BAB V
HASIL DAN PEMBAHASAN

5.1. Hasil

Implementasi filter *bad word* menggunakan algoritma Naïve Bayes pada *chatbot* discord berhasil dirancang, model dibuat dengan penggunaan data sebanyak 4 set data yang memiliki 2 label data. Data dibagi sebagai data latih sebesar 80% dan data test 20%, dengan proporsi 55,9% data ungkapan negatif (*bad words*) dan 44,1% data ungkapan positif. Model dapat melakukan filter terhadap *bad word* yang dikirimkan *user* pada aplikasi discord dan memberikan warning kepada user yang mengirimkan *bad word* tersebut. Selain itu, *chatbot* memiliki beberapa fitur untuk memudahkan penggunanya dalam melihat maupun mengelola data member yang sering mengirimkan *bad word*.

5.1.1. Spesifikasi Perangkat Keras

Dalam mengembangkan aplikasi perangkat keras yang digunakan untuk menjalankan program dengan detail spesifikasi yang tercantum pada Tabel 5.1.

Tabel 5. 1 Spesifikasi Perangkat Keras

Perangkat	Konfigurasi
Prosesor	3,1 GHz Dual-Core Intel Pentium Gold
Memori	16 GB 1600 MHz DDR3
Sistem Operasi	Windows 11
Tipe Sistem	64-bit

5.1.2. Spesifikasi Perangkat Lunak

Dalam pengembangan aplikasi dibutuhkan juga *tools* serta terknologi untuk menjalankan aplikasi seperti *Integrated Developed Environment* (IDE) untuk menulis code, Bahasa pemrograman, dan lain-lainnya. Spesifikasi perangkat lunak dapat dilihat pada Tabel 5.2.

Tabel 5. 2 Spesifikasi Perangkat Lunak

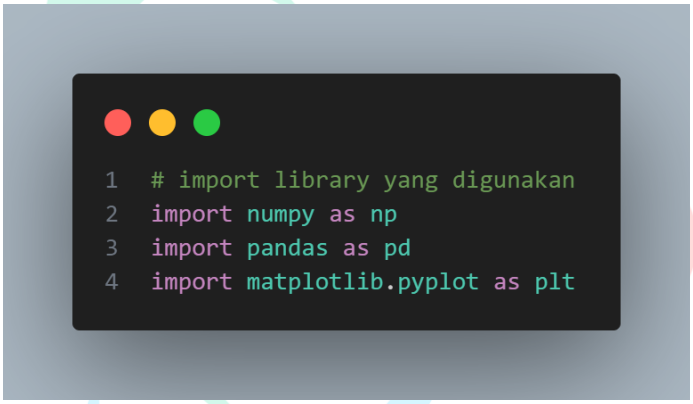
Perangkat	Konfigurasi
Bahasa Pemrograman	Python
Browser	Google Chrome
Library	Numpy, pandas, matplotlib
Tools	PyCharm, Jupyter

5.2. Pembahasan

Pada sub bab ini akan dibahas mengenai detail hasil perancangan yang telah dibuat.

5.2.1. *Pre-Processing* Data dan Pembuatan Model

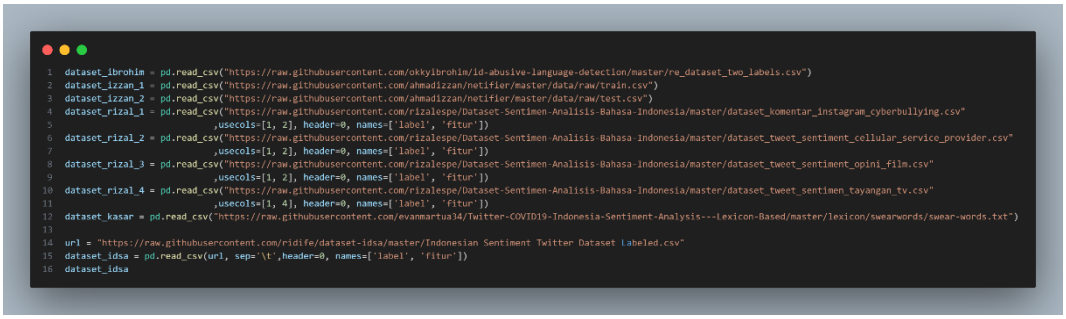
I. Import Library yang dibutuhkan



Gambar 5. 1 Penggalan Code Import Library

Melakukan Import pada library yang dibutuhkan yaitu numpy numerical python yang digunakan untuk melakukan perhitungan *scientific*, matplotlib untuk melakukan plotting 2 dimensi dari data ataupun model yang dibuat, dan pandas adalah Python Data Analysis Library digunakan untuk melakukan analisis data secara cepat.

II. Pengumpulan Dataset



Gambar 5. 2 Penggalan Code Pengumpulan Dataset

Dataset yang digunakan berasal dari 9 sumber data yang kredibel dan telah digunakan sebagai dataset untuk melakukan sentimen analisis. Data – data tersebut kemudian diolah agar hanya memiliki 2 *field*, yaitu *feature* dan label, dengan feature adalah kalimat atau ungkapan dan label adalah penanda bahwa feature tersebut merupakan negatif atau positif ditandai dengan 0 untuk positif dan 1 untuk negatif.

III. Penggabungan Dataset dan *Split test train data*

Dataset yang telah diolah kemudian digabungkan untuk memperoleh dataset secara utuh yang nantinya akan digunakan sebagai data latih dan data test dalam pembuatan model. Dataset dibagi menjadi 4 bagian sesuai sumbernya, dan disamakan antara fieldnya.

```
df1 = dataset_ibrohim[['Tweet']]
df1.columns = ['feature']
df1['label'] = dataset_ibrohim['Label'] - 1


df2 = dataset_izzan[['original_text', 'label']]
df2.columns = ['feature', 'label']
df2

df3 = dataset_rizal
df3.columns = ['feature', 'label']
df3['label'] = np.where(df3['label'] == "negative", 1, 0)
df3

dataset_idsa = pd.DataFrame({'feature': dataset_idsa['fitur'],'label': dataset_idsa['label']})
df4 = dataset_idsa

df = df1._append(df2, ignore_index=True)._append(df4, ignore_index=True)
```

Gambar 5. 3 Penggalan Code Penggabungan Dataset

Kemudian dataset digabungkan untuk dijadikan data latih dan data test dalam pembuatan model. Dataset yang digabung adalah dataset df1 sumber ibrohim, dataset df2 sumber izzan dan dataset df 4 sumber idsa. Dataset ini nantinya akan dipecah lagi menjadi dataset untuk data latih model, dan dataset untuk data testing model.

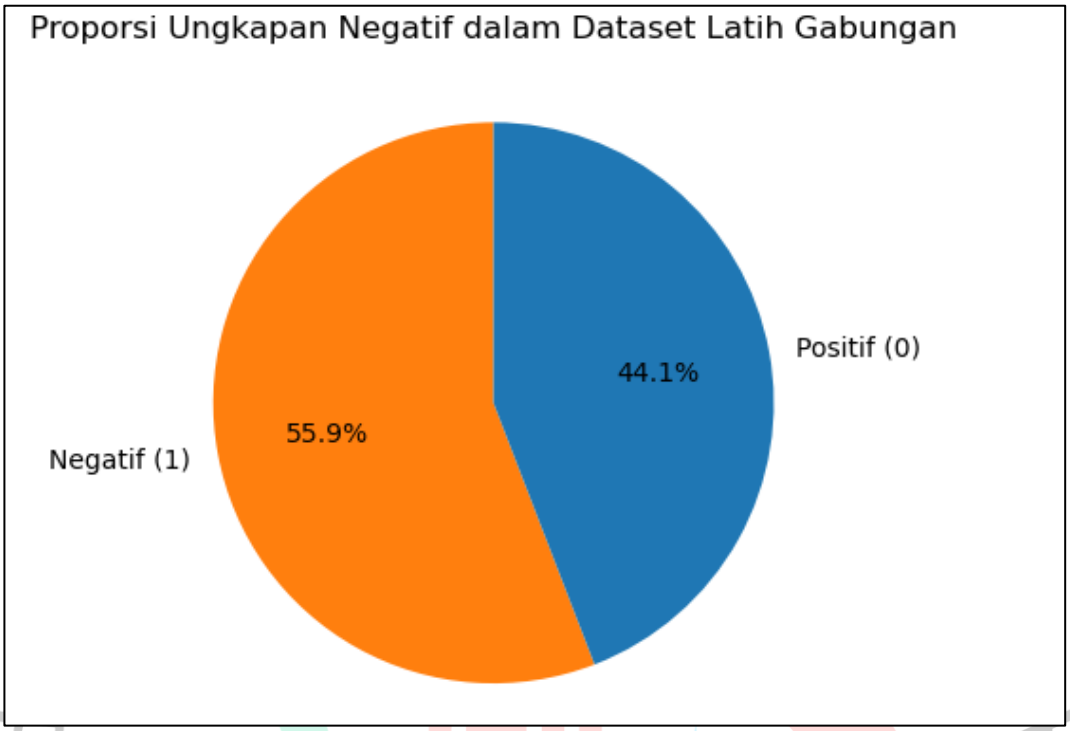
```
# # Train Test Split

from sklearn.model_selection import train_test_split
X_train, X_test, y_train, y_test = train_test_split(df['feature'], df['label'], stratify=df['label'],
                                                    test_size=.20, random_state=42)

print(X_train.shape)
print(y_train.shape)
print(X_test.shape)
print(y_test.shape)


X_train = X_train._append(df3['feature'], ignore_index=True)._append(dataset_kasar['feature'], ignore_index=True)
y_train = y_train._append(df3['label'], ignore_index=True)._append(dataset_kasar['label'], ignore_index=True)

print(X_train.shape)
print(y_train.shape)

print(X_train.tail())
print(y_train.tail())

```

Gambar 5. 4 Penggalan Code Train Test Data Split

Mengimport train_test_split dari sklearn.model_selection untuk membagi dataset menjadi data latih dan data test dengan 4 variabel yaitu X_train untuk feature data train, X_test untuk feature data test, y_train untuk label data train, dan y_test untuk label data test dengan proporsi pembagian sebesar 20% data test dan 80% data latih dari dataset df. Kemudian data train ditambah dengan dataset df3 yaitu dataset rizal dan ditambah dengan dataset kata kasar, untuk membuat model semakin baik.

```
ax = (y_train.value_counts()
      .plot(kind='pie', label="", autopct='%.1f', labels=["Negatif (1)", "Positif (0)"],
            startangle=90, title="Proporsi Ungkapan Negatif dalam Dataset Latih Gabungan",
            colors=['C1', 'C0']))

for t in ax.texts:
    try:
        float(t.get_text())
        t.set_text(t.get_text() + "%")
    except ValueError:
        pass
```

Gambar 5. 5 Penggalan Code Plot Data



Gambar 5. 6 Proporsi Data

Pada gambar 5.6 terlihat proporsi untuk ungkapan positif dan ungkapan negatif pada dataset latih yang telah digabung menunjukkan 55,9% untuk ungkapan negatif ditandai dengan label 1 dan 44,1% untuk ungkapan positif ditandai dengan label 0.

IV. Pre-processing Data

```python
def cleanse(data):
    # lowercase
    data = data.lower()

    # remove newline
    data = data.replace(r'\n', ' ')

    # remove links
    data = re.sub(r'http\S+', '', data)
    data = re.sub(r'bit.ly\S+', '', data)
    data = re.sub(r't.co\S+', '', data)
    data = re.sub(r's.id\S+', '', data)

    # remove mentions
    data = data.replace("_", "")
    data = re.sub("(@[A-Za-z0-9]+)", " ", data)

    # remove punctutations
    punct = string.punctuation.replace("<", "").replace(">", "").replace("[", "").replace("]", "")
    translator = str.maketrans(punct, ' '*len(punct))
    data = data.translate(translator)

    # remove characters between [], <>, or {}
    data = re.sub("[\[].*?[\]]", " ", data)
    data = re.sub(r'<[^>]+>', ' ', data)
    data = re.sub(r'{[^>]+}', ' ', data)

    # remove numeric digits
    data = re.sub(r'[0-9]', '', data)

    # remove emoji
    emoji_pattern = re.compile("["
                               u"\U0001F600-\U0001F64F"  # emoticons
                               u"\U0001F300-\U0001F5FF"  # simbol & piktogram
                               u"\U0001F680-\U0001F6FF"  # transportasi & simbol map
                               u"\U0001F700-\U0001F77F"  # alat teknis
                               u"\U0001F780-\U0001F7FF"  # panah
                               u"\U0001F800-\U0001F8FF"  # kategori ekstra
                               u"\U0001F900-\U0001F9FF"  # emoji wajah
                               u"\U0001FA00-\U0001FA6F"  # emoji
                               u"\U0001FA70-\U0001FAFF"  # emoji kategori tambahan
                               u"\U00002702-\U000027B0"  # dingbat
                               u"\U000024C2-\U0001F251"
                               "]+", flags=re.UNICODE)
    data = emoji_pattern.sub(r'', data)

    # strip excess characters on both ends (aaah -> ah, siapp -> siap)
    data_list = [re.sub(r'([a-z])\1+$', r'\1', token) for token in data.split()]
    data_list = [re.sub(r'^([a-z])\1+', r'\1', token) for token in data_list]
    #data_list = [re.sub(r'^([a-z])\1+&&([a-z])\1+$', r'\1', token) for token in data.split()]
    data = " ".join(data_list)

    # remove extra space
    data = ' '.join(data.split())

    return data

X_train_clean = X_train.apply(lambda x: cleanse(x))
X_test_clean  = X_test.apply(lambda x: cleanse(x))
```

Gambar 5. 7 Penggalan Code Pre-Processing Data

Data yang sudah dibagi kemudian dilakukan *pre-processing* agar data latih menjadi data yang bersih dan dapat dipahami dengan mudah oleh model. Gambar 5.7 merupakan penggalan *code* fungsi *cleansing* data. Tahap pertama

pre-processing data adalah melakukan *cleansing* pada data, proses – prosesnya adalah:

a. Membuat semua data menjadi lower
b. Mengganti baris baru dengan spasi
c. Menghapus links
d. Menghapus *mentions*
e. Menghapus tanda baca
f. Menghapus karakter yang terdapat di dalam kurung [], <>, atau {}
g. Menghapus angka
h. Menghapus emoji
i. Menghapus abjad yang berulang dalam kata
j. Menghapus spasi yang lebih dari satu

Gambar 5.8 merupakan hasil dari proses cleansing data pada data latih dan contoh pada kalimat baru.

```
cleanse("@asaoao_qeq coba dddongg #pakeotak https://s.id/qwe 🤗 hehehttp {<bomb>} <asd> <bomb> [DUPLICATE] uwwuuu kerennnnn")
'coba dong pakeotak hehehttp uwwu keren'
```

Gambar 5. 8 Hasil Cleansing Data

Tahap kedua dalam pre-processing data adalah melakukan normalisasi data, yaitu proses mengganti kata gaul atau kata tidak baku pada data menjadi kata baku. Data kata gaul dan formal diambil dari 3 sumber lalu digabungkan untuk dilakukan normalisasi data pada data latih.

```
# ## Normalisasi with Tokenization

slang_1 = pd.read_csv("https://raw.githubusercontent.com/ramaprakoso/analisis-sentimen/master/kamus/kbba.txt",
                      sep='\t', header=None, names=["slang", "formal"])

slang_2 = pd.read_csv("https://raw.githubusercontent.com/nasalsabila/kamus-alay/master/colloquial-indonesian-lexicon.csv",
                      usecols=[0, 1])

slang_3 = pd.read_csv("https://raw.githubusercontent.com/pzidaneh/netifier/master/data/external/slangword.csv",
                       header=0, names=["slang", "formal"])

kamus_slang = slang_1._append(slang_2, ignore_index=True)._append(slang_3, ignore_index=True).drop_duplicates()
```

Gambar 5. 9 Penggalan Code Data Normalisasi

```
import nltk
nltk.download('punkt')
from nltk import word_tokenize

def normalize_text(data):
    word_tokens = word_tokenize(data)
    result = [dict_slang.get(wt, wt) for wt in word_tokens]  # if dict_slang(wt) doesn't exist, output is wt itself
    return ' '.join(result)

X_train_norm = X_train_clean.apply(lambda x: normalize_text(x))
X_test_norm = X_test_clean.apply(lambda x: normalize_text(x))

print(pd.concat([X_train_clean.head(5), X_train_norm.head(5)], axis=1))
print(pd.concat([X_test_clean.tail(5), X_test_norm.tail(5)], axis=1))
```

Gambar 5. 10 Penggalan Code Proses Normalisasi

Untuk melakukan proses normalisasi data perlu mengimport nltk dan word_tokenize, untuk melakukan *replace* kata kata data latih yang terdeteksi sama dengan dataset kamus_slang untuk diganti menjadi formal. Data latih kemudian diupdate dengan data yang sudah dilakukan normalisasi.

Tahap ketiga dalam pre-processing data adalah melakukan *stopword removal*, yaitu menghapus kata kata yang terindikasi *stopword*. Proses menghilangkan stopword terdapat pada fungsi remove_stopword pada gambar 5.11 jika terdapat stopwords pada kalimat atau ungkapan maka akan dihapus atau diganti dengan spasi.

```
# ## Stopword Removal

from nltk import sent_tokenize
from nltk.corpus import stopwords
import requests


stopwords_yasir = pd.read_csv("https://raw.githubusercontent.com/yasirabd/sentiment-dkk-semarang/main/data/stop_words.txt",
                              header=None)


def remove_stopword(text, stopwords_yasir=stopwords_yasir):
    word_tokens = word_tokenize(text)
    filtered_sentence = [w for w in word_tokens if w not in stopwords_yasir]
    return ' '.join(filtered_sentence)

X_train_nosw = X_train_norm.apply(lambda x: remove_stopword(x))
X_test_nosw = X_test_norm.apply(lambda x: remove_stopword(x))
```

Gambar 5. 11 Penggalan Code Stopword Removal

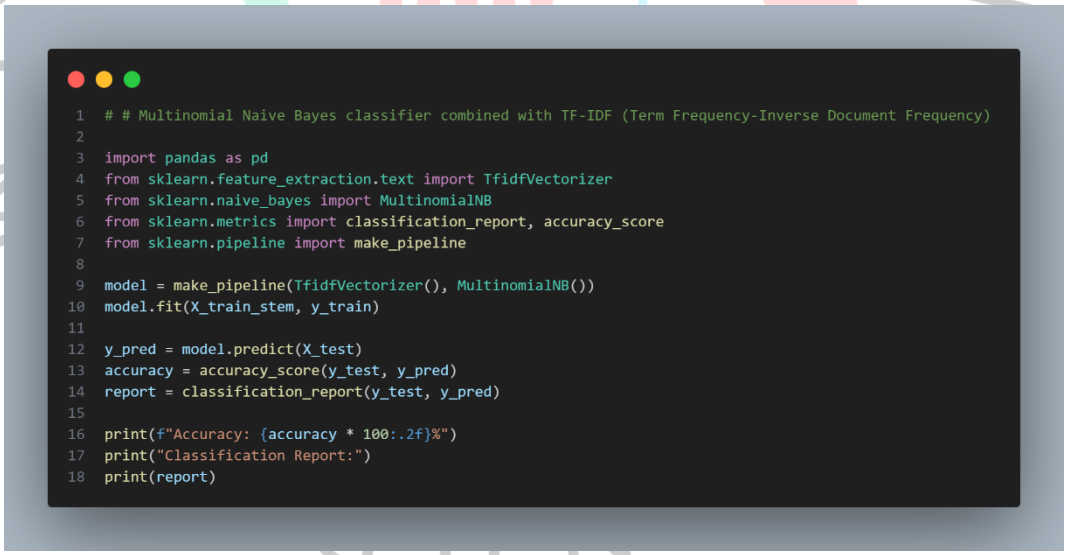
Tahap keempat dalam pre-processing data adalah stemming, yaitu proses menghilangkan imbuhan dalam awal dan akhir kata, sehingga hanya ada kata

dasar. Pertama diperlukan code untuk menginstall library mpstemmer untuk melakukan proses *stemming* pada dataset. Setelah library diinstall, library sudah bisa diimport dan digunakan dengan cara memanggil *function* stem_kalimat dari MPStemmer(). Dataset kemudian diperbaru dengan data yang sudah di *stemming*.

```
# ## Stemming

get_ipython().system('pip install --upgrade git+https://github.com/ariaghora/mpstemmer.git')

get_ipython().system('pip install python-Levenshtein')


from mpstemmer import MPStemmer
stemmer = MPStemmer()


X_train_stem = X_train_nosw.apply(lambda x: stemmer.stem_kalimat(x))
X_test_stem = X_test_nosw.apply(lambda x: stemmer.stem_kalimat(x))
```

Gambar 5. 12 Penggalan Code Stemming

V. Pembuatan Model Naïve Bayes dengan TF-IDF



```
# # Multinomial Naive Bayes classifier combined with TF-IDF (Term Frequency-Inverse Document Frequency)

import pandas as pd
from sklearn.feature_extraction.text import TfidfVectorizer
from sklearn.naive_bayes import MultinomialNB
from sklearn.metrics import classification_report, accuracy_score
from sklearn.pipeline import make_pipeline

model = make_pipeline(TfidfVectorizer(), MultinomialNB())
model.fit(X_train_stem, y_train)

y_pred = model.predict(X_test)
accuracy = accuracy_score(y_test, y_pred)
report = classification_report(y_test, y_pred)

print(f"Accuracy: {accuracy * 100:.2f}%")
print("Classification Report:")
print(report)
```

Gambar 5. 13 Penggalan Code Pembuatan Model

Setelah data selesai di pre-processing, data kemudian dipakai untuk membuat model, model dibuat dengan menggunakan pipeline yang berisi tfidfvectorizer dan juga multinomial Naïve Bayes serta data latih yang telah dilakukan pre-processing. Setelah model terbuat model ditest menggunakan data

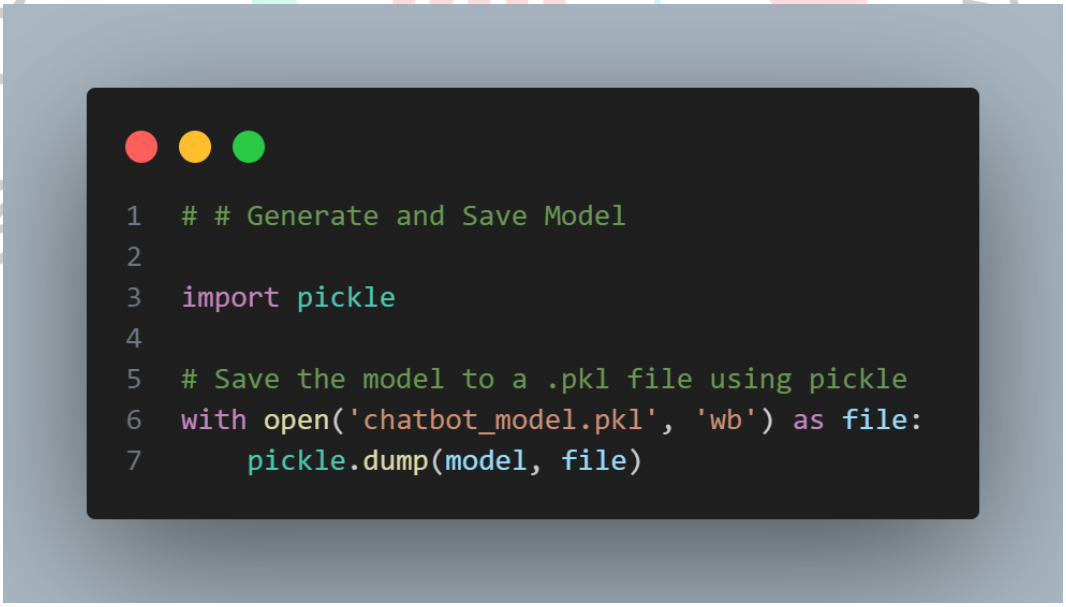
test, setelah model di test, dapat dilihat berapa total akurasi yang dihasilkan model terhadap data test. Report hasil model dapat dilihat pada gambar 5.14.

```
Accuracy: 75.80%
Classification Report:
              precision    recall  f1-score   support

           0       0.74      0.88      0.80      2332
           1       0.79      0.60      0.68      1787

    accuracy                           0.76      4119
   macro avg       0.77      0.74      0.74      4119
weighted avg       0.76      0.76      0.75      4119
```

Gambar 5. 14 Hasil Report Model

VI. Generate Model

Tahap terakhir adalah melakukan *generate* model menjadi .pkl *file* agar model dapat diintegrasikan dengan discord atau pihak ketiga lainnya. Untuk dapat *generate* model menjadi .pkl dibutuhkan sebuah library bernama pickle.



Gambar 5. 15 Penggalan Code Generate Model

Gambar 5.15 merupakan sepenggalan code untuk melakukan penyimpanan model kedalam bentuk file dengan format pkl.

```
df_export = pd.DataFrame({'feature': X_train_stem, 'label': y_train})

# Export the DataFrame to a CSV file
df_export.to_csv('train_data_updated.csv', index=False)
```

Gambar 5. 16 Penggalan Code Export Data Train

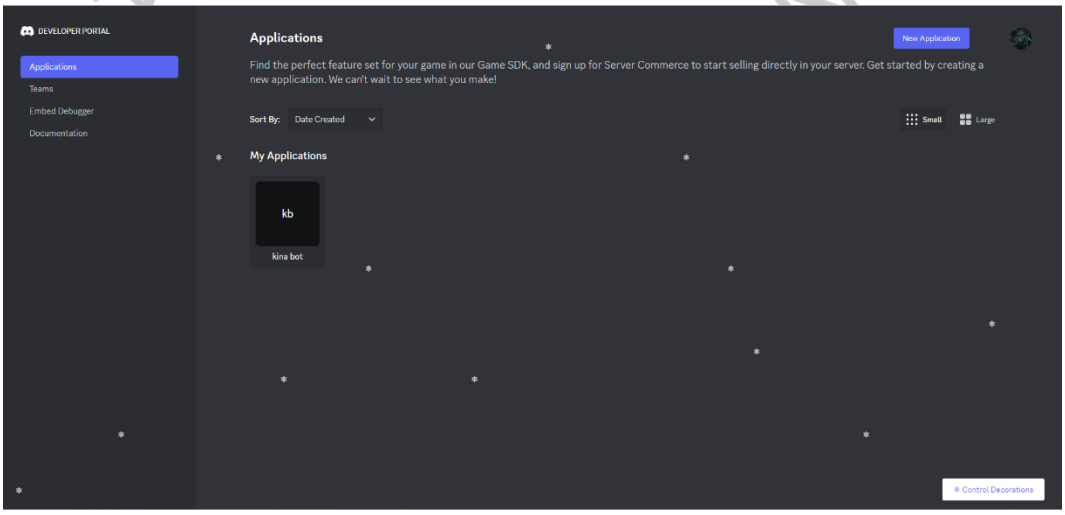
Selain *generate* model ke dalam bentuk file, perlu dilakukan juga untuk melakukan *generate* data latih ke dalam bentuk CSV file. Hal ini diperlukan untuk terus melakukan update terhadap model agar menghasilkan akurasi yang lebih tinggi dan lebih sesuai dengan konteks.

5.2.2. Integrasi Model dengan Discord Bot

Setelah model terbuat, langkah selanjutnya adalah mengintegrasikan model dengan discord *bot*. Discord menyediakan sebuah portal untuk para penggunanya agar dapat membuat aplikasi yang nantinya dapat diintegrasikan dengan discord, yang bernama Developer Portal. Berikut adalah langkah – langkah membuat aplikasi / bot di Developer Portal Discord :

1. Buka Developer portal Discord

   Langkah pertama adalah buka developer portal discord di link https://discord.com/developers/applications. Gambar 5.17 merupakan tampilan awal dari dashboard developer portal discord.

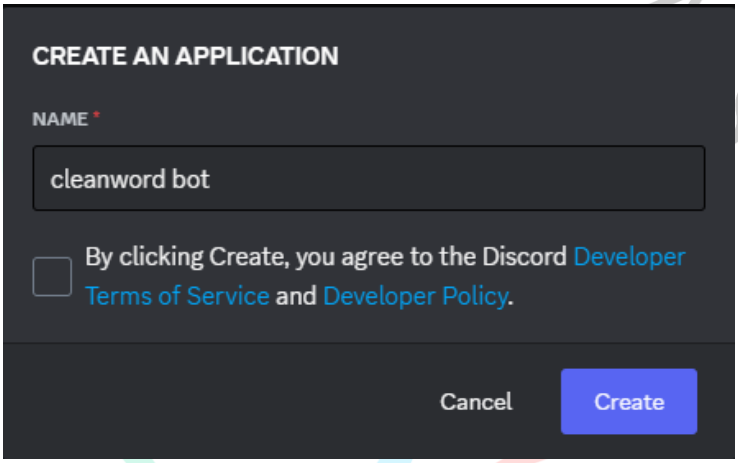


Gambar 5. 17 Dashboard Developer Portal

Terdapat 4 menu utama di dashboard ini, yaitu *Application, Teams, Embed Debugger,* dan *Documentation*. *Application* untuk mengurusi aplikasi yang akan kia buat, *teams* untuk membuat tim yang berkontribusi dalam aplikasi yang dibuat, *debugger* untuk menautkan log ke dalam aplikasi yang kita buat, *documentation* berisi dokumentasi tata cara penggunaan dari developer portal discord.

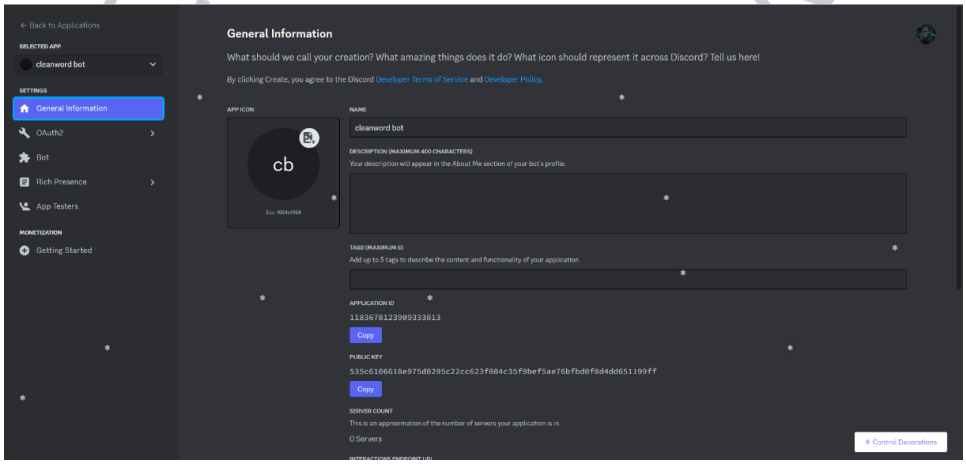
2. Membuat aplikasi baru

Untuk membuat aplikasi baru dapat memilih tombol *New Application* di pojok kanan atas, lalu akan muncul kotak dialog seperti gambar 5.18.



Gambar 5. 18 Create Application

Masukkan nama aplikasi yang ingin dibuat, dan klik persetujuan *Discord Developer Terms of Service and Developer Policy.* Lalu klik tombol *Create.* Setelah terbuat maka akan muncul dashboard baru dari aplikasi kita, seperti pada gambar 5.19.

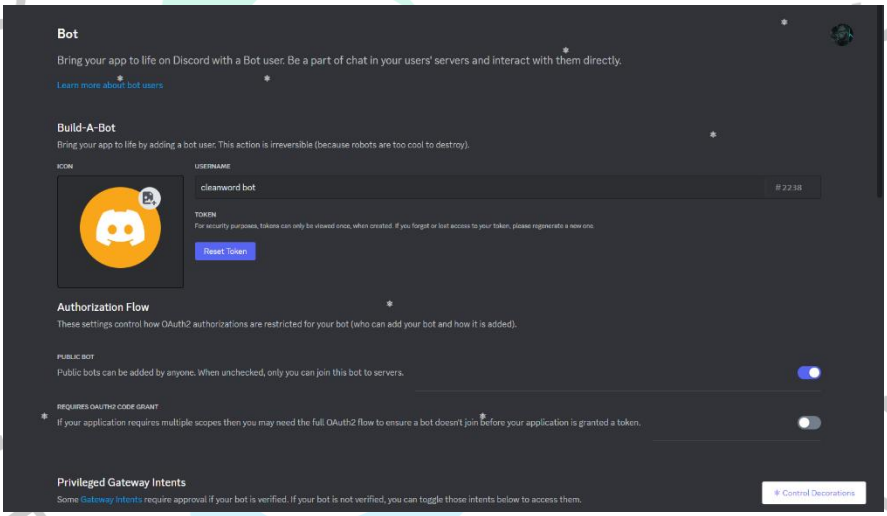


Gambar 5. 19 Dashboard Aplikasi

Pada tampilan dashboard aplikasi Terdapat 6 menu utama di dashboard ini, yaitu *General Information*, OAuth2, Bot, *Rich Presence* dan *App Testers*. General Information berkaitan dengan informasi umum mengenai aplikasi, OAuth2 untuk menerapkan authentikasi terhadap aplikasi, Bot untuk memanage Bot yang akan kita buat, *Rich Presence* semua asset yang akan kita gunakan ketika memakai aplikasi, *App testers* untuk melakukan test terhadap aplikasi dengan mengundang user discord lainnya.

3. Setting Bot

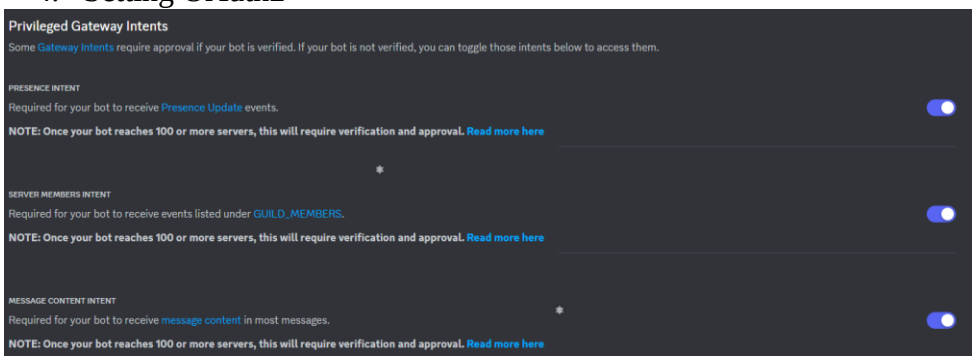
Langkah berikutnya adalah setting bot di menu *Bot*, untuk menentukan *privilege* apa saja yang akan dimiliki bot. Berikut adalah tampilan pada menu Bot.



Gambar 5. 20 Halaman Bot

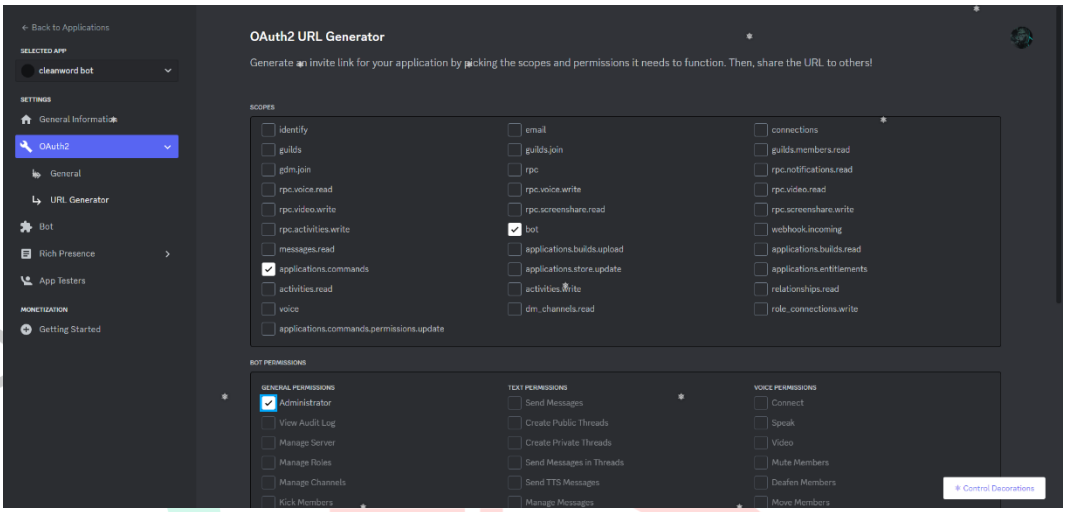
Masuk ke dalam bagian *Privileged Gateway Intents* dan *checklist* pada *presence intent*, *server members intent*, dan *message content intent*. Lakukan seperti pada gambar 5.20 dan jangan lupa untuk melakukan *save changes*.

4. Setting OAuth2



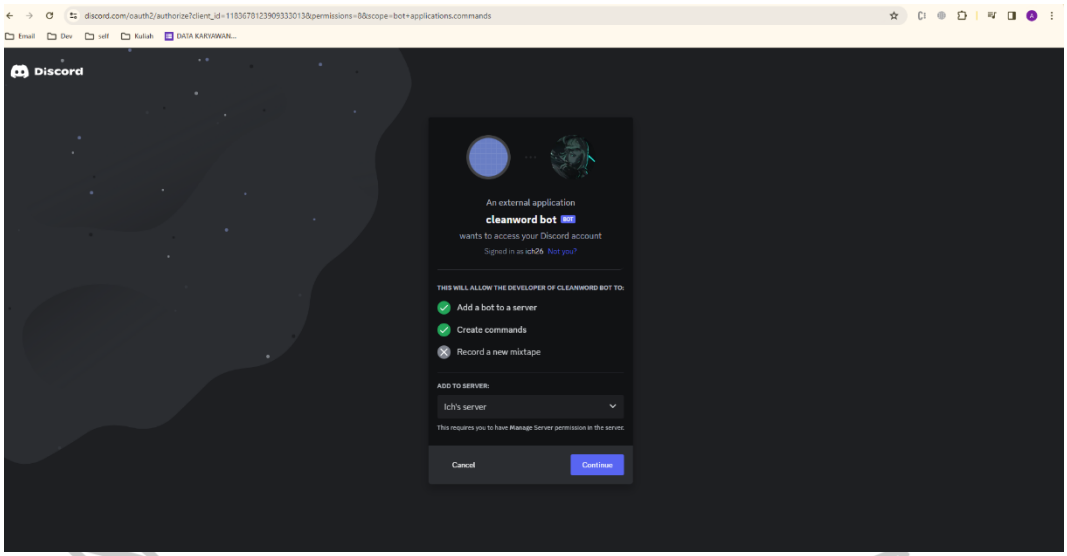
Gambar 5. 21 Setting OAuth2

Langkah berikutnya adalah melakukan setting pada OAuth2, untuk memilih cakupan dan *permissions* yang dapat dilakukan oleh bot, selain itu juga terdapat *URL Generator* untuk mengundang bot kedalam server pengguna. Lakukan setting pada menu OAuth2 lalu *URL Generator* seperti pada gambar 5.22.

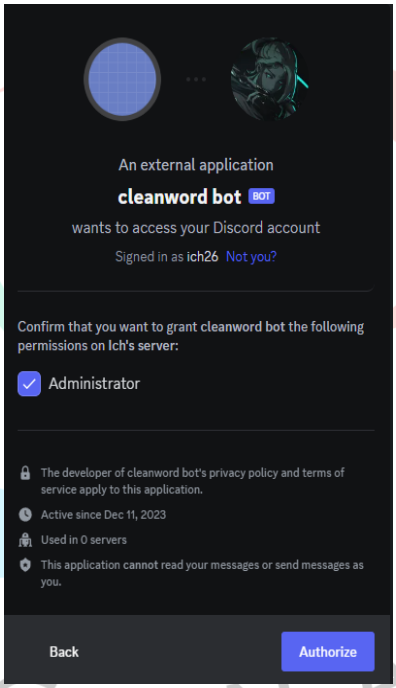


Gambar 5. 22 Setting OAuth2 URL Generator

Pilih bot dan *application.commands* pada *scopes*, ini menandakan aplikasi memiliki *scopes* sebagai bot, dan juga dapat melakukan *commands* pada kolom *chat* di discord. Lalu pada bot *permissions checklist* pada Administrator, yang menandakan aplikasi/bot memiliki *permission* seperti administrator pada server yang nantinya ditempati aplikasi/bot. Setelah melakukan *checklist* maka akan muncul *generated url* di bagian bawah menu, kita dapat menggunakan url tersebut untuk mengundang bot ke dalam server di discord. Copy link tersebut dan buka di dalam tab baru, maka akan muncul kotak dialog seperti pada gambar 5.23, untuk melakukan *invite* bot ke dalam server.

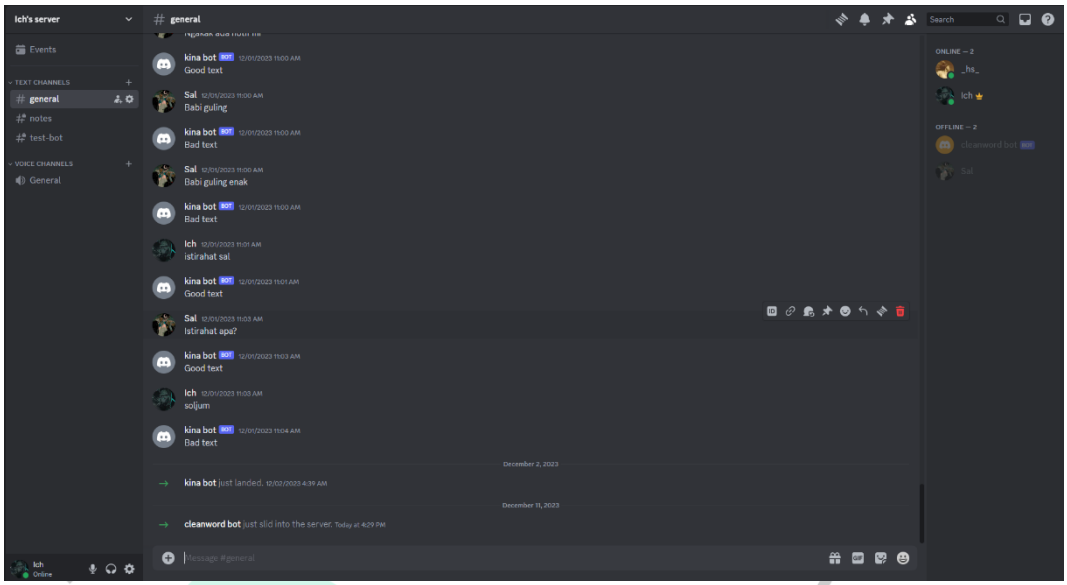


Gambar 5. 23 Invite Bot ke Dalam Server



Gambar 5. 24 Authorize Bot

Pilih server yang akan ditempati bot lalu klik tombol *Continue.* Lalu kemudian klik tombol Authorize. Dan Bot sudah masuk ke dalam server yang dipilih, terlihat seperti di gambar 5.25.



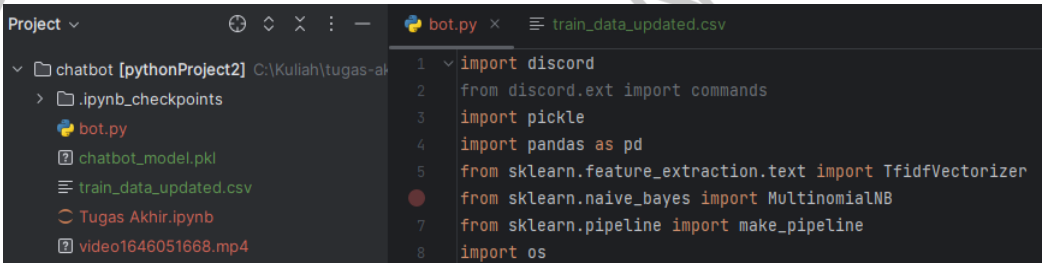
Gambar 5. 25 Bot Sudah Terdaftar di Server

5. Integrasi discord bot dengan model

Seperti yang terlihat pada gambar 5.25 bot masih dalam keadaan offline, artinya bot masih belum dapat bekerja. Untuk mengaktifkan bot dan membuat bot bekerja kita perlu melakukan pengaktifan melalui code dan token yang telah disediakan. Code akan ditulis menggunakan bahasa python.

a. Buat project baru

Buat project baru dan masukkan model .pkl serta data latih yang telah menjadi .csv. Serta lakukan import pada library yang dibutuhkan. Seperti yang terlihat pada gambar 5.26.



Gambar 5. 26 Penggalan Code Import Library

b. Membuat intent serta client dengan token aplikasi

Setelah mengimport buat variable intent serta client menggunakan function discord.Client dan juga buat variable token yang diambil dari developer portal bagian bot.

```
intents = discord.Intents.default()
intents.message_content = True

client = discord.Client(intents=intents)
token = "MTE3OTgyNzQzNzA2NDk1Mzk1Ng.GtCuV5._4PlyNor2Xh8_CdZevKWmfHKV7WqjyiTFPIPPg"

client.run(token)
```

Gambar 5. 27 Penggalan Code Membuat Intent

c. Membuat client event untuk menyapa

Client event merupakan sebuah annotasi untuk menyatakan apa yang dilakukan bot terhadap suatu event, berikut contohnya :

```
@client.event
async def on_ready():
    print(f'Clean word Bot | Logged in as {client.user}')


@client.event
async def on_message(message):
    if message.author == client.user:
        return

    if message.content.startswith('hello clean word bot'):
        await message.reply('Hi aku clean world bot')

client.run(token)
```

Gambar 5. 28 Penggalan Code Client Event

Pada gambar 5.28 terdapat 2 client.event, client.event yang pertama dengan fungsi on_ready menunjukkan jika aplikasi dijalankan maka akan menampilkan tulisan yang tertera pada fungsi print di terminal. Client.event yang kedua dengan fungsi on_message dan parameter message artinya jika ada message tertentu maka aplikasi akan bereaksi terhadap message tersebut, contoh pada gambar 5.28 diatas adalah jika mengirimkan “hello clean word bot”, bot akan membalas dengan “Hi aku clean world bot”

d. Menambahkan fungsi filter badword

Pada fungsi on_message kita akan menambahkan code untuk melakukan pemfilteran apakah chat yang dikirimkan merupakan bad word atau bukan. Code dapat dilihat pada gambar 5.29

```
with open('chatbot_model.pkl', 'rb') as file:
    model = pickle.load(file)

users_with_bad_words = set()


@client.event
async def on_message(message):
    if message.author == client.user:
        return

    if message.content.startswith('hello clean word bot'):
        await message.reply('Hi aku clean world bot')

    mention = message.author.mention

    prediction = model.predict([message.content])[0]
    response = 'Bad Words' if prediction == 1 else None
    if response is not None:
        users_with_bad_words.add(message.author.id)
        await message.channel.send("Peringatan kepada " + mention + " Anda telah mengirimkan " + response)

client.run(token)
```

Gambar 5. 29 Penggalan Code Fungsi On_Message

Pertama – tama perlu mengekstrak file model.pkl menjadi model dengan menggunakan pickle.load. setelah itu membuat variable users_with_bad_words untuk menampung user yang mengirimkan bad words nantinya. Pada code diatas bot akan meresponse jika ada user yang mengirimkan bad words dan memberikan peringatan kepadanya. Selanjutnya user tersebut disimpan ke dalam set users_with_bad_words.

e. Menambahkan fungsi list users with bad words

Untuk menampilkan variable users_with_bad_words diperlukanlah sebuah perintah. Berikut adalah penggalan kode untuk menambahkan perintah menampilkan list user yang mengirimkan bad words.

```
    # Print the list of users with bad words
    if message.content.startswith('/users'):
        user_mentions = [f'<@{user_id}>' for user_id in users_with_bad_words]
        users_list = ', '.join(user_mentions)
        await message.channel.send(f'Users with bad words: {users_list}')
```

Gambar 5. 30 Penggalan Code List Users

Jika mengirimkan perintah “/users” maka bot akan menampilkan list *user* yang telah mengirimkan bad words pada server tersebut, dengan format @namauser sehingga user termention di dalam chat dari bot.

f. Menambahkan fungsi penambahan data train dan melatih ulang model

Karena model dibuat dengan data – data yang ada sebelum model dibuat, adakalanya banyak kata – kata atau kalimat yang belum masuk dalam data latih model, sehingga diperlukanlah penambahan data dan pembuatan model ulang, agar bot atau model semakin update dan dapat memahami gaya bahasa terkini. Namun untuk melakukan update model diperlukan restart terhadap bot, sehingga bot akan berhenti bekerja selama beberapa detik.

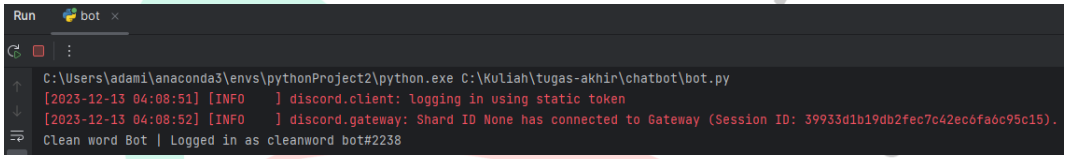
```
50      if message.content.startswith('add to train'):
51          df_existing = pd.read_csv('train_data_updated.csv')
52          split_string = message.content.split()
53          index_of_train = split_string.index("train")
54          feature = " ".join(split_string[index_of_train + 1:-1])
55          label = split_string[-1]
56
57          new_features = [
58              feature,
59          ]
60          new_labels = [
61              label,
62          ]
63
64          df_new_data = pd.DataFrame({'feature': new_features, 'label': new_labels})
65          df_combined = pd.concat( objs: [df_existing, df_new_data], ignore_index=True)
66          df_combined.to_csv( path_or_buf: 'train_data_updated.csv', index=False)
67          csv_file_path = 'train_data_updated.csv'
68          df = pd.read_csv(csv_file_path)
69
70          X_train_stem = df['feature'].tolist()
71          y_train= df['label'].tolist()
72
73          model_updated = make_pipeline( *steps: TfidfVectorizer(), MultinomialNB())
74          model_updated.fit(X_train_stem, y_train)
75
76          pickle_file_path = 'chatbot_model.pkl'
77          new_pickle_file_path = 'new_model.pkl'
78          with open(new_pickle_file_path, 'wb') as new_pickle_file:
79              pickle.dump(model_updated, new_pickle_file)
80          os.remove(pickle_file_path)
81          os.rename(new_pickle_file_path, pickle_file_path)
82          response = 'model updated need restart bot'
83          await message.channel.send(response)
```

Gambar 5. 31 Penggalan Code Add to Train

Gambar 5.31 merupakan penggalan kode untuk melatih ulang model, perintah yang digunakan adalah *chat* yang diawali dengan kata "add to train" ditambah dengan kata atau kalimat yang ingin dimasukkan beserta labelnya. Data selanjutnya dimasukkan ke dalam row baru csv, dan setelah itu dilakukan lagi pelatihan model serta mereplace model lama dengan model baru.

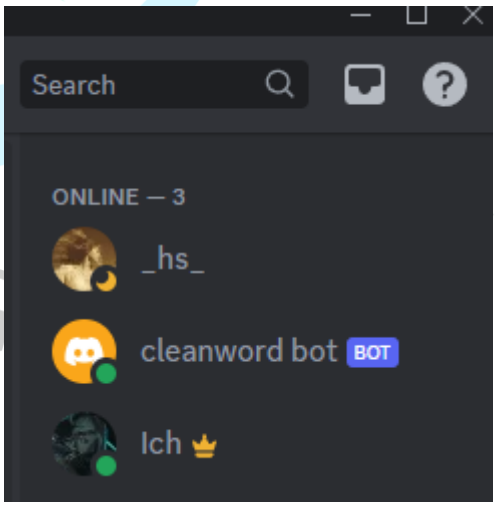
6. Menjalankan Aplikasi / Bot

Langkah terakhir adalah menjalankan Bot untuk mengetahui apakah bot sudah berjalan dengan baik atau belum.



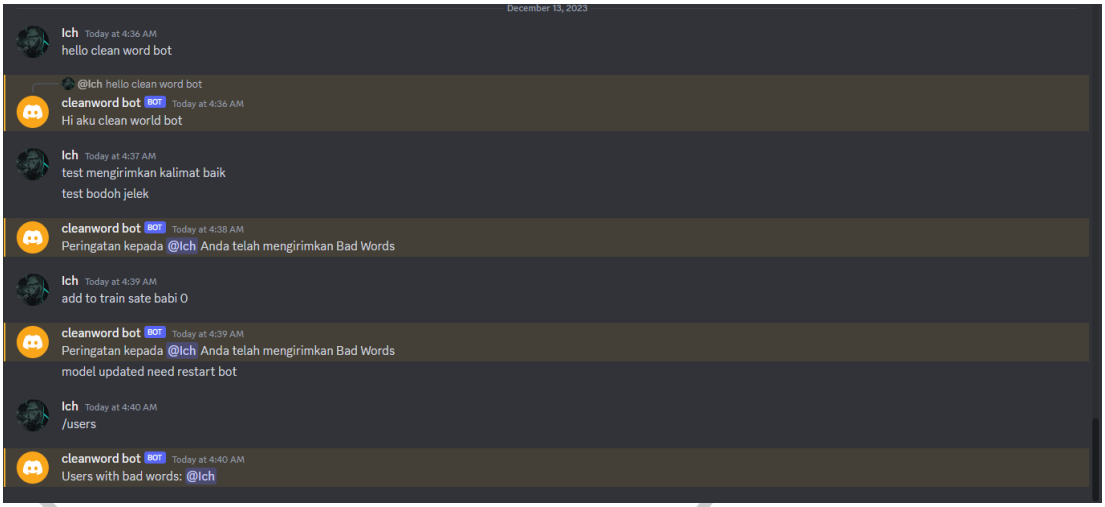
Gambar 5. 32 Aplikasi Berjalan

Gambar 5.32 adalah log terminal ketika aplikasi / bot dijalankan, terlihat terdapat pesan *cleanword* Bot | logged in as cleanword bot#2238. Dan jika kita lihat pada aplikasi discord, maka bot sudah online seperti pada gambar 5.33.



Gambar 5. 33 Bot Online

Dan berikut adalah contoh response yang diberikan bot terhadap pesan yang terdapat dalam chat.



Gambar 5. 34 Bot Meresponse

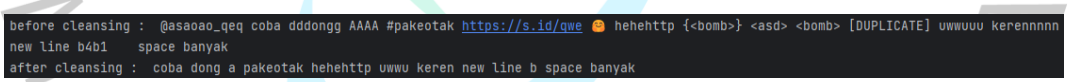
5.2.3. Hasil Pengujian

Aplikasi yang sudah selesai dibangun perlu dilakukan pengujian untuk memastikan bahwa aplikasi / bot berjalan dengan semestinya. Bot diuji dengan 2 metode yaitu pengujian kotak hitam untuk menguji fungsionalitasnya, dan pengujian kotak putih untuk menguji algoritma atau logika code yang telah dibuat.

5.2.3.1 Pengujian Kotak Putih

Tabel 5. 3 Scenario Pengujian Kotak Putih

	Algoritma	Perancangan Code
1	*Cleansing data*	*def cleanse(data):* *# lowercase* *data = data.lower()* *# remove newline* *data = data.replace(r'\n', ' ')* *# remove links* *data = re.sub(r'http\S+', '', data)* *data = re.sub(r'bit.ly\S+', '', data)* *data = re.sub(r't.co\S+', '', data)* *data = re.sub(r's.id\S+', '', data)* *# remove mentions* *data = data.replace("_", "")* *data = re.sub("(@[A-Za-z0-9]+)", " ", data)* *# remove punctutations*

		punct = string.punctuation.replace("<", "").replace(">", "").replace("[", "").replace("]", "") *translator = str.maketrans(punct, ' ' * len(punct))* *data = data.translate(translator)* *# remove characters between [], <>, or {}* *data = re.sub("[\[].*?[\]]", " ", data)* *data = re.sub(r'<[^>]+>', ' ', data)* *data = re.sub(r'{[^>]+}', ' ', data)* *# remove emoji* *emoji_pattern = re.compile("["* *u"\U0001F600-\U0001F64F" # emoticons* *u"\U0001F300-\U0001F5FF" # simbol & piktogram* *u"\U0001F680-\U0001F6FF" # transportasi & simbol map* *u"\U0001F700-\U0001F77F" # alat teknis* *u"\U0001F780-\U0001F7FF" # panah* *u"\U0001F800-\U0001F8FF" # kategori ekstra* *u"\U0001F900-\U0001F9FF" # emoji wajah* *u"\U0001FA00-\U0001FA6F" # emoji* *u"\U0001FA70-\U0001FAFF" # emoji kategori tambahan* *u"\U00002702-\U000027B0" # dingbat* *u"\U000024C2-\U0001F251"* *"]+", flags=re.UNICODE)* *data = emoji_pattern.sub(r'', data)* *# remove extra space* *data = ' '.join(data.split())* *return data*
Hasil		
		
Pembahasan		
Cleansing data berfungsi untuk mengubah *message* menjadi *lowercase*, menghapus baris baru, menghapus link, menghapus mentions, menghapus tanda kurung, menghapus emoji, serta menghapus spasi yang terlalu banyak .		
2	Integrasi model dengan bot discord	*intents = discord.Intents.default()* *intents.message_content = True* *client = discord.Client(intents=intents)* *token = "MTE4MzY3ODEyMzkwOTMzMzAxMw.GHnM33.QZcIu8rsH6De6N2hIYE8F2xo4Y3L75vvFcXJ-4"* *with open('chatbot_model.pkl', 'rb') as file:* *model = pickle.load(file)* *@client.event* *async def on_ready():* *print(f'Clean word Bot | Logged in as {client.user}')*

Hasil

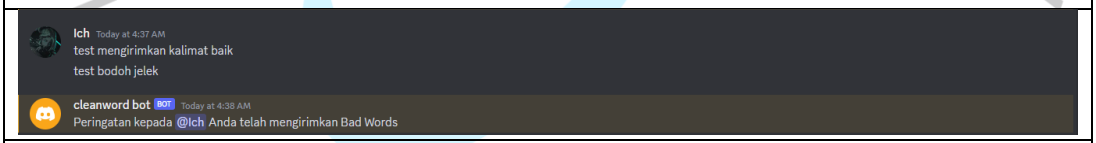
```
C:\Users\adami\anaconda3\envs\pythonProject2\python.exe C:\Kuliah\tugas-akhir\chatbot\bot.py
[2023-12-13 22:30:19] [INFO    ] discord.client: logging in using static token
[2023-12-13 22:30:20] [INFO    ] discord.gateway: Shard ID None has connected to Gateway (Session ID: d4462476f0e00eeca8e17e830da32a16).
Clean word Bot | Logged in as cleanword bot#2238
```

Pembahasan

Untuk mengintegrasikan model dengan bot discord diperlukan library discord, lalu membuat intents dan menjalankan client dengan token yang ada pada dashboard developer portal. Langkah terakhir adalah melakukan *load* model dari file berformat .pkl.

3	filter *bad word*	*@client.event* *async def on_message(message):* *if message.author == client.user:* *return* *if message.content.startswith('hello clean word bot'):* *await message.reply('Hi aku clean world bot')* *print("before cleansing : ", message.content)* *message.content = cleanse(message.content)* *print("after cleansing : ",message.content)* *mention = message.author.mention* *prediction = model.predict([message.content])[0]* *response = 'Bad Words' if prediction == 1 else None* *if response is not None:* *users_with_bad_words.add(message.author.id)* *await message.channel.send("Peringatan kepada " + mention + "* *Anda telah mengirimkan " + response)*

Hasil



Pembahasan

Fungsi on_message berfungsi jika ada message yang masuk ke dalam server bot akan merespons, pada code diatas bot akan merespons jika disapa, dan akan meresponse jika model yang telah dibuat sebelumnya mengidentifikasi *bad word* di dalam *message chat* yang dikirimkan member/user discord di dalam server. Jika tidak ada *bad word* maka bot tidak meresponse apapun. User yang mengirimkan *bad words* kemudian dimasukkan ke dalam list.

4	List User yang mengirimkan *bad words*	@client.event async def on_message(message): ---------- ---------- ---------- # Print the list of users with bad words if message.content.startswith('/users'): user_mentions = [f'<@{user_id}>' for user_id in users_with_bad_words] users_list = ', '.join(user_mentions)

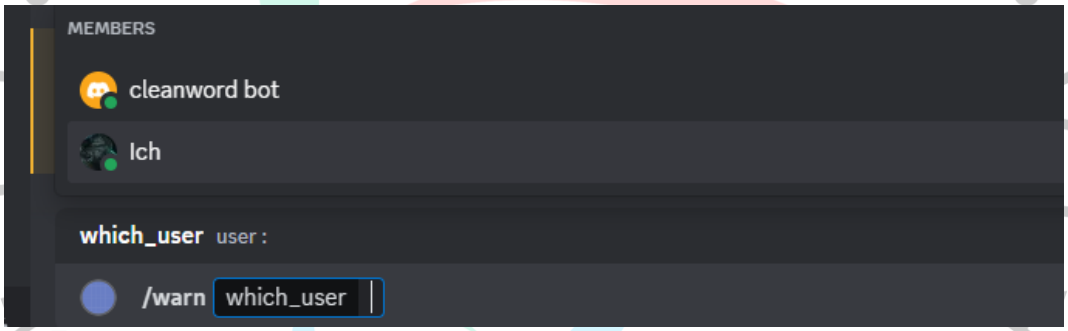
		await message.channel.send(f'Users with bad words: {users_list}')
Hasil		
		
Pembahasan		
Masih di dalam *function* on_message, untuk dapat menampilan list user yang mengirimkan *bad words* diperlukan pesan "/users" kemudian bot akan meresponse dengan list user disertai mention kepada user tersebut.		
5	Update data dan Model	@client.event async def on_message(message): ---------- ---------- ---------- if message.content.startswith('add to train'): df_existing = pd.read_csv('train_data_updated.csv') split_string = message.content.split() index_of_train = split_string.index("train") feature = " ".join(split_string[index_of_train + 1:-1]) label = split_string[-1] new_features = [feature,] new_labels = [label,] df_new_data = pd.DataFrame({'feature': new_features, 'label': new_labels}) df_combined = pd.concat([df_existing, df_new_data], ignore_index=True) df_combined.to_csv('train_data_updated.csv', index=False) csv_file_path = 'train_data_updated.csv' df = pd.read_csv(csv_file_path) X_train_stem = df['feature'].tolist() y_train= df['label'].tolist() model_updated = make_pipeline(TfidfVectorizer(), MultinomialNB()) model_updated.fit(X_train_stem, y_train)

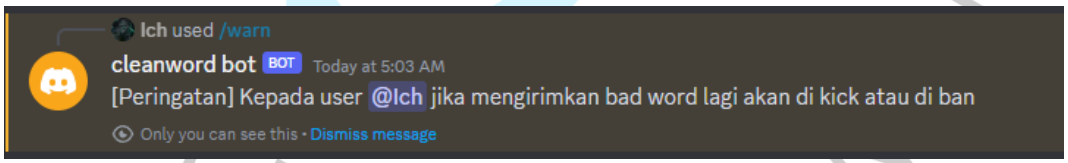
		pickle_file_path = 'chatbot_model.pkl' new_pickle_file_path = 'new_model.pkl' with open(new_pickle_file_path, 'wb') as new_pickle_file: pickle.dump(model_updated, new_pickle_file) os.remove(pickle_file_path) os.rename(new_pickle_file_path, pickle_file_path) response = 'model updated need restart bot' await message.channel.send(response)
Hasil		
		
Pembahasan		
Masih di dalam *function* on_message, untuk menambahkan data train dan melatih ulang model atau mengupdate model maka *user* harus menggunakan pesan “add to train” + ungkapan yang ingin ditambah + label ungkapan. Label berisikan nilai 1 atau 0. Data kemudian ditambahkan ke dalam csv train data, dan kemudian model dilatih ulang lagi. Setelah pelatihan model selesai, model lama *direplace* dengan model yang baru saja diupdate. Namun untuk mengimplementasi model terbaru diperlukan restart terhadap bot.		

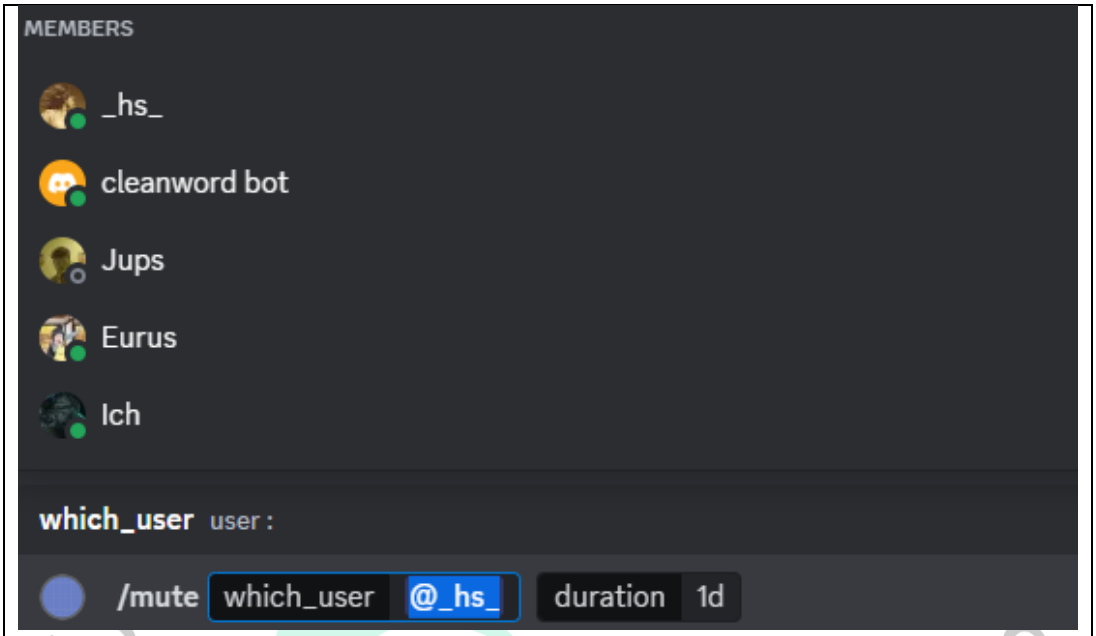
5.2.3.2. Pengujian Kotak Hitam

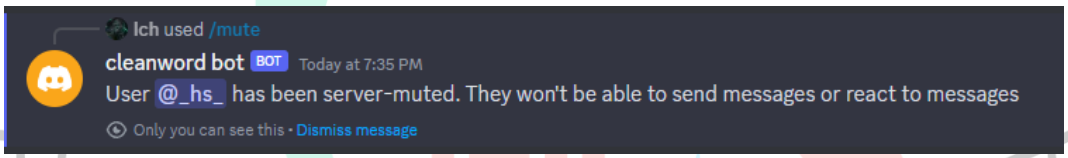
Tabel 5. 4 Skenario Pengujian Kotak Hitam

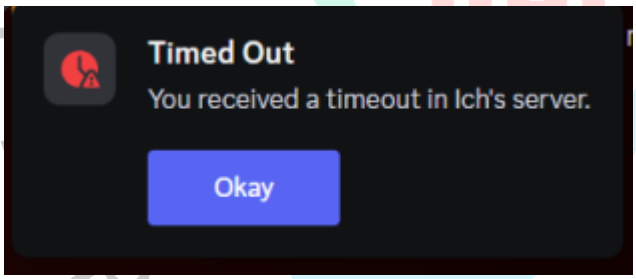
No	Skenario Pengujian	Hasil yang Diharapkan
1	*User* menambahkan bot ke dalam server	Bot berhasil masuk ke dalam server
Hasil		
		
Kesimpulan: Bot berhasil masuk ke dalam server		
2	*User* menginput ‘bot pintar sekali’	Bot tidak meresponse apa apa
Hasil		
		

<table>
<tr><td colspan="3">Kesimpulan: Karena user mengirimkan ungkapan positif, maka bot tidak memberikan response</td></tr>
<tr><td>3</td><td>User menginput ‘bot bodoh dan jelek’</td><td>Bot meresponse memberikan warning terhadap pengirim</td></tr>
<tr><td colspan="3">Hasil
 
</td></tr>
<tr><td colspan="3">Kesimpulan: Karena user mengirimkan bad word, maka bot memberikan response</td></tr>
<tr><td>5</td><td>User (admin) mengakses list pengirim bad words</td><td>Menampilkan list pengirim bad words</td></tr>
<tr><td colspan="3">Hasil
 
</td></tr>
<tr><td colspan="3">Kesimpulan: Sistem berhasil menampilkan list pengirim bad words</td></tr>
<tr><td>6</td><td>User mengirimkan peringatan kepada salah satu pengirim bad words</td><td>Bot mengirimkan peringatan kepada pengirim bad words</td></tr>
<tr><td colspan="3">Hasil
 


</td></tr>
<tr><td colspan="3">Kesimpulan: Bot berhasil mengirimkan peringatan kepada Member</td></tr>
<tr><td>7</td><td>User (admin) melakukan mute kepada salah satu pengirim bad words</td><td>Melakukan mute terhadap member</td></tr>
<tr><td colspan="3">Hasil</td></tr>
</table>

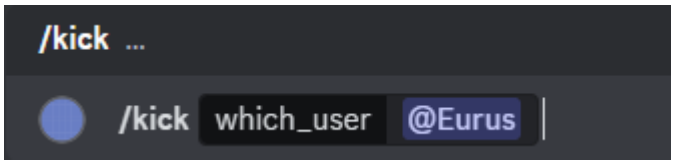


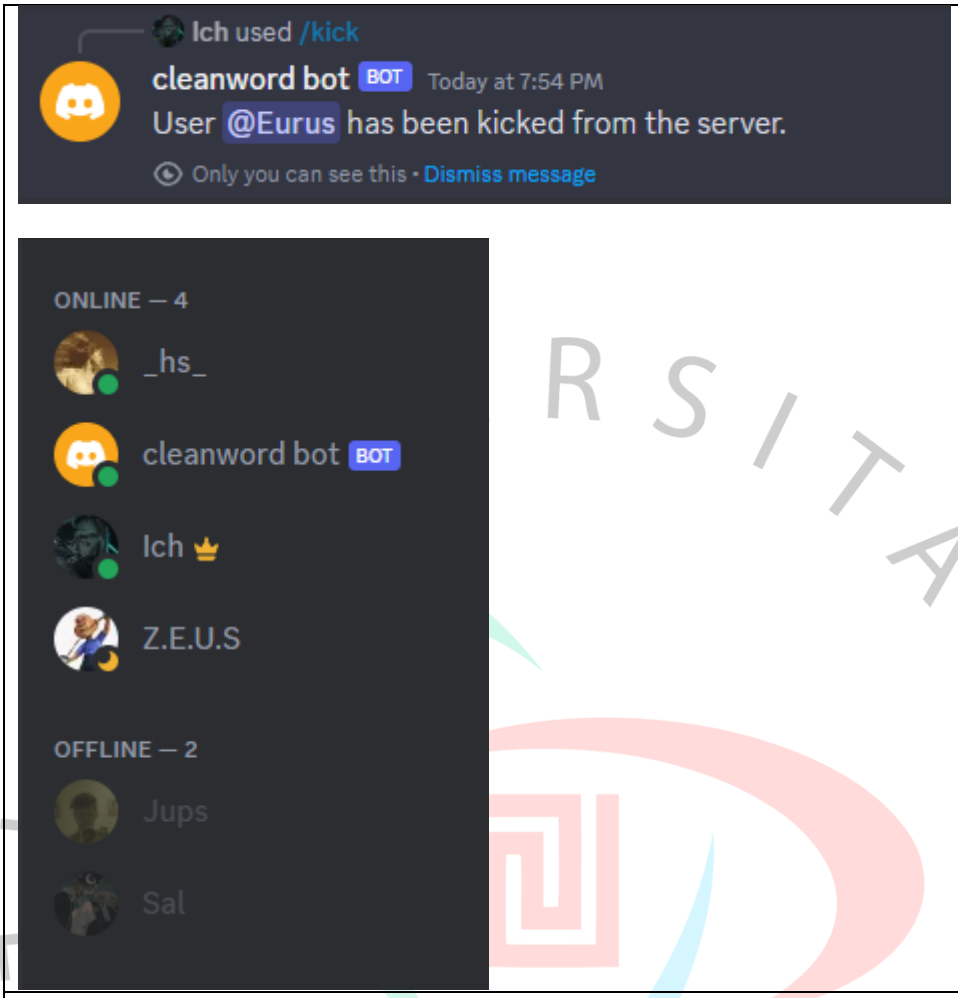


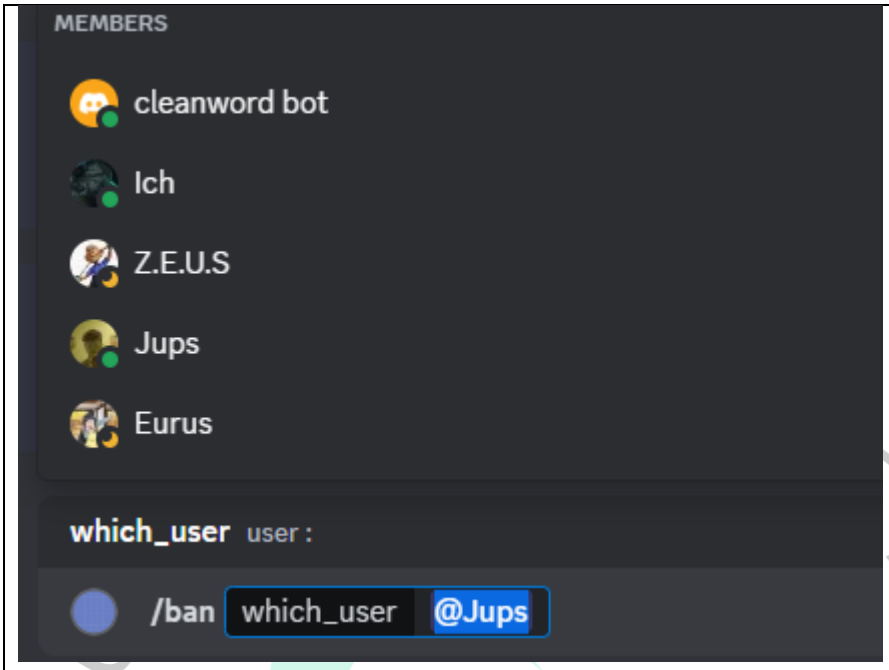


Kesimpulan: Sistem berhasil melakukan *mute*, dan member yang di *mute* tidak bisa mengirimkan pesan ataupun bereaksi terhadap pesan member lain selama durasi *mute*		
8	*User* (admin) melakukan kick kepada salah satu pengirim *bad words*	Bot melakukan *kick* kepada member, dan member keluar dari server

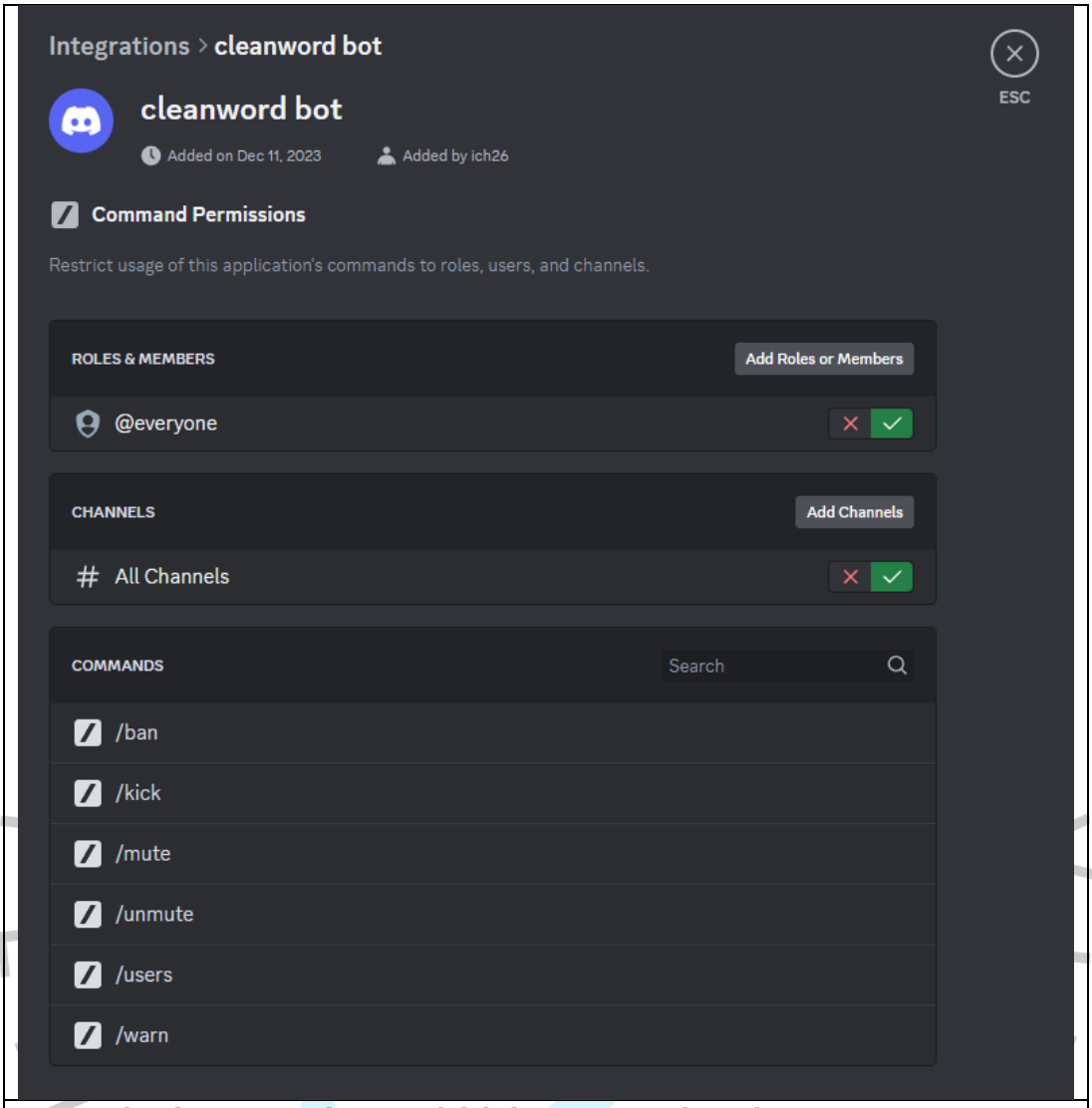
Hasil

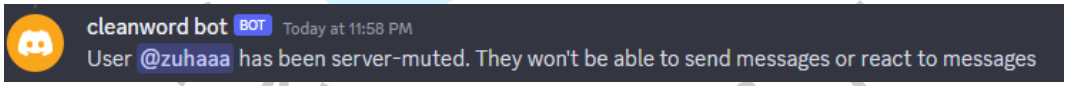
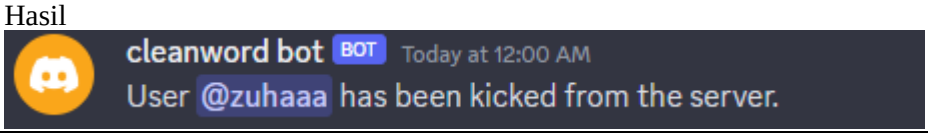


<table>
<tr><td colspan="3"> 
</td></tr>
<tr><td colspan="3">Kesimpulan: Bot berhasil melakukan kick terhadap user Eurus</td></tr>
<tr><td>9</td><td>User (admin) melakukan ban kepada salah satu pengirim bad words</td><td>Bot melakukan ban terhadap member dan member tidak bisa join lagi ke dalam server</td></tr>
<tr><td colspan="3">Hasil</td></tr>
</table>



Kesimpulan: Bot berhasil melakukan ban terhadap user Jups dengan nama jups_z		
10	*User* memilih menu setting untuk melihat list fitur pada bot	Menampilan list command pada bot
Hasil		



Kesimpulan: list command tertampil di halaman setting discord		
11	*User* mengirim *bad words* sebanyak 3 kali	Bot melakukan mute terhadap *user* selama 1 hari
Hasil  		
Kesimpulan: Bot melakukan mute terhadap user yang mengirimkan *bad word* 3 kali		
12	User yang sudah berakhir masa *mute* lalu mengirimkan *bad word* lagi	Bot melakukan kick terhadap user tersebut
Hasil  		
Kesimpulan: Bot berhasil melakukan kick terhadap *user* tersebut		
13	User yang sudah terkick, dan masuk lagi, lalu mengirim *bad word* lagi	Bot melakukan kick terhadap user tersebut
Hasil		


Kesimpulan: Bot berhasil melakukan banned terhadap user