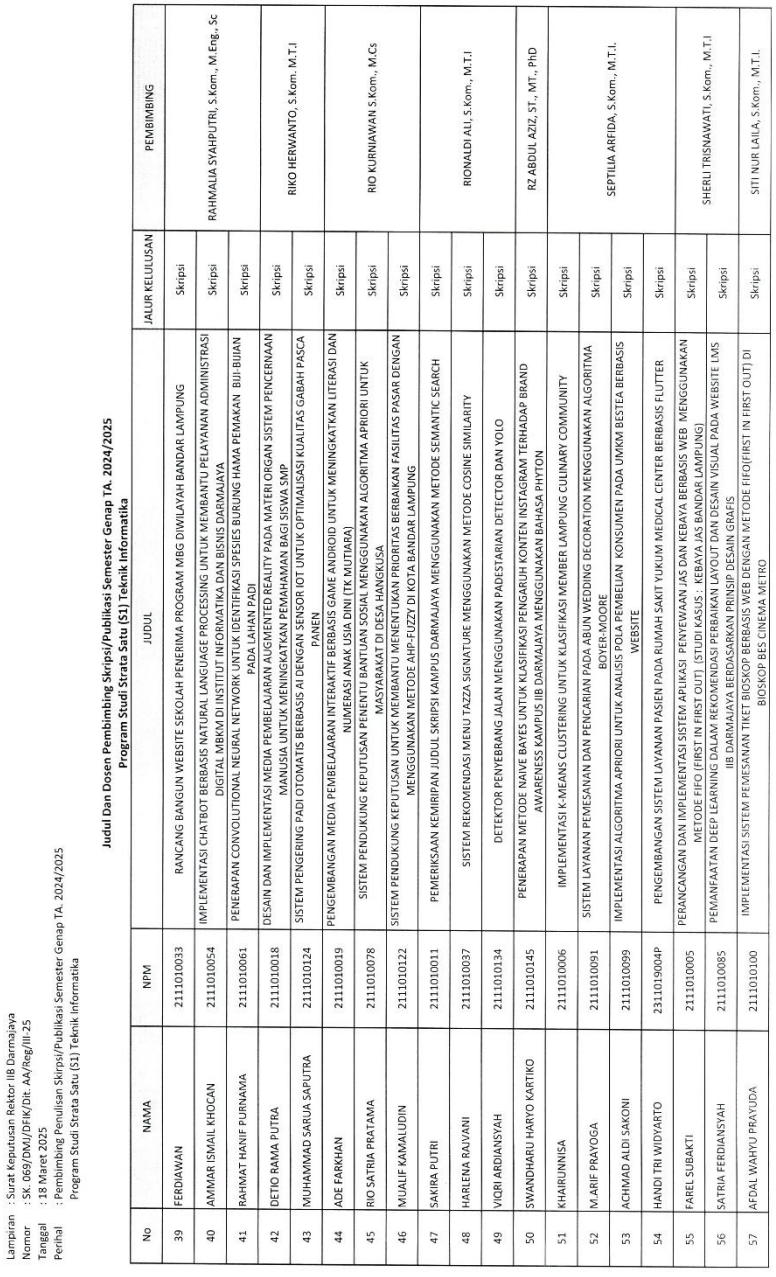
LAMPIRAN

Lampiran 1

Surat keputusan



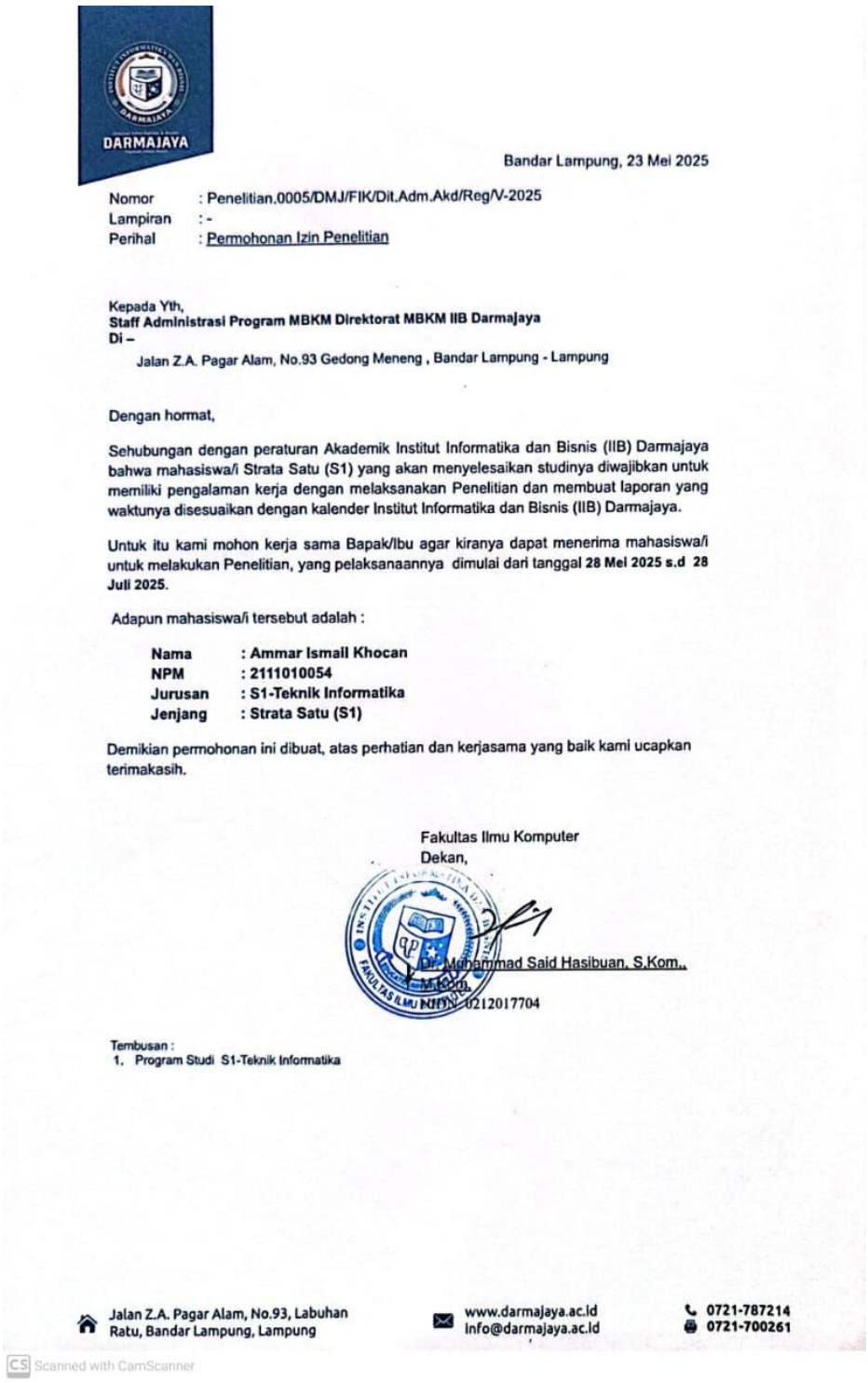
Lampiran : Surat Keputusan Rektor IIB Darmajaya
Nomor : SK. 069/DMJ/DFIK/Dit. AA/Reg/III 25
Tanggal : 18 Maret 2025
Perihal : Pembimbing Penulisan Skripsi/Publikasi Semester Genap TA. 2024/2025
Program Studi Strata Satu (S1) Teknik Informatika

Judul Dan Dosen Pembimbing Skripsi/Publikasi Semester Genap TA. 2024/2025
Program Studi Strata Satu (S1) Teknik Informatika

No	NAMA	NPM	JUDUL	JALUR KELULUSAN	PEMBIMBING
39	FERDIAWAN	2111010033	RANCANG BANGUN WEBSITE SEKOLAH PENERIMA PROGRAM MBG DIWILAYAH BANDAR LAMPUNG	Skripsi	RAHMALIA SYAHPUTRI, S.Kom., M.Eng., Sc
40	AMMAR ISMAIL KHOCAN	2111010054	IMPLEMENTASI CHATBOT BERBASIS NATURAL LANGUAGE PROCESSING UNTUK MEMBANTU PELAYANAN ADMINISTRASI DIGITAL MBKM DI INSTITUT INFORMATIKA DAN BISNIS DARMAJAYA	Skripsi	
41	RAHMAT HANIF PURNAMA	2111010061	PENERAPAN CONVOLUTIONAL NEURAL NETWORK UNTUK IDENTIFIKASI SPESIES BURUNG HAMA PEMAKAN BIJI-BIJIAN PADA LAHAN PADI	Skripsi	
42	DETIO RAMA PUTRA	2111010018	DESAIN DAN IMPLEMENTASI MEDIA PEMBELAJARAN AUGMENTED REALITY PADA MATERI ORGAN SISTEM PENCERNAAN MANUSIA UNTUK MENINGKATKAN PEMAHAMAN BAGI SISWA SMP	Skripsi	RIKO HERWANTO, S.Kom. M.T.I
43	MUHAMMAD SARUA SAPUTRA	2111010124	SISTEM PENGERING PADI OTOMATIS BERBASIS AI DENGAN SENSOR IOT UNTUK OPTIMALISASI KUALITAS GABAH PASCA PANEN	Skripsi	
44	ADE FARKHAN	2111010019	PENGEMBANGAN MEDIA PEMBELAJARAN INTERAKTIF BERBASIS GAME ANDROID UNTUK MENINGKATKAN LITERASI DAN NUMERASI ANAK USIA DINI (TK MUTIARA)	Skripsi	RIO KURNIAWAN S.Kom., M.Cs
45	RIO SATRIA PRATAMA	2111010078	SISTEM PENDUKUNG KEPUTUSAN PENENTU BANTUAN SOSIAL MENGGUNAKAN ALGORITMA APRIORI UNTUK MASYARAKAT DI DESA HANGKUSA	Skripsi	
46	MUALIF KAMALUDIN	2111010122	SISTEM PENDUKUNG KEPUTUSAN UNTUK MEMBANTU MENENTUKAN PRIORITAS BERBAIKAN FASILITAS PASAR DENGAN MENGGUNAKAN METODE AHP-FUZZY DI KOTA BANDAR LAMPUNG	Skripsi	
47	SAKIRA PUTRI	2111010011	PEMERIKSAAN KEMIRIPAN JUDUL SKRIPSI KAMPUS DARMAJAYA MENGGUNAKAN METODE SEMANTIC SEARCH	Skripsi	RIONALDI ALI, S.Kom., M.T.I
48	HARLENA RAUVANI	2111010037	SISTEM REKOMENDASI MENU TAZZA SIGNATURE MENGGUNAKAN METODE COSINE SIMILARITY	Skripsi	
49	VIQBI ARDIANSYAH	2111010134	DETEKTOR PENYEBRANG JALAN MENGGUNAKAN PADESTARIAN DETECTOR DAN YOLO	Skripsi	
50	SWANDHARU HARYO KARTIKO	2111010145	PENERAPAN METODE NAIVE BAYES UNTUK KLASIFIKASI PENGARUH KONTEN INSTAGRAM TERHADAP BRAND AWARENESS KAMPUS IIB DARMAJAYA MENGGUNAKAN BAHASA PHYTON	Skripsi	RZ ABDUL AZIZ, ST., MT., PhD
51	KHAIRUNNISA	2111010006	IMPLEMENTASI K-MEANS CLUSTERING UNTUK KLASIFIKASI MEMBER LAMPUNG CULINARY COMMUNITY	Skripsi	SEPTILIA ARFIDA, S.Kom., M.T.I.
52	M.ARIF PRAYOGA	2111010091	SISTEM LAYANAN PEMESANAN DAN PENCARIAN PADA ABUN WEDDING DECORATION MENGGUNAKAN ALGORITMA BOYER-MOORE	Skripsi	
53	ACHMAD ALDI SAKONI	2111010099	IMPLEMENTASI ALGORITMA APRIORI UNTUK ANALISIS POLA PEMBELIAN KONSUMEN PADA UMKM BESTEA BERBASIS WEBSITE	Skripsi	
54	HANDI TRI WIDYARTO	2311019004P	PENGEMBANGAN SISTEM LAYANAN PASIEN PADA RUMAH SAKIT YUKUM MEDICAL CENTER BERBASIS FLUTTER	Skripsi	
55	FAREL SUBAKTI	2111010005	PERANCANGAN DAN IMPLEMENTASI SISTEM APLIKASI PENYEWAAN JAS DAN KEBAYA BERBASIS WEB MENGGUNAKAN METODE FIFO (FIRST IN FIRST OUT) (STUDI KASUS : KEBAYA JAS BANDAR LAMPUNG)	Skripsi	SHERLI TRISNAWATI, S.Kom., M.T.I
56	SATRIA FERDIANSYAH	2111010085	PEMANFAATAN DEEP LEARNING DALAM REKOMENDASI PERBAIKAN LAYOUT DAN DESAIN VISUAL PADA WEBSITE LMS IIB DARMAJAYA BERDASARKAN PRINSIP DESAIN GRAFIS	Skripsi	
57	AFDAL WAHYU PRAYUDA	2111010100	IMPLEMENTASI SISTEM PEMESANAN TIKET BIOSKOP BERBASIS WEB DENGAN METODE FIFO(FIRST IN FIRST OUT) DI BIOSKOP BES CINEMA METRO	Skripsi	SITI NUR LAILA, S.Kom., M.T.I.

Lampiran 1

Surat Permohonan Izin Penelitian



DARMAJAYA

Bandar Lampung, 23 Mei 2025

Nomor : Penelitian.0005/DMJ/FIK/Dit.Adm.Akd/Reg/V-2025
Lampiran : -
Perihal : Permohonan Izin Penelitian

Kepada Yth,
Staff Administrasi Program MBKM Direktorat MBKM IIB Darmajaya
Di –
Jalan Z.A. Pagar Alam, No.93 Gedong Meneng , Bandar Lampung - Lampung

Dengan hormat,

Sehubungan dengan peraturan Akademik Institut Informatika dan Bisnis (IIB) Darmajaya bahwa mahasiswa/i Strata Satu (S1) yang akan menyelesaikan studinya diwajibkan untuk memiliki pengalaman kerja dengan melaksanakan Penelitian dan membuat laporan yang waktunya disesuaikan dengan kalender Institut Informatika dan Bisnis (IIB) Darmajaya.

Untuk itu kami mohon kerja sama Bapak/Ibu agar kiranya dapat menerima mahasiswa/i untuk melakukan Penelitian, yang pelaksanaannya dimulai dari tanggal **28 Mei 2025 s.d 28 Juli 2025**.

Adapun mahasiswa/i tersebut adalah :

Nama : Ammar Ismail Khocan
NPM : 2111010054
Jurusan : S1-Teknik Informatika
Jenjang : Strata Satu (S1)

Demikian permohonan ini dibuat, atas perhatian dan kerjasama yang baik kami ucapkan terimakasih.

Fakultas Ilmu Komputer
Dekan,

Dr. Muhammad Said Hasibuan, S.Kom., M.Kom.
NIDN. 0212017704

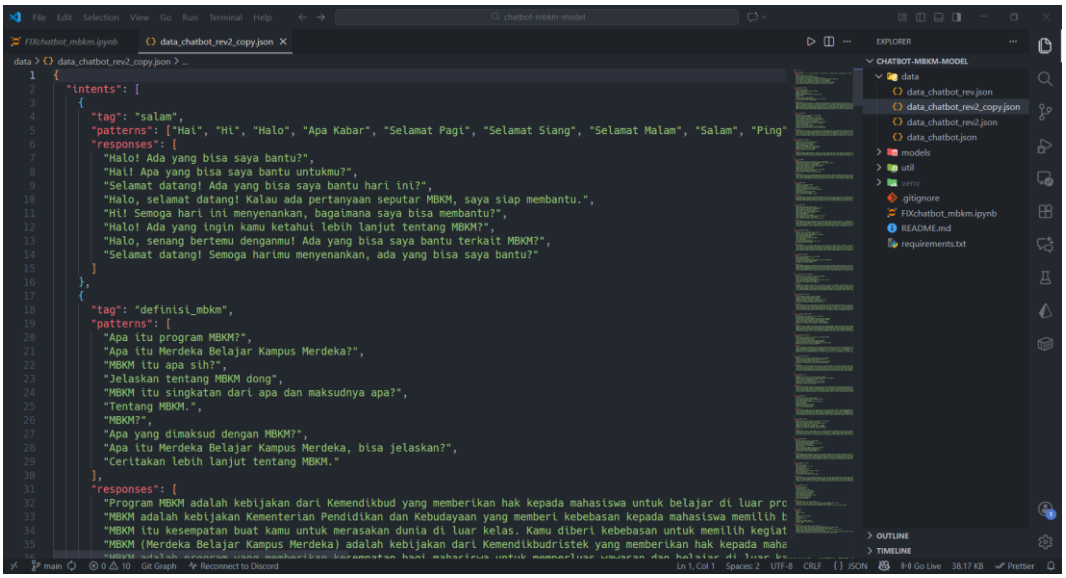
Tembusan :
1. Program Studi S1-Teknik Informatika

Jalan Z.A. Pagar Alam, No.93, Labuhan Ratu, Bandar Lampung, Lampung
www.darmajaya.ac.id
Info@darmajaya.ac.id
0721-787214
0721-700261

Lampiran 2

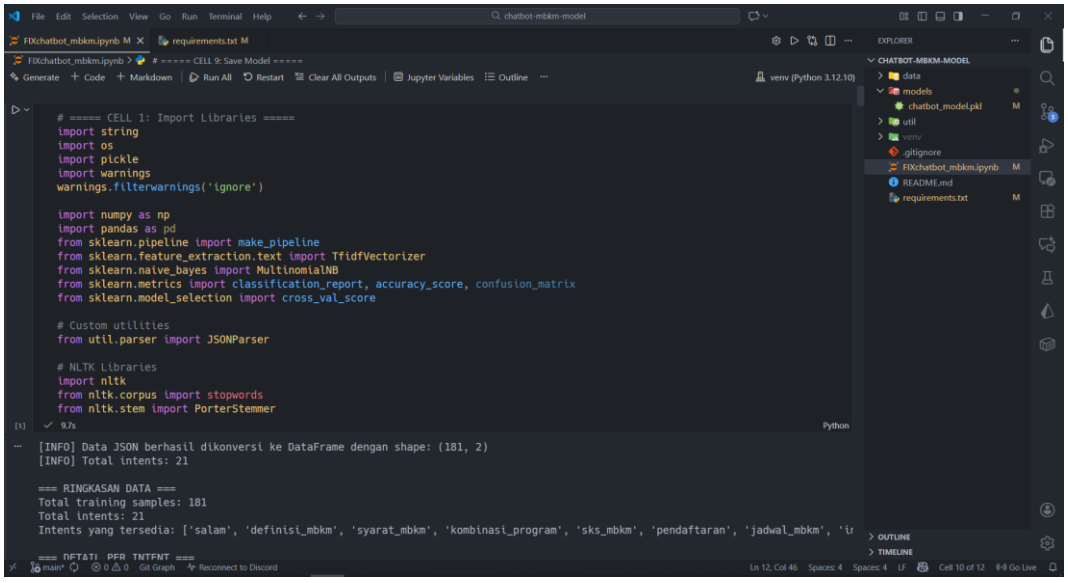
Lampiran 3

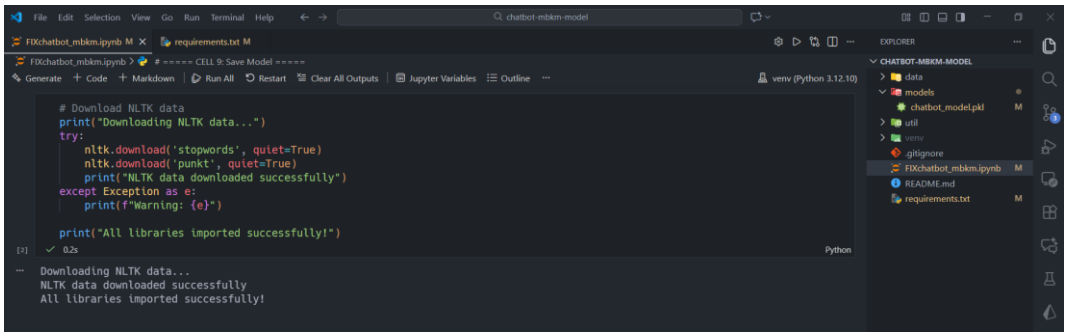
Data Chatbot JSON

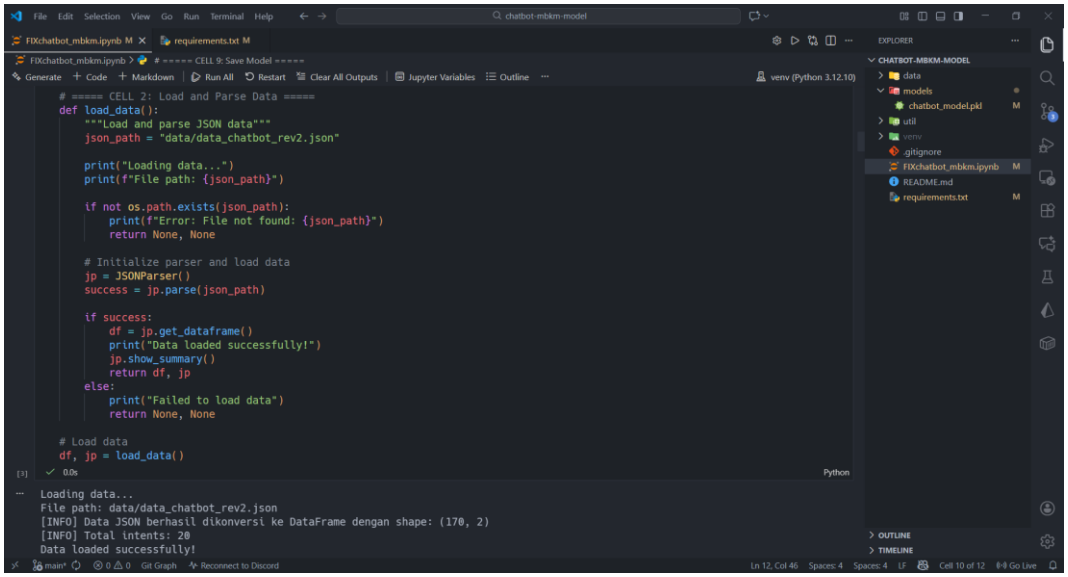


Lampiran 4

Proses Pembuatan Model

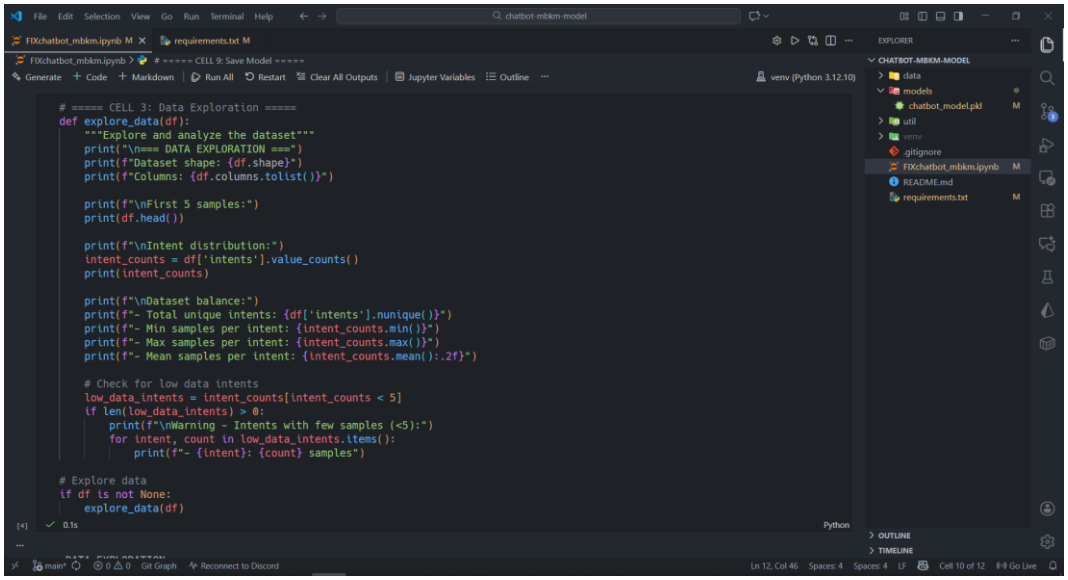




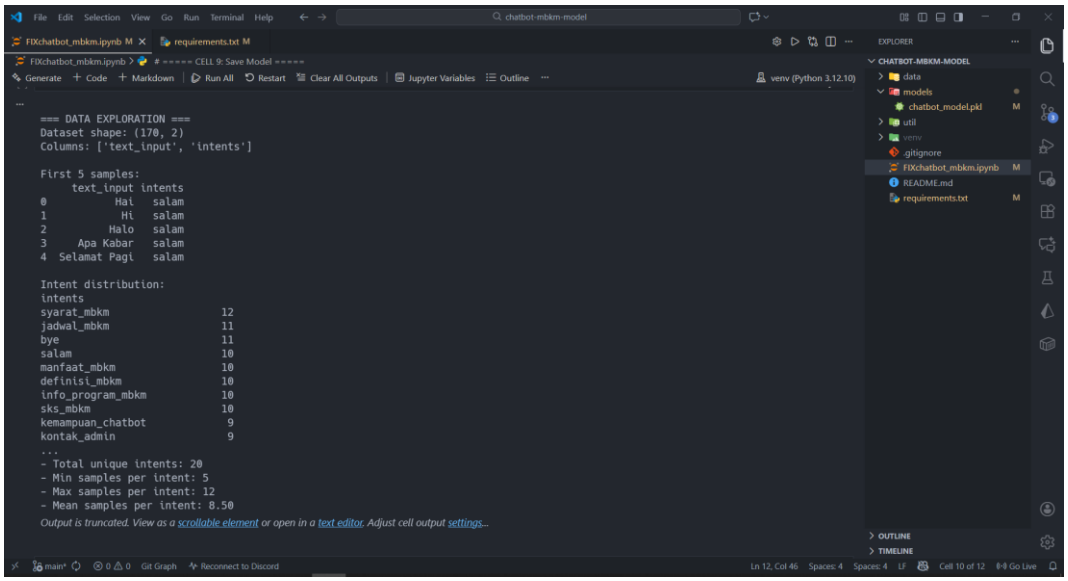


```python
# ===== CELL 2: Load and Parse Data =====
def load_data():
    """Load and parse JSON data"""
    json_path = "data/data_chatbot_rev2.json"

    print("Loading data...")
    print(f"File path: {json_path}")

    if not os.path.exists(json_path):
        print(f"Error: File not found: {json_path}")
        return None, None

    # Initialize parser and load data
    jp = JSONParser()
    success = jp.parse(json_path)

    if success:
        df = jp.get_dataframe()
        print("Data loaded successfully!")
        jp.show_summary()
        return df, jp
    else:
        print("Failed to load data")
        return None, None

# Load data
df, jp = load_data()
```

```
Loading data...
File path: data/data_chatbot_rev2.json
[INFO] Data JSON berhasil dikonversi ke DataFrame dengan shape: (170, 2)
[INFO] Total intents: 20
Data loaded successfully!
```



```python
# ===== CELL 3: Data Exploration =====
def explore_data(df):
    """Explore and analyze the dataset"""
    print("\n=== DATA EXPLORATION ===")
    print(f"Dataset shape: {df.shape}")
    print(f"Columns: {df.columns.tolist()}")

    print(f"\nFirst 5 samples:")
    print(df.head())

    print(f"\nIntent distribution:")
    intent_counts = df['intents'].value_counts()
    print(intent_counts)

    print(f"\nDataset balance:")
    print(f"- Total unique intents: {df['intents'].nunique()}")
    print(f"- Min samples per intent: {intent_counts.min()}")
    print(f"- Max samples per intent: {intent_counts.max()}")
    print(f"- Mean samples per intent: {intent_counts.mean():.2f}")

    # Check for low data intents
    low_data_intents = intent_counts[intent_counts < 5]
    if len(low_data_intents) > 0:
        print(f"\nWarning - Intents with few samples (<5):")
        for intent, count in low_data_intents.items():
            print(f"- {intent}: {count} samples")

# Explore data
if df is not None:
    explore_data(df)
```



```
=== DATA EXPLORATION ===
Dataset shape: (170, 2)
Columns: ['text_input', 'intents']

First 5 samples:
    text_input intents
0          Hai   salam
1           Hi   salam
2         Halo   salam
3    Apa Kabar   salam
4  Selamat Pagi   salam

Intent distribution:
intents
syarat_mbkm                12
jadwal_mbkm                11
bye                        11
salam                      10
manfaat_mbkm               10
definisi_mbkm              10
info_program_mbkm          10
sks_mbkm                   10
kemampuan_chatbot           9
kontak_admin                9
...
- Total unique intents: 20
- Min samples per intent: 5
- Max samples per intent: 12
- Mean samples per intent: 8.50
Output is truncated. View as a scrollable element or open in a text editor. Adjust cell output settings...
```

```python
# ===== CELL 4: Text Preprocessing =====
def setup_preprocessing():
    """Setup text preprocessing components"""
    print("\n=== SETTING UP TEXT PREPROCESSING ===")

    # Setup stopwords
    try:
        stop_words = set(stopwords.words('indonesian'))
        print(f"Loaded {len(stop_words)} Indonesian stopwords")
    except:
        print("Indonesian stopwords not available, using English")
        stop_words = set(stopwords.words('english'))

    # Setup stemmer
    stemmer = PorterStemmer()

    return stop_words, stemmer

def preprocess_text(text, stop_words, stemmer):
    """
    Preprocess text for chatbot

    Args:
        text (str): Input text
        stop_words (set): Set of stopwords
        stemmer: Stemmer object

    Returns:
        str: Processed text
    """
    if not isinstance(text, str):
        return ""

    # 1. Lowercase
    text = text.lower()

    # 2. Remove punctuation
    text = ''.join(ch for ch in text if ch not in string.punctuation)

    # 3. Remove digits
    text = ''.join(ch for ch in text if not ch.isdigit())

    # 4. Remove extra spaces
    text = ' '.join(text.split())

    # 5. Remove stopwords
    words = [word for word in text.split() if word not in stop_words]
    text = ' '.join(words)

    # 6. Stemming
    words = [stemmer.stem(word) for word in text.split()]
    text = ' '.join(words)

    return text

# Setup preprocessing
stop_words, stemmer = setup_preprocessing()

# Test preprocessing
print("\nTesting preprocessing:")
test_texts = [
    "Halo, bagaimana kabar Anda?",
    "Saya ingin mendaftar program MBKM 2024",
    "Apa saja syarat-syarat yang diperlukan?",
    "Terima kasih banyak atas informasinya!"
]

for test_text in test_texts:
    processed = preprocess_text(test_text, stop_words, stemmer)
    print(f"Original: {test_text}")
    print(f"Processed: {processed}")
    print("-" * 40)
```

✓ 0.0s

```python
# ===== CELL 5: Apply Preprocessing to Dataset =====
def apply_preprocessing(df, stop_words, stemmer):
    """Apply preprocessing to the entire dataset"""
    print("\n=== APPLYING PREPROCESSING ===")

    print("Processing all text data...")
    df['text_processed'] = df['text_input'].apply(
        lambda x: preprocess_text(x, stop_words, stemmer)
    )

    # Remove empty texts
    original_len = len(df)
    df_clean = df[df['text_processed'].str.len() > 0].reset_index(drop=True)
    removed_count = original_len - len(df_clean)

    print(f"Preprocessing results:")
    print(f"- Original data: {original_len} rows")
    print(f"- After preprocessing: {len(df_clean)} rows")
    print(f"- Removed (empty): {removed_count} rows")

    if removed_count > 0:
        print(f"Warning: {removed_count} rows removed due to empty text")

    return df_clean

# Apply preprocessing
if df is not None:
    df = apply_preprocessing(df, stop_words, stemmer)

    # Show comparison
    print(f"\nPreprocessing comparison (first 3 rows):")
    for idx in range(min(3, len(df))):
        row = df.iloc[idx]
        print(f"\n{idx+1}. Original: {row['text_input']}")
        print(f"   Processed: {row['text_processed']}")
        print(f"   Intent: {row['intents']}")
```

✓ 0.0s

```
=== APPLYING PREPROCESSING ===
Processing all text data...
Preprocessing results:
- Original data: 170 rows
- After preprocessing: 169 rows
- Removed (empty): 1 rows
Warning: 1 rows removed due to empty text

Preprocessing comparison (first 3 rows):

1. Original: Hai
   Processed: hai
   Intent: salam

2. Original: Hi
   Processed: hi
   Intent: salam

3. Original: Halo
   Processed: halo
   Intent: salam
```

```python
# ===== CELL 6: Model Training =====
def train_model(df):
    """Train the Multinomial Naive Bayes model"""
    print("\n=== MODEL TRAINING ===")

    # Create pipeline
    pipeline = make_pipeline(
        TfidfVectorizer(
            max_features=1000,
            min_df=1,
            max_df=0.9,
            ngram_range=(1, 2),
            lowercase=True,
            strip_accents='unicode'
        ),
        MultinomialNB(alpha=1.0)
    )

    # Train model
    print("Training model...")
    X = df['text_processed']
    y = df['intents']

    pipeline.fit(X, y)

    # Evaluate on training data
    y_pred = pipeline.predict(X)
    train_accuracy = accuracy_score(y, y_pred)

    print(f"Model training completed!")
    print(f"- Training accuracy: {train_accuracy:.4f}")
    print(f"- Number of classes: {len(pipeline.classes_)}")
    print(f"- Classes: {list(pipeline.classes_)}")

    # Cross-validation
    print("\nPerforming cross-validation...")
    cv_scores = cross_val_score(pipeline, X, y, cv=3, scoring='accuracy')
    print(f"Cross-validation scores: {cv_scores}")
    print(f"Mean CV accuracy: {cv_scores.mean():.4f} (+/- {cv_scores.std() * 2:.4f})")

    # Classification report
    print(f"\nClassification Report:")
    print(classification_report(y, y_pred, zero_division=0))

    return pipeline, train_accuracy

# Train model
if df is not None:
    pipeline, train_accuracy = train_model(df)
```

✓ 0.0s

```
=== MODEL TRAINING ===
Training model...
Model training completed!
- Training accuracy: 0.9290
- Number of classes: 20
- Classes: [np.str_('bantuan_teknis'), np.str_('bye'), np.str_('definisi_magang'), np.str_('definisi_mbkm'), np.str_('definisi_pertukaran'), np.str_('definisi_studi_independen'),

Performing cross-validation...
Cross-validation scores: [0.38596491 0.21428571 0.53571429]
Mean CV accuracy: 0.3787 (+/- 0.2626)
```

```
Classification Report:
                          precision    recall  f1-score   support

          bantuan_teknis       1.00      1.00      1.00         7
                     bye       1.00      1.00      1.00        11
         definisi_magang       1.00      1.00      1.00         6
           definisi_mbkm       0.90      0.90      0.90        10
      definisi_pertukaran       1.00      0.83      0.91         6
definisi_studi_independen       0.83      0.83      0.83         6
      dokumen_pendaftaran       1.00      0.86      0.92         7
        info_program_mbkm       0.64      0.90      0.75        10
             jadwal_mbkm       0.92      1.00      0.96        11
        kemampuan_chatbot       1.00      0.88      0.93         8
        kombinasi_program       1.00      0.75      0.86         8
             kontak_admin       1.00      1.00      1.00         9
             konversi_sks       1.00      1.00      1.00         7
             manfaat_mbkm       1.00      0.80      0.89        10
              pendaftaran       0.90      1.00      0.95         9
  perbedaan_magang_stupen       1.00      1.00      1.00         7
       persetujuan_prodi       1.00      1.00      1.00         5
                   salam       1.00      0.90      0.95        10
                sks_mbkm       1.00      1.00      1.00        10
             syarat_mbkm       0.79      0.92      0.85        12

                accuracy                           0.93       169
               macro avg       0.95      0.93      0.93       169
            weighted avg       0.94      0.93      0.93       169
```

```python
# ===== CELL 9: Save Model =====
def save_model(pipeline, filename="models/chatbot_model.pkl"):
    """Save the trained model"""
    try:
        # Create models directory if it doesn't exist
        os.makedirs(os.path.dirname(filename), exist_ok=True)

        # Save model
        with open(filename, 'wb') as f:
            pickle.dump(pipeline, f)

        print(f"Model saved successfully to {filename}")
        return True
    except Exception as e:
        print(f"Error saving model: {e}")
        return False

# Save model
if 'pipeline' in locals() and pipeline is not None:
    save_model(pipeline)
```

✓ 0.0s

```
Model saved successfully to models/chatbot_model.pkl
```