4. IMPLEMENTASI SISTEM

Pada bab ini akan membahas mengenai program yang sudah dibuat. Berikut merupakan tahapan dalam pembuatan program visualisasi data berbasis web yang ditulis dengan bahasa pemrograman python dan memanfaatkan *database* sqlite. *Framework front-end* dan *data visualization* pendukung yaitu streamlit*,* dan juga *beberapa library* python mendukung yang berguna dalam pembuatan visualisasi data berbasis web ini.

4.1 Implementasi Sistem

Tahapan penerapan implementasi sistem dari rancangan yang sudah dibuat, dan pembuatan dilakukan secara bertahap.

Tabel 4.1

Tabel Hubungan *Activity Diagram* dengan Segmen Program

Activity Diagram	**Segmen Program**	**Nama Segmen Program**	**Keterangan**
Gambar 3.17	4.1	User view homepage	Halaman pertama yang ditampilkan kepada *user* saat menginputkan URL *website*.
Gambar 3.18	4.2	Dashboard mahasiswa	Halaman *dashboard* yang dapat dilihat oleh *user* mahasiswa sendiri.
Gambar 3.19	4.3	KHS	Halaman untuk melihat KHS mahasiswa, informasi akademik mahasiswa setiap periode.
Gambar 3.20	4.4	Transkrip mahasiswa	Halaman untuk melihat keseluruhan transkrip nilai

			mahasiswa.
Gambar 3.21	4.5	Dashboard dosen wali	Halaman *dashboard* yang dapat dilihat oleh *user* dosen.
Gambar 3.22	4.6	List Mata kuliah dosen	Halaman bagi dosen untuk melihat mata kuliah yang diajarkan setiap semester.
Gambar 3.23	4.7	List mahasiswa wali	Halaman untuk melihat informasi mengenai mahasiswa wali dari dosen.
Gambar 3.24	4.8	Dashboard program studi	Halaman *dashboard* yang dapat dilihat oleh *user* program studi.
Gambar 3.25	4.9	List mahasiswa program studi	Halaman untuk melihat informasi mengenai mahasiswa dalam program studi.
Gambar 3.26	4.10	List dosen program studi	Halaman untuk melihat informasi mengenai dosen pengajar dalam program studi.
Gambar 3.27	4.11	Dashboard yudisium	Halaman untuk melihat visualisasi serta detail mahasiswa yang berhasil yudisium.
Gambar 3.28	4.12	List mata kuliah dan kelas	Halaman untuk melihat informasi mengenai mata kuliah yang dibuka dalam

			program studi.
Gambar 3.29	4.13	Simulasi Yudisium	halaman sistem yang dirancang untuk program studi dalam melakukan simulasi terhadap persyaratan yudisium mahasiswa.

4.1.1 User View Homepage

Halaman ini digunakan sebagai *landing page* atau halaman pertama yang akan dilihat oleh *user* saat menggunakan prototipe visualisasi data, dimana pada halaman ini user diharuskan melakukan *login* dimana pada prototipe visualisasi data ini menggunakan *username* dan *password* akun *dummy* yang dibuat khusus untuk prorotipe ini. Dengan menggunakan akun *dummy* yang sudah dibuat user dapat masuk sesuai dengan *role* atau *user ty*pe yang ada yaitu mahasiswa, dosen, atau sebagai program studi.

Segmen Program 4.1 User View Homepage

```
#Set all Pages
show_pages(
    [
        #Homepage
        Page('homepage.py', 'Homepage'),

        #Page Prodi
        Page("prodi/dashboard_prodi.py", "Dashboard Program
Studi"),
        Page("prodi/dashboard_yudisium.py", "Dashboard
Yudisium"),
        Page("prodi/prodi_dosen.py", "Dosen Prodi"),
        Page("prodi/prodi_mahasiswa.py", "Mahasiswa Prodi"),
        Page("prodi/matkul_kelas.py", 'Mata Kuliah dan
Kelas'),
        Page("prodi/yudisium.py", 'Yudisium'),

        Page("prodi/dosen/kelas_ajar.py", "Prodi_Kelas Ajar"),
        Page("prodi/dosen/matkul_ajar.py", "Prodi_Mata Kuliah
Ajar"),
        Page("prodi/dosen/dashboard_dosen_wali.py",
"Prodi_Dashboard Dosen"),
        Page("prodi/dosen/mahasiswa_wali.py", "Prodi_Mahasiswa
Wali"),
```

```
        Page("prodi/mahasiswa/dashboard_mahasiswa.py",
"Prodi_Dashboard Mahasiswa"),
        Page("prodi/mahasiswa/khs_transkrip.py", "Prodi_KHS-
Transkrip"),

        #Page Dosen
        Page("dosen/dashboard_dosen.py", "Dashboard Dosen"),
        Page("dosen/mahasiswa_wali.py", "Mahasiswa Wali"),
        Page("dosen/matkul_ajar.py", "Mata Kuliah Ajar"),

        Page("dosen/mahasiswa/dashboard_mahasiswa.py",
"Dosen_Dashboard Mahasiswa"),
        Page("dosen/mahasiswa/khs_transkrip.py", "Dosen_KHS-
Transkrip"),

        #Page Mahasiswa
        Page("mahasiswa/dashboard_mahasiswa.py", "Dashboard
Mahasiswa"),
        Page("mahasiswa/khs_transkrip.py", "KHS-Transkrip")
    ]
)
def authenticate_user(username, password):
    conn = sqlite3.connect('db_skripsi.db')  # Connect to your
SQLite database
    cursor = conn.cursor()

    # Hash the password input to match with stored hashed
password
    hashed_password =
hashlib.sha256(password.encode()).hexdigest()

    # Query to check username and hashed password
    query = "SELECT usertype, userid, prodi FROM user WHERE
username=? AND password=?"
    cursor.execute(query, (username, hashed_password))
    user_data = cursor.fetchone()

    conn.close()
    return user_data

#Set Session
# st.session_state.clear()
if 'logged_in' not in st.session_state:
    st.session_state.logged_in = False
if 'usertype' not in st.session_state:
    st.session_state.usertype = None
if 'userid' not in st.session_state:
    st.session_state.userid = None
if 'prodi' not in st.session_state:
    st.session_state.prodi = None
if 'username' not in st.session_state:
    st.session_state.username = None
```

```
if not st.session_state.logged_in:
    page_prodi = ['Dashboard Program Studi', 'Dashboard
Yudisium', 'Dosen Prodi', 'Mahasiswa Prodi', 'Mata Kuliah dan
Kelas', 'Prodi_Kelas Ajar', 'Prodi_Mata Kuliah Ajar',
'Prodi_Dashboard Dosen', 'Prodi_Mahasiswa Wali', 'Yudisium',
'Prodi_Dashboard Mahasiswa', 'Prodi_KHS-Transkrip']
    page_dosen = ['Dashboard Dosen', 'Mahasiswa Wali', 'Mata
Kuliah Ajar', 'Dosen_Dashboard Mahasiswa', 'Dosen_KHS-
Transkrip']
    page_mahasiswa = ['Dashboard Mahasiswa', 'KHS-Transkrip']
    hide_pages(page_prodi+page_dosen+page_mahasiswa)

    st.title("Demo Visualisasi Data Akademik Program Studi
UKP")
    st.write("Please log in to continue.")

    username_input = st.text_input("Username",
key='username_input')
    password = st.text_input("Password", type="password",
key='password')

    if st.button("Log in", key="login_button"):
        user_data = authenticate_user(username_input,
password)
        if user_data:
            st.session_state.logged_in = True
            st.session_state.usertype = user_data[0]
            st.session_state.userid = user_data[1]
            st.session_state.prodi = user_data[2]
            st.session_state.username = username_input

            st.success("Logged in successfully!")
            sleep(0.5)
            st.rerun()

        else:
            st.error("Incorrect username or password")
else:
    # Display user information after successful login
    st.title("Welcome!")
    st.write('---')
    st.write("You are logged in as:")
    st.write(f"**Username:** {st.session_state.username}")
    st.write(f"**Usertype:** {st.session_state.usertype}")
    st.write(f"**User ID:** {st.session_state.userid}")
    st.write(f"**Prodi:** {st.session_state.prodi}")

    if st.session_state.usertype == 'mahasiswa':
        page_prodi = ['Dashboard Program Studi', 'Dashboard
Yudisium', 'Dosen Prodi', 'Mahasiswa Prodi', 'Mata Kuliah dan
Kelas', 'Prodi_Kelas Ajar', 'Prodi_Mata Kuliah Ajar',
```

```
'Prodi_Dashboard Dosen', 'Prodi_Mahasiswa Wali', 'Yudisium',
'Prodi_Dashboard Mahasiswa', 'Prodi_KHS-Transkrip']
        page_dosen = ['Dashboard Dosen', 'Mahasiswa Wali',
'Mata Kuliah Ajar', 'Dosen_Dashboard Mahasiswa', 'Dosen_KHS-
Transkrip']
        hide_pages(page_prodi+page_dosen)
    elif st.session_state.usertype == 'dosen':
        page_prodi = ['Dashboard Program Studi', 'Dashboard
Yudisium', 'Dosen Prodi', 'Mahasiswa Prodi', 'Mata Kuliah dan
Kelas', 'Prodi_Kelas Ajar', 'Prodi_Mata Kuliah Ajar',
'Prodi_Dashboard Dosen', 'Prodi_Mahasiswa Wali', 'Yudisium',
'Prodi_Dashboard Mahasiswa', 'Prodi_KHS-Transkrip']
        page_dosen = ['Dosen_Dashboard Mahasiswa', 'Dosen_KHS-
Transkrip']
        page_mahasiswa = ['Dashboard Mahasiswa', 'KHS-
Transkrip']
        hide_pages(page_prodi+page_dosen+page_mahasiswa)
    elif st.session_state.usertype == 'prodi':
        page_prodi = ['Prodi_Kelas Ajar', 'Prodi_Mata Kuliah
Ajar', 'Prodi_Dashboard Dosen', 'Prodi_Mahasiswa Wali',
'Prodi_Dashboard Mahasiswa', 'Prodi_KHS-Transkrip']
        page_dosen = ['Dashboard Dosen', 'Mahasiswa Wali',
'Mata Kuliah Ajar', 'Dosen_Dashboard Mahasiswa', 'Dosen_KHS-
Transkrip']
        page_mahasiswa = ['Dashboard Mahasiswa', 'KHS-
Transkrip']
        hide_pages(page_prodi+page_dosen+page_mahasiswa)

    # Provide a logout button
    st.write('')
    if st.button("Log out", type='primary',
use_container_width=True):
        st.session_state.logged_in = False
        st.session_state.usertype = None
        st.session_state.userid = None
        st.session_state.selected_prodi = None
        st.session_state.username = None
        st.rerun()
```

4.1.2 Dashboard Mahasiswa

Halaman dashboard mahasiswa yang dapat diakses oleh user untuk memantau status akademik mahasiswa. Halaman ini menyajikan beberapa grafik yang membentuk dashboard tersebut, dengan data akademik yang diambil dari periode akademik terakhir. Visualisasi data pada dashboard ini mencakup berbagai informasi penting, antara lain durasi masa studi mahasiswa, jumlah SKS yang telah diselesaikan, status pemenuhan syarat yudisium, histori IPK dan IPS dari periode awal hingga periode terakhir, serta status prasyarat mata kuliah wajib yang sudah atau belum diselesaikan oleh mahasiswa.

Segmen Program 4.2 Dashboard Mahasiswa

```
#Hide Pages
page_prodi = ['Dashboard Program Studi', 'Dashboard Yudisium',
'Dosen Prodi', 'Mahasiswa Prodi', 'Mata Kuliah dan Kelas',
'Prodi_Kelas Ajar', 'Prodi_Mata Kuliah Ajar', 'Prodi_Dashboard
Dosen', 'Prodi_Mahasiswa Wali', 'Yudisium', 'Prodi_Dashboard
Mahasiswa', 'Prodi_KHS-Transkrip']
page_dosen = ['Dashboard Dosen', 'Mahasiswa Wali', 'Mata
Kuliah Ajar', 'Dosen_Dashboard Mahasiswa', 'Dosen_KHS-
Transkrip']
hide_pages(page_prodi+page_dosen)

#SQLITE connection
conn = sqlite3.connect('db_skripsi.db')
cursor = conn.cursor()

#Check Prodi State and other Variable
prodi = st.session_state.prodi
nim = st.session_state.userid

#Get latest periode
cursor.execute("SELECT MAX(periode) FROM krsmahasiswa WHERE
namaunit = ?", (prodi,))
latest_periode = cursor.fetchone()[0]

#Functions
def hitung_masa_studi(row):
    periodelulus = row.get('periodelulus')
    if periodelulus is None:
        periodelulus = latest_periode

    tahun_masuk, semester_masuk = row['periodemasuk'][:-2],
row['periodemasuk'][-2:]
    tahun_lulus, semester_lulus = periodelulus[:-2],
periodelulus[-2:]
    tahun_masuk, tahun_lulus = int(tahun_masuk),
int(tahun_lulus)

    total_tahun = tahun_lulus - tahun_masuk
    total_semesters = total_tahun * 2

    if semester_lulus == 'S2':
        total_semesters += 1
    if semester_masuk == 'S2':
        total_semesters -= 1

    return total_semesters + 1

def hitung_prasyarat(student, syarat_yudisium):
    min_skslulus = int(syarat_yudisium['Minimum'][0])
    min_ipk = float(syarat_yudisium['Minimum'][1])
    min_ept = int(syarat_yudisium['Minimum'][2])
    max_matkulD = int(syarat_yudisium['Minimum'][3])
```

```
    mkwajiblulus = int(syarat_yudisium['Minimum'][4])

    requirements = {
        f'Minimum {min_skslulus} SKS Lulus': student['SKS
Lulus'] >= min_skslulus,
        f'IPK >= {min_ipk}': student['IPK'] >= min_ipk,
        f'EPT score >= {min_ept}': student['Nilai EPT'] >=
min_ept,
        'Lulus Semua MK Wajib': student['MK Wajib Lulus']>=
mkwajiblulus,
        f'SKS dengan nilai D <= {max_matkulD} SKS':
student['SKS_D'] <= max_matkulD
    }
    completed = sum(requirements.values())
    total = len(requirements)
    return requirements, completed, total


#Title
st.markdown(f'### Dashboard Mahasiswa')
st.markdown(f'#### nim: {nim} - {prodi}')
st.markdown("---")

#Get Data Kurikulum
query = f"""
    SELECT kodemk, namamk, namaunit, wajibpilihan as kurikulum
    FROM kurikulum
    WHERE namaunit = '{prodi}' AND tahun = (SELECT MAX(tahun)
FROM kurikulum WHERE namaunit = '{prodi}')
"""
df_kurikulum = pd.read_sql(query, conn)
df_kurikulum['kurikulum'] =
df_kurikulum['kurikulum'].replace({'W': 'Wajib', 'P':
'Pilihan'})
df_matkulwajib = df_kurikulum[df_kurikulum['kurikulum'] ==
'Wajib']
total_matkulwajib = str(len(df_matkulwajib))

#Get Data Syarat Yudisium
query = f"SELECT minskslulus, minipk, minept, maxsksd FROM
syaratyudisium WHERE namaunit ='{prodi}' AND periode =(SELECT
MAX(periode) FROM syaratyudisium WHERE namaunit = '{prodi}')"
df_syaratyudisium = pd.read_sql(query, conn)
syarat_yudisium = {'Syarat': ['Total SKS Lulus >=', 'IPK >=',
'Nilai EPT >=', 'SKS dengan Nilai D <=', 'Lulus Semua Matkul
Wajib'],
                   'Minimum':
[int(df_syaratyudisium['minskslulus'].iloc[0]),
float(df_syaratyudisium['minipk'].iloc[0]),
int(df_syaratyudisium['minept'].iloc[0]),
int(df_syaratyudisium['maxsksd'].iloc[0]), total_matkulwajib]}
df_syarat = pd.DataFrame(syarat_yudisium)
df_syarat['Minimum'] = df_syarat['Minimum'].astype(str)
```

```
df_syarat.loc[df_syarat['Syarat'] == 'IPK', 'Minimum'] =
df_syarat.loc[df_syarat['Syarat'] == 'IPK',
'Minimum'].astype(float)

#Get Data Transkrip Mahasiswa
query_transkrip = f"""
    SELECT periode, nim, kodemk, tahunkurikulum, sks, nangka,
nhuruf
    FROM krsmahasiswa
    WHERE nim LIKE '{nim}'
"""
df_transkrip = pd.read_sql_query(query_transkrip, conn)
df_transkrip = pd.merge(df_transkrip, df_kurikulum,
on='kodemk', how='left')

data_sks_lulus =
df_transkrip.loc[~df_transkrip['nhuruf'].isin(['D', 'E', 'T',
None]), 'sks'].sum()
data_sks_tdklulus =
df_transkrip.loc[df_transkrip['nhuruf'].isin(['D', 'E']),
'sks'].sum() #Dibawah C Dianggap Tidak lulus
data_mkwajib_lulus =
df_transkrip[(~df_transkrip['nhuruf'].isin(['D', 'E', 'T',
None])) & (df_transkrip['kurikulum'] ==
'Wajib')].reset_index(drop=True)

#Hitung IPK
filtered_df_ipk =
df_transkrip.drop_duplicates(subset=['kodemk'],
keep='last').reset_index(drop=True)
filtered_df_ipk =
filtered_df_ipk[~df_transkrip['nhuruf'].isin([None, 'T'])]
total_grade_poin = (filtered_df_ipk['nangka'] *
filtered_df_ipk['sks']).sum()
total_sks = filtered_df_ipk['sks'].sum()
data_ipk = round(total_grade_poin / total_sks,2)

cursor.execute(f"SELECT nilai_ept FROM detailmahasiswa WHERE
nim = '{nim}'")
data_ept = cursor.fetchone()[0]

#Dashboard Metric Atas
left_col, mid_col, right_col = st.columns(3)
with left_col:
    left_col.subheader('Semester Aktif', anchor=False)

    query = f"""
        SELECT periodemasuk, periodelulus FROM listmahasiswa
WHERE nim like '{nim}'
    """
    cursor.execute(query)
    df = pd.DataFrame(cursor.fetchall(),
columns=['periodemasuk', 'periodelulus'])
```

```
    df['masastudi'] = df.apply(hitung_masa_studi, axis=1)

    # Data
    current_semester = df['masastudi'][0]
    limit_semester = 14

    # Buat gauge chart dengan step yang lebih kecil
    fig = go.Figure(go.Indicator(
        mode="gauge+number",
        value=current_semester,
        gauge={
            'axis': {'range': [0, limit_semester], 'tickmode':
'array', 'tickvals': list(range(0, limit_semester+1, 2))},
            'bar': {'color': "lightblue"},
            'steps': [
                {'range': [0, 8], 'color': "#F8EEEC"},
                {'range': [8, limit_semester], 'color':
"#B22222"}],
            'threshold': {
                'line': {'color': "lightblue", 'width': 4},
                'thickness': 0.75,
                'value': current_semester},
        }
    ))

    fig.update_layout(
        height=280,
        margin=dict(l=0, r=0, t=50, b=50),
        title={
            'text': "Semester",
            'y':0.15,
            'x':0.500,
            'xanchor': 'center',
            'font': {'size': 23}
        },
    )
    st.plotly_chart(fig)

with mid_col:
    mid_col.subheader('Jumlah SKS Lulus', anchor=False)
    min_sks = int(df_syarat['Minimum'][0])

    fig = go.Figure(go.Indicator(
        mode="gauge+number",
        value=data_sks_lulus,
        gauge={
            'axis': {'range': [0, min_sks], 'tickvals': [0,
36, 72, 110, 144]},
            'bar': {'color': "lightblue"},
            'steps': [
                {'range': [0,  min_sks], 'color': "#F8EEEC"}],
            'threshold': {
                'line': {'color': "lightblue", 'width': 4},
```

```
                'thickness': 0.75,
                'value': data_sks_lulus},
        }
    ))

    fig.update_layout(
        height=280,
        margin=dict(l=0, r=0, t=50, b=50),
        title={
            'text': "SKS",
            'y': 0.15,
            'x': 0.500,
            'xanchor': 'center',
            'font': {'size': 23}
        },
    )

    st.plotly_chart(fig)

with right_col:
    right_col.subheader('Syarat Yudisium', anchor=False)

    status_list = []

    for index, row in df_syarat.iterrows():
        syarat = row['Syarat']
        minimum = row['Minimum']

        if syarat == 'Total SKS Lulus >=':
            status = '❌' if data_sks_lulus is None else ('✔' 
if data_sks_lulus >= int(minimum) else '❌')
        elif syarat == 'Nilai EPT >=':
            status = '❌' if data_ept is None else ('✔' if 
data_ept >= int(minimum) else '❌')
        elif syarat == 'SKS dengan Nilai D <=':
            status = '❌' if data_sks_tdklulus is None else 
('✔' if data_sks_tdklulus < int(minimum) else '❌')
        elif syarat == 'Lulus Semua Matkul Wajib':
            status = '❌' if data_mkwajib_lulus is None else 
('✔' if len(data_mkwajib_lulus) >= int(minimum) else '❌')
        elif syarat == 'IPK >=':
            status = '❌' if data_ipk is None else ('✔' if 
data_ipk >= float(minimum) else '❌')
        else:
            status = '❌'

        if syarat != 'Lulus Semua Matkul Wajib':
            syarat = f'{syarat} {str(minimum)}'

        status_list.append((syarat, status))
```

```
    result_df = pd.DataFrame(status_list, columns=['Syarat',
'Status'])

    right_col.markdown(
        """
        <style>
        [data-testid="stElementToolbar"] {
            display: none;
        }
        </style>
        """,
        unsafe_allow_html=True
    )
    right_col.dataframe(result_df, use_container_width=True,
hide_index=True)

#Dashboard Metric Bawah
bleft_col, pad, bright_col = st.columns((1,0.1,1))
with bleft_col:
    bleft_col.subheader("Histori IPS dan IPK", anchor=False)

    query = f"SELECT periode, ips, ipk FROM akademikmahasiswa
WHERE nim = '{nim}'"
    df_ip = pd.read_sql_query(query, conn)

    fig_history = go.Figure()
    #IPS Line
    fig_history.add_trace(
        go.Scatter(
            x=df_ip['periode'],
            y=df_ip['ips'],
            name='IPS',
            mode='lines+markers',
            marker={'size':9},
            line = dict(color='blue', width=3)
        )
    )
    #IPK Line
    fig_history.add_trace(
        go.Scatter(
            x=df_ip['periode'],
            y=df_ip['ipk'],
            name='IPK',
            mode='lines+markers',
            marker={'size':9},
            line = dict(color='red', width=3)
        )
    )
    fig_history.update_layout(
        showlegend=True,
        margin=dict(t=50,l=10,b=10,r=10),
        legend_title_text='Legend',
```

```
        xaxis_tickfont_size=18, xaxis_title="Periode",
xaxis_title_font_size=20,
        yaxis_tickfont_size=18, yaxis_title="Indeks Prestasi",
yaxis_title_font_size=20,
        legend=dict(
            x=0.01,
            y=1.15,
            orientation='h',
            bgcolor='rgba(255, 255, 255, 0)',
            bordercolor='rgba(255, 255, 255, 0)',
            font_size=15
        ),
    )

    fig_history.update_traces(texttemplate='%{text:.2s}')
    fig_history.update_layout(hovermode="x unified")

    st.plotly_chart(fig_history, use_container_width=True)

with bright_col:
    bright_col.subheader("Prasyarat Lulus Matkul Wajib",
anchor=False)

    lulus_mkwajib = [len(data_mkwajib_lulus),
int(df_syarat['Minimum'][4]) - len(data_mkwajib_lulus)]
    total_matkul = lulus_mkwajib[0] + lulus_mkwajib[1]

    labels=['MK Wajib Lulus', 'MK Wajib Belum']
    custom_text = [f'{val}-MatKul' for label, val in
zip(labels, [lulus_mkwajib[0], lulus_mkwajib[1]])]
    colors = ['#1f77b4', '#d1324a']
    fig_mkwajib = go.Figure(data=[go.Pie(labels=labels,
                                         values=lulus_mkwajib,
                                         text=custom_text,
                                         hole=.4,
                                         textfont_size = 20,

marker=dict(colors=colors),)])
    fig_mkwajib.update_layout(
        margin=dict(l=0, r=0, t=60, b=90),
        legend=dict(orientation='v', x=0.75, y=1.1,
bordercolor='#f5e4e6', borderwidth=0.8, font=dict(size=18)),
        height=500,
        hoverlabel=dict(font_size=15),
    )
    fig_mkwajib.update_traces(
        hovertemplate ="<b>%{label}</b> <br> Jumlah:
%{value}",
    )

    st.plotly_chart(fig_mkwajib)
```

4.1.3 KHS

Pada halaman Kartu Hasil Studi (KHS) ini, pengguna dapat melihat rangkuman pembelajaran mahasiswa untuk setiap periode akademik. Halaman ini memungkinkan pengguna untuk mengubah filter periode untuk menampilkan tabel yang berisikan data nilai mata kuliah yang telah diambil pada periode tersebut. Dengan fitur ini pengguna dapat secara mudah memantau dan mengevaluasi kinerja akademik mereka dari satu periode ke periode lainnya.

Segmen Program 4.3 KHS

```
#Hide Pages
page_prodi = ['Dashboard Program Studi', 'Dashboard Yudisium',
'Dosen Prodi', 'Mahasiswa Prodi', 'Mata Kuliah dan Kelas',
'Prodi_Kelas Ajar', 'Prodi_Mata Kuliah Ajar', 'Prodi_Dashboard
Dosen', 'Prodi_Mahasiswa Wali', 'Yudisium', 'Prodi_Dashboard
Mahasiswa', 'Prodi_KHS-Transkrip']
page_dosen = ['Dashboard Dosen', 'Mahasiswa Wali', 'Mata
Kuliah Ajar', 'Dosen_Dashboard Mahasiswa', 'Dosen_KHS-
Transkrip']
hide_pages(page_prodi+page_dosen)

#SQLITE connection
conn = sqlite3.connect('db_skripsi.db')
cursor = conn.cursor()

#Check Prodi State and other Variable
prodi = st.session_state.prodi
nim = st.session_state.userid

#Get Data Kurikulum
query = f"""
    SELECT kodemk, namamk, namaunit, wajibpilihan as kurikulum
    FROM kurikulum
    WHERE namaunit = '{prodi}' AND tahun = (SELECT MAX(tahun)
FROM kurikulum WHERE namaunit = '{prodi}')
"""
df_kurikulum = pd.read_sql(query, conn)
df_kurikulum['kurikulum'] =
df_kurikulum['kurikulum'].replace({'W': 'Wajib', 'P':
'Pilihan'})
filtered_df = df_kurikulum[df_kurikulum['kurikulum'] ==
'Wajib']

#Functions
def extract_year(semester):
    return semester[:4]

def color_cells(row):
    if row['Kode MK'] in filtered_df['kodemk'].values:
        if row['Nilai'] in ['D', 'E', 'T']:
```

```
            return ['background-color: #EE4E4E'] * len(row)
        else:
            return ['background-color: #41dc8e'] * len(row)
    elif 'DU' in row['Kode MK']:
        if row['Nilai'] in ['D', 'E', 'T']:
            return ['background-color: #DD761C'] * len(row)
        else:
            return ['background-color: #3ABEF9'] * len(row)
    return [None] * len(row)

@st.experimental_dialog("Prasyarat Mata Kuliah Wajib",
width='large')
def show_prasyaratwkWajib(df_transkrip, df_kurikulum):
    st.write('List Mata Kuliah Wajib yang belum diselesaikan
oleh Mahasiswa')

    mk_wajib_belum_diambil =
df_kurikulum[~df_kurikulum['kodemk'].isin(df_transkrip['Kode
MK'])]
    mk_wajib_belum_diambil['Nilai'] = None

    mk_belum_lulus =
df_transkrip[df_transkrip['Nilai'].isin(['D', 'E', 'T'])]
    mk_wajib_belum_lulus = mk_belum_lulus.merge(df_kurikulum,
left_on='Kode MK', right_on='kodemk', how='inner')
    mk_wajib_belum_lulus = mk_wajib_belum_lulus[['kodemk',
'namamk', 'Nilai']]

    df_result = pd.concat([mk_wajib_belum_diambil[['kodemk',
'namamk', 'Nilai']], mk_wajib_belum_lulus])
    df_result.columns = ['Kode MK', 'Nama MK', 'Nilai']

    df_result = df_result.sort_values('Nama MK',
ascending=True)
    df_result.reset_index(drop=True, inplace=True)
    df_result.index += 1

    st.dataframe(df_result, use_container_width=True)

tabkhs, tabtranskrip = st.tabs(["KHS","Transkrip"])
with tabtranskrip:
    with st.sidebar:
        st.write('')
        btn_prasyaratmk = st.button('Prasyarat MK Wajib',
help='List MK Wajib yang belum terselesaikan',
use_container_width=True)

    query = f"""
        SELECT krs.periode, krs.kodemk, krs.namamk, krs.sks,
krs.nhuruf, nangka
        FROM krsmahasiswa krs
        WHERE krs.nim ='{nim}' AND krs.nhuruf is not NULL
        ORDER by krs.periode asc
```

```
    """
    df_transkrip = pd.read_sql_query(query, conn)
    df_transkrip.columns = ['Periode', 'Kode MK', 'Nama MK',
'SKS', 'Nilai', 'Nilai Angka']

    df_transkrip = df_transkrip.sort_values(by='Periode')
    df_transkrip = df_transkrip.drop_duplicates(subset=['Kode
MK', 'Nama MK'], keep='last').reset_index(drop=True)
    df_transkrip['Nilai'] = df_transkrip['Nilai'].str.strip()

    df_transkrip['Total Poin'] = df_transkrip['Nilai Angka'] *
df_transkrip['SKS']
    df_filterT = df_transkrip[df_transkrip['Nilai'] != 'T']
    total_poin = df_filterT['Total Poin'].sum()
    total_sks = df_filterT['SKS'].sum()

    df_filterD = df_transkrip[(df_transkrip['Nilai'] == 'D') |
(df_transkrip['Nilai'] == 'E')]
    total_sks_tidaklulus = df_filterD['SKS'].sum()

    ipk = round(total_poin / total_sks, 2) if total_sks != 0
else 0

    cursor.execute(f"SELECT nilai_ept FROM detailmahasiswa
WHERE nim = '{nim}'")
    data_ept = cursor.fetchone()[0]
    nilai_ept = data_ept if data_ept is not None else '-'

    transkrip_view = df_transkrip[['Periode', 'Kode MK', 'Nama
MK', 'SKS', 'Nilai']]
    cursor.execute('SELECT periodemasuk FROM listmahasiswa
WHERE nim = ?', (nim, ))
    angakatan = extract_year(cursor.fetchone()[0])

    st.markdown(f"<h3 style='text-align: center;'>Tranksrip",
unsafe_allow_html=True)

    if len(transkrip_view)<=20:
        pad,info,val,empty,pad = st.columns((2,1,2,7,2))
        with info:
            st.write('**NIM**')
            st.write('**Prodi**')
            st.write('**Angkatan**')
        with val:
            st.write(f'**{nim}**')
            st.write(f"**{prodi}**")
            st.write(f"**{angakatan}**")

        pad,mid,pad = st.columns((2,10,2))
        with mid:
            toggle_mkwajib = st.toggle('Cek Kurikulum')
            if toggle_mkwajib:
```

```
                styled_transkrip_view =
transkrip_view.style.apply(color_cells, axis=1)
            else: styled_transkrip_view = transkrip_view

            st.table(styled_transkrip_view)
            st.code(f'''
                Indeks Prestasi Kumulatif       {ipk}
                Jumlah SKS Ambil                {total_sks}
                Jumlah SKS Lulus                {total_sks-
total_sks_tidaklulus}
                Nilai EPT                       {nilai_ept}
            ''')
            if toggle_mkwajib:
                st.code(
                    """
                    Catatan Warna Fitur Cek Kurikulum:
                    - Hijau:    MK Wajib yang Sudah Lulus
                    - Merah:    MK Wajib yang Belum Lulus

                    - Biru:     MKDU yang Sudah Lulus
                    - Oranye:   MKDU yang Belum Lulus
                    """,
                )
    else:
        pad,info,val,empty,pad = st.columns((2,1.5,2.5,6,2))
        with info:
            st.write('**NIM**')
            st.write('**Prodi**')
            st.write('**Angkatan**')
            toggle_mkwajib = st.toggle('Cek Kurikulum')

        with val:
            st.write(f'**{nim}**')
            st.write(f"**{prodi}**")
            st.write(f"**{angakatan}**")

        midpoint = int(len(transkrip_view)/2)
        table_left = transkrip_view.iloc[:midpoint+1]
        table_right = transkrip_view.iloc[midpoint+1:]

        pad,left,right,pad = st.columns((2,5,5,2))
        with left:
            if toggle_mkwajib:
                styled_table_left =
table_left.style.apply(color_cells, axis=1)
            else: styled_table_left = table_left

            st.table(styled_table_left)

            st.code(f'''
                Indeks Prestasi Kumulatif       {ipk}
                Jumlah SKS Ambil                {total_sks}
```

```
                Jumlah SKS Lulus                {total_sks-
total_sks_tidaklulus}
                Nilai EPT                       {nilai_ept}
            ''')
            if toggle_mkwajib:
                st.code(
                    """
                    Catatan Warna Fitur Cek Kurikulum:
                    - Hijau:    MK Wajib yang Sudah Lulus
                    - Merah:    MK Wajib yang Belum Lulus

                    - Biru:     MKDU yang Sudah Lulus
                    - Oranye:   MKDU yang Belum Lulus
                    """,
                )

        with right:
            if toggle_mkwajib:
                styled_table_right =
table_right.style.apply(color_cells, axis=1)
            else: styled_table_right = table_right

            st.table(styled_table_right)

    if btn_prasyaratmk:
            show_prasyaratwkWajib(df_transkrip, filtered_df)
```

4.1.4 Transkrip Mahasiswa

Pada halaman ini, pengguna dapat melihat keseluruhan transkrip nilai mata kuliah yang telah diselesaikan dan dinyatakan lulus. Halaman ini akan menampilkan informasi mengenai mata kuliah beserta nilai yang telah diperoleh, serta informasi mengenai total SKS yang telah diselesaikan dan IPK terakhir mahasiswa.

Segmen Program 4.4 Transkrip Mahasiswa

```
with tabkhs:
    st.markdown("<h2 style='text-align: center;'>Nilai Mata
Kuliah Mahasiswa</h2>", unsafe_allow_html=True)
    st.markdown("<br>",True)

    #Center container
    container_style = "align: center;"

    #Info Mahasiswa
    cursor.execute("SELECT kodeunit, namaunit, ipk,
nipdosenwali FROM listmahasiswa WHERE nim = ?",(nim,))
    data = cursor.fetchall()
    detail_mahasiswa = pd.DataFrame(data, columns=['kodeunit',
'namaunit', 'ipk', 'nipdosenwali'])
```

```
    cursor.execute("SELECT kodefakultas FROM listunit WHERE
kodeunit = ?",(detail_mahasiswa['kodeunit'][0],))
    data = cursor.fetchone()[0]
    nama_fakultas = data

    cursor.execute('SELECT nama FROM pegawai WHERE nip = ?',
(detail_mahasiswa['nipdosenwali'][0], ))
    dosen_wali = cursor.fetchone()[0]

    with st.markdown(f"<div style='{container_style}'>",
unsafe_allow_html=True):
        infocol1, valuecol1, padmid, infocol2, valuecol2=
st.columns((2,3,1,2,3))
        with infocol1:
            st.write("##### NIM")
            st.write("##### Dosen Wali")
            st.write("##### Total SKS Lulus")
        with valuecol1:
            st.write(f"##### {nim}")
            st.write(f"##### {dosen_wali}")
            st.write(f"##### {total_sks-
total_sks_tidaklulus}")
        with infocol2:
            st.write("##### Fakultas")
            st.write("##### Jurusan")
            st.write("##### IPK")
        with valuecol2:
            st.write(f"##### {nama_fakultas}")
            st.write(f"#####
{detail_mahasiswa['namaunit'][0]}")
            st.write(f"##### {detail_mahasiswa['ipk'][0]}")

    st.markdown("</div>", unsafe_allow_html=True)

    query = f"""
        SELECT periode, kodemk, namamk, sks, nhuruf, nangka
        FROM raw_krsmahasiswa
        WHERE nim = ?
        ORDER BY periode, kodemk ASC
    """
    cursor.execute(query,(nim,))
    data = cursor.fetchall()
    khs = pd.DataFrame(data, columns=['Periode', 'Kode MK',
'Mata Kuliah', 'SKS', 'Nilai', 'nangka'])
    khs = khs[(khs['Periode'] >= '20231') | ((khs['Periode'] <
'20231') & khs['Nilai'].notna())]

    temp_khs = khs.sort_values(by='Periode')
    non_null_khs = khs.dropna(subset=['Nilai'])
    temp_khs = non_null_khs.drop_duplicates(subset=['Kode
MK'], keep='last').reset_index(drop=True)
```

```
    null_khs = khs[khs['Nilai'].isna()]
    temp_khs = pd.concat([temp_khs,
null_khs]).reset_index(drop=True)


    temp_khs['poin'] = temp_khs['SKS'] * temp_khs['nangka']

    periode_list = []
    total_sks_list = []
    total_sks_cumulative_list = []
    ips_list = []
    ipk_list = []

    total_sks_cumulative = 0
    total_poin_cumulative = 0
    previous_ipk = 0
    previous_sks_cumulative = 0

    periods = temp_khs['Periode'].unique()
    for i, period in enumerate(temp_khs['Periode'].unique()):
        period_data = temp_khs[temp_khs['Periode'] == period]
        # print(period_data)

        if len(period_data) == 1 and
period_data['Nilai'].iloc[0] == 'T ':
            total_sks = period_data['SKS'].sum()
            total_poin = period_data['poin'].sum()
            ips = round(total_poin / total_sks,2) if total_sks
> 0 else 0
            total_sks_cumulative = previous_sks_cumulative
            ipk = previous_ipk
        elif period_data['Nilai'].isnull().all():
            total_sks = period_data['SKS'].sum()
            total_poin = period_data['poin'].sum()
            ips = round(total_poin / total_sks,2) if total_sks
> 0 else 0
            total_sks_cumulative = previous_sks_cumulative
            ipk = previous_ipk
        else:
            total_sks = period_data['SKS'].sum()
            total_poin = period_data['poin'].sum()
            ips = round(total_poin / total_sks,2) if total_sks
> 0 else 0

            total_sks_cumulative += total_sks
            total_poin_cumulative += total_poin
            ipk = round(total_poin_cumulative /
total_sks_cumulative,2) if total_sks_cumulative > 0 else 0

            previous_ipk = ipk
            previous_sks_cumulative = total_sks_cumulative

        periode_list.append(period)
```

```
        total_sks_list.append(total_sks)
        total_sks_cumulative_list.append(total_sks_cumulative)
        ips_list.append(ips)
        ipk_list.append(ipk)

    df_ip = pd.DataFrame({
        'periode': periode_list,
        'sks_now': total_sks_list,
        'sks_cumulative': total_sks_cumulative_list,
        'ips': ips_list,
        'ipk': ipk_list
    })

    cursor.execute("SELECT DISTINCT periode FROM krsmahasiswa
WHERE nim = ? ORDER BY periode DESC", (nim,))
    list_periode = [str(periode[0]) for periode in
cursor.fetchall()]

    periode_filter, pad= st.columns((2,9))
    selected_periode = periode_filter.selectbox('Pilih
Periode', options=list_periode)

    if selected_periode:
        filtered_khs = khs[khs['Periode'] ==
selected_periode].reset_index(drop=True)
        filtered_khs.index = filtered_khs.index + 1
        filtered_khs = filtered_khs.drop(columns=['nangka'])
        filtered_khs['Nilai'] =
filtered_khs['Nilai'].fillna('')
        st.markdown(f"<h3 style='text-align: center;'>Nilai
Mata Kuliah Periode {selected_periode}</h3>",
unsafe_allow_html=True)
        st.table(filtered_khs)

        filtered_ip = df_ip[df_ip['periode'] ==
selected_periode]

        infoakademik, pad= st.columns((5,6))
        with infoakademik:
            st.code(f'''
                Jumlah SKS yang diambil semester ini
{filtered_ip['sks_now'].to_string(index=False)}
                IPS yang diperoleh
{filtered_ip['ips'].to_string(index=False)}
                Total SKS yang telah ditempuh
{filtered_ip['sks_cumulative'].to_string(index=False)}
                IPK mata kuliah yang telah ditempuh
{filtered_ip['ipk'].to_string(index=False)}
            ''')
```

4.1.5 Dashboard Dosen Wali

Halaman *dashboard* dosen wali ini dapat dilihat oleh *user* dosen dengan tujuan memudahkan dosen wali untuk memantau progres akademik mahasiswa walinya secara keseluruhan dengan memvisualisasikan data akademik mahasiswa wali, dimana data yang dapat dilihat pada dashboard ini adalah jumlah mahasiswa wali yang aktif, mahasiswa wali yang tidak lulus mata kuliah di periode sebelumnya, mahasiswa yang mengambil ulang mata kuliah yang pernah diambilnya atau mengulang di periode ini, range penyelesaian SKS dan IPK mahasiswa wali.

Segmen Program 4.5 Dashboard Dosen Wali

```
#Hide Pages
page_prodi = ['Dashboard Program Studi', 'Dashboard Yudisium',
'Dosen Prodi', 'Mahasiswa Prodi', 'Mata Kuliah dan Kelas',
'Prodi_Kelas Ajar', 'Prodi_Mata Kuliah Ajar', 'Prodi_Dashboard
Dosen', 'Prodi_Mahasiswa Wali', 'Yudisium', 'Prodi_Dashboard
Mahasiswa', 'Prodi_KHS-Transkrip']
page_dosen = ['Dosen_Dashboard Mahasiswa', 'Dosen_KHS-
Transkrip']
page_mahasiswa = ['Dashboard Mahasiswa', 'KHS-Transkrip']
hide_pages(page_prodi+page_dosen+page_mahasiswa)

#SQLITE connection
conn = sqlite3.connect('db_skripsi.db')
cursor = conn.cursor()

#Check Prodi State and other Variable
prodi = st.session_state.prodi
nip = st.session_state.userid

cursor.execute(f"SELECT nama FROM pegawai WHERE nip = '{nip}'")
dosen = cursor.fetchone()[0]
st.session_state.dosen = dosen

#Get latest periode
cursor.execute("SELECT MAX(periodelulus) FROM akademikmahasiswa
WHERE namaunit = ?", (prodi,))
latest_periode = cursor.fetchone()[0]

#Function
def extract_year(semester):
    return semester[:4]

@st.experimental_dialog("Mahasiswa Wali Tidak Lulus MK",
width='large')
def show_mhsTidakLulus(df, periode):
    st.write(f'List Mahasiswa yang tidak lulus pada satu atau
lebih mata kuliah periode {periode}')
```

```
    gb = GridOptionsBuilder.from_dataframe(df)
    gb.configure_default_column(groupable=True,)
    gb.configure_column("nim", pinned="left", width=50)
    gb.configure_pagination(paginationAutoPageSize=True)
    gridOptions = gb.build()
    custom_css = {
        ".ag-theme-streamlit .ag-cell": {"font-size": "18px"}
    }
    AgGrid(df, gridOptions=gridOptions, custom_css=custom_css)
    if not df.empty:
        st.download_button(
            label="Download as CSV",
            data=df.to_csv(index=False).encode('utf-8'),
            file_name=f'list_mhswali_tdklulusmk_{dosen}.csv',
            mime='text/csv',
        )

@st.experimental_dialog("Mahasiswa Wali Mengambil Ulang MK",
width='large')
def show_mhsUlangMK(df, periode):
    st.write(f'List Mahasiswa yang mengambil ulang mata kuliah
di awal periode {periode}')

    def aggregate_kodemk(kodemk_list):
        kode_counts = {}
        for kode in kodemk_list:
            kode_counts[kode] = kode_counts.get(kode, 0) + 1
        return ', '.join([f"{count}x {kode}" for kode, count in
kode_counts.items()])

    df['Pengulangan MK'] = df['kodemk_namamk'].apply(lambda x:
aggregate_kodemk(x.split(', ')))
    df = df.drop('kodemk_namamk', axis=1)

    gb = GridOptionsBuilder.from_dataframe(df)
    gb.configure_default_column(groupable=True,)
    gb.configure_column("nim", pinned="left")
    gb.configure_pagination(paginationAutoPageSize=True)
    gridOptions = gb.build()
    custom_css = {
        ".ag-theme-streamlit .ag-cell": {"font-size": "18px"}
    }
    AgGrid(df, gridOptions=gridOptions, height=300,
custom_css=custom_css)

    if not df.empty:
        st.download_button(
            label="Download as CSV",
            data=df.to_csv(index=False).encode('utf-8'),
            file_name=f'list_mhswali_ulangmk_{dosen}.csv',
            mime='text/csv',
        )
```

```
with st.sidebar:
    cursor.execute(f"SELECT DISTINCT periode FROM krsmahasiswa
WHERE namaunit like '{prodi}' ORDER BY periode DESC")
    list_periode = [str(periode[0]) for periode in
cursor.fetchall()]
    selected_periode = st.selectbox("Pilih Periode:",
options=list_periode)

#Title and Sub-Title
st.markdown(f'### Dashboard Dosen Wali - {prodi}')
st.markdown(f'#### {dosen} - {selected_periode}')
st.markdown("---")

st.markdown("""
    <style>
    button[kind="primary"] {background: none!important; border:
none; padding: 0!important; margin: 0!important;
        color: blue !important; text-decoration: none; cursor:
pointer; border: none !important;
    }
    button[kind="primary"]:hover {
        text-decoration: none; color: red !important;
    }
    button[kind="primary"]:focus {
        outline: none !important; box-shadow: none !important;
color: blue !important;
    }

    button[kind="secondary"] {background: none!important;
border: none; padding: 0!important; margin: 0!important;
        color: black !important; text-decoration: none; cursor:
pointer; border: none !important;
    }
    button[kind="secondary"]:hover {
        text-decoration: none; color: black !important;
    }
    button[kind="secondary"]:focus {
        outline: none !important; box-shadow: none !important;
color: black !important;
    }
    </style>
    """,
    unsafe_allow_html=True,
)

if dosen:
    #Dasboard Bagian Atas
    topleft_col, topmid_col, topright_col = st.columns(3)
    with topleft_col:
        topleft_col.subheader("Mahasiswa Wali Aktif",
anchor=False)
```

```
        topleft_col.button(f'Periode - {selected_periode}',
disabled=True)

        query= f"SELECT nim, periodemasuk, periodelulus FROM
listmahasiswa WHERE nipdosenwali LIKE '{nip}' AND (periodelulus
> '{selected_periode}' OR periodelulus is NULL) AND
periodemasuk <= '{selected_periode}' AND namaunit LIKE
'{prodi}'"
        df_nim = pd.read_sql_query(query, conn)
        list_nim = tuple(df_nim['nim'].unique())
        topleft_col.title(len(df_nim), anchor=False)

    with topmid_col:
        topmid_col.subheader("Mahasiswa Wali Tidak Lulus MK",
anchor=False)

        index = list_periode.index(selected_periode)

        if index + 1 < len(list_periode):
            periode_behind = list_periode[index + 1]
        else:
            periode_behind = None

        query = """
            SELECT krs.nim, krs.angkatan, krs.kodemk,
krs.namamk, krs.kelasmk, krs.nhuruf
            FROM krsmahasiswa krs
            JOIN listmahasiswa lm ON krs.nim = lm.nim
            JOIN pegawai pg ON lm.nipdosenwali = pg.nip
            WHERE krs.periode like '{}' AND krs.nhuruf >= 'D'
AND pg.nip = '{}' AND lm.namaunit = '{}'
            ORDER BY krs.nim
        """

        df =
pd.read_sql_query(query.format(selected_periode,nip,prodi),
conn)
        if not df.empty:
            df_grouping = df.groupby(['nim',
'angkatan']).apply(lambda x: ', '.join(x['kodemk'] + '-' +
x['namamk'])).reset_index(name='kodemk_namamk')
            if topmid_col.button(f"Periode -
{selected_periode}", help='Click to see more details',
type='primary', key='topmid1'):
                show_mhsTidakLulus(df_grouping,
selected_periode)

            topmid_col.title(len(df_grouping), anchor=False)
        else:
            topmid_col.button(f"Periode - {selected_periode}",
help='Empty Data', type='primary', key='topmid2')
            topmid_col.title(len(df), anchor=False)
```

```
    with topright_col:
        topright_col.subheader("Mahasiswa Wali Mengambil Ulang
MK", anchor=False)

        query_previous = f"""
            SELECT nim, angkatan, kodemk
            FROM krsmahasiswa
            WHERE periode < '{selected_periode}' AND nhuruf IS
NOT NULL AND namaunit like '{prodi}'
            AND nim IN {list_nim}
        """

        query_current = f"""
            SELECT DISTINCT nim, angkatan, kodemk, namamk
            FROM krsmahasiswa
            WHERE periode = '{selected_periode}' AND namaunit
like '{prodi}'
            AND nim IN {list_nim}
        """

        df_previous = pd.read_sql_query(query_previous, conn)
        df_current = pd.read_sql_query(query_current, conn)
        merged_df = pd.merge(df_current, df_previous,
on=['nim', 'angkatan', 'kodemk'], how='inner')

        if not merged_df.empty:
            df_grouping2 = merged_df.groupby(['nim',
'angkatan']).apply(lambda x: ', '.join(x['kodemk'] + '-' +
x['namamk'])).reset_index(name='kodemk_namamk')

            if topright_col.button(f"Periode -
{selected_periode}", help='Click to see more details',
type='primary', key='topright1'):
                show_mhsUlangMK(df_grouping2, selected_periode)

            topright_col.title(len(df_grouping2), anchor=False)
        else:
            topright_col.button(f"Periode -
{selected_periode}", help='Empty Data', type='primary',
key='topright2')
            topright_col.title(len(merged_df), anchor=False)

    #Bawah
    st.markdown('<br><br>', True)
    bottomleft_col, bottomright_col = st.columns(2)
    with bottomleft_col:
        if selected_periode == latest_periode:
            fig_sks = go.Figure()
            fig_sks.update_layout(
                title=dict(text="Range Penyelesaian SKS
Mahasiswa Wali", font=dict(size=25)),
                xaxis =  { "visible": False },
                yaxis = { "visible": False },
```

```
                annotations = [
                    {
                        "text": "Periode Masih Berjalan",
                        "showarrow": False,
                        "font": {
                            "size": 28
                        }
                    }
                ]
            )
            st.plotly_chart(fig_sks, True)
        else:
            #Query Hitung semua termasuk yang cuti
            query_sks = f"""
                SELECT a.nim, a.totalsks, a.periodemasuk,
periode
                FROM akademikmahasiswa a
                WHERE a.nim in {list_nim} AND a.periode = (
                    SELECT MAX(b.periode)
                    FROM akademikmahasiswa AS b
                    WHERE a.nim = b.nim AND b.periode <=
'{selected_periode}'
                )
            """
            cursor.execute(query_sks)
            data = cursor.fetchall()
            df_sks= pd.DataFrame(data, columns=['nim', 'sks',
'periodemasuk', 'periode'])
            df_sks['angkatan'] =
df_sks['periodemasuk'].apply(extract_year)

            sks_ranges = [(0, 19), (20, 39), (40, 59), (60,
79), (80, 99), (100, 119), (120, 139), (140, float('inf'))]
            bins = pd.IntervalIndex.from_tuples([*sks_ranges[:-
1], (sks_ranges[-1][0], sks_ranges[-1][1])], closed='both')
            labels = [f"{start} - {end}" for start, end in
sks_ranges[:-1]] + ["140+"]
            df_sks['SKS Range'] = pd.cut(df_sks['sks'],
bins=bins, labels=labels)

            grouped_df_sks = df_sks.groupby(['SKS Range',
'angkatan']).size().reset_index(name='Count')
            grouped_df_sks['SKS Range'] = grouped_df_sks['SKS
Range'].astype(str)

            fig_sks = px.bar(grouped_df_sks, x='SKS Range',
y='Count', color='angkatan', barmode='stack', labels={'SKS
Range': 'Range SKS', 'Count': 'Banyak Mahasiswa',

'angkatan': 'Angkatan'}, text='Count')

fig_sks.update_traces(textposition='inside',textfont_size=18,
textangle=0)
```

```
            fig_sks.update_layout(title=dict(text="Range
Penyelesaian SKS Mahasiswa Wali", font=dict(size=25)),
hoverlabel=dict(font_size=15),

legend=dict(font=dict(size=15)), uniformtext=dict(mode='hide',
minsize=15))
            fig_sks.update_xaxes(tickvals=grouped_df_sks['SKS
Range'].unique(), ticktext=labels, tickfont=dict(size=15),)
            fig_sks.update_yaxes(tickfont=dict(size=15),)
            bottomleft_col.plotly_chart(fig_sks,True)

    with bottomright_col:
        if selected_periode == latest_periode:
            fig_sks = go.Figure()
            fig_sks.update_layout(
                title=dict(text="Range IPK Mahasiswa Wali",
font=dict(size=25)),
                xaxis =  { "visible": False },
                yaxis = { "visible": False },
                annotations = [
                    {
                        "text": "Periode Masih Berjalan",
                        "showarrow": False,
                        "font": {
                            "size": 28
                        }
                    }
                ]
            )
            st.plotly_chart(fig_sks, True)
        else:
            query_ipk = f"""
                SELECT a.nim, a.ipk, a.periodemasuk
                FROM akademikmahasiswa a
                WHERE a.nim in {list_nim} AND a.periode = (
                    SELECT MAX(b.periode)
                    FROM akademikmahasiswa AS b
                    WHERE a.nim = b.nim AND b.periode <=
'{selected_periode}'
                )
            """
            cursor.execute(query_ipk)
            data = cursor.fetchall()
            df_ipk = pd.DataFrame(data, columns=['nim', 'ipk',
'periodemasuk'])
            df_ipk['angkatan'] =
df_ipk['periodemasuk'].apply(extract_year)

            ipk_ranges = [(0.00, 1.00), (1.01, 1.50), (1.51,
2.00), (2.01, 2.50), (2.51, 3.00), (3.01, 3.50), (3.51, 4.00)]
            bins = pd.IntervalIndex.from_tuples([*ipk_ranges[:-
1], (ipk_ranges[-1][0], ipk_ranges[-1][1])], closed='both')
```

```
            labels = [f"{start:.2f}-{end:.2f}" for start, end
in ipk_ranges]
            df_ipk['IPK Range'] = pd.cut(df_ipk['ipk'],
bins=bins, labels=labels)

            grouped_df_ipk = df_ipk.groupby(['IPK Range',
'angkatan']).size().reset_index(name='Count')
            grouped_df_ipk['IPK Range'] = grouped_df_ipk['IPK
Range'].astype(str)

            fig_ipk = px.bar(grouped_df_ipk, x='IPK Range',
y='Count', color='angkatan', barmode='group', labels={'IPK
Range': 'Range IPK', 'Count': 'Banyak Mahasiswa',

'angkatan': 'Angkatan'}, text='Count')

fig_ipk.update_traces(textposition='inside',textfont_size=18,
textangle=0)
            fig_ipk.update_layout(title=dict(text="Range IPK
Mahasiswa Wali", font=dict(size=25)),
hoverlabel=dict(font_size=15),

legend=dict(font=dict(size=15)), uniformtext=dict(mode='hide',
minsize=15))
            fig_ipk.update_xaxes(tickvals=grouped_df_ipk['IPK
Range'].unique(), ticktext=labels, tickfont=dict(size=15),)
            fig_ipk.update_yaxes(tickfont=dict(size=15),)
            bottomright_col.plotly_chart(fig_ipk,True)
```

4.1.6 Mata Kuliah Ajaran Dosen

Halaman mata kuliah dosen berisikan informasi mengenai mata kuliah yang diajarkan oleh dosen dalam periode tertentu, pada halaman ini terdapat tabel berisikan semua mata kuliah yang diajarkan dosen per periode, *filter* periode untuk mengatur data yang ditampilkan, *filter* pemilihan kelas untuk melihat hasil kegiatan pengajaran per kelas mata kuliah, *summary* dari hasil perkuliahan kelas tersebut dan juga chart distribusi dari nilai akhir menggunakan chart *emperical cumulative distribution function* atau *quantile-quantile* plot sesuai input dari dosen, dimana nilai yang digunakan disini di dapatkan melalui konversi nilai Huruf (E-A) menjadi nilai simulasi mengikut range-range tertentu, adanya tabel hasil nilai pembelajaran dan tabel juga dapat didownload menjadi file CSV untuk kebutuhan lebih lanjutnya.

Segmen Program 4.6 List Mata Kuliah Dosen

```
#Hide Pages Outside User Prodi
page_prodi = ['Dashboard Program Studi', 'Dashboard Yudisium',
'Dosen Prodi', 'Mahasiswa Prodi', 'Mata Kuliah dan Kelas',
'Prodi_Kelas Ajar', 'Prodi_Mata Kuliah Ajar', 'Prodi_Dashboard
```

```
Dosen', 'Prodi_Mahasiswa Wali', 'Yudisium', 'Prodi_Dashboard
Mahasiswa', 'Prodi_KHS-Transkrip']
page_dosen = ['Dosen_Dashboard Mahasiswa', 'Dosen_KHS-
Transkrip']
page_mahasiswa = ['Dashboard Mahasiswa', 'KHS-Transkrip']
hide_pages(page_prodi+page_dosen+page_mahasiswa)

#SQLITE connection
conn = sqlite3.connect('db_skripsi.db')
cursor = conn.cursor()

#Check Prodi State
prodi = st.session_state.prodi
dosen = st.session_state.dosen
nip_dosen = st.session_state.userid

#Get latest periode
cursor.execute("SELECT MAX(periode) FROM krsmahasiswa WHERE
namaunit = ?", (prodi,))
latest_periode = cursor.fetchone()[0]

#Functions
def get_mhs_matkulajar(periode, nip_dosen, prodi, cursor):
    query= f"""
        SELECT krs.nim, krs.angkatan, krs.namaunit,
krs.kodemk, krs.namamk, krs.kelasmk, krs.nangka, krs.nhuruf,
krs.nsimulasi
        FROM krsmahasiswa krs
        JOIN pengajar rp ON krs.periode || krs.kodemk ||
krs.kelasmk = rp.periode || rp.kodemk || rp.kelasmk
        WHERE rp.periode = ? AND rp.nip = ? AND rp.namaunit =
?
    """
    if periode != latest_periode:
        query += " AND krs.nhuruf IS NOT NULL"

    cursor.execute(query, (periode, nip_dosen, prodi))
    data = cursor.fetchall()
    df = pd.DataFrame(data, columns=['NIM', 'Angkatan',
'Jurusan', 'Kode MK', 'Mata Kuliah', 'Kelas MK', 'Nilai
Angka', 'Nilai Huruf', 'Nilai Simulasi'])

    return df

def get_mhsulangmk(periode, kodemk, nim_query, conn):
    query_previous = f"""
        SELECT nim, kodemk
        FROM krsmahasiswa
        WHERE periode < '{periode}'
        AND kodemk LIKE '{kodemk}'
        AND nim {nim_query}
    """
    query_current = f"""
```

```
        SELECT DISTINCT nim, kodemk
        FROM krsmahasiswa
        WHERE periode = '{periode}'
        AND kodemk LIKE '{kodemk}'
        AND nim {nim_query}
    """

    df_previous = pd.read_sql_query(query_previous, conn)
    df_current = pd.read_sql_query(query_current, conn)
    merged_df = pd.merge(df_current, df_previous, on=['nim',
'kodemk'], how='inner')
    print(merged_df)

    retake_counts = merged_df['nim'].value_counts()
    retake_counts = retake_counts[retake_counts > 0]
    return retake_counts

with st.sidebar:
    cursor.execute(f"SELECT DISTINCT periode FROM krsmahasiswa
WHERE namaunit like '{prodi}' ORDER BY periode DESC")
    list_periode = [str(periode[0]) for periode in
cursor.fetchall()]
    select_periode = st.selectbox("Pilih Periode:",
options=list_periode)

#Title and Sub-Title
st.write(f'### Mata Kuliah Ajaran - {prodi}')
st.markdown("---")

df_mhs_matkulajar = get_mhs_matkulajar(select_periode,
nip_dosen, prodi,  cursor) #CEK LAGI

jumlahmatkul = df_mhs_matkulajar['Mata Kuliah'].nunique()
jumlahkelas = df_mhs_matkulajar[['Mata Kuliah', 'Kelas
MK']].drop_duplicates().shape[0]

jumlahmhs = df_mhs_matkulajar.dropna(subset=['Mata Kuliah',
'Nilai Huruf']).shape[0]
jumlahmhs_lulus = df_mhs_matkulajar[df_mhs_matkulajar['Nilai
Angka'] >= 2].shape[0]
if jumlahmhs != 0:
    percentage_lulus = round((jumlahmhs_lulus / jumlahmhs) *
100, 2)
else:
    percentage_lulus = 0

st.markdown("<br>",True)
#Detail Info Pengajar
infodosen_left, valuedosen_left, padmid, infodosen_right,
valuedosen_right = st.columns((2,3,1,2,3))
with infodosen_left:
    st.write('##### NIP')
    st.write('##### Nama Dosen')
```

```
    st.write('##### Jurusan')
with valuedosen_left:
    st.write(f'##### {nip_dosen}')
    st.write(f'##### {dosen}')
    st.write(f'##### {prodi}')
with infodosen_right:
    st.write('##### Jumlah Matkul Ajar')
    st.write('##### Jumlah Mhs Ajar')
    st.write('##### Persentase Kelulusan')
with valuedosen_right:
    st.write(f'##### {jumlahmatkul} Mata Kuliah |
{jumlahkelas} Kelas')
    st.write(f'##### {jumlahmhs} Mahasiswa')
    st.write(f'##### {percentage_lulus} %')

st.markdown("<br>",True)
st.markdown(f"<h3 style='text-align: center;'>Mata Kuliah
Ajaran - {select_periode}</h3>", unsafe_allow_html=True)
group_df_kelas = df_mhs_matkulajar.groupby(['Kode MK', 'Mata
Kuliah', 'Kelas MK']).NIM.count().reset_index(name='Jumlah
Mahasiswa')

mhs_lulus = df_mhs_matkulajar[df_mhs_matkulajar['Nilai Angka']
>= 2]
group_df_lulus = mhs_lulus.groupby(['Kode MK', 'Mata Kuliah',
'Kelas MK']).NIM.count().reset_index(name='Jumlah Lulus')

merged_df = pd.merge(group_df_kelas, group_df_lulus, on=['Kode
MK', 'Mata Kuliah', 'Kelas MK'], how='left')
merged_df['Jumlah Lulus'] = merged_df['Jumlah
Lulus'].fillna(0)

merged_df['Persentase Kelulusan'] = ((merged_df['Jumlah
Lulus'] / merged_df['Jumlah Mahasiswa']) * 100).round(2)

view_df = merged_df[['Kode MK', 'Mata Kuliah', 'Kelas MK',
'Jumlah Mahasiswa', 'Persentase Kelulusan']]
st.dataframe(view_df, hide_index=True,
use_container_width=True)

options = ['Pilih Kelas'] + [f"{row['Mata Kuliah']} -
{row['Kelas MK']}" for index, row in merged_df.iterrows()]
selected_index = st.selectbox("Pilih Kelas",
range(len(options)), format_func=lambda x: options[x])

pie_chart, info_col, bar_chart = st.columns((1,1,2))

if selected_index:
    index = selected_index - 1
    kodemk = merged_df.iloc[index]['Kode MK']
    matakuliah = merged_df.iloc[index]['Mata Kuliah']
    kelasmk = merged_df.iloc[index]['Kelas MK']
    count_tidaklulus = merged_df
```

```
    filtered_data = df_mhs_matkulajar[(df_mhs_matkulajar['Kode
MK'] == kodemk) & (df_mhs_matkulajar['Kelas MK'] ==
kelasmk)].reset_index(drop=True)
    st.markdown("<br>",True)
    st.markdown(f"<h3 style='text-align: center;'>{matakuliah}
- {kelasmk}</h3>", unsafe_allow_html=True)
    st.markdown("<br>",True)
    st.markdown("#### Summary")

    pie_chart, info_col, bar_chart = st.columns((1,1,2))
    with pie_chart:
        angkatan_counts =
filtered_data['Angkatan'].value_counts().reset_index()
        angkatan_counts.columns = ['Angkatan', 'Count']

        fig = px.pie(angkatan_counts, values='Count',
names='Angkatan', hole=0.5)
        fig.update_traces(textfont_size=15,
hoverinfo='label+percent', textinfo='label',
insidetextfont_size=20, outsidetextfont_size=20)
        fig.update_layout(
            width=350, height=200,
            margin=dict(l=0, r=0, t=30, b=0),
            showlegend=False,
            hoverlabel=dict(font_size=17)
        )
        st.plotly_chart(fig)

    with info_col:
        if len(filtered_data) > 1:
            nim_query = 'IN ' +
str(tuple(filtered_data['NIM']))
        else:
            nim_query = f'LIKE "{filtered_data["NIM"][0]}"'
        index_periode = list_periode.index(select_periode)
        prev_periode = list_periode[index_periode + 1]

        query = f"""
            SELECT AVG(ips), AVG(ipk)
            FROM akademikmahasiswa
            WHERE nim  {nim_query} and periode like
'{prev_periode}'
        """
        cursor.execute(query)
        data = cursor.fetchall()
        df_ip = pd.DataFrame(data, columns=['avg_ips',
'avg_ipk'])

        if df_ip.isna().all().all():
            avg_ipk = "-"
            avg_ips = "-"
            count_ulang = "-"
```

```
            count_tidaklulus = "-"
        else:
            avg_ipk = format(df_ip['avg_ipk'][0], '.2f')
            avg_ips = format(df_ip['avg_ips'][0], '.2f')
            count_ulang = get_mhsulangmk(select_periode,
kodemk, nim_query, conn)

            filtered_row = merged_df[(merged_df['Kode MK'] ==
kodemk) & (merged_df['Kelas MK'] == kelasmk)]
            if filtered_row['Jumlah Lulus'].iloc[0] == 0:
                count_tidaklulus = 0
            else:
                count_tidaklulus = int((filtered_row['Jumlah
Mahasiswa'].iloc[0] if not filtered_row.empty else 0) -
(filtered_row['Jumlah Lulus'].iloc[0] if not
filtered_row.empty else 0))
        st.markdown('<br>',True)
        infosummary, infovalue = st.columns(2)
        with infosummary:
            st.write('##### Average IPK')
            st.write('##### Average IPS')
            st.write('##### Mengulang')
            st.write('##### Tidak Lulus')
        with infovalue:
            st.write(f'##### {avg_ipk}')
            st.write(f'##### {avg_ips}')
            st.write(f'##### {len(count_ulang)}')
            st.write(f'##### {count_tidaklulus}')

    with bar_chart:
        nilai_huruf_counts = filtered_data['Nilai
Huruf'].value_counts().reset_index()
        nilai_huruf_counts.columns = ['Nilai Huruf', 'Count']
        nilai_huruf_counts =
nilai_huruf_counts.sort_values(by='Nilai Huruf',
ascending=False)

        color_scale = pc.sequential.deep_r
        if not nilai_huruf_counts.empty:
            fig = px.bar(nilai_huruf_counts, x='Count',
y='Nilai Huruf', orientation='h',
                         color='Nilai Huruf',
color_discrete_sequence=color_scale, text_auto=True)

            fig.update_traces(textposition='outside',
outsidetextfont_size=15)

            fig.update_layout(
                title = 'Distribusi Nilai Mahasiswa di Kelas',
                xaxis_title="Jumlah Mahasiswa",
xaxis_tickfont_size=18,
                yaxis_title="Nilai Huruf",
yaxis_tickfont_size=18,
```

```
                margin=dict(l=0, r=0, t=30, b=0),
                height=250,
                legend=dict(x=1, y=0.5, orientation='v',
font_size=15 ),
            )
        else:
            fig = go.Figure()
            fig.update_layout(
                title = 'Distribusi Nilai Mahasiswa di Kelas',
                xaxis =  { "visible": False },
                yaxis = { "visible": False },
                annotations = [
                    {
                        "text": "No Data Nilai Huruf Found",
                        "showarrow": False,
                        "font": {
                            "size": 28
                        }
                    }
                ],
                margin=dict(l=0, r=0, t=30, b=0),
                height=250,
            )
        bar_chart.plotly_chart(fig)

    st.markdown("<br><br>",True)
    selected_plottype = st.radio("Select Chart Distribution",
["Emperical CDF Plot", "Q-Q Plot"], key='radio_plottype',
horizontal=True)
    chart, table = st.columns(2)
    with chart:
        if selected_plottype == "Emperical CDF Plot":
            df_ecdf = filtered_data.dropna(subset=['Nilai
Simulasi'])
            if len(df_ecdf):
                fig = px.ecdf(title="Emperical Cumulative
Distribution Function", data_frame=df_ecdf, x="Nilai
Simulasi", markers=True)
                fig.update_layout(yaxis=dict(title="Cumulative
Probability"), yaxis_tickformat="", hovermode="y", width= 700,
height= 600, hoverlabel_font_size=15)
                fig.add_shape(
                    dict(
                        type="line",
                        x0=55.5,
                        y0=0,
                        x1=55.5,
                        y1=1,
                        line=dict(
                            color="Red",
                            width=2,
                            dash="dashdot"
                        )
```

```
                    )
                )
            else:
                fig = go.Figure()
                fig.update_layout(
                    xaxis =  { "visible": False },
                    yaxis = { "visible": False },
                    annotations = [
                        {
                            "text": "No Data Nilai Simulasi
Found",
                            "showarrow": False,
                            "font": {
                                "size": 28
                            }
                        }
                    ]
                )
            st.plotly_chart(fig)

        elif selected_plottype == "Q-Q Plot":
            # Extract test scores from the DataFrame
            df_qq = filtered_data
            score = df_qq['Nilai Simulasi']

            fig = go.Figure()
            if len(df_qq.dropna(subset=['Nilai Simulasi'])):
                # Create QQ plot using test scores
                qqplot_data = qqplot(score,
line='s').gca().lines

                fig.add_trace({
                    'type': 'scatter',
                    'x': qqplot_data[0].get_xdata(),
                    'y': qqplot_data[0].get_ydata(),
                    'mode': 'markers',
                    'marker': {
                        'color': 'blue'
                    }
                })

                fig.add_trace({
                    'type': 'scatter',
                    'x': qqplot_data[1].get_xdata(),
                    'y': qqplot_data[1].get_ydata(),
                    'mode': 'lines',
                    'line': {
                        'color': 'red'
                    }

                })

                fig['layout'].update({
```

```
                    'title': 'Quantile-Quantile Plot',
                    'xaxis': {
                        'title': 'Theoretical Quantiles',
                        'zeroline': False
                    },
                    'yaxis': {
                        'title': 'Sample Quantiles'
                    },
                    'showlegend': False,
                    'width': 700,
                    'height': 600,
                })
            else:
                fig.update_layout(
                    xaxis =  { "visible": False },
                    yaxis = { "visible": False },
                    annotations = [
                        {
                            "text": "No Data Nilai Simulasi
Found",
                            "showarrow": False,
                            "font": {
                                "size": 28
                            }
                        }
                    ]
                )
            st.plotly_chart(fig)

    with table:
        sorted_data_mahasiswa =
filtered_data.sort_values(by=['Nilai Simulasi'],
ascending=False).reset_index(drop=True)
        sorted_data_mahasiswa['No'] =
sorted_data_mahasiswa.index + 1
        sorted_data_mahasiswa = sorted_data_mahasiswa[['No',
'NIM', 'Jurusan', 'Angkatan', 'Nilai Angka', 'Nilai Huruf',
'Nilai Simulasi']]

        if not sorted_data_mahasiswa.empty:
            filename = f'{matakuliah}-
{kelasmk}_{select_periode}'
            st.download_button(
                label="Download as CSV",

data=sorted_data_mahasiswa.to_csv(index=False).encode('utf-
8'),
                file_name=f'data_{filename}.csv',
                mime='text/csv',
            )
        gd =
GridOptionsBuilder.from_dataframe(sorted_data_mahasiswa)
        gd.configure_default_column(groupable=True)
```

```
        gd.configure_pagination(paginationAutoPageSize=True)
        gd.configure_grid_options(suppressMovableColumns=True)
        gridoption = gd.build()

        grid_table = AgGrid(sorted_data_mahasiswa,
                            gridOptions=gridoption,
                            fit_columns_on_grid_load=True,
                            height=700,
                            theme= 'streamlit')
```

4.1.7 Mahasiswa Wali

Halaman daftar mahasiswa wali ini adalah tampilan yang dapat diakses oleh dosen untuk melihat informasi lengkap seputar seluruh mahasiswa walinya. Pada halaman ini, terdapat tabel yang menampilkan informasi akademik mahasiswa wali serta grafik distribusi mahasiswa berdasarkan jumlah mata kuliah atau nilai yang tidak lulus pada transkrip, dengan rentang 1-3, 4-7, dan 8+. Baik tabel maupun grafik distribusi ini dapat disesuaikan sesuai dengan filter yang digunakan oleh dosen. Filter yang tersedia meliputi filter angkatan (menyesuaikan dengan angkatan mahasiswa wali), status mahasiswa (dengan pilihan aktif, lulus, non-aktif), filter EPT (dengan pilihan sesuai dengan syarat yudisium program studi; misalnya, Informatika minimum 450 dan Business Management serta Desain Komunikasi Visual minimum 500), filter rentang IPK (yang bisa diatur dari 0.00 hingga 4.00), dan filter mahasiswa dengan nilai D (di mana filter ini akan memberikan warna pada latar belakang tabel untuk mahasiswa yang memiliki dengan mata kuliah tidak lulus, dengan warna latar belakang mengikuti rentang pada grafik distribusi: 1-3 abu-abu, 4-7 oranye, dan 8+ merah tua).

Segmen Program 4.7 List Mahasiswa Wali

```
#Hide Pages
page_prodi = ['Dashboard Program Studi', 'Dashboard Yudisium',
'Dosen Prodi', 'Mahasiswa Prodi', 'Mata Kuliah dan Kelas',
'Prodi_Kelas Ajar', 'Prodi_Mata Kuliah Ajar', 'Prodi_Dashboard
Dosen', 'Prodi_Mahasiswa Wali', 'Yudisium', 'Prodi_Dashboard
Mahasiswa', 'Prodi_KHS-Transkrip']
page_dosen = ['Dosen_Dashboard Mahasiswa', 'Dosen_KHS-
Transkrip']
page_mahasiswa = ['Dashboard Mahasiswa', 'KHS-Transkrip']
hide_pages(page_prodi+page_dosen+page_mahasiswa)

#SQLITE connection
conn = sqlite3.connect('db_skripsi.db')
cursor = conn.cursor()

#Check Prodi State and other Variable
prodi = st.session_state.prodi
dosen = st.session_state.dosen
```

```
nip_dosen = st.session_state.userid

#Get minimal EPT Score
cursor.execute(f"SELECT minept FROM syaratyudisium WHERE
namaunit = '{prodi}' AND periode = (SELECT MAX(periode) FROM
syaratyudisium WHERE namaunit = '{prodi}')")
min_ept = cursor.fetchone()[0]

#Functions
def extract_year(semester):
    return semester[:4]

#Title and Sub-Title
st.write('### Mahasiswa Wali')
st.write(f'#### Dosen: {dosen}')
st.markdown("---")

query = f"""
    SELECT nim, periodemasuk, periodelulus, ipk, statusmhs
    FROM listmahasiswa mhs
    JOIN pegawai pgw ON mhs.nipdosenwali = pgw.nip
    WHERE mhs.namaunit = ? AND pgw.nama = ?
"""
cursor.execute(query, (prodi, dosen))
data = cursor.fetchall()
df = pd.DataFrame(data, columns=['NIM', 'periodemasuk',
'periodelulus', 'IPK', 'Status Mahasiswa'])
df['Angkatan'] = df['periodemasuk'].apply(extract_year)
df_mhs = df[['NIM', 'Angkatan', 'Status Mahasiswa', 'IPK']]

query = f"""
    SELECT nim, totalsks, nilai_ept
    FROM akademikmahasiswa AS a
    WHERE periode = (
        SELECT MAX(periode)
        FROM akademikmahasiswa AS b
        WHERE a.nim = b.nim
    )
"""
cursor.execute(query)
data = cursor.fetchall()
df_akademkikmhs = pd.DataFrame(data, columns=['NIM', 'SKS
Ambil', 'Nilai EPT'])
final_df = pd.merge(df_akademkikmhs, df_mhs, on='NIM',
how='inner')
final_df = final_df[['NIM', 'Angkatan', 'SKS Ambil', 'IPK',
'Status Mahasiswa', 'Nilai EPT']]

mapping = {'A': 'Aktif', 'N': 'Lulus'}
final_df['Status Mahasiswa'] = final_df['Status
Mahasiswa'].replace(mapping)
final_df.loc[~final_df['Status Mahasiswa'].isin(['Aktif',
'Lulus']), 'Status Mahasiswa'] = 'Non Aktif'
```

```
left_col, pad_col, right_col = st.columns((3, 0.05, 2))
with left_col:
    #Filter Atas
    filter_angkatan, filter_statusmhs = st.columns(2)
    with filter_angkatan:
        list_angkatan = list(final_df['Angkatan'].unique())
        list_angkatan.sort(reverse=True)
        selected_angkatan = filter_angkatan.multiselect('Pilih
Angkatan', options=list_angkatan)
    with filter_statusmhs:
        selected_statusmhs = st.multiselect('Pilih Status
Mahasiswa', options=["Aktif", "Lulus", "Non Aktif"],
default=['Aktif'])

    #Filter Bawah
    filter_ept, filter_ipkmin, filter_ipkmax , pad=
st.columns((1,0.5,0.5,1))
    with filter_ept:
        selected_ept = st.multiselect('Pilih Nilai EPT',
options= [f'< {min_ept}', f'{min_ept} and above'])
        switch_color_nilaid = st.toggle("Mahasiswa dengan MK
Tidak Lulus")
    with filter_ipkmin:
        min_ipk = st.number_input('Min IPK:', min_value=0.0,
max_value=4.0, value=0.0)
    with filter_ipkmax:
        max_ipk = st.number_input('Max IPK:', min_value=0.0,
max_value=4.0, value=4.0)

    #Kondisional Filter Angkatan
    if selected_angkatan:
        filtered_df =
final_df[final_df['Angkatan'].isin(selected_angkatan)]
    else:
        filtered_df = final_df

    #Kondisional Status Mahasiswa
    if selected_statusmhs:
        filtered_df = filtered_df[filtered_df['Status
Mahasiswa'].isin(selected_statusmhs)]
    else:
        filtered_df = filtered_df

    #Conditioning Nilai EPT
    if selected_statusmhs:
        if f'< {min_ept}' in selected_ept and f'{min_ept} and
above' in selected_ept:
            pass
        elif f'< {min_ept}' in selected_ept:
            filtered_df = filtered_df[(filtered_df['Nilai
EPT'] < min_ept) | (filtered_df['Nilai EPT'].isna())]
        elif f'{min_ept} and above' in selected_ept:
```

```
            filtered_df = filtered_df[filtered_df['Nilai EPT']
>= min_ept]
    else:
        filtered_df = filtered_df

    filtered_df = filtered_df[(filtered_df['IPK'] >= min_ipk)
& (filtered_df['IPK'] <= max_ipk)]

with right_col:
    list_mahasiswa = tuple(filtered_df['NIM'].unique())
    query = f"""
        SELECT periode, nim, kodemk, namamk, nhuruf
        FROM krsmahasiswa
        WHERE nim IN {list_mahasiswa} AND nhuruf IS NOT NULL
        ORDER BY periode, nim, kodemk ASC
    """
    df_mhs = pd.read_sql_query(query, conn)
    df_mhs = df_mhs.drop_duplicates(subset=['kodemk','nim'],
keep='last').reset_index(drop=True)
    df_mhs['nhuruf'] = df_mhs['nhuruf'].str.strip()

    query = f"""
        SELECT kodemk, wajibpilihan as kurikulum
        FROM kurikulum
        WHERE namaunit = '{prodi}' AND tahun = (SELECT
MAX(tahun) FROM kurikulum WHERE namaunit = '{prodi}')
    """
    df_kurikulum = pd.read_sql(query, conn)
    df_kurikulum['kurikulum'] =
df_kurikulum['kurikulum'].replace({'W': 'Wajib', 'P':
'Pilihan'})

    df_mhs_kurikulum = pd.merge(df_mhs, df_kurikulum,
on='kodemk', how='left')
    df_mhs_kurikulum['kurikulum'] =
df_mhs_kurikulum['kurikulum'].fillna('None')

    filter =
df_mhs_kurikulum[df_mhs_kurikulum['nhuruf'].isin(['D', 'E',
'T'])].reset_index(drop=True)
    grouped_df = filter.groupby('nim').agg(
        mk_tidaklulus=('nhuruf', 'size'),
        list_mk_tdklulus=('kodemk', lambda x: ', '.join(
            sorted(set([f"{row['kodemk']} - {row['namamk']} -
{row['kurikulum']}"
                        for _, row in
df_mhs_kurikulum[df_mhs_kurikulum['kodemk'].isin(x)].iterrows(
)]))
        ))
    ).reset_index()

    #Custom Range and Label untuk Bar Chart
    range_mhs = [(1, 3), (4, 7), (8, float('inf'))]
```

```
    labels = [f"{r[0]}-{r[1]}" if r[1] != float('inf') else
f"{r[0]}++" for r in range_mhs]

    bins = [r[0] - 0.5 for r in range_mhs] + [float('inf')]
    grouped_df['nim_bin'] =
pd.cut(grouped_df['mk_tidaklulus'], bins=bins, labels=labels,
right=False)
    bar_data =
grouped_df['nim_bin'].value_counts().reset_index()
    bar_data.columns = ['Range', 'Count']

    color_mapping = {
        '1-3': '#999999',
        '4-7': '#ff5252',
        '8++': '#a70000'
    }
    fig = go.Figure()

    for idx, row in bar_data.iterrows():
        fig.add_trace(go.Bar(
            x=[row['Range']],
            y=[row['Count']],
            text=[row['Count']],
            name=row['Range'],
            marker_color=color_mapping.get(row['Range'],
'lightgrey')  # Assign color based on the range
        ))

    fig.update_traces(textposition='outside',
outsidetextfont_size=15, hovertemplate ="<b>Range: %{x}</b>
<br> Jumlah: %{y}",)

    fig.update_layout(
        title='Distribusi Mahasiswa berdasarkan Jumlah Mata
Kuliah Tidak Lulus',
        xaxis_tickfont_size=17, xaxis_title='Range',
        yaxis_tickfont_size=16, yaxis_title='Banyak
Mahasiswa',
        margin=dict(t=30, l=0, r=0, b=0),
        legend_title='Range MK Tidak Lulus',
        legend=dict(font=dict(size=15))
    )
    st.plotly_chart(fig)

with left_col:
    filtered_df = filtered_df.merge(grouped_df[['nim',
'nim_bin', 'list_mk_tdklulus']], how='left', left_on='NIM',
right_on='nim')
    filtered_df = filtered_df.drop('nim', axis=1)

    color_mapping_json = json.dumps(color_mapping)

    cell_style_jscode = JsCode(f"""
```

```
        function(params) {{
            var colorMapping = {color_mapping_json};
            var nimBin = params.data.nim_bin;
            var color = colorMapping[nimBin];
            if (color) {{
                return {{
                    'background-color': color,
                    'color': 'black'
                }};
            }}
            return {{}};
        }}
    """)

    # Configure ag-Grid options
    gb = GridOptionsBuilder.from_dataframe(filtered_df)
    gb.configure_selection(selection_mode='single',
use_checkbox=True)
    gb.configure_pagination(paginationAutoPageSize=True)
    gb.configure_grid_options(suppressMovableColumns=True)
    gb.configure_column('nim_bin', hide=True)
    gb.configure_column('list_mk_tdklulus', hide=True)


    if switch_color_nilaid:

gb.configure_default_column(cellStyle=cell_style_jscode)

    gridoption = gb.build()

    grid_table = AgGrid(filtered_df,
                        gridOptions=gridoption,
                        fit_columns_on_grid_load=True,
                        height=500,
                        theme='streamlit',
                        allow_unsafe_jscode=True)

    selected_row = grid_table['selected_rows']
    if selected_row is not None:
        temp_df = filtered_df[filtered_df['NIM'] ==
selected_row['NIM'][0]]
        with st.container(border=True):

            st.write("### Detail Mahasiswa Wali",
anchor=False)
            col1,col2 = st.columns(2)
            col1.write(f"**NIM:** {selected_row['NIM'][0]}")
            col1.write(f"**Angkatan:**
{selected_row['Angkatan'][0]}")
            col1.write(f"**Total SKS Ambil:**
{selected_row['SKS Ambil'][0]}")
            col1.write(f"**IPK:** {selected_row['IPK'][0]}")
```

```
            col2.write(f"**Dosen Wali:** {dosen}")
            col2.write(f"**Status Mahasiswa:**
{selected_row['Status Mahasiswa'][0]}")
            col2.write(f"**Nilai EPT:** {selected_row['Nilai
EPT'][0]}")

            if not temp_df['list_mk_tdklulus'].iloc[0] is
np.nan:
                concat_list = [item.strip() for sublist in
temp_df['list_mk_tdklulus'].str.split(", ") for item in
sublist]
                if concat_list:
                    concatenated_str = ', '.join(concat_list)

                    with st.expander("**List MK Tidak
Lulus**"):
                        for i, item in enumerate(concat_list,
start=1):
                            st.write(f'{i}. {item}')

        nim = selected_row["NIM"][0]
        but1,but2 = st.columns(2)
        but1.link_button("Dashboard Mahasiswa",
url=f'/Dosen_Dashboard%20Mahasiswa?prodi={prodi}&nim={nim}',us
e_container_width=True)
        but2.link_button("KHS dan Transkrip",
url=f'/Dosen_KHS-
Transkrip?prodi={prodi}&nim={nim}',use_container_width=True,
type='primary')

if not filtered_df.empty:
    download_df = filtered_df.drop('nim_bin', axis=1)
    right_col.download_button(
        label="Download as CSV",
        data=download_df.to_csv(index=False).encode('utf-8'),
        file_name=f'Mahasiswa_Wali_{dosen}.csv',
        mime='text/csv',
    )
```

4.1.8 Dashboard Program Studi

Halaman dashboard ini dapat diakses oleh pengguna dari program studi dengan tujuan untuk memberikan visualisasi data progres akademik mahasiswa dalam satu program studi selama satu periode. Pada dashboard ini, pengguna dapat memilih periode yang ingin diperiksa dan melihat detail pengelompokan berdasarkan rentang IPK, IPS, maupun SKS, serta jumlah mahasiswa yang aktif, total mahasiswa yang telah yudisium, dan kelas mata kuliah yang dibuka selama periode tersebut. Fitur-fitur ini memungkinkan program studi untuk memantau dan menganalisis perkembangan akademik mahasiswa secara terperinci, yang dapat membantu dalam pengambilan keputusan dan penilaian kinerja mahasiswa secara keseluruhan.

Segmen Program 4.8 Halaman Dashboard Program Studi

```
#Hide Pages
page_prodi = ['Prodi_Kelas Ajar', 'Prodi_Mata Kuliah Ajar',
'Prodi_Dashboard Dosen', 'Prodi_Mahasiswa Wali',
'Prodi_Dashboard Mahasiswa', 'Prodi_KHS-Transkrip']
page_dosen = ['Dashboard Dosen', 'Mahasiswa Wali', 'Mata
Kuliah Ajar', 'Dosen_Dashboard Mahasiswa', 'Dosen_KHS-
Transkrip']
page_mahasiswa = ['Dashboard Mahasiswa', 'KHS-Transkrip']
hide_pages(page_prodi+page_dosen+page_mahasiswa)

#SQLITE connection
conn = sqlite3.connect('db_skripsi.db')
cursor = conn.cursor()

#Check Variable
prodi = st.session_state['prodi']

#Functions
def extract_year(semester):
    return semester[:4]

def get_periode_from_list(list_periode, periodeawal,
num_periods=5):
    start_index = list_periode.index(periodeawal)

    result = list_periode[start_index:start_index +
num_periods]
    return result

with st.sidebar:
    #Get periode list
    cursor.execute(f"SELECT DISTINCT periode FROM kelas WHERE
namaunit like '{prodi}' ORDER BY periode DESC")
    list_periode = [str(periode[0]) for periode in
cursor.fetchall()]
```

```
    select_periode = st.sidebar.selectbox(label="Pilih
Periode", options=list_periode)

#Title
st.markdown(f'### Dashboard Program Studi - {prodi}')
st.markdown(f'#### Periode: {select_periode}')
st.markdown("---")

if select_periode:
    #Dashboard Bagian Atas
    topleft_col, topmid_col, topright_col = st.columns(3)
    with topleft_col:
        st.subheader("Jumlah Mhs. Aktif", anchor=False)
        latest_periode = list_periode[0]
        if select_periode == latest_periode:
            cursor.execute('SELECT COUNT(DISTINCT nim) FROM
listmahasiswa WHERE statusmhs = "A" AND namaunit = ?',
(prodi,))
            total_mhs = cursor.fetchone()[0]
        else:
            cursor.execute('SELECT COUNT(DISTINCT nim) FROM
akademikmahasiswa WHERE periode = ? AND namaunit = ?',
(select_periode, prodi))
            total_mhs = cursor.fetchone()[0]
        st.title(total_mhs, anchor=False)

    with topmid_col:
        st.subheader("Total Yudisium", anchor=False)

        cursor.execute('SELECT COUNT(*) FROM listlulusan WHERE
periodelulus = ? AND namaunit = ?', (select_periode, prodi))
        total_yudisium = cursor.fetchone()[0]
        st.title(total_yudisium, anchor=False)

    with topright_col:
        st.subheader("Kelas Mata Kuliah Yang Dibuka",
anchor=False)

        cursor.execute('SELECT COUNT(*) FROM kelas WHERE
periode = ? AND namaunit = ? AND kodeunit IS NOT NULL',
(select_periode, prodi))
        total_kelas = cursor.fetchone()[0]
        st.title(total_kelas, anchor=False)

    #Dashboard Bagian Bawah
    cursor.execute('SELECT periode, nim, periodemasuk,
totalsks, ips, ipk FROM akademikmahasiswa WHERE periode = ?
AND periode != ? AND namaunit = ?', (select_periode,
latest_periode, prodi))
    data = cursor.fetchall()
    df = pd.DataFrame(data, columns=['periode', 'nim',
'angkatan', 'totalsks','ips', 'ipk'])
    df['angkatan'] = df['angkatan'].apply(extract_year)
```

```
    st.markdown('<br>',True)
    bottomleft_col, bottommid_col, bottomright_col =
st.columns(3)
    with bottomleft_col:
        st.subheader("Range IPK Keseluruhan", anchor=False)
        visualization_type = st.radio("Select Visualization
Type", ["Histogram", "Stack", "Group"], key="radio_ipk",
horizontal=True)

        if len(df):
            ipk_ranges = [(0.00, 1.00), (1.01, 1.50), (1.51,
2.00), (2.01, 2.50), (2.51, 3.00), (3.01, 3.50), (3.51, 4.00)]
            bins =
pd.IntervalIndex.from_tuples([*ipk_ranges[:-1], (ipk_ranges[-
1][0], ipk_ranges[-1][1])], closed='both')
            labels = [f"{start:.2f}-{end:.2f}" for start, end
in ipk_ranges]
            df['Range IPK'] = pd.cut(df['ipk'], bins=bins,
labels=labels)

            if visualization_type == "Histogram":
                grouped_df_ipk = df.groupby(['Range
IPK']).size().reset_index(name='Count')
                grouped_df_ipk['Range IPK'] =
grouped_df_ipk['Range IPK'].astype(str)
                fig_ipk = px.bar(grouped_df_ipk, x='Range
IPK', y='Count', labels={'Count': 'Banyak Mahasiswa'},
text_auto=True)
                fig_ipk.update_traces(textposition='outside',
textfont_size=15)
                fig_ipk.update_layout(margin=dict(t=25, b=50),
hoverlabel=dict(font_size=15))
                fig_ipk.update_xaxes(
                    tickvals=grouped_df_ipk['Range IPK'],
ticktext=labels, tickfont=dict(size=15)
                )
            else:
                grouped_df_ipk = df.groupby(['Range IPK',
'angkatan']).size().reset_index(name='Count')
                grouped_df_ipk['Range IPK'] =
grouped_df_ipk['Range IPK'].astype(str)
                fig_ipk = px.bar(grouped_df_ipk, x='Range
IPK', y='Count', color='angkatan',
barmode=visualization_type.lower(), labels={'Count': 'Banyak
Mahasiswa', 'angkatan': 'Angkatan'}, text_auto=True)
                fig_ipk.update_xaxes(
                    tickvals=grouped_df_ipk['Range
IPK'].unique(), ticktext=labels, tickfont=dict(size=15),
                )
                fig_ipk.update_layout(margin=dict(t=25, b=50),
hoverlabel=dict(font_size=15),
```

```
legend=dict(orientation="h", yanchor="bottom", y=1.02,
xanchor="right", x=1, font_size=13),
                                        )
                if visualization_type == "Stack":

fig_ipk.update_traces(textposition='inside', textangle=0)

fig_ipk.update_layout(uniformtext_minsize=15,
uniformtext_mode='hide',)
                elif visualization_type == "Group":

fig_ipk.update_traces(textposition='outside',
textfont=dict(size=15))
        else:
            fig_ipk = go.Figure()
            fig_ipk.update_layout(
                xaxis =  { "visible": False },
                yaxis = { "visible": False },
                annotations = [
                    {
                        "text": "No Data Found",
                        "showarrow": False,
                        "font": {
                            "size": 28
                        }
                    }
                ]
            )
        st.plotly_chart(fig_ipk, True)

    with bottommid_col:
        st.subheader("Range IPS Keseluruhan", anchor=False)
        visualization_type = st.radio("Select Visualization
Type", ["Histogram", "Stack", "Group"], key="radio_ips",
horizontal=True)

        if len(df):
            ips_range = [(0.00, 1.00), (1.01, 1.50), (1.51,
2.00), (2.01, 2.50), (2.51, 3.00), (3.01, 3.50), (3.51, 4.00)]
            bins = pd.IntervalIndex.from_tuples([*ips_range[:-
1], (ips_range[-1][0], ips_range[-1][1])], closed='both')
            labels = [f"{start:.2f}-{end:.2f}" for start, end
in ips_range]
            df['Range IPS'] = pd.cut(df['ips'], bins=bins,
labels=labels)

            if visualization_type == "Histogram":
                grouped_df_ips = df.groupby(['Range
IPS']).size().reset_index(name='Count')
                grouped_df_ips['Range IPS'] =
grouped_df_ips['Range IPS'].astype(str)
```

```
                fig_ips = px.bar(grouped_df_ips, x='Range
IPS', y='Count', labels={'Count': 'Banyak Mahasiswa'},
text_auto=True)
                fig_ips.update_traces(textposition='outside',
textfont_size=15)
                fig_ips.update_layout(margin=dict(t=25, b=50),
hoverlabel=dict(font_size=15))
                fig_ips.update_xaxes(
                    tickvals=grouped_df_ips['Range IPS'],
ticktext=labels, tickfont=dict(size=15)
                )
            else:
                grouped_df_ips = df.groupby(['Range IPS',
'angkatan']).size().reset_index(name='Count')
                grouped_df_ips['Range IPS'] =
grouped_df_ips['Range IPS'].astype(str)
                fig_ips = px.bar(grouped_df_ips, x='Range
IPS', y='Count', color='angkatan',
barmode=visualization_type.lower(), labels={'Count': 'Banyak
Mahasiswa', 'angkatan': 'Angkatan'}, text_auto=True)
                fig_ips.update_xaxes(
                    tickvals=grouped_df_ips['Range
IPS'].unique(), ticktext=labels, tickfont=dict(size=15),
                )
                fig_ips.update_layout(margin=dict(t=25, b=50),
hoverlabel=dict(font_size=15),

legend=dict(orientation="h", yanchor="bottom", y=1.02,
xanchor="right", x=1, font_size=13),
                                      )
                if visualization_type == "Stack":

fig_ips.update_traces(textposition='inside', textangle=0)

fig_ips.update_layout(uniformtext_minsize=15,
uniformtext_mode='hide',)
                elif visualization_type == "Group":

fig_ips.update_traces(textposition='outside',
textfont=dict(size=15))

        else:
            fig_ips = go.Figure()
            fig_ips.update_layout(
                xaxis =  { "visible": False },
                yaxis = { "visible": False },
                annotations = [
                    {
                        "text": "No Data Found",
                        "showarrow": False,
                        "font": {
                            "size": 28
                        }
```

```
                    }
                ]
            )
        st.plotly_chart(fig_ips, True)

    with bottomright_col:
        st.subheader("Range Penyelesaian SKS", anchor=False)
        visualization_type = st.radio("Select Visualization
Type", ["Histogram", "Stack", "Group"], key="radio_sks",
horizontal=True)

        if len(df):
            sks_ranges = [(0, 19), (20, 39), (40, 59), (60,
79), (80, 99), (100, 119), (120, 139), (140, float('inf'))]
            bins =
pd.IntervalIndex.from_tuples([*sks_ranges[:-1], (sks_ranges[-
1][0], sks_ranges[-1][1])], closed='both')
            labels = [f"{start} - {end}" for start, end in
sks_ranges[:-1]] + ["140+"]
            df['Range SKS'] = pd.cut(df['totalsks'],
bins=bins, labels=labels)

            if visualization_type == "Histogram":
                grouped_df_sks = df.groupby(['Range
SKS']).size().reset_index(name='Count')
                grouped_df_sks['Range SKS'] =
grouped_df_sks['Range SKS'].astype(str)
                fig_sks = px.bar(grouped_df_sks, x='Range
SKS', y='Count', labels={'Count': 'Banyak Mahasiswa'},
text_auto=True)
                fig_sks.update_traces(textposition='outside',
textfont_size=15)
                fig_sks.update_layout(margin=dict(t=25, b=50),
hoverlabel=dict(font_size=15))
                fig_sks.update_xaxes(
                    tickvals=grouped_df_sks['Range SKS'],
ticktext=labels, tickfont=dict(size=15)
                )
            else:
                grouped_df_sks = df.groupby(['Range SKS',
'angkatan']).size().reset_index(name='Count')
                grouped_df_sks['Range SKS'] =
grouped_df_sks['Range SKS'].astype(str)
                fig_sks = px.bar(grouped_df_sks, x='Range
SKS', y='Count', color='angkatan',
barmode=visualization_type.lower(), labels={'Count': 'Banyak
Mahasiswa', 'angkatan': 'Angkatan'}, text_auto=True)
                fig_sks.update_xaxes(
                    tickvals=grouped_df_sks['Range
SKS'].unique(), ticktext=labels, tickfont=dict(size=15),
                )
                fig_sks.update_layout(margin=dict(t=25, b=50),
hoverlabel=dict(font_size=15),
```

```
legend=dict(orientation="h", yanchor="bottom", y=1.02,
xanchor="right", x=1, font_size=13),
                                        )
                if visualization_type == "Stack":

fig_sks.update_traces(textposition='inside', textangle=0)

fig_sks.update_layout(uniformtext_minsize=18,
uniformtext_mode='hide',)
                elif visualization_type == "Group":

fig_sks.update_traces(textposition='outside',
textfont=dict(size=15))
        else:
            fig_sks = go.Figure()
            fig_sks.update_layout(
                xaxis =  { "visible": False },
                yaxis = { "visible": False },
                annotations = [
                    {
                        "text": "No Data Found",
                        "showarrow": False,
                        "font": {
                            "size": 28
                        }
                    }
                ]
            )
        st.plotly_chart(fig_sks, True)

    periods = get_periode_from_list(list_periode,
select_periode)
    if len(periods) == 1:
        query = f"SELECT nim, periode, ips, periodemasuk FROM
akademikmahasiswa WHERE periode = '{periods[0]}' AND namaunit
LIKE '{prodi}'"
    else:
        query = f"SELECT nim, periode, ips, periodemasuk FROM
akademikmahasiswa WHERE periode IN {tuple(periods)} AND
namaunit LIKE '{prodi}'"
    df = pd.read_sql_query(query, conn)
    df['angkatan'] = df['periodemasuk'].apply(extract_year)

    grouped_df = df.groupby(['angkatan',
'periode']).agg({'ips': 'mean'}).reset_index()
    grouped_df['ips'] = grouped_df['ips'].round(2)
    grouped_df = grouped_df.rename(columns={'angkatan':
'Angkatan', 'periode': 'Periode', 'ips': 'Average IPS'})

    all_angkatan_avg = grouped_df.groupby('Periode')['Average
IPS'].mean().reset_index()
    all_angkatan_avg['Angkatan'] = 'ALL'
```

```
    df_with_all = pd.concat([grouped_df, all_angkatan_avg],
ignore_index=True)

    order_periods = periods
    pivot_df = df_with_all.pivot(index='Angkatan',
columns='Periode', values='Average IPS')
    pivot_df = pivot_df[order_periods]
    pivot_df = pivot_df.reindex(['ALL'] + [x for x in
pivot_df.index if x != 'ALL'])
    pivot_df = pivot_df.reset_index()
    pivot_df = pivot_df.fillna('')

    max_row_index = len(pivot_df) - 1
    max_col_index = len(pivot_df.columns)-1

    def highlight_cells(x):
        color = 'background-color: yellow'
        df_styler = pd.DataFrame('', index=x.index,
columns=x.columns)
        df_styler.iloc[max_row_index, 1] = color
        if max_col_index >= 3:
            df_styler.iloc[max_row_index-1, 3] = color
        if max_col_index >= 5:
            df_styler.iloc[max_row_index-2, 5] = color  # Cell
(3, 5)
        return df_styler

    styled_df = pivot_df.style.apply(highlight_cells,
axis=None).format(precision=2)
    all_top_value = pivot_df.loc[0][1]
    if all_top_value != 0:
        st.markdown('<br>',True)
        st.markdown(f""" <div style='text-align: center;'>
<h2>Rata-Rata IPS Per Angkatan - Periode {select_periode}</h2>
</div> """, unsafe_allow_html=True)
        st.table(styled_df)
```

4.1.9 Mahasiswa Program Studi

Halaman mahasiswa program studi ini adalah tampilan yang dapat dilihat oleh *user* program studi, untuk melihat informasi seputar seluruh mahasiswa dalam satu program studi. Pada halaman ini terdapat tabel dengan informasi akademik mahasiswa dan juga *chart* distribusi mahasiswa berdasarkan angkatannya. Dimana kedua tabel dan *chart* distribusi dapat berubah sesuai dengan filter yang digunakan oleh *user*, yaitu filter angkatan dimana pilihan angkatan dimulai dari 2018 hingga 2023, filter pilihan dosen wali, status mahasiswa dengan pilihan aktif, lulus, non aktif, lalu filter EPT dengan pilihan sesuai dengan syarat yudisium program studi (informatika minimal 450 dan *business management* dan desain komunikasi visual minimal 500), *filter range* ipk yang bisa diatur dari 0.00 – 4.00. *User* dapat memilih satu mahasiswa untuk

melihat detail mahasiswa dibawah tabel dimana *user* atau program studi dapat melihat dashboard mahasiswa serta khs dan transkrip mahasiswa dengan menekan salah satu tombol pada detail mahasiswa.

Segmen Program 4.9 List Mahasiswa Program Studi

```
#Hide Pages
page_prodi = ['Prodi_Kelas Ajar', 'Prodi_Mata Kuliah Ajar',
'Prodi_Dashboard Dosen', 'Prodi_Mahasiswa Wali',
'Prodi_Dashboard Mahasiswa', 'Prodi_KHS-Transkrip']
page_dosen = ['Dashboard Dosen', 'Mahasiswa Wali', 'Mata Kuliah
Ajar', 'Dosen_Dashboard Mahasiswa', 'Dosen_KHS-Transkrip']
page_mahasiswa = ['Dashboard Mahasiswa', 'KHS-Transkrip']
hide_pages(page_prodi+page_dosen+page_mahasiswa)

#SQLITE connection
conn = sqlite3.connect('db_skripsi.db')
cursor = conn.cursor()

#Check Prodi State
prodi = st.session_state['prodi']

#Get minimal EPT Score
cursor.execute(f"SELECT minept FROM syaratyudisium WHERE
namaunit = '{prodi}' AND periode = (SELECT MAX(periode) FROM
syaratyudisium WHERE namaunit = '{prodi}')")
min_ept = cursor.fetchone()[0]

#Functions
def extract_year(semester):
    return semester[:4]

#Title
st.markdown(f'### Mahasiswa Prodi - {prodi}')
st.markdown("---")

cursor.execute("SELECT nim, periodemasuk, periodelulus, ipk,
statusmhs, nipdosenwali FROM listmahasiswa WHERE namaunit = ?",
(prodi,))
data = cursor.fetchall()
df = pd.DataFrame(data,
columns=['NIM','periodemasuk','periodelulus','IPK','Status
Mahasiswa','nip'])
# df['Masa Studi'] = df.apply(hitung_masa_studi, axis=1)
df['Angkatan'] =
df['periodemasuk'].apply(extract_year).astype(str)

mapping = {'A': 'Aktif', 'N': 'Lulus'}
df['Status Mahasiswa'] = df['Status
Mahasiswa'].replace(mapping)
```

```
df.loc[~df['Status    Mahasiswa'].isin(['Aktif',    'Lulus']),
'Status Mahasiswa'] = 'Non Aktif'

cursor.execute("SELECT  nim,  nilai_ept  FROM  detailmahasiswa
WHERE namaunit = ?",(prodi,))
data = cursor.fetchall()
df_ept = pd.DataFrame(data, columns=['NIM', 'Nilai EPT'])

cursor.execute("SELECT nip, nama FROM pegawai WHERE namaunit =
?",(prodi,))
data = cursor.fetchall()
df_dosen = pd.DataFrame(data, columns=['nip', 'Dosen Wali'])

df_merge = pd.merge(df, df_ept, on='NIM', how='inner')
df_merge = pd.merge(df_merge, df_dosen, on='nip', how='inner')
final_df  =  df_merge[['NIM',  'Angkatan',  'IPK',  'Nilai  EPT',
'Status Mahasiswa', 'Dosen Wali']]

left_col, pad_col, right_col = st.columns((3, 0.05, 2))
with left_col:
    #Filter Atas
    filter_angkatan, filter_dosen = st.columns(2)
    with filter_angkatan:
        list_angkatan = list(df['Angkatan'].unique())
        list_angkatan.sort(reverse=True)
        selected_angkatan  =  filter_angkatan.multiselect('Pilih
Angkatan', options=list_angkatan)

    with filter_dosen:
        list_dosenwali = list(final_df['Dosen Wali'].unique())
        list_dosenwali.sort()
        selected_dosen  =  st.multiselect('Pilih  Dosen  Wali',
options=list_dosenwali)

    #Filter Bawah
    filter_statusmhs, filter_ept, filter_ipkmin, filter_ipkmax
= st.columns((1,1,0.5,0.5))
    with filter_statusmhs:
        selected_statusmhs   =   st.multiselect('Pilih   Status
Mahasiswa',   options=["Aktif",   "Lulus",   "Non   Aktif"],
default=['Aktif'])
    with filter_ept:
        selected_ept   =   st.multiselect('Pilih   Nilai   EPT',
options= [f'< {min_ept}', f'{min_ept} and above'])
    with filter_ipkmin:
        min_ipk  =  st.number_input('Min  IPK:',  min_value=0.0,
max_value=4.0, value=0.0)
    with filter_ipkmax:
        max_ipk  =  st.number_input('Max  IPK:',  min_value=0.0,
max_value=4.0, value=4.0)

    #Conditioning Filter Angkatan dan Dosen
    if selected_angkatan and selected_dosen:
```

```
        filtered_df                                                    =
final_df[final_df['Angkatan'].isin(selected_angkatan)          &
final_df['Dosen Wali'].isin(selected_dosen)]
    elif selected_angkatan :
        filtered_df                                                    =
final_df[final_df['Angkatan'].isin(selected_angkatan)]
    elif selected_dosen:
        filtered_df          =          final_df[final_df['Dosen
Wali'].isin(selected_dosen)]
    else:
        filtered_df = final_df

    #Conditioning Status Mahasiswa
    if selected_statusmhs:
        filtered_df      =      filtered_df[filtered_df['Status
Mahasiswa'].isin(selected_statusmhs)]
    else:
        filtered_df = filtered_df

    filtered_df = filtered_df[(filtered_df['IPK'] >= min_ipk) &
(filtered_df['IPK'] <= max_ipk)]

   #Conditioning Nilai EPT
    if selected_statusmhs:
        if f'< {min_ept}' in selected_ept and f'{min_ept} and
above' in selected_ept:
            pass
        elif f'< {min_ept}' in selected_ept:
            filtered_df = filtered_df[(filtered_df['Nilai EPT']
< min_ept) | (filtered_df['Nilai EPT'].isna())]
        elif f'{min_ept} and above' in selected_ept:
            filtered_df = filtered_df[filtered_df['Nilai EPT']
>= min_ept]
    else:
        filtered_df = filtered_df

    unique_nim = filtered_df['NIM'].unique()
    selected_nim   =   st.multiselect('Pilih   NIM   Mahasiswa',
options=unique_nim)
    if selected_nim:
        filtered_df                                                    =
filtered_df[filtered_df['NIM'].isin(selected_nim)]
    else:
        filtered_df = filtered_df

    gd = GridOptionsBuilder.from_dataframe(filtered_df)
    gd.configure_default_column(groupable=True)
    gd.configure_selection(selection_mode='single',
use_checkbox=True)
    gd.configure_pagination(paginationAutoPageSize=True)
    gd.configure_grid_options(suppressMovableColumns=True)
    gridoption = gd.build()
```

```
    grid_table = AgGrid(filtered_df,
                        gridOptions=gridoption,
                        fit_columns_on_grid_load=True,
                        height=500,
                        theme= 'streamlit')

    selected_row = grid_table['selected_rows']
    if selected_row is not None:
        with st.container(border=True):
            st.subheader("Student Details")
            col1,col2 = st.columns(2)
            col1.write(f"**NIM:** {selected_row['NIM'][0]}")
            col1.write(f"**Angkatan:**
{selected_row['Angkatan'][0]}")
            col1.write(f"**IPK:** {selected_row['IPK'][0]}")
            col1.write(f"**Nilai  EPT:**  {selected_row['Nilai
EPT'][0]}")

            col2.write(f"**Dosen  Wali:**  {selected_row['Dosen
Wali'][0]}")
            col2.write(f"**Status                  Mahasiswa:**
{selected_row['Status Mahasiswa'][0]}")
            col2.markdown('<font       color="white">Invisible
Text</font>', unsafe_allow_html=True)
            col2.markdown('<font       color="white">Invisible
Text</font>', unsafe_allow_html=True)

        # Buttons for Dashboard and Transkrip
        col1.link_button("Dashboard                 Mahasiswa",
url=f'/Prodi_Dashboard%20Mahasiswa?prodi={prodi}&nim={selected
_row["NIM"][0]}', use_container_width=True)
        col2.link_button("KHS dan Transkrip", url=f'/Prodi_KHS-
Transkrip?prodi={prodi}&nim={selected_row["NIM"][0]}',use_cont
ainer_width=True, type='primary')

with right_col:
    sunburst_df  =  filtered_df.groupby(['Status  Mahasiswa',
'Angkatan']).size().reset_index(name='Jumlah Mahasiswa')
    fig = px.sunburst(
        title= "Distribusi Mahasiswa per Angkatan",
        data_frame=sunburst_df,
        path=["Status Mahasiswa", 'Angkatan'],
        values='Jumlah Mahasiswa',
        custom_data=['Status Mahasiswa','Jumlah Mahasiswa']
    )
    # fig.update_traces(textinfo='label+percententry')
    fig.update_layout(
        margin = dict(t=50, l=0, r=0, b=10),
        hoverlabel=dict(font_size=18)
        )
    fig.update_traces(
        hovertemplate  ="<b>%{customdata[0]}</b>  <br>  Jumlah:
%{customdata[1]}",
```

```
        insidetextorientation='radial',
        insidetextfont_size=25,
    )

    st.plotly_chart(fig, use_container_width=True)
    st.write('')

    if not filtered_df.empty:
        st.download_button(
            label="Download as CSV",
            data=filtered_df.to_csv(index=False).encode('utf-
8'),
            file_name=f'list_mahasiswa_{prodi}.csv',
            mime='text/csv',
        )
```

4.1.10 Dosen Program Studi

Halaman dosen program studi ini adalah tampilan yang dapat diakses oleh *user* program studi, untuk melihat informasi seputar seluruh dosen dalam satu program studi. Pada halaman ini terdapat tabel dengan informasi dosen, dimana tabel berubah sesuai dengan filter yang digunakan oleh *user*, yaitu *filter* aktif mengajar dengan pilihan aktif dan non aktif, *filter* mahasiswa wali dengan pilihan ada dan tidak ada, *filter search* nama dosen atau pilih nama dosen. *User* dapat memilih satu data dosen untuk melihat detail dosen dibawah tabel dimana *user* atau program studi dapat melihat halaman mata kuliah ajaran dengan menekan tombol mata kuliah ajaran, dan jika dosen yang dipilih mempunyai mahasiswa wali maka akan ada tombol untuk melihat *dashboard* dosen wali dan list mahasiswa wali.

Segmen Program 4.10 List Dosen Program Studi

```
#Hide Pages
page_prodi = ['Prodi_Kelas Ajar', 'Prodi_Mata Kuliah Ajar',
'Prodi_Dashboard Dosen', 'Prodi_Mahasiswa Wali',
'Prodi_Dashboard Mahasiswa', 'Prodi_KHS-Transkrip']
page_dosen = ['Dashboard Dosen', 'Mahasiswa Wali', 'Mata Kuliah
Ajar', 'Dosen_Dashboard Mahasiswa', 'Dosen_KHS-Transkrip']
page_mahasiswa = ['Dashboard Mahasiswa', 'KHS-Transkrip']
hide_pages(page_prodi+page_dosen+page_mahasiswa)


#SQLITE connection
conn = sqlite3.connect('db_skripsi.db')
cursor = conn.cursor()

#Check Prodi State
prodi = st.session_state['prodi']
```

```
#Get latest periode pengajar
cursor.execute("SELECT MAX(periode) FROM pengajar WHERE
namaunit = ?", (prodi,))
periode = cursor.fetchone()[0]

#Title
st.markdown(f'### Dosen - {prodi}')
st.markdown(f'#### Periode: {periode}')
st.markdown("---")

#Ambil Data Dosen Ajar + Dosen Wali
query = f"""
    SELECT
        rp.nip as NIP,
        p.nama as 'Nama Dosen',
        COALESCE(rp2.jumlah_kodemk, 0) AS 'Status Mengajar'
    FROM
        (SELECT nip FROM pegawai WHERE namaunit like '{prodi}'
GROUP BY nip) AS rp
    LEFT JOIN
        (SELECT nip, COUNT(DISTINCT kodemk) AS jumlah_kodemk
        FROM pengajar
        WHERE periode = ?
        GROUP BY nip) AS rp2
    ON rp.nip = rp2.nip
    LEFT JOIN pegawai p ON rp.nip = p.nip
    ORDER BY p.nama asc;
"""
df_dosenajar = pd.read_sql_query(query, conn,
params=(periode,))
query = f"""
    SELECT nipdosenwali as NIP, COUNT(nim) AS 'Mahasiswa Wali'
    FROM listmahasiswa
    WHERE namaunit = ? AND statusmhs like 'A'
    GROUP BY nipdosenwali;
"""
df_dosenwali = pd.read_sql_query(query, conn, params=(prodi,))

final_df = pd.merge(df_dosenajar, df_dosenwali, on='NIP',
how='left')
final_df['Status Mengajar'] = final_df['Status
Mengajar'].apply(lambda x: 'Aktif' if x > 0 else 'Non Aktif')
final_df['Mahasiswa Wali'] = final_df['Mahasiswa
Wali'].fillna(0)

left, right = st.columns(2)
with left:
    filter_kiri, filter_kanan, pad = st.columns((1,1,1))
    with filter_kiri:
        aktif_ajar_option = st.multiselect("Aktif Mengajar:",
options=['Aktif', 'Non Aktif'], default='Aktif')
    with filter_kanan:
```

```
        mahasiswa_wali_options   =   st.multiselect("Mahasiswa
Wali:", options=['Ada', 'Tidak Ada'])

    if aktif_ajar_option:
        filtered_df         =         final_df[final_df['Status
Mengajar'].isin(aktif_ajar_option)]
    else:
        filtered_df = final_df

    if  'Ada'  in  mahasiswa_wali_options  and  'Tidak  Ada'  in
mahasiswa_wali_options:
        pass
    elif mahasiswa_wali_options:
        if 'Ada' in mahasiswa_wali_options:
            filtered_df  =  filtered_df[filtered_df['Mahasiswa
Wali'] != 0]

        if 'Tidak Ada' in mahasiswa_wali_options:
            filtered_df  =  filtered_df[filtered_df['Mahasiswa
Wali'] == 0]

    selected_dosen   =   st.multiselect('Pilih   Nama   Dosen:',
options=filtered_df['Nama Dosen'].unique())
    if selected_dosen:
        filtered_df       =       filtered_df[filtered_df['Nama
Dosen'].isin(selected_dosen)]
    else: filtered_df = filtered_df
with right:
    if not filtered_df.empty:
        add_vertical_space(7)
        st.download_button(
            label="Download as CSV",
            data=filtered_df.to_csv(index=False).encode('utf-
8'),
            file_name=f'Data_Dosen_{prodi}_{periode}.csv',
            mime='text/csv',
        )

gb = GridOptionsBuilder.from_dataframe(filtered_df)
gb.configure_default_column(groupable=True)
gb.configure_selection(selection_mode='single',
use_checkbox=True)
gb.configure_pagination(paginationAutoPageSize=True)
gb.configure_grid_options(suppressMovableColumns=True)
gridoption = gb.build()

grid_table = AgGrid(filtered_df,
                    gridOptions=gridoption,
                    fit_columns_on_grid_load=True,
                    height=500,
                    theme= 'streamlit')

selected_row = grid_table['selected_rows']
```

```
if selected_row is not None:
    with st.container(border=True):
        st.subheader("Dosen Details", anchor=False)
        st.write(f"**NIP:** {selected_row['NIP'][0]}")
        st.write(f"**Nama     Dosen:**     {selected_row['Nama
Dosen'][0]}")
        st.write(f"**Mahasiswa Wali:** {selected_row['Mahasiswa
Wali'][0]}")
        st.write(f"**Status  Mengajar:**  {selected_row['Status
Mengajar'][0]}")

    if selected_row['Mahasiswa Wali'][0]:
        btn1, btn2, btn3 = st.columns(3)
        btn1.link_button("**Mata       Kuliah       Ajaran**",
url=f'/Prodi_Mata%20Kuliah%20Ajar?prodi={prodi}&dosen={selecte
d_row["Nama          Dosen"][0]}&nip={selected_row["NIP"][0]}',
use_container_width=True)
        btn2.link_button("**Dashboard      Dosen      Wali**",
url=f'/Prodi_Dashboard%20Dosen?prodi={prodi}&dosen={selected_r
ow["Nama             Dosen"][0]}&nip={selected_row["NIP"][0]}',
use_container_width=True)
        btn3.link_button("**Mahasiswa                    Wali**",
url=f'/Prodi_Mahasiswa%20Wali?prodi={prodi}&dosen={selected_ro
w["Nama              Dosen"][0]}&nip={selected_row["NIP"][0]}',
use_container_width=True)
    else:
        st.link_button("**Mata        Kuliah        Ajaran**",
url=f'/Prodi_Mata%20Kuliah%20Ajar?prodi={prodi}&dosen={selecte
d_row["Nama          Dosen"][0]}&nip={selected_row["NIP"][0]}',
use_container_width=True)
```

4.1.11 Dashboard Yudisium

Halaman *dashboard* yudisium ini dapat diakses oleh *user* program studi dengan tujuan memberikan visualisasi data hasil pembelajaran mahasiswa dalam satu program studi pada waktu yudisium. *User* dapat memilih periode yudisium yang ingin diperiksa dan juga angkatan apa saja yang akan muncul pada *dashboard* dimana pilihan angkatan akan mengikuti angkatan apa saja yang yudisium pada periode tersebut.

Segmen Program 4.11 Dashboard Yudisium

```
#Hide Pages
page_prodi = ['Prodi_Kelas Ajar', 'Prodi_Mata Kuliah Ajar',
'Prodi_Dashboard Dosen', 'Prodi_Mahasiswa Wali',
'Prodi_Dashboard Mahasiswa', 'Prodi_KHS-Transkrip']
page_dosen = ['Dashboard Dosen', 'Mahasiswa Wali', 'Mata
Kuliah Ajar', 'Dosen_Dashboard Mahasiswa', 'Dosen_KHS-
Transkrip']
page_mahasiswa = ['Dashboard Mahasiswa', 'KHS-Transkrip']
hide_pages(page_prodi+page_dosen+page_mahasiswa)
```

```
#SQLITE connection
conn = sqlite3.connect('db_skripsi.db')
cursor = conn.cursor()

#Check Prodi State
prodi = st.session_state['prodi']

#Get latest periode
cursor.execute("SELECT MAX(periodelulus) FROM listlulusan
WHERE namaunit = ?", (prodi,))
latest_periode = cursor.fetchone()[0]

#Functions
def extract_year(semester):
    return semester[:4]

def hitung_masa_studi(row):
    periodelulus = row.get('periodelulus')
    if periodelulus is None:
        periodelulus = latest_periode

    tahun_masuk, semester_masuk = row['periodemasuk'][:-2],
row['periodemasuk'][-2:]
    tahun_lulus, semester_lulus = periodelulus[:-2],
periodelulus[-2:]
    tahun_masuk, tahun_lulus = int(tahun_masuk),
int(tahun_lulus)

    total_tahun = tahun_lulus - tahun_masuk
    total_semesters = total_tahun * 2

    if semester_lulus == 'S2':
        total_semesters += 1
    if semester_masuk == 'S2':
        total_semesters -= 1

    return total_semesters + 1

#Data Yudisium
cursor.execute("SELECT nim, periodemasuk, periodelulus, ipk
FROM listlulusan WHERE namaunit = ?", (prodi,))
data = cursor.fetchall()
df = pd.DataFrame(data, columns=['nim', 'periodemasuk',
'periodelulus', 'ipk'])
df['masastudi'] = df.apply(hitung_masa_studi, axis=1)
df['angkatan'] = df['periodemasuk'].apply(extract_year)

#Data Yudisium
cursor.execute("SELECT nim, periodemasuk, periodelulus, ipk
FROM listlulusan WHERE namaunit = ?", (prodi,))
data = cursor.fetchall()
```

```
df = pd.DataFrame(data, columns=['nim', 'periodemasuk',
'periodelulus', 'ipk'])
df['masastudi'] = df.apply(hitung_masa_studi, axis=1)
df['angkatan'] = df['periodemasuk'].apply(extract_year)

with st.sidebar:
    #Get periode list
    list_periode =
df['periodelulus'].drop_duplicates().sort_values(ascending=Fal
se).astype(str).tolist()
    select_periode = st.sidebar.selectbox(label="Pilih
Periode", options=list_periode)

    temp = df[df['periodelulus'] == select_periode]
    list_angkatan = sorted(temp['angkatan'].unique(),
reverse=True)
    selected_angkatan = st.sidebar.multiselect(label="Pilih
Angkatan", options=list_angkatan, default=list_angkatan)

#Title
st.markdown(f'### Dashboard Yudisium - {prodi}')
st.markdown(f'#### Periode: {select_periode}')
st.markdown("---")

#Data Yudisium Hasil Filter Angkatan
if selected_angkatan:
    df_angkatan =
temp[temp['angkatan'].isin(selected_angkatan)]
else:
    df_angkatan = temp

if select_periode:
    #Dashboard Bagian Atas
    topleft_col, topmid_col, topright_col = st.columns(3)

    with topleft_col:
        st.subheader("Jumlah Mahasiswa Yudisium",
anchor=False)
        st.title(len(df_angkatan), anchor=False)

    with topmid_col:
        st.subheader("Mean Masa Studi per Yudisium",
anchor=False)
        avg_studi = round(df_angkatan['masastudi'].mean(),2)
        st.title(avg_studi, anchor=False)

    with topright_col:
        st.subheader("Mean IPK per Yudisium", anchor=False)
        avg_ipk = round(df_angkatan['ipk'].mean(),2)
        st.title(avg_ipk, anchor=False)

    #Dashboard Bagian Bawah
    st.markdown('<br><br>',True)
```

```
    bottomleft_col, bottommid_col, bottomright_col =
st.columns(3)
    with bottomleft_col:
        st.subheader(f"Yudisium per Angkatan Periode
{select_periode}", anchor=False)

        angkatan_counts =
df_angkatan['angkatan'].value_counts().reset_index()
        angkatan_counts.columns = ['angkatan', 'count']

        fig_angkatan =
go.Figure(data=[go.Pie(labels=angkatan_counts['angkatan'],
values=angkatan_counts['count'])])
        fig_angkatan.update_layout(font_size=20,
legend_font_size=15, hoverlabel_font_size=20,
legend=dict(title='Angkatan', x=0,y=1,))
        fig_angkatan.update_traces(textposition='outside',
textinfo='percent+label', hovertemplate ="<b>%{label}</b> <br>
Jumlah Mhs: %{value}")
        st.plotly_chart(fig_angkatan, True)

    filtered_df = df[df['periodelulus'] <= select_periode]
    # filtered_df =
df[(df['angkatan'].isin(selected_angkatan)) &
(df['periodelulus'] <= select_periode)]
    with bottommid_col:
        st.subheader(f"Trend Masa Studi Yudisium",
anchor=False)
        df_masastudi =
filtered_df.groupby('periodelulus')['masastudi'].mean().reset_
index().round(2)
        df_masastudi['periode'] = range(0, len(df_masastudi))

        fig_studi = px.scatter(df_masastudi, x='periode',
y='masastudi', color_discrete_sequence=['blue'],
                       trendline="lowess",
trendline_options=dict(frac=0.7),
trendline_color_override="grey")
        fig_studi.add_trace(
            go.Scatter(
                x=df_masastudi['periode'],
                y=df_masastudi['masastudi'],
                name='Avg Masa Studi',
                mode='lines',
                marker={'size':5},
                line = dict(color='blue', width=2.5)
            )
        )

        fig_studi.update_layout(
            showlegend=True,
            # margin=dict(t=50,l=10,b=10,r=10),
            legend_title_text='Legend',
```

```
            legend=dict(
                x=0.01,
                y=1.1,
                bgcolor='rgba(255, 255, 255, 0)',
                bordercolor='rgba(255, 255, 255, 0)'
            )
        )
        fig_studi.update_layout(legend_font_size=15,
hoverlabel_font_size=15)
        fig_studi.update_traces(hovertemplate ="<b>%{x}</b>
<br> Mean Masa Studi: %{y}")

fig_studi.update_xaxes(tickvals=df_masastudi.index.tolist(),
ticktext=df_masastudi['periodelulus'], tickfont_size=13)
        fig_studi.update_yaxes(tickfont_size=13)
        st.plotly_chart(fig_studi, True)


    with bottomright_col:
        st.subheader(f"Trend IPK Yudisium", anchor=False)
        df_ipk =
filtered_df.groupby('periodelulus')['ipk'].mean().reset_index(
).round(2)
        df_ipk['periode'] = range(0, len(df_masastudi))

        fig_ipk = px.scatter(df_ipk, x='periode', y='ipk',
color_discrete_sequence=['red'],
                     trendline="lowess",
trendline_options=dict(frac=0.7),
trendline_color_override="grey")
        fig_ipk.add_trace(
            go.Scatter(
                x=df_ipk['periode'],
                y=df_ipk['ipk'],
                name='Avg IPK',
                mode='lines',
                marker={'size':5},
                line = dict(color='red', width=2.5)
            )
        )

        fig_ipk.update_layout(
            showlegend=True,
            # margin=dict(t=50,l=10,b=10,r=10),
            legend_title_text='Legend',
            legend=dict(
                x=0.01,
                y=1.1,
                bgcolor='rgba(255, 255, 255, 0)',
                bordercolor='rgba(255, 255, 255, 0)'
            )
        )
```

```
        fig_ipk.update_layout(legend_font_size=15,
hoverlabel_font_size=15)
        fig_ipk.update_traces(hovertemplate ="<b>%{x}</b> <br>
Mean IPK: %{y}")
        fig_ipk.update_xaxes(tickvals=df_ipk.index.tolist(),
ticktext=df_ipk['periodelulus'], tickfont_size=13)
        fig_ipk.update_yaxes(tickfont_size=13)
        st.plotly_chart(fig_ipk, True)
```

4.1.12 Mata Kuliah dan Kelas

Halaman mata kuliah dan kelas ini dapat diakses oleh *user* program studi dengan tujuan memberikan informasi terkait dengan mata kuliah yang dibuka pada periode tertentu, diaman *user* dapat memilih periode akademik yang ingin di munculkan pada halaman. Informasi yang diberikan berupa jumlah mata kuliah dan jumlah kelas yang dibuka, perbandingan mata kuliah pilihan dan wajib yang dibuka mengikuti kurikulum terbaru, dan tabel informasi mata kuliah yang dapat di *filter* atau *search* dengan menggunakan *filter* pilih kodemk/namamk. *User* dapat memilih satu mata kuliah dan kelas untuk melihat detail mata kuliah dibawah tabel dan dapat melihat halaman detail kelas ajar dengan menekan tombol *show* kelas ajar.

Segmen Program 4.12 List Mata Kuliah dan Kelas

```
#Hide Pages
page_prodi = ['Prodi_Kelas Ajar', 'Prodi_Mata Kuliah Ajar',
'Prodi_Dashboard Dosen', 'Prodi_Mahasiswa Wali',
'Prodi_Dashboard Mahasiswa', 'Prodi_KHS-Transkrip']
page_dosen = ['Dashboard Dosen', 'Mahasiswa Wali', 'Mata
Kuliah Ajar', 'Dosen_Dashboard Mahasiswa', 'Dosen_KHS-
Transkrip']
page_mahasiswa = ['Dashboard Mahasiswa', 'KHS-Transkrip']
hide_pages(page_prodi+page_dosen+page_mahasiswa)

#SQLITE connection
conn = sqlite3.connect('db_skripsi.db')
cursor = conn.cursor()

#Check Prodi State
prodi = st.session_state['prodi']

#Functions
def get_persenkelulusan(periode, formatmk, cursor):
    query = f"""
        SELECT nim, nangka
        FROM krsmahasiswa
        WHERE kodemk || kelasmk LIKE '{formatmk}' AND periode
LIKE '{periode}'
    """
    cursor.execute(query)
```

```
    data = cursor.fetchall()
    df = pd.DataFrame(data, columns=['nim', 'nangka'])
    df['nangka'] = df['nangka'].astype(float)

    total_mahasiswa = len(df)
    if total_mahasiswa == 0:
        return '0.00 %'

    mahasiswa_lulus = (df['nangka'] >= 2.00).sum()
    persenlulus = (mahasiswa_lulus / total_mahasiswa) * 100
    persenlulus_rounded_ = round(persenlulus, 2)
    return (f'{persenlulus_rounded_} %')

with st.sidebar:
    #Get periode list
    cursor.execute(f"SELECT DISTINCT(periode) FROM kelas WHERE
namaunit = ? ORDER BY periode DESC", (prodi,))
    list_periode = [str(periode[0]) for periode in
cursor.fetchall()]
    selected_periode = st.sidebar.selectbox(label="Pilih
Periode", options=list_periode)

#Title and Sub-Title
st.write(f'### Mata Kuliah - Prodi {prodi}')
st.write(f'##### Periode - {selected_periode}')
st.write('---')

query = f"""
    SELECT kl.periode, kl.kodeunit, kl.tahun, kl.kodemk,
kl.namamk, kl.kelasmk, ku.wajibpilihan
    FROM kelas kl
    LEFT JOIN kurikulum ku ON kl.namaunit || kl.tahun ||
kl.kodemk = ku.namaunit || ku.tahun || ku.kodemk
    WHERE kl.namaunit = '{prodi}' AND kl.periode =
'{selected_periode}'
"""
df_matkul = pd.read_sql_query(query, conn)

query = f"""
    SELECT pgj.periode, pgj.kodemk, pgj.kelasmk, pgw.nama as
namadosen
    FROM pengajar pgj
    LEFT JOIN pegawai pgw ON pgj.nip = pgw.nip
    WHERE pgj.namaunit = '{prodi}' AND pgj.periode =
'{selected_periode}'
"""
df_dosen = pd.read_sql_query(query, conn)
merged_df = pd.merge(df_matkul, df_dosen, on=['periode',
'kodemk', 'kelasmk'], how='outer')
merged_df['wajibpilihan'] =
merged_df['wajibpilihan'].replace({'W': 'Wajib', 'P':
'Pilihan'})
```

```
query = f"""
    SELECT periode, kodemk, kelasmk, count(nim) as jumlahmhs
    FROM krsmahasiswa
    WHERE periode = '{selected_periode}'
    GROUP BY periode, kodemk, namamk, kelasmk
"""
df_kelas = pd.read_sql_query(query, conn)
merged_df = pd.merge(merged_df, df_kelas, on=['periode',
'kodemk', 'kelasmk'], how='left')
merged_df = merged_df.dropna(subset=['jumlahmhs'])

filtered_df = merged_df
filtered_df['persenlulus'] = filtered_df.apply(lambda row:
get_persenkelulusan(row['periode'],
row['kodemk']+row['kelasmk'], cursor), axis=1)

# Metric Atas
left, mid, right = st.columns((1,1,1))
with left:
    left.subheader('Perbandingan Mata Kuliah', anchor=False)

    unique_df = filtered_df.drop_duplicates(subset=['periode',
'kodeunit', 'tahun', 'kodemk', 'wajibpilihan'])
    grouped_df = unique_df.groupby(['periode', 'kodeunit',
'tahun', 'kodemk',
'wajibpilihan']).size().reset_index(name='count')
    wp_count =
grouped_df['wajibpilihan'].value_counts().reset_index()
    wp_count.columns = ['wajibpilihan', 'Count']
    fig = px.pie(data_frame=wp_count, values='Count',
names='wajibpilihan', hole=0.5)
    fig.update_traces(textfont_size=15,
                      hoverinfo='label+percent',
                      insidetextfont_size=18,
                      textposition='inside',
                      textinfo='percent+label',
                      hovertemplate ="<b>%{label}</b> <br>
Jumlah: %{value}",
                      rotation=35)
    fig.update_layout(
        width=300,
        height=250,
        margin=dict(l=25, r=25, t=20, b=30),
        showlegend=False,
        hoverlabel=dict(font_size=15),
    )
    st.plotly_chart(fig)

with mid:
    total_mk = str(filtered_df['kodemk'].nunique())
    mid.subheader('Jumlah Mata kuliah', anchor=False)
    mid.title(total_mk, anchor=False)
```

```
with right:
    total_kelas = str(filtered_df['kelasmk'].count())
    right.subheader('Jumlah Kelas', anchor=False)
    right.title(total_kelas, anchor=False)

st.markdown('<br>',True)
final_df = filtered_df[['namamk', 'kelasmk', 'namadosen',
'persenlulus', 'jumlahmhs', 'kodemk']]
final_df['infomatkul'] = final_df['kodemk'] + ' - ' +
final_df['namamk']

left_col, right_col = st.columns((9,1))
selected_infomatkul = left_col.multiselect('Pilih Kode MK/Nama
MK',options=final_df['infomatkul'].unique())
if not filtered_df.empty:
    with right_col:
        st.markdown(" ",True)
        st.markdown(" ",True)
        right_col.download_button(
            label="Download as CSV",
            data=filtered_df.to_csv(index=False).encode('utf-
8'),
            file_name=f'Mata_Kuliah-
Kelas_{prodi}_{selected_periode}.csv',
            mime='text/csv',
        )

if selected_infomatkul:
    grid_df =
final_df[final_df['infomatkul'].isin(selected_infomatkul)]
else:
    grid_df = final_df

gd = GridOptionsBuilder.from_dataframe(grid_df)
gd.configure_column('infomatkul', headerName='Mata Kuliah',
rowGroup=True, hide=True)
gd.configure_column('kodemk', header_name='Kode MK',
hide=True)
gd.configure_column('namamk', header_name='Nama MK')
gd.configure_column('kelasmk', headerName='Kelas')
gd.configure_column('namadosen', headerName='Nama Dosen')
gd.configure_column('persenlulus', header_name='% Kelulusan')
gd.configure_column('jumlahmhs', headerName='Mahasiswa',
aggFunc='sum', valueFormatter="value")

gd.configure_default_column(groupable=True)
gd.configure_selection(selection_mode='single',
use_checkbox=True)
gd.configure_pagination(paginationAutoPageSize=False,
paginationPageSize=20)
gd.configure_grid_options(suppressMovableColumns=True)
gridoption = gd.build()
```

```
grid_table = AgGrid(grid_df,
                    gridOptions=gridoption,
                    fit_columns_on_grid_load=True,
                    height=650,
                    theme= 'streamlit')

selected_row = grid_table['selected_rows']
if selected_row is not None:
    with st.container(border=True):
        st.subheader("Detail Mata Kuliah", anchor=False)
        st.write(f"**Kode MK:** {selected_row['kodemk'][0]}")
        st.write(f"**Nama MK:** {selected_row['namamk'][0]}")
        st.write(f"**Kelas:** {selected_row['kelasmk'][0]}")
        st.write(f"**Dosen Pengajar:**
{selected_row['namadosen'][0]}")

    st.link_button('Show Kelas Ajar',
url=f"Prodi_Kelas%20Ajar?periode={selected_periode}&kodemk={se
lected_row['kodemk'][0]}&kelas={selected_row['kelasmk'][0]}&do
sen={selected_row['namadosen'][0]}", use_container_width=True)
```

4.1.13 Simulasi Yudisium

Pada halaman ini *user* selaku program studi dapat mengakses fitur simulasi yudisium dan melihat syarat-syarat yudisium. Dimana *user* dapat memasukan NIM atau NRP mahasiswa yang ingin masukan kedalam simulasi yudisium dan mendapatkan informasi mengenai apakah mahasiswa sudah memenuhi syarat yudisium program studi, Pada halaman juga sistem akan memberikan visualisasi dan total syarat apa saja yang dipenuhi atau tidak dipenuhi oleh mahasiswa sehingga dapat memudahkan *user* untuk memutuskan langkah selanjutnya. *User* juga dapat melakukan *crosscheck* terhadap syarat matakuliah wajib dan informasi SKS dengan KHS atau Transkrip mahasiswa dengan menekan tombol khs dan transkrip, yang akan membuka halaman KHS dan Transkrip.

Segmen Program 4.13 Simulasi Yudisium

```
#Hide Pages
page_prodi = ['Prodi_Kelas Ajar', 'Prodi_Mata Kuliah Ajar',
'Prodi_Dashboard Dosen', 'Prodi_Mahasiswa Wali',
'Prodi_Dashboard Mahasiswa', 'Prodi_KHS-Transkrip']
page_dosen = ['Dashboard Dosen', 'Mahasiswa Wali', 'Mata
Kuliah Ajar', 'Dosen_Dashboard Mahasiswa', 'Dosen_KHS-
Transkrip']
page_mahasiswa = ['Dashboard Mahasiswa', 'KHS-Transkrip']
hide_pages(page_prodi+page_dosen+page_mahasiswa)

#SQLITE connection
conn = sqlite3.connect('db_skripsi.db')
```

```
cursor = conn.cursor()

#Check Prodi State
prodi = st.session_state['prodi']

#Get latest periode
cursor.execute("SELECT MAX(periode) FROM krsmahasiswa WHERE
namaunit = ?", (prodi,))
latest_periode = cursor.fetchone()[0]

#Functions
def extract_year(semester):
    return semester[:4]

def run_query(query, conn):
    return pd.read_sql_query(query, conn)

def hitung_prasyarat(student, syarat_yudisium):
    min_skslulus = int(syarat_yudisium['Minimum'][0])
    min_ipk = float(syarat_yudisium['Minimum'][1])
    min_ept = int(syarat_yudisium['Minimum'][2])
    max_matkulD = int(syarat_yudisium['Minimum'][3])
    mkwajiblulus = int(syarat_yudisium['Minimum'][4])

    requirements = {
        f'Minimum {min_skslulus} SKS Lulus': student['SKS
Lulus'] >= min_skslulus,
        f'IPK >= {min_ipk}': student['IPK'] >= min_ipk,
        f'EPT score >= {min_ept}': student['Nilai EPT'] >=
min_ept,
        'Lulus Semua MK Wajib': student['MK Wajib Lulus']>=
mkwajiblulus,
        f'SKS dengan nilai D <= {max_matkulD} SKS':
student['SKS_D'] <= max_matkulD
    }
    completed = sum(requirements.values())
    total = len(requirements)
    return requirements, completed, total

def color_cells(val):
    if str(val).strip() == 'D' or str(val).strip() == 'E':
        style = 'background-color: #FF6666;'  # Light red
    elif pd.isna(val):
        style = 'background-color: #CCCCCC;'  # Light gray
    else:
        return None
    return style

@st.experimental_dialog("Tambahkan Data Syarat Yudisium",
width='large')
def add_syarat_yudisium(prodi):
    with sqlite3.connect('db_skripsi.db') as conn:
        cursor = conn.cursor()
```

```
        periode_input = st.text_input(label='Periode')
        minskslulus_input = st.text_input(label='Min SKS
Lulus')
        minipk_input = st.text_input(label='Min IPK')
        minept_input = st.text_input(label='Min EPT')
        maxsksd_input = st.text_input(label='Max SKS Nilai D')

        minipk_input = minipk_input.replace(',', '.')
        if st.button('Confirm Add Data'):
            query = """
                INSERT INTO syaratyudisium (namaunit, periode,
minskslulus, minipk, minept, maxsksd)
                VALUES (?, ?, ?, ?, ?, ?)
            """
            cursor.execute(query, (prodi, periode_input,
minskslulus_input, minipk_input, minept_input, maxsksd_input))
            conn.commit()
            st.rerun()

@st.experimental_dialog("Edit Data Syarat Yudisium",
width='large')
def edit_syarat_yudisium(prodi, listperiode):
    with sqlite3.connect('db_skripsi.db') as conn:
        cursor = conn.cursor()

        periode = st.selectbox('Pilih Periode Syarat
Yudisium', options=listperiode, index=0)
        if periode:
            query = f"SELECT minskslulus, minipk, minept,
maxsksd FROM syaratyudisium WHERE namaunit = '{prodi}' AND
periode = '{periode}'"
            df_temp = pd.read_sql(query, conn)

        minskslulus_input = st.text_input(label='Min SKS
Lulus', value = int(df_temp['minskslulus']))
        minipk_input = st.text_input(label='Min IPK', value =
float(df_temp['minipk']))
        minept_input = st.text_input(label='Min EPT', value =
int(df_temp['minept']))
        maxsksd_input = st.text_input(label='Max SKS Nilai D',
value = int(df_temp['maxsksd']))

        minipk_input = minipk_input.replace(',', '.')
        if st.button('Confirm Edit Data'):
            query = """
                INSERT OR REPLACE INTO syaratyudisium
(namaunit, periode, minskslulus, minipk, minept, maxsksd)
                VALUES (?, ?, ?, ?, ?, ?)
            """
            cursor.execute(query, (prodi, periode,
minskslulus_input, minipk_input, minept_input, maxsksd_input))
            conn.commit()
```

```
            st.rerun()


with st.sidebar:
    #Get periode list
    st.sidebar.write(f"Prodi: {prodi}")

    if st.button("**Clear Data Simulasi**", type='primary',
use_container_width=True):
        st.session_state.nim_list = []

#Title and Sub-Title
st.write(f'### Yudisum - Prodi {prodi}')

tabsimulasi, tabsyarat = st.tabs(['Simulasi', 'Syarat
Yudisium'])
with tabsyarat:
    st.write("#### Syarat Yudisium")

    query = f"""
        SELECT kodemk, namamk, namaunit, wajibpilihan as
kurikulum
        FROM kurikulum
        WHERE namaunit = '{prodi}' AND tahun = (SELECT
MAX(tahun) FROM kurikulum WHERE namaunit = '{prodi}')
    """
    df_kurikulum = pd.read_sql(query, conn)
    df_kurikulum['kurikulum'] =
df_kurikulum['kurikulum'].replace({'W': 'Wajib', 'P':
'Pilihan'})

    df_matkulwajib = df_kurikulum[df_kurikulum['kurikulum'] ==
'Wajib']
    total_makulwajib = str(len(df_matkulwajib))

    query = f"""
        SELECT periode
        FROM syaratyudisium
        WHERE namaunit = '{prodi}'
        ORDER BY periode desc
    """
    df_periode = pd.read_sql(query,conn)
    list_periode =
df_periode['periode'].sort_values(ascending=False).tolist()

    left,right,pad = st.columns((1,1,5))
    select_periode = left.selectbox(label="Cek Syarat Yudisium
Periode", options=list_periode, index=0)

    query = f"""
        SELECT minskslulus as 'Min SKS Lulus', minipk as 'Min
IPK', minept as 'Min EPT Score', maxsksd as 'Max SKS Nilai D'
        FROM syaratyudisium
```

```
        WHERE namaunit = '{prodi}' and periode =
'{select_periode}'
    """
    df_yudisiumreq = pd.read_sql(query, conn)
    df_yudisiumreq['Lulus Semua Matkul Wajib'] =
total_makulwajib
    df_transformed = df_yudisiumreq.melt(var_name='Syarat',
value_name='Value')

    gb = GridOptionsBuilder.from_dataframe(df_transformed)
    gridOptions = gb.build()

    grid, pad = st.columns(2)
    with grid:
        st.dataframe(df_transformed,hide_index=True,
use_container_width=True)

    # st.write(df_transformed)

    hleft,hright = st.columns(2)
    left2,mid2,right2,pad = hleft.columns((1,1,1,3))
    if left2.button('Add Data', key='add_syarat',
use_container_width=True):
        add_syarat_yudisium(prodi)
    if mid2.button('Edit Data', key='edit_syarat',
use_container_width=True):
        edit_syarat_yudisium(prodi, list_periode)

    st.markdown('<br>',True)
    st.write('#### Mata Kuliah Wajib Sesuai Kurikulum')
    filterwajib = st.toggle('Filter Wajib Saja')
    if filterwajib == True:
        filtered_df = df_kurikulum[df_kurikulum['kurikulum']
== 'Wajib']
    else:
        filtered_df = df_kurikulum

    gb = GridOptionsBuilder.from_dataframe(filtered_df)
    gb.configure_default_column(resizable=True,
filterable=True, sortable=True)
    gridOptions = gb.build()

    grid_response = AgGrid(
        filtered_df,
        gridOptions=gridOptions,
        update_mode='no_update',
        height=400,
        fit_columns_on_grid_load=True  # Ensure columns fit
within the grid
    )

with tabsimulasi:
    st.write('### Simulasi Yudisium')
```

```
    if 'nim_list' not in st.session_state:
        st.session_state.nim_list = []

    input_method = st.radio("Select Input Method:", ["Manual
Input", "Mahasiswa Tugas Akhir"], horizontal=True)
    if input_method == "Manual Input":
        nim_input = st.text_area("Enter Student IDs (NIM),
separated by commas")
        if st.button("Submit", help="Click to submit manually
entered NIMs"):
            new_nim = [id.strip() for id in
nim_input.split(',')]
            st.session_state.nim_list += new_nim
            if st.session_state.nim_list:
                nim_list = [nim for nim in
st.session_state.nim_list if nim.strip() != '' and nim.strip()
not in ['', None]]
                nim_list = list(set(nim_list))
                st.session_state.nim_list = nim_list
            else:
                nim_list = []
    elif input_method == "Mahasiswa Tugas Akhir":
        if st.button('Get NIM', help="Click to Get NIM
Mahasiswa Tugas Akhir"):
            if prodi in ['Informatika', 'Business
Management']:
                query = f"SELECT DISTINCT nim FROM
krsmahasiswa WHERE namaunit LIKE '{prodi}' AND periode LIKE
'2023S1' AND namamk LIKE 'Skripsi'"
                df_nim = pd.read_sql_query(query, conn)
                new_nim = df_nim['nim'].tolist()
                st.session_state.nim_list += new_nim
                if st.session_state.nim_list:
                    nim_list = [nim for nim in
st.session_state.nim_list if nim.strip() != '' and nim.strip()
not in ['', None]]
                    nim_list = list(set(nim_list))
                    st.session_state.nim_list = nim_list
                else:
                    nim_list = []

    list_of_nim = st.session_state.nim_list
    valid_nims = [nim for nim in list_of_nim if len(nim) ==
32]
    df_nim = pd.DataFrame()
    df_transkrip = pd.DataFrame()
    df_ept = pd.DataFrame()

    #Query Syarat Yudisium (Auto ambil syarat dengan periode
teratas atau terbesar atau terbaru dari tabel syaratyudisium)
```

```
    query = f"SELECT minskslulus, minipk, minept, maxsksd FROM
syaratyudisium WHERE namaunit ='{prodi}' AND periode =(SELECT
MAX(periode) FROM syaratyudisium WHERE namaunit = '{prodi}')"
    df_syaratyudisium = pd.read_sql(query, conn)
    syarat_yudisium = {'Syarat': ['Total SKS Lulus', 'IPK',
'Nilai EPT', 'SKS dengan Nilai D', 'Lulus Matkul Wajib'],
                       'Minimum':
[int(df_syaratyudisium['minskslulus']),
float(df_syaratyudisium['minipk']),
int(df_syaratyudisium['minept']),
int(df_syaratyudisium['maxsksd']), total_makulwajib]}

    query=''
    if len(valid_nims) > 1:
        query = f"SELECT nim, periodemasuk as angkatan FROM
listmahasiswa WHERE nim in {tuple(valid_nims)} AND namaunit
LIKE '{prodi}'"
    elif len(valid_nims) == 1:
        query = f"SELECT nim, periodemasuk as angkatan FROM
listmahasiswa WHERE nim LIKE '{valid_nims[0]}' AND namaunit
LIKE '{prodi}'"

    if query:
        if df_nim.empty:
            df_nim = pd.read_sql_query(query, conn)
            df_nim['angkatan'] =
df_nim['angkatan'].apply(extract_year)
        else:
            df_temp = pd.read_sql_query(query, conn)
            df_temp['angkatan'] =
df_temp['angkatan'].apply(extract_year)
            df_nim = pd.concat([df_nim, df_temp],
ignore_index=True)


        for _, row in df_nim.iterrows():
            nim = row['nim']

            # Query for transkrip
            query_transkrip = f"""
                SELECT periode, nim, kodemk, namamk,
tahunkurikulum, sks, nangka, nhuruf
                FROM krsmahasiswa
                WHERE nim LIKE '{nim}' AND periode <
'{latest_periode}' AND nhuruf is not NULL
            """
            df_transkrip_temp = run_query(query_transkrip,
conn)
            df_transkrip_temp['nim'] = nim  # Add nim column
to identify the records
            df_transkrip_temp =
df_transkrip_temp.drop_duplicates(subset=['kodemk', 'namamk'],
keep='last').reset_index(drop=True)
```

```
            df_transkrip = pd.concat([df_transkrip,
df_transkrip_temp], ignore_index=True)

            # Query for ept
            query_ept = f"""
                SELECT nim, nilai_ept
                FROM detailmahasiswa
                WHERE nim LIKE '{nim}'
            """
            df_ept_temp = run_query(query_ept, conn)
            df_ept_temp['nim'] = nim  # Add nim column to
identify the records
            df_ept = pd.concat([df_ept, df_ept_temp],
ignore_index=True)

        df_summary = df_nim.copy()
        df_kurikulumtemp = df_kurikulum.copy()
        df_transkrip_copy = df_transkrip.copy()

        df_kurikulumtemp =
df_kurikulumtemp.drop(columns=['namamk', 'namaunit'])
        df_transkrip = pd.merge(df_transkrip,
df_kurikulumtemp, on='kodemk', how='left')

        #Perhitungan Bobot Nilai dan IPK
        df_transkrip['grade_points'] = df_transkrip['sks'] *
df_transkrip['nangka']
        df_ipk =
df_transkrip.groupby('nim').agg({'grade_points': 'sum', 'sks':
'sum'}).reset_index()
        df_ipk['IPK'] = (df_ipk['grade_points'] /
df_ipk['sks']).round(2)

        #Perhitungan Total SKS Yang Sudah Diambil
        df_totalsks =
df_transkrip.groupby('nim')['sks'].sum().reset_index()
        df_totalsks.columns = ['nim', 'totalsks']

        #Perhitungan Total SKS Yang Sudah Diambil dan Lulus
(Diatas D)
        df_totalskslulus =
df_transkrip[~df_transkrip['nhuruf'].isin(['D',
'E'])].groupby('nim')['sks'].sum().reset_index()
        df_totalskslulus.columns = ['nim', 'totalskslulus']

        #Perhitungan Total SKS Yang Sudah Diambil dan Tidak
Lulus (Dibawah C)
        df_totalsksnilaid =
df_transkrip[df_transkrip['nhuruf'].isin(['D',
'E'])].groupby('nim')['sks'].sum().reset_index()
        df_totalsksnilaid.columns = ['nim', 'totalsksnilaid']

        #Perhitungan MK Wajib yang Lulus (Nilai Diatas D)
```

```
        df_transkrip_filtered =
df_transkrip[(df_transkrip['nhuruf'] != 'D') &
(df_transkrip['nhuruf'] != 'E') & (df_transkrip['kurikulum']
== 'Wajib')]
        df_mkwajiblulus =
df_transkrip_filtered.groupby('nim')['kodemk'].nunique().reset
_index()
        df_mkwajiblulus.columns = ['nim', 'mkwajiblulus']

        df_summary = df_summary.merge(df_totalsks, on='nim',
how='left')
        df_summary = df_summary.merge(df_totalskslulus,
on='nim', how='left')
        df_summary = df_summary.merge(df_totalsksnilaid,
on='nim', how='left')
        df_summary['totalskslulus'] =
df_summary['totalskslulus'].fillna(0)
        df_summary['totalsksnilaid'] =
df_summary['totalsksnilaid'].fillna(0)

        df_summary = df_summary.merge(df_ept[['nim',
'nilai_ept']], on='nim', how='left')
        df_summary['nilai_ept'] =
df_summary['nilai_ept'].fillna(0)

        df_summary = df_summary.merge(df_ipk[['nim', 'IPK']],
on='nim', how='left')

        df_summary = pd.merge(df_summary, df_mkwajiblulus,
on='nim', how='left')
        df_summary['mkwajiblulus'] =
df_summary['mkwajiblulus'].fillna(0).astype(int)
        df_summary = df_summary.rename(columns={
            'nim': 'NIM',
            'angkatan': 'Angkatan',
            'totalsks': 'Total SKS Ambil',
            'totalskslulus': 'SKS Lulus',
            'totalsksnilaid': 'SKS_D',
            'nilai_ept': 'Nilai EPT',
            'mkwajiblulus': 'MK Wajib Lulus'
        })

        #View Summary Data
        st.markdown('<br>',True)
        st.subheader('Summary of Simulasi Yudisium Checker',
anchor=False)
        summary_result = {
            'NIM': [],
            'Angkatan': [],
            'Syarat Terpenuhi': []
        }

        for idx, mahasiswa in df_summary.iterrows():
```

```
            requirements, completed, total =
hitung_prasyarat(mahasiswa, syarat_yudisium)
            summary_result['NIM'].append(mahasiswa['NIM'])

summary_result['Angkatan'].append(mahasiswa['Angkatan'])
            summary_result['Syarat
Terpenuhi'].append(f"{completed}/{total}")

        summary_df = pd.DataFrame(summary_result)
        # st.table(summary_df)

        grid_table =
GridOptionsBuilder.from_dataframe(summary_df)
        grid_table.configure_default_column(groupable=True)

grid_table.configure_pagination(paginationAutoPageSize=True)

grid_table.configure_grid_options(suppressMovableColumns=True)

        gridoption = grid_table.build()
        grid_table = AgGrid(summary_df,
                            gridOptions=gridoption,
                            fit_columns_on_grid_load=True,
                            height=350,
                            theme= 'streamlit')

        for idx, student in df_summary.iterrows():
            with st.expander(f"{student['NIM']} -
{student['Angkatan']}"):
                st.write(f"##### Details for {student['NIM']}
- Angkatan {student['Angkatan']}")

                requirements, _, _ = hitung_prasyarat(student,
syarat_yudisium)
                requirement_df =
pd.DataFrame(list(requirements.items()),
columns=['Requirement', 'Status'])
                requirement_df['Status'] =
requirement_df['Status'].apply(lambda x: '✔' if x else '✘')
                requirement_df.index = requirement_df.index +
1
                st.table(requirement_df)

                col1, col2, col3 = st.columns((1.5,1.5,2))

                with col1:
                    labels = ['SKS Lulus','SKS Tidak Lulus']
                    values = [student['SKS Lulus'],
student['SKS_D']]
                    colors = ['#1f77b4', '#ff7f0e']

                    fig =
go.Figure(data=[go.Pie(labels=labels,
```

```
values=values,
                                                    pull=[0,
0.15],
                                                    textfont_size
= 15,

marker=dict(colors=colors),)])
                    fig.update_layout(title='Distribusi SKS
Keseluruhan',

legend=dict(orientation='v', x=0.9, y=1.1,
bordercolor='#f5e4e6', borderwidth=0.8, font=dict(size=13)),
height=420)
                    st.plotly_chart(fig)

                with col2:
                    lulus_mkwajib = [student['MK Wajib
Lulus'], int(syarat_yudisium['Minimum'][4]) - student['MK
Wajib Lulus']]
                    colors = ['#1f77b4', '#d1324a']
                    labels=['MK Wajib Lulus', 'MK Wajib
Belum']
                    custom_text = [f'{val}-MatKul' for label,
val in zip(labels, [lulus_mkwajib[0], lulus_mkwajib[1]])]
                    fig = go.Figure(data=[go.Pie(labels=['MK
Wajib Terpenuhi', 'MK Wajib Kurang'],

values=lulus_mkwajib,

text=custom_text,
                                                    hole=.3,
                                                    textfont_size
= 15,

marker=dict(colors=colors),)])
                    fig.update_layout(title_text='Prasyarat MK
Wajib',

legend=dict(orientation='v', x=0.9, y=1.1,
bordercolor='#f5e4e6', borderwidth=0.8, font=dict(size=13)),
                                        height=400,
hoverlabel=dict(font_size=15),)

                    fig.update_traces(
                        hovertemplate ="<b>%{label}</b> <br>
Jumlah: %{value}",
                    )
                    st.plotly_chart(fig,
use_container_width=True)
                with col3:
                    #Dataframe for Tabel in Expanded row for
MK Wajib
```

```
                    df_kurikulum_wajib =
df_kurikulum[df_kurikulum['kurikulum'] ==
'Wajib'].reset_index(drop=True)
                    df_transkrip_temp =
df_transkrip_copy[df_transkrip_copy['nim'] ==
student['NIM']].reset_index(drop=True)
                    df_transkrip_temp =
df_transkrip_temp.drop_duplicates(subset=['kodemk', 'namamk'],
keep='last').reset_index(drop=True)
                    merged_df = pd.merge(df_kurikulum_wajib,
df_transkrip_temp, on=['kodemk', 'namamk'], how='left')
                    result_df = merged_df[['kodemk', 'namamk',
'kurikulum', 'nhuruf']]
                    result_df.index = result_df.index + 1

                    # Apply conditional formatting to the
'nhuruf' column

                    toggle_mkwajib = st.toggle('Belum Lulus',
key=f'toggle_mkwajib{student["NIM"]}')
                    if toggle_mkwajib == True:
                        temp_filtered =
result_df[result_df['nhuruf'].isin(['D', 'E']) |
result_df['nhuruf'].isna()].reset_index(drop=True)
                        temp_filtered.index =
temp_filtered.index + 1
                        temp_filtered =
temp_filtered.style.map(color_cells, subset=['nhuruf'])
                    else:
                        temp_filtered =
result_df.style.map(color_cells, subset=['nhuruf'])

                    st.dataframe(temp_filtered,
use_container_width=True)
                st.link_button("**KHS/Transkrip**",
use_container_width=True, url=f'/Prodi_KHS-
Transkrip?prodi={prodi}&nim={student["NIM"]}')
```