# BAB IV
# IMPLEMENTASI DAN PEMBAHASAN

Pada bab ini akan membahas tentang implementasi dan pengujian sistem. Implementasi dilakukan setelah perancangan sistem telah dilakukan, kemudian diimplementasikan pada bahasa pemograman yang akan menghasilkan suatu program. Pada tahap selanjutnya, pengujian dilakukan terhadap sistem yang dikembangkan untuk melihat kekurangan atau bug yang terdapat dalam sistem untuk pengembangan sistem selanjutnya.

## 1.1 Implementasi

Implementasi sistem adalah proses pembuatan program atau *development* dari rancangan sistem yang telah dibuat. Proses implementasi dibangun berdasarkan hasil dari tahap analisis dan desain sistem yang telah dilakukan pada bab sebelumnya. Pada tahap ini dilakukan implementasi atau kode pembuatan *chatbot* menggunakan Bahasa pemrograman javascript untuk membuat *endpoint* dari *chatbot* dan python untuk membuat *processing* yaitu pembobotan kata dengan TF-IDF dan VSM (*Term Frequency – Inverse Document Frequency* dan *Vector Space Model*) serta perhitungan *similarity* (kemiripan) menggunakan *cosine similarity.*

### 1.1.1 Implementasi *Whatsapp Api*

Implementasi pada tahap ini adalah implementasi pembuatan whatsapp api menggunakan modul atau library whatsapp web js yang dikembangkan oleh pedroslopez. Pembuatan whatsapp api ini menggunakan Bahasa

pemrograman javascript dan node js sebagai runtime untuk menjalankan javascript dari sisi server.

Berikut adalah implementasi dari pembuatan whatsapp api :

1. Routing whatsapp api

```
app.get('/', (req, res) => {
    res.sendFile('index.html', {
        root: __dirname
    });
});
```

Gambar 4.1 Kode *routing* whatsapp api

Kode program ini berfungsi sebagai routing dari whatsapp api sehingga dapat diakses melalui browser. Dimana index.html berisi kode program User Interface atau tampilan untuk menampilkan qr code kedalam browser.

*2. listen program*

```
server.listen(port, function () {
    console.log('App running on *: ' + port);
});
```

Gambar 4.2 Kode *listen program*

Kode program ini berfungsi untuk menjalankan *server* pada *port* yang ditentukan. Disini penulis menyimpan *port* kedalam *variable port* dengan settingan default yaitu :

Port = 8000

Maka saat program dijalankan, program akan berjalan pada port 8000, dengan mengetikan https://*localhost:8000.* Cara untuk menjalankan program adalah dengan mengetikan run npm panca pada jendela terminal run npm adalah perintah default dari node.js sedangkan panca adalah nama alias dari file app.js yang telah disimpan.

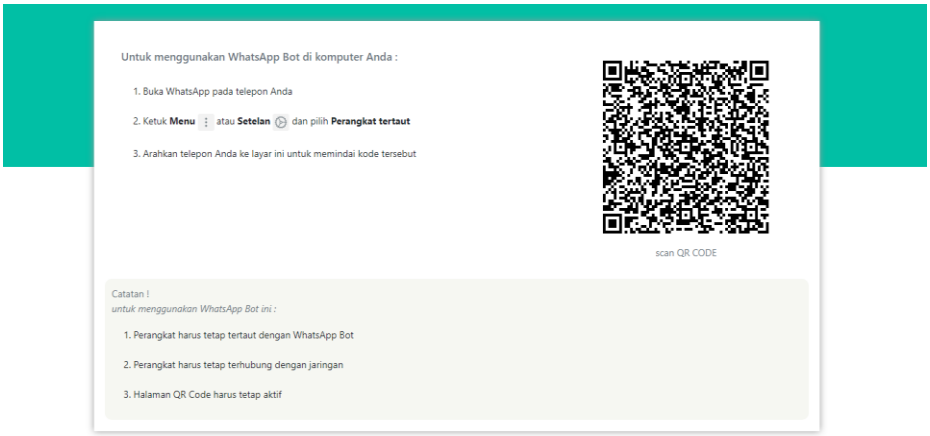
*3. show qr code to browser*

```
client.on('qr', (qr) => {
    qrcode.toDataURL(qr, (err, url) => {
        socket.emit('qr', url);
        socket.emit('message', 'QR Code received, scan please!');
        console.log('QR Code received, scan please!');
    });
});
```

Gambar 4.3 Kode *show qr code*

Kode program ini berfungsi untuk menampilkan qr code kedalam browser, dimana qr code yang semula berupa sebuah kode ditransformasikan menjadi kode batang menggunakan library qr-code.

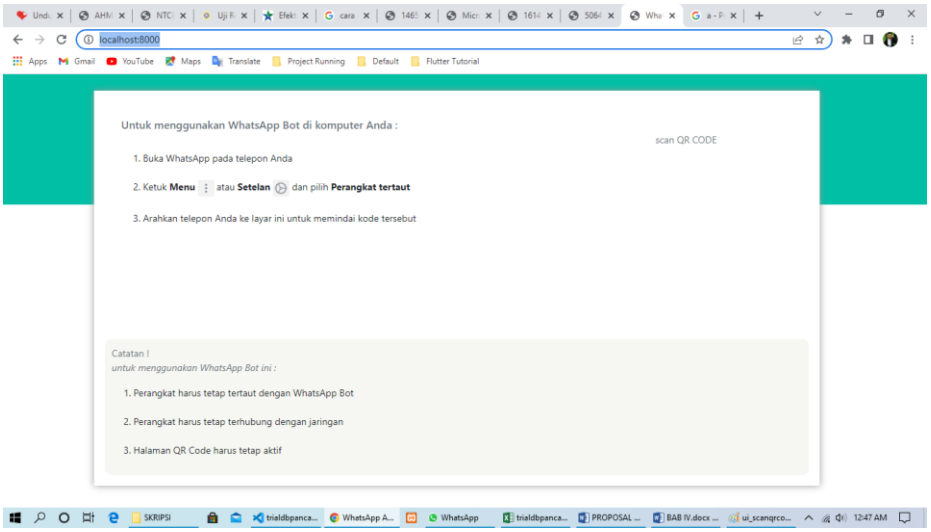
Berikut adalah tampilan atau user interface dari halaman scan qr code :




Gambar 4.4 Halaman *Scan qr code*

Tampilan ini mengadaptasi dari tampilan whatsapp browser dengan sedikit perubahan pada penambahan catatan untuk menggunakan WhatsApp Api ini.

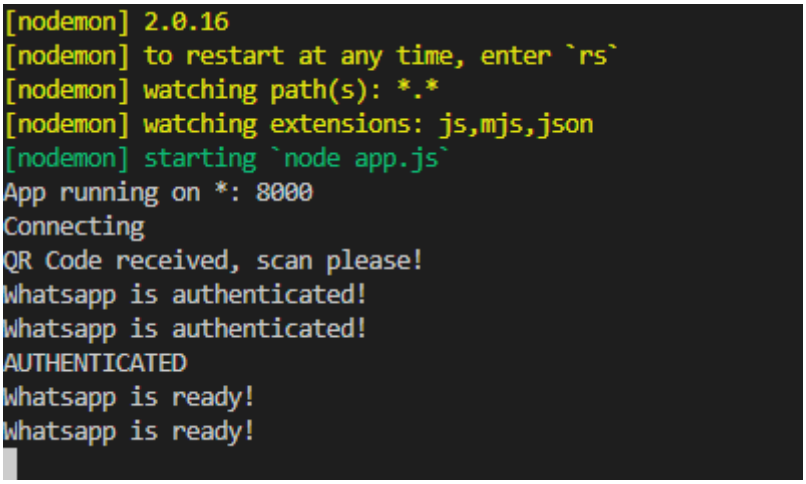
Dan berikut ini adalah tampilan saat qr code berhasil discan :




Gambar 4.5 Halaman *success scan qr code*

Kode batang atau *qr code* secara otomatis akan menghilang apabila *qr code* berhasil discan. Kode *qr* dihilangkan untuk menghindari pengguna melakukan scanning 2 (dua) kali terhadap qr code yang ditampilkan.

Dibawah ini adalah pesan (*log*) yang akan ditampilkan dari sisi server bahwa *qr code* berhasil di scan oleh pengguna dan whatsapp api sudah berjalan dan siap untuk digunakan.




Gambar 4.6 Pesan *success scan qr code*

4. *Endpoint* send and receive messages

```
client.on('message', async msg => {
    console.log('[', msg.from.replace(/\D/g, '') + ' ] ' + msg.body);
    // console.log('Pesan : ', msg);
    var dataToSend;
    const python = spawn('python', ['processing.py', msg.body, msg.from.replace(/\D/g, '')]);
    python.stdout.on('data', async function (data) {
        dataToSend = data.toString();
        console.log("[ Panca ] " + dataToSend)
        if (msg.body !== false && dataToSend.length > 0) {
            await msg.reply(dataToSend);
        }
    });
});
```

Gambar 4.7 Kode *endpoint send* and *receive messages*

Pada gambar 4.7 adalah kode program untuk menerima *request* dan mengirim respon kepada pengguna, dimana *request* disini berupa *query* yang akan dihitung pada tahap selanjutnya yaitu pembobotan dan perhitungan kemiripan.

*Endpoint* akan menerima pesan yang direpresentasikan kedalam *arrow function msg*, kemudian perintah *spawn* dengan parameter pesan dan juga nomor pengirim kepada file pyhton menggunakan *library child process* untuk diproses ditahap selanjutnya. Dari hasil proses tersebut, akan disimpan kedalam *variable dataToSend* yang kemudian dikirimkan kepada pengguna.

### 1.1.2 Implementasi *Tf-Idf (Term frequency – Inverse Document Frequency)* dan *VSM (Vector Space Model)*

Implementasi pada tahap ini adalah proses menghitung bobot dokumen dan bobot query. Untuk melakukan impelementasi processing pada tahap ini penulis menggunakan Bahasa pemrograman python dan beberapa library pendukung seperti pysastrawi dan lain sebagainya. Sebagai studi kasus kedepan, penulis mengambil beberapa sampel data untuk dilakukan proses pembobotan mengggunakan Tf-Idf (Term frequency – Inverse Document Frequency) dan VSM (Vector Space Model).

Berikut adalah sampel dokumen yang akan menjadi bahan studi kasus pada tahap impelemntasi Tf-Idf (Term frequency – Inverse Document Frequency), VSM (Vector Space Model) dan Cosine similairy

Tabel 4.1 sampel dokumen

| **Dokumen** | **Document type** |
|---|---|
| alamat Univerisitas Panca Marga (UPM) ada di Jl Yos Sudarso, No 107 Pabean, Dringu - Probolinggo | dimana |
| ruang pendaftaran mahasiswa baru terletak digedung utara lantai I | dimana |
| Dekan Fakultas Teknik adalah Ir. Haryono, M.H. | siapa |
| Wakil Dekan I (Wadek I) Fakultas Teknik adalah Ahmad Izzuddin, S.T., M.Kom. | siapa |
| Wakil Dekan II (Wadek II) Fakultas Teknik adalah Ira Aprilia, S.Pd., M.Pd. | siapa |
| Wakil Dekan III (Wadek III) Fakultas Teknik adalah Tri Prihatingsih, S.T., M.T. | siapa |
| Ketua Program Studi (Kaprodi) Teknik Elektro adalah Nuzul Hikmah, S.T., M.Kom. | siapa |
| Ketua Program Studi (Kaprodi) Teknik Industri adalah Yustina Suhandini Tjahjaningsih, S.T., M.T. | siapa |
| Ketua Program Studi (Kaprodi) Teknik Mesin adalah Djoko Wahyudi, S.T., M.T. | siapa |
| Registrasi Administrasi adalah layanan terhadap mahasiswa untuk memperoleh status terdaftar pada suatu jenis program studi / jurusan tertentu dilingkungan Universitas Panca Marga | Apa |
| Registrasi Akademik adalah layanan terhadap mahasiswa untuk memperoleh hak dan ijin mengikuti perkuliahan pada suatu program studi / jurusan tertentu dilingkungan Universitas Panca Marga. | apa |
| Registrasi Ulang atau Heregistrasi Adalah Registrasi bagi mahasiswa lama | apa |
| Registrasi Harus dilakukan untuk memberikan status terdaftar kepada mahasiswa, sehingga berhak menggunakan semua fasilitas yang ada di Universitas Panca Marga | kenapa |
| Penyelenggaraan ujian dimaksudkan untuk<br>1) Menilai apakah mahasiswa telah memahami atau menguasai bahan bahasan yang<br>disajikan dalam kuliah.<br>2) Mengelompokkan mahasiswa ke dalam beberapa golongan berdasarkan kemapuan.<br>3) Menilai apakah bahan kuliah disajkan sesuai dalam kurikulum.<br>4) Mengetahuiapakah cara penyajian dosen cukup baik. | kenapa |

1. *Filter Question*

```
for index in range(len(pertanyaan)):
    if pertanyaan[index] in message.split(" "):
        kategori = pertanyaan[index]
        break
    else:
        kategori = ""

for data in contacts:
    if(data[1] == kategori):
        dokumen.append(data[0])
```

Gambar 4.8 Kode *filter questions*

Pada gambar 4.8 adalah kode program untuk memfilter pertanyaan dari *query* yang dikirim pengguna. Dimana pada tahap ini menentukan dokumen yang ada pada sistem untuk dilakukan perhitungan. Sebagai contoh penulis disini menanyakan tentang “berapa biaya spp di universitas panca marga ?”, maka dari *query* tersebut difilter dari kalimat tanya.

Daftar kalimat tanya penulis simpan kedalam *variable* pertanyaan berupa array. Berikut daftar pertanyaan tersebut :

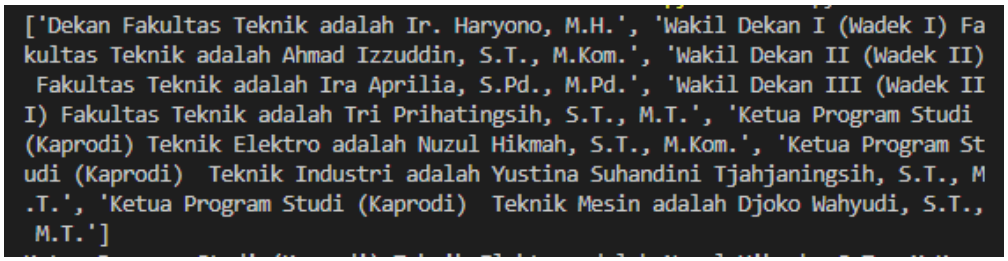
pertanyaan = ["apa", "siapa", "kenapa", "mengapa", "kapan", "dimana", "bagaimana", "berapa"]

dari daftar pertanyaan diatas, terdapat 8 kata tanya yang didukung. Pada contoh query yang tulis berikan, yaitu “siapa ketua program studi Teknik elektro ?”, mengandung kata tanya “siapa”. Maka dokumen yang akan dihitung adalah dokumen yang mengandung *document type* kata tanya siapa.

Berikut ini adalah hasil dokumen yang terpilih berdasarkan *filter question* :

Tabel 4.2 Hasil *filter question*

| **Dokumen** |
|---|
| Dekan Fakultas Teknik adalah Ir. Haryono, M.H. |
| Wakil Dekan I (Wadek I) Fakultas Teknik adalah Ahmad Izzuddin, S.T., M.Kom. |
| Wakil Dekan II (Wadek II) Fakultas Teknik adalah Ira Aprilia, S.Pd., M.Pd. |
| Wakil Dekan III (Wadek III) Fakultas Teknik adalah Tri Prihatingsih, S.T., M.T. |
| Ketua Program Studi (Kaprodi) Teknik Elektro adalah Nuzul Hikmah, S.T., M.Kom. |
| Ketua Program Studi (Kaprodi) Teknik Industri adalah Yustina Suhandini Tjahjaningsih, S.T., M.T. |
| Ketua Program Studi (Kaprodi) Teknik Mesin adalah Djoko Wahyudi, S.T., M.T. |




Gambar 4.9 Hasil *filter question*

Gambar 4.9 merupakan hasil dari *filter question* berdasarkan kata tanya yang terkandung dalam *query*. Pada gambar tersebut, penulis tampilkan kedalam jendela terminal menggunakan fungsi *print*() yang ada didalam Bahasa pemrograman python.

2. *Case Folding*

Case Folding adalah tahap mengubah semua huruf menjadi huruf kecil. karena penulis untuk text preprocessing menggunakan library *pysastrawi*, maka proses case folding sudah ditangani oleh library tersebut.

Namun dalam python, cara untuk mnegubah huruf besar menjadi huruf kecil adalah dengan menggunakan fungsi "*lower()*".

Berikut contoh hasil *case folding* :

Tabel 4.3 Hasil filter question

| **Sebelum Case Folding** | **Sesudah Case Folding** |
|---|---|
| Dekan Fakultas Teknik adalah Ir. Haryono, M.H. | dekan fakultas teknik adalah ir. haryono, m.h. |
| Wakil Dekan I (Wadek I) Fakultas Teknik adalah Ahmad Izzuddin, S.T., M.Kom. | wakil dekan i (wadek i) fakultas teknik adalah ahmad izzuddin, s.t., m.kom. |
| Wakil Dekan II (Wadek II) Fakultas Teknik adalah Ira Aprilia, S.Pd., M.Pd. | wakil dekan ii (wadek ii) fakultas teknik adalah ira aprilia, s.pd., m.pd. |
| Wakil Dekan III (Wadek III) Fakultas Teknik adalah Tri Prihatingsih, S.T., M.T. | wakil dekan iii (wadek iii) fakultas teknik adalah tri prihatingsih, s.t., m.t. |
| Ketua Program Studi (Kaprodi) Teknik Elektro adalah Nuzul Hikmah, S.T., M.Kom. | ketua program studi (kaprodi) teknik elektro adalah nuzul hikmah, s.t., m.kom. |
| Ketua Program Studi (Kaprodi) Teknik Industri adalah Yustina Suhandini Tjahjaningsih, S.T., M.T. | ketua program studi (kaprodi) teknik industri adalah yustina suhandini tjahjaningsih, s.t., m.t. |
| Ketua Program Studi (Kaprodi) Teknik Mesin adalah Djoko Wahyudi, S.T., M.T. | ketua program studi (kaprodi) teknik mesin adalah djoko wahyudi, s.t., m.t. |

3. *Tokenizing*

```
def STEMMINGDOKUMEN(dokumen):
    listkata = []

    for kalimat in dokumen:
        for kata in stemmer.stem(kalimat).split(" "):
            if kata not in stopwords and kata not in listkata:
                listkata.append(kata)

    return listkata
```

Gambar 4.9 Spliting dokumen

Pada Gambar 4.9 adalah tahap tokenizing dengan memecah kalimat menjadi daftar kata atau token. Untuk melalukan hal itu disini penulis menggunakan fungsi “split()” yang ada dalam python.

Kalimat dipisahkan berdasarkan spasi, Kemudian hasil dari splitting disimpan kedalam sebuah variable array untuk dibawa pada tahap selanjutnya. Berikut adalah hasil untuk tokenizing :

Tabel 4.4 Hasil Tokenizing

| |
|---|
| 'dekan', 'fakultas', 'teknik', 'adalah', 'ir.', 'haryono,', 'm.h.', 'wakil', 'i', '(wadek', 'i)', 'ahmad', 'izzuddin,', 's.t.,', 'm.kom.', 'ii', 'ii)', 'ira', 'aprilia,', 's.pd.,', 'm.pd.', 'iii', 'iii)', 'tri', 'prihatingsih,', 'm.t.', 'ketua', 'program', 'studi', '(kaprodi)', 'elektro', 'nuzul', 'hikmah,', '', 'industri', 'yustina', 'suhandini', 'tjahjaningsih,', 'mesin', 'djoko', 'wahyudi,' |

4. Stopword Removal

```
        if kata not in stopwords and kata not in listquery:
            listquery.append(kata)
```

Gambar 4.11 *Stopword removal*

Stopword Removal adalah proses menghilangkan kata-kata yang biasanya dan sering muncul dan akan diabaikan dalam pemrosesan. Gambar 4.11 adalah kode program untuk melakukan stopword removal.

Berikut hasil *Stopword Removal* :

Tabel 4.5 Hasil Stopword Removal

| **Sebelum stopword removal** | **Sesudah stopword removal** |
|---|---|
| 'dekan', 'fakultas', 'teknik', 'adalah', 'ir.', 'haryono,', 'm.h.', 'wakil', 'i', '(wadek', 'i)', 'ahmad', 'izzuddin,', 's.t.,', 'm.kom.', 'ii', 'ii)', 'ira', 'aprilia,', 's.pd.,', 'm.pd.', 'iii', 'iii)', 'tri', 'prihatingsih,', 'm.t.', 'ketua', 'program', 'studi', '(kaprodi)', 'elektro', 'nuzul', 'hikmah,', '', 'industri', 'yustina', 'suhandini', 'tjahjaningsih,', 'mesin', 'djoko', 'wahyudi,' | 'dekan', 'fakultas', 'teknik', 'ir.', 'haryono,', 'm.h.', 'wakil', 'i', '(wadek', 'i)', 'ahmad', 'izzuddin,', 's.t.,', 'm.kom.', 'ii', 'ii)', 'ira', 'aprilia,', 's.pd.,', 'm.pd.', 'iii', 'iii)', 'tri', 'prihatingsih,', 'm.t.', 'ketua', 'program', 'studi', '(kaprodi)', 'elektro', 'nuzul', 'hikmah,', '', 'industri', 'yustina', 'suhandini', 'tjahjaningsih,', 'mesin', 'djoko', 'wahyudi,' |

5. *Stemming*

```
def STEMMINGDOKUMEN(dokumen):
    listkata = []

    for kalimat in dokumen:
        for kata in stemmer.stem(kalimat).split(" "):
            if kata not in stopwords and kata not in listkata:
                listkata.append(kata)

    return listkata
```

Gambar 4.12 *Stemming* Dokumen

pada gambar 4.12 adalah kode program untuk melakukan *stemming* pada dokumen dan *query*. *Stemming* adalah proses untuk mengembalikan kata menjadi kata dasar, didalam *pysastrawi* tesedia fungsi untuk menjadikan kata menjadi kata dasar, *pysastrawi* juga dapat menghilangkan karakter seperti “,;-=&*%$] dll.

Berikut adalah contoh hasil dari *Stemming* setelah melalui beberapa tahap t*ext preprocessing* tersebut :

Tabel 4.6 Hasil *Stemming*

| **Sebelum Stemming** | **Sesudah Stemming** |
|---|---|
| 'dekan', 'fakultas', 'teknik', 'ir.', 'haryono,', 'm.h.', 'wakil', 'i', '(wadek', 'i)', 'ahmad', 'izzuddin,', 's.t.,', 'm.kom.', 'ii', 'ii)', 'ira', 'aprilia,', 's.pd.,', 'm.pd.', 'iii', 'iii)', 'tri', 'prihatingsih,', 'm.t.', 'ketua', 'program', 'studi', '(kaprodi)', 'elektro', 'nuzul', 'hikmah,', '', 'industri', 'yustina', 'suhandini', 'tjahjaningsih,', 'mesin', 'djoko', 'wahyudi,' | 'dekan', 'fakultas', 'teknik', 'ir', 'haryono', 'wakil', 'wadek', 'ahmad', 'izzuddin', 'kom', 'ii', 'ira', 'aprilia', 'pd', 'iii', 'tri', 'prihatingsih', 'ketua', 'program', 'studi', 'kaprodi', 'elektro', 'nuzul', 'hikmah', 'industri', 'yustina', 'suhandini', 'tjahjaningsih', 'mesin', 'djoko', 'wahyudi' |

6. *Term Frequency Document*

```
def TERMFREQUENCYDOKUMEN(stemming, dokumen):
    tf = []

    for _ in range(len(dokumen)):
        tf.append(dict(zip(stemming, [0 for _ in range(len(stemming))])))

    for index, kalimat in enumerate(dokumen):
        for kata in stemmer.stem(kalimat).split(" "):
            if kata in tf[index]:
                tf[index][kata] += 1
                # tf[index][kata] = round(log10(1 + (tf[index][kata])), 3)

    return tf
```

Gambar 4.14 *Term frequency* dokumen

Pada gambar 4.14 adalah kode program untuk menghitung tf dari semua dokumen, dimana setiap term dihitung dan diberikan nilai berdasarkan seberapa banyak term atau kata tersebut dalam sebuah dokumen.

pada gambar 4.14 fungsi *termfrequencydokumen*() menerima sebuah parameter yaitu hasil stemming dan seluruh dokumen yang sudah melalui tahap filter questions. Langkah pertama penulis memberikan nilai 0 pada semua term yang sudah distemming, menggunakan perulangan *for*(). Kemudian penulis tambahkan atau *append*() kedalam array untuk setiap term yang diberikan nilai 0 (nol) ke *variable* tf.

Pada perulangan kedua, lakukan perulangan sebanyak dokumen kemudian lakukan stemming kembali, jika kata yang sudah distemming ada didalam tf[index] maka tambahkan nilai 1. Berikut hasil *term frequency* :

Tabel 4.7 Hasil *term frequency*

| **Term** | **D1** | **D2** | **D3** | **D4** | **D5** | **D6** | **D7** |
|---|---|---|---|---|---|---|---|
| dekan | 1 | 1 | 1 | 1 | 0 | 0 | 0 |
| fakultas | 1 | 1 | 1 | 1 | 0 | 0 | 0 |
| teknik | 1 | 1 | 1 | 0 | 0 | 0 | 0 |
| ir | 1 | 0 | 0 | 0 | 0 | 0 | 0 |
| haryono | 1 | 0 | 0 | 0 | 0 | 0 | 0 |

| Term | D1 | D2 | D3 | D4 | D5 | D6 | D7 |
|---|---|---|---|---|---|---|---|
| wakil | 0 | 1 | 1 | 1 | 0 | 0 | 0 |
| wadek | 0 | 1 | 1 | 1 | 0 | 0 | 0 |
| ahmad | 0 | 1 | 0 | 0 | 0 | 0 | 0 |
| izzuddin | 0 | 1 | 0 | 0 | 0 | 0 | 0 |
| kom | 0 | 1 | 1 | 0 | 1 | 1 | 0 |
| ii | 0 | 0 | 2 | 0 | 0 | 0 | 0 |
| ira | 0 | 0 | 1 | 0 | 0 | 0 | 0 |
| aprilia | 0 | 0 | 1 | 0 | 0 | 0 | 0 |
| pd | 0 | 0 | 2 | 0 | 0 | 0 | 0 |
| iii | 0 | 0 | 0 | 2 | 0 | 0 | 0 |
| tri | 0 | 0 | 0 | 1 | 0 | 0 | 0 |
| prihatingsih | 0 | 0 | 0 | 1 | 0 | 0 | 0 |
| ketua | 0 | 0 | 0 | 0 | 1 | 1 | 1 |
| program | 0 | 0 | 0 | 0 | 1 | 1 | 1 |
| studi | 0 | 0 | 0 | 0 | 1 | 1 | 1 |
| kaprodi | 0 | 0 | 0 | 0 | 1 | 1 | 1 |
| elektro | 0 | 0 | 0 | 0 | 1 | 0 | 0 |
| nuzul | 0 | 0 | 0 | 0 | 1 | 0 | 0 |
| hikmah | 0 | 0 | 0 | 0 | 1 | 0 | 0 |
| industri | 0 | 0 | 0 | 0 | 0 | 1 | 0 |
| yustina | 0 | 0 | 0 | 0 | 0 | 1 | 0 |
| suhandini | 0 | 0 | 0 | 0 | 0 | 1 | 0 |
| tjahjaningsih | 0 | 0 | 0 | 0 | 0 | 1 | 0 |
| mesin | 0 | 0 | 0 | 0 | 0 | 0 | 1 |
| djoko | 0 | 0 | 0 | 0 | 0 | 0 | 1 |
| wahyudi | 0 | 0 | 0 | 0 | 0 | 0 | 1 |

7. *Document Frequency*

```
def DOCUMENTFREQUENCY(stemming, tf):
    df = (dict(zip(stemming, [0 for _ in range(len(stemming))])))

    for index, kalimat in enumerate(tf):
        for kunci, nilai in kalimat.items():
            if nilai:
                df[kunci] += 1

    return df
```

Gambar 4.15 *Document Frequency*

Pada gambar 4.15 adalah kode progam untuk menghitung dokumen frekuensi, dimana dokumen frekuensi ini adalah tahap menghitung banyaknya term dalam keseluruhan dokumen.

Fungsi documentfrequency() menerima 2 (dua) parameter yaitu stemming dan tf, Langkah pertama penulis memberikan nilai 0 pada semua term yang sudah distemming, kemudian penulis menggunakan fungsi zip untuk menggabungkan setiap term dengan nilai 0 (nol) ke variable df.

Pada langkah kedua, lakukan perulangan sebanyak tf kemudian tambahkan nilai 1 untuk setiap df yang tersimpan. Berikut hasil document frequency :

Tabel 4.8 Hasil *document frequency*

| |
|---|
| 'dekan': 4, 'fakultas': 4, 'teknik': 7, 'ir': 1, 'haryono': 1, 'wakil': 3, 'wadek': 3, 'ahmad': 1, 'izzuddin': 1, 'kom': 2, 'ii': 1, 'ira': 1, 'aprilia': 1, 'pd': 1, 'iii': 1, 'tri': 1, 'prihatingsih': 1, 'ketua': 3, 'program': 3, 'studi': 3, 'kaprodi': 3, 'elektro': 1, 'nuzul': 1, 'hikmah': 1, 'industri': 1, 'yustina': 1, 'suhandini': 1, 'tjahjaningsih': 1, 'mesin': 1, 'djoko': 1, 'wahyudi': 1 |

8. *Inverse Document Frequency*

```
def INVERSEDOCUMENTFREQUENCY(df, dokumen):
    idf = {}

    for kunci, nilai in df.items():
        idf[kunci] = len(dokumen) / nilai

    for kunci, nilai in idf.items():
        idf[kunci] = log10(nilai)
    return idf
```

Gambar 4.16 *Inverse document frequency*

Pada Gambar 4.17 adalah kode program untuk menghitung inverse document frequency, Panjang dokumen dibagi dengan nilai df (document frequency) kemudian dari hasil pembagian tersebut di log10. Kemudian Hasil perhitungan tersebut disimpan kedalam variable array idf,

Berikut adalah hasil perhitungan idf :

Tabel 4.9 Hasil *Inverse document frequency*

| **Token** | **Bobot** | **Token** | **Bobot** |
|---|---|---|---|
| dekan | 0.24303804868629 | aprilia | 0.84509804001425 |
| fakultas | 0.24303804868629 | pd | 0.84509804001425 |
| teknik | 0.0 | iii | 0.84509804001425 |
| ir | 0.8450980400142 | tri | 0.84509804001425 |
| haryono | 0.8450980400142 | prihatingsih | 0.84509804001425 |
| wakil | 0.36797678529459 | ketua | 0.36797678529459 |
| wadek | 0.36797678529459 | program | 0.36797678529459 |
| ahmad | 0.84509804001425 | studi | 0.36797678529459 |
| izzuddin | 0.84509804001425 | kaprodi | 0.36797678529459 |
| kom | 0.54406804435027 | elektro | 0.84509804001425 |
| ii | 0.84509804001425 | nuzul | 0.84509804001425 |
| ira | 0.84509804001425 | hikmah | 0.84509804001425 |
| yustina | 0.84509804001425 | industry | 0.84509804001425 |
| suhandini | 0.84509804001425 | tjahjaningsih | 0.84509804001425 |
| mesin | 0.84509804001425 | djoko | 0.84509804001425 |
| wahyudi | 0.84509804001425 | | |

9. *Term Weighting*

```
def WEIGHT(tf, idf):
    w = []

    for index, dokumen in enumerate(tf):
        w.append({})
        for kunci, nilai in dokumen.items():
            w[index][kunci] = nilai * idf[kunci]

    return w
```

Gambar 4.17 Kode *Term weighting*

Pada Gambar 4.17 adalah kode program untuk menghitung *term weighting*, pada tahap perhitungan *term weighting* semua hasil perhitungan tf dikalikan dengan df. Perhitungan *term weighting* dilakukan sebanyak $tf_i$, pada setiap *term*.

Berdasarkan hasil perhitungan pada tahap sebelumnya, yaitu tahap menghitung tf (*term frequency*) dan idf (*inverse document frequency*), maka dapat dihitung bobot kata pada setiap dokumen.

Berikut ini adalah hasil perhitungan *term weighting* (bobot kata) setelah melalui beberapa tahap *processing* :

Tabel 4.10 Hasil *term weighting*

| **Term** | **D1** | **D2** | **D3** | **D4** | **D5** | **D6** | **D7** |
|---|---|---|---|---|---|---|---|
| dekan | 0.243 | 0.243 | 0.243 | 0.243 | 0.0 | 0.0 | 0.0 |
| fakultas | 0.243 | 0.243 | 0.243 | 0.243 | 0.0 | 0.0 | 0.0 |
| teknik | 0.0 | 0.0 | 0.0 | 0.0 | 0.0 | 0.0 | 0.0 |
| ir | 0.845 | 0.0 | 0.0 | 0.0 | 0.0 | 0.0 | 0.0 |
| haryono | 0.845 | 0.0 | 0.0 | 0.0 | 0.0 | 0.0 | 0.0 |
| wakil | 0.0 | 0.367 | 0.367 | 0.367 | 0.0 | 0.0 | 0.0 |
| wadek | 0.0 | 0.367 | 0.367 | 0.367 | 0.0 | 0.0 | 0.0 |
| ahmad | 0.0 | 0.845 | 0.0 | 0.0 | 0.0 | 0.0 | 0.0 |
| izzuddin | 0.0 | 0.845 | 0.0 | 0.0 | 0.0 | 0.0 | 0.0 |
| kom | 0.0 | 0.544 | 0.0 | 0.0 | 0.544 | 0.0 | 0.0 |
| ii | 0.0 | 0.0 | 1.690 | 0.0 |  | 0.0 | 0.0 |
| ira | 0.0 | 0.0 | 0.845 | 0.0 |  | 0.0 | 0.0 |
| aprilia | 0.0 | 0.0 | 0.845 | 0.0 |  | 0.0 | 0.0 |
| pd | 0.0 | 0.0 | 1.690 | 0.0 |  | 0.0 | 0.0 |
| iii | 0.0 | 0.0 | 0.0 | 1.690 |  | 0.0 | 0.0 |
| tri | 0.0 | 0.0 | 0.0 | 0.845 |  | 0.0 | 0.0 |
| prihatingsih | 0.0 | 0.0 | 0.0 | 0.845 |  | 0.0 | 0.0 |
| ketua | 0.0 | 0.0 | 0.0 | 0.0 | 0.367 | 0.367 | 0.367 |
| program | 0.0 | 0.0 | 0.0 | 0.0 | 0.367 | 0.367 | 0.367 |
| studi | 0.0 | 0.0 | 0.0 | 0.0 | 0.367 | 0.367 | 0.367 |
| kaprodi | 0.0 | 0.0 | 0.0 | 0.0 | 0.845 | 0.367 | 0.367 |
| elektro | 0.0 | 0.0 | 0.0 | 0.0 | 0.845 | 0.0 | 0.0 |
| nuzul | 0.0 | 0.0 | 0.0 | 0.0 | 0.845 | 0.0 | 0.0 |
| hikmah | 0.0 | 0.0 | 0.0 | 0.0 | 0.845 | 0.0 | 0.0 |
| industri | 0.0 | 0.0 | 0.0 | 0.0 | 0.0 | 0.845 | 0.0 |
| yustina | 0.0 | 0.0 | 0.0 | 0.0 | 0.0 | 0.845 | 0.0 |
| suhandini | 0.0 | 0.0 | 0.0 | 0.0 | 0.0 | 0.845 | 0.0 |
| tjahjaningsih | 0.0 | 0.0 | 0.0 | 0.0 | 0.0 | 0.845 | 0.0 |
| mesin | 0.0 | 0.0 | 0.0 | 0.0 | 0.0 | 0.0 | 0.845 |
| djoko | 0.0 | 0.0 | 0.0 | 0.0 | 0.0 | 0.0 | 0.845 |
| wahyudi | 0.0 | 0.0 | 0.0 | 0.0 | 0.0 | 0.0 | 0.845 |

10.*Vector document*

```
def JARAKDOKUMEN(w):
    jd = []
    hasil = []

    for index, dokumen in enumerate(w):
        jd.append({})
        total = 0
        for kunci, nilai in dokumen.items():
            jd[index][kunci] = nilai * nilai
            total += jd[index][kunci]
        hasil.append(sqrt(total))

    return hasil
```

Gambar 4.18 Kode *vector document*

Pada gambar 4.18 adalah kode program untuk menghitung vector document, dimana fungsi jarakdokumen() menerima 1 paramter yaitu hasil perhitungan dari term weighting. Masing masing bobot *term* pada setiap dokumen dikuadratkan kemudian dijumlah, sehingga akan menghasilkan *vector document*. hal ini dilakukan berulang sebanyak dokumen $_{i}$ .

Berikut ini adalah hasil perhitungan *vector document* setelah melalui beberapa tahap *processing* sebelumnya:

Tabel 4.11 Hasil *vector document*

| **dokumen** | **Vector document** |
|---|---|
| d1 | 1.2435901176393933 |
| d2 | 1.4537332102572953 |
| d3 | 2.7442404755431613 |
| d4 | 2.161965078040254 |
| d5 | 1.7263284121666784 |
| d6 | 1.8434723884579152 |
| d7 | 1.6383527549861756 |

### 1.1.3 Implementasi *Cosine Similarity*

Implementasi Cosine similarity digunakan untuk menghitung kemiripan antara query dan dokumen, penulis membagi menjadi 4 tahap dalam menghitung cosine similarity, yaitu sum bobot, menghitung semua vector document, menghitung similaritas kemudian ranking berdasarkan nilai bobot tertinggi.

Pada studi kasus ini, penulis menggunakan pertanyaan “Siapa ketua program studi Teknik elektro ?” sebagai query. Setelah query melalui tahap processing diatas, maka dapat dihasilkan beberapa hal sebagai berikut :

Tabel 4.12 Hasil *processing query*

| | |
|---|---|
| ***Query*** | Siapa ketua program studi Teknik elektro ? |
| ***Case folding*** | siapa ketua program studi teknik elektro ? |
| ***Tokenizing*** | 'siapa', 'ketua', 'program', 'studi', 'teknik', 'elektro', '?' |
| ***Stopword Removal*** | 'ketua', 'program', 'studi', 'teknik', 'elektro', '?' |
| ***Stemming*** | 'ketua', 'program', 'studi', 'teknik', 'elektro' |
| ***Term frequency*** | 'dekan': 0, 'fakultas': 0, 'teknik': 1, 'ir': 0, 'haryono': 0, 'wakil': 0, 'wadek': 0, 'ahmad': 0, 'izzuddin': 0, 'kom': 0, 'ii': 0, 'ira': 0, 'aprilia': 0, 'pd': 0, 'iii': 0, 'tri': 0, 'prihatingsih': 0, 'ketua': 1, 'program': 1, 'studi': 1, 'kaprodi': 0, 'elektro': 1, 'nuzul': 0, 'hikmah': 0, 'industri': 0, 'yustina': 0, 'suhandini': 0, 'tjahjaningsih': 0, 'mesin': 0, 'djoko': 0, 'wahyudi': 0 |
| ***Term weighting*** | 'dekan': 0.0, 'fakultas': 0.0, 'teknik': 0.0, 'ir': 0.0, 'haryono': 0.0, 'wakil': 0.0, 'wadek': 0.0, 'ahmad': 0.0, 'izzuddin': 0.0, 'kom': 0.0, 'ii': 0.0, 'ira': 0.0, 'aprilia': 0.0, 'pd': 0.0, 'iii': 0.0, 'tri': 0.0, 'prihatingsih': 0.0, 'ketua': 0.36797678529459443, 'program': 0.36797678529459443, 'studi': 0.36797678529459443, 'kaprodi': 0.0, 'elektro': 0.8450980400142568, 'nuzul': 0.0, 'hikmah': 0.0, 'industri': 0.0, 'yustina': 0.0, 'suhandini': 0.0, 'tjahjaningsih': 0.0, 'mesin': 0.0, 'djoko': 0.0, 'wahyudi': 0.0 |
| ***Vector Query*** | 1.0584948940751535 |

Dari data query diatas, maka perhitungan similarity dapat dilakukan. Berikut adalah tahapan dalam menghitung Cosine similarity :

1. Sum Bobot

```
def SUMBOBOT(w, wq):
    sb = []
    hasil = []

    for index, dokumen in enumerate(w):
        sb.append({})
        total = 0
        for kunci, nilai in dokumen.items():
            sb[index][kunci] = nilai * wq[0][kunci]
            total += sb[index][kunci]
        hasil.append(total)

    return hasil
```

Gambar 4.19 Kode sum bobot

Pada gambar 4.19 adalah kode program untuk menghitung jumlah bobot *query* dengan bobot dokumen, pada fungsi *sumbobot()* menerima parameter dari masing masing bobot query dan bobot setiap dokumen. Perhitungan dilakukan dengan cara melakukan perkalian antara nilai bobot setiap dokumen dengan nilai bobot *query*, kemudian hasil dari perhitungan tersebut disimpan kedalam sebuah array, Perhitungan ini dilakukan sebanyak dokumen yang sudah melewati tahap *filter questions.*

Berikut adalah hasil perhitungan dari sumbobot :

Tabel 4.13 Hasil sum bobot

| **Dokumen** | **Hasil sum bobot** |
|---|---|
| d1 | 0.0 |
| d2 | 0.0 |
| d3 | 0.0 |
| d4 | 0.0 |
| d5 | 1.1204114407831707 |

| Dokumen | Hasil sum bobot |
|---|---|
| d6 | 0.40622074354723214 |
| d7 | 0.40622074354723214 |

2. Hitung vector document *dan* vector query

```
def HITUNGSEMUAJARAK(jd, jq):
    hsj = []

    for index in range(len(jd)):
        hsj.append(jd[index] * jq[0])

    return hsj
```

Gambar 4.20 Kode hitung *vector document* dan *vector query*

Pada gambar 4.20 adalah kode program untuk menghitung *vector document* dan *vector query* dengan perkalian *matrix*. Dimana setiap *vector document* dari masing masing dokumen dikali dengan *vector query*. Hasil dari perkalian tersebut disimpan kedalam array.

Berikut ini adalah hasil dari perhitungan perkalian matrix *vector document* $_i$ dengan *vector query*.

Tabel 4.14 Hasil perkalian matrix *vector document* $_i$ dengan *vector query*

| **Dokumen** | **Hasil perkalian matrix *vector document* $_i$ dengan *vector query*** |
|---|---|
| **d1** | 1.3163337898436174 |
| **d2** | 1.5387691804048287 |
| **d3** | 2.9047645314768076 |
| **d4** | 2.2884289962744 |
| **d5** | 1.8273098097752962 |
| **d6** | 1.9513061105512313 |
| **d7** | 1.7341880258468279 |

3. *Similarity*

```
def SIMILARITY(sb, hsj):

    sim = []

    for index in range(len(sb)):
        sim.append(sb[index] / hsj[index])

    return sim
```

Gambar 4.21 Kode hitung *similarity*

Pada gambar 4.21 adalah kode program untuk menghitung *similarity* (kemiripan) antara dokumen dengan *query*. Perhitungan ini dilakukan dengan cara membagi nilai sumbobot ke $_{i}$ dengan nilai *vector document* dan *vector query* ke $_{i.}$ hasil dari perhitungan tersebut disimpan kedalam array.

Berikut ini adalah hasil perhitungan *similarity* (kemiripan) antara dokumen dengan *query* setelah melalui tahap *processing* sebelumnya.

Tabel 4.15 Hasil *similarity*

| **Dokumen** | **Hasil *similarity*** |
|---|---|
| **d1** | 0.0 |
| **d2** | 0.0 |
| **d3** | 0.0 |
| **d4** | 0.0 |
| **d5** | 0.613148046811475 |
| **d6** | 0.20817889174368312 |
| **d7** | 0.23424261815489641 |

Data pada table 4.15 merupakan hasil perhitungan mengunakan metode *cosine similarity* sebelum dilakukan tahap ranking berdasarkan nilai terbesar.

4. *Ranking* dokumen

```
def SORTIR(sim, dokumen):

    kunci = []
    for index in range(len(dokumen)):
        kunci.append(dokumen[index])

    nilai = []
    for index in range(len(dokumen)):
        nilai.append(sim[index])

    sortir = []
    sortir.append(
        dict(sorted(zip(kunci, nilai),  key=lambda item: item[1], reverse=True)))
    sortir = sortir[0]

    hasil = []
    for kunci, nilai in sortir.items():
        hasil.append(kunci)

    return hasil[0]
```

Gambar 4.22 Kode Hitung similarity

Pada gambar 4.22 adalah kode program untuk ranking dokumen berdasarkan hasil perhitungan *similarity*. Proses ranking dilakukan dengan tahapan-tahapan sebagai berikut :

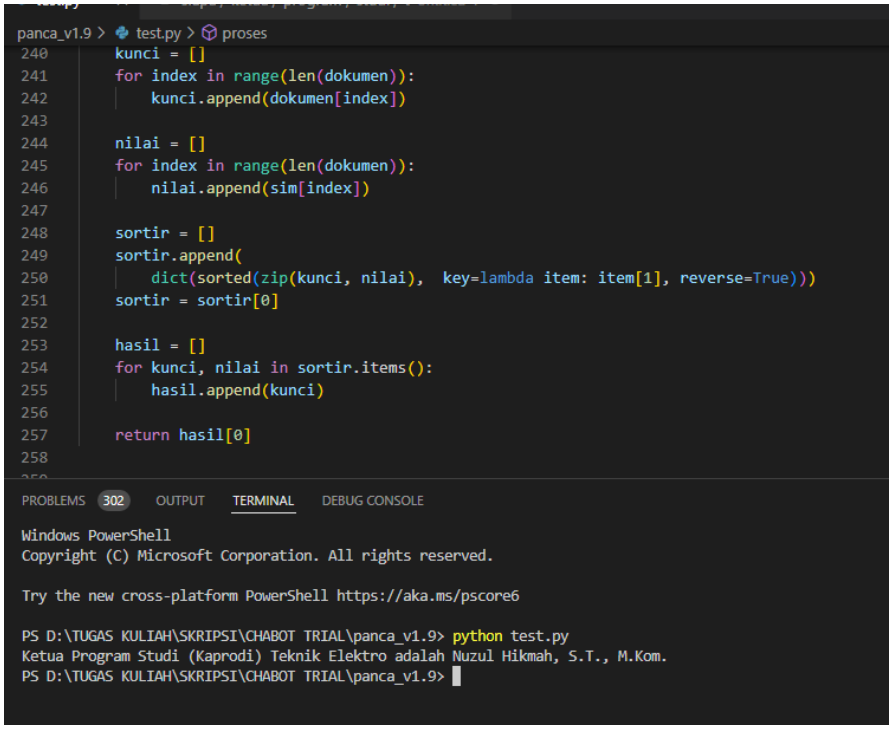
1. Pertama, menyimpan semua dokumen yang sudah melalui tahap *filter questions* kedalam sebuah array
2. Kedua, menyimpan semua nilai *similarity* dari masing masing dokumen kedalam sebuah array
3. Ketiga, gabungkan antara dokumen ke $_{i}$ dengan nilai similarity ke $_{i}$ menggunakan fungsi zip(). Kemudian sortir berdasarkan value dari nilai terbesar ke nilai terkecil menggunakan fungsi sorted().
4. Keempat, pecah hasil sortir antara key dan value, kemudian ambil value dari setiap dokumen dan simpan kedalam array.
5. Kelima, kembalikan nilai atau value dari tahap keempat dengan indeks ke 0,yaitu indeks dengan nilai similarity terbesar.

Maka dari hasil ranking dokumen dapat diurutkan menjadi sebagai berikut :

Tabel 4.23 Hasil *ranking*

| **dokumen** | **Hasil *ranking*** |
|---|---|
| **d5** | 0.613148046811475 |
| **d7** | 0.23424261815489641 |
| **d6** | 0.20817889174368312 |
| **d1** | 0.0 |
| **d2** | 0.0 |
| **d3** | 0.0 |
| **d4** | 0.0 |

Dari semua hasil perhitungan diatas dapat disimpulkan bahwa dokumen ke-6 memiliki tingkat *similarity* paling tinggi terhadap *query*, Dengan nilai 0.613148046811475. maka dokumen yang menjadi respon terhadap *query* adalah "Ketua Program Studi (Kaprodi) Teknik Elektro adalah Nuzul Hikmah, S.T., M.Kom.".




Gambar 4.23 Hasil respon terhadap *query* dari sistem

### 1.1.4 Implementasi simpan *history* chat

Impelementasi penyimpan histori chat berfungsi sebagai rekam jejak percakapan antara chatbot dengan pengguna. Dari rekam jejak tersebut, terdapat data nomor pengguna, pesan dari pengguna, respon dari chatbot serta waktu timestamp dari percakapan.

Data tersebut dapat digunakan sebagai acuan untuk memperbaiki database dari pertanyaan pertanyaan yang tidak dapat dijawab oleh chatbot karena respon dari pertanyaan tersebut tidak terdapat didalam database. Hal ini juga sebagai langkah maintenance agar chatbot jauh lebih cerdas kedepannya.

```
historys = {
    "phonenumber": phonenumber,
    "message": message,
    "respon": proses(),
    "date": date
}

fname = 'historys.text'
a = []

if not os.path.isfile(fname):
    a.append(historys)
    with open(fname, mode='w') as f:
        f.write(json.dumps(a, indent=4))
else:
    with open(fname) as feedsjson:
        feeds = json.load(feedsjson)

    feeds.append(historys)
    with open(fname, mode='w') as f:
        f.write(json.dumps(feeds, indent=4))
```

Gambar 4.24 Kode menyimpan data pesan

Pada gambar 4.24 adalah kode program untuk menyimpan data pesan dari pengguna, dimana langkah pertama melihat sebuah file txt apakah ada atau tidak dalam sebuah system. apabila “tidak”, maka buatkan file txt dengan nama *historys* yang disimpan didalam variable fname. Jika iya (file ada), maka buka file txt lalu tambahkan data yang tersimpan didalam *variable historys* kedalam file historys.txt.

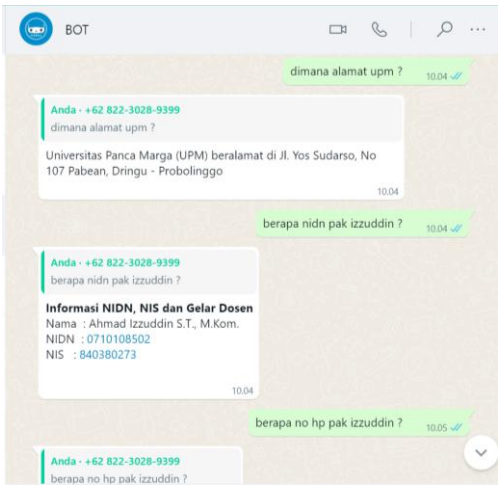
## 1.2 Pengujian

Pengujian sistem dilakukan setelah tahap imlementasi, pada tahap ini akan dilakukan uji kelayakan terhadap aplikasi yang telah dibuat. Pengujian sistem dilakukan dengan beberapa scenario pengujian, pengujian yang dilakukan oleh penulis diantaranya yaitu metode *black box testing*, mengukur tingkat akurasi, dan *user acceptance test.*

Pada saat pertama kali user mengirim pesan kepada *chatbot*, *chatbot* akan memperkenalkan diri dan memberikan sedikit *rule* bagaimana cara berinteraksi dengan *chatbot*, hal ini dapat dilihat pada gambar 4.25. perkenalan ini berguna untuk membuat pengguna lebih memahami cara penggunaan *chatbot*.

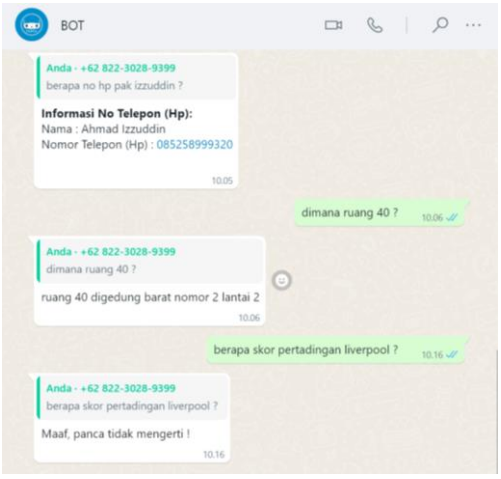

Gambar 4.25 salam perkenalan *chatbot*

Pada gambar 4.25 menunjukan percakapan antara pengguna dengan *chatbot*, dimana *chatbot* mengirimkan pesan *default* untuk memperkenalkan diri kepada pengguna, Hal ini bisa terjadi karena pengguna baru pertama kali menggunakan *chatbot.*




Gambar 4.26 percakapan pengguna dengan chatbot

Pada gambar 4.26 adalah beberapa percakapan pengguna dengan *chatbot*, dimana pengguna mengajukan bebeberapa pertanyaan kepada *chatbot*. *respon time* dari percakapan tersebut tidak sampai 1 (satu) menit, hal ini menunjukan bahwa *chatbot* sangat *responsive* dalam merespon pengguna.




Gambar 4.27 respon tidak menemukan jawaban

Gambar 4.27 menunjukan *respon default* bahwa *chatbot* tidak mengerti pertanyaan dari pengguna, Hal ini terjadi karena perhitungan antara *query* dan dokumen yang ada didalam database tidak menemukan kemiripan, maka secara otomatis *chatbot* akan mengirimkan respon default "Maaf, panca tidak mengerti!" untuk menunjukan bahwa *query* atau pertanyaan dari pengguna tidak ditemukan atau tidak memiliki topik yang dianggap mirip dengan dokumen yang terdapat didalam database.

### 1.2.1 *Black Box Testing*

1. Tabel Pengujian Scan QR Code

Tabel 4.24 Pengujian Scan QR Code

| Interface | Yang diuji | Kondisi | Output | Status |
|---|---|---|---|---|
| Halaman index | menampilkan qr code | Belum discan | Qr code ditampilkan dibrowser | Benar |
| Halaman index | Scan qr code | Sudah discan | QR Code hilang | Benar |

2. Tabel Pengujian Percakapan dengan Chatbot

Tabel 4.25 Pengujian Percakapan dengan Chatbot

| Interface | Yang diuji | Kondisi | Output | Status |
|---|---|---|---|---|
| Aplikasi Whatsapp | Kirim pesan | Seputar akademik | Respon sesuai perhitungan | Benar |
| Aplikasi Whatsapp | Kirim pesan | Diluar seputar akademik | Respon default "Maaf, panca tidak mengerti ! Coba lagi ya dengan bahasa yang lebih baku agar panca mengerti" | Benar |

3. Tabel Pengujian Simpan Pesan

Tabel 4.26 Pengujian Simpan Pesan

| Interface | Yang diuji | Kondisi | Output | Status |
|---|---|---|---|---|
| - | menyimpan pesan | - | Pesan tersimpan didalam file txt | Benar |

### 1.2.2 *Akurasi*

1. Tabel Pengujian akurasi

Tabel 4.27 Pengujian akurasi

| Tipe | Benar | Salah | Jumlah |
|---|---|---|---|
| **Apa** | 4 | 1 | 5 |
| **Dimana** | 19 | 1 | 20 |
| **Kapan** | 3 | 2 | 5 |
| **Kenapa** | 4 | 1 | 5 |
| **Siapa** | 18 | 2 | 20 |
| **Bagaimana** | 4 | 1 | 5 |
| **Berapa** | 8 | 2 | 10 |
| **Total** | **60** | **10** | **70** |

Sehingga dari hasil pengujian diatas dapat dihitung menggunakan rumus akurasi sebagai berikut :

$$\text{Akurasi} = \frac{\text{Jumlah jawaban benar}}{\text{Jumlah total pertanyaan}} \text{ X } 100\ \%$$

$$\text{Akurasi} = \frac{60}{70} \text{ X } 100\ \%$$

Akurasi = **85,7%**

### 1.2.3 *User Acceptance Test*

Pengujian *User Acceptance Test* merupakan pengujian yang melibatkan pengguna, dimana pengguna yang dimaksud adalah mahasiswa universitas panca marga probolinggo dan masyarakat umum khususnya masyarakat dengan status pelajar SMA.

Pengujian dengan melibatkan mahasiswa universitas panca marga probolinggo khususnya mahasiswa baru, mahasiswa baru dianggap sebagai responden paling relevan karena mahasiswa baru akan memiliki problem atau banyak pertanyaan seputar akademik, hal ini dikarenakan proses transisi antara masa SMA ke masa perkuliahan. Dimana sistem dari masa sekolah dengan masa kuliah sangat berbeda.

Untuk melihat seberapa efesien dan efektive aplikasi chatbot yang sudah dikembangkan maka peneliti juga melibatkan masyarakat umum dalam pengujian UAT. Seperti yang telah dijelaskan sebelumnya, masyarakat umum disini adalah masyarakat dengan status sebagai pelajar SMA (Sekolah Menengah Atas), diambilnya responden dari pelajar SMA karena siswa / siswi SMA ini merupakan responden kedua paling dekat setelah Mahasiswa baru yang ada diuniversitas panca marga probolinggo.

1. Pengujian Pertama

Berikut hasil pengujian *user acceptance test* dengan melakukan survei menggunakan kuesioner. Kuesioner ini ditujukan kepada 30 pengguna (Mahasiswa Universitas Panca Marga Probolinggo dan Masyarakat Umum).

Tabel 4.28 menunjukkan beberapa rencana pengujian yang dijalankan penulis saat menguji menggunakan *user acceptance test*.

Tabel 4.28 rencana pengujian *user acceptance test*

| **No** | **Jenis Pengujian** | | | | | | |
|---|---|---|---|---|---|---|---|
| **1** | **Pengujian Kinerja** | | | | | | |
| | No | Pertanyaan | SS | S | RR | TS | STS |
| | 1 | Apakah anda setuju *respon time* (waktu respon) chatbot cukup cepat ? | √ | | | | |
| | 2 | Apakah anda setuju respon yang diberikan *chatbot* sesuai dengan pertanyaan yang anda ajukan ? | | √ | | | |
| | 3 | Apakah anda setuju *chatbot* dapat memperkenalkan diri dengan baik sehingga pengguna mengerti dan memahami cara penggunaan dari *chatbot* ? | | | √ | | |
| **2** | **Pengujian Kepuasan Pengguna** | | | | | | |
| | 1 | Apakah anda setuju *chatbot* dapat membantu anda mendapatkan informasi seputar akademik ? | | √ | | | |
| | 2 | Apakah anda setuju *chatbot* ini mempermudah dan mempercepat anda mendapatkan informasi seputar akademik universitas panca marga probolinggo ? | | √ | | | |
| | 3 | Apakah anda setuju *chatbot* dapat menjadi prioritas utama anda sebagai *virtual assistant* untuk mendapatkan informasi seputar akademik universitas panca marga probolinggo ? | | | √ | | |

Data Tabel 4.28 merupakan salah satu hasil dari kuesioner untuk pengguna (Mahasiswa di Universitas Panca Marga Probolinggo dan masyarakat umum).

Setelah melakukan survei dengan menyebar kuesioner kepada pengguna menggunakan *google form* (lampiran), maka didapatkan data sebagai berikut :

Tabel 4.29 Hasil semua kuesioner

| **No** | **Jenis Pengujian** | | | | | | |
|---|---|---|---|---|---|---|---|
| **1** | **Pengujian Kinerja** | | | | | | |
| | No | Pertanyaan | SS | S | RR | TS | STS |
| | 1 | Apakah anda setuju *respon time* (waktu respon) chatbot cukup cepat ? | 14 | 16 | - | - | - |
| | 2 | Apakah anda setuju respon yang diberikan *chatbot* sesuai dengan pertanyaan yang anda ajukan ? | 6 | 20 | 4 | - | - |
| | 3 | Apakah anda setuju *chatbot* dapat memperkenalkan diri dengan baik sehingga pengguna mengerti dan memahami cara penggunaan dari *chatbot* ? | 10 | 15 | 4 | 1 | - |
| **2** | **Pengujian Kepuasan Pengguna** | | | | | | |
| | 1 | Apakah anda setuju *chatbot* dapat membantu anda mendapatkan informasi seputar akademik ? | 12 | 17 | 1 | - | - |
| | 2 | Apakah anda setuju *chatbot* ini mempermudah dan mempercepat anda mendapatkan informasi seputar akademik universitas panca marga probolinggo ? | 10 | 19 | - | 1 | - |
| | 3 | Apakah anda setuju *chatbot* dapat menjadi prioritas utama anda sebagai *virtual assistant* untuk mendapatkan informasi seputar akademik universitas panca marga probolinggo ? | 6 | 17 | 6 | 1 | - |

Tabel 2.29 adalah hasil rekap jawaban kuesioner (lampiran) tentang kinerja dan kepuasan pengguna.

Berdasarkan rekap jawaban kuesioner (lampiran) pada table 2.29, berikut merupakan hasil perhitungan bobot atau presentase dari pengujian *user acceptance test* yang dijelaskan pada Tabel 2.30.

Tabel 2.30 Hasil perhitungan UAT

| Pertanyaan | Jawaban Responden | | | | | Total Responden |
|---|---|---|---|---|---|---|
| | **SS** | **S** | **RR** | **TS** | **STS** | |
| **P1** | 14 | 16 | 0 | 0 | 0 | 30 |
| **P2** | 6 | 20 | 4 | 0 | 0 | |
| **P3** | 10 | 15 | 4 | 1 | 0 | |
| **P4** | 12 | 17 | 1 | 0 | 0 | |
| **P5** | 10 | 19 | 0 | 1 | 0 | |
| **P6** | 6 | 17 | 6 | 1 | 0 | |
| **Jumlah** | **58** | **104** | **15** | **3** | **0** | |

Dari data tersebut, penulis menganalisisnya dengan menghitung rata-rata tanggapan berdasarkan skor yang diperoleh dari masing-masing tanggapan responden.

Berdasarkan data yang terdapat pada tabel 2.30 dapat dihitung sebagai mana berikut :

- Jumlah responden yang menjawab SS = 58 x 5 = 290
- Jumlah responden yang menjawab S = 104 x 4 = 416
- Jumlah responden yang menjawab RR = 15 x 3 = 45
- Jumlah responden yang menjawab TS = 3 x 2 = 6
- Jumlah responden yang menjawab STS = 0 x 1 = 0

**Total Nilai = 757**

Presentase menjawab SS : 290 / 757 x 100% = 38.3 %

Presentase menjawab S : 416 / 757 x 100% = 54.9 %

Presentase menjawab RR : 45 / 757 x 100% = 5.9 %

Presentase menjawab TS : 6 / 757 x 100% = 0.7 %

Presentase menjawab STS : 0 / 757 x 100% = 0

Hasil tanggapan dari 30 responden dapat dihitung nilai tertinggi dan terendah sebagai berikut :

Nilai Tertinggi = 30 x 6 x 5 = **900** (apabila semua menjawab SS).

Nilai Terendah = 30 x 6 x 1 = **180** (apabila semua menjawab SS).

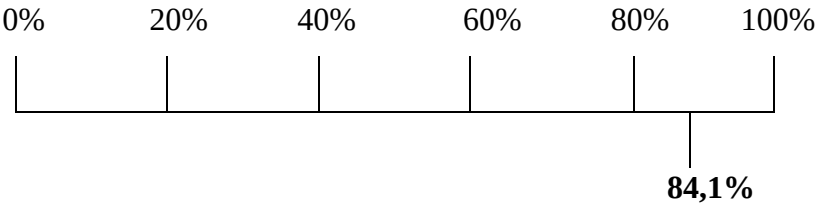
Berdasarkan perhitungan yang menyatakan nilai tertinggi adalah 900, dapat dicari presentase sebagaimana berikut :

Presentase = total nilai / nilai tertinggi x 100%

= 757 / 900 x 100%

= **84,1%**

Berdasarkan *user acceptance test*, dapat dilihat bahwa kinerja chatbot dan kepuasan pengguna sangat tinggi, apabila dilihat berdasarkan gambar 4.25 maka hasil dari pengujian tergolong sangat kuat.




Gambar 4.25 Tingkat *usability* sistem

Keterangan :

0 – 20 % = Sangat Lemah

20% - 40% = Lemah

41% - 60% = Cukup

61% - 80% = Kuat

**80% - 100% = Sangat Kuat**

Dari hasil presentase diatas yaitu **84,1%** *chatbot* ini tergolong sangat kuat dari penilaian pengguna.

2. Pengujian Kedua

Berikut hasil pengujian *user acceptance test* tahap kedua dengan melakukan survei menggunakan kuesioner. Kuesioner ini ditujukan kepada 92 pengguna (Mahasiswa Universitas Panca Marga Probolinggo dan Masyarakat Umum).

Tabel 4.31 menunjukkan beberapa rencana pengujian yang dijalankan penulis saat menguji menggunakan user acceptance test.

Tabel 4.31 rencana pengujian *user acceptance test*

| **No** | **Jenis Pengujian** | | | | | | |
|---|---|---|---|---|---|---|---|
| **1** | **Pengujian Kinerja** | | | | | | |
| | No | Pertanyaan | SS | S | RR | TS | STS |
| | 1 | Apakah anda setuju *respon time* (waktu respon) chatbot cukup cepat ? | √ | | | | |
| | 2 | Apakah anda setuju respon yang diberikan *chatbot* sesuai dengan pertanyaan yang anda ajukan ? | | √ | | | |
| | 3 | Apakah anda setuju *chatbot* dapat memperkenalkan diri dengan baik sehingga pengguna mengerti dan memahami cara penggunaan dari *chatbot* ? | | √ | | | |
| | 4 | Apakah anda setuju chatbot dapat berjalan dengan baik ? | | √ | | | |
| **2** | **Pengujian Kepuasan Pengguna** | | | | | | |
| | 1 | Apakah anda setuju *chatbot* dapat membantu anda mendapatkan informasi seputar akademik universitas panca marga probolinggo ? | √ | | | | |
| | 2 | Apakah anda setuju *chatbot* ini mempermudah dan mempercepat anda mendapatkan informasi seputar akademik universitas panca marga probolinggo ? | √ | | | | |
| | 3 | Apakah anda setuju *chatbot* dapat menjadi prioritas utama | | | √ | | |

| **No** | **Jenis Pengujian** | | | | | | |
|---|---|---|---|---|---|---|---|
| | **Pengujian Kinerja** | | | | | | |
| | No | Pertanyaan | SS | S | RR | TS | STS |
| **2** | | anda sebagai *virtual assistant* untuk mendapatkan informasi seputar akademik universitas panca marga probolinggo ? | | | | | |
| | 4 | Apakah anda setuju chatbot ini bermanfaat apabila digunakan sebagai asisten virtual ? | √ | | | | |

Data Tabel 4.31 merupakan salah satu hasil dari kuesioner untuk pengguna (Mahasiswa di Universitas Panca Marga Probolinggo dan masyarakat umum).

Setelah melakukan survei dengan menyebar kuesioner kepada pengguna menggunakan google form (lampiran), maka didapatkan data sebagai berikut :

Tabel 4.32 Hasil semua kuesioner Pengujian kedua

| **No** | **Jenis Pengujian** | | | | | | |
|---|---|---|---|---|---|---|---|
| **1** | **Pengujian Kinerja** | | | | | | |
| | No | Pertanyaan | SS | S | RR | TS | STS |
| | 1 | Apakah anda setuju *respon time* (waktu respon) chatbot cukup cepat ? | 27 | 59 | 5 | 1 | - |
| | 2 | Apakah anda setuju respon yang diberikan *chatbot* sesuai dengan pertanyaan yang anda ajukan ? | 15 | 57 | 16 | 4 | - |
| | 3 | Apakah anda setuju *chatbot* dapat memperkenalkan diri dengan baik sehingga pengguna mengerti dan memahami cara penggunaan dari *chatbot* ? | 21 | 50 | 18 | 3 | - |
| | 4 | Apakah anda setuju chatbot dapat berjalan dengan baik ? | 17 | 59 | 15 | 1 | - |
| **2** | **Pengujian Kepuasan Pengguna** | | | | | | |
| | 1 | Apakah anda setuju *chatbot* dapat membantu anda mendapatkan informasi seputar Akademik Universitas Panca Marga? | 7 | 68 | 17 | - | - |

| **No** | **Jenis Pengujian** | | | | | | |
|---|---|---|---|---|---|---|---|
| **2** | **Pengujian Kinerja** | | | | | | |
| | No | Pertanyaan | SS | S | RR | TS | STS |
| | 2 | Apakah anda setuju *chatbot* ini mempermudah dan mempercepat anda mendapatkan informasi seputar akademik universitas panca marga probolinggo ? | 74 | 9 | 9 | - | - |
| | 3 | Apakah anda setuju *chatbot* dapat menjadi prioritas utama anda sebagai *virtual assistant* untuk mendapatkan informasi seputar akademik universitas panca marga probolinggo ? | 5 | 46 | 26 | 15 | - |
| | 4 | Apakah anda setuju chatbot ini bermanfaat apabila digunakan sebagai asisten virtual ? | 74 | 10 | 8 | - | - |

Tabel 2.32 adalah hasil rekap jawaban kuesioner (lampiran) tentang kinerja dan kepuasan pengguna.

Berdasarkan rekap jawaban kuesioner (lampiran) pada table 2.32, peneliti kelompokan menjadi 2 (dua) kategori pengujian, yaitu pengujian terhadap kinerja chatbot dan pengujian terhadap kepuasaan pengguna.

Berikut merupakan hasil perhitungan bobot atau presentase dari pengujian *user acceptance test* terhadap kinerja chatbot yang dijelaskan pada Tabel 2.33.

Tabel 2.33 Hasil perhitungan *UAT* pengujian kedua

| **Pertanyaan** | **Jawaban Responden** | | | | | **Total Responden** |
|---|---|---|---|---|---|---|
| | **SS** | **S** | **RR** | **TS** | **STS** | |
| **P1** | 27 | 59 | 5 | 1 | 0 | **92** |
| **P2** | 15 | 57 | 16 | 4 | 0 | |
| **P3** | 21 | 50 | 18 | 3 | 0 | |
| **P4** | 17 | 59 | 15 | 1 | 0 | |
| **P5** | 7 | 68 | 17 | 0 | 0 | |
| **P6** | 74 | 9 | 9 | 0 | 0 | |
| **P7** | 5 | 46 | 26 | 15 | 0 | |
| **P8** | 74 | 10 | 8 | 0 | 0 | |
| **Jumlah** | **240** | **358** | **114** | **24** | **0** | |

Dari data tersebut, penulis menganalisisnya dengan menghitung rata-rata tanggapan berdasarkan skor yang diperoleh dari masing-masing tanggapan responden. Berdasarkan data yang terdapat pada tabel 2.33 dapat dihitung sebagai mana berikut :

- Jumlah responden yang menjawab SS = 240 x 5 = 1200
- Jumlah responden yang menjawab S = 358 x 4 = 1432
- Jumlah responden yang menjawab RR = 114 x 3 = 342
- Jumlah responden yang menjawab TS = 24 x 2 = 48
- Jumlah responden yang menjawab STS = 0 x 1 = 0

---

**Total Nilai = 3022**

Presentase menjawab SS : 1200 / 3022 x 100% = 39,7 %

Presentase menjawab S : 1432 / 3022 x 100% = 47,3 %

Presentase menjawab RR : 342 / 3022 x 100% = 11,3 %

Presentase menjawab TS : 48 / 3022 x 100% = 1,5 %

Presentase menjawab STS : 0 / 3022 x 100% = 0

Hasil tanggapan dari 92 responden dapat dihitung nilai tertinggi dan terendah sebagai berikut :

Nilai Tertinggi = 92 x 8 x 5 = **3680** (apabila semua menjawab SS).

Nilai Terendah = 92 x 8 x 1 = **736** (apabila semua menjawab SS).

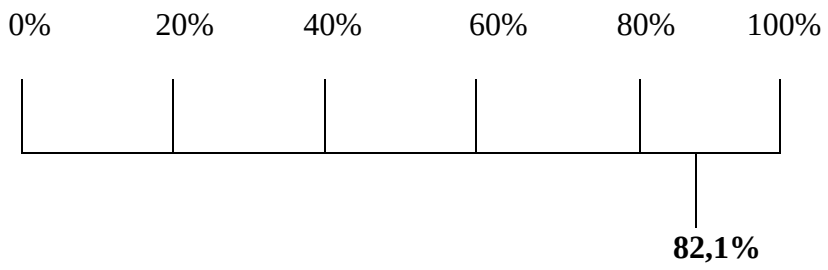
Berdasarkan perhitungan yang menyatakan nilai tertinggi adalah 3680, dapat dicari presentase sebagaimana berikut :

Presentase = Jumlah Skor total / Nilai tertinggi x 100%

= 3022 / 3680 x 100%

= **82,1%**

Berdasarkan user acceptance test, dapat dilihat bahwa kinerja chatbot dan kepuasan pengguna sangat tinggi, apabila dilihat berdasarkan gambar 4.25 maka hasil dari pengujian tergolong sangat kuat.




Gambar 4.26 Tingkat *usability* sistem

Keterangan :

0 – 20 % = Sangat Lemah

20% - 40% = Lemah

41% - 60% = Cukup

61% - 80% = Kuat

**80% - 100% = Sangat Kuat**

Dari hasil presentase diatas yaitu **82,1%** *chatbot* ini tergolong sangat kuat dari penilaian pengguna.