**USE CASE DIAGRAM:**

A use case diagram in the Unified Modeling Language (UML) is a type of behavioral diagram defined by and created from a Use-case analysis. Its purpose is to present a graphical overview of the functionality provided by a system in terms of actors, their goals (represented as use cases), and any dependencies between those use cases. The main purpose of a use case diagram is to show what system functions are performed for which actor. Roles of the actors in the system can be depicted.

User



Admin



**CLASS DIAGRAM:**

In software engineering, a class diagram in the Unified Modeling Language (UML) is a type of static structure diagram that describes the structure of a system by showing the system's classes, their attributes, operations (or methods), and the relationships among the classes. It explains which class contains information.

****

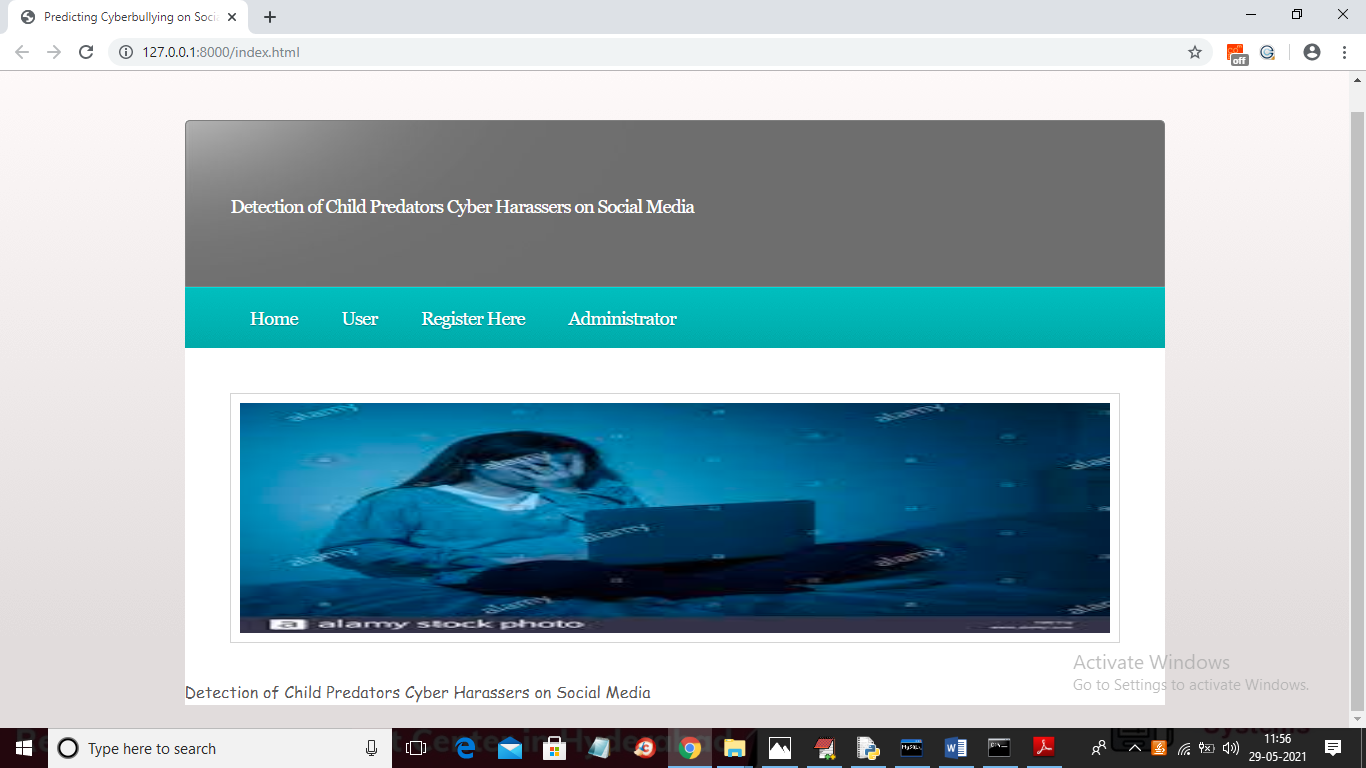
**SEQUENCE DIAGRAM:**

A sequence diagram in Unified Modeling Language (UML) is a kind of interaction diagram that shows how processes operate with one another and in what order. It is a construct of a Message Sequence Chart. Sequence diagrams are sometimes called event diagrams, event scenarios, and timing diagrams.

****

**COLLABORATION DIAGRAM:**





import numpy as np

from sklearn import svm

import pandas as pd

from sklearn.metrics import accuracy\_score

from sklearn.model\_selection import train\_test\_split

import nltk

from nltk.corpus import stopwords

from sklearn.metrics import classification\_report

from sklearn.metrics import confusion\_matrix

from sklearn.ensemble import RandomForestClassifier

from sklearn.naive\_bayes import MultinomialNB

from sklearn.tree import DecisionTreeClassifier

from sklearn.neighbors import KNeighborsClassifier

from sklearn.ensemble import BaggingClassifier

from sklearn\_extensions.extreme\_learning\_machines.elm import GenELMClassifier

from sklearn\_extensions.extreme\_learning\_machines.random\_layer import RBFRandomLayer, MLPRandomLayer

from sklearn.linear\_model import LogisticRegression

from django.core.files.storage import FileSystemStorage

import datetime

from numpy.linalg import norm

from numpy import dot

global classifier

global label\_count

global X

global Y

corpus = []

def index(request):

if request.method == 'GET':

return render(request, 'index.html', {})

def SendPost(request):

if request.method == 'GET':

return render(request, 'SendPost.html', {})

def Register(request):

if request.method == 'GET':

return render(request, 'Register.html', {})