BAB IV

HASIL DAN PEMBAHASAN

4.1. Rancang Bangun Aplikasi

4.1.1. Perancangan Sistem & Implementasi

Arsitektur sistem aplikasi, berdasarkan analisis kebutuhan pada Bab III, diimplementasikan seperti yang ditunjukkan pada Gambar 3.2. Model client-server dengan integrasi *Google AI* (*Gemini API*) dipilih untuk mencapai otomatisasi dan efisiensi. Berikut adalah beberapa dependensi kunci yang digunakan:

Tabel 4.4. Tabel Depedensi *Flutter*

Dependensi	**Versi**	**Fungsi**
curved_navigation_bar	^1.0.6	Menyediakan *widget* navigasi bawah dengan tampilan melengkung yang modern dan interaktif, meningkatkan *user experience* navigasi antar halaman aplikasi.
device_info_plus	^11.0.0	Mengambil informasi detail tentang perangkat yang digunakan pengguna, seperti model perangkat, sistem operasi, versi OS, dan informasi lainnya. Data ini dapat digunakan untuk keperluan *debugging*, analitik, dan personalisasi pengalaman pengguna.
*file*_picker	^8.0.0+1	Memungkinkan pengguna untuk memilih dan mengunggah *file* dari berbagai sumber di perangkat, seperti penyimpanan internal, kartu SD, atau layanan penyimpanan cloud. Dependensi ini penting untuk fungsionalitas pengunggahan *CV*.
firebase_core	^3.6.0	Inisialisasi dan koneksi dasar ke layanan Firebase. Dependensi ini wajib ada untuk menggunakan layanan Firebase lainnya, seperti autentikasi, database, dan penyimpanan.

firebase_auth	5.3.2	Menyediakan layanan autentikasi pengguna menggunakan Firebase Authentication. Memungkinkan implementasi fitur *login*, registrasi, dan manajemen akun pengguna dengan aman.
firebase_database	^11.1.4	Mengakses *Firebase Realtime Database*, yang memungkinkan penyimpanan dan pengambilan data secara *real-time*. Data lowongan, kriteria, dan hasil analisis *CV* disimpan di database ini.
firebase_storage	^12.3.3	Mengakses Firebase Cloud Storage untuk menyimpan dan mengambil *file*, seperti *CV* yang diunggah oleh pelamar. Cloud Storage menyediakan solusi penyimpanan yang skalabilitas dan aman.
google_generative_ai	^0.4.6	Menyediakan integrasi dengan *Google AI* (*Gemini API*). Dependensi ini memungkinkan aplikasi untuk berkomunikasi dengan *Gemini API*, mengirimkan data *CV* untuk dianalisis, dan menerima hasil analisis.
permission_handler	^11.3.1	Mengelola permintaan dan penanganan izin aplikasi pada perangkat pengguna. Berguna untuk meminta izin akses ke fitur-fitur perangkat, seperti penyimpanan *file* dan kamera.
path_provider	^2.1.4	Menyediakan akses ke jalur direktori umum pada sistem *file* perangkat, seperti direktori untuk menyimpan *file* sementara atau *cache*.

Dengan menggunakan dependensi ini, aplikasi Anda dapat mengelola autentikasi pengguna, menyimpan dan mengambil data dari Firebase, mengunggah *file*, dan menggunakan AI untuk analisis *CV*, semuanya dengan efisien dan efektif. Jika Anda memiliki pertanyaan lebih lanjut atau memerlukan bantuan tambahan, jangan ragu untuk bertanya!

4.1.2. Perancangan Database

Perancangan database adalah langkah penting dalam pengembangan aplikasi untuk memastikan data disimpan dengan cara yang efisien dan

terstruktur. Berdasarkan konteks yang diberikan, kita akan menggunakan *Firebase Realtime Database* untuk menyimpan data pengguna, kriteria pekerjaan, dan analisis *CV*, *Firebase Realtime Database* dipilih karena kemampuan sinkronisasi data secara real-time yang memudahkan pengembangan aplikasi dan menjaga konsistensi data antar pengguna. Struktur data pada *Firebase Realtime Database* dirancang untuk menyimpan data lowongan, kriteria, pengguna, dan hasil analisis *CV*. Berikut adalah representasi JSON dari struktur database dan penjelasannya:

Gambar 4.9. Gambar Struktur Database berbentuk JSON

Penjelasan Struktur Database :

1. criteria: Menyimpan kriteria pekerjaan yang digunakan untuk analisis *CV*.
   - criteria_id: ID unik untuk setiap kriteria.
   - text: Deskripsi kriteria.

2. users: Menyimpan data pengguna dan informasi terkait lowongan pekerjaan yang mereka lamar.
   - user_id: ID unik untuk setiap pengguna.
   - data_user: Menyimpan informasi dasar pengguna seperti email, nama, dan URL gambar profil.
     - email: Email pengguna.
     - nama: Nama pengguna.
     - *Profile*ImageUrl: URL gambar profil pengguna.
   - lowongan: Menyimpan informasi lowongan pekerjaan yang dilamar oleh pengguna.
     - job_id: ID unik untuk setiap lowongan pekerjaan.
     - job_title: Judul pekerjaan.
     - job_criteria: Daftar ID kriteria yang terkait dengan pekerjaan.
     - *CV*_analysis: Menyimpan hasil analisis *CV* untuk pekerjaan tertentu.
       - *CV*_id: ID unik untuk setiap analisis *CV*.
       - job_id: ID pekerjaan yang terkait dengan analisis *CV*.
       - job_title: Judul pekerjaan yang terkait dengan analisis *CV*.
       - *CV_file*name: Nama *file CV*.
       - *CV*_path: Path *file CV* di perangkat pengguna.
       - *CV*_url: URL *file CV* di Firebase Storage.
       - main_relevance: Relevansi utama *CV* terhadap pekerjaan.
       - saran_relevance: Saran relevansi berdasarkan analisis *CV*.
       - *checkbox*_values: Daftar nilai relevansi untuk setiap kriteria.
         - description: Deskripsi kriteria.
         - value: Nilai relevansi kriteria.

- user_id: ID pengguna yang terkait dengan analisis *CV*.

Struktur database ini dirancang untuk menyimpan data pengguna, kriteria pekerjaan, dan analisis *CV* dengan cara yang terstruktur dan efisien di *Firebase Realtime Database*. Dengan struktur ini, aplikasi dapat dengan mudah mengakses dan mengelola data yang diperlukan untuk memberikan analisis *CV* yang relevan dan bermanfaat bagi pengguna. Struktur ini juga memungkinkan untuk pengembangan lebih lanjut, seperti menambahkan fitur baru atau memperluas skala aplikasi tanpa mengubah dasar dari struktur database yang sudah ada. Dengan menggunakan *Firebase Realtime Database*, aplikasi dapat memanfaatkan kemampuan real-time untuk memberikan pengalaman pengguna yang responsif dan interaktif.

4.1.3. Konfigurasi *Google AI* (Gemini)

Pada bagian ini, kita akan membahas langkah-langkah untuk mengonfigurasi Gemini dalam aplikasi *Flutter*. Gemini adalah antarmuka pemrograman aplikasi yang memungkinkan integrasi model bahasa generatif dari Google ke dalam aplikasi. Dalam konteks aplikasi ini, Gemini digunakan untuk menganalisis *CV* pelamar dan memberikan skor relevansi terhadap deskripsi pekerjaan yang dipilih.

a. Mengimpor Paket *Google AI*

Langkah pertama adalah mengimpor paket google_generative_ai yang menyediakan fungsi-fungsi untuk berinteraksi dengan *Gemini API*. Paket ini diimpor di *file InputScreen*.dart karena di sinilah proses analisis *CV* dimulai :

```
1  import 'package:google_generative_ai/google_generative_ai.dart';
```

Gambar 4.10. Import Paket *Google AI*

Dengan mengimpor paket ini, kita dapat mengakses kelas dan fungsi yang dibutuhkan untuk berkomunikasi dengan *Gemini API*.

b. Mendefinisikan API Key

API Key diperlukan untuk mengautentikasi permintaan ke *Gemini API*. API Key ini didapatkan dari https://aistudio.google.com/app/apikey dan disimpan dalam variabel apiKey :

```dart
final String apiKey = "YOUR_API_KEY_HERE";
```

Gambar 4.11. Deklarasi Variable API key

Gunakan metode penyimpanan yang aman, seperti environment variables atau secure storage yang disediakan oleh *platform*. Kode di atas hanya untuk ilustrasi.

c. Mengonfigurasi Model Generatif

Model generatif Gemini dikonfigurasi dengan parameter-parameter spesifik untuk analisis *CV* :

```dart
late final GenerativeModel? model;
  @override
  void initState() {
    super.initState();
    final User? user = _auth.currentUser;
    if (user != null) {
      usersKey = user.uid;
      final schema = Schema.object(
          description: 'Evaluation result for job applicant',
          properties: {
            'main_relevance': Schema.integer(
                description: 'Overall relevance score', nullable: false),
            'checkbox_values': Schema.array(
                description:
                    'List of relevance scores for each job description',
                items: Schema.object(properties: {
                  'description': Schema.string(
                      description: 'Job description', nullable: false),
                  'value' : Schema.integer(description: 'Relevance score for the job description',
                    nullable: false)
                }, requiredProperties: [
                  'description','value'
                ])),'saran_relevance': Schema.string(
                description: 'Advice paragraph for HR', nullable: false)
          },requiredProperties: ['main_relevance','checkbox_values','saran_relevance'
          ]);
      model = GenerativeModel(
        model: 'gemini-1.5-flash', apiKey: apiKey,
        safetySettings: [
          SafetySetting(HarmCategory.harassment, HarmBlockThreshold.none),
          SafetySetting(HarmCategory.hateSpeech, HarmBlockThreshold.none),
          SafetySetting(HarmCategory.sexuallyExplicit, HarmBlockThreshold.none),
          SafetySetting(HarmCategory.dangerousContent, HarmBlockThreshold.none),
        ],
        generationConfig: GenerationConfig(
          temperature: 0.95, topK: 64, topP: 0.95,
          maxOutputTokens: 8192, responseMimeType: 'application/json',
          responseSchema: schema,));
      fetchLowongan().then((data) {
        setState(() {
          lowonganList = data;
        });}).catchError((error) {print('Error fetching data: $error');
      });
    } else {
      Navigator.pushReplacement(context, MaterialPageRoute(builder: (context) => LoginScreen()),
      );
    }}
```

Gambar 4.12. Inisialisasi model di initState

Penjelasan Konfigurasi :

- model: Menentukan model generatif yang digunakan, dalam hal ini 'gemini-1.5-pro'.
- apiKey: API Key Anda untuk mengakses *Gemini API*.
- safetySettings: Mengatur pengaturan keamanan untuk memfilter konten yang berpotensi berbahaya atau tidak pantas.
- generationConfig: Mengonfigurasi parameter pembangkitan teks, seperti:
    - temperature: Mengontrol kreativitas output model (nilai lebih tinggi menghasilkan output yang lebih beragam).
    - topK dan topP: Mengontrol pemilihan kata berikutnya berdasarkan probabilitas.
    - maxOutputTokens: Jumlah maksimum token dalam output.
    - responseMimeType: Tipe MIME dari respons yang diharapkan, di sini disetel ke 'application/json'.
    - responseSchema: Skema JSON dari respons yang diharapkan untuk memvalidasi format output.

d. Membuat Prompt untuk Model

Fungsi combineChatSession membuat prompt yang akan dikirim ke *Gemini API*. Prompt ini berisi instruksi dan konteks untuk analisis *CV*. Definisikan fungsi untuk membuat prompt yang akan dikirim ke model:

```
combineChatSession(String jobTitle, List<String> jobDescription) {
  String prompt = """
  Saya ingin Anda bertindak sebagai seorang perekrut. Saya akan memberikan informasi
  tentang lowongan pekerjaan dan CV pelamar. Tugas Anda adalah menilai kecocokan
  pelamar dengan posisi $jobTitle berdasarkan deskripsi pekerjaan berikut:
  """;
  for (int i = 0; i < jobDescription.length; i++) {
    prompt += "- ${jobDescription[i]}\n";
  prompt += """
  Format JSON yang diharapkan:
  {
      "main_relevance": integer,
      "checkbox_values": [
          {"description": "Deskripsi pekerjaan 1", "value": integer},
          {"description": "Deskripsi pekerjaan 2", "value": integer},
          // ... nilai lainnya ...
      ],
      "saran_relevance": "Paragraf saran untuk HRD."
  }
  """;
  return prompt;
}
```

Gambar 4.13. Membuat Fungsi Prompt

Instruksi:

- Berikan nilai relevansi utama secara keseluruhan (main_relevance) dengan rentang 1 hingga 100.
- Berikan nilai relevansi (value) untuk setiap deskripsi pekerjaan dalam '*checkbox*_values'.
- Jika pengalaman, pendidikan, atau keterampilan tidak sesuai dengan deskripsi pekerjaan, berikan nilai 0.
- Setelah memberikan penilaian, berikan saran dalam bentuk paragraf untuk *HRD* dalam 'saran_relevance'.

Format JSON yang diharapkan ini menerima jobTitle dan jobDescription sebagai *Input* dan menghasilkan prompt yang terstruktur dalam format yang dipahami oleh *Gemini API*.

e. Mengirim Permintaan ke Multi Model

Setelah model dan prompt disiapkan, permintaan dikirim ke *Gemini API* beserta *file CV*. Gunakan model untuk menghasilkan konten berdasarkan prompt dan *file CV* yang dipilih:

```
String prompt = combineChatSession(jobTitle, jobDescription);
final fileCV = await _pickedFile!.readAsBytes();
final fileExtension = _pickedFile!.path.split('.').last;
final mimeType = (fileExtension == 'jpg' || fileExtension == 'jpeg' || fileExtension == 'png')
    ? 'image/jpeg'
    : 'application/pdf';
final filePart = DataPart(
  mimeType,
  fileCV,
);
final response = await model?.generateContent([
  Content.multi([TextPart(prompt), filePart])
]);
```

Gambar 4.14. Mengirim Permintaan ke Multi Model

Kode ini membaca *file CV*, menentukan tipe MIME-nya, membuat data part untuk *file*, dan mengirimkan permintaan ke model menggunakan generateContent. Metode Content.multi digunakan untuk mengirimkan teks (prompt) dan *file CV* secara bersamaan.

f. Memproses Respons Model

Setelah menerima respons dari model, data Respons dari *Gemini API* dalam format JSON di-parse untuk mendapatkan informasi yang dibutuhkan :

```dart
if (response?.text != null) {
  final responseData = jsonDecode(response!.text);

  // Mendapatkan skor relevansi utama
  int mainRelevance = responseData['main_relevance'];

  // Mendapatkan nilai relevansi untuk setiap deskripsi pekerjaan
  List<Map<String, dynamic>> checkboxValues = List<Map<String, dynamic>
>.from(
    responseData['checkbox_values'],
  );

  // Mendapatkan saran untuk HRD
  String saranRelevance = responseData['saran_relevance'];

  // Lakukan penyimpanan atau manipulasi data sesuai kebutuhan
Navigator.of(context).push(MaterialPageRoute(
  builder: (context) => ResultScreen(
    cvAnalysis: jobKey!,
    cv_url: downloadUrl,
    cv_path: cv_path,
    jobTitle: jobTitle,
    mainRelevance: mainRelevance,
    checkboxValues: checkboxValues,
    saranRelevance: saranRelevance,
  ),
));
}
```

Gambar 4.15. Parsing Respon Model

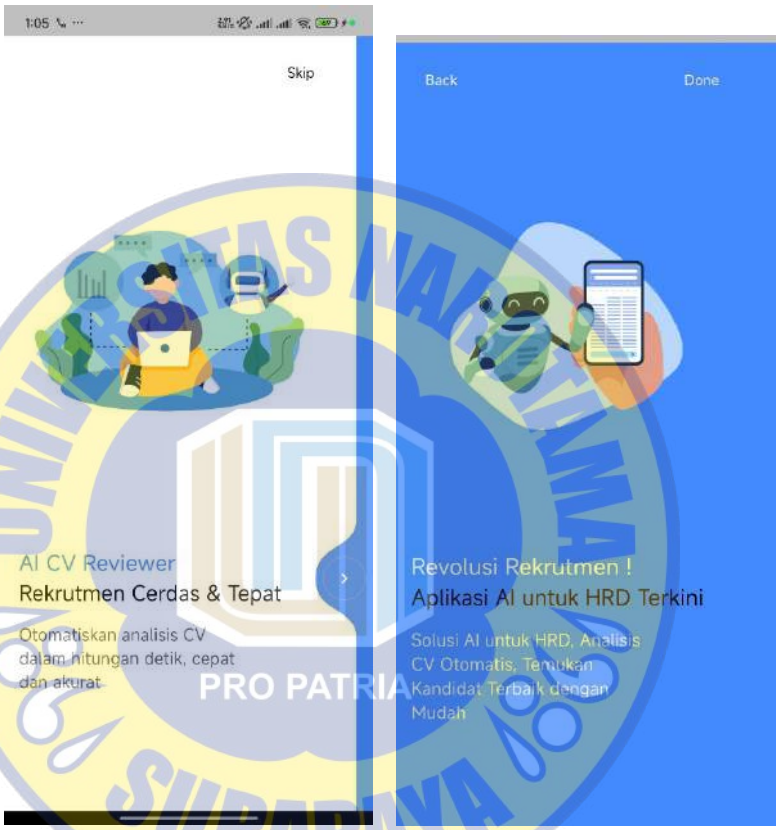
Kode ini mengekstrak main_relevance, *checkbox*_values, dan saran_relevance dari respons JSON. Data ini kemudian digunakan untuk ditampilkan pada Result*Screen*.

4.1.4. Implementasi Antarmuka Pengguna dan Hasil

Antarmuka pengguna (UI) aplikasi diimplementasikan menggunakan *framework Flutter*, mengikuti perancangan UI/UX yang

dijelaskan pada Bab III. *Flutter* dipilih karena kemampuannya dalam membangun UI yang modern, responsif, dan cross-*platform*. Widget-*widget Flutter* digunakan untuk menciptakan tampilan yang intuitif dan mudah dinavigasi. Berikut adalah *Screen*shot dan penjelasan masing-masing tampilan dalam aplikasi :

a. Tampilan Splash *Screen*

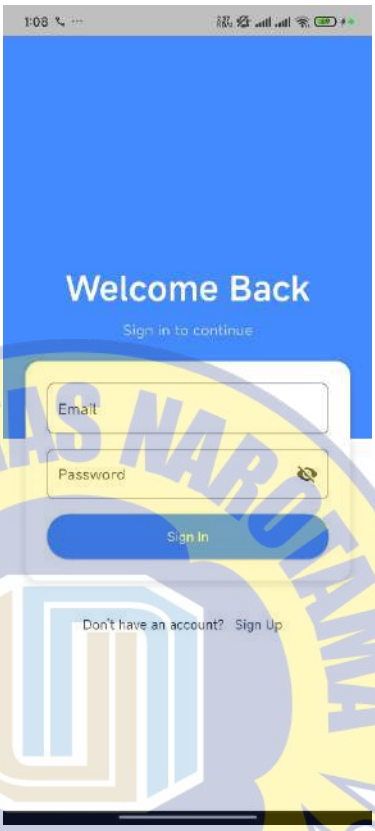


Gambar 4.16. *Screen*shot Tampilan Splash *Screen*

Tampilan splash *Screen* pada aplikasi ini memiliki desain modern dengan transisi animasi mulus seperti air yang digeser ke kiri, menciptakan pengalaman pengguna yang intuitif dan profesional. Halaman pertama menampilkan ilustrasi robot AI dengan latar biru, dilengkapi teks utama "Revolusi Rekrutmen!" yang memperkenalkan aplikasi sebagai solusi *HRD* berbasis AI untuk analisis *CV* otomatis. Halaman kedua beralih ke latar putih dengan ilustrasi pengguna laptop yang dikelilingi simbol AI, mempertegas fungsi utama aplikasi untuk rekrutmen cerdas dan cepat. Kombinasi desain visual yang menarik dan

efek animasi yang seamless memberikan kesan profesional sejak awal penggunaan.

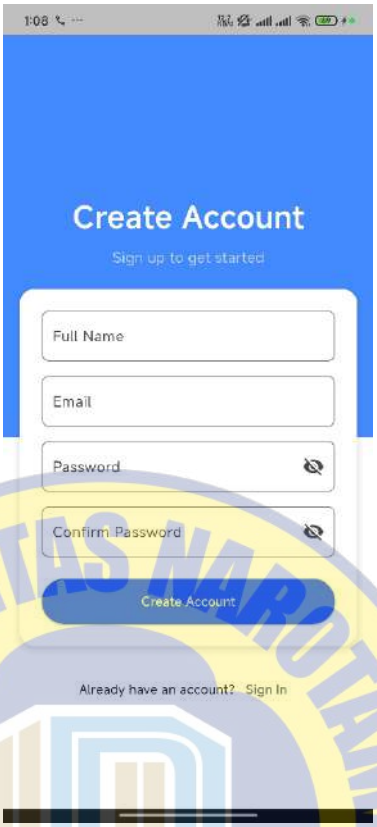
b. Tampilan Login *Screen*



Gambar 4.17. *Screen*shot Tampilan Login *Screen*

Tampilan login *Screen* merupakan gerbang masuk bagi *HRD* untuk mengakses aplikasi. Tampilan ini terdiri dari *Input field* untuk email dan password, serta tombol "Login". Integrasi dengan Firebase Authentication memastikan keamanan dan kevalidan proses login.

c. Tampilan Signup *Screen*

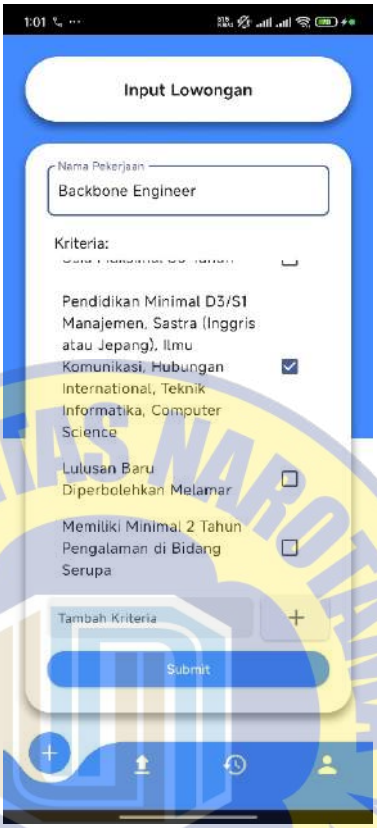


Gambar 4.18. *Screen*shot Tampilan Signup *Screen*

Tampilan Signup *Screen* dirancang khusus untuk memfasilitasi pendaftaran *HRD* baru ke dalam aplikasi. Formulir pendaftaran mencakup *field Input* untuk nama lengkap, email, dan password. Validasi *Input* diimplementasikan untuk memastikan data yang dimasukkan valid dan sesuai format yang diharapkan, misalnya, format email yang benar dan password yang memenuhi kriteria keamanan minimum. Setelah data yang dimasukkan lolos validasi, data tersebut dikirim ke Firebase Authentication untuk membuat akun pengguna baru. Proses pendaftaran yang mudah dan aman ini menjamin hanya *HRD* yang berwenang yang dapat mengakses dan menggunakan aplikasi. Tampilan ini juga terintegrasi dengan *Firebase Realtime Database* untuk menyimpan data pengguna yang baru terdaftar, sehingga data profil dapat diakses dan dikelola melalui aplikasi. Selain itu, tampilan Signup *Screen* didesain dengan mempertimbangkan

prinsip usability dan estetika, menciptakan pengalaman pendaftaran yang positif bagi *HRD* baru.

d. Tampilan *Input* Lowongan *Screen*



Gambar 4.19. *Screen*shot Tampilan *Input* Lowongan *Screen*

Tampilan "*Input* Lowongan *Screen*" merupakan fitur penting bagi *HRD* untuk mengelola dan menambahkan lowongan pekerjaan baru ke dalam sistem. Pada tampilan ini, *HRD* dapat memasukkan informasi detail lowongan pekerjaan, dimulai dengan "Nama Pekerjaan" yang di*Input*kan melalui *TextFormField*, sebuah *widget Flutter* yang menyediakan *Input field* untuk teks. Selain nama pekerjaan, *HRD* juga dapat menentukan kriteria yang dibutuhkan untuk lowongan tersebut. Pemilihan kriteria dilakukan dengan menggunakan *CheckboxListTile*, sebuah *widget Flutter* yang menggabungkan *checkbox* dengan teks deskriptif untuk setiap kriteria. Penggunaan *CheckboxListTile* memudahkan *HRD* untuk memilih beberapa kriteria sekaligus sesuai dengan kebutuhan lowongan pekerjaan. Kriteria-kriteria yang

ditampilkan diambil dari data yang tersimpan di *Firebase Realtime Database*, sehingga *HRD* dapat memilih dari daftar kriteria yang sudah tersedia atau menambahkan kriteria baru jika diperlukan. Setelah *HRD* selesai memasukkan nama pekerjaan dan memilih kriteria, data lowongan pekerjaan yang baru akan disimpan ke *Firebase Realtime Database* ketika tombol "Tambah Lowongan" ditekan. Tampilan "*Input* Lowongan *Screen*" dirancang dengan antarmuka yang intuitif dan mudah digunakan, sehingga *HRD* dapat dengan mudah dan efisien menambahkan lowongan pekerjaan baru ke dalam sistem.

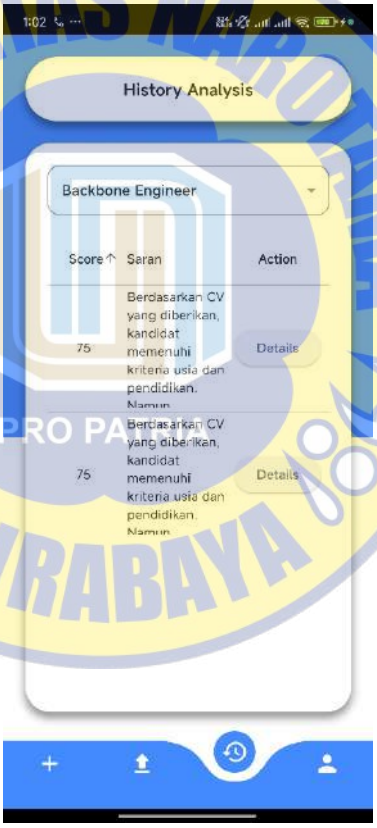
e. Tampilan *Submit CV Screen*

Gambar 4.20. *Screen*shot Tampilan *Submit CV Screen*

Tampilan "*Submit CV Screen*" merupakan tahapan krusial di mana *HRD* dapat mengunggah *CV* mereka untuk dievaluasi oleh sistem. Tampilan ini dirancang dengan alur yang sederhana dan mudah diikuti. Langkah pertama yang harus dilakukan oleh *HRD* adalah memilih lowongan pekerjaan yang ingin dilamar. Pemilihan ini dilakukan

melalui dropdown dengan daftar lowongan pekerjaan yang tersedia di database. Hal ini penting untuk memastikan bahwa *CV* yang diunggah akan dianalisis berdasarkan kriteria yang sesuai dengan lowongan pekerjaan yang dipilih. Setelah *HRD* memilih lowongan pekerjaan, langkah selanjutnya adalah meng-upload *file CV*. Proses pengunggahan dilakukan dengan menggunakan *FilePicker*, sebuah plugin *Flutter* yang memungkinkan pelamar untuk memilih *file CV* dari penyimpanan lokal perangkat. Aplikasi ini khusus mendukung format *file* PDF dengan ukuran minimum 0,3MB untuk memastikan kualitas *scan* dan kejelasan teks yang dibutuhkan oleh *Gemini API* untuk proses analisis.

f. Tampilan *History Screen*



Gambar 4.21. *Screen*shot Tampilan *History Screen*

Tampilan "*History Screen*" memberikan *HRD* akses ke riwayat lengkap analisis *CV* yang pernah dilakukan. Tampilan ini berfungsi sebagai arsip dan memudahkan *HRD*, dalam hal ini *HRD*, untuk memantau dan mereview hasil penilaian kandidat dari waktu ke waktu.

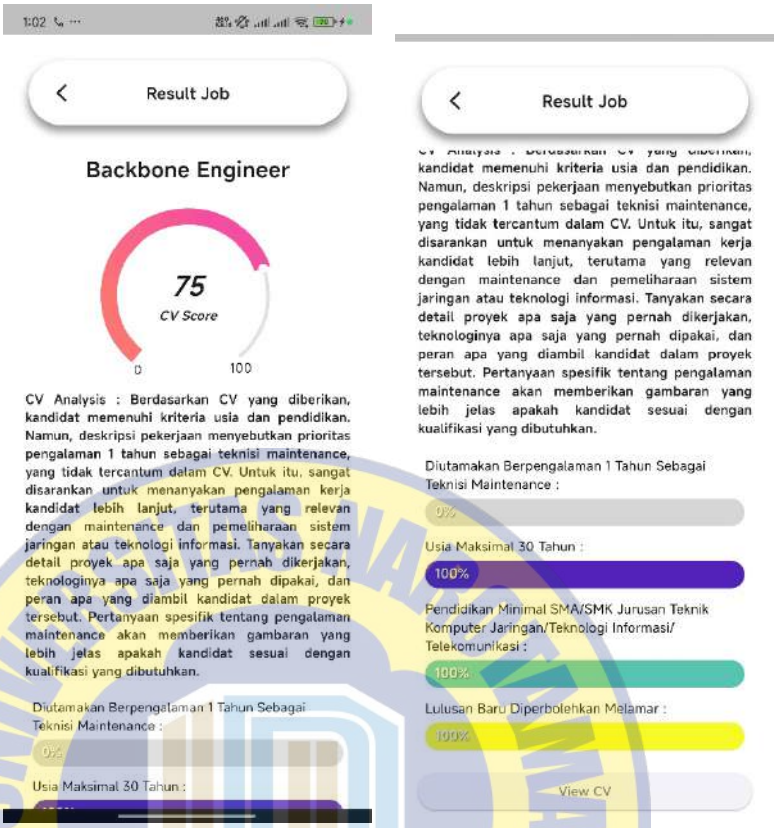
"*History Screen*" menampilkan data dalam bentuk tabel yang terstruktur dan mudah dibaca. Setiap baris pada tabel merepresentasikan satu proses analisis *CV*, Dengan memilih lowongan spesifik, tabel akan menampilkan hanya riwayat analisis *CV* yang terkait dengan lowongan tersebut. Untuk melihat detail dari setiap analisis, pengguna dapat menekan baris yang bersesuaian pada tabel. Aksi ini akan membuka tampilan "Result Analysis *Screen*" yang menampilkan informasi lengkap mengenai hasil analisis *CV* yang dipilih, termasuk skor relevansi untuk setiap kriteria, saran dari *Gemini API*, dan pratinjau *CV* kandidat.

g. Tampilan *Profile Screen*

Gambar 4.22. *Screen*shot Tampilan *History Screen*

Layar profil pengguna di mana informasi pengguna ditampilkan dan dapat diedit. Mengambil data dari *Firebase Realtime Database* dan menampilkan gambar profil, nama, dan email.

h. Tampilan *Result Screen*



Gambar 4.23. *Screen*shot Tampilan *Result Screen*

Tampilan "Result Analysis *Screen*" merupakan puncak dari proses review *CV* menggunakan aplikasi. Di sinilah hasil analisis *CV* oleh *Gemini API* ditampilkan secara lengkap dan terstruktur. Tampilan ini menyajikan informasi kunci yang dibutuhkan *HRD* untuk mengevaluasi kesesuaian seorang kandidat dengan lowongan pekerjaan tertentu. Elemen utama pada tampilan ini adalah "Judul Pekerjaan" yang dianalisis, memberikan konteks yang jelas mengenai posisi yang dilamar. Selanjutnya, "Relevansi Utama" ditampilkan secara prominen, biasanya dalam bentuk persentase dan disertai *Visualisasi* grafik menggunakan *SfLinearGauge*. *Visualisasi* ini memberikan gambaran cepat mengenai skor kesesuaian kandidat secara keseluruhan. Selain skor keseluruhan, tampilan ini juga merinci penilaian untuk setiap kriteria yang ditetapkan untuk lowongan pekerjaan tersebut. Setiap kriteria ditampilkan beserta skor atau tingkat kesesuaiannya,

memudahkan *HRD* untuk memahami kekuatan dan kelemahan kandidat berdasarkan persyaratan spesifik dari lowongan tersebut. Salah satu elemen penting lainnya adalah "Saran" dari analisis yang dihasilkan oleh *Gemini API*. Saran ini berupa teks yang dihasilkan oleh model AI berdasarkan analisis *CV* dan deskripsi pekerjaan. Saran ini dapat berisi poin-poin penting yang perlu diperhatikan oleh *HRD*, rekomendasi untuk tahap selanjutnya dalam proses rekrutmen, atau pertanyaan wawancara yang disarankan. Dengan menyajikan informasi yang lengkap dan terstruktur, "Result Analysis *Screen*" memungkinkan *HRD* untuk mengambil keputusan yang lebih informasi dan objektif dalam proses penyaringan *CV*. Tampilan ini juga meningkatkan efisiensi karena *HRD* tidak perlu lagi membaca dan menganalisis *CV* secara manual.

4.2. Analisis Kinerja Aplikasi

4.2.1. Data Uji

Data uji yang digunakan dalam pengujian ini terdiri dari sejumlah *CV* dengan berbagai latar belakang keahlian dan pengalaman kerja. Data uji terdiri dari 4 *CV* dan 3 lowongan pekerjaan dengan kriteria yang berbeda. *CV*-*CV* ini dipilih untuk merepresentasikan variasi keahlian dan pengalaman pelamar. Lowongan pekerjaan dipilih untuk merepresentasikan berbagai jenis posisi dan persyaratan.:

Tabel 4.5. Data Uji *CV*

CV ID	Keterangan
*CV*001	Abdullah adalah lulusan Sistem Informasi dari UIN Sunan Ampel Surabaya dan telah bekerja sebagai penyelenggara Pemilihan Bupati di KPU Tuban, pendidik di Yayasan Miftahul Falah, tenaga pemasaran di Morning Glory Enterprise, mitra BPS dalam sensus penduduk, dan terakhir sebagai koordinator area di divisi penjualan PT Supra Primatama Nusantara. *CV*-nya juga mencantumkan penelitian yang dilakukannya di UIN Sunan Ampel Surabaya tentang pemetaan daerah rawan bencana banjir di Kabupaten Tuban. Keahliannya meliputi Microsoft Office dan internet, serta kemampuan percakapan. Ia menunjukkan minat yang beragam dalam pemerintahan, pendidikan, penjualan, dan

	teknologi.
*CV*002	Lucky adalah lulusan Universitas Airlangga Surabaya dengan gelar S1 Ilmu Informasi dan Perpustakaan. *CV*-nya menunjukkan pengalamannya sebagai pustakawan, project leader pembuatan perpustakaan, dan sales/marketing freelance. Ia juga mengelola toko online dan menekankan kemampuannya dalam Microsoft Office, MySQL, desain, dan pemasaran. Lucky memiliki pengalaman bekerja di Universitas Dr. Soetomo dan berbagai perusahaan lainnya. Ia menggambarkan dirinya sebagai pekerja keras, jujur, dan disiplin yang tertarik dengan hal-hal baru.
*CV*003	Marwah adalah lulusan Universitas Jember dengan gelar di bidang Teknik Elektro. *CV*-nya menekankan pengalamannya dalam manajemen telekomunikasi, pemasaran, dan pengoperasian mesin. Ia juga aktif dalam organisasi mahasiswa, seperti Resimen Mahasiswa dan PASKIBRA. Marwah mencantumkan pelatihan yang ia ikuti, seperti "Fiber to the Home" dan kursus kader eksekutif. Ia juga menekankan keahliannya dalam komputer, termasuk Microsoft Office, Adobe Photoshop, Corel Draw, dan AutoCAD.
*CV*004	Nur Laily adalah Lulusan Universitas Wijaya Kusuma Surabaya dengan gelar S1 Ekonomi Pembangunan, Keuangan dan Perbankan. *CV*-nya berfokus pada pengalamannya di administrasi dan layanan pelanggan, termasuk peran di PT. Bangun Indopralon, PT. Safari Global Perkasa, dan PT. Permodalan Nasional Madani Persero. Ia juga aktif dalam berbagai organisasi mahasiswa, seperti UKM Lakapanza dan kegiatan duta anti narkoba. *CV*-nya menyoroti keahliannya dalam Microsoft Office, presentasi, dan komunikasi verbal. Ia mencari posisi administrasi yang menawarkan peluang pengembangan.

Keempat *CV* ini dipilih karena mewakili berbagai bidang keahlian yang umum dicari dalam dunia kerja, sehingga dapat memberikan gambaran yang representatif tentang kinerja aplikasi dalam menganalisis *CV* dari berbagai latar belakang.

Tabel 4.6. Data Uji Lowongan Pekerjaan

Lowongan ID	**Jenis Lowongan**	**Kriteria (Singkat)**
L001	Store Specialist	• Pendidikan Minimal SMA/SMK atau Diploma Jurusan Manajemen/Akuntansi/Administrasi • Terbiasa untuk Bekerja dengan Sistem (diutamakan familiar dengan SAP) • Perempuan, Usia 21-30 Tahun

		• Freshgraduate Dipersilahkan Melamar • Berpengalaman Sebagai Administrasi dan/atau Customer Service Selama 1 Tahun
L002	Area Coordinator	• Memiliki Komunikasi yang Baik dan Kemampuan Persuasif • Usia Maksimal 35 Tahun • Memiliki Pengalaman Dalam Penjualan Produk (Lulusan Baru Diperkenankan Melamar) • Terbiasa Bekerja dengan Target • Pendidikan Minimal D3 Jurusan Manajemen Pemasaran/Ekonomi Manajemen/Sistem Informasi/Ilmu Komunikasi
L003	Backbone Engineer	• Diutamakan Berpengalaman 1 Tahun Sebagai Teknisi Maintenance • Usia Maksimal 30 Tahun • Pendidikan Minimal SMA/SMK Jurusan Teknik Komputer Jaringan/Teknologi Informasi/Telekomunikasi • Lulusan Baru Diperbolehkan Melamar

Lowongan pekerjaan ini dipilih untuk merepresentasikan berbagai jenis kriteria dan persyaratan yang umum ditemukan dalam proses rekrutmen. Hal ini memungkinkan pengujian yang lebih komprehensif terhadap kemampuan aplikasi dalam menganalisis kesesuaian *CV* dengan berbagai jenis lowongan.

4.2.2. Pengujian Ketepatan

Pengujian ini dilakukan dengan menguji 4 *CV* terhadap 3 lowongan yang menurut saya sesuai, menghasilkan total 4 pengujian. Output rekomendasi aplikasi dibandingkan dengan penilaian manual oleh leader saya, menggunakan *Main Relevance*, metrik *MAE* dan *RMSE*. Pengujian ketepatan aplikasi dilakukan dengan membandingkan output rekomendasi sistem dengan penilaian manual oleh leader.

Tiga metrik yang digunakan adalah *Main Relevance*, *Mean Absolute Error* (*MAE*), dan *Root Mean Squared Error* (*RMSE*), Berikut adalah penjelasan cara perhitungan metrik :

- *Main Relevance* : Metrik ini mengukur kesesuaian keseluruhan antara *CV* dan lowongan pekerjaan. Nilai ini dihitung dengan menjumlahkan nilai aktual dari setiap kriteria yang relevan, kemudian membaginya dengan jumlah total kriteria yang dinilai. Semakin tinggi nilai *Main Relevance* (mendekati 100%), semakin besar kesesuaian *CV* tersebut dengan kualifikasi yang dicari oleh perusahaan. Sebaliknya, nilai yang rendah menunjukkan adanya ketidakcocokan yang lebih signifikan, Rumus perhitungan *Main Relevance* :
  Main Relevance = Jumlah Nilai / Banyak Kriteria
- *Mean Absolute Error* (*MAE*): *MAE* mengukur rata-rata besarnya kesalahan antara nilai yang diharapkan (berdasarkan persyaratan lowongan) dan nilai aktual (yang terdapat dalam *CV*) untuk setiap kriteria. Perbedaan antara nilai yang diharapkan dan nilai aktual dihitung secara absolut (diabaikan tanda positif/negatifnya) dan kemudian dirata-ratakan, Semakin rendah nilai *MAE*, semakin kecil rata-rata kesalahan, yang berarti semakin akurat rekomendasi sistem. dengan rumus:
  $MAE = (1/n) * \Sigma|yi - xi|$
  Penjelasan :
  n: Jumlah total data poin atau prediksi.
  yi: Nilai aktual (atau yang diamati) untuk data poin ke-i.
  xi: Nilai aktual yang ditemukan dalam *CV* untuk kriteria ke-i.
  Σ: Simbol sigma, yang berarti penjumlahan. Jadi, kita menjumlahkan selisih absolut untuk semua data poin.
- *Root Mean Squared Error* (*RMSE*): *RMSE RMSE* juga mengukur rata-rata kesalahan antara nilai yang diharapkan dan nilai aktual, tetapi memberikan penalti yang lebih besar untuk kesalahan yang besar. Hal ini dilakukan dengan mengkuadratkan selisih antara nilai yang diharapkan dan nilai aktual sebelum dirata-ratakan dan diakarkan, Sama seperti *MAE*, nilai *RMSE* yang lebih rendah menunjukkan kinerja sistem yang lebih baik. Namun, karena *RMSE* lebih sensitif terhadap kesalahan besar, metrik ini berguna ketika akurasi pada kriteria tertentu sangat penting, dengan rumus:
  $RMSE = \sqrt{[(1/n) * \Sigma(yi - xi)^2]}$.

Penjelasan :

n: Jumlah total data poin atau prediksi.

yi: Nilai aktual (atau yang diamati) untuk data poin ke-i.

xi: Nilai aktual yang ditemukan dalam *CV* untuk kriteria ke-i.

Σ: Simbol sigma, yang berarti penjumlahan. Jadi, kita menjumlahkan selisih absolut untuk semua data poin.

Berikut tabel hasil pengujian ketepatan aplikasi:

Tabel 4.7. Pengujian Ketepatan

CV ID	Lowongan ID	Output yang Diharapkan	Output Aktual	*Main Relevance*	*Mean Absolute Error* (%)	*Root Mean Squared Error* (%)
CV 001	L002	• Memiliki Komunikasi yang Baik dan Kemampuan Persuasif (≥60). • Usia Maksimal 35 Tahun (≥80). • Memiliki Pengalaman Dalam Penjualan Produk (Lulusan Baru Diperkenankan Melamar) (≥40). • Terbiasa Bekerja dengan Target (≥40). • Pendidikan Minimal D3 Jurusan	• Memiliki Komunikasi yang Baik dan Kemampuan Persuasif (70). • Usia Maksimal 35 Tahun (100). • Memiliki Pengalaman Dalam Penjualan Produk (Lulusan Baru Diperkenankan Melamar) (80). • Terbiasa Bekerja dengan Target (70). • Pendidikan Minimal D3 Jurusan Manajemen Pemasaran/Eko	74	28.00	30.33

		Manajemen Pemasaran/Ekonomi Manajemen/Sistem Informasi/Ilmu Komunikasi (≤10).	nomi Manajemen/Sistem Informasi/Ilmu Komunikasi (50)			
CV 002	L002	• Memiliki Komunikasi yang Baik dan Kemampuan Persuasif (≥70). • Usia Maksimal 35 Tahun (≥80). • Memiliki Pengalaman Dalam Penjualan Produk (Lulusan Baru Diperkenankan Melamar) (≥60). • Terbiasa Bekerja dengan Target (≥50). • 5. Pendidikan Minimal D3 Jurusan Manajemen Pemasaran/Ekonomi Manajemen/Sistem Informasi/Ilmu Komunikasi (≤10).	• Memiliki Komunikasi yang Baik dan Kemampuan Persuasif (70). • Usia Maksimal 35 Tahun (100). • Memiliki Pengalaman Dalam Penjualan Produk (Lulusan Baru Diperkenankan Melamar) (80). • Terbiasa Bekerja dengan Target (70). • Pendidikan Minimal D3 Jurusan Manajemen Pemasaran/Ekonomi Manajemen/Sistem Informasi/Ilmu Komunikasi (0)	64	14.00	16.12
CV 003	L003	• Diutamakan Berpengalaman 1 Tahun Sebagai Teknisi Maintenance (≤20). • Usia Maksimal	• Diutamakan Berpengalaman 1 Tahun Sebagai Teknisi Maintenance (0). • Usia Maksimal	75	27.50	30.41

		30 Tahun (≥80). • Pendidikan Minimal SMA/SMK Jurusan Teknik Komputer Jaringan/Teknologi Informasi/Telekomunikasi (≥50). • Lulusan Baru Diperbolehkan Melamar (≥80).	30 Tahun (100). • Pendidikan Minimal SMA/SMK Jurusan Teknik Komputer Jaringan/Teknologi Informasi/Telekomunikasi (100). • Lulusan Baru Diperbolehkan Melamar (100)			
*CV*004	L001	• Pendidikan Minimal SMA/SMK atau Diploma Jurusan Manajemen/Akuntansi/Administrasi (≤20). • Terbiasa untuk Bekerja dengan Sistem (diutamakan familiar dengan SAP) (≤40). • Perempuan, Usia 21-30 Tahun (≥80). • Freshgraduate Dipersilahkan Melamar (≥90). • Berpengalaman Sebagai Administrasi dan/atau Customer Service Selama 1 Tahun (≤10).	• Pendidikan Minimal SMA/SMK atau Diploma Jurusan Manajemen/Akuntansi/Administrasi (80). • Terbiasa untuk Bekerja dengan Sistem (diutamakan familiar dengan SAP) (30). • Perempuan, Usia 21-30 Tahun (100). • Freshgraduate Dipersilahkan Melamar (100). • Berpengalaman Sebagai Administrasi dan/atau Customer Service Selama 1 Tahun (70)	70	32.00	39.50
Rata-rata				70.75	25.38	29.09

Tabel 4.7. menyajikan hasil pengujian ketepatan aplikasi. *Main Relevance* dihitung berdasarkan persentase kriteria yang terpenuhi. Setiap kriteria yang terpenuhi diberi nilai 100%, dan yang tidak terpenuhi diberi nilai 0%. Skor akhir (*Main Relevance*) adalah rata-rata dari nilai semua kriteria, Metrik *Mean Absolute Error* (*MAE*), dan *Root Mean Squared Error* (*RMSE*) digunakan untuk mengukur seberapa akurat aplikasi dalam merekomendasikan kandidat yang sesuai untuk setiap lowongan.

Dapat disimpulkan bahwa terdapat rata-rata kesalahan Metrik *Mean Absolute Error* (*MAE*) sebesar 25.38% dan rata-rata kesalahan *Root Mean Squared Error* (*RMSE*) sebesar 29.09% antara output sistem dan penilaian manual. Meskipun *Main Relevance* menunjukkan kesesuaian rata-rata sebesar 70.75%, tingkat kesalahan yang relatif tinggi ini mengindikasikan perlunya evaluasi dan perbaikan lebih lanjut pada algoritma atau model yang digunakan oleh aplikasi. Analisis lebih lanjut diperlukan untuk memahami sumber kesalahan dan meningkatkan akurasi rekomendasi.

4.2.3. Pengujian Kecepatan

Bertujuan untuk mengukur waktu respons aplikasi dalam memproses *CV*. Waktu respons actual dengan koneksi internet yang stabil.

Tabel 4.8. Pengujian Kecepatan

No	*CV* ID	Lowongan ID	Ukuran *File CV* (MB)	Waktu Pemrosesan (ms)
1	*CV*001	L002	0.330	5528
2	*CV*002	L002	0.285	5011
3	*CV*003	L003	0.768	6503
4	*CV*004	L001	0.157	5195
Rata-rata			0.385	5559.25

Tabel 4.8. ini menunjukkan waktu pemrosesan yang dibutuhkan aplikasi untuk setiap kombinasi *CV* dan lowongan pekerjaan. Data ini akan digunakan untuk mengevaluasi kecepatan aplikasi menggunakan internet.