CREATE TABLE dosen(

dosen\_id int(11) NOT NULL AUTO\_INCREMENT PRIMARY KEY,

nama varchar(50) NOT NULL,

univ varchar(50) NOT NULL,

jurusan varchar(50) NOT NULL)

INSERT INTO dosen(nama, univ, jurusan) VALUES

('milia', 'TelU', 'sisfo'),

('agata', 'UI', 'tekdus');

app = Flask(\_\_name\_\_)

# Setup MongoDB connection

client = pymongo.MongoClient("mongodb://localhost:27017/")

db = client["TTS"]

collection = db["tts\_history"]

@app.route('/')

def index():

    return render\_template('convert.html')

@app.route('/history', methods=['GET'])

def history():

    history\_data = list(collection.find())

    return render\_template('history.html', history=history\_data)

@app.route('/convert', methods=['POST'])

def convert\_text\_to\_speech():

    text = request.form['text']

    lang = request.form['voice']

    timestamp = datetime.now().strftime("%Y-%m-%d %H:%M:%S")

    entry\_id = str(ObjectId())  # Generate a new ObjectId and convert it to string

    # Convert text to speech using gTTS with the selected language

    tts = gTTS(text=text, lang=lang)

    # Save audio data to MongoDB

    audio\_data = BytesIO()

    tts.write\_to\_fp(audio\_data)

    # Save data to MongoDB along with the generated ObjectId

    data = {

        '\_id': entry\_id,  # Save the generated ObjectId

        'text': text,

        'language': lang,  # Save selected language

        'timestamp': timestamp,

        'audio\_data': audio\_data.getvalue()  # Save audio data as binary

    }

    collection.insert\_one(data)

    # Render convert.html with audio\_path

    audio\_path = url\_for('play\_audio', entry\_id=entry\_id)

    return render\_template('convert.html', audio\_path=audio\_path)

@app.route('/audio/<entry\_id>')

def play\_audio(entry\_id):

    # Retrieve audio data from MongoDB

    data = collection.find\_one({'\_id': entry\_id})

    if data:

        audio\_data = data['audio\_data']

        return send\_file(BytesIO(audio\_data), mimetype='audio/mp3')

    else:

        return "Audio not found"

@app.route('/edit/<entry\_id>', methods=['GET'])

def edit\_entry(entry\_id):

    entry = collection.find\_one({'\_id': entry\_id})

    if entry:

        entry\_text = entry['text']

        selected\_voice = None

        return render\_template('edit.html', entry\_id=entry\_id, entry\_text=entry\_text, selected\_voice=selected\_voice)

    else:

        return "Entry not found", 404

@app.route('/edit/<entry\_id>', methods=['PUT'])

def update\_entry(entry\_id):

    lang = request.json['voice']  # Get selected voice from the request JSON

    # Retrieve text from the database

    entry = collection.find\_one({'\_id': entry\_id})

    text = entry['text']

    # Convert text to speech using gTTS with the selected language

    tts = gTTS(text=text, lang=lang)

    # Save audio data to MongoDB

    audio\_data = BytesIO()

    tts.write\_to\_fp(audio\_data)

    # Update data in MongoDB

    collection.update\_one({'\_id': entry\_id}, {'$set': {'audio\_data': audio\_data.getvalue(), 'language': lang}})

    return jsonify({'message': 'Entry updated successfully'})

@app.route('/search', methods=['GET'])

def search():

    search\_query = request.args.get('search\_query', '')

    # Cari entri yang cocok dengan query

    search\_results = list(collection.find({'text': {'$regex': search\_query, '$options': 'i'}}))

    return render\_template('search.html', search\_query=search\_query, search\_results=search\_results)

@app.route('/delete/<entry\_id>', methods=['POST'])

def delete\_entry(entry\_id):

    # Delete entry from MongoDB

    collection.delete\_one({'\_id': entry\_id})

    return redirect(url\_for('history'))

if \_\_name\_\_ == "\_\_main\_\_":

    app.run(host='0.0.0.0', port='2363', debug=True)

Tentu, mari kita jelaskan kode di setiap rute aplikasi:

1. \*\*Rute `/` (index)\*\*:

```python

@app.route('/')

def index():

return render\_template('convert.html')

```

- Saat pengguna mengakses rute utama, fungsi `index()` akan dipanggil.

- Fungsi ini merender template `convert.html`, yang berisi formulir untuk memasukkan teks yang akan dikonversi menjadi ucapan.

2. \*\*Rute `/history`\*\*:

```python

@app.route('/history', methods=['GET'])

def history():

history\_data = list(collection.find())

return render\_template('history.html', history=history\_data)

```

- Saat pengguna mengakses rute `/history`, fungsi `history()` akan dipanggil.

- Fungsi ini akan mengambil semua data riwayat konversi dari koleksi MongoDB dan merender template `history.html`, yang menampilkan daftar riwayat konversi.

3. \*\*Rute `/convert`\*\*:

```python

@app.route('/convert', methods=['POST'])

def convert\_text\_to\_speech():

text = request.form['text']

lang = request.form['voice']

timestamp = datetime.now().strftime("%Y-%m-%d %H:%M:%S")

entry\_id = str(ObjectId())

tts = gTTS(text=text, lang=lang)

audio\_data = BytesIO()

tts.write\_to\_fp(audio\_data)

data = {

'\_id': entry\_id,

'text': text,

'language': lang,

'timestamp': timestamp,

'audio\_data': audio\_data.getvalue()

}

collection.insert\_one(data)

audio\_path = url\_for('play\_audio', entry\_id=entry\_id)

return render\_template('convert.html', audio\_path=audio\_path)

```

- Saat pengguna mengirimkan formulir konversi, fungsi `convert\_text\_to\_speech()` akan dipanggil.

- Fungsi ini mengambil teks yang dikirim oleh pengguna, mengonversinya menjadi ucapan menggunakan gTTS, dan menyimpan hasil konversi ke dalam database MongoDB.

- Kemudian, fungsi merender kembali template `convert.html` dengan menambahkan jalur file audio hasil konversi untuk diputar oleh pengguna.

4. \*\*Rute `/audio/<entry\_id>`\*\*:

```python

@app.route('/audio/<entry\_id>')

def play\_audio(entry\_id):

data = collection.find\_one({'\_id': entry\_id})

if data:

audio\_data = data['audio\_data']

return send\_file(BytesIO(audio\_data), mimetype='audio/mp3')

else:

return "Audio not found"

```

- Saat pengguna memutar audio, fungsi `play\_audio()` akan dipanggil dengan ID entri sebagai parameter.

- Fungsi ini akan mencari data audio dalam database berdasarkan ID entri, dan mengirimkan file audio ke klien menggunakan `send\_file`.

5. \*\*Rute `/edit/<entry\_id>`\*\*:

```python

@app.route('/edit/<entry\_id>', methods=['GET'])

def edit\_entry(entry\_id):

entry = collection.find\_one({'\_id': entry\_id})

if entry:

entry\_text = entry['text']

selected\_voice = None

return render\_template('edit.html', entry\_id=entry\_id, entry\_text=entry\_text, selected\_voice=selected\_voice)

else:

return "Entry not found", 404

```

- Saat pengguna mengakses rute `/edit/<entry\_id>` dengan metode GET, fungsi `edit\_entry()` akan dipanggil.

- Fungsi ini akan mencari entri dalam database berdasarkan ID entri yang diberikan dan merender template `edit.html` dengan teks entri yang akan diedit.

6. \*\*Rute `/edit/<entry\_id>` (PUT)\*\*:

```python

@app.route('/edit/<entry\_id>', methods=['PUT'])

def update\_entry(entry\_id):

lang = request.json['voice']

entry = collection.find\_one({'\_id': entry\_id})

text = entry['text']

tts = gTTS(text=text, lang=lang)

audio\_data = BytesIO()

tts.write\_to\_fp(audio\_data)

collection.update\_one({'\_id': entry\_id}, {'$set': {'audio\_data': audio\_data.getvalue(), 'language': lang}})

return jsonify({'message': 'Entry updated successfully'})

```

- Saat pengguna mengirimkan permintaan PUT ke rute `/edit/<entry\_id>`, fungsi `update\_entry()` akan dipanggil.

- Fungsi ini akan mengambil data bahasa yang dikirim oleh pengguna dari permintaan JSON, mengonversi teks ke ucapan dengan bahasa yang baru, dan memperbarui entri dalam database MongoDB dengan audio baru dan bahasa yang baru.

Selain itu, terdapat juga rute untuk pencarian dan penghapusan data yang dijelaskan dengan