BAB 3
PELAKSANAAN KERJA MAGANG

3.1 Kedudukan dan Koordinasi

Pelaksanaan kegiatan magang di PT Bank Central Asia Tbk berada di bawah naungan divisi Group Strategic Information Technology (GSIT), tepatnya pada grup Data & IT Management Office, biro Data Management C (DTM C), dan tim Artificial Intelligence (AI). Biro DTM C dipimpin oleh Kepala Biro, Bapak Adhitya Bhaswara. Anggota biro DTM C tersebar di dua lokasi kantor yang berbeda, yaitu Menara BCA di Jakarta Pusat dan Wisma BCA Foresta di BSD.

Biro DTM C memiliki 43 anggota yang terbagi ke dalam tiga tim, yakni tim AI, tim ML, dan tim SQI. Tim AI terdiri dari 11 karyawan tetap dan 8 karyawan magang, yang mencakup jabatan Team Lead Data Scientist, Data Scientist Staff, serta Data Scientist Intern.

Kegiatan magang pada tim AI dilaksanakan di bawah bimbingan mentor magang, yaitu Ibu Salma Safira Ramadhanti selaku Data Scientist. Koordinasi proyek tim AI dipimpin oleh Bapak Timotius Nico dan Bapak Frederick Harison Wijaya selaku Team Lead Data Scientist. Proses koordinasi dengan mentor dan pemimpin tim umumnya dilakukan secara luring apabila seluruh anggota tim hadir di Menara BCA. Namun, apabila terdapat anggota tim yang sedang tidak berada di Menara BCA atau sedang menjalani skema *work from home* (WFH), maka koordinasi dilakukan secara daring menggunakan platform Microsoft Teams. Pola koordinasi campuran ini mendukung fleksibilitas kerja sekaligus menjaga efektivitas komunikasi dan kolaborasi antaranggota tim.

3.2 Tugas yang Dilakukan

Selama pelaksanaan kerja magang di PT Bank Central Asia Tbk, tim Artificial Intelligence (AI) di bawah naungan Group Strategic Information Technology (GSIT), tanggung jawab utama berupa pengembangan *frontend* untuk beberapa model *Natural Language Processing* (NLP) dan *rule-based masking*. Tugas-tugas yang dilakukan meliputi:

1. **Pengembangan *Frontend* untuk Model *Named Entity Recognition* (NER), *Part-of-Speech* (POS) Tagging, *Personal Identifiable Information Masking***

(PII):, Bertanggung jawab dalam merancang dan mengimplementasikan antarmuka pengguna (*User Interface*) berbasis (textitStreamlit) untuk model (*NER*), (*Part-of-Speech*), (*Personal Identifiable Information Masking*). Dirancang dengan antarmuka pengguna berbasis Streamlit, sistem ini menyediakan Homepage yang intuitif. Dari sana, pengguna dapat dengan mudah memilih halaman yang berbeda untuk menginput teks atau file untuk menampilkan entitas entitas yang dikenali oleh model (*NER*), memvisualisasikan hasil *tagging* dari model (*Part-of-Speech*) sebagai bagian ucapan (POS tagging) dari teks yang dimasukkan pengguna , atau menyamarkan kata yang mengandung (*Personal Identifiable Information* melalui (*Personal Identifiable Information Masking*).

2. **Integrasi *Frontend* dengan *Backend* Model:** Melakukan integrasi antara antarmuka *frontend* berbasis Streamlit dengan layanan *backend* yang mengekspos model NER, POS tagging, dan PII masking. Integrasi ini melibatkan pengiriman parameter dari *frontend* ke *backend* dan menampilkan hasil pemrosesan model kembali ke *frontend*. Layanan *backend* diimplementasikan menggunakan *Docker container*.

3. **Deployment Aplikasi *Frontend* dan *Backend* menggunakan Docker:** Mengelola dan melakukan *deployment* aplikasi *frontend* (Streamlit) dan *backend* (layanan model) ke dalam lingkungan *Docker container* untuk memastikan konsistensi dan kemudahan dalam pengembangan serta produksi.

Tugas-tugas ini berfokus pada penyediaan visualisasi dan interaksi yang intuitif bagi pengguna untuk berinteraksi dengan model-model AI yang dikembangkan oleh tim.

3.3 Uraian Pelaksanaan Magang

Pelaksanaan kerja magang diuraikan secara rinci pada sub-bagian berikut, yang berfokus pada tiga tugas utama yaitu pengembangan *frontend* untuk model NLP dan *rule-based masking*, integrasi *frontend* dengan *backend*, serta *deployment* aplikasi menggunakan Docker. Periode kerja magang berlangsung selama 16 minggu dengan rincian pekerjaan mingguan seperti yang tercantum pada Tabel 3.1.

Tabel 3.1. Pekerjaan yang dilakukan tiap minggu selama pelaksanaan kerja magang

Minggu Ke-	**Pekerjaan yang dilakukan**
1	Pengenalan lingkungan kerja, pengambilan alat, briefing alur sistem, pengerjaan e-learning, dan mengikuti *Town Hall*.
2	Eksplorasi *server*, *Docker*, dan kerangka kerja *Streamlit* untuk *frontend*. Pembuatan *Docker container* awal.
3	Eksplorasi *environment* (*Jupyter*, *Mobaxterm*), briefing proyek, mulai setup awal *frontend chatbot Streamlit*.
4	Menjalankan aplikasi *Streamlit chatbot* pertama kali, pengujian *API*, dan meeting bersama *user*. Memahami struktur kode *NER (HERA)*.
5	Migrasi awal dari *Gradio* ke *Streamlit*, *handover* resmi proyek *NER*, pembuatan *Docker container* untuk *NER*.
6	Fokus migrasi *frontend* ke *Streamlit*, penyelesaian masalah *dependency Docker*, optimasi ukuran *image Docker*.
7	Finalisasi *Dockerfile* dan *requirements.txt*, penambahan komponen UI seperti *modal*, *background*, contoh input, dan penghapusan *container* lama.
8	Penambahan dan revisi desain UI, *disclaimer*, tema visual, dan *troubleshooting regex rule-based* untuk *PII masking*.
9	*Troubleshooting API*, integrasi *NER* ke *ORION*, *tracking user visit*, modifikasi akses melalui *query params*.
10	Implementasi visualisasi *Entity Detection* dan *Subject Detection*, *feedback* dari tim, presentasi progres UI ke *Team Lead NLP*.
11	Pengubahan akses *NER*, validasi akses lewat *query*, pengerjaan *technical documentation*.
12	Task tambahan untuk halaman *AskSekre* berbasis *Regex*, konfigurasi *proxy DLP OCR*, *bug fixing* dan *stress test* untuk *API HERA*.
13	Revisi *logo HERA*, analisis *response time*, pengalihan dari *GPU* ke *CPU* untuk *inference*.
14	Kategorisasi *prompt log GAIA chatbot*, integrasi *MCP* ke *GAIA Gateway*, *stress test* akhir *API*.
15	Eksplorasi *MLflow*, *push model* ke *MLflow*, eksplorasi *GAIA Gateway* dan integrasi *MCP client-server*.
16	Eksplorasi dan implementasi *tools MCP*, pengujian *client alternatif*, penyederhanaan *chatbot structure*, dan penggunaan *agentic tools*.

3.3.1 Pengembangan *Frontend* berbasis *Streamlit* untuk Model NLP dan *Rule-based Masking*

Tugas utama pertama dalam pelaksanaan magang adalah mengembangkan antarmuka pengguna berbasis Streamlit untuk tiga model utama: *Named Entity Recognition* (NER), *Part-of-Speech* (POS) Tagging, dan *Personal Identifiable Information* (PII) Masking.

Streamlit adalah kerangka kerja open-source berbasis Python yang dirancang khusus untuk membangun antarmuka web secara cepat dan interaktif, terutama untuk keperluan data science dan machine learning. Dengan sintaksis yang sederhana, Streamlit memungkinkan integrasi langsung antara logika pemrosesan data dan tampilan antarmuka, sehingga sangat cocok untuk menampilkan hasil dari model NLP secara langsung kepada pengguna.

Pengembangan ini bertujuan untuk menyediakan interface yang intuitif dan mudah digunakan bagi pengguna untuk berinteraksi dengan model-model AI yang telah dikembangkan oleh tim.

Proses pengembangan dimulai dengan studi mendalam terhadap karakteristik dan kebutuhan masing-masing model. Model NER memerlukan visualisasi yang jelas untuk menampilkan entitas-entitas yang terdeteksi beserta kategorinya. Model POS Tagging membutuhkan representasi visual untuk menunjukkan bagian-bagian ucapan dalam teks. Sedangkan model PII Masking memerlukan interface yang dapat menampilkan teks asli dan teks yang telah disamarkan secara bersamaan.

Implementasi dilakukan dengan pendekatan modular, dimana setiap model memiliki halaman terpisah namun mengikuti pola desain yang konsisten. Hal ini memudahkan maintenance dan pengembangan lebih lanjut. Setiap modul dirancang dengan dua mode input utama: input teks langsung dan upload file, memberikan fleksibilitas bagi pengguna dengan kebutuhan yang berbeda-beda.

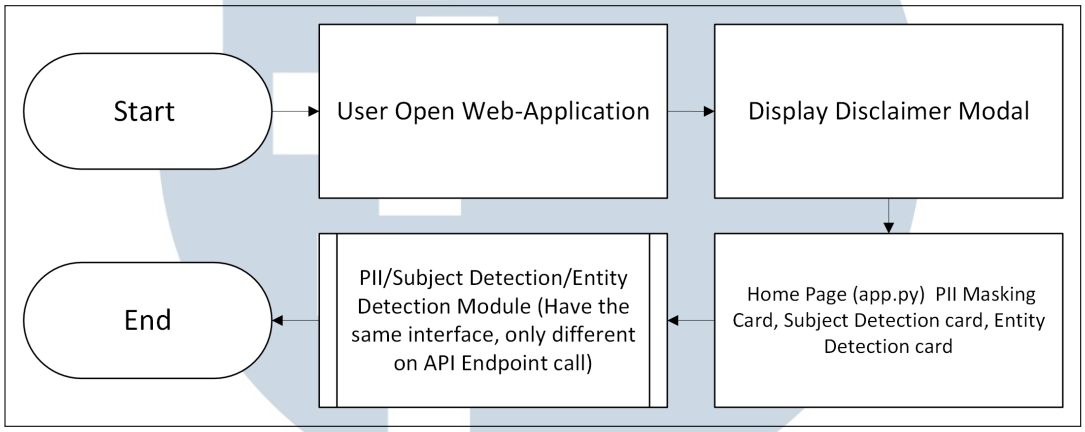
Tantangan utama dalam pengembangan ini adalah merancang visualisasi yang efektif untuk output masing-masing model. Untuk NER dan POS Tagging, digunakan library *annotated_text* yang memungkinkan highlighting dan labeling teks secara real-time. Untuk PII Masking, dikembangkan tampilan dual-pane yang menampilkan teks asli dan teks yang telah dimasking secara berdampingan.

Antarmuka pengguna aplikasi HEERA dirancang menggunakan Streamlit untuk memberikan pengalaman interaktif yang intuitif bagi pengguna dalam berinteraksi dengan model-model NLP. Perancangan didasarkan pada flowchart

sistem yang menggambarkan alur interaksi pengguna dengan aplikasi.

A Flowchart Sistem Aplikasi HEERA

Alur kerja aplikasi HEERA dapat digambarkan dalam flowchart utama yang menunjukkan navigasi pengguna dari halaman utama ke berbagai modul yang tersedia.



Gambar 3.1. Flowchart Utama Aplikasi HEERA

Flowchart utama pada Gambar 3.1 menggambarkan bahwa pengguna pertama kali akan melihat *disclaimer modal* saat membuka aplikasi. *Disclaimer modal* ini berfungsi sebagai langkah awal untuk memastikan bahwa pengguna memahami kebijakan privasi dan tanggung jawab penggunaan data, terutama karena aplikasi HEERA berkaitan erat dengan entitas pribadi dan informasi sensitif.

Setelah pengguna menyetujui *disclaimer* tersebut, mereka akan diarahkan ke halaman utama aplikasi. Halaman utama ini menampilkan tiga pilihan modul utama dalam bentuk kartu (*card layout*), yaitu modul untuk *PII Masking*, *Subject Detection*, dan *Entity Detection*. Ketiga modul ini ditampilkan secara paralel untuk mempermudah navigasi dan memberikan fleksibilitas kepada pengguna dalam memilih jenis pemrosesan teks yang diinginkan.

Setiap modul memiliki *interface* yang konsisten dalam hal tata letak dan pengalaman pengguna (*user experience*), namun di balik keseragaman antarmuka tersebut, terdapat perbedaan pada endpoint API yang digunakan, serta algoritma pemrosesan yang mendasarinya. Misalnya, modul *PII Masking* berfokus pada deteksi dan penyamaran informasi identitas pribadi menggunakan teknik berbasis aturan dan model NER, sedangkan *Entity Detection* menggunakan pendekatan *Named Entity Recognition* untuk menandai entitas umum seperti lokasi, organisasi,

dan nama orang.

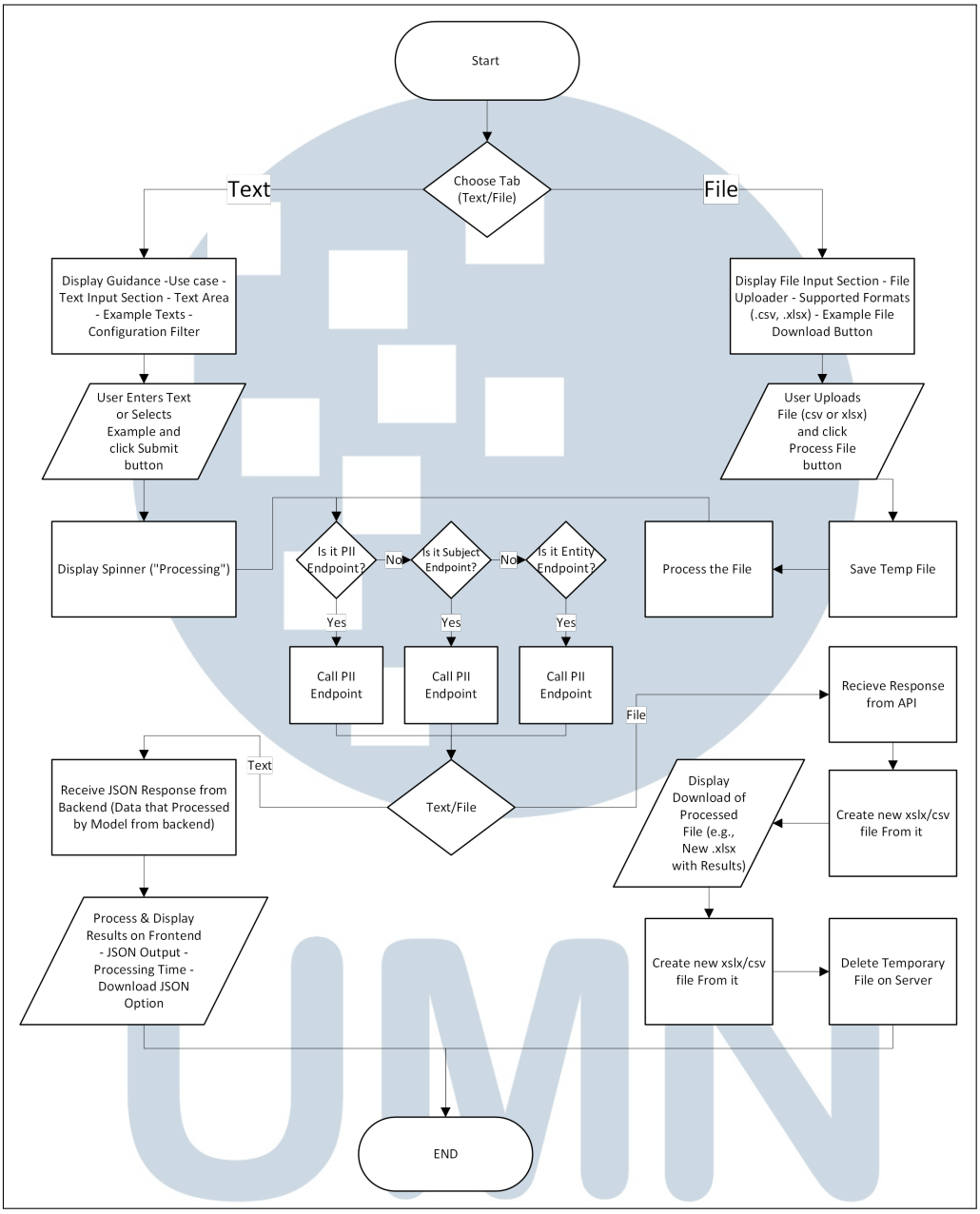
Flowchart ini tidak hanya menunjukkan urutan langkah navigasi, tetapi juga mencerminkan pendekatan sistematis dalam perancangan pengalaman pengguna yang intuitif, aman, dan modular. Hal ini penting agar aplikasi dapat digunakan secara efisien oleh pengguna yang memiliki kebutuhan berbeda, baik untuk eksplorasi data, perlindungan informasi, maupun pengembangan lebih lanjut dalam konteks *Natural Language Processing* (NLP).

Dengan desain ini, aplikasi HEERA menekankan prinsip modularitas, keamanan data, dan kemudahan penggunaan. Proses yang digambarkan dalam flowchart memastikan bahwa setiap pengguna diarahkan melalui alur yang logis dan terdokumentasi dengan baik, mulai dari akses awal hingga interaksi spesifik dengan modul yang dipilih.

Flowchart utama pada Gambar 3.1 menggambarkan bahwa pengguna pertama kali akan melihat disclaimer modal saat membuka aplikasi, kemudian diarahkan ke halaman utama yang menampilkan tiga kartu pilihan modul. Setiap modul memiliki interface yang serupa namun dengan fungsionalitas yang berbeda sesuai dengan model yang digunakan.

B Flowchart Subprocess untuk Setiap Modul

Setiap modul (PII Masking, Subject Detection, Entity Detection) mengikuti alur kerja yang sama dengan dua *path* utama: input teks dan input file.



Gambar 3.2. Flowchart Subprocess untuk Modul HEERA

Flowchart subprocess pada Gambar 3.2 menunjukkan dua jalur pemrosesan utama, yaitu jalur input teks dan jalur input file. Pemisahan ini memungkinkan fleksibilitas dalam penggunaan aplikasi, tergantung pada kebutuhan dan preferensi pengguna dalam memberikan data masukan.

Pada jalur *Text*, pengguna terlebih dahulu memilih tab *Text* pada halaman aplikasi. Setelah itu, akan ditampilkan antarmuka bimbingan berupa area input teks, contoh teks, serta konfigurasi filter yang relevan dengan masing-masing modul. Pengguna dapat langsung mengetikkan teks yang ingin dianalisis atau memilih

dari teks contoh yang telah disediakan, kemudian menekan tombol *Submit* untuk melanjutkan proses.

Setelah input dikirim, sistem akan menampilkan indikator pemrosesan berupa *spinner*. Di belakang layar, sistem akan mengevaluasi jenis endpoint yang digunakan (PII, Subject, atau Entity), dan memanggil endpoint API yang sesuai. Hasil dari pemrosesan akan dikembalikan dalam format *JSON*, yang kemudian ditampilkan di antarmuka pengguna bersama informasi tambahan seperti waktu pemrosesan dan opsi untuk mengunduh hasil dalam bentuk *JSON file*.

Sementara itu, pada jalur *File*, pengguna memilih tab *File*, lalu diarahkan ke bagian pengunggahan file. Sistem mendukung format berkas seperti `.csv` dan `.xlsx`. Pengguna juga disediakan contoh file yang dapat diunduh untuk referensi. Setelah file diunggah dan tombol *Process File* ditekan, sistem akan menyimpan file sementara di server dan mulai melakukan batch processing.

Pada tahap selanjutnya, backend akan memproses file yang telah disimpan dengan cara yang serupa dengan input teks, yakni memanggil endpoint sesuai modul yang dipilih. Setelah respons diterima dari API, sistem membuat file baru (biasanya dalam format `.xlsx` atau `.csv`) yang memuat hasil pemrosesan. Pengguna kemudian dapat mengunduh file hasil tersebut secara langsung dari aplikasi. File sementara yang sebelumnya disimpan akan dihapus untuk menjaga efisiensi penyimpanan dan keamanan data.

Kedua jalur ini akhirnya bergabung kembali menuju satu akhir proses yang sama, yaitu penyampaian hasil akhir kepada pengguna. Flowchart ini secara menyeluruh mencerminkan modularitas dan fleksibilitas desain HEERA, serta memisahkan dengan jelas proses berbasis teks dan berbasis file tanpa mengubah pengalaman pengguna secara signifikan.

C Struktur Navigasi dan Halaman Utama

Aplikasi HEERA memiliki struktur navigasi multi-halaman yang jelas, memungkinkan pengguna untuk berpindah antar modul dengan mudah dan intuitif. Navigasi ini diimplementasikan menggunakan fitur `st.switch_page()` dari pustaka Streamlit Navigator, yang menangani perpindahan antar halaman dalam antarmuka pengguna. Setiap modul utama (PII Masking, Subject Detection, dan Entity Detection) diatur dalam berkas Python terpisah untuk menjamin modularitas dan skalabilitas aplikasi.

Halaman utama (`app.py`) berperan sebagai titik awal bagi seluruh

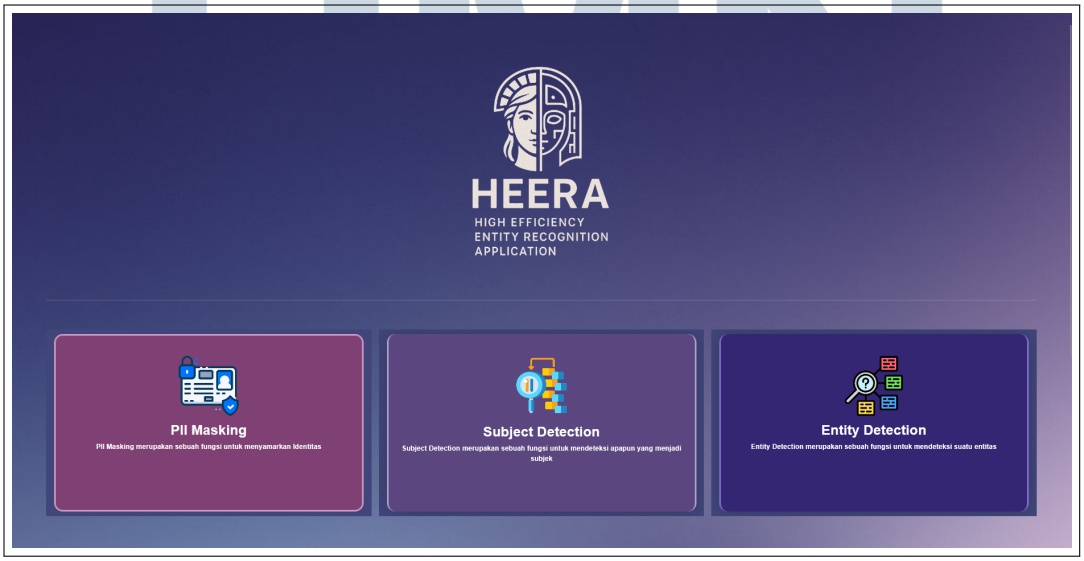
pengguna, termasuk akses langsung (*direct access*) dan melalui integrasi token SSO. Halaman ini juga mencakup mekanisme validasi `session state` untuk menjaga konsistensi autentikasi dan pengalaman pengguna.

C.1 Halaman Utama (*Home Page*)

Halaman ini menampilkan tiga kartu utama: ”PII Masking”, ”Subject Detection”, dan ”Entity Detection”. Setiap kartu dirancang menggunakan pustaka `streamlit-card` dan didesain secara estetis dengan pendekatan *glassmorphism* melalui CSS kustom. Efek visual seperti *hover animation*, *drop shadow*, dan `backdrop-filter` digunakan untuk memberikan pengalaman pengguna yang modern dan dinamis.

Pemilihan desain kartu didasarkan pada kebutuhan antarmuka yang intuitif dan informatif, dengan ikon yang merepresentasikan fungsi utama dari masing-masing modul. Selain itu, ilustrasi latar belakang halaman (*background gradient*) juga disesuaikan menggunakan gambar `JPEG` terenkode Base64 agar kompatibel di semua browser, serta menjaga konsistensi estetika antarmuka.

Logo aplikasi HEERA ditampilkan di bagian atas sebagai identitas visual, sementara modal *disclaimer* secara otomatis muncul jika pengguna belum menyetujui syarat penggunaan. Modal ini memberikan edukasi awal mengenai keterbatasan sistem dan tanggung jawab pengguna dalam menggunakan aplikasi untuk keperluan eksplorasi NLP.



Gambar 3.3. Tampilan Halaman Utama Aplikasi HEERA

C.1.1 Logika Navigasi Antar Halaman

Navigasi antar halaman diimplementasikan menggunakan fungsi `st.switch_page()` dari pustaka `streamlit-extras`, yang memudahkan pemisahan modul berdasarkan berkas. Klik pada salah satu kartu akan men-trigger perpindahan ke modul terkait, yaitu:

- `pages/1_PII_Masking.py` untuk fungsi penyamaran entitas pribadi,
- `pages/2_Subject_Detection.py` untuk fungsi deteksi subjek,
- `pages/3_Entity_Detection.py` untuk fungsi deteksi entitas umum.

Setiap halaman memiliki struktur antarmuka independen namun tetap mengikuti konsistensi desain yang ditetapkan dari `main.css` dan `custom_css` pada halaman utama. Navigasi ini juga mempertahankan `session state` pengguna, termasuk `token`, `username`, dan mode akses (*direct* atau terautentikasi), untuk memastikan pengalaman pengguna tetap mulus dan aman saat berpindah halaman.

C.1.2 Fitur Tambahan Halaman Utama

Halaman utama juga memuat bagian *Feature Showcase* sebagai simulasi interaktif yang menunjukkan perbandingan hasil sebelum dan sesudah proses masking data PII. Elemen ini disusun dalam bentuk `demo-card` untuk memberikan visualisasi langsung dan pemahaman konteks terhadap pengguna awam.

Selain itu, elemen-elemen seperti header dan footer turut didesain secara responsif dan minimalis, dengan penggunaan ikon `FontAwesome`, tata letak berbasis grid `st.columns()`, serta pengaturan *background transparency* untuk seluruh komponen Streamlit, memberikan tampilan bersih dan profesional.

`pages/3_Entity_Detection.py`) untuk modularitas kode.

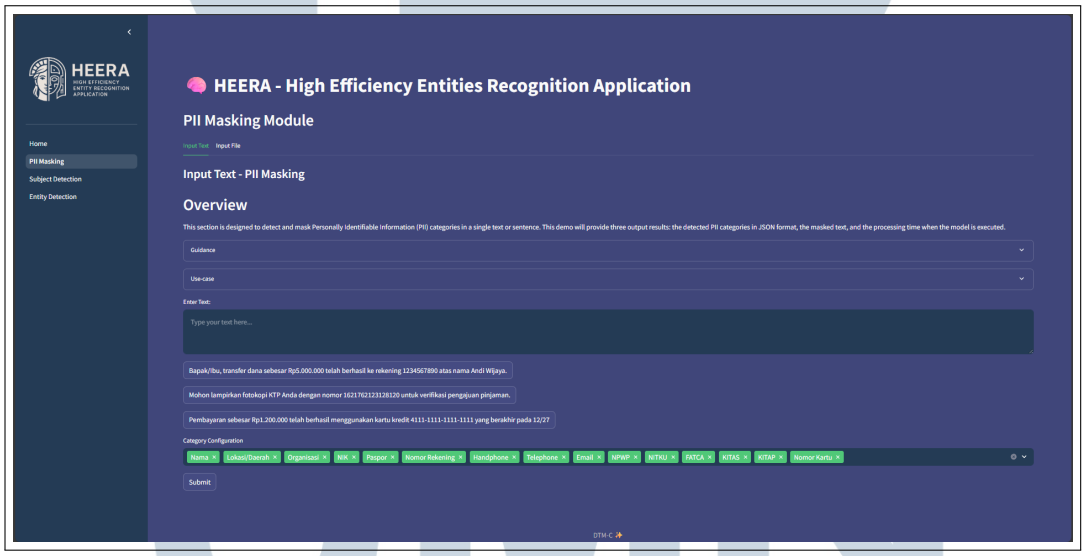
D Modul PII Masking

Modul ini dirancang untuk mendeteksi dan menyamarkan Informasi Identitas Pribadi (PII) dalam teks, baik secara individual maupun massal. Tujuan utama modul ini adalah mendukung kepatuhan terhadap regulasi privasi data serta meningkatkan keamanan informasi melalui deteksi otomatis terhadap entitas sensitif seperti nama, nomor rekening, kartu identitas, nomor telepon, dan data

pribadi lainnya. Antarmuka pengguna pada modul ini dibangun menggunakan Streamlit dan dibagi menjadi dua tab utama: ”Input Text” dan ”Input File”, masing-masing dirancang untuk memenuhi kebutuhan pengguna dalam skenario penggunaan yang berbeda.

D.1 Fitur Input Teks

Tab pertama, yaitu ”Input Text”, menyediakan antarmuka bagi pengguna untuk memasukkan teks secara manual atau memilih dari tiga contoh kalimat yang tersedia. Contoh-contoh ini mencerminkan situasi umum yang melibatkan PII, seperti transaksi perbankan, verifikasi KTP, dan pembayaran kartu kredit. Fitur ini sangat berguna untuk pengujian cepat dan pemahaman intuitif terhadap cara kerja masking.



Gambar 3.4. Tampilan Antarmuka Modul PII Masking (Input Teks)

Pengguna dapat memilih kategori PII yang ingin dideteksi melalui *multiselect* berbasis konfigurasi (misalnya: `['name', 'phone', 'account number']`), yang memungkinkan fleksibilitas dalam cakupan masking. Setelah teks dimasukkan dan kategori dipilih, pengguna menekan tombol `Submit` untuk memproses data. Hasilnya meliputi:

- Deteksi entitas PII dalam format `JSON`,
- Teks yang telah disamarkan (*masked*),
- Waktu pemrosesan dalam satuan detik.

Seluruh hasil ditampilkan secara berdampingan, sehingga memudahkan perbandingan dan verifikasi. Selain itu, pengguna dapat mengunduh hasil deteksi dalam bentuk berkas `.json` untuk analisis lanjutan atau pelaporan.

Penanganan Kesalahan: Sistem secara otomatis akan memberikan peringatan apabila pengguna tidak memasukkan teks sebelum menekan tombol `Submit`. Selain itu, jika proses masking gagal karena gangguan sistem atau hasil deteksi kosong, aplikasi akan menampilkan pesan kesalahan yang informatif agar pengguna dapat mengambil tindakan korektif.

D.2 Fitur Input File

Tab kedua, ”Input File”, dirancang untuk mendukung pemrosesan data dalam skala lebih besar. Modul ini menerima unggahan file dalam format `.csv` atau `.xlsx`, di mana teks yang akan dianalisis harus diletakkan pada kolom pertama. Tujuan dari desain ini adalah memastikan kemudahan integrasi dengan dataset nyata, seperti laporan pelanggan, dokumen administratif, atau kumpulan catatan medis.

Antarmuka memberikan panduan terperinci dalam bentuk `expander` mengenai cara penggunaan, serta menyediakan tombol unduhan untuk dua jenis file contoh:

- **Example Input**: berisi struktur template yang dapat diisi pengguna,
- **Sample File**: berisi data simulasi sebagai referensi.

Setelah file diunggah dan kategori deteksi dipilih, pengguna dapat memulai proses masking dengan menekan tombol `Process File`. Output-nya adalah file `.xlsx` yang telah diperluas dengan kolom-kolom tambahan:

1. **Input Text**: teks asli dari file,
2. **Detected PII**: entitas yang berhasil dideteksi (format JSON),
3. **Masked Text**: teks hasil penyamaran,
4. **Processing Time**: waktu proses per baris data.

Berkas hasil dapat diunduh langsung melalui tombol unduh yang muncul setelah proses selesai.

Penanganan Kesalahan: Sistem mendeteksi beberapa kondisi kesalahan yang umum terjadi, seperti:

- *File tidak diunggah:* Pengguna akan menerima notifikasi untuk mengunggah file terlebih dahulu.
- *Format file tidak sesuai:* Sistem hanya menerima file dengan ekstensi `.csv` dan `.xlsx`.
- *Struktur kolom salah:* Jika kolom pertama tidak berisi teks atau file kosong, proses masking akan dibatalkan dengan pesan kesalahan.
- *Gagal memproses atau menyimpan file:* Error saat pembacaan atau penulisan file ditangani dengan logika penanganan `try-except`, dan pengguna akan diberi tahu melalui pesan kesalahan eksplisit.

Pendekatan ini memastikan bahwa pengguna tidak hanya dipandu dalam interaksi normal, tetapi juga dilindungi dari kegagalan teknis yang dapat menghambat pengalaman pengguna.

Gambar 3.5. Tampilan Antarmuka Modul PII Masking (Input File)

Modul PII Masking pada HEERA tidak hanya memperhatikan aspek fungsionalitas, tetapi juga memberikan pengalaman pengguna yang informatif melalui panduan penggunaan, studi kasus (*use-case*), sistem umpan balik langsung, serta mekanisme penanganan kesalahan yang dirancang dengan cermat. Pendekatan ini menunjukkan kesesuaian antara antarmuka pengguna dan kebutuhan analisis data sensitif dalam berbagai domain seperti keuangan, hukum, dan kesehatan.

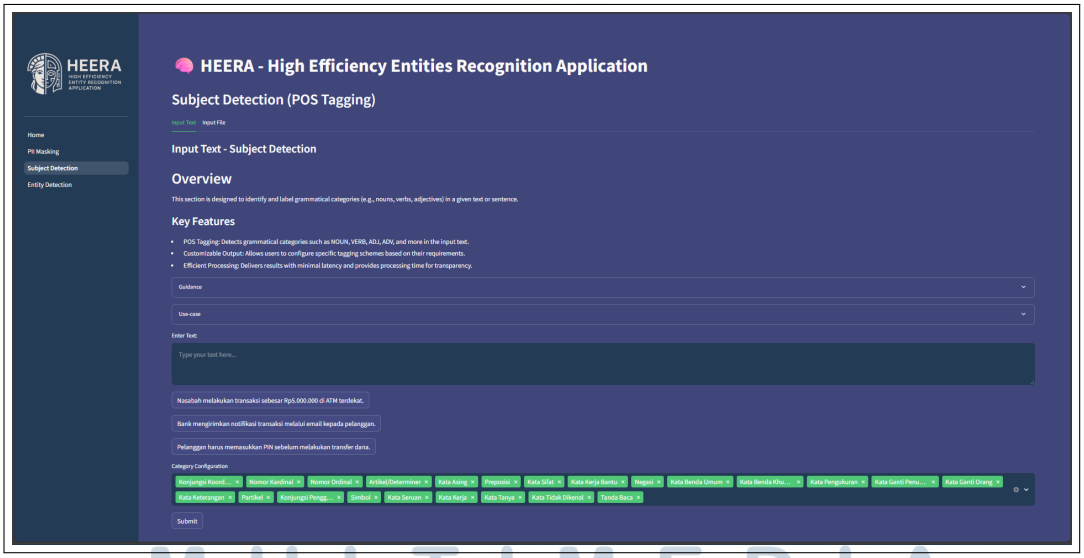
E Modul Subject Detection (POS Tagging)

Modul ini berfungsi untuk mengidentifikasi dan melabeli kategori tata bahasa (*Part-of-Speech*, POS), seperti kata benda (noun), kata kerja (verb), dan kata sifat (adjective) dalam teks. Keluaran dari modul ini dapat digunakan dalam berbagai aplikasi seperti analisis sentimen, pemrosesan bahasa alami lanjutan, hingga sistem manajemen konten berbasis linguistik.

Antarmuka dibagi menjadi dua tab utama: **Input Text** dan **Input File**, masing-masing memiliki tampilan dan mekanisme kerja yang disesuaikan dengan kebutuhan pengguna.

E.1 Fitur Input Teks

Pada tab ini, pengguna dapat memasukkan teks secara manual atau memilih dari beberapa contoh kalimat yang disediakan. Setelah menekan tombol *Submit*, sistem akan memproses teks dan menampilkan hasil POS Tagging secara visual dengan menggunakan pustaka `st-annotated-text`. Setiap kata akan disorot bersama label kategorinya (misal: NOUN, VERB, ADJ).



Gambar 3.6. Tampilan Antarmuka Modul Subject Detection (Input Teks)

Pengguna juga dapat mengunduh hasil deteksi dalam format JSON. Informasi waktu pemrosesan ditampilkan untuk memberikan transparansi terhadap efisiensi modul.

Modul ini dilengkapi dengan **error handling** untuk berbagai kemungkinan kesalahan:

- Jika teks kosong atau tidak valid, sistem akan memberikan peringatan agar pengguna mengisi teks terlebih dahulu.
- Jika hasil deteksi tidak valid atau format entitas tidak sesuai (misal: tidak terdapat atribut `word` atau `entity_group`), sistem akan menampilkan galat atau peringatan dan melewati entitas tersebut.
- Jika terjadi kesalahan saat mengurai JSON atau format output tidak dapat diproses, sistem akan memberikan notifikasi kesalahan secara eksplisit.

E.2 Fitur Input File

Tab ini memungkinkan pengguna mengunggah file dalam format `.csv` atau `.xlsx` yang berisi satu atau lebih baris teks. File yang diunggah akan diproses oleh model POS Tagging, dan hasilnya akan dituliskan ke dalam file `.xlsx` baru. File hasil berisi tiga kolom: teks asli, kategori yang terdeteksi, dan waktu pemrosesan.

Gambar 3.7. Tampilan Antarmuka Modul Subject Detection (Input File)

Beberapa validasi penting juga diterapkan:

- Hanya satu file yang dapat diunggah pada satu waktu, dan harus berada dalam format `.csv` atau `.xlsx`.
- Teks yang akan diproses harus berada di kolom pertama.

- Jika terjadi kesalahan saat menyimpan file, sistem akan menampilkan pesan galat yang menjelaskan permasalahan secara langsung.
- Bila terjadi error selama proses tagging, pengguna akan menerima pesan kesalahan yang informatif untuk membantu proses debugging.

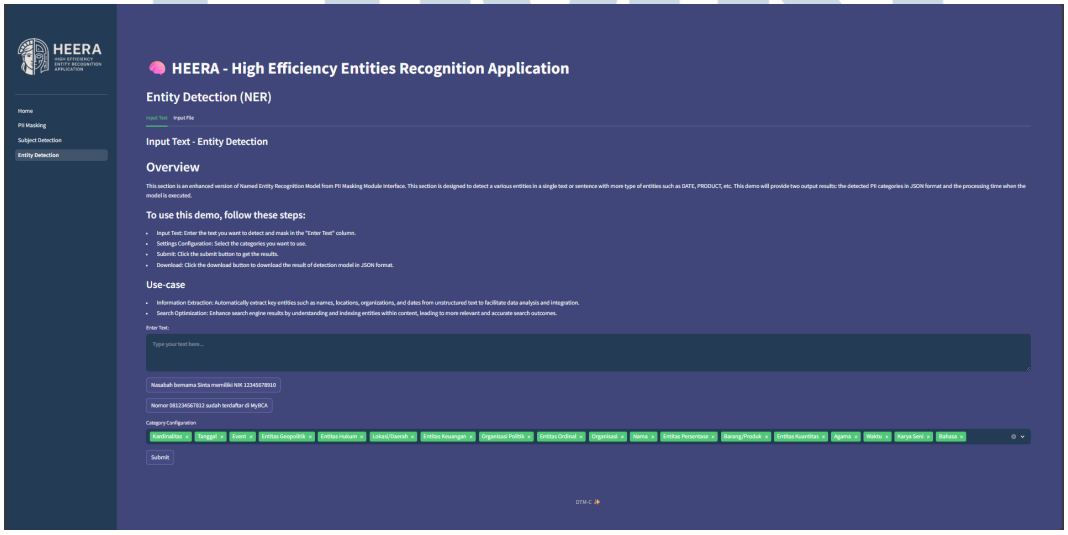
Pengguna juga dapat mengunduh berkas contoh input dan sampel keluaran melalui tombol *Download Example* yang tersedia.

F Modul Entity Detection (NER)

Modul ini dirancang untuk mendeteksi berbagai jenis entitas bernama (*named entities*) seperti nama, lokasi, organisasi, tanggal, produk, dan lain sebagainya dalam teks. Deteksi dilakukan secara otomatis dengan menggunakan model berbasis *transformer*, dan hasilnya divisualisasikan secara interaktif.

F.1 Fitur Input Teks

Pengguna dapat memasukkan teks secara manual atau memilih dari contoh yang disediakan. Setelah menekan tombol *Submit*, sistem akan menampilkan hasil deteksi berupa teks beranotasi menggunakan `annotated_text`, yang menyoroti entitas yang dikenali beserta kategorinya. Sistem juga menampilkan waktu pemrosesan dalam satuan detik dan menyediakan tombol unduh untuk menyimpan hasil deteksi dalam format JSON.



Gambar 3.8. Tampilan Antarmuka Modul Entity Detection (Input Teks)

- Tersedia beberapa kategori entitas yang dapat dikonfigurasi melalui `multiselect`.
- Terdapat tombol contoh (*preset*) untuk mempermudah pengguna dalam memahami proses deteksi.
- Hasil akan muncul hanya jika proses deteksi berhasil dan entitas valid ditemukan dalam teks input.

Penanganan Kesalahan:

- Jika input kosong saat tombol *Submit* ditekan, akan muncul peringatan kepada pengguna untuk mengisi teks terlebih dahulu.
- Jika hasil deteksi tidak memuat entitas valid (misalnya entitas kosong atau format JSON tidak sesuai), sistem akan memberikan peringatan atau pesan kesalahan.
- Setiap kegagalan dalam proses parsing JSON atau entitas yang tidak memiliki atribut penting seperti `word` atau `entity_group` akan ditangani dengan notifikasi `st.warning()` atau `st.error()`.

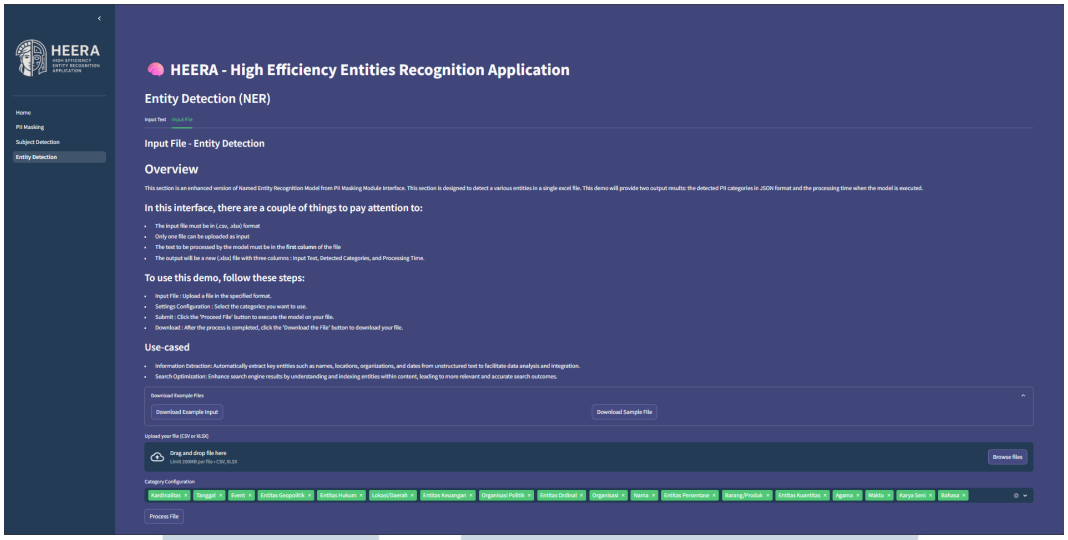
F.2 Fitur Input File

Modul ini juga mendukung pemrosesan entitas dari berkas masukan. Pengguna dapat mengunggah berkas dengan format `.csv` atau `.xlsx`, yang berisi kumpulan teks pada kolom pertama. Setelah diproses, sistem menghasilkan berkas baru dengan tiga kolom utama: teks input, kategori entitas yang terdeteksi, dan waktu pemrosesan.

- Hanya satu file yang dapat diproses dalam satu waktu.
- File harus memiliki teks pada kolom pertama; kolom lain akan diabaikan.
- Setelah pemrosesan selesai, pengguna dapat mengunduh hasilnya dalam format `.xlsx`.

Penanganan Kesalahan:

- Sistem memverifikasi format file yang diunggah. Jika format tidak sesuai, akan muncul peringatan.



Gambar 3.9. Tampilan Antarmuka Modul Entity Detection (Input File)

- Jika tidak ada file yang diunggah, sistem akan menampilkan pesan agar pengguna mengunggah terlebih dahulu.
- Bila terjadi kegagalan dalam proses tagging, pengguna akan diberi notifikasi bahwa pemrosesan gagal dan diminta memeriksa log atau format file.
- Pada proses penyimpanan file temporer, pengecualian ditangani menggunakan blok `try-except` untuk mencegah gangguan antarmuka.

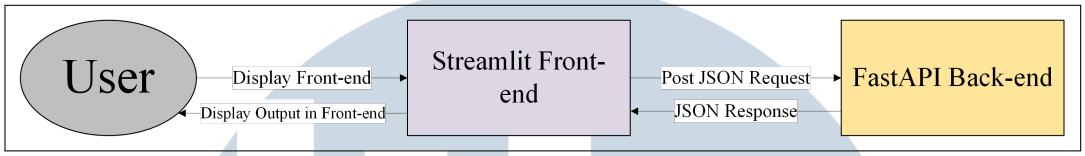
3.3.2 Integrasi *Frontend* dengan *Backend* Model

Tugas kedua melibatkan integrasi komprehensif antara antarmuka *frontend* Streamlit dengan layanan *backend* yang mengekspos model-model AI. Integrasi ini merupakan aspek kritis yang menentukan performa dan responsivitas aplikasi secara keseluruhan.

Proses integrasi dimulai dengan analisis mendalam terhadap API *backend* yang tersedia, termasuk struktur endpoint, format data input dan output, serta mekanisme error handling. Setiap model memiliki endpoint terpisah dengan parameter dan response format yang spesifik, sehingga diperlukan implementasi yang disesuaikan untuk masing-masing model.

Implementasi komunikasi dilakukan menggunakan protokol HTTP dengan metode POST, dimana data dikirimkan dalam format JSON. Sistem dirancang dengan mekanisme asynchronous untuk menghindari blocking UI selama proses pemrosesan model yang memerlukan waktu relatif lama. Indikator loading berupa

spinner ditampilkan untuk memberikan feedback kepada pengguna bahwa sistem sedang memproses request.



Gambar 3.10. Diagram Integrasi antara Frontend Streamlit dan Backend FastAPI

Penanganan error dikembangkan secara komprehensif untuk mengantisipasi berbagai skenario kegagalan, mulai dari timeout connection, invalid response format, hingga server error. Setiap error ditampilkan dengan pesan yang informatif kepada pengguna, membantu dalam troubleshooting dan meningkatkan user experience.

Aplikasi Streamlit ini terintegrasi dengan layanan *FastAPI* sebagai *backend* yang mengekspos model-model AI (NER, POS Tagging, PII Masking).

A Mekanisme Integrasi

A.1 Pemanggilan API

Interaksi antara *frontend* Streamlit dan *backend* model dilakukan melalui permintaan HTTP POST ke *endpoint* API yang relevan. Data dikirimkan dalam format JSON dan respon dari *backend* juga diharapkan dalam format JSON, yang kemudian diproses dan ditampilkan di *frontend*. Penanganan error dasar juga diterapkan untuk memberikan pesan informatif kepada pengguna.

A.2 Parameter dan Respon

Pengiriman parameter (teks input, konfigurasi kategori) dari *frontend* ke *backend* dilakukan melalui tubuh permintaan JSON. Hasil pemrosesan dari model *backend* dikembalikan dalam format JSON, yang kemudian diuraikan oleh *frontend* untuk menampilkan deteksi entitas, teks yang dimasker, atau label POS, beserta waktu pemrosesan.

3.4 Kendala dan Solusi yang Ditemukan

Selama pengembangan aplikasi HEERA, beberapa kendala dihadapi, terutama terkait integrasi, performa, dan pengalaman pengguna. Solusi yang

diterapkan untuk mengatasi kendala-kendala ini diuraikan sebagai berikut:

3.4.1 Kendala dalam Integrasi Frontend dan Backend

A Penanganan Data dan Format Respon API

A.1 Kendala

Model NLP sering mengembalikan data dalam format JSON yang kompleks, yang memerlukan penguraian dan visualisasi yang hati-hati di Streamlit. Inkonsistensi format JSON dari *backend* juga menjadi tantangan.

A.2 Solusi

Diimplementasikan fungsi konversi data (`st-annotated-text`) yang robust untuk mengubah struktur JSON dari *backend* menjadi format yang dapat diterima oleh komponen visualisasi Streamlit. Dilakukan validasi format JSON untuk menangani respon yang tidak valid dan memberikan pesan error yang informatif kepada pengguna.

3.4.2 Kendala Performa dan Pengalaman Pengguna

A Waktu Pemrosesan Model

A.1 Kendala

Beberapa model NLP, terutama saat memproses teks panjang atau file besar, dapat memakan waktu cukup lama, menyebabkan aplikasi terlihat tidak responsif.

A.2 Solusi

Digunakan indikator *spinner* (`st.spinner()`) untuk memberikan umpan balik visual kepada pengguna bahwa proses sedang berlangsung, sehingga meningkatkan persepsi responsivitas. Untuk kasus pemrosesan file besar, di masa depan dapat dipertimbangkan implementasi antrean tugas asinkron untuk memisahkan proses berat dari *frontend* Streamlit.

B Pengelolaan *State* di Streamlit

B.1 Kendala

Streamlit melakukan *rerun* seluruh skrip setiap kali ada interaksi pengguna, yang dapat menyebabkan hilangnya *state* atau eksekusi ulang operasi yang tidak perlu.

B.2 Solusi

Menggunakan `st.session_state` secara ekstensif untuk menyimpan dan mempertahankan *state* aplikasi antar *rerun*, seperti teks input, pilihan kategori, dan hasil pemrosesan. Ini memastikan pengalaman pengguna yang mulus dan data tidak hilang setelah interaksi.

3.4.3 Kendala Tampilan UI/UX Streamlit

A Kustomisasi Tampilan

A.1 Kendala

Gaya default Streamlit tidak selalu sesuai dengan estetika yang diinginkan. Kustomisasi mendalam pada efek visual seperti *glassmorphism* atau gradien latar belakang memerlukan pendekatan khusus.

A.2 Solusi

Menggunakan `st.markdown(custom_css, unsafe_allow_html=True)` untuk menyuntikkan CSS kustom. Ini memungkinkan penyesuaian mendalam pada gaya elemen Streamlit, termasuk gradien latar belakang, efek kartu *glassmorphism*, dan penyembunyian elemen UI bawaan Streamlit (seperti header atau sidebar navigasi default pada kondisi tertentu).

B Penanganan File Temporer

B.1 Kendala

Saat memproses file yang diunggah, diperlukan mekanisme untuk menyimpan file secara temporer di server sebelum diproses oleh model, dan

kemudian menghapusnya setelah tidak lagi diperlukan.

B.2 Solusi

Mengimplementasikan fungsi `save_uploaded_file` menggunakan `tempfile.NamedTemporaryFile`. Ini memastikan file yang diunggah disimpan di lokasi sementara yang aman dan dapat diakses untuk diproses. Setelah pemrosesan selesai dan file output diunduh, file sementara dihapus menggunakan `os.remove()` untuk menjaga kebersihan sistem.