BAB 4
HASIL PENELITIAN

4.1 RINGKASAN HASIL PENELITIAN

Penelitian ini memiliki tujuan untuk menganalisis ulasan pengguna terhadap aplikasi Wattpad yang terdapat di *Google Play Store*, dengan pendekatan *K-Means Clustering* dan interpretasi berdasarkan dimensi UEQ. Data diperoleh melalui proses web scraping sebanyak 3.626 ulasan, kemudian dilakukan *preprocessing* yang mencakup tahapan *cleaning*, *case folding*, *normalization*, *tokenizing*, *stopword removal*, dan *stemming*. Hasil *preprocessing* menghasilkan 2.274 data bersih yang selanjutnya direpresentasikan dalam bentuk numerik menggunakan metode TF-IDF.

Selanjutnya, dilakukan pengelompokan ulasan menggunakan algoritma *K-Means*. Evaluasi menggunakan *Elbow Method* dan *Silhouette Score* menunjukkan bahwa jumlah klaster optimal adalah 8 klaster. Masing-masing klaster dianalisis berdasarkan kata-kata dominan, dan hasilnya diinterpretasikan ke dalam enam dimensi UEQ, yaitu *Attractiveness*, *Perspicuity*, *Efficiency*, *Dependability*, *Stimulation*, dan *Novelty*. Hasil analisis menunjukkan bahwa dimensi *Attractiveness* paling dominan di hampir seluruh klaster, yang berarti pengguna banyak menyoroti daya tarik tampilan dan fitur aplikasi Wattpad. Selain itu, *Dependability* dan *Perspicuity* juga cukup sering muncul, menandakan adanya perhatian terhadap keandalan sistem dan kemudahan penggunaan. Sedangkan dimensi lain seperti *Efficiency*, *Stimulation*, dan *Novelty* cenderung kurang mendapat perhatian pengguna.

Secara keseluruhan, penelitian ini memberikan gambaran bahwa persepsi pengguna terhadap Wattpad secara umum cukup positif dari segi tampilan dan konten, namun masih ditemukan beberapa keluhan teknis terkait performa dan kejelasan antarmuka.

4.2 HASIL PENGAMBILAN DATA (*SCRAPING*)

Proses pengambilan data dilakukan untuk memperoleh umpan balik dari pengguna aplikasi Wattpad dalam bentuk ulasan. Pada tahap ini digunakan *Google Colab* sebagai platform bantu, yang telah dipasang library *Google-Play-Scraper* untuk melakukan *scraping data* dari *Google Play Store*. Pengambilan data dilakukan dengan cara memasukkan id aplikasi Wattpad dan mengunduh seluruh ulasan berdasarkan rentang tanggal yang ditentukan. Informasi yang berhasil dikumpulkan meliputi *userName*, *score*, *at*, *content*. Proses *scraping* dilaksanakan pada tanggal 19 April 2025, dengan cakupan data ulasan dari periode 1 Januari 2025 hingga 19 April 2025. Jumlah total ulasan yang berhasil diperoleh adalah sebanyak 3.626 data yang disimpan dalam format csv. Gambar 4.1 merupakan hasil pengambilan data yang sudah dikonversi ke dalam bentuk *DataFrame.*

	userName	score	at	content
0	Pengguna Google	3	2025-04-19 15:43:53	kenapa setiap masuk harus disuruh login lagi d...
1	Pengguna Google	5	2025-04-19 15:21:18	apk nya sangat bagusssss ceritanya juga banyak...
2	Pengguna Google	5	2025-04-19 15:19:22	harus pake data melulu
3	Pengguna Google	2	2025-04-19 15:13:18	ini kenapa ga bisa masuk ke akun ya? mana ga b...
4	Pengguna Google	1	2025-04-19 15:09:20	ga bisa masuk jing
...	...	...	...	...
3622	Pengguna Google	2	2025-01-01 02:41:14	Sekarang yg offline cuma dua! Mana puas bacany...
3623	Pengguna Google	4	2025-01-01 02:36:53	
3624	Pengguna Google	1	2025-01-01 02:22:36	nggak bisa login,tiap mau login ada tulisan"ma...
3625	Pengguna Google	5	2025-01-01 01:43:15	Aplikasi ini bagus banget banyak pilihan novel...
3626	Pengguna Google	4	2025-01-01 01:00:36	apk nya bgus cmn klok mau login ulang knp susa...

3627 rows × 4 columns

Gambar 4.1 Hasil Pengambilan Data

Gambar 4.1 memperlihatkan bahwa data ulasan yang diperoleh masih mengandung berbagai elemen yang perlu dibersihkan. Hal ini terlihat dari adanya kata-kata yang tidak baku, tidak relevan, serta penggunaan bahasa yang tidak dapat diartikan secara langsung atau bersifat kasar. Dengan demikian, tahapan *preprocessing* diperlukan untuk menjamin bahwa data yang digunakan dalam

analisis berada dalam keadaan yang bersih, terorganisir, dan relevan dengan tujuan penelitian

4.3 *PREPROCESSING DATA*

Preprocessing data adalah tahapan pengolahan data untuk mempersiapkan data yang siap digunakan untuk analisis atau pelatihan model. Sebelum masuk ke dalam tahapan *preprocessing*, terlebih dahulu dilakukan pengecekan kualitas data, seperti penghapusan duplikat dan nilai kosong (*missing values*), agar data siap diproses lebih lanjut. Data yang digunakan dalam penelitian ini hanya data ulasan yang ada dalam kolom 'content'. Selanjutnya, *preprocessing* data terdiri dari beberapa tahapan, diantaranya :

1. *Cleaning*

   Tahapan pertama dalam preprocessing adalah data cleaning, yaitu tahap pembersihan data dari karakter-karakter yang tidak sesuai dengan menggunakan bantuan *library regular expression* (re). Proses pembersihan ini bertujuan untuk menghilangkan elemen-elemen seperti URL, simbol, angka, tanda baca, dan spasi berlebih yang tidak diperlukan dalam analisis. Implementasi proses ini ditunjukkan pada kode program dibawah ini :

```
1. def clean_text(text):
2.     text = re.sub(r'http\S+|www\S+|https\S+', '', text,
       flags=re.MULTILINE)
3.     text = re.sub(r'@\w+|#\w+', '', text)
4.     text = re.sub(r'\d+', '', text)
5.     text = re.sub(r'[^\w\s]', '', text)
6.     text = re.sub(r'\s+', ' ', text).strip()
```

   Berikut tahapan-tahapan dari fungsi 'clean_text':

   a. re.sub(r'http\S+|www\S+|https\S+', '', text, flags=re.MULTILINE) : menghapus semua URL dari sebuah teks, misalnya https://example.com.
   b. re.sub(r'@\w+|#\w+', '', text) : Menghapus *mention* dan *hashtag*, misalnya @namauser atau #topik.
   c. re.sub(r'\d+', '', text) : Menghapus semua angka dari dalam teks.
   d. re.sub(r'\s+', ' ', text).strip() : Menghapus spasi yang berlebihan, tab, atau

baris kosong, serta memastikan tidak ada spasi di awal atau akhir kalimat.

Hasil dari proses *data cleaning* dapat dilihat pada Gambar 4.2, yang menunjukkan bahwa teks ulasan telah dibersihkan dari elemen-elemen yang tidak relevan dan siap digunakan untuk tahap *preprocessing* selanjutnya..

	content	cleaned
0	kenapa setiap masuk harus disuruh login lagi d...	kenapa setiap masuk harus disuruh login lagi d...
1	apk nya sangat bagusssss ceritanya juga banyak...	apk nya sangat bagusssss ceritanya juga banyak...
2	harus pake data melulu	harus pake data melulu
3	ini kenapa ga bisa masuk ke akun ya? mana ga b...	ini kenapa ga bisa masuk ke akun ya? mana ga b...
4	ga bisa masuk jing	ga bisa masuk jing

Gambar 4.2 Hasil *Data Cleaning*

2. *Case Folding*

Tahapan kedua adalah *case folding*, yaitu proses mengubah seluruh huruf kapital dalam teks menjadi huruf kecil. Tujuan dari tahapan ini adalah untuk menyamakan bentuk kata agar tidak dianggap berbeda dalam proses analisis, misalnya kata "Baca" dan "baca" akan diperlakukan sama. Proses *case folding* ini diimplementasikan sebagaimana ditunjukkan pada kode program dibawah ini :

```
1. df['casefold'] = df['cleaned'].str.lower()
2. df.head()
```

Proses mengubah seluruh huruf dalam kolom *cleaned* menjadi huruf kecil dengan menggunakan perintah (*lower*). Hasil *Case Folding* ditunjukkan pada Gambar 4.3.

cleaned	casefold
kenapa setiap masuk harus disuruh login lagi d...	kenapa setiap masuk harus disuruh login lagi d...
apk nya sangat bagusssss ceritanya juga banyak...	apk nya sangat bagusssss ceritanya juga banyak...
harus pake data melulu	harus pake data melulu
ini kenapa ga bisa masuk ke akun ya mana ga bi...	ini kenapa ga bisa masuk ke akun ya mana ga bi...
ga bisa masuk jing	ga bisa masuk jing

Gambar 4.3 Hasil *Case Folding*

3. *Normalization*

Tahapan selanjutnya adalah *normalization*, yaitu proses mengubah kata-kata tidak baku menjadi bentuk baku. Langkah ini bertujuan untuk mengurangi keragaman kata dengan makna serupa, sehingga dapat meningkatkan konsistensi data. Kata tidak baku yang dimaksud biasanya berasal dari bahasa gaul, singkatan, atau bentuk informal lainnya, yang lazim digunakan dalam komunikasi sehari-hari dan dikenal dengan istilah *slangword*. Gambar 4.4 menampilkan contoh daftar *slangword* yang digunakan dalam penelitian ini sebagai bagian dari proses normalisasi teks.

```
slang_dict = {
    "gk": "tidak", "ga": "tidak", "nggak": "tidak", "tdk": "tidak", "dgn": "dengan",
    "bgt": "banget", "bgtu": "begitu", "bgt2": "banget banget", "trs": "terus",
    "udh": "sudah", "blm": "belum", "aja": "saja", "sm": "sama", "tp": "tapi",
    "bikin": "membuat", "nonton": "menonton", "ngulang": "mengulang", "dong": "",
    "kyk": "seperti", "jg": "juga", "dr": "dari", "krn": "karena", "utk": "untuk",
    "ni": "ini", "gt": "gitu", "ny": "nya", "y": "", "ajaib": "unik", "bosen": "bosan"
}
```

Gambar 4.4 Contoh Daftar *Slangword*

Pada proses *normalization* ini, dilakukan penggabungan antara daftar *slangword* yang diperoleh dari file CSV dan kamus tambahan (*dictionary*) untuk menangani kata-kata tidak baku yang belum tercakup dalam file tersebut. Selanjutnya, setiap kata slang dalam teks ulasan digantikan dengan bentuk baku yang sesuai berdasarkan daftar *slangword* yang telah disusun. Proses normalisasi ini diimplementasikan seperti pada kode program dibawah ini :

```
1. def normalize_slang(text):
2.     tokens = text.split()
3.     normalized = [slang_dict.get(word, word) for
        word in tokens]
4.     return " ".join(normalized)
```

Fungsi ‘normalize_slang’ digunakan untuk melakukan proses normalisasi terhadap kata-kata tidak baku (slang) dengan menggunakan kamus ‘slang_dict’. Fungsi ini bekerja dengan memproses setiap kata dalam teks, kemudian memeriksa apakah kata tersebut termasuk dalam daftar slang yang tersedia. Jika kata tersebut ditemukan dalam kamus slang, maka akan digantikan dengan

padanan kata baku yang sesuai. Sebaliknya, jika tidak termasuk dalam daftar slang, kata tersebut akan tetap dipertahankan. Hasil dari proses ini adalah teks ulasan yang telah dinormalisasi, di mana seluruh kata disusun kembali menjadi satu kalimat utuh, sebagaimana ditampilkan pada Gambar 4.5.

casefold	normalized
kenapa setiap masuk harus disuruh login lagi d...	kenapa setiap masuk harus disuruh login lagi d...
apk nya sangat bagusssss ceritanya juga banyak...	apk nya sangat bagusssss ceritanya juga banyak...
harus pake data melulu	harus pake data melulu
ini kenapa ga bisa masuk ke akun ya mana ga bi...	ini kenapa tidak bisa masuk ke akun ya mana ti...
ga bisa masuk jing	tidak bisa masuk jing

Gambar 4.5 Hasil *Normalization*

Untuk memastikan semua kata tidak baku yang digunakan oleh pengguna dapat ditangani dengan baik, dilakukan pengecekan data ulasan secara manual. Hal ini dilakukan guna mengidentifikasi kata slang yang belum tercakup dalam file CSV, lalu menambahkannya secara manual ke dalam kamus slang tambahan.

4. *Tokenizing* dan *Stopword Removal*

Tokenizing merupakan proses memecah teks ulasan menjadi satuan kata atau token. Tahapan ini dilakukan sebelum proses *stopword removal*, merupakan proses untuk menghapus kata-kata umum atau tidak bermakna signifikan dalam analisis, seperti "dan", "di", "yang", dan sebagainya. Dalam penelitian ini, proses *tokenizing* dilakukan menggunakan fungsi ‘word_tokenize’ dari pustaka *Natural Language Toolkit* (NLTK), yang memecah teks berdasarkan spasi dan tanda baca. Sementara itu, proses *stopword removal* dilakukan dengan mengacu pada daftar *stopword* dari pustaka Sastrawi. Daftar kata-kata yang diidentifikasi sebagai *stopword* ditunjukkan pada Tabel 4.1.

Tabel 4.1 Daftar *Stopword*

Daftar *Stopword*
['yang', 'untuk', 'pada', 'ke', 'para', 'namun', 'menurut', 'antara', 'dia', 'dua', 'ia', 'seperti', 'jika', 'jika', 'sehingga', 'kembali', 'dan', 'tidak', 'ini', 'karena', 'kepada', 'oleh', 'saat', 'harus', 'sementara', 'setelah', 'belum', 'kami', 'sekitar', 'bagi', 'serta', 'di', 'dari', 'telah', 'sebagai', 'masih', 'hal', 'ketika', 'adalah', 'itu', 'dalam', 'bisa', 'bahwa', 'atau', 'hanya', 'kita', 'dengan', 'akan', 'juga', 'ada', 'mereka', 'sudah', 'saya', 'terhadap', 'secara', 'agar', 'lain', 'anda', 'begitu', 'mengapa', 'kenapa', 'yaitu', 'yakni', 'daripada', 'itulah', 'lagi', 'maka', 'tentang', 'demi', 'dimana', 'kemana', 'pula', 'sambil', 'sebelum', 'sesudah', 'supaya', 'guna', 'kah', 'pun', 'sampai', 'sedangkan', 'selagi', 'sementara', 'tetapi', 'apakah', 'kecuali', 'sebab', 'selain', 'seolah', 'seraya', 'seterusnya', 'tanpa', 'agak', 'boleh', 'dapat', 'dsb', 'dst', 'dll', 'dahulu', 'dulunya', 'anu', 'demikian', 'tapi', 'ingin', 'juga', 'nggak', 'mari', 'nanti', 'melainkan', 'oh', 'ok', 'seharusnya', 'sebetulnya', 'setiap', 'setidaknya', 'sesuatu', 'pasti', 'saja', 'toh', 'ya', 'walau', 'tolong', 'tentu', 'amat', 'apalagi', 'bagaimanapun']

Proses *tokenizing* dan *stopword removal* ditunjukkan pada kode program dibawah ini :

```
1.custom_stopwords = {
      'nya', 'banget', 'aja', 'deh', 'dong', 'loh',
      'nih', 'ya', 'si', 'tau', 'kayak', 'mah',
      'lah', 'pun', 'nyaa', 'gak'
  ]
2.stop_words.update(custom_stopwords)
3.def remove_stopwords(text):
4.    tokens = word_tokenize(text)
5.    filtered = [word for word in tokens if word not
       in stop_words]
```

Proses *stopword removal* terdapat beberapa kata yang dipertahankan agar tidak dihapus, karena dinilai memiliki makna penting dalam konteks ulasan pengguna, sehingga dapat memengaruhi hasil analisis apabila dihapus secara otomatis. Hasil *stopword removal* dapat diamati melalui Gambar 4.6.

normalized	stopwords_removed
kenapa setiap masuk harus disuruh login lagi d...	masuk disuruh login dah pas login error
apk nya sangat bagusssss ceritanya juga banyak...	apk bagusssss ceritanya teman yg baca iklan wa...
harus pake data melulu	pake data melulu
ini kenapa tidak bisa masuk ke akun ya mana ti...	tidak masuk akun tidak akses pakai wifi data s...
tidak bisa masuk jing	tidak masuk jing

Gambar 4.6 Hasil *Stopword Removal*

5. *Stemming*

Tahapan *preprocessing* yang terakhir yaitu melakukan *stemming*. *Stemming* mengubah kata berimbuhan menjadi bentuk dasar dengan menghapus awalan, akhiran, sisipan, atau gabungan imbuhan, misal ‘bermain’ ketika di stemming akan menjadi ‘main’. Proses *stemming* ditunjukkan pada kode program dibawah ini:

```
1. factory = StemmerFactory()
2. stemmer = factory.create_stemmer()
3. df['stemmed'] = df['stopwords_removed'].
   swifter.apply(lambda x: stemmer.stem(x))
4. df.head()
```

Proses *stemming* dilakukan dengan bantuan pustaka Sastrawi menggunakan metode ‘StemmerFactory’. Setiap kalimat yang telah melalui tahap penghapusan *stopword* kemudian diproses untuk mengembalikan kata-kata ke bentuk dasarnya. Kolom *stemmed* berisi hasil stemming dari data, yang divisualisasikan pada Gambar 4.7.

stopwords_removed	stemmed
masuk disuruh login dah pas login error	masuk suruh login dah pas login error
apk bagusssss ceritanya teman yg baca iklan wa...	apk bagusssss cerita teman yg baca iklan wattp...
pake data melulu	pake data melulu
tidak masuk akun tidak akses pakai wifi data s...	tidak masuk akun tidak akses pakai wifi data s...
tidak masuk jing	tidak masuk jing

Gambar 4.7 Hasil *Stemming*

Hasil akhir data setelah dilakukan *preprocessing* dan penghapusan data kosong ditunjukkan pada ambar 4.8

	content	casefold	cleaned	normalized	stopwords_removed	stemmed
0	kenapa setiap masuk harus disuruh login lagi d...	kenapa setiap masuk harus disuruh login lagi d...	kenapa setiap masuk harus disuruh login lagi d...	kenapa setiap masuk harus disuruh login lagi d...	masuk disuruh login dah pas login error	masuk suruh login pas login error
1	apk nya sangat bagusssss ceritanya juga banyak...	apk nya sangat bagusssss ceritanya juga banyak...	apk nya sangat bagusssss ceritanya juga banyak...	apk nya sangat bagusssss ceritanya juga banyak...	apk bagusssss ceritanya teman yg baca iklan wa...	aplikasinya bagus dan cerita banyak cocok buat...
2	harus pake data melulu	harus pake data melulu	harus pake data melulu	harus pake data melulu	pake data melulu	harus pake data melulu
3	ini kenapa ga bisa masuk ke akun ya? mana ga b...	ini kenapa ga bisa masuk ke akun ya? mana ga b...	ini kenapa ga bisa masuk ke akun ya mana ga bi...	ini kenapa tidak bisa masuk ke akun ya mana ti...	tidak masuk akun tidak akses pakai wifi data s...	kenapa ya tidak bisa masuk ke akun Aplikasi ju...
4	ga bisa masuk jing	ga bisa masuk jing	ga bisa masuk jing	tidak bisa masuk jing	tidak masuk jing	tidak masuk jing
...	...	...	...	...	...	...
3242	Makin ngeselin tiap update.	makin ngeselin tiap update.	makin ngeselin tiap update	makin ngeselin tiap update	ngeselin update	tiap update ngeselin
3243	Menurut saya ini apk membaca yang worth it And...	menurut saya ini apk membaca yang worth it and...	menurut saya ini apk membaca yang worth it and...	menurut saya ini apk membaca yang worth it and...	apk membaca worth it and this is the most exci...	menurut saya ini apk membaca yang worth it and...
3244	Sekarang yg offline cuma dual Mana puas bacany...	sekarang yg offline cuma dual mana puas bacany...	sekarang yg offline cuma dua mana puas bacanya...	sekarang yg offline cuma dua mana puas bacanya...	yg offline puas bacanya baca online kebanyakan...	banyak iklan cerita offline hanya dua balikin ...
3246	Aplikasi ini bagus banget banyak pilihan novel...	aplikasi ini bagus banget banyak pilihan novel...	aplikasi ini bagus banget banyak pilihan novel...	aplikasi ini bagus banget banyak pilihan novel...	aplikasi bagus pilihan novel novel bagus iklan...	aplikasi ini bagus banget banyak pilihan novel...
3247	apk nya bgus cmn klok mau login ulang knp susa...	apk nya bgus cmn klok mau login ulang knp susa...	apk nya bgus cmn klok mau login ulang knp susa...	apk nya bgus cmn klok mau login ulang knp susa...	apk bgus cmn klok login ulang knp susah akun k...	apk bagus susah login ulang

2274 rows × 6 columns

Gambar 4.8 Hasil Akhir *Preprocessing Data*

Sebagaimana ditampilkan pada Gambar 4.8, sebanyak 2.273 data siap digunakan untuk proses interpretasi UEQ.

4.4 TF-IDF

Setelah melalui tahapan preprocessing seperti *cleaning*, *case folding*, *normalization*, *tokenizing*, *stopword removal*, dan *stemming*, data teks ulasan pengguna siap untuk diubah menjadi bentuk numerik menggunakan metode TF-

IDF. Metode ini digunakan untuk menilai seberapa besar kontribusi suatu kata dalam satu dokumen tertentu terhadap keseluruhan kumpulan teks atau kumpulan dokumen. Proses TF-IDF ditunjukkan pada kode program dibawah ini :

```
1. vectorizer = TfidfVectorizer(max_df=0.9, min_df=2)
2. X = vectorizer.fit_transform(df['stemmed'])
3. feature_names = vectorizer.get_feature_names_out()
4. tfidf_df = pd.DataFrame(X.toarray(), columns=feature_names)
5. tfidf_df.head()
```

	20	25	abis	acak	account	ada	adain	add	ads	again	...	wow	woy	wp	ya	yah	yaitu	yang	yg	you	youtube
0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	...	0.0	0.0	0.0	0.000000	0.0	0.0	0.00000	0.0	0.0	0.0
1	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	...	0.0	0.0	0.0	0.000000	0.0	0.0	0.11436	0.0	0.0	0.0
2	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	...	0.0	0.0	0.0	0.000000	0.0	0.0	0.00000	0.0	0.0	0.0
3	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	...	0.0	0.0	0.0	0.271693	0.0	0.0	0.00000	0.0	0.0	0.0
4	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	...	0.0	0.0	0.0	0.000000	0.0	0.0	0.00000	0.0	0.0	0.0

Gambar 4.9 Hasil TF-IDF

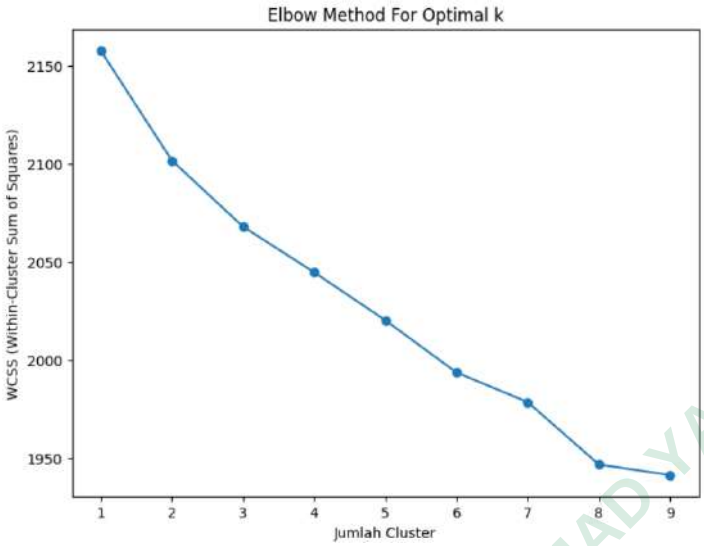
Hasil dari proses TF-IDF ditampilkan pada Gambar 4.9. Setiap baris dalam tabel mewakili satu ulasan pengguna, sedangkan setiap kolom mewakili kata (fitur) yang dihasilkan setelah proses *preprocessing*. Nilai-nilai di dalam tabel merupakan skor TF-IDF. Nilai 0.0 menunjukkan bahwa kata tersebut tidak muncul pada ulasan terkait, sedangkan nilai yang lebih tinggi menunjukkan bahwa kata tersebut memiliki bobot penting yang lebih besar dalam ulasan itu. Sebagai contoh, pada baris ke-3 (ulasan ke-4), kata "ya" memiliki nilai 0.271693, menunjukkan bahwa kata tersebut sangat menonjol dalam ulasan tersebut dibandingkan ulasan lainnya.

4.5 *CLUSTERING*

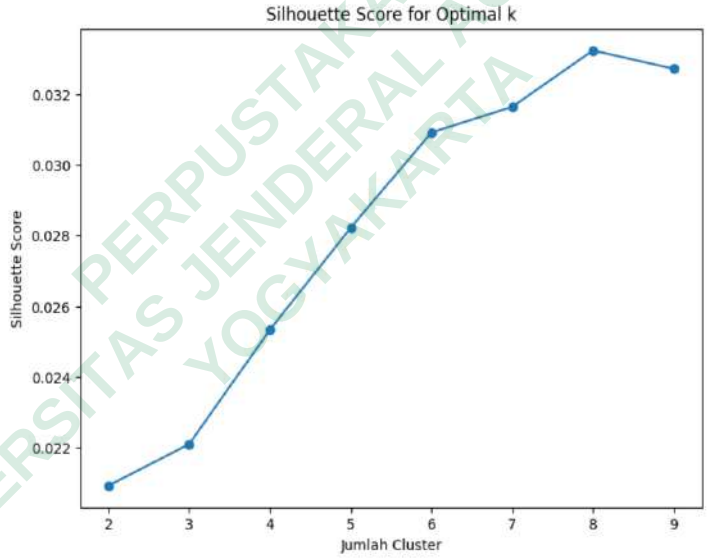
Setelah data ulasan dikonversi dengan TF-IDF, langkah berikutnya adalah menentukan jumlah klaster (k) optimal sebelum menerapkan K-Means. Metode yang digunakan yaitu Elbow Method untuk melihat titik tekukan dari nilai WCSS, dan Silhouette Score untuk menilai kualitas pemisahan klaster. Visualisasi dari kedua metode ini membantu menentukan nilai k yang sesuai. Implementasi perhitungannya ditampilkan pada kode program dibawah ini :

```
wcss = []
for i in range(1, 10):
    kmeans = KMeans(n_clusters=i, random_state=42)
    kmeans.fit(X)  # X adalah hasil dari TF-IDF
    wcss.append(kmeans.inertia_)
plt.figure(figsize=(8, 6))
plt.plot(range(1, 10), wcss, marker='o')
plt.title('Elbow Method For Optimal k')
plt.xlabel('Jumlah Cluster')
plt.ylabel('WCSS')
plt.show()
sil_scores = []
for i in range(2, 10):
    kmeans = KMeans(n_clusters=i, random_state=42)
    kmeans.fit(X)
    sil_score = silhouette_score(X, kmeans.labels_)
    sil_scores.append(sil_score)
plt.figure(figsize=(8, 6))
plt.plot(range(2, 10), sil_scores, marker='o')
plt.title('Silhouette Score for Optimal k')
plt.xlabel('Jumlah Cluster')
plt.ylabel('Silhouette Score')
plt.show()
```


Berdasarkan proses evaluasi yang dilakukan, metode *Elbow* menunjukkan bahwa klaster ke-8 merupakan titik di mana penurunan nilai WCSS mulai melandai, yang mengindikasikan bahwa penambahan jumlah klaster setelah angka tersebut tidak memberikan penurunan WCSS yang signifikan. Sementara itu, hasil evaluasi menggunakan metode *Silhouette Score* juga menunjukkan bahwa nilai tertinggi dicapai pada klaster ke-8, yang berarti kualitas pemisahan antar klaster berada pada tingkat optimal. Oleh karena itu, jumlah klaster optimal yang dipilih adalah sebanyak 8 klaster, sebagaimana ditunjukkan melalui Gambar 4.9 dan Gambar 4.10.



Gambar 4.10 Grafik Evaluasi Jumlah Klaster Menggunakan *Elbow Method*



Gambar 4.11 Grafik Evaluasi Jumlah Klaster Menggunakan *Silhouette Score*

Setelah jumlah klaster optimal ditetapkan sebanyak 8 klaster, dilakukan proses pengelompokan data ulasan menggunakan algoritma *K-Means*. Proses ini bertujuan untuk mengelompokkan ulasan yang memiliki kemiripan kata atau topik ke dalam satu klaster yang sama. Implementasi kode ditunjukkan pada kode program dibawah ini:

```
1. k = 8
2. kmeans = KMeans(n_clusters=k, random_state=42,
   n_init=10)
3. clusters = kmeans.fit_predict(X)
4. df['cluster'] = clusters
5. print(df['cluster'].value_counts())21.
   plt.xlabel('Jumlah Cluster')
6. plt.ylabel('Silhouette Score')
7. plt.show()
```

Setelah data ulasan pengguna dikonversi ke dalam bentuk numerik menggunakan metode *TF-IDF* dan jumlah klaster optimal ditentukan sebanyak 8, dilakukan proses pengelompokan data menggunakan algoritma *K-Means*. Proses ini bertujuan untuk mengelompokkan ulasan yang memiliki kemiripan kata ke dalam satu klaster yang sama. Hasil dari pengelompokan ini disimpan dalam kolom baru bernama *cluster* pada *DataFrame*, sehingga setiap ulasan dapat diidentifikasi termasuk dalam klaster mana. Untuk mengetahui distribusi jumlah data dalam masing-masing klaster, dilakukan pengecekan terhadap nilai frekuensi dari setiap label klaster. Pada Gambar 4.11, disajikan jumlah data yang terdistribusi di setiap klaster.


```
cluster
2    945
5    339
4    295
3    176
1    153
7    145
6    117
0    104
Name: count, dtype: int64
```

Gambar 4.12 Jumlah Data per Klaster

Setelah menentukan jumlah data per klaster, kode program dibawah ini digunakan untuk menampilkan 10 kata dengan skor TF-IDF tertinggi. Kata-kata ini diambil berdasarkan hasil transformasi TF-IDF, dijumlahkan dari seluruh dokumen, lalu diurutkan dari yang tertinggi. Hasilnya menunjukkan kata yang paling penting dalam kumpulan ulasan.

```
1. feature_array = vectorizer.get_feature_names_out()
2. tfidf_sorting = X.toarray().sum(axis=0).
   argsort()[::-1]
3. top_n = 10
4. top_features = feature_array[tfidf_sorting][:top_n]
5. print(f"Top {top_n} fitur TF-IDF:", top_features)
```

Proses ini dimulai dengan mengambil daftar fitur dari hasil transformasi TF-IDF (*feature_array*), kemudian menjumlahkan skor TF-IDF dari setiap kata di seluruh dokumen (X.toarray().sum(axis=0)), dan mengurutkannya dari nilai tertinggi ke terendah. Sepuluh kata paling dominan dipilih untuk mewakili kata-kata penting yang paling sering ditemukan dalam ulasan. Hasil pemilihan tersebut disajikan pada Gambar 4.13.

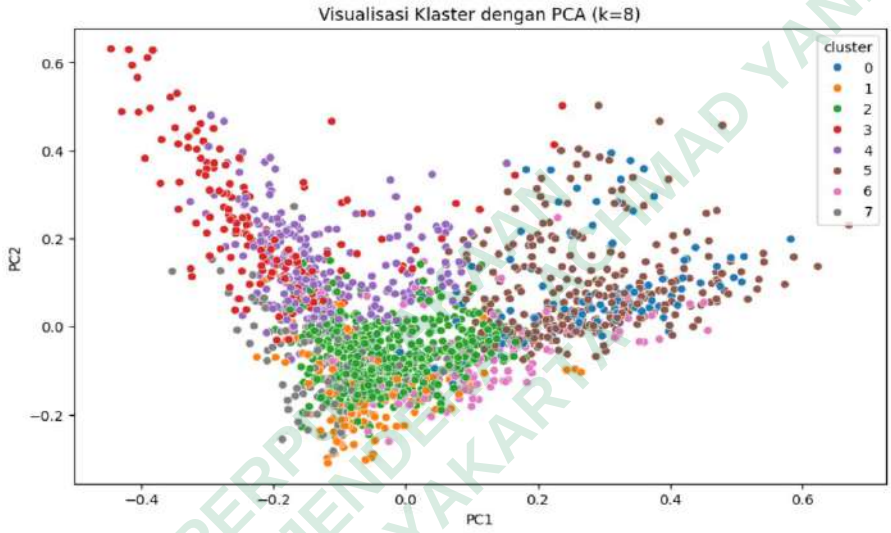
```
Top 10 fitur TF-IDF: ['tidak' 'bagus' 'bisa' 'cerita' 'wattpad' 'iklan' 'login' 'banyak'
 'aplikasi' 'apk']
```

Gambar 4.13 10 Kata dengan Skor TF-IDF Tertinggi

Langkah selanjutnya setelah menampilkan 10 kata dengan skor TF-IDF tertinggi adalah memvisualisasikan hasil *clustering*. kode program dibawah ini digunakan untuk menampilkan visualisasi dua dimensi dari data berdimensi tinggi dengan bantuan metode *Principal Component Analysis (PCA)*. Visualisasi ini memungkinkan untuk melihat pola persebaran data ulasan berdasarkan klaster yang telah terbentuk.

```
1. pca = PCA(n_components=2, random_state=42)
2. reduced_X = pca.fit_transform(X.toarray())
3. pca_df = pd.DataFrame(reduced_X, columns=['PC1',
   'PC2'])
4. pca_df['cluster'] = clusters
5. plt.figure(figsize=(10, 6))
6. sns.scatterplot(data=pca_df, x='PC1', y='PC2',
   hue='cluster', palette='tab10')
7. plt.title("Visualisasi Klaster dengan PCA (k=8)")
8. plt.show()
```

Gambar 4.14 merupakan hasil pengelompokan ulasan pengguna aplikasi Wattpad menggunakan algoritma *K-Means* dengan 8 klaster. Karena data awal memiliki banyak dimensi, digunakan teknik *Principal Component Analysis* (PCA) untuk mereduksinya menjadi dua dimensi agar dapat divisualisasikan. Setiap titik pada grafik mewakili satu ulasan, dan warna yang berbeda menunjukkan klaster yang berbeda. Ulasan dikelompokkan berdasarkan kemiripan isi. Semakin jelas pemisahan antar warna, semakin baik kualitas pemisahan antar klaster.



Gambar 4.14 Hasil Visualisasi Klaster

4.6 INTERPRETASI DIMENSI UEQ

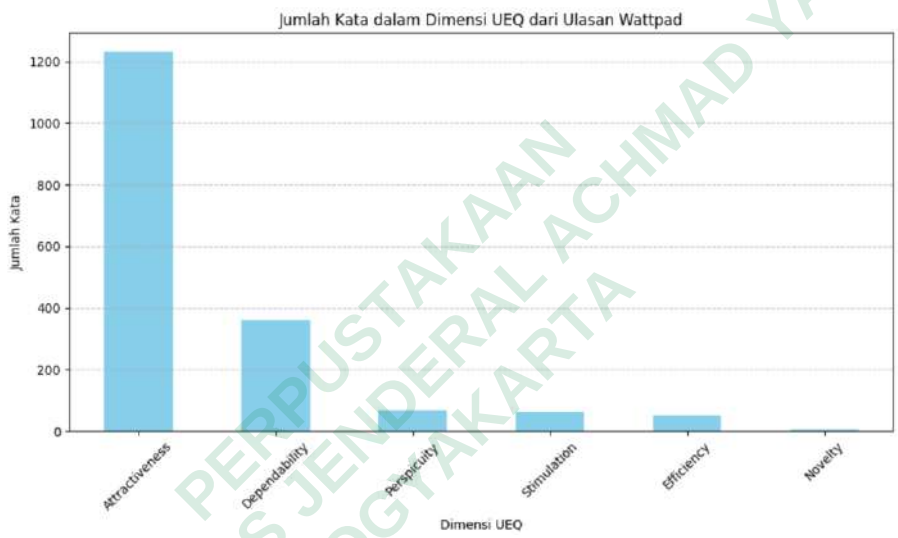
Setelah tahap *clustering* dengan *K-Means*, langkah selanjutnya adalah menginterpretasikan setiap klaster berdasarkan dimensi UEQ, yaitu *Attractiveness* (daya tarik), *Perspicuity* (kejelasan), *Efficiency* (efisiensi), *Dependability* (keandalan), *Stimulation* (stimulasi), dan *Novelty* (kebaruan). Untuk mendukung proses interpretasi ini, digunakan sebuah kamus UEQ yang berisi daftar kata-kata representatif untuk masing-masing dimensi. Kode program di bawah ini digunakan untuk menyesuaikan setiap kata dalam ulasan pengguna dengan daftar kosakata yang terdapat dalam kamus tersebut.

```
1.  dimensi_ueq = {
    'Attractiveness': [
        'menarik', 'menyenangkan', 'bagus', 'suka', 'tertarik',
'keren', 'oke', 'nyaman',
       'tidak menarik', 'membosankan', 'jelek', 'buruk', 'tidak
suka', 'tidak menyenangkan'
    ],
    'Efficiency': [
        'cepat', 'efisien', 'praktis', 'mudah digunakan',
         'lambat', 'lemot', 'bertele-tele', 'tidak efisien',
'mengganggu', 'boros waktu'
    ],
    'Dependability': [
        'dapat diandalkan', 'stabil', 'berfungsi dengan baik',
'responsif',
         'bug', 'error', 'masalah', 'crash', 'sering macet',
'tidak stabil', 'tidak bisa digunakan', 'sering gagal'
    ],
    'Perspicuity': [
         'jelas', 'dipahami', 'sederhana', 'mudah dimengerti',
'terstruktur', 'gampang',
         'membingungkan', 'bingung', 'rumit', 'tidak jelas',
'susah digunakan'
    ],
    'Novelty': [
        'unik', 'inovatif', 'kreatif', 'berbeda', 'modern',
         'biasa saja', 'tidak inovatif', 'ketinggalan zaman',
'membosankan', 'jadul'
    ],
    'Stimulation': [
         'menghibur', 'menarik', 'menggugah', 'menginspirasi',
'memotivasi', 'bermanfaat',
       'kurang menarik', 'tidak seru', 'membosankan', 'monoton'
    ]
}
```

Kode program di bawah ini digunakan untuk menghitung jumlah kata atau ulasan dalam setiap klaster yang sesuai dengan daftar kata pada kamus masing-masing dimensi UEQ. Proses ini dilakukan dengan mencocokkan kata-kata dalam ulasan pengguna dengan kata-kata representatif dari tiap dimensi. Hasil dari perhitungan ini digunakan untuk mengidentifikasi dimensi yang paling dominan dalam setiap klaster, dengan tujuan memberikan wawasan yang lebih luas terkait persepsi pengalaman pengguna selama menjalankan aplikasi berdasarkan dimensi UEQ.

```
1.  def klasifikasi_ulasan(text, dimensi_ueq):
2.      hasil = {dimensi: 0 for dimensi in
        dimensi_ueq}
3.      for kata in text.split():
4.          for dimensi, keywords in
            dimensi_ueq.items():
5.              if kata in keywords:
6.                  hasil[dimensi] += 1
7.      return hasil
8.  hasil_klasifikasi = df['cleaned'].apply(lambda x:
    klasifikasi_ulasan(x, dimensi_ueq))
9.  df_dimensi = pd.DataFrame
    (hasil_klasifikasi.tolist())
10. total_per_dimensi = df_dimensi.sum()
11. plt.figure(figsize=(10, 6))
12. total_per_dimensi.sort_values(ascending=False)
    .plot(kind='bar', color='skyblue')
13. plt.title('Jumlah Kata dalam Dimensi UEQ dari
    Ulasan Wattpad')
14. plt.ylabel('Jumlah Kata')
15. plt.xlabel('Dimensi UEQ')
16. plt.xticks(rotation=45)
17. plt.grid(axis='y', linestyle='--', alpha=0.7)
18. plt.tight_layout()
19. plt.show()
```

Dimensi *Attractiveness* paling dominan dengan lebih dari 1200 kata, menunjukkan bahwa daya tarik Wattpad menjadi fokus utama ulasan pengguna. *Dependability* juga cukup sering disebut, menandakan pentingnya keandalan aplikasi. Di sisi lain, dimensi lain seperti *Perspicuity*, *Stimulation*, *Efficiency*, dan *Novelty* menunjukkan jumlah kata yang relatif rendah, yang mengindikasikan bahwa aspek-aspek tersebut tidak terlalu menjadi sorotan bagi pengguna. Kondisi ini dapat diamati pada Gambar 4.15.

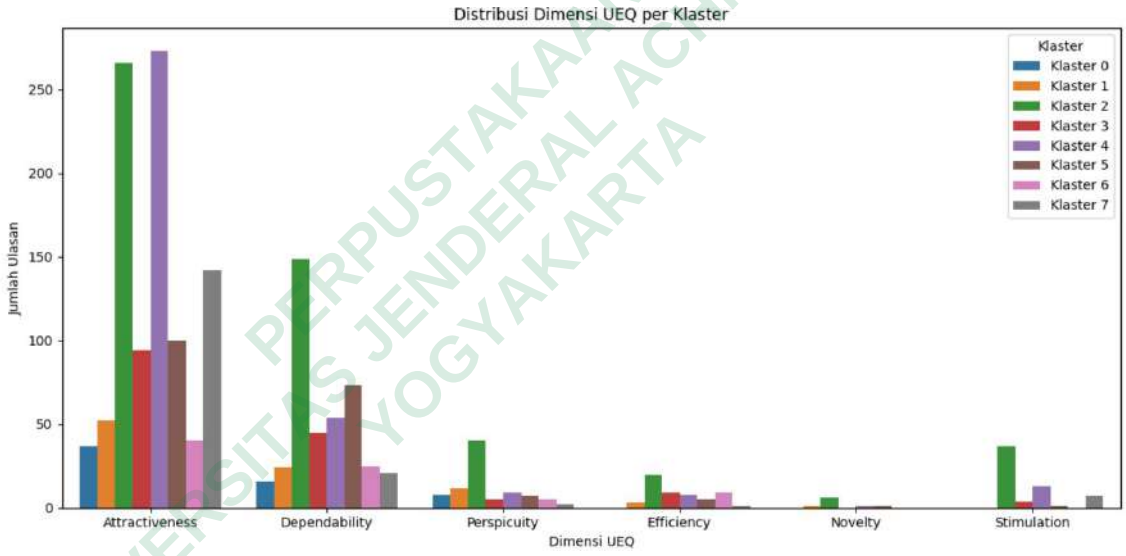


Gambar 4.15 Jumlah Kata dalam Dimensi UEQ

Langkah selanjutnya setelah menentukan klaster dan mendeteksi dimensi UEQ adalah menganalisis distribusi dimensi UEQ di setiap klaster. Kode program di bawah ini digunakan untuk mendeteksi dimensi berdasarkan kata kunci dalam ulasan, lalu menghitung frekuensi kemunculannya per klaster. Hasilnya divisualisasikan dalam bentuk diagram batang, sehingga memudahkan pemahaman mengenai persebaran aspek pengalaman pengguna seperti *Attractiveness* dan *Dependability* pada masing-masing klaster.

```
1.   def deteksi_dimensi(teks):
2.     dimensi_ditemukan = []
3.       for dimensi, kata_kunci in
         dimensi_ueq.items():
4.         for kata in kata_kunci:
             If kata in teks:
5.               dimensi_ditemukan.append(dimensi)
                 break
6.     return dimensi_ditemukan
7.  df['dimensi_ueq'] = 8.  df['cleaned']
    .apply(deteksi_dimensi)
8.  data_visualisasi = []
9.  from collections import Counter
10. for i in range(k):
11.   subset = df[df['cluster'] == i]
12.   semua_dimensi = sum(subset['dimensi_ueq']
      .tolist(), [])
13.   counter = Counter(semua_dimensi)
14.   print(f"\n ◇  Klaster {i}:")
15.   for dimensi, jumlah in counter.items():
16.        print(f"  - {dimensi}: {jumlah} ulasan")
17.        data_visualisasi.append({
            'Klaster': f'Klaster {i}',
            'Dimensi_UEQ': dimensi,
            'Jumlah': jumlah
           })
18. df_viz = pd.DataFrame(data_visualisasi)
19. plt.figure(figsize=(12, 6))
20. sns.barplot(data=df_viz, x='Dimensi_UEQ',
    y='Jumlah', hue='Klaster')
21. plt.title('Distribusi Dimensi UEQ per Klaster')
22. plt.xlabel('Dimensi UEQ')
23. plt.ylabel('Jumlah Ulasan')
24. plt.legend(title='Klaster')
25. plt.tight_layout()
26. plt.show()
```

Berdasarkan Gambar 4.16, terlihat bahwa dimensi *Attractiveness* merupakan yang paling dominan di hampir seluruh klaster, dengan total lebih dari 1200 ulasan. Hal ini menunjukkan bahwa aspek daya tarik visual dan kesan umum dari Wattpad menjadi fokus utama perhatian pengguna. Selain itu, dimensi *Dependability* juga cukup sering muncul, khususnya pada Klaster 2 dan Klaster 5, yang mencerminkan bahwa keandalan dan konsistensi aplikasi dianggap penting oleh sebagian pengguna. Sementara itu, dimensi lain seperti *Perspicuity*, *Stimulation*, *Efficiency*, dan *Novelty* memiliki jumlah ulasan yang jauh lebih rendah di semua klaster. Ini mengindikasikan bahwa aspek-aspek tersebut kurang disorot atau kurang dirasakan oleh pengguna dalam pengalaman mereka menggunakan Wattpad.



Gambar 4.16 Jumlah Dimensi UEQ per Klaster

Untuk memahami lebih jauh karakteristik masing-masing klaster, berikut adalah uraian lengkapnya:

1. Klaster 0 : Dominan daya Tarik (*Attractiveness)*, menunjukkan pengguna menganggap aplikasi menarik secara tampilan atau fitur. Namun, terdapat ulasan terkait ketidakstabilan **(***Dependability***)** dan kebingungan dalam penggunaan **(***Perspicuity***).**
2. Klaster 1 : Ulasan cenderung positif (daya tarik), namun sebagian mengeluhkan masalah teknis dan kemudahan penggunaan.

3. Klaster 2 : Klaster dengan ulasan terbanyak dan paling lengkap, menunjukkan pengguna menikmati fitur dan tampilan, namun tetap menyoroti banyak masalah teknis.
4. Klaster 3 : Mirip dengan klaster 2 tetapi dengan intensitas yang lebih rendah. Kombinasi antara apresiasi dan keluhan teknis.
5. Klaster 4 **:** Ulasan sangat positif tentang daya tarik aplikasi, tetapi tetap menyuarakan perbaikan untuk fitur-fitur tertentu.
6. Klaster 5 : Dominasi ulasan positif, meskipun keluhan teknis (*Dependability*) juga menonjol.
7. Klaster 6 : Fokus pada daya tarik, tetapi juga banyak menyebutkan error dan gangguan performa.
8. Klaster 7 : Ulasan menggambarkan pengalaman menyenangkan secara umum, namun terdapat sedikit keluhan terhadap keandalan dan kejelasan fitur.

Berdasarkan hasil analisis clustering dan interpretasi dimensi *User Experience Questionnaire* (UEQ), ditemukan bahwa *Attractiveness* merupakan dimensi yang paling dominan dalam ulasan pengguna aplikasi Wattpad. Hasil ini memperlihatkan bahwa pengguna cenderung memberikan persepsi positif terhadap elemen desain visual, *user interface*, dan fungsionalitas aplikasi. Daya tarik estetika aplikasi menjadi aspek yang paling disorot, dengan lebih dari 1.200 ulasan terkait dimensi ini.

Selain itu, dimensi *Dependability* dan *Perspicuity* juga muncul secara konsisten, terutama pada Klaster 2 dan Klaster 5. Hal ini mengindikasikan adanya keluhan terhadap kestabilan aplikasi, kemunculan *error*, dan kebingungan dalam penggunaan fitur tertentu. Artinya, meskipun Wattpad menarik secara visual, sebagian pengguna mengalami hambatan dalam hal keandalan dan kejelasan penggunaan.

Sementara itu, dimensi lain seperti *Efficiency*, *Stimulation*, dan *Novelty* memiliki frekuensi ulasan yang jauh lebih rendah. Hal ini menunjukkan bahwa

aspek-aspek tersebut belum menjadi perhatian utama pengguna, atau belum cukup menonjol dalam pengalaman mereka menggunakan aplikasi.