Spesifikasi Tugas Besar IF2211 – Asisten Pengingat (Versi Revisi)

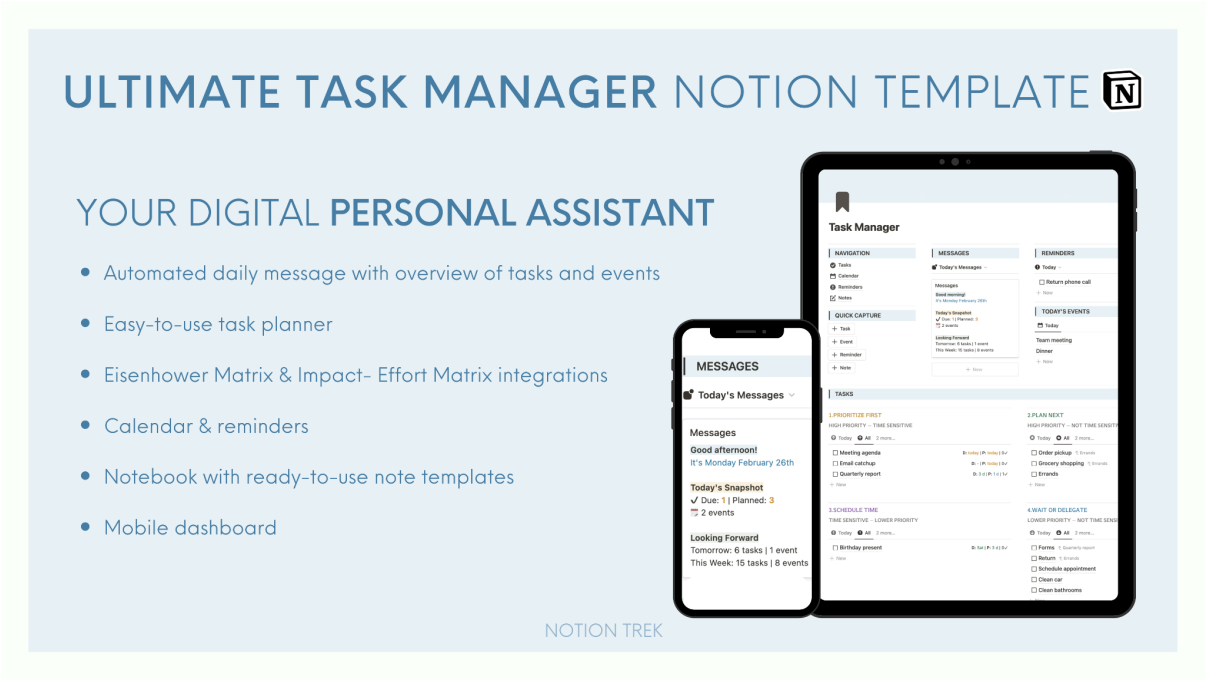
Ada yang tahu humagear? Personal Assistant di Zero-One (Ceritanya), hehe

I. Latar Belakang dan Tujuan Pembelajaran

1.1 Latar Belakang: Menaklukkan Badai Tenggat Waktu

Sebagai mahasiswa informatika, Anda hidup di tengah badai informasi: tenggat waktu tugas, jadwal kuis, ujian, dan praktikum yang datang silih berganti. Mengelola semua ini adalah sebuah tantangan algoritmik tersendiri. Tak jarang, sebuah tenggat waktu penting terlewat bukan karena kurangnya niat, tetapi karena sulitnya melacak semua komitmen yang ada.

Dalam tugas ini, Anda akan membangun solusi untuk masalah ini. Anda akan merancang dan mengimplementasikan sebuah **Asisten Pengingat Cerdas**—sebuah *chatbot* interaktif yang berfungsi sebagai tangan kanan digital Anda dalam menaklukkan jadwal akademik. Dengan memanfaatkan kekuatan algoritma pencocokan string dan *Regular Expression*, Anda akan menciptakan sebuah alat yang mampu memahami perintah bahasa alami untuk mencatat, mencari, dan mengelola semua tenggat waktu Anda. Proyek ini bukan hanya tentang menulis kode; ini tentang menerapkan teori algoritma untuk memecahkan masalah nyata yang relevan dengan kehidupan Anda sehari-hari.



Gambar 1: Mock-up Konseptual Interaksi dengan Asisten Pengingat Cerdas

1.2 Tujuan Pembelajaran

Setelah menyelesaikan tugas besar ini, Anda diharapkan mampu:

Tujuan Inti (Algoritmik):

- Mengimplementasikan algoritma pencocokan string Knuth-Morris-Pratt (KMP) dan Boyer-Moore dari prinsip dasar teoretis menjadi kode fungsional.
- Merancang dan menerapkan pola *Regular Expression* (Regex) yang kompleks untuk mengekstraksi informasi terstruktur (seperti tanggal, kode mata kuliah, dan topik) dari teks bahasa alami yang tidak terstruktur.
- Menganalisis dan menerapkan metrik kemiripan string (seperti Jarak Levenshtein) untuk membangun fitur koreksi kesalahan ketik (*typo*).

Tujuan Terapan (Rekayasa Perangkat Lunak):

- Merancang dan membangun aplikasi *full-stack* berbasis web dari tahap konseptualisasi hingga implementasi fungsional.
- Menerapkan prinsip-prinsip pengembangan perangkat lunak yang baik, termasuk modularitas kode dan penggunaan komentar yang efektif.
- Mengkomunikasikan analisis, perancangan, dan hasil kerja teknis secara efektif melalui dokumentasi tertulis yang terstruktur (laporan teknis dan README.md).

II. Spesifikasi Wajib Aplikasi

Anda akan membangun Asisten Pengingat Cerdas menggunakan sistem Tanya-Jawab. Sistem ini harus dapat mendeteksi berbagai format perintah dari pengguna. Untuk melakukannya, Anda perlu merancang serangkaian aturan deteksi perintah, terutama menggunakan *Regular Expression*, untuk mem-*parsing* masukan dari pengguna. Berikut adalah fitur-fitur wajib yang harus dimiliki oleh aplikasi Anda.

2.1 Menambahkan Tugas Baru

- Pengguna dapat menambahkan tugas baru dengan mengirimkan pesan dalam format bahasa alami. Sistem harus mampu mengekstrak informasi penting dari pesan tersebut dan menyimpannya.
- **Kriteria Penerimaan:**
    1. Sebuah pesan diklasifikasikan sebagai perintah penambahan tugas jika mengandung empat komponen informasi yang didefinisikan dalam **Tabel 2.1**.
    2. Ekstraksi keempat komponen ini wajib menggunakan *Regular Expression*.
    3. Jika berhasil, asisten akan memberikan respons konfirmasi yang berisi ID unik (bilangan bulat yang meningkat secara sekuensial), tanggal, kode mata kuliah, jenis, dan topik tugas.
- **Contoh Interaksi:**

```
Pengguna: Tubes IF2211 String Matching pada 14 April 2024

Asisten: (ID: 1) 14/04/2024 - IF2211 - Tubes - String Matching
```

Tabel berikut mendefinisikan batasan minimum untuk format input yang harus didukung oleh aplikasi Anda.

Komponen	Format Wajib Didukung	Contoh
Tanggal	`DD/MM/YYYY` (misal, 05/04/2024) `DD-MM-YYYY` (misal, 05-04-2024) `D MMMM YYYY` (misal, 5 April 2024)	`28/04/2024` `14-04-2024` `5 Mei 2024`
Kode Mata Kuliah	Dua huruf kapital, diikuti empat digit angka.	`IF2211` `KU1001` `MA1201`
Jenis Tugas	Harus cocok (secara *case-insensitive*) dengan salah satu kata dari daftar kata penting yang telah didefinisikan (lihat Fitur 2.7).	`Kuis` `Tubes` `Praktikum`

Topik Tugas	Teks bebas yang mengikuti komponen lainnya.	String Matching Bab 2 sampai 3 Membuat Asisten Cerdas

2.2 Melihat Daftar Tugas

- Pengguna dapat meminta daftar tugas yang tersimpan dengan berbagai kriteria penyaringan.
- **Kriteria Penerimaan:**
    1. **Semua Tugas:** Perintah untuk menampilkan semua tugas yang belum selesai.
    2. **Berdasarkan Periode Waktu:** Aplikasi harus bisa mem-parsing dan memfilter tugas berdasarkan:
        - Rentang tanggal spesifik (misal, "antara 5 April 2024 sampai 15 April 2024").
        - N hari ke depan (misal, "7 hari ke depan").
        - N minggu ke depan (misal, "2 minggu ke depan").
        - Hari ini.
    3. **Berdasarkan Jenis Tugas:** Aplikasi harus bisa memfilter tugas berdasarkan jenisnya (misal, "kuis apa saja").
    4. **Kombinasi Filter:** Aplikasi harus mampu menangani kombinasi filter, misalnya "kuis 2 minggu ke depan"
- **Contoh Interaksi:**

```
Pengguna: Apa saja deadline sejauh ini?

Asisten:
1. (ID: 1) 14/04/2024 - IF2211 - Tubes - String Matching
2. (ID: 2) 22/04/2024 - IF3110 - Kuis - Bab 2 sampai 3
```

2.3 Menampilkan Tenggat Waktu Tugas Tertentu

- Pengguna dapat menanyakan tenggat waktu dari sebuah tugas spesifik dengan menyebutkan kode mata kuliah atau topik.
- **Kriteria Penerimaan:**
    1. Aplikasi harus dapat mencari tugas berdasarkan kode mata kuliah atau kata kunci pada topiknya.
    2. Jika ditemukan, asisten akan merespons dengan tanggal tenggat waktu tugas tersebut.
- **Contoh interaksi:**

```
Pengguna: Deadline tubes IF2211 kapan?
Asisten: 14/04/2024
```

2.4 Memperbarui Tugas

- Pengguna dapat mengubah tanggal tenggat waktu dari tugas yang sudah ada, misalnya karena ada pengunduran.
- **Kriteria Penerimaan:**
    1. Perintah harus mengandung ID tugas yang akan diperbarui dan tanggal baru.
    2. Jika ID valid, asisten akan memperbarui data dan memberikan pesan konfirmasi.
    3. Jika ID tidak valid, asisten akan memberikan pesan galat.
- Contoh Interaksi:

```
Pengguna: Saya sudah selesai mengerjakan task 2

Asisten: Mantap! Tugas (ID: 2) telah ditandai selesai.
```

2.5 Menandai Tugas Selesai

- Pengguna dapat menandai sebuah tugas sebagai "selesai". Tugas yang sudah selesai tidak akan muncul lagi dalam daftar tugas.
- **Kriteria Penerimaan:**
    1. Perintah harus mengandung ID tugas yang akan ditandai selesai.
    2. Jika ID valid, asisten akan mengubah status tugas dan memberikan pesan konfirmasi.
    3. Jika ID tidak valid, asisten akan memberikan pesan galat.
- **Contoh Interaksi:**

```
Pengguna: Saya sudah selesai mengerjakan task 2

Asisten: Mantap! Tugas (ID: 2) telah ditandai selesai.
```

2.6 Menampilkan Bantuan

- Pengguna dapat meminta daftar perintah yang dapat dipahami oleh asisten.
- **Kriteria Penerimaan:** Saat pengguna mengirim perintah bantuan (misal, "help", "bantuan", "apa yang bisa kamu lakukan?"), asisten akan merespons dengan daftar fitur dan contoh penggunaannya
- Contoh Interaksi:

```
Pengguna: Bantuan

Asisten:
1. Menambah tugas: [Kode MK]
```

```
2. Melihat daftar tugas: "deadline minggu ini", "kuis apa saja"
```

2.7 Mendefinisikan Daftar Kata Penting

- **Deskripsi:** Aplikasi harus memiliki daftar kata kunci statis untuk mengidentifikasi jenis tugas.
- **Kriteria Penerimaan:**
    1. Daftar ini harus didefinisikan secara *hardcoded* di dalam program.
    2. Daftar harus berisi minimal 5 kata yang berbeda.
    3. Kata-kata ini digunakan untuk mem-parsing "Jenis Tugas" pada Fitur 2.1.

2.8 Menangani Pesan Tidak Dikenali

- **Deskripsi:** Jika masukan pengguna tidak cocok dengan pola perintah manapun, asisten harus memberikan respons yang menandakan bahwa pesan tidak dipahami.
- **Kriteria Penerimaan:**
    1. Setiap pesan yang tidak dapat dipetakan ke Fitur 2.1 hingga 2.6 harus menghasilkan pesan galat yang ramah.
- **Contoh Interaksi:**

```
Pengguna: Apakah mayones sebuah instrumen?

Asisten: Maaf, saya tidak mengerti perintah tersebut. Coba ketik "bantuan" untuk melihat apa saja yang bisa saya lakukan.
```

III. Spesifikasi Teknis: Arsitektur dan Algoritma

3.1. *Tech Stack*

- **Platform:** Aplikasi wajib dibuat berbasis **web**.
- **Bahasa Pemrograman *Backend*:** Anda dapat memilih salah satu dari bahasa berikut yang mendukung *Regular Expression* secara native: Python, JavaScript/TypeScript, Java, atau PHP.
- **Kerangka Kerja (*Framework*):** Disarankan menggunakan kerangka kerja web yang umum untuk bahasa yang dipilih, misalnya: Flask/Django (Python), Express/Next.js (JS/TS), Spring Boot (Java), atau Laravel (PHP).

3.2. Implementasi Algoritma Inti

Implementasi algoritma berikut adalah inti dari penilaian tugas ini.

- ***Regular Expression:*** Wajib digunakan untuk tugas utama *parsing* perintah dan ekstraksi entitas (tanggal, kode mata kuliah, topik) dari input pengguna, seperti yang dijelaskan pada Spesifikasi Fungsional.
- **Algoritma KMP & Boyer-Moore:**
    1. Anda wajib mengimplementasikan kedua algoritma ini **dari nol** (tidak diizinkan menggunakan *library* atau fungsi bawaan yang sudah jadi).
    2. Implementasi ini harus digunakan secara fungsional di dalam aplikasi Anda untuk salah satu dari dua skenario berikut (pilih salah satu):
        - **Skenario A (Pencarian Topik):** Membuat fitur pencarian yang memungkinkan pengguna mencari tugas berdasarkan kata kunci yang cocok persis (*exact match*) di dalam topik tugas. Misalnya, perintah "cari tugas dengan topik 'String Matching'".
        - **Skenario B (Filter Kata Kunci):** Menggunakan algoritma ini untuk memfilter daftar tugas yang mengandung kata kunci tertentu di dalam topiknya. Tujuan dari persyaratan ini adalah untuk memastikan Anda tidak hanya menulis kode algoritma secara terisolasi, tetapi juga mampu mengintegrasikannya ke dalam sistem yang lebih besar.

3.3. Persistensi Data

Anda harus menyimpan data tugas agar tidak hilang saat aplikasi dimatikan. Pilih salah satu dari dua metode berikut:

- **Opsi A (Basis Data Sederhana):** Gunakan sistem basis data relasional seperti SQLite (disarankan karena kemudahannya) atau PostgreSQL. Anda tidak diwajibkan melakukan normalisasi skema, tetapi skema basis data yang Anda gunakan harus didokumentasikan dalam laporan.

- **Opsi B (Sistem Berkas):** Simpan data dalam format file terstruktur seperti JSON (disarankan) atau CSV. Aplikasi Anda harus mengimplementasikan mekanisme untuk memuat data dari file saat dimulai dan menyimpan data ke file saat ada perubahan.

IV. Ketentuan Pengerjaan dan Integritas Akademik

4.1. Kolaborasi Tim

- Tugas ini dikerjakan secara berkelompok yang terdiri dari **2 hingga 3 mahasiswa**.
- Anggota kelompok boleh berasal dari kelas yang berbeda, namun tidak boleh memiliki anggota yang sama persis dengan kelompok Tugas Besar IF2211 sebelumnya.
- **Penting:** Setiap anggota kelompok wajib memahami keseluruhan kode program, termasuk bagian yang tidak mereka kerjakan secara langsung. Pemahaman ini akan diuji secara individual pada saat demonstrasi.

4.2. Kebijakan Integritas Akademik

Prinsip utama tugas ini adalah belajar dengan membuat sendiri.

- **Plagiarisme:** Dilarang keras melakukan plagiarisme, yaitu menyalin kode sumber dari kelompok lain atau dari sumber daring tanpa atribusi dan pemahaman. Program yang terdeteksi memiliki tingkat kesamaan yang tidak wajar dengan program lain akan dikenakan sanksi akademik yang berat.
- **Belajar dari Sumber Lain:** Anda diizinkan untuk mempelajari contoh-contoh program atau tutorial yang ada di internet. Namun, kode yang Anda kumpulkan harus merupakan hasil ketikan dan pemahaman Anda sendiri.

4.3 Penggunaan Alat Bantu (Termasuk AI Generatif)

Di era pengembangan perangkat lunak modern, penggunaan alat bantu seperti AI generatif (misalnya, ChatGPT, GitHub Copilot) adalah hal yang wajar. Kebijakan berikut bertujuan untuk memandu Anda dalam menggunakan alat-alat ini secara etis dan produktif, tanpa mengurangi tujuan pembelajaran inti.

- **Prinsip Utama: Anda bertanggung jawab penuh dan harus dapat menjelaskan setiap baris kode yang Anda kumpulkan.** Penggunaan alat bantu tidak menghilangkan tanggung jawab Anda atas kebenaran, efisiensi, dan orisinalitas pekerjaan Anda.
- **Penggunaan yang Diizinkan:**
    - **Membangkitkan Kode *Boilerplate*:** Menggunakan AI untuk menghasilkan kode kerangka awal, seperti struktur dasar server Flask/Express, templat HTML/CSS untuk antarmuka, atau kode koneksi basis data.
    - **Debugging:** Meminta penjelasan mengenai pesan galat atau bantuan untuk menemukan *bug* logis pada kode Anda.
    - **Refactoring dan Optimisasi:** Meminta saran untuk memperbaiki atau menyederhanakan kode non-algoritmik yang telah Anda tulis.
- **Penggunaan yang Dilarang Keras:**
    - **Membangkitkan Implementasi Algoritma Inti:** Dilarang menggunakan AI untuk menghasilkan implementasi dari algoritma **KMP, Boyer-Moore, atau metrik**

Jarak Levenshtein. Bagian-bagian ini harus Anda tulis sendiri berdasarkan pemahaman teoretis Anda.

- **Kewajiban Dokumentasi:** Jika Anda menggunakan AI secara signifikan (misalnya, untuk menghasilkan seluruh struktur *backend*), Anda wajib mendokumentasikan penggunaannya dalam sebuah bagian khusus di file README.md Anda. Jelaskan *prompt* apa yang Anda gunakan dan bagaimana Anda mengadaptasi hasilnya.

V. Prosedur Pengumpulan dan Penilaian

5.1 Batas Waktu

- **Batas Akhir Pengumpulan:** Rabu, 28 April 2024, pukul 23.59 WIB.
- Keterlambatan pengumpulan akan dikenakan penalti pengurangan nilai yang signifikan.

5.2 Format Pengumpulan

- Seluruh proyek (kode sumber, dokumentasi, dan data pendukung) harus dikumpulkan dalam satu arsip .zip dengan format nama Tubes3_IF2211_NIMKetua.zip.
- Struktur direktori di dalam arsip tersebut wajib mengikuti format berikut:

```
Tubes3_IF2211_NIMKetua/
├── src/      # Berisi seluruh kode sumber aplikasi
├── doc/      # Berisi file laporan dalam format PDF
├── test/     # Berisi data awal (jika ada) atau hasil dump database
└── README.md  # File dokumentasi utama
```

- **README.md:** File ini harus lengkap dan informatif, berisi:
    - Judul proyek dan nama/NIM anggota kelompok.
    - Deskripsi singkat tentang aplikasi.
    - Petunjuk yang jelas tentang cara instalasi dependensi dan cara menjalankan program.
    - Pembagian tugas antar anggota kelompok.
    - Bagian "Dokumentasi Penggunaan AI" jika relevan (lihat Bagian 4.3).

5.3 Kriteria Penilaian

Penilaian akan didasarkan pada kualitas implementasi program, pemahaman konseptual, dan dokumentasi. Demonstrasi program secara langsung akan dijadwalkan setelah batas waktu pengumpulan.

Komponen	Bobot	Deskripsi Kriteria Penilaian
Kebenaran Program	30%	Seluruh fitur wajib (2.1–2.8) berfungsi dengan benar sesuai spesifikasi. Aplikasi stabil dan dapat menangani berbagai kasus input dengan baik.
Implementasi & Pemahaman Algoritma	25%	Implementasi algoritma KMP, Boyer-Moore, dan Regex dilakukan dengan benar dan efisien. Pemahaman mendalam terhadap cara kerja dan kompleksitas algoritma ini ditunjukkan saat sesi demo lisan.

Kualitas Kode & Arsitektur	15%	Kode program ditulis dengan bersih, terstruktur secara modular, mudah dibaca, dan dilengkapi dengan komentar yang relevan dan membantu.
Dokumentasi	15%	Kualitas laporan teknis (sesuai kerangka yang ditentukan) dan kelengkapan serta kejelasan file `README.md`.
Antarmuka & Kreativitas	15%	Desain antarmuka pengguna yang fungsional, intuitif, dan responsif. Adanya unsur kreativitas, seperti fitur tambahan yang relevan atau penyelesaian masalah yang elegan.
Total	**100%**	

VI. Bonus

Anda dapat memperoleh poin tambahan dengan mengimplementasikan fitur-fitur bonus berikut.

6.1 Koreksi *Typo* Cerdas (Bonus: +10 poin)

- Implementasikan fitur yang dapat mendeteksi dan memberikan saran perbaikan jika pengguna melakukan kesalahan ketik (*typo*) pada kata kunci perintah (misal, "dedline" bukan "deadline").
- **Persyaratan:**
    1. Gunakan metrik kemiripan string, seperti **Jarak Levenshtein**, untuk membandingkan kata yang tidak dikenal dengan daftar kata kunci perintah yang valid.
    2. Jika kemiripan berada di atas ambang batas tertentu (misal, 75%), tawarkan kata yang benar sebagai saran kepada pengguna.
    3. Implementasi algoritma Jarak Levenshtein juga harus dibuat **dari nol**.

6.2 *Deployment* Aplikasi (Bonus: +5 poin)

- Lakukan *deployment* aplikasi web Anda ke salah satu layanan *hosting* publik sehingga dapat diakses melalui internet.
- **Persyaratan:**
    1. Aplikasi harus dapat diakses melalui URL publik hingga periode demonstrasi selesai.
    2. Contoh layanan: Vercel, Railway, Heroku, atau penyedia *cloud* lainnya.
    3. Sertakan URL aplikasi yang telah di-*deploy* di dalam `README.md`.

6.3 Video Demonstrasi (Bonus: +5 poin)

- Buat sebuah video demonstrasi (durasi 3–5 menit) yang menjelaskan dan menampilkan cara kerja aplikasi Anda.
- **Persyaratan:**
    1. Video harus diunggah ke YouTube dengan visibilitas "Publik" atau "Tidak Publik".
    2. Video harus memiliki narasi audio yang jelas yang menjelaskan setiap fitur yang didemokan.
    3. Sertakan tautan ke video di dalam README.md.