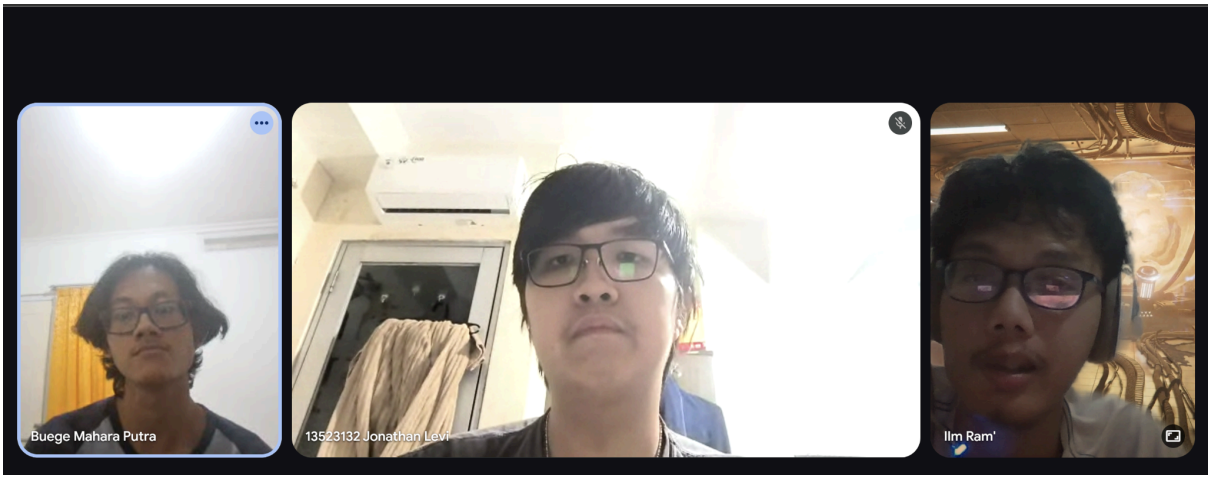
Pemanfaatan Pattern Matching untuk Membangun Sistem ATS (Applicant Tracking System) Berbasis CV Digital

Laporan Tugas Besar 3

IF2211 Strategi Algoritma



Disusun Oleh Kelompok 7 (hr):

10122010	Dzubyan Ilman Ramadhan
13523037	Buege Mahara Putra
13523132	Jonathan Levi

Program Studi Teknik Informatika

Sekolah Teknik Elektro dan Informatika

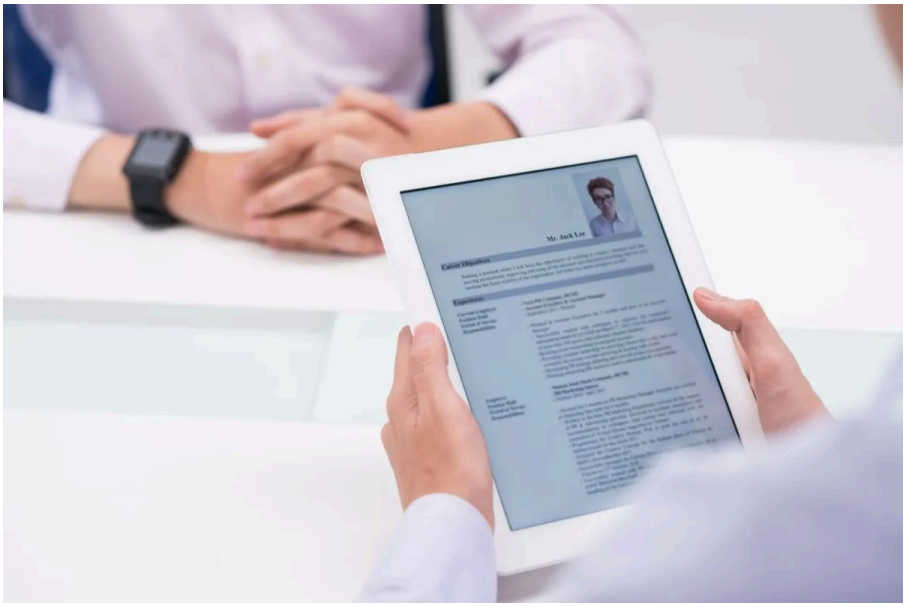
Institut Teknologi Bandung

2025

DAFTAR ISI

Bab 1

Deskripsi Tugas



Gambar 1. CV ATS dalam Dunia Kerja
(Sumber: https://www.antaranews.com/)

Di era digital ini, keamanan data dan akses menjadi semakin penting. Perkembangan proses rekrutmen tenaga kerja telah mengalami perubahan signifikan dengan memanfaatkan teknologi untuk meningkatkan efisiensi dan akurasi. Salah satu inovasi yang menjadi solusi utama adalah Applicant Tracking System (ATS), yang dirancang untuk mempermudah perusahaan dalam menyaring dan mencocokkan informasi kandidat dari berkas lamaran, khususnya Curriculum Vitae (CV). ATS memungkinkan perusahaan untuk mengelola ribuan dokumen lamaran secara otomatis dan memastikan kandidat yang relevan dapat ditemukan dengan cepat.

Meskipun demikian, salah satu tantangan besar dalam pengembangan sistem ATS adalah kemampuan untuk memproses dokumen CV dalam format PDF yang tidak selalu terstruktur. Dokumen seperti ini memerlukan metode canggih untuk mengekstrak informasi penting seperti identitas, pengalaman kerja, keahlian, dan riwayat pendidikan secara efisien. Pattern matching menjadi solusi ideal dalam menghadapi tantangan ini.

Pattern matching adalah teknik untuk menemukan dan mencocokkan pola tertentu dalam teks. Dalam konteks ini, algoritma Boyer-Moore dan Knuth-Morris-Pratt (KMP) sering digunakan karena keduanya menawarkan efisiensi tinggi untuk pencarian teks di dokumen

besar. Algoritma ini memungkinkan sistem ATS untuk mengidentifikasi informasi penting dari CV pelamar dengan kecepatan dan akurasi yang optimal.

Di dalam Tugas Besar 3 ini, Anda diminta untuk mengimplementasikan sistem yang dapat melakukan deteksi informasi pelamar berbasis dokumen CV digital. Metode yang akan digunakan untuk melakukan deteksi pola dalam CV adalah Boyer-Moore dan Knuth-Morris-Pratt. Selain itu, sistem ini akan dihubungkan dengan identitas kandidat melalui basis data sehingga harapannya terbentuk sebuah sistem yang dapat mengenali profil pelamar secara lengkap hanya dengan menggunakan CV digital.

Bab 2
Landasan Teori

2.1 Dasar Teori

a. Algoritma Knuth-Morris-Pratt

Algoritma Knuth-Morris-Pratt mencari *pattern* P sepanjang m karakter di dalam teks T sepanjang n karakter dengan cara membangun tabel LPS (*longest proper prefix which is also suffix*) sepanjang m. LPS[j] menunjukkan panjang terpanjang *proper prefix* dari P[0..j] yang juga merupakan suffix dari P[0..j]. Dengan LPS ini, jika terjadi *mismatch* pada P[j], algoritma ini akan "melompat" ke indeks LPS[j-1] daripada menggeser kembali pola kembali ke awal, sehingga tidak ada karakter yang terbaca ulang.

Tahapan:

1. Hitung tabel LPS untuk *pattern* P.
2. Bandingkan P dengan T
   - Jika *match*, lanjutkan.
   - Jika *mismatch* di posisi j, geser P ke posisi j = LPS[j−1], lalu teruskan tanpa mengurangi indeks teks.
3. Jika pencarian mencapai akhir teks, berhenti.

Keunggulan algoritma ini adalah memiliki waktu yang terjamin linear, sederhana untuk diimplementasikan, dan membutuhkan tambahan memori yang minimum (hanya tabel LPS sepanjang m).

b. Algoritma Boyer-Moore

Algoritma Boyer-Moore mencari *pattern* P di dalam teks T, tetapi bisa melompat lebih jauh saat terjadi *mismatch*, sehingga waktu yang dibutuhkan menjadi lebih optimal.

Kasus:

- Jika suatu posisi P[j] dengan karakter teks T[i] *mismatch*, geser pola sehingga kemunculan terakhir karakter T[i] di P (sebelum posisi j) sejajar dengan i.
  - Jika karakter tersebut tidak ada di P[0..j−1], geser P melewati i sepenuhnya.

- Jika karakter teks T[i] ada di *pattern* P, tetapi semua kemunculannya berada pada indeks lebih besar dari j (P[j+1..m−1]), sehingga tidak ada kemunculan sebelum indeks j yang dapat disejajarkan dengan teks, geser P sebanyak 1 karakter ke kanan.

Tahapan:

1. Bandingkan P dengan T
   - Jika *match*, lanjutkan.
   - Jika *mismatch*, *handle* berdasarkan kasus di atas.
2. Jika pencarian mencapai akhir teks, berhenti.

Keunggulannya adalah pada teks acak, bisa terjadi lompatan besar, sehingga dapat menghemat waktu daripada linear.

c. Algoritma Aho-Corasick

Algoritma Aho-Corasick mencari serangkaian *pattern* {P1,…,Pk} di dalam teks T dengan cara membangun automaton, sehingga semua kemunculan dari semua *pattern* hanya terdeteksi dalam sekali iterasi saja. Automaton terdiri dari:

- *Root Node*
- *Child Node*
  Karakter dari *pattern*
- *Failure link*
  Kembali ke *prefix* terdekat yang mungkin cocok jika tidak ada *child node* yang cocok (mirip tabel LPS pada algoritma Knuth-Morris-Pratt)
- *Output link*
  Jika tiba di *node* karakter terakhir salah satu *pattern* lain melalui *failure link*, sehingga dapat menghasilkan *output* pola tersebut juga.
- *Node output*
  Node daun atau *node* yang merupakan huruf terakhir dari *pattern* tersebut

Tahapan:

1. *Build* sebuah automaton atau *trie* berdasarkan bagian-bagian automaton di atas.
2. Inisialisasi di root *trie*.
3. Untuk setiap karakter pada teks (dimulai dari i = 0), cek *node* berikutnya (*child node* jika merupakan karakter tersebut, atau melalui *failure link*).

- Jika *node* tersebut adalah *node output*, maka *pattern* tersebut *match*. Catat indeks kemunculan (i sekarang - panjang karakter pattern + 1) untuk *pattern* tersebut.

4. Jika pencarian mencapai akhir teks, berhenti.

Keunggulan algoritma ini adalah mampu menangani banyak *pattern* sekaligus dalam waktu pencarian yang linear, sehingga bisa lebih cepat daripada algoritma Boyer-Moore dan Knuth-Morris-Pratt. Namun, jika untuk pencarian satu *pattern* saja, Boyer-Moore lebih cepat.

2.2 Penjelasan Singkat

Aplikasi ini memiliki fungsi utama untuk membantu seorang pengguna dalam pencarian CV yang paling sesuai dengan keyword-keyword yang diberikan. Semakin banyak kemunculan keyword di suatu CV, semakin cocok CV tersebut dengan yang diinginkan. Lalu, pengguna dapat memilih metode pencarian string yang diinginkan (Metode KMP atau BM), dan menentukan banyak hasil yang diinginkan. Setelah pengguna menekan tombol untuk mencari, aplikasi akan menampilkan hasil pencarian sekaligus juga beberapa data seperti waktu pencarian dan banyak kemunculan keyword untuk tiap hasil. Apabila tidak ditemukan satupun CV dengan keyword yang diinginkan, aplikasi akan mencari beberapa CV yang isinya cukup dekat dengan keyword yang diinginkan menggunakan perhitungan jarak Levenshtein. Selain itu, untuk tiap hasil, terdapat tombol agar pengguna juga melihat summary dari hasil pencarian tersebut yang data-data diperoleh dari regex dan tombol untuk membuka file pdf aslinya. Terdapat juga database untuk menyimpan data-data penting dari tiap CV.

Aplikasi akan dibangun menggunakan bahasa Python dengan library PyQt5 dan MySQL untuk penyimpanan data.

Bab 3
Analisis Pemecahan Masalah

3.1 Langkah-langkah pemecahan masalah.

Dalam pengimplementasian ATS CV yang sesuai dengan spesifikasi, kita dapat meninjau permasalahannya menjadi 2, yaitu pencarian CV dengan kemunculan keyword paling banyak, dan pencarian data dalam CV.

Pertama, untuk mencari CV yang memiliki kemunculan terbanyak, kita perlu mengubah semua file-file PDF menjadi teks dalam file txt agar pencarian dapat dilakukan dengan mudah. Dalam Python, terdapat beberapa library untuk melakukan ini seperti PyMuPDF, PyPDF2, atau Pdfminer.six. Setelah itu, kita dapat menggunakan algoritma KMP maupun BM untuk melakukan string pattern searching untuk semua keyword yang diinginkan pada semua file txt yang telah dibangkitkan. Lalu, banyak kecocokan string keyword ditotalkan dan diurutkan. Apabila tidak ada pdf dengan keyword yang diinginkan, maka akan File akan ditampilkan berdasarkan banyak CV yang ingin ditampilkan yang angkanya telah ditentukan.

Untuk pencarian data dari CV untuk ditampilkan di halaman Summary, kita akan melakukan pencarian string dengan Regex karena bentuk template untuk CV yang ATS-friendly itu sendiri, sehingga data-data yang kita perlukan untungnya akan dekat dengan kata-kata kunci yang sesuai. Contoh : Untuk mencari pendidikan seorang aplikan, kita bisa mencari kata-kata yang dekat dengan kata “Pendidikan”. Data-data ini lalu disimpan di database dan untuk digunakan nanti saat menampilkan *summary* dari *applicant*.

3.2 Proses pemetaan masalah menjadi elemen-elemen

1. Konversi pdf ke txt: Modul `pdf_to_text_convert.py` dan `loader.py`

2. Data model & penyimpanan
   - Tabel MySQL:
     - ApplicantProfile (entitas pelamar)
     - ApplicationDetail (detail aplikasi, path CV)
   - *Seeder* (seeder.py / tubes3_seeding.sql) utk pengisian *database*

3. Pencarian (*string matching*)

- Algoritma pencarian tepat/*exact*: KMP (`kmp_search.py`), Boyer–Moore (`bm_search.py`), Aho–Corasick (`ac_search.py`)
- Algoritma pencarian *fuzzy*: Levenshtein (`levensthein.py`) dengan skor kemiripan

4. Ranking: *function* `search()` dan `sort()` pada `main_menu.py`

5. Ekstraksi Data untuk Summary: `skill_extractor.py`, `education_extractor.py`, `job_extractor.py`

6. UI dan *Controller*
   - `main_menu_ui.py`
   - `main_menu.py`
   - `summary_menu.py`

7. Integrasi
   - `main.py`
     - *Load* `.env` dan koneksi MySQL
     - Inisiasi `MainMenu(db_conn)`
   - Alur data: pdf > txt > *database* > *loader* > *search* > GUI

3.3 Fitur fungsional dan arsitektur aplikasi yang dibangun.

3.3.1 Arsitektur Aplikasi

Aplikasi ini dibangun dengan arsitektur MVC (Model-View-Controller) dengan database.

1. *Entry point*:
   - `main.py` - Men-*load* konfigurasi (`.env`, koneksi MySQL), menginisiasi QApplication, menjalankan *loop* GUI, serta membuat dan menampilkan MainMenu (*view*) dengan `db_conn`
   - `requirements.txt` - *Dependency* yang dibutuhkan

2. *Layer* Presentasi (*View* atau UI)
   - `src/views` - Kelas PyQt5 untuk antarmuka (windows, dialogs)
   - `main_menu_ui.py` - Wrapper dari file .ui (Qt designer)
   - `src/ui` - File .ui mentah atau modul UI helper

3. *Layer* Logika (*Controller*)
    - `src/controller` - Mengatur alur data antara *View* dan Model
    - Menangani *event*, validasi, pemanggilan Model, dan *update View*

4. *Layer* Data (Model)
    - `src/models` - Representasi *entity* (misal ApplicantProfile, ApplicationDetail)

5. *Utility* dan *Helper*
    - `src/utils` - *Tool* (kode/fungsi) yang akan digunakan ulang (misal algoritma *search*, Levenshtein *distance*, skor kemiripan)

6. Folder data
    - *Script* untuk *seeding database* (tubes3_seeding.sql, seeder.py) dari PDF
    - Untuk *preprocessing database*

Alur singkat: main.py > *load environment* dan koneksi *database* > *View* memicu *Controller* > *Controller read*/*write* di Model > *View* men-*update* antarmuka.

3.3.2 Fungsi Fungsional

Fungsi Pencarian CV Berdasarkan Keyword

- Menerima input **daftar keyword** dari pengguna.
- Menghitung **jumlah kemunculan keyword** di setiap CV.
- Semakin banyak keyword yang muncul, semakin relevan CV tersebut.

Pemilihan Metode String Matching

- Pengguna dapat memilih metode pencarian string:
    - **KMP (Knuth-Morris-Pratt)**
    - **BM (Boyer-Moore)**
    - **AC (Aho-Corasick)**

Pengaturan dan Output Hasil Pencarian

- Pengguna dapat menentukan **jumlah hasil** yang ingin ditampilkan.
- Setelah pencarian:
    - Menampilkan **daftar CV** paling relevan.
    - Menyediakan **waktu pencarian**.
    - Menampilkan **jumlah kemunculan keyword** untuk setiap hasil.

- Fallback dengan Levenshtein Distance
- Jika tidak ada CV yang mengandung keyword secara langsung:
    - Aplikasi mencari CV yang **"mirip" atau relevan secara konten** menggunakan **Levenshtein Distance**.

Fitur Ekstra pada Hasil Pencarian

- **Tombol untuk melihat summary CV**, yang:
    - Dibuat dengan **regex extraction** dari isi CV.
- **Tombol untuk membuka file PDF** asli dari CV tersebut.

Database Penyimpanan

- Menyimpan **data penting dari tiap CV**, seperti:
    - Nama file
    - Isi CV
    - Jumlah keyword
    - Informasi ringkasan
    - Metadata lainnya

3.4 Contoh ilustrasi kasus.

Misalkan kita memiliki 2 CV dengan konten berikut dan kita ingin mencari kata kunci “cuisine” (Kata “cuisine” ditebalkan untuk contoh)

```
Name : Salman Halim
Address : 6th Street
Phone : 0888-888-8888
FOOD PREP CHEF
Skills

Highly skilled in cooking and preparing a variety of cuisines
Inborn ability to explore new cooking avenues

Accomplishments

Sandwich preparation experience
Knowledge of basic food preparation
Food handling knowledge
Italian cuisine
Asian cuisine
American cuisine
Ethnic cuisine preparation
Plate presentation skills
```

Banquet operations and off-site catering expert

Education
2011
MIT
Mathematics

Experience
Chez Gusteau 2020-2022
Head chef

Courses in Hospitality and Restaurant Management
Classes in Restaurant and Facility Operations
Basic Vocational : Prep Cook
Courses in: Food Preparation, Kitchen Management,Patisserie and Confectionery, International **Cuisine**

Name : Martana Zulkarnain
Address : 5th Street
Phone : 0888-888-8888
PROFESSIONAL RESTAURANT WAITER
Skills

Highly skilled in serving guest
Inborn ability to explore new management avenues

Accomplishments

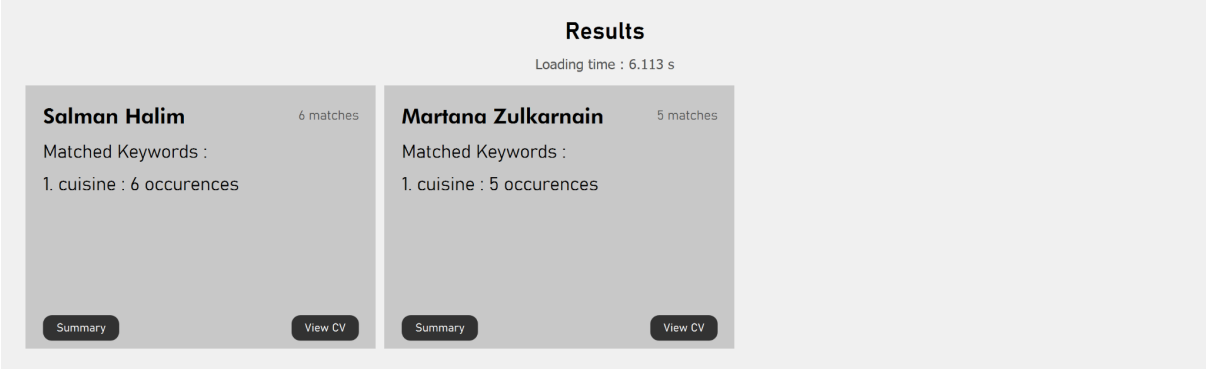
Knowledge of basic food preparation
Basic restaurant waitering
Food handling knowledge

Speciality
American **Cuisine**
French **Cuisine**
Asian **cuisine**
Ethnic **cuisine**
Italian **cuisine**

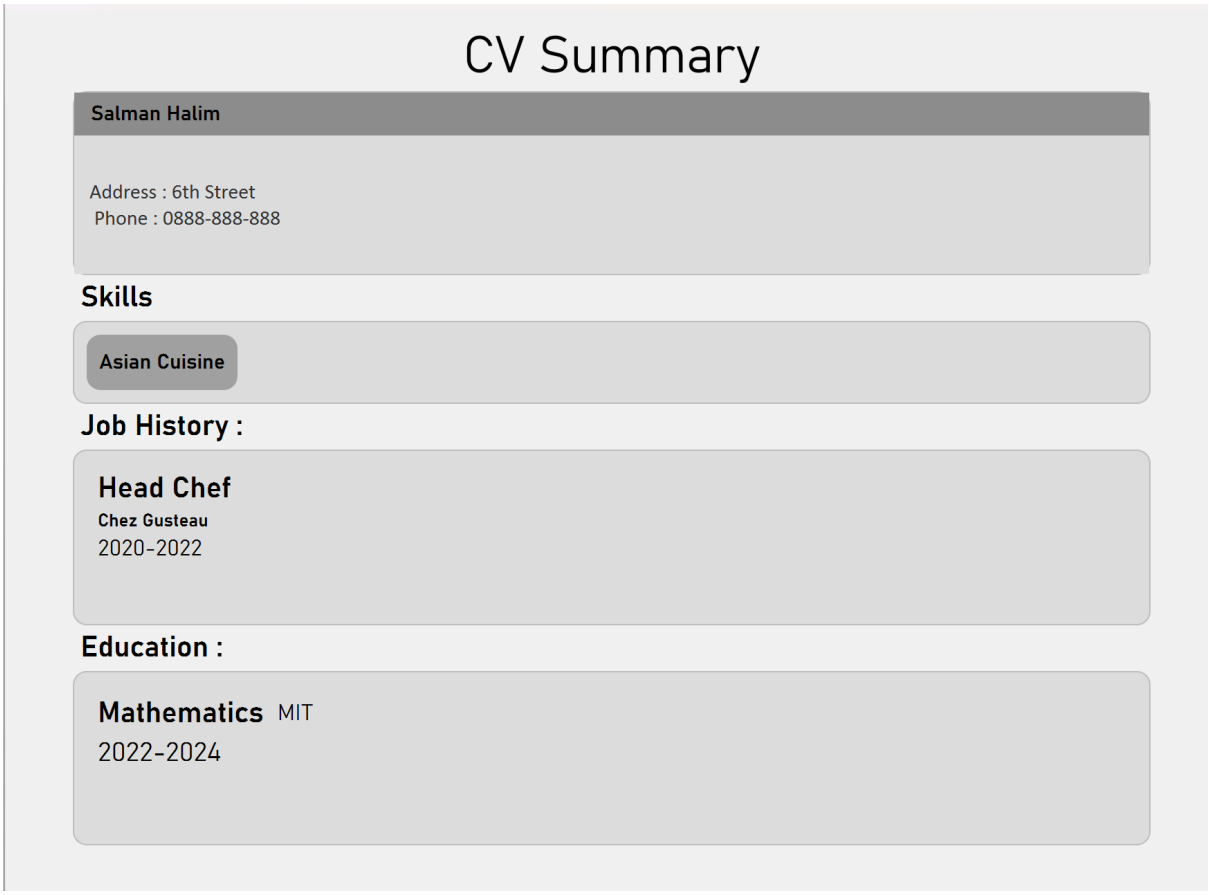
Education
2011
MIT
Mathematics

Experience :
Waiter for 2 years in Le French
Waiter for 2 years in La Italian

Jika kita mencari keyword “Italian, Catering, Restaurant”, maka aplikasi harus memunculkan kartu sebagai berikut :



Perhatikan bahwa hasil pencarian terurut berdasarkan banyak kemunculan keyword. Lalu, apabila kita membuka summary dari aplikan pertama, haruslah muncul laman seperti berikut.



Bab 4

Implementasi dan Pengujian

4.1 Spesifikasi Teknis Program

[Spesifikasi teknis program (struktur data, fungsi, dan prosedur yang dibangun).]

1. Struktur Data

1. pdf_data (`loader.load_pdf`)
   - Tipe data: `dict{int->dict}`
   - *Key*: detail_id (*primary key* dari tabel ApplicationDetail)
   - *Value*: {
     `"regex": str` (teks CV untuk regex)
     `"pattern_match": str` (teks CV satu baris, huruf kecil semua, siap diproses oleh algoritma *matching*)
     }

2. Hasil pencarian
   - Tipe data: `list[dict]`
   - Setiap elemen: {
     `"detail_id": int`
     `"applicant_name": str`
     `"total": int` (jumlah total match)
     `"keywords": dict{str->(occurrence:int, is_fuzzy:bool)}`
     }

3. Struktur *trie* (Aho-Corasick)
   - Node: objek dengan:
     - children: `dict{char->Node}`
     - fail: `Node (link fallback)`
     - output: `list[str]` (*pattern* yang berakhir di node ini)

4. Tabel pendukung algoritma
   – KMP: `lps_table: list[int]` (prefix‑function)
   – BM: `last_occur: dict{char->int}`
   – Levenshtein: dp: `list[list[int]]` (matrix ukuran (m+1)×(n+1))

2. Modul & Fungsi Utama

1. Utils / PDF Processing
   - `loader.load_pdf(conn)`
     - Ambil list `(detail_id, cv_path)`
     - Multiprocessing: ekstrak teks PDF → dua bentuk (regex, pattern_match)
     - Kembalikan dict `pdf_data`
   - `pdf_to_text_convert.convert_pdf_to_txt(...)`
     - Ekstrak teks via PyMuPDF
     - Simpan dua file txt (untuk pencarian menggunakan regex dan pencarian dengan *pattern matching*)
   - `seeder.seed_database(conn)`
     - Buat/migrasi tabel MySQL
     - Konversi pdf ke txt (opsional)
     - Generate data palsu (Faker), lalu dimasukkan ke *database*
2. Algoritma Pencarian
   - **KMP** - `kmp_search(text, pattern)`
   - **Boyer-Moore** - `bm_search(text, pattern)`
   - **Aho-Corasick** - `ac_search(text, patternList)`
   - **Fuzzy** (Levenshtein) - `similarity_score(str1, str2)`, `levenshtein_distance(str1, str2)`, `fuzzy_search(text, pattern, threshold=0.8)`

4.2 Tata Cara Penggunaan Program

4.2.1 Instalasi Program

1. *Clone* kode aplikasi

```
git clone https://github.com/RealNath/Tubes3_hr.git
```

2. Cd ke folder data, buat *database*

```
cd data
mysql -u root -p < cv_database.sql
```

3. Pastikan ada folder pdf yang berisi file cv (pdf), cd ke root

```
cd ..
```

4. Edit *password* pada file .env sesuai dengan password yang dimasukkan pada tahap 2 tadi

5. Lakukan *seeding database* (jika menggunakan dataset dari Kaggle)
   Windows: `python data/seeder.py`
   Linux: `python3 data/seeder.py`

6. Buat *virtual environment* lokal
   Windows: `python -m venv venv`
   Linux: `python3 -m venv venv`

7. Aktivasi *virtual environment*
   Windows: `venv\Scripts\activate`
   Linux: `source venv/bin/activate`

8. *Install library-library* Python yang dibutuhkan

```
pip install -r requirements.txt
```

9. Jalankan program utamanya
   Windows: `python main.py`
   Linux: `python3 main.py`

4.2.2. Cara Menggunakan Aplikasi

1. Dalam aplikasi, masukkan *keyword*, mode pencarian, dan banyak hasil yang ingin ditampilkan ke kotak prompt yang telah disediakan.

2. Tekan tombol "Search".

3. Setelah beberapa saat, akan muncul hasil pencarian yang sesuai dengan parameter yang dimasukkan. Tiap hasil akan dilengkapi dengan tombol untuk melihat *summary* CV dan tombol untuk membuka *file* PDF aslinya.

4.3 Hasil Pengujian

Pengujian ke-	Hasil Percobaan
1	Keyword : “Java, React”, Algoritma : KMP, Banyak : 3

2	Keyword : “Chef, Restaurant, Italian”, Algoritma : BM Banyak : 5
3	Keyword : “Powerpoint, forecast, excel, sap” Algoritma : AC Banyak : 2
4	Keyword : “Jawascript” Algoritma : KMP Banyak : 9

4.4 Analisis Hasil Pengujian

Pada hasil pengujian, dapat dilihat bahwa algoritma Boyer-Moore adalah algoritma pencarian tercepat untuk satu buah kata kunci. Namun, untuk beberapa kata kunci sekaligus, algoritma Aho-Corasick lebih cepat karena algoritma ini mampu mencari kumpulan kata kunci sekaligus dalam satu kali iterasi teks saja.

Bab 5
Kesimpulan dan Saran

5.1. Kesimpulan

Pada Tugas Besar 3 IF2211-Strategi Algoritma Semester 2 Tahun Ajaran 2024/2025, kami diminta untuk membuat suatu aplikasi berbasis Python dengan memanfaatkan algoritma pencarian string Knuth-Morris-Pratt, Boyer-Moore, dan Aho-Corasick dan pencarian dengan implementasi skor Levenshtein.

Kami berhasil membuat aplikasi yang dapat mencarikan CV-CV yang mengandung keyword terbanyak dengan metode-metode yang telah disebutkan. Dan membuat laman untuk summary dari CV tersebut dengan pencarian menggunakan regex. Semua ini dibuat dengan Python dan library PyQt5 untuk pembuatan aplikasi.

5.2. Saran

Pelaksanaan Tugas Besar 3 IF2211-Strategi Algoritma Semester 2 Tahun Ajaran 2024/2025 merupakan pengalaman berharga dan juga berkesan bagi kami. Dari pengalaman ini, selain belajar tentang macam-macam algoritma pencarian string, dan juga pembuatan aplikasi menggunakan PyQt5, dan juga perancangan UI dengan QT Designer. Kami juga ingin memberikan beberapa saran kepada pembaca yang mungkin akan menghadapi tugas serupa di masa depan atau tertarik untuk mengembangkan topik ini lebih lanjut:

1. Jadwalkan dan buat *roadmap* yang jelas pada pengembangan aplikasi. Karena aplikasi yang ingin dibangun secara *scalability*-nya menengah sampai tinggi, maka hal yang bijak jika dibangun suatu jadwal dan *roadmap* terlebih dahulu agar proses pengerjaan lebih terarah dan juga terstruktur.
2. Komunikasi antar anggota tim. Jika pengembangan aplikasi dilakukan dengan tim, maka komunikasi antar anggota sangat penting untuk mengetahui *progress* antar anggota dan juga menghindari suatu konflik dalam pekerjaan yang sedang dikerjakan.
3. Eksplorasi bahasa pemrograman dan *framework*. Library PyQt5 adalah library yang sangat luas dan penggunaannya harus dipelajari terlebih dahulu. Karena banyak mekanisme mekanisme yang mungkin bagi orang lain kurang familiar.

5.3. Refleksi

Ruang perbaikan dan pengembangan dalam membangun aplikasi ini dapat difokuskan pada beberapa aspek. Pertama, manajemen dan perencanaan waktu adalah bagian utama dalam membangun sebuah aplikasi dengan waktu hanya dua minggu yang diganggu dengan ulangan.

Selain itu, pemahaman terhadap spesifikasi yang diberikan juga harus diperhatikan. Memahami secara hati-hati dan menyeluruh tentang apa saja yang perlu dibangun, diminta, dan juga yang harus ada pada aplikasi. Hal ini dapat mencegah adanya risiko kesalahan pada pembuatan aplikasi dan juga aplikasi dapat berjalan sesuai dengan harapan. Selain itu, penempatan perhatian pada revisi spesifikasi juga harus selalu diperhatikan untuk menyesuaikan hal-hal yang perlu disesuaikan pada aplikasi.

Komunikasi tim yang berjalan dan baik juga harus dibangun dan bisa diperbaiki. Membuat sebuah ruang komunikasi untuk tim yang jelas, terbuka, dan inklusif di antara anggota tim untuk menghindari adanya miskomunikasi dan juga *update progress* dari hal yang telah dikerjakan oleh masing-masing. Diskusi yang diadakan secara reguler dan terjadwal terkait pembangunan dan pengembangan aplikasi juga dapat meningkatkan keterlibatan dan pemahaman untuk semua anggota tim.

Terakhir, evaluasi pada masing-masing anggota tim. Evaluasi dapat berbentuk umpan balik untuk antar anggota tim maupun dari asisten. Umpan balik ini dapat dimanfaatkan sebagai sarana pengembangan diri dari masing-masing anggota tim untuk perbaikan pada waktu selanjutnya dan juga sebagai media untuk terus berkembang. Selalu terbuka dan juga mau untuk mendengarkan saran serta komitmen untuk belajar dari setiap pengalaman juga dapat meningkatkan kinerja dan juga kompetensi yang dimiliki pada waktu selanjutnya.

LAMPIRAN

Tautan *repository* GitHub : Tubes3_hr

No	Poin	Ya	Tidak
1	Aplikasi dapat dijalankan.	✓	
2	Aplikasi menggunakan basis data berbasis SQL dan berjalan dengan lancar.	✓	
3	Aplikasi dapat mengekstrak informasi penting menggunakan Regular Expression (Regex).	✓	
4	Algoritma *Knuth-Morris-Pratt (KMP)* dan *Boyer-Moore (BM)* dapat menemukan kata kunci dengan benar.	✓	
5	Algoritma Levenshtein Distance dapat mengukur kemiripan kata kunci dengan benar.	✓	
6	Aplikasi dapat menampilkan *summary CV applicant*.	✓	
7	Aplikasi dapat menampilkan *CV applicant* secara keseluruhan.	✓	
8	Membuat laporan sesuai dengan spesifikasi.	✓	
9	Membuat bonus enkripsi data profil *applicant*.		✓
10	Membuat bonus algoritma Aho-Corasick.	✓	
11	Membuat bonus video dan diunggah pada Youtube.		✓

DAFTAR PUSTAKA

Munir, R. (2025). *Pencocokan string (*string matching*) dengan algoritma brute force, KMP, Boyer-Moore*. Homepage Rinaldi Munir. Retrieved May 13, 2025, from https://informatika.stei.itb.ac.id/~rinaldi.munir/Stmik/2024-2025/23-Pencocokan-string-(2025).pdf

ComputerBread (2025). *Ctrl+F on steroids - Aho-Corasick Algorithm (pt. 1)*. YouTube. Retrieved May 14, 2025, from https://www.youtube.com/watch?v=XWujo7KQL54