BAB IV

PERANCANGAN

4.1 Analisis Sistem terdahulu

Sistem tradisional yang digunakan untuk membuat pengingat pada lansia seringkali bersifat manual, seperti menggunakan catatan fisik atau alat sederhana, yang memiliki banyak keterbatasan. Lansia sering kali menghadapi kesulitan dalam mengingat jadwal harian mereka tanpa bantuan, sehingga meningkatkan ketergantungan pada pengasuh atau keluarga. Selain itu, kurangnya dukungan untuk teknologi modern, seperti pengingat berbasis suara yang ramah pengguna, membuat pengelolaan jadwal menjadi tugas yang menantang. Aplikasi yang ada saat ini sering kali tidak dirancang dengan mempertimbangkan kebutuhan dan keterbatasan lansia, seperti antarmuka yang rumit atau kurangnya dukungan untuk bahasa lokal.

Kesulitan ini diperburuk oleh fakta bahwa teknologi pengingat modern sering kali tidak mendukung perintah suara dalam bahasa lokal atau dialek yang dikenal oleh lansia. Akibatnya, banyak dari mereka merasa enggan atau tidak mampu menggunakan aplikasi modern, sehingga kehilangan kesempatan untuk mandiri dalam mengelola waktu dan menerima informasi penting. Ketergantungan pada metode manual atau bantuan pengasuh juga mengurangi efisiensi dalam kehidupan sehari-hari, yang pada akhirnya dapat meningkatkan stres baik pada lansia maupun pengasuh mereka.

Dengan latar belakang ini, menjadi jelas bahwa solusi inovatif berbasis teknologi, seperti aplikasi pengingat berbasis *Android* dengan dukungan perintah suara dan bahasa lokal, sangat dibutuhkan. Solusi semacam itu dapat membantu meningkatkan kemandirian lansia, mempermudah pengelolaan jadwal harian, dan memberikan pengalaman yang lebih inklusif serta ramah pengguna. Hal ini akan menjadi langkah penting untuk mengatasi keterbatasan sistem tradisional dan memberikan dampak positif pada kehidupan lansia.

4.2 Spesifikasi Kebutuhan Sistem Baru

Sistem baru dirancang dengan fungsionalitas utama yang mencakup fitur penjadwalan, pengenalan suara, dan penggunaan model *Natural Language Processing (NLP)*. Pada fitur penjadwalan, pengguna dapat menambahkan jadwal secara manual melalui menu di halaman utama dengan mengisi form detail pengingat. Alternatifnya, pengguna dapat menggunakan perintah suara untuk menambahkan jadwal, di mana sistem memproses perintah tersebut menggunakan *regex* (regular expression) untuk mengekstrak informasi seperti label, deskripsi, dan waktu pembuatan. Sebagai contoh, ketika pengguna mengucapkan, “Buat jadwal minum obat paracetamol jam 10 pagi,” sistem akan mengenali pola suara, mengekstrak informasi, dan menyiapkan jadwal dengan label "Minum Obat," deskripsi "Paracetamol," dan waktu "10:00."

Fitur pengenalan suara mendukung Bahasa Indonesia dengan integrasi Google *Speech-to-Text* API. Setelah perintah suara diubah menjadi teks, sistem menggunakan algoritma *regex* untuk mengekstrak informasi seperti waktu dan aktivitas yang dimaksud. Selain itu, jika pengguna mengajukan pertanyaan terkait kesehatan, seperti “Kenapa saya sering pusing?”, aplikasi akan menggunakan model NLP berbasis *IndoBERT* untuk memprediksi label dan memilih jawaban relevan dari *answer_bank.json* sesuai label yang diprediksi.

Secara teknis, framework *ONNX Runtime* digunakan untuk mengimplementasikan model *machine learning* pada perangkat *Android*. Model *IndoBERT* yang telah di-fine-tuned dioptimalkan untuk klasifikasi teks dalam Bahasa Indonesia, sehingga mampu memberikan prediksi yang akurat. Model ini bekerja secara lokal di perangkat *Android* tanpa memerlukan koneksi internet, memungkinkan performa optimal dan pengalaman pengguna yang lebih baik. Integrasi dengan *answer_bank.json* memastikan jawaban yang diberikan sesuai dengan label yang telah diprediksi oleh model.

4.2.1 Alur Aplikasi

Aplikasi ini dirancang untuk membantu lansia mengatur jadwal dan memberikan jawaban atas pertanyaan kesehatan umum melalui fitur pengingat berbasis suara dan *Natural Language Processing (NLP)*. Alur aplikasi mencakup tiga proses :

(1) Pengelolaan Jadwal: Pengguna dapat menambahkan, mengedit, dan menghapus jadwal harian baik secara manual melalui antarmuka aplikasi maupun dengan menggunakan perintah suara. Perintah suara diproses untuk mengekstrak informasi seperti aktivitas, waktu, dan deskripsi.

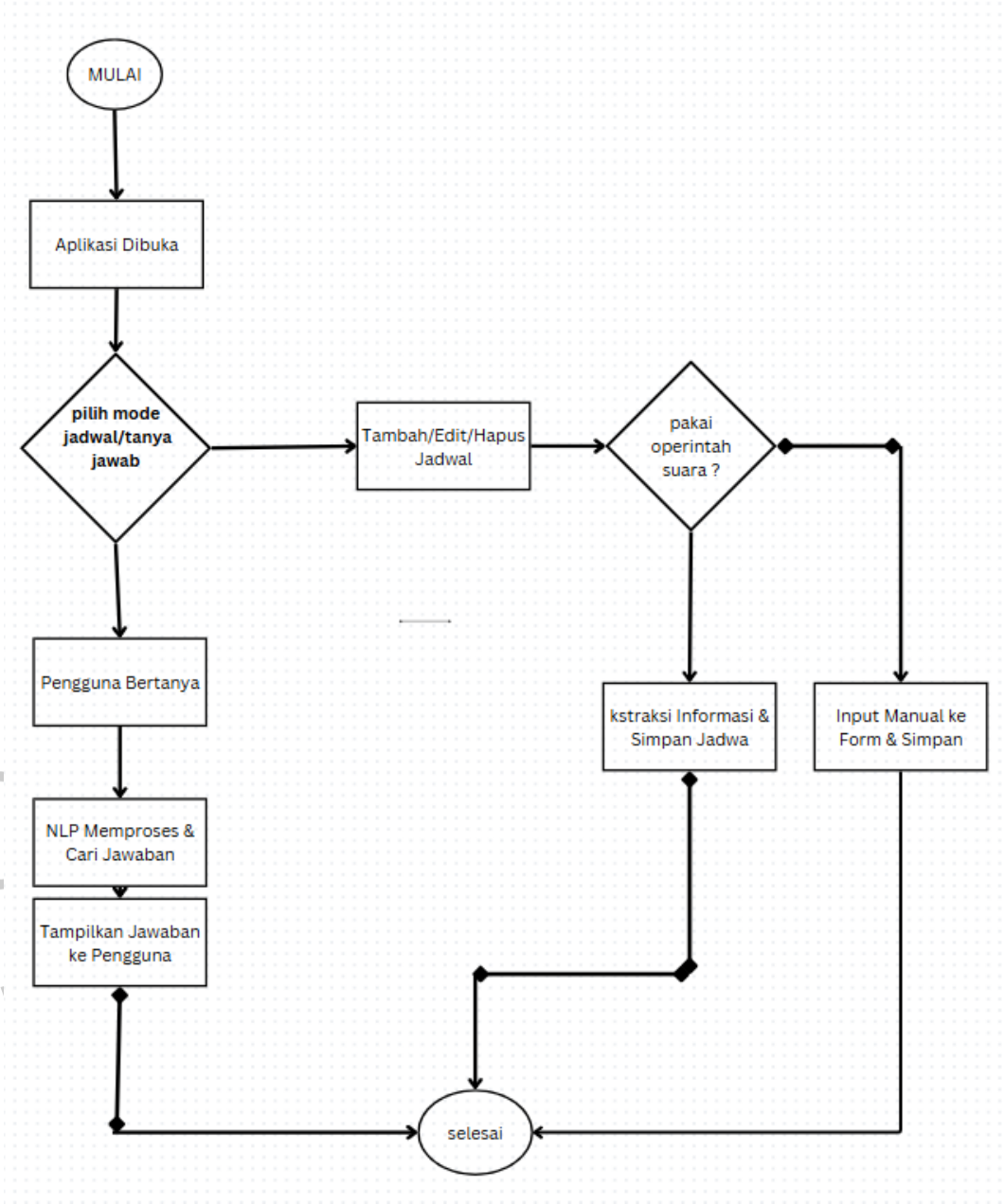
(2) Penyimpanan Data: Data jadwal disimpan secara efisien dan aman di *database* lokal menggunakan *ROOM Database*, memastikan akses cepat dan tidak memerlukan koneksi internet.

(3) Penggunaan Perintah Suara: Aplikasi mendukung fitur pengenalan suara menggunakan teknologi *SpeechRecognizer* dan *TextToSpeech*, memungkinkan pengguna untuk memberikan perintah suara dalam Bahasa Indonesia secara mudah.

(4) Proses Pertanyaan Kesehatan: Pengguna dapat mengajukan pertanyaan kesehatan umum melalui aplikasi. Sistem akan memproses pertanyaan tersebut menggunakan model NLP berbasis *IndoBERT* untuk memprediksi label.

(5) Pencarian Jawaban: Berdasarkan label yang diprediksi, sistem mengambil jawaban yang sesuai dari file *answer_bank.json*.

(6) Penyajian Jawaban: Jawaban yang relevan ditampilkan kepada pengguna melalui antarmuka aplikasi, memberikan informasi yang akurat dan membantu.



Gambar 4. 1 *Flowchart* Alur Aplikasi

Gambar 4.1 – Flowchart Alur Aplikasi menunjukkan bagaimana pengguna dapat memberikan input suara atau teks untuk mengatur jadwal dan melakukan tanya jawab kesehatan. Proses ini dimulai dari pengguna membuka aplikasi, memilih fitur yang diinginkan, hingga sistem merespons dengan jawaban atau jadwal yang telah diproses.

4.2.2 *Hardware Specification*

Hardware yang digunakan oleh peneliti, sebuah laptop yang memiliki spesifikasi berikut, digunakan untuk mendukung pengembangan sistem ini:

Tabel 4. 1 Spesifikasi Perangkat Keras

Prosesor	Processor Intel Core i5 – 10300H 3.0GHz
Memori	16 GB
Operating Sistem	Windows 11
System type	64-bit
Smartphone	*Android* 10
RAM Smartphone	6GB

4.2.3 Spesifikasi Perangkat Lunak

Aplikasi ini membutuhkan dukungan perangkat lunak, yang ditunjukkan dalam tabel berikut:

Tabel 4. 2 Spesifikasi Perangkat Lunak

IDE	*Android* Studio Ladybug (Versi 2024.2.1)
Database Android	*ROOM Database*
Build Tool *Android*	Gradle
Language	Kotlin
Android Library for *machine learning* implementation	*ONNX Runtime*
Browser	Google Chrome
Pembuatan Dataset	Microsoft Excel
Perancangan Desain Antarmuka	Figma
NLP Model Development Platform	*Google Colab*
NLP Model Type	*IndoBERT*-based Text Classification Model
Output Format of NLP Model	.safetensor and *.onnx*
Dataset Type	CSV files with health-related text labels
Training Dataset Tool	Hugging Face datasets library
Model Training Framework	PyTorch, *Transformer*s by Hugging Face

4.3 Perancangan Sistem

Bagian ini menjelaskan perancangan sistem yang dilakukan untuk mengimplementasikan solusi berbasis teknologi dalam membantu lansia mengelola jadwal dan mendapatkan jawaban atas pertanyaan kesehatan umum. Perancangan sistem mencakup representasi visual seperti *flowchart*, *diagram use case*, *diagram* aktivitas, dan *sequence diagram* untuk memvisualisasikan proses kerja sistem secara lebih mendalam.

4.3.1 Alur Sistem

Bagian ini menjelaskan alur kerja sistem yang dirancang menggunakan *flowchart* sebagai representasi visual dari proses utama dalam aplikasi. *Flowchart* ini memberikan gambaran sistematis tentang bagaimana aplikasi bekerja untuk memenuhi kebutuhan pengguna, terutama dalam mengelola jadwal dan menjawab pertanyaan kesehatan.

Pada *flowchart*, proses dimulai ketika pengguna membuka aplikasi. Pengguna diberikan dua opsi utama pada antarmuka awal, yaitu:

(1) Mengelola jadwal pengingat:

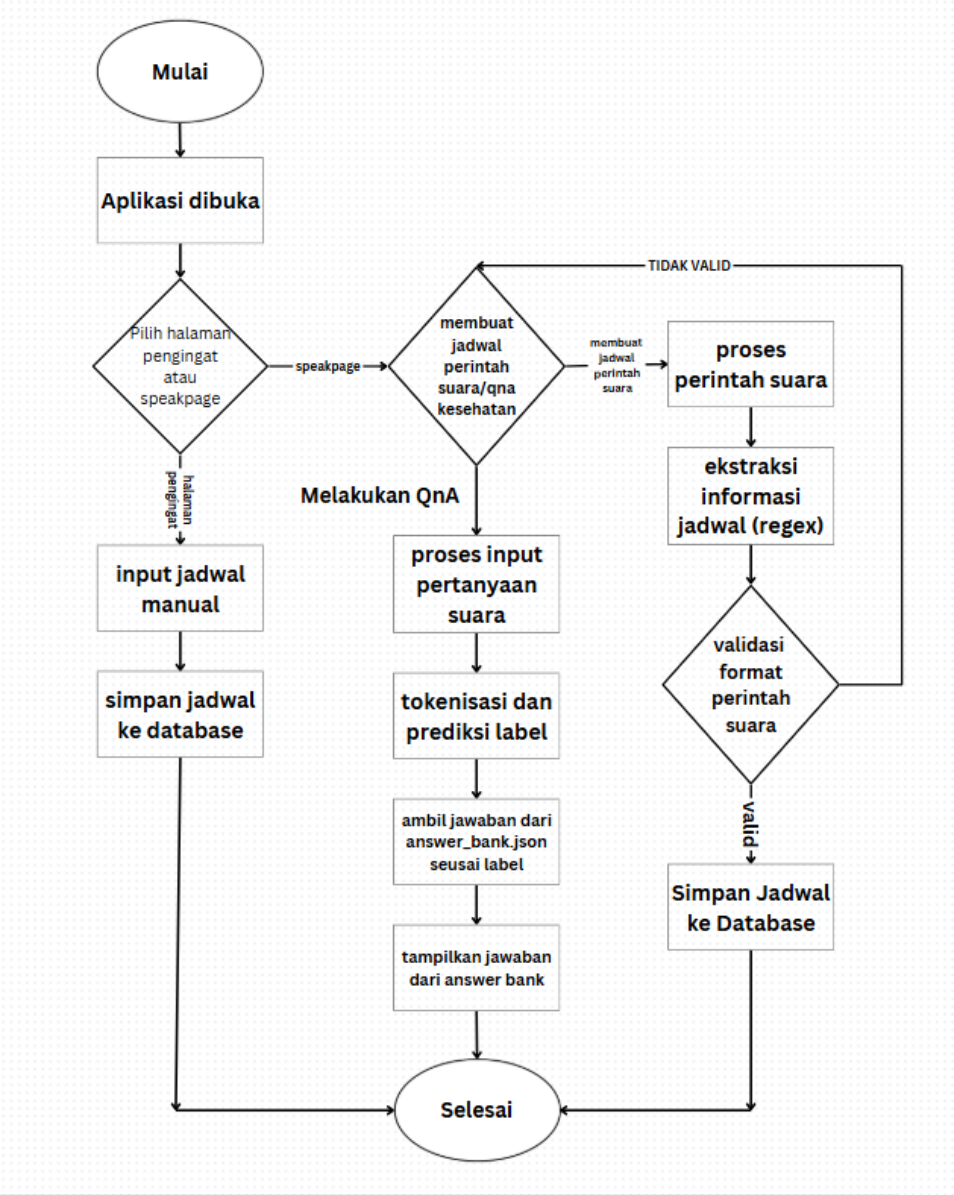
Pengguna dapat menambahkan jadwal secara manual melalui form atau menggunakan perintah suara. Jika menggunakan perintah suara, sistem akan memproses input suara, mengekstraksi informasi dengan metode *regex*, memvalidasi formatnya, dan menyimpan jadwal ke *database* lokal jika valid. Jika terjadi kesalahan pada format perintah suara, sistem akan memberikan umpan balik kepada pengguna untuk memperbaiki input.

(2) Melakukan sesi tanya jawab kesehatan (QnA):

Pengguna dapat memberikan input berupa pertanyaan kesehatan melalui fitur perintah suara. Proses pengolahan pertanyaan kesehatan pada aplikasi dimulai dengan tokenisasi dan prediksi label menggunakan model NLP berbasis *IndoBERT*. Setelah label pertanyaan berhasil diprediksi, label tersebut digunakan untuk mengambil jawaban yang relevan dari file *answer_bank.json*. Jawaban yang sesuai kemudian ditampilkan kepada

pengguna sebagai output, memberikan informasi yang akurat dan sesuai dengan kebutuhan pengguna.

Flowchart ini mengilustrasikan setiap langkah secara terstruktur, dimulai dari input pengguna hingga hasil akhir yang ditampilkan. Proses ini dirancang agar sederhana dan ramah pengguna, khususnya untuk lansia, dengan memastikan integrasi antara fitur pengingat dan sesi tanya jawab kesehatan berjalan efisien.



Gambar 4. 2 *Flowchart* Alur sistem

Flowchart ini menggambarkan alur sistem asisten digital pintar berbasis Android yang dirancang untuk membantu lansia dalam mengelola jadwal dan mendapatkan informasi kesehatan melalui pengenalan suara dan Natural Language Processing (NLP). Sistem dimulai ketika pengguna membuka aplikasi dan memilih antara pengelolaan jadwal atau tanya jawab kesehatan (QnA). Jika pengguna ingin mengelola jadwal, mereka dapat memasukkan jadwal secara manual atau menggunakan perintah suara, yang akan diproses dan divalidasi sebelum disimpan ke database. Jika pengguna memilih fitur QnA, sistem akan menerima input suara, mengolahnya menggunakan NLP untuk memprediksi intent, mencocokkannya dengan jawaban yang tersedia dalam basis data, lalu menampilkan respons kepada pengguna. Proses ini memastikan bahwa aplikasi dapat berfungsi secara efisien untuk membantu lansia dalam aktivitas sehari-hari dengan antarmuka yang sederhana dan ramah pengguna.

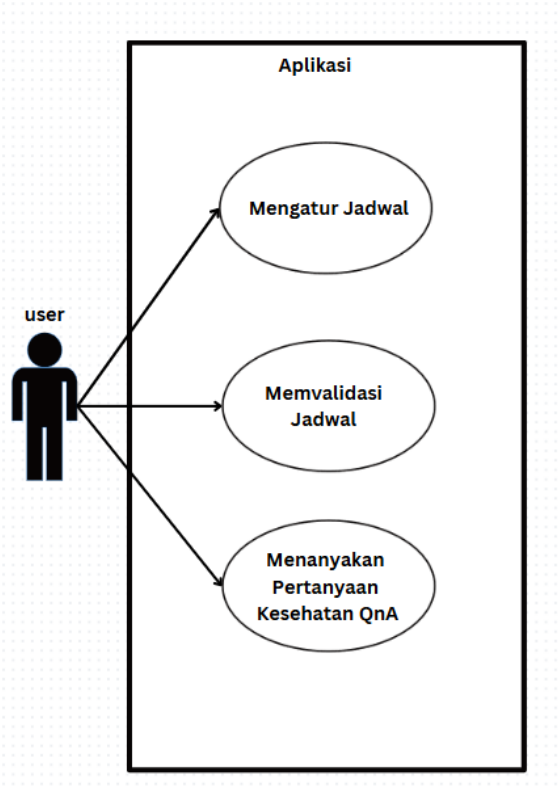
4.3.2 Use Case

Use Case Diagram yang ditampilkan pada Gambar 4.2 menggambarkan interaksi antara pengguna (lansia) dan sistem aplikasi yang dirancang. *Diagram* ini memvisualisasikan fungsi utama aplikasi, yaitu mengatur jadwal, perintah suara, dan pertanyaan kesehatan.

Pada fungsi mengatur jadwal, pengguna dapat melakukan berbagai aktivitas, seperti menambahkan jadwal, mengedit jadwal, dan menghapus jadwal yang sudah ada. Fungsi ini memberikan fleksibilitas kepada pengguna dalam mengelola aktivitas harian mereka. Untuk fungsi perintah suara, pengguna dapat memberikan input berupa suara untuk membuat atau mengelola jadwal. Sistem akan memproses input suara tersebut menggunakan *regex* (regular expression) untuk memastikan informasi seperti waktu dan aktivitas telah tervalidasi sebelum disimpan ke *database* lokal.

Selain itu, aplikasi menyediakan fungsi pertanyaan kesehatan, yang memungkinkan pengguna untuk bertanya mengenai masalah kesehatan umum. Sistem akan memproses pertanyaan tersebut menggunakan model *Natural Language Processing (NLP)* berbasis *IndoBERT* untuk memprediksi label yang relevan, kemudian

mencocokkan jawaban dari *answer_bank.json*. Jawaban yang sesuai akan ditampilkan kepada pengguna secara langsung.



Gambar 4. 3 *Use Case Diagram*

Gambar 4.2 – *Use Case Diagram* menggambarkan interaksi antara pengguna dan sistem. Dalam diagram ini, pengguna memiliki dua peran utama, yaitu mengatur jadwal dan melakukan tanya jawab kesehatan. Diagram ini membantu dalam memahami bagaimana setiap fitur terhubung dengan pengguna.

Agar informasi tentang skenario *Use Case* dapat disajikan dengan jelas dan mudah dipahami, maka perlu dibuat tabel skenario. Tabel skenario harus dibuat dengan pihak yang terlibat, nama skenario, ringkasan singkat, skenario normal, dan skenario alternatif. Tabel skenario ini harus dibuat agar informasi tentang skenario *use case* jelas dan mudah dipahami.

Tabel 4. 3 Tabel Skenario Mengatur Jadwal

Use Case	Mengatur Jadwal
Penjelasan	Pengguna dapat mengelola jadwal, seperti menambah, melihat, mengedit, dan menghapus jadwal.

Aktor	*User*
Skenario Utama	1. Pengguna memilih fitur mengatur jadwal. 2. Pengguna memilih aksi (Tambah/Edit/Hapus). 3. Sistem memproses permintaan dan menyimpan perubahan di *database*.
Skenario Alternatif	1. Pengguna memasukkan data jadwal yang tidak valid. 2. Sistem memberikan pesan error dan meminta pengguna memperbaiki data.
Kondisi Akir	Jadwal berhasil dikelola (ditambahkan, diedit, atau dihapus) sesuai permintaan pengguna.

Tabel 4. 4 Tabel Skenario validasi jadwal

Use Case	Memvalidasi Jadwal
Penjelasan	Sistem memvalidasi data jadwal sebelum disimpan ke *database*.
Aktor	*User*, Sistem
Skenario Utama	1. Pengguna mengirim data jadwal. 2. Sistem memvalidasi format data. 3. Jika valid, data disimpan di *database*. 4. Sistem memberikan konfirmasi penyimpanan kepada pengguna.
Skenario Alternatif	1. Data tidak valid. 2. Sistem memberikan pesan error kepada pengguna.
Kondisi Akir	Jadwal tersimpan di *database* jika data valid, atau pengguna menerima pesan error jika data tidak valid.

Tabel 4. 5 Tabel Skenario Pertanyaan Kesehatan

Use Case	Menanyakan Pertanyaan Kesehatan (QnA)
Penjelasan	Pengguna bertanya tentang masalah kesehatan umum menggunakan aplikasi.
Aktor	*User*
Skenario Utama	1. Pengguna memilih fitur pertanyaan kesehatan. 2. Pengguna memberikan input pertanyaan. 3. Sistem memproses pertanyaan dengan NLP. 4. Sistem mengambil jawaban dari bank jawaban. 5. Jawaban ditampilkan kepada pengguna.

Skenario Alternatif	1. Sistem gagal memproses pertanyaan karena input tidak sesuai format. 2. Sistem memberikan pesan error kepada pengguna.
Kondisi Akhir	Jawaban pertanyaan kesehatan berhasil ditampilkan kepada pengguna, atau pengguna menerima pesan error jika input tidak sesuai format.

4.3.3 *Diagram Activity*

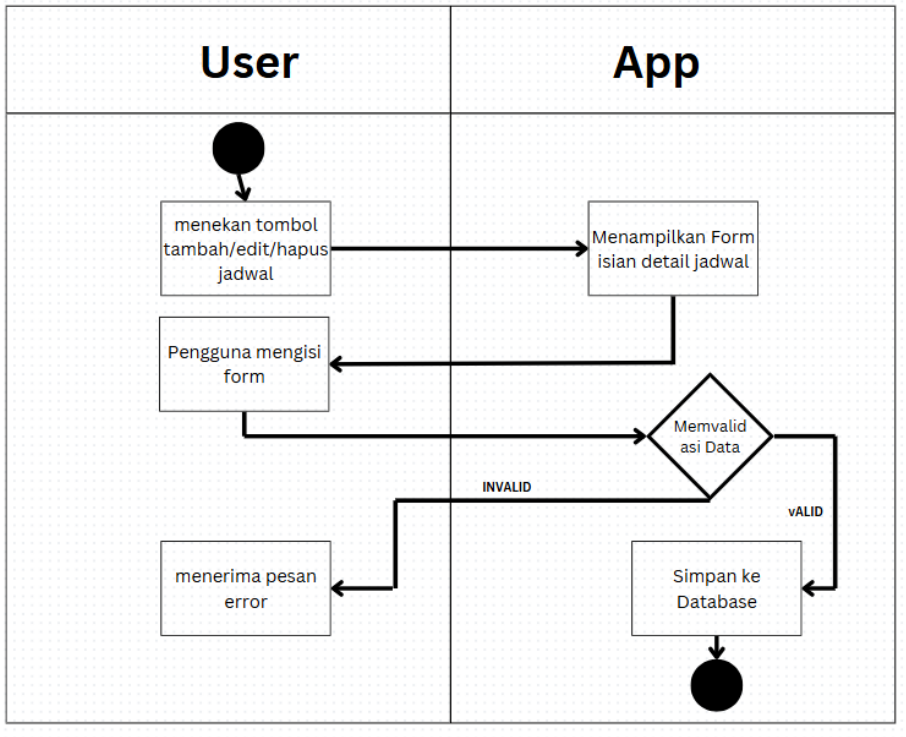
Diagram Activity dalam sistem ini dirancang untuk memberikan gambaran alur kerja utama yang terjadi antara pengguna dan aplikasi, mencakup fitur-fitur inti seperti pengaturan jadwal, perintah suara, dan tanya jawab kesehatan. *Diagram* ini memvisualisasikan langkah-langkah yang dimulai dari input pengguna, proses validasi dan pengolahan data oleh aplikasi, hingga penyelesaian aktivitas dengan menyimpan data atau menampilkan hasil yang sesuai. Setiap proses dirancang untuk memastikan alur kerja berjalan dengan lancar dan memenuhi kebutuhan pengguna.

Pada fitur pengaturan jadwal, *diagram* menunjukkan proses ketika pengguna memulai dengan menekan tombol tambah jadwal, mengisi formulir sesuai format yang ditentukan, hingga aplikasi menyimpan data ke dalam *database* lokal. Untuk fitur perintah suara, *diagram* menjelaskan bagaimana aplikasi memproses input suara menggunakan algoritma *regex* untuk mengekstrak informasi seperti waktu dan deskripsi aktivitas, yang kemudian divalidasi dan disimpan. Kedua fitur ini memastikan data dikelola secara efisien dan aman.

Selain itu, pada fitur tanya jawab kesehatan, *diagram* menggambarkan proses ketika pengguna memberikan pertanyaan, yang kemudian diolah oleh model NLP berbasis *IndoBERT* untuk memprediksi label yang relevan. Label ini digunakan untuk mengambil jawaban yang sesuai dari *answer_bank.json*, yang ditampilkan kembali kepada pengguna. *Diagram Activity* ini memberikan pemahaman yang jelas mengenai bagaimana aplikasi bekerja secara sistematis untuk menangani kebutuhan pengguna, baik untuk manajemen jadwal maupun informasi kesehatan.

(1) *Diagram Activity* Mengatur Jadwal

Diagram Activity untuk fitur Mengatur Jadwal menggambarkan alur proses yang dimulai dari pengguna menekan tombol tambah jadwal di aplikasi. Aplikasi kemudian menampilkan formulir isian detail jadwal yang harus diisi oleh pengguna sesuai dengan format yang telah ditentukan. Setelah formulir diisi, data dikirimkan kembali ke aplikasi untuk dilakukan validasi. Jika data valid, aplikasi akan menyimpan jadwal tersebut ke dalam *database* lokal, menandai selesainya proses. *Diagram* ini menunjukkan interaksi yang jelas antara pengguna dan aplikasi, memastikan pengelolaan jadwal dilakukan dengan efisien dan aman.



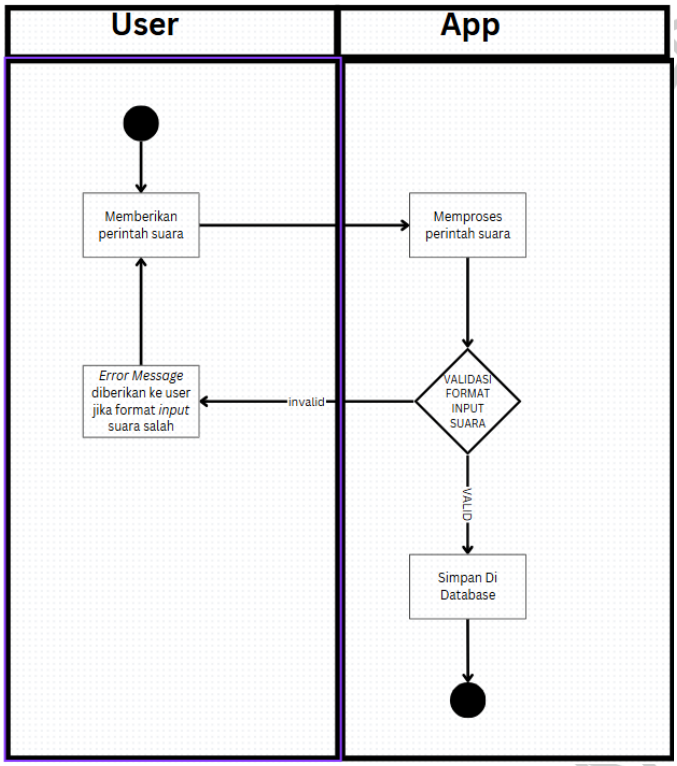
Gambar 4. 4 *Diagram Activity* Mengatur Jadwal

Gambar 4.4 Diagram Activity Mengatur Jadwal menjelaskan langkah-langkah yang dilakukan pengguna dalam menambahkan jadwal melalui input suara atau teks.

(2) *Diagram Activity* validasi perintah suara

Diagram aktivitas Penjadwalan Perintah Suara menunjukkan alur kerja fitur aplikasi saat pengguna membuat jadwal menggunakan perintah suara. Terdapat dua swimlane, yaitu *User* dan App, yang menggambarkan peran pengguna dan sistem dalam proses ini.

Proses dimulai ketika pengguna memberikan perintah suara melalui aplikasi. Sistem mengolah perintah tersebut menggunakan teknologi pengenalan suara untuk mengonversinya menjadi teks. Selanjutnya, sistem memvalidasi format teks menggunakan algoritma *regex* untuk memastikan informasi seperti waktu dan deskripsi sudah sesuai. Jika ditemukan kesalahan pada format, aplikasi akan memberikan pesan kesalahan kepada pengguna. Namun, jika format valid, sistem menyimpan data jadwal ke *database* lokal. Alur ini memastikan proses pembuatan jadwal melalui perintah suara berjalan lancar dan data yang dimasukkan sesuai standar.



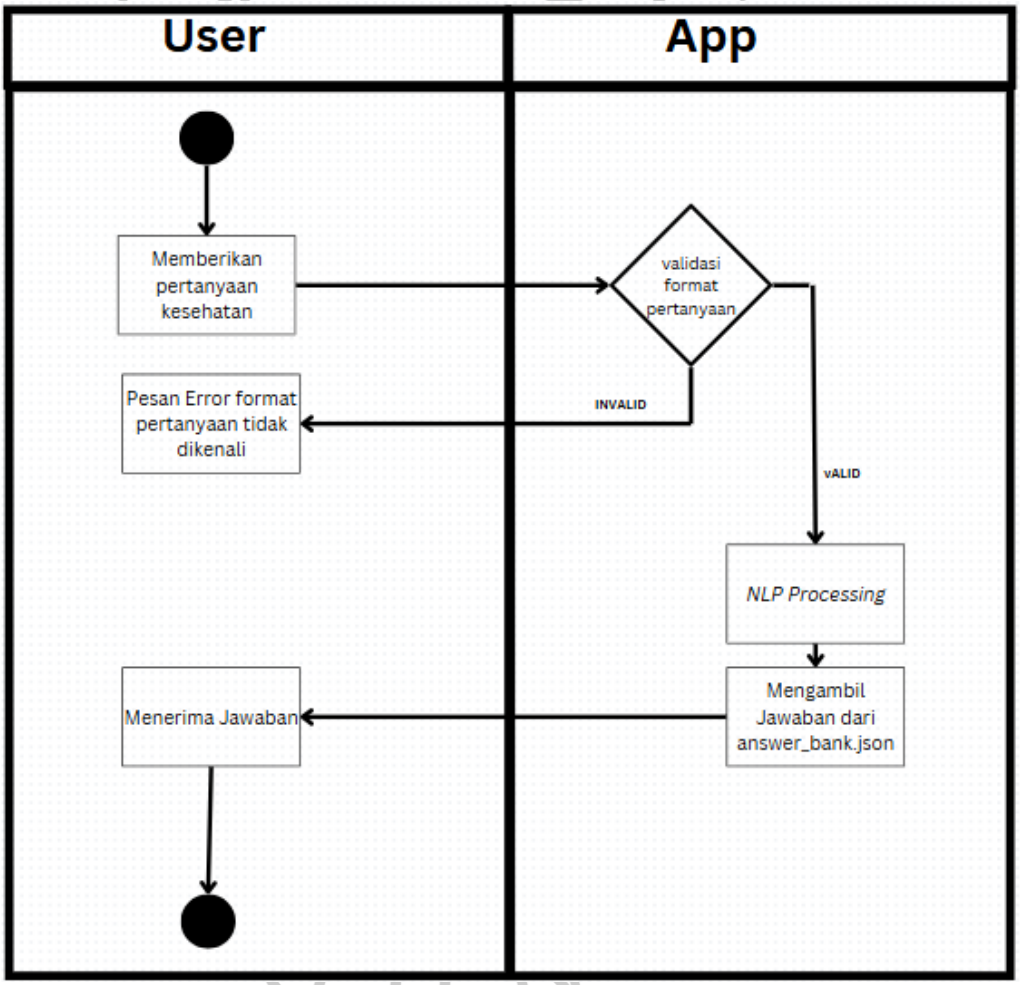
Gambar 4. 5 *Diagram Activity* Penjadwalan Perintah Suara

Gambar 4.5 – Diagram Activity Penjadwalan Perintah Suara menunjukkan bagaimana sistem mengonversi perintah suara menjadi teks dan memvalidasi formatnya sebelum menyimpannya dalam database.

(3) *Diagram Activity* Pertanyaan Kesehatan

Diagram aktivitas ini menggambarkan proses alur kerja ketika pengguna mengajukan pertanyaan kesehatan melalui aplikasi. Proses dimulai dari pengguna memberikan input pertanyaan di antarmuka aplikasi. Aplikasi

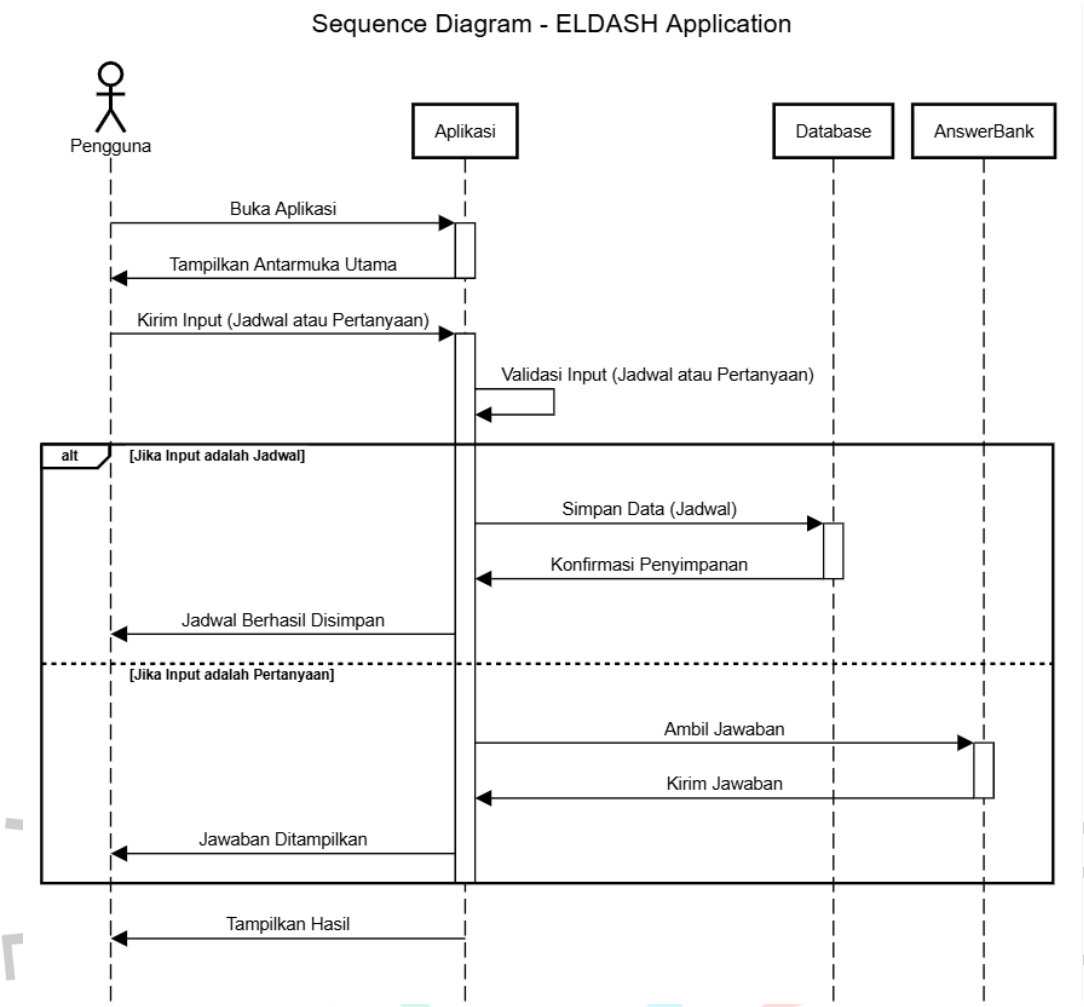
kemudian menggunakan model *Natural Language Processing (NLP)* berbasis *IndoBERT* untuk memprediksi label pertanyaan yang diajukan. Prediksi label ini digunakan untuk mencocokkan jawaban yang relevan di Answer Bank. Setelah jawaban ditemukan, aplikasi mengirimkan informasi tersebut kepada pengguna. *Diagram* ini memastikan bahwa setiap pertanyaan diproses secara sistematis, memberikan hasil yang relevan dan informatif kepada pengguna.).



Gambar 4. 6 *Diagram Activity* Pertanyaan Kesehatan

Gambar 4.6 – Diagram Activity Pertanyaan Kesehatan menggambarkan bagaimana sistem memproses pertanyaan pengguna dan mencari jawaban dari answer bank.

4.3.4 *Sequence Diagram*



Gambar 4. 7 *Sequence Diagram*

Sequence diagram pada Gambar 4.6 menggambarkan alur kerja aplikasi ELDASH yang dirancang untuk membantu pengguna, terutama lansia, dalam mengelola jadwal harian dan mendapatkan jawaban atas pertanyaan kesehatan. *Diagram* ini melibatkan empat komponen utama, yaitu Pengguna, Aplikasi, *Database*, dan *AnswerBank*. Proses dimulai ketika pengguna membuka aplikasi, yang kemudian menampilkan antarmuka utama. Selanjutnya, pengguna dapat memberikan input berupa jadwal atau pertanyaan kesehatan. Jika input berupa jadwal, aplikasi akan memvalidasi format input, menyimpan data ke dalam *database* lokal, dan memberikan konfirmasi bahwa jadwal telah berhasil disimpan. Sebaliknya, jika input berupa pertanyaan kesehatan, aplikasi akan memproses pertanyaan menggunakan model NLP berbasis *IndoBERT* untuk memprediksi label yang sesuai, mengambil jawaban dari *AnswerBank*, dan menampilkan jawaban tersebut kepada pengguna. Aktivitas dari masing-masing komponen

divisualisasikan melalui batang aktivasi (*activation bar*), yang menunjukkan kapan komponen tersebut aktif selama proses berlangsung. *Sequence diagram* ini menegaskan efisiensi aplikasi dalam menangani dua jenis input yang berbeda dengan alur kerja yang terstruktur dan responsif.

4.3.5 Perancangan Desain Antarmuka

(1) Halaman *Splash Screen*

Gambar 4. 8 Halaman *Splashcreen*

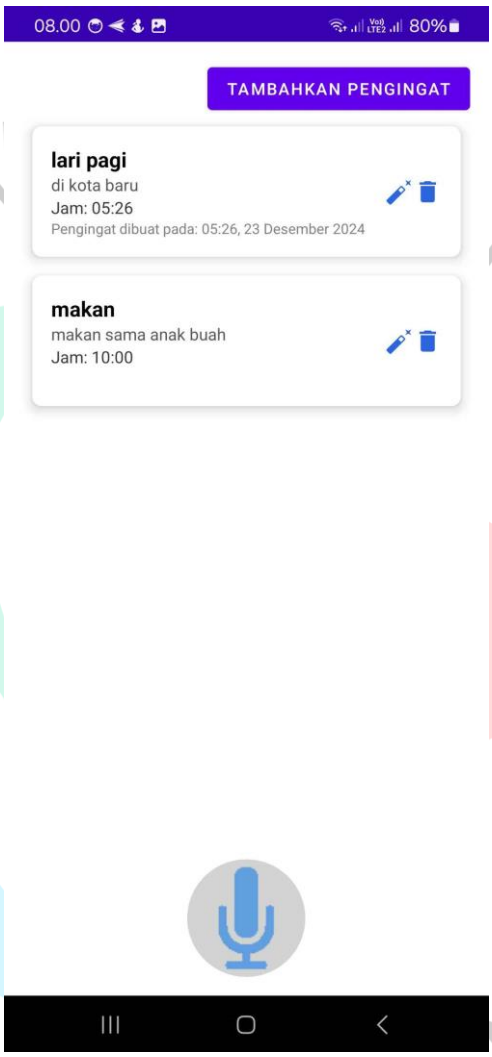
Gambar 4.8 Halaman *Splashscreen* merupakan tampilan awal aplikasi sebelum pengguna masuk ke menu utama.

Tabel 4. 6 komponen *splashscreen*

Komponen	Fungsi
Logo ELDASH	Menampilkan identitas aplikasi

Tagline	Menunjukan tujuan aplikasi “Sahabat Lansia Anda”.
Background Putih	Membuat Tampilan Bersih dan Sederhna

(2) Halaman Penjadwalan



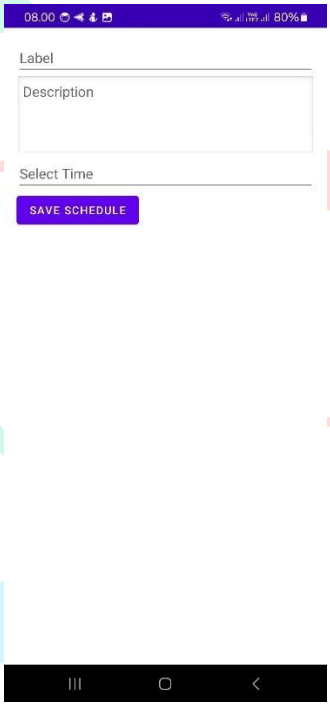
Gambar 4. 9 *MainActivity*

Gambar 4.8 – MainActivity adalah tampilan utama aplikasi yang berisi akses ke fitur utama.

Tabel 4. 7 Komponen *MainActivity*

Komponen	Fungsi
Header (Tambahkan Pengingat)	Tombol untuk menambahkan pengingat baru secara manual dengan membuka form input pengingat.
Daftar Pengingat (Card)	Menampilkan daftar pengingat yang telah dibuat pengguna.
Ikon Edit (Pensil)	Memberikan opsi kepada pengguna untuk mengedit informasi yang ada pada pengingat.
Ikon Hapus (Tempat Sampah)	Memberikan opsi kepada pengguna untuk menghapus pengingat yang tidak lagi diperlukan.
Tombol Mikrofon (Lingkaran Mikrofon)	Tombol untuk mengaktifkan fitur perintah suara yang memungkinkan pengguna menambahkan jadwal melalui input suara.

(3) Halaman Tambah Jadwal



Gambar 4. 10 Tambah Jadwal

Gambar 4.9 – Tambah Jadwal menunjukkan bagaimana pengguna dapat menambahkan jadwal baru.

Tabel 4. 8 Komponen Tambah Jadwal

Komponen	Fungsi
Input Label	Mengisi nama atau label pengingat.
Input Deskripsi	Menambahkan penjelasan tambahan untuk pengingat
Input Waktu	Menyimpan data jadwal ke dalam *database* aplikasi.

Tombol Simpan Jadwal	Menyimpan data jadwal ke dalam *database* aplikasi.

(4) Halaman Edit Jadwal

Gambar 4. 11 *Edit* Jadwal

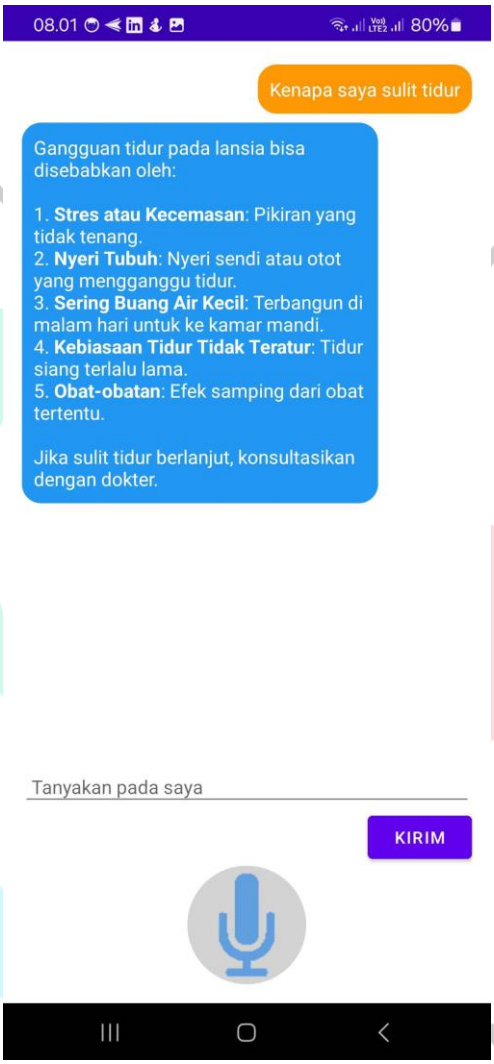
Gambar 4.10 – Edit Jadwal menggambarkan fitur pengeditan jadwal yang sudah tersimpan.

Tabel 4. 9 Komponen *Edit* Jadwal

Komponen	Fungsi
Input Label	Memperbarui nama atau label pengingat.
Input Deskripsi	Memperbarui detail atau deskripsi pengingat.

Input Waktu	Mengubah waktu pengingat sesuai kebutuhan.
Tombol Simpan Perubahan	Menyimpan perubahan jadwal ke dalam *database*.

(5) Halaman *SpeakPage*



Gambar 4. 12 *SpeakPage*

Gambar 4.11 – SpeakPage merupakan antarmuka untuk memberikan perintah suara.

Tabel 4. 10 Komponen *SpeakPage*

Komponen	Fungsi
Tombol Mikrofon	Memulai rekaman suara untuk pengingat atau pertanyaan.

Kotak Percakapan	Menampilkan perintah suara dan jawaban dari sistem.
Area Input Teks	Memungkinkan pengguna mengetikkan pertanyaan manual.
Tombol Kirim	Mengirim input teks ke sistem untuk diproses oleh model NLP.

4.3.6 Spesifikasi Model NLP

(1) Pendekatan dan Algoritma

Model yang digunakan adalah IndoBERT, sebuah model transformer berbasis BERT yang telah diadaptasi untuk Bahasa Indonesia (Koto et al., 2020). Pendekatan ini mengandalkan transfer learning, di mana model pre-trained dilatih ulang pada dataset spesifik untuk tugas klasifikasi teks (Do & Ng, 2006).

Pendekatan utamanya adalah memanfaatkan model IndoBERT yang telah dilatih pada korpus Bahasa Indonesia umum, kemudian model tersebut dilatih ulang pada dataset pertanyaan kesehatan lansia untuk menghasilkan prediksi yang lebih relevan. Model ini digunakan karena memiliki kemampuan dalam memahami konteks Bahasa Indonesia dengan lebih baik dibandingkan model berbasis bahasa lain.

Metode yang digunakan :

Tabel 4. 11 Metode NLP

Metode	Penjelasan
Tokenization	Mengubah teks menjadi representasi numerik (token) menggunakan *tokenizer IndoBERT*.
Transformer Encoding	Memahami konteks dengan mempertimbangkan kata sebelum dan sesudah.
Softmax Classification	Menentukan label output berdasarkan probabilitas tertinggi.

(2) Desain Dataset

Dataset ini terdiri dari pertanyaan dan label yang mewakili berbagai kategori jawaban. Untuk keperluan pengembangan model, dataset ini dibagi menjadi tiga kelompok utama, yaitu: dataset pelatihan (*training dataset*) yang digunakan untuk melatih model, dataset validasi (*validation dataset*) yang

digunakan untuk mengevaluasi performa model selama pelatihan, dan dataset uji *(test dataset*) yang digunakan untuk mengukur kemampuan generalisasi model. Dataset ini mencakup teks dan label yang mengelompokkan jawaban sesuai dengan kategori yang relevan, sehingga dapat digunakan secara efektif dalam melatih dan menguji model.

Tabel 4. 12 Contoh Struktur Dataset

Text	Label
Kenapa kepala saya sering pusing?	penyebab_pusing
Bagaimana cara meredakan pusing?	meredakan_pusing
Apa yang bisa dilakukan agar tidak pusing?	pencegahan_pusing
Kenapa sendi saya terasa nyeri?	penyebab_nyeri_sendi
Bagaimana cara mengurangi nyeri sendi?	mengurangi_nyeri_sendi
Apa yang bisa dilakukan untuk mencegah nyeri?	pencegahan_nyeri_sendi
Aktivitas apa yang aman saat nyeri sendi?	aktivitas_saat_nyeri_sendi
Kenapa saya sulit tidur?	penyebab_gangguan_tidur
Bagaimana cara tidur nyenyak?	cara_tidur_nyenyak
Apa pola tidur yang sehat untuk lansia?	pola_tidur_sehat
Kenapa nafsu makan saya menurun?	penyebab_nafsu_makan_berkurang
Bagaimana cara meningkatkan nafsu makan?	meningkatkan_nafsu_makan
Apa pola makan sehat untuk lansia?	pola_makan_sehat
Bagaimana cara merawat kulit kering?	merawat_kulit_kering
Kebiasaan apa yang baik untuk menjaga kulit?	kebiasaan_menjaga_kulit
Kenapa rambut saya rontok?	penyebab_rambut_rontok
Bagaimana cara merawat rambut rontok?	perawatan_rambut
Bagaimana menjaga kesehatan rambut?	menjaga_kesehatan_rambut
Apa yang bisa dilakukan saat rambut rontok?	aktivitas_saat_rambut_rontok
Obat apa yang aman untuk rambut rontok?	obat_rambut_rontok
Bagaimana cara mencegah rambut rontok?	pencegahan_rambut_rontok

Dataset ini telah diproses untuk tokenisasi dan *encoding* agar dapat digunakan dalam model berbasis *IndoBERT*.

(3) Evaluasi Model

Model dilatih menggunakan pendekatan *Transfer Learning* pada *IndoBERT* dengan konfigurasi yang dirancang untuk menghasilkan performa optimal. Proses pelatihan menggunakan *optimizer AdamW* dan fungsi *loss CrossEntropyLoss*. *Hyperparameter* yang digunakan meliputi learning rate sebesar 2e-5, jumlah epoch sebanyak 8, dan ukuran *batch* sebesar 8. Dataset yang digunakan dibagi menjadi tiga kelompok, yaitu 80% untuk training dataset, 10% untuk *validation dataset*, dan 10% untuk *test dataset*. Selama proses pelatihan, model menunjukkan penurunan nilai training loss dan *validation loss* secara konsisten, yang mengindikasikan bahwa model mampu mempelajari pola data secara efektif. Hasil evaluasi pelatihan dirangkum pada tabel berikut untuk memberikan gambaran performa model.

Tabel 4. 13 Proses Pelatihan Model

Epoch	Training Loss	Validation Loss
1	0.268	0.347
2	0.085	0.392
3	0.030	0.432
4	0.028	0.408
5	0.025	0.407
6	0.024	0.406
7	0.021	0.404
8	0.020	0.403

Setelah pelatihan, model diuji menggunakan test dataset untuk mengukur performa generalisasi. Hasil evaluasi berdasarkan *Classification Report* adalah sebagai berikut:

Tabel 4. 14 *Classification Report*

Label	*Precision*	*Recall*	*F1-Score*	Support
penyebab_pusing	1.00	1.00	1.00	10
meredakan_pusing	0.91	1.00	0.95	10
pencegahan_pusing	1.00	0.90	0.95	10

penyebab_nyeri_sendi	1.00	1.00	1.00	10
mengurangi_nyeri_sendi	0.83	1.00	0.91	10
pencegahan_nyeri_sendi	1.00	1.00	1.00	10
aktivitas_saat_nyeri_sendi	1.00	0.80	0.89	10
penyebab_gangguan_tidur	1.00	1.00	1.00	10
cara_tidur_nyenyak	1.00	1.00	1.00	10
pola_tidur_sehat	1.00	1.00	1.00	10
penyebab_nafsu_makan_berkurang	1.00	1.00	1.00	10
meningkatkan_nafsu_makan	0.91	1.00	0.95	10
pola_makan_sehat	1.00	0.90	0.95	10
merawat_kulit_kering	1.00	0.80	0.89	10
kebiasaan_menjaga_kulit	0.83	1.00	0.91	10
penyebab_rambut_rontok	1.00	1.00	1.00	10
perawatan_rambut	0.90	0.90	0.90	10
menjaga_kesehatan_rambut	0.78	0.70	0.74	10
aktivitas_saat_rambut_rontok	1.00	1.00	1.00	10
obat_rambut_rontok	1.00	1.00	1.00	10
pencegahan_rambut_rontok	0.73	0.80	0.76	10
small_talk	1.00	1.00	1.00	10
pertanyaan_di_luar_medical_qna	1.00	1.00	1.00	10

(4) Arsitektur Model

Arsitektur model yang digunakan pada sistem ini berbasis *IndoBERT*, sebuah model *transformer* yang telah diadaptasi untuk Bahasa Indonesia. Model ini dirancang untuk tugas klasifikasi teks, di mana input berupa teks diubah menjadi representasi numerik melalui proses tokenisasi. Setelah itu, representasi ini diproses oleh beberapa lapisan *transformer* untuk memahami konteks antar kata dalam kalimat.

Model terdiri dari beberapa komponen utama:

1. *Tokenization*: Menggunakan *tokenizer IndoBERT* untuk mengubah teks menjadi token numerik yang dapat dipahami oleh model.
2. *Transformer* Encoder: Lapisan bertumpuk yang digunakan untuk memproses token dan memahami konteks antar kata.

3. *Output Layer*: Menggunakan fungsi *Softmax* untuk menentukan probabilitas setiap label, sehingga model dapat memilih label dengan probabilitas tertinggi sebagai prediksi.

Arsitektur model ini memanfaatkan transfer learning dari model *IndoBERT* yang telah dilatih pada corpus Bahasa Indonesia. Model kemudian di-fine-tuned menggunakan dataset khusus yang dikembangkan untuk aplikasi ini, sehingga dapat memberikan prediksi label yang relevan dengan kebutuhan sistem.

4.3.7 Pendekatan dan Rumus NLP

Model NLP yang digunakan untuk aplikasi ini berbasis *IndoBERT*, dengan mekanisme klasifikasi label berdasarkan probabilitas tertinggi. Berikut adalah langkah-langkah utama dan rumus yang digunakan:

1. **Tokenisasi**

   Input berupa teks diubah menjadi token numerik menggunakan tokenizer *IndoBERT*. Token ini akan digunakan sebagai masukan untuk model

2. **Transformer Encoding**

   Token yang dihasilkan dari langkah sebelumnya diproses oleh lapisan *transformer* untuk memahami konteks antar kata. Lapisan *transformer* ini menggunakan mekanisme perhatian (*attention mechanism*) untuk memprioritaskan informasi yang relevan.

3. **Softmax untuk Prediksi Label**

   Setelah *encoding* oleh *transformer*, sistem menghasilkan skor logit (z) untuk setiap label. Skor ini diubah menjadi probabilitas menggunakan fungsi *Softmax*, yang dirumuskan sebagai berikut:

$$P(yi \mid x) = \frac{e^{Zi}}{\sum_{j=1}^{n} Zj}$$

1) P(yi|x)

Probabilitas P(yi|x) dalah probabilitas bahwa label yi benar, diberikan input x. Nilai ini berada dalam rentang antara 0 dan 1, dan semua probabilitas untuk seluruh label dijumlahkan hingga 1. Probabilitas ini menunjukkan seberapa besar keyakinan model bahwa label tertentu adalah hasil yang benar.

2) e^{Zi}

Skor logit e^{Zi} merupakan output mentah dari model sebelum fungsi Softmax diterapkan. Eksponen alami e^{Zi} digunakan untuk mengubah skor logit e^{Zi} menjadi nilai positif yang dapat dinormalisasi. Proses ini memungkinkan semua skor mentah dikonversi menjadi bentuk yang lebih dapat dibandingkan.

3) $\sum_{j=1}^{n} Zj$

Adalah jumlah dari semua eksponen skor logit untuk seluruh label j. Penyebut ini memastikan bahwa total probabilitas untuk semua label y1,y2,….,yn selalu sama dengan 1. Hal ini memungkinkan setiap probabilitas menjadi proporsi dari keseluruhan.

4) n

Adalah jumlah total label yang tersedia dalam model. Sebagai contoh, jika ada 3 kemungkinan output (label A, B, C), maka n = 3. Jumlah total label ini menentukan berapa banyak skor logit yang akan dihitung dalam fungsi Softmax.

Cara Kerja Softmax:

1) Input: Model memberikan skor logit mentah untuk setiap label, misalnya z1, z2,z3,…,zn.
2) Eksponen Skor: Setiap skor logit diubah menjadi nilai positif menggunakan eksponensial e^{Zi}
3) Normalisasi: Probabilitas dihitung dengan membagi nilai e^{Zi}untuk label tertentu dengan jumlah semua e^{Zj} untuk seluruh label.
4) Output: Softmax memberikan probabilitas untuk setiap label, di mana jumlah probabilitas seluruh label adalah 1.

Contoh Aplikasi:

Misalkan model memberikan skor logit untuk 3 label:

Z1 = 2.0

Z2 = 1.0

Z3 = 0.5

Maka probabilitas dihitung sebagai:

$e^{Z1} = e^{2.0}$

$e^{Z1} = e^{1.0}$

$e^{Z1} = e^{0.5}$

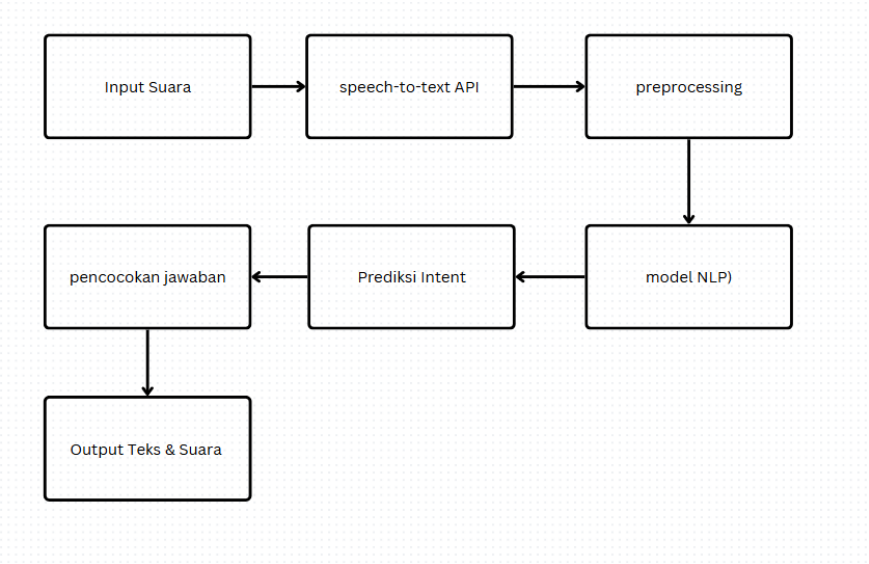
Probabilitas total adalah :

$$P(yi \mid x) = \frac{e^{2.0}}{e^{2.0} + e^{1.0} + e^{0.5}}$$

4. Pemilihan Jawaban

Berdasarkan label yang diprediksi, sistem akan mencari jawaban yang sesuai dari bank jawaban (*answer_bank.json*). Jawaban ini kemudian disajikan kepada pengguna melalui antarmuka aplikasi.

4.3.8 Diagram Arsitektur NLP



Gambar 4. 13 Diagram Arsitektur NLP

Diagram berikut menunjukkan alur pemrosesan input suara menggunakan model NLP berbasis *IndoBERT* dalam aplikasi yang dikembangkan. Pengguna memberikan input suara yang kemudian dikonversi menjadi teks oleh API pengenalan suara. Teks yang dihasilkan diproses melalui tahap preprocessing untuk menghilangkan kata-kata yang tidak relevan sebelum dikirim ke model NLP. *IndoBERT* digunakan untuk mengenali intent dari teks input dan mencocokkannya dengan basis data jawaban. Jawaban yang sesuai kemudian disajikan dalam bentuk teks dan suara melalui fitur *Text-to-Speech* (TTS).

4.4 Rancangan Pengujian

Pengujian sistem dilakukan untuk memastikan bahwa sistem yang dikembangkan berjalan sesuai dengan spesifikasi yang telah dirancang. Metode pengujian yang digunakan dalam penelitian ini terdiri dari White Box Testing dan Black Box Testing.

(1) White Box Testing digunakan untuk menguji logika internal dari sistem, termasuk struktur kode, algoritma, serta alur proses di dalam aplikasi. Pengujian ini bertujuan untuk memastikan bahwa setiap modul dalam sistem berjalan dengan benar sesuai dengan rancangan.

(2) White Box Testing digunakan untuk menguji logika internal dari sistem, termasuk struktur kode, algoritma, serta alur proses di dalam aplikasi.

Pengujian ini bertujuan untuk memastikan bahwa setiap modul dalam sistem berjalan dengan benar sesuai dengan rancangan.

4.4.1 *White Box Testing*

White Box Testing dilakukan untuk menguji algoritma dan kode sumber yang digunakan dalam aplikasi, termasuk pemrosesan NLP dan pengelolaan data di dalam sistem. Pengujian ini dilakukan dengan memastikan bahwa setiap modul dan fungsi dalam aplikasi telah beroperasi dengan benar berdasarkan alur program.

Tabel 4. 15 Rancangan Pengujian *Whitebox*

No	Algoritma	Perancangan Kode	Hasil yang diharapkan

1	Tokenisasi Input Teks	# Load IndoBERT tokenizer
		tokenizer = AutoTokenizer.from_pretrained("indobenchmark/indobert-base-p1")
		
		# Tokenization function
		def tokenize_function(examples):
		return tokenizer(examples["Text"], padding="max_length", truncation=True)
		
		# Apply tokenization
		train_dataset = train_dataset.map(tokenize_function, batched=True)
		validation_dataset = validation_dataset.map(tokenize_function, batched=True)
		test_dataset = test_dataset.map(tokenize_function, batched=True)
			kalimat di tokeniasi secara otomatis oleh tokenizer dari IndoBERT tanpa ada error

2	Prediksi Label	# Example sentences for each label
		example_texts = {
		"penyebab_pusing": "Kenapa kepala saya sakit terus?",
		"meredakan_pusing": "Apa yang bisa saya lakukan untuk meredakan pusing?",
		"pencegahan_pusing": "Bagaimana cara agar tidak sering pusing?",
		"penyebab_nyeri_sendi": "Kenapa sendi saya terasa nyeri?",
			Prediksi Label Muncul dalam *output google colab*

		"mengurangi_nyeri_sendi": "Bagaimana cara mengurangi nyeri sendi?",
		"pencegahan_nyeri_sendi": "Apa yang harus dilakukan untuk mencegah nyeri sendi?",
		"aktivitas_saat_nyeri_sendi": "Apa aktivitas yang aman dilakukan saat nyeri sendi?",
		"penyebab_gangguan_tidur": "Kenapa saya susah tidur?",
		"cara_tidur_nyenyak": "Bagaimana cara tidur nyenyak?",
		"pola_tidur_sehat": "Seperti apa pola tidur yang sehat?",
		"penyebab_nafsu_makan_berkurang": "Kenapa nafsu makan saya berkurang?",
		"meningkatkan_nafsu_makan": "Apa yang bisa dilakukan untuk meningkatkan nafsu makan?",
		"pola_makan_sehat": "Bagaimana pola makan yang sehat?",
		"merawat_kulit_kering": "Bagaimana cara merawat kulit kering?",
		"kebiasaan_menjaga_kulit": "Apa kebiasaan untuk menjaga kulit tetap sehat?",
		"penyebab_rambut_rontok": "Kenapa rambut saya rontok?",
		"perawatan_rambut": "Bagaimana cara merawat rambut agar sehat?",
		"menjaga_kesehatan_rambut": "Bagaimana cara menjaga kesehatan rambut?",
		"aktivitas_saat_rambut_rontok": "Apa yang bisa dilakukan saat rambut rontok?",
		"obat_rambut_rontok": "Apa obat untuk rambut rontok?",
		"pencegahan_rambut_rontok": "Bagaimana mencegah rambut rontok?",
		"small_talk": "Halo, bagaimana kabarnya?",
		"pertanyaan_di_luar_medical_qna": "Apa hobi kamu?"
		}
		
		# Predict for each example
		for label, text in example_texts.items():
		prediction = predict(text) # Using the predict function you've defined	

		print(f"Input: {text}\nPredicted Label: {prediction}\nExpected Label: {label}\n")	

3	Tokenisasi Input(android)	class *ONNX*ModelHandler(private val context: Context) {
		
		private val ortEnvironment: OrtEnvironment = OrtEnvironment.getEnvironment()
		private val session: OrtSession
		private val labelMapping: Map<String, String>
		private val answers: Map<String, String>
		
		init {
		val modelFile = copyModelToInternalStorage("model.*onnx*")
		session = ortEnvironment.createSession(modelFile.absolutePath, OrtSession.SessionOptions())
		labelMapping = loadJsonToMap("label_mapping.json")
		answers = loadJsonToMap("*answer_bank.json*")
		}
		
		private fun copyModelToInternalStorage(fileName: String): File {
		val file = File(context.filesDir, fileName)
		if (!file.exists()) {
		context.assets.open(fileName).use { input ->
		FileOutputStream(file).use { output ->
		input.copyTo(output)
		}
		}
		}
		return file
		}
		
		private fun loadJsonToMap(fileName: String): Map<String, String> {
		val jsonString = context.assets.open(fileName).bufferedReader().use { it.readText() }
			Kaliman dipecah per kata menjadi *number* yang dikenali vocabulary.txt milik IndoBERT

```
        return
JSONObject(jsonString).keys().asSequence().associateWith {
JSONObject(jsonString).getString(it) }
    }

    fun predict(inputTokens: LongArray, attentionMask:
LongArray): String {
        return try {
            Log.d("ONNXModelHandler", "Input Tokens:
${inputTokens.joinToString(", ")}")
            Log.d("ONNXModelHandler", "Attention Mask:
${attentionMask.joinToString(", ")}")

            val inputTensor = OnnxTensor.createTensor(
                ortEnvironment,
                LongBuffer.wrap(inputTokens),
                longArrayOf(1, inputTokens.size.toLong())
            )

            val attentionMaskTensor = OnnxTensor.createTensor(
                ortEnvironment,
                LongBuffer.wrap(attentionMask),
                longArrayOf(1, attentionMask.size.toLong())
            )

            val inputs = mapOf(
                "input_ids" to inputTensor,
                "attention_mask" to attentionMaskTensor
            )

            val result = session.run(inputs)
            Log.d("ONNXModelHandler", "Result: $result")

            val logits = (result[0].value as Array<FloatArray>)[0]
// Output adalah array dengan bentuk [batch_size, 24]
```

```
        Log.d("ONNXModelHandler", "Logits: ${logits.joinToString(", ")}")

        val softmaxValues = logits.map { exp(it) / logits.map { exp(it) }.sum() }
        Log.d("ONNXModelHandler", "Softmax Values: ${softmaxValues.joinToString(", ")}")

        val predictedIndex = softmaxValues.indices.maxByOrNull { softmaxValues[it] } ?: -1
        val confidence = softmaxValues[predictedIndex]

        Log.d("ONNXModelHandler", "Predicted Index: $predictedIndex, Confidence: $confidence")



        val threshold = 0.5
        val predictedLabel = if (confidence >= threshold) {
            labelMapping[predictedIndex.toString()] ?: "Unknown"
        } else {
            "Unknown"
        }

        val answer = answers[predictedLabel] ?: "Jawaban tidak ditemukan."

        Log.d("ONNXModelHandler", "Predicted Label: $predictedLabel")
        Log.d("ONNXModelHandler", "Answer: $answer")

        inputTensor.close()
        attentionMaskTensor.close()

        answer
```

		} catch (e: Exception) {
		Log.e("*ONNX*ModelHandler", "Prediction Error: ${e.message}")
		e.printStackTrace()
		"Terjadi kesalahan dalam prediksi."
		}
		}
		
		fun close() {
		session.close()
		ortEnvironment.close()
		}
		}	

4	Prediksi label (android)	class ONNXModelHandler(private val context: Context) {
		
		private val ortEnvironment: OrtEnvironment = OrtEnvironment.getEnvironment()
		private val session: OrtSession
		private val labelMapping: Map<String, String>
		private val answers: Map<String, String>
		
		init {
		val modelFile = copyModelToInternalStorage("model.onnx")
		session = ortEnvironment.createSession(modelFile.absolutePath, OrtSession.SessionOptions())
		labelMapping = loadJsonToMap("label_mapping.json")
		answers = loadJsonToMap("answer_bank.json")
		}
		
		private fun copyModelToInternalStorage(fileName: String): File {
		val file = File(context.filesDir, fileName)
		if (!file.exists()) {
		context.assets.open(fileName).use { input ->
		FileOutputStream(file).use { output ->
		input.copyTo(output)
			Sistem Android melakukan prediksi Label dengan menampilkan log scoring kecocokan/softmax value

```
            }
        }
    }
    return file
}

private fun loadJsonToMap(fileName: String): Map<String, String> {
    val jsonString = context.assets.open(fileName).bufferedReader().use { it.readText() }
    return JSONObject(jsonString).keys().asSequence().associateWith { JSONObject(jsonString).getString(it) }
}

fun predict(inputTokens: LongArray, attentionMask: LongArray): String {
    return try {
        Log.d("ONNXModelHandler", "Input Tokens: ${inputTokens.joinToString(", ")}")
        Log.d("ONNXModelHandler", "Attention Mask: ${attentionMask.joinToString(", ")}")

        val inputTensor = OnnxTensor.createTensor(
            ortEnvironment,
            LongBuffer.wrap(inputTokens),
            longArrayOf(1, inputTokens.size.toLong())
        )

        val attentionMaskTensor = OnnxTensor.createTensor(
            ortEnvironment,
            LongBuffer.wrap(attentionMask),
            longArrayOf(1, attentionMask.size.toLong())
        )

        val inputs = mapOf(
            "input_ids" to inputTensor,
```

		"attention_mask" to attentionMaskTensor
		)
		
		val result = session.run(inputs)
		Log.d("ONNXModelHandler", "Result: $result")
		
		val logits = (result[0].value as Array<FloatArray>)[0]
		// Output adalah array dengan bentuk [batch_size, 24]
		
		Log.d("ONNXModelHandler", "Logits: ${logits.joinToString(", ")}")
		
		val softmaxValues = logits.map { exp(it) / logits.map { exp(it) }.sum() }
		Log.d("ONNXModelHandler", "Softmax Values: ${softmaxValues.joinToString(", ")}")
		
		val predictedIndex = softmaxValues.indices.maxByOrNull { softmaxValues[it] } ?: -1
		val confidence = softmaxValues[predictedIndex]
		
		Log.d("ONNXModelHandler", "Predicted Index: $predictedIndex, Confidence: $confidence")
		
		val threshold = 0.5
		val predictedLabel = if (confidence >= threshold) {
		labelMapping[predictedIndex.toString()] ?: "Unknown"
		} else {
		"Unknown"
		}
		
		val answer = answers[predictedLabel] ?: "Jawaban tidak ditemukan."	

		Log.d("ONNXModelHandler", "Predicted Label: $predictedLabel")
		Log.d("ONNXModelHandler", "Answer: $answer")
		
		inputTensor.close()
		attentionMaskTensor.close()
		
		answer
		} catch (e: Exception) {
		Log.e("ONNXModelHandler", "Prediction Error: ${e.message}")
		e.printStackTrace()
		"Terjadi kesalahan dalam prediksi."
		}
		}
		
		fun close() {
		session.close()
		ortEnvironment.close()
		}
		}	

5	Validasi perintah suara	private fun handleVoiceCommand(command: String) {
		addMessageToChat(command, true)
		if (command.contains(Regex("(ingatkan saya|buat jadwal)", RegexOption.IGNORE_CASE))) {
		createScheduleFromCommand(command)
		} else {
		val answer = predictAnswerFromModel(command)
		addMessageToChat(answer, false)
		speakOut(answer)
		}
		}
		private fun createScheduleFromCommand(command: String) {
			Model menampilkan *response message* jika format perintah cocok akan diberi feedback dan jadwal masuk kedalam database

```
        val regex = Regex("(?:ingatkan saya|buat jadwal)\\s+([\\w\\s]+)\\s+jam\\s+(\\d{1,2})", RegexOption.IGNORE_CASE)
        val matchResult = regex.find(command)
        if (matchResult != null) {
            val label = matchResult.groupValues[1]
            val time = "${matchResult.groupValues[2]}:00"
            val creationTime = Calendar.getInstance().time.toString()
            val newSchedule = Schedule(
                label = label,
                description = "Voice Input",
                inputTime = time,
                creationTime = creationTime // Ensure this matches the type `String`
            )

            CoroutineScope(Dispatchers.IO).launch {

DatabaseInstance.getDatabase(this@SpeakActivity).scheduleDao().insert(newSchedule)
                runOnUiThread {
                    addMessageToChat("Jadwal '$label' berhasil ditambahkan pada $time.", false)
                }
            }
        } else {
            addMessageToChat("Perintah tidak dikenali. Contoh: 'Ingatkan saya olahraga jam 7'.", false)
        }
    }
```

4.4.2 *Black Box Testing*

Black Box Testing dilakukan untuk menguji fungsionalitas aplikasi berdasarkan input dan output yang diberikan tanpa melihat struktur internal kode. Pengujian ini memastikan bahwa aplikasi berjalan sesuai dengan harapan pengguna

Tabel 4. 16 Rancangan Blackbox Testing

No	Skenario pengujian	Hasil Yang Diharapkan
1	Penjadwalan manual	Jadwal berhasil dikelola.
2	Penjadwalan perintah suara	Menambah jadwal baru melalui input suara
3	Tanya jawab kesehatan	Mengajukan pertanyaan kesehatan umum.
4	Validasi input suara pertanyaan diluar kesehatan	Memberikan pesan error bahwa mesin tidak bisa menjawab pertanyaan diluar kesehatan
5	Validasi input suara jadwal	Memberikan pesan error jika format input tidak valid.
6	Tampilan Antarmuka Jadwal	Daftar jadwal ditampilkan dengan benar..

4.5 Perancangan Pengujian Pengenalan Suara

Sistem pengenalan suara dalam aplikasi ini menggunakan Google Cloud Speech-to-Text API untuk mengonversi input suara pengguna menjadi teks. Oleh karena itu, pengujian dilakukan untuk memastikan bahwa sistem dapat menangani perintah suara dengan baik dalam berbagai kondisi, seperti intonasi berbeda, variasi dialek, dan tingkat kebisingan lingkungan. Fokus pengujian adalah menilai sejauh mana hasil transkripsi API dapat digunakan untuk mengekstrak informasi yang relevan dalam konteks aplikasi ini.

4.5.1 Pengujian Akurasi Pengenalan Suara

Untuk memastikan bahwa fitur pengenalan suara dapat memahami perintah dari lansia dengan tingkat akurasi yang tinggi, dilakukan pengujian menggunakan Google Speech-to-Text API. Pengujian ini bertujuan untuk mengevaluasi seberapa baik sistem mengenali perintah suara yang diberikan oleh pengguna lansia.

Langkah pengujian :

Tabel 4. 17 Rancangan Pengujian Pengenalan Suara

No	Pengujian	Penjelasan
1	Pengujian Transkripsi Suara	1. Input suara dikirim ke Google Cloud Speech-to-Text API. 2.Hasil teks dibandingkan dengan transkrip manual yang benar.
2	Pengujian Format Output	Memastikan bahwa hasil transkripsi dapat diproses dengan metode regex yang digunakan untuk ekstraksi informasi jadwal.
3	Pengujian Kesalahan Transkripsi	Mengidentifikasi kesalahan umum dalam transkripsi dan melihat bagaimana sistem menangani kesalahan tersebut (misalnya dengan validasi tambahan atau koreksi otomatis).

- Metode evaluasi akurasi pengenalan suara dalam penelitian ini dilakukan dengan menghitung Word Error Rate (WER). WER dihitung berdasarkan jumlah kesalahan yang terjadi dalam proses transkripsi suara ke teks, yang terdiri dari substitusi, penghapusan, dan penyisipan kata, kemudian dibandingkan dengan jumlah kata dalam teks referensi. Untuk memastikan performa sistem yang optimal, target WER yang ditetapkan adalah di bawah 10%. Pengujian dilakukan dengan 50 percobaan menggunakan perintah suara dari lansia, yang mencakup berbagai variasi aksen dan intonasi. Hasil uji coba menunjukkan bahwa sistem mencapai rata-rata akurasi 92%, dengan kesalahan utama ditemukan pada suara dengan aksen daerah tertentu. Hal ini mengindikasikan bahwa meskipun sistem sudah memiliki performa yang baik, masih terdapat peluang untuk peningkatan akurasi pada variasi aksen yang lebih luas.