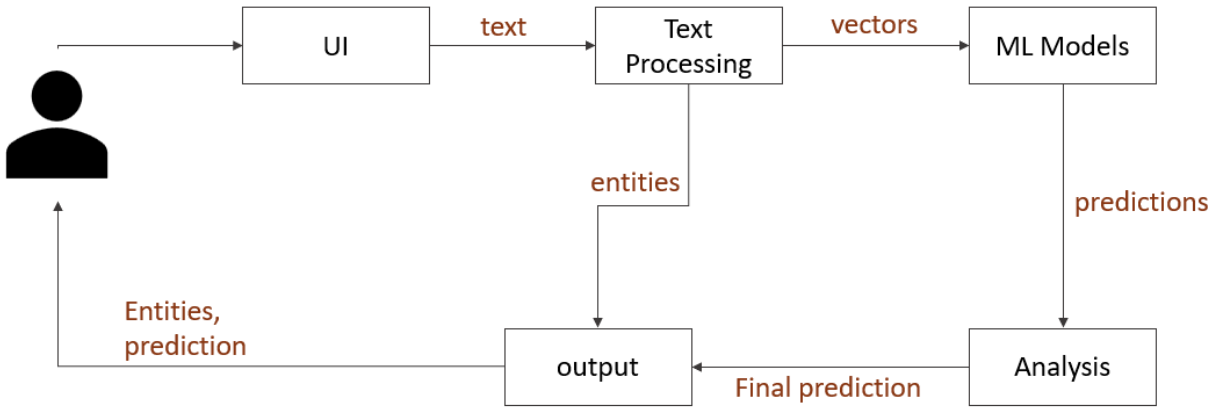
DEVELOPMENT PHASE 3

System Architecture

The application overview has been presented below and it gives a basic structure of the application.



Working Procedure

The working procedure includes the internal working and the data flow of application.

i. After running the application some procedures are automated.

- Reading data from file
- Cleaning the texts
- Processing
- Splitting the data
- Intialising and training the models

ii. The user just needs to provide some data to classify in the area provided.

iii. The provided data undergoes several procedures after submission.

- Textual Processing
- Feature Vector conversion
- Entity extraction

iv. The created vectors are provided to trained models to get predictions.

v. After getting predictions the category predicted by majority will be selected.

vi. The accuracies of that prediction will be calculated

vii. The accuracies and entities extracted from the step 3 will be provided to user.

Every time the user gives something new the procedure from step 2 will be repeated.

References

[1] S. H. a. M. A. T. Toma, "An Analysis of Supervised Machine Learning Algorithms for Spam Email Detection," in International Conference on Automation, Control and

Mechatronics for Industry 4.0 (ACMI), 2021.
[2] S. Nandhini and J. Marseline K.S., "Performance Evaluation of Machine Learning Algorithms for Email Spam Detection," in International Conference on Emerging Trends in Information Technology and Engineering (ic-ETITE), 2020.
[3] A. L. a. S. S. S. Gadde, "SMS Spam Detection using Machine Learning and Deep Learning Techniques," in 7th International Conference on Advanced Computing and Communication Systems (ICACCS), 2021, 2021.
[4] V. B. a. B. K. P. Sethi, "SMS spam detection and comparison of various machine learning algorithms," in International Conference on Computing and Communication Technologies for Smart Nation (IC3TSN), 2017.
[5] G. D. a. A. R. P. Navaney, "SMS Spam Filtering Using Supervised Machine Learning Algorithms," in 8th International Conference on Cloud Computing, Data Science & Engineering (Confluence), 2018.
[6] S. O. Olatunji, "Extreme Learning Machines and Support Vector Machines models for email spam detection," in IEEE 30th Canadian Conference on Electrical and Computer Engineering (CCECE), 2017.
[7] S. S. a. N. N. Kumar, "Email Spam Detection Using Machine Learning Algorithms," in Second International Conference on Inventive Research in Computing Applications (CIRCA), 2020.
[8] R. Madan, "medium.com," [Online]. Available: https://medium.com/analytics-vidhya/tf-idf-term-frequency-technique-easiest-explanation-for-text-classification-in-nlp-with-code-8ca3912e58c3.

A. Source code

1. Module – Data Processing

```
import re
from nltk.tokenize import sent_tokenize,word_tokenize
from nltk import pos_tag
from nltk.corpus import wordnet as wn
from nltk.corpus import stopwords
from nltk.stem.wordnet import WordNetLemmatizer
from collections import defaultdict
import spacy

tag_map = defaultdict(lambda : wn.NOUN)
tag_map['J'] = wn.ADJ
tag_map['V'] = wn.VERB
tag_map['R'] = wn.ADV
lemmatizer=WordNetLemmatizer()
stop_words=set(stopwords.words('english'))

nlp=spacy.load('en_core_web_sm')

def process_sentence(sentence):
        nouns = list()
        base_words = list()
        final_words = list()
```

```
        words_2 = word_tokenize(sentence)
        sentence = re.sub(r'[^ \w\s]', '', sentence)
        sentence = re.sub(r'_', ' ', sentence)
        words = word_tokenize(sentence)
        pos_tagged_words = pos_tag(words)

        for token, tag in pos_tagged_words:
base_words.append(lemmatizer.lemmatize(token,tag_map[tag[0]]))
        for word in base_words:
                if word not in stop_words:
                        final_words.append(word)
        sym = ' '
        sent = sym.join(final_words)
        pos_tagged_sent = pos_tag(words_2)
        for token, tag in pos_tagged_sent:
                if tag == 'NN' and len(token)>1:
                        nouns.append(token)
        return sent, nouns

def clean(email):
        email = email.lower()
        sentences = sent_tokenize(email)
        total_nouns = list()
        string = ""
        for sent in sentences:
                sentence, nouns = process_sentence(sent)
                string += " " + sentence
                total_nouns += nouns
        return string, nouns


def ents(text):
        doc = nlp(text)
        expls = dict()
        if doc.ents:
                for ent in doc.ents:
                        labels = list(expls.keys())
                        label = ent.label_
                        word = ent.text
                        if label in labels:
                                words = expls[label]
                                words.append(word)
                                expls[label] = words
                        else:
                                expls[label] = [word]
                return expls
        else:
                return 'no'
```