DAFTAR LAMPIRAN

Lampiran 1. Surat Keterangan Bebas Plagiasi

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 : Implementasi Datamining Untuk Mendukung Program Reduksi Sampah Di Daerah Khusus Jakarta Dengan Algoritma Time Series dan K-Means Clustering

Penulis : Muhammad Krisna Adiputro

NIM : **2019230157**

Tgl pemeriksaan : 21 Februari 2025

Dengan hasil Tingkat Kesamaan (*similarity index*) **19%**

Demikian Surat Keterangan kami buat, untuk dipergunakan sebagaimana mestinya.

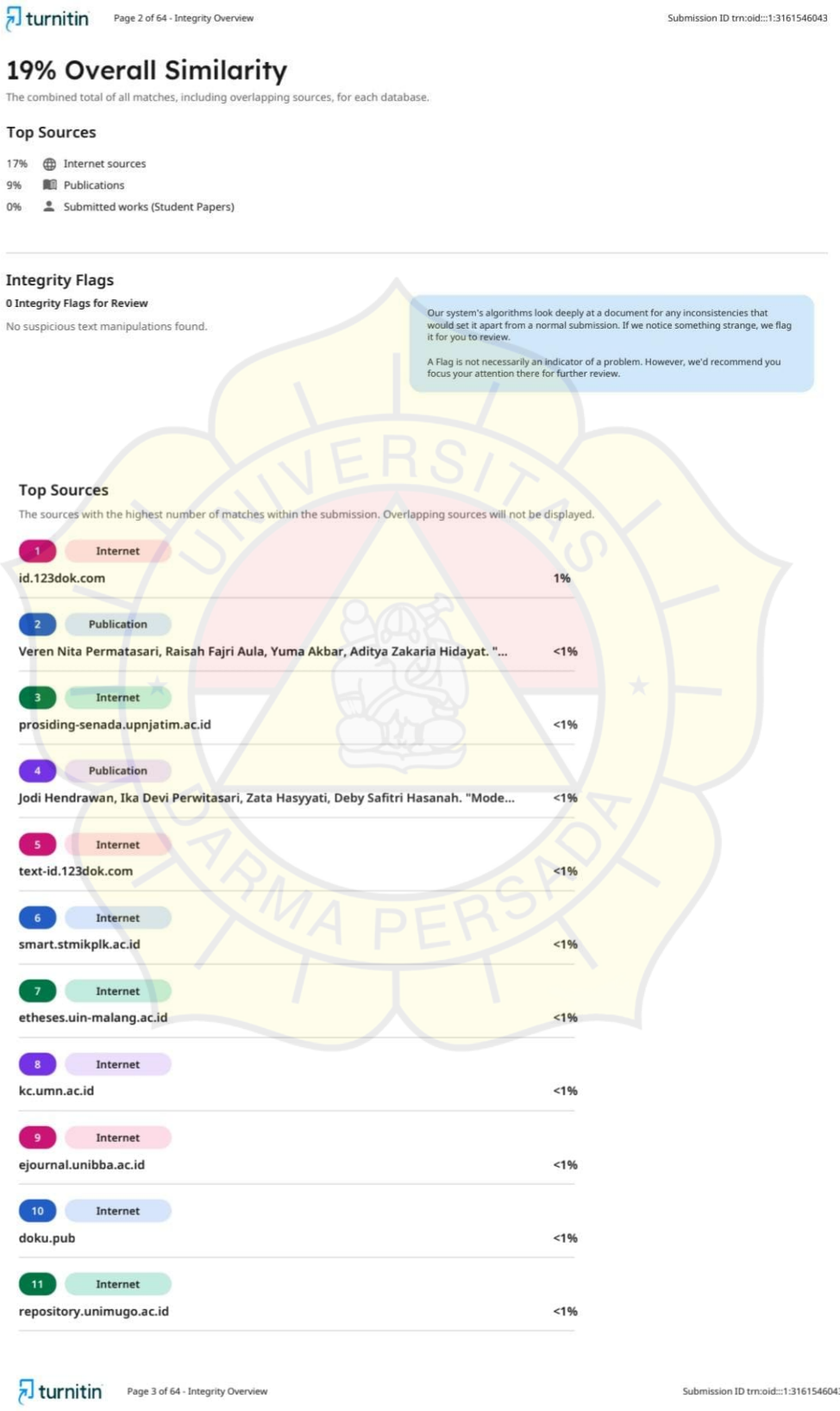
Jakarta, 21 Februari 2025

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. Hasil Turnitin



turnitin Page 2 of 64 - Integrity Overview Submission ID trn:oid:::1:3161546043

19% Overall Similarity

The combined total of all matches, including overlapping sources, for each database.

Top Sources

17% Internet sources
9% Publications
0% Submitted works (Student Papers)

Integrity Flags

0 Integrity Flags for Review

No suspicious text manipulations found.

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.

Top Sources

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

1	Internet	
id.123dok.com		1%
2	Publication	
Veren Nita Permatasari, Raisah Fajri Aula, Yuma Akbar, Aditya Zakaria Hidayat. "...		<1%
3	Internet	
prosiding-senada.upnjatim.ac.id		<1%
4	Publication	
Jodi Hendrawan, Ika Devi Perwitasari, Zata Hasyyati, Deby Safitri Hasanah. "Mode...		<1%
5	Internet	
text-id.123dok.com		<1%
6	Internet	
smart.stmikplk.ac.id		<1%
7	Internet	
etheses.uin-malang.ac.id		<1%
8	Internet	
kc.umn.ac.id		<1%
9	Internet	
ejournal.unibba.ac.id		<1%
10	Internet	
doku.pub		<1%
11	Internet	
repository.unimugo.ac.id		<1%

turnitin Page 3 of 64 - Integrity Overview Submission ID trn:oid:::1:3161546043

12 Internet

ejournal.itn.ac.id <1%

13 Internet

laakfkb.telkomuniversity.ac.id <1%

14 Internet

repository.fe.unj.ac.id <1%

15 Internet

bkddki.jakarta.go.id <1%

16 Internet

dspace.uii.ac.id <1%

17 Internet

www.kompasiana.com <1%

18 Publication

Marlen T Lesnussa, Melianus Salakory, Edward G Tetelepta. "Waste Management ... <1%

19 Internet

digilib.itb.ac.id <1%

20 Internet

123dok.com <1%

21 Publication

Asri Fornika Sari, Seno D Panjaitan, Bomo W Sanjaya. "Optimasi Pemantauan Kua... <1%

22 Publication

Admin Teika, Yusran Timur Samuel, Frengky Simbolon. "Perancangan Aplikasi Un... <1%

23 Publication

Sri Lestari, Maryana Febryanti. "Analisis Sentimen Mengenai Produk Inovasi Invis... <1%

24 Publication

Zahra Purwanti, Sugiyono. "Pemodelan Text Mining untuk Analisis Sentimen Ter... <1%

25 Internet

eprints.upj.ac.id <1%

26	Internet	
id.scribd.com		<1%
27	Internet	
library.binus.ac.id		<1%
28	Internet	
simki.unpkediri.ac.id		<1%
29	Internet	
swa.co.id		<1%
30	Internet	
vegiwilandari.wordpress.com		<1%
31	Internet	
lib.ui.ac.id		<1%
32	Internet	
jogodebola.net		<1%
33	Internet	
ailima.co.id		<1%
34	Internet	
geograf.id		<1%
35	Internet	
6b1aa66e-13bb-4ed4-a7fe-e6007b3f5822.filesusr.com		<1%
36	Publication	
Abd Azis, Sutisna. "Penerapan Data Mining untuk Menentukan Ketersediaan Sto...		<1%
37	Internet	
gaya.tempo.co		<1%
38	Internet	
repository.itsb.ac.id		<1%
39	Internet	
rinjani.unitri.ac.id		<1%

40	Publication	
Amrizal Novianto, Fadil Indra Sanjaya. "PERANCANGAN SISTEM INFORMASI UNTU...		<1%
41	Publication	
Iksan Mule. "Peramalan Jumlah Penduduk Miskin di Provinsi Maluku Tahun 2021 ...		<1%
42	Publication	
NAUFAL ARIF HIDAYATULLAH, Willy Prihartono, Fathur Rohman. "CLUSTERING PE...		<1%
43	Publication	
Novri Hadinata, Kurniawan Kurniawan. "ANALISIS POLA PEMBELIAN PRODUK MA...		<1%
44	Internet	
berbagireferensiilmiah.blogspot.com		<1%
45	Internet	
ejournal.ust.ac.id		<1%
46	Internet	
eprints.uad.ac.id		<1%
47	Internet	
fkom.almaata.ac.id		<1%
48	Internet	
academic-accelerator.com		<1%
49	Internet	
dokumen.tips		<1%
50	Internet	
es.scribd.com		<1%
51	Internet	
naniksuharti.wordpress.com		<1%
52	Internet	
repository.maranatha.edu		<1%
53	Internet	
www.ijettjournal.org		<1%

54	Internet	
www.kemenperin.go.id		<1%
55	Publication	
Dimas Bayu Anjasmara, Mochamad Alfan Rosid, Ade Eviyanti. "Implementasi Fitu...		<1%
56	Publication	
Susy Katarina Sianturi, Dina Satriani Fansuri, Wiwin Najmiatul Aini. "Algoritma Ap...		<1%
57	Internet	
contohlaporansusunanpkl.blogspot.com		<1%
58	Internet	
dias.library.tuc.gr		<1%
59	Internet	
dspace.umkt.ac.id		<1%
60	Internet	
ejurnal.univbatam.ac.id		<1%
61	Internet	
id.parkwaycancercentre.com		<1%
62	Internet	
jimfeb.ub.ac.id		<1%
63	Internet	
kmbappeda.jakarta.go.id		<1%
64	Internet	
ojs.unigal.ac.id		<1%
65	Internet	
repository.trisakti.ac.id		<1%
66	Internet	
www.coursehero.com		<1%
67	Publication	
Andhyagis Suatrat, Daniel A Sihasale. "Community Behavior in Waste Disposal Al...		<1%

68	Internet	
adoc.pub		<1%
69	Internet	
akademik.unsoed.ac.id		<1%
70	Internet	
de.scribd.com		<1%
71	Internet	
diyanaeunhyuk.blogspot.com		<1%
72	Internet	
duta.co		<1%
73	Internet	
guru-id.github.io		<1%
74	Internet	
journal.unj.ac.id		<1%
75	Internet	
mediaindonesia.com		<1%
76	Internet	
repository.its.ac.id		<1%
77	Internet	
valexeev.yolasite.com		<1%
78	Internet	
wahjoecyber.blogspot.com		<1%
79	Internet	
www.hdewcameras.co.uk		<1%
80	Publication	
Ferdiansyah Aditya Soliata, Ryan Randy Suryono. "Sistem Otomatisasi Pengairan ...		<1%
81	Publication	
Zaid Romegar Mair, Helen Yunita Sari. "Aplikasi Kasir Pada Adibah Boutique Berb...		<1%

82	Publication	
Herdy Hardana, Ruben Edward, Sri Mardiyati. "Perancangan Sistem Informasi Ad...		<1%
83	Publication	
Meri Fitriani, Gigih Forda Nama, Mardiana Mardiana. "Implementasi Association ...		<1%
84	Publication	
Muhammad Syani, Tundo, Sugiyono, Tri Wahyudi. "Klasterisasi Penggunaan Ban...		<1%
85	Publication	
Royani Wulandari. "Strategi Berkelanjutan dalam Mengatasi Krisis Sampah Di Kot...		<1%
86	Publication	
Veri Arinal, Irma Rusmarhadi. "Implementasi Data Mining Untuk Menentukan Str...		<1%
87	Internet	
repository.ub.ac.id		<1%

Lampiran 3. Codingan Python

```
import streamlit as st
import pandas as pd
import matplotlib.pyplot as plt
import io
import base64
import time
from pymongo import MongoClient, errors
from statsmodels.tsa.arima.model import ARIMA
from sklearn.cluster import KMeans
from sklearn.preprocessing import StandardScaler, LabelEncoder
from datetime import datetime
from sklearn.metrics import silhouette_score,
calinski_harabasz_score
from sklearn.decomposition import PCA
import numpy as np
from statsmodels.tsa.stattools import adfuller
from sklearn.metrics import mean_squared_error
from sklearn.metrics import mean_absolute_error
import itertools

# MongoDB connection with error handlingecho "# ForecastApp" >>
README.md
try:
    client =
MongoClient("mongodb+srv://krisna:krisna@cluster0.3mao11f.mongodb.
net/?retryWrites=true&w=majority&appName=Cluster0")
    db = client['dbkrisna']
    collection_visualisasi = db['visualisasi']
    collection_history = db['history']
except errors.ConnectionError as e:
    st.error("Could not connect to MongoDB. Please check your
connection settings.")
    st.stop()  # Stop the app if the connection fails

# User management
users = {
    "admin": {"password": "admin", "role": "admin"},
    "upst": {"password": "admin", "role": "user"},
    "psm": {"password": "admin", "role": "user"}
}

def add_user(username, password):
    if username not in users:
        users[username] = {"password": password, "role": "user"}
        return True
    return False

def edit_user(username, new_password=None):
    if username in users:
        if new_password:
            users[username]["password"] = new_password
```

```
        return True
    return False

def delete_user(username):
    if username in users and username != "admin":   # Prevent
deletion of admin
        del users[username]
        return True
    return False

def login():
    st.title("Login")
    username = st.text_input("Username")
    password = st.text_input("Password", type="password")
    if st.button("Login"):
        if username in users and users[username]  ["password"] ==
password:
            st.session_state['logged_in'] = True
            st.session_state['role'] = users[username]["role"]
            st.session_state['username'] = username
            st.rerun()
        else:
            st.error("Username atau password salah")

def clear_collection(collection):
    with st.spinner("Mengosongkan collection..."):
        collection.delete_many({})
        st.success("Collection berhasil dikosongkan!")

def delete_history_by_collection_name(collection_name):
    with        st.spinner(f"Menghapus        semua        riwayat
'{collection_name}'..."):
        collection_history.delete_many({"collection_name":
collection_name})

def load_data(collection):
    progress_bar = st.progress(0)
    status_text = st.empty()
    cursor = collection.find()
    data_list = []
    total = collection.count_documents({})
    count = 0

    for doc in cursor:
        data_list.append(doc)
        count += 1
        progress = int((count / total) * 100)
        progress_bar.progress(progress)
        status_text.text(f"Memuat data... {progress}%")
        time.sleep(0.01)

    progress_bar.empty()
    status_text.text("Memuat data selesai!")
    data = pd.DataFrame(data_list)
    return data
```

```
def save_data(collection, data):
    collection.delete_many({})
    progress_bar = st.progress(0)
    status_text = st.empty()
    records = data.to_dict('records')
    total = len(records)
    batch_size = 10

    for i in range(0, total, batch_size):
        batch = records[i:i+batch_size]
        collection.insert_many(batch)
        progress = int(((i + len(batch)) / total) * 100)
        progress_bar.progress(progress)
        status_text.text(f"Menyimpan data... {progress}%")
        time.sleep(0.01)

    progress_bar.empty()
    status_text.success("Data berhasil diunggah!")

def save_history(*args, collection_name):
    with st.spinner("Menyimpan riwayat data..."):
        history_data = {}
        for i, data in enumerate(args):
            if isinstance(data, plt.Figure):
                img_stream = io.BytesIO()

data.savefig(img_stream, format='png')
                img_stream.seek(0)
                img_data                                                =
base64.b64encode(img_stream.read()).decode('utf-8')
                history_data[f"fig_{i+1}"] = img_data
            else:
                history_data[f"data_{i+1}"]                             =
data.to_dict('records')

        history = {
            "collection_name": collection_name,
            "data": history_data,
            "timestamp":        datetime.now().strftime("%Y-%m-%d
%H:%M:%S")
        }
        collection_history.insert_one(history)

def load_history(collection_name=None):
    with st.spinner("Memuat riwayat data..."):
        if collection_name:
            history                                                     =
pd.DataFrame(list(collection_history.find({"collection_name":
collection_name})))
        else:
            history                                                     =
pd.DataFrame(list(collection_history.find()))
    return history

def delete_history(history_id):
```

```
    with st.spinner("Menghapus riwayat data..."):
        collection_history.delete_one({'_id': history_id})

def show_image_from_history(img_data):
    img = base64.b64decode(img_data)
    st.image(img, use_container_width=True)

def arima_prediction(data):
    with st.spinner("Membuat prediksi ARIMA..."):
        month_mapping = {
            "Januari": 1, "Februari": 2, "Maret": 3, "April": 4,
            "Mei": 5, "Juni": 6, "Juli": 7, "Agustus": 8,
            "September":  9,  "Oktober":  10,  "November":  11,
"Desember": 12
        }

        df_arima = data.copy()
        df_arima['month'] = df_arima['Bulan'].map(month_mapping)
        df_arima['year'] = df_arima['Tahun'].astype(int)
        df_arima['date'] = pd.to_datetime(
            df_arima[['year',          'month']].assign(Day=1),
errors='coerce'
        )

        df_arima                                                =
pd.DataFrame(df_arima.groupby('date')['Tonase'].sum())
        ts_data = df_arima['Tonase']

        st.write("Hasil ADF Test sebelum differencing:")
        adf_test(ts_data)

        if adfuller(ts_data)[1] > 0.05:
            ts_data = ts_data.diff().dropna()
            st.write("\nHasil ADF Test setelah differencing:")
            adf_test(ts_data)

        train_size = int(len(ts_data) * 0.95)
        train, test = ts_data[:train_size], ts_data[train_size:]

        p, d, q = 9, 5, 4
        model = ARIMA(train, order=(p, d, q))
        model_fit = model.fit()

        forecast = model_fit.forecast(steps=len(test))
        forecast_series = pd.Series(forecast, index=test.index)

        mae = mean_absolute_error(test, forecast_series)
        mape = np.mean(np.abs((test - forecast_series) / test)) *
100
        rmse = np.sqrt(mean_squared_error(test, forecast_series))

        st.write("\nEvaluasi Model:")
        st.write(f"MAE: {mae}")
        st.write(f"MAPE: {mape}%")
        st.write(f"RMSE: {rmse}")
```

```
        full_series = pd.concat([train, test])
        forecast_series_full = pd.concat([train, forecast_series])

        fig, ax = plt.subplots(figsize=(10, 6))
        ax.plot(forecast_series_full,            label="Forecast",
color='red')
        ax.plot(train, label="Actual", color='blue')
        ax.set_title("ARIMA Model - Actual vs Forecast")
        ax.set_xlabel(None)
        ax.set_ylabel("Tonase")
        ax.legend()
        ax.grid()

        st.pyplot(fig)

    return fig

def adf_test(series):
    result = adfuller(series)
    st.write("ADF Test Statistic:", result[0])
    st.write("p-value:", result[1])
    st.write("Critical Values:")
    for key, value in result[4].items():
        st.write(f"   {key}: {value}")
    if result[1] <= 0.05:
        st.write("Data stasioner (H0 ditolak).")
    else:
        st.write("Data tidak stasioner (H0 diterima).")

def kmeans_prediction(data):
    with st.spinner("Membuat prediksi KMeans..."):
        data = data.copy()
        features    =    ['kota_kabupaten_code',    'tahun_code',
'bulan_code', 'Tonase']
        X = data[features]
        scaler = StandardScaler()
        X_scaled = scaler.fit_transform(X)

        inertia = []
        k_range = range(2, 11)
        for k in k_range:
            kmeans = KMeans(n_clusters=k, random_state=42)
            kmeans.

fit(X_scaled)
            inertia.append(kmeans.inertia_)

        fig_elbow, ax = plt.subplots(figsize=(8, 5))
        ax.plot(k_range, inertia, marker='o')
        ax.set_title("Elbow Method")
        ax.set_xlabel("Number of Clusters")
        ax.set_ylabel("Inertia")
        ax.set_xticks(k_range)
        ax.grid()
```

```
        st.pyplot(fig_elbow)

        silhouette_scores = []
        for k in k_range:
            kmeans = KMeans(n_clusters=k, random_state=42)
            kmeans.fit(X_scaled)
            labels = kmeans.labels_
            score = silhouette_score(X_scaled, labels)
            silhouette_scores.append(score)

        fig_silhoute, ax = plt.subplots(figsize=(8, 5))
        ax.plot(k_range,      silhouette_scores,      marker='o',
color='orange')
        ax.set_title("Silhouette Score")
        ax.set_xlabel("Number of Clusters")
        ax.set_ylabel("Silhouette Score")
        ax.set_xticks(k_range)
        ax.grid()
        st.pyplot(fig_silhoute)

        optimal_k = k_range[np.argmax(silhouette_scores)]
        st.write(f'Jumlah cluster optimal berdasarkan Silhouette
Score: {optimal_k}')

        kmeans_optimal       =       KMeans(n_clusters=optimal_k,
random_state=42)
        kmeans_optimal.fit(X_scaled)
        data['cluster'] = kmeans_optimal.labels_

        # Define waste categories based on Tonase
        def categorize_waste(t):
            if t > 5000:
                return 'Large'
            elif t > 2000:
                return 'Medium'
            else:
                return 'Small'

        data['Waste_Category']                                    =
data['Tonase'].apply(categorize_waste)

        # Create a summary table for cities with their waste
categories and tonnage
        summary_table = data[['Kota / Kabupaten', 'Waste_Category',
'Tonase']].copy()
        summary_table = summary_table.groupby(['Kota / Kabupaten',
'Waste_Category']).agg(
            Total_Tonase=('Tonase', 'sum')
        ).reset_index()

        st.subheader("City Waste Summary")
        st.dataframe(summary_table)

        # Visualize the clusters
        pca = PCA(n_components=2)
```

```
        X_pca = pca.fit_transform(X_scaled)

        fig, ax = plt.subplots(figsize=(10, 5))
        scatter    =    ax.scatter(X_pca[:,    0],    X_pca[:,    1],
c=data['cluster'], cmap='viridis')
        ax.set_title('KMeans Clustering Visualization')
        ax.set_xlabel('PCA Component 1')
        ax.set_ylabel('PCA Component 2')
        plt.colorbar(scatter, ax=ax, label='Cluster')
        st.pyplot(fig)

    return fig_elbow, fig_silhoute, fig, summary_table

def main():
    if 'logged_in' not in st.session_state:
        st.session_state['logged_in'] = False
    if 'role' not in st.session_state:
        st.session_state['role'] = None

    if not st.session_state['logged_in']:
        login()
    else:
        st.sidebar.title("Navigasi")
        page = st.sidebar.radio("Pilih Halaman", ["Data", "Time",
"Area", "Admin", "Log Out"])

        if page == "Admin" and st.session_state['role'] == "admin":
            st.title("User  Management")
            user_to_manage   =   st.selectbox("Pilih   User   untuk
Dikelola", ["upst", "psm"])
            action = st.selectbox("Pilih Aksi", ["Add User", "Edit
User", "Delete User"])
            if action == "Add User":
                new_username = st.text_input("Username")
                new_password      =      st.text_input("Password",
type="password")
                if st.button("Add User"):
                    if add_user(new_username, new_password):
                        st.success("User  berhasil ditambahkan!")
                    else:
                        st.error("User  sudah ada!")

            elif action == "Edit User":
                edit_username = user_to_manage
                new_password   =   st.text_input("New   Password",
type="password")
                if st.button("Edit User"):
                    if edit_user(edit_username, new_password):
                        st.success("User  berhasil diedit!")
                    else:
                        st.error("User  tidak ditemukan!")

            elif action == "Delete User":
delete_username = user_to_manage
                if st.button("Delete User"):
```

```
                    if delete_user(delete_username):
                        st.success("User  berhasil dihapus!")
                    else:
                        st.error("User  tidak ditemukan!")

        elif page == "Data":
            st.title("Data XLS")
            uploaded_file    =    st.file_uploader("Upload    XLS",
type="xlsx")

            if uploaded_file is not None:
                data = pd.read_excel(uploaded_file)
                df   =   data.applymap(lambda   x:   x.strip()   if
isinstance(x, str) else x)
                df   =   df.applymap(lambda   x:   x.capitalize()   if
isinstance(x, str) else x)
                df  =  df[['Kota  /  Kabupaten',  'Tahun',  'Bulan',
'Tonase']]
                df = df[(df['Kota / Kabupaten'].isna() == False) &
(df['Kota / Kabupaten'] != 'Lembaga')]
                df  =  df[(df['Tonase']  >=  100)  &  (df['Tonase']  <=
10000)]

                encoder = LabelEncoder()
                df['kota_kabupaten_code']                              =
encoder.fit_transform(df['Kota / Kabupaten'])
                df['tahun_code']                                       =
encoder.fit_transform(df['Tahun'])
                df['bulan_code']                                       =
encoder.fit_transform(df['Bulan'])

                save_data(collection_visualisasi, df)
                st.session_state['data_loaded'] = True
                st.subheader("Dataset Head")
                st.write(df.head())
                save_history(df.head(),
collection_name="visualisasi")

            total_rows                                                 =
collection_visualisasi.count_documents({})
            st.write(f"Total   data   saat   ini:   **{total_rows}
baris**")
            if st.button("Kosongkan Collection Visualisasi"):
                clear_collection(collection_visualisasi)
                st.rerun()

            st.title("Riwayat Visualisasi")
            history = load_history('visualisasi')
            if history.empty:
                st.warning("Belum ada riwayat visualisasi.")
            else:
                if st.button("Kosongkan Riwayat"):

delete_history_by_collection_name("visualisasi")
                    st.rerun()
```

```
                for index, row in history.iterrows():
                    st.write(f"{row['timestamp']}                -
{row['collection_name']}")
                    col1, col2 = st.columns([1, 1])
                    with col1:
                        if st.button(f"Lihat Hasil {row['_id']}"):
                            st.dataframe(row['data']['data_1'])
                    with col2:
                        if st.button(f"Hapus {row['_id']}"):
                            delete_history(row['_id'])
                            st.success(f"Riwayat      {row['_id']}
berhasil dihapus")
                            time.sleep(1)
                            st.rerun()

        elif page == "Time" and st.session_state['role'] == "user"
and st.session_state['username'] == "upst":
            if   'data'   not   in   st.session_state   or   not
st.session_state.get('data_loaded', False):
                data = load_data(collection_visualisasi)
                st.session_state['data'] = data
                st.session_state['data_loaded'] = True

            data = st.session_state['data'].copy()

            if st.button("Tampilkan Prediksi ARIMA"):
                plt = arima_prediction(data)
                save_history(plt, collection_name="arima")

            st.title("Riwayat Prediksi ARIMA")
            history = load_history('arima')
            if history.empty:
                st.warning("Belum ada riwayat prediksi.")
            else:
                if st.button("Kosongkan Riwayat"):
                    delete_history_by_collection_name("arima")
                    st.rerun()

                for index, row in history.iterrows():
                    st.write(f"{row['timestamp']}                -
{row['collection_name']}")
                    col1 = st.columns(1)  # Create one column
                    with col1[0]:  # Use the first (and only) column
                        if st.button(f"Lihat Hasil {row['_id']}"):
                            if 'fig_1' in row['data']:
fig_base64 = row['data']['fig_1']

show_image_from_history(fig_base64)

        elif page == "Area" and st.session_state['role'] == "user"
and st.session_state['username'] == "psm":
            if   'data'   not   in   st.session_state   or   not
st.session_state.get('data_loaded', False):
                data = load_data(collection_visualisasi)
```

```
                st.session_state['data'] = data
                st.session_state['data_loaded'] = True

            data = st.session_state['data'].copy()

            if st.button("Tampilkan Prediksi KMeans"):
                fig_elbow, fig_silhoute, fig, cluster_summary =
kmeans_prediction(data)
                save_history(fig,     fig_elbow,     fig_silhoute,
cluster_summary, collection_name="kmeans")

            st.title("Riwayat Prediksi KMeans")
            history = load_history('kmeans')
            if history.empty:
                st.warning("Belum ada riwayat prediksi KMeans.")
            else:
                if st.button("Kosongkan Riwayat"):
                    delete_history_by_collection_name("kmeans")
                    st.rerun()

                for index, row in history.iterrows():
                    st.write(f"{row['timestamp']}                -
{row['collection_name']}")
                    col1, col2 = st.columns([1, 1])
                    with col1:
                        if st.button(f"Lihat Hasil {row['_id']}"):
                            if 'fig_1' in row['data']:
                                fig_base64 = row['data']['fig_1']

show_image_from_history(fig_base64)
                            if 'fig_2' in row['data']:
                                elbow_fig_base64                =
row['data']['fig_2']

show_image_from_history(elbow_fig_base64)
                            if 'fig_3' in row['data']:
                                elbow_fig_base64                =
row['data']['fig_3']

show_image_from_history(elbow_fig_base64)
                            if 'data_4' in row['data']:
                                df                              =
pd.DataFrame(row['data']['data_4'])
                                st.dataframe(df)
                    with col2:
                        if st.button(f"Hapus {row['_id']}"):
                            delete_history(row['_id'])
                            st.success(f"Riwayat      {row['_id']}
berhasil dihapus")
                            time.sleep(1)
                            st.rerun()

        elif page == "Log Out":
            st.session_state['logged_in'] = False
            st.session_state['role'] = None
```

```
            st.session_state['username'] = None
            st.rerun()

if name == "__main__":
    main()
```