Message Extraction Documentation

Running the code:

1. In order to run the code, extract the zip file which has been submitted. Go inside of the extracted folder and open the terminal.
2. (Using any virtual environment) Create a new virtual environment.
3. Install requirements using command

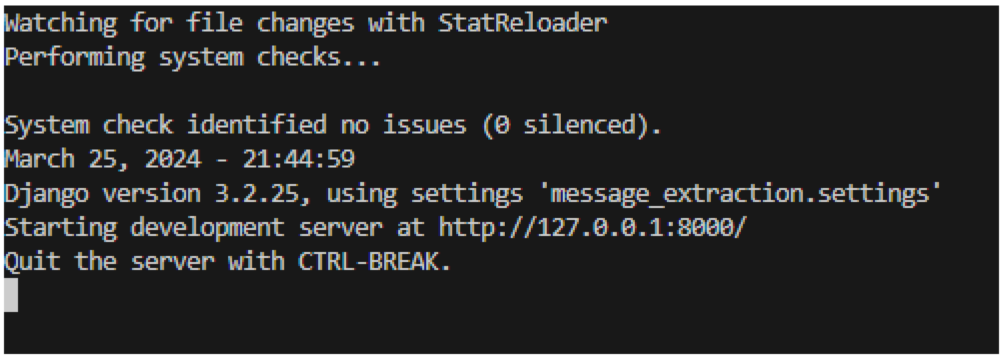
```
pip install -r requirements.txt
```

(Note: If you're using GPU then use paddlepaddle-gpu otherwise use paddlepaddle)

4. After requirements are installed, run the django server using command:

```
python manage.py runserver 8000
```

This will run a django server at localhost:8000 (you can change to other port)



```
Watching for file changes with StatReloader
Performing system checks...

System check identified no issues (0 silenced).
March 25, 2024 - 21:44:59
Django version 3.2.25, using settings 'message_extraction.settings'
Starting development server at http://127.0.0.1:8000/
Quit the server with CTRL-BREAK.
```

If the above is displayed in the terminal, then our django server is up and running.

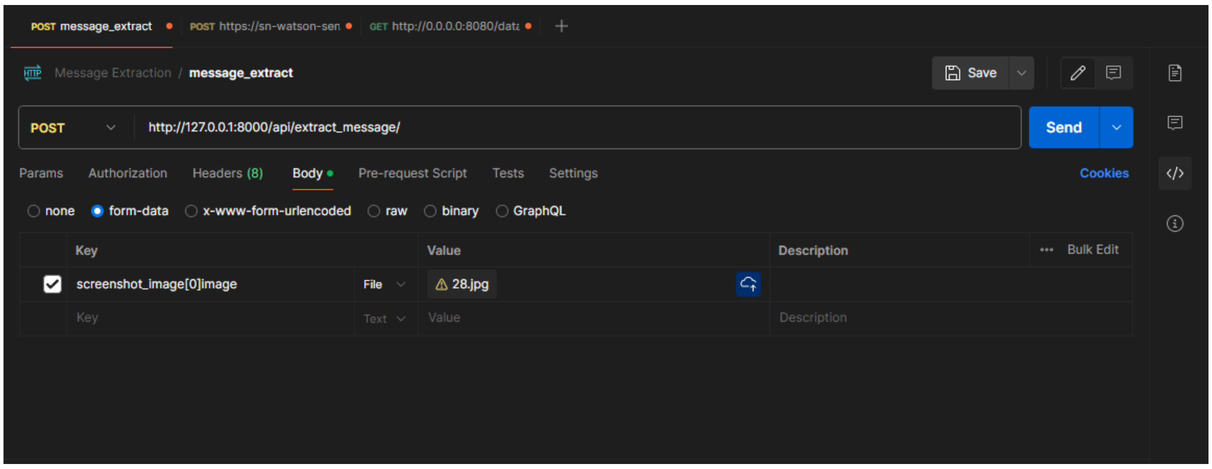
API Request:

1. After the django server is running, we are ready to hit POST requests to the server.
2. Use the following curl command in order to hit a request to the server.

```
curl --location 'http://127.0.0.1:8000/api/extract_message/' \
--form 'screenshot_image[0]image=@"<path_to_screenshot_image>"
```

Note: Provide path to the image in above.

3. To use POSTMAN for API request, past the above curl command in the postman. In body / form-data you can see a key value pair.


POST message_extract | POST https://sn-watson-sen | GET http://0.0.0.0:8080/data

Message Extraction / message_extract

POST http://127.0.0.1:8000/api/extract_message/

Params | Authorization | Headers (8) | Body | Pre-request Script | Tests | Settings

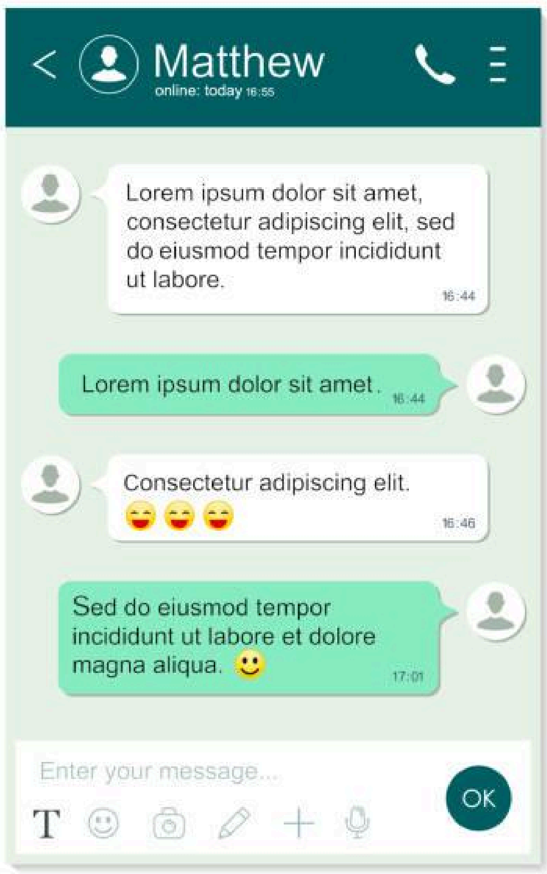
none [✓ form-data] x-www-form-urlencoded raw binary GraphQL

	Key		Value	Description
✓	screenshot_image[0]image	File	28.jpg	
	Key	Text	Value	Description

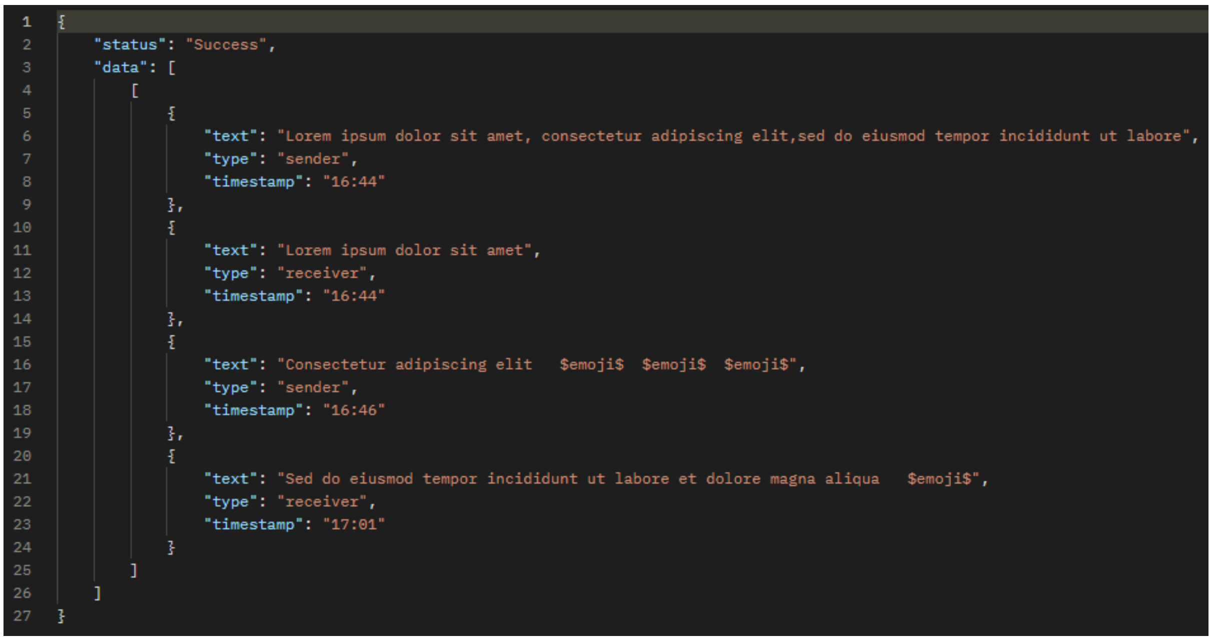
The key should be the same as above. The value is the screenshot image file.

API Response Format

1. If the file uploaded is a valid screenshot image. For example:


Matthew
online: today 16:55
Lorem ipsum dolor sit amet, consectetur adipiscing elit, sed do eiusmod tempor incididunt ut labore. 16:44
Lorem ipsum dolor sit amet. 16:44
Consectetur adipiscing elit. 16:46
Sed do eiusmod tempor incididunt ut labore et dolore magna aliqua. 17:01
Enter your message...
OK

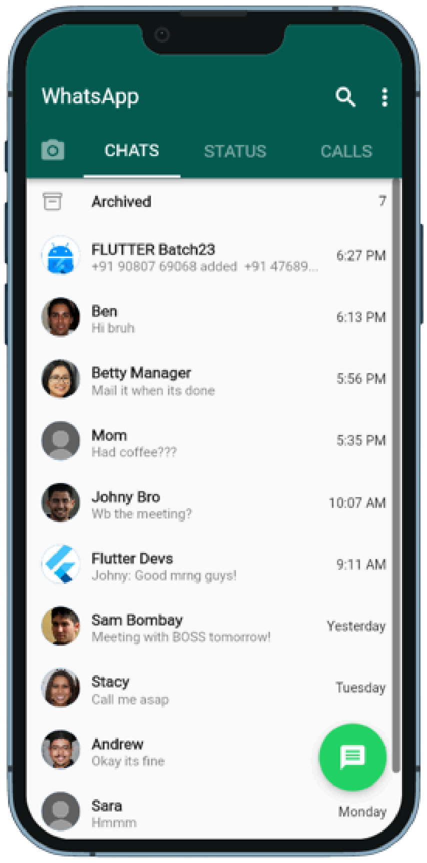
Response will be :



```
{
    "status": "Success",
    "data": [
        [
            {
                "text": "Lorem ipsum dolor sit amet, consectetur adipiscing elit,sed do eiusmod tempor incididunt ut labore",
                "type": "sender",
                "timestamp": "16:44"
            },
            {
                "text": "Lorem ipsum dolor sit amet",
                "type": "receiver",
                "timestamp": "16:44"
            },
            {
                "text": "Consectetur adipiscing elit   $emoji$  $emoji$  $emoji$",
                "type": "sender",
                "timestamp": "16:46"
            },
            {
                "text": "Sed do eiusmod tempor incididunt ut labore et dolore magna aliqua   $emoji$",
                "type": "receiver",
                "timestamp": "17:01"
            }
        ]
    ]
}
```

If the emoji is present then it will show **`$$emoji$$`** for each emoji present in the text.

2. If the image is not a chat screenshot. For example:



WhatsApp

CHATS STATUS CALLS

Archived 7

FLUTTER Batch23 6:27 PM
+91 90807 69068 added +91 47689...

Ben 6:13 PM
Hi bruh

Betty Manager 5:56 PM
Mail it when its done

Mom 5:35 PM
Had coffee???

Johny Bro 10:07 AM
Wb the meeting?

Flutter Devs 9:11 AM
Johny: Good mrng guys!

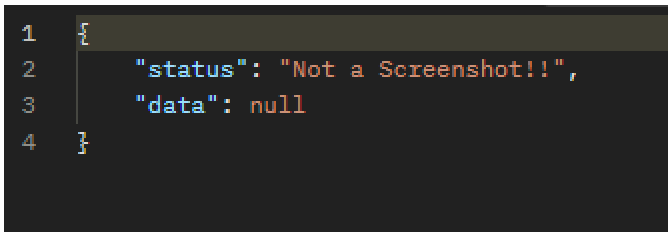
Sam Bombay Yesterday
Meeting with BOSS tomorrow!

Stacy Tuesday
Call me asap

Andrew
Okay its fine

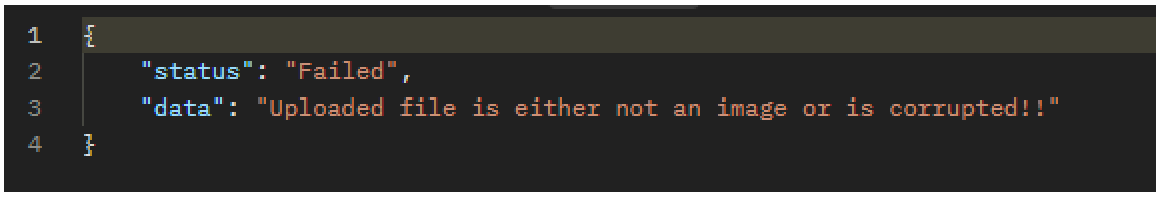
Sara Monday
Hmmm

For this kind of image the API response will be:



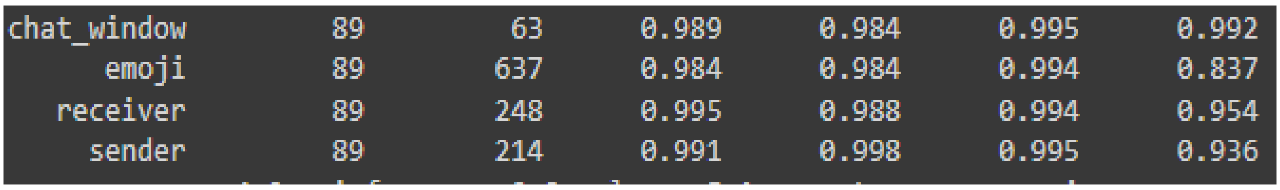
```
{
    "status": "Not a Screenshot!!",
    "data": null
}
```

3. If corrupted image is uploaded or file that is not an image is uploaded then the API response will be:



```
{
    "status": "Failed",
    "data": "Uploaded file is either not an image or is corrupted!!"
}
```

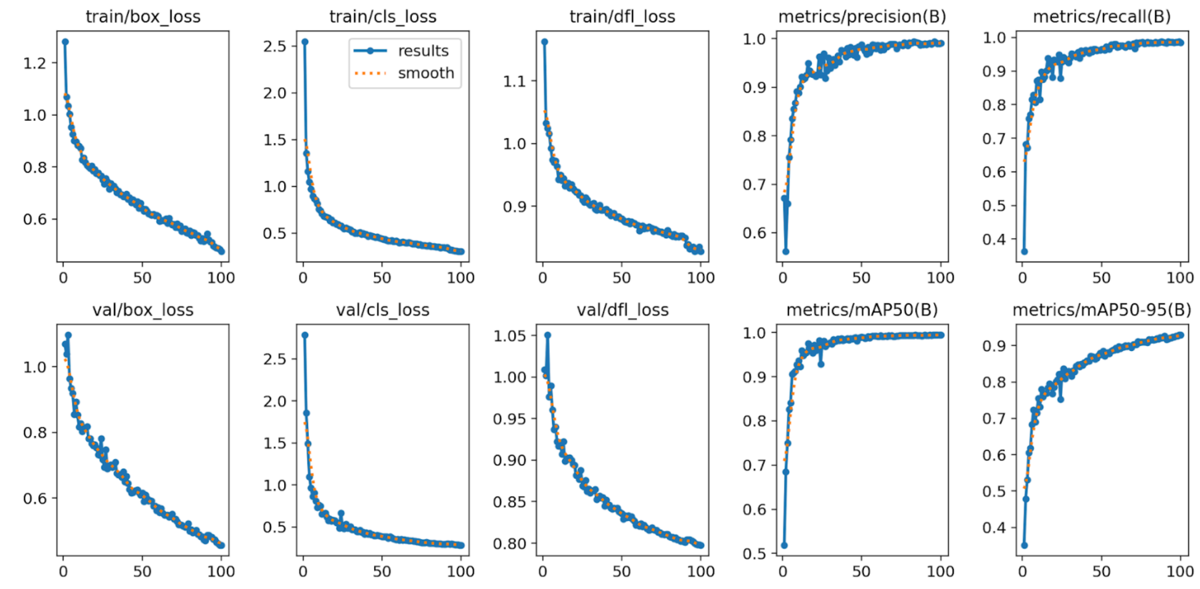
Model Performance Metrics

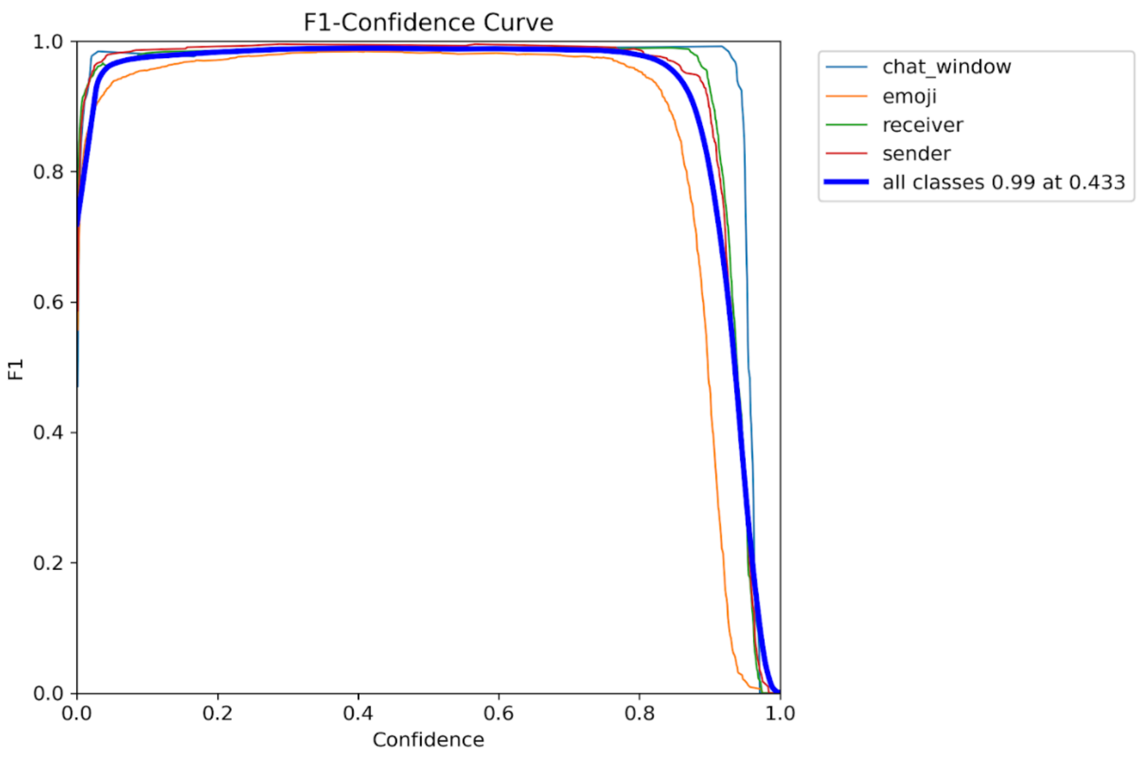


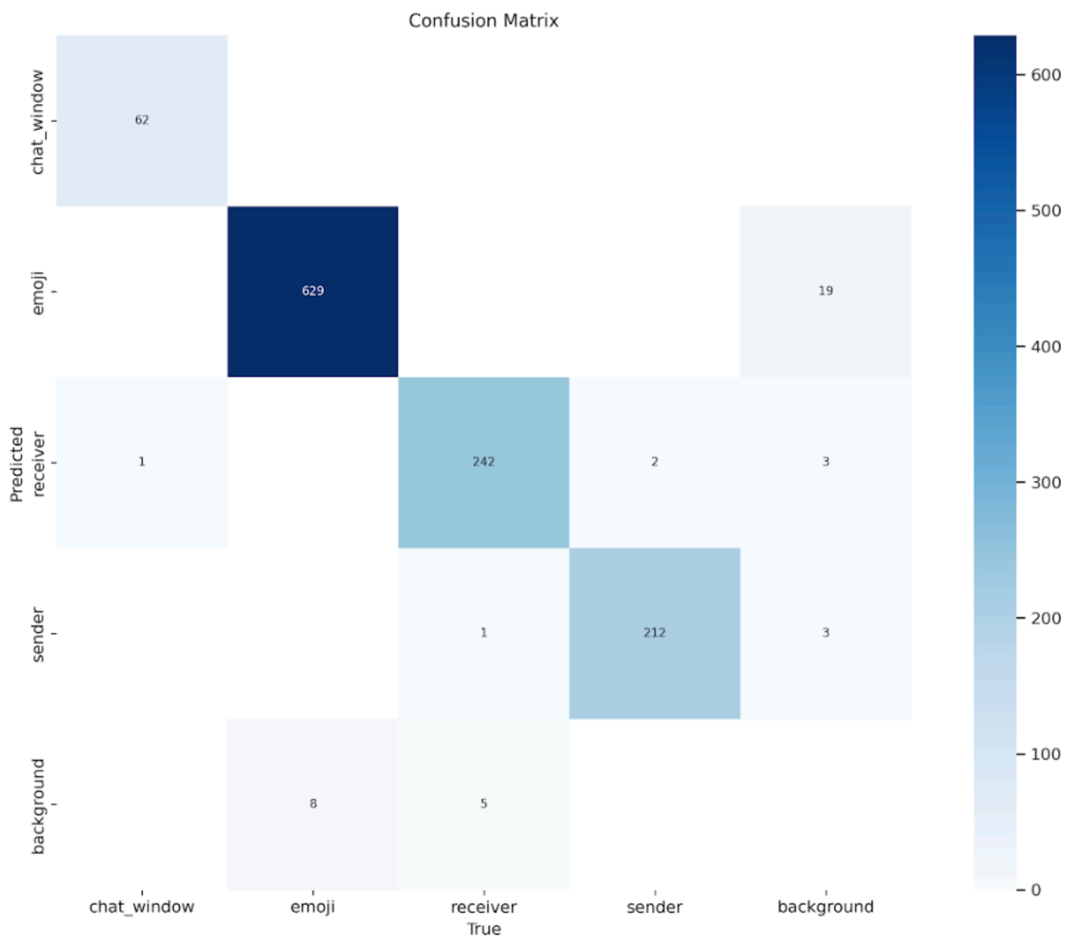
chat_window	89	63	0.989	0.984	0.995	0.992
emoji	89	637	0.984	0.984	0.994	0.837
receiver	89	248	0.995	0.988	0.994	0.954
sender	89	214	0.991	0.998	0.995	0.936

Here, receiver and sender represents the extraction of text and timestamp along with classifying it to receiver or sender.
Chat_window represents the screenshot part where the chat is present
Emoji represents all the emoji present in the chat screenshots.

Training Results:






Confusion Matrix
Predicted
chat_window
emoji
receiver
sender
background
True
chat_window
emoji
receiver
sender
background
62
629
19
1
242
2
3
1
212
3
8
5
600
500
400
300
200
100
0