LAMPIRAN

Source Code Library Python Yang Di Gunakan

```
from flask import Flask, request, redirect, session,
url_for, flash, render_template, send_file
from flask_login import LoginManager, login_user,
login_required, logout_user, current_user
from math import ceil
import pandas as pd
import os
import numpy as np
from sklearn.preprocessing import MinMaxScaler
from fcmeans import FCM
from sklearn.metrics import silhouette_score
import matplotlib.pyplot as plt
import io
import base64
from io import BytesIO
import folium
from folium.plugins import HeatMap
from datetime import datetime
from models import db,
Earthquake,ClusteredEarthquake, User
from sqlalchemy import func


app = Flask(__name__)
app.config['SQLALCHEMY_DATABASE_URI']='mysql+mysqlcon
nector://root:@localhost/clusterdb'
app.config['SQLALCHEMY_TRACK_MODIFICATIONS'] = False
app.config['SECRET_KEY'] = 'admin'

# Register the max and min filters
app.jinja_env.globals['max'] = max
app.jinja_env.globals['min'] = min

db.init_app(app)
```

Source Code Models.py digunakan untuk menyimpan database

```
from flask_sqlalchemy import SQLAlchemy
from flask_login import UserMixin

db = SQLAlchemy()
```

```
class User(db.Model, UserMixin):
    id = db.Column(db.Integer, primary_key=True)
    username = db.Column(db.String(150), unique=True,
nullable=False)
    password = db.Column(db.String(150),
nullable=False)

class Earthquake(db.Model):
    __tablename__ = 'earthquake'

    id = db.Column(db.Integer, primary_key=True)
    tanggal = db.Column(db.Date, nullable=False)
    waktu = db.Column(db.String(5),
nullable=False)  # MM.SS
    latitude = db.Column(db.Float, nullable=False)
    longitude = db.Column(db.Float, nullable=False)
    magnitude = db.Column(db.Float, nullable=False)
    depth = db.Column(db.Float, nullable=False)
    # Tambahkan kolom cluster_label
    # cluster_label = db.Column(db.Integer,
nullable=False)  # Nilai default None jika belum di-
cluster

    def __init__(self, tanggal, waktu, latitude,
longitude, magnitude, depth,):
        self.tanggal = tanggal
        self.waktu = waktu
        self.latitude = latitude
        self.longitude = longitude
        self.magnitude = magnitude
        self.depth = depth

class ClusteredEarthquake(db.Model):
    __tablename__ = 'clustered_earthquake'


    id = db.Column(db.Integer, primary_key=True)
    earthquake_id = db.Column(db.Integer,
db.ForeignKey('earthquake.id'), nullable=False)
    cluster_label = db.Column(db.Integer,
nullable=True)

    def __init__(self, earthquake_id, cluster_label):
        self.earthquake_id = earthquake_id
        self.cluster_label = cluster_label
```

Source Code Scripts.js untuk menyimpan sintaks *javasript*

```
// Mengatur active state saat link di-klik
document.querySelectorAll('nav a').forEach(link => {
    link.addEventListener('click', function () {
        // Menghapus kelas 'active' dari semua link
        document.querySelectorAll('nav
a').forEach(item => item.classList.remove('active'));
        // Menambahkan kelas 'active' ke link yang
di-klik
        this.classList.add('active');
    });
});


//----Modal Manual Add--- const
modalManualAdd =
document.getElementById('modalManualAdd')
const openModalManualAdd =
document.getElementById('openManualAdd')
const closeModalManualAdd =
document.getElementById('closeModalManualAdd')

// Open modal
openModalManualAdd.addEventListener('click', function
() {
    modalManualAdd.classList.remove('hidden')
})

// Close modal
closeModalManualAdd.addEventListener('click',
function () {
    modalManualAdd.classList.add('hidden')
})

// Close modal when clicking outside of the modal
window.onclick = function (event) {
    if (event.target === modalManualAdd) {
        modalManualAdd.classList.add('hidden')
    }
}

//Modal Edit data
function openEditModal(id, tanggal, waktu, latitude,
longitude, magnitude, depth) {
    document.getElementById("editId").value = id;
    document.getElementById("editTanggal").value =
tanggal;
    document.getElementById("editWaktu").value =
waktu;
```

```
    document.getElementById("editLatitude").value =
latitude;
    document.getElementById("editLongitude").value =
longitude;
    document.getElementById("editMagnitude").value =
magnitude;
    document.getElementById("editDepth").value =
depth;
    document.getElementById("editModal").classList.re
move("hidden");
}

function closeEditModal() {
    document.getElementById("editModal").classList.ad
d("hidden");
}

//Modal drag and drop file csv
//Referencing HTML elements
const csvModal = document.getElementById("csvModal");
const openCsvAdd =
document.getElementById("openCsvAdd");
const dropArea = document.getElementById("dropArea");
const fileInput =
document.getElementById("fileInput");

// Buka modal saat tombol "Tambah Data CSV" diklik
openCsvAdd.addEventListener("click", function () {
    csvModal.classList.remove("hidden");
});

// Tutup modal
function closeCsvModal() {
    csvModal.classList.add("hidden");
}

// Prevent default drag behaviors
['dragenter', 'dragover', 'dragleave',
'drop'].forEach(eventName => {
    dropArea.addEventListener(eventName,
preventDefaults, false);
});

function preventDefaults(e) {
    e.preventDefault();
    e.stopPropagation();
}
```

```
// Highlight drop area when item is dragged over it
['dragenter', 'dragover'].forEach(eventName => {
    dropArea.addEventListener(eventName, () => {
        dropArea.classList.add('border-blue-500');
    }, false);
});

['dragleave', 'drop'].forEach(eventName => {
    dropArea.addEventListener(eventName, () => {
        dropArea.classList.remove('border-blue-500');
    }, false);
});

// Handle dropped files
dropArea.addEventListener('drop', handleDrop, false);

function handleDrop(e) {
    const dt = e.dataTransfer;
    const files = dt.files;
    handleFiles(files);
}

function handleFiles(files) {
    // Assign dropped files to the file input
    fileInput.files = files;
    const fileName = files[0].name;
    dropArea.innerHTML = `<p class="text-gray-
600">Selected File: ${fileName}</p>`;
}

// Optional: Preview file name when a file is
selected via "Choose File"
fileInput.addEventListener("change", function () {
    const fileName = fileInput.files[0].name;
    dropArea.innerHTML = `<p class="text-gray-
600">Selected File: ${fileName}</p>`;
});

setTimeout(function() {
    const flashMessage =
document.querySelector('.flash-message');
    if (flashMessage) {
      flashMessage.classList.add('opacity-0');  //
Menggunakan Tailwind untuk fade-out
```

```
      setTimeout(() => flashMessage.style.display =
'none', 300);  // Setelah fade-out selesai,
sembunyikan
    }
  }, 3000);  // 5000ms = 5 detik
```

Source Code Halaman *Login.html*

```
<!DOCTYPE html>
<html lang="en">
  <head>
    <meta charset="UTF-8" />
    <meta name="viewport" content="width=device-
width, initial-scale=1.0" />
    <title>Clusteringapps</title>
    <link href="{{ url_for('static',
filename='dist/output.css') }}" rel="stylesheet" />
  </head>
  <body class="bg-gray-100 flex items-center justify-
center min-h-screen">
    <div class="w-full max-w-md bg-white rounded-lg
shadow-md p-8">
      <h2 class="text-2xl font-bold mb-6 text-center
text-gray-800">Login</h2>
        {# {% with messages =
get_flashed_messages(with_categories=true) %}
          {% if messages %}
              <ul class="space-y-4">
              {% for category, message in messages %}
                <li class="flash-message {{ 'bg-red-
500 text-white' if category == 'error' else 'bg-blue-
500 text-white' }}
                              p-4 rounded-md shadow-md
text-center">
                  {{ message }}
                </li>
              {% endfor %}
              </ul>
        {% endif %}
    {% endwith %} #}
      {# Menampilkan Harus Login Terlebih dahulu #}
      {% with messages =
get_flashed_messages(with_categories=true) %}
        {% if messages %}
```

```
            <ul class="space-y-4">
            {% for category, message in messages %}
                <li class="flash-message {{ 'bg-red-
500 text-white' if category == 'info' else 'bg-red-
500 text-white' }}
                                p-2 rounded-md shadow-md
text-center">
                    {{ message }}
                </li>
            {% endfor %}
            </ul>
        {% endif %}
      {% endwith %}
      <form method="POST" action="{{ url_for('login')
}}">
        <div class="mb-4">
          <label for="username" class="block text-
gray-700 text-sm font-bold mb-2">Username</label>
          <input type="username" id="username"
name="username" required class="w-full px-3 py-2
border rounded-lg focus:outline-none focus:ring-2
focus:ring-blue-500 focus:border-transparent" />
        </div>
        <div class="mb-6">
          <label for="password" class="block text-
gray-700 text-sm font-bold mb-2">Password</label>
          <input type="password" id="password"
name="password" required class="w-full px-3 py-2
border rounded-lg focus:outline-none focus:ring-2
focus:ring-blue-500 focus:border-transparent" />
        </div>
        <div class="flex items-center justify-
between">
          <button type="submit" class="w-full bg-
blue-500 hover:bg-blue-700 text-white font-bold py-2
px-4 rounded focus:outline-none focus:shadow-
outline">LogIn</button>
        </div>
      </form>
    </div>

  </body>
  <script>// Menghilangkan pesan setelah 5 detik
Pesan harus login
      setTimeout(function() {
        const flashMessage =
document.querySelector('.flash-message');
```

```
        if (flashMessage) {
          flashMessage.classList.add('opacity-
0');  // Menggunakan Tailwind untuk fade-out
          setTimeout(() => flashMessage.style.display
= 'none', 300);  // Setelah fade-out selesai,
sembunyikan
        }
      }, 3000);  // 5000ms = 5 detik
    </script>
</html>
```

Source Code Function Login

```
@app.route('/', methods=['GET', 'POST'])
def login():
    if request.method == 'POST':
        username = request.form['username']
        password = request.form['password']

        user =
User.query.filter_by(username=username,
password=password).first()
        if user:
            login_user(user)
            return redirect(url_for('inputdata'))
        # render_template("dashboard.html") #
Mengarahkan ke dashboard
        else:
            flash('Invalid username or password',
'error')
            return render_template('login.html')
    return render_template('login.html')

# Tambahkan pengaturan untuk menangani pesan ketika
user belum login
@login_manager.unauthorized_handler
def unauthorized():
    flash('Tolong login terlebih dahulu', 'info')  #
Pesan flash khusus
    return redirect(url_for('login'))  # Redirect ke
halaman login jika tidak login
```

Source Code base.hmtl digunakan untuk *layout* utama sistem

```
<!DOCTYPE html>
<html lang="en">

<head>
  <meta charset="UTF-8" />
  <meta name="viewport" content="width=device-width,
initial-scale=1.0" />
  <link href="{{ url_for('static',
filename='dist/output.css') }}" rel="stylesheet" />
  <title>Clustering apps</title>
  <style>
    /* CSS Custom untuk Active State */
    .active {
      background-color: #22c55e;
      /* bg-green-500 */
    }

    table {
      font-size: 15px;
      /* Ukuran font yang lebih kecil */
    }

    .centerkolom {
      padding: 4px 8px;
      text-align: center;
      /* Mengecilkan padding */
    }

    .table-container {
      max-width: 90%;
      /* Batasi lebar tabel */
      margin: auto;
      /* Agar tabel berada di tengah */
    }

    button.buttonkolom {
      align-items: center;
      font-size: 10px;
      /* Ukuran font tombol lebih kecil */
      padding: 4px 8px;
      /* Mengecilkan tombol */
    }
    .action-buttons {
```

```
    display: flex;
    justify-content: center; /* Pusatkan tombol
secara horizontal */
    align-items: center; /* Pusatkan tombol secara
vertikal */
  }
  </style>
</head>

<body class="bg-gray-100">
  <div class="min-h-screen flex">
    <!-- Sidebar / Navigation Bar -->
    <nav class="bg-gray-800 text-white fixed h-full"
style="width:200px;">
      <div class="p-6">
        <h1 class="text-2xl font-bold mb-6">Sistem
Clustering</h1>
        <ul>
          <li class="mb-4 mt-2">
            <a href="/inputdata"
              class="block py-2 px-4 text-lg
hover:bg-green-600 rounded {% if request.path ==
'/inputdata' %}active{% endif %}">Input
              Data</a>
          </li>
          <li class="mb-4 mt-2">
            <a href="/prosesdata"
              class="block py-2 px-4 text-lg
hover:bg-green-600 rounded {% if request.path ==
'/prosesdata' %}active{% endif %}">Proses
              Data</a>
          </li>
          <li class="mb-4 mt-2"></li>
          <a href="/proseslanjut"
            class="block py-2 px-4 text-lg hover:bg-
green-600 rounded {% if request.path ==
'/proseslanjut' %}active{% endif %}">Proses
            Lanjut</a>
          </li>
          <li class="mb-4 mt-2">
            <a href="/hasil"
              class="block py-2 px-4 text-lg
hover:bg-green-600 rounded {% if request.path ==
'/hasil' %}active{% endif %}">Hasil
              Clustering</a>
          </li>
          <li class="mb-4 mt-2">
```

```
            <a href="/visualisasi"
              class="block py-2 px-4 text-lg
hover:bg-green-600 rounded {% if request.path ==
'/visualisasi' %}active{% endif %}">Visualisasi
              Data</a>
          </li>
          <li class="mb-4 mt-2">
            <a  href="{{ url_for('logout') }}"
class="block py-2 px-4 text-lg bg-red-500 hover:bg-
red-400 rounded ">Logout</a>
          </li>
        </ul>
      </div>
    </nav>
    <!-- Main Content -->
    <div class="ml-[200px] w-full">
      {% block content %}
      <!-- Content goes here -->
      {% endblock %}
    </div>
  </div>
</body>
<script src="{{ url_for('static',
filename='src/scripts.js') }}"></script>
</html>
```

Source Code Halaman Input Data

```
{% extends 'base.html' %}

{% block content %}
<div class="flex-1 p-10">
  <h1 class="font-bold mb-6" style="font-size: xx-
large;">Halaman Input Data</h1>
  <div class="bg-gray-100">
    <!-- Button to open the modal -->
    <div class="flex justify-end space-x-4">
      <button id="openManualAdd"
        class="bg-blue-500 text-white px-4 py-2
rounded-md hover:bg-blue-600 focus:outline-
none">Tambah Data
        Manual
      </button>
      <button id="openCsvAdd"
```

```
        class="bg-green-500 text-white px-4 py-2
rounded-md hover:bg-green-700 focus:outline-
none">Tambah Data
        CSV
      </button>
      <form action="/hapusdatainput" method="POST">
        <button type="submit" class="bg-red-500
hover:bg-red-700 text-white font-bold py-2 px-4
rounded">
          Hapus Data
        </button>
      </form>
    </div>

    <!--alert data sudah di hapus-->
  {% with messages =
get_flashed_messages(with_categories=true) %}
    {% if messages %}
      <div class="space-y-4">
        {% for category, message in messages %}
          {% if category == 'success' %}
            <div class="flash-message bg-green-500
text-white py-2 px-4 mt-2 mb-2 rounded-md shadow-md
text-center">
              {{ message }}
            </div>
          {% elif category == 'error' %}
            <div class="flash-message bg-red-500
text-white py-2 px-4 mt-2 mb-2 rounded-md shadow-md
text-center">
              {{ message }}
            </div>
          {% endif %}
        {% endfor %}
      </div>
    {% endif %}
  {% endwith %}

    <!-- Modal Manual Add -->
    <div id="modalManualAdd" class="flex hidden fixed
inset-0 bg-gray-900 bg-opacity-50 items-center
justify-center">
      <div class="bg-white rounded-lg shadow-lg w-1/3
p-6">
        {% with messages =
get_flashed_messages(with_categories=True) %}
        {% if messages %}
```

```
        <ul>
          {% for category, message in messages %}
          <li class="{{ category }}">{{ message
}}</li>
          {% endfor %}
        </ul>
        {% endif %}
        {% endwith %}
        <h2 class="text-lg font-semibold mb-4">Tambah
Data Manual</h2>
        <!-- Form for input -->
        <form action="/submit" method="POST">
          <div class="mb-4">
            <label for="tanggal" class="block text-
gray-700 text-sm font-bold mb-2">Tanggal:</label>
            <input type="date" name="tanggal"
id="tanggal"
              class="shadow appearance-none border
rounded w-full py-2 px-3 text-gray-700 leading-tight
focus:outline-none focus:shadow-outline"
              required />
          </div>
          <div class="mb-4">
            <label for="waktu" class="block text-
gray-700 text-sm font-bold mb-2">Waktu
(MM.SS):</label>
            <input type="text" name="waktu"
id="waktu" pattern="^[0-9]{1,2}\.[0-9]{2}$"
              class="shadow appearance-none border
rounded w-full py-2 px-3 text-gray-700 leading-tight
focus:outline-none focus:shadow-outline"
              required>
            <p class="text-gray-600 text-xs
italic">Format: MM.SS (Contoh: 31.23 untuk 31 menit
23 detik)</p>
          </div>
          <div class="mb-4">
            <label for="latitude" class="block text-
gray-700">Latitude</label>
            <input type="text" id="latitudeForm"
name="latitude"
              class="border border-gray-300 rounded-
md w-full p-2 focus:ring focus:ring-blue-200"
required />
          </div>
          <div class="mb-4">
```

```
            <label for="longitude" class="block text-
gray-700">Longitude</label>
            <input type="text" id="longitudeForm"
name="longitude"
              class="border border-gray-300 rounded-
md w-full p-2 focus:ring focus:ring-blue-200"
required />
          </div>
          <div class="mb-4">
            <label for="magnitude" class="block text-
gray-700">Magnitude</label>
            <input type="text" id="magnitudeForm"
name="magnitude"
              class="border border-gray-300 rounded-
md w-full p-2 focus:ring focus:ring-blue-200"
required />
          </div>
          <div class="mb-4">
            <label for="depth" class="block text-
gray-700">Depth</label>
            <input type="text" id="depthForm"
name="depth"
              class="border border-gray-300 rounded-
md w-full p-2 focus:ring focus:ring-blue-200"
required />
          </div>

          <!-- Modal action buttons -->
          <div class="flex justify-end space-x-4">
            <button type="button"
id="closeModalManualAdd"
              class="bg-gray-500 text-white px-4 py-2
rounded-md hover:bg-gray-600">Batal</button>
            <button type="submit" class="bg-blue-500
text-white px-4 py-2 rounded-md hover:bg-blue-
600">Simpan</button>
          </div>
        </form>
      </div>
    </div>
    <h1 class="text-2xl font-bold mt-5 mb-5">Data
Gempa Bumi</h1>

    <!--Tabel data Gempa-->
    <div class="table-container"></div>
    <table class="min-w-full bg-white border border-
gray-300">
```

```
      <thead>
        <tr class="bg-neutral-400">
          <th class="py-1 px-2 border
centerkolom">No</th>
          <th class="py-1 px-2 border
centerkolom">Tanggal</th>
          <th class="py-1 px-2 border
centerkolom">Waktu</th>
          <th class="py-1 px-2 border
centerkolom">Latitude</th>
          <th class="py-1 px-2 border
centerkolom">Longitude</th>
          <th class="py-1 px-2 border
centerkolom">Magnitude</th>
          <th class="py-1 px-2 border
centerkolom">Depth</th>
          <th class="py-1 px-2 border
centerkolom">Aksi</th>
        </tr>
      </thead>
      <tbody>
        {% for eq in earthquakes %}
        <tr class="hover:bg-gray-200">
          <td class="py-2 px-4 border centerkolom">{{
loop.index + (page - 1) * 100 }}</td>
          <td class="py-1 px-2 border centerkolom">{{
eq.tanggal }}</td>
          <td class="py-1 px-2 border centerkolom">{{
eq.waktu }}</td>
          <td class="py-1 px-2 border centerkolom">{{
eq.latitude }}</td>
          <td class="py-1 px-2 border centerkolom">{{
eq.longitude }}</td>
          <td class="py-1 px-2 border centerkolom">{{
eq.magnitude }}</td>
          <td class="py-1 px-2 border centerkolom">{{
eq.depth }}</td>
          <td class="py-1 px-2 border ">
            <div class="flex items-center action-
buttons">
              <button
                onclick="openEditModal({{ eq.id }},
'{{ eq.tanggal }}', '{{ eq.waktu }}', {{ eq.latitude
}}, {{ eq.longitude }}, {{ eq.magnitude }}, {{
eq.depth }})"
```

```
                class="bg-green-500 text-white
rounded-md hover:bg-green-600 ml-2
buttonkolom">Ubah</button>
              <form action="/delete/{{ eq.id }}"
method="POST">
                <button type="submit"
                  class="ml-2 bg-red-500 text-white
px-4 py-2 rounded-md hover:bg-red-600
buttonkolom">Hapus</button>
              </form>
            </div>
          </td>
        </tr>
        {% endfor %}
      </tbody>
    </table>
    <!-- Pagination -->
    <div class="pagination mt-4 flex justify-center
overflow-x-auto">
      <!-- Previous Button -->
      {% if page > 1 %}
      <a href="?page={{ page - 1 }}"
        class="bg-blue-500 text-white px-4 py-2
rounded-md hover:bg-blue-600 mr-2">Previous</a>
      {% endif %}

      <!-- Page Number Buttons with Scrolling -->
      <div class="flex space-x-1 overflow-x-auto"
style="max-width: 200px;">
        {% if total_pages <= 5 %} <!-- Display all
pages if total pages <=5 -->
          {% for p in range(1, total_pages + 1) %}
          <a href="?page={{ p }}"
            class="bg-gray-300 text-black px-3 py-2
rounded-md hover:bg-gray-400 {{ 'font-bold' if p ==
page else '' }}">
            {{ p }}
          </a>
          {% endfor %}
          {% else %}
          <!-- Conditionally display pages with
ellipsis if there are more than 5 pages -->
          {% if page > 2 %}
          <a href="?page=1" class="bg-gray-300 text-
black px-3 py-2 rounded-md hover:bg-gray-400">1</a>
          {% if page > 3 %}<span class="px-
2">...</span>{% endif %}
```

```
          {% endif %}

          <!-- Display 2 pages before and after the
current page -->
          {% for p in range(max(1, page - 2),
min(total_pages, page + 2) + 1) %}
          <a href="?page={{ p }}"
            class="bg-gray-300 text-black px-3 py-2
rounded-md hover:bg-gray-400 {{ 'font-bold' if p ==
page else '' }}">
            {{ p }}
          </a>
          {% endfor %}

          {% if page < total_pages - 2 %} {% if page
< total_pages - 3 %}<span class="px-2">...</span>{%
endif %}
            <a href="?page={{ total_pages }}"
class="bg-gray-300 text-black px-3 py-2 rounded-md
hover:bg-gray-400">{{
              total_pages }}</a>
            {% endif %}
            {% endif %}
      </div>

      <!-- Next Button -->
      {% if page < total_pages %} <a href="?page={{
page + 1 }}"
        class="bg-blue-500 text-white px-4 py-2
rounded-md hover:bg-blue-600 ml-2">Next</a>
        {% endif %}
    </div>
  </div>

  <!--Modal Edit data gempa-->
  <div id="editModal" class="fixed flex inset-0 bg-
gray-600 hidden bg-opacity-50 items-center justify-
center">
    <div class="bg-white p-6 rounded-md w-1/3">
      <h2 class="text-xl font-bold mb-4">Edit Data
Gempa</h2>
      <form action="/update" method="POST">
        <input type="hidden" id="editId"
name="id_gempa">
        <div class="mb-4">
          <label for="editTanggal" class="block text-
gray-700 text-sm font-bold mb-2">Tanggal:</label>
```

```
          <input type="date" id="editTanggal"
name="tanggal"
            class="shadow appearance-none border
rounded w-full py-2 px-3 text-gray-700 leading-tight
focus:outline-none focus:shadow-outline"
            required>
        </div>
        <div class="mb-4">
          <label for="editWaktu" class="block text-
gray-700 text-sm font-bold mb-2">Waktu
(MM.SS):</label>
          <input type="text" id="editWaktu"
name="waktu" pattern="^\d{1,2}\.\d{2}$"
            class="shadow appearance-none border
rounded w-full py-2 px-3 text-gray-700 leading-tight
focus:outline-none focus:shadow-outline"
            required>
        </div>
        <div class="mb-4">
          <label for="editLatitude" class="block
text-gray-700">Latitude</label>
          <input type="text" id="editLatitude"
name="latitude"
            class="border border-gray-300 rounded-md
w-full p-2 focus:ring focus:ring-blue-200" required
/>
        </div>
        <div class="mb-4">
          <label for="editLongitude" class="block
text-gray-700">Longitude</label>
          <input type="text" id="editLongitude"
name="longitude"
            class="border border-gray-300 rounded-md
w-full p-2 focus:ring focus:ring-blue-200" required
/>
        </div>
        <div class="mb-4">
          <label for="editMagnitude" class="block
text-gray-700">Magnitude</label>
          <input type="text" id="editMagnitude"
name="magnitude"
            class="border border-gray-300 rounded-md
w-full p-2 focus:ring focus:ring-blue-200" required
/>
        </div>
        <div class="mb-4">
```

```
          <label for="editDepth" class="block text-
gray-700">Depth</label>
          <input type="text" id="editDepth"
name="depth"
            class="border border-gray-300 rounded-md
w-full p-2 focus:ring focus:ring-blue-200" required
/>
        </div>

        <!-- Tombol di dalam Modal -->
        <div class="flex justify-end space-x-4">
          <button type="button"
onclick="closeEditModal()"
            class="bg-gray-500 text-white px-4 py-2
rounded-md hover:bg-gray-600">Batal</button>
          <button type="submit" class="bg-blue-500
text-white px-4 py-2 rounded-md hover:bg-blue-
600">Simpan
            Perubahan</button>
        </div>
      </form>
    </div>
  </div>

  <!--Modal CSV Add data-->
  <div id="csvModal" class="fixed flex inset-0 bg-
gray-600 bg-opacity-50 hidden items-center justify-
center">
    <div class="bg-white p-6 rounded-md w-1/3">
      <h2 class="text-xl font-bold mb-4">Upload File
CSV</h2>
      <!-- Form upload CSV -->
      <form action="/upload" method="POST"
enctype="multipart/form-data">
        <div id="dropArea" class="border-2 border-
dashed border-gray-400 p-4 rounded-md text-center mb-
4">
          <p class="text-gray-600 mb-2">Seret dan
letakkan file CSV Anda di sini</p>
          <p class="text-gray-600 mb-2">atau</p>
          <label for="fileInput"
            class="bg-blue-500 text-white px-4 py-2
rounded-md cursor-pointer hover:bg-blue-600">Pilih
File</label>
        </div>
        <input type="file" id="fileInput" name="file"
class="hidden">
```

```
        <!-- Tombol di dalam modal -->
        <div class="flex justify-end space-x-4">
          <button type="button"
onclick="closeCsvModal()"
            class="bg-gray-500 text-white px-4 py-2
rounded-md hover:bg-gray-600">Batal</button>
          <button type="submit" class="bg-blue-500
text-white px-4 py-2 rounded-md hover:bg-blue-
600">Upload</button>
        </div>
      </form>
    </div>
  </div>
</div>
</div>
{% endblock %}
```

Source Code Function Halaman Input Data

```
@app.route("/inputdata")
@login_required
def inputdata():
   # Ambil halaman dari query parameter atau default
ke halaman 1
    page = request.args.get('page', 1, type=int)
    per_page = 100  # Jumlah data per halaman

    # Hitung total data
    total_data = Earthquake.query.count()

    # Hitung total halaman
    total_pages = ceil(total_data / per_page)

    # Ambil data sesuai halaman dan batasi 100 data
per halaman
    earthquakes = Earthquake.query.offset((page - 1)
* per_page).limit(per_page).all()

    # Render template dan kirim data untuk paginasi
    return render_template('inputdata.html',
earthquakes=earthquakes, page=page,
total_pages=total_pages)

@app.route("/submit", methods=["POST"])
@login_required
```

```
def submit():
    tanggal = request.form['tanggal']
    waktu = request.form['waktu']
    latitude = float(request.form['latitude'])
    longitude = float(request.form['longitude'])
    magnitude = float(request.form['magnitude'])
    depth = float(request.form['depth'])

    # Membuat objek Earthquake baru
    new_earthquake = Earthquake(tanggal=tanggal,
waktu=waktu, latitude=latitude,
                                longitude=longitude,
magnitude=magnitude, depth=depth)


    # Menyimpan ke database
    db.session.add(new_earthquake)
    db.session.commit()

    flash('Data gempa bumi berhasil disimpan!',
'success')
    return redirect(url_for('inputdata'))

#hapus data
@app.route("/delete/<int:id>", methods=["POST"])
@login_required
def delete(id):
    # Ambil data gempa yang ingin dihapus berdasarkan
id_gempa
    earthquake_to_delete =
Earthquake.query.get_or_404(id)

    try:
        # Hapus data dari database
        db.session.delete(earthquake_to_delete)
        db.session.commit()
        flash('Data gempa berhasil dihapus',
'success')
    except:
        db.session.rollback()
        flash('Terjadi kesalahan saat menghapus
data', 'error')

    return redirect("/inputdata")

#edit data gempa
@app.route("/update", methods=["POST"])
@login_required
```

```
def update():
    # Tangkap data dari form edit
    id_gempa = request.form.get('id_gempa')
    earthquake = Earthquake.query.get(id_gempa)

    # Update field berdasarkan input dari form
    earthquake.tanggal = request.form['tanggal']
    earthquake.waktu = request.form['waktu']
    earthquake.latitude = request.form['latitude']
    earthquake.longitude = request.form['longitude']
    earthquake.magnitude = request.form['magnitude']
    earthquake.depth = request.form['depth']

    db.session.commit()
    flash('Data berhasil diperbarui', 'success')

    return redirect("/inputdata")

# Add data csv
# Rute untuk menangani upload file CSV
@app.route('/upload', methods=['POST'])
@login_required
def upload():
    if 'file' not in request.files:
        return "No file part"

    file = request.files['file']

    if file.filename == '':
        return "No selected file"

    # Baca CSV dengan pandas
    try:
        data = pd.read_csv(file)

        # Loop dan masukkan ke database
        for index, row in data.iterrows():

           # Pastikan bahwa nilai 'tanggal' bukan NaN
            if pd.notnull(row['tanggal']) and
isinstance(row['tanggal'], str):
                try:
                    # Ubah format parsing tanggal ke
'm/d/Y'
                    tanggal =
datetime.strptime(row['tanggal'], '%m/%d/%Y')
                except ValueError:
```

```
                    return f"Invalid date format in
row {index + 1}"
            else:

                return f"Missing or invalid date in
row {index + 1}"


            new_item = Earthquake(
                tanggal=tanggal,
                waktu=row['waktu'],
                latitude=row['latitude'],
                longitude=row['longitude'],
                depth=row['depth'],
                magnitude=row['magnitude'])
            db.session.add(new_item)
        db.session.commit()
        return redirect("/inputdata")

    except Exception as e:
        return f"Error: {e}"

@app.route('/hapusdatainput', methods=['POST'])
@login_required
def hapus_data():
    db.session.query(ClusteredEarthquake).delete()
    db.session.query(Earthquake).delete()
    db.session.commit()
    flash('Data berhasil dihapus!', 'error')
    return redirect(url_for('inputdata'))
```

Source Code Halaman Proses Data

```
{% extends 'base.html' %}

{% block content %}
<div class="flex-1 p-10">
    <h1 class="font-bold mb-6" style="font-size: xx-
large;">Halaman Proses Data</h1>
    <div class="bg-gray-100">
        <h2 class="font-bold mb-6 text-center
decoration-solid" style="font-size: x-large;">Hasil
Normalisasi Data</h2>

        <!-- Hasil Normalisasi Ditampilkan di Tabel -
->
        {% if data %}
```

```
        <table class="min-w-full bg-white border
border-gray-300">
            <thead>
                <tr class="bg-neutral-400">
                    <th class="px-4 py-2 border-
b">Latitude</th>
                    <th class="px-4 py-2 border-
b">Longitude</th>
                    <th class="px-4 py-2 border-
b">Magnitude</th>
                    <th class="px-4 py-2 border-
b">Depth</th>

                </tr>
            </thead>
            <tbody>
                {% for row in data %}
                <tr>
                    <td class="border px-4 py-2
border-b">{{ row[0] }}</td>
                    <td class="border px-4 py-2
border-b">{{ row[1] }}</td>
                    <td class="border px-4 py-2
border-b">{{ row[2] }}</td>
                    <td class="border px-4 py-2
border-b">{{ row[3] }}</td>
                </tr>
                {% endfor %}
            </tbody>
        </table>
        {% endif %}

        <div class="pagination mt-4 flex justify-
center overflow-x-auto">
            <!-- Previous Button -->
            {% if page > 1 %}
            <a href="?page={{ page - 1 }}"
                class="bg-blue-500 text-white px-4
py-2 rounded-md hover:bg-blue-600 mr-2">Previous</a>
            {% endif %}

            <!-- Page Number Buttons with Scrolling -
->
            <div class="flex space-x-1 overflow-x-
auto" style="max-width: 200px;">
                {% if total_pages <= 5 %} <!--
Display all pages if total pages <=5 -->
```

```
                    {% for p in range(1, total_pages
+ 1) %}
                    <a href="?page={{ p }}"
                        class="bg-gray-300 text-black
px-3 py-2 rounded-md hover:bg-gray-400 {{ 'font-bold'
if p == page else '' }}">
                        {{ p }}
                    </a>
                    {% endfor %}
                    {% else %}
                    <!-- Conditionally display pages
with ellipsis if there are more than 5 pages -->
                    {% if page > 2 %}
                    <a href="?page=1" class="bg-gray-
300 text-black px-3 py-2 rounded-md hover:bg-gray-
400">1</a>
                    {% if page > 3 %}<span class="px-
2">...</span>{% endif %}
                    {% endif %}

                    <!-- Display 2 pages before and
after the current page -->
                    {% for p in range(max(1, page -
2), min(total_pages, page + 2) + 1) %}
                    <a href="?page={{ p }}"
                        class="bg-gray-300 text-black
px-3 py-2 rounded-md hover:bg-gray-400 {{ 'font-bold'
if p == page else '' }}">
                        {{ p }}
                    </a>
                    {% endfor %}

                    {% if page < total_pages - 2 %}
{% if page < total_pages - 3 %}<span class="px-
2">...</span>{% endif
                        %}
                        <a href="?page={{ total_pages
}}"
                            class="bg-gray-300 text-
black px-3 py-2 rounded-md hover:bg-gray-400">{{
total_pages }}</a>
                        {% endif %}
                        {% endif %}
            </div>


            <!-- Next Button -->
```

```
            {% if page < total_pages %} <a
href="?page={{ page + 1 }}"
                class="bg-blue-500 text-white px-4
py-2 rounded-md hover:bg-blue-600 ml-2">Next</a>
                {% endif %}
        </div>

    </div>
</div>
{% endblock %}
```

Source Code Function Halaman Proses Data

```
@app.route('/prosesdata', methods=['GET'])
@login_required
def prosesdata():
    # Get the current page number from the request
(default to page 1)
    page = request.args.get('page', 1, type=int)

    # Set the number of records per page (in this
case, 100)
    records_per_page = 100

    # Query the data and paginate
    earthquakes = Earthquake.query.with_entities(
        Earthquake.latitude,
        Earthquake.longitude,
        Earthquake.magnitude,
        Earthquake.depth
    ).paginate(page=page, per_page=records_per_page,
error_out=False)

    # Extract data from the paginated result
    data_array = np.array([(eq.latitude,
eq.longitude, eq.magnitude, eq.depth) for eq in
earthquakes.items])

    # Normalize the data
    scaler = MinMaxScaler()
    normalized_data =
scaler.fit_transform(data_array)

    # Convert the normalized data to a list
    normalized_data_list = normalized_data.tolist()
```

```
    # Calculate the total number of pages
    total_pages = earthquakes.pages

    return render_template(
        'prosesdata.html',
        data=normalized_data_list,
        page=page,
        total_pages=total_pages
    )
```

Source Code Halaman Proses Data Lanjut

```
{% extends 'base.html' %}

{% block content %}
<div class="flex-1 p-10">
    <h1 class="font-bold mb-6" style="font-size: xx-
large;">Halaman Proses Data Lanjut</h1>
    <div class="bg-gray-100">
<!-- Tampilkan hasil jika sudah ada data plot_url -->
        {% if plot_url %}
        <div class="flex justify-center items-start">
            <div class="w-1/2">
                <!-- Grafik Silhouette -->
                <div class="text-center">
                    <img
src="{{ plot_url }}"
alt="Silhouette Score Plot" class="mx-auto">
                </div>
            </div>
            <div class="w-1/2" style="margin-left:
50px;">
                <div class=" text-left pl-8"></div>
                <h3 class="text-lg font-
semibold">Skor Silhouette untuk Setiap Kluster:</h3>
                <ul class="list-disc list-inside">
                    {% for n_clusters, silhouette_avg
in silhouette_data %}
                    <li>Jumlah Kluster = {{
n_clusters }}, Skor Silhouette = {{ silhouette_avg
}}</li>
                    {% endfor %}
                </ul>
```

```
                <div class="p-2 mt-4 bg-green-500"
style="width: 300px;">
                    <h2 class="text-xl ">Jumlah
kluster terbaik adalah: {{ optimal_clusters }}</h2>
                </div>
            </div>
        </div>

        <!-- Tampilkan Centroid di Tabel -->
        <h2 class="font-bold mb-6 decoration-solid
mt-6" style="font-size: x-large;">Hasil Centroid</h2>
        {% if fcm_centers %}
        <table class="min-w-full bg-white border
border-gray-300">
            <thead>
                <tr class="bg-gray-300">
                    <th class="border px-4 py-2
border-gray-400 text-center">Kluster</th>
                    <th class="border px-4 py-2
border-gray-400 text-center ">Latitude</th>
                    <th class="border px-4 py-2
border-gray-400 text-center ">Longitude</th>
                    <th class="border px-4 py-2
border-gray-400 text-center ">Magnitude</th>
                    <th class="border px-4 py-2
border-gray-400 text-center ">Depth</th>
                </tr>
            </thead>
            <tbody>
                {% for center in fcm_centers %}
                <tr>
                    <td class="border px-4 py-2
border-gray-400 text-center">Kluster {{ loop.index
}}</td>
                    <td class="border px-4 py-2
border-gray-400 text-center">{{ center[0] }}</td>
                    <td class="border px-4 py-2
border-gray-400 text-center">{{ center[1] }}</td>
                    <td class="border px-4 py-2
border-gray-400 text-center">{{ center[2] }}</td>
                    <td class="border px-4 py-2
border-gray-400 text-center">{{ center[3] }}</td>
                </tr>
                {% endfor %}
            </tbody>
        </table>
        {% endif %}
```

```
    </div>
</div>
</div>
</div>
<!-- Tampilkan centroid kluster dalam bentuk tabel --
>

{% endif %}
</div>
</div>
</div>
{% endblock %}
```

Source Code Function Halaman Proses Data Lanjut

```
optimal_clusters= None
standardized_data=None
@app.route('/proseslanjut', methods=['GET', 'POST'])
@login_required
def proseslanjut():
    global optimal_clusters
    plot_url = None
    optimal_clusters = None
    fcm_centers = None
    silhouette_data = []  # Menyimpan data jumlah
kluster dan skor silhouette

    # Ambil data dari kolom latitude, longitude,
magnitude, dan depth
    earthquakes = Earthquake.query.with_entities(
            Earthquake.latitude,
            Earthquake.longitude,
            Earthquake.magnitude,
            Earthquake.depth
    ).all()

    # Ubah data ke dalam format numpy array
    data_array = np.array(earthquakes)

    # Inisialisasi MinMaxScaler
    scaler = MinMaxScaler()

    # Normalisasi data
    standardized_data =
scaler.fit_transform(data_array)
```

```
    # Pencarian jumlah kluster terbaik menggunakan
FCM dan Silhouette Score
    cluster_range = range(2, 10)
    silhouette_scores = []

    for n_clusters in cluster_range:
        # Fuzzy C-Means clustering
        fcm = FCM(n_clusters=n_clusters,
max_iter=100, m=2, error=0.01)
        fcm.fit(standardized_data)

        # Menentukan keanggotaan terbesar untuk
setiap titik data
        cluster_labels = np.argmax(fcm.u, axis=1)

        # Menghitung skor silhouette
        silhouette_avg =
silhouette_score(standardized_data, cluster_labels)
        silhouette_scores.append(silhouette_avg)

        # Simpan jumlah kluster dan skor silhouette
ke dalam list
        silhouette_data.append((n_clusters,
silhouette_avg))

    # Menentukan jumlah kluster terbaik berdasarkan
skor silhouette tertinggi
    optimal_clusters =
cluster_range[np.argmax(silhouette_scores)]
    # Sekarang gunakan jumlah kluster terbaik untuk
FCM
    fcm = FCM(n_clusters=optimal_clusters,
max_iter=100, m=2, error=0.01)
    fcm.fit(standardized_data)

    # Ambil pusat kluster (centroid)
    fcm_centers = fcm.centers.tolist()

    # Plot Skor Silhouette
    plt.figure()
    plt.plot(cluster_range, silhouette_scores,
marker='o')
    plt.xlabel('Jumlah Kluster')
    plt.ylabel('Skor Silhouette')
```

```
    plt.title('Skor Silhouette untuk berbagai jumlah
kluster')

    # Simpan plot ke dalam buffer
    img = io.BytesIO()
    plt.savefig(img, format='png')
    img.seek(0)
    plot_url =
base64.b64encode(img.getvalue()).decode()

    # Kirim plot, centroid kluster, dan data
silhouette ke template
    return render_template('proseslanjut.html',
plot_url=plot_url, optimal_clusters=optimal_clusters,
fcm_centers=fcm_centers,
silhouette_data=silhouette_data)
closest_events = []
```

Source Code Halaman Hasil

```
{% extends 'base.html' %}

{% block content %}

<div class="bg-gray-100">
    <div class="container mx-auto p-4">
        <h1 class="text-3xl font-bold text-center
text-gray-800 mb-6">
            Hasil Clustering
        </h1>
        <h2 class="text-xl">Jumlah kluster terbaik
adalah: {{ optimal }}</h2>
        <div class="overflow-x-auto">
            <table class="min-w-full bg-white border
border-gray-300">
                <thead class="bg-gray-200">
                    <tr>
                        <th class="px-4 py-2 border
border-gray-300 text-left">No</th>
```

```
                        <th class="px-4 py-2 border
border-gray-300 text-left">Tanggal</th>
                        <th class="px-4 py-2 border
border-gray-300 text-left">Waktu</th>
                        <th class="px-4 py-2 border
border-gray-300 text-left">Latitude</th>
                        <th class="px-4 py-2 border
border-gray-300 text-left">
                            Longitude
                        </th>
                        <th class="px-4 py-2 border
border-gray-300 text-left">
                            Magnitude
                        </th>
                        <th class="px-4 py-2 border
border-gray-300 text-left">Depth</th>
                        <th class="px-4 py-2 border
border-gray-300 text-left">Cluster</th>
                    </tr>
                </thead>
                <tbody>
                    {% for data, label in
labeled_data %}
                    <tr class="hover:bg-gray-100">
                        <td class="px-4 py-2 border
border-gray-300">
                            {{ (current_page - 1) *
100 + loop.index }}
                        </td>
                        <td class="px-4 py-2 border
border-gray-300">{{ data[1] }}</td>
                        <td class="px-4 py-2 border
border-gray-300">{{ data[2] }}</td>
                        <td class="px-4 py-2 border
border-gray-300">{{ data[3] }}</td>
                        <td class="px-4 py-2 border
border-gray-300">{{ data[4] }}</td>
                        <td class="px-4 py-2 border
border-gray-300">{{ data[5] }}</td>
                        <td class="px-4 py-2 border
border-gray-300">{{ data[6] }}</td>
                        <td class="px-4 py-2 border
border-gray-300">{{ label }}</td>
                    </tr>
                    {% endfor %}
                </tbody>
            </table>
```

```
            <!-- Pagination -->
            <div class="pagination mt-4 flex justify-
center overflow-x-auto">
                <!-- Previous Button -->
                {% if current_page > 1 %}
                <a href="{{ url_for('hasil',
page=current_page-1) }}"
                    class="bg-blue-500 text-white px-
4 py-2 rounded-md hover:bg-blue-600 mr-2">&laquo;
Previous</a>
                {% endif %}

                <!-- Page Number Buttons with Scroll
and Ellipses -->
                <div class="flex space-x-1 overflow-
x-auto" style="max-width: 200px">
                    {% if total_pages <= 5 %} <!--
Display all pages if total pages <=5 -->
                        {% for p in range(1,
total_pages + 1) %}
                        <a href="{{ url_for('hasil',
page=p) }}"
                            class="bg-gray-300 text-

black px-3 py-2 m-1 rounded-md hover:bg-gray-400 {{
'font-bold' if p == current_page else '' }}">{{
                            p }}</a>
                        {% endfor %} {% else %}
                        <!-- Display first page and
ellipsis if current page > 2 -->
                        {% if current_page > 2 %}
                        <a href="{{ url_for('hasil',
page=1) }}"
                            class="bg-gray-300 text-
black px-3 py-2 m-1 rounded-md hover:bg-gray-
400">1</a>
                        {% if current_page > 3
%}<span class="px-2">...</span>{% endif %} {%
                        endif %}

                        <!-- Display 2 pages before
and after the current page -->
                        {% for p in range(max(1,
current_page - 2), min(total_pages,
                        current_page + 2) + 1) %}
                        <a href="{{ url_for('hasil',
page=p) }}"
```

```
                                class="bg-gray-300 text-
black px-3 py-2 m-1 rounded-md hover:bg-gray-400 {{
'font-bold' if p == current_page else '' }}">{{
                                p }}</a>
                            {% endfor %}

                            <!-- Display last page and
ellipsis if current page is far from the end -->
                            {% if current_page <
total_pages - 2 %} {% if current_page < total_pages -
3 %}<span
                                class="px-2">...</span>{%
endif %}
                                <a href="{{
url_for('hasil', page=total_pages) }}"
                                    class="bg-gray-300
text-black px-3 py-2 m-1 rounded-md hover:bg-gray-
400">{{ total_pages
                                    }}</a>
                                {% endif %} {% endif %}
                </div>


                <!-- Next Button -->
                {% if current_page < total_pages %}
<a href="{{ url_for('hasil', page=current_page+1) }}"
                    class="bg-blue-500 text-white px-
4 py-2 rounded-md hover:bg-blue-600 ml-2">Next
&raquo;</a>
                    {% endif %}
            </div>
        </div>

        <div class="mb-4">
            <h2 class="text-xl">Centroid Sesuai
Data</h2>
            <div class="overflow-x-auto">
                <table class="min-w-full bg-white
border border-gray-300">
                    <thead class="bg-gray-200">
                        <tr>
                            <th class="px-4 py-2
border border-gray-300 text-left">Centroid</th>
                            <th class="px-4 py-2
border border-gray-300 text-left">No</th>
                            <th class="px-4 py-2
border border-gray-300 text-left">Date</th>
                            <th class="px-4 py-2
border border-gray-300 text-left">Time</th>
```

```
                            <th class="px-4 py-2
border border-gray-300 text-left">Latitude</th>
                            <th class="px-4 py-2
border border-gray-300 text-left">Longitude</th>
                            <th class="px-4 py-2
border border-gray-300 text-left">Depth</th>
                            <th class="px-4 py-2
border border-gray-300 text-left">Mag</th>
                        </tr>
                    </thead>
                    <tbody>
                        {% for event in
closest_events %}
                        <tr>
                            <td class="px-4 py-2
border border-gray-300">{{ event.Centroid }}</td>
                            <td class="px-4 py-2
border border-gray-300">{{ event.No }}</td>
                            <td class="px-4 py-2
border border-gray-300">{{ event.Date }}</td>
                            <td class="px-4 py-2
border border-gray-300">{{ event.Time }}</td>
                            <td class="px-4 py-2
border border-gray-300">{{ event.Latitude }}</td>
                            <td class="px-4 py-2
border border-gray-300">{{ event.Longitude }}</td>
                            <td class="px-4 py-2
border border-gray-300">{{ event.Depth }}</td>
                            <td class="px-4 py-2
border border-gray-300">{{ event.Mag }}</td>

                        </tr>
                        {% endfor %}
                    </tbody>
                </table>
            </div>
        </div>
        <div class="mb-4">
            <!-- Tabel Hasil Cluster -->
            <h2 class="text-xl">Statistik Tiap
Cluster</h2>
            <table class="min-w-full bg-white border
border-gray-300">
                <thead class="bg-gray-200">
                    <tr>
```

```
                        <th class="px-4 py-2 border
border-gray-300 text-left">Cluster</th>
                        <th class="px-4 py-2 border
border-gray-300 text-left">Min Latitude</th>
                        <th class="px-4 py-2 border
border-gray-300 text-left">Max Latitude</th>
                        <th class="px-4 py-2 border
border-gray-300 text-left">Mean Latitude</th>
                        <th class="px-4 py-2 border
border-gray-300 text-left">Min Longitude</th>
                        <th class="px-4 py-2 border
border-gray-300 text-left">Max Longitude</th>
                        <th class="px-4 py-2 border
border-gray-300 text-left">Mean Longitude</th>
                        <th class="px-4 py-2 border
border-gray-300 text-left">Min Magnitude</th>
                        <th class="px-4 py-2 border
border-gray-300 text-left">Max Magnitude</th>
                        <th class="px-4 py-2 border
border-gray-300 text-left">Mean Magnitude</th>
                        <th class="px-4 py-2 border
border-gray-300 text-left">Min Depth</th>
                        <th class="px-4 py-2 border
border-gray-300 text-left">Max Depth</th>
                        <th class="px-4 py-2 border
border-gray-300 text-left">Mean Depth</th>
                    </tr>
                </thead>
                <tbody>
                    {% for stats in cluster_stats %}
                    <tr>
                        <td class="px-4 py-2 border
border-gray-300">{{ stats['Cluster'] }}</td>
                        <td class="px-4 py-2 border
border-gray-300">{{ stats['Min Latitude'] }}</td>
                        <td class="px-4 py-2 border
border-gray-300">{{ stats['Max Latitude'] }}</td>
                        <td class="px-4 py-2 border
border-gray-300">{{ stats['Mean Latitude'] | round(2)
}}</td>
                        <td class="px-4 py-2 border
border-gray-300">{{ stats['Min Longitude'] }}</td>
                        <td class="px-4 py-2 border
border-gray-300">{{ stats['Max Longitude'] }}</td>
                        <td class="px-4 py-2 border
border-gray-300">{{ stats['Mean Longitude'] |
round(2) }}</td>
```

```
                        <td class="px-4 py-2 border
border-gray-300">{{ stats['Min Magnitude'] | round(2)
}}</td>
                        <td class="px-4 py-2 border
border-gray-300">{{ stats['Max Magnitude'] |
round(2)}}</td>
                        <td class="px-4 py-2 border
border-gray-300">{{ stats['Mean Magnitude'] |
round(2) }}</td>
                        <td class="px-4 py-2 border
border-gray-300">{{ stats['Min Depth'] | round(2)
}}</td>
                        <td class="px-4 py-2 border
border-gray-300">{{ stats['Max Depth'] | round(2)
}}</td>
                        <td class="px-4 py-2 border
border-gray-300">{{ stats['Mean Depth'] | round(2)
}}</td>
                    </tr>
                    {% endfor %}
                </tbody>
            </table>
        </div>

        <div class="mb-4">
            <h2 class="mb-4 text-xl">Jumlah
Keanggotaan Tiap Cluster</h2>
            <table class="min-w-full bg-white">
                <thead>
                    <tr>
                        <th class="py-2 px-4 bg-gray-
300">Cluster</th>
                        <th class="py-2 px-4 bg-gray-
300">Jumlah Anggota</th>
                        <th class="py-2 px-4 bg-gray-
300">Aksi</th>

                    </tr>
                </thead>
                <tbody>
                    {% for label, jumlah in
label_counts %}
                    <tr class="border-b">
                        <td class="py-2 px-4 text-
center">Cluster {{ label }}</td>
                        <td class="py-2 px-4 text-
center">{{ jumlah }} anggota</td>
```

```
                        <td class="py-2 px-4 text-
center">
                            <a href="{{
url_for('export_excel', label_id=label) }}"
                                class="px-4 py-2 bg-
green-500 text-white text-sm font-medium rounded
hover:bg-green-600">
                                Export ke Excel
                            </a>
                        </td>
                    </tr>
                    {% endfor %}
                </tbody>
            </table>
        </div>

        {% endblock %}
    </div>
```

Source Code Function Halaman Hasil

```
@app.route('/hasil')
@login_required
def hasil():
    global closest_events
    # Setel seed untuk hasil yang konsisten
    np.random.seed(42)

    # Ambil parameter halaman dari URL, default ke
halaman 1
    page = request.args.get('page', 1, type=int)

    # Batas per halaman
    per_page = 100

    # Ambil data untuk tampilan dengan paginasi
    dataview = Earthquake.query.with_entities(
        Earthquake.id,
        Earthquake.tanggal,
        Earthquake.waktu,
        Earthquake.latitude,
        Earthquake.longitude,
        Earthquake.magnitude,
        Earthquake.depth
```

```
    ).paginate(page=page, per_page=per_page,
error_out=False)

    # Ambil semua data untuk proses clustering dan
tampilan centroid
    earthquakes = Earthquake.query.with_entities(
        Earthquake.id,
        Earthquake.tanggal,
        Earthquake.waktu,
        Earthquake.latitude,
        Earthquake.longitude,
        Earthquake.magnitude,
        Earthquake.depth
    ).all()

    # Konversi data menjadi format array numpy
    data_array = np.array([(e.latitude, e.longitude,
e.magnitude, e.depth) for e in earthquakes])

    # Inisialisasi MinMaxScaler
    scaler = MinMaxScaler()

    # Normalisasi data
    normalized_data =
scaler.fit_transform(data_array)

    # Inisialisasi model Fuzzy C-Means dengan seed
yang sama
    fcm = FCM(n_clusters=optimal_clusters,
max_iter=100, m=2, error=0.01, random_state=42)

    # Fit data ke model
    fcm.fit(normalized_data)

    # Dapatkan keanggotaan untuk setiap titik data
    membership_matrix = fcm.u

    # Tetapkan setiap titik data ke cluster dengan
keanggotaan tertinggi
    labels = np.argmax(membership_matrix, axis=1) + 1

    # Gabungkan data dengan label cluster untuk
tampilan
    labeled_data = list(zip(dataview.items, labels))

    # Simpan label ke tabel ClusteredEarthquake jika
belum disimpan
```

```
    for i, label in enumerate(labels):
        earthquake_id = earthquakes[i].id
        existing_cluster =
ClusteredEarthquake.query.filter_by(earthquake_id=ear
thquake_id).first()
        if existing_cluster is None:
            new_cluster = ClusteredEarthquake(
                earthquake_id=earthquake_id,
                cluster_label=int(label)
            )
            db.session.add(new_cluster)

    # Commit perubahan ke database
    db.session.commit()

    # Hitung jumlah untuk setiap label cluster
    label_counts = db.session.query(
        ClusteredEarthquake.cluster_label,
        db.func.count(ClusteredEarthquake.id).label('
jumlah')
    ).group_by(ClusteredEarthquake.cluster_label).all
()

    # Hitung centroid
    centroids = fcm.centers

    # Hitung jarak dari setiap titik data ke setiap
centroid
    distances = np.linalg.norm(normalized_data[:,
np.newaxis] - centroids, axis=2)

    # Identifikasi kejadian terdekat dengan setiap
centroid dan posisi mereka
    closest_events = []
    for i in range(centroids.shape[0]):
        closest_index = np.argmin(distances[:, i])
        closest_event = earthquakes[closest_index]
        table_position = (page - 1) * per_page +
closest_index + 1

        closest_events.append({
            'Centroid': f'Cluster {i + 1}',
            'No': table_position,
            'Date': closest_event.tanggal,
            'Time': closest_event.waktu,
            'Latitude': closest_event.latitude,
            'Longitude': closest_event.longitude,
```

```
            'Depth': closest_event.depth,
            'Mag': closest_event.magnitude
        })

    # Hitung min, max, dan rata-rata untuk setiap
cluster
    cluster_stats = []
    for cluster_label in range(1, optimal_clusters +
1):
        stats = db.session.query(
            db.func.min(Earthquake.latitude).label('m
in_latitude'),
            db.func.max(Earthquake.latitude).label('m
ax_latitude'),
            db.func.avg(Earthquake.latitude).label('m
ean_latitude'),
            db.func.min(Earthquake.longitude).label('
min_longitude'),
            db.func.max(Earthquake.longitude).label('
max_longitude'),
            db.func.avg(Earthquake.longitude).label('
mean_longitude'),
            db.func.min(Earthquake.magnitude).label('
min_magnitude'),
            db.func.max(Earthquake.magnitude).label('
max_magnitude'),
            db.func.avg(Earthquake.magnitude).label('
mean_magnitude'),
            db.func.min(Earthquake.depth).label('min_
depth'),
            db.func.max(Earthquake.depth).label('max_
depth'),
            db.func.avg(Earthquake.depth).label('mean
_depth')

        ).join(ClusteredEarthquake, Earthquake.id ==
ClusteredEarthquake.earthquake_id)\
         .filter(ClusteredEarthquake.cluster_label ==
cluster_label).first()

        cluster_stats.append({
            'Cluster': f'Cluster {cluster_label}',
            'Min Latitude': stats.min_latitude,
            'Max Latitude': stats.max_latitude,
            'Mean Latitude': stats.mean_latitude,
            'Min Longitude': stats.min_longitude,
            'Max Longitude': stats.max_longitude,
            'Mean Longitude': stats.mean_longitude,
```

```
            'Min Magnitude': stats.min_magnitude,
            'Max Magnitude': stats.max_magnitude,
            'Mean Magnitude': stats.mean_magnitude,
            'Min Depth': stats.min_depth,
            'Max Depth': stats.max_depth,
            'Mean Depth': stats.mean_depth
        })

    # Hitung total halaman
    total_data = Earthquake.query.count()
    total_pages = ceil(total_data / per_page)

    # Render halaman hasil menampilkan cluster
optimal, data yang dilabeli, dan paginasi
    return render_template(
        'hasil.html',
        optimal=optimal_clusters,
        labeled_data=labeled_data,
        total_pages=total_pages,
        current_page=page,
        label_counts=label_counts,
        closest_events=closest_events,
        cluster_stats=cluster_stats  # Kirimkan
statistik cluster untuk tampilan
    )


@app.route('/export_excel/<int:label_id>')
@login_required
def export_excel(label_id):
    # Ambil data dari database untuk label yang
diberikan dengan join ke tabel Earthquake
    data = db.session.query(
        Earthquake.id,
        Earthquake.tanggal,
        Earthquake.waktu,
        Earthquake.latitude,
        Earthquake.longitude,
        Earthquake.magnitude,
        Earthquake.depth,
        ClusteredEarthquake.cluster_label
    ).join(ClusteredEarthquake, Earthquake.id ==
ClusteredEarthquake.earthquake_id) \
     .filter(ClusteredEarthquake.cluster_label ==
label_id).all()

    # Ubah data menjadi DataFrame
```

```
    data_list = [{
        "ID": d.id,
        "Tanggal": d.tanggal,
        "Waktu": d.waktu,
        "Latitude": d.latitude,
        "Longitude": d.longitude,
        "Magnitude": d.magnitude,
        "Depth": d.depth,
        "Cluster Label": d.cluster_label
    } for d in data]

    df = pd.DataFrame(data_list)

    # Simpan DataFrame ke dalam buffer Excel
    output = BytesIO()
    with pd.ExcelWriter(output, engine='xlsxwriter')
as writer:
        df.to_excel(writer, index=False,
sheet_name=f'Cluster_{label_id}')
    output.seek(0)

    # Kirim file Excel sebagai respons untuk diunduh
    return send_file(output, as_attachment=True,
download_name=f'Cluster_{label_id}.xlsx',
mimetype='application/vnd.openxmlformats-
officedocument.spreadsheetml.sheet')
```

Source Code Halaman Visualisasi

```
{% extends 'base.html' %}

{% block content %}

<div class="flex-1 p-10">
  <h1 class="font-bold mb-6" style="font-size: xx-
large;">Visualisasi Data Dengan Maps</h1>
  <div class="bg-gray-100">
     {{ map|safe }}
  </div>
</div>


{% endblock %}
```

Source Code Function Halaman Visualisasi

```
@app.route('/visualisasi')
@login_required
def visualisasi():
    # Membuat peta dengan Folium
    m =
folium.Map(location=[closest_events[0]['Latitude'],
closest_events[0]['Longitude']], zoom_start=5)

    # Tentukan warna untuk setiap cluster
    cluster_colors = {
        1: 'red',
        2: 'blue',
        3: 'green',
        4: 'purple',
        5: 'orange',
        # Tambahkan lebih banyak warna jika
diperlukan
    }

    for event in closest_events:
        cluster_id = int(event['Centroid'].split()[-
1])  # Mengambil nomor cluster dari 'Cluster X'
        color = cluster_colors.get(cluster_id,
'gray')  # Dapatkan warna berdasarkan nomor cluster
        folium.Marker(
            location=[event['Latitude'],
event['Longitude']],
            popup=folium.Popup(f"Cluster:
{event['Centroid']}<br>No: {event['No']}<br>Date:
{event['Date']}<br>Time:
{event['Time']}<br>Magnitude:
{event['Mag']}<br>Depth: {event['Depth']}",
max_width=300),
            icon=folium.Icon(color=color)
        ).add_to(m)

    # Mengembalikan peta ke template sebagai HTML
    return render_template('visualisasi.html',
map=m._repr_html_())
```

Source code Inisialisasi Run Sistem

```
if __name__ == "__main__":

    with app.app_context():
        db.create_all()  # Membuat tabel jika belum
ada
    app.run(debug=True)
```