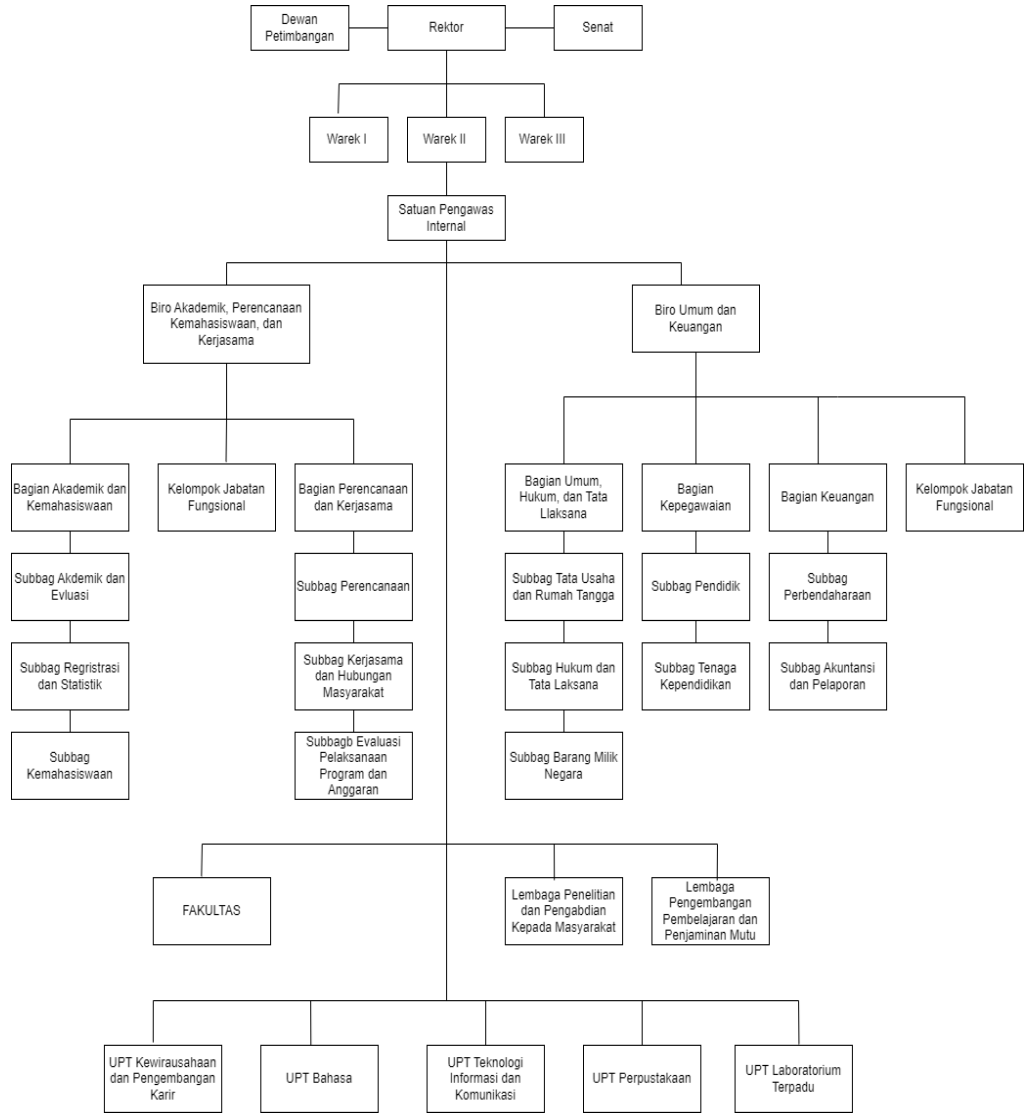
BAB IV
HASIL DAN PEMBAHASAN

4.1 Profil UPN "Veteran" Jakarta

Universitas Pembangunan Nasional "Veteran" Jakarta adalah perguruan tinggi negeri yang mempunyai lokasi di Ibu Kota Jakarta, mencakup wilayah Depok dan juga Jakarta Selatan. Sejarah UPN "Veteran" Jakarta dimulai pada tanggal 15 Desember 1958, ketika didirikan dengan nama Akademi Pembangunan Nasional "Veteran" di Yogyakarta. Akademi ini kemudian berkembang menjadi Perguruan Tinggi Pembangunan Nasional (PTPN). Pada 21 Februari 1967, PTPN, yang saat itu terdiri dari tiga akademi, bergabung menjadi satu dengan nama PTPN "Veteran" Cabang Jakarta. Pada 30 November 1977, nama tersebut diubah menjadi Universitas Pembangunan Nasional (UPN) "Veteran" Cabang Jakarta. Pada 27 Februari 1993, UPN "Veteran" Jakarta memisahkan diri dari induknya, Universitas Pembangunan Nasional "Veteran" Yogyakarta. Kemudian, pada 29 November 1994, statusnya berubah dari Perguruan Tinggi Kedinasan menjadi Perguruan Tinggi Swasta. Pada tahun 2000, semua program studi di UPN "Veteran" Jakarta memperoleh akreditasi dari BAN-PT. Akhirnya, pada tanggal 6 Oktober 2014, UPN "Veteran" Jakarta ditetapkan menjadi Perguruan Tinggi Negeri.

4.1.1 Struktur Organisasi UPN "Veteran" Jakarta

Berdasarkan Surat Keputusan Rektor UPN "Veteran"Jakarta Nomor : KEP/858/UN61.0/2017. Tanggal 22 November 2017 Tentang Organisasi UPN "Veteran"Jakarta. Berikut ini merupakan susunan organisasi tata kerja UPN "Veteran" Jakarta:



Gambar 4.1 Struktur Organisasi UPN "Veteran" Jakarta

4.2 Dataset

Penelitian ini menggunakan dataset berdasarkan observasi pada Universitas Pembangunan Nasional "Veteran" Jakarta, yang berisi mengenai atribut – atribut yang digunakan untuk memperoleh sistem prediksi mahasiswa berpotensi *drop-out*. Pada penelitian ini data yang digunakan bersumber dari *endpoint* API yang meliputi data histori mahasiswa dan data list mahasiswa Universitas Pembangunan Nasional "Veteran" Jakarta. Jumlah data yang dihasilkan dari data histori mahasiswa yaitu 176.063 *rows* dan list mahasiswa yaitu 29.771 *rows*. Kemudian setelah melewati tahap *preprocessing*, jumlah data yang tersedia dan digunakan untuk melatih model yaitu 23.420 *rows* . Berikut ini merupakan inisisalisasi penarikan *endpoint* API:

1. Pemodelan Data

```python
import requests
import pandas as pd
from datetime import datetime

# Nonaktifkan peringatan SSL jika diperlukan
requests.packages.urllib3.disable_warnings()

# API endpoint URL
url = ''

# API key, secret key, and Basic Auth credentials
api_key_name = ''
api_secret_key = ''
authorization = ''

headers = {
    'API_KEY_NAME': api_key_name,
    'API_SECRET_KEY': api_secret_key,
    'Authorization': authorization
}

# Mendapatkan tahun sekarang
current_year = datetime.now().year

# Membuat list untuk menyimpan DataFrame
dfs = []
```

Gambar 4.2 Inisialisasi API Key

Gambar diatas merupakan tahapan inisialisasi *endpoint* API yang mencakup *API key*, *secret key*, dan *auth credential.*

```python
for year in range(current_year, 2014, -1):
    for periode in [2, 1]:
        id_periode = f'{year}{periode}'
        parameter_list = {'id_periode': id_periode}

        # Kirim permintaan POST
        response = requests.post(url, headers=headers, data=parameter_list, verify=False)

        if response.status_code == 200:
            data = response.json()["data"]
            df = pd.DataFrame(data)
            dfs.append(df)
        else:
            print("Error:", response.status_code)

# Menggabungkan semua DataFrame menjadi satu DataFrame
data_mhs = pd.concat(dfs, ignore_index=True)
data_mhs
```

Gambar 4.3 Get Data Histori Mahasiswa

```python
for year in range(current_year, 2014, -1):
    angkatan = f'{year}'
    parameter_list = {'angkatan': angkatan}

    response = requests.post(url, headers=headers, data=parameter_list, verify=False)
    if response.status_code == 200:
        data = response.json()["data"]
        df = pd.DataFrame(data)
        dfs.append(df)
    else:
        print("Error:", response.status_code)

# Menggabungkan semua DataFrame menjadi satu DataFrame
data_ipk = pd.concat(dfs, ignore_index=True)
data_ipk
```

Gambar 4.4 Get API Data List Mahasiswa

Setelah melakukan inisisalisasi *endpoint* API, dilakukan penarikan data dengan

method POST. Pengambilan data dilakukan pada periode 2015 ganjil hingga periode 2023 genap. Gambar diatas merupakan proses penarikan data yang dilakukan pada data histori mahasiswa dan list mahasiswa. Berikut ini merupakan *field* dataset untuk data histori mahasiswa dan data list mahasiswa.

```
nim                          int64
nama_mahasiswa              object
kode_program_studi           int64
program_pendidikan          object
program_studi               object
ips                        float64
sks_semester                 int64
status_mahasiswa_semester   object
id_periode                   int64
dtype: object
```

Gambar 4.5 Field Data Histori Mahasiswa

```
nim                          int64
nama_mahasiswa              object
jenis_kelamin               object
jalur_penerimaan            object
tahun_angkatan               int64
semester_masuk             float64
tanggal_mulai_masuk         object
kode_program_studi           int64
nama_program_studi          object
propinsi                    object
kota_kab                    object
kecamatan                   object
ipk                        float64
status_mahasiswa_terakhir   object
penerima_beasiswa_kipk      object
penerima_beasiswa_kjmu      object
kelompok_ukt                object
nominal_ukt                float64
dtype: object
```

Gambar 4.6 Field Data List Mahasiswa

2. Prediksi Data

```
api_key_name = ''
api_secret_key = ''
authorization = '='
headers = {
'API_KEY_NAME': api_key_name,
'API_SECRET_KEY': api_secret_key,
'Authorization': authorization,
}
```

Gambar 4.7 Inisialisasi API Key Prediksi

```python
# GET DATA FROM API
def get_histori_mahasiswa(url):
    current_year = datetime.now().year
    dfs = []
    for year in range(current_year, 2014, -1):
        for periode in [2, 1]:
            id_periode = f'{year}{periode}'
            parameter_list = {'id_periode': id_periode}
            response = requests.post(url, headers=headers, data=parameter_list, verify=False)
            if response.status_code == 200:
                data = response.json()["data"]
                df = pd.DataFrame(data)
                dfs.append(df)
            else:
                print("Error:", response.status_code)

    data_mhs = pd.concat(dfs, ignore_index=True)
    return data_mhs
```

Gambar 4.8 Get Data Prediksi (Histori Mahasiswa)

```python
def get_list_mahasiswa(url):
    current_year = datetime.now().year
    dfs = []

    for year in range(current_year, 2014, -1):
        angkatan = f'{year}'
        parameter_list = {'angkatan': angkatan}
        response = requests.post(url, headers=headers, data=parameter_list, verify=False)
        if response.status_code == 200:
            data = response.json()["data"]
            df = pd.DataFrame(data)
            dfs.append(df)
        else:
            print("Error:", response.status_code)

    data_ipk = pd.concat(dfs, ignore_index=True)
    return data_ipk
```

Gambar 4.9 Get Data Prediksi (List Mahasiswa)

Gambar diatas merupakan *get* data *endpoint* API yang dilakukan dalam tahap prediksi data. Penarikan data dilakukan pada dua *endpoint* yang berbeda, yaitu data histori mahasiswa dan data list mahasiswa. Pada tahapan ini, penarikan data dilakukan secara dinamis sehingga data menjadi *real time.* Langkah pertama yaitu melakukan inisialisasi API *key*, selanjutnya melakukan *looping* untuk data histori dan list mahasiswa. Sebelum dilakukan *looping*, terlebih dahulu dilakukan inisialisasi *dataFrame.*

Pada data histori mahasiswa dilakukan *looping* dari tahun 2015 hingga saat ini dengan parameter_list berupa id_periode. Parameter 1 merupakan periode ganjil dan parameter 2 merupakan periode genap. Kemudian setelah dilakukan penarikan data, data digabungkan dan disimpan dalam variabel data_mhs. Pada data list mahasiswa, dilakukan *looping* dari tahun 2015 sampai dengan saat ini dengan parameter_list berupa angkatan. Kemudian data disimpan dalam variabel data_ipk.

4.3 Preprocessing Data

Setelah melakukan penarikan data kemudian dilakukan tahap *preprocessing.* Tahap *preprocessing* data pada penelitian ini terdiri dari beberapa bagian seperti

data cleaning, data transformation, feature engineering, dan *data scalling.*

4.3.1 *Data Cleaning*

Pada tahap ini, dilakukan pemilihan atribut-atribut yang diperlukan dan yang tidak diperlukan baik dalam pembuatan model maupun prediksi dan visualisasi. Berikut ini merupakan tahapan pembersihan data yang dilakukan:

1. **Pemodelan Data**

   Pada tahap ini dilakukan persiapan atribut-atribut yang digunakan:

```
# Filter program pendidikan S1
data_s1_only = data_mhs[data_mhs['program_pendidikan'] == 'S1']
data_s1_only
```

Gambar 4.10 Memilih Atribut Mahasiswa Tingkat S1

Pada gambar diatas dilakukan penarikan data khusus mahasiswa dengan tingkat S1 yang kemudian disimpan pada variabel data_s1_only.

```
# Merge Dataframe df_mhs & df_ipk
data_mahasiswa = pd.merge(data_s1_only.drop(columns=['ips']),
                          data_ipk[['nim', 'ipk']], on='nim', how='inner')
data_mahasiswa
```

Gambar 4.11 Drop Column IPS

Pada gambar diatas dilakukan *merge* data yang bersumber dari 2 *endpoint* API. Penggabungan data dilakukan untuk mendapatkan data mahasiswa tingkat S1 dengan dilengkapi IPK. Penggabungan ini dilakukan dengan proses *merge* data_s1_only dengan data_ipk. Pada tahap ini juga dilakukan penghapusan *field* 'ips'. Data yang telah digabungkan kemudian disimpan dalam variabel data_mahasiswa.

```
# Create DataFrame sks
sks = data_s1_only.groupby('nim')['sks_semester'].sum().reset_index()
sks.columns = ['nim', 'sks']
sks
```

Gambar 4.12 Membuat DataFrame SKS Tempuh

Pada gambar diatas dilakukan pembuatan df sks yang berisi nim dan sks pada data_s1_only.

```
# Merge DataFrame sks & ipk
sks_with_ipk = pd.merge(sks, data_ipk[['nim', 'ipk']], on='nim', how='inner')
sks_with_ipk
```

Gambar 4.13 Merge DataFrame SKS dan Data List Mahasiswa

Setelah membuat df nim dan sks, selanjutnya adalah menambahkan *field* 'ipk' dengan melakukan *merge* antara df sks dengan data_ipk yang mencakup 'nim' dan 'ipk' dan diberi nama df sks_with_ipk.

```
import datetime

current_year = datetime.datetime.now().year

# Initial digits
first_nim = data_s1_only.groupby('nim')['id_periode'].min().reset_index()
first_nim.columns = ['nim', 'first_id_periode']

# Occurrences of ID periods
id_periode_count = data_s1_only.groupby(['nim']).size().reset_index(name='id_periode_count')
merged_df = pd.merge(data_s1_only, first_nim, on='nim', how='left')
merged_df['tahun_masuk'] = merged_df['first_id_periode'].astype(str).str[:4].astype(int)

# Counting cuti & non-aktif
inactive_status = merged_df[merged_df['status_mahasiswa_semester'].isin(['cuti', 'non-aktif'])].
groupby('nim').size().reset_index(name='inactive_count')

merged_df = pd.merge(merged_df, inactive_status, on='nim', how='left')
merged_df['inactive_count'].fillna(0, inplace=True)

merged_df = pd.merge(merged_df, id_periode_count, on='nim', how='left')
merged_df['id_periode_count'].fillna(0, inplace=True)

merged_df['masa_studi'] = merged_df['id_periode_count'] - merged_df['inactive_count']
merged_df[['nim', 'masa_studi']]
```

Gambar 4.14 Penghitungan Masa Studi

Kemudian setelah mendapatkan df nim, sks, dan ipk, selanjutnya adalah membuat penghitungan masa studi. Langkah pertama adalah menentukan tahun angkatan mahasiswa dari nim awal mahasiswa yang diambil pada data_s1_only. Kemudian dilakukan penghitungan berapa kali nim mahasiswa muncul dalam dataset. Selanjutnya dilakukan penggabungan data S1 dengan digit awal nim. Kemudian di inisialisasi bahwa digit nim awal merupakan tahun masuk mahasiswa. Selanjutnya dilakukan inisialisasi status mahasiswa yang *inactive* yaitu dengan melakukan penggabungan df merged_df dengan status mahasiswa *inactive.* Kemudian dilakukan penggabungan antara merged_df dengan id_periode_count. Selanjutnya dilakukan penanganan *missing values*, yaitu dengan mengubahnya ke nilai 0. Langkah terakhir adalah melakukan penghitungan masa studi yaitu menghitung banyaknya id periode mahasiswa yang muncul dalam df dan dikurang dengan status mahasiswa yang *incative.*

```
# Merge DataFrame sks & data_ipk
sks_with_ipk = pd.merge(sks, data_ipk[['nim', 'ipk']], on='nim', how='inner')

# Merge DataFrame merged_df & sks_with_ipk
merged_df_all = pd.merge(merged_df[['nim', 'masa_studi']], sks_with_ipk, on='nim', how='inner')

merged_df_all
```

Gambar 4.15 Merge Data IPK, SKS, Masa Studi

Pada gambar diatas dilakukan penggabungan antara df sks_with_ipk dengan merged_df yaitu beirisi ‘nim’ dan ‘masa_studi’ dan disimpan dalam df merged_df_all.

```
# Removing duplicates based on the 'nim' column value
df_mahasiswa = merged_df_all.drop_duplicates(subset=['nim'], ignore_index=True)
df_mahasiswa
```

Gambar 4.16 Menghapus Duplikasi Data

Dalam tahap ini, dilakukan penghapusan *duplicate* pada nim sehingga tidak ada lagi duplikasi data pada df_mahasiswa.

2. Prediksi Data

Pada tahap ini dilakukan *preprocessing data* untuk data baru yang akan dilakukan prediksi. Berikut merupakan tahapan-tahapannya.

```
def preprocess_data(data_mahasiswa):
    # Create DataFrame sks
    data_mahasiswa['sks_semester'] = data_mahasiswa['sks_semester'].astype(np.int64)
    sks = data_mahasiswa.groupby(['nim'])['sks_semester'].sum().reset_index()
    sks.columns = ['nim','sks']
    sks
    # Merge DataFrame sks & ipk
    sks_with_ipk = pd.merge(sks, data_mahasiswa[['nim', 'nama_mahasiswa',
                                                 'status_mahasiswa_terakhir', 'ipk', 'tahun_angkatan',
                                                 'nama_program_studi']], on='nim', how='inner')
    sks_with_ipk
```

Gambar 4.17 Membuat Fungsi Preprocess Data Prediksi

```
    # Initial digits
    first_nim = data_mahasiswa.groupby('nim')['id_periode'].min().reset_index()
    first_nim.columns = ['nim', 'first_id_periode']
    # Occurrences of ID periods
    id_periode_count = data_mahasiswa.groupby(['nim']).size().reset_index(name='id_periode_count')
    merged_df = pd.merge(data_mahasiswa, first_nim, on='nim', how='left')
    merged_df['tahun_masuk'] = merged_df['first_id_periode'].astype(str).str[:4].astype(int)
    # Counting cuti & non-aktif
    inactive_status = merged_df[merged_df[
        'status_mahasiswa_semester'].isin(['cuti','non-aktif'])].groupby(
            'nim').size().reset_index(name='inactive_count')

    merged_df = pd.merge(merged_df, inactive_status, on='nim', how='left')
    merged_df['inactive_count'].fillna(0, inplace=True)

    merged_df = pd.merge(merged_df, id_periode_count, on='nim', how='left')
    merged_df['id_periode_count'].fillna(0, inplace=True)

    merged_df['masa_studi'] = merged_df['id_periode_count'] - merged_df['inactive_count']
    merged_df[['nim', 'masa_studi']]

    # Merge DataFrame sks_with_ipk & merged_df by nim
    merged_df_all = pd.merge(sks_with_ipk, merged_df[['nim', 'masa_studi']], on='nim', how='inner')
    df_mahasiswa = merged_df_all.drop_duplicates(subset=['nim'], ignore_index=True)

    return df_mahasiswa
```

Gambar 4.18 Penghitungan Masa Studi Data Prediksi

Setelah dilakukan inisialisasi, sama halnya dalam *data modeling*, dilakukan tahapan-tahapan seperti melakukan *preprocessing data* sehingga membentuk df ipk, sks, dan masa studi. Namun perbedaan terletak pada *field* df. Pada gambar diatas

menunjukkan bahwa pada bagian *merge* sks_with_ipk, tidak hanya atribut-atribut untuk prediksi saja yang tersedia, melainkan terdapat atribut tambahan yang dapat berguna untuk visualisasi data. Hal ini dilakukan agar proses lebih efisien.

```python
@dropout_bp.route('/get_data/<year>/', methods=['GET'])
def get_data_endpoint(year):
    histori_mahasiswa = get_histori_mahasiswa('')
    list_mahasiswa = get_list_mahasiswa('')

    data_s1_only = histori_mahasiswa[histori_mahasiswa['program_pendidikan'] == 'S1']

    data_mahasiswa = pd.merge(data_s1_only.drop(columns=['ips']),
                              list_mahasiswa[['nim', 'ipk', 'status_mahasiswa_terakhir', 'tahun_angkatan',
                                              'nama_program_studi']], on='nim', how='inner')
    df_pred = preprocess_data(data_mahasiswa)

    data_mahasiswa_json = df_pred.to_json(orient='records')

    file_path_predict = f"routes/dropout/data/predict_{year}.json"
    with open(file_path_predict, 'w') as f:
        f.write(data_mahasiswa_json)

    return jsonify(data_mahasiswa_json = data_mahasiswa_json)
```

Gambar 4.19 Drop Column Data Prediksi

Dalam data prediksi, setelah data melalui *function preprocessing*, selanjutnya dimasukkan kedalam *function* 'get_data_endpoint'. Data yang dihasilkan dari tahapan *preprocess* selanjutnya dimasukkan kedalam variabel df_pred dan disimpan ke dalam bentuk JSON. Data ini dapat digunakan baik untuk prediksi maupun visualisasi data.

4.3.2 *Data Transformation (Label Encoding)*

Pada tahapan ini dilakukan transformasi data berupa *label encoding.*

1. **Pemodelan Data**

```python
# Label Encoding
label_encoder = LabelEncoder()

for c in data_s1_only.columns:
    if data_s1_only[c].dtype == 'object':
        data_s1_only.loc[:, c] = label_encoder.fit_transform(data_s1_only[c].values)

data_s1_only
```

Gambar 4.20 Label Encoding

	nim	nama_mahasiswa	kode_program_studi	program_pendidikan	program_studi	ips	sks_semester	status_mahasiswa_semester	id_periode
0	1510411001	8655	411	0	8	3.95	20	0	20151
1	1510313001	18764	313	0	21	3.93	19	0	20151
2	1510611002	494	611	0	7	3.84	40	0	20151
3	1510714001	18314	714	0	5	3.59	20	0	20151
4	1510313002	21503	313	0	21	0.00	0	0	20151
...	...	...	...	...	...	...	...	...	...
175784	2310412235	13336	412	0	6	0.00	0	0	20232
175785	2310512174	5621	512	0	17	0.00	0	0	20232
175786	2310111279	11932	111	0	14	0.00	0	0	20232
175787	2310412236	1769	412	0	6	0.00	0	0	20232
175791	2310414080	16384	414	0	16	0.00	0	7	20232

146433 rows × 9 columns

Gambar 4.21 Hasil Label Encoding

Pada tahapan ini, setelah melakukan *grab* data S1 mahasiswa, langkah berikutnya adalah melakukan *encoding*. *Label encoding* bertujuan untuk mengonversi data kategori menjadi numerik. Dalam hal ini, untuk setiap atribut yang memiliki tipe data *object* akan dikonversi menjadi numerik.

4.3.3 *Feature Engineering*

Pada tahap ini dilakukan pembuatan atribut baru yaitu adalah pembuatan atribut status mahasiswa.

1. Pemodelan Data

```
# Inisialisasi kolom 'status_mahasiswa' dengan nilai 'prediksi dropout' secara default
df_mahasiswa['status_mahasiswa'] = 'terprediksi dropout'
```

Gambar 4.22 Inisialisasi Status Mahasiswa

Langkah pertama adalah melakukan inisialisasi pada df_mahasiswa sehingga memiliki nilai awal ‘terprediksi_dropout’.

```
# Cek kriteria untuk 'mahasiswa aktif' dan set nilai 'status_mahasiswa' berdasarkan kriteria
for index, row in df_mahasiswa.iterrows():
    if row['masa_studi'] == 2.0 and row['sks'] >= 19:
        df_mahasiswa.at[index, 'status_mahasiswa'] = 'tidak terprediksi dropout'

    elif row['masa_studi'] == 3.0 and row['sks'] >= 40:
        df_mahasiswa.at[index, 'status_mahasiswa'] = 'tidak terprediksi dropout'

    elif row['masa_studi'] == 4.0 and row['sks'] >= 40:
        df_mahasiswa.at[index, 'status_mahasiswa'] = 'tidak terprediksi dropout'

    elif row['masa_studi'] == 5.0 and row['sks'] >= 60:
        df_mahasiswa.at[index, 'status_mahasiswa'] = 'tidak terprediksi dropout'

    elif row['masa_studi'] == 6.0 and row['sks'] >= 60:
        df_mahasiswa.at[index, 'status_mahasiswa'] = 'tidak terprediksi dropout'

    elif row['masa_studi'] == 7.0 and row['sks'] >= 80:
        df_mahasiswa.at[index, 'status_mahasiswa'] = 'tidak terprediksi dropout'

    elif row['masa_studi'] == 8.0 and row['sks'] >= 80:
        df_mahasiswa.at[index, 'status_mahasiswa'] = 'tidak terprediksi dropout'

    elif row['masa_studi'] == 9.0 and row['sks'] >= 100:
        df_mahasiswa.at[index, 'status_mahasiswa'] = 'tidak terprediksi dropout'

    elif row['masa_studi'] == 10.0 and row['sks'] >= 100:
        df_mahasiswa.at[index, 'status_mahasiswa'] = 'tidak terprediksi dropout'

    elif row['masa_studi'] == 11.0 and row['sks'] >= 120:
        df_mahasiswa.at[index, 'status_mahasiswa'] = 'tidak terprediksi dropout'

    elif row['masa_studi'] == 12.0 and row['sks'] >= 120:
        df_mahasiswa.at[index, 'status_mahasiswa'] = 'tidak terprediksi dropout'

    elif row['masa_studi'] == 13.0 and row['sks'] >= 144:
        df_mahasiswa.at[index, 'status_mahasiswa'] = 'tidak terprediksi dropout'

    elif row['masa_studi'] == 14.0 and row['sks'] >= 144:
        df_mahasiswa.at[index, 'status_mahasiswa'] = 'tidak terprediksi dropout'
```

Gambar 4.23 Membuat Kriteria Masa Studi dan SKS

Kemudian membuat kriteria terkait masa studi dan sks. Apabila sesuai dengan ketentuan atau acuan maka bernilai ‘tidak terprediksi dropout’.

```
# Set nilai 'status_mahasiswa' berdasarkan kriteria dropout
criteria_dropout = (
    (df_mahasiswa['masa_studi'] >= 14.0) |
    (df_mahasiswa['ipk'] < 2.00) |
    (df_mahasiswa['masa_studi'] == 0.0) |
    (df_mahasiswa['sks'] == 0.0) |
    (df_mahasiswa['ipk'] == 0.00)
)

df_mahasiswa.loc[criteria_dropout, 'status_mahasiswa'] = 'terprediksi dropout'
df_mahasiswa[['nim', 'masa_studi', 'sks', 'ipk', 'status_mahasiswa']]
```

Gambar 4.24 Membuat Atribut Status Mahasiswa

Selanjutnya adalah melakukan pengecekkan kriteria *dropout.* Apabila df_mahasiswa sesuai dengan 'criteria_dropout', maka akan mengembalikan hasil 'terprediksi dropout'. Kemudian dilakukan pembuatan df yang akan menampilkan nim, masa studi, sks, ipk, dan status mahasiswa.

4.3.4 Data Scalling

Data scalling digunakan untuk mengubah fitur ke dalam skala tertentu, dan akan memastikan bahwa seluruh fitur memiliki skala yang sama. *Data scalling* memiliki tujuan untuk meningkatkan kinerja model. Salah satu algoritma *machine learning* yang berbasis jarak adalah Algoritma *KNN*. Fitur yang memiliki skala yang lebih besar akan mendominasi dari fitur lainnya sehingga dapat menyebabkan kesalahan dalam membuat keputusan.

1. **Pemodelan Data**

```
#scalling data
sc = StandardScaler()
X_train = sc.fit_transform(X_train)
X_test = sc.transform(X_test)
```

Gambar 4.25 Scalling Data Model

Dalam pemodelan data, setelah dilakukan inisialisasi *data training, data testing* dan proses *data splitting* selanjutnya adalah melakukan *scalling data* untuk *data training* dan *data* testing dengan menggunakan StandardScaler().

2. **Prediksi Data**

```
model = predict_model(f"routes/dropout/data/KNN_model_fix.pkl")
sc = StandardScaler()
data_predict = data_prediction[['masa_studi', 'sks', 'ipk']]
sc.fit(data_predict)
new_data_scaled = sc.transform(data_predict)
predicted_class = model.predict(new_data_scaled)
data_prediction['status_mahasiswa'] = predicted_class
```

Gambar 4.26 Scalling Data Prediksi

Sama halnya dalam pemodelan data, pada prediksi data setelah model disimpan dan diinisialisasikan dengan nama 'predict_model', kemudian model disimpan dalam *variable* model. Selanjutnya dilakukan inisisalisasi data_predict yang merupakan df yang sebelumnya sudah diinisialisasikan. Selanjutnya dilakukan penyesuaian skala terhadap data yang ingin di prediksi. Tahapan berikutnya dilakukan *scalling data* pada data baru. Setelah *scalling data* dilakukan, kemudian dilakukan prediksi data dengan menggunakan model yang sebelumnya telah diinisialisasikan.

4.4 Modeling

4.4.1 Data Preparation

```
X = df_mahasiswa[['masa_studi', 'sks', 'ipk']]
y = df_mahasiswa['status_mahasiswa']
```

Gambar 4.27 Membuat Variable Fitur dan Label

```
df_mahasiswa.info()
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 23420 entries, 0 to 23419
Data columns (total 5 columns):
 #   Column           Non-Null Count  Dtype
---  ------           --------------  -----
 0   nim              23420 non-null  int64
 1   masa_studi       23420 non-null  float64
 2   sks              23420 non-null  int64
 3   ipk              23420 non-null  float64
 4   status_mahasiswa  23420 non-null  int32
dtypes: float64(2), int32(1), int64(2)
memory usage: 823.5 KB
```

Gambar 4.28 Informasi Field DataFrame df_mahasiswa

Pada Gambar 4.27 dilakukan inisialisasi *data training* dan *data testing. Variable* 'X' berisi tentang fitur dataset yaitu masa studi, sks, dan ipk. Sedangkan variable 'y' berisi label atau target berupa status mahasiswa yang telah ditentukan pada tahap *feature engineering.*

Tabel 4.1 Menentukan Parameter k

Parameter k	*Mean Accuracy*
k = 4	0.9786 (97.86%)
k = 5	0.9900 (99.00%)
k = 6	0.9773 (97.73%)

Berdasarkan ke-3 percobaan pada tabel diatas, dapat disimpulkan bahwa, *mean accuracy* tertinggi terdapat pada parameter k=5, sehingga parameter k=5 digunakan sebagai acuan untuk melakukan *training* model.

```
#splitting data
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size = 0.2, random_state = 420)
```

Gambar 4.29 Splitting Data Rasio (80:20)

```
#splitting data
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size = 0.3, random_state = 420)
```

Gambar 4.30 Splitting Data Rasio (70:30)

Pada gambar diatas, dilakukan dua kali percobaan dalam melakukan *splitting data* dengan penjelasan sebagai berikut.

Tabel 4.2 Perbandingan Rasio

Perbandingan	***Accuracy***
Rasio 80:20	0.9977 (99.77%)
Rasio 70:30	0.9974 (99.74%)

Berdasarkan perbandingan diatas, dengan perbedaan yang tidak terlalu signifikan, rasio (80:20) memiliki *accuracy* yang lebih tinggi dibandingkan dengan rasio (70:30), sehingga rasio (80:20) dijadikan acuan untuk melakukan *training model* agar mendapatkan hasil yang maksimal. Setelah dilakukan *splitting data*, selanjutnya adalah melakukan *scalling data* dan uji *metric* untuk memperoleh hasil yang maksimal.

Tabel 4.3 Perbandingan Metric

Metric	***Accuracy***
Manhattan Distance	0.9977 (99.97%)
Chebyshev Distance	0.9974 (99.74%)
Minkowski p = 2 Distance / Euclidean Distance	0.9977 (99.77%)

Tabel diatas merupakan uji *metric* yang dilakukan pada tiga jenis *metric* yaitu *Manhattan Distance*, *Chebyshev metric*, dan *Minkowski p = 2* atau *Euclidean Distance*. Pada tabel diatas, penggunaan *metric Manhattan Distance* dan *Euclidean Distance* memiliki hasil *accuracy* yang sama yaitu pada 0.9977, sedangkan untuk *metric Chebyshev* memiliki nilai *accuracy* 0.9974. Pemilihan *Euclidean Distance* atau setara dengan *minkowski p = 2*, sebagai atribut dilakukan karena data memiliki fitur yang hanya terbatas pada tiga atribut dan *Euclidean Distance* merupakan *metric* yang paling umum digunakan pada algoritma KNN.

4.4.2 Modeling K-Nearest Neighbor

```
#training model
classifier = KNeighborsClassifier(n_neighbors = 5, metric = 'minkowski', p = 2)
classifier.fit(X_train, y_train)
```

Gambar 4.31 Training Model KNN

Gambar diatas merupakan proses *training* model pada KNN dengan menggunakan *library* KneighborsClassifier dengan parameter k=5, metric=’minkowski’, dan p = 2.

4.4.3 Evaluasi Model K-Nearest Neighbor

Tabel 4.4 Confusion Matrix Data Training KNN

		Predicted Label	
		Terprediksi *Dropout* (0)	Tidak Terprediksi *Dorpout* (1)
Actual Label	Terprediksi *Dropout* (0)	1518 (TN)	35 (FP)
	Tidak Terprediksi *Dorpout* (1)	7 (FN)	17176 (TP)

Keterangan:

1. True Positives (TP): Jumlah contoh yang benar-benar "Tidak Terprediksi Dropout (1)" dan diklasifikasikan dengan benar sebagai "Tidak Terprediksi Dropout (1)" bernilai 17176.
2. True Negatives (TN): Jumlah contoh yang benar-benar "Terprediksi Dropout (0)" dan diklasifikasikan dengan benar sebagai "Terprediksi

Dropout (0)" bernilai 1518.

3. False Positives (FP): Jumlah contoh yang benar-benar "Terprediksi Dropout (0)" tetapi salah diklasifikasikan sebagai "Tidak Terprediksi Dropout (1)" bernilai 35.
4. False Negatives (FN): Jumlah contoh yang benar-benar "Tidak Terprediksi Dropout (1)" tetapi salah diklasifikasikan sebagai "Terprediksi Dropout (0)" bernilai 7.

- *Accuracy*

$$Accuracy = \frac{17176 + 1518}{17176 + 1518 + 35 + 7} = 0.9978$$

- *Precision*

$$Precision = \frac{17176}{17176 + 35} = 0.9980$$

- *Recall*

$$Recall = \frac{17176}{17176 + 7} = 0.9995$$

- *F1-Score*

$$F1 - Score = 2 \times \frac{0.9980 \times 0.9995}{0.9980 + 0.9995} = 0.9988$$

Tabel 4.5 Confusion Matrix Data Testing KNN

		Predicted Label	
		Terprediksi *Dropout* (0)	Tidak Terprediksi *Dorpout* (1)
Actual Label	Terprediksi *Dropout* (0)	400 (TN)	6 (FP)
	Tidak Terprediksi *Dorpout* (1)	4 (FN)	4274 (TP)

Keterangan:

1. True Positives (TP): Jumlah contoh yang benar-benar "Tidak Terprediksi Dropout (1)" dan diklasifikasikan dengan benar sebagai "Tidak Terprediksi Dropout (1)" bernilai 4274.

2. True Negatives (TN): Jumlah contoh yang benar-benar "Terprediksi Dropout (0)" dan diklasifikasikan dengan benar sebagai "Terprediksi Dropout (0)" bernilai 400.
3. False Positives (FP): Jumlah contoh yang benar-benar "Terprediksi Dropout (0)" tetapi salah diklasifikasikan sebagai "Tidak Terprediksi Dropout (1)" bernilai 6.
4. False Negatives (FN): Jumlah contoh yang benar-benar "Tidak Terprediksi Dropout (1)" tetapi salah diklasifikasikan sebagai "Terprediksi Dropout (0)" bernilai 4.

- *Accuracy*

$$Accuracy = \frac{4274 + 400}{4274 + 400 + 6 + 4} = 0.9978$$

- *Precision*

$$Precision = \frac{4274}{4274 + 6} = 0.9985$$

- *Recall*

$$Recall = \frac{4274}{4274 + 4} = 0.9990$$

- *F1-Score*

$$F1 - Score = 2 \times \frac{0.9985 \times 0.9990}{0.9985 + 0.9990} = 0.9988$$

4.4.4 Modeling Naïve Bayes

```
gnb = GaussianNB()
gnb.fit(X_train, y_train.ravel())
```

Gambar 4.32 Training Model Naive Bayes

Gambar 4.32 merupakan proses *training* pada algoritma Naïve Bayes menggunakan *library* Gaussian Naïve Bayes.

4.4.5 Evaluasi Model Naïve Bayes

Tabel 4.6 Confusion Matrix Data Training Naive Bayes

		Predicted Label	
		Terprediksi *Dropout* (0)	Tidak Terprediksi *Dorpout* (1)
Actual Label	Terprediksi *Dropout* (0)	1091 (TN)	462 (FP)

	Tidak Terprediksi *Dorpout* (1)	274 (FN)	16909 (TP)

Keterangan:

1. True Positives (TP): Jumlah contoh yang benar-benar "Tidak Terprediksi Dropout (1)" dan diklasifikasikan dengan benar sebagai "Tidak Terprediksi Dropout (1)" bernilai 16909.
2. True Negatives (TN): Jumlah contoh yang benar-benar "Terprediksi Dropout (0)" dan diklasifikasikan dengan benar sebagai "Terprediksi Dropout (0)" bernilai 1091.
3. False Positives (FP): Jumlah contoh yang benar-benar "Terprediksi Dropout (0)" tetapi salah diklasifikasikan sebagai "Tidak Terprediksi Dropout (1)" bernilai 462.
4. False Negatives (FN): Jumlah contoh yang benar-benar "Tidak Terprediksi Dropout (1)" tetapi salah diklasifikasikan sebagai "Terprediksi Dropout (0)" bernilai 274.

- *Accuracy*

$$Accuracy = \frac{16909 + 1091}{16909 + 1091 + 462 + 274} = 0.9658$$

- *Precision*

$$Precision = \frac{16909}{16909 + 462} = 0.9734$$

- *Recall*

$$Recall = \frac{16909}{16909 + 274} = 0.9840$$

- *F1-Score*

$$F1 - Score = 2 \times \frac{0.9734 \times 0.9840}{0.9734 + 0.9840} = 0.9787$$

Tabel 4.7 Confusion Matrix Data Testing Naive Bayes

		Predicted Label	
		Terprediksi *Dropout* (0)	Tidak Terprediksi *Dorpout* (1)
Actual	Terprediksi	298 (TN)	108 (FP)

Label	*Dropout* (0)		
	Tidak Terprediksi *Dorpout* (1)	64 (FN)	4214 (TP)

Keterangan:

1. True Positives (TP): Jumlah contoh yang benar-benar "Tidak Terprediksi Dropout (1)" dan diklasifikasikan dengan benar sebagai "Tidak Terprediksi Dropout (1)" bernilai 4214.
2. True Negatives (TN): Jumlah contoh yang benar-benar "Terprediksi Dropout (0)" dan diklasifikasikan dengan benar sebagai "Terprediksi Dropout (0)" bernilai 298.
3. False Positives (FP): Jumlah contoh yang benar-benar "Terprediksi Dropout (0)" tetapi salah diklasifikasikan sebagai "Tidak Terprediksi Dropout (1)" bernilai 108.
4. False Negatives (FN): Jumlah contoh yang benar-benar "Tidak Terprediksi Dropout (1)" tetapi salah diklasifikasikan sebagai "Terprediksi Dropout (0)" bernilai 64.

- *Accuracy*

$$Accuracy = \frac{4214 + 298}{4214 + 298 + 108 + 64} = 0.9633$$

- *Precision*

$$Precision = \frac{4214}{4214 + 108} = 0.9750$$

- *Recall*

$$Recall = \frac{4214}{4214 + 64} = 0.9849$$

- *F1-Score*

$$F1 - Score = 2 \times \frac{0.9750 \times 0.9849}{0.9750 + 0.9849} = 0.9799$$

4.5 Pemilihan Model Terbaik

Tabel 4.8 Pemilihan Model Terbaik

Algoritma	Data	Evaluasi Model
KNN	*Training*	*Accuracy*: 0.9978
		Precision: 0.9980
		Recall: 0.9995
		F1-Score: 0.9988
	Testing	*Accuracy*: 0.9978
		Precision: 0.9985
		Recall: 0.9990
		F1-Score: 0.9988
Naïve Bayes	*Training*	*Accuracy*: 0.9658
		Precision: 0.9734
		Recall: 0.9840
		F1-Score: 0.9787
	Testing	*Accuracy*: 0.9633
		Precision: 0.9750
		Recall: 0.9849
		F1-Score: 0.9799

Berdasarkan Tabel 4.14, algoritma KNN memiliki performa yang lebih baik dibandingkan dengan algoritma Naïve Bayes. Dengan demikian, model yang digunakan adalah KNN.

4.7 Perancangan dan Pengujian Sistem

4.7.1 Analisis Kebutuhan

Pada tahapan ini, terdapat kebutuhan fungsional maupun kebutuhan non-fungsional seperti yang dijelaskan dibawah ini.

- Kebutuhan Fungsional
    1. Sistem harus dapat menarik data secara periodik dari API dan data tersimpan dengan baik.
    2. Sistem harus dapat memproses data yang diambil dan dilakukan *preprocessing.*
    3. Sistem harus dapat mendukung untuk *training* model *machine learning.*
    4. Sistem harus dapat menyimpan model *machine learning* yang telah dilatih.

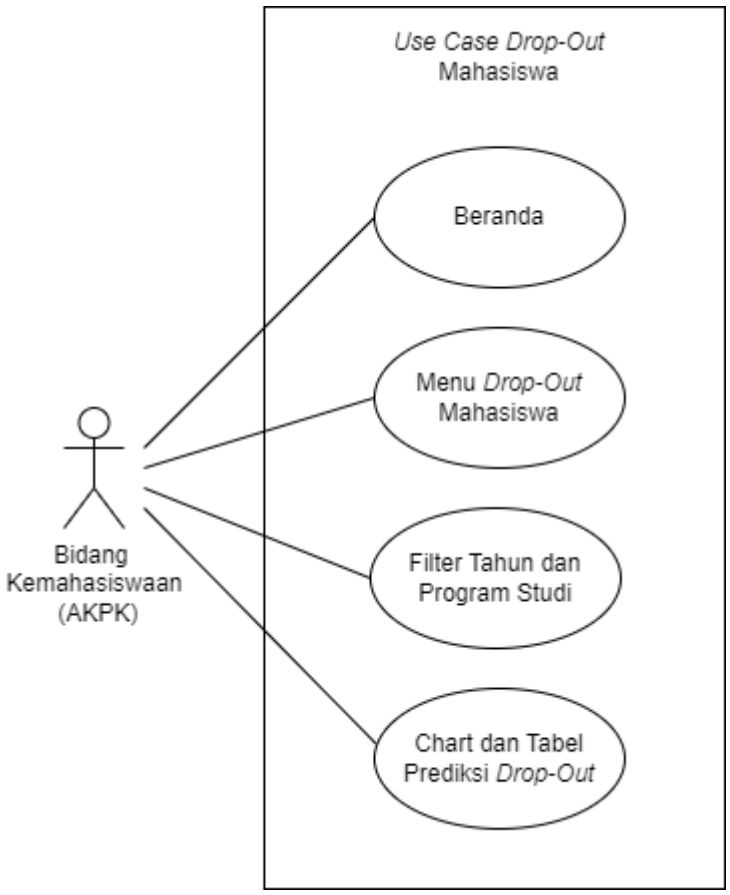
5. Sistem harus dapat melakukan prediksi berdasarkan model yang telah dilatih.
6. Sistem harus dapat menyediakan visualisasi yang interaktif.

- Kebutuhan Non-Fungsional
    1. Sistem operasi menggunakan Windows 11.
    2. Bahasa pemrograman yang digunakan untuk *backend* yaitu *python* dengan menggunakan *framework Flask.* Sedangkan untuk *frontend* menggunakan *HTML*, *CSS*, dan *Javascript.*
    3. Data bersumber dari API eksternal dan difasilitasi dengan *'requests'.*
    4. *Library* yang digunakan meliputi: *numpy*, *pandas, matplotlib,* dan *scikit-learn* yang digunakan untuk berbagai macam proses.
    5. *Tools* yang digunakan untuk visualisasi data adalah *ApexCharts.js.*

4.7.2 Sistem Desain

Dalam tahapan ini, dilakukan perancangan UML yaitu *Use Case Diagram*, *Activity diagram*, dan *Class Diagram* sebagai perancangan sistem, kemudian dilakukan perancangan *wireframe* sebagai perancangan *user interface.*

- *Use Case Diagram*



Gambar 4.33 Use Case Dropout Mahasiswa

Use Case Diagram Scenario

Tabel 4.9 Use Case Beranda

Use Case	**Beranda**
Aktor	Bidang Kemahasiswaan (AKPK)
Kondisi Awal	Pengguna mengakses halaman web
Kondisi Akhir	Pengguna melihat tampilan halaman beranda
Deskripsi	Mengakses halaman beranda. Halaman beranda berisi informasi mengenai dashboard terkait
Scenario	1. Pengguna mengakses halaman web 2. Sistem menampilkan halaman beranda

Tabel 4.10 Use Case Menu Dropout Mahasiswa

Use Case	**Menu *Drop-out* Mahasiswa**
Aktor	Bidang Kemahasiswaan (AKPK)
Kondisi Awal	Pengguna mengakses menu *drop-out* mahasiswa
Kondisi Akhir	Pengguna melihat tampilan halaman *dropout* mahasiswa
Deskripsi	Mengakses halaman *drop-out* mahasiswa. Halaman *drop-out* mahasiswa berisi diagram-diagram informasi data mahasiswa
Scenario	1. Pengguna mengakses halaman *drop-out* mahasiswa 2. Sistem menampilkan halaman *drop-out* mahasiswa

Tabel 4.11 Use Case Filter Tahun dan Program Studi

Use Case	**Filter Tahun dan Program Studi**
Aktor	Bidang Kemahasiswaan (AKPK)
Kondisi Awal	Pengguna melakukan *click button* filter tahun dan program studi
Kondisi Akhir	Pengguna melihat data-data mahasiswa yang telah di filter
Deskripsi	Menggunakan filter untuk mempermudah pencarian data
Scenario	1. Pada halaman menu *drop-out* mahasiswa, pengguna melakukan - *click button* pada filter 2. Sistem menampilkan visualisasi tabel hasil prediksi

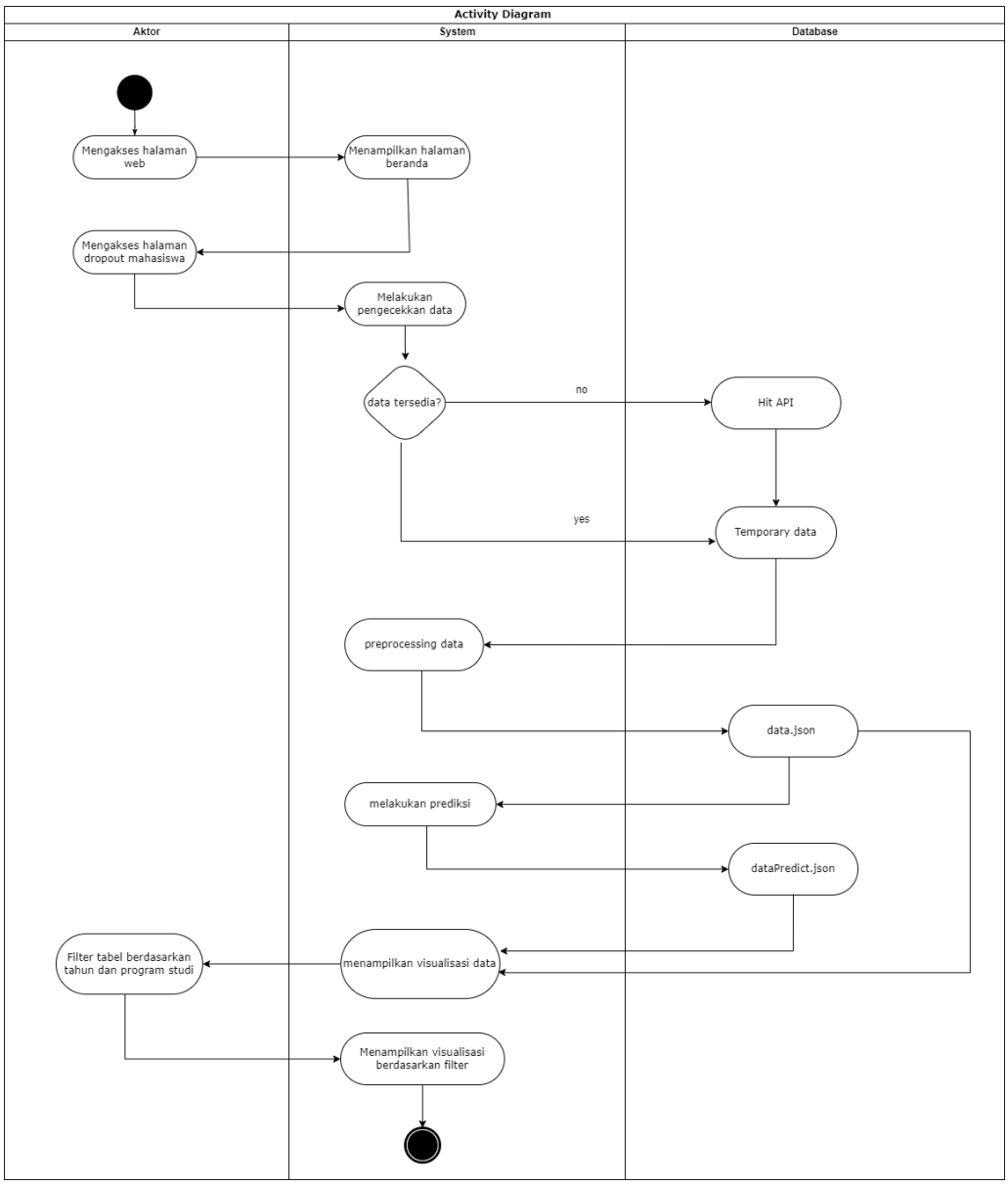
Tabel 4.12 Use case Chart dan Tabel Prediksi Dropout

Use Case	**Chart dan Tabel Prediksi *Dropout***
Aktor	Bidang Kemahasiswaan (AKPK)
Kondisi Awal	Pengguna mengakses halaman *dropout* mahasiswa
Kondisi Akhir	Pengguna melihat daigram-diagram dan hasil prediksi
Deskripsi	Pengguna mengakses dan melakukan eksplorasi pada halaman *dropout* mahasiswa
Scenario	1. Pengguna mengakses halaman *dropout* mahasiswa 2. Sistem menampikan halaman *dropout* mahasiswa yang berisikan diagram-diagram informatif terkait data prediksi mahasiswa teridentifikasi *dropout* dan informasi penunjang lainnya.

- *Activity Diagram*

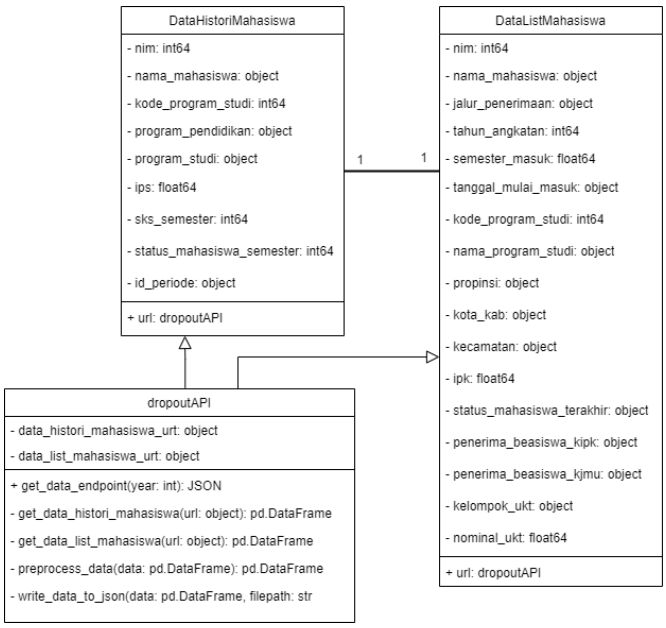
Activity diagram dibawah ini menjelaskan mengenai alur pada sistem yang dibangun. *Acticity diagram* pada penelitian ini menggambarkan analisis proses serta menjelaskan mengenai langkah-langkah serta keputusan yang diambil.

Tabel 4.13 Activity Diagram Dropout Mahasiswa



- *Class Diagram*

  Class Diagram menggambarkan struktur dan hubungan antara kelas-kelas dalam sebuah sistem. *Class Diagram* memberikan pandangan mengenai bagaimana kelas-kelas saling terkait, atribut-atribut yang dimiliki oleh setiap kelas, serta metode yang dimiliki oleh setiap kelas tersebut.



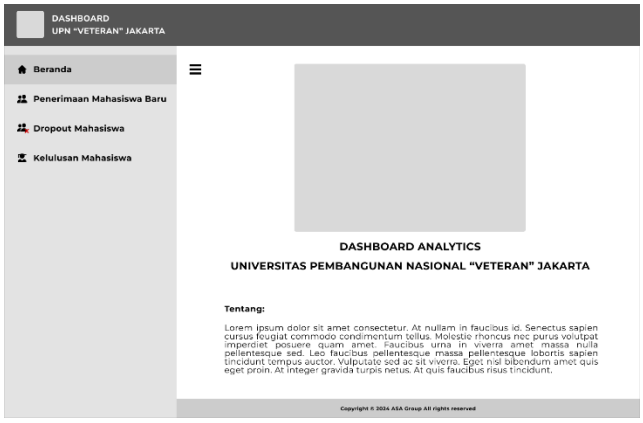
Gambar 4.34 Class Diagram Dropout Mahasiswa

Keterangan:

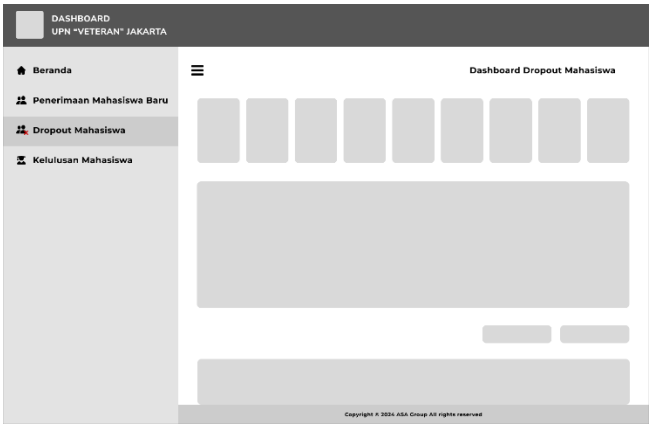
- *dropoutAPI* merupakan kelas utama yang mengelola interaksi dengan API untuk mengambil data histori mahasiswa dan data *list* mahasiswa.
- *data_histori_mahasiswa_url* merupakan url untuk *endpoint* data histori mahasiswa
- *data_list_mahasiswa_url* merupakan url untuk *endpoint* data list mahasiswa
- *get_data_endpoint(year: int)* merupakan metode untuk mengambil data histori mahasiswa dan data list mahasiswa untuk tahun tertentu, menggabungkannya, dan mengembalikan hasilnya dalam format JSON.
- *get_histori_mahasiswa()* merupakakn metode internal untuk mengambil data histori mahasiswa dari *data_histori_mahasiswa* dan mengembalikan data dalam bentuk Pandas DataFrame.
- *get_list_mahasiswa()* merupakan metode internal untuk mengambil data daftar mahasiswa dari *data_list_mahasiswa* dan mengembalikan data dalam bentuk Pandas DataFrame.

- *preprocess_data(data: pd.DataFrame)* merupakan metode untuk melakukan pra-pemrosesan data mahasiswa yang diperlukan sebelum penyimpanan atau penggunaan lebih lanjut.
- *write_data_to_json(data: pd.DataFrame, file_path: str) merupakan* metode untuk menulis data yang telah diproses ke dalam file JSON di lokasi yang ditentukan.

- Rancangan *Wireframe*



Gambar 4.35 Rancangan Wireframe Menu Beranda

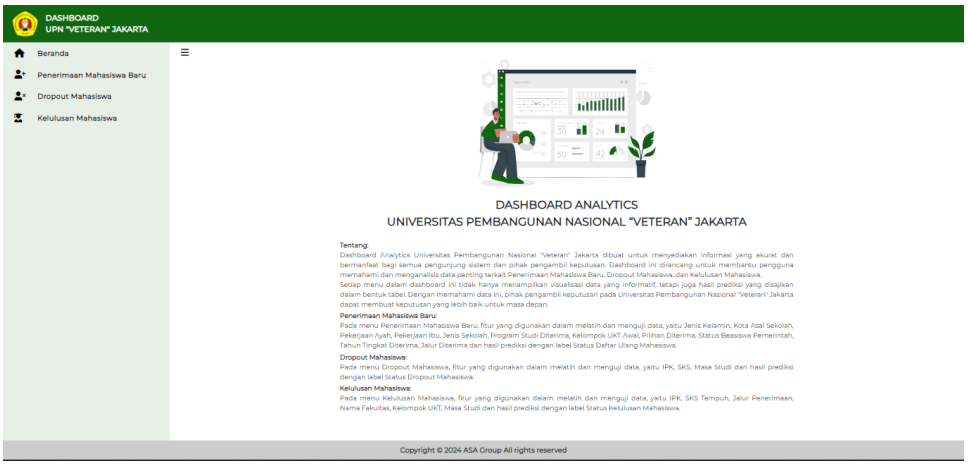


Gambar 4.36 Rancangan Wireframe Menu Dropout Mahasiswa

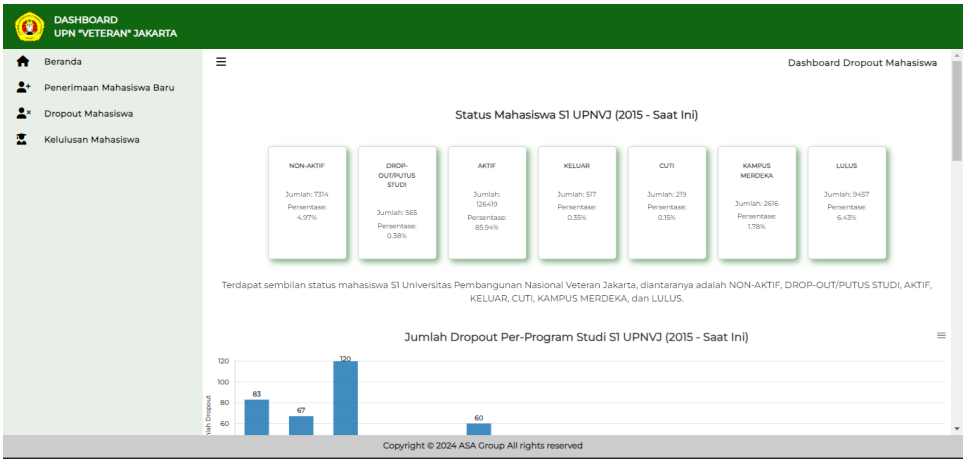
4.7.3 Implementasi

Pada tahap ini, dilakukan implementasi perancangan sistem. Setelah melakukan pemilihan model terbaik, langkah berikutnya adalah mengimplementasikan model pada sistem *website* yang dibangun. Pada perancangan sistem web bagian *frontend*, sistem menggunakan *HTML*,

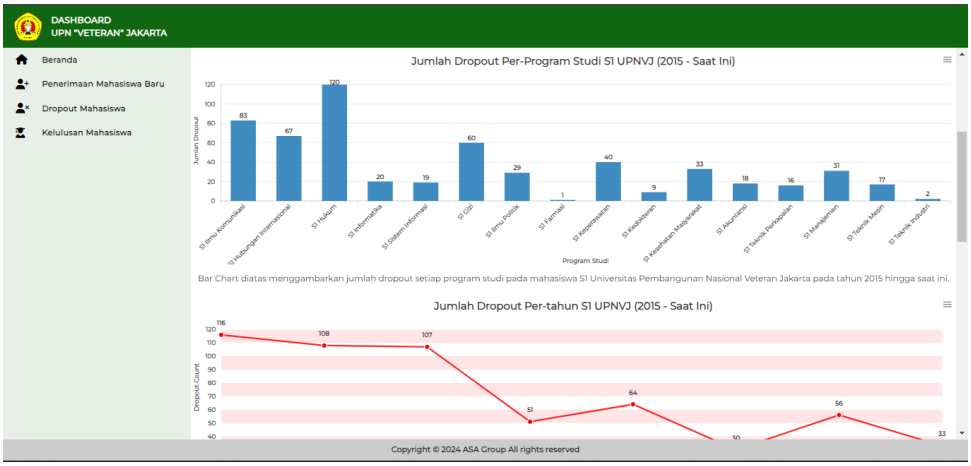
CSS, dan *Javascript*. Kemudian pada bagian *backend*, menggunakan *Flask*. Berikut ini merupakan hasil dari implementasi perancangan sistem berbasis *website*.



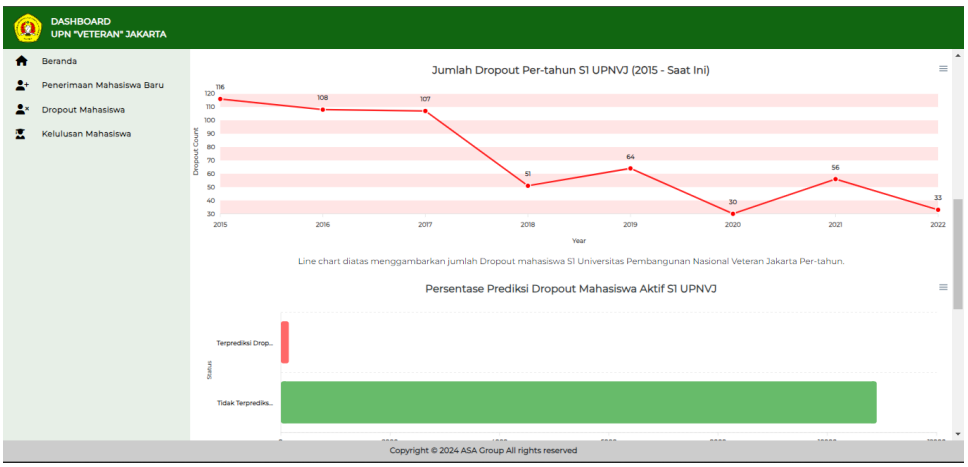
Gambar 4.37 User Interface Menu Beranda



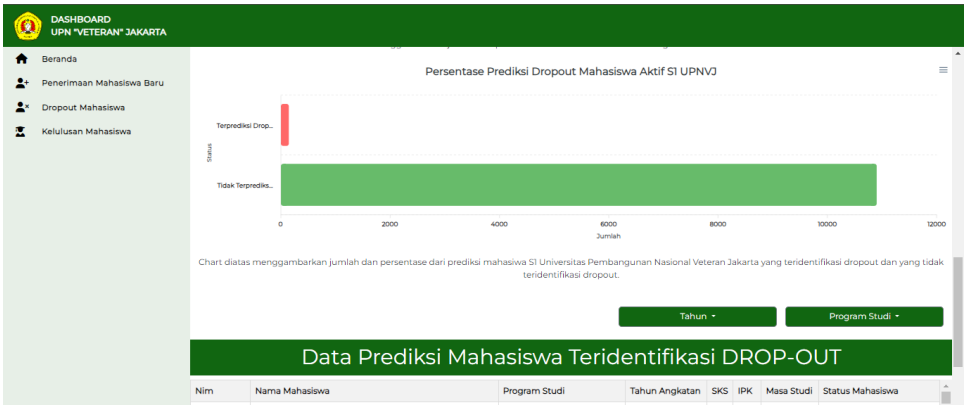
Gambar 4.38 User Interface Menu Dropout Mahasiswa (1)



Gambar 4.39 User Interface Menu Dropout Mahasiswa (2)



Gambar 4.40 User Interface Menu Dropout Mahasiswa (3)



Gambar 4.41 User Interface Menu Dropout Mahasiswa (4)

Berdasarkan hasil penelitian, jumlah mahasiswa terprediksi *dropout* untuk jenjang S1 sebanyak 144 orang pada tahun ajaran 2023/2024. Sementara mahasiswa tidak terprediksi *dropout* berjumlah 10.902 orang dengan presentase 98.70%.

4.7.4 Pengujian Sistem

Pada tahap ini, pengujian sistem dilakukan menggunakan *Black Box Testing.*

Tabel 4.14 Black Box Testing

Aktivitas	Skenario	*Output* yang diharapkan	Kesimpulan
Akses halaman Beranda	Mengakses halaman melalui *URL*	*View* halaman beranda	Berhasil

Akses halaman *dropout* mahasiswa	Mengakses halaman *dropout* mahasiswa	*View* halaman *dropout* mahasiswa berisi diagram – diagram informasi mahasiswa	Berhasil
Click button dropdown filter	Melakukan klik *button*	Data mahasiswa teridentifikasi *dropout* berdasarkan filter	Berhasil

4.7.5 Penerapan

Pada tahap ini, setelah pengujian sistem berhasil dilakukan, sistem dapat diterapkan kepada pengguna

4.7.6 Pemeliharaan

Tahap pemeliharaan dan pembaruan sistem merupakan tahap terakhir dari alur proses pada sistem perancangan dan pengujian sistem yang terdapat pada penelitian ini. Dalam tahap ini, perlu dilakukan *maintenance system* serta *update system* berdasarkan kebutuhan seperti pengecekkan *system error* maupun pembaruan *machine learning model* maupun pembaruan konten visualisasi.