TITLE OF THE PROJECT

FLUTO - MESSENGER APPLICATION

MINI PROJECT REPORT

Submitted in partial fulfillment of the requirements

for the award of the degree of

BACHELOR OF ENGINEERING

IN

INFORMATION TECHNOLOGY

By

<K.Asha><1602-19-737-005>
<K.Pallavi Suma><1602-19-737-025>
<P.Keerthana Reddy><1602-19-737-183>

Department of Information Technology
Vasavi College of Engineering (Autonomous)
(Affiliated to Osmania University)
Ibrahimbagh, Hyderabad – 31

BONAFIDE CERTIFICATE

This is to certify that the project report titled “FLUTO - MESSENGER APPLICATION” project work of K.Asha, K.Pallavi Suma and P.Keerthana Reddy bearing roll numbers 1602-19-737-005,1602-19-737-025 and 1602-19-737-183 respectively who carried out this project under my supervision in the VI semester for the academic year 2021-2022.

SIGNATURE
Internal Examiner

SIGNATURE
External Examiner

TABLE OF CONTENTS

ABSTRACT

In the present generation there are many more social media applications where we can interact with each other. People who are in social media are prone to mental abuses by unwanted messages , spam messages, threat and hate messages. Due to these problems, people are facing many severe issues.Our Fluto messenger Application is an attempt to free people from such situations by categorizing messages into spam messages and asking the user's permission for viewing them.

INTRODUCTION

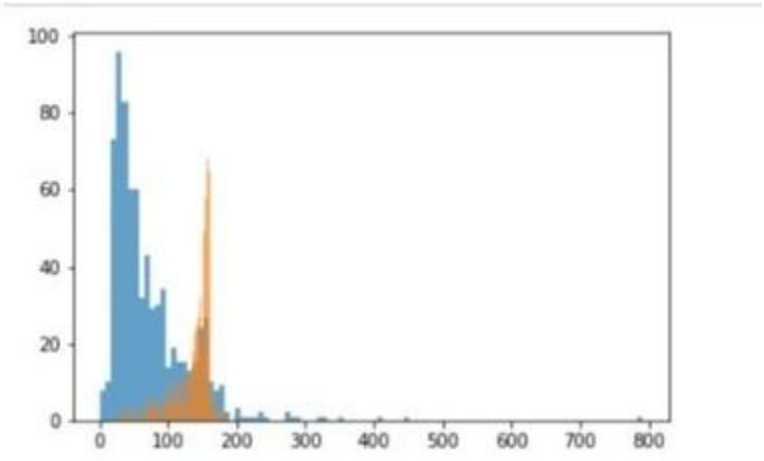
The main feature of our messenger application “Fluto” is categorizing spam messages in the messaging or chat area .It is a user-friendly social media application which is used for communicating with others.It can be also integrated for voice calling, image file sending, emojis sending, gif sending etc. The user interface and working of the application is designed in such a way that is easy to understand and easy to use for any age groups.

LITERATURE REVIEW

The main objective of our project and research is to offer users a comfortable,user-friendly web messenger application with a new feature of spam message detection by infusing new technology such as machine learning techniques and algorithms. So we have chosen support vector algorithm and randomforest algorithm to solve our problem statement.

For the spam message detection we used machine learning algorithms like Random Forest, Support Vector Algorithm(SVM), and analysed the accuracy of the both algorithms and predicted the results. We used spam.tsv dataset which is imported from website kaggle.com. We also tested the model with real time data chat from WhatsApp Messenger, some of them worked while the other didnot.

Accuracies of the SVM is noted to be 95% which is more than Random Forest accuracy that is 93%. Hence,SVM is more suitable for classification of spam message detection.



EXISTING METHODS/SYSTEMS

While there are many existing methods/systems of spam detections in emails and SMS but almost none of the system has incorporated a method of spam detection in an messenger application, which makes our project one of its kind.Technical papers such as Neural network for spam detection in emails by navya jyoti journal which focuses on using backpropagation algorithm to classify mails under spam, while sms spam detection using machine learning by iopscience organisation through text classification and TF-TDF vectorization used in machine learningas a weighting factor and for identifying words features. These existing paper works have given us an idea of how to implement spam detection in a messenger application,which algorithm should be used for better accuracy, how to obtain data and how to classify messages under spam. Drawbacks:

- No existing system of spam detection in messenger app
- Overall accuracy of existing works is around 90-95%.
- No proper balance between accuracies of spam(unsafe) and ham(safe)which means and algorithm works fine with one but not upto the mark with the other.

REQUIREMENTS/SPECIFICATIONS

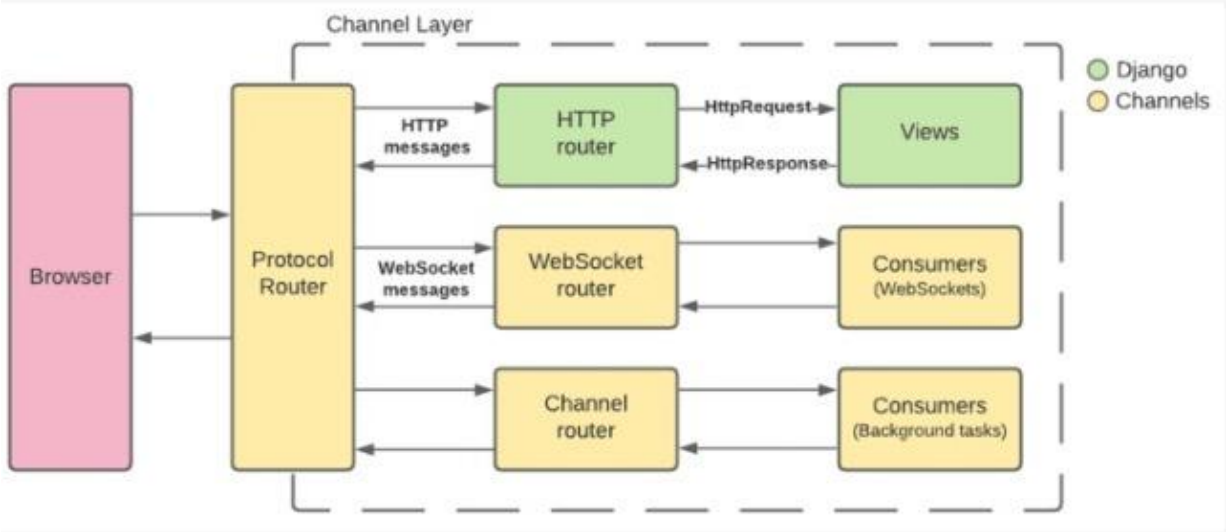
- HTML and CSS
- Java Script
- Django Framework
- Jupyter Notebook - for python
- Libraries -
    - ➔ numpy
    - ➔ Sklearn
    - ➔ Mathplot
    - ➔ Pandas

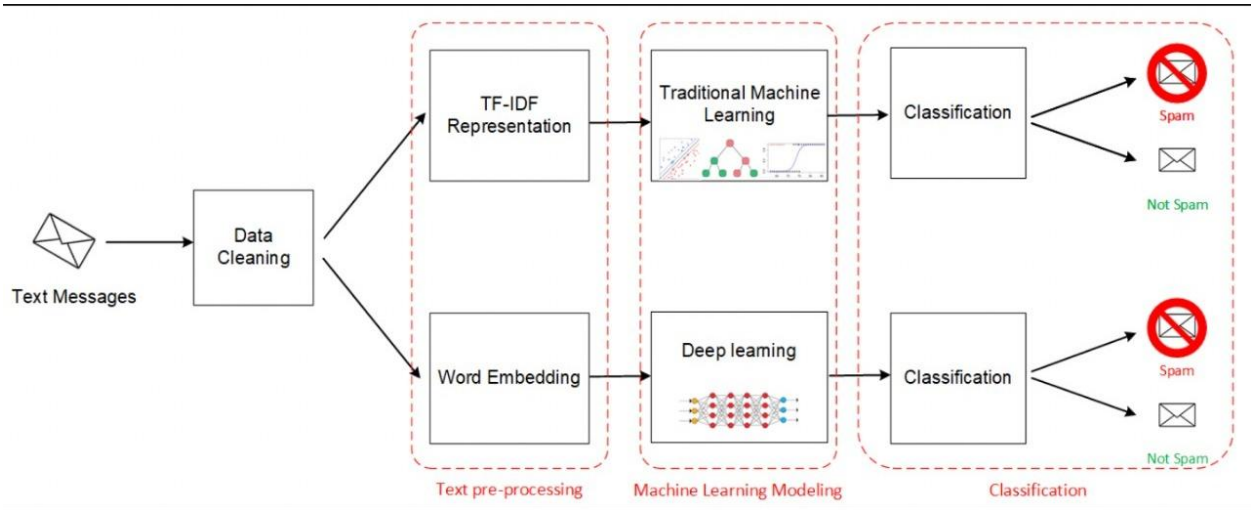
PROPOSED METHOD

For the messenger application we have used Django framework to develop web messenger. Technologies used in Django are channels- which facilitate support of web sockets similar to traditional HTTP views. Django's native asynchronous view support, allowing Django projects to handle not only HTTP, but also protocols that require long-running connections, such as WebSockets, MQTT, chatbots.Further to store the messages we used Django database.

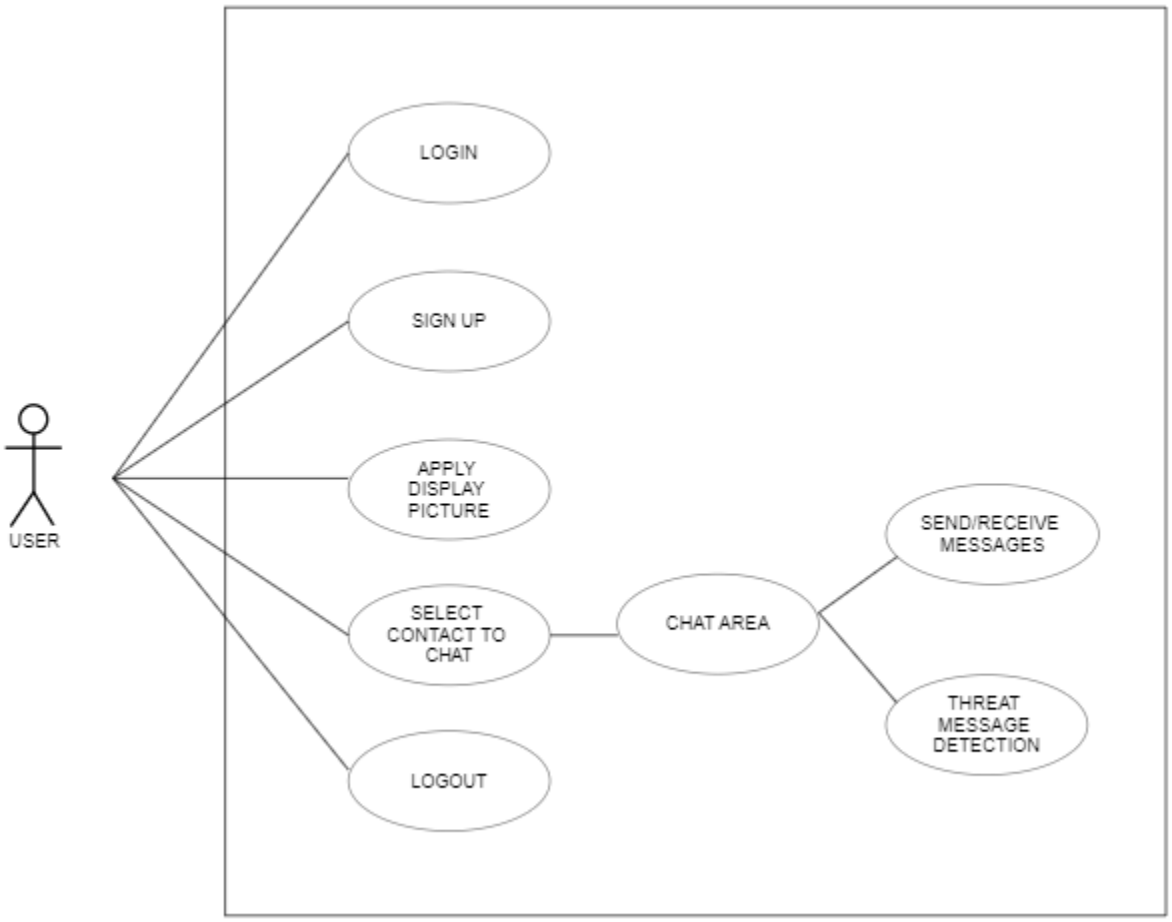
For the spam message detection we used machine learning algorithms like Random Forest, Support Vector Algorithm(SVM), and analysed the accuracy of the both algorithms and predicted the results. We used spam.tsv dataset which is imported from website kaggle.com. We also tested the model with real time data chat from WhatsApp Messenger, some of them worked while the other didnot. Accuracies of the SVM is noted to be more than Random Forest. Hence,SVM is more suitable for classification of spam message detection.

ARCHITECTURE





USE CASE DIAGRAM



IMPLEMENTATION

Jagochat .py files

Asgi.py

```
import os

import django

from channels.auth import AuthMiddlewareStack

from channels.routing import ProtocolTypeRouter, URLRouter

from django.core.asgi import get_asgi_application

from room import routing

# Initialize Django ASGI application early to ensure the AppRegistry

# is populated before importing code that may import ORM models.

os.environ.setdefault("DJANGO_SETTINGS_MODULE",
"jangochat.settings")

application = ProtocolTypeRouter({

    "http": get_asgi_application(),

    "websocket": AuthMiddlewareStack(

        URLRouter(

            Routing.websocket_urlpatterns

)

    )

})
```

Settings.py

"""

Django settings for jangochat project.

Generated by 'django-admin startproject' using Django 4.0.4.

For more information on this file, see

https://docs.djangoproject.com/en/4.0/topics/settings/

For the full list of settings and their values, see

https://docs.djangoproject.com/en/4.0/ref/settings/

"""

```
from pathlib import Path

# Build paths inside the project like this: BASE_DIR / 'subdir'.

BASE_DIR = Path(__file__).resolve().parent.parent

# Quick-start development settings - unsuitable for production

# See https://docs.djangoproject.com/en/4.0/howto/deployment/checklist/

# SECURITY WARNING: keep the secret key used in production secret!

SECRET_KEY =
'django-insecure-l$%swp17j!5dsv5hz8s=)jrid$rg2x!%en@xq3hg866fgu9%t$
'

# SECURITY WARNING: don't run with debug turned on in production!

DEBUG = True

ALLOWED_HOSTS = []

LOGOUT_REDIRECT_URL='/'

LOGIN_REDIRECT_URL='/rooms/'
```

```
LOGIN_URL='/login/'
# Application definition
INSTALLED_APPS = [
    'django.contrib.admin',
    'django.contrib.auth',
    'django.contrib.contenttypes',
    'django.contrib.sessions',
    'django.contrib.messages',
    'django.contrib.staticfiles',
    'channels',
    'core',
    'room',
]
MIDDLEWARE = [
    'django.middleware.security.SecurityMiddleware',
    'django.contrib.sessions.middleware.SessionMiddleware',
    'django.middleware.common.CommonMiddleware',
    'django.middleware.csrf.CsrfViewMiddleware',
    'django.contrib.auth.middleware.AuthenticationMiddleware',
    'django.contrib.messages.middleware.MessageMiddleware',
    'django.middleware.clickjacking.XFrameOptionsMiddleware',
]
```

```
ROOT_URLCONF = 'jangochat.urls'
ASGI_APPLICATION = 'jangochat.routing.application'
TEMPLATES = [
    {
        'BACKEND': 'django.template.backends.django.DjangoTemplates',
        'DIRS': [],
        'APP_DIRS': True,
        'OPTIONS': {
            'context_processors': [
                'django.template.context_processors.debug',
                'django.template.context_processors.request',
                'django.contrib.auth.context_processors.auth',
                'django.contrib.messages.context_processors.messages',
            ],
        },
    },
]
WSGI_APPLICATION = 'jangochat.wsgi.application'
ASGI_APPLICATION='jangochat.asgi.application'

CHANNEL_LAYERS={
    'default': {
```

```
        'BACKEND':'channels.layers.InMemoryChannelLayer'
    }
}
# Database
# https://docs.djangoproject.com/en/4.0/ref/settings/#databases
DATABASES = {
    'default': {
        'ENGINE': 'django.db.backends.sqlite3',
        'NAME': BASE_DIR / 'db.sqlite3',
    }
}
# Password validation
# https://docs.djangoproject.com/en/4.0/ref/settings/#auth-password-validators


AUTH_PASSWORD_VALIDATORS = [
    {
        'NAME': 'django.contrib.auth.password_validation.UserAttributeSimilarityValidator',
    },
    {
        'NAME': 'django.contrib.auth.password_validation.MinimumLengthValidator',
```

```
    },
    {
        'NAME':
'django.contrib.auth.password_validation.CommonPasswordValidator',
    },
    {
        'NAME':
'django.contrib.auth.password_validation.NumericPasswordValidator',
    },
]
# Internationalization
# https://docs.djangoproject.com/en/4.0/topics/i18n/
LANGUAGE_CODE = 'en-us'
TIME_ZONE = 'UTC'
USE_I18N = True
USE_TZ = True
# Static files (CSS, JavaScript, Images)
# https://docs.djangoproject.com/en/4.0/howto/static-files/
STATIC_URL = 'static/'
# Default primary key field type
# https://docs.djangoproject.com/en/4.0/ref/settings/#default-auto-field
DEFAULT_AUTO_FIELD = 'django.db.models.BigAutoField'
```

Urls.py

```
"""jangochat URL Configuration
The `urlpatterns` list routes URLs to views. For more information please see:
    https://docs.djangoproject.com/en/4.0/topics/http/urls/
Examples:
Function views
    1. Add an import:  from my_app import views
    2. Add a URL to urlpatterns:  path('', views.home, name='home')
Class-based views
    1. Add an import:  from other_app.views import Home
    2. Add a URL to urlpatterns:  path('', Home.as_view(), name='home')
Including another URLconf
    1. Import the include() function: from django.urls import include, path
    2. Add a URL to urlpatterns:  path('blog/', include('blog.urls'))
"""
from django.contrib import admin
from django.urls import path, include
urlpatterns = [
    path('',include('core.urls')),
    path('rooms/', include('room.urls')),
    path('admin/', admin.site.urls),
]
```

Wsgi.py

"""

WSGI config for jangochat project.

It exposes the WSGI callable as a module-level variable named ``application``.

For more information on this file, see

https://docs.djangoproject.com/en/4.0/howto/deployment/wsgi/

"""

```
import os

from django.core.wsgi import get_wsgi_application

os.environ.setdefault('DJANGO_SETTINGS_MODULE', 'jangochat.settings')

application = get_wsgi_application()
```

Room .py files

Admin.py

```
from django.contrib import admin

# Register your models here.

from .models import Room

admin.site.register(Room)
```

apps.py

```
from django.apps import AppConfig

class RoomConfig(AppConfig):
    default_auto_field = 'django.db.models.BigAutoField'
```

name = 'room'

Consumers.py

```
from copyreg import pickle
import json
from channels.generic.websocket import AsyncWebsocketConsumer
from asgiref.sync import sync_to_async
from django.contrib.auth.models import User
from .models import Message, Room
class ChatConsumer(AsyncWebsocketConsumer):
    async def connect(self):
        self.room_name = self.scope['url_route']['kwargs']['room_name']
        self.room_group_name = 'chat_%s' % self.room_name

        await self.channel_layer.group_add(

            self.room_group_name,
            self.channel_name
        )
        await self.accept()

    async def disconect(self):
        await self.channel_layer.group_discard(
```

```
            self.room_group_name,
            self.channel_name,
        )
async def receive(self, text_data):
        data = json.loads(text_data)
       message = data['message']
     username = data['username']
        room = data['room']
        await self.save_message(username, room, message)
 await self.channel_layer.group_send(
self.room_group_name,
            {
                'type': 'chat_message',
                'message': message,
                'username': username,
                'room': room,
            }
        )
    async def chat_message(self, event):
        message = event['message']

        username = event['username']
```

```
        room = event['room']
        await self.send(text_data=json.dumps({
            'message': message,
            'username': username,
            'room': room,
        }))
@sync_to_async
    def save_message(self,username,room,message):
        user=User.objects.get(username=username)
        room=Room.objects.get(slug=room)
Message.objects.create(user=user, room=room, content=message)
```

Models.py

```
from django.contrib.auth.models import User
from django.db import models
# Create your models here.
class Room(models.Model):
    name=models.CharField(max_length=255)
    slug = models.SlugField(unique=True)
class Message(models.Model):
    room = models.ForeignKey(Room, related_name='messages',
on_delete=models.CASCADE)
    user = models.ForeignKey(User, related_name='messages',
on_delete=models.CASCADE)
```

```
    content = models.TextField()
    date_added = models.DateTimeField(auto_now_add = True)
 class Meta:
        ordering =('date_added',)
```

Routing.py

```
from django.urls import path
from . import consumers
websocket_urlpatterns = [
    path('ws/<str:room_name>/', consumers.ChatConsumer.as_asgi()),
]
```

Tests.py

```
from django.test import TestCase
# Create your tests here.
```

Urls.py

```
from django.contrib import admin
from django.urls import path, include
from . import views
urlpatterns = [
            path('',views.rooms,name='rooms'),
            path('<slug:slug>/',views.room,name='room'),
              ]
```

Views.py

```
from django.contrib.auth.decorators import login_required
from django.shortcuts import render
# Create your views here.
from .models import Room , Message
@login_required
def rooms(request):
    rooms = Room.objects.all()
 return render(request , 'room/rooms.html' , {'rooms': rooms} )
@login_required
def room(request,slug):
    room = Room.objects.get(slug=slug)
    messages = Message.objects.filter(room = room)[0:25]
    return render(request, 'room/room.html',{'room':room,
'messages':messages})
```

Templates→room.html

```
{% extends 'core/base.html' %}
{% block title %}{{ room.name }} | {% endblock %}
{% block content %}
<div class="p-10 lg:p-20 text-center">
    <h1 class="text-3xl lg:text-6xl text-white">{{ room.name }}</h1>
</div>
<div class="lg:w-2/4 mx-4 lg:mx-auto p-4 bg-white rounded-xl">
```

```
    <div class="chat-messages space-y-3" id="chat-messages">
        {% for message in messages %}
        <div class="p-4 bg-gray-200 rounded-xl">
          <p class="font-semibold">{{ message.user.username }}</p>
          <p>{{ message.content }}</p>
        </div>
        {% endfor %}


    </div>
</div>
<div class="lg:w-2/4 mt-6 mx-4 lg:mx-auto p-4 bg-white rounded-xl">
    <form method="post" action="." class="flex">
        {% csrf_token %}
        <input type="text" name="content" class="flex-1 mr-3"
placeholder="Your message..." id="chat-message-input">


        <button
            class="px-5 py-3 rounded-xl text-white bg-teal-600
hover:bg-teal-700"
            id="chat-message-submit"
        >Submit</button>
    </form>
</div>
```

```
{% endblock %}


{% block scripts %}
{{ room.slug|json_script:"json-roomname" }}
{{ request.user.username|json_script:"json-username" }}
<script>
    const roomName =
JSON.parse(document.getElementById('json-roomname').textContent);
    const userName =
JSON.parse(document.getElementById('json-username').textContent);
   const chatSocket = new WebSocket(
            'ws://'
            + window.location.host
            + '/ws/'
            + roomName
            + '/'
        );
 chatSocket.onmessage = function(e) {
        console.log('onmessage');
        const data = JSON.parse(e.data);
if(data.message){
let html=' <div class="p-4 bg-gray-200 rounded-xl">';
           html+= '<p class="font-semibold">'+ data.username+'</p>';
```

```
            html+= '<p>' + data.message +'</p> </div>';
            document.querySelector('#chat-messages').innerHTML+=html;
scrollToBottom();
        }
        else {
        alert("this message is empty");
        }

    };
 chatSocket.onclose = function(e) {
        console.log('onclose')
    }
    document.querySelector('#chat-message-submit').onclick =function(e){
    const messageInputDom =
document.querySelector('#chat-message-input');
    const message = messageInputDom.value;
    chatSocket.send(JSON.stringify({
    'message':message,
    'username':userName,
    'room':roomName
    }));
    messageInputDom.value='';
```

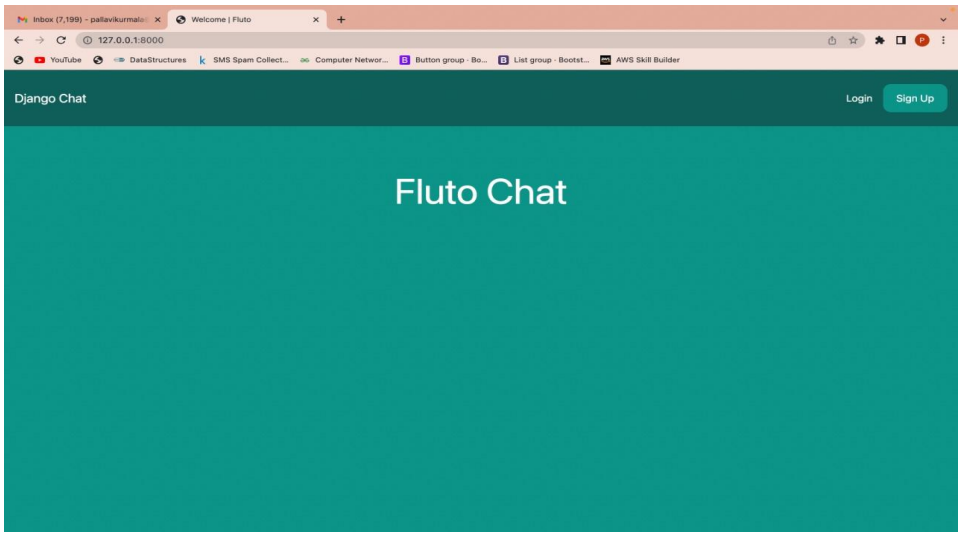
```
    return false;
    }
function scrollToBottom(){
 const objDiv =document.querySelector('#chat-messages');
 objDiv.scrollTop = objDiv.scrollHeight;
}
scrollToBottom();
</script>
{% endblock %}
```

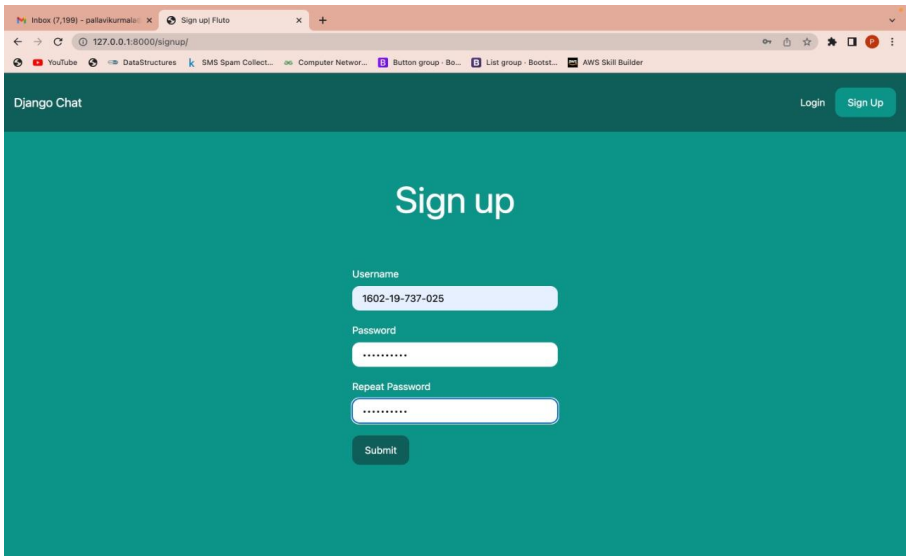
Rooms.html

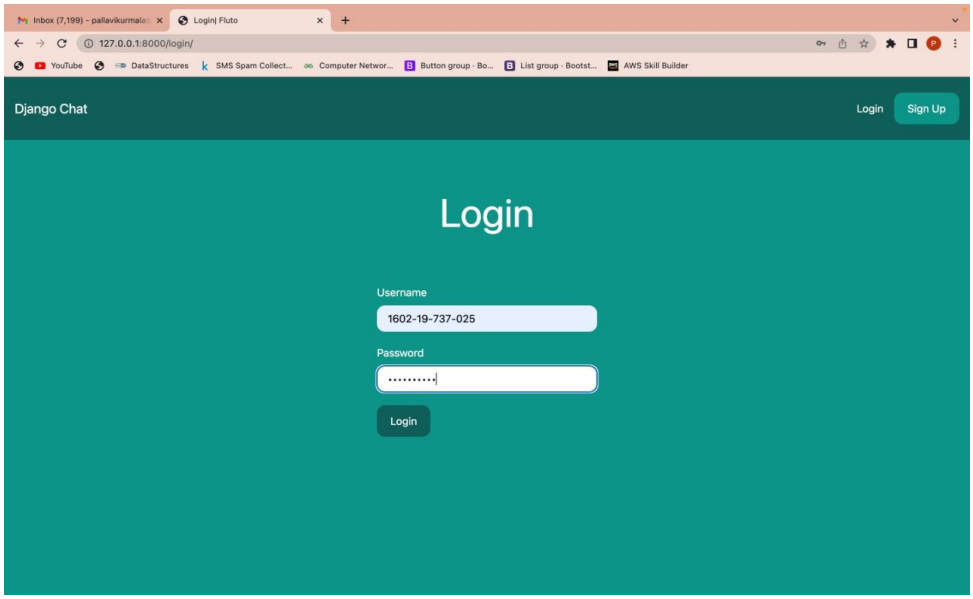
```
{% extends 'core/base.html' %}
{% block title %} Rooms| {%  endblock %}
{% block content %}
<div class="p-10 lg:p-20 text-center">
    <h1 class="text-3xl lg:text-6xl text-white">Rooms</h1>
</div>
<div class="w-full flex flex-wrap items-center">


 {% for room in rooms %}
    <div class="w-full lg:w-1/4 px-3 py-3">
        <div class="p-4 bg-white chadow-xl text-center">
```

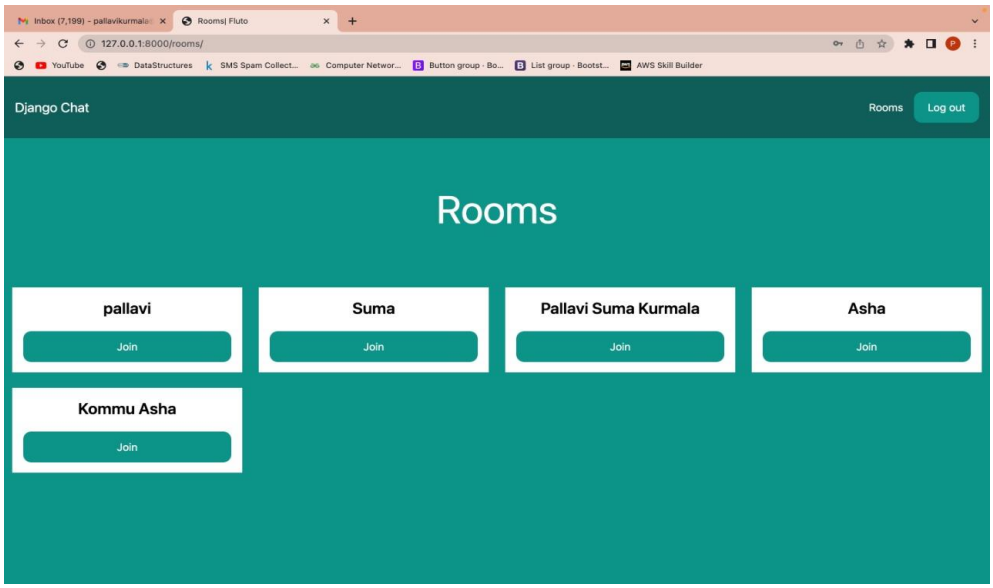
```
            <h2 class ="mb-5 text-2xl font-semibold">{{ room.name }}</h2>

            <a href="{% url 'room' room.slug%}" class="px-5 py-3 block
rounded-xl text-white bg-teal-600 hover:bg-teal-700">Join</a>

        </div></div>

    {% endfor %}

</div>

{% endblock %}
```

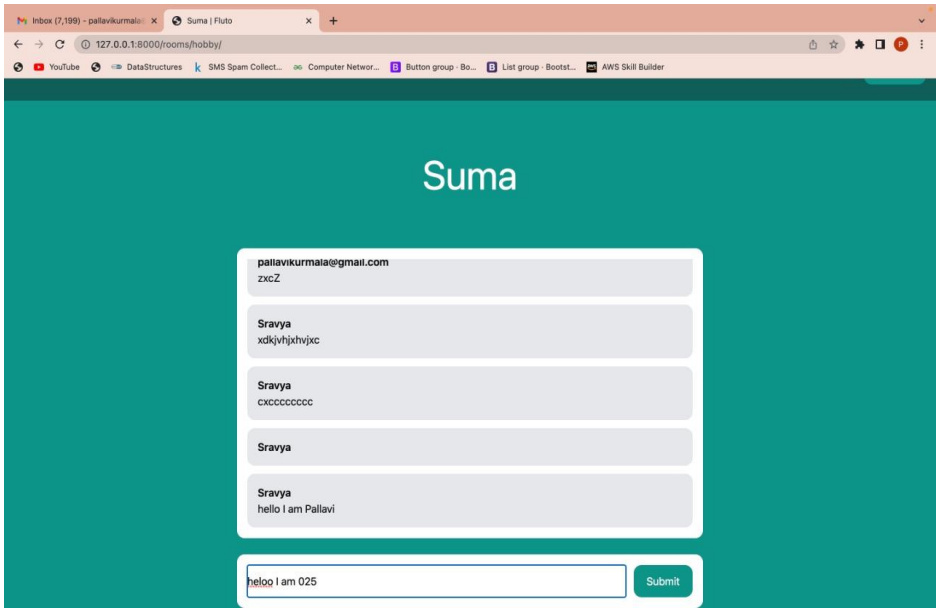
TESTING

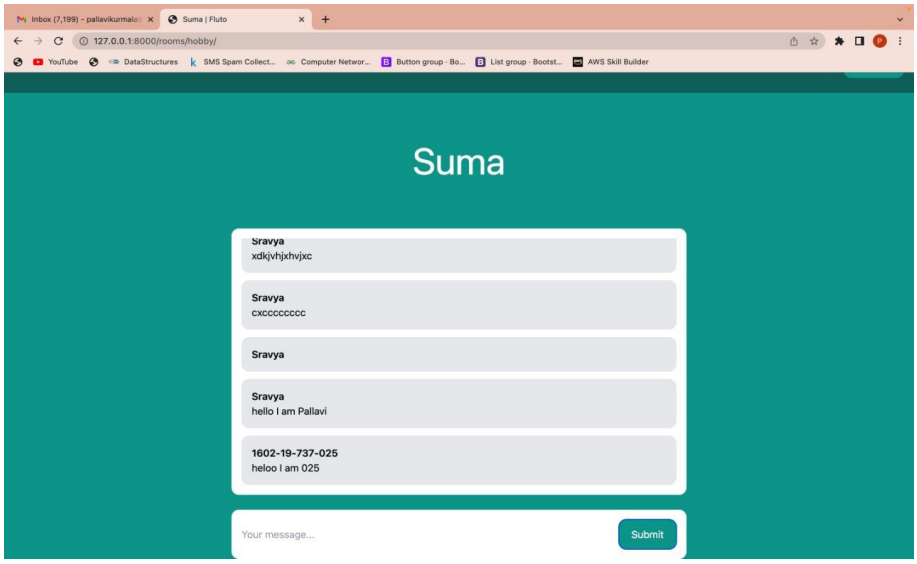

Inbox (7,199) - pallavikurmala
Welcome | Fluto
127.0.0.1:8000
YouTube
DataStructures
SMS Spam Collect...
Computer Networ...
Button group · Bo...
List group · Bootst...
AWS Skill Builder
Django Chat
Login
Sign Up
Fluto Chat

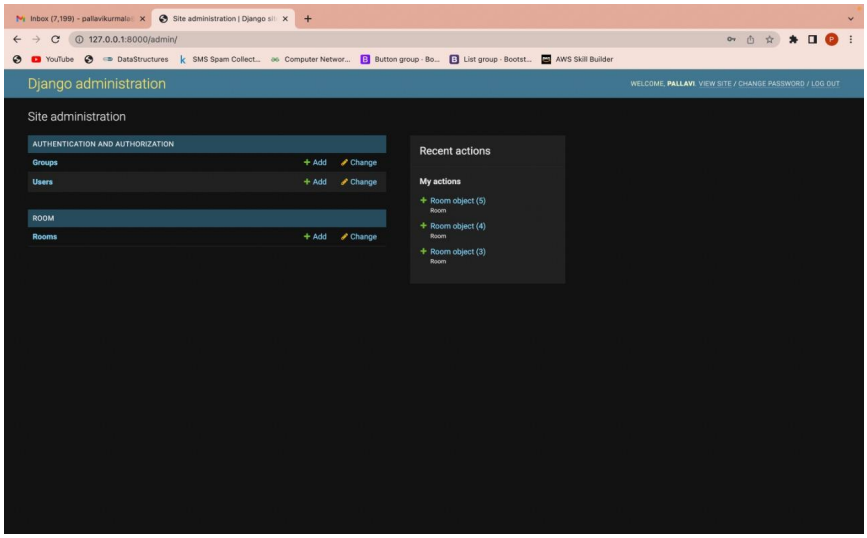

Inbox (7,199) - pallavikurmala
Sign up| Fluto
127.0.0.1:8000/signup/
YouTube
DataStructures
SMS Spam Collect...
Computer Networ...
Button group · Bo...
List group · Bootst...
AWS Skill Builder
Django Chat
Login
Sign Up
Sign up
Username
1602-19-737-025
Password
Repeat Password
Submit

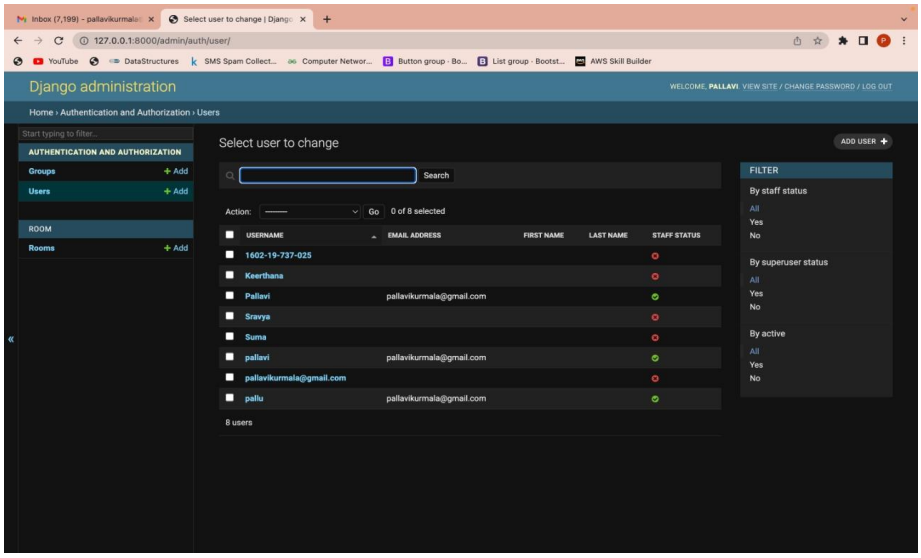

Inbox (7,199) - pallavikurmala
Login| Fluto
127.0.0.1:8000/login/
YouTube
DataStructures
SMS Spam Collect...
Computer Networ...
Button group · Bo...
List group · Bootst...
AWS Skill Builder
Django Chat
Login
Sign Up
Login
Username
1602-19-737-025
Password
Login

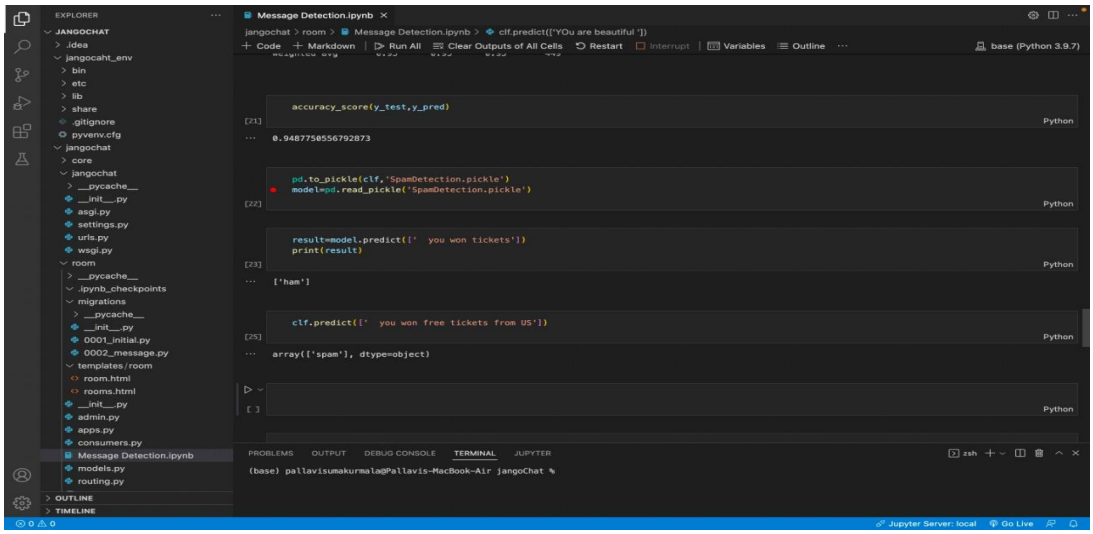

Inbox (7,199) - pallavikurmala
Rooms| Fluto
127.0.0.1:8000/rooms/
YouTube
DataStructures
SMS Spam Collect...
Computer Networ...
Button group · Bo...
List group · Bootst...
AWS Skill Builder
Django Chat
Rooms
Log out
Rooms
pallavi
Join
Suma
Join
Pallavi Suma Kurmala
Join
Asha
Join
Kommu Asha
Join

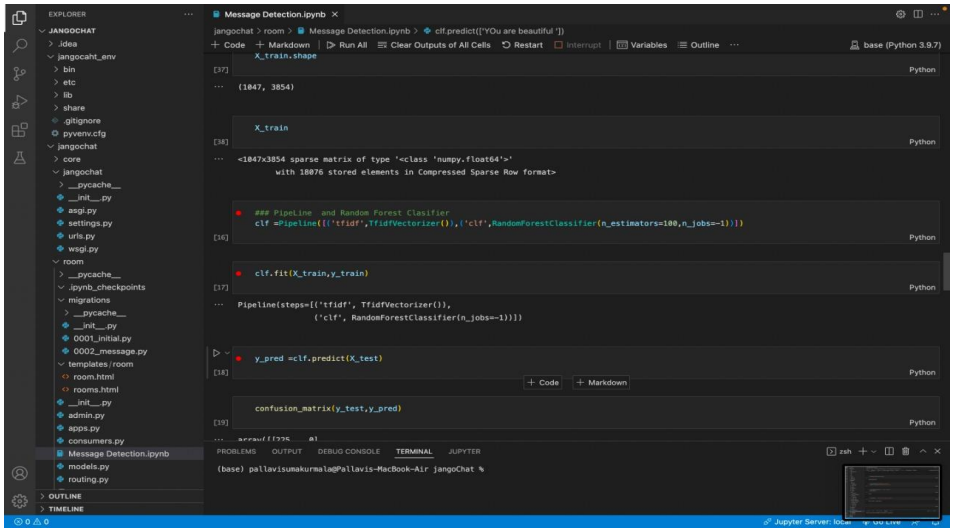

Inbox (7,199) - pallavikurmala
Suma | Fluto
127.0.0.1:8000/rooms/hobby/
YouTube
DataStructures
SMS Spam Collect...
Computer Networ...
Button group · Bo...
List group · Bootst...
AWS Skill Builder
Suma
pallavikurmala@gmail.com
zxcZ
Sravya
xdkjvhjxhvjxc
Sravya
cxccccccccc
Sravya
Sravya
hello I am Pallavi
heloo I am 025
Submit

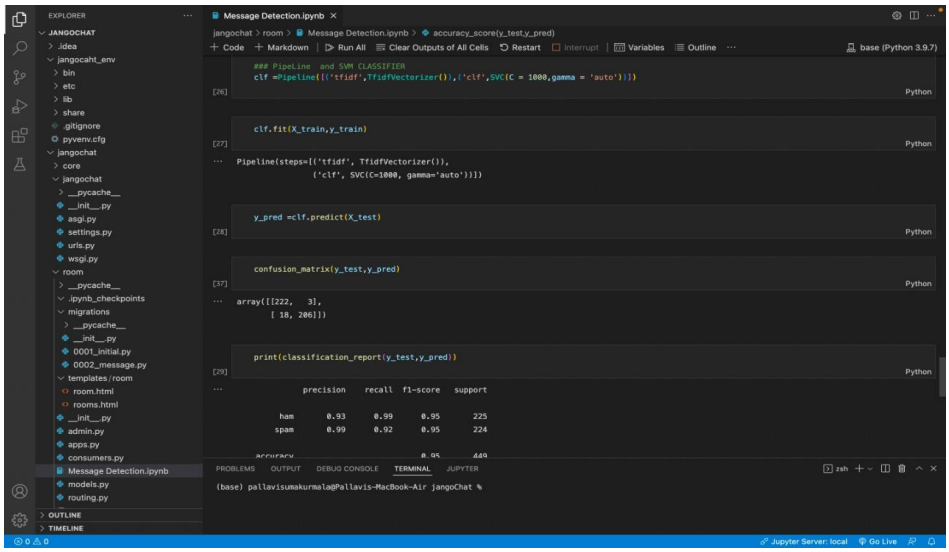

Inbox (7,199) - pallavikurmala
Suma | Fluto
127.0.0.1:8000/rooms/hobby/
YouTube
DataStructures
SMS Spam Collect...
Computer Networ...
Button group · Bo...
List group · Bootst...
AWS Skill Builder
Suma
Sravya
xdkjvhjxhvjxc
Sravya
cxccccccccc
Sravya
Sravya
hello I am Pallavi
1602-19-737-025
heloo I am 025
Your message...
Submit

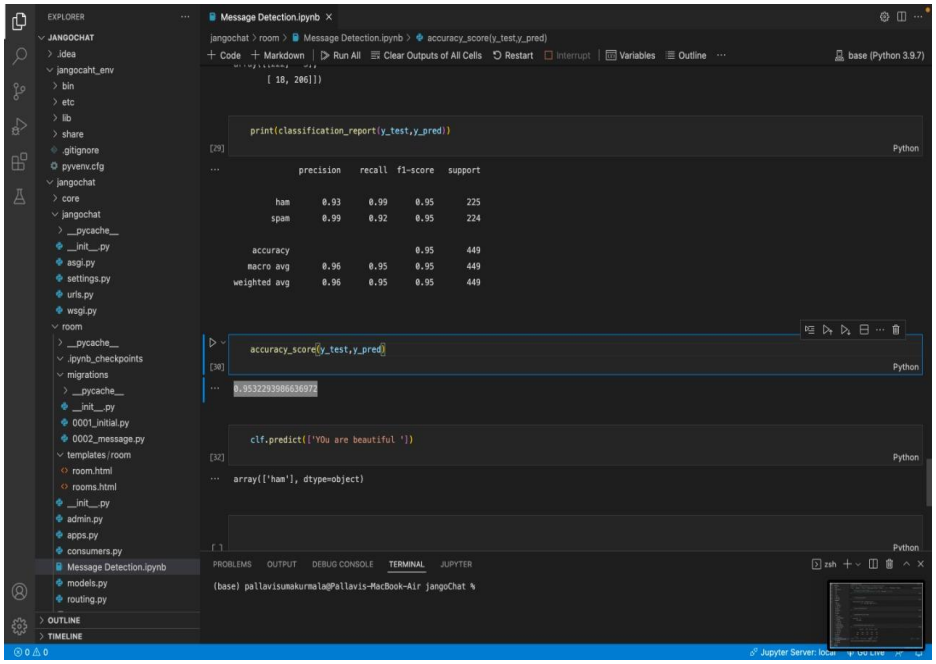

Inbox (7,199) - pallavikurmala
Site administration | Django site
127.0.0.1:8000/admin/
YouTube
DataStructures
SMS Spam Collect...
Computer Networ...
Button group · Bo...
List group · Bootst...
AWS Skill Builder
Django administration
WELCOME, PALLAVI. VIEW SITE / CHANGE PASSWORD / LOG OUT
Site administration
AUTHENTICATION AND AUTHORIZATION
Groups + Add Change
Users + Add Change
ROOM
Rooms + Add Change
Recent actions
My actions
Room object (5)
Room
Room object (4)
Room
Room object (3)
Room


Inbox (7,199) - pallavikurmala
Select user to change | Django site
127.0.0.1:8000/admin/auth/user/
YouTube
DataStructures
SMS Spam Collect...
Computer Networ...
Button group · Bo...
List group · Bootst...
AWS Skill Builder
Django administration
WELCOME, PALLAVI. VIEW SITE / CHANGE PASSWORD / LOG OUT
Home › Authentication and Authorization › Users
Start typing to filter...
AUTHENTICATION AND AUTHORIZATION
Groups + Add
Users + Add
ROOM
Rooms + Add
Select user to change
ADD USER +
Search
Action: Go 0 of 8 selected
USERNAME
EMAIL ADDRESS
FIRST NAME
LAST NAME
STAFF STATUS
1602-19-737-025
Keerthana
Pallavi pallavikurmala@gmail.com
Sravya
Suma
pallavi pallavikurmala@gmail.com
pallavikurmala@gmail.com
pallu pallavikurmala@gmail.com
8 users
FILTER
By staff status
All
Yes
No
By superuser status
All
Yes
No
By active
All
Yes
No


EXPLORER
JANGOCHAT
.idea
jangocaht_env
bin
etc
lib
share
.gitignore
pyvenv.cfg
jangochat
core
jangochat
__pycache__
__init__.py
asgi.py
settings.py
urls.py
wsgi.py
room
__pycache__
.ipynb_checkpoints
migrations
__pycache__
__init__.py
0001_initial.py
0002_message.py
templates/room
room.html
rooms.html
__init__.py
admin.py
apps.py
consumers.py
Message Detection.ipynb
models.py
routing.py
OUTLINE
TIMELINE
Message Detection.ipynb
jangochat > room > Message Detection.ipynb > clf.predict(['YOu are beautiful '])
Code
Markdown
Run All
Clear Outputs of All Cells
Restart
Interrupt
Variables
Outline
base (Python 3.9.7)
accuracy_score(y_test,y_pred)
[21]
Python
0.9487750556792873
pd.to_pickle(clf,'SpamDetection.pickle')
model=pd.read_pickle('SpamDetection.pickle')
[22]
Python
result=model.predict([' you won tickets'])
print(result)
[23]
Python
['ham']
clf.predict([' you won free tickets from US'])
[25]
Python
array(['spam'], dtype=object)
Python
PROBLEMS
OUTPUT
DEBUG CONSOLE
TERMINAL
JUPYTER
zsh
(base) pallavisumakurmala@Pallavis-MacBook-Air jangoChat %
Jupyter Server: local
Go Live

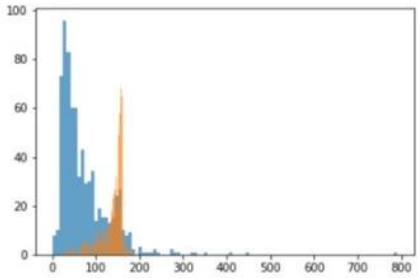

EXPLORER
JANGOCHAT
.idea
jangocaht_env
bin
etc
lib
share
.gitignore
pyvenv.cfg
jangochat
core
jangochat
__pycache__
__init__.py
asgi.py
settings.py
urls.py
wsgi.py
room
__pycache__
.ipynb_checkpoints
migrations
__pycache__
__init__.py
0001_initial.py
0002_message.py
templates/room
room.html
rooms.html
__init__.py
admin.py
apps.py
consumers.py
Message Detection.ipynb
models.py
routing.py
OUTLINE
TIMELINE
Message Detection.ipynb
jangochat > room > Message Detection.ipynb > clf.predict(['YOu are beautiful '])
Code
Markdown
Run All
Clear Outputs of All Cells
Restart
Interrupt
Variables
Outline
base (Python 3.9.7)
X_train.shape
[37]
Python
(1047, 3854)
X_train
[38]
Python
<1047x3854 sparse matrix of type '<class 'numpy.float64'>'
with 18076 stored elements in Compressed Sparse Row format>
PipeLine and Random Forest Clasifier
clf =Pipeline([('tfidf',TfidfVectorizer()),('clf',RandomForestClassifier(n_estimators=100,n_jobs=-1))])
[36]
Python
clf.fit(X_train,y_train)
[17]
Python
Pipeline(steps=[('tfidf', TfidfVectorizer()),
('clf', RandomForestClassifier(n_jobs=-1))])
y_pred =clf.predict(X_test)
[18]
Python
Code
Markdown
confusion_matrix(y_test,y_pred)
[19]
Python
PROBLEMS
OUTPUT
DEBUG CONSOLE
TERMINAL
JUPYTER
zsh
(base) pallavisumakurmala@Pallavis-MacBook-Air jangoChat %
Jupyter Server: local
Go Live


EXPLORER
JANGOCHAT
.idea
jangocaht_env
bin
etc
lib
share
.gitignore
pyvenv.cfg
jangochat
core
jangochat
__pycache__
__init__.py
asgi.py
settings.py
urls.py
wsgi.py
room
__pycache__
.ipynb_checkpoints
migrations
__pycache__
__init__.py
0001_initial.py
0002_message.py
templates/room
room.html
rooms.html
__init__.py
admin.py
apps.py
consumers.py
Message Detection.ipynb
models.py
routing.py
OUTLINE
TIMELINE
Message Detection.ipynb
jangochat > room > Message Detection.ipynb > accuracy_score(y_test,y_pred)
Code
Markdown
Run All
Clear Outputs of All Cells
Restart
Interrupt
Variables
Outline
base (Python 3.9.7)
PipeLine and SVM CLASSIFIER
clf =Pipeline([('tfidf',TfidfVectorizer()),('clf',SVC(C = 1000,gamma = 'auto'))])
[26]
Python
clf.fit(X_train,y_train)
[27]
Python
Pipeline(steps=[('tfidf', TfidfVectorizer()),
('clf', SVC(C=1000, gamma='auto'))])
y_pred =clf.predict(X_test)
[28]
Python
confusion_matrix(y_test,y_pred)
[37]
Python
array([[222, 3],
[18, 206]])
print(classification_report(y_test,y_pred))
[29]
Python
precision recall f1-score support
ham 0.93 0.99 0.95 225
spam 0.99 0.92 0.95 224
accuracy 0.95 449
PROBLEMS
OUTPUT
DEBUG CONSOLE
TERMINAL
JUPYTER
zsh
(base) pallavisumakurmala@Pallavis-MacBook-Air jangoChat %
Jupyter Server: local
Go Live



Accuracy

In [14]:
```
###Exploratory Data Analsys
plt.hist(data[data['label']=='ham']['length'],bins =100,alpha=0.7)
plt.hist(data[data['label']=='spam']['length'],bins =100,alpha=0.7)
plt.show()
```



CONCLUSION

This project has helped us very much to know more about machine learning algorithms,how they work and when to use which algorithm. It helped us to implement the knowledge of machine learning in a live project and it was fascinating to watch so many machine learning algorithms solve a single problem in different ways and with different accuracies. And also made us curious to implement more such algorithms and solve more such problems.

FUTURE SCOPE

Many features such as voice mailing,calling,video calling are being looked into to add in the project in the future. And also web search a word by long pressing on it is one of the unique ideas we are considering to implement, protecting the app by a password are few of the ideas.

REFERENCES

- https://iopscience.iop.org/article/10.1088/1742-6596/1797/1/012017
- https://www.hindawi.com/journals/scn/2020/8873639/
- http://cs229.stanford.edu/proj2013/ShiraniMehr-SMSSpamDetectionUsingMachineLearningApproach.pdf
- http://navajyotijournal.org/Aug_2017_issue/Aug2017_7.pdf
- https://www.globaltechcouncil.org/machine-learning/how-does-machine-learning-works-in-mobile-messaging/