

Laporan Proyek Akhir
Pemetaan Bawah Permukaan
IF-B



Disusun Oleh:
Khatama Putra 123230053

Dosen Pengampu:
Dr. Frans Richard Kodong, S.T., M.Kom.

Program Studi Informatika
Jurusan Teknik Informatika
Fakultas Teknik Industri
Universitas Pembangunan Nasional “Veteran” Yogyakarta
2025

DAFTAR ISI

DAFTAR ISI.....	ii
BAB I PENDAHULUAN.....	3
1.1 Latar Belakang	3
1.2 Rumusan Masalah.....	3
1.3 Tujuan	3
1.4 Manfaat	4
BAB II LANDASAN TEORI.....	5
2.1 Konsep Dasar Pemetaan Bawah Permukaan	5
2.2 Data dan Koordinat dalam Pemetaan Reservoir	5
2.3 Interpolasi Permukaan	5
2.4 Kontak Fluida.....	5
2.5 Volumetrik Reservoir.....	5
2.6 Visualisasi 2D dan 3D	5
2.7 Pemodelan Berbasis Aplikasi	6
BAB III METODOLOGI.....	7
3.1 Alur Kerja Proyek	7
3.2 Pengumpulan dan Persiapan Data.....	7
3.3 Interpolasi dan Struktur Reservoir	7
3.4 Perhitungan Ketebalan dan Volume	7
3.5 Permodelan 3D.....	7
BAB IV HASIL DAN PEMBAHASAN	8
4.1 Gambaran Aplikasi	8
4.2 Peta Kontur 2D	17
4.3 Model 3D Reservoir.....	17
4.4 Perhitungan Volumetrik.....	17
4.5 Evaluasi Implementasi Aplikasi	18
4.6 Keterbatasan.....	18
4.7 Kontributor Proyek	18
BAB V PENUTUP	19
5.1 Kesimpulan	19
5.2 Saran	19

BAB I

PENDAHULUAN

1.1 Latar Belakang

Pemetaan bawah permukaan merupakan salah satu komponen fundamental dalam ilmu geologi, geofisika, dan teknik permifyakan. Proses ini bertujuan untuk memahami geometri reservoir, distribusi batuan, dan kondisi fluida yang berada di bawah permukaan bumi. Dalam industri migas, pemodelan struktur reservoir menjadi dasar dalam menghitung cadangan hidrokarbon, merancang strategi produksi, hingga mengoptimalkan keputusan ekonomi.

Perkembangan teknologi komputasi dan visualisasi telah memungkinkan penggunaan aplikasi digital untuk melakukan pemodelan reservoir secara interaktif. Salah satu pendekatan yang relevan untuk tugas akademik adalah pengembangan aplikasi sederhana yang mampu menampilkan struktur 3D, peta kontur 2D, serta estimasi volumetrik reservoir secara otomatis. Aplikasi yang dikembangkan dalam proyek ini disusun menggunakan Python dan Streamlit. Fungsionalitas mencakup input data koordinat X–Y–Z, interpolasi struktur, visualisasi kontur, pemodelan 3D, hingga perhitungan volume dan cadangan hidrokarbon. Seluruh proses ini terimplementasi dalam kode sumber yang disediakan.

1.2 Rumusan Masalah

1. Bagaimana melakukan pemetaan struktur bawah permukaan berdasarkan titik koordinat reservoir?
2. Bagaimana menentukan kontak fluida (GOC dan WOC) dalam model reservoir?
3. Bagaimana menghitung volumetrik reservoir (GRV, gas cap, oil zone, total reservoir)?
4. Bagaimana mengimplementasikan pemetaan dan pemodelan tersebut ke dalam sebuah aplikasi interaktif berbasis Python?

1.3 Tujuan

1. Menyusun model struktur reservoir berdasarkan data koordinat X–Y–Z.
2. Menyajikan peta kontur 2D dan model 3D permukaan reservoir.
3. Menghitung parameter volumetrik, termasuk GRV, oil zone volume, gas cap volume, dan total reservoir volume.
4. Mengembangkan aplikasi pemetaan bawah permukaan sebagai sarana analisis dan visualisasi.

1.4 Manfaat

1. Memberikan pemahaman teknis mengenai proses pemetaan bawah permukaan.
2. Menjadi sarana pembelajaran mengenai pemodelan reservoir dan perhitungan volumetrik.
3. Menghasilkan aplikasi sederhana yang dapat digunakan untuk eksplorasi data reservoir.
4. Memberikan dasar pengembangan aplikasi geosains yang lebih kompleks.

BAB II

LANDASAN TEORI

2.1 Konsep Dasar Pemetaan Bawah Permukaan

Pemetaan bawah permukaan bertujuan merekonstruksi geometri lapisan batuan berdasarkan data titik permukaan, sumur, maupun data seismik. Interpretasi struktur membantu memahami perangkap hidrokarbon, distribusi litologi, dan arah migrasi fluida. Peta struktur (structural contour map) adalah representasi kontur dari kedalaman atau elevasi suatu horizon geologi.

2.2 Data dan Koordinat dalam Pemetaan Reservoir

Data biasanya berupa koordinat X (Timur–Barat), Y (Utara–Selatan), dan Z (kedalaman). Kedalaman yang lebih besar berarti posisi lebih dalam dari datum referensi. Kualitas model sangat bergantung pada sebaran titik data.

2.3 Interpolasi Permukaan

Dalam pemetaan reservoir, interpolasi digunakan untuk memperkirakan nilai Z pada titik yang tidak diukur. Aplikasi ini menggunakan metode griddata dengan opsi cubic atau linear (SciPy). Interpolasi grid 100×100 menghasilkan permukaan halus untuk visualisasi.

2.4 Kontak Fluida

- Gas-Oil Contact (GOC) adalah batas antara gas dan minyak.
- Water-Oil Contact (WOC) adalah batas antara air dan minyak.

Kontak fluida penting dalam pembagian zona volumetrik reservoir: gas cap, oil zone, dan aquifer.

2.5 Volumetrik Reservoir

Perhitungan volumetrik dilakukan berdasarkan Gross Rock Volume (GRV), yaitu volume batuan di atas horizon tertentu.

- Gas Cap Volume: volume batuan di atas GOC
- Oil Zone Volume: volume antara GOC dan WOC
- Total Reservoir Colume: volume di atas WOC

Formula STOIP dan GIIP digunakan untuk estimasi cadangan awal.

2.6 Visualisasi 2D dan 3D

Visualisasi sangat penting dalam interpretasi struktur.

- Peta kontur 2D menunjukkan isoline kedalaman.

- Model 3D memperlihatkan geometri topografi bawah permukaan dan bidang GOC/WOC.

Implementasi ini menggunakan Plotly untuk visualisasi grafis.

2.7 Pemodelan Berbasis Aplikasi

Aplikasi berbasis Python memberikan fleksibilitas pengembangan. README proyek (README) menjelaskan fitur, dependensi, serta tujuan dari aplikasinya, sehingga landasan teorinya sekaligus berkaitan dengan pemrograman geosains modern.

BAB III

METODOLOGI

3.1 Alur Kerja Proyek

- Input titik data X–Y–Z
- Pengolahan data menjadi grid
- Interpolasi permukaan
- Penentuan kontak fluida
- Perhitungan volumetrik
- Visualisasi 2D dan 3D
- Ekspor laporan otomatis
- Dokumentasi dan penulisan laporan

3.2 Pengumpulan dan Persiapan Data

Data awal berupa koordinat titik struktur reservoir. Aplikasi menerima input manual, upload CSV/Excel, atau data demo.

3.3 Interpolasi dan Struktur Reservoir

Interpolasi menggunakan fungsi griddata dengan parameter:

- 100 grid pada sumbu X
- 100 grid pada sumbu Y
- Metode cubic/linear jika error

Grid Z kemudian digunakan untuk kontur dan pemodelan.

3.4 Perhitungan Ketebalan dan Volume

Ketebalan dihitung dari selisih antara permukaan reservoir dan kontak fluida.

Volume dihitung:

$$V = \sum(\text{thickness}) \times \Delta x \times \Delta y$$

3.5 Permodelan 3D

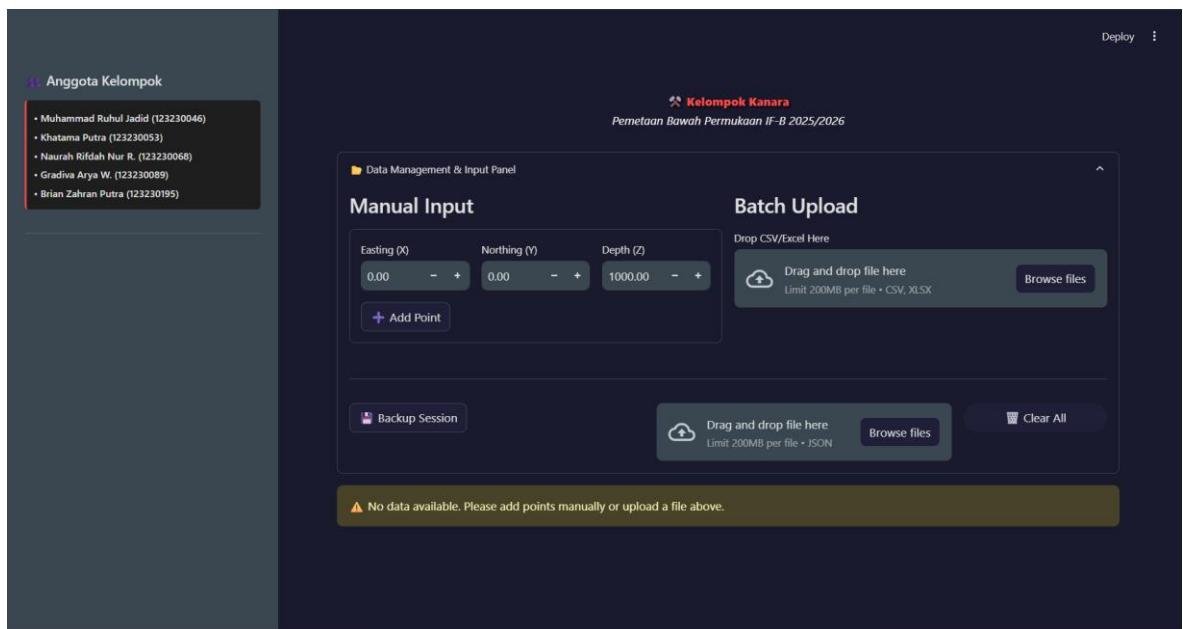
Plotly digunakan untuk permukaan utama dan bidang datar GOC/WOC. Visualisasi ditampilkan secara interaktif dengan fungsi orbiting, zooming, dan layering.

BAB IV

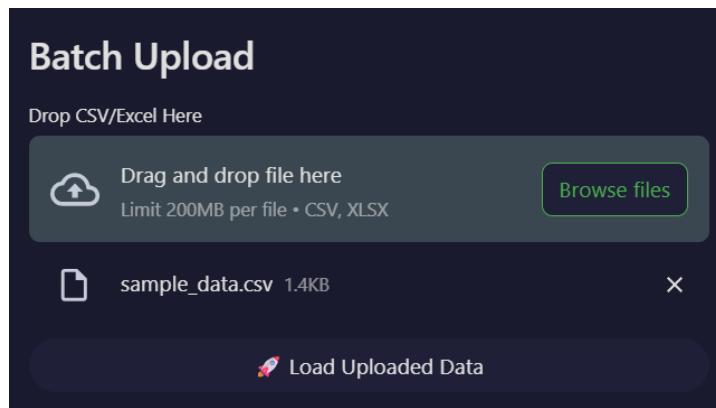
HASIL DAN PEMBAHASAN

4.1 Gambaran Aplikasi

Aplikasi menampilkan panel input, peta kontur, model 3D, tabel data mentah, serta ekspor hasil:



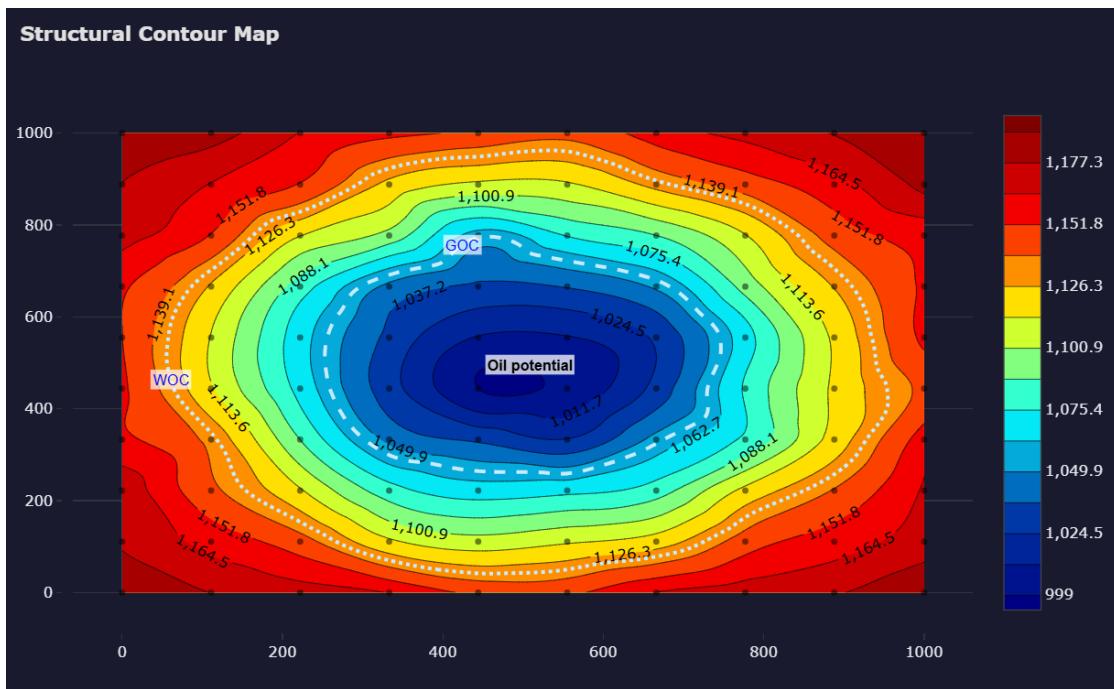
Gambar 1. Halaman Utama



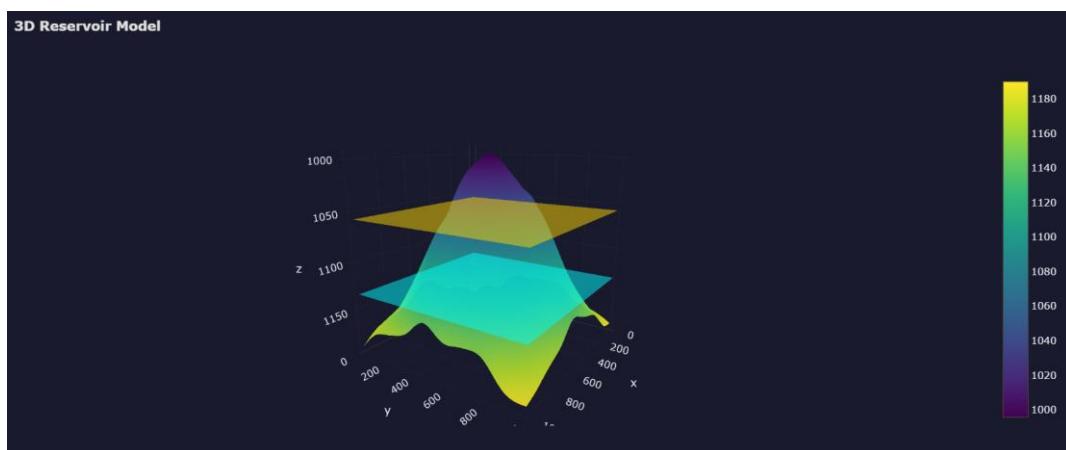
Gambar 2. Fitur Unggah Data



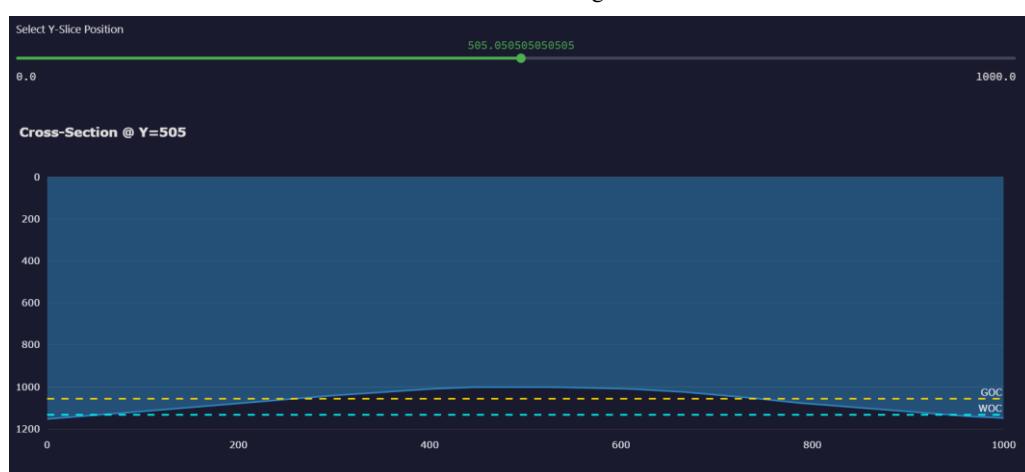
Gambar 3. Volumetric Assessment



Gambar 4. Peta Kontur Dua Dimensi



Gambar 5. Peta Kontur Tiga Dimensi



Gambar 6. Peta Kontur Tiga Dimensi

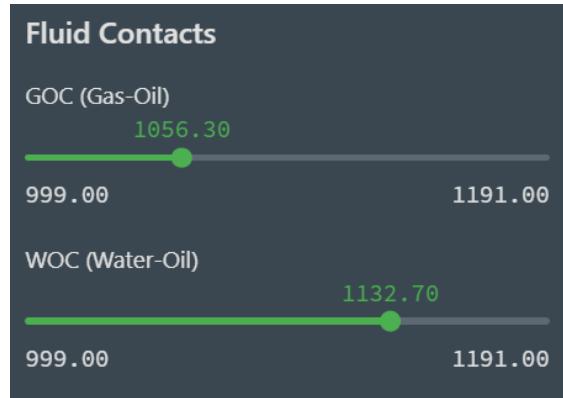
Export Center

[Download PDF Report](#) [Download Excel Report](#) [Download Grid CSV](#)

Raw Data Preview

	X	Y	Z
0		0	1190
1		111	1177
2		222	1170
3		333	1164
4		444	1149
5		555	1149
6		666	1165
7		777	1171

Gambar 7. Data Mentah dan Ekspor Data



Gambar 8. Fluid Contacts



Gambar 9. Petrophysics

Source Code:

```
import streamlit as st
import pandas as pd
```

```

import plotly.graph_objects as go
import numpy as np
from scipy.interpolate import griddata
from datetime import datetime
import io
import json

from reportlab.lib.pagesizes import A4
from reportlab.lib import colors
from reportlab.lib.units import inch
from reportlab.platypus import SimpleDocTemplate, Paragraph, Spacer, Table, TableStyle
from reportlab.lib.styles import getSampleStyleSheet, ParagraphStyle
from reportlab.lib.enums import TA_CENTER

st.set_page_config(page_title="Kelompok Kanara", layout="wide",
page_icon="🌟")

st.markdown("""
<style>
html, body, [class*="css"] {
    font-family: 'Segoe UI', Tahoma, Geneva, Verdana, sans-serif;
}
.main-title {
    font-size: 3rem;
    font-weight: 800;
    color: #FF4B4B;
    text-align: center;
    margin-bottom: 0px;
    text-shadow: 2px 2px 4px #000000;
}
.sub-title {
    font-size: 1.2rem;
    color: #FAFAFA;
    text-align: center;
    margin-bottom: 30px;
    font-style: italic;
}
div[data-testid="stMetric"] {
    background-color: #262730;
    border: 1px solid #4e4f57;
    padding: 15px;
    border-radius: 10px;
    box-shadow: 2px 2px 5px rgba(0,0,0,0.3);
}
.stButton>button {
    width: 100%;
    border-radius: 20px;
    font-weight: bold;
    border: none;
    transition: 0.3s;
}
.stButton>button:hover {
    transform: scale(1.02);
}
</style>
""", unsafe_allow_html=True)

def create_volumetric_report_pdf(vol_gas_cap, vol_oil_zone, vol_total_res,
                                 goc_input, woc_input,
                                 num_points, x_range, y_range, z_range):
    buffer = io.BytesIO()
    doc = SimpleDocTemplate(buffer, pagesize=A4)
    story = []
    styles = getSampleStyleSheet()


```

```

        title_style = ParagraphStyle('CustomTitle', parent=styles['Heading1'],
fontStyle=20, textColor=colors.HexColor('#FF4B4B'), spaceAfter=30,
alignment=TA_CENTER)

    story.append(Paragraph("ReserView Calculation Report", title_style))
    story.append(Spacer(1, 0.2*inch))
    story.append(Paragraph(f"<i>Generated: {datetime.now().strftime('%d %B %Y, %H:%M:%S')}</i>", styles['Normal']))
    story.append(Spacer(1, 0.3*inch))
    story.append(Paragraph("Input Parameters Summary", styles['Heading2']))
    story.append(Spacer(1, 0.1*inch))

    summary_data = [
        ['Parameter', 'Value'],
        ['Total Points', f"{num_points}"],
        ['GOC', f"{goc_input:.2f} m"],
        ['WOC', f"{woc_input:.2f} m"],
        ['X Range', f"{x_range[0]:.2f} - {x_range[1]:.2f}"],
        ['Y Range', f"{y_range[0]:.2f} - {y_range[1]:.2f}"],
        ['Z Range', f"{z_range[0]:.2f} - {z_range[1]:.2f} m"],
    ]

    summary_table = Table(summary_data, colWidths=[3*inch, 3*inch])
    summary_table.setStyle(TableStyle([
        ('BACKGROUND', (0, 0), (-1, 0), colors.darkgrey),
        ('TEXTCOLOR', (0, 0), (-1, 0), colors.white),
        ('ALIGN', (0, 0), (-1, -1), 'LEFT'),
        ('GRID', (0, 0), (-1, -1), 1, colors.black)
    ]))
    story.append(summary_table)
    story.append(Spacer(1, 0.3*inch))

    story.append(Paragraph("Volume Calculation Results",
styles['Heading2']))
    story.append(Spacer(1, 0.1*inch))

    volume_data = [
        ['Zone', 'Volume (m³)', 'Volume (MMm³)'],
        ['Gas Cap', f"{vol_gas_cap:.2f}", f"{vol_gas_cap/1e6:.2f}"],
        ['Oil Zone', f"{vol_oil_zone:.2f}", f"{vol_oil_zone/1e6:.2f}"],
        ['Total Reservoir', f"{vol_total_res:.2f}",
f"{vol_total_res/1e6:.2f}"],
    ]

    volume_table = Table(volume_data, colWidths=[2*inch, 2*inch, 2*inch])
    volume_table.setStyle(TableStyle([
        ('BACKGROUND', (0, 0), (-1, 0), colors.firebrick),
        ('TEXTCOLOR', (0, 0), (-1, 0), colors.white),
        ('ALIGN', (0, 0), (-1, -1), 'CENTER'),
        ('GRID', (0, 0), (-1, -1), 1, colors.black),
    ]))
    story.append(volume_table)

    doc.build(story)
    buffer.seek(0)
    return buffer

def create_volumetric_report_excel(vol_gas_cap, vol_oil_zone, vol_total_res,
goc_input, woc_input, num_points, x_range, y_range, z_range, df):
    buffer = io.BytesIO()
    with pd.ExcelWriter(buffer, engine='openpyxl') as writer:
        summary_df = pd.DataFrame({
            'Parameter': ['Total Data Points', 'GOC (m)', 'WOC (m)', 'X Min', 'X Max', 'Y Min', 'Y Max', 'Z Min (m)', 'Z Max (m)'],
            'Nilai': [num_points, goc_input, woc_input, x_range[0], x_range[1], y_range[0], y_range[1], z_range[0], z_range[1]]
        })
        summary_df.to_excel(writer, sheet_name='Summary', index=False)

```

```

volume_df = pd.DataFrame({
    'Zona': ['Gas Cap', 'Oil Zone', 'Total Reservoir'],
    'Volume (m³)': [vol_gas_cap, vol_oil_zone, vol_total_res],
    'Volume (Juta m³)': [vol_gas_cap/1e6, vol_oil_zone/1e6,
vol_total_res/1e6]
})
volume_df.to_excel(writer, sheet_name='Volume Results', index=False)
df.to_excel(writer, sheet_name='Raw Data', index=False)
buffer.seek(0)
return buffer

st.markdown('<p class="main-title">🛠 Kelompok Kanara</p>',
unsafe_allow_html=True)
st.markdown('<p class="sub-title">Pemetaan Bawah Permukaan IF-B  
2025/2026</p>', unsafe_allow_html=True)

if 'data_points' not in st.session_state:
    st.session_state['data_points'] = []

with st.expander("📁 Data Management & Input Panel", expanded=False):
    col_input1, col_input2 = st.columns([1, 1])

    with col_input1:
        st.subheader("Manual Input")
        with st.form(key='input_form', clear_on_submit=True):
            cil, ci2, ci3 = st.columns(3)
            with cil: x_val = st.number_input("Easting (X)", value=0.0,
step=50.0)
            with ci2: y_val = st.number_input("Northing (Y)", value=0.0,
step=50.0)
            with ci3: z_val = st.number_input("Depth (Z)", value=1000.0,
step=10.0)
            if st.form_submit_button('➕ Add Point'):
                st.session_state['data_points'].append({'X': x_val, 'Y':
y_val, 'Z': z_val})
                st.success("Point Added!")

    with col_input2:
        st.subheader("Batch Upload")
        uploaded_file = st.file_uploader("Drop CSV/Excel Here", type=["csv",
"xlsx"])
        if uploaded_file:
            try:
                df_upload = pd.read_csv(uploaded_file) if
uploaded_file.name.endswith('.csv') else pd.read_excel(uploaded_file)
                df_upload.columns = [c.upper() for c in df_upload.columns]
                if {'X', 'Y', 'Z'}.issubset(df_upload.columns):
                    if st.button('⚡ Load Uploaded Data'):
                        st.session_state['data_points'].extend(df_upload[['X',
'Y', 'Z']].to_dict('records'))
                        st.rerun()
                    else:
                        st.error("Invalid Columns! Need X, Y, Z.")
            except Exception as e:
                st.error(f"Error: {e}")

    st.divider()
    c_sess1, c_sess2, c_sess3 = st.columns([2, 2, 1])
    with c_sess1:
        st.download_button("💾 Backup Session",
data=json.dumps(st.session_state['data_points']), file_name="backup.json",
mime="application/json")
    with c_sess2:
        up_sess = st.file_uploader("Restore Session", type=["json"],
label_visibility="collapsed")
        if up_sess and st.button("Restore"):
            st.session_state['data_points'] = json.load(up_sess)

```

```

        st.rerun()
    with c_sess3:
        if st.button("Clear All"):
            st.session_state['data_points'] = []
            st.rerun()

    df = pd.DataFrame(st.session_state['data_points'])

    with st.sidebar:
        st.markdown("### 🧑‍🤝‍🧑 Anggota Kelompok")
        st.markdown("""
<div style="background-color: #1E1E1E; padding: 10px; border-radius: 5px; border-left: 3px solid #FF4B4B;">
    <small>
        • <b>Muhammad Ruhul Jadiid (123230046)</b><br>
        • <b>Khatama Putra (123230053)</b><br>
        • <b>Naurah Rifdah Nur R. (123230068)</b><br>
        • <b>Gradiva Arya W. (123230089)</b><br>
        • <b>Brian Zahran Putra (123230195)</b><br>
    </small>
</div>
""", unsafe_allow_html=True)
        st.divider()

    if not df.empty:
        st.header("🌐 Control Panel")

        st.subheader("Fluid Contacts")
        min_z, max_z = df['Z'].min(), df['Z'].max()
        goc_input = st.slider("GOC (Gas-Oil)", float(min_z), float(max_z),
        float(min_z + (max_z - min_z) * 0.3))
        woc_input = st.slider("WOC (Water-Oil)", float(min_z), float(max_z),
        float(min_z + (max_z - min_z) * 0.7))

        st.subheader("Petrophysics")
        porosity = st.number_input("Porosity ( $\phi$ )", 0.05, 0.40, 0.20, 0.01)
        sw = st.number_input("Water Saturation (Sw)", 0.1, 1.0, 0.3, 0.05)
        ntg = st.number_input("Net-to-Gross", 0.1, 1.0, 0.8, 0.05)
        bo = st.number_input("Bo (Oil FVF)", 1.0, 2.0, 1.2)
        bg = st.number_input("Bg (Gas FVF)", 0.001, 0.1, 0.005,
        format=".4f")

        st.info(f"Dataset: {len(df)} Points\nMax Depth: {max_z} m")

    if df.empty:
        st.warning("⚠️ No data available. Please add points manually or upload a file above.")
    else:
        if len(df) >= 4:
            df_unique = df.groupby(['X', 'Y'], as_index=False)['Z'].mean()
            grid_x, grid_y = np.meshgrid(
                np.linspace(df['X'].min(), df['X'].max(), 100),
                np.linspace(df['Y'].min(), df['Y'].max(), 100))
            )
            try:
                grid_z = griddata((df_unique['X'], df_unique['Y']),
                df_unique['Z'], (grid_x, grid_y), method='cubic')
            except:
                grid_z = griddata((df_unique['X'], df_unique['Y']),
                df_unique['Z'], (grid_x, grid_y), method='linear')

            dx = (df['X'].max() - df['X'].min()) / 99
            dy = (df['Y'].max() - df['Y'].min()) / 99
            cell_area = dx * dy

            thick_above_woc = np.maximum(0, woc_input - grid_z)
            vol_total_res = np.nansum(thick_above_woc) * cell_area

```

```

thick_above_goc = np.maximum(0, goc_input - grid_z)
vol_gas_cap = np.nansum(thick_above_goc) * cell_area

vol_oil_zone = max(0, vol_total_res - vol_gas_cap)

stoiip = (vol_oil_zone * ntg * porosity * (1 - sw)) / bo
giip = (vol_gas_cap * ntg * porosity * (1 - sw)) / bg

st.markdown("### 📈 Volumetric Assessment")
m1, m2, m3, m4, m5 = st.columns(5)
m1.metric("Gas Cap Vol", f"{vol_gas_cap/1e6:.2f} MMm³",
delta="Gross")
m2.metric("Oil Zone Vol", f"{vol_oil_zone/1e6:.2f} MMm³",
delta="Gross")
m3.metric("Total Reservoir", f"{vol_total_res/1e6:.2f} MMm³",
delta="Gross")
m4.metric("🔥 Est. GIIP", f"{giip/1e9:.2f} BCF", delta_color="off")
m5.metric("🌐 Est. STOIIP", f"{stoiip/1e6:.2f} MMbbl",
delta_color="off")

st.divider()

v_tab1, v_tab2, v_tab3, v_tab4 = st.tabs([".mapbox Contour Map", "📦 3D Model", "gMaps Cross-Section", "📝 Reports & Data"])

with v_tab1:
    fig_2d = go.Figure(data=go.Contour(
        z=grid_z, x=grid_x[0], y=grid_y[:,0],
        colorscale='Jet', contours=dict(start=min_z, end=max_z,
size=(max_z-min_z)/15, showlabels=True)
    ))

    try:
        fig_2d.add_trace(go.Contour(
            z=grid_z, x=grid_x[0], y=grid_y[:,0],
            contours=dict(start=goc_input, end=goc_input,
size=(max_z-min_z)/1000, showlabels=False, coloring='lines'),
            line=dict(color='blue', width=3, dash='dash'),
            showscale=False, name='GOC (line)')
        )
    except Exception:
        pass

    try:
        fig_2d.add_trace(go.Contour(
            z=grid_z, x=grid_x[0], y=grid_y[:,0],
            contours=dict(start=woc_input, end=woc_input,
size=(max_z-min_z)/1000, showlabels=False, coloring='lines'),
            line=dict(color='blue', width=3, dash='dot'),
            showscale=False, name='WOC (line)')
        )
    except Exception:
        pass

    try:
        gz_flat = grid_z.flatten()
        gx_flat = grid_x.flatten()
        gy_flat = grid_y.flatten()

        if np.isfinite(gz_flat).any() and np.isfinite(goc_input):
            idx_goc = int(np.nanargmin(np.abs(gz_flat - goc_input)))
            fig_2d.add_annotation(x=float(gx_flat[idx_goc]),
y=float(gy_flat[idx_goc]), text='GOC', showarrow=False,
font=dict(color='blue', size=12,
family='Arial'), bgcolor='rgba(255,255,255,0.7)')

        if np.isfinite(gz_flat).any() and np.isfinite(woc_input):
            idx_woc = int(np.nanargmin(np.abs(gz_flat - woc_input)))
    
```

```

        fig_2d.add_annotation(x=float(gx_flat[idx_woc]),
y=float(gy_flat[idx_woc]), text='WOC', showarrow=False,
font=dict(color='blue', size=12,
family='Arial'), bgcolor='rgba(255,255,255,0.7)')

        low_c, high_c = min(goc_input, woc_input), max(goc_input,
woc_input)
        mask = np.isfinite(gz_flat) & (gz_flat >= low_c) & (gz_flat
<= high_c)
        if np.any(mask):
            cx, cy = float(np.nanmean(gx_flat[mask])),
float(np.nanmean(gy_flat[mask]))
            fig_2d.add_annotation(x=cx, y=cy, text='Oil potential',
showarrow=False,
font=dict(color='black', size=12,
family='Arial', weight='bold'),
bgcolor='rgba(255,255,255,0.75)')
        except Exception:
            pass

        fig_2d.add_trace(go.Scatter(x=df['X'], y=df['Y'],
mode='markers', marker=dict(color='black', size=5, opacity=0.5),
name='Wells'))
        fig_2d.update_layout(title="Structural Contour Map", height=600,
plot_bgcolor='rgba(0,0,0,0)')
        st.plotly_chart(fig_2d, use_container_width=True)

    with v_tab2:
        fig_3d = go.Figure(data=[go.Surface(z=grid_z, x=grid_x,
y=grid_y, colorscale='Viridis', opacity=0.9)])
        fig_3d.add_trace(go.Surface(z=goc_input*np.ones_like(grid_z),
x=grid_x, y=grid_y, colorscale=[[0,'gold'],[1,'gold']], opacity=0.5,
showscale=False, name='GOC'))
        fig_3d.add_trace(go.Surface(z=woc_input*np.ones_like(grid_z),
x=grid_x, y=grid_y, colorscale=[[0,'cyan'],[1,'cyan']], opacity=0.5,
showscale=False, name='WOC'))

        fig_3d.update_layout(title="3D Reservoir Model", height=600,
scene=dict(zaxis=dict(autorange="reversed")))
        st.plotly_chart(fig_3d, use_container_width=True)

    with v_tab3:
        unique_ys = np.unique(grid_y.flatten())
        default_y = unique_ys[len(unique_ys)//2]

        slice_y = st.select_slider("Select Y-Slice Position",
options=unique_ys,
value=default_y)
        idx_y = (np.abs(grid_y[:, 0] - slice_y)).argmin()

        fig_xs = go.Figure()
        fig_xs.add_trace(go.Scatter(x=grid_x[0, :], y=grid_z[idx_y, :],
mode='lines', fill='tozeroxy', line=dict(color='#2E86C1'), name='Structure'))
        fig_xs.add_hline(y=goc_input, line_dash="dash",
line_color="gold", annotation_text="GOC")
        fig_xs.add_hline(y=woc_input, line_dash="dash",
line_color="cyan", annotation_text="WOC")
        fig_xs.update_layout(title=f"Cross-Section @ Y={slice_y:.0f}",
height=500, yaxis=dict(autorange="reversed"))
        st.plotly_chart(fig_xs, use_container_width=True)

    with v_tab4:
        st.markdown("#### Export Center")
        cel, ce2, ce3 = st.columns(3)
        with cel:
            pdf_data = create_volumetric_report_pdf(vol_gas_cap,
vol_oil_zone, vol_total_res, goc_input, woc_input, len(df), (df['X'].min(),

```

```

df['X'].max(), (df['Y'].min(), df['Y'].max()), (df['Z'].min(),
df['Z'].max()))
        st.download_button("📄 Download PDF Report", pdf_data,
"report.pdf", "application/pdf")
    with ce2:
        xlsx_data = create_volumetric_report_excel(vol_gas_cap,
vol_oil_zone, vol_total_res, goc_input, woc_input, len(df), (df['X'].min(),
df['X'].max()), (df['Y'].min(), df['Y'].max()), (df['Z'].min(),
df['Z'].max()), df)
        st.download_button("📊 Download Excel Report", xlsx_data,
"report.xlsx", "application/vnd.openxmlformats-
officedocument.spreadsheetml.sheet")
    with ce3:
        st.download_button("💾 Download Grid CSV",
pd.DataFrame({'X': grid_x.flatten(), 'Y': grid_y.flatten(), 'Z':
grid_z.flatten()}).to_csv(index=False), "grid.csv", "text/csv")

        st.divider()
        st.caption("Raw Data Preview")
        st.dataframe(df.style.background_gradient(cmap='Blues'),
use_container_width=True)

    else:
        st.error("Insufficient data points for interpolation (Min. 4
required).")

```

4.2 Peta Kontur 2D

Visualisasi kontur menampilkan:

- Permukaan reservoir.
- Titik data beserta nilai kedalaman.
- Zona fluida (gas cap, oil zone, aquifer).
- Skala warna dan label kontur.

4.3 Model 3D Reservoir

Model menunjukkan:

- Permukaan reservoir hasil interpolasi
- Bidang GOC (merah)
- Bidang WOC (biru)
- Batang vertikal sumur-titik data

Model ini memberikan konteks ruang tiga dimensi untuk interpretasi.

4.4 Perhitungan Volumetrik

Aplikasi menghitung:

- Gas Cap Volume
- Oil Zone Volume
- Total Reservoir Volume
- Parameter petrofisika
- STOIIP & GIIP

Walaupun laporan ini tidak memuat dataset aktual, struktur laporan memungkinkan pengguna memasukkan data mereka sendiri dan mengisi tabel hasil.

4.5 Evaluasi Implementasi Aplikasi

Kelebihan:

- Interaktif dan mudah digunakan
- Visualiasi 2D & 3D lengkap
- Perhitungan volumetrik otomatis
- Ekspor laporan terintegrasi

Keterbatasan:

- Interpolasi bergantung pada sebaran titik
- Tidak mendukung multiple horizons
- Tidak menangani fault atau struktur kompleks

4.6 Keterbatasan

Aplikasi masih dalam tahap dasar. Untuk skenario industri, diperlukan:

- Integrasi data seismik
- Model reservoir multi-layer
- Dukungan grid geologi 3D (corner-point grid)
- Analisis ketidakpastian

4.7 Kontributor Proyek

No.	Nama	NIM
1.	Muhammad Ruhul Jadid	123230046
2.	Khatama Putra	123230053
3.	Naurah Rifdah Nur Ramadhani	123230068
4.	Gradiva Arya Wicaksana	123230089
5.	Brian Zahran Putra	123230195

BAB V

PENUTUP

5.1 Kesimpulan

1. Pemetaan bawah permukaan dapat dilakukan menggunakan data koordinat X–Y–Z melalui proses interpolasi dan visualisasi berbasis Python.
2. Kontak fluida GOC dan WOC menjadi batas penting dalam mendefinisikan zona reservoir.
3. Volume reservoir dapat dihitung secara numerik menggunakan grid dan ketebalan di atas horizon tertentu.
4. Aplikasi berbasis Streamlit berhasil mengintegrasikan pengolahan data, visualisasi 2D–3D, serta perhitungan volumetrik secara interaktif.

5.2 Saran

1. Pengembangan fitur interpretasi fault, horizon multipel, dan model stratigrafi.
2. Integrasi dengan data pengeboran dan seismik.
3. Peningkatan algoritma interpolasi menggunakan geostatistik (Kriging).
4. Penambahan fitur analisis sensitivitas dan ketidakpastian.