LAMPIRAN

Lampiran 1-Surat Keterangan Bebas Plagiat

UNIVERSITAS DARMA PERSADA
UPT PERPUSTAKAAN
Gedung Rektorat Lantai 3,
Jl.Taman Malaka Selatan, Pondok Kelapa – Jakarta Timur 13450

SURAT KETERANGAN
HASIL PENGECEKAN TURNITIN

UPT Perpustakaan Universitas Darma Persada menerangkan telah selesai melakukan pemeriksaan duplikasi/*similarity* menggunakan perangkat lunak Turnitin terhadap hasil karya sebagai berikut:

Judul : PERBANDINGAN ARIMA DAN INFORMER UNTUK FORECASTING KEJADIAN GEMPA DI SELAT SUNDA
Penulis : **Lukman Farid**
NIM : **2021230012**
Tgl pemeriksaan : 11 Februari 2025

Dengan hasil Tingkat Kesamaan (*similarity index*) **14%**

Demikian Surat Keterangan kami buat, untuk dipergunakan sebagaimana mestinya.

Jakarta, 11 Februari 2025

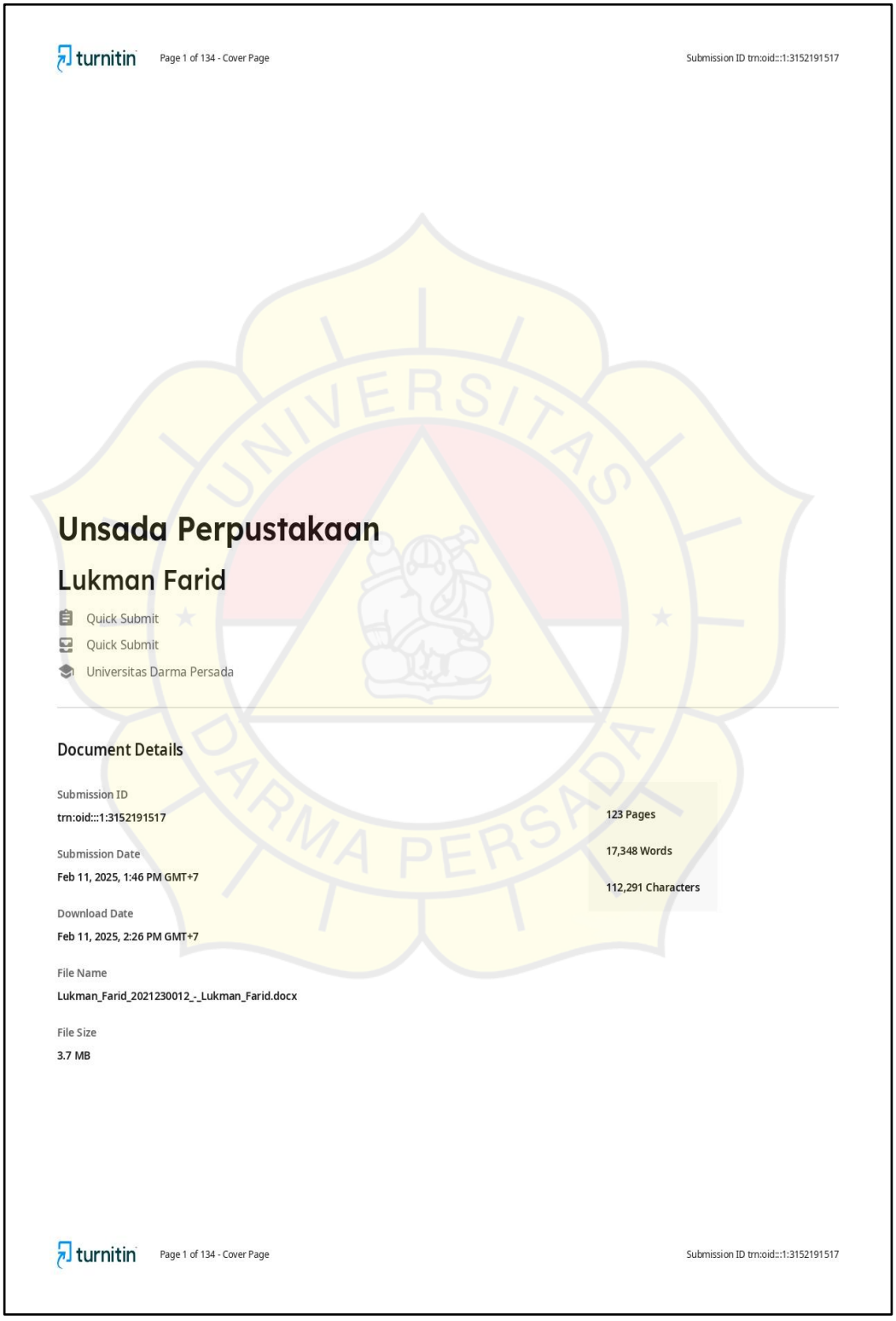
Ka.UPT Perpustakaan Unsada

Yus Rusmiyati, SS., MM

Batas maksimal similarity 30% untuk Fakultas Sastra dan Ekonomi

Batas maksimal similarity 25% untuk Fakultas Teknik, Kelautan dan Pasca Sarjana

Lampiran 2 Surat Hasil Turnitin



turnitin Page 1 of 134 - Cover Page Submission ID trn:oid:::1:3152191517

Unsada Perpustakaan

Lukman Farid

Quick Submit

Quick Submit

Universitas Darma Persada

Document Details

Submission ID
trn:oid:::1:3152191517

Submission Date
Feb 11, 2025, 1:46 PM GMT+7

Download Date
Feb 11, 2025, 2:26 PM GMT+7

File Name
Lukman_Farid_2021230012_-_Lukman_Farid.docx

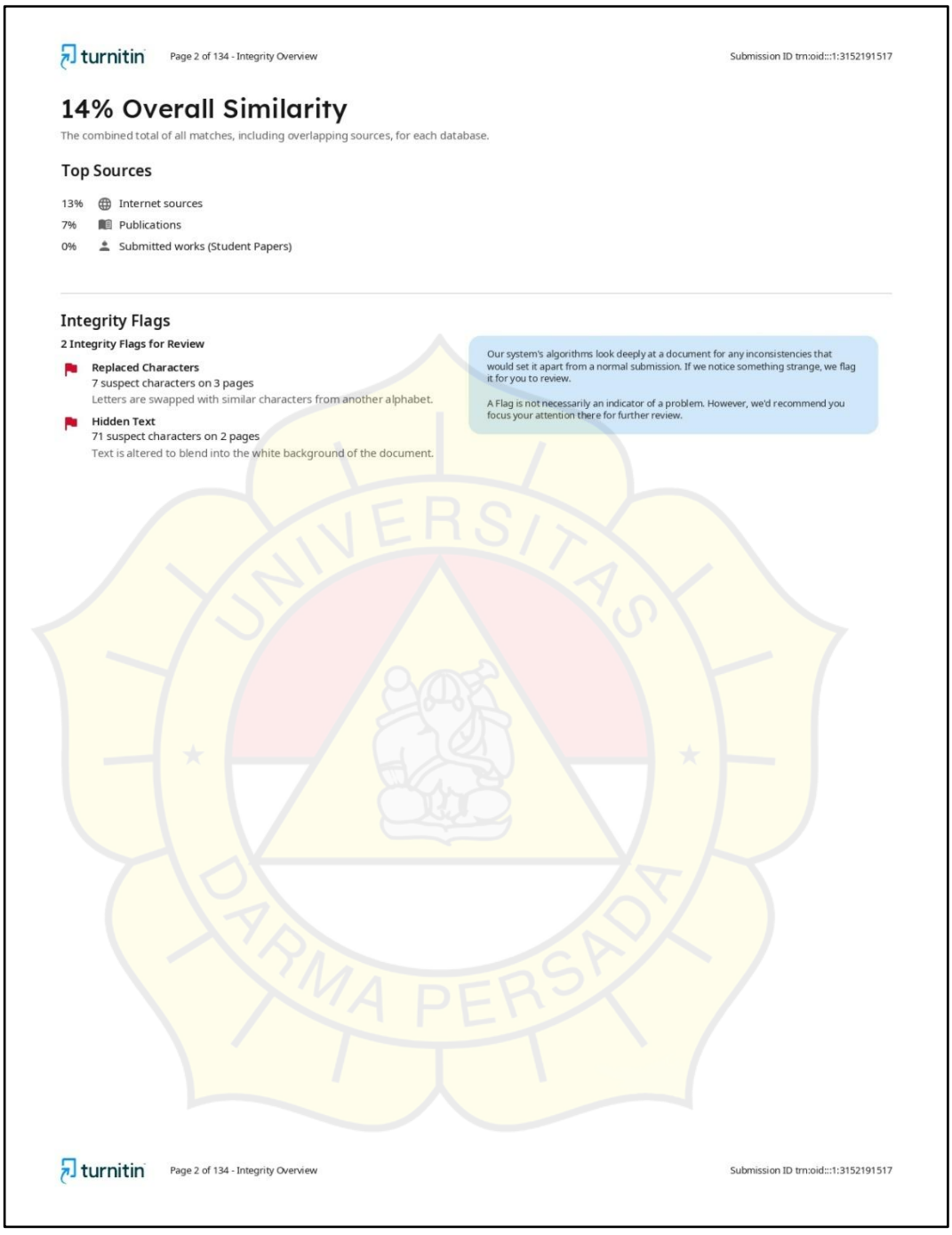
File Size
3.7 MB

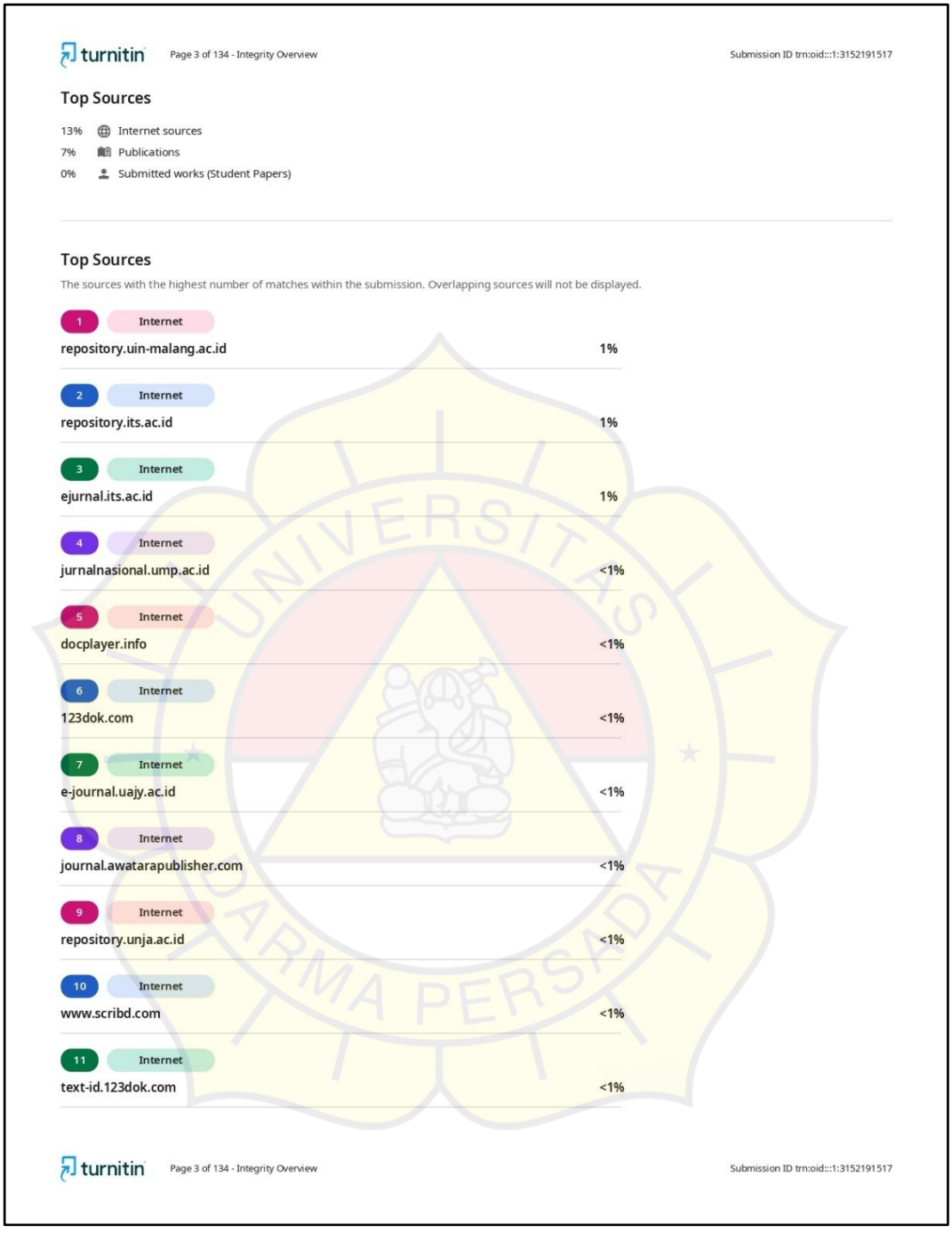
123 Pages

17,348 Words

112,291 Characters

turnitin Page 1 of 134 - Cover Page Submission ID trn:oid:::1:3152191517


turnitin
Page 2 of 134 - Integrity Overview
Submission ID trn:oid:::1:3152191517
14% Overall Similarity
The combined total of all matches, including overlapping sources, for each database.
Top Sources
13% Internet sources
7% Publications
0% Submitted works (Student Papers)
Integrity Flags
2 Integrity Flags for Review
Replaced Characters
7 suspect characters on 3 pages
Letters are swapped with similar characters from another alphabet.
Hidden Text
71 suspect characters on 2 pages
Text is altered to blend into the white background of the document.
Our system's algorithms look deeply at a document for any inconsistencies that would set it apart from a normal submission. If we notice something strange, we flag it for you to review.
A Flag is not necessarily an indicator of a problem. However, we'd recommend you focus your attention there for further review.
turnitin
Page 2 of 134 - Integrity Overview
Submission ID trn:oid:::1:3152191517



turnitin Page 3 of 134 - Integrity Overview Submission ID trn:oid:::1:3152191517

Top Sources

13% Internet sources
7% Publications
0% Submitted works (Student Papers)

Top Sources

The sources with the highest number of matches within the submission. Overlapping sources will not be displayed.

1	Internet	repository.uin-malang.ac.id 1%
2	Internet	repository.its.ac.id 1%
3	Internet	ejurnal.its.ac.id 1%
4	Internet	jurnalnasional.ump.ac.id <1%
5	Internet	docplayer.info <1%
6	Internet	123dok.com <1%
7	Internet	e-journal.uajy.ac.id <1%
8	Internet	journal.awatarapublisher.com <1%
9	Internet	repository.unja.ac.id <1%
10	Internet	www.scribd.com <1%
11	Internet	text-id.123dok.com <1%

turnitin Page 3 of 134 - Integrity Overview Submission ID trn:oid:::1:3152191517

12	Internet	
core.ac.uk		<1%
13	Internet	
doaj.org		<1%
14	Publication	
Meili Christabel, Dedi Trisnawarman. "Implementasi Algoritma Arima Dalam Pred...		<1%
15	Internet	
artikelpendidikan.id		<1%
16	Internet	
repositori.usu.ac.id		<1%
17	Internet	
garuda.kemdikbud.go.id		<1%
18	Publication	
Wilianto Wilianto, Yuliana Yuliana, Albert Suwandhi, Jimmy Jimmy, Jati Putra. "Pe...		<1%
19	Internet	
id.scribd.com		<1%
20	Publication	
Fanny Ramadhani, Dian Septiana, Sisti Nadia Amalia, Putri Maulidina Fadilah, An...		<1%
21	Internet	
repository.unej.ac.id		<1%
22	Internet	
id.123dok.com		<1%
23	Internet	
jurnal.unsur.ac.id		<1%
24	Internet	
toffeedev.com		<1%
25	Internet	
dqlab.id		<1%

26	Internet	
fr.scribd.com		<1%
27	Internet	
juandomingofarnos.wordpress.com		<1%
28	Internet	
repository.amikom.ac.id		<1%
29	Internet	
doku.pub		<1%
30	Internet	
repo.itera.ac.id		<1%
31	Publication	
Tan Eric Wijaya, Yohana Tri Widayati, Yusup Yusup. "Optimalisasi Performa Penju...		<1%
32	Internet	
wardanhumaira02.blogspot.com		<1%
33	Internet	
community.algostudio.net		<1%
34	Internet	
elibrary.stipram.ac.id		<1%
35	Internet	
eprints.utdi.ac.id		<1%
36	Internet	
kumparan.com		<1%
37	Internet	
media.neliti.com		<1%
38	Internet	
www.sridianti.com		<1%
39	Publication	
Khairunnisa Raihani. "Pembuatan Website Penjualan Toko Aksesoris Dengan Me...		<1%

turnitin Page 6 of 134 - Integrity Overview Submission ID trn:oid:::1:3152191517

40	Internet	
bpbd.jogjaprov.go.id		<1%
41	Internet	
repository.usd.ac.id		<1%
42	Internet	
www.mathsjournal.com		<1%
43	Publication	
Yusuf Kurnia, Ellysha Dwiyanthi Kusuma, Lianny Wydiastuty Kusuma, Suwitno, W...		<1%
44	Internet	
www.researchgate.net		<1%
45	Internet	
repository.unifa.ac.id		<1%
46	Internet	
fliphtml5.com		<1%
47	Internet	
hes-gotappointment-newspaper.icu		<1%
48	Internet	
repositori.uin-alauddin.ac.id		<1%
49	Internet	
jurnal.fp.unila.ac.id		<1%
50	Internet	
library.binus.ac.id		<1%
51	Internet	
mubadalah.id		<1%
52	Internet	
tirto.id		<1%
53	Internet	
www.kompas.com		<1%

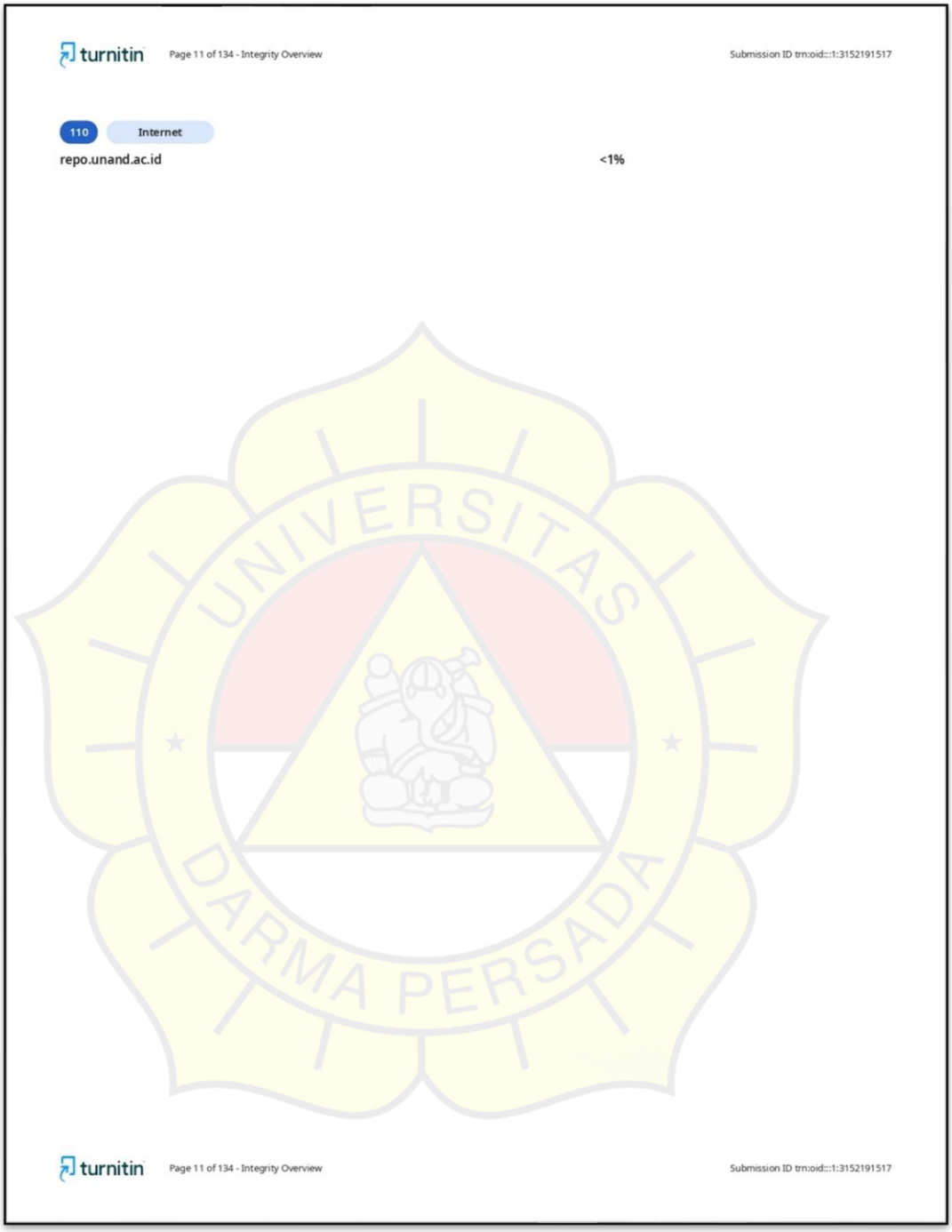
turnitin Page 6 of 134 - Integrity Overview Submission ID trn:oid:::1:3152191517

54	Internet	
eprints.akakom.ac.id		<1%
55	Publication	
Frisda Dita Isnaini, Yisti Vita Via, Eka Prakarsa Mandyartha. "PENERAPAN HOLT-W...		<1%
56	Publication	
Muhammad Teguh Febrian, Nuriadi Manurung, Pristiyanilicia Putri. "SISTEM PER...		<1%
57	Internet	
digilib.unila.ac.id		<1%
58	Internet	
dspace.uii.ac.id		<1%
59	Internet	
eprints.undip.ac.id		<1%
60	Internet	
es.scribd.com		<1%
61	Internet	
repository.unp.ac.id		<1%
62	Internet	
text.123docz.net		<1%
63	Internet	
www.mask-ryant.co.cc		<1%
64	Internet	
jurnal-online.um.ac.id		<1%
65	Internet	
kc.umn.ac.id		<1%
66	Internet	
repository.ipb.ac.id		<1%
67	Internet	
simpel.its.ac.id		<1%

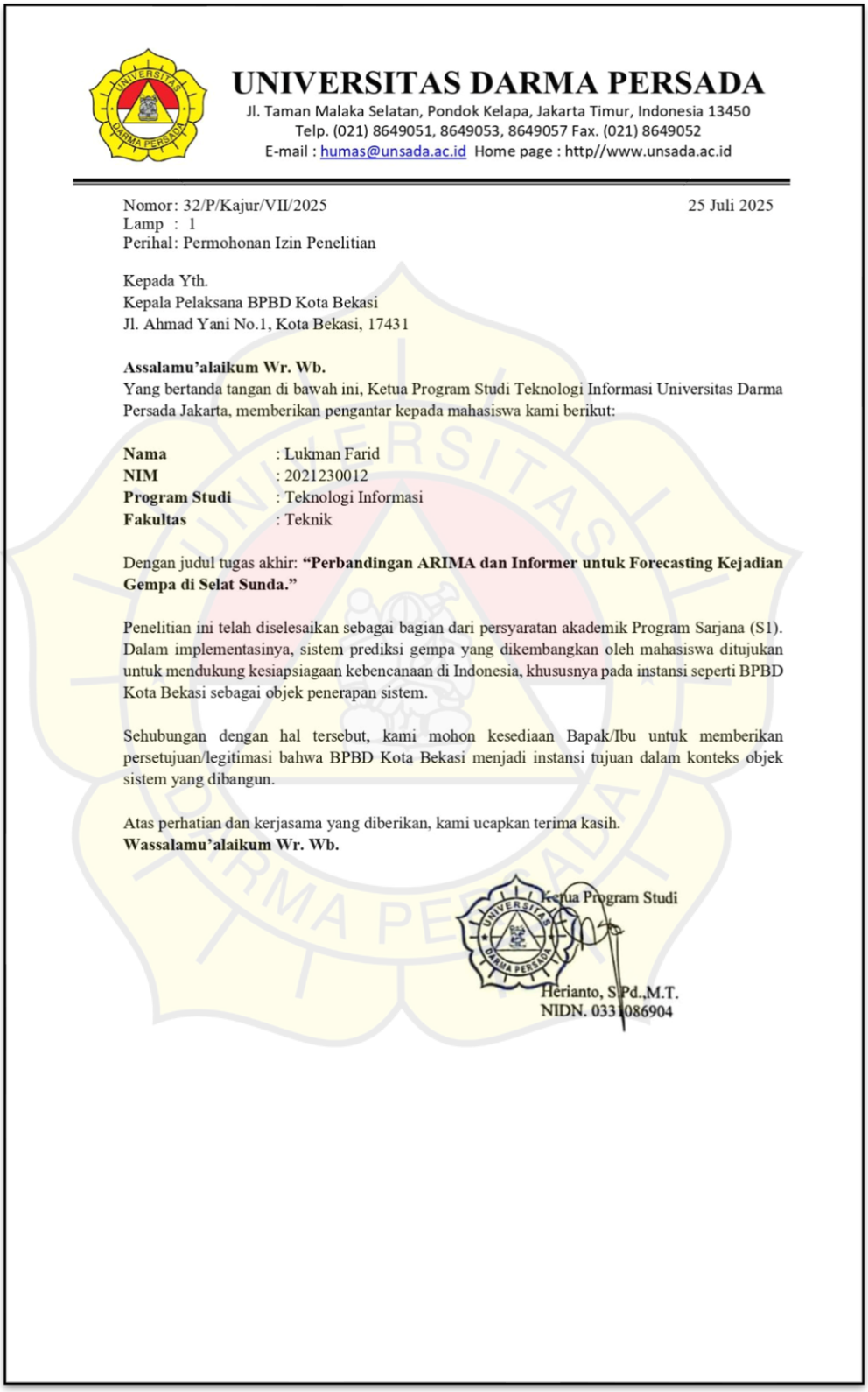
68	Internet	
sudardjattanusukma.wordpress.com		<1%
69	Publication	
Hary Murcahyanto, Mohzana, Hartini Haritani. "Evaluation of Planning for The C...		<1%
70	Publication	
Zian Asti Dwiyanti, Roni Habibi. "Penerapan Algoritma K-Means untuk Mengukur ...		<1%
71	Internet	
adoc.pub		<1%
72	Internet	
docobook.com		<1%
73	Internet	
dokumen.tips		<1%
74	Internet	
ejournal.kemenperin.go.id		<1%
75	Internet	
etheses.uin-malang.ac.id		<1%
76	Internet	
implantgigi.com		<1%
77	Internet	
journal2.um.ac.id		<1%
78	Internet	
komunikasi.us		<1%
79	Internet	
lipi.go.id		<1%
80	Internet	
repository.untar.ac.id		<1%
81	Internet	
repository.widyatama.ac.id		<1%

82	Internet	
www.aisi555.com		<1%
83	Internet	
www.laptophia.com		<1%
84	Publication	
Bajeng Nurul Widyaningrum, Lingga Kurnia Ramadhani. "Penerapan Metode Resi...		<1%
85	Publication	
Malim Muhammad, Harjono Harjono, Lukmanul Akhsani. "Peramalan Mahasiswa ...		<1%
86	Publication	
Muhammad Dwison Alizah, Arifin Nugroho, Ummu Radiyah, Windu Gata. "Sentim...		<1%
87	Publication	
Nadia Elisabet Br. Hutapea, Lisya Junus, Putri Puspita Ningrum, Hani Wahyunida I...		<1%
88	Publication	
Oki Dermawan, Putri Octavia, Wan Jamaludin. "Optimizing Human Resource Man...		<1%
89	Publication	
Sulaeman Nurman, Muhammad Nusrang, Sudarmin. "Analysis of Rice Productio...		<1%
90	Internet	
businessdocbox.com		<1%
91	Internet	
elibrary.unikom.ac.id		<1%
92	Internet	
eprints.mdp.ac.id		<1%
93	Internet	
eprints.unpam.ac.id		<1%
94	Internet	
erepository.uwks.ac.id		<1%
95	Internet	
export.arxiv.org		<1%

96	Internet	
gojleng.wordpress.com		<1%
97	Internet	
id.bitdegree.org		<1%
98	Internet	
johannessimatupang.wordpress.com		<1%
99	Internet	
katalog.ukdw.ac.id		<1%
100	Internet	
lib.ui.ac.id		<1%
101	Internet	
migracionesforzadas.org		<1%
102	Internet	
prosiding.unimus.ac.id		<1%
103	Internet	
repository.stiesia.ac.id		<1%
104	Internet	
repository.uisu.ac.id		<1%
105	Internet	
www.e-journal.stmiklombok.ac.id		<1%
106	Internet	
www.freediverindonesia.com		<1%
107	Publication	
Ega Evinda Putri, Agung Putra Yunanda. "INOVASI LAYANAN KESEHATAN ISLAMI: ...		<1%
108	Internet	
geologi2016.blogspot.com		<1%
109	Internet	
lib.ibs.ac.id		<1%


turnitin
Page 11 of 134 - Integrity Overview
Submission ID trn:oid:::1:3152191517
110
Internet
repo.unand.ac.id
<1%
turnitin
Page 11 of 134 - Integrity Overview
Submission ID trn:oid:::1:3152191517

Lampiran 3 Surat Permohonan Kepada Instansi



UNIVERSITAS DARMA PERSADA

Jl. Taman Malaka Selatan, Pondok Kelapa, Jakarta Timur, Indonesia 13450
Telp. (021) 8649051, 8649053, 8649057 Fax. (021) 8649052
E-mail : humas@unsada.ac.id Home page : http//www.unsada.ac.id

Nomor: 32/P/Kajur/VII/2025 25 Juli 2025
Lamp : 1
Perihal: Permohonan Izin Penelitian

Kepada Yth.
Kepala Pelaksana BPBD Kota Bekasi
Jl. Ahmad Yani No.1, Kota Bekasi, 17431

Assalamu'alaikum Wr. Wb.
Yang bertanda tangan di bawah ini, Ketua Program Studi Teknologi Informasi Universitas Darma Persada Jakarta, memberikan pengantar kepada mahasiswa kami berikut:

Nama : Lukman Farid
NIM : 2021230012
Program Studi : Teknologi Informasi
Fakultas : Teknik

Dengan judul tugas akhir: **"Perbandingan ARIMA dan Informer untuk Forecasting Kejadian Gempa di Selat Sunda."**

Penelitian ini telah diselesaikan sebagai bagian dari persyaratan akademik Program Sarjana (S1). Dalam implementasinya, sistem prediksi gempa yang dikembangkan oleh mahasiswa ditujukan untuk mendukung kesiapsiagaan kebencanaan di Indonesia, khususnya pada instansi seperti BPBD Kota Bekasi sebagai objek penerapan sistem.

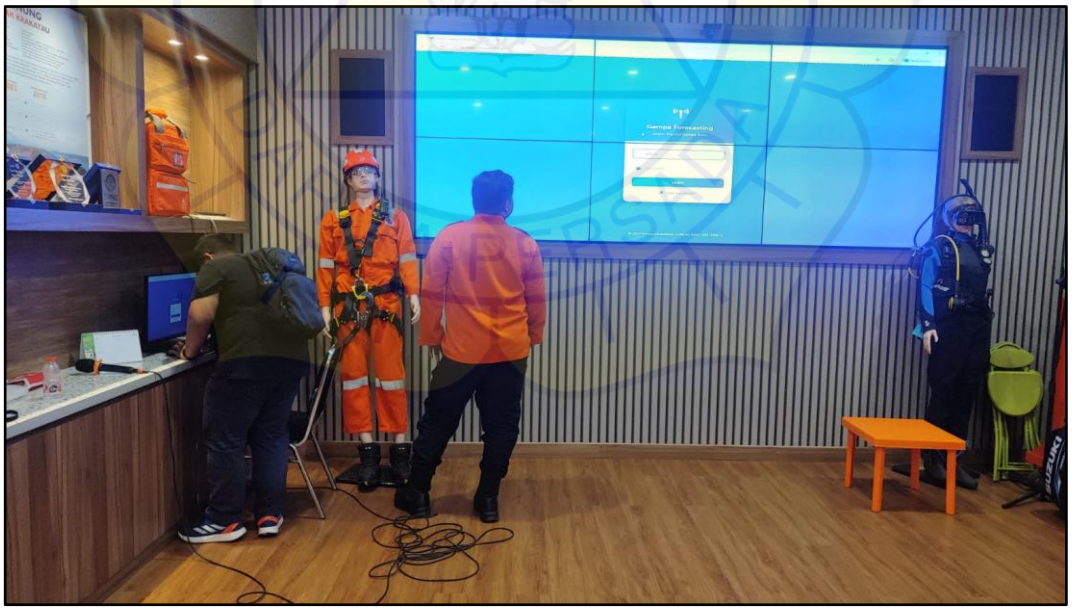
Sehubungan dengan hal tersebut, kami mohon kesediaan Bapak/Ibu untuk memberikan persetujuan/legitimasi bahwa BPBD Kota Bekasi menjadi instansi tujuan dalam konteks objek sistem yang dibangun.

Atas perhatian dan kerjasama yang diberikan, kami ucapkan terima kasih.
Wassalamu'alaikum Wr. Wb.

Ketua Program Studi

Herianto, S.Pd.,M.T.
NIDN. 0331086904

Lampiran 4 Dokumentasi Bersama Instansi



Lampiran 5 *Source Code* Aplikasi Utama

```
from flask import Flask, render_template, request, jsonify,
redirect, url_for, flash, session
import pandas as pd
import os
from flask_sqlalchemy import SQLAlchemy
from flask_login import LoginManager, UserMixin, login_user,
login_required, logout_user, current_user
from datetime import datetime, timedelta
from functools import wraps
from statsmodels.tsa.arima.model import ARIMA
import matplotlib.pyplot as plt
import requests
import io
import base64
from sklearn.metrics import mean_absolute_error,
mean_squared_error
import numpy as np


# Setup Flask
app = Flask(__name__)

# Konfigurasi database
app.config['SQLALCHEMY_DATABASE_URI'] = 'sqlite:///app.db'  #
Database SQLite untuk log
app.config['SQLALCHEMY_TRACK_MODIFICATIONS'] = False
app.secret_key = 'your_secret_key'

# Set waktu kedaluwarsa sesi menjadi 30 menit
app.permanent_session_lifetime = timedelta(minutes=30)  # Sesi
akan kedaluwarsa setelah 30 menit


# Inisialisasi SQLAlchemy dan LoginManager
db = SQLAlchemy(app)
login_manager = LoginManager()
login_manager.init_app(app)
login_manager.login_view = 'login'


# Model untuk mencatat log aktivitas user
class UserLog(db.Model):
    id = db.Column(db.Integer, primary_key=True)
    username = db.Column(db.String(80), nullable=False)
```

```
    activity = db.Column(db.String(200), nullable=False)
    timestamp = db.Column(db.DateTime, default=datetime.utcnow)

    def __repr__(self):
        return f'<UserLog {self.username} - {self.activity}>'

# Model User untuk login
class User(UserMixin, db.Model):
    id = db.Column(db.Integer, primary_key=True,
autoincrement=True)
    username = db.Column(db.String(80), unique=True,
nullable=False)
    password = db.Column(db.String(200), nullable=False)
    role = db.Column(db.String(50), nullable=False,
default='user')

# Fungsi user_loader
@login_manager.user_loader
def load_user(user_id):
    return User.query.get(user_id)

# Membuat tabel jika belum ada
with app.app_context():
    db.create_all()

    # Tambahkan user default jika belum ada
    if not User.query.filter_by(username='admin').first():
        admin_user = User(username='admin', password='admin123',
role='admin')
        db.session.add(admin_user)
        db.session.commit()

# Fungsi untuk mencatat log aktivitas user
def log_user_activity(username, activity):
    new_log = UserLog(username=username, activity=activity)
    db.session.add(new_log)
    db.session.commit()

# Middleware untuk membatasi akses berdasarkan peran
def role_required(role):
    def decorator(func):
        @wraps(func)
        def wrapped_view(*args, **kwargs):
            if current_user.role != role:
```

```
                return "Akses ditolak: Anda tidak memiliki izin
untuk mengakses halaman ini."
            return func(*args, **kwargs)
        return wrapped_view
    return decorator

# Halaman Utama
@app.route('/')
@login_required
def index():
    if current_user.role == 'admin':
        return redirect(url_for('admin_dashboard'))
    return redirect(url_for('user_dashboard'))

# Fungsi login
@app.route('/login', methods=['GET', 'POST'])
def login():
    if current_user.is_authenticated:
        return redirect(url_for('index'))  # Jika sudah login,
arahkan ke halaman utama (index)

    if request.method == 'POST':
        username = request.form['username']
        password = request.form['password']

        # Cek kredensial
        user = User.query.filter_by(username=username).first()
        if user and user.password == password:
            login_user(user)
            log_user_activity(username, 'Logged in')  # Catat
aktivitas login
            return redirect(url_for('index'))
        else:
            return 'Invalid username or password', 401

    return render_template('login.html')

# Fungsi untuk menangani lupa password
@app.route('/forgot_password', methods=['GET', 'POST'])
def forgot_password():
    if request.method == 'POST':
        username = request.form['username']
        user = User.query.filter_by(username=username).first()

        if user:
```

```
            # Jika username ditemukan, tampilkan password asli
(untuk logika dasar)
            return render_template('forgot_password.html',
success=f'Password Anda adalah: {user.password}')
        else:
            # Jika username tidak ditemukan, tampilkan pesan
kesalahan
            return render_template('forgot_password.html',
error='Username tidak ditemukan.')

    return render_template('forgot_password.html')

# Fungsi logout
@app.route('/logout')
@login_required
def logout():
    log_user_activity(current_user.username, 'Logged out')  #
Catat aktivitas logout
    logout_user()
    session.clear()  # Hapus semua data sesi
    return redirect(url_for('login'))  # Redirect ke halaman login
setelah logout

# Dashboard Admin
@app.route('/admin')
@login_required
@role_required('admin')
def admin_dashboard():
    logs = UserLog.query.all()
    return render_template('admin_dashboard.html', logs=logs)

# Dashboard User
@app.route('/dashboard')
@login_required
@role_required('user')
def user_dashboard():
    return render_template('user_dashboard.html')

DATA_FOLDER = './data'

# Definisikan path file CSV yang digunakan untuk prediksi
UPLOAD_FILEPATH = os.path.join(DATA_FOLDER, 'uploaded_data.csv')

if not os.path.exists(DATA_FOLDER):
    os.makedirs(DATA_FOLDER)
```

```python
# CRUD User
@app.route('/manage_users', methods=['GET', 'POST'])
@login_required
@role_required('admin')
def manage_users():
    if request.method == 'POST':
        username = request.form['username']
        password = request.form['password']
        role = request.form['role']

        # Tambah user baru
        new_user = User(username=username, role=role,
password=password)
        db.session.add(new_user)
        db.session.commit()
        log_user_activity(current_user.username, f'Added new user:
{username}')
        return redirect(url_for('manage_users'))

    users = User.query.all()
    return render_template('manage_user.html', users=users)

# Route to add a user
@app.route('/add_user', methods=['POST'])
@login_required
@role_required('admin')
def add_user():
    username = request.form['username']
    password = request.form['password']
    role = request.form['role']

    if User.query.filter_by(username=username).first():
        flash('Username already exists!', 'error')
        return redirect(url_for('manage_users'))

    new_user = User(username=username, password=password,
role=role)

    db.session.add(new_user)
    db.session.commit()
    print(f"User added with ID: {new_user.id}")  # Debugging
    flash('User added successfully!', 'success')
    return redirect(url_for('manage_users'))
```

```
@app.route('/edit_user/<user_id>', methods=['GET', 'POST'])
@login_required
@role_required('admin')
def edit_user(user_id):
    user = User.query.get(user_id)
    if request.method == 'POST':
        user.username = request.form['username']
        if request.form['password']:
            user.password = request.form['password']
        user.role = request.form['role']
        db.session.commit()
        log_user_activity(current_user.username, f'Edited user:
{user.username}')
        return redirect(url_for('manage_users'))

    return render_template('edit_user.html', user=user)

@app.route('/delete_user/<user_id>', methods=['POST'])
@login_required
@role_required('admin')
def delete_user(user_id):
    user = User.query.get(user_id)
    db.session.delete(user)
    db.session.commit()
    log_user_activity(current_user.username, f'Deleted user:
{user.username}')
    return redirect(url_for('manage_users'))


@app.route('/maps', methods=['GET'])
@login_required
def train_model():
    # Logic untuk mempersiapkan data dan model
    return render_template('map1.html')


@app.route('/upload', methods=['GET', 'POST'])
@login_required
def upload_data():
    if request.method == 'POST':
        file = request.files['file']
        if file:
            # Simpan file CSV yang diunggah
```

```
            filepath = os.path.join(DATA_FOLDER,
'uploaded_data.csv')
            file.save(filepath)

            # Log aktivitas user
            log_user_activity(current_user.username, 'Uploaded new
data')

            # Redirect ke halaman visualisasi
            return redirect(url_for('visualize_csv',
filename='uploaded_data.csv'))

    return render_template('upload.html')

# Halaman untuk menampilkan grafik dari file CSV
@app.route('/visualize/<filename>', methods=['GET'])
@login_required
def visualize_csv(filename):
    filepath = os.path.join(DATA_FOLDER, filename)

    try:
        # Membaca file CSV
        df = pd.read_csv(filepath)

        # Memeriksa apakah CSV memiliki setidaknya dua kolom
        if len(df.columns) < 2:
            return "File CSV harus memiliki minimal 2 kolom.", 400

        # Membuat grafik
        fig, ax = plt.subplots(figsize=(10, 6))
        ax.plot(df.iloc[:, 0], df.iloc[:, 1], label='Data',
color='b')

        # Menambahkan label
        ax.set_xlabel(df.columns[0])
        ax.set_ylabel(df.columns[4])
        ax.set_title(f'Grafik {filename}')

        # Menyimpan grafik dalam format PNG ke dalam memori
        img = io.BytesIO()
        plt.savefig(img, format='png')
        img.seek(0)

        # Mengencode gambar menjadi base64
```

```
        img_base64 =
base64.b64encode(img.getvalue()).decode('utf8')

        # Tampilkan grafik pada halaman HTML
        return render_template('visualize.html',
img_data=img_base64)

    except Exception as e:
        return f"Terjadi kesalahan saat memproses file: {str(e)}",
500

@app.route('/pengetahuan')
@login_required
def pengetahuan():
    return render_template('pengetahuan.html')


@app.route('/redirect_informer')
def redirect_informer():
    return redirect('http://127.0.0.1:5001/')  # Port Flask
Informer

# Route untuk mengunggah dan memproses data dengan ARIMA

@app.route('/predict_arima', methods=['GET', 'POST'])
def predict_arima():
    if not os.path.exists(UPLOAD_FILEPATH):
        flash("Tidak ada file yang tersedia untuk prediksi.
Silakan unggah file terlebih dahulu.", "danger")
        return redirect(url_for('upload_file'))

    if request.method == 'POST':
        try:
            steps = int(request.form.get('steps', 10))  # Ambil
input jumlah hari prediksi, default 10
            if steps <= 0:
                flash("Jumlah hari prediksi harus lebih dari 0.",
"danger")
                return redirect(url_for('predict_arima'))

            df = pd.read_csv(UPLOAD_FILEPATH, parse_dates=[0],
index_col=0)

            if df.empty or len(df.columns) < 1:
```

```
                flash("File CSV kosong atau tidak memiliki cukup
data.", "danger")
                return redirect(url_for('predict_arima'))

            series = df.iloc[:, 0]  # Ambil kolom pertama

            # Melatih model ARIMA
            model = ARIMA(series, order=(5,1,0))  # Order bisa
disesuaikan
            model_fit = model.fit()

            # Prediksi sesuai jumlah hari yang diinput
            forecast = model_fit.forecast(steps=steps)
            forecast_dates = pd.date_range(start=series.index[-1],
periods=steps+1, freq='D')[1:]

            # Menghitung MSE & MAPE menggunakan data terakhir
sebagai perbandingan
            actual_values = series[-steps:] if len(series) >=
steps else series
            mse = mean_squared_error(actual_values,
forecast[:len(actual_values)])

            # MAPE = Mean Absolute Percentage Error
            mape = np.mean(np.abs((actual_values -
forecast[:len(actual_values)]) / actual_values))

            # Menampilkan hasil dalam bentuk grafik
            fig, ax = plt.subplots(figsize=(10, 6))
            ax.plot(forecast_dates, forecast, label='Prediksi',
color='red', marker='o')
            ax.legend()
            ax.set_title(f'Prediksi ARIMA {steps} Hari ke Depan')

            img = io.BytesIO()
            plt.savefig(img, format='png')
            img.seek(0)
            img_base64 =
base64.b64encode(img.getvalue()).decode('utf8')

            # Membuat tabel prediksi dalam format list
            prediction_table = pd.DataFrame({
                'Tanggal': forecast_dates.strftime('%Y-%m-%d'),
                'Prediksi': forecast.values
            }).to_dict(orient='records')
```

```
            return render_template(
                'predict_arima.html',
                img_data=img_base64,
                predictions=prediction_table,
                mse=mse,
                mape=mape,
                steps=steps
            )

        except Exception as e:
            flash(f"Terjadi kesalahan: {str(e)}", "danger")
            return redirect(url_for('predict_arima'))

    return render_template('input_steps.html')  # Redirect ke
halaman upload jika error

# Lihat Log Aktivitas
@app.route('/logs')
@login_required
@role_required('admin')
def view_logs():
    logs = UserLog.query.all()
    return render_template('logs.html', logs=logs)

if __name__ == '__main__':
    app.run(debug=True)
```

Lampiran 5 *Source Code* Aplikasi Informer

```
from flask import Flask, request, jsonify, render_template
import torch
import numpy as np
import pandas as pd
from sklearn.preprocessing import MinMaxScaler
from models.model import Informer
import logging
import requests
from flask_cors import CORS

# --- FUNGSI AMBIL DATA USGS (SESUAI KOORDINAT SELAT SUNDA) ---
def download_usgs_earthquake_data(starttime, endtime, minlatitude,
maxlatitude,
```

```
                                    minlongitude, maxlongitude,
                                    minmagnitude=2.0,
                                    limit=200):
    url = "https://earthquake.usgs.gov/fdsnws/event/1/query"
    params = {
        "format": "geojson",
        "starttime": starttime,
        "endtime": endtime,
        "minlatitude": minlatitude,
        "maxlatitude": maxlatitude,
        "minlongitude": minlongitude,
        "maxlongitude": maxlongitude,
        "minmagnitude": minmagnitude,
        "orderby": "time",  # urutan terbaru
        "limit": limit
    }

    response = requests.get(url, params=params)
    data = response.json()
    records = []
    for feature in data.get("features", []):
        props = feature["properties"]
        coords = feature["geometry"]["coordinates"]
        records.append({
            "time": props["time"],
            "place": props["place"],
            "mag": props["mag"],
            "longitude": coords[0],
            "latitude": coords[1],
            "depth": coords[2],
        })
    df = pd.DataFrame(records)
    if not df.empty:
        df["time"] = pd.to_datetime(df["time"], unit="ms")
        df = df.sort_values("time").reset_index(drop=True)
    return df

# --- FUNGSI WAKTU ---
def time_features(df, freq):
    if "date" not in df.columns:
        raise ValueError("Kolom 'date' tidak ditemukan dalam
DataFrame.")
    if not pd.api.types.is_datetime64_any_dtype(df["date"]):
        df["date"] = pd.to_datetime(df["date"])
    time_feats = []
```

```
    if freq == "d":
        time_feats.append(df["date"].dt.day.values)
        time_feats.append(df["date"].dt.dayofweek.values)
        time_feats.append(df["date"].dt.month.values)
    else:
        raise ValueError(f"Frekuensi {freq} tidak didukung.")
    return np.stack(time_feats, axis=1)

# --- FLASK APP ---
app = Flask(__name__)
CORS(app)

@app.route("/")
def index():
    return render_template("index.html")

# --- LOAD MODEL ---
device = torch.device("cuda" if torch.cuda.is_available() else
"cpu")
model = Informer(
    enc_in=1, dec_in=1, c_out=1,
    seq_len=100, label_len=7, out_len=7,
    factor=5, d_model=512, n_heads=8,
    e_layers=2, d_layers=1, d_ff=512,
    dropout=0.05, attn="prob", embed="timeF",
    freq="d", activation="gelu",
    output_attention=False, distil=True,
    mix=True, device=device
)
model.load_state_dict(torch.load("informer_model.pth",
map_location=device), strict=False)
model.to(device)
model.eval()
scaler = MinMaxScaler()
logging.basicConfig(level=logging.DEBUG)

@app.route('/predict', methods=['POST'])
def predict():
    try:
        logging.debug("Masuk ke endpoint /predict")

        df_realtime = download_usgs_earthquake_data(
            starttime="1990-01-01",
            endtime="2024-12-31",
            minlatitude=-8.494,
```

```
            maxlatitude=-5.703,
            minlongitude=103.052,
            maxlongitude=106.743,
            minmagnitude=2.0,
            limit=107
        )

        df_realtime =
df_realtime.sort_values("time").tail(107).reset_index(drop=True)


        if df_realtime.empty:
            return jsonify({"error": "Data gempa tidak
ditemukan!"}), 404

        # ambil 107 data terakhir
        df_realtime = df_realtime.tail(107).reset_index(drop=True)
        print("Jumlah data akhir:", len(df_realtime))
        if len(df_realtime) < 107:
            return jsonify({"error": "Data tidak cukup untuk
prediksi!"}), 400

        df_test = df_realtime.rename(columns={"time": "date",
"mag": "value"})[["date", "value"]]
        df_test["date"] = pd.to_datetime(df_test["date"])
        df_test["value"] = pd.to_numeric(df_test["value"],
errors="coerce")
        df_test["value"].fillna(df_test["value"].mean(),
inplace=True)
        df_test["value"] =
scaler.fit_transform(df_test["value"].values.reshape(-1, 1))

        if len(df_test) < 2:
            return jsonify({"error": "Data terlalu sedikit untuk
hitung delta waktu"}), 400

        delta = df_test["date"].iloc[1] - df_test["date"].iloc[0]
        for _ in range(7):
            new_row = {"date": df_test["date"].iloc[-1] + delta,
"value": 0}
            df_test = pd.concat([df_test,
pd.DataFrame([new_row])], ignore_index=True)
        df_test.fillna(0, inplace=True)

        df_test_x = df_test.iloc[-100-7:-7].copy()
```

```
        df_test_y = df_test.iloc[-7:].copy()
        test_time_x = time_features(df_test_x, freq="d")
        test_time_y = time_features(df_test_y, freq="d")
        df_test_numpy = df_test.to_numpy()[:, 1:].astype("float")

        test_data_x = df_test_numpy[-100-7:-7]
        test_data_y = np.zeros((7, 1))

        batch_x =
torch.tensor(test_data_x).unsqueeze(0).float().to(device)
        batch_x_mark =
torch.tensor(test_time_x).unsqueeze(0).float().to(device)
        batch_y =
torch.tensor(test_data_y).unsqueeze(0).float().to(device)
        batch_y_mark =
torch.tensor(test_time_y).unsqueeze(0).float().to(device)

        with torch.no_grad():
            outputs = model(batch_x, batch_x_mark, batch_y,
batch_y_mark)
            preds = outputs.detach().cpu().numpy()

        preds = scaler.inverse_transform(preds[0])

        def classify_value(value):
            if value < 4:
                return "waspada"
            elif 4 <= value < 6:
                return "siaga"
            else:
                return "berbahaya"

        pred_dates = [str(df_test["date"].iloc[-7 + i]) for i in
range(7)]
        response = {
            "predictions": [
                {
                    "date": date,
                    "value": float(value),
                    "classification": classify_value(value)
                }
                for date, value in zip(pred_dates,
preds.flatten().tolist())
            ]
        }
```

```
        print("Prediksi sukses:\n", response)
        return jsonify(response)

    except Exception as e:
        logging.error(f"Error terjadi: {e}")
        return jsonify({"error": str(e)}), 500

if __name__ == '__main__':
    app.run(debug=True, host='0.0.0.0', port=5001)
```