1. Pengembangan *web service* dapat melibatkan aspek keamanan seperti akses terhadap layanan-layanan yang akan diintegrasikan dan menjamin keamanan data yang didistribusikan dari *server* ke *client.*
2. Melakukan percobaan *image preprocessing* yang lebih beragam untuk menghasilkan *dataset* yang siap diolah lebih lanjut.

DAFTAR PUSTAKA

Ahmadi, A. (2017, April 18). *Perbedaan REST dengan RESTfull API*. Retrieved January 23, 2021, from Medium: medium.com/@ahmad.fight/perbedaan-rest-dengan-restfull-api-c08025d6d59e

Bosak, J. (1997). Xml, java, and the future of the web. *Worl Wide Web Journal*, 219-227.

Bradley, J., Jones, M., & Sakimura, N. (2015). *JSON Web Token.* Internet Engineering Task Force (IETF) .

Brittenham, P. (2002, June 1). An Overview of the Web Service Inspection Language. *Distribute Web Service Discovery Using WS-Inspection documents*.

Damrongchai, A. (2017, June 14). *Test your APIs with Insomnia REST client*. Retrieved January 23, 2021, from Medium: https://medium.com/@artiwarahdamrongchai/test-your-apis-with-insomnia-rest-client-355093f32755

Dharwiyanti, S., & Wahono, R. S. (2003). Pengantar Unified Modeling Language (UML). *IlmuKomputer.com*, 02.

Ferdinandus, S., Wowor, H. F., Lumenta, A. S., & Rumagit, A. (2012). Perancangan Aplikasi Surat Masuk dan Surat Keluar Pada PT.PLN (Persero) Wilayah Suluttenggo. *Jurnal Teknik Elektro dan Komputer*, 1-7.

Ian, H. W., Eibe, F., & Mark, A. H. (2011). *Data Mining Practical Machine Learning Tools and Techniques, 3rd ed.* USA: Morgan Kaufmann Publishers.

Irsan, M. (2015). Rancang Bangun Aplikasi Mobile Notifikasi Berbasis Android untuk Mendukung Kinerja Instansi Pemerintahan. *Jurnal Sistem dan Teknologi Informasi*, 02.

JSON. (n.d.). Retrieved June 03, 2021, from json.org: https://www.json.org

Krismadi, A., Lestari, A. F., Pitriyah, A., Mardangga, W. P., Astuti, M., & Saifuddin, A. (2019). Pengujian Black Box Berbasis Equivalence Partitions pada Aplikasi Seleksi Promosi Kenaikan Jabatan. *Jurnal Teknologi Sistem Informasi dan Aplikasi*, 155-161.

Kurniawan, Y. K., Oslan, Y., & Kristanto, H. (2013). Implementasi REST API untuk Portal Akademik UKDW Berbasis Android. *Jurnal EKSIS Vol 06*, 02-03.

Leiva, A. (2017). *Kotlin for Android Developers.* Lean Publishing.

Lutz, M. (2010). *Programming Python, 4th Edition.* Sebastopol: O'Reilly Media, Inc.

Malik, F. (2019, March 10). *Everything About Python - Beginner To Advanced*. Retrieved January 25, 2021, from Medium: https://medium.com/fintechexplained/everything-about-python-from-beginner-to-advance-level-227d52ef32d2

Microsoft. (2000). *Application Service Provider: Evolution and Resources.* USA: White Paper.

Mulyanto, A. R. (2008). *Rekayasa Perangkat Lunak.* Jakarta: Direktorat Pembinaan Sekolah Menengah Kejuruan.

Newcomer, E. (2002). *Understanding Web Service XML, WSDL, SOAP, and UDDI.* Independent Technology Guide.

Rudianto, A. M. (2011). *Pemogramana Web Dinamis Menggunakan PHP dan MySQL*. Yogyakarta: C.V ANDI OFFSET.

Samudera, N. A., Mulyana, A., & Rakhman, E. (2015). Keamanan Ruangan Menggunakan Raspberry Pi(bagian: Aplikasi). *e-Proceeding of Engineering : Vol.2* (p. 3743). Bandung: Universitas Telkom.

Saputra, A. (2012). Manajemen Basis Data MySQL Pada Situs FTP Lapan Bandung. *Berita Dirgantara*, 03.

Sogen, M. T., & Kusuma, T. M. (2015). Rancang Bangun Purwarupa Sistem Pedeteksi Kendaraan Menggunakan Pustaka Opencv. *Politeknik Katolik Saint Paul Sorong-Jurnal Elektronic System*, 05.

Suprayogi, B., & Rahmanesa, A. (2019). Penenrapan Framework Bootstrap Dalam Sisitem Informasi Pendidikan SMA Negeri 1 Pacet Cianjur Jawa Barat. *TEMATIK - Jurnal Teknologi Informasi dan Komunikasi*, 120.

Sutanta, E., & Mustofa, K. (2012). Kebutuhan Web Service Untuk Sinkronisasi Data Antar Sistem Informasi Dalam E-GOV di Pemkab Bantuk Yogyakarta . *JURTIK - STIMIK BANDUNG*, 2.

Sutrisno, N. A. (2017, February 14). *REST API Client Sederhana dengan Retrofit pada Android Studio*. Retrieved July 9, 2021, from CODEPOLITAN: https://www.codepolitan.com/rest-api-client-sederhana-dengan-retrofit-pada-android-studio-58986d62c46ae

Syaban, R. M., & Bunyamin. (2015). Pengembangan SIstem Informasi Pengelolaan Surat Masuk dan Surat Keluar Berbasis Web di Dinas Sosial Tenaga Kerja dan Transmigrasi Kabupaten Garut Menggunakan Framework PHP. *Jurnal Algoritma*, 310-311.

Triasanti, D. (2001). *Konsep Dasar Python.* Jakarta: Universitas Gunadarma.

Tyas, A. A., & Ashari, A. (2016). Pemanfaatan Teknologi web Service untuk Integrasi Sistem Layanan Materi Pelajaran Terdistribusi. *Jurnal Angkas*.

W3C. (2004, February 11). Retrieved April 27, 2021, from https://www.w3.org/TR/2004/NOTE-ws-arch-20040211/

Wang , H. Y., Liao, C., & Yang, L. H. (2013). What Affects Mobile Application Use ? The Roles of Consumption Values. *International Journal of Marketing Studies*, 12.

Zaky, A. (2008). *36 Menit BelajarKomputer PHP dan MySQL*. Jakarta: PT. Elex Media Komputindo.

LAMPIRAN

Lampiran 1. Sintaks API

```
from datetime import timedelta
from posixpath import basename
from flask import Flask, request, jsonify, send_file, make_r
esponse
from flask_jwt_extended import create_access_token
from flask_jwt_extended import get_jwt_identity
from flask_jwt_extended import jwt_required
from flask_jwt_extended import JWTManager
from flask_mysqldb import MySQL
import MySQLdb.cursors
import time

from preprocessing import preprocess
import os
import io
import zipfile
import time
import shutil
```

```
baseDir = "Dataset"
datasetDir = "Dataset"
ALLOWED_EXTENSIONS = {'jpeg', 'jpg'}
app = Flask(__name__)
app.config['JWT_SECRET_KEY'] = 'skripsi-awesome-secret'
app.config['JWT_ACCESS_TOKEN_EXPIRES'] = timedelta(seconds=1
440)
app.config['MYSQL_HOST'] = 'localhost'
app.config['MYSQL_USER'] = 'root'
app.config['MYSQL_PASSWORD'] = ''
app.config['MYSQL_DB'] = 'image_preprocessing_dataset'
jwt = JWTManager(app)
mysql = MySQL(app)

def allowed_file(filename):
    return '.' in filename and filename.rsplit('.', 1)[1].lo
wer() in ALLOWED_EXTENSIONS

@app.route('/api/test')
@jwt_required()
def test():
    response = {
        "code": 401,
        "message": "Invalid credential",
        "data": None,
    }

    authenticated = get_jwt_identity()
    if authenticated is None:
        return jsonify(response), 401

    response = {
        "code": 200,
        "message": "Success",
        "data": authenticated
    }
    return jsonify(response)

@app.route('/api/auth', methods=['POST'])
def auth():
    response = {
        "code": 401,
        "message": "Invalid Credential",
        "data": None
    }
```

```
    username = request.json.get("username", None)
    password = request.json.get("password", None)

    result = None
    cursor = mysql.connection.cursor(MySQLdb.cursors.DictCur
sor)
    cursor.execute(
        "SELECT * FROM user WHERE uname = %s AND passw = %s"
,
        (username, password))
    row = cursor.fetchone()

    if row is None:
        return jsonify(response), 401

    result = {
        'username': row['uname'],
        'token': create_access_token(identity=row['uname'])
    }

    response = {
        "code": 200,
        "message": "Token generated successfully",
        "data": result
    }

    return jsonify(response)

@app.route('/api/users', methods=['POST'])
@jwt_required()
def users():
    response = {
        "success": 0,
        "msg": "Failed getting list user",
        "data": None
    }

    authenticated = get_jwt_identity()
    if authenticated is None:
        return jsonify(response), 401

    result = None
    query = "SELECT uname FROM user"
    cursor = mysql.connection.cursor(MySQLdb.cursors.DictCur
sor)
    cursor.execute(query)
```

```
    result = cursor.fetchall()
    if not result:
        return jsonify(response)
    users = [res['uname'] for res in result]
    response = {
        "success": 1,
        "msg": "Success getting users",
        "users": users
    }
    return jsonify(response)

@app.route('/api/users/<uname>/delete', methods=['POST'])
@jwt_required()
def user_delete(uname):
    response = {
        "success": 0,
        "msg": "Failed deleting list user",
        "data": None
    }

    authenticated = get_jwt_identity()
    if authenticated is None:
        return jsonify(response), 401

    cursor = mysql.connection.cursor(MySQLdb.cursors.DictCur
sor)
    try:
        cursor.execute(
            "DELETE FROM verified WHERE uname = %s", (uname,
))
        cursor.execute(
            "DELETE FROM user WHERE uname = %s", (uname,))
        mysql.connection.commit()
    except Exception as e:
        print(str(e))
        return jsonify(response)
    global baseDir
    pathDataset = os.path.join(baseDir, uname)
    if os.path.exists(pathDataset):
        try:
            shutil.rmtree(pathDataset)
        except OSError as e:
            print("Error: %s : %s" % (pathDataset, e.strerro
r))
    response = {
        "success": 1,
```

```
        "msg": "Success deleting users",
        "users": uname
    }
    return jsonify(response)

@app.route('/api/create_dataset', methods=['POST'])
@jwt_required()
def create_dataset():
    response = {
        "success": 0,
        "msg": "Failed create dataset",
        "data": None
    }

    authenticated = get_jwt_identity()
    if authenticated is None:
        return jsonify(response), 401

    dataset = request.form.get("dataset")
    uname = request.form.get("uname")
    global baseDir
    pathDataset = os.path.join(baseDir, uname, dataset)
    if not os.path.exists(pathDataset):
        os.makedirs(pathDataset)
    response = {
        "success": 1,
        "msg": "Success create dataset",
        "data": {
            "path": pathDataset
        }
    }
    return jsonify(response)

@app.route('/api/create_sub_dataset', methods=['POST'])
@jwt_required()
def create_sub_dataset():
    response = {
        "success": 0,
        "msg": "Failed create dataset",
        "data": None
    }

    authenticated = get_jwt_identity()
    if authenticated is None:
        return jsonify(response), 401
```

```
    subdataset = request.form.get("subdataset")
    dataset = request.form.get("dataset")
    uname = request.form.get("uname")
    global baseDir
    pathSubDataset = os.path.join(baseDir, uname, dataset, s
ubdataset)
    if not os.path.exists(pathSubDataset):
        os.makedirs(pathSubDataset)
    response = {
        "success": 1,
        "msg": "Success create dataset",
        "data": {
            "path": pathSubDataset
        }
    }
    return jsonify(response)

@app.route('/api/upload_gambar', methods=['POST'])
@jwt_required()
def upload_gambar():
    response = {
        "success": 0,
        "msg": "Failed upload image",
        "data": None
    }

    authenticated = get_jwt_identity()
    if authenticated is None:
        return jsonify(response), 401

    if 'gambar' not in request.files:
        return(jsonify(response))
    gambar = request.files['gambar']
    subdataset = request.form.get("subdataset")
    dataset = request.form.get("dataset")
    uname = request.form.get("uname")

    if not gambar or gambar.filename == "" or not allowed_fi
le(gambar.filename):
        return(jsonify(response))

    cursor = mysql.connection.cursor(MySQLdb.cursors.DictCur
sor)
    cursor.execute(
        "DELETE FROM verified WHERE dataset = %s AND subdata
set = %s AND uname = %s", (dataset, subdataset, uname))
```

```
    mysql.connection.commit()
    now = int(round(time.time()*1000))
    global baseDir
    pathSubDataset = os.path.join(baseDir, uname, dataset, s
ubdataset)
    app.config['UPLOAD_FOLDER'] = pathSubDataset
    if not os.path.exists(app.config['UPLOAD_FOLDER']):
        os.makedirs(app.config['UPLOAD_FOLDER'])
    path = os.path.join(app.config['UPLOAD_FOLDER'],
                        f"{now}{os.path.splitext(gambar.file
name)[1]}")
    gambar.save(path)
    pRes = preprocess(app.config['UPLOAD_FOLDER'], path)
    response = {
        "success": 1,
        "msg": "Success upload image",
        "data": {
            "uname": uname,
            "path": path,
            "preprocess": pRes
        }
    }
    return(jsonify(response))

@app.route('/api/register', methods=['POST'])
@jwt_required(optional=True)
def register():
    response = {
        "success": 0,
        "msg": "Failed register",
        "data": None
    }

    uname = request.form.get("uname")
    nama = request.form.get("nama")
    passw = request.form.get("passw")
    level = request.form.get("level")

    cursor = mysql.connection.cursor(MySQLdb.cursors.DictCur
sor)
    res = cursor.execute(
        "INSERT INTO user VALUES (%s, %s, %s, %s)",
        (uname, nama, passw, level))
    mysql.connection.commit()
    global baseDir
    global datasetDir
```

```
    if os.path.exists(baseDir):
        subdir = os.listdir(os.path.join(baseDir, "administr
ator"))
        for dataset in subdir:
            if os.path.isdir(os.path.join(baseDir, "administ
rator", dataset)):
                for subdataset in os.listdir(os.path.join(ba
seDir, "administrator", dataset)):
                    os.makedirs(os.path.join(
                        baseDir, uname, dataset, subdataset)
)
    if res:
        response = {
            "success": 1,
            "msg": "Success register",
            "data": res
        }
        return jsonify(response)
    else:
        response["data"] = res
        return jsonify(response)

@ app.route('/api/login', methods=['POST'])
@jwt_required()
def login():
    response = {
        "success": 0,
        "msg": "Failed login",
        "data": None
    }

    authenticated = get_jwt_identity()
    if authenticated is None:
        return jsonify(response), 401

    uname = request.form.get("uname")
    passw = request.form.get("passw")
    result = None
    cursor = mysql.connection.cursor(MySQLdb.cursors.DictCur
sor)
    cursor.execute(
        "SELECT * FROM user WHERE uname = %s AND passw = %s"
,
        (uname, passw))
    row = cursor.fetchone()
    if row != None:
```

```
            result = {
                'uname': row['uname'],
                'nama': row['name'],
                'passw': row['passw'],
                'level': row['level']
            }
        if result == None:
            return jsonify(response)
        else:
            global baseDir
            userPath = os.path.join(baseDir, uname)
            if not os.path.exists(userPath):
                os.makedirs(userPath)
            response = {
                "success": 1,
                "msg": "Success login",
                "data": result
            }
            return jsonify(response)

@ app.route('/api/list_dataset', methods=['POST'])
@jwt_required(optional=True)
def list_dataset():
    response = {
        "success": 0,
        "msg": "Failed getting list dataset",
        "data": None
    }

    uname = request.form.get("uname")
    global baseDir
    if uname != None:
        userPath = os.path.join(baseDir, uname)
        if os.path.exists(userPath):
            subdir = os.listdir(userPath)
            listDataset = [name for name in subdir if os.pat
h.isdir(
                os.path.join(userPath, name))]
        else:
            return jsonify(response)
        response = {
            "success": 1,
            "msg": "Success getting list dataset",
            "data": listDataset
        }
    else:
```

```
        if os.path.exists(baseDir):
            subdir = os.listdir(baseDir)
            listUser = [name for name in subdir if os.path.i
sdir(
                os.path.join(baseDir, name))]
            listDataset = []
            for user in listUser:
                userPath = os.path.join(baseDir, user)
                subdirUser = os.listdir(userPath)
                for name in subdirUser:
                    if os.path.isdir(os.path.join(userPath,
name)):
                        listDataset.append((user, name))
                response = {
                    "success": 1,
                    "msg": "Success getting list dataset",
                    "data": listDataset
                }
    return jsonify(response)

@ app.route('/api/dataset', methods=['POST'])
@jwt_required(optional=True)
def dataset():
    response = {
        "success": 0,
        "msg": "Failed getting list dataset",
        "data": None
    }

    dataset = request.form.get("dataset")
    global baseDir
    if os.path.exists(baseDir):
        subdir = os.listdir(baseDir)
        listUser = [name for name in subdir if os.path.isdir
(
            os.path.join(baseDir, name))]
        listDataset = []
        for user in listUser:
            userPath = os.path.join(baseDir, user)
            subdirUser = os.listdir(userPath)
            for name in subdirUser:
                if os.path.isdir(os.path.join(userPath, name
)):
                    if name == dataset:
                        listDataset.append((user, name))
            response = {
```

```
                "success": 1,
                "msg": "Success getting list dataset",
                "data": listDataset
            }
    return jsonify(response)

@ app.route('/api/verify', methods=['POST'])
@jwt_required()
def verify():
    response = {
        "success": 0,
        "msg": "Failed verify subdataset user",
        "data": None
    }

    authenticated = get_jwt_identity()
    if authenticated is None:
        return jsonify(response), 401

    uname = request.form.get("uname")
    dataset = request.form.get("dataset")
    subdataset = request.form.get("subdataset")

    cursor = mysql.connection.cursor(MySQLdb.cursors.DictCur
sor)
    res = cursor.execute("INSERT INTO verified VALUES (%s, %
s, %s)",
                         (dataset, subdataset, uname))
    mysql.connection.commit()
    if res == None:
        response = {
            "success": 1,
            "msg": "Success verify subdataset user",
            "data": {
                "uname": uname,
                "dataset": dataset,
                "subdataset": subdataset
            }
        }
    return jsonify(response)

@ app.route('/api/verified/<user>/<dataset>/<subdataset>')
@jwt_required(optional=True)
def verified(user, dataset, subdataset):
    cursor = mysql.connection.cursor(MySQLdb.cursors.DictCur
sor)
```

```
    cursor.execute(
        "SELECT * FROM verified WHERE dataset = %s AND subda
taset = %s AND uname = %s",
        (dataset, subdataset, user)
    )
    res = cursor.fetchone()
    if res != None:
        return jsonify({
            "result": 1
        })
    else:
        return jsonify({
            "result": 0
        })

@ app.route('/api/list_sub_dataset', methods=['POST'])
@jwt_required(optional=True)
def list_sub_dataset():
    response = {
        "success": 0,
        "msg": "Failed getting list sub-dataset",
        "data": None
    }

    uname = request.form.get("uname")
    dataset = request.form.get("dataset")
    global baseDir
    userPath = os.path.join(baseDir, uname, dataset)
    if os.path.exists(userPath):
        subdir = os.listdir(userPath)
        listDataset = [name for name in subdir if os.path.is
dir(
            os.path.join(userPath, name))]
    else:
        return jsonify(response)
    response = {
        "success": 1,
        "msg": "Success getting list sub-dataset",
        "data": listDataset
    }
    return jsonify(response)

@ app.route('/api/preprocess/<user>/<dataset>/<subdataset>/<
gambar>')
@jwt_required(optional=True)
def preprocessing(user, dataset, subdataset, gambar):
```

```
    global baseDir
    imgPath = os.path.join(baseDir, user, dataset,
                           subdataset, "preprocess", gambar)
    return send_file(imgPath)

@ app.route('/api/gambar/<user>/<dataset>/<subdataset>/<gamb
ar>')
@jwt_required(optional=True)
def gambar(user, dataset, subdataset, gambar):
    global baseDir
    imgPath = os.path.join(baseDir, user, dataset, subdatase
t, gambar)
    return send_file(imgPath)

@ app.route('/api/list_gambar', methods=['POST'])
@jwt_required(optional=True)
def list_gambar():
    response = {
        "success": 0,
        "msg": "Failed getting list image",
        "data": None
    }

    uname = request.form.get("uname")
    dataset = request.form.get("dataset")
    subdataset = request.form.get("subdataset")
    global baseDir
    userPath = os.path.join(baseDir, uname, dataset, subdata
set)
    if os.path.exists(userPath):
        subfile = os.listdir(userPath)
        listImage = [name for name in subfile if not os.path
.isdir(
            os.path.join(userPath, name))]
    else:
        return jsonify(response)
    response = {
        "success": 1,
        "msg": "Success getting list image",
        "data": listImage
    }
    return jsonify(response)

@app.route('/api/downloads', methods=['GET', 'POST'])
@jwt_required(optional=True)
def downloads():
```

```
    username = request.json.get("username", None)
    dataset = request.json.get("dataset", None)
    subdataset = request.json.get("subdataset", "")
    response = {
        "code": 401,
        "message": "Invalid Credential",
        "data": None
    }

    if username is None or dataset is None:
        return jsonify(response)

    global baseDir
    PATH = os.path.join(baseDir, username, dataset, subdatas
et)
    fileobj = io.BytesIO()
    with zipfile.ZipFile(fileobj, 'w') as zip_file:
        zip_info = zipfile.ZipInfo(PATH)
        zip_info.date_time = time.localtime(time.time())[:6]
        zip_info.compress_type = zipfile.ZIP_DEFLATED
        for file in os.listdir(os.path.join(PATH, "preproces
s")):
            file_path = os.path.join(PATH, "preprocess", fil
e)
            zip_file.write(file_path, basename(file_path))
    fileobj.seek(0)
    response = make_response(fileobj.read())
    response.headers.set('Content-Type', 'zip')
    response.headers.set('Content-
Disposition', 'attachment',
                         filename='%s.zip' % os.path.basenam
e(PATH))
    return response

@ app.route('/api/list_preproc', methods=['POST'])
@jwt_required(optional=True)
def list_preproc():
    response = {
        "success": 0,
        "msg": "Failed getting list preprocessed image",
        "data": None,
        "array": None,
    }

    uname = request.form.get("uname")
    dataset = request.form.get("dataset")
```

```
    subdataset = request.form.get("subdataset")
    global baseDir
    userPath = os.path.join(baseDir, uname, dataset, subdata
set)
    if os.path.exists(userPath):
        subfile = os.listdir(userPath)
        for imageName in subfile:
            imagePath = os.path.join(userPath, imageName)
            if not os.path.isdir(imagePath):
                preprocess(userPath, imagePath)
    userPathPre = os.path.join(userPath, "preprocess")
    if os.path.exists(userPathPre):
        subfile = os.listdir(userPathPre)
        listPre = [name for name in subfile if not os.path.i
sdir(
            os.path.join(userPathPre, name))]
    else:
        return jsonify(response)
    response = {
        "success": 1,
        "msg": "Success getting preprocessed list image",
        "data": listPre,
    }
    return jsonify(response)

def fun_del_preproc(dir, gambar):
    name, ext = os.path.splitext(gambar)
    res = []
    for i in range(1, 4):
        path = os.path.join(dir, "preprocess", f"{name}_{i}{
ext}")
        if os.path.exists(path):
            os.remove(path)
            res.append(path)
    return res

@ app.route('/api/del_gambar', methods=['POST'])
@jwt_required()
def del_gambar():
    response = {
        "success": 0,
        "msg": "Failed delete image",
        "data": None
    }

    authenticated = get_jwt_identity()
```

```
    if authenticated is None:
        return jsonify(response), 401

    uname = request.form.get("uname")
    dataset = request.form.get("dataset")
    subdataset = request.form.get("subdataset")
    gambar = request.form.get("gambar")
    global baseDir
    dir = os.path.join(baseDir, uname, dataset, subdataset)
    userPath = os.path.join(dir, gambar)
    pRes = None
    if os.path.exists(userPath):
        os.remove(userPath)
        pRes = fun_del_preproc(dir, gambar)
    else:
        return jsonify(response)
    response = {
        "success": 1,
        "msg": "Success delete image",
        "data": {
            "uname": uname,
            "path": userPath,
            "preprocess": pRes
        }
    }
    return jsonify(response)

@ app.route('/api/del_preproc', methods=['POST'])
@jwt_required()
def del_preproc():
    response = {
        "success": 0,
        "msg": "Failed delete preprocessed image",
        "data": None
    }

    authenticated = get_jwt_identity()
    if authenticated is None:
        return jsonify(response), 401

    uname = request.form.get("uname")
    dataset = request.form.get("dataset")
    subdataset = request.form.get("subdataset")
    gambar = request.form.get("gambar")
    global baseDir
    dir = os.path.join(baseDir, uname, dataset, subdataset)
```

```
    userPath = os.path.join(dir, "preprocess", gambar)
    if os.path.exists(userPath):
        os.remove(userPath)
    else:
        return jsonify(response)
    response = {
        "success": 1,
        "msg": "Success delete preprocessed image",
        "data": {
            "uname": uname,
            "path": userPath
        }
    }
    return jsonify(response)

if __name__ == '__main__':
    app.run(host='0.0.0.0')
```

Lampiran 2. Sintaks *Preprocessing*

```
def preprocess(filedir, file):
    import numpy as np
    import cv2 as cv
    import os

    file_basename,                file_ext                =
os.path.splitext(os.path.basename(file))

    ppDir = os.path.join(filedir, "preprocess")
    if not os.path.exists(ppDir):
        os.makedirs(ppDir)

    preprocess_Res = [
        os.path.join(ppDir, f"{file_basename}_1{file_ext}"),
        os.path.join(ppDir, f"{file_basename}_2{file_ext}"),
        os.path.join(ppDir, f"{file_basename}_3{file_ext}"),
    ]

    allExists = True
    for pRes in preprocess_Res:
        if not os.path.exists(pRes):
            allExists = False

    if allExists:
        return None
```

```
    img = cv.imread(file)

    image = cv.resize(img, (640, 480))

    den = cv.GaussianBlur(image, (5, 5), 0)
    rgb_dst = cv.fastNlMeansDenoisingColored(den, None, 10,
10, 7, 15)

    import random
    aug_img_H_Flip = cv.flip(rgb_dst, 0)  # vertical flipping
    aug_img_V_Flip = cv.flip(rgb_dst, 1)  # vertical flipping
    aug_img_HV_Flip = cv.flip(rgb_dst, -1)   # vertical and
horizontal flipping

    if not os.path.exists(preprocess_Res[0]):
        cv.imwrite(preprocess_Res[0], aug_img_HV_Flip)
    if not os.path.exists(preprocess_Res[1]):
        cv.imwrite(preprocess_Res[1], aug_img_H_Flip)
    if not os.path.exists(preprocess_Res[2]):
        cv.imwrite(preprocess_Res[2], aug_img_V_Flip)

    return preprocess_Res
```

Lampiran 3. Contoh sintaks Aplikasi *Website*

- ***Login.php***

```
<?php
session_start();
if               (isset($_SESSION['uname'])               &&
isset($_SESSION['access_token'])) {
    header("Location: index.php");
    exit;
}
?>
<!doctype html>
<html lang="en">

<head>
    <!-- Required meta tags -->
    <meta charset="utf-8">
    <meta    name="viewport"    content="width=device-width,
initial-scale=1, shrink-to-fit=no">

    <!-- Bootstrap CSS -->
    <link rel="stylesheet" href="lib/css/bootstrap.min.css">
```

```
    <script src="lib/js/jquery.min.js"></script>

    <title>Dataset Image - Flask</title>
</head>

<body style="background-color: #435f7c;">
    <div class="container-fluid">

        <?php
        if (isset($_POST['login'])) {
            $query = array(
                "uname" => $_POST['uname'],
                "passw" => $_POST['pass']
            );

            include 'lib/Config.php';
            $auth = curl_init();
            $payload = json_encode(array(
                "username" => $_POST['uname'],
                "password" => $_POST['pass']
            ));
            curl_setopt($auth,   CURLOPT_URL,   API_URL   .
"auth");
            curl_setopt($auth, CURLOPT_POST, 1);
            curl_setopt($auth,           CURLOPT_POSTFIELDS,
$payload);
            curl_setopt($auth,           CURLOPT_HTTPHEADER,
array("Content-Type: application/json"));
            curl_setopt($auth,       CURLOPT_RETURNTRANSFER,
true);
            $res = json_decode(curl_exec($auth));
            $_SESSION["access_token"] = $res->data->token;

            $ch = curl_init();

            $authorization  =  "Authorization:  Bearer  "  .
$_SESSION['access_token'];
            curl_setopt($ch, CURLOPT_URL, API_URL . "login");
            curl_setopt($ch, CURLOPT_POST, 1);
            curl_setopt($ch,             CURLOPT_POSTFIELDS,
http_build_query($query));
            curl_setopt($ch, CURLOPT_HTTPHEADER, array(
                'Content-Type:       application/x-www-form-
urlencoded',
                $authorization
```

```
            ));
            curl_setopt($ch, CURLOPT_RETURNTRANSFER, true);
            curl_setopt($ch, CURLOPT_HEADER, false);

            $c_res = curl_exec($ch);
            $response = json_decode($c_res, true);

            if ($response["success"] == 1) {
                $user = $response['data'];
                $_SESSION['nama'] =  $user['nama'];
                $_SESSION['uname'] =  $user['uname'];
                $_SESSION['pass'] = $user['passw'];
                $_SESSION['level'] =  $user['level'];
                if (isset($_POST['remember'])) {
                    setcookie('save_uname',  $user['uname'],
time() + (86400 * 30), "/");
                    setcookie('save_pass',   $user['passw'],
time() + (86400 * 30), "/");
                } else {
                    clear_cookie('save_uname');
                    clear_cookie('save_pass');
                }
                header("Location: index.php");
            } else {
                showAlert($response["msg"]);
            }
        }

        function showAlert($text)
        {
            echo "
                <div   class=\"alert   alert-danger   alert-
dismissible  fade  show\"  role=\"alert\"  style=\"margin-top:
1rem;\">
                    $text
                    <button  type=\"button\"  class=\"close\"
data-dismiss=\"alert\" aria-label=\"Close\">
                        <span                            aria-
hidden=\"true\">&times;</span>
                    </button>
                </div>";
        }

        function clear_cookie($cookie_name)
        {
            if (isset($_COOKIE[$cookie_name])) {
```

```
                unset($_COOKIE[$cookie_name]);
                setcookie($cookie_name, null, -1, '/');
            }
        }
        ?>

        <div class="col" style="margin-bottom: 3rem;">
            <h1 class="text-center mt-5">Halaman Login</h1>

            <div     class="row     justify-content-center"
style="margin-top: 3rem;">
                <div class="card" style="width: 20rem;">
                    <div class="card-body">
                        <form                  method="post"
enctype="multipart/form-data">
                            <div class="form-group">
                                <label
for="uname">Username</label>
                                <input           type="text"
class="form-control" id="uname" name="uname" required>
                            </div>
                            <div class="form-group">
                                <label
for="pass">Password</label>
                                <input       type="password"
class="form-control" id="pass" name="pass" required>
                            </div>
                            <div   class="form-group   form-
check">
                                <input       type="checkbox"
class="form-check-input"    id="remember"    name="remember"
value="checked">
                                <label    class="form-check-
label" for="remember">Ingat Saya</label>
                            </div>
                            <button type="submit" class="btn
btn-primary"             id="login"             name="login"
value="login">Login</button>
                            <a           href="register.php"
type="button" class="btn btn-success">Register</a>
                            <a              href="index.php"
type="button" class="btn btn-secondary">Guest</a>
                        </form>
                    </div>
                </div>
            </div>
```

```
        </div>

        <script>
            var        cond        =        <?php        if
(isset($_COOKIE['save_uname'])                              &&
isset($_COOKIE['save_pass'])) {
                            echo 1;
                        } else {
                            echo 0;
                        } ?>;
            if (cond == 1) {
                document.getElementById("remember").checked
= true;
                document.getElementById("uname").value     =
'<?php      if      (isset($_COOKIE['save_uname']))      echo
$_COOKIE['save_uname']; ?>';
                document.getElementById("pass").value      =
'<?php      if      (isset($_COOKIE['save_pass']))      echo
$_COOKIE['save_pass']; ?>';
            }
        </script>

    </div>
    <script src="lib/js/popper.min.js"></script>
    <script src="lib/js/bootstrap.min.js"></script>
</body>

</html>
```

- ***Create_dataset.php***

```
<?php
session_start();
if (!isset($_SESSION['uname'])) {
    header("Location: login.php");
    exit;
}

if (!isset($_SESSION['access_token'])) {
    header("Location: login.php");
    exit;
}

?>
<!doctype html>
<html lang="en">
```

```
<head>
    <!-- Required meta tags -->
    <meta charset="utf-8">
    <meta    name="viewport"    content="width=device-width,
initial-scale=1, shrink-to-fit=no">

    <!-- Bootstrap CSS -->
    <link rel="stylesheet" href="lib/css/bootstrap.min.css">

    <script src="lib/js/jquery.min.js"></script>

    <title>Dataset Image - Flask</title>
</head>

<body style="background-color: #435f7c;">
    <div class="container-fluid">

        <?php
        if (isset($_POST['create'])) {
            $dataset = $_POST['nama'];

            $msg = null;
            $arrds = unpack("C*", $dataset);
            foreach ($arrds as $c) {
                if (!(($c >= 65 && $c <= 90) || ($c >= 97 &&
$c <= 132))) {
                    $msg   =   "Nama   dataset   hanya   boleh
mengandung huruf";
                    break;
                }
            }

            if ($msg == null) {

                include 'lib/Config.php';
                $get_users = curl_init();

                $authorization = "Authorization: Bearer " .
$_SESSION['access_token'];
                curl_setopt($get_users, CURLOPT_URL, API_URL
. "users");
                curl_setopt($get_users, CURLOPT_POST, 1);
                curl_setopt($get_users,  CURLOPT_HTTPHEADER,
array('Content-Type:     application/x-www-form-urlencoded',
$authorization));
```

```
                curl_setopt($get_users,
CURLOPT_RETURNTRANSFER, true);
                curl_setopt($get_users,      CURLOPT_HEADER,
false);

                $users = curl_exec($get_users);
                $users_response = json_decode($users, true);
                $is_scuccess = FALSE;
                $message = null;
                foreach ($users_response['users'] as $key =>
$uname) {
                    if (!$uname || empty($uname)) {
                        continue;
                    }

                    $query = array(
                        "uname" => $uname,
                        "dataset" => $dataset
                    );

                    $ch = curl_init();

                    curl_setopt($ch, CURLOPT_URL, API_URL .
"create_dataset");
                    curl_setopt($ch, CURLOPT_POST, 1);
                    curl_setopt($ch,     CURLOPT_POSTFIELDS,
http_build_query($query));
                    curl_setopt($ch,     CURLOPT_HTTPHEADER,
array(
                        'Content-Type:    application/x-www-
form-urlencoded',
                        $authorization
                    ));
                    curl_setopt($ch, CURLOPT_RETURNTRANSFER,
true);
                    curl_setopt($ch, CURLOPT_HEADER, false);

                    $c_res = curl_exec($ch);
                    $response = json_decode($c_res, true);

                    $is_scuccess = $response["success"] == 1;
                    $message = $response["msg"];
                }

                if ($is_scuccess) {
                    header("Location: index.php");
```

```
                } else {
                    showAlert($message);
                }
            } else {
                showAlert($msg);
            }
        }

        function showAlert($text)
        {
            echo "
                <div   class=\"alert   alert-danger   alert-
dismissible  fade  show\"  role=\"alert\"  style=\"margin-top:
1rem;\">
                    $text
                    <button  type=\"button\"  class=\"close\"
data-dismiss=\"alert\" aria-label=\"Close\">
                        <span                            aria-
hidden=\"true\">&times;</span>
                    </button>
                </div>";
        }
        ?>

        <div class="col" style="margin-bottom: 3rem;">
            <h1 class="text-center mt-5">Dataset Baru</h1>

            <div     class="row     justify-content-center"
style="margin-top: 3rem;">
                <div class="card" style="width: 20rem;">
                    <div class="card-body">
                        <form                   method="post"
enctype="multipart/form-data">
                            <div class="form-group">
                                <label       for="nama">Nama
Dataset</label>
                                <input           type="text"
class="form-control" id="nama" name="nama" required>
                            </div>
                            <button type="submit" class="btn
btn-success"            id="create"            name="create"
value="create">Create</button>
                            <a              href="index.php"
type="button" class="btn btn-danger">Kembali</a>
                        </form>
                    </div>
```

```
                </div>
            </div>
        </div>

    </div>
    <script src="lib/js/popper.min.js"></script>
    <script src="lib/js/bootstrap.min.js"></script>
</body>

</html>
```

- ***upload_gambar.php***

```
<?php
session_start();
if (!isset($_SESSION['uname'])) {
    header("Location: login.php");
    exit;
}
if (!(isset($_GET['page']) && isset($_GET['ds']))) {
    header("Location: index.php");
    exit;
}

$user = null;
$is_admin = FALSE;

if (isset($_GET['is_admin']) && $_GET['is_admin']) {
    $user = $_GET['uname'];
    $is_admin = !$is_admin;
} else {
    $user = $_SESSION['uname'];
}
if (!isset($_SESSION['access_token'])) {
    header("Location: login.php");
    exit;
}
?>
<!doctype html>
<html lang="en">

<head>
    <!-- Required meta tags -->
    <meta charset="utf-8">
    <meta    name="viewport"    content="width=device-width,
initial-scale=1, shrink-to-fit=no">
```

```
    <!-- Bootstrap CSS -->
    <link rel="stylesheet" href="lib/css/bootstrap.min.css">

    <script src="lib/js/jquery.min.js"></script>

    <title>Dataset Image - Flask</title>
</head>

<body style="background-color: #435f7c;">
    <div class="container-fluid">

        <?php
        if (isset($_POST['create'])) {
            $gambar = $_FILES['gambar'];



            $query = array(
                "uname" => $user,
                "dataset" => $_GET['ds'],
                "subdataset" => $_GET['page'],
                "gambar"                                        =>
curl_file_create($gambar['tmp_name'],
mime_content_type($gambar['tmp_name']), $gambar['name'])
            );

            include 'lib/Config.php';
            $ch = curl_init();

            $authorization  =  "Authorization:  Bearer  "  .
$_SESSION['access_token'];
            curl_setopt($ch,    CURLOPT_URL,    API_URL    .
"upload_gambar");
            curl_setopt($ch, CURLOPT_POST, 1);
            curl_setopt($ch, CURLOPT_POSTFIELDS, $query);
            curl_setopt($ch, CURLOPT_HTTPHEADER, array(
                'Content-Type: multipart/form-data',
                $authorization
            ));
            curl_setopt($ch, CURLOPT_RETURNTRANSFER, true);
            curl_setopt($ch, CURLOPT_HEADER, false);

            $c_res = curl_exec($ch);
            $response = json_decode($c_res, true);

            if ($response["success"] == 1) {
```

```
                if ($is_admin) {
                    header("Location:  gambar.php?uname="  .
$user . "&page=" . $_GET['page'] . "&ds=" . $_GET['ds']);
                } else {
                    header("Location:   gambar.php?page="   .
$_GET['page'] . "&ds=" . $_GET['ds']);
                }
            } else {
                showAlert($response["msg"]);
            }
        }

        function showAlert($text)
        {
            echo "
                <div   class=\"alert   alert-danger   alert-
dismissible  fade  show\"  role=\"alert\"  style=\"margin-top:
1rem;\">
                    $text
                    <button  type=\"button\"  class=\"close\"
data-dismiss=\"alert\" aria-label=\"Close\">
                        <span                           aria-
hidden=\"true\">&times;</span>
                    </button>
                </div>";
        }
        ?>

        <div class="col" style="margin-bottom: 3rem;">
            <h1 class="text-center mt-5">Gambar Baru</h1>

            <div     class="row     justify-content-center"
style="margin-top: 3rem;">
                <div class="card" style="width: 20rem;">
                    <div class="card-body">
                        <form                  method="post"
enctype="multipart/form-data">
                            <div class="custom-file mb-3">
                                <input           type="file"
accept="image/jpeg"  class="custom-file-input"  name="gambar"
id="gambar" required>
                                <label   class="custom-file-
label" for="gambar">Pilih gambar...</label>
                            </div>
```

```
                            <button type="submit" class="btn
btn-success"            id="create"            name="create"
value="create">Upload</button>
                            <?php if ($is_admin) : ?>
                                <a
href="gambar.php?uname=<?=  $user  ?>&page=<?=  $_GET['page']
?>&ds=<?=  $_GET['ds']  ?>"  type="button"  class="btn  btn-
danger">
                                    Kembali
                                </a>
                            <?php else : ?>
                                <a href="gambar.php?page=<?=
$_GET['page']   ?>&ds=<?=   $_GET['ds']   ?>"   type="button"
class="btn btn-danger">
                                    Kembali
                                </a>
                            <?php endif; ?>
                        </form>
                    </div>
                </div>
            </div>
        </div>

    </div>

    <script>
        document.querySelector('.custom-file-
input').addEventListener('change', function(e) {
            var                     nama                    =
document.getElementById("gambar").files[0].name;
            var sib = e.target.nextElementSibling;
            sib.innerText = nama;
        })
    </script>

    <script src="lib/js/popper.min.js"></script>
    <script src="lib/js/bootstrap.min.js"></script>
</body>

</html>
```

- ***delete_gambar.php***

```
<?php
session_start();
$is_admin = FALSE;
```

```
$user = null;

if (!isset($_SESSION['uname'])) {
    header("Location: login.php");
    exit;
}

if   (!(isset($_GET['page'])   &&   isset($_GET['ds'])   &&
isset($_GET['sds']))) {
    header("Location: index.php");
    exit;
}

if (isset($_GET['is_admin']) && $_GET['is_admin']) {
    $user = $_GET['uname'];
    $is_admin = !$is_admin;
} else {
    $user = $_SESSION['uname'];
}

$query = array(
    "uname" => $user,
    "gambar" => $_GET['page'],
    "dataset" => $_GET['ds'],
    "subdataset" => $_GET['sds']
);

if (!isset($_SESSION['access_token'])) {
    header("Location: login.php");
    exit;
}

include 'lib/Config.php';
$ch = curl_init();

$authorization    =    "Authorization:    Bearer    "    .
$_SESSION['access_token'];
curl_setopt($ch, CURLOPT_URL, API_URL . "del_gambar");
curl_setopt($ch, CURLOPT_POST, 1);
curl_setopt($ch,                          CURLOPT_POSTFIELDS,
http_build_query($query));
curl_setopt($ch, CURLOPT_HTTPHEADER, array(
    'Content-Type: application/x-www-form-urlencoded',
    $authorization
));
curl_setopt($ch, CURLOPT_RETURNTRANSFER, true);
```

```
curl_setopt($ch, CURLOPT_HEADER, false);

$c_res = curl_exec($ch);
$response = json_decode($c_res, true);

if ($is_admin) {
    header("Location:  gambar.php?page="  .  $_GET['sds']  .
"&uname=" . $user . "&ds=" . $_GET['ds']);
} else {
    header("Location:  gambar.php?page="  .  $_GET['sds']  .
"&ds=" . $_GET['ds']);
}
```

Lampiran 4. Contoh sintaks Aplikasi *Mobile*

- ***AuthRepo.kt***

```
package com.android.datasetflask.repo

import androidx.lifecycle.MutableLiveData
import com.android.datasetflask.api.ApiClient
import com.android.datasetflask.model.AuthRequest
import com.android.datasetflask.model.AuthResponse
import com.android.datasetflask.model.UserDataResponse
import com.android.datasetflask.model.UserResponse
import retrofit2.Call
import retrofit2.Response

object AuthRepo {

    fun auth(user: AuthRequest):
MutableLiveData<AuthResponse> {
        val res = MutableLiveData<AuthResponse>()
        val call = ApiClient.apiInterface.auth(user)
        call?.enqueue(object :
retrofit2.Callback<AuthResponse?> {
            override fun onResponse(call:
Call<AuthResponse?>, response: Response<AuthResponse?>) {
                res.postValue(response.body())
            }

            override fun onFailure(call:
```

```
Call<AuthResponse?>, t: Throwable) {
                TODO("Not yet implemented")
            }

        })
        return res
    }

    fun login(token: String?, user: UserDataResponse):
MutableLiveData<UserResponse> {
        val res = MutableLiveData<UserResponse>()
        val bearer = "Bearer $token"

        val call = ApiClient.apiInterface.login(bearer,
user.uname, user.passw)

        call?.enqueue(object :
retrofit2.Callback<UserResponse?> {
            override fun onResponse(call:
Call<UserResponse?>, response: Response<UserResponse?>) {
                res.postValue(response.body())
            }

            override fun onFailure(call:
Call<UserResponse?>, t: Throwable) {
                val obj = UserResponse(success = 0, msg =
t.message, data = null)
                res.postValue(obj)
            }
        })

        return res
    }
}
```

- ***DatasetRepo.kt***

```
package com.android.datasetflask.repo

import androidx.lifecycle.MutableLiveData
import com.android.datasetflask.api.ApiClient
import com.android.datasetflask.model.DatasetResponse
import com.android.datasetflask.model.ListDatasetResponse
import okhttp3.MediaType.Companion.toMediaTypeOrNull
import okhttp3.MultipartBody
import okhttp3.RequestBody.Companion.asRequestBody
```

```
import okhttp3.RequestBody.Companion.toRequestBody
import retrofit2.Call
import retrofit2.Response
import java.io.File


object DatasetRepo {
    private val dsMld = MutableLiveData<DatasetResponse>()
    private val sdsMld = MutableLiveData<DatasetResponse>()
    private val gMld = MutableLiveData<DatasetResponse>()
    private val ldsMld =
MutableLiveData<ListDatasetResponse>()
    private val lsdsMld =
MutableLiveData<ListDatasetResponse>()
    private val lgMld =
MutableLiveData<ListDatasetResponse>()

    fun createDataset(nama: String, uname: String):
MutableLiveData<DatasetResponse> {
        val call =
ApiClient.apiInterface.createDataset(nama, uname)

        call?.enqueue(object :
retrofit2.Callback<DatasetResponse?> {
            override fun onResponse(
                call: Call<DatasetResponse?>,
                response: Response<DatasetResponse?>
            ) {
                dsMld.postValue(response.body())
            }

            override fun onFailure(call:
Call<DatasetResponse?>, t: Throwable) {
                val obj = DatasetResponse(success = 0, msg =
t.message, data = null)
                dsMld.postValue(obj)
            }

        })

        return dsMld
    }

    fun uploadGambar(
        token: String?,
        gambar: File,
```

```
        data: Array<String?>
    ): MutableLiveData<DatasetResponse> {
        val gambarPart: MultipartBody.Part =
MultipartBody.Part.createFormData(
            "gambar",
            gambar.name,

gambar.asRequestBody("image/*".toMediaTypeOrNull())
        )

        val plainType = "text/plain;charset=utf-
8".toMediaTypeOrNull()
        val bearer = "Bearer $token"

        val call = ApiClient.apiInterface.uploadGambar(
            bearer,
            gambar = gambarPart,
            uname = data[0]?.toRequestBody(plainType),
            dataset = data[1]?.toRequestBody(plainType),
            subdataset = data[2]?.toRequestBody(plainType)
        )

        call?.enqueue(object :
retrofit2.Callback<DatasetResponse?> {
            override fun onResponse(
                call: Call<DatasetResponse?>,
                response: Response<DatasetResponse?>
            ) {
                gMld.postValue(response.body())
            }

            override fun onFailure(call:
Call<DatasetResponse?>, t: Throwable) {
                val obj = DatasetResponse(success = 0, msg =
t.message, data = null)
                gMld.postValue(obj)
            }

        })

        return gMld
    }

    fun createSubdataset(
        nama: String,
        dataset: String,
```

```
        uname: String
    ): MutableLiveData<DatasetResponse> {
        val call =
ApiClient.apiInterface.createSubDataset(nama, dataset,
uname)

        call?.enqueue(object :
retrofit2.Callback<DatasetResponse?> {
            override fun onResponse(
                call: Call<DatasetResponse?>,
                response: Response<DatasetResponse?>
            ) {
                sdsMld.postValue(response.body())
            }

            override fun onFailure(call:
Call<DatasetResponse?>, t: Throwable) {
                val obj = DatasetResponse(success = 0, msg =
t.message, data = null)
                sdsMld.postValue(obj)
            }

        })

        return sdsMld
    }

    fun listDataset(uname: String):
MutableLiveData<ListDatasetResponse> {
        val call = ApiClient.apiInterface.listDataset(uname)

        call?.enqueue(object :
retrofit2.Callback<ListDatasetResponse?> {
            override fun onResponse(
                call: Call<ListDatasetResponse?>,
                response: Response<ListDatasetResponse?>
            ) {
                ldsMld.postValue(response.body())
            }

            override fun onFailure(call:
Call<ListDatasetResponse?>, t: Throwable) {
                val obj = ListDatasetResponse(success = 0,
msg = t.message, data = null)
                ldsMld.postValue(obj)
            }
```

```
        })

        return ldsMld
    }

    fun listSubdataset(dataset: String, uname: String):
MutableLiveData<ListDatasetResponse> {
        val call =
ApiClient.apiInterface.listSubDataset(uname, dataset)

        call?.enqueue(object :
retrofit2.Callback<ListDatasetResponse?> {
            override fun onResponse(
                call: Call<ListDatasetResponse?>,
                response: Response<ListDatasetResponse?>
            ) {
                lsdsMld.postValue(response.body())
            }

            override fun onFailure(call:
Call<ListDatasetResponse?>, t: Throwable) {
                val obj = ListDatasetResponse(success = 0,
msg = t.message, data = null)
                lsdsMld.postValue(obj)
            }


        })

        return lsdsMld
    }

    fun listGambar(
        subdataset: String,
        dataset: String,
        uname: String,
        preproc: Boolean
    ): MutableLiveData<ListDatasetResponse> {
        val call =
            if (!preproc)

ApiClient.apiInterface.listGambar(subdataset, dataset,
uname)
            else
```

```
ApiClient.apiInterface.listPreproc(subdataset, dataset,
uname)

        call?.enqueue(object :
retrofit2.Callback<ListDatasetResponse?> {
            override fun onResponse(
                call: Call<ListDatasetResponse?>,
                response: Response<ListDatasetResponse?>
            ) {
                lgMld.postValue(response.body())
            }

            override fun onFailure(call:
Call<ListDatasetResponse?>, t: Throwable) {
                val obj = ListDatasetResponse(success = 0,
msg = t.message, data = null)
                lgMld.postValue(obj)
            }


        })

        return lgMld
    }
}
```

- ***TokenManager.kt***

```
package com.android.datasetflask.util

import android.content.Context
import android.content.SharedPreferences
import com.android.datasetflask.R

class TokenManager(ctx: Context) {
    private var preferences: SharedPreferences =
ctx.getSharedPreferences(
        ctx.getString(R.string.app_name),
        Context.MODE_PRIVATE
    )

    companion object {
        const val TOKEN = "token"
    }

    fun store(token: String) {
```

```
        var editor = preferences.edit()
        editor.putString(TOKEN, token)
        editor.apply()
    }

    fun get(): String? {
        return preferences.getString(TOKEN, null)
    }

    fun destroy() {
        var editor = preferences.edit()
        editor.remove(TOKEN)
        editor.apply()
    }
}
```

- ***GambarAdapter.kt***

```
package com.android.datasetflask.view.adapter

import android.content.Context
import android.view.LayoutInflater
import android.view.View
import android.view.ViewGroup
import android.widget.ImageView
import androidx.recyclerview.widget.RecyclerView
import com.android.datasetflask.R
import com.android.datasetflask.api.ApiClient
import com.bumptech.glide.Glide

class GambarAdapter(private val data1: Array<String?>,
private val data: ArrayList<String>, private val ctx:
Context):
RecyclerView.Adapter<GambarAdapter.GambarViewHolder>() {
    var mode = "gambar"

    override fun onCreateViewHolder(
        parent: ViewGroup,
        viewType: Int
    ): GambarViewHolder {
        val view =
LayoutInflater.from(parent.context).inflate(R.layout.item_ga
mbar, parent, false)
        return GambarViewHolder(view)
    }
```

```
    override fun onBindViewHolder(holder: GambarViewHolder,
position: Int) {
        val loc = data[position]
        val url =
"${ApiClient.ServerURL}$mode/${data1[0]}/${data1[1]}/${data1
[2]}/$loc"
        Glide.with(ctx).load(url)
            .centerInside()
            .placeholder(R.drawable.ic_baseline_image_24)
            .into(holder.gambar)
    }

    override fun getItemCount(): Int = data.size

    class GambarViewHolder(item: View):
RecyclerView.ViewHolder(item) {
        var gambar: ImageView

        init {
            gambar = item.findViewById(R.id.gambar)
        }
    }
}
```

- ***GambarActivity.kt***

```
package com.android.datasetflask.view

import android.annotation.SuppressLint
import android.app.Activity
import android.content.Intent
import android.database.Cursor
import android.net.Uri
import android.os.Bundle
import android.provider.OpenableColumns
import android.util.Log
import android.widget.Button
import android.widget.Switch
import android.widget.Toast
import androidx.appcompat.app.AppCompatActivity
import androidx.appcompat.app.AppCompatDelegate
import androidx.lifecycle.ViewModelProvider
import androidx.recyclerview.widget.GridLayoutManager
import androidx.recyclerview.widget.RecyclerView
import com.android.datasetflask.R
import com.android.datasetflask.model.UserDataResponse
import com.android.datasetflask.util.Constant
```

```
import com.android.datasetflask.util.TokenManager
import com.android.datasetflask.view.adapter.GambarAdapter
import com.android.datasetflask.viewmodel.GambarViewModel
import com.github.dhaval2404.imagepicker.ImagePicker
import com.google.gson.Gson
import java.io.File
import java.io.FileOutputStream
import java.io.InputStream
import java.io.OutputStream


class GambarActivity : AppCompatActivity() {
    private lateinit var staticData: Array<String?>
    private lateinit var gvm: GambarViewModel
    private lateinit var gambarAdapter: GambarAdapter

    @SuppressLint("UseSwitchCompatOrMaterialCode")
    override fun onCreate(savedInstanceState: Bundle?) {
        super.onCreate(savedInstanceState)
        setContentView(R.layout.activity_gambar)



AppCompatDelegate.setDefaultNightMode(AppCompatDelegate.MODE
_NIGHT_NO)

        val sp =
getSharedPreferences(Constant.MY_SHARED_PREF, MODE_PRIVATE)

        val userJson =
sp.getString(Constant.USER_SHARED_PREF, null)
        if (userJson == null)
            startActivity(Intent(this,
MainActivity::class.java))
        val userActive = Gson().fromJson(userJson,
UserDataResponse::class.java)

        val dataset = intent.getStringExtra("dataset")
        val subdataset = intent.getStringExtra("subdataset")
        if (subdataset == null || dataset == null)
            onBackPressed()

        gvm =
ViewModelProvider(this).get(GambarViewModel::class.java)

        val createButton = findViewById<Button>(R.id.create)
```

```
        val backButton = findViewById<Button>(R.id.back)
        val preSwitch =
findViewById<Switch>(R.id.preprocess)
        val gambarList =
findViewById<RecyclerView>(R.id.dataset)

        val dataAdapter = ArrayList<String>()
        staticData = arrayOf(
            userActive.uname,
            dataset,
            subdataset
        )

        gambarAdapter = GambarAdapter(staticData,
dataAdapter, this)
        gambarList.adapter = gambarAdapter
        gambarList.layoutManager = GridLayoutManager(this,
2)

        val lgLd = gvm.listGambar(subdataset!!, dataset!!,
userActive.uname)

        lgLd.observe(this, { lg ->
            lg.let {
                if (it.success == 1 && it.data != null) {
                    dataAdapter.clear()
                    dataAdapter.addAll(it.data)
                    gambarAdapter.notifyDataSetChanged()
                }
            }
        })

        preSwitch.setOnCheckedChangeListener { _, isChecked
->
            gambarAdapter.mode =
                if (isChecked)
                    "preprocess"
                else
                    "gambar"

            gvm.listGambar(subdataset, dataset,
userActive.uname, isChecked)
        }

        createButton.setOnClickListener {
            ImagePicker.with(this).compress(4096)
```

```
                .galleryMimeTypes(
                    mimeTypes = arrayOf(
                        "image/jpg",
                        "image/jpeg"
                    )
                )
                .start()
        }

        backButton.setOnClickListener {
            onBackPressed()
        }
    }

    @Suppress("DEPRECATION")
    override fun onActivityResult(requestCode: Int,
resultCode: Int, data: Intent?) {
        super.onActivityResult(requestCode, resultCode,
data)
        if (resultCode == Activity.RESULT_OK) {
            val uri: Uri = data?.data!!
            val fileName = getFileName(uri)
            val fileData = fileName?.split(".")
            Log.v("filename", "$fileName")
            val inps: InputStream? =
contentResolver.openInputStream(uri)
            val file = File.createTempFile(fileData![0],
".${fileData[1]}", cacheDir)
            val out: OutputStream = FileOutputStream(file)
            val buf = ByteArray(1024)
            var len: Int
            while (inps?.read(buf).also { len = it!! }!! >
0) {
                out.write(buf, 0, len)
            }
            out.close()
            inps?.close()
            val tokenManager = TokenManager(this)
            val gLd = gvm.uploadGambar(tokenManager.get(),
file, staticData)
            gLd.observe(this, {
                if (it.success == 1 && it.data != null) {
                    gvm.listGambar(
                        staticData[2]!!,
                        staticData[1]!!,
                        staticData[0]!!,
```

```
                        gambarAdapter.mode == "preprocess"
                    )
                } else {
                    Toast.makeText(this, it.msg,
Toast.LENGTH_SHORT).show()
                }
            })
        } else if (resultCode == ImagePicker.RESULT_ERROR) {
            Toast.makeText(this, ImagePicker.getError(data),
Toast.LENGTH_SHORT).show()
        }
    }

    private fun getFileName(uri: Uri): String? {
        var result: String? = null
        if (uri.scheme == "content") {
            val cursor: Cursor? = contentResolver.query(uri,
null, null, null, null)
            try {
                if (cursor != null && cursor.moveToFirst())
{
                    result =
cursor.getString(cursor.getColumnIndex(OpenableColumns.DISPL
AY_NAME))
                }
            } finally {
                cursor?.close()
            }
        }
        if (result == null) {
            result = uri.path
            val cut = result!!.lastIndexOf('/')
            if (cut != -1) {
                result = result.substring(cut + 1)
            }
        }
        return result
    }
}
```

- ***GambarViewModel.kt***

```
package com.android.datasetflask.viewmodel

import androidx.lifecycle.LiveData
import androidx.lifecycle.ViewModel
import com.android.datasetflask.model.DatasetResponse
```

```
import com.android.datasetflask.model.ListDatasetResponse
import com.android.datasetflask.repo.DatasetRepo
import java.io.File

class GambarViewModel : ViewModel() {
    fun uploadGambar(
        token: String?,
        gambar: File,
        data: Array<String?>
    ): LiveData<DatasetResponse> {
        return DatasetRepo.uploadGambar(token, gambar, data)
    }

    fun listGambar(
        subdataset: String,
        dataset: String,
        uname: String,
        preproc: Boolean = false
    ): LiveData<ListDatasetResponse> {
        return DatasetRepo.listGambar(subdataset, dataset,
uname, preproc)
    }
}
```