

BAB V

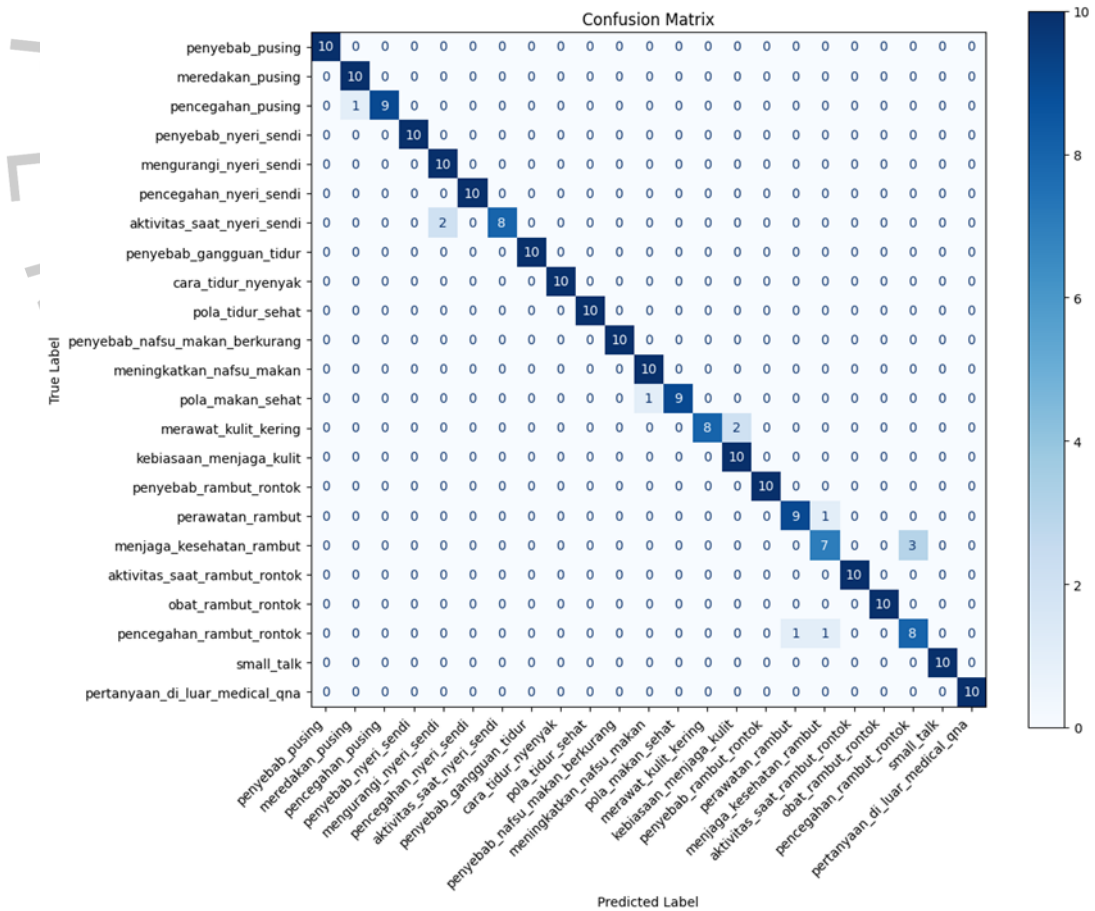
HASIL DAN PEMBAHASAN

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*

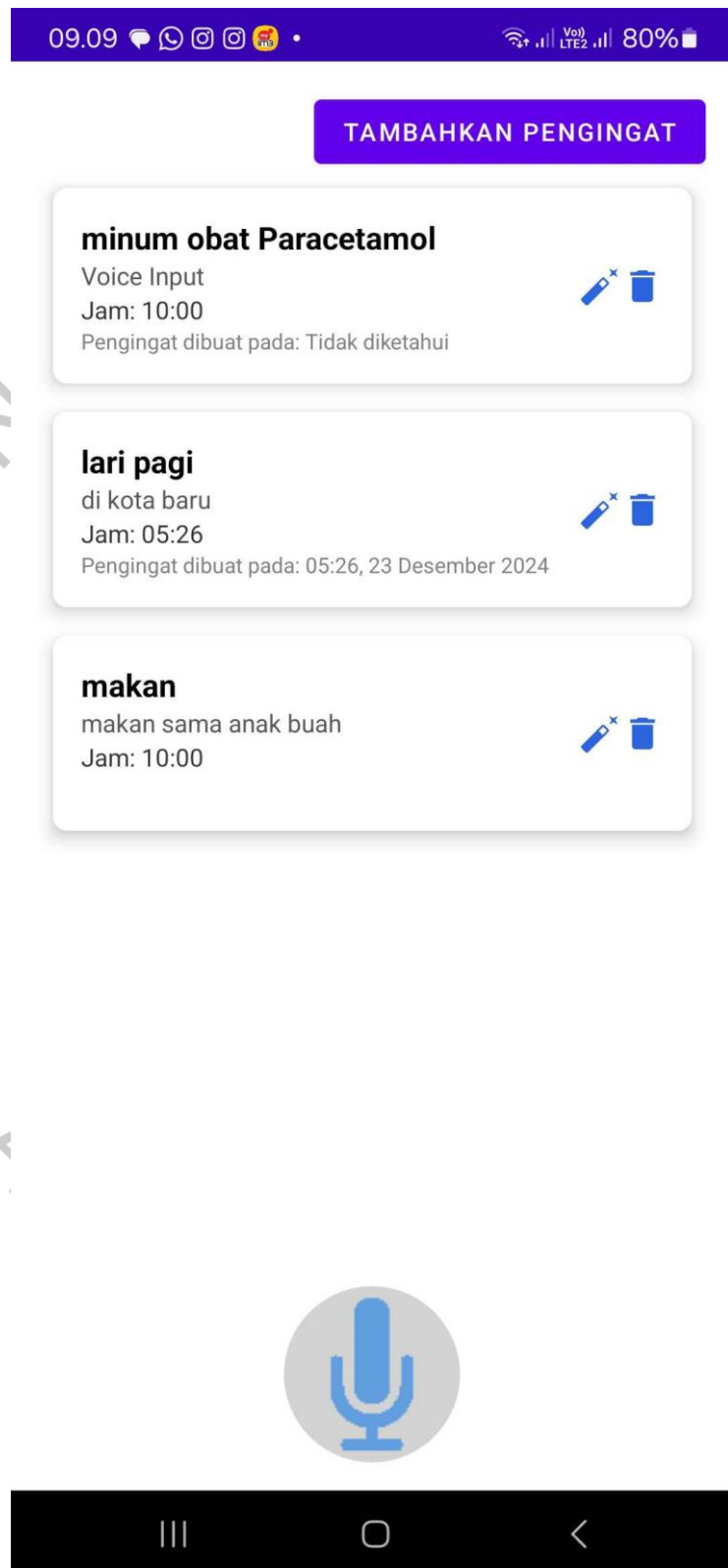


ELDASH
sahabat lansia anda



Gambar 5. 2 *Splashscreen*

(2) Main Activity



Gambar 5. 3 *MainActivity*

(3) Tambah Jadwal

09.09 80%

minum obat

paracetamol 1000mg

12:09

SAVE SCHEDULE

Gambar 5. 4 Form Tambah Jadwal

(4) Edit Jadwal

09:10 [social media icons] [LTE2] 80%

minum obat

paracetamol 1000mg (halaman edit yang tadi)

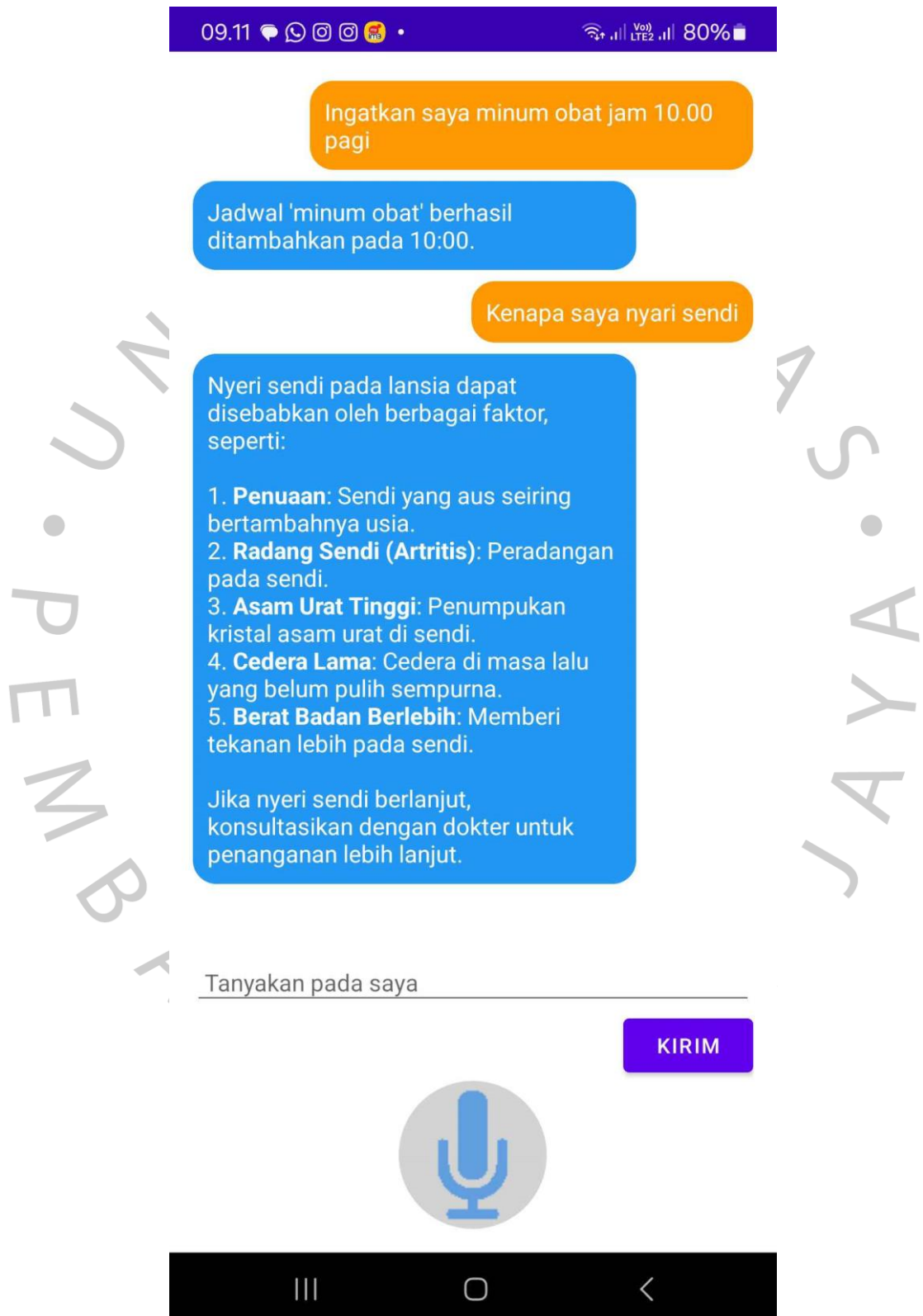
12:09

SAVE

[list icon] [home icon] [back icon]

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*

No	Algoritma	Perancangan kode	Hasil yang diharapkan
1	Tokenisasi Input Teks	<pre># 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)</pre>	kalimat di tokeniasi secara otomatis oleh tokenizer dari IndoBERT tanpa ada error
Hasil			
<pre>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 predefined 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]</pre>			
Pembahasan			
Sistem berhasil melakukan perbandingan antar kriteria. Setelah pembobotan perhasil di input dan melakukan perbandingan maka hasil perbandingan akan muncul.			
No	Algoritma	Perancangan Kode	Hasil yang di harapkan

2	Prediksi Label	<pre># 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?",</pre>	Prediksi Label Muncul dalam <i>output google colab</i>
---	----------------	--	--

	<pre> "penyebab_rambut_rontok": "Kenapa rambut saya rontok?", "perawatan_rambut": "Bagaimana cara merawat rambut agar sehat?", "menjaga_kesehatan_rambut": "Bagaimana cara menjaga kesehatan rambut?", "aktivitas_saar_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") </pre>	
Hasil		

<p>Input: Apa yang bisa dilakukan untuk meningkatkan nafsu makan? Predicted Label: meningkatkan_nafsu_makan Expected Label: meningkatkan_nafsu_makan</p> <p>Input: Bagaimana pola makan yang sehat? Predicted Label: pola_makan_sehat Expected Label: pola_makan_sehat</p> <p>Input: Bagaimana cara merawat kulit kering? Predicted Label: merawat_kulit_kering Expected Label: merawat_kulit_kering</p> <p>Input: Apa kebiasaan untuk menjaga kulit tetap sehat? Predicted Label: kebiasaan_menjaga_kulit Expected Label: kebiasaan_menjaga_kulit</p> <p>Input: Kenapa rambut saya rontok? Predicted Label: penyebab_rambut_rontok Expected Label: penyebab_rambut_rontok</p> <p>Input: Bagaimana cara merawat rambut agar sehat? Predicted Label: perawatan_rambut Expected Label: perawatan_rambut</p> <p>Input: Bagaimana cara menjaga kesehatan rambut? Predicted Label: menjaga_kesehatan_rambut Expected Label: menjaga_kesehatan_rambut</p> <p>Input: Apa yang bisa dilakukan saat rambut rontok? Predicted Label: aktivitas_saat_rambut_rontok Expected Label: aktivitas_saat_rambut_rontok</p> <p>Input: Apa obat untuk rambut rontok? Predicted Label: obat_rambut_rontok Expected Label: obat_rambut_rontok</p> <p>Input: Bagaimana mencegah rambut rontok? Predicted Label: pencegahan_rambut_rontok Expected Label: pencegahan_rambut_rontok</p>			
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	Hasil Yang diharapkan

3	Tokenisasi Input Teks (Android)	<pre> 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): LongArray { val clsToken = specialTokens["[CLS]"]!!.toLong() val sepToken = specialTokens["[SEP]"]!!.toLong() // 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 </pre>	Kaliman dipecah per kata menjadi number yang dikenali vocabular y.txt milik IndoBERT
---	---------------------------------	---	--


		<pre> Log.d("TokenizerDebug", "Word: \$it -> Token ID: \$tokenId") tokenId.toLong() }.toMutableList() // Add [CLS] at the beginning and [SEP] at the end tokens.add(0, clsToken) tokens.add(sepToken) Log.d("TokenizerDebug", "Final Tokens: \${tokens.joinToString(", ")}") return tokens.toLongArray() } } </pre>	
Hasil			
		<pre> D Vocab Loaded: 30521 entries D Special Tokens: {[PAD]=0, [UNK]=1, [CLS]=2, [SEP]=3, [MASK] D Vocab Loaded: 30521 entries D Special Tokens: {[PAD]=0, [UNK]=1, [CLS]=2, [SEP]=3, [MASK] D Word: kenapa -> Token ID: 2124 D Word: saya -> Token ID: 209 D Word: pusing -> Token ID: 8277 D Final Tokens: 2, 2124, 209, 8277, 3 </pre>	
Pembahasan			
		Setelah menghitung perkalian matriks dengan prioritas, kemudian akan menentukan nilai CI, RI, dan CR. Nilai akhir yang menjadi penentuan adalah <i>Consistency Ratio</i> (CR), jika nilai CR <0,1 maka data bisa digunakan (konsisten)	
No	Algoritma	Perancangan Kode	Hasil yang diharapkan
4	Prediksi Label (<i>Android</i>)	<pre> 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> </pre>	Sistem Android melakukan prediksi Label dengan menampilkan

	<pre> 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().asSequence().associateWith { JSONObject(jsonString).getString(it) } } fun predict(inputTokens: LongArray, attentionMask: LongArray): String { return try { </pre>	<pre> kan log scoring kecocokan /softmax value </pre>
--	---	---

		<pre> 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(", ")}") </pre>	
--	--	--	--

	<pre> 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()] } 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() { </pre>	
--	---	--

		<pre> session.close() ortEnvironment.close() } } </pre>	
Hasil			
<pre> D Input Tokens: 2, 2124, 209, 8277, 3 D Attention Mask: 1, 1, 1, 1, 1 D Result: ai.onnxruntime.OrtSession\$Result@229e699 D Logits: 11.354839, 0.17706949, 0.36240685, -0.75606 D Softmax Values: 0.9997742, 1.3978422E-5, 1.6824772E D Predicted Index: 0, Confidence: 0.9997742 D Predicted Label: penyebab_pusing D Answer: Pusing pada lansia merupakan keluhan yang d </pre>			
Pembahasan			
Setelah menghitung perkalian matriks dengan prioritas, kemudian akan menentukan nilai CI, RI, dan CR. Nilai akhir yang menjadi penentuan adalah <i>Consistency Ratio</i> (CR), jika nilai CR <0,1 maka data bisa digunakan (konsisten)			
No	Algoritma	Perancangan Kode	Hasil yang diharapkan
5	Validasi Perintah Suara (Android)	<pre> 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) { 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) { </pre>	Model menampilkan <i>response message</i> jika format perintah cocok akan diberi feedback dan jadwal masuk kedalam database

	<pre> 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).scheduledD ao().insert(newSchedule) runOnUiThread { addMessageToChat("Jadwal '\$label' berhasil ditambahkan pada \$time.", false) } } else { addMessageToChat("Perintah tidak dikenali. Contoh: 'Ingatkan saya olahraga jam 7'.", false) } } </pre>	
Hasil		
Pembahasan		
	<p>Setelah menghitung perkalian matriks dengan prioritas, kemudian akan menentukan nilai CI, RI, dan CR. Nilai akhir yang menjadi penentuan adalah <i>Consistency Ratio</i> (CR), jika nilai $CR < 0,1$ maka data bisa digunakan (konsisten)</p>	

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		

<p style="text-align: center;">Kenapa saya nyari sendi</p> <p>Nyeri sendi pada lansia dapat disebabkan oleh berbagai faktor, seperti:</p> <ol style="list-style-type: none"> 1. Penuaan: Sendi yang aus seiring bertambahnya usia. 2. Radang Sendi (Artritis): Peradangan pada sendi. 3. Asam Urat Tinggi: Penumpukan kristal asam urat di sendi. 4. Cedera Lama: Cedera di masa lalu yang belum pulih sempurna. 5. Berat Badan Berlebih: Memberi tekanan lebih pada sendi. <p>Jika nyeri sendi berlanjut, konsultasikan dengan dokter untuk penanganan lebih lanjut.</p>		
Pembahasan		
Jawaban yang relevan ditampilkan.		
4	Validasi input suara pertanyaan diluar kesehatan	Memberikan pesan error bahwa mesin tidak bisa menjawab pertanyaan diluar kesehatan
Hasil		
<p style="text-align: center;">Bagaimana cuaca hari ini</p> <p>Halo! Semoga harimu menyenangkan. Untuk saat ini saya belum bisa mengobrol dengan anda hanya pertanyaan seputar medis saja yang bisa dijawab oleh saya, jika ada saran silahkan hubungi pembuat saya farisonlynew@gmail.com</p>		
Pembahasan		
Pesan error ditampilkan.		
5	Validasi input suara jadwal	Memberikan pesan error jika format input tidak valid.
Hasil		
<p style="text-align: center;">Buatkan jadwal minum obat jam 10 pagi</p> <p>Format perintah tidak dikenali. Contoh: - "Ingatkan saya makan siang jam 12 siang" - "Tambah jadwal olahraga jam 6 sore" - "Setel pengingat minum obat"</p>		
Pembahasan		
Pesan error ditampilkan.		
6	Tampilan Antarmuka Jadwal	Daftar jadwal ditampilkan dengan benar..
Hasil		

TAMBAHKAN PENGINGAT
<div style="border: 1px solid #ccc; padding: 5px; margin-bottom: 5px;"> <p>minum obat Paracetamol Voice Input Jam: 10:00 Peringat dibuat pada: Tidak diketahui</p> </div> <div style="border: 1px solid #ccc; padding: 5px; margin-bottom: 5px;"> <p>lari pagi di kota baru Jam: 05:26 Peringat dibuat pada: 05:26, 23 Desember 2024</p> </div> <div style="border: 1px solid #ccc; padding: 5px;"> <p>makan makan sama anak buah Jam: 10:00</p> </div>
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.

