PENGEMBANGAN *KNOWLEDGE – BASED SYSTEM* MENGGUNAKAN METODE *KNOWLEDGE GRAPH* DALAM MENDIAGNOSIS PENYAKIT PARU – PARU

Proposal Tugas Akhir

Oleh

Clement Nathanael Lim
18222032

PROGRAM STUDI SISTEM DAN TEKNOLOGI INFORMASI
SEKOLAH TEKNIK ELEKTRO DAN INFORMATIKA
INSTITUT TEKNOLOGI BANDUNG

NOVEMBER 2025

LEMBAR PENGESAHAN

PENGEMBANGAN *KNOWLEDGE – BASED SYSTEM* MENGGUNAKAN METODE *KNOWLEDGE GRAPH* DALAM MENDIAGNOSIS PENYAKIT PARU – PARU

Proposal Tugas Akhir

Oleh

Clement Nathanael Lim
18222032

Program Studi Sistem dan Teknologi Informasi
Sekolah Teknik Elektro dan Informatika
Institut Teknologi Bandung

Proposal Tugas Akhir ini telah disetujui dan disahkan
di Bandung, pada tanggal 23 November 2025

Pembimbing

Dr. Nur Ulfa Maulidevi, S.T., M.Sc.

NIP. 197603092008012010

DAFTAR ISI

DAFTAR GAMBAR

DAFTAR TABEL

DAFTAR KODE

BAB I
PENDAHULUAN

I.1 Latar Belakang Masalah

Sektor kesehatan merupakan pilar fundamental dalam pembangunan kualitas hidup masyarakat. Seiring dengan kemajuan teknologi, tuntutan akan layanan kesehatan yang cepat, akurat, dan dapat diakses oleh semua kalangan semakin meningkat. Namun, di balik kemajuan tersebut, industri kesehatan masih menghadapi berbagai tantangan signifikan yang menghambat efisiensi dan efektivitas pelayanan. Kondisi ini menciptakan urgensi untuk mencari solusi inovatif yang dapat menjembatani kesenjangan antara kebutuhan pasien dan kapasitas penyedia layanan kesehatan.

Salah satu permasalahan utama yang telah lama menjadi perhatian adalah keterbatasan jumlah tenaga medis, khususnya dokter, dibandingkan dengan populasi yang harus dilayani. Berdasarkan studi oleh Trenggono dan Bachtiar (April 2023), rasio dokter dan pasien yang tidak seimbang menyebabkan beban kerja yang berlebihan pada dokter, waktu tunggu yang lama bagi pasien, dan potensi penurunan kualitas layanan akibat kelelahan. Masalah ini semakin terasa dalam era digital, di mana *platform telemedicine* seperti Halodoc menjadi populer. Data dari Halodoc menunjukkan bahwa dokter tidak selalu dapat melayani pasien secara paralel atau *real-time* karena sedang melayani pasien lain, sehingga mengurangi aksesibilitas konsultasi instan yang diharapkan. Selain tantangan pada aspek pelayanan, dunia medis juga dihadapkan pada isu efektivitas pengobatan. Pendekatan pengobatan yang seringkali bersifat umum terbukti kurang efektif untuk pasien dengan kondisi klinis yang unik dan spesifik, semisal untuk mengobati kanker, di mana terkadang pengobatan seperti kemoterapi tidak cocok untuk diterapkan kepada semua pasien yang mengalami kanker. Proses untuk mengembangkan pengobatan yang dipersonalisasi memakan waktu yang sangat lama dan biaya yang besar, padahal hal ini krusial untuk hasil terapi yang optimal (Schork 2019). Oleh karena itu, diperlukan suatu sistem yang dapat mendeteksi penyakit kepada pasien, dengan menggunakan bantuan pakar (*expert system*), dengan metode *knowledge graph* dalam melakukan representasi pengetahuan.

Knowledge – based system, dalam kasus ini adalah *expert system*, masih relevan untuk digunakan, terutama dalam melakukan serangkaian pemecahan masalah. Saat ini, *expert system* masih menjadi salah satu aplikasi utama dari AI di dalam

bidang kesehatan (Akil 2020). *Knowledge – based system* sendiri pada dasarnya akan membantu dokter dalam melakukan diagnosis, di mana sistem akan melakukan proses diagnosis hingga dapat menghasilkan suatu keputusan. Keputusan tersebut nantinya yang akan menjadi bahan pertimbangan untuk dokter dalam menentukan diagnosis final dari penyakit paru – paru yang diderita oleh pasien.

Penyakit paru – paru merupakan penyakit di bidang kesehatan yang tepat untuk menjadi domain pengembangan *Knowledge – based system*. Berdasarkan data dari Hello Sehat tahun 2023, penyakit paru – paru, antara lain tuberkulosis dan PPOK (Penyakit Paru Obstruktif Kronik), menjadi 10 besar penyakit paling mematikan di Indonesia. Untuk tuberkulosis sendiri, menurut data WHO tahun 2014, jumlah kematian akibat TBC terus meningkat, bahkan diperkirakan lebih dari 100.000 kasus setiap tahunnya.

Dengan demikian, *Knowledge – based system* untuk mendiagnosis penyakit paru – paru masih penting dan relevan. Dengan masifnya perkembangan teknologi, *Knowledge – based system* juga masih terus berkembang hingga sekarang. Saat ini, representasi pengetahuan dari KBS sendiri terdiri atas beberapa macam representasi yang dapat dilakukan, antara lain menggunakan *rule – based* (representasi berdasarkan sekelompok aturan), *formal logic* (representasi berdasarkan logika proposisional), *knowledge graph* (representasi berdasarkan keterhubungan antar entitas), maupun menggunakan *machine learning*.

Dibandingkan dengan *rule – based* maupun dengan *formal logic*, *knowledge graph* unggul baik dalam sisi teknis maupun dalam sisi implementasi. Hal ini disebabkan adanya fleksibilitas dalam menghubungkan *edges* dengan *node* (dalam kasus ini gejala – gejala) dari penyakit paru – paru). Hal ini tentu sangat sulit jika menggunakan *rule – based* maupun *logic* karena akan melibatkan banyak sekali aturan yang saling tumpang tindih. Penggunaan *knowledge graph* yang dibangun dengan baik dapat digunakan untuk berbagai aplikasi *downstream* dan dapat membuat model memiliki kemampuan *reasoning* yang masuk akal (Ji dkk. 2020). Dengan demikian, *knowledge graph* menjadi teknik yang paling optimal untuk menjadi representasi pengetahuan untuk *knowledge – based system*. Untuk representasi menggunakan *machine learning* tidak akan dilakukan disebabkan batasan dari tugas akhir ini untuk tidak mengembangkan *learning agent* dalam KBS ini.

Penggunaan *knowledge – based agent* dalam bidang kesehatan juga sudah banyak

dilakukan. Beberapa pengembangan dalam beberapa tahun terakhir antara lain digunakan untuk mendeteksi kehamilan, penyakit COVID - 19, stroke, dan sebagainya.

I.2 Rumusan Masalah

Berdasarkan latar belakang tersebut, keterlibatan sistem pakar diperlukan untuk memperoleh sekumpulan pengetahuan yang dibutuhkan untuk menyelesaikan masalah kesehatan yang akan ditentukan kemudian. Pakar tersebut, dalam hal ini akan menargetkan dokter, berguna untuk menguji kebenaran solusi yang diusulkan oleh sistem. Permasalahan yang ingin dijawab adalah bagaimana merepresentasikan sekumpulan pengetahuan yang didapat oleh sistem pakar, menjadi suatu sistem cerdas yang dapat menjawab pertanyaan dari pasien itu sendiri. Terdapat beberapa hal yang dapat menjadi perumusan masalah, antara lain:

1. Bagaimana representasi *knowledge graph* dalam membangun suatu *knowledge – based system* untuk domain kesehatan?
2. Bagaimana menyusun *knowledge graph* dengan *input* pengetahuan yang berasal dari pakar?
3. Bagaimana kinerja dari *knowledge – based system* yang memanfaatkan *knowledge graph*?

I.3 Tujuan

Tujuan dari Tugas Akhir ini adalah untuk mengimplementasikan pengetahuan – pengetahuan dari suatu sistem pakar, agar dapat menghasilkan suatu solusi maupun jawaban dari setiap pertanyaan yang diajukan oleh pasien. Untuk solusi yang nanti dihasilkan adalah diagnosis dari penyakit paru – paru berdasarkan gejala yang dimasukkan oleh pasien.

I.4 Batasan Masalah

Pada tugas akhir ini, akan diberikan batasan – batasan sebagai berikut:

1. Pengumpulan basis pengetahuan untuk *knowledge – based system* diakuisisi dari pakar.
2. *Knowledge – based system* yang dikembangkan tidak memiliki *learning component* / *learning agent*, sehingga sistem tidak dapat memperbaharui *dataset* maupun pengetahuan yang dimiliki.

I.5 Metodologi

Beberapa tahapan yang digunakan untuk menyelesaikan tugas akhir, antara lain:

1. Investigasi

   Tahapan investigasi dilakukan dengan cara mencari literatur berupa *paper* terkait dengan *knowledge – based system*, *expert system*, AI *in Healthcare*, serta dengan laporan Tugas Akhir mahasiswa terdahulu.

2. Analisis Pembangunan Solusi

   Terdapat beberapa hal yang diperlukan dalam membangun *knowledge – based system*, antara lain:

   1) Identifikasi

      i. Pada tahap ini, akan ditentukan analisis kebutuhan awal dari KBS, yakni tujuan, batasan, lingkungan penggunaan, dan sebagainya.

      ii. Selain itu, akan dilakukan uji kelayakan untuk memastikan apakah suatu domain layak untuk dijadikan *knowledge – based system* atau tidak. Dalam kasus ini, akan dilakukan analisis kelayakan yang dilakukan oleh seorang dokter spesialis yang akan menjadi pakar dalam pengembangan sistem ini.

   2) Konseptualisasi

      i. Pada tahap ini, akan mulai dilakukan wawancara dengan pakar untuk mendapatkan pengetahuan yang dimiliki dan mengorganisasikan fakta yang didapat.

   3) Formalisasi

      i. Pada tahap ini, akan dilakukan representasi pengetahuan dalam bentuk *knowledge graph* berdasarkan pengetahuan yang sudah didapat dari sistem pakar.

   4) Implementasi

      i. Pada tahap ini, prototipe dari *knowledge graph* akan dihasilkan.

   5) Pengujian

i. Pada tahap ini, akan dilakukan pengujian dari sistem yang sudah dibangun, baik dari pakar maupun calon pengguna sistem.

BAB II
STUDI LITERATUR

II.1 Sistem Pakar

Sistem pakar (*expert system*) merupakan salah satu cabang dari *knowledge – based system*, di mana KBS sendiri memiliki representasi pengetahuan yang cukup umum, bisa melalui *indirect acquisition* (yakni menggunakan pengumpulan data historis,*workshop*, diskusi kelompok, observasi lapangan, studi literatur, dan sebagainya), maupun *direct acquisition*, yakni akuisisi yang melibatkan pakar. Dalam kasus ini, akuisisi dapat dilakukan dengan menggunakan konsultasi dengan dokter, dimana dokter dapat memberikan parameter maupun gejala – gejala untuk suatu penyakit tertentu.

Akuisisi langsung (*direct acquisition*) merupakan pendekatan fundamental dalam pembangunan sistem pakar di mana pengetahuan diekstraksi secara manual melalui interaksi langsung antara seorang *knowledge engineer* dengan seorang pakar. Metode utamanya adalah wawancara / diskusi dengan pakar menggunakan berbagai cara seperti pemaparan pengetahuan yang dimiliki pakar maupun diskusi berdasarkan kasus - kasus tertentu. Dengan akusisi langsung, *knowledge engineer* berusaha melakukan proses penalaran (*reasoning*) untuk dapat menggali pengetahuan yang lebih optimal dari pakar tersebut.

Untuk melengkapi dan memvalidasi informasi dari pakar, pendekatan ini juga memerlukan studi literatur yang intensif, di mana pengembang sistem harus mempelajari buku teks, jurnal ilmiah, dan manual untuk membangun fondasi pengetahuan domain yang kokoh. Selain itu, observasi terstruktur dapat digunakan untuk mengamati pakar saat bekerja di lingkungan nyata, memungkinkan penangkapan pengetahuan prosedural dan kontekstual yang mungkin terlewat dalam wawancara.

Jika akuisisi langsung memerlukan interaksi secara intensif dengan *expert*, berbeda halnya dengan akuisisi tidak langsung (*indirect acquisition*). *Indirect acquisition* menggunakan data dan observasi untuk mengurangi subjektivitas dan mempercepat proses. Metode ini mencakup observasi lapangan, *workshop*, studi literatur, dan diskusi kelompok untuk memahami konteks kerja pakar atau mensintesis pengetahuan dari beberapa ahli sekaligus (Leake December 1991).

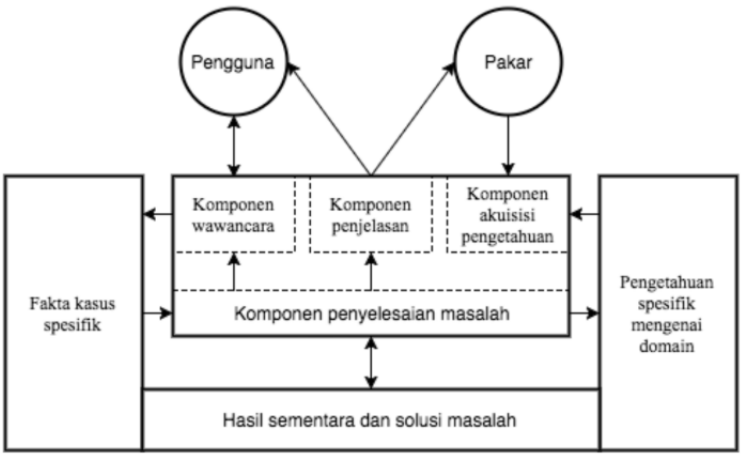
Pendekatan yang lebih modern adalah menggunakan pengumpulan data historis, di mana algoritma *machine learning* seperti ID3 yang dipelopori oleh Quinlan (1986), diterapkan untuk menginduksi atau menemukan aturan-aturan (misalnya, gejala-penyakit) secara otomatis dari ribuan contoh kasus. Pada praktiknya, pendekatan campuran yang mengombinasikan beberapa metode ini, seperti memulai dengan studi literatur, dilanjutkan dengan wawancara pakar, dan divalidasi dengan data historis, merupakan strategi yang paling efektif untuk membangun sistem pakar yang komprehensif dan andal.

Sistem pakar yang akan dikembangkan ini pada dasarnya adalah KBS dengan representasi pengetahuan menggunakan seorang *expert* / pakar dengan kombinasi menggunakan data historis. Dalam kasus ini, pakar yang dimaksud adalah orang yang berpengalaman dalam permasalahan tersebut, yakni menggunakan dokter sebagai pakarnya, disebabkan dokter mampu memberikan gejala – gejala dari suatu penyakit tertentu, sehingga untuk representasi pengetahuannya dapat menjadi lebih jelas.

Arsitektur sistem pakar berbasis KBS terdiri atas 2 modul, yakni:

1. *Problem solving component*, yang terdiri atas *interviewer* dan *explanation component*.
2. Basis pengetahuan, yang terdiri atas *case – specific facts*, *intermediate results*, *problem solution*, serta *knowledge* yang didapat dari hasil *learning*.

Arsitektur umum dari sistem pakar antara lain seperti gambar II.1:



Gambar II.1 Arsitektur umum dari *knowledge – based system* (Puppe 1993)

Berdasarkan Barrett, Jones, dan Thompson (1992), proses penyusunan *expert system* antara lain sebagai berikut:

1. Identifikasi

   (a) Tahap identifikasi adalah langkah analisis kebutuhan awal dengan tujuan menentukan kebutuhan eksternal, format *input* dan *output*, lingkungan penggunaan, dan pengguna akhir dari sistem.

   (b) Selain menentukan tujuan, akan diidentifikasi juga pakar yang akan digunakan, sumber daya, dan waktu.

   (c) *Constraints* / batasan juga menjadi aspek penting untuk membatasi domain masalah agar sistem bisa menargetkan satu bagian tertentu saja secara spesifik untuk dilakukan analisis yang mendalam.

   (d) Pada tahapan ini, diperlukan juga uji kelayakan untuk memastikan bahwa *knowledge – based system* yang dikembangkan benar – benar diperlukan.

2. Konseptualisasi

   (a) Tahap konseptualisasi merupakan proses desain awal program dibuat, dengan fokus untuk mengidentifikasi konsep, hubungan antar sub – sistem, dan mekanisme kontrol (alur penalaran) dalam domain masalah.

   (b) *Knowledge engineer* akan menanyakan pertanyaan-pertanyaan mendalam untuk memahami cara kerja pakar, seperti:

      i. Keputusan apa yang dibuat oleh pakar?

      ii. Informasi (*input*) apa yang dibutuhkan untuk membuat keputusan tersebut?

      iii. Kondisi apa saja yang mengarah pada hasil tertentu?

   (c) Tujuannya adalah untuk mendekomposisi proses pengambilan keputusan pakar, mulai dari yang kompleks menjadi komponen-komponen yang lebih kecil dan dapat dipahami.

3. Formalisasi

   (a) Tahap formalisasi merupakan tahap di mana konsep, sub-masalah, dan

alur informasi yang telah diidentifikasi sebelumnya diorganisasikan ke dalam representasi formal (misalnya aturan, graf, atau *decision tree*). Dalam kasus ini, representasi pengetahuan yang akan digunakan adalah graf pengetahuan / *knowledge graph*.

(b) Dalam membuat suatu *knowledge graph*, sistem akan melakukan penelurusan (traversal) graf dari satu *node* ke *node* lain melalui serangkaian hubungan (*edge*). Keberadaan, panjang, atau jenis jalur ini menjadi dasar keputusan.

(c) Selain mencari jalur, dapat dilakukan juga penalaran berbasis aturan (*rule – based reasoning*), di mana akan didefinisikan suatu aturan logis (semisal dalam format IF – ELSE) untuk dapat menyimpulkan hubungan atau properti baru yang tidak ada secara eksplisit / kompleks.

4. Implementasi

(a) Tahap implementasi merupakan tahap di mana pengetahuan yang telah diformalkan (aturan, struktur, alur kontrol) ditransformasikan ke dalam *software* atau *tools* pengembangan, di mana hasil dari tahap ini adalah sebuah prototipe sistem pakar yang berfungsi (*working prototype*).

(b) Sistem harus menyertakan penjelasan dari sistem yang sudah dikembangkan, dengan tujuan untuk membantu pengguna memahami pertanyaan yang diajukan oleh sistem, memungkinkan pengguna memanfaatkan *output* atau rekomendasi secara efektif, serta menunjukkan kepada pengguna alur logika bagaimana sistem sampai pada sebuah kesimpulan, untuk membangun kepercayaan.

(c) Dalam pengembangan ini, sistem juga dapat diintegrasikan dengan basis data rekam medis elektronik (RME) untuk dapat menarik data terkait pasien yang dianalisis oleh sistem.

5. Pengujian

(a) Pengujian bertujuan untuk menguji kebenaran pengetahuan dan kegunaan sistem secara keseluruhan.

(b) Terdapat 3 aktivitas utama dalam pengujian ini, antara lain:

i. Verifikasi: Verifikasi akan memastikan bahwa model dan aturan di dalam program secara akurat mencerminkan pengetahuan yang

sebenarnya. Hal ini dapat diuji dengan meminta pakar menguji sistem untuk semua kemungkinan skenario untuk memastikan logikanya benar dan tidak ada yang terlewat.

ii. Validasi: Validasi akan memastikan bahwa program yang dibangun benar-benar menyelesaikan masalah yang dimaksud dan *output*–nya sesuai dengan solusi dari pakar manusia.

iii. Evaluasi: Evaluasi akan menilai kegunaan dan nilai praktis dari sistem di dunia nyata setelah diimplementasikan.

(c) Diperlukan juga *user acceptance testing* (UAT) untuk memastikan bahwa pengguna memahami alur dan kinerja dari sistem.

II.2 *Rule – based Expert System*

Salah satu *tools* untuk mengembangkan sistem pakar adalah CLIPS. CLIPS (*C Language Integrated Production System*) adalah kakas pengembangan sistem pakar, ditulis dalam bahasa C, yang menyediakan sarana untuk membuat konstruksi berbasis aturan dan/atau objek (Riley 1999). Beberapa komponen utama dari CLIPS, antara lain:

1. *Fact List*

   (a) Merupakan *working memory* dari sistem.

   (b) Berisi semua fakta atau data yang diketahui oleh sistem pada saat itu. Fakta ini bisa berasal dari *input* pengguna atau hasil dari eksekusi aturan sebelumnya.

   (c) Contoh fakta: (Pasien mengalami batuk), (Suhu pasien 39,5ºC).

2. *Knowledge – Base*

   (a) Merupakan sistem pakar yang dibangun.

   (b) Berisi kumpulan aturan-aturan dalam format IF–THEN. Aturan-aturan ini disebut *productions*.

3. *Inference Engine* / Mesin Inferensi

   (a) Merupakan bagian yang akan melakukan penalaran.

   (b) Tugasnya adalah melakukan pengecekan terus-menerus dengan membandingkan fakta di *Fact List* dengan kondisi dari semua aturan di

Knowledge Base.

Berikut ini merupakan contoh dari kode CLIPS untuk mendeteksi penyakit paru – paru:

Kode II.1 Contoh Kode CLIPS

```
(defrule rule-diagnosis-tb
  "Aturan untuk mendeteksi kemungkinan Tuberkulosis (TB)"
  ?f <- (pasien (jenis-batuk dahak-kronis)
            (keringat-malam ya)
            (diagnosis belum-diketahui))
  =>
  (printout t "Aturan TB terpicu: Gejala batuk kronis dan
     keringat malam cocok." crlf)
  (modify ?f (diagnosis Tuberkulosis))
)

(defrule rule-diagnosis-pneumonia
  "Aturan untuk mendeteksi kemungkinan Pneumonia"
  ?f <- (pasien (jenis-batuk dahak-akut)
            (demam ya)
            (diagnosis belum-diketahui))
  =>
  (printout t "Aturan Pneumonia terpicu: Gejala batuk dahak
     akut dan demam cocok." crlf)
  (modify ?f (diagnosis Pneumonia))
)

(defrule rule-diagnosis-ppok
  "Aturan untuk mendeteksi kemungkinan PPOK"
  ?f <- (pasien (jenis-batuk dahak-kronis)
            (perokok-berat ya)
            (usia ?u &:(> ?u 40))
            (diagnosis belum-diketahui))
  =>
  (printout t "Aturan PPOK terpicu: Gejala batuk kronis,
     perokok, dan usia > 40 cocok." crlf)
  (modify ?f (diagnosis PPOK))
)
```

II.3 *Knowledge Graph*

Knowledge Graph (KG) merupakan sebuah cara merepresentasikan pengetahuan di dalam *knowledge based system*, di mana informasi disimpan sebagai jaringan

entitas berbentuk graf, dengan *node* untuk merepresentasikan entitasnya (seperti "Diabetes Tipe 2", "Metformin") yang saling terhubung dengan *edges* / sisi oleh relasi spesifik (seperti "diobatiDengan", "memilikiGejala"). Berbeda dengan aturan IF-THEN pada *rule – based expert system*, *knowledge graph* menangkap hubungan yang lebih kompleks, sehingga memungkinkan KBS untuk melakukan penalaran yang lebih efektif, misalnya menemukan hubungan tersembunyi antara obat dan efek samping melalui jalur koneksi. Fleksibilitas ini membuat *knowledge graph* sangat ideal untuk domain kompleks seperti kesehatan, di mana pengetahuan terus berkembang dan data berasal dari berbagai sumber yang terpisah.

Berdasarkan Nicholson dan Greene (2020), penggunaan *knowledge graph*, terutama untuk bidang biomedis, mampu untuk menghubungkan data mutasi genetik seorang pasien dari sekuens genom, dengan jalur protein yang terdampak dari basis data biomedis yang ada, lalu ke artikel penelitian terbaru yang membahas jalur tersebut, dan akhirnya ke uji klinis yang relevan. Dengan demikian, *knowledge graph* mampu untuk mengubah kumpulan fakta yang disampaikan oleh *user* menjadi sebuah graf kompleks yang dapat dieksplorasi oleh mesin, sehingga memungkinkan aplikasi canggih seperti penemuan obat (*drug discovery*) dengan mengidentifikasi target obat baru melalui analisis jalur dalam graf, dan *personalized medicine*, di mana sistem dapat merekomendasikan terapi yang paling sesuai dengan menganalisis graf unik yang merepresentasikan profil molekuler dan klinis seorang pasien.

Berikut ini merupakan contoh *knowledge graph* untuk penyakit kanker paru – paru:

Gambar II.2 Contoh *knowledge graph* untuk kanker paru – paru.

Berikut ini merupakan langkah – langkah merancang *knowledge graph* berdasarkan referensi dari Stegeman (2025):

1. Merancang *blueprint* pengetahuan dari graf.
   (a) Menentukan entitas (*nodes*) utama seperti penyakit, gejala, faktor risiko, pemeriksaan, dan obat.
   (b) Menentukan hubungan (*edges* antar entitas, seperti "memiliki gejala", "disebabkan oleh", "didiagnosis dengan", dan sebagainya).
2. Mengisi graf pengetahuan.
   (a) Akan dilakukan pengumpulkan data dan fakta dari berbagai sumber untuk mengisi graf pengetahuan.
   (b) Pada kasus ini, akan dilakukan validasi dengan pakar, yakni dokter spesialis paru untuk memverifikasi hubungan dan menambahkan pengetahuan praktis yang tidak tertulis.
3. Membangun graf pengetahuan.

(a) Merupakan tahap rekayasa perangkat lunak di mana akan dimasukkan semua data yang telah dikumpulkan ke graf pengetahuan.

(b) Untuk hal tersebut, akan dipilih basis data graf, seperti Neo4j.

4. Evaluasi graf pengetahuan.

(a) Setelah graf dibuat, akan diajukan beberapa pertanyaan (*query*) berdasarkan data pasien untuk mendapatkan kemungkinan diagnosis.

(b) Selanjutnya akan ditampilkan daftar kemungkinan diagnosis yang diurutkan dari yang paling mungkin hingga yang kurang mungkin, beserta alasan (jalur di dalam graf) mengapa kesimpulan itu diambil.

(c) Terakhir, akan dilakukan validasi untuk memastikan diagnosis yang diberikan oleh *knowledge graph* sudah sesuai dengan pengetahuan dari dokter spesialis tersebut.

II.4 Penelitian Terkait

Expert system merupakan salah satu aplikasi dari AI yang cukup sering digunakan di dalam bidang kesehatan. Beberapa penelitian terkait dengan sistem pakar, antara lain untuk sistem pakar untuk mendeteksi penyakit COVID-19 (Fadli dkk. 2024), sistem pakar untuk mendeteksi *stroke* (Yimenu, Adege, dan Yitagesu October 2025), beserta sistem pakar untuk mendeteksi kehamilan (Aditia dkk. May 2023).

Penelitian terkait dengan *knowledge graph* dalam bidang kesehatan juga akan diulas pada bagian ini, sehingga akan memberikan tinjauan yang lebih komprehensif terkait penggunakan *knowledge graph* dalam bidang kesehatan.

II.4.1 Sistem Pakar untuk COVID – 19 (Fadli dkk. 2024)

Pada jurnal ini, sistem pakar yang dianalisis menggunakan metode *forward chaining* dalam menentukan konklusi dari fakta – fakta yang dimasukkan oleh pasien, dimana sistem bekerja dengan cara menelusuri data dari fakta-fakta yang ada (gejala yang dimasukkan oleh pengguna) menuju sebuah kesimpulan (diagnosis penyakit). Prosesnya dimulai dari aturan "IF" (jika pengguna mengalami gejala X), lalu bergerak menuju kesimpulan "THEN" (maka kemungkinan diagnosisnya adalah Y).

Metode *knowledge based* yang digunakan memiliki teknik untuk menangani ketidakpastian dalam diagnosis (misalnya, tingkat keyakinan pengguna terhadap

gejala yang dialami), yakni menggunakan *certainty factor*. Metode ini menghitung nilai keyakinan (dalam bentuk persentase) untuk setiap kemungkinan diagnosis berdasarkan jawaban pengguna dan bobot yang telah ditentukan oleh pakar. Berikut ini merupakan contoh tabel nilai *certainty factor* per diagnosis yang diberikan:

Tabel II.1 Contoh tabel *certainty factor* yang dibangun (Fadli dkk. 2024).

Code	Symptom Name	Diagnosis		
		D001	D002	D003
G001	Cough	0.4		
G002	Have a cold	0.4		
G003	Fever	0.2		
G004	Fatigue	0.2		
G005	Headache	0.2		
G006	Loss of Appetite (Anorexia)	0.4		
G007	Loss of Sense of Smell (Anosmia)		0.6	
G008	Loss of Sense of Taste (Ageusia)		0.6	
G009	Sore throat		0.2	
G010	Coughs and colds accompanied by shortness of breath		0.6	
G011	Asthma		0.6	
G012	Red Eyes/Irritation		0.2	
G013	Diarrhea		0.2	
G014	Muscle ache		0.2	
G015	Pulmonary Hypertension			0.8
G016	Diabetes Mellitus			0.4
G017	Heart failure (Decompression of the Heart)			0.8
G018	Pneumonia (Pneumonia)			0.8
G019	Pain in the Chest			0.6
G020	Hard to breath			0.6

Akuisisi pengetahuan untuk sistem ini diperoleh dari studi literatur, basis data medis, dan wawancara dengan pakar dari Puskesmas Praya yang telah menangani kasus COVID-19. Proses pengujian dilakukan dengan menggunakan metode *Black-Box Testing* untuk memastikan fungsionalitasnya berjalan sesuai yang diharapkan tanpa melihat kode internalnya.

Sistem pakar yang dibuat dirancang untuk mendiagnosis COVID-19 ke dalam tiga kategori, yakni negatif, reaktif, dan positif. Pengetahuan yang didapat dari pakar diterjemahkan ke dalam basis aturan (*production rules*) yang mencakup 20 gejala yang terkait dengan COVID-19. Pengguna kemudian berinteraksi dengan sistem melalui serangkaian pertanyaan mengenai gejala yang mereka rasakan. Jawaban pengguna (misalnya, "Sangat Yakin", "Yakin", "Tidak Yakin") kemudian dikonversi menjadi nilai numerik. Sistem kemudian melakukan perhitungan menggunakan metode *forward chaining* dan *certainty factor* untuk menghasilkan

persentase kemungkinan dari ketiga diagnosis tersebut. Hasil yang didapatkan antara lain sebagai berikut:

Tabel II.2 Hasil kalkulasi manual dan kalkulasi sistem (Fadli dkk. 2024).

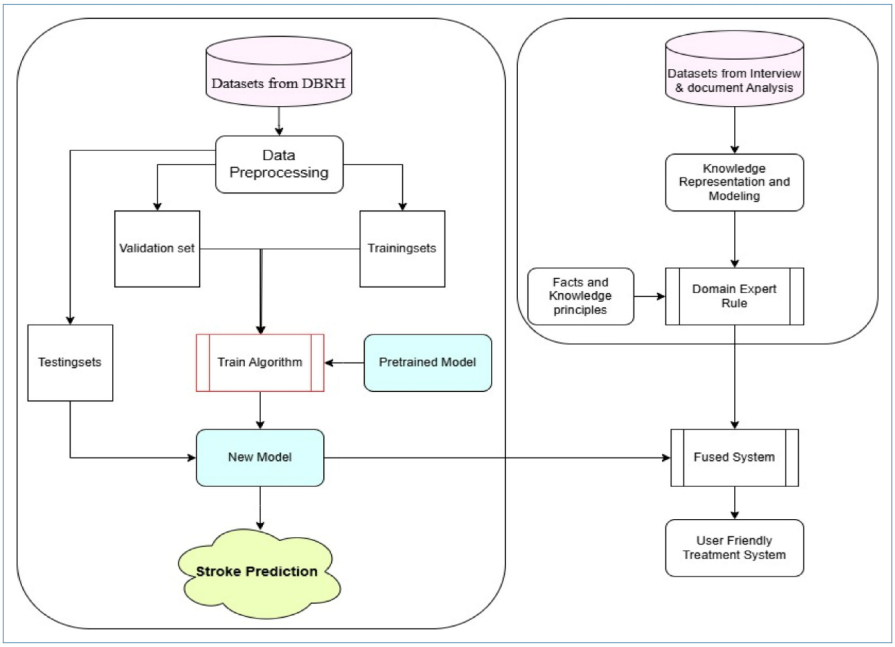
Diagnosis	**The calculation results**	
	Manual Calculation	**System Calculation**
Negative	82%	82%
Reactive	7%	8%
Positive	12%	12%
Diagnostic Results	**Negative**	**Negative**

Berdasarkan data hasil diagnosis antara perhitungan manual dan perhitungan sistem, hasil perhitungan sistem sangat akurat dan konsisten dengan perhitungan manual, yakni diagnosis negatif 82%, reaktif 8%, dan positif 12%.

II.4.2 Sistem Pakar untuk Mendeteksi Stroke (Yimenu, Adege, dan Yitagesu October 2025)

Pada jurnal ini, sistem pakar yang dibangun dikombinasikan dengan *machine learning*. Tujuannya adalah untuk menciptakan alat pendukung keputusan yang kuat bagi tenaga medis, memungkinkan mereka membuat diagnosis dan rencana perawatan stroke yang lebih baik, bahkan tanpa masukan langsung dari spesialis.

Berikut ini merupakan arsitektur yang digunakan untuk merancang sistem pakar tersebut:



Gambar II.3 Arsitektur sistem pakar menggunakan *machine learning* (Yimenu, Adege, dan Yitagesu October 2025)

Berdasarkan arsitektur sistem pakar tersebut, pada bagian kiri, alur *machine learning* memproses data kuantitatif pasien melalui tahap *preprocessing* dan pelatihan algoritma untuk menghasilkan sebuah *new model* yang mampu melakukan *stroke prediction* berbasis data. Sementara itu, pada bagian kanan, alur sistem pakar mengekstraksi pengetahuan kualitatif dari wawancara dan analisis dokumen, lalu merepresentasikannya menjadi *domain expert rule* yang logis menggunakan *rule – based*. Hasil dari arsitektur ini adalah model prediktif dari *machine learning* dan aturan berbasis pengetahuan dari pakar yang digabungkan menjadi sebuah *fused system*. Sistem gabungan inilah yang pada akhirnya menghasilkan sebuah *user friendly treatment system* yang komprehensif, memberikan rekomendasi yang tidak hanya akurat secara statistik tetapi juga andal dan dapat dijelaskan secara klinis.

Untuk komponen *machine learning*, digunakan data pasien yang dikumpulkan dari rumah sakit dan *dataset* publik di *platform* Kaggle. Beberapa model *machine learning* yang dievaluasi, antara lain *decision tree*, *random forest*, dan *support vector machine* (SVM), yang diimplementasikan menggunakan Python. Untuk pemrosesan data, dilakukan seleksi Fitur untuk mengidentifikasi faktor risiko paling signifikan (seperti usia, kadar glukosa). Selain itu, digunakan juga SHAP (*Shapley Additive Explanations*), untuk membuat model lebih dapat diinterpretasikan, yakni menjelaskan mengapa model membuat prediksi tertentu,

dan SMOTE (*Synthetic Minority Over – sampling Technique*) untuk menangani ketidakseimbangan data.

Setelah dilakukan pembuatan model, didapatkan hasil percobaan sebagai berikut:

Tabel II.3 Hasil *training model* untuk KBS (Yimenu, Adege, dan Yitagesu October 2025).

Model	Feature	K-fold technique	Precision	recall	F1-score	Accuracy	Macro avg
Decision tree	All	No	90	89.5	90	90	90
	All	Yes	92	92	92	92	92
	Selected	Yes	98	97.5	97.8	97.5	97.5
Random forest	All	No	92.5	90	91	91	91
	All	Yes	98.5	98.5	99	99	99
	Selected	Yes	99.4	99.4	99.4	99.4	99.4
SVM	All	No	76	75.5	75.5	75	76
	All	Yes	75	76	75	75	75
	Selected	Yes	82	80.8	87.7	77	80.8

Dari evaluasi model *machine learning*, didapatkan model *random forest* menunjukkan performa terbaik dengan akurasi yang sangat tinggi, yaitu 99.4%, setelah melalui proses validasi silang (*k-fold cross - validation*). Selain itu, analisis menggunakan SHAP mengonfirmasi bahwa usia dan kadar glukosa adalah prediktor paling kuat untuk risiko stroke. Risiko stroke meningkat secara signifikan seiring bertambahnya usia dan kadar glukosa yang tinggi. Sistem gabungan ini juga berhasil menggabungkan *rule – based system* dari Prolog (*Programming in logic*) dengan kemampuan prediksi dari model *random forest*. Hasilnya adalah sistem pendukung keputusan yang tidak hanya akurat tetapi juga dapat memberikan penjelasan logis. Sistem ini diuji oleh para profesional medis (ahli saraf dan dokter umum), yang mengonfirmasi efektivitas dan kegunaannya sebagai alat bantu diagnosis dan perawatan stroke di lingkungan klinis.

II.4.3 Sistem Pakar untuk Mendeteksi Kehamilan (Aditia dkk. May 2023)

Pada jurnal ini, sistem pakar yang dianalisis menggunakan metode *forward chaining* dengan cara mengumpulkan fakta-fakta (gejala yang dimasukkan oleh pengguna) terlebih dahulu, kemudian menelusuri aturan-aturan yang ada untuk sampai pada sebuah kesimpulan (diagnosis penyakit). Pengetahuan mengenai gejala dan penyakit kehamilan dikumpulkan melalui studi literatur dan konsultasi dengan pakar (dokter) untuk membangun aturan-aturan dalam sistem. Sistem ini dikembangkan sebagai aplikasi *web* agar mudah diakses oleh siapa saja yang memiliki koneksi internet.

Hasil dari sistem akan menampilkan daftar kemungkinan penyakit, lengkap dengan persentase tingkat kepercayaan (*confidence level*) untuk setiap diagnosis. Selain hasil diagnosis, sistem juga memberikan halaman kesimpulan yang berisi penjelasan mengenai penyakit yang paling mungkin diderita, serta saran dan anjuran tindakan selanjutnya yang harus dilakukan. Namun, peneliti juga menekankan bahwa sistem ini hanyalah alat bantu diagnosis awal dan bukan pengganti konsultasi medis profesional. Pengguna sangat dianjurkan untuk tetap berkonsultasi langsung dengan dokter untuk mendapatkan diagnosis yang lebih akurat dan penanganan yang tepat.

II.4.4 Penggunaan *Knowledge Graph* dalam Bidang Kesehatan (Cui dkk. 2025)

Jurnal ini memberikan tinjauan komprehensif tentang kondisi terkini dari *healthcare knowledge graph* (HKG), yang mencakup metode konstruksi (pembangunan), model pemanfaatan, dan berbagai aplikasinya di domain penelitian biomedis dan layanan kesehatan.

Berdasarkan jurnal tersebut, berikut ini merupakan cara melakukan konstruksi *healthcare knowledge graphs* (HKG) dari *scratch*:

1. Menentukan ruang lingkup dan struktur dari graf.
    (a) Sebelum memasukkan data apapun, akan didefinisikan skema dari graf. Skema menentukan jenis-jenis informasi apa yang akan disimpan dan bagaimana keterhubungannya.
2. Melakukan pengumpulan data.
    (a) Akan dilakukan pengumpulan semua sumber informasi dari pengetahuan yang akan di ekstraksi.
3. Melakukan transformasi data.
    (a) Data mentah yang didapatkan akan dilakukan transformasi dan mengubahnya menjadi bentuk SUBJEK – PREDIKAT – OBJEK.
4. Melakukan normalisasi entitas dan relasi.
    (a) Akan dilakukan standarisasi entitas yang telah diekstrak agar tidak ada duplikasi dan dapat terhubung dengan sistem lain. Ini adalah proses menghubungkan entitas dengan sebuah standar tertentu (misal menggunakan standar ICD10 (*International Classification of Diseases, 10th Revision*)).

5. Melakukan validasi berkelanjutan.

    (a) Akan dilakukan iterasi / perulangan pada proses sebelumnya untuk memastikan akurasi dan relevansi dari HKG yang dibangun.

II.5 *Analytical Hierarchy Process* (AHP)

Analytical Hierarchy Process (AHP) adalah sebuah teori pengukuran yang dirancang untuk mengatasi masalah pengambilan keputusan yang kompleks. Menurut jurnal yang ditulis oleh Saaty (November 2008), banyak keputusan melibatkan faktor-faktor tak berwujud (*intangibles*) yang sulit diukur, seperti reputasi, keamanan, atau fleksibilitas. AHP menyediakan kerangka kerja matematis untuk mengukur faktor-faktor ini secara relatif dan menggabungkannya dengan faktor-faktor yang berwujud (*tangibles*) untuk sampai pada sebuah keputusan yang logis dan transparan. Berdasarkan jurnal tersebut, terdapat 4 proses dalam merancang AHP, antara lain sebagai berikut:

1. Mendefinisikan masalah dan menyusun hierarki keputusan.
   Pada tahap pertama, akan dilakukan pemecahan keputusan menjadi sebuah struktur hierarki, antara lain adalah:

    (a) Tingkat puncak: Merupakan tujuan utama (*goals*).

    (b) Tingkat menengah: Kriteria dan subkriteria yang memengaruhi tujuan.

    (c) Tingkat terbawah: Alternatif pilihan yang akan dievaluasi.

2. Membuat matriks perbandingan berpasangan.
   Setelah hierarki terbentuk, dilakukan perbandingan berpasangan untuk setiap elemen pada satu tingkat terhadap elemen lain dalam tingkat yang sama, dengan mengacu pada elemen di tingkat atasnya. Pada tahapan ini, perbandingan akan didefinisikan menggunakan skala absolut 1 sampai 9, di mana kala ini menunjukkan seberapa jauh lebih penting atau dominan satu elemen dibandingkan yang lain, dengan nilai 1 dengan kriteria sama pentingnya, dan 9 adalah mutlak lebih penting.
   Penilaian ini dimasukkan ke dalam matriks. Jika elemen A dinilai 3 kali lebih penting dari elemen B, maka di matriks, posisi (A,B) diisi angka 3, dan posisi (B,A) diisi kebalikannya, yaitu 1/3.

3. Menghitung prioritas.
   Dari setiap matriks perbandingan berpasangan, sebuah vektor prioritas (disebut juga bobot atau *eigen vector*) dihitung. Vektor ini

merepresentasikan bobot relatif dari setiap elemen. Jurnal ini menyebutkan bahwa prioritas ini dapat dihitung secara aproksimasi dengan menjumlahkan nilai di setiap baris matriks dan menormalisasikannya (membagi setiap jumlah baris dengan total jumlah keseluruhan). Hasil dari perhitungan ini adalah adanya suatu prioritas dengan bobot setiap elemen relatif terhadap induknya dalam hierarki.

4. Melakukan sintesis untuk mendapatkan prioritas global.
   Tahapan terakhir adalah menggabungkan semua prioritas lokal untuk mendapatkan peringkat akhir dari setiap alternatif. Proses ini dilakukan dengan cara:

   (a) Mengalikan prioritas setiap alternatif (yang dihitung berdasarkan kriteria di atasnya) dengan bobot (prioritas) kriteria tersebut.

   (b) Menjumlahkan semua hasil perkalian bobot untuk setiap alternatif.

BAB III
ANALISIS MASALAH

III.1 Analisis Kondisi Saat Ini

Kondisi pengembangan *knowledge - based system* (KBS) saat ini, khususnya dalam domain kesehatan, ditandai adanya perubahan, yakni dari sistem pakar berbasis aturan (*rule-based*) yang kaku menuju arsitektur yang lebih dinamis dan fleksibel berbasis graf (*knowledge graph*). Sistem berbasis aturan tradisional dengan logika IF-THEN terbukti tidak mampu menangani kompleksitas, ambiguitas, dan volume data medis modern yang sangat besar (Sutton dkk. 2020).

Sebaliknya, penggunaan *knowledge graph* memungkinkan pemodelan hubungan yang rumit dan *multi-level*, seperti interaksi antara gen, protein, obat, dan penyakit secara lebih intuitif dan mudah untuk dikembangkan (skalabilitas). Adopsi pendekatan ini merupakan respons langsung terhadap tantangan yang muncul dari perkembangan teknologi di dunia kesehatan, mulai dari rekam medis elektronik (EHR) yang terstruktur, data genomik, hingga literatur penelitian dan catatan klinis yang tidak terstruktur. *Knowledge graph* dirancang untuk mengintegrasikan data ini menjadi satu jaringan pengetahuan yang koheren dan dapat diproses oleh mesin (Cui dkk. 2025). *Knowledge graph* dapat mengintegrasikan data ini menjadi satu jaringan pengetahuan yang koheren dan dapat diproses oleh mesin.

Pada praktiknya, perkembangan KBS di bidang kesehatan telah menunjukkan hasil yang signifikan di berbagai domain aplikasi. Salah satu area domain biomedis yang sedang gencar dilakukan penelitian adalah adanya penemuan dan penargetan ulang obat (*drug discovery and repurposing*). *Knowledge graph* seperti HetioNet, yang merupakan suatu *knowledge graph* menggunakan Neo4j, digunakan untuk mengidentifikasi interaksi baru antara obat, komponen struktur biologis manusia, dan penyakit (Himmelstein dkk. 2017).

Selain bidang penelitian, berbagai KBS juga dikembangkan untuk mendeteksi adanya penyakit. Sebagai contoh, saat masa pandemi, KBS dikembangkan untuk menganalisis literatur COVID-19 dan memprediksi obat-obatan yang sudah ada yang berpotensi efektif untuk pengobatan. Selain itu, dalam dukungan keputusan klinis, KBS diterapkan untuk penyakit kompleks seperti kanker, di mana sistem dapat mengintegrasikan data genomik, proteomik, dan klinis pasien untuk

merekomendasikan terapi yang dipersonalisasi (Cui dkk. 2025).

Hasil dari implementasi ini bervariasi dari identifikasi kandidat gen baru hingga peningkatan akurasi model prediksi risiko penyakit. Sebagaimana dirangkum dalam penelitian yang dilakukan oleh Cui dkk. (2025), kondisi saat ini menunjukkan bahwa meskipun banyak dari sistem ini masih dalam tahap prototipe penelitian, dampaknya dalam menyajikan pengetahuan medis yang dapat dijelaskan (*explainable*) dan terstruktur sudah cukup menjanjikan. Tren terkini bahkan mendorong integrasi dengan *large language models* (LLM), di mana *knowledge graph* berperan sebagai representasi pengetahuan yang mumpuni untuk memastikan sistem dapat menghasilkan keputusan yang faktual dan dapat dipercaya.

Berdasarkan kondisi saat ini, dapat disimpulkan bahwa penggunaan KBS sudah banyak dilakukan untuk domain kesehatan, yakni untuk mendeteksi berbagai penyakit dengan menggunakan metode tertentu. Penggunaan *knowledge graph* juga sudah diterapkan dalam berbagai penelitian dalam bidang *drug discovery* yang menunjukkan hasil yang menjanjikan. Penggunaan teknologi lainnya seperti *machine learning* dan LLM juga membantu dalam meningkatkan akurasi dari hasil yang didapatkan.

Namun, berdasarkan riset yang sudah dilakukan, hingga saat ini, belum ada kombinasi KBS yang dibuat untuk mendiagnosis penyakit yang menggunakan representasi pengetahuan menggunakan *knowledge graph*. Dengan demikian, dari *gap* yang ada di kondisi sekarang, akan dikembangkan suatu KBS yang menggunakan representasi pengetahuan menggunakan *knowledge graph*, dengan harapan untuk bisa meningkatkan akurasi dari hasil analisis yang dilakukan.

III.2 Analisis Masalah Saat Ini

Dalam pembangunan sistem pakar, terdapat beberapa tahapan yang akan dilakukan untuk memastikan bahwa hasil akhir dari sistem yang dikembangkan akurat dan dapat digunakan oleh pengguna. Namun, tentu di setiap tahapan akan terdapat permasalahan yang mungkin dapat menghambat proses pengerjaaan sistem tersebut. Oleh karena itu, akan ditentukan beberapa masalah yang mungkin dihadapi berdasarkan tahapan pembangunan dari KBS, yakni identifikasi, konseptualisasi, formalisasi, implementasi, dan juga pengujian (Barrett, Jones, dan Thompson 1992).

1. Identifikasi – Penentuan pakar yang sesuai.

(a) Dalam menyusun *knowledge – based system* dengan menggunakan pakar sebagai akuisisi pengetahuan, maka menemukan pakar yang benar-benar kompeten, bisa berkomunikasi dengan baik, dan memiliki waktu untuk kesulitan tersendiri.

(b) Kemampuan komunikasi dari pakar menjadi penentu dalam keberhasilan *knowledge graph* yang dibangun, di mana jika pakar sulit berkomunikasi atau tidak memberikan detail yang cukup jelas, maka pengembang akan kesulitan untuk menentukan fakta – fakta apa saja yang perlu untuk dimodelkan.

2. Konseptualisasi – Pemodelan data yang tidak tepat.

   (a) Dalam memodelkan data yang akan dipakai di dalam graf, tentu harus dilakukan pemodelan data dalam bentuk graf tersebut. Bagian pemodelan data graf menjadi bagian yang cukup krusial karena akan menjadi fondasi dari keseluruhan model yang akan dibangun.

   (b) Beberapa kesalahan yang mungkin terjadi, antara lain adalah generalisasi suatu gejala tertentu. Sebagai contoh, terdapat berbagai macam batuk, antara lain batuk berdahak, batuk kering, dan sebagainya. Namun, untuk kemudahan, maka kita akan men-generalisasikan menjadi satu *node* "batuk" saja.

   (c) Hal ini tentu dapat menyebabkan kesalahan diagnosis, karena bisa saja gejala batuk tersebut merupakan indikasi spesifik suatu penyakit tertentu, yang dapat berakibat kepada kesalahan diagnosis penyakit.

3. Formulasi – *Knowledge graph* yang tidak dapat mengenal konteks.

   (a) Sistem pakar tidak memiliki pemahaman mendasar tentang dunia, di mana sistem hanya mengikuti aturan secara harafiah tanpa adanya pemahaman konteks.

   (b) Semisal sistem pakar dalam bidang kesehatan menerima *input* dari pengguna. Jika pengguna salah memasukkan jumlah obat yang diminum, yakni 50 obat, padahal seharusnya hanya sekitar 5 obat, maka sistem pakar akan memproses data tersebut secara terang – terangan tanpa berpikir di balik layar jika ingin bukan merupakan hal yang logis (jika memang tidak ada *constraints* / batasan yang membatasinya).

(c) Sistem pakar juga didesain untuk bisa menjawab hal apapun yang bahkan diluar domain pengetahuannya, asalkan *input* yang diterima masih dalam format yang sesuai. Semisal sistem pakar didesain untuk menganalisis penyakit TBC. Namun, jika ternyata terdapat pasien yang men– *input* gejala Pneumonia, maka mungkin saja sistem masih dapat memberikan diagnosis untuk pasien tersebut berdasarkan gejala yang sesuai dari penyakit yang lain (jika misalkan penyakit yang di–*input* oleh pasien memang diprogram secara eksplisit).

4. Implementasi – Kinerja dan skalabilitas dari *knowledge graph*.

   (a) Saat *knowledge graph* yang dibuat menjadi sangat besar, *query* yang tadinya cepat bisa menjadi sangat lambat. Implementasi graf tanpa dilakukan optimasi akan menimbulkan *bottleneck* jika data yang diterima berskala besar.

   (b) Selain itu, implementasi dari *rule engine* yang cukup banyak dan hanya untuk melakukan inferensi untuk satu penyakit saja dapat membuat sistem menjadi sangat lambat.

5. Evaluasi – Proses Perawatan dan pemeliharaan / *maintenance*.

   (a) Ketika jumlah aturan bertambah (dari ratusan menjadi ribuan), menjadi sangat sulit untuk menambahkan aturan baru tanpa menimbulkan kontradiksi atau efek samping dari aturan yang sudah ada.

   (b) Penting bagi pengembang untuk terus memperbarui sistem pakarnya, terutama untuk domain kesehatan, di mana penyakit tidak bersifat statis, namun bersifat dinamis, yang artinya setiap hari pasti ada informasi baru terkait suatu penyakit (semisal ada gejala baru, atau ada metode penyembuhan yang baru yang lebih efektif, dan sebagainya).

III.3 Analisis Kebutuhan

Berdasarkan analisis masalah di atas, dapat disimpulkan bahwa *knowledge-based system* dengan menggunakan *knowledge graph* sebagai representasi pengetahuan memerlukan serangkaian kebutuhan yang terdefinisi dengan baik agar dapat mengatasi tantangan yang ada dan memberikan solusi yang efektif. Kebutuhan ini dapat diklasifikasikan menjadi identifikasi masalah pengguna, kebutuhan fungsional, dan kebutuhan non-fungsional.

III.3.1 Identifikasi Masalah Pengguna

Sistem ini akan memiliki dua kelompok pengguna utama, antara lain adalah pasien (pengguna umum) dan dokter spesialis (pakar).

1. Pasien / pengguna umum.
    (a) Kesulitan mendapatkan informasi awal yang terpercaya mengenai gejala penyakit paru-paru yang mereka alami.
    (b) Merasa cemas dan tidak tahu kapan harus mengambil tindakan untuk berkonsultasi dengan dokter.
    (c) Membutuhkan penjelasan yang mudah dipahami tentang kemungkinan penyakit berdasarkan gejala yang dirasakan.
2. Dokter spesialis / pakar.
    (a) Membutuhkan alat bantu untuk mempercepat proses diagnosis awal, terutama untuk kasus-kasus umum.
    (b) Menghadapi tantangan dalam mengelola dan menghubungkan pengetahuan medis yang kompleks dan terus berkembang (misalnya, hubungan antara gejala, faktor risiko, dan penyakit).
    (c) Membutuhkan *platform* untuk memvalidasi dan memperbarui basis pengetahuan secara sistematis dan fleksibel.

III.3.2 Kebutuhan Fungsional

Kebutuhan fungsional akan mendefinisikan fitur – fitur spesifik yang harus dimiliki oleh sistem sebagai solusi dari setiap peluang masalah yang sudah didefinisikan di bagian sebelumnya, antara lain:

1. Sistem memerlukan seorang *knowledge engineer* untuk dapat mengelola pengetahuan yang didapat dari pakar.
    (a) Dengan adanya *knowledge engineer*, hubungan antara pakar dengan sistem dapat terfailitasi dengan baik.
2. Sistem dapat menerima *input* berupa gejala dari pasien.
    (a) Sistem harus dapat menerima *input* berupa nama gejala penyakit secara spesifik, misalnya sistem harus memiliki *node* untuk gejala "batuk",

"batuk kering", "batuk berdahak", dan sebagainya, bukan hanya satu *node* umum untuk batuk.

(b) Hal ini bertujuan untuk menghindari generalisasi berlebih dari suatu gejala penyakit yang dapat menyebabkan kesalahan / ketidakakuratan dalam diagnosis.

3. Sistem dapat mengeluarkan *output* berupa diagnosis penyakit berdasarkan gejala yang dimasukkan pasien.

   (a) Sistem harus dapat mengeluarkan *output* berupa diagnosis penyakit berdasarkan analisis gejala yang dimasukkan pasien melalui *knowledge graph*.

4. Sistem harus memiliki mekanisme validasi *input* dan pembatasan konteks.

   (a) Sistem harus memiliki aturan validasi untuk *input* pengguna, misalnya menolak *input* yang salah atau tidak logis (misal sistem dapat memberikan peringatan jika ada data umur pasien yang tidak logis, misal berusia 200 tahun).

   (b) Sistem juga dapat memberikan pemberitahuan jika tidak memiliki pengetahuan nama penyakit berdasarkan gejala - gejala yang sudah diberikan.

III.3.3 Kebutuhan Non – Fungsional

Kebutuhan non-fungsional mendefinisikan standar kualitas dan performa sistem untuk memastikan solusi yang dibangun tidak hanya berfungsi, tetapi juga efisien dan andal.

1. Kinerja dan Skalabilitas (*Scalability*).

   (a) Waktu respons sistem untuk melakukan *query* diagnosis harus di bawah 3 detik, bahkan ketika basis pengetahuan sudah besar.

   (b) Arsitektur sistem harus dirancang untuk dapat menangani pertumbuhan *knowledge graph* hingga ratusan *node* dan relasi tanpa penurunan kinerja yang signifikan, yang dapat diukur dengan menggunakan kompleksitas ruang dan waktu dari *knowledge graph* tersebut.

2. Pemeliharaan (*Maintainability*).

   (a) Desain basis pengetahuan (skema graf) harus fleksibel dan modular,

memungkinkan penambahan tipe entitas atau relasi baru di masa depan tanpa harus merombak keseluruhan sistem.

3. Keandalan (*Reliability*).

   (a) Sistem harus memberikan hasil diagnosis yang konsisten untuk *input* gejala yang sama.

   (b) Setiap perubahan pada basis pengetahuan harus melalui proses validasi oleh pakar untuk memastikan akurasi tetap terjaga.

III.4 Analisis Pemilihan Solusi

Karena penggunaan graf pengetahuan (*knowledge graph*) telah ditetapkan sebagai metode representasi utama, analisis ini berfokus pada pemilihan strategi akuisisi pengetahuan dan konstruksi *knowledge graph* yang paling optimal. Pemilihan strategi akan menentukan kualitas, akurasi, dan kelayakan implementasi sistem. Untuk itu, akan dilakukan evaluasi terhadap tiga alternatif strategi menggunakan metode *Analytic Hierarchy Process* (AHP).

III.4.1 Alternatif Solusi

Terdapat tiga alternatif utama untuk strategi akuisisi pengetahuan dari *knowledge graph* dalam konteks Tugas Akhir ini:

1. Akuisisi manual berbasis pakar (*Expert – Driven*):

   (a) Pendekatan ini mengandalkan interaksi langsung dengan pakar domain (dokter spesialis) untuk mengekstraksi, memformalkan, dan memvalidasi pengetahuan.

   (b) Proses melibatkan wawancara terstruktur, studi literatur yang direkomendasikan pakar, dan pemodelan manual setiap entitas dan relasi ke dalam *knowledge graph*.

   (c) Keunggulan utamanya adalah akurasi dan kualitas pengetahuan yang sangat tinggi, namun kelemahannya adalah proses yang lambat dan tidak skalabel.

2. Akuisisi otomatis berbasis teks (NLP – *Driven*):

   (a) Pendekatan ini memanfaatkan teknik *Natural Language Processing* (NLP) untuk secara otomatis mengekstrak entitas dan relasi dari teks

yang besar (misalnya, artikel jurnal medis).

(b) Tujuannya adalah membangun *knowledge graph* dalam skala besar dengan intervensi manusia minimal.

(c) Keunggulannya adalah skalabilitas dan efisiensi, namun sangat rentan menghasilkan informasi yang salah (*noise*) dan memiliki kompleksitas implementasi yang sangat tinggi.

3. Pendekatan *hybrid* (otomatis dengan validasi pakar):

(a) Strategi ini menggabungkan kedua pendekatan sebelumnya. Sebuah *pipeline* NLP digunakan untuk mengekstrak pengetahuan secara otomatis, yang kemudian disajikan kepada pakar untuk divalidasi (disetujui, ditolak, atau diubah) sebelum dimasukkan ke dalam *knowledge graph* final.

(b) Pendekatan ini menyeimbangkan antara kualitas dan skala, namun memiliki kompleksitas implementasi tertinggi karena memerlukan pengembangan sistem ekstraksi otomatis dan antarmuka validasi.

III.4.2 Analisis Penentuan Solusi

Analisis dilakukan melalui tiga langkah utama AHP, antara lain adalah penyusunan hierarki, perbandingan berpasangan untuk menghitung bobot, dan sintesis hasil.

1. Penyusunan hierarki.
   Hierarki keputusan terdiri dari tujuan (memilih strategi akuisisi), empat kriteria utama, dan tiga alternatif yang sudah disebutkan. Kriteria yang digunakan adalah:

(a) K1: Kualitas & akurasi pengetahuan.
Seberapa valid dan andal pengetahuan yang dihasilkan.

(b) K2: Kelayakan implementasi (*feasibility*) untuk TA.
Seberapa realistis strategi ini untuk diselesaikan dalam batasan waktu dan sumber daya tugas akhir.

(c) K3: Skalabilitas dan cakupan pengetahuan
Kemampuan strategi untuk membangun. *knowledge graph* yang luas dan mudah diperluas.

(d) K4: Efisiensi keterlibatan pakar.

Seberapa efisien waktu pakar dimanfaatkan.

2. Perbandingan & perhitungan bobot.
   Pada tahap ini, penilaian kualitatif diubah menjadi angka menggunakan Skala Saaty (1-9). Hasil dari perhitungan ini adalah perbandingan pemilihan bobot dan penentuan solusi mana yang akan dipilih berdasarkan perhitungan bobot tersebut.

III.4.3 Penentuan Bobot Prioritas Kriteria

Kriteria-kriteria di atas dibandingkan satu sama lain untuk menentukan tingkat kepentingannya.

1. Justifikasi penilaian:

   (a) K1 vs K2 (Akurasi vs Kelayakan): Akurasi dinilai lebih penting (nilai 3) daripada kelayakan. Sebuah sistem yang layak dibuat namun tidak akurat tidak memiliki nilai guna.

   (b) K1 vs K3 (Akurasi vs Skalabilitas): Akurasi dinilai cukup lebih penting (nilai 5) daripada skalabilitas. Lebih baik memiliki *knowledge graph* kecil yang akurat daripada *knowledge graph* besar yang tidak akurat.

   (c) K1 vs K4 (Akurasi vs Efisiensi Pakar): Akurasi dinilai sangat lebih penting (nilai 7) daripada efisiensi waktu pakar. Kualitas hasil akhir adalah prioritas utama.

   (d) K2 vs K3 (Kelayakan vs Skalabilitas): Kelayakan (*feasibility*) untuk tugas akhir dinilai lebih penting (nilai 3) daripada skalabilitas. Proyek harus dapat diselesaikan terlebih dahulu sebelum memikirkan skalabilitas.

   Hasil dari perbandingan tersebut dapat dilihat pada matriks Tabel III.1.

Tabel III.1 Matriks perbandingan berpasangan untuk tiap kriteria

Kriteria	**K1**	**K2**	**K3**	**K4**
K1: Akurasi	1	3	5	7
K2: Kelayakan TA	1/3	1	3	5
K3: Skalabilitas	1/5	1/3	1	3
K4: Efisiensi Pakar	1/7	1/5	1/3	1

2. Rincian perhitungan bobot prioritas kriteria.
   Berdasarkan hasil perhitungan yang sudah dilakukan (rincian perhitungan ada

di bagian Lampiran), didapatkan hasil sebagai berikut:

Tabel III.2 Matriks perbandingan berpasangan dan bobot prioritas kriteria

Kriteria	**K1**	**K2**	**K3**	**K4**	**Bobot Prioritas**
K1: Akurasi	1	3	5	7	0.55
K2: Kelayakan TA	1/3	1	3	5	0.26
K3: Skalabilitas	1/5	1/3	1	3	0.13
K4: Efisiensi Pakar	1/7	1/5	1/3	1	0.06

III.4.4 Sintesis & Keputusan Akhir

Bobot dari kriteria dan alternatif digabungkan untuk mendapatkan skor akhir.

(a) Ringkasan bobot prioritas
Tabel III.3 merangkum semua bobot yang telah dihitung.

Tabel III.3 Ringkasan Bobot Prioritas dari AHP

Alternatif	vs K1: Akurasi (Bobot=0.55)	vs K2: Kelayakan (Bobot=0.26)	vs K3: Skalabilitas (Bobot=0.13)	vs K4: Efisiensi (Bobot=0.06)
A1: Manual	0.63	0.70	0.10	0.11
A2: Otomatis	0.10	0.10	0.65	0.62
A3: *Hybrid*	0.27	0.20	0.25	0.27

(b) Perhitungan skor akhir
Skor akhir setiap alternatif dihitung dengan mengalikan bobot pada setiap kolom dengan bobot kriteria di atasnya.

i. Skor Akuisisi Manual:
$(0.55 \times 0.63) + (0.26 \times 0.70) + (0.13 \times 0.10) + (0.06 \times 0.11) = 0.347 + 0.182 + 0.013 + 0.007 = \mathbf{0.549}$

ii. Skor Akuisisi Otomatis:
$(0.55 \times 0.10) + (0.26 \times 0.10) + (0.13 \times 0.65) + (0.06 \times 0.62) = 0.055 + 0.026 + 0.085 + 0.037 = \mathbf{0.203}$

iii. Skor Pendekatan *Hybrid*:
$(0.55 \times 0.27) + (0.26 \times 0.20) + (0.13 \times 0.25) + (0.06 \times 0.27) = 0.149 + 0.052 + 0.033 + 0.016 = \mathbf{0.250}$

(c) Keputusan akhir
Hasil perhitungan AHP menunjukkan peringkat dari setiap alternatif, antara lain sebagai berikut:

Tabel III.4 Hasil Akhir Peringkat Alternatif Strategi Akuisisi

Alternatif Strategi Akuisisi	Skor Akhir	Peringkat
Akuisisi Manual Berbasis Pakar	**0.549**	**1**
Pendekatan *Hybrid*	0.250	2
Akuisisi Otomatis Berbasis Teks	0.203	3

Berdasarkan analisis AHP yang komprehensif, strategi akuisisi manual berbasis pakar terpilih sebagai pendekatan yang paling optimal untuk Tugas Akhir ini. Keputusan ini didasarkan pada skornya yang paling tinggi dibandingkan dengan alternatif solusi lain, di mana berasal dari keunggulannya pada dua kriteria dengan bobot tertinggi, yakni akurasi pengetahuan medis dan kelayakan implementasi. Dalam konteks tugas akhir, kualitas dan mitigasi risiko proyek lebih diutamakan daripada skalabilitas. Oleh karena itu, strategi pengembangan akan berfokus pada kolaborasi intensif dengan pakar untuk membangun sebuah *knowledge graph* yang valid dan dapat dipertanggung jawabkan.

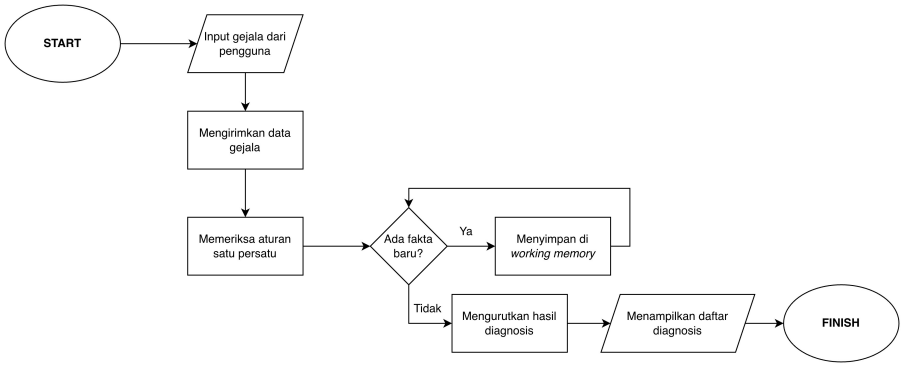
BAB IV
DESAIN DAN RANCANGAN SOLUSI

IV.1 Tahapan Desain

Bab ini akan menguraikan secara rinci mengenai proses desain dan rancangan dari *knowledge-based system* yang akan dibangun. Rancangan ini didasarkan pada analisis kebutuhan dan pemilihan solusi yang telah dibahas pada bab sebelumnya, di mana diputuskan untuk menggunakan *knowledge graph* dengan strategi akuisisi pengetahuan manual berbasis pakar.

IV.1.1 Tahapan Pertama

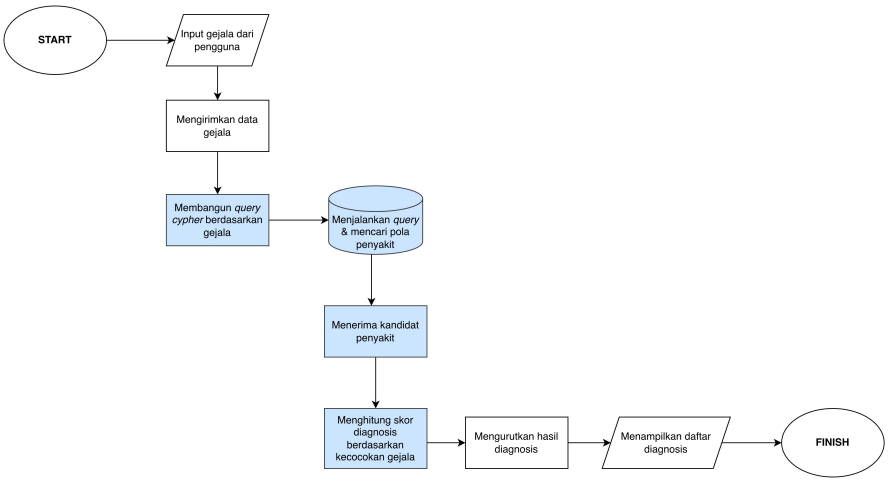
Pada tahapan pertama ini, untuk membantu memahami proses dari *knowledge – based system* yang akan dibangun, akan didefinisikan diagram alur proses dari *knowledge – based system as is*, antara lain sebagai berikut:



Gambar IV.1 Diagram alur proses KBS *as – is*.

Gambar IV.1 menggambarkan alur kinerja dari *knowledge – based system* pada umumnya, di mana akan menggunakan *rule – based*, sehingga dari setiap fakta yang didapat, akan dilakukan pengecekan di dalam *rule*, apakah sesuai atau tidak. Lalu, jika sesuai dan ada fakta baru, fakta tersebut akan ditambahkan ke dalam *working memory*. Selanjutnya, sistem akan mengurutkan hasil diagnosis serta menampilkan daftar diagnosis yang mungkin kepada pengguna.

Selanjutnya, gambar IV.2 menggambarkan *knowledge – based system* yang akan dibangun menggunakan *knowledge graph*:



Gambar IV.2 Diagram alur proses dari KBS yang akan dibangun.

Berikut ini merupakan penjelasan dari *flowchart* diatas:

(a) Pasien memasukkan gejala yang dialami.

- Pasien berinteraksi dengan program untuk memasukkan data yang dialami. Mereka memilih gejala-gejala yang mereka alami dari daftar yang disediakan.

(b) Sistem mengirimkan data gejala ke bagian CSF (*Case – Specific Fact*).

- Sistem akan mengumpulkan data gejala yang dipilih dan mengirimkannya ke *knowledge – based system* yang dibangun untuk diproses.

(c) Mesin inferensi akan membangun *query cypher* berdasarkan gejala.

- Mesin inferensi akan menerima daftar gejala dari komponen CSF dan secara dinamis membuat sebuah perintah *query* dalam bahasa Cypher. *Query* ini dirancang untuk mencari pola yang relevan di dalam *knowledge graph*.

(d) DSKB (*Domain – specific knowledge based*) akan menjalankan *query* dan mencari pola penyakit.

- *Query* yang sudah dibuat dikirimkan ke dalam basis data berbentuk *knowledge graph* (Neo4j). Basis data graf tersebut akan mengeksekusi *query* tersebut, menelusuri *node* yang ada dan relasi untuk menemukan semua *node* penyakit yang terhubung dengan

node gejala yang dimasukkan oleh pengguna. Hasil dari proses ini adalah daftar kandidat penyakit.

(e) Mesin inferensi (Komponen Hasil Sementara) akan menerima kandidat penyakit.

- Komponen Hasil Sementara akan menerima daftar kandidat penyakit yang dikembalikan oleh DSKB.

(f) Komponen Hasil Sementara akan menghitung skor diagnosis berdasarkan kecocokan gejala.

- Untuk setiap kandidat penyakit, Komponen Hasil Sementara akan menjalankan algoritma *scoring*. Skor dihitung berdasarkan faktor-faktor seperti jumlah gejala yang cocok, bobot spesifik dari setiap gejala, dan faktor lain yang akan ditentukan kemudian.

(g) Komponen Hasil Sementara akan mengurutkan hasil penyakit.

- Setelah semua kandidat penyakit diberi skor, daftar tersebut diurutkan dari yang memiliki skor tertinggi (paling mungkin) hingga yang terendah.

(h) Sistem akan menampilkan daftar diagnosis.

- Hasil akhir yang sudah diurutkan dikirim kembali ke pasien yang akan ditampilkan dalam format yang mudah dibaca, termasuk nama penyakit, persentase keyakinan, dan penjelasan mengapa sistem sampai pada kesimpulan tersebut (misalnya, "Didiagnosis karena gejala X, Y, dan Z cocok").

IV.1.2 Tahapan Kedua

Pada tahapan kedua ini, akan dirancang arsitektur dari *knowledge – based system*. Berikut ini merupakan komponen arsitektur KBS yang akan dibangun:

(a) Aktor eksternal.

- Pasien: Pasien merupakan pengguna akhir sistem yang tidak memiliki pengetahuan medis yang mendalam. Mereka berinteraksi dengan sistem untuk mendapatkan diagnosis awal berdasarkan gejala yang mereka alami. Mereka berinteraksi terutama melalui

Komponen Wawancara dan menerima *output* dari Komponen Penjelasan.

- Pakar / dokter spesialis: Ini adalah sumber pengetahuan domain. Berdasarkan strategi akuisisi manual yang dipilih, pakar (bersama dengan *knowledge engineer*) bertanggung jawab untuk membangun, memvalidasi, dan memperbarui *knowledge graph*. Mereka berinteraksi dengan sistem melalui Modul Akuisisi Pengetahuan.

(b) Basis pengetahuan (*knowledge bases*).

- *Case – specific facts* (CSF): CSF merupakan data temporer dan spesifik untuk setiap sesi konsultasi. Isinya adalah fakta, atau dalam kasus ini merupakan daftar gejala yang dimasukkan oleh seorang pengguna pada satu waktu tertentu. Informasi ini menjadi *input* utama bagi Mesin Inferensi untuk memulai proses diagnosis.
- *Domain – specific knowledge bases* (DSKB): DSKB berupa basis data graf (Neo4j) yang berisi model pengetahuan domain: *node-node* (penyakit, gejala) dan relasi-relasi yang menghubungkannya. Pengetahuan ini bersifat statis selama sesi konsultasi tetapi dapat diperbarui oleh pakar.

(c) Mesin inferensi:

- Mesin inferensi yang akan dibangun merupakan komponen sentral yang menjalankan proses diagnosis. Fungsi dari mesin inferensi, antara lain:
  i. Menerima CSF dari pengguna.
  ii. Mengonstruksi dan mengirimkan *query cypher* kepada DSKB.
  iii. Menerima hasil dari *query cypher*.
  iv. Melakukan *scoring* untuk menentukan diagnosis yang paling mungkin.
  v. Menyiapkan solusi masalah (hasil diagnosis) dan "Jalur Penalaran" untuk dikirim ke Komponen Penjelasan.
- Komponen Wawancara: Bagian yang berfungsi untuk

mendapatkan data / fakta dari pengguna. Komponen ini menyediakan daftar gejala yang akan diisi oleh pengguna untuk memasukkan gejala mereka.

- Komponen Akusisi Pengetahuan: Bagian yang berfungsi untuk memasukkan pengetahuan yang didapat oleh pakar ke dalam *knowledge graph*. Pada tahap ini, *knowledge engineer* akan memasukkan langsung ke basis data Neo4j.
- Komponen Hasil Sementara: Bagian ini akan menerima hasil diagnosis sementara dari DSKB dan akan melakukan perhitungan dan *sorting* untuk menghasilkan diagnosis penyakit mana yang paling mungkin.
- Komponen Penjelasan: Bagian ini yang akan memberikan penjelasan kepada pengguna terkait dengan hasil diagnosis yang didapatkan.

IV.1.3 Tahapan Ketiga

Pada tahapan ini, akan dirancang struktur dasar atau skema dari *knowledge graph* yang akan dibangun. Skema ini berfungsi sebagai *blueprint* yang mendefinisikan jenis-jenis informasi apa saja yang dapat disimpan dan bagaimana mereka dapat saling terhubung. Proses perancangan skema ini akan dilakukan secara iteratif dengan melibatkan pakar dalam proses pengembangannya. Langkah – langkah penyusunan *knowledge graph* antara lain sebagai berikut:

(a) Menentukan ruang lingkup pengembangan.
Sebelum mendefinisikan entitas, ruang lingkup dari *knowledge graph* harus dibatasi dengan jelas. Tujuannya adalah untuk memastikan *knowledge graph* dapat menjawab pertanyaan-pertanyaan spesifik yang relevan dengan tujuan sistem. Beberapa pertanyaan yang akan dijawab oleh *knowledge graph*, antara lain:

i. "Diberikan sekumpulan gejala, apa saja kemungkinan penyakit paru-paru yang diderita?"

ii. "Apa saja faktor risiko yang terkait dengan suatu penyakit?"

iii. "Pemeriksaan apa yang biasanya digunakan untuk mendiagnosis penyakit ini?"

(b) Mengidentifikasi tipe entitas (*node*).
Berdasarkan pertanyaan kunci, akan diidentifikasi kategori-kategori utama dari informasi yang akan menjadi *node* di dalam graf. Setiap kategori akan diberi label yang jelas, antara lain:

i. Penyakit: Entitas utama yang merepresentasikan kondisi medis. Contoh: "Tuberkulosis", "Pneumonia", "Asma".

ii. Gejala: Entitas yang merepresentasikan tanda atau keluhan. Contoh: "Batuk kronis", "Demam", "sesak napas".

iii. Faktor risiko: Entitas yang merepresentasikan kondisi atau kebiasaan yang meningkatkan risiko. Contoh: "Merokok", "Paparan debu silika".

iv. Pemeriksaan: Entitas yang merepresentasikan prosedur diagnostik yang perlu dilakukan. Contoh: "*Rontgen* dada", "Tes dahak BTA".

(c) Mengidentifikasi tipe relasi (*edge*).
Selanjutnya, didefinisikan hubungan semantik yang akan menjadi *edge* (panah) yang menghubungkan antar *node*. Relasi ini harus memiliki arah dan nama yang deskriptif.

i. MEMILIKI_GEJALA: Menghubungkan *node* Penyakit ke *node* Gejala. Contoh: (:Penyakit {nama: "Tuberkulosis"})-[:MEMILIKI_GEJALA]->(:Gejala {nama: "Batuk Kronis"}).

ii. MEMILIKI_FAKTOR_RISIKO: Menghubungkan *node* Penyakit ke *node* FaktorRisiko. Contoh: (:Penyakit {nama: "PPOK"})-[:MEMILIKI_FAKTOR_RISIKO]->(:FaktorRisiko {nama: "Merokok"}).

iii. DIDIAGNOSIS_DENGAN: Menghubungkan *node* Penyakit ke *node* Pemeriksaan. Contoh: (:Penyakit {nama: "Pneumonia"})-[:DIDIAGNOSIS_DENGAN]->(:Pemeriksaan {nama: "Rontgen Dada"}).

(d) Mendefinisikan properti untuk *node* dan relasi.
Untuk menambah informasi, setiap *node* dan relasi akan memiliki properti (atribut). Properti ini akan menjadi *input* bagi algoritma *scoring* diagnosis.

i. Properti *node*: Setiap *node* minimal akan memiliki properti nama (misal: ‘nama: "Tuberkulosis"‘) dan deskripsi (misal: ‘deskripsi: "Penyakit infeksi yang disebabkan oleh bakteri..."‘).

ii. Properti relasi: Untuk mendukung logika *scoring* pada mesin inferensi, relasi MEMILIKI_GEJALA akan memiliki properti tambahan. Contohnya:

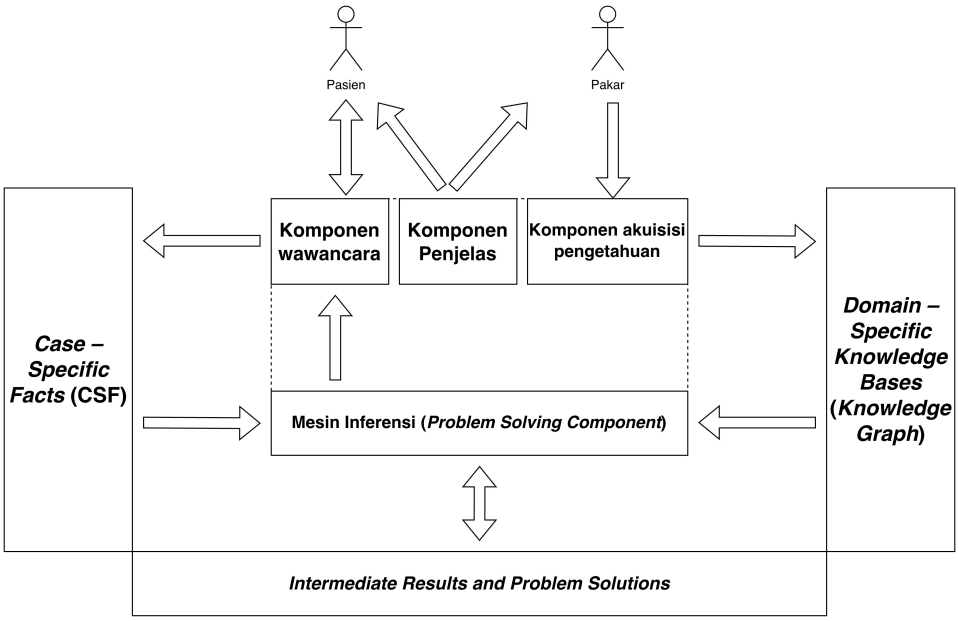
A. Bobot: Nilai numerik (misal 1 hingga 5) yang menandakan seberapa kuat atau umum sebuah gejala untuk suatu penyakit. Gejala kunci akan memiliki bobot tinggi.

B. Frekuensi: Keterangan kualitatif seperti "Umum", "Jarang", "Spesifik".

(e) Melakukan finalisasi skema graf.
Hasil akhir dari semua langkah di atas adalah sebuah dokumen skema graf yang berisi diagram visual dan penjelasan dari semua tipe *node*, tipe relasi, properti yang dimiliki, serta aturan bagaimana mereka boleh terhubung. Skema ini akan berfungsi sebagai kamus data dan panduan utama selama tahap formalisasi (populasi basis data).

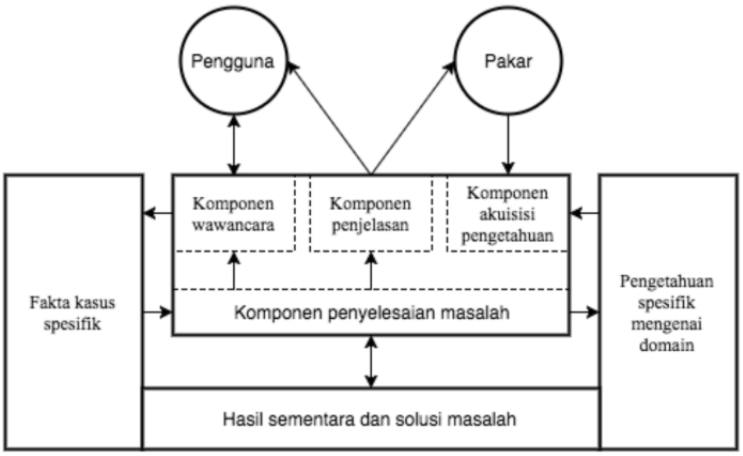
IV.2 Hasil Desain

Berdasarkan hasil perancangan komponen pada tahapan kedua, berikut ini merupakan arsitektur *knowledge – based system* yang akan dibuat:



Gambar IV.3 Rancangan hasil arsitektur KBS.

Hasil desain arsitektur tersebut sedikit berbeda dengan arsitektur *knowledge – based system* pada umumnya, antara lain sebagai berikut:



Gambar IV.4 Arsitektur *Knowledge – Based System* pada umumnya. (Puppe 1993)

Secara umum, perbedaan dari arsitektur KBS *as – is* dan juga rancangan arsitektur KBS *to – be* terletak pada representasi pengetahuan yang dibangun, di mana dalam kasus yang dikembangkan, akan digunakan *knowledge graph*, sedangkan pada arsitektur *as – is* tidak dispesifikkan representasi pengetahuan apa yang digunakan.

BAB V

RENCANA SELANJUTNYA

V.1 Tahapan Implementasi dan Evaluasi

Tahapan implementasi dan evaluasi dari pembuatan *knowledge – based system* antara lain terdiri atas 5 hal, yakni:

(a) Identifikasi

i. Pada tahap identifikasi, akan ditentukan pakar yang menjadi sumber pengetahuan dari KBS ini. Pakar merupakan seorang dokter spesialis paru – paru. Setelah terjadi kesepakatan dengan pakar, selanjutnya, saya akan melakukan diskusi terkait dengan ruang lingkup penyakit paru-paru yang akan dicakup dalam prototipe awal (misalnya, fokus pada 5-7 penyakit paling umum seperti Tuberkulosis, PPOK, Pneumonia, Asma, dan sebagainya).

ii. Selain itu, akan dilakukan konfigurasi semua perangkat lunak yang dibutuhkan, yakni Neo4j Desktop sebagai basis data graf untuk representasi pengetahuan.

(b) Konseptualisasi

i. Pada tahapan konseptualisasi, saya akan melakukan beberapa sesi wawancara dengan pakar untuk mengumpulkan pengetahuan tentang hubungan antara penyakit, gejala, faktor risiko, dan pemeriksaan.

ii. Setiap dilakukannya sesi wawancara, saya akan membuat *knowledge graph* berdasarkan dengan keterhubungan antara setiap komponen yang didapat selama sesi wawancara.

(c) Formalisasi

i. Pada tahapan ini, akan ditulis *script* dalam bahasa Cypher untuk membuat semua *node* (Penyakit, Gejala) dan relasi (MEMILIKI_GEJALA, dan sebagainya) di dalam Neo4j berdasarkan pengetahuan dari tahap sebelumnya.

ii. Setelah *script* ditulis, *script* akan dijalankan untuk mengisi basis data Neo4j. Setelah itu, untuk validasi, akan dibuat *query* sederhana untuk memverifikasi bahwa semua data telah dimasukkan dengan benar sesuai dengan model.

(d) Implementasi

i. Pada tahap implementasi, akan dibangun mesin inferensi yang akan terintegrasi dengan basis data Neo4j. Tahapan pembuatan / alur dari mesin inferensinya, antara lain:

A. Melakukan koneksi dengan *database* Neo4j.

B. Menerima *input* gejala dari pengguna.

C. Membangun *query* Cypher secara dinamis.

D. Mengeksekusi *query* dan menerima hasil.

E. Menjalankan algoritma *scoring* untuk menghitung dan mengurutkan diagnosis.

F. Mengembalikan hasil akhir yang sudah terstruktur untuk ditampilkan.

(e) Pengujian

i. Verifikasi sistem.

A. Akan dilakukan evaluasi apakah program dapat menangani *input* pengguna dengan benar (misalnya, *input* angka, *input* yang dipisahkan koma) dan apakah *output* yang ditampilkan sesuai dengan format yang telah dirancang.

B. Selanjutnya, akan diukur waktu eksekusi dari saat pengguna memasukkan gejala hingga hasil diagnosis tercetak sebagai *output*.

ii. Validasi pengetahuan.

A. Untuk memastikan *knowledge graph* yang disusun sudah tepat, pakar akan menjalankan program secara langsung atau didampingi oleh pengembang. Pakar akan diminta untuk memasukkan serangkaian skenario kasus klinis (kombinasi

gejala). Untuk setiap skenario, akan dicek apakah diagnosis yang benar muncul di peringkat 1, 2, atau 3 teratas dari *output*.

V.2 Lingkungan, Alat, dan Bahan

Bagian ini merinci semua komponen perangkat keras, perangkat lunak, dan sumber daya data yang diperlukan untuk pengembangan, pengujian, dan eksekusi dari *knowledge-based system* yang akan dibangun.

V.2.1 Lingkungan Pengembangan

Lingkungan merujuk pada *platform* dan konfigurasi di mana sistem akan dikembangkan dan dijalankan.

(a) Perangkat Keras (*Hardware*)
Seluruh proses pengembangan, pengujian, dan eksekusi akan dilakukan pada satu unit MacBook Pro. Spesifikasi dari perangkat yang digunakan:

i. Model: MacBook Pro (13-*inch*, M1, 2020)

ii. Prosesor: Apple M1 *chip*.

iii. Memori (RAM): 8 GB *Unified Memory*.

iv. Penyimpanan: 256 GB *Solid State Drive* (SSD).

(b) Sistem Operasi (*Operating System*)

i. Pengembangan akan dilakukan pada sistem operasi macOS Tahoe 26.1. Lingkungan berbasis UNIX dari macOS menyediakan dukungan *native* yang sangat baik untuk alat-alat pengembangan yang akan digunakan.

V.2.2 Alat Pengembangan (*Tools*)

Alat merujuk pada perangkat lunak, *library*, dan *framework* yang akan digunakan untuk membangun sistem.

(a) Bahasa Pemrograman.

i. Python (versi 3.11): Python dipilih sebagai bahasa utama karena *syntax* yang mudah dipahami, ekosistem *library* yang mumpuni, dan dukungan *driver* resmi untuk Neo4j.

(b) Basis Data.

i. Neo4j Desktop (*Community Edition*): Neo4j Desktop merupakan versi *desktop* dari Neo4j yang akan digunakan sebagai *server* basis data graf lokal. Neo4j Desktop menyediakan antarmuka visual yang sangat membantu, yaitu Neo4j Browser, yang akan digunakan untuk memvisualisasikan, memvalidasi, dan melakukan *query* pada *knowledge graph* selama tahap pengembangan.

(c) *Integrated Development Tools* (IDE).

i. Visual Studio Code (VS Code): VS Code dipilih sebagai editor kode utama karena ringan, fleksibel, dan memiliki ekosistem *extension* yang sangat luas.

(d) *Virtual Environment*.

i. venv (*Python Standard Library*): venv merupakan lingkungan virtual akan digunakan untuk mengisolasi *dependency* proyek agar tidak berkonflik dengan instalasi Python sistem.

(e) *Version Control*.

i. GitHub: GitHub digunakan sebagai *platform hosting repository* Git secara daring untuk penyimpanan versi dari kode (*versioning*) dan manajemen proyek.

(f) Terminal.

i. Aplikasi Terminal: Terminal bawaan yang digunakan sebagai antarmuka utama untuk menjalankan dan berinteraksi dengan program *knowledge – based system* yang dibangun.

V.2.3 Bahan Penelitian

Bahan merujuk pada sumber data dan pengetahuan yang menjadi *input* untuk membangun dan mengevaluasi sistem.

(a) Pengetahuan Domain.

i. Sumber utama pengetahuan adalah pakar (dokter spesialis paru – paru).

ii. Pengetahuan dari pakar akan diakusisi dari hasil wawancara

terstruktur, catatan diskusi kasus, dan diagram konseptual yang dibuat selama sesi akuisi pengetahuan. Pengetahuan ini mencakup hubungan antara penyakit, gejala, faktor risiko, dan pemeriksaan yang relevan.

(b) Data Uji.

i. Serangkaian kombinasi gejala yang representatif dari kasus-kasus nyata akan disusun bersama pakar.

ii. Skenario-skenario ini akan digunakan sebagai data *input* selama tahap validasi untuk mengukur akurasi dan relevansi dari hasil diagnosis yang diberikan oleh sistem.

V.3 Estimasi Biaya

Karena pada dasarnya *software* dari Neo4j *Community Edition* gratis, maka biaya pengembangan dari *knowledge – based system* tidak ada untuk saat ini (akan ditambahkan jika terdapat biaya tambahan).

V.4 Linimasa Pengerjaan

Tabel V.1 merupakan *gantt chart* rencana pengerjaan *knowledge – based system*.

Tabel V.1 *Gantt chart* perencanaan pengembangan *knowledge – based system*.

Kegiatan	Januari 2026					Februari 2026				Maret 2026				April 2026					Mei 2026				
	W1	W2	W3	W4	W5	W1	W2	W3	W4	W1	W2	W3	W4	W1	W2	W3	W4	W5	W1	W2	W3	W4	W5
Fase 1: Identifikasi																							
Menyiapkan lingkungan pengembangan yang dibutuhkan.																							
Menyusun jadwal wawancara secara bertahap dengan pakar.																							
Melakukan studi literatur mendalam terkait implementasi *knowledge graph*.																							
Melakukan studi kelayakan KBS dengan pakar.																							
Fase 2: Konseptualisasi																							
Melakukan wawancara dengan pakar.																							
Mempersiapkan struktur awal dari *knowledge graph*.																							
Melakukan validasi model *knowledge graph* dengan pakar.																							
Melakukan finalisasi model *knowledge graph* dengan pakar.																							
Fase 3: Formalisasi																							
Menulis *script query* Neo4j.																							
Melakukan populasi basis data Neo4j.																							
Melakukan validasi data awal dengan *query* sederhana.																							
Fase 4: Implementasi																							
Membangun mesin inferensi / *back end*.																							
Mengimplementasikan algoritma *scoring* di mesin inferensi.																							
Mengimplementasikan antarmuka program *console*.																							
Melakukan pengujian integrasi keseluruhan komponen.																							
Fase 5: Pengujian																							
Melakukan verifikasi sistem & pengujian fungsional sistem.																							
Melakukan UAT (*user acceptance testing*) dengan pakar.																							
Menganalisis & merevisi *feedback* dari pakar.																							

Berdasarkan *gantt chart* tersebut, akan dialokasikan sekitar 5 bulan untuk melakukan pengembangan KBS, mulai dari fase identifikasi pada minggu pertama dan kedua di bulan Januari 2026, fase konseptualisasi pada minggu ketiga bulan Januari 2026 hingga minggu kedua bulan Februari 2026, fase

formalisasi dari minggu ketiga bulan Februari 2026 hingga minggu ketiga bulan Maret 2026, fase implementasi dari minggu keempat bulan Maret 2026 hingga minggu pertama bulan Mei 2026, dan terakhir untuk fase pengujian dimulai dari minggu kedua hingga minggu kelima bulan Mei 2026.

Terdapat *spare* waktu 2 bulan (Juni dan Juli) untuk menjadi cadangan jika semisal terjadi hambatan dalam fase tertentu yang dapat mengakibatkan mundurnya *timeline* yang sudah ditetapkan.

V.5 Analisis Risiko dan Mitigasi

Berikut ini merupakan analisis risiko dan mitigasi dari *knowledge – based system* yang akan dibangun.

Tabel V.2 Analisis Risiko dan Mitigasi Proyek

No.	Kategori Risiko	Deskripsi Risiko	Tingkat	Rencana Mitigasi
A. Risiko Pengetahuan (*Knowledge-related*)				
1.	Akuisisi Pengetahuan	Ketersediaan waktu pakar terbatas. Jadwal pakar (dokter spesialis) yang cukup padat dapat menyebabkan sesi wawancara dan akuisisi pengetahuan tertunda.	Tinggi	1. Menjadwalkan semua sesi jauh-jauh hari. 2. Menyiapkan agenda yang jelas untuk memaksimalkan efisiensi. 3. Meminta rekomendasi literatur sebagai cadangan.
2.	Akuisisi Pengetahuan	Pengetahuan yang ambigu atau implisit. Pakar memberikan informasi yang sulit untuk diformalkan ke dalam struktur graf.	Sedang	1. Menggunakan teknik wawancara berbasis skenario. 2. Melakukan validasi iteratif dengan menunjukkan model graf sementara kepada pakar.
Lanjut ke halaman berikutnya...				

No.	Kategori Risiko	Deskripsi Risiko	Tingkat	Rencana Mitigasi
3.	Validasi Pengetahuan	Akurasi sistem rendah. Hasil diagnosis prototipe tidak sesuai dengan ekspektasi pakar saat evaluasi.	Tinggi	1. Memvalidasi *knowledge graph* secara iteratif setiap kali ada *input* pengetahuan yang baru. 2. Merancang algoritma *scoring* yang fleksibel untuk penyesuaian berdasarkan umpan balik.
B. Risiko Teknis (*Technical*)				
4.	Desain Basis Data	Skema graf tidak cukup. Skema yang dirancang di awal kurang lengkap.	Sedang	1. Memulai dengan skema inti yang sederhana dan mengembangkannya secara iteratif. 2. Memvalidasi skema graf dengan pakar di awal proyek.
5.	Implementasi	Logika inferensi tidak optimal. *Query Cypher* atau algoritma *scoring* menghasilkan diagnosis yang kurang relevan.	Sedang	1. Memulai dengan algoritma *scoring* yang sederhana dan melakukan evaluasi dari perhitungan yang dilakukan. 2. Mencatat (*log*) semua *query Cypher* untuk kemudahan *debugging*.
6.	Implementasi	Kesulitan teknis yang tidak terduga, seperti masalah dalam menghubungkan Python dengan Neo4j.	Rendah	1. Menggunakan *library* dan *driver* yang stabil dan terdokumentasi dengan baik. 2. Mengisolasi lingkungan pengembangan menggunakan `virtual environment`.
C. Risiko Manajemen Proyek (*Project Management*)				

Lanjut ke halaman berikutnya...

No.	Kategori Risiko	Deskripsi Risiko	Tingkat	Rencana Mitigasi
7.	Ruang lingkup	Pelebaran ruang lingkup (*Scope Creep*). Muncul keinginan untuk menambah fitur atau penyakit di tengah pengerjaan.	Sedang	1. Menetapkan batasan ruang lingkup yang jelas dan terukur di awal. 2. Mencatat semua ide tambahan yang *feasible* untuk dilanjutkan di tesis.
8.	Jadwal	Keterlambatan proyek. Salah satu fase pengembangan memakan waktu lebih lama dari yang dijadwalkan.	Sedang	1. Membuat linimasa pengerjaan (*gantt chart*) yang realistis. 2. Melakukan evaluasi mingguan untuk identifikasi keterlambatan dini.

V.6 Rencana Evaluasi

Rencana evaluasi ini bertujuan untuk mengukur kualitas, akurasi, dan kegunaan dari prototipe KBS yang telah dibangun. Evaluasi akan dilakukan dalam dua fase utama, yakni verifikasi sistem untuk memastikan fungsionalitas teknis, serta validasi pengetahuan untuk mengukur keakuratan dan manfaat dari hasil diagnosis. Evaluasi akan menggunakan pendekatan campuran (*mixed-method*), yang menggabungkan metrik kuantitatif dan *feedback* kualitatif.

(a) Verifikasi sistem: Akan dilakukan pengujian fungsional dan non – fungsional oleh pengembang.

(b) Validasi pengetahuan: Akan dilakukan melalui validasi berbasis skenario oleh pakar yang merupakan bentuk dari *user acceptance testing* (UAT).

V.6.1 Prosedur Evaluasi

Berikut ini merupakan prosedur evaluasi yang akan dilakukan:

(a) Pengujian fungsional.
Pengujian fungsional memastikan setiap komponen program berjalan sesuai rancangan. Berikut ini merupakan *use – case* pengujian fungsional yang akan dilakukan:

i. Program dapat dimulai tanpa adanya *error*.

ii. Daftar gejala dapat ditampilkan di antarmuka pengguna.

iii. Program dapat menangani berbagai format *input* dari pengguna (misalnya, angka tunggal, beberapa angka dipisah koma, *input*yang salah/tidak valid).

iv. Program berhasil terhubung ke basis data Neo4j.

v. Format *output* teks (hasil diagnosis dan penjelasan) sesuai dengan yang telah dirancang.

(b) Pengujian non – fungsional.
Pengujian non – fungsional bertujuan untuk mengukur performa sistem dalam menjalankan tugas yang diberikan. Berikut ini merupakan *use – case* pengujian non - fungsional yang akan dilakukan:

i. Pengukuran waktu proses eksekusi program saat pengguna selesai memasukkan gejala hingga hasil diagnosis pertama kali tercetak di antarmuka. Pengujian akan dilakukan sebanyak 10 kali dengan skenario yang berbeda, lalu dihitung waktu rata-ratanya.

ii. Perhitungan kompleksitas waktu dan ruang dari sistem yang sudah dibuat.

(c) Validasi pengetahuan KBS.
Sebelum sesi evaluasi, pengembang akan bekerja sama dengan pakar untuk menyusun 10-15 skenario kasus klinis. Setiap skenario berisi daftar gejala yang representatif untuk penyakit tertentu, dengan tingkat kesulitan yang bervariasi dengan tujuan untuk mengukur akurasi dari sistem. Prosedur yang dapat dilakukan, antara lain sebagai berikut:

i. Pakar akan diberikan prototipe program.

ii. Untuk setiap skenario kasus, pakar akan memasukkan gejala yang sesuai ke dalam program.

iii. Pakar akan mengamati dan mencatat daftar diagnosis terurut yang dihasilkan oleh sistem.

Untuk setiap skenario, dicatat apakah diagnosis yang benar muncul di peringkat 1 (top 1), peringkat 3 teratas (top 3), atau tidak sama sekali.

Selain itu, setelah sesi, pakar akan memberikan *feedback* terkait dengan hasil, ataupun *knowledge engineer* dapat melakukan wawancara singkat dengan pakar untuk mengevaluasi hasil yang diberikan.

(d) Validasi kegunaan untuk pengguna umum.
Pengetesan sistem juga akan melibatkan pengguna umum untuk memastikan bahwa pengguna umum juga dapat memahami penggunaan dari sistem. Pengguna akan mencoba untuk memasukkan gejala yang dialami, lalu melihat hasil diagnosis dan juga penjelasan yang diberikan, apakah dipahami atau tidak penjelasan hasil diagnosis tersebut.

V.6.2 Kriteria Keberhasilan

Prototipe sistem akan dianggap berhasil jika memenuhi kriteria berikut:

(a) Verifikasi:

i. Semua *use – case* dalam *list* pengujian fungsional berhasil dieksekusi.

ii. Rata-rata waktu respons diagnosis berada di bawah 3 detik, sesuai dengan kebutuhan non-fungsional.

iii. Kompleksitas ruang dan waktu algoritma tidak melebihi O(2^n).

(b) Validasi:

i. Akurasi top 3 (diagnosis yang benar muncul dalam 3 peringkat teratas) mencapai minimal 80% dari seluruh skenario uji.

ii. Pakar memberikan penilaian positif secara umum, menyatakan bahwa sistem bermanfaat sebagai alat bantu diagnosis awal.

iii. Pengguna umum menyatakan bahwa *output* teks jelas dan dapat dipahami.

DAFTAR PUSTAKA

Aditia, Gilang, Afzal Ziqri, Aldhan Tri Maulana, dan Faisal Dharma Adhinata. May 2023. "Expert System for Identifying Pregnant Using Forward Chaining". *Journal of Informatics Information System Software Engineering and Applications (INISTA)* 5 (2): 136–143. https://doi.org/10.20895/inista.v5i2.494.

Akil, Ibnu. 2020. *Expert System That Detects COVID-19 Using Forward Chaining Algorithm.* Technical report. https://iocscience.org/ejournal/index.php/mantik.

Barrett, John R, Don D Jones, dan Thomas L Thompson. 1992. "Knowledge systems development in U.S. agriculture". *Expert Systems with Applications* 4 (1): 45–51. ISSN: 0957-4174. https://doi.org/https://doi.org/10.1016/0957-4174(92)90039-U. https://www.sciencedirect.com/science/article/pii/095741749290039U.

Cui, Hejie, Jiaying Lu, Ran Xu, Shiyu Wang, Wenjing Ma, Yue Yu, Shaojun Yu, dkk. 2025. "A review on knowledge graphs for healthcare: Resources, applications, and promises". *Journal of Biomedical Informatics* 169:104861. ISSN: 1532-0464. https://doi.org/https://doi.org/10.1016/j.jbi.2025.104861. https://www.sciencedirect.com/science/article/pii/S1532046425000905.

Fadli, Sofiansyah, Maulana Ashari, Ria Septiani, Saikin Saikin, dan Didik Sudyana. 2024. "Expert System for Diagnosing Covid-19 Disease Using Method Forward Chaining". *JISA(Jurnal Informatika dan Sains),* https://api.semanticscholar.org/CorpusID:270890816.

Himmelstein, Daniel Scott, Antoine Lizee, Christine Hessler, Leo Brueggeman, Sabrina L Chen, Dexter Hadley, Ari Green, Pouya Khankhanian, dan Sergio E Baranzini. 2017. "Systematic integration of biomedical knowledge prioritizes drugs for repurposing". Disunting oleh Alfonso Valencia. *eLife* 6:e26726. ISSN: 2050-084X. https://doi.org/10.7554/eLife.26726. https://doi.org/10.7554/eLife.26726.

Ji, Shaoxiong, Shirui Pan, Erik Cambria, Pekka Marttinen, dan Philip S Yu. 2020. "A Survey on Knowledge Graphs: Representation, Acquisition and Applications". *CoRR* abs/2002.00388. https://arxiv.org/abs/2002.00388.

Leake, Charles R. December 1991. "Knowledge Acquisition: Principles and Guidelines". Doi: 10.1057/jors.1991.209, *Journal of the Operational Research Society* 42 (12): 1125. ISSN: 0160-5682. https : / / doi . org / 10 . 1057 / jors . 1991 . 209. https://doi.org/10.1057/jors.1991.209.

Nicholson, David N, dan Casey S Greene. 2020. "Constructing knowledge graphs and their biomedical applications". *Computational and Structural Biotechnology Journal* 18:1414–1428. ISSN: 2001-0370. https://doi.org/ https://doi.org/10.1016/j.csbj.2020.05.017. https://www.sciencedirect.com/science/article/pii/S2001037020302804.

Puppe, Frank. 1993. *Systematic Introduction to Expert Systems: Knowledge Representations and Problem Solving Methods.* Springer-Verlag. ISBN: 0387562559.

Quinlan, J R. 1986. "Induction of decision trees". *Machine Learning* 1 (1): 81–106. ISSN: 1573-0565. https://doi.org/10.1007/BF00116251. https://doi.org/10.1007/BF00116251.

Riley, Gary. 1999. "CLIPS: A tool for building expert systems". *línea], disponible en: http://www. ghg. net/clips/CLIPS. html, recuperado.*

Saaty, Thomas. November 2008. "Decision making with the Analytic Hierarchy Process". *Int. J. Services Sciences Int. J. Services Sciences* 1 (): 83–98. https://doi.org/10.1504/IJSSCI.2008.017590.

Schork, Nicholas J. 2019. "Artificial Intelligence and Personalized Medicine". Dalam *Cancer Treatment and Research,* 178:265–283. Springer International Publishing. https://doi.org/10.1007/978-3-030-16391-4_11.

Sutton, Reed T, David Pincock, Daniel C Baumgart, Daniel C Sadowski, Richard N Fedorak, dan Karen I Kroeker. 2020. “An overview of clinical decision support systems: benefits, risks, and strategies for success”. *npj Digital Medicine* 3 (1): 17. ISSN: 2398-6352. https : / / doi . org / 10 . 1038 / s41746 - 020 - 0221 - y. https://doi.org/10.1038/s41746-020-0221-y.

Trenggono, Patriot Haryo, dan Adang Bachtiar. April 2023. “PERAN ARTIFICIAL INTELLIGENCE DALAM PELAYANAN KESEHATAN : A SYSTEMATIC REVIEW”. *Jurnal Ners* 7 (1): 444–451. https : / / doi . org / 10 . 31004 / jn . v7i1 . 13612. https : //journal.universitaspahlawan.ac.id/index.php/ners/article/view/13612.

Yimenu, Taddesse, Abebe Adege, dan Sofonias Yitagesu. October 2025. “Integrating expert knowledge with machine learning for AI-based stroke identifications and treatment systems”. *DIGITAL HEALTH* 11 (). https://doi.org/10.1177/20552076251336853.

LAMPIRAN 1

RINCIAN PERHITUNGAN AHP

.1 Rincian Perhitungan Bobot Prioritas Kriteria

Berikut ini merupakan rincian perhitungan bobot prioritas kriteria dari AHP:

(a) Langkah 1: Menjumlahkan setiap kolom matriks.
Setiap kolom dari matriks perbandingan (dengan nilai pecahan diubah ke desimal) dijumlahkan.

i. Jumlah Kolom K1: $1.000 + 0.333 + 0.200 + 0.143 = 1.676$

ii. Jumlah Kolom K2: $3.000 + 1.000 + 0.333 + 0.200 = 4.533$

iii. Jumlah Kolom K3: $5.000 + 3.000 + 1.000 + 0.333 = 9.333$

iv. Jumlah Kolom K4: $7.000 + 5.000 + 3.000 + 1.000 = 16.000$

(b) Langkah 2: Normalisasi matriks.
Setiap elemen dalam matriks perbandingan awal (Tabel .4) dibagi dengan total kolomnya masing-masing. Rincian perhitungannya adalah sebagai berikut:

i. Perhitungan untuk baris K1 (Akurasi):

A. Kolom K1: $\frac{1.000}{1.676} = 0.597$

B. Kolom K2: $\frac{3.000}{4.533} = 0.662$

C. Kolom K3: $\frac{5.000}{9.333} = 0.536$

D. Kolom K4: $\frac{7.000}{16.000} = 0.438$

ii. Perhitungan untuk baris K2 (Kelayakan TA):

A. Kolom K1: $\frac{0.333}{1.676} = 0.199$

B. Kolom K2: $\frac{1.000}{4.533} = 0.221$

C. Kolom K3: $\frac{3.000}{9.333} = 0.321$

D. Kolom K4: $\frac{5.000}{16.000} = 0.313$

iii. Perhitungan untuk baris K3 (Skalabilitas):

A. Kolom K1: $\frac{0.200}{1.676} = 0.119$

B. Kolom K2: $\frac{0.333}{4.533} = 0.073$

C. Kolom K3: $\frac{1.000}{9.333} = 0.107$

D. Kolom K4: $\frac{3.000}{16.000} = 0.188$

iv. Perhitungan untuk baris K4 (Efisiensi Pakar):

A. Kolom K1: $\frac{0.143}{1.676} = 0.085$

B. Kolom K2: $\frac{0.200}{4.533} = 0.044$

C. Kolom K3: $\frac{0.333}{9.333} = 0.036$

D. Kolom K4: $\frac{1.000}{16.000} = 0.063$

Hasil dari perhitungan ini kemudian disusun ke dalam Tabel .3.

Tabel .3 Matriks normalisasi untuk tiap kriteria

Kriteria	K1	K2	K3	K4
K1: Akurasi	0.597	0.662	0.536	0.438
K2: Kelayakan TA	0.199	0.221	0.321	0.313
K3: Skalabilitas	0.119	0.073	0.107	0.188
K4: Efisiensi Pakar	0.085	0.044	0.036	0.063

(c) Langkah 3: Menghitung Rata-rata Baris (Hasil Bobot Prioritas).
Bobot prioritas adalah nilai rata-rata dari setiap baris pada matriks ternormalisasi.

i. Bobot K1 (Akurasi):
$\frac{0.597+0.662+0.536+0.438}{4} = 0.558 \approx \mathbf{0.55}$

ii. Bobot K2 (Kelayakan):
$\frac{0.199+0.221+0.321+0.313}{4} = 0.264 \approx \mathbf{0.26}$

iii. Bobot K3 (Skalabilitas):
$\frac{0.119+0.073+0.107+0.188}{4} = 0.122 \approx \mathbf{0.13}$

iv. Bobot K4 (Efisiensi):
$\frac{0.085+0.044+0.036+0.063}{4} = 0.057 \approx \mathbf{0.06}$

Dengan demikian, didapatkan matriks hasil prioritas sebagai berikut:

Tabel .4 Matriks perbandingan berpasangan dan bobot prioritas kriteria

Kriteria	**K1**	**K2**	**K3**	**K4**	**Bobot Prioritas**
K1: Akurasi	1	3	5	7	0.55
K2: Kelayakan TA	1/3	1	3	5	0.26
K3: Skalabilitas	1/5	1/3	1	3	0.13
K4: Efisiensi Pakar	1/7	1/5	1/3	1	0.06

.2 Perhitungan Evaluasi Alternatif Terhadap Setiap Kriteria

Setiap alternatif dievaluasi untuk masing-masing kriteria melalui proses normalisasi matriks perbandingan berpasangan.

(a) Terhadap K1: Akurasi pengetahuan

i. Justifikasi: Manual paling akurat (diawasi 100% oleh pakar). *Hybrid* di urutan kedua (ada validasi). Otomatis paling tidak akurat (risiko *noise*). Manual dinilai lebih akurat (7) dari otomatis dan sedikit lebih akurat (3) dari *Hybrid*.

ii. Rincian Perhitungan:

A. Matriks Perbandingan (dalam desimal)

Tabel .5 Matriks Perbandingan Alternatif terhadap K1

vs K1	Manual	*Hybrid*	Otomatis
Manual	1.000	3.000	7.000
Hybrid	0.333	1.000	3.000
Otomatis	0.143	0.333	1.000
Jumlah	**1.476**	**4.333**	**11.000**

B. Matriks Ternormalisasi

- Perhitungan untuk baris Manual:
  - Kolom Manual: $\frac{1.000}{1.476} = 0.677$
  - Kolom *Hybrid*: $\frac{3.000}{4.333} = 0.692$
  - Kolom Otomatis: $\frac{7.000}{11.000} = 0.636$
- Perhitungan untuk baris *Hybrid*:

- Kolom Manual: $\frac{0.333}{1.476} = 0.226$
- Kolom *Hybrid*: $\frac{1.000}{4.333} = 0.231$
- Kolom Otomatis: $\frac{3.000}{11.000} = 0.273$

• Perhitungan untuk baris Otomatis:

- Kolom Manual: $\frac{0.143}{1.476} = 0.097$
- Kolom *Hybrid*: $\frac{0.333}{4.333} = 0.077$
- Kolom Otomatis: $\frac{1.000}{11.000} = 0.091$

Hasilnya disusun dalam Tabel 6.

Tabel .6 Matriks Ternormalisasi untuk Kriteria K1

vs K1	Manual	*Hybrid*	Otomatis
Manual	0.677	0.692	0.636
Hybrid	0.226	0.231	0.273
Otomatis	0.097	0.077	0.091

C. Rata-rata Baris (Hasil Bobot)

• Bobot Manual: $\frac{0.677+0.692+0.636}{3} = 0.668 \approx \mathbf{0.66}$

• Bobot *Hybrid*: $\frac{0.226+0.231+0.273}{3} = 0.243 \approx \mathbf{0.24}$

• Bobot Otomatis: $\frac{0.097+0.077+0.091}{3} = 0.088 \approx \mathbf{0.08}$

iii. Hasil Bobot: Manual (0.66), *Hybrid* (0.24), Otomatis (0.08).

(b) Terhadap K2: Kelayakan implementasi Tugas Akhir

i. Justifikasi: Manual paling layak (risiko teknis rendah). Otomatis dan *Hybrid* tidak begitu *feasible* karena memerlukan keahlian NLP yang berada di luar lingkup Tugas Akhir ini. Manual dinilai mutlak lebih layak (9) dari otomatis dan sangat lebih layak (7) dari *Hybrid*.

ii. Rincian Perhitungan:

A. Matriks Perbandingan (dalam Desimal)

Tabel .7 Matriks Perbandingan Alternatif terhadap K2

vs K2	Manual	*Hybrid*	Otomatis
Manual	1.000	7.000	9.000
Hybrid	0.143	1.000	2.000
Otomatis	0.111	0.500	1.000
Jumlah	**1.254**	**8.500**	**12.000**

B. Matriks Ternormalisasi

- Perhitungan untuk baris Manual:
  - Kolom Manual: $\frac{1.000}{1.254} = 0.797$
  - Kolom *Hybrid*: $\frac{7.000}{8.500} = 0.824$
  - Kolom Otomatis: $\frac{9.000}{12.000} = 0.750$
- Perhitungan untuk baris *Hybrid*:
  - Kolom Manual: $\frac{0.143}{1.254} = 0.114$
  - Kolom *Hybrid*: $\frac{1.000}{8.500} = 0.118$
  - Kolom Otomatis: $\frac{2.000}{12.000} = 0.167$
- Perhitungan untuk baris Otomatis:
  - Kolom Manual: $\frac{0.111}{1.254} = 0.089$
  - Kolom *Hybrid*: $\frac{0.500}{8.500} = 0.059$
  - Kolom Otomatis: $\frac{1.000}{12.000} = 0.083$

Hasilnya disusun dalam Tabel 8.

Tabel .8 Matriks Ternormalisasi untuk Kriteria K2

vs K2	Manual	Hybrid	Otomatis
Manual	0.797	0.824	0.750
Hybrid	0.114	0.118	0.167
Otomatis	0.089	0.059	0.083

C. Rata-rata Baris (Hasil Bobot)

- Bobot Manual: $\frac{0.797+0.824+0.750}{3} = 0.790 \approx \mathbf{0.79}$
- Bobot Hybrid: $\frac{0.114+0.118+0.167}{3} = 0.133 \approx \mathbf{0.13}$
- Bobot Otomatis: $\frac{0.089+0.059+0.083}{3} = 0.077 \approx \mathbf{0.10}$

iii. Hasil Bobot: Manual (0.79), *Hybrid* (0.13), Otomatis (0.10).

(c) Terhadap K3: Skalabilitas

i. Justifikasi: Otomatis paling skalabel karena dapat memproses ribuan dokumen. Hybrid di urutan kedua, dan Manual sangat tidak skalabel. Berdasarkan penilaian, Otomatis dinilai sangat lebih skalabel (7) dari Manual, dan sedikit lebih skalabel (3) dari Hybrid. Selain itu, Hybrid juga dinilai sedikit lebih skalabel (3) dari Manual.

ii. Rincian Perhitungan:

A. Matriks Perbandingan (dalam Desimal)

Tabel .9 Matriks Perbandingan Alternatif terhadap K3

vs K3	Manual	Hybrid	Otomatis
Manual	1.000	0.333	0.143
Hybrid	3.000	1.000	0.333
Otomatis	7.000	3.000	1.000
Jumlah	**11.000**	**4.333**	**1.476**

B. Matriks Ternormalisasi

- Perhitungan untuk baris Manual:
  - Kolom Manual: $\frac{1.000}{11.000} = 0.091$
  - Kolom Hybrid: $\frac{0.333}{4.333} = 0.077$
  - Kolom Otomatis: $\frac{0.143}{1.476} = 0.097$
- Perhitungan untuk baris Hybrid:
  - Kolom Manual: $\frac{3.000}{11.000} = 0.273$
  - Kolom Hybrid: $\frac{1.000}{4.333} = 0.231$
  - Kolom Otomatis: $\frac{0.333}{1.476} = 0.226$

- Perhitungan untuk baris Otomatis:
  - Kolom Manual: $\frac{7.000}{11.000} = 0.636$
  - Kolom Hybrid: $\frac{3.000}{4.333} = 0.692$
  - Kolom Otomatis: $\frac{1.000}{1.476} = 0.677$

Hasilnya disusun dalam Tabel 10.

Tabel .10 Matriks Ternormalisasi untuk Kriteria K3

vs K3	Manual	Hybrid	Otomatis
Manual	0.091	0.077	0.097
Hybrid	0.273	0.231	0.226
Otomatis	0.636	0.692	0.677

C. Rata-rata Baris (Hasil Bobot)

- Bobot Otomatis: $\frac{0.636+0.692+0.677}{3} = 0.668 \approx \mathbf{0.65}$
- Bobot Hybrid: $\frac{0.273+0.231+0.226}{3} = 0.243 \approx \mathbf{0.25}$
- Bobot Manual: $\frac{0.091+0.077+0.097}{3} = 0.088 \approx \mathbf{0.10}$

iii. Hasil Bobot: Otomatis (0.65), *Hybrid* (0.25), Manual (0.10).

(d) Terhadap K4: Efisiensi keterlibatan pakar

i. Justifikasi: Otomatis paling efisien (intervensi pakar minimal). *Hybrid* membutuhkan waktu untuk validasi. Manual paling boros karena membutuhkan waktu lama dari pakar. Otomatis dinilai sangat lebih efisien (7) dari manual, dan sedikit lebih efisien (3) dari Hybrid.

ii. Rincian Perhitungan:

A. Matriks Perbandingan (dalam Desimal)

Tabel .11 Matriks Perbandingan Alternatif terhadap K4

vs K4	Manual	Hybrid	Otomatis
Manual	1.000	0.333	0.143
Hybrid	3.000	1.000	0.333
Otomatis	7.000	3.000	1.000
Jumlah	**11.000**	**4.333**	**1.476**

B. Matriks Ternormalisasi

- Perhitungan untuk baris Manual:
  - Kolom Manual: $\frac{1.000}{11.000} = 0.091$
  - Kolom Hybrid: $\frac{0.333}{4.333} = 0.077$
  - Kolom Otomatis: $\frac{0.143}{1.476} = 0.097$
- Perhitungan untuk baris Hybrid:
  - Kolom Manual: $\frac{3.000}{11.000} = 0.273$
  - Kolom Hybrid: $\frac{1.000}{4.333} = 0.231$
  - Kolom Otomatis: $\frac{0.333}{1.476} = 0.226$
- Perhitungan untuk baris Otomatis:
  - Kolom Manual: $\frac{7.000}{11.000} = 0.636$
  - Kolom Hybrid: $\frac{3.000}{4.333} = 0.692$
  - Kolom Otomatis: $\frac{1.000}{1.476} = 0.677$

Hasilnya disusun dalam Tabel 12.

Tabel .12 Matriks Ternormalisasi untuk Kriteria K4

vs K4	Manual	Hybrid	Otomatis
Manual	0.091	0.077	0.097
Hybrid	0.273	0.231	0.226
Otomatis	0.636	0.692	0.677

C. Rata-rata Baris (Hasil Bobot)

- Bobot Otomatis: $\frac{0.636+0.692+0.677}{3} = 0.668 \approx \mathbf{0.62}$
- Bobot Hybrid: $\frac{0.273+0.231+0.226}{3} = 0.243 \approx \mathbf{0.27}$
- Bobot Manual: $\frac{0.091+0.077+0.097}{3} = 0.088 \approx \mathbf{0.11}$

iii. Hasil Bobot: Otomatis (0.62), *Hybrid* (0.27), Manual (0.11).