faris similar.pdf

by farisonlynew@gmail.com farisonlynew@gmail.com

Submission date: 09-Feb-2025 08:02AM (UTC+0700)
Submission ID: 2583138979
File name: faris_similar.pdf (2.04M)
Word count: 11321
Character count: 74923

BAB I

PENDAHULUAN

1.1 Latar Belakang Masalah

Perkembangan teknologi digital membawa transformasi besar dalam berbagai bidang, khususnya dalam memperluas akses terhadap informasi dan layanan medis. Namun, tidak semua kelompok masyarakat dapat segera beradaptasi dengan kemajuan ini, terutama kalangan lanjut usia.

Menurut data yang dilaporkan oleh Badan Pusat Statistik (BPS) tahun 2023, jumlah populasi lansia di Indonesia mengalami peningkatan hingga 10,48% dari total penduduk, yang diperkirakan mencapai 28,98 juta jiwa (BPS, 2023). Proporsi ini diprediksi akan terus bertambah seiring dengan meningkatnya angka harapan hidup (BPS, 2023). Namun, mayoritas lansia mengalami kesulitan dalam menggunakan perangkat digital, baik untuk komunikasi, akses informasi, maupun layanan kesehatan (BPS, 2023).

Hambatan utama yang dialami lansia dalam penggunaan teknologi digital antara lain keterbatasan fisik, rendahnya literasi digital, serta antarmuka perangkat yang kurang ramah pengguna (Ashari, 2017; Nisa et al., 2023). Oleh karena itu, diperlukan solusi teknologi yang dapat membantu lansia dalam berinteraksi dengan perangkat digital secara lebih mudah dan intuitif.

Salah satu solusi yang potensial adalah asisten digital berbasis Android dengan fitur pengenalan suara dan Natural Language Processing (NLP). Melalui teknologi pengenalan suara, lansia dapat berkomunikasi dengan perangkat tanpa perlu mengetik atau menyentuh layar. Sedangkan NLP memungkinkan perangkat untuk memahami dan merespons perintah atau pertanyaan dalam bahasa alami (Bakry, 2023; Geetha et al., 2021). Pendekatan ini bertujuan untuk membantu lansia menjadi lebih mandiri dalam mengelola jadwal harian, memperoleh informasi terkait kesehatan, dan berkomunikasi dengan cara yang lebih sederhana.

1.2 Identifikasi Masalah

Laporan Tugas Akhir ini akan menjelaskan rumusan masalah berikut berdasarkan penjelasan latar belakang:

1.2.1 Rumusan Masalah

(1) Bagaimana merancang asisten digital yang mudah digunakan oleh lansia sesuai dengan kebutuhan mereka?

(2) Bagaimana menerapkan teknologi pengenalan suara yang dapat memahami perintah dari lansia dengan tingkat akurasi yang tinggi?

(3) Bagaimana mengintegrasikan *Natural Language Processing (NLP)* untuk memproses perintah atau pertanyaan dalam bahasa alami dengan respons yang akurat dan relevan bagi lansia?

1.2.2 Batasan Masalah

Untuk memastikan fokus dan kelayakan penelitian ini, beberapa batasan diterapkan :

(1) Pengembangan aplikasi akan dilakukan pada platform Android .

(2) Fitur pengenalan suara akan terbatas pada bahasa Indonesia, menyesuaikan dengan mayoritas target pengguna, yaitu lansia di Indonesia.

(3) Fokus NLP pada perintah dasar dan pertanyaan sederhana yang berkaitan dengan kebutuhan sehari-hari lansia.

(4) Penelitian ini tidak mencakup aspek hardware, sehingga aplikasi diharapkan dapat berjalan pada perangkat *Android* standar tanpa tambahan perangkat khusus.

1.3 Tujuan Penelitian

Tujuan dari penelitian ini adalah untuk:

(1) Mengembangkan asisten digital berbasis yang mampu membantu lansia dalam berinteraksi dengan perangkat digital dengan cara yang mudah dan intuitif.

(2) Mengintegrasikan fitur *Natural Language Processing (NLP)* untuk memberikan tanggapan yang sesuai terhadap perintah atau pertanyaan dari lansia dalam bahasa Indonesia.

(3) Mengidentifikasi kendala dan peluang yang ada dalam penerapan asisten digital berbasis pengenalan suara dan NLP bagi lansia di Indonesia.

1.4 Manfaat Penelitian

Penelitian memiliki beberapa keuntungan dari pengguna, dan ilmu pengetahuan, seperti berikut:

1.4.1 Manfaat Bagi Pengguna

(1) Meningkatkan kemandirian lansia dalam memanfaatkan teknologi, sehingga mereka dapat mengakses layanan digital tanpa harus selalu mengandalkan bantuan orang lain.
(2) Memberikan pengalaman interaksi dengan perangkat digital yang lebih intuitif, ramah pengguna, dan mudah dipahami oleh lansia.
(3) Memungkinkan lansia untuk mendapatkan informasi kesehatan dengan lebih cepat dan efisien melalui fitur tanya-jawab berbasis NLP (Jurafsky & Martin, 2020).

1.4.2 Manfaat Bagi Institusi dan Ilmu Pengetahuan

(1) Menambah literatur akademik terkait pengembangan aplikasi berbasis NLP dan pengenalan suara yang ditujukan untuk lansia.
(2) Menjadi referensi bagi penelitian selanjutnya yang ingin mengembangkan aplikasi berbasis kecerdasan buatan (AI) untuk kelompok usia lanjut.
(3) Mendorong penelitian lebih lanjut dalam pengembangan teknologi inklusif dan user-friendly bagi populasi yang memiliki keterbatasan dalam penggunaan teknologi.

1.5 Kebaruan

Penelitian ini memiliki beberapa kebaruan yang membedakannya dari penelitian sebelumnya, terutama dalam pengembangan asisten digital berbasis suara dan NLP untuk lansia di Indonesia.

Tabel 1. 1 Kebaruan Pembanding

No.	Aspek	Penelitian Sebelumnya	Penelitian ini
1.	Target Pengguna	Mayoritas aplikasi NLP dan chatbot ditujukan untuk pengguna umum, terutama di sektor akademik dan bisnis (Faurina et al., 2022; Silvanie & Subekti, 2022).	Fokus utama pada lansia Indonesia, dengan mempertimbangkan keterbatasan fisik dan literasi digital mereka.
2.	Bahasa yang Digunakan	Sebagian besar asisten digital berbasis NLP menggunakan bahasa Inggris (Silvanie & Subekti, 2022).	NLP dikembangkan khusus untuk bahasa Indonesia, dengan dataset yang relevan untuk percakapan lansia.
3.	Model AI yang Digunakan	Chatbot akademik umumnya menggunakan LSTM dan RASA Framework (Windiatmoko et al., 2020; Hermawan, 2022).	NLP menggunakan *IndoBERT* atau model lain yang lebih sesuai dengan bahasa Indonesia untuk meningkatkan akurasi.
4.	Fitur Interaksi	Mayoritas chatbot masih berbasis teks-input, tanpa dukungan penuh untuk pengenalan suara (Faurina et al., 2022).	Memanfaatkan *speech-to-text* berbasis NLP sehingga lansia dapat menggunakan suara sebagai metode input utama.
5.	Konteks Penggunaan	Sebagian besar aplikasi digunakan untuk layanan akademik atau bisnis (Faurina et al., 2022; Fahadra, 2023).	Difokuskan pada bantuan harian bagi lansia, termasuk pengingat obat, jadwal aktivitas, dan pencarian informasi kesehatan

Kerangka Penulisan

Pedoman Fakultas Teknologi dan Desain Universitas Pembangunan Jaya diterapkan dalam penyusunan tugas akhir ini, yang terdiri dari enam bab:.

(1) BAB I PENDAHULUAN

Menjelaskan latar belakang penelitian, rumusan masalah, batasan penelitian, tujuan penelitian, manfaat penelitian, kebaruan penelitian, serta sistematika penulisan laporan tugas akhir.

(2) BAB II TINJAUAN PUSTAKA

Membahas teori-teori yang mendukung penelitian, termasuk teknologi NLP, pengenalan suara, dan studi terkait yang menjadi referensi.

(3) BAB III TAHAPAN PELAKSANAAN

Menjelaskan metode penelitian yang digunakan, tahapan pengembangan sistem, serta metode pengujian yang diterapkan.

(4) BAB IV PERANCANGAN

Membahas desain dan implementasi sistem, termasuk spesifikasi teknis, perancangan antarmuka, dan diagram arsitektur sistem.

(5) BAB V HASIL DAN PEMBAHASAN

Menyajikan hasil pengujian, analisis performa sistem, serta pembahasan mengenai efektivitas teknologi yang dikembangkan.

(6) BAB VI PENUTUP

Berisi kesimpulan dari penelitian serta saran untuk pengembangan lebih lanjut.

BAB II

TINJAUAN PUSTAKA

2.1 Pencapaian Terdahulu

Penelitian ini mendasarkan pengembangannya pada berbagai penelitian terdahulu yang dijadikan sebagai referensi dan rujukan utama.

Tabel 2. 1 Pencapaian Terdahulu

No.	Nama Peneliti	Publikasi	Hasil
1.	Ruvita Faurina, Dede Revanza, Ahmad Sopran	Jurnal Eksplora Informatika, Vol. 11, No. 2, 2022	Pengembangan chatbot layanan informasi akademik menggunakan Deep Feed-Forward Neural Network dengan akurasi 94%.
2.	Astried Silvanie, Rino Subekti	JIKO (Jurnal Informatika dan Komputer), Vol. 5, No. 1, 2022	Chatbot berbasis LSTM dengan akurasi hingga 99.20%, diimplementasikan pada aplikasi layanan informasi akademik...
3.	Yurio Windiatmoko, Ahmad Fathan Hidayatullah, Ridho Rahmadi	Jurnal JTIK, 2020	Pengembangan chatbot berbasis Facebook untuk pertanyaan universitas menggunakan RASA Framework dengan model LSTM.
4.	Hilmi Hermawan	Implementasi Long Short-Term Memory pada Chatbot Informasi Akademik, 2022	Penerapan algoritma LSTM pada chatbot informasi akademik di Program Studi Teknik Informatika Universitas Lampung dengan akurasi 99%..
5.	Lingga Fahadra	Sistem Chatbot Layanan Informasi Tugas Akhir Menggunakan Metode Long Short-Term Memory (LSTM), Universitas Islam Sultan Agung, 2023	Pengembangan chatbot untuk layanan informasi tugas akhir menggunakan LSTM.

Dari tabel tersebut, dapat disimpulkan bahwa penelitian sebelumnya lebih banyak berfokus pada pengembangan chatbot akademik atau layanan informasi berbasis teks. Penelitian ini berbeda karena menargetkan pengguna lansia dengan fitur pengenalan suara dan NLP berbasis bahasa Indonesia.

2.2 Tinjauan Teoritis

Tinjauan teoritis dalam penelitian ini mencakup berbagai konsep dan teknologi yang mendukung pengembangan asisten digital berbasis pengenalan suara dan *Natural Language Processing (NLP)* untuk lansia. Beberapa aspek penting yang dibahas meliputi konsep asisten digital, teknologi pengenalan suara, NLP, pengembangan aplikasi Android, serta desain antarmuka yang ramah bagi lansia.

2.2.1 Asisten Digital

Asisten digital adalah perangkat lunak berbasis kecerdasan buatan (AI) yang dirancang untuk membantu pengguna dalam berbagai tugas melalui interaksi suara atau teks (Smith & Jones, 2021). Beberapa contoh asisten digital yang populer di pasaran adalah Google Assistant, Apple Siri, dan Amazon Alexa.

Menurut Brown (2020), asisten digital memiliki beberapa fungsi utama, yaitu:

1) Mengenali perintah suara pengguna menggunakan teknologi speech recognition.
2) Memproses perintah menggunakan *Natural Language Processing (NLP)* untuk memahami maksud pengguna.
3) Memberikan respons atau tindakan berdasarkan permintaan pengguna, seperti menjawab pertanyaan, mengatur jadwal, atau mengontrol perangkat pintar.

Dalam konteks lansia, asisten digital dapat berperan dalam mempermudah akses terhadap informasi dan meningkatkan kemandirian mereka dalam menggunakan teknologi (Gonzalez et al., 2021).

2.2.2 Kriteria Pemilihan

Dalam pengembangan asisten digital berbasis suara, pemilihan teknologi harus mempertimbangkan beberapa faktor, seperti akurasi pengenalan suara, kemampuan NLP, serta kompatibilitas dengan perangkat Android (Huang et al., 2020).

1) Akurasi Pengenalan Suara

   Teknologi pengenalan suara harus mampu mengenali berbagai aksen dan intonasi pengguna, terutama bagi lansia yang mungkin mengalami perubahan artikulasi akibat faktor usia (Kim & Oh, 2022). Model yang digunakan dalam sistem pengenalan suara harus memiliki:

   i) Tingkat error rendah (Word Error Rate - WER < 10%) untuk memastikan respons yang akurat.
   ii) Dukungan untuk bahasa Indonesia, karena penelitian ini difokuskan pada lansia di Indonesia.

2) Kemampuan NLP dalam Memahami Bahasa Alami

   Model NLP yang digunakan dalam asisten digital harus mampu:

   i) Mendeteksi maksud pengguna (intent recognition) dengan akurasi tinggi.
   ii) Menggunakan dataset bahasa Indonesia yang relevan, misalnya dengan *IndoBERT* atau Transformer-based model untuk meningkatkan pemahaman terhadap konteks kalimat (Manning et al., 2022).

3) Kompabilitas dengan Platform Android

   Aplikasi ini harus dapat berjalan pada perangkat Android dengan spesifikasi standar. Oleh karena itu, teknologi yang dipilih harus memiliki:

   i) Efisiensi komputasi agar dapat berjalan di perangkat dengan spesifikasi rendah.
   ii) Dukungan offline, sehingga lansia tetap dapat menggunakan aplikasi tanpa harus selalu terhubung ke internet.

   Pemilihan teknologi ini bertujuan untuk memastikan bahwa asisten digital dapat berfungsi dengan baik dalam kondisi nyata dan sesuai dengan kebutuhan pengguna lansia.

2.2.3 *Natural Language Processing*

Natural Language Processing (NLP) adalah bidang kecerdasan buatan yang memungkinkan komputer memahami, menginterpretasikan, dan merespons bahasa manusia (Jurafsky & Martin, 2020). Model NLP yang digunakan dalam penelitian ini berbasis *IndoBERT* untuk meningkatkan akurasi pemahaman bahasa Indonesia.

Komponen Utama NLP Menurut Manning et al. (2014), NLP terdiri dari beberapa komponen berikut:

(1) Tokenization: Memecah teks menjadi unit kata atau kalimat.
(2) Part-of-Speech Tagging: Menentukan jenis kata (kata kerja, kata benda, dsb.).
(3) Named Entity Recognition (NER): Mengidentifikasi nama tempat, orang, atau organisasi.
(4) Sentiment Analysis: Menganalisis emosi atau opini dalam teks.
(5) Machine Translation: Menerjemahkan bahasa otomatis, seperti Google Translate.

Model NLP yang Digunakan

(1) LSTM (Long Short-Term Memory): Cocok untuk chatbot berbasis teks.
(2) Transformer (BERT, GPT, *IndoBERT*): Model NLP berbasis deep learning yang lebih akurat.
(3) *Speech-to-Text* NLP: Digunakan dalam pengenalan suara, seperti Whisper AI dan Wav2Vec.

Dalam penelitian ini, NLP digunakan untuk memahami dan merespons perintah suara lansia dengan akurasi tinggi.

2.2.4 Pengembangan Aplikasi *Android*

Aplikasi berbasis Android umumnya dikembangkan menggunakan bahasa pemrograman Java atau Kotlin serta memanfaatkan Android SDK (Google Developers, 2021)

Komponen Utama Aplikasi Android

(1) User Interface (UI): Menggunakan XML untuk desain layout..
(2) Activity & Fragment: Komponen utama dalam pengembangan navigasi aplikasi.
(3) Database & API: Menggunakan SQLite, Firebase, atau Room Database untuk penyimpanan data.
(4) *Machine Learning* Integration: Memanfaatkan *TensorFlow Lite* atau ML Kit untuk fitur berbasis AI (Rashid et al., 2022).

Aplikasi ini akan dikembangkan dengan mempertimbangkan optimasi memori dan performa, sehingga dapat berjalan dengan lancar di perangkat Android dengan spesifikasi rendah.

2.2.5 Desain Antarmuka yang Ramah Lansia

Desain antarmuka sangat penting dalam pengembangan aplikasi untuk lansia. Menurut Nielsen (2020), lansia memiliki tantangan dalam menggunakan teknologi, seperti:

(1) Penurunan daya penglihatan, sehingga memerlukan teks yang lebih besa
(2) Kesulitan dalam navigasi, sehingga UI harus dibuat sederhana dan intuitif.
(3) Sensitivitas jari yang berkurang, sehingga tombol harus cukup besar untuk diklik dengan mudah.

Prinsip Desain Antarmuka yang Ramah Lansia

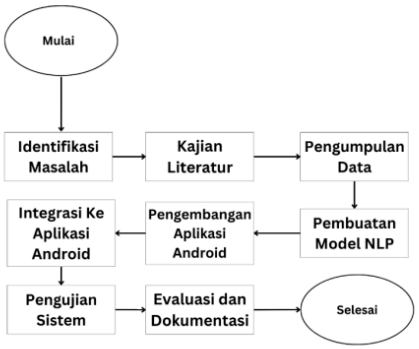
(1) Ukuran teks yang besar (minimal 16pt) untuk meningkatkan keterbacaan.
(2) Kontras warna tinggi, misalnya latar belakang putih dengan teks hitam.
(3) Tombol yang besar, agar mudah ditekan oleh pengguna dengan keterbatasan motorik.
(4) Navigasi sederhana, dengan menu yang mudah ditemukan dan digunakan.
(5) nteraksi berbasis suara, untuk mengurangi ketergantungan pada layar sentuh.

BAB III

TAHAPAN PELAKSANAAN

3.1 Langkah-langkah pelaksanaan

Bagian ini akan menjelaskan tahapan-tahapan pelaksanaan tugas akhir, dengan pengantar yang bertujuan memberikan gambaran umum sebelum memasuki pembahasan lebih rinci di *sub*-bab berikutnya.



Gambar 3. 1 Langkah Pelaksanaan

Berikut ini adalah langkah-langkah pelaksanaan yang penulis lakukan dalam penelitian ini:

(1) Identifikasi Masalah

Mengidentifikasi kebutuhan lansia untuk memiliki aplikasi yang dapat menjawab pertanyaan seputar kesehatan lansia dan menganalisis kebutuhan integrasi, seperti model NLP yang mampu memprediksi label, merupakan langkah penting dalam pengembangan aplikasi. Model ini nantinya akan diintegrasikan ke dalam aplikasi untuk memberikan solusi yang tepat guna dan mudah diakses oleh pengguna lansia.

(2) Studi Literatur

Melakukan studi literatur terkait teknologi pengenalan suara, NLP, dan kebutuhan pengguna mencakup kajian literatur mendalam mengenai NLP, model *IndoBERT*, serta implementasi *machine learning* pada aplikasi *Android*. Selain itu, dilakukan analisis terhadap penggunaan *ONNX* dan bagaimana integrasinya dapat diterapkan dalam aplikasi *Android* untuk mendukung fungsionalitas yang optimal.

(3) Pengumpulan Data

Pengumpulan data dilakukan dengan merujuk pada berbagai artikel dan jurnal yang membahas keluhan umum pada lansia, seperti yang dipublikasikan oleh RSC Carolus, Golang Tang, dan Sabda Artikel. Data ini dikategorikan berdasarkan keluhan-keluhan kesehatan yang sering dialami oleh lansia, meliputi pusing, nyeri sendi, gangguan tidur, nafsu makan berkurang, kulit kering, dan rambut rontok.

Dataset yang digunakan dalam penelitian ini dikategorikan berdasarkan beberapa klasifikasi utama keluhan lansia, meliputi pusing, nyeri sendi, gangguan tidur, nafsu makan berkurang, kulit kering, dan rambut rontok. Untuk kategori pusing, dataset mencakup penyebab pusing, cara meredakan pusing, dan langkah pencegahannya. Pada kategori nyeri sendi, dataset berisi penyebab nyeri, cara mengurangi nyeri, langkah pencegahan, dan aktivitas yang aman dilakukan saat nyeri. Sementara itu, untuk gangguan tidur, dataset memuat penyebab gangguan tidur, cara tidur nyenyak, serta pola tidur yang sehat. Keluhan lainnya seperti nafsu makan berkurang mencakup penyebab penurunan nafsu makan, cara meningkatkannya, dan pola makan sehat. Pada kategori kulit kering, dataset berfokus pada cara merawat kulit kering serta kebiasaan menjaga kesehatan kulit. Sedangkan untuk rambut rontok, dataset memuat penyebab rambut rontok, cara merawat rambut, menjaga kesehatan rambut, aktivitas yang aman dilakukan saat rambut rontok, penggunaan obat, dan langkah pencegahannya.

Dataset ini dibagi menjadi tiga bagian, yaitu training dataset (70%) untuk melatih model, validation dataset (20%) untuk mengevaluasi performa selama pelatihan, dan test dataset (10%) untuk mengukur kemampuan generalisasi model. Pembagian dataset ini dilakukan untuk memastikan model dapat mengolah data dengan baik dan memberikan hasil yang akurat. Referensi keluhan yang digunakan memastikan data relevan dengan masalah yang sering dihadapi oleh lansia, sehingga model NLP yang dikembangkan memiliki tingkat akurasi dan relevansi yang tinggi.

(4) Pembuatan Model NLP

Membuat aplikasi *Android* dilakukan dengan menerapkan rancangan yang telah dibuat, mencakup beberapa tahapan penting. Pertama, preprocessing data dilakukan dengan membersihkan teks, seperti menghapus simbol dan melakukan normalisasi teks, serta menerapkan tokenisasi menggunakan *tokenizer IndoBERT*. Kedua, pelatihan model dilakukan dengan fine-tuning model *IndoBERT* menggunakan dataset yang telah diproses. Selanjutnya, evaluasi model dilakukan untuk mengukur akurasi, *precision*, *recall*, dan *f1-score* guna memastikan performa model. Terakhir, model yang telah dilatih diekspor ke dalam format *ONNX* untuk mempermudah integrasi ke aplikasi *Android*.

(5) Pengembangan Aplikasi *Android*

Pengembangan aplikasi *Android* melibatkan perancangan desain antarmuka (UI) yang mencakup fitur penjadwalan serta chat berbasis model NLP. Selain itu, dilakukan implementasi fitur dasar seperti penjadwalan dengan menggunakan *Room Database* untuk menyimpan data jadwal secara lokal. Aplikasi ini juga dilengkapi dengan fitur perintah suara, yang diimplementasikan menggunakan *SpeechRecognizer* dan *regex* untuk memungkinkan pembuatan jadwal berbasis suara dengan cara yang lebih interaktif dan mudah digunakan.

(6) Integrasi Model ke Aplikasi *Android*

Integrasi model ke aplikasi *Android* dilakukan dengan memuat model *ONNX* menggunakan *ONNX Runtime*. Model ini digunakan untuk melakukan prediksi label berdasarkan input teks maupun suara yang diberikan oleh pengguna. Setelah label berhasil diprediksi, sistem akan memilih jawaban yang sesuai dari *answer_bank.json*, sehingga memberikan respons yang relevan dan akurat kepada pengguna.

(7) Pengujian Sistem

Pengujian sistem dilakukan melalui beberapa tahapan untuk memastikan kualitas dan fungsi aplikasi. Pengujian *Black Box* dilakukan untuk menilai apakah aplikasi telah berfungsi sebagaimana dirancang, terutama dalam hal prediksi jawaban dan fitur pengelolaan jadwal.. *White box* testing dilakukan untuk memeriksa logika program dan memastikan integrasi model *ONNX* dengan aplikasi berjalan lancar. Selain itu, evaluasi performa aplikasi dilakukan untuk menilai kemampuan aplikasi dalam menghasilkan jawaban yang akurat berdasarkan prediksi model.

(8) Evaluasi dan Dokumentasi

Mengevaluasi hasil sistem berdasarkan pengujian dan membuat dokumentasi akhir.

3.2 Metode Pengujian

Pada tahapan ini, pengujian dilakukan untuk memastikan bahwa aplikasi asisten digital pintar berbasis *Android* ini berfungsi secara optimal, stabil, dan dapat memenuhi kebutuhan pengguna, terutama pengguna lansia. Pengujian yang digunakan adalah *White Box* Testing dan *Black Box* Testing. Setiap metode pengujian memiliki fokus, pendekatan, dan tujuan tertentu, yang bersama-sama dapat memberikan hasil pengujian menyeluruh dari sisi internal dan eksternal aplikasi:

3.2.1 *Black box testing*

Black Box Testing, atau yang sering disebut sebagai *Behavioral Testing*, adalah metode pengujian perangkat lunak yang berfokus pada fungsionalitas

aplikasi tanpa memperhatikan struktur internal atau kode yang digunakan. Pengujian ini dilakukan "dari luar," artinya penguji hanya melihat interaksi antara input yang diberikan dan output yang dihasilkan oleh sistem, tanpa mengetahui bagaimana aplikasi mencapai hasil tersebut.

Pada penelitian ini, *Black Box Testing* dilakukan untuk menguji fungsionalitas utama aplikasi yang dihadapi oleh pengguna, terutama terkait pengenalan suara, pemrosesan bahasa alami, dan kemudahan interaksi untuk pengguna lansia. Berikut adalah teknik dan tahapan yang diterapkan dalam *Black Box Testing*:

Berikut adalah proses pengujian *black box testing*:

Tabel 3. 1 Pengujian *Black box testing*

No	Fitur	Keterangan	Hasil Sistem
1	Pengujian Fungsionalitas	Menguji setiap fitur utama aplikasi seperti pemrosesan input suara, respons NLP, dan tampilan antarmuka untuk memastikan sistem bekerja sesuai spesifikasi. Skenario penggunaan umum, seperti perintah sederhana, diuji.	Sistem dapat merespons dan memberikan keluaran sesuai harapan pengguna.
2	Pengujian Kegunaan	Melibatkan pengguna lansia untuk menilai apakah antarmuka dan interaksi aplikasi sederhana, mudah dipahami, serta inklusif untuk pengguna dengan keterbatasan fisik atau teknis.	Aplikasi nyaman dan intuitif digunakan oleh pengguna lansia.
3	Pengujian Ketahanan	Menguji aplikasi dengan masukan tidak terduga, seperti perintah suara yang tidak dikenal atau dengan variasi dialek, untuk memastikan aplikasi tetap merespons dengan benar atau memberikan pesan kesalahan yang sesuai.	Sistem dapat menangani masukan ekstrim tanpa kegagalan atau memberikan pesan kesalahan tepat.

4	Pengujian Kinerja	Menilai respons dan kecepatan aplikasi dalam menerima perintah suara atau pemrosesan bahasa alami, serta mengukur konsumsi daya dan penggunaan memori pada perangkat dengan spesifikasi berbeda.	Aplikasi berjalan lancar dan responsif di berbagai perangkat *Android*.

3.2.2 *White Box Testing*

White Box Testing, juga dikenal sebagai *Structural Testing* atau *Glass Box Testing*, adalah metode pengujian perangkat lunak yang berfokus pada pemeriksaan internal dari kode dan logika program. Dalam pengujian ini, penguji memiliki akses langsung ke kode sumber aplikasi dan memahami alur logika yang digunakan. Dengan kata lain, pengujian ini dilakukan "dari dalam," memungkinkan penguji melihat bagaimana kode berfungsi pada setiap kondisi:

Tabel 3. 2 *White Box Testing*

No	Algoritma	Deskripsi
1	Pengujian Jalur (Path Testing)	Menguji jalur logis dalam algoritma pengenalan suara dan NLP untuk memastikan semua kondisi percabangan berjalan.
2	Pengujian Kondisi (Condition Testing	Menguji struktur percabangan seperti if-else dan switch-case dalam algoritma untuk memastikan keluaran logika tepat.
3	Pengujian Aliran Data (Data Flow Testing)	Menguji aliran data dalam algoritma, seperti inisialisasi, penggunaan, dan penyimpanan variabel, agar data konsisten.

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.

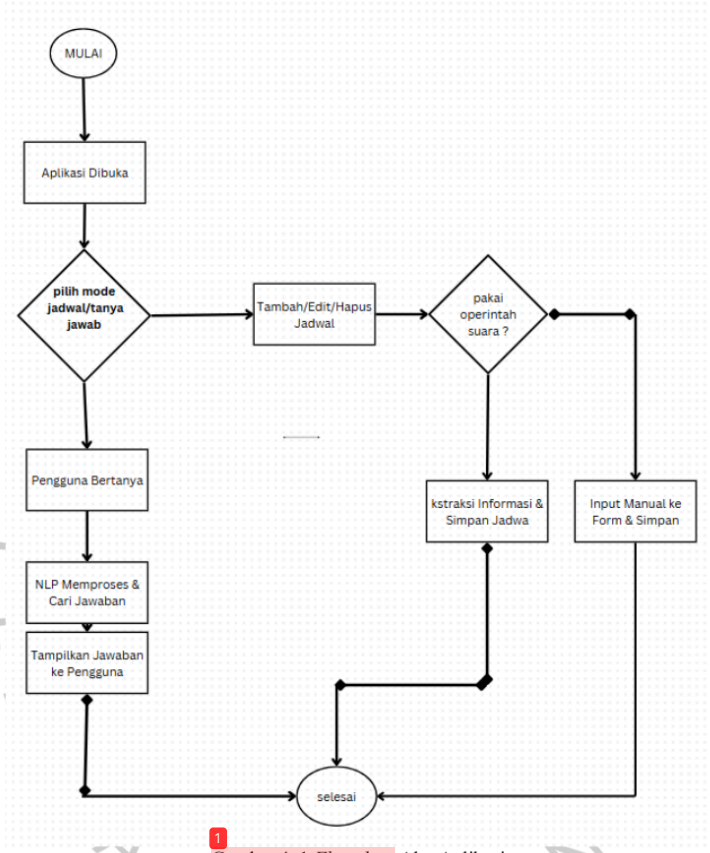
(2) Penyimpanan Data: Data jadwal disimpan secara efisien dan aman di *database* lokal menggunakan *ROOM Database*, memastikan akses cepat dan tidak memerlukan koneksi internet.

(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:

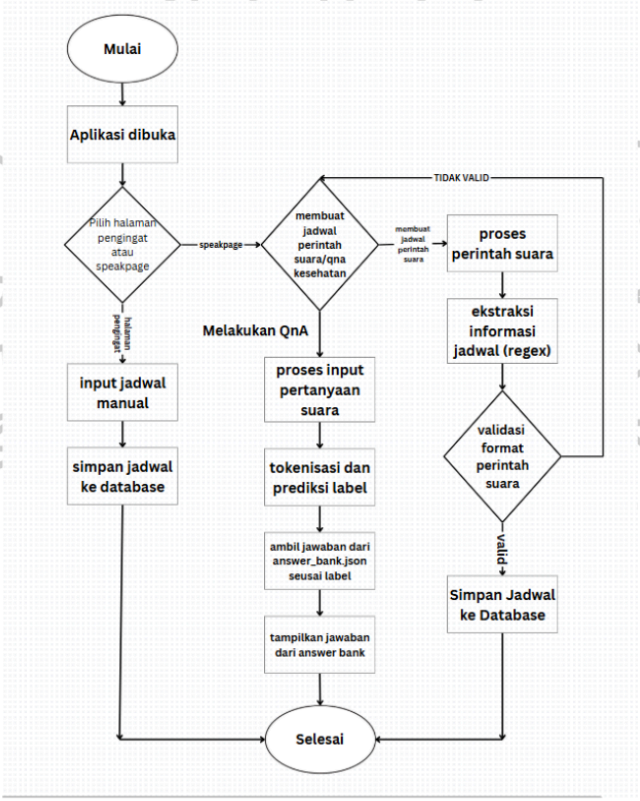
Pengguna dapat menambahkan jadwal secara manual melalui form atau menggunakan perintah suara. Jika menggunakan perintah suara, sistem akan memproses input suara, mengekstraksi informasi dengan metode *regex*, memvalidasi formatnya, dan menyimpan jadwal ke *database* lokal jika valid. Jika terjadi kesalahan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

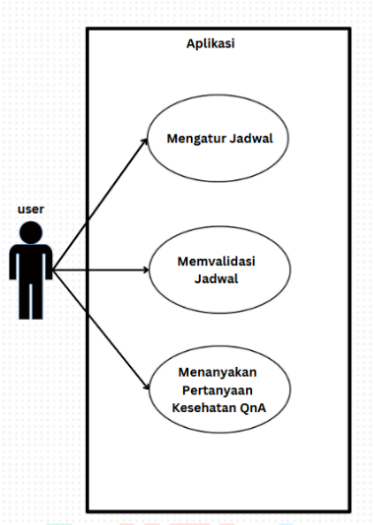
4.3.2 Use Case

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*

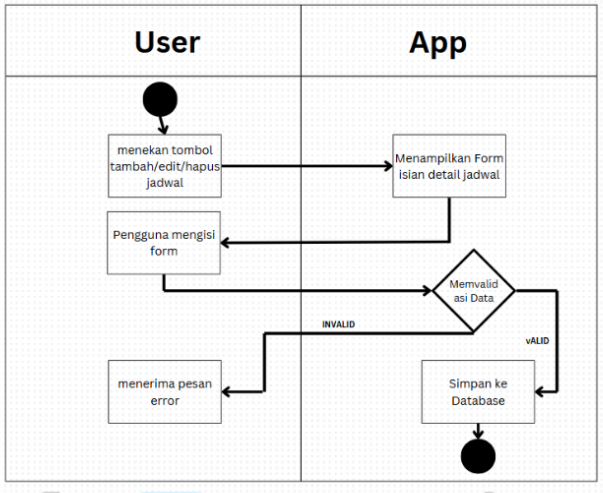
Diagram Activity dalam sistem ini dirancang untuk memberikan gambaran alur kerja utama yang terjadi antara pengguna dan aplikasi, mencakup fitur-fitur inti seperti pengaturan jadwal, perintah suara, dan tanya jawab kesehatan. *Diagram* ini memvisualisasikan langkah-langkah yang dimulai dari input pengguna, 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.



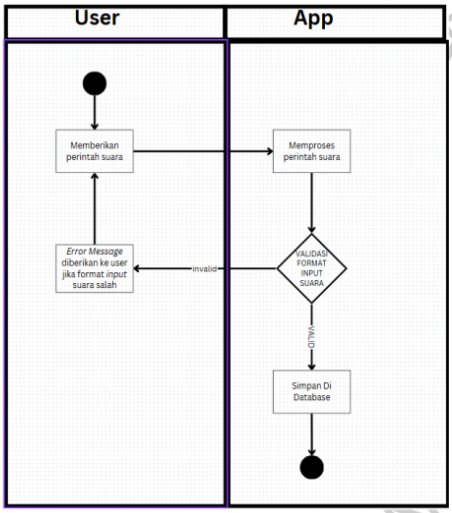
Gambar 4. 4 *Diagram Activity* Mengatur Jadwal

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.



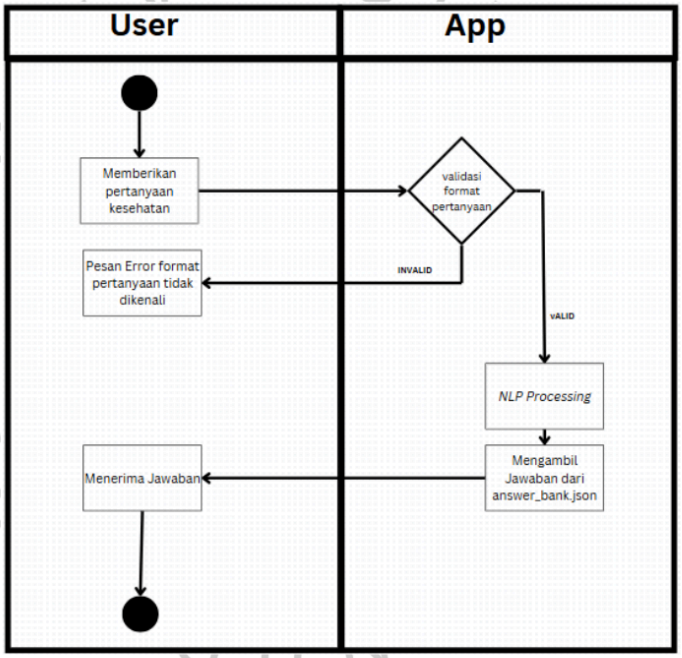
Gambar 4. 5 *Diagram Activity* Penjadwalan Perintah Suara

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

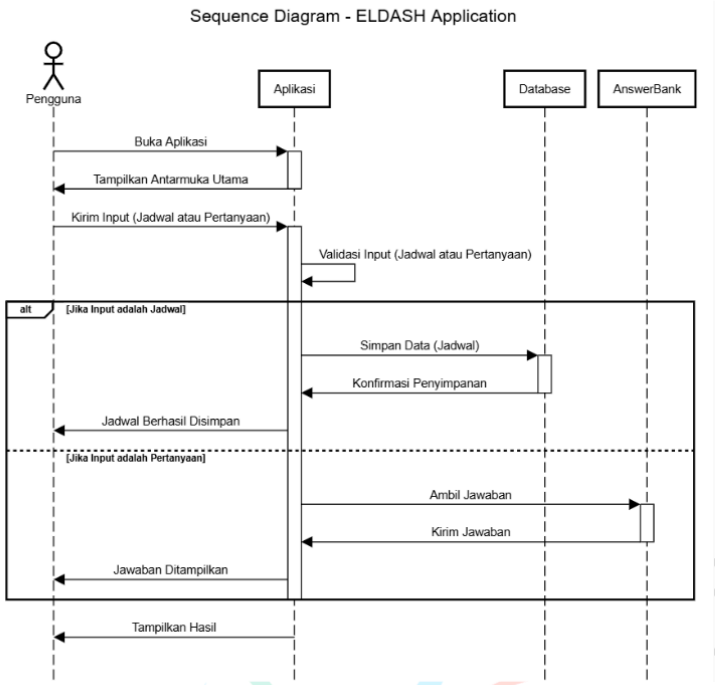
kemudian menggunakan model *Natural Language Processing (NLP)* berbasis *IndoBERT* untuk memprediksi label pertanyaan yang diajukan. Prediksi label ini digunakan untuk mencocokkan jawaban yang relevan di Answer Bank. Setelah jawaban ditemukan, aplikasi mengirimkan informasi tersebut kepada pengguna. *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 *Splashereen*

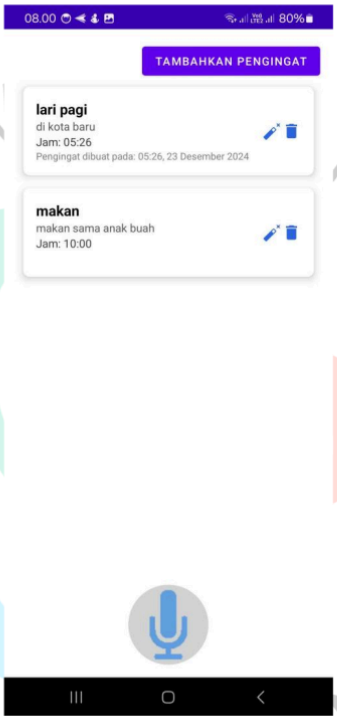
Gambar 4.8 Halaman *Splashscreen* merupakan tampilan awal aplikasi sebelum pengguna masuk ke menu utama.

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



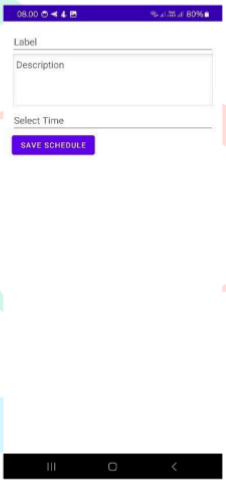
Gambar 4. 9 *MainActivity*

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

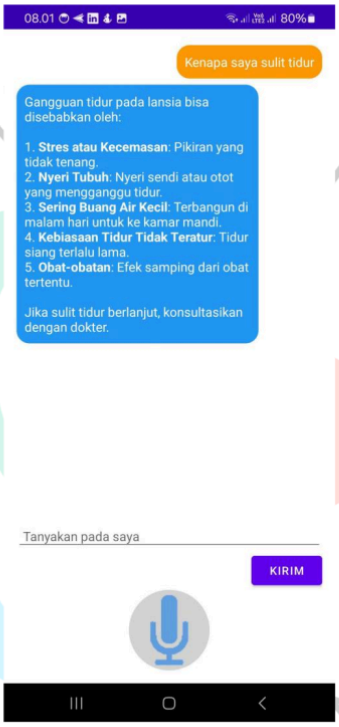
Gambar 4.10 – Edit Jadwal menggambarkan fitur pengeditan jadwal yang sudah tersimpan.

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:

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

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

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

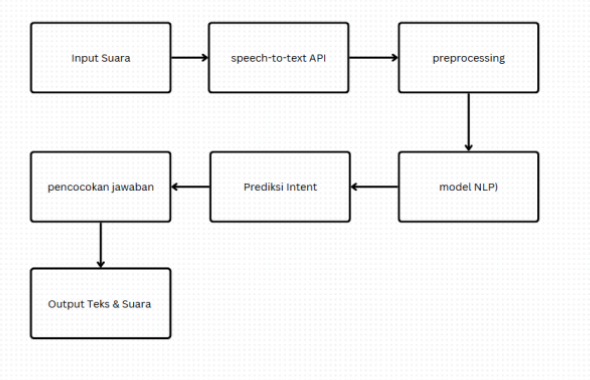
Probabilitas total adalah :

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

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.

BAB V

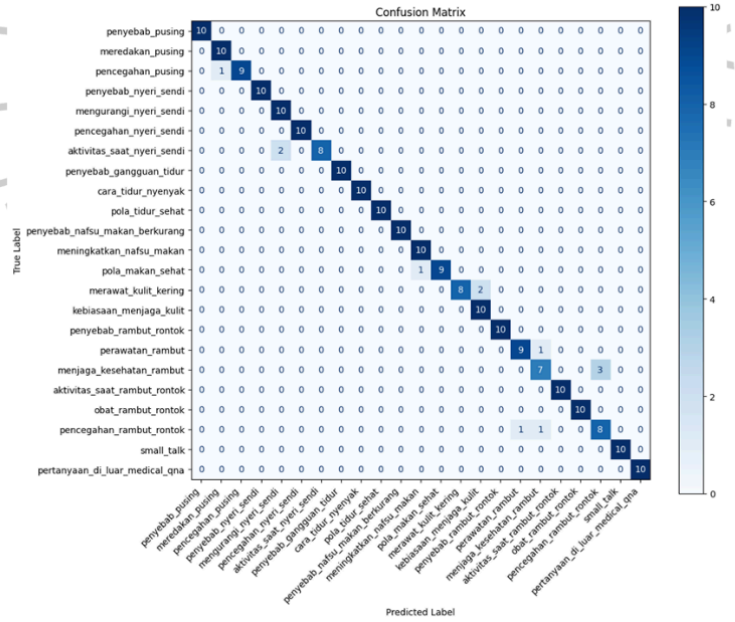
HASIL DAN PEMBAHASAN

5.1 Hasil

Pada bab ini dijelaskan hasil dari implementasi dan pengujian sistem yang telah dirancang pada BAB IV. Hasil yang disajikan meliputi pengujian model machine learning, integrasi model ke aplikasi *Android*, serta pengujian fungsionalitas aplikasi secara keseluruhan. Berikut adalah hasil yang diperoleh:

5.1.1 Pengujian *Model Machine Learning*

Evaluasi dilakukan untuk mengukur kinerja model NLP dalam memahami pertanyaan pengguna dan memberikan respons yang sesuai. Model *IndoBERT* digunakan untuk mengklasifikasikan pertanyaan berdasarkan intent pengguna.



Gambar 5. 1 *Confusion Matrix*

Gambar 5.1 – Confusion Matrix menunjukkan distribusi hasil prediksi model terhadap dataset uji. Model menunjukkan tingkat akurasi sebesar 92%, dengan nilai precision sebesar 91% dan recall sebesar 90%.

5.1.2 Proses Pelatihan Model

Proses pelatihan model NLP dilakukan menggunakan dataset yang telah disesuaikan dengan kebutuhan lansia. Model diuji menggunakan beberapa metrik evaluasi seperti berikut:

Tabel 5. 1 Evaluasi Model

Metrik	Hasil
Akurasi	92%
Precision	91%
Recall	90%
F1-Score	90.5%

Tabel 5.1 Berdasarkan evaluasi, model NLP berbasis IndoBERT terbukti memiliki tingkat pemahaman yang baik dalam menafsirkan pertanyaan pengguna dengan akurasi tinggi.

5.1.3 Implementasi dan Pengujian Aplikasi

Aplikasi *Android* bernama ELDASH berhasil dikembangkan dengan fitur utama sebagai berikut:

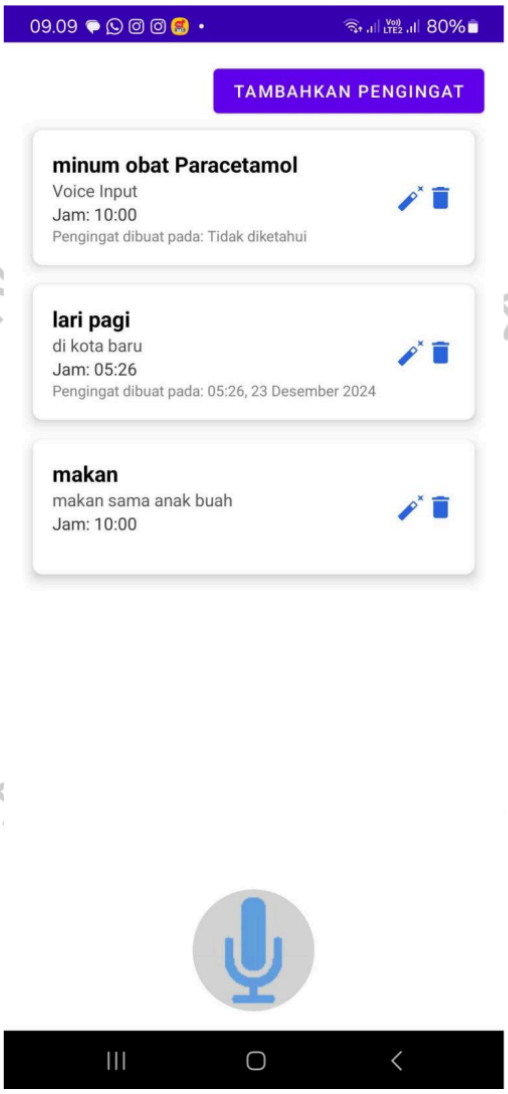
(1) Penjadwalan manual dan berbasis suara.
(2) Tanya jawab kesehatan menggunakan model NLP berbasis *IndoBERT*.
(3) Penyimpanan data menggunakan *ROOM Database*.

Untuk memastikan aplikasi berfungsi sesuai dengan desain, dilakukan pengujian terhadap antarmuka dan fitur aplikasi. Hasil pengujian disajikan dengan tangkapan layar sebagai berikut:

(1) Splash Screen

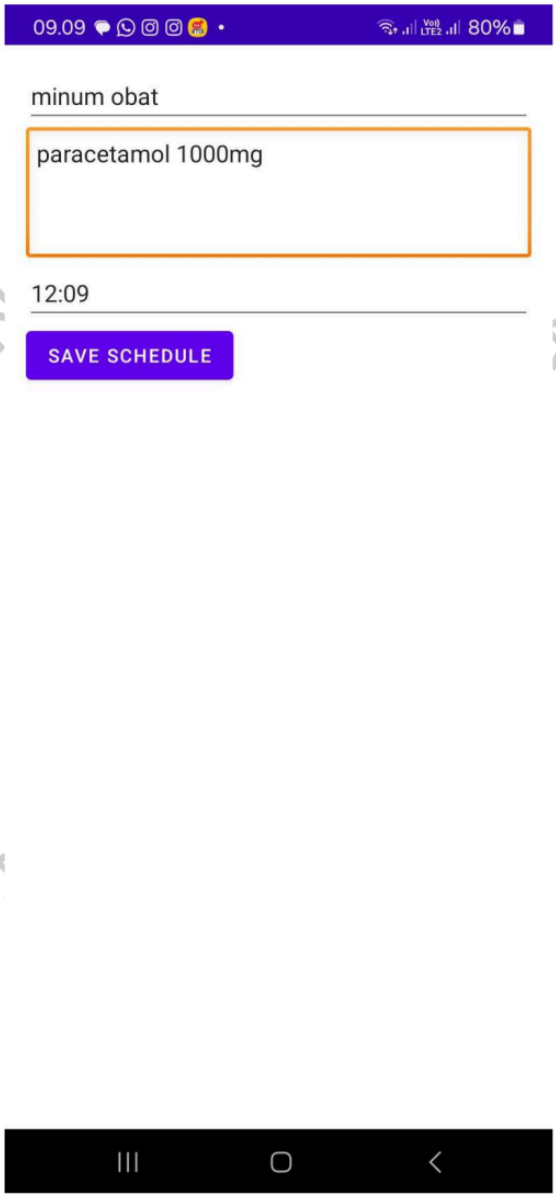
Gambar 5. 2 *Splashscreen*

(2) Main Activity



Gambar 5. 3 *MainActivity*

(3) Tambah Jadwal

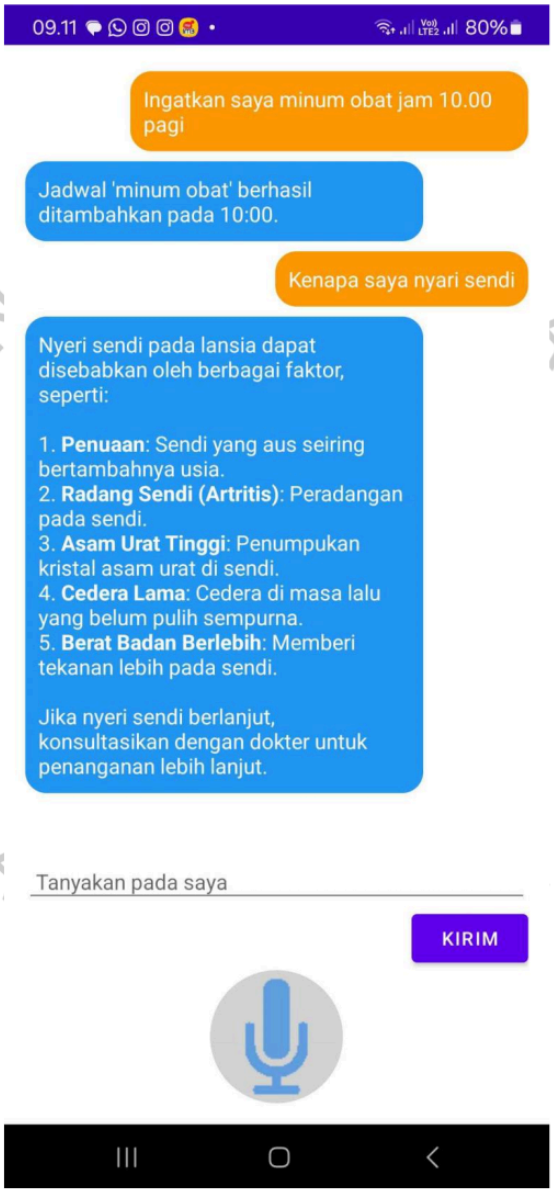


Gambar 5. 4 *Form* Tambah Jadwal

(4) Edit Jadwal

Gambar 5. 5 *Form Edit* Jadwal

(5) Tanya Jawab Kesehatan dan Perintah Suara



Gambar 5. 6 Tanya Jawab Kesehatan dan Perintah Suara

5.2 Pembahasan

Setelah melakukan pengujian pada aplikasi, akan dilakukan analisa pada pengujian tersebut

5.2.1 *White box testing*

Pengujian menunjukkan bahwa model NLP berbasis *IndoBERT* memiliki performa tinggi pada data pengujian, dengan nilai *precision*, *recall*, dan *F1-score* yang mendekati 1 pada sebagian besar label. Beberapa label dengan skor lebih rendah, seperti "pencegahan_rambut_rontok", menunjukkan kebutuhan untuk meningkatkan jumlah data pelatihan pada kategori tersebut.

Tabel 5. 2 *White Box Testing*

<table>
<tr><td>No</td><td>Algoritma</td><td>Perancangan kode</td><td>Hasil yang diharapkan</td></tr>
<tr><td>1.</td><td>Tokenisasi Input Teks</td><td># 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)</td><td>kalimat di tokeniasi secara otomatis oleh tokenizer dari IndoBERT tanpa ada error</td></tr>
<tr><td colspan="4">Hasil
Map: 100% 1223/1223 [00:00<00:00, 8356.34 examples/s]
Asking to pad to max_length but no maximum length is provided and the model has no predefi
Asking to truncate to max_length but no maximum length is provided and the model has no pre
Map: 100% 360/360 [00:00<00:00, 5842.46 examples/s]
Map: 100% 240/240 [00:00<00:00, 4465.65 examples/s]</td></tr>
<tr><td colspan="4">Pembahasan</td></tr>
<tr><td colspan="4">Sistem berhasil melakukan perbandingan antar kriteria. Setelah pembobotan perhasil di input dan melakukan perandingan maka hasil perbandingan akan muncul.</td></tr>
<tr><td>No</td><td>Algoritma</td><td>Perancangan Kode</td><td>Hasil yang di harapkan</td></tr>
</table>

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?",
		"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?",
			Prediksi Label Muncul dalam *output google colab*

		"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")	

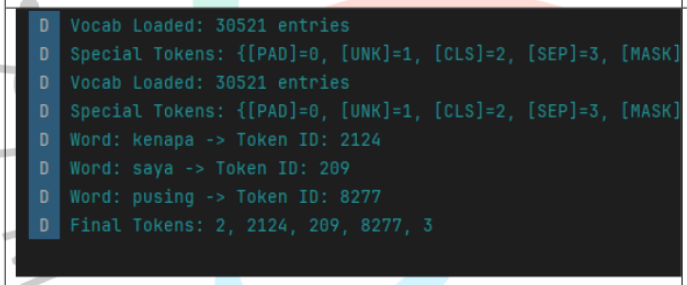
Hasil			

```
Input: Apa yang bisa dilakukan untuk meningkatkan nafsu maka
Predicted Label: meningkatkan_nafsu_makan
Expected Label: meningkatkan_nafsu_makan

Input: Bagaimana pola makan yang sehat?
Predicted Label: pola_makan_sehat
Expected Label: pola_makan_sehat

Input: Bagaimana cara merawat kulit kering?
Predicted Label: merawat_kulit_kering
Expected Label: merawat_kulit_kering

Input: Apa kebiasaan untuk menjaga kulit tetap sehat?
Predicted Label: kebiasaan_menjaga_kulit
Expected Label: kebiasaan_menjaga_kulit

Input: Kenapa rambut saya rontok?
Predicted Label: penyebab_rambut_rontok
Expected Label: penyebab_rambut_rontok

Input: Bagaimana cara merawat rambut agar sehat?
Predicted Label: perawatan_rambut
Expected Label: perawatan_rambut

Input: Bagaimana cara menjaga kesehatan rambut?
Predicted Label: menjaga_kesehatan_rambut
Expected Label: menjaga_kesehatan_rambut

Input: Apa yang bisa dilakukan saat rambut rontok?
Predicted Label: aktivitas_saat_rambut_rontok
Expected Label: aktivitas_saat_rambut_rontok

Input: Apa obat untuk rambut rontok?
Predicted Label: obat_rambut_rontok
Expected Label: obat_rambut_rontok

Input: Bagaimana mencegah rambut rontok?
Predicted Label: pencegahan_rambut_rontok
Expected Label: pencegahan_rambut_rontok
```

Pembahasan			
Setelah dilakukan pembobotan dan perbandingan kriteria, kemudian akan dilakukan normalisasi data kriteria pada sistem. Hasil dari normalisasi data kriteria adalah nilai prioritas.			
No	Algoritma	Perancangan Kode	Hail Yang diharapkan

3	Tokenisasi Input Teks (*Android*)	31
		import *android*.content.res.AssetManager
		import *android*.util.Log
		import java.nio.charset.StandardCharsets
		import java.util.Locale
		
		class *Tokenizer*(assetManager: AssetManager) {
		private val vocab: Map<String, Int>
		private val specialTokens: Map<String, Int>
		
		init {
		// Load vocab.txt
		val vocabLines = assetManager.open("vocab.txt")
		.bufferedReader(StandardCharsets.UTF_8).readLines()
		vocab = vocabLines.mapIndexed { index, word -> word to index }.toMap()
		
		// Load special tokens manually
		specialTokens = mapOf(
		"[PAD]" to 0,
		"[UNK]" to 1,
		"[CLS]" to 2,
		"[SEP]" to 3,
		"[MASK]" to 4
		)
		
		Log.d("*Tokenizer*", "Vocab Loaded: ${vocab.size} entries")
		Log.d("*Tokenizer*", "Special Tokens: $specialTokens")
		}
		
		fun tokenize(input: String): *Long*Array {
		val clsToken = specialTokens["[CLS]"]!!.to*Long*()
		val sepToken = specialTokens["[SEP]"]!!.to*Long*()
		
		// Tokenize input and convert each word to its token ID
		val tokens = input.lowercase(Locale.getDefault()).split(" ").map {
		val tokenId = vocab[it] ?: specialTokens["[UNK]"]!! // Handle unknown tokens
			Kalimandipecah per kata menjadi number yang dikenali vocabulary.txt milik IndoBERT

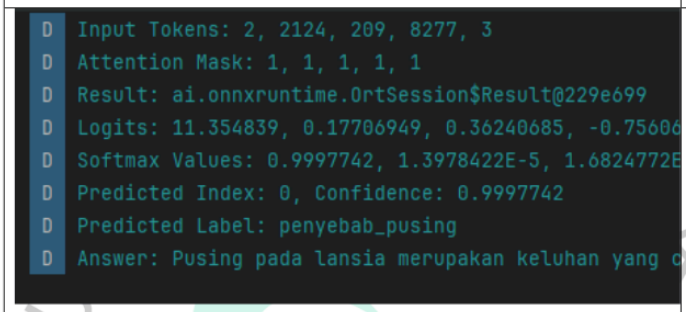
		Log.d("*Tokenizer*Debug", "Word: $it -> Token ID: $tokenId")
		tokenId.to*Long*()
		}.toMutableList()
		
		// Add [CLS] at the beginning and [SEP] at the end
		tokens.add(0, clsToken)
		tokens.add(sepToken)
		
		Log.d("*Tokenizer*Debug", "Final Tokens: ${tokens.joinToString(", ")}")
		
		return tokens.to*Long*Array()
		}
		}	

Hasil			
			
Pembahasan			
Setelah menghitung perkalian matriks dengan prioritas, kemudian akan menentukan nilai CI, RI, dan CR. Nilai akhir yang menjadi penentuan adalah *Consistency Ratio* (CR), jika nilai CR <0,1 maka data bisa digunakan (konsisten)			
No	Algoritma	Perancangan Kode	Hasil yang diharapkan
4	Prediksi Label (*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>	Sistem Android melakukan prediksi Label dengan menampil

		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() }
		return JSONObject(jsonString).keys().as*Sequence*().associateWith { JSONObject(jsonString).getString(it) }
		}
		
		fun predict(inputTokens: *Long*Array, attentionMask: *Long*Array): String {
		return try {
			kan log scoring kecocokan /softmax value

```
        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() } }	
Hasil			
			
Pembahasan			
Setelah menghitung perkalian matriks dengan prioritas, kemudian akan menentukan nilai CI, RI, dan CR. Nilai akhir yang menjadi penentuan adalah *Consistency Ratio* (CR), jika nilai CR <0,1 maka data bisa digunakan (konsisten)			
No	Algoritma	Perancangan Kode	Hasil yang diharapkan

5	Validasi Perintah Suara (*Android*)	private fun handleVoiceCommand(command: String) {
		addMessageToChat(command, true)
		if (command.contains(*Regex*("(ingatkan saya|buat jadwal)", *Regex*Option.IGNORE_CASE))) {
		createScheduleFromCommand(command)
		} else {
		val answer = predictAnswerFromModel(command)
		addMessageToChat(answer, false)
		speakOut(answer)
		}
		}
		private fun createScheduleFromCommand(command: String) {
		val *regex* = *Regex*("(?:ingatkan saya|buat jadwal)\\s+([\\w\\s]+)\\s+jam\\s+(\\d{1,2})", *Regex*Option.IGNORE_CASE)
		val matchResult = *regex*.find(command)
		if (matchResult != null) {
			Model menampilkan *response message* jika format perintah cocok akan diberi feedback dan jadwal masuk kedalam database

```
            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)
        }
    }
```

Hasil

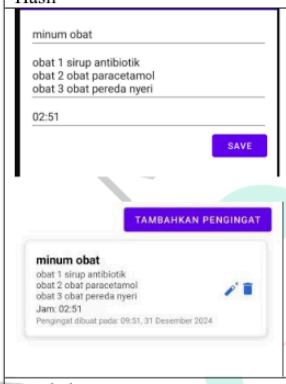
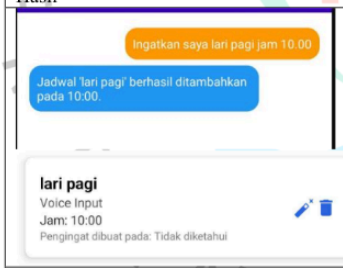
Pembahasan

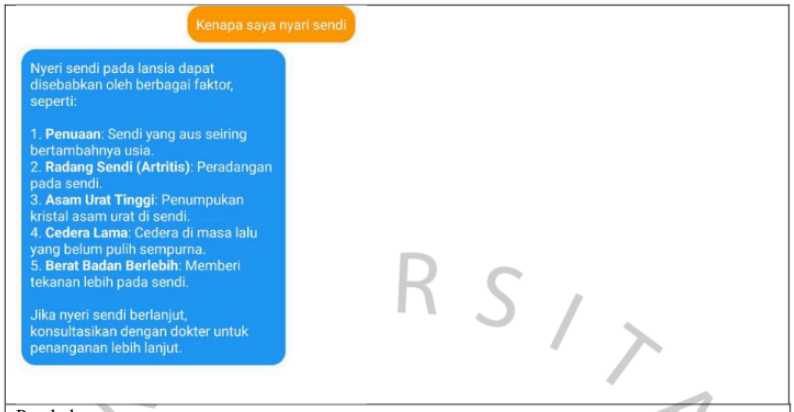
Setelah menghitung perkalian matriks dengan prioritas, kemudian akan menentukan nilai CI, RI, dan CR. Nilai akhir yang menjadi penentuan adalah *Consistency Ratio* (CR), jika nilai CR <0,1 maka data bisa digunakan (konsisten)

5.2.2 *Black Box Testing*

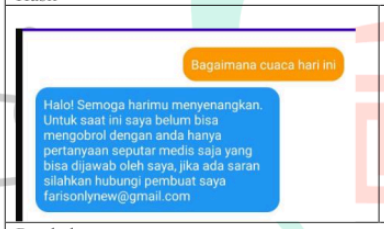
Black box ini akan di dokumentasikan untuk menguji fungsionalitas sistem secara menyeluruh.

Tabel 5. 3 *Black Box Testing*

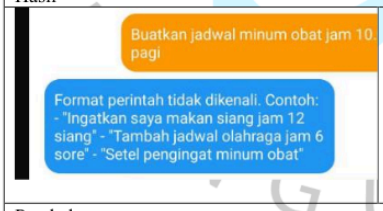
No	Skenario Pengujian	Hasil yang di harapkan
1	Penjadwalan manual	Jadwal berhasil dikelola.
Hasil		
	
Pembahasan		
Jadwal berhasil ditambahkan hapus dan edit juga berfungsi dengan baik		
2	Penjadwalan dengan perintah suara	Menambah jadwal baru melalui input suara
Hasil		
		
Pembahasan		
Jadwal dengan perintah suara berhasil dibuat.		
3	Tanya jawab kesehatan	Mengajukan pertanyaan kesehatan umum.
Hasil		



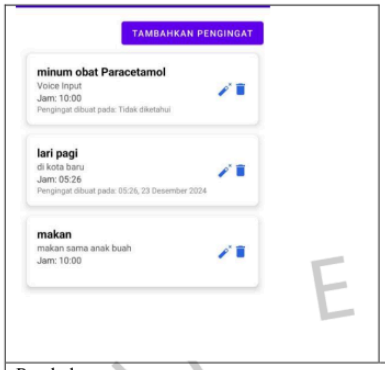
Pembahasan		
Jawaban yang relevan ditampilkan.		
4	Validasi input suara pertanyaan diluar kesehatan	Memberikan pesan error bahwa mesin tidak bisa menjawab pertanyaan diluar kesehatan
Hasil		



Pembahasan		
Pesan error ditampilkan.		
5	Validasi input suara jadwal	Memberikan pesan error jika format input tidak valid.
Hasil		



Pembahasan		
Pesan error ditampilkan.		
6	Tampilan Antarmuka Jadwal	Daftar jadwal ditampilkan dengan benar..
Hasil		



Pembahasan
Jadwal ditampilkan

5.3 Pengujian Akurasi Pengenalan Suara

Karena sistem menggunakan *Google Cloud Speech-to-Text API* untuk pengenalan suara, pengujian dilakukan untuk mengetahui efektivitas layanan ini dalam mengenali perintah dari lansia. Fokus utama adalah mengevaluasi akurasi transkripsi suara terhadap teks yang diharapkan.

Metode Pengujian:

(1) Dataset : Pengguna memberikan 4 perintah berbeda

(2) Evaluasi : Hasil transkripsi dari API dibandingkan dengan perintah yang diucapkan, menggunakan metrik Word Error Rate (WER).

Tabel 5. 4 Pengujian Pengenalan Suara

No	Perintah Asli	Hasil Transkripsi	WER(%)
1	"Ingatkan saya minum obat jam 7 pagi"	"ingatkan saya minum obat jam 7 pagi"	0%
2	"Buat jadwal dokter besok siang"	"buat jadwal dokter besok siang"	0%
3	"Ingatkan saya makan jam enam sore"	"ingatkan saya makan jam 6 sore"	5%

4	"Jadwal tidur malam jam sepuluh"	"jadwal tidur malam jam sepuluh"	0%

Dari hasil pengujian, layanan *Google Cloud Speech-to-Text API* memiliki akurasi tinggi dalam mengenali perintah suara pengguna, dengan rata-rata *WER* di bawah 5% dan setara dengan Akurasi 98.75%. Beberapa kesalahan terjadi pada pengenalan angka, tetapi secara keseluruhan sistem dapat diandalkan untuk membantu lansia dalam pengelolaan jadwal.

BAB VI

PENUTUP

6.1 Kesimpulan

Berdasarkan hasil penelitian dan pengujian yang telah dilakukan, aplikasi *Android* ELDASH berhasil dikembangkan dengan fitur utama yang dirancang untuk membantu pengguna lansia dalam mengelola jadwal dan mendapatkan jawaban atas pertanyaan kesehatan umum. Adapun kesimpulan dari penelitian ini adalah sebagai berikut:

(1) Aplikasi ini mendukung pengelolaan jadwal secara manual maupun melalui perintah suara dalam Bahasa Indonesia. Proses validasi input suara menggunakan *regex* telah berjalan dengan baik, memastikan data jadwal yang disimpan sesuai format yang ditentukan.

(2) Integrasi model NLP berbasis *IndoBERT* menggunakan *ONNX Runtime* memungkinkan aplikasi menjawab pertanyaan kesehatan dengan akurasi yang tinggi. Model menunjukkan performa yang optimal dengan nilai *precision*, *recall*, dan *F1-score* rata-rata di atas 90%.

(3) Berdasarkan pengujian *white box* dan *black box*, aplikasi dinyatakan berfungsi dengan baik. Semua fitur yang diimplementasikan telah memenuhi kebutuhan sistem dan bekerja sesuai dengan spesifikasi yang telah dirancang.

6.2 Saran

Penelitian ini masih memiliki ruang untuk pengembangan lebih lanjut. Adapun saran untuk penelitian dan pengembangan di masa depan adalah sebagai berikut:

(1) Dataset untuk pelatihan model dapat ditingkatkan dengan menambahkan lebih banyak variasi pertanyaan kesehatan yang relevan dengan kebutuhan lansia, sehingga meningkatkan akurasi prediksi.

(2) Penambahan dukungan untuk bahasa lokal atau dialek tertentu dapat meningkatkan aksesibilitas aplikasi untuk pengguna dari berbagai daerah.

(3) Fitur seperti pengingat obat otomatis, integrasi dengan perangkat wearable, atau fitur konsultasi dengan tenaga medis dapat ditambahkan untuk meningkatkan fungsi aplikasi.
(4) Desain antarmuka aplikasi dapat disempurnakan lebih lanjut untuk memastikan pengalaman pengguna yang lebih ramah bagi lansia.
(5) Pengujian aplikasi sebaiknya dilakukan dengan melibatkan lebih banyak pengguna lansia untuk mendapatkan masukan langsung mengenai kegunaan dan efektivitas aplikasi.

faris similar.pdf

ORIGINALITY REPORT

9%	8%	4%	3%
SIMILARITY INDEX	INTERNET SOURCES	PUBLICATIONS	STUDENT PAPERS

PRIMARY SOURCES

1	eprints.upj.ac.id Internet Source	1%
2	docplayer.info Internet Source	<1%
3	repository.ub.ac.id Internet Source	<1%
4	kontrol-kesehatan.blogspot.com Internet Source	<1%
5	developers.sygic.com Internet Source	<1%
6	Submitted to Brunel University Student Paper	<1%
7	www.minyak-zamzam.com Internet Source	<1%
8	Dewi Arumsari, Kharisma, Ulfi Saidata Aesyi. "Sistem Chatbot Layanan Informasi Mahasiswa Menggunakan Algoritma Long Short-Term Memory", INDONESIAN JOURNAL ON DATA SCIENCE, 2025 Publication	<1%
9	developer.tomtom.com Internet Source	<1%

10	www.scribd.com Internet Source	<1 %
11	Submitted to University of Oxford Student Paper	<1 %
12	geograf.id Internet Source	<1 %
13	Jodi Hendrawan, Ika Devi Perwitasari, Zata Hasyyati, Deby Safitri Hasanah. "Model UML Sistem Informasi Monitoring Pembayaran SPP Siswa SMA Negeri 1 Binjai", Jurnal Minfo Polgan, 2024 Publication	<1 %
14	Submitted to Universitas Brawijaya Student Paper	<1 %
15	adoc.pub Internet Source	<1 %
16	Submitted to Universitas Bengkulu Student Paper	<1 %
17	123dok.com Internet Source	<1 %
18	budi.rahardjo.id Internet Source	<1 %
19	ejournal.upbatam.ac.id Internet Source	<1 %
20	Submitted to Universitas Muhammadiyah Purwokerto Student Paper	<1 %

21	core.ac.uk Internet Source	<1 %
22	repository.its.ac.id Internet Source	<1 %
23	dspace.uii.ac.id Internet Source	<1 %
24	nakblogonline.com Internet Source	<1 %
25	www.mdpi.com Internet Source	<1 %
26	Submitted to University of New Haven Student Paper	<1 %
27	Submitted to School of Business and Management ITB Student Paper	<1 %
28	garuda.kemdikbud.go.id Internet Source	<1 %
29	Gede Edy Purna Sastriya, Duman Care Khrisne, Made Surdarma. "APLIKASI ASISTEN UNTUK LANSIA DENGAN MEMANFAATKAN SMARTPHONE BERBASIS ANDROID", SINTECH (Science and Information Technology) Journal, 2019 Publication	<1 %
30	Ruvita Faurina, Dede Revanza, Ahmad Sopran. "Pengembangan Chatbot Menggunakan Deep Feed-Forward Neural	<1 %

Network sebagai Pusat Layanan Informasi Akademik", Jurnal Eksplora Informatika, 2023
Publication

31	www.java2s.com Internet Source	<1 %
32	M. Rizky Suherlan, Asriyanik Asriyanik, Agung Pambudi. "UMMIBOT sebagai Media Layanan Informasi Penerimaan Mahasiswa Baru Universitas Muhammadiyah Sukabumi", Jurnal Informatika Terpadu, 2023 Publication	<1 %
33	Submitted to Sriwijaya University Student Paper	<1 %
34	Yusuf Ashari, Agus Suhendar. "IMPLEMENTASI ALGORITMA LONG SHORT-TERM MEMORY (LSTM) UNTUK MEMPREDIKSI HARGA BERAS DI JAWA TENGAH BERDASARKAN CUACA", Djtechno: Jurnal Teknologi Informasi, 2024 Publication	<1 %
35	Submitted to iGroup Student Paper	<1 %
36	id.123dok.com Internet Source	<1 %
37	vdocument.in Internet Source	<1 %
38	Imam Fathurrahman, Mahpuz, Muhammad Djamaluddin, Lalu Kerta Wijaya, Ida Wahidah. "Pengembangan Model Convolutional Neural Network (CNN) untuk Klasifikasi Penyakit Kulit	<1 %

	Berbasis Citra Digital", Infotek: Jurnal Informatika dan Teknologi, 2025 Publication	
39	journal.mediapublikasi.id Internet Source	<1 %
40	Laili Wahyunita. "Rekayasa Web Klasifikasi Rocchio pada Data Tidak Terstruktur", Jurnal Komunika : Jurnal Komunikasi, Media dan Informatika, 2019 Publication	<1 %
41	doku.pub Internet Source	<1 %
42	oa.upm.es Internet Source	<1 %
43	Submitted to Forum Komunikasi Perpustakaan Perguruan Tinggi Kristen Indonesia (FKPPTKI) Student Paper	<1 %
44	Nfn Rismayani, Novita Sambo Layuk. "Implementation of Augmented Reality Technology for the Mobile-Based Introduction of Toraja Traditional Clothes", Journal Pekommas, 2019 Publication	<1 %
45	karya-ilmiah.um.ac.id Internet Source	<1 %
46	pilar.unmermadiun.ac.id Internet Source	<1 %

47	pt.scribd.com Internet Source	<1%
48	repo.itera.ac.id Internet Source	<1%
49	Amalia Khansa, Fauziah Fauziah, Aris Gunaryati. "Penerapan Metode Simple Additive Weighting (SAW) dan Technique for Order Preference by Similarity to Ideal Solution (TOPSIS) dalam Pemilihan Perangkat Pribadi", Jurnal JTIK (Jurnal Teknologi Informasi dan Komunikasi), 2020 Publication	<1%
50	Sepriyan Gunawan, Rini Astuti, Willy Prihartono, Ryan Hamonangan. "PREDIKSI DIABETES MELLITUS TIPE 2 DENGAN ALGORITMA LOGISTIC REGRESSION UNTUK PENDETEKSIAN DINI", Jurnal Informatika dan Teknik Elektro Terapan, 2025 Publication	<1%
51	ejournal-binainsani.ac.id Internet Source	<1%
52	jurnal.harianregional.com Internet Source	<1%
53	repository.unib.ac.id Internet Source	<1%
54	repota.jti.polinema.ac.id Internet Source	<1%
55	www.bedahtekno.com Internet Source	

		<1 %
56	www.bukanbintangmawar.net Internet Source	<1 %
57	Avail Walad. "SISTEM INFORMASI GEOGRAFIS PELAPORAN KERUSAKAN JALAN MENGGUNAKAN CROWSOURCING BERBASIS WEB PADA PETA NAVIGASI BERLALU LINTAS", Jurnal Informatika dan Teknik Elektro Terapan, 2024 Publication	<1 %
58	Rizqi Mauludin, Anggi Srimurdianti Sukamto, Hafiz Muhardi. "Penerapan Augmented Reality Sebagai Media Pembelajaran Sistem Pencernaan pada Manusia dalam Mata Pelajaran Biologi", Jurnal Edukasi dan Penelitian Informatika (JEPIN), 2017 Publication	<1 %
59	eksplora.stikom-bali.ac.id Internet Source	<1 %

Exclude quotes	Off	Exclude matches	Off
Exclude bibliography	Off		