            words = [w for w in words if len(w) > 2]
            if gram > 1:
                w = []
                for i in range(len(words) - gram + 1):
                    w += [' '.join(words[i:i + gram])]
                return w
            if stop_words:
                sw = stopwords.words('english')
                words = [word for word in words if word not in sw]
            if stem:
                stemmer = PorterStemmer()
                words = [stemmer.stem(word) for word in words]
            return words
```

Figure 7: Pre process the data

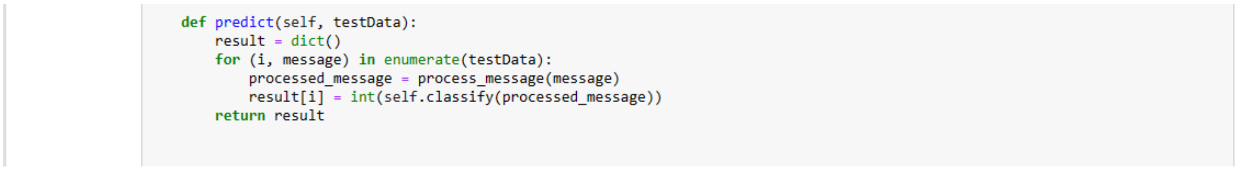
After preprocessing the data, we will do a classification process to the data. This classification will create the calculation method for NLP techniques that are used in this project which is TF-IDF.



```
In [8]: class ReviewClassifier(object):
            def __init__(self, trainData, method = 'tf-idf'):
                self.review, self.labels = trainData['review'], trainData['label']
                self.method = method

            def train(self):
                self.calc_TF_and_IDF()
                if self.method == 'tf-idf':
                    self.calc_TF_IDF()     #TF-IDF

            def calc_prob(self):
                self.prob_positive = dict()
                self.prob_negative = dict()
                for word in self.tf_positive:
                    self.prob_positive[word] = (self.tf_positive[word] + 1) / (self.positive_words + \
                                                                                len(list(self.tf_positive.keys())))
                for word in self.tf_negative:
                    self.prob_negative[word] = (self.tf_negative[word] + 1) / (self.negative_words + \
                                                                                len(list(self.tf_negative.keys())))
                self.prob_positive_review, self.prob_negative_review = self.positive_review / self.total_review, self.negative_review /

            def calc_TF_and_IDF(self):
                noOfReview = self.review.shape[0]
                self.positive_review, self.negative_review = self.labels.value_counts()[1], self.labels.value_counts()[0]
                self.total_review = self.positive_review + self.negative_review
                self.positive_words = 0
                self.negative_words = 0
                self.tf_positive = dict()
                self.tf_negative = dict()
                self.idf_positive = dict()
                self.idf_negative = dict()
                for i in range(noOfReview):
                    message_processed = process_message(self.review[i])
                    count = list() #To keep track of whether the word has ocured in the message or not.
                                   #For IDF
                    for word in message_processed:
                        if self.labels[i]:
                            self.tf_positive[word] = self.tf_positive.get(word, 0) + 1
                            self.positive_words += 1
                        else:
                            self.tf_negative[word] = self.tf_negative.get(word, 0) + 1
                            self.negative_words += 1
                        if word not in count:
                            count += [word]
```



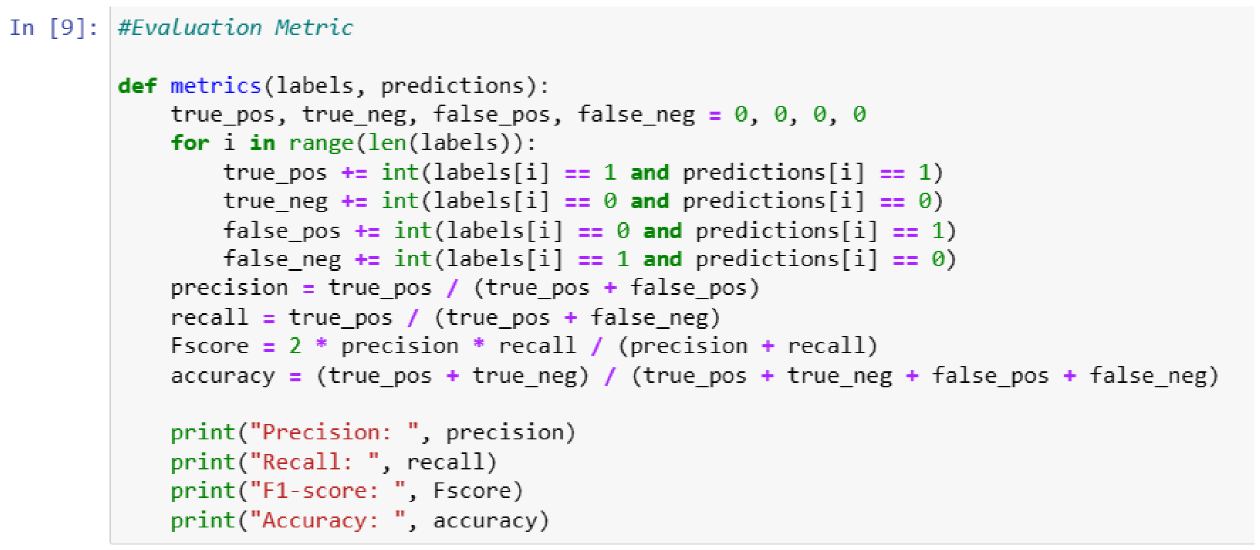
```
            for word in count:
                if self.labels[i]:
                    self.idf_positive[word] = self.idf_positive.get(word, 0) + 1
                else:
                    self.idf_negative[word] = self.idf_negative.get(word, 0) + 1

    def calc_TF_IDF(self):
        self.prob_positive = dict()
        self.prob_negative = dict()
        self.sum_tf_idf_positive = 0
        self.sum_tf_idf_negative = 0
        for word in self.tf_positive:
            self.prob_positive[word] = (self.tf_positive[word]) * log((self.positive_review + self.negative_review) \
                                                          / (self.idf_positive[word] + self.idf_negative.get(word, 0)))
            self.sum_tf_idf_positive += self.prob_positive[word]
        for word in self.tf_positive:
            self.prob_positive[word] = (self.prob_positive[word] + 1) / (self.sum_tf_idf_positive + len(list(self.prob_positive.k

        for word in self.tf_negative:
            self.prob_negative[word] = (self.tf_negative[word]) * log((self.positive_review + self.negative_review) \
                                                          / (self.idf_positive.get(word, 0) + self.idf_negative[word]))
            self.sum_tf_idf_negative += self.prob_negative[word]
        for word in self.tf_negative:
            self.prob_negative[word] = (self.prob_negative[word] + 1) / (self.sum_tf_idf_negative + len(list(self.prob_negative.k


        self.prob_positive_review, self.prob_negative_review = self.positive_review / self.total_review, self.negative_review / s

    def classify(self, processed_message):
        pPositive, pNegative = 0, 0
        for word in processed_message:
            if word in self.prob_positive:
                pPositive += log(self.prob_positive[word])
            else:
                if self.method == 'tf-idf':
                    pPositive -= log(self.sum_tf_idf_positive + len(list(self.prob_positive.keys())))
                else:
                    pPositive -= log(self.positive_words + len(list(self.prob_posiitive.keys())))
            if word in self.prob_negative:
                pNegative += log(self.prob_negative[word])
            else:
                if self.method == 'tf-idf':
                    pNegative -= log(self.sum_tf_idf_negative + len(list(self.prob_negative.keys())))
                else:
                    pNegative -= log(self.negative_words + len(list(self.prob_negative.keys())))
            pPositive += log(self.prob_positive_review)
            pNegative += log(self.prob_negative_review)
        return pPositive >= pNegative
```



```
    def predict(self, testData):
        result = dict()
        for (i, message) in enumerate(testData):
            processed_message = process_message(message)
            result[i] = int(self.classify(processed_message))
        return result
```

Figure 8: Classification process

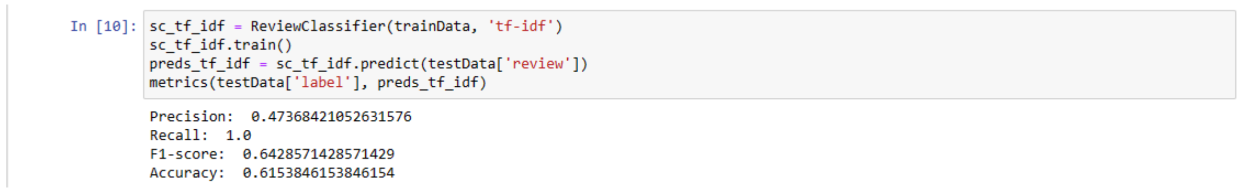
Next, we will do an evaluation metric to get the value of Precision, Recall, F-1 Score and also the Accuracy for both NLP techniques. This is shown as follows.



```
In [9]: #Evaluation Metric

        def metrics(labels, predictions):
            true_pos, true_neg, false_pos, false_neg = 0, 0, 0, 0
            for i in range(len(labels)):
                true_pos += int(labels[i] == 1 and predictions[i] == 1)
                true_neg += int(labels[i] == 0 and predictions[i] == 0)
                false_pos += int(labels[i] == 0 and predictions[i] == 1)
                false_neg += int(labels[i] == 1 and predictions[i] == 0)
            precision = true_pos / (true_pos + false_pos)
            recall = true_pos / (true_pos + false_neg)
            Fscore = 2 * precision * recall / (precision + recall)
            accuracy = (true_pos + true_neg) / (true_pos + true_neg + false_pos + false_neg)

            print("Precision: ", precision)
            print("Recall: ", recall)
            print("F1-score: ", Fscore)
            print("Accuracy: ", accuracy)
```

Figure 9: Evaluation Metric

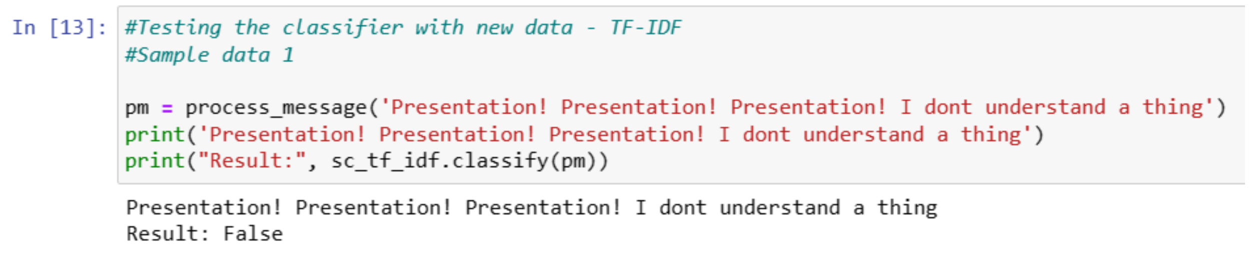
For TF-IDF technique, after implementing the classifier and performing the evaluation metric, we get the result for the Precision = 0.47, Recall = 1.0 , F-1 Score = 0.64 and the Accuracy = 0.61 for AIPM subject.



```
In [10]: sc_tf_idf = ReviewClassifier(trainData, 'tf-idf')
         sc_tf_idf.train()
         preds_tf_idf = sc_tf_idf.predict(testData['review'])
         metrics(testData['label'], preds_tf_idf)

         Precision:  0.47368421052631576
         Recall:  1.0
         F1-score:  0.6428571428571429
         Accuracy:  0.6153846153846154
```

Figure 10: Feature Extraction for TF-IDF

We also tested this technique with some other examples aside from the chosen comments to see whether it can predict the labels correctly or not. Figure below shows the result.



```
In [13]: #Testing the classifier with new data - TF-IDF
         #Sample data 1

         pm = process_message('Presentation! Presentation! Presentation! I dont understand a thing')
         print('Presentation! Presentation! Presentation! I dont understand a thing')
         print("Result:", sc_tf_idf.classify(pm))

         Presentation! Presentation! Presentation! I dont understand a thing
         Result: False
```

Figure 11: Testing sample data for TF-IDF

The same method is applied for another three subjects which are AIRA, EPI and also NLP. We have calculated the Precision value, Recall value, F-1 score and Accuracy for all four subjects.

5.0 CONCLUSION

The strength of the system is that it uses sentiment analysis to represent feedbacks' received from students in a simple form like pie charts and word clouds for users, in this case lecturers to get a better understanding of students' opinions about them. This system can also reduce paper usage instead of collecting data manually from students and is less time consuming for lecturers to get the feedback about their method of teaching rather than checking one by one.

Despite its strengths, the system also has one downside which is it is too simple and straightforward whereby the students can also access the lecturer pages. This is because the login page is not developed by us as we are only portraying the prototype.

Suggestions to enhance this web-based system that we can improve the design of the system to make it more interesting and user friendly. Furthermore, we can include the prediction module that can predict the future feedback from the student based on its pattern. This can give a valuable insight to the lecturer to strategize for the future planning. Moreover, a recommendation module should also be one of the modules to be considered adding to give suggestions to the lecturer on how to improve the feedback.

Upon completing the system, we have found that sentiment analysis is indeed very important in Natural Language Processing. Not only it helps in monitoring the lecturer performance but it also helps in categorizing the opinions expressed by the student. From this data, we can use it to perform data visualization such as area chart, histogram and map visualization.

In conclusion, this web-based system can definitely be improved for future use. With the improvement made, this system has the potential to be promoted at school, college or university to improve the education system. Additionally, the feedback from students is important to improve the performance of the lecturer and to provide a method of facilitating development. The development in the education area will then determine the success of our education system.