BAB III
ANALISA DAN PERANCANGAN SISTEM

3.1 Analisa System

Aplikasi chat dengan fitur penerjemahan otomatis merupakan inovasi dalam komunikasi lintas bahasa yang memungkinkan pengguna berinteraksi secara real-time tanpa batasan bahasa. Sistem ini bekerja dengan menerima input pesan dari pengguna, kemudian memeriksa kemungkinan adanya kesalahan ketik (typo) yang dapat memengaruhi akurasi pesan. Jika ditemukan kesalahan, sistem akan memperbaikinya sebelum meneruskan pesan untuk diterjemahkan. Namun, dalam implementasi sistem penerjemahan otomatis yang ada saat ini, sering kali terjadi kendala akibat kesalahan pengetikan yang tidak diperbaiki secara otomatis. Akibatnya, kata yang salah tetap diterjemahkan tanpa koreksi terlebih dahulu, yang berpotensi menyebabkan kesalahpahaman dalam komunikasi antar pengguna.

Sistem ini masih menghadapi beberapa permasalahan yang dapat memengaruhi akurasi penerjemahan. Salah satu masalah utama adalah kesalahan ketik (typo) yang dapat mengganggu proses penerjemahan. Sistem penerjemah otomatis yang ada saat ini belum memiliki fitur perbaikan otomatis terhadap kesalahan ketik, sehingga kata yang salah tetap diterjemahkan tanpa koreksi terlebih dahulu, yang menyebabkan hasil terjemahan kurang sesuai dengan maksud asli pengguna.

3.2 Hasil Analisa System

Hasil analisa sistem menunjukkan bahwa proses kesalahan kata (autocorrect) dalam aplikasi chat dilakukan secara menggunakan pendekatan, *N-Gram* (*Unigram* dan *Bigram*) dan *Levenshtein Distance* untuk autocorrect. Sistem ini dirancang untuk membantu pengguna memperbaiki kesalahan ketik sebelum diterjemahkan ke bahasa target.

Autocorrect menggunakan kombinasi *Unigram* dan *Bigram* untuk mengevaluasi konteks kata, serta *Levenshtein Distance* untuk menghitung jarak

edit antar kata. Probabilitas Bigram diberi bobot lebih tinggi karena mampu menangkap konteks kalimat secara lebih spesifik. Levenshtein Distance berperan penting dalam menentukan saran koreksi yang paling mirip berdasarkan operasi penyisipan, penghapusan, dan substitusi karakter.

Penelitian ini mengintegrasikan metode *N-Gram* dan *Levenshtein Distance* dalam aplikasi chat untuk mendukung fitur penerjemahan otomatis, yang mencakup perbaikan kesalahan ketik (typo). Sistem ini mendukung tiga bahasa, yaitu Indonesia, Inggris, dan Spanish.Model yang digunakan berbasis *Unigram dan Bigram* dengan pendekatan probabilistik menggunakan *Maximum Likelihood Estimation (MLE)* serta teknik *Laplace Smoothing* untuk mengatasi probabilitas nol dan *Levenshtein Distance* digunakan untuk menghitung jarak kemiripan kata dan memberikan saran koreksi yang paling mendekati.

3.2.1 Pembentukan Unigram dan Bigram

Proses pertama adalah preprocessing yaitu case folding, filtering, dan tokenisasi. Setelah tahapan preprocessing, langkah selanjutnya adalah pembentukan *Unigram* dengan mengambil $N = 1$ dan *Bigram* dengan mengambil $N = 2$ dari setiap kata dalam kalimat. Teks korpus *N-Gram* dari buku Game Of Thrones serta dari platform Kaggle dan Teks korpus untuk deteksi kesalahan penulisan kata diambil dari Kamus Besar Bahasa Indonesia (KBBI) dari Github.

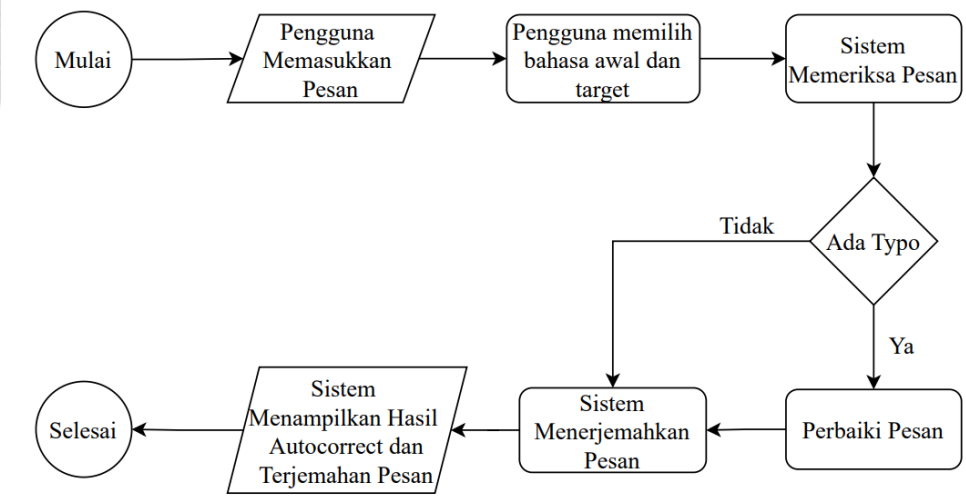
Contoh kalimat sebagai berikut.

Kita harus kembali mendesak

- Hasil *Unigram* sebagai berikut:
  A. Kita
  B. Harus
  C. Kembali
  D. Mendesak
- Hasil *Bigram* sebagai berikut
  A. Kita Harus
  B. Harus Kembali
  C. Kembali Mendesak

3.2.2 Flowchart Proses Pengolahan Pesan

Flowchart proses pengujian alur kerja sistem dalam mengolah pesan pengguna, mulai dari pemilihan bahasa, pemeriksaan pesan, perbaikan teks jika diperlukan, hingga penerjemahan dan menampilkan hasil akhir dapat di lihat pada Gambar 3.1



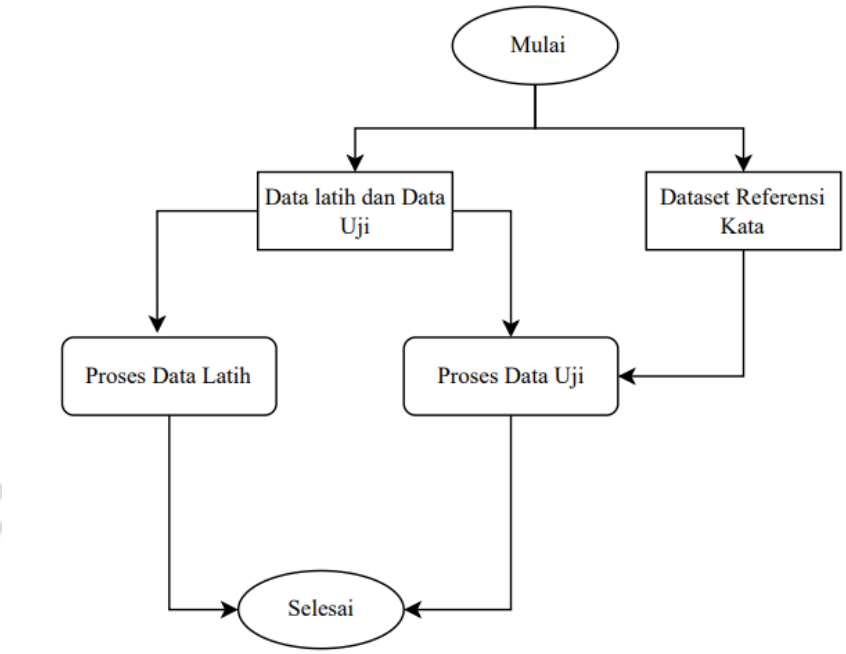
Gambar 3.1. Flowchart Proses Pengolahan Pesan

Penjelasan Flowchart pengolahan pesan pada gambar 3.1 adalah sebagai berikut:

1. Proses dimulai ketika pengguna ingin menerjemahkan sebuah pesan
2. Pengguna mengetikkan pesan yang ingin di terjemahkan
3. Pengguna akan memilih bahasa awal dan bahasa tujuan untuk di terjemahkan
4. Sistem akan memeriksa apakah di pesan yang di kirim oleh pengguna terdapat kesalah pengetikan
5. Keputusan apakah pesan valid ?
    - Jika pesan valid: Maka pesan akan langsung diterjemahkan
    - Jika tidak valid : Maka sistem akan memperbaiki kesalahan pada pesan
6. Sistem akan menerjemahkan ke bahasa yang telah di pilih sebelumnya
7. Sistem akan menampilkan hasil terjemahan ke pengguna

3.2.3 Flowchart Pengolahan Data

Flowchart ini menggambarkan tahapan dalam proses pelatihan dan pengujian model menggunakan metode *N-Gram* dan *Levenshtein Distance*. Model dilatih menggunakan data latih yang bersumber dari buku Game Of Thrones. Setiap data diproses secara terpisah. dapat di lihat pada Gambar 3.2 sebagai berikut.



Gambar 3.2. Flowchart Pengolahan Data Autocorrect

Penjelasan Flowchart Pengholahan dataset pada gambar 3.2 adalah sebagai berikut:

1. Mulai

   Proses dimulai sebagai langkah awal dari seluruh rangkaian sistem.
2. Data Latih dan Data Uji

   Memasukkan dua sumber data berbeda, yaitu:

- Data Latih berasal dari buku Game Of Thrones dan digunakan untuk proses pelatihan model.
- Data uji dirancang dan dibuat secara mandiri,Teks uji ini berisi kalimat-kalimat yang sengaja diberikan kesalahan pengetikan untuk keperluan pengujian model

  Kedua data ini akan diproses secara terpisah sesuai fungsinya.

3. Dataset Referensi Kata

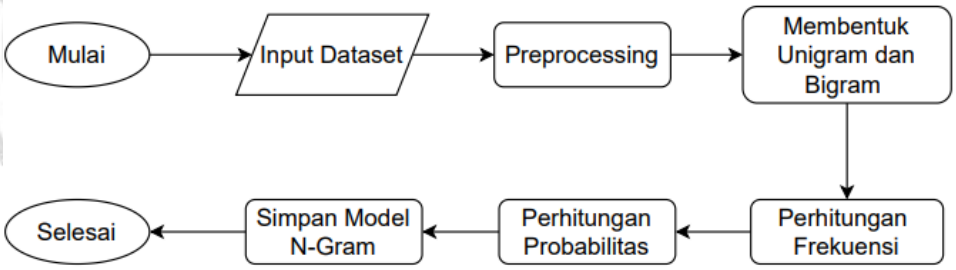
   Korpus kata baku yang berfungsi sebagai kamus referensi untuk memvalidasi kata-kata dalam proses pengujian. Dataset ini

digunakan untuk menentukan apakah suatu kata merupakan typo atau bukan dengan cara membandingkan kata dalam data uji dengan kata-kata yang ada dalam korpus referensi.

4. Proses Data Latih Pada tahap ini, data latih diproses melalui pembersihan teks, pembentukan *Unigram dan Bigram*, perhitungan frekuensi dan probabilitas, kemudian disimpan sebagai model *N-gram* untuk tahap pengujian.
5. Proses Data Uji, Data uji diproses menggunakan model *N-gram* yang telah dibentuk dan dibandingkan dengan Dataset Referensi Kata. Setelah melalui preprocessing, data diuji untuk proses evaluasi, autocorrect. Kata-kata yang tidak ditemukan dalam Dataset Referensi Kata akan diperiksa menggunakan model *Unigram* dan *Bigram* untuk mendeteksi kemungkinan typo dan memberikan saran koreksi.

3.2.4 Flowchart Algoritma N-Gram Data Latih

Flowchart ini menggambarkan tahapan dalam proses pelatihan model menggunakan algoritma *N-Gram* pada data latih. dapat di lihat pada Gambar 3.3 sebagai berikut.



Gambar 3.3 Proses Ekstraksi Data Menjadi *N-Gram* untuk Autocorrect

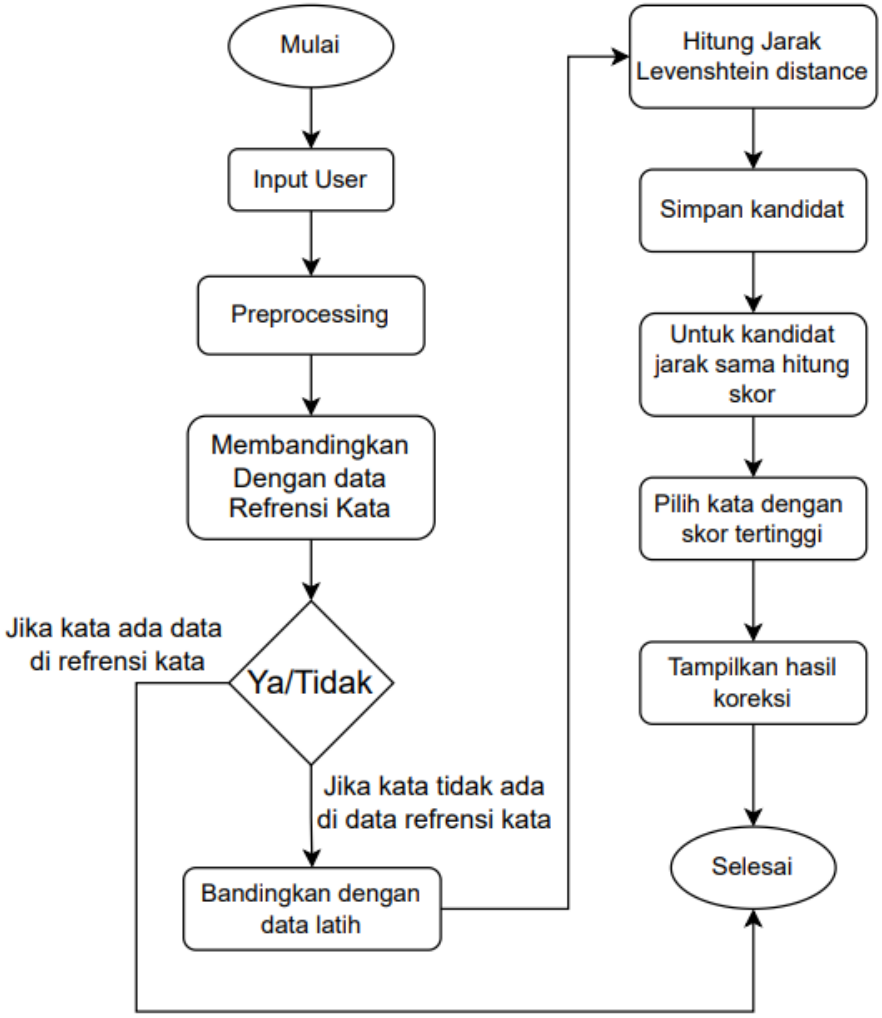
Penjelasan Flowchart Ekstraksi Data menjadi *N-Gram* pada gambar 3.3 adalah sebagai berikut:

Tahapan dalam proses pembentukan model *N-Gram* (Unigram dan Bigram) yang digunakan untuk keperluan autocorrect. Adapun penjelasan tiap tahapan sebagai berikut:

1. Memasukkan dataset teks yang digunakan untuk proses pembelajaran model.
2. Proses Pembersihan dataset di lakukan berbagai cara yaitu Case folding, Filtering dan Tokenizing
3. Dataset yang sudah bersih diubah menjadi kumpulan *Unigram* dan *Bigram* yaitu susunan tiga kata berturut-turut dari teks.
4. Menghitung Frekuensi atau jumlah kemunculan Setiap *Unigram* dan *Bigram* yang terbentuk dihitung frekuensinya, yaitu berapa kali kata tersebut muncul dalam seluruh dataset. Ini bertujuan untuk mengetahui pola kemunculan kata.
5. Menghitung probabilitas atau menghitung kemunculan suatu kata berdasarkan konteks dua kata sebelumnya
6. Hasil dari perhitungan frekuensi dan probabilitas disimpan sebagai model *N-Gram* yang nantinya akan digunakan dalam proses pengujian autocorrect

3.2.5 Flowchart Levenshtein Distance dan N-Gram pada Autocorrect

Proses kerja sistem autocorrect dalam memperbaiki kesalahan pengetikan (typo) menggunakan algoritma Levenshtein Distance. Proses ini bekerja dengan cara membandingkan setiap kata yang tidak ditemukan dalam kamus dengan semua kata dalam dataset referensi untuk menemukan kandidat kata dengan kemiripan tertinggi.



Gambar 3.4 Proses *Levenshtein Distance* dan *N-Gram* pada Autocorrect

Penjelasan Flowchart Autocorrect pada gambar 3.4 adalah sebagai berikut:

1. Proses dimulai setelah sistem mendeteksi bahwa sebuah kata tidak ditemukan dalam kamus referensi.
2. Sistem mengambil kata hasil input dari pengguna yang diduga mengandung kesalahan.
3. Kata input dibandingkan satu per satu dengan semua kata data latih
4. *Levenshtein Distance* dihitung antara kata input dan data latih.

5. Sistem menyimpan kata-kata yang memiliki nilai terendah. Algoritma mempertahankan kata-kata dengan jarak *Levenshtein* maksimal 3 sebagai kandidat potensial untuk koreksi.
6. Untuk setiap kandidat, sistem menghitung skor gabungan probabilitas *Unigram* dan *Bigram*
7. Kandidat dengan skor tertinggi (LD rendah + *N-Gram* tinggi) dipilih sebagai koreksi terbaik.
8. Sistem mengembalikan kata hasil koreksi ke proses utama (untuk diterjemahkan)

3.3 Representasi Model

3.3.1 Perhitungan Frekuensi

Proses pembentukan *Unigram* dan *Bigram*, dilakukan perhitungan frekuensi, yaitu dengan menjumlahkan kemunculan kata-kata yang sama dalam seluruh dataset yang telah melalui tahap preprocessing. Proses ini bertujuan untuk menghitung seberapa sering sebuah *Unigram* dan *Bigram* muncul dalam data, yang selanjutnya akan digunakan dalam perhitungan probabilitas untuk model, Hasil perhitungan frekuensi *Unigram* dan *Bigram* disajikan dalam Tabel berikut.

Tabel 3.1. Data Latih Bahasa Indonesia *Unigram*

No	Unigram IDN	Frekuensi
1	kembali	423
2	gelap	161
3	mereka	2891
4	orang	1227
5	itu	4058
…	…	…
195	memasak	7
196	mengubah	19
197	menulis	30
198	mengajar	1
199	menonton	22
200	minuman	20

Tabel 3.2. Data Latih Bahasa Indonesia *Bigram*

No	Bigram IDN	Frekuensi
1	yang panjang	134
2	itu besar	162
3	ini adalah	321
4	saya mengerti	32
5	dia melihat	43
..	……	….
96	memasak makanan	19
97	dia adalah	30
98	itu adalah	1
99	membuka pintu	22
100	membeli makanan	20

Tabel 3.3. Data Latih Bahasa Inggris *Unigram*

No	Unigram ENG	Frekuensi
1	book	112
2	house	32
3	table	42
4	we	234
5	dark	56
…	…	…
196	search	23
197	market	12
198	earlier	46
199	busy	23
200	pen	3

Tabel 3.4. Data Latih Bahasa Inggris *Bigram*

No	Bigram ENG	Frekuensi
1	the book	232
2	this is	43
3	that was	454
4	i understand	145
…	…	…
5	she sees	45
96	wearing clothes	65
97	cooking food	23
98	she is	15
99	this is	8
100	that is	4

Tabel 3.5. Data Latih Spanyol *Unigram*

No	Unigram SPS	Frekuensi
1	que	453
2	largo	352
3	amigo	163
4	viejo	76
5	hermano	75
…	…	…
196	escribir	34
197	explicar	64
198	perezoso	36
199	dibujar	22
200	salado	36

Tabel 3.6. Data Latih Spanyol *Bigram*

No	Bigram SPS	Frekuensi
1	el libro	123
2	esto es	234
3	eso fue	165
4	yo entiendo	65
5	ella ve	32
…	…	…
96	vistiendo ropa	17
97	amando familia	24
98	ella es	3
99	esto es	101
100	eso es	97

3.3.2 Perhitungan Probabilitas Menggunakkan Metode *Maximum Likelihood Estimation* (MLE) dan *Laplace Smoothing*

Proses selanjutnya menghitung probabilitas kemunculan kata menggunakan metode *Maximum Likelihood Estimation* (MLE) yang dikombinasikan dengan teknik *Laplace Smoothing* untuk mengatasi probabilitas nol.

3.3.2.1 Perhitungan Probabilitas Unigram

Proses perhitungan probabilitas *Unigram* dilakukan dengan membagi frekuensi kata dengan jumlah kata yang ada ditambah kata uniq, Menghitung probabilitas *Unigram* menggunakkan rumus 2.2 Seperti Berikut:

$$\text{P(Kembali)} = \frac{423 + 1}{243741 + 11715} = \frac{424}{255.456} = 0{,}0017$$

3.3.2.2 Perhitungan Probabilitas Bigram

Proses perhitungan probabilitas, dilakukan penyortiran terhadap satu kata pertama dari setiap *Bigram* untuk mencari kemunculan yang sama dalam dataset.

Seperti berikut :

Menghitung nilai *Bigram* dari kata (memasak makanan) Maka di lakukan penyortiran di dalam dataset yang memiliki 2 kata awal dari kata *Bigram* yaitu memasak. pada tabel 3.4 hasil sorting

Tabel 3.7. Hasil Sorting

Bigram	Frekuensi
Memasak makanan	1
memasak hidangan	1
memasak semur	1
memasak untuknya	1
memasak sayur	1
memasak nasi	1
memasak kunang	1

1. Frekuensi *Bigram* "memasak makanan" beserta *Bigram* lain yang memiliki awalan "memasak" dijumlahkan sebagai berikut:

$$1+1+1+1+1+1+1 = 7$$

2. Dataset yang digunakan untuk model dalam Bahasa Indonesia mengandung total 11715 kata unik bahasa inggris 11335 dan bahasa spanyol 16775. Jumlah ini digunakan sebagai variabel dalam metode *Laplace Smoothing* untuk menghindari nilai probabilitas nol.
3. Menghitung Probabilitas kata di sekeliling mereka dengan jumlah frekuensi 7. menggunakkan rumus probabilitas 2.4.

$$\mathrm{P(Makanan|Memasak)} = \frac{1+1}{6+11715} = \frac{2}{11721} = 0{,}002$$

Hasil perhitungan probabilitas kemunculan *Unigram* di dalam korpus yang dianalisis adalah sebesar 0,0017,dan hasil perhitungan *Bigram* 0,002. Hasil perhitungan probabilitas *Unigram* dan *Bigram* ditampilkan pada Tabel berikut. Pemilihan dilakukan secara acak dengan tujuan menunjukkan bagaimana variasi probabilitas dalam dataset. Hal ini memberikan gambaran mengenai distribusi dalam korpus yang digunakan.

Tabel 3.8. Hasil Perhitungan Probabilitas *Unigram* Dataset Bahasa Indonesia.

No	Unigram	Jumlah Kata(V)	Frekuensi	N	Probabilitas
1	kembali	11715	423	243741	0,0017
2	gelap	11715	161	243741	0.0006
3	mereka	11715	2891	243741	0.0113
4	orang	11715	1227	243741	0.0048
5	itu	11715	4058	243741	0.0159

Tabel 3.9. Hasil Perhitungan Probabilitas *Bigram* Dataset Bahasa Indonesia.

No	Bigram	Jumlah Kata(V)	Frekuensi	Total Frekuensi	Probabilitas
1	dia melihat	11715	100	43	0.0004
2	memasak makanan	11715	1	6	0.0002
3	dia adalah	11715	6	43	0.0004
4	itu adalah	11715	13	36	0.0009
5	membuka pintu	11715	18	26	0.0016

Tabel 3.10. Hasil Perhitungan Probabilitas *Unigram* Dataset Bahasa Inggris

No	Unigram	Jumlah Kata(V)	Frekuensi	N	Probabilitas
1	back	11335	693	216426	0.0031
2	and	11335	8997	216426	0.0396
3	should	11335	226	216426	0.0010
4	them	11335	1190	216426	0.0052
5	you	11335	3493	216426	0.0154

Tabel 3.11. Hasil Perhitungan Probabilitas *Bigram* Dataset Bahasa Inggris

No	Bigram	Jumlah Kata(V)	Frekuensi	Total Frekuensi	Probabilitas
1	i understand	11335	6	48	0.0005
2	of them	11335	153	30	0.0088
3	going to	11335	66	23	0.0059
4	he said	111335	348	35	0.0208
5	this is	111335	100	22	0.0084

Tabel 3.12. Hasil Perhitungan Probabilitas *Unigram* Dataset Bahasa Spanyol

No	Unigram	Jumlah Kata(V)	Frekuensi	N	Probabilitas
1	que	16775	8675	275916	0.0296
2	largo	16775	129	275916	0.0004
3	amigo	16775	56	275916	0.0002
4	donde	16775	211	275916	0.0007
5	hermano	16775	445	275916	0.0015

Tabel 3.13. Hasil Perhitungan Probabilitas *Bigram* Dataset Bahasa Spanyol

No	Bigram	Jumlah Kata(V)	Frekuensi	Total Frekuensi	Probabilitas
1	el libro	16775	15	18	0.0006
2	esto es	16775	23	12	0.0014

No	Bigram	Jumlah Kata(V)	Frekuensi	Total Frekuensi	Probabilitas
3	como un	16775	175	19	0.0096
4	pero no	16775	278	27	0.0150
5	casa grande	16775	6	43	0.0004

Setelah didapatkan probabilitas *Unigram* dan *Bigram*, nilai probabilitas tersebut dihitung menggunakan metode *Maximum Likelihood Estimation (MLE)* dan *Laplace Smoothing*. Hasil perhitungan probabilitas yang ditampilkan dalam tabel digunakan untuk memperkirakan kemungkinan kemunculan suatu kata berdasarkan dua kata sebelumnya dalam teks. Perhitungan ini berperan penting dalam penelitian untuk meningkatkan akurasi koreksi kesalahan ketik

3.3.3 Perhitungan Menggunakan Levenshtein Distance

Proses selanjutnya, setelah sistem menghitung probabilitas berdasarkan model *N-Gram*, yaitu *Maximum Likelihood Estimation* (MLE) dan *Laplace Smoothing*, adalah melakukan koreksi kesalahan ketik (typo) menggunakan metode *Levenshtein Distance.*

Levenshtein Distance digunakan untuk mengukur jarak kedekatan antara dua kata, yaitu kata hasil input pengguna dan kata yang benar berdasarkan kamus. Metode ini menghitung jumlah minimum operasi yang diperlukan untuk mengubah satu kata menjadi kata lainnya.

Contoh Kasus Autoccorect:

- Kata benar : memasak
- Kata salah (typo) : memask

Dalam kasus ini, terdapat kesalahan ketik berupa hilangnya huruf “a” kedua dalam kata memasak.

3.3.3.1 Definisi dan Matriks Awal

a. String a = memask (Panjang m = 6)
b. String b = memasak (Panjang n = 7)

Tabel 3.14. Matriks Perhitungan *levenshetin Distance*

		m	e	m	a	s	a	k
	0	1	2	3	4	5	6	7
m	1							
e	2							
m	3							
a	4							
s	5							
k	6							

3.3.3.2 Hasil Perhitungan Levenshtein Distance

Tabel 3.15. Matriks Perhitungan *levenshetin Distance*

		m	e	m	a	s	a	k
	0	1	2	3	4	5	6	7
m	1	0	1	1	3	4	5	6
e	2	1	0	1	2	3	4	5
m	3	2	1	0	1	2	3	4
a	4	3	2	1	0	1	2	3
s	5	4	3	2	1	0	1	2
k	6	5	4	3	2	1	1	1

Setelah perhitungan akhir di dapatkan nilai *Levenshtein Distance* = 1*,* Artinya, dibutuhkan satu operasi edit penyisipan huruf **(a)** antara (s) dan (k) untuk mengubah "memask" menjadi "memasak", Dalam sistem yang dikembangkan, jumlah kandidat kata yang dipilih sebanyak 5 kata dengan jarak *Levenshtein* terdekat.

3.3.4 Perhitungan Skor

Setelah kandidat kata koreksi dihasilkan melalui *Levenshtein Distance*, jika jarak *Levenshtein* antar kandidat sama, maka sistem memilih kata terbaik dengan mempertimbangkan konteks bahasa menggunakan skor *N-Gram*. Skor ini menggabungkan probabilitas *Unigram* dan *Bigram* untuk menilai kemungkinan suatu kata muncul dalam konteks tertentu. Rumus skoring yang digunakan adalah:

$$totalScore = (bigramScore\ x\ 1000) + unigramScore$$

Keterangan:

Bigram Score menunjukkan peluang kemunculan dua kata berurutan, sedangkan *unigram* Score menunjukkan peluang kata tunggal. Bobot *bigram* dikalikan 1000 karena lebih merepresentasikan konteks kalimat dibanding *unigram* yang hanya berdasarkan frekuensi kata.

Contoh Implementasi:

Misalkan kandidat koreksi untuk kata "memask" adalah:

- memasak (bigramScore = 0.002, unigramScore = 0.0017)
- memasok (bigramScore = 0.0004, unigramScore = 0.0003)

Perhitungan skor:

- memasak: (0.002 × 1000) + 0.0017 = 2.0017 (0.002 × 1000) + 0.0017 = 2.0017
- memasok: (0.0004 × 1000) + 0.0003 = 0.4003 (0.0004 × 1000) + 0.0003 =0.4003

Kata memasak dipilih karena memiliki skor tertinggi.

3.3.5 Pengujian dan Penilaian Kinerja Sistem

Pengujian dilakukan untuk menilai kinerja sistem autocorrect yang dikembangkan. Salah satu parameter utama untuk mengukur kinerja sistem adalah akurasi koreksi.

3.3.5.1 Perhitungan Akurasi

Pengujian dilakukan dengan 20 data pengujian per jenis kesalahan. Setiap jenis pengujian akan menghasilkan nilai Kb (koreksi benar) dan Ks (koreksi salah).di gunakan rumus 2.6

Contoh Untuk pengujian pertama:

- Kb = 15 (koreksi benar)
- Ks = 5 (koreksi salah)

$$\text{Akurasi} = \frac{15}{15+5}\ x\ 100\% = 75\%$$

3.3.5.2 Perhitungan Akurasi Rata-rata

Perhitungan akurasi rata-rata dilakukan untuk mengetahui performa keseluruhan sistem dalam melakukan koreksi otomatis terhadap berbagai jenis kesalahan. Akurasi rata-rata dihitung dari hasil akurasi masing-masing jenis pengujian, kemudian dirata-ratakan.

Contoh Perhitungan Akurasi Rata-rata:

- Pengujian 1: 100%
- Pengujian 2: 90%
- Pengujian 3: 100%
- Pengujian 4: 75%
- Pengujian 5: 70%
- Pengujian 6: 70%

$$\text{Rata-Rata} = \frac{100+ 90 +100 + 75 +70+70}{6} = \frac{505}{6} = 84$$

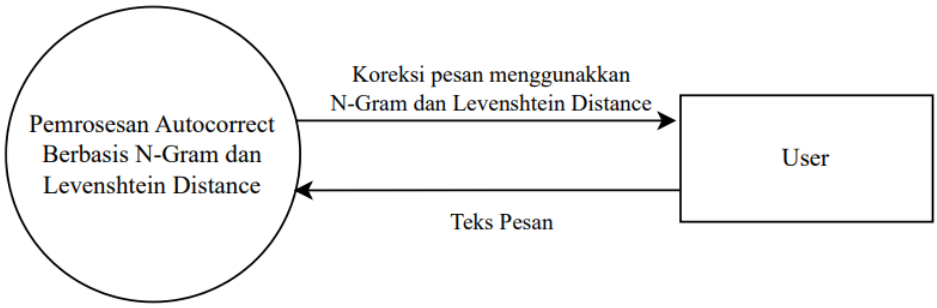
akurasi rata-rata yang di dapat dari seluruh pengujian adalah 84%.

3.4 Perancangan Sistem

3.4.1 Diagram Konteks

Diagram konteks ini menggambarkan alur kerja sistem secara keseluruhan yang menunjukkan bagaimana data mengalir antara pengguna dan sistem. Sistem yang dikembangkan adalah Aplikasi Chat dengan Fitur Penerjemahan yang dilengkapi dengan kemampuan autocorrect. Pada sistem ini, metode yang digunakan adalah *N-Gram* dan *Levenshtein Distance*. dilakukan menggunakan metode *N-Gram* (*Unigram*) yang menganalisis pola distribusi kata dalam teks untuk mengenali bahasa yang digunakan. Sedangkan untuk fitur autocorrect, sistem memanfaatkan kombinasi metode *N-Gram Unigram* dan *Bigram* untuk menghitung probabilitas kata dan pasangan kata, serta metode *Levenshtein Distance* untuk menemukan kandidat kata yang paling mendekati secara struktur karakter. Setelah teks diproses, diperbaiki, dan diterjemahkan, hasil akhirnya akan ditampilkan kembali kepada

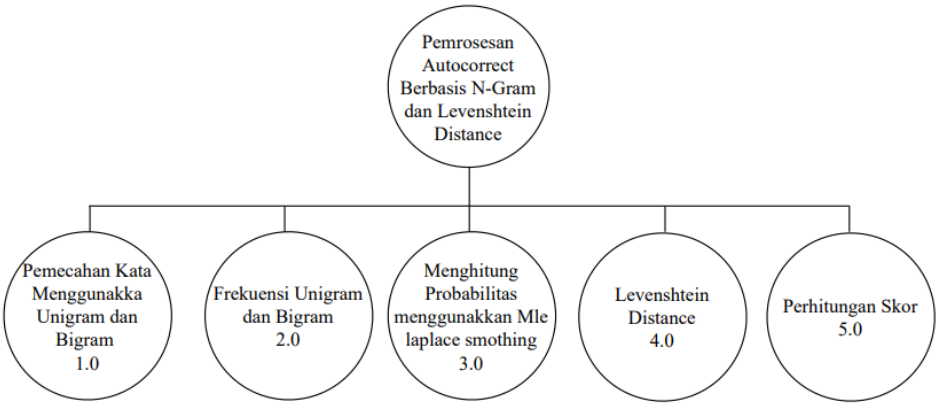
pengguna.. Rancangan diagram konteks sistem ini dapat dilihat pada Gambar 3.5.



Gambar 3.5. Diagram Konteks Sistem Chat dengan Integrasi *N-Gram* dan *Levenshtein Distance*

3.4.2 Diagram Berjenjang

Diagram berjenjang merupakan urutan proses dalam suatu sistem. Dalam pengembangan sistem penerjemah otomatis, perbaikan kata, diperlukan bagan berjenjang sebagai dasar untuk menggambarkan Data Flow Diagram (DFD) pada level-level yang lebih rinci. Diagram berjenjang dari sistem ini ditampilkan pada Gambar 3.6



Gambar 3.6. Diagram Berjenjang Sistem Chat dengan *Integrasi N-Gram*

Penjelasan dari Diagram Berjenjang adalah sebagai berikut:

1. Level 0 Sistem Utama

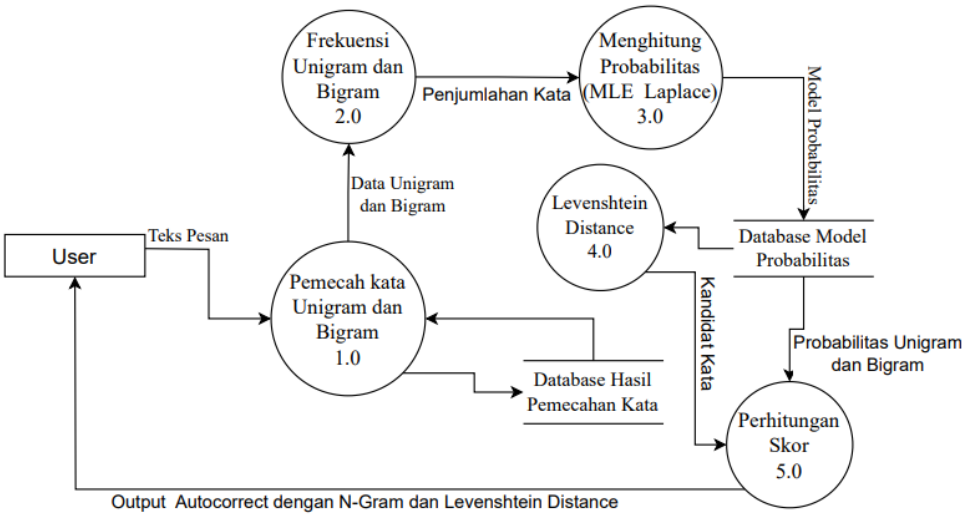
   Pada tingkat ini, sistem secara keseluruhan direpresentasikan sebagai satu entitas utama, yaitu "Sistem Penerjemah, Perbaikan Kesalahan Kata". Diagram pada level ini memberikan gambaran umum dari keseluruhan proses yang ada dalam sistem.

2. Level 1 Proses Utama

   - Preprocessing Data Latih (1.0):
     1. Pemecahan Kata Menggunakan *Unigram* dan *Bigram* (1.0) Memisahkan kata ke dalam bentuk *Unigram* dan Bigram untuk membantu analisis bahasa.
     2. Frekuensi *Unigram* dan *Bigram* (2.0) Menghitung frekuensi kemunculan *Unigram* dan *Bigram* dari korpus yang digunakan.
     3. Perhitungan Probabilitas dengan MLE dan *Laplace Smoothing* (3.0) Menghitung probabilitas kata berdasarkan data latih menggunakan metode *Maximum Likelihood Estimation (MLE)* serta menerapkan *Laplace Smoothing*
     4. Levenshtein Distance (4.0) untuk mengukur tingkat kemiripan antara kata yang diketik oleh pengguna dengan kata-kata dalam korpus. Hal ini bertujuan untuk mendeteksi kesalahan penulisan dan memberikan saran kata yang lebih akurat.
     5. Perhitungan Skor (5.0) yang menggabungkan hasil probabilitas dan jarak edit guna menentukan kata yang paling sesuai sebagai hasil koreksi.

3.4.3 Data Flow Diagram (DFD) Level 1

DFD level 2 proses 1 merupakan penurunan dari DFD level 1 yang terjadi pada proses Preprocessing data latih



Gambar 3.7. Data Flow Diagram (DFD) Level 1 Sistem Chat dengan Integrasi *N-Gram*

Penjelasan diagram diatas adalah sebagai berikut:

1. Pemecahan Kata dengan *Unigram* dan *Bigram* (1.0)

   Proses ini membagi teks menjadi kelompok N1 dan N2 (Unigram dan Bigram) untuk mengidentifikasi pola dalam data. Hasil pemecahan kata ini kemudian digunakan dalam perhitungan probabilitas dan disimpan dalam database hasil pemecahan kata sebagai referensi untuk proses selanjutnya.

2. Frekuensi *Unigram* dan *Bigram* (2.0)

   Sistem menghitung jumlah kemunculan masing-masing Unigram dan Bigram dalam teks. Informasi tentang frekuensi ini menjadi dasar untuk analisis probabilitas yang akan dilakukan pada tahap berikutnya.

3. Perhitungan Probabilitas (*MLE & Laplace*) (3.0)

   Proses ini data *Unigram* dan *Bigram* yang telah terbentuk digunakan untuk menghitung probabilitas kemunculan kata menggunakan metode *Maximum Likelihood Estimation (MLE)* dan *Laplace Smoothing*. Perhitungan ini bertujuan untuk menentukan kemungkinan suatu kata muncul dalam suatu konteks berdasarkan pola yang telah dianalisis.

Hasil dari proses ini disimpan dalam Database Model Probabilitas untuk digunakan dalam perbaikan teks pengguna.

4. Levenshtein Distance (4.0)

   Proses ini mengevaluasi kemungkinan kesalahan penulisan kata dengan mengukur jarak edit (jumlah operasi yang dibutuhkan untuk mengubah satu kata menjadi kata lain) antara kata input dan kandidat kata dalam database.

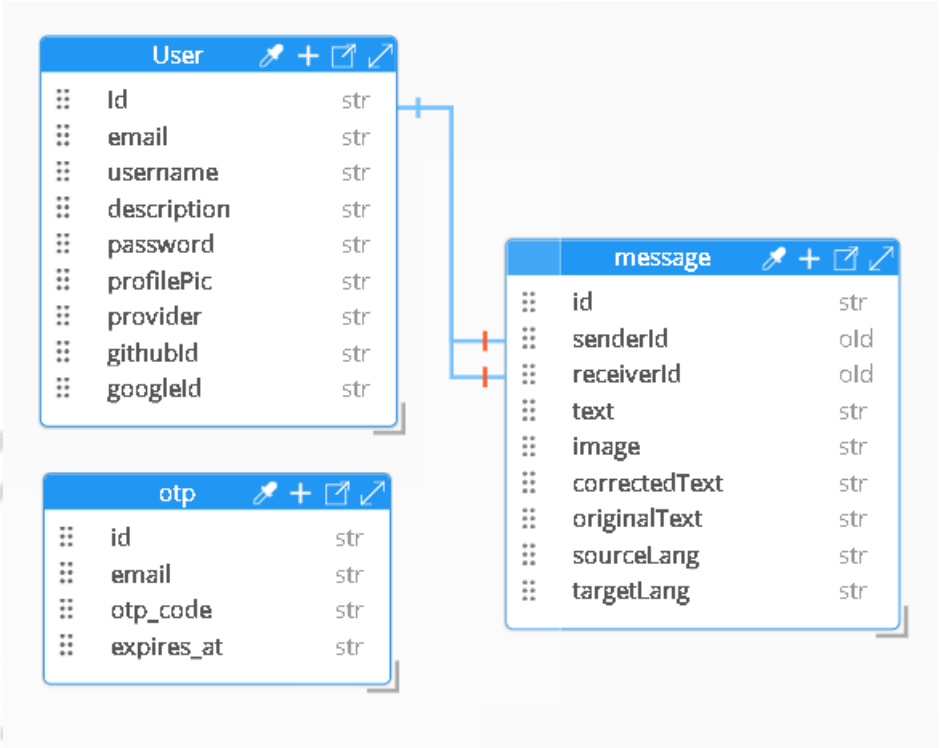
5. Perhitungan Skor (5.0)

   Proses akhir dalam sistem adalah penggabungan antara hasil probabilitas *Unigram* dan *Bigram* dan skor *Levenshtein Distance* untuk menentukan kata koreksi terbaik. Proses ini menghitung nilai gabungan dari kedua metode untuk menentukan hasil koreksi yang paling akurat. Hasil akhirnya adalah output deteksi dan autocorrect yang dikembalikan kepada pengguna.

3.4.4 Perancangan Basis Data

Perancangan basis data adalah proses penyusunan struktur penyimpanan data dalam sistem yang bertujuan untuk memastikan data tersimpan secara terstruktur dan dapat diakses dengan efisien. Salah satu perancangan basis data yang digunakan dalam sistem ini adalah

ERD (Entity Relationship Diagram), Tampilan ERD dapat dilihat pada Gambar 3.9.



Gambar 3.8. Diagram ERD App Chat

Diagram ERD ini dirancang untuk mendukung sistem komunikasi berbasis pesan, yang mencakup fitur percakapan individu. Tabel profile menyimpan data pengguna, yang terhubung dengan tabel message untuk mencatat pesan yang dikirim.

Sistem ini mendukung penerjemahan pesan, tabel otp menyimpan kode verifikasi untuk autentikasi pengguna,. Struktur ERD ini memastikan penyimpanan data yang terorganisir, memungkinkan akses yang efisien, serta mendukung fitur komunikasi yang aman dan fleksibel. Diagram ERD diatas terdapat 3 tabel yang digunakan dalam menyusun basis data yang digunakan dalam sistem. Berikut rincian dari tiap tabel:

1. Tabel profile

Tabel ini digunakan untuk menyimpan data pengguna yang memiliki akses ke sistem. Selain menyimpan informasi pribadi seperti nama dan email, tabel ini juga berfungsi untuk mengelola peran pengguna dalam sistem. Pengguna yang terdaftar di dalam tabel ini harus melakukan autentikasi sebelum dapat menggunakan fitur sistem. Struktur tabel profile dapat dilihat pada tabel 3.16

Tabel 3.16. Tabel profile

Field	Type
id	String
email	String
username	String
description	String
password	String
profilePic	String
provider	String
githubId	String
googleId	String

2. Tabel message

Tabel ini berfungsi untuk menyimpan data pesan yang dikirim dalam sistem. Setiap pesan yang dikirim oleh pengguna akan disimpan dalam tabel ini, termasuk informasi tentang pengirim, isi pesan, serta status pesan seperti terkirim atau telah dibaca. Struktur tabel message dapat dilihat pada tabel 3.17

Tabel 3.17. Tabel message

Field	Type
id	String
senderId	ObjectId
receiverId	ObjectId
text	String
image	String

Field	Type
correctedText	String
originalText	String
sourceLang	String
targetLang	String

3. Tabel otp

Tabel ini digunakan untuk menyimpan kode verifikasi satu kali (OTP) yang dikirim kepada pengguna saat proses autentikasi pembuatan akun atau saat ketika lupa password.Struktur tabel otp dapat dilihat pada tabel 3.18

Tabel 3.18. Tabel otp

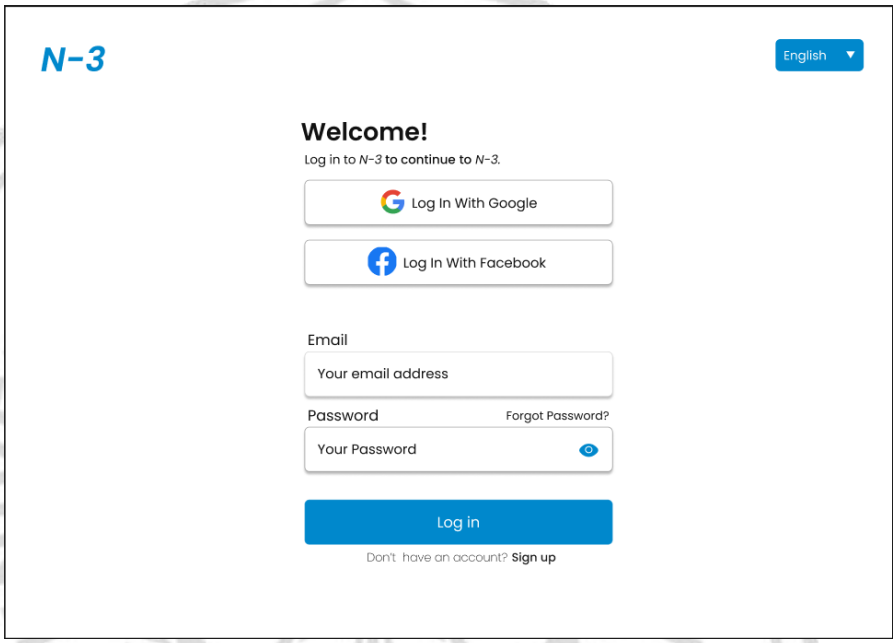
Field	Type
id	String
email	String
otp_code	String
expires_at	String

3.4.5 Perancangan Antar Muka

Aplikasi Chat ini adalah sistem berbasis web yang dikembangkan menggunakan bahasa pemrograman JavaScript dengan framework React.js untuk frontend dan Express.js untuk backend. Antarmuka sistem berfungsi sebagai penghubung antara pengguna dan sistem untuk melakukan input data berupa teks yang akan diterjemahkan, memproses koreksi kesalahan ketik, dan menampilkan hasil terjemahan. Sistem ini memiliki beberapa halaman utama, antara lain:

a. Halaman Login

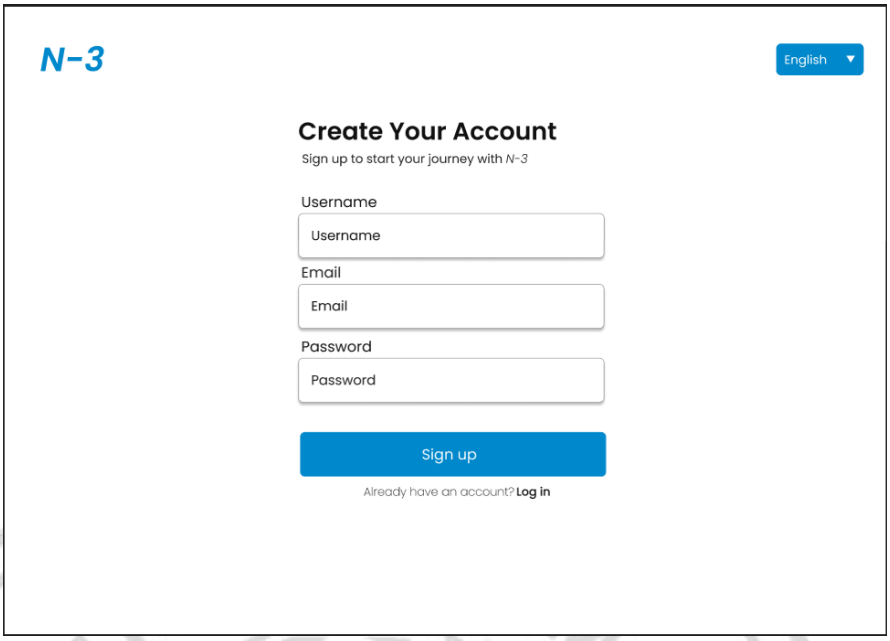
Halaman ini mengharuskan pengguna memasukkan username dan password yang sesuai dengan akun mereka untuk dapat mengakses sistem. Proses ini bertujuan untuk memastikan bahwa setiap pengguna mendapatkan hak akses yang sesuai dengan perannya, yaitu sebagai user atau admin. Rancangan halaman login dapat dilihat pada gambar 3.10



Gambar 3.9. Halaman Login

b. Halaman Registrasi

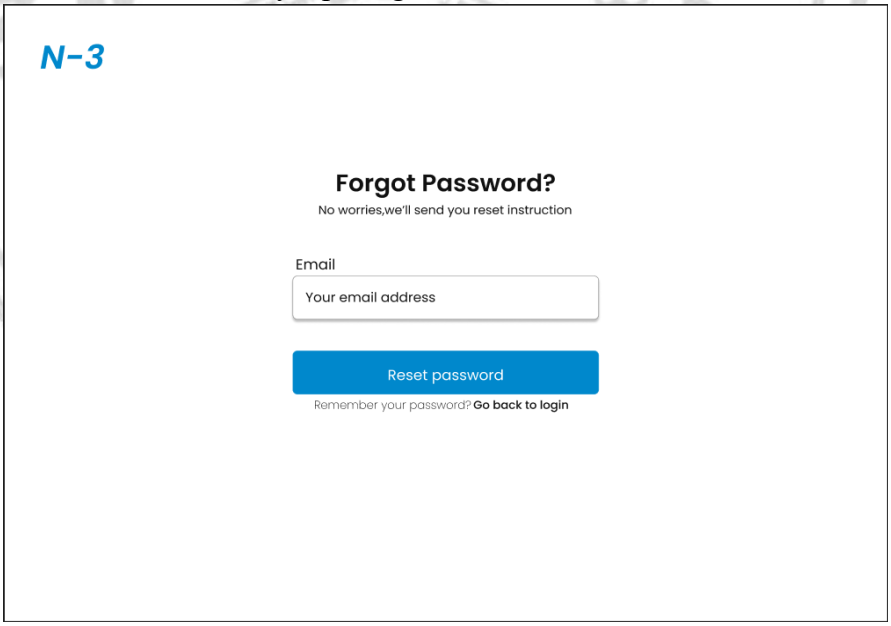
Halaman ini digunakan untuk mendaftarkan akun baru dengan mengisi beberapa informasi yang diperlukan, seperti nama depan, nama belakang, email, password, dan region. Pengguna harus memastikan bahwa data yang dimasukkan valid untuk dapat melanjutkan proses pendaftaran. Rancangan halaman Registrasi dapat dilihat pada gambar 3.11



Gambar 3.10. Halaman Registrasi

c. Halaman Lupa Pasasword Verifikasi Email

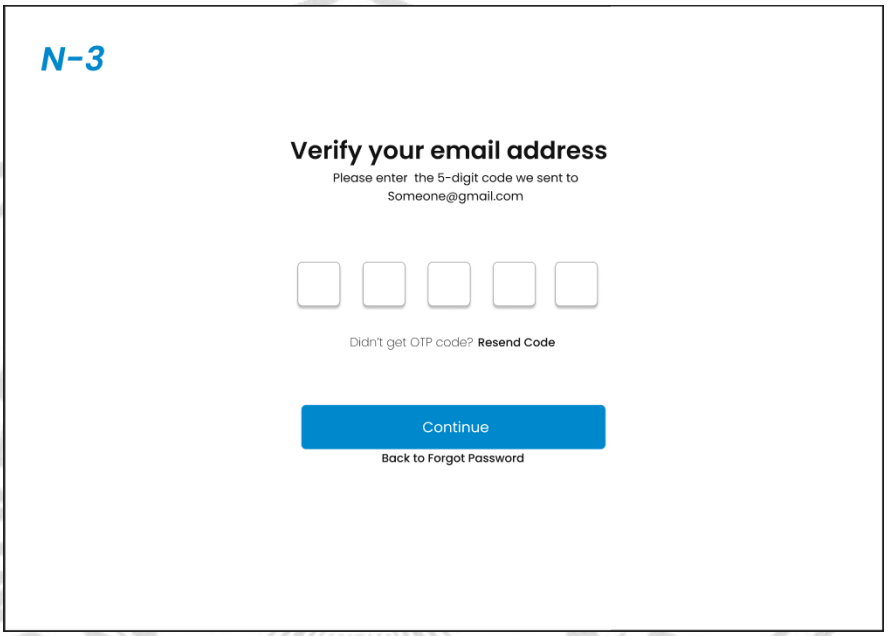
Halaman ini digunakan untuk memfasilitasi pengguna yang lupa kata sandi (password) akun mereka. Dalam sistem ini, pengguna diminta untuk memasukkan alamat email yang telah terdaftar sebelumnya. pada gambar 3.12



Gambar 3.11. Halaman Verifikasi Email

d. Halaman Verifikasi OTP

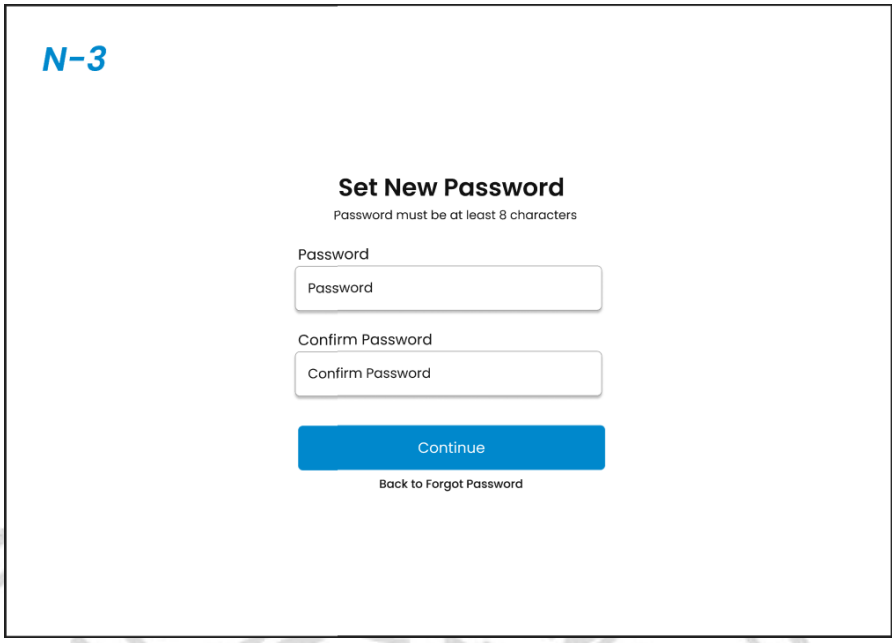
Halaman ini digunakan untuk memverifikasi identitas pengguna dengan cara memasukkan kode OTP (One-Time Password) yang dikirimkan ke alamat email yang telah dimasukkan sebelumnya pada proses lupa password.halaman Verifikasi OTP dapat dilihat pada gambar 3.13



Gambar 3.12. Halaman Verifikasi OTP

e. Halaman Atur Ulang Kata Sandi

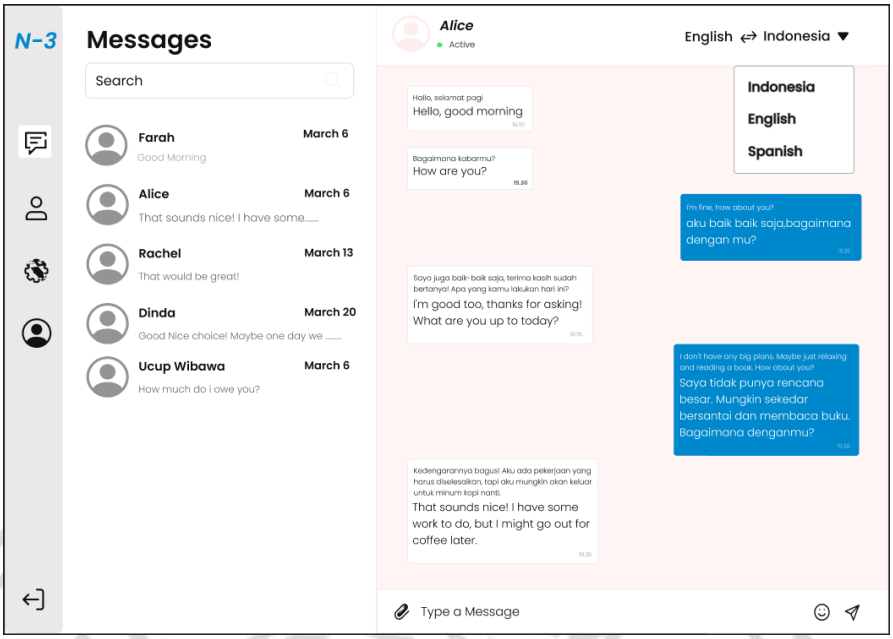
Halaman ini digunakan oleh pengguna untuk mengatur ulang kata sandi baru setelah berhasil melewati proses verifikasi kode OTP. halaman Contact dapat dilihat pada gambar 3.14



Gambar 3.13. Halaman Atur Ulang Sandi

f. Halaman Utama

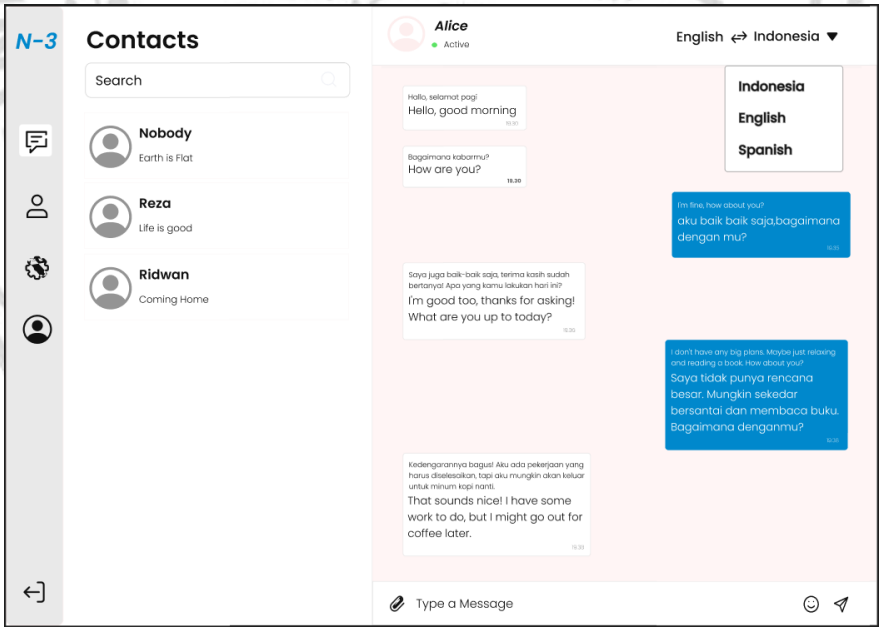
Halaman ini merupakan halaman utama dari fitur pesan, di mana pengguna dapat berkomunikasi melalui chat dengan kontak mereka. Selain fungsi utama untuk berkirim pesan, halaman ini juga dilengkapi dengan fitur autocorrect yang akan memperbaiki kesalahan ketik (typo) sebelum pesan diterjemahkan. Setelah dikoreksi, pengguna akan memilih bahasa awal dan bahasa target, lalu menerjemahkannya ke dalam bahasa yang dipilih melalui menu dropdown di bagian atas. Rancangan halaman Utama dapat dilihat pada gambar 3.14



Gambar 3.14. Halaman Utama

g. Halaman Contact

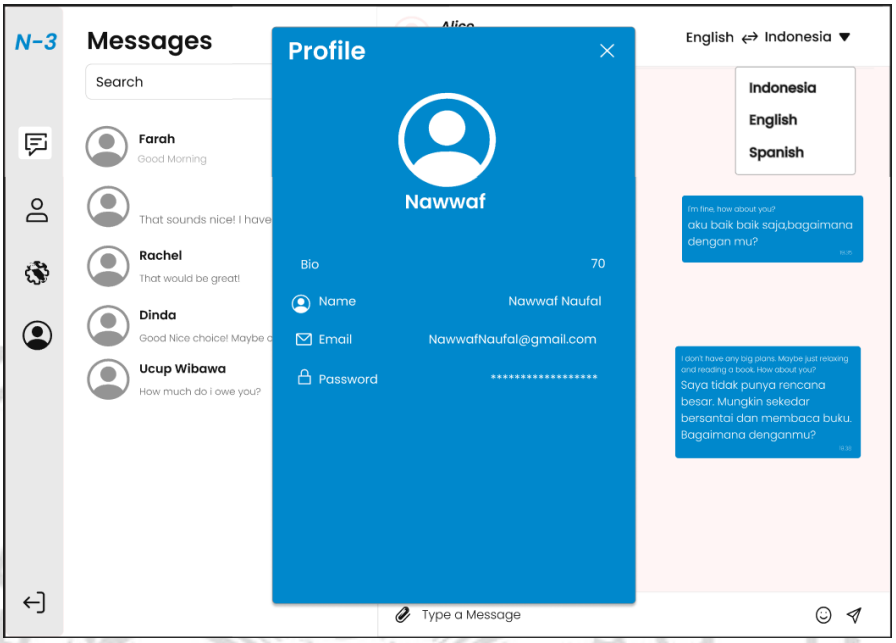
Halaman ini bertujuan untuk mencari pengguna lain untuk berkomunikasi melalui chat. Rancangan halaman Contact dapat dilihat pada gambar 3.15



Gambar 3.15. Halaman Contact

h. Halaman Profile

Halaman ini bertujuan untuk memungkinkan pengguna mengelola dan memperbarui informasi profil mereka. Rancangan halaman Profile dapat dilihat pada gambar 3.16



Gambar 3.16. Halaman Profile

3.4.6 Spesifikasi Pengembangan System

1. Kebutuhan Perangkat Keras

Perangkat keras adalah alat yang digunakan untuk menunjang dalam pembuatan sistem. Dalam pembuatan sistem ini perangkat keras yang digunakan yaitu laptop dengan spesifikasi:

a. Processor AMD Ryzen 3 3200U

b. RAM 8 GB

c. Mouse

2. Kebutuhan Perangkat Lunak

Perangkat lunak adalah program atau aplikasi yang digunakan untuk membangun sistem. Perangkat lunak yang dibutuhkan dalam pembuatan sistem ini adalah:

a. Windows 11
b. Visual Studio Code
c. Postman
d. JavaScript
e. Node.js
f. Express.js
g. React.js
h. MongoDb dengan Monggose
i. Google Chrome

3.4.7 Perencanaan Pengujian System

Adapun perencanaan pengujian dilakukan sebagai berikut:

1. Menguji koreksi otomatis pada kalimat dengan hanya satu huruf yang salah pada setiap kata. Ini untuk menilai seberapa efektif sistem dalam menangani kesalahan ketik ringan dan apakah dapat memperbaiki kata yang dimaksud dengan benar.

   Contoh Kalimat : Dia akna kembali ke rumah setelah bekerja.

2. Menguji koreksi otomatis jika terdapat lebih dari satu kata yang salah dalam satu kalimat, termasuk jika terdapat kesalahan ketik yang parah (misalnya 2 hingga 3 huruf dalam satu kata yang salah). Ini untuk menilai apakah sistem bisa melakukan koreksi secara konsisten dan tidak hanya fokus pada kata pertama yang salah

   Contoh kalimat: Kenla kmu nmenggir aku jahat.

3. Pengujian memasukkan kalimat yang benar tanpa kesalahan dan memverifikasi bahwa sistem tidak mengubah kata yang seharusnya tidak perlu dikoreksi.

   Contoh kalimat*:* Kami pergi ke pasar untuk membeli sayur dan buah.

4. Pengujian dengan kombinasi typo berat penggunaan simbol/angka dalam kata. Pengujian ini bertujuan untuk mengukur batas kemampuan koreksi otomatis sistem.

Contoh kalimat: Ak0 pergi ke m@ll untk beli sep@tuh baru

5. Menguji koreksi otomatis untuk memastikan bahwa sistem tidak hanya memperbaiki kesalahan ketik, tetapi juga dapat mempertahankan konteks kalimat secara keseluruhan. Hal ini untuk memastikan bahwa koreksi yang diberikan relevan dengan konteks pesan dan tidak mengubah makna atau tujuan dari pesan yang dimaksud

   Contoh Kalimat : Aku sedang fari di pantai

6. Menguji ketika pengguna memasukkan nama, alamat, atau nama tempat, untuk memastikan sistem tidak mengganti atau mengoreksi kata-kata yang memang sesuai dan tidak perlu diperbaiki.

   Contoh kalimat: Ibu melihat ilham berjalan di Jalan Sudirman

   Setiap jenis pengujian di atas akan dilakukan dengan menggunakan 20 data uji. Seluruh data uji disusun secara manual data uji tersebut tidak diambil dari pengguna nyata, namun dirancang agar menyerupai kondisi sebenarnya dalam penggunaan aplikasi chat Setelah pengujian dilakukan, akan diambil nilai rata-rata akurasi dari seluruh pengujian sebagai evaluasi keseluruhan terhadap performa sistem.