

Dokumentasi Pembangunan Aplikasi Pendeteksi Gambar yang Telah Melewati Perombakan Digital dengan Menggunakan Flask dan Connexion

A. Pendahuluan

Pada dokumen ini, akan dijelaskan teknologi-teknologi yang digunakan dalam pembangunan sebuah *Application Programming Interface* (API) untuk memberikan service pendeteksian gambar yang sudah melalui proses perombakan digital (*tampered image*). Sistem ini dibangun dengan menggunakan Flask untuk pembangunan *Application Programming Interface*, serta menggunakan Connexion untuk membuat dokumentasi dari API tersebut, dimana akan memanfaatkan Swagger UI dan OpenAPI3.

Repository yang digunakan untuk proyek ini terdapat di <https://github.com/jamesvincentsiau/tampered-image-detector>. Untuk setup dan run aplikasi dapat membaca langkah-langkahnya pada file README.md. Untuk dokumentasi lengkap penggunaan Flask dapat dilihat pada link <https://pypi.org/project/Flask/> dan dokumentasi lengkap penggunaan Connexion dapat dilihat pada link <https://connexion.readthedocs.io/en/latest/index.html>.

B. Requirements

Untuk menjalankan program ini, dibutuhkan beberapa *requirements* yang harus dipenuhi, tetapi terdapat 2 metode untuk menjalankannya, yaitu dengan menggunakan Docker, atau dengan menggunakan Python.

Untuk metode pertama, yang perlu dilakukan hanyalah Docker yang dapat di-install melalui link <https://www.docker.com/>. Setelah itu, kita hanya perlu mengikuti panduan yang ada di file README.md yang terdapat pada repository.

Metode kedua adalah dengan menggunakan Python dengan versi 3.4.x yang dapat di-install melalui link <https://www.python.org/>. Setelah itu, kita perlu meng-install *library-library* yang dibutuhkan yang ada pada file requirements.txt pada repository.

C. Implementasi

Pada implementasinya, *library* Connexion ini adalah *library* untuk pengintegrasian antara aplikasi Flask itu sendiri dengan dokumentasi API Swagger. Jika pada saat menggunakan *library* Flask seluruhnya, kita mendefinisikan sebuah file utama sebagai *main app*, dan di-run dengan definisi

```
app = Flask(__name__)
```

Namun, saat memanfaatkan *library* Connexion kita akan melihat *main program* kita sebagai sebuah modul, dan *main program* nya adalah Swagger. Lengkapnya adalah

sebagai berikut.

Saya memiliki method `tampered_image_processing()` yang terdapat pada file `app.py` dimana `app.py` ini dianggap sebagai sebuah modul dengan nama `app` dan memiliki method `tampered_image_processing()`.

```
# Main Method. Will be called by Connexion and Connected with Swagger
def tampered_image_processing():
    try:
        requested_model = request.form['model']

        if choose_model(requested_model) == "error":
            val = {
                'status': "error",
                'message': 'Model Not Found!'
            }
            return jsonify(val), 400

        # Load the model
        model = load_model(choose_model(requested_model))

        # Process file
        if not request.files['img']:
            return jsonify({
                'message': "Bad Parameter! Please upload file",
                'status': "error"
            }), 400

        filepath = files_handler(request.files['img'])
        if filepath == "error":
            val = {
                'status': "error",
                'message': 'Bad Parameter, Check File Extension!'
            }
            return jsonify(val), 400

        img = load(filepath)

        result = process_prediction(model, img)
        return jsonify(result), 200
```

Kemudian, pada dokumentasi API yang terdapat pada file `swagger.yaml` saya tinggal memanggil method `tampered_image_processing()` yang ada pada modul `app` pada path yang terkait dengan method tersebut pada atribut `operationID` pada `swagger.yaml`. Contohnya adalah sebagai berikut.

```

paths:
  /predictor:
    post:
      tags:
        - Main
      summary: Predict Images
      operationId: app.tampered_image_processing
      description: |
        By passing the chosen model and the image file, you can get prediction whether it is real or tampered
      requestBody:
        content:
          multipart/form-data:
            schema:
              type: object
              required:
                - img
                - model
              properties:
                img:
                  type: string
                  format: binary
                  description: Please Upload .jpeg/.jpg/.png/.tif File
                model:
                  type: string
                  format: model
                  example: v2
                  description: choose v1/v2/v3

```

Setelah itu, kita perlu mendefinisikan aplikasi kita akan berjalan dengan menggunakan *library* Connexion dengan menggunakan kode seperti berikut ini.

```


if __name__ == "__main__":
    # Create the application instance
    app = connexion.App(__name__, specification_dir='openapi/')

    # Add a flask route to expose information
    app.add_url_rule("/api/predictor/health", "healthcheck", view_func=lambda: health.run())

    # Read the swagger.yml file to configure the endpoints
    app.add_api('swagger.yml')
    app.run(threaded=False)

```

Pada awalnya, *library* Connexion tidak akan menampilkan UI dari dokumentasi API Swagger, tetapi kita dapat membuatnya menampilkan UI dengan cara menjalankan perintah `pip install connexion[swagger-ui]` dari command prompt. Kemudian, kita dapat mengakses UI dokumentasi API kita pada link `{{basepath_swagger_url}}/ui`, contohnya pada aplikasi ini terdapat pada <http://34.83.91.7:5000/api/ui/>. Kemudian akan muncul tampilan seperti berikut.

 **Swagger**
Powered by SMARTBEAR

/api/openapi.json

Explore

DOKU - Tampered Image Detection System

1.0.0OAS3

/api/openapi.json

DOKU - Tampered Image Detection System

[Contact the developer](#)

Apache 2.0

Servers

/api

Main

POST

/predictor

Predict Images

Schemas

Result