BAB 2
LANDASAN TEORI

Bab ini memaparkan landasan teori yang menjadi dasar pengembangan dan perancangan sistem pada penelitian ini. Pembahasan meliputi konsep umum hingga metrik evaluasi yang akan digunakan.

2.1 Tanda Vital

Tanda vital merupakan sekumpulan pengukuran fisiologis yang digunakan oleh tenaga medis untuk mengetahui kondisi fisik seseorang secara umum. Dalam praktik klinis, terdapat lima parameter utama yang rutin dipantau, yaitu: laju pernapasan, detak jantung, suhu tubuh, tekanan darah, dan saturasi oksigen [11]. Adapun perbandingan rentang normal dan abnormal pada tanda vital dapat dilihat pada Tabel 2.1.

Tabel 2.1. Perbandingan rentang normal dan abnormal tanda vital manusia

Komponen	Rentang Normal	Rentang Abnormal
Laju pernapasan	12–20 kali per menit.	Kurang dari 10 atau lebih dari 24 kali per menit.
Detak jantung	60–100 bpm.	Kurang dari 50 bpm atau lebih dari 120 bpm.
Suhu tubuh	36,5–37,5°C atau kisaran 36,6°C.	Kurang dari 35,5°C atau $\geq$ 37,5°C.
Tekanan darah	Sistolik $<$ 120 mmHg dan Diastolik $<$ 80 mmHg.	Sistolik $>$ 160 atau $<$ 90 mmHg. Diastolik $>$ 100 atau $<$ 60 mmHg.
Saturasi oksigen	Optimal pada rentang 95% hingga 100%.	Di bawah 95%.

Sumber: [12]

2.2 Sistem Deteksi Dini (Early Warning System)

Secara umum, *Early Warning System* (EWS) merupakan sebuah sistem terintegrasi yang menggabungkan teknologi, prosedur serta kesiapsiagaan untuk mengurangi risiko bencana sebelum peristiwa berbahaya terjadi [13]. Sebuah sistem deteksi dini terdiri empat komponen yaitu:

1. *Disaster risk knowledge*: komponen ini merupakan fondasi dari EWS. Fokus utama pada komponen ini adalah memahami ancaman secara mendalam sebelum bencana terjadi.

2. *Observations, monitoring, analysis, and forecasting*: komponen kedua ini berfokus pada deteksi ancaman, penetapan ambang batas (*threshold*), dan analisis presisi. Agar dapat mendeteksi sebuah ancaman, diperlukan implementasi teknologi atau sebuah mekanisme pemantauan untuk mengawasi parameter alam. Penetapan ambang batas dilakukan untuk menentukan nilai indikator tertentu yang menjadi pemicu (*trigger*) peringatan bahwa suatu ancaman sedang mendekat. Analisis presisi dilakukan untuk memastikan pemantauan dianalisis dengan akurat.

3. *Warning dissemination*: komponen ini memastikan seluruh informasi peringatan sampai kepada semua orang dengan menggunakan berbagai saluran komunikasi.

4. *Preparedness and response capabilities*: komponen terakhir berkaitan dengan kelompok penduduk yang rentan sehingga rencana perlindungan dan kesiapan harus dikembangkan.

Pada dunia kesehatan, *Early Warning System* (EWS) merupakan sebuat alat deteksi dini yang digunakan untuk mengenali tanda-tanda awal perburukan pada seseorang melalui pemberian skor terhadap parameter fisiologis. Secara teknis, sistem ini mengukur lima tanda vital utama yaitu, laju pernapasan, detak jantung, suhu tubuh, tekanan darah, dan saturasi oksigen. Keberhasilan sistem deteksi ini sangat bergantung pada perawat dalam menganalisis skor numerik sebagai landasan pengambilan keputusan klinis [14]. Penggunaan EWS terbukti meningkatkan hasil klinis pasien dengan menurunkan risiko kematian hingga 20,26 kali lipat dibanding tanpa menggunakan EWS [15]. Selain itu, penggunaan EWS di IGD berhasil meningkatkan kepatuhan akan pemantauan tanda vital sebesar 72,5% serta mempercepat respon klinis [16].

2.3 Standar Interoperabilitas FHIR

Health Level 7 (HL7) merupakan sebuah organisasi pengembang standar yang berfokus pada peningkatan pertukaran informasi antar sistem kesehatan. *Fast Healthcare Interoperability Resources* (FHIR) merupakan sebuah standar generasi terbaru dari HL7 yang dibuat untuk memfasilitasi pertukaran data kesehatan secara elektronik. FHIR berfokus pada kemudahan interoperabilitas semantik. Standar ini menggunakan konsep "*Resources*" sebagai unit dasar pertukaran data sehingga memungkinkan informasi klinis seperti tanda vital untuk direpresentasikan secara terstruktur dan konsisten di berbagai sistem kesehatan yang berbeda [17].

Secara teknis, HL7 FHIR mendukung tiga format dalam merepresentasikan data untuk pertukaran informasi yaitu, *Extensible Markup Language* (XML), *JavaScript Object Notation* (JSON), dan *Resource Description Framework* (RDF). XML merupakan format tradisional yang sering digunakan pada sistem warisan. JSON merupakan format yang ringan dan efisien, sangat populer dalam pengembangan aplikasi *web* dan integrasi dengan *database* NoSQL [18]. RDF merupakan format yang digunakan untuk pemodelan data berbasis *linked data*. Perbandingan representasi kode dalam tiga format berbeda dapat dilihat pada Tabel 2.2.

Tabel 2.2. Perbandingan representasi FHIR dalam XML, JSON, dan RDF

Fitur	**XML**	**JSON**	**RDF (Turtle)**
Pembungkus Utama	Elemen `<Observation>`	Properti `"resourceType": "Observation"`	Subjek `a fhir:Observation`
Gaya Penulisan	Berbasis *Tag*/Elemen (`<tag>`)	Pasangan *Key-Value* (`"key": "value"`)	*Triples* (Subjek-Predikat-Objek)
Status Klinis	`<status value="..."/>`	`"status": "..."`	`fhir:Observation .status`
Identitas (LOINC)	Elemen `<code>`	Objek `"code"`	`fhir:Observation .code`
Nilai (UCUM)	Elemen `<valueQuantity>`	Objek `"valueQuantity"`	`fhir:Observation .valueQuantity`

Implementasi format XML pada Kode 2.1 menunjukkan struktur hierarki yang kaku dengan penggunaan elemen bersarang sebagai pembungkus data. Dalam standar FHIR, XML memanfaatkan atribut `value` di dalam elemen untuk menyimpan data primitif, seperti yang terlihat pada penetapan status dan nilai kuantitas. Penggunaan *namespace* pada elemen utama menjamin bahwa seluruh terminologi yang digunakan merujuk pada skema standar HL7, sehingga struktur data tetap valid dan terorganisir secara formal.

```
<Observation xmlns="http://hl7.org/fhir">
  <status value="final"/>
  <code>
    <coding>
      <system value="http://loinc.org"/>
      <code value="8867-4"/>
      <display value="Heart rate"/>
    </coding>
  </code>
  <valueQuantity>
    <value value="60"/>
    <unit value="bpm"/>
  </valueQuantity>
</Observation>
```

Kode 2.1: Contoh potongan kode XML FHIR

Berbeda dengan XML, format JSON pada Kode 2.2 menawarkan representasi yang lebih ringkas dan ringan melalui pasangan *key-value*. Jenis sumber daya diidentifikasi secara eksplisit menggunakan properti `"resourceType"`, yang memudahkan mesin pencari dan aplikasi web untuk melakukan pemetaan data. Struktur *array* pada bagian `"coding"` memberikan fleksibilitas dalam menampung informasi kode terminologi tanpa beban *tag* pembuka dan penutup yang repetitif, menjadikan JSON sebagai sebuah format yang paling populer dalam pertukaran data API modern.

```
{
  "resourceType": "Observation",
  "status": "final",
  "code": {
    "coding": [{
      "system": "http://loinc.org",
      "code": "8867-4",
      "display": "Heart rate"
    }]
  },
  "valueQuantity": {
    "value": 60,
    "unit": "bpm"
  }
}
```

Kode 2.2: Contoh potongan kode JSON FHIR

Format RDF dengan sintaks Turtle pada Kode 2.3 menyajikan data dalam bentuk pernyataan *triple* (Subjek-Predikat-Objek) yang sangat mendukung interoperabilitas semantik. Setiap elemen data dihubungkan melalui *prefix* `fhir:` yang merujuk pada ontologi global, memungkinkan integrasi data antar sistem yang berbeda tanpa kehilangan makna konteksnya. Penggunaan format ini sangat krusial pada arsitektur *Semantic Web*, di mana setiap informasi klinis tidak hanya sekadar teks, melainkan entitas yang saling terhubung secara logis dalam graf data yang luas.

```
@prefix fhir: <http://hl7.org/fhir/> .

<http://hl7.org/fhir/Observation/example> a fhir:Observation
    ;
  fhir:Observation.status [ fhir:value "final" ] ;
  fhir:Observation.code [
    fhir:CodeableConcept.coding ( [
      fhir:Coding.code [ fhir:value "8867-4" ] ;
      fhir:Coding.display [ fhir:value "Heart rate" ]
    ] )
  ] ;
  fhir:Observation.valueQuantity [
    fhir:Quantity.value [ fhir:value "60" ] ;
    fhir:Quantity.unit [ fhir:value "bpm" ]
  ] .
```

Kode 2.3: Contoh potongan kode RDF FHIR

2.4 Agentic Artificial Intelligence (Agentic AI)

Agentic AI merupakan generasi terbaru dalam bidang kecerdasan buatan yang memiliki karakteristik utama berupa otonomi, adaptabilitas, dan kemampuan penalaran yang memungkinkan untuk mengatasi tantangan dalam manajemen medis. Dalam konteks kesehatan, *Agentic AI* berperan sebagai sebuah sistem pengambil keputusan yang cerdas dengan mengintegrasikan berbagai sumber data multimodal [19]. *Agentic AI* mampu untuk merencanakan langkah-langkah penyelesaian secara mandiri serta berinteraksi dengan alat atau basis pengetahuan eksternal untuk mencapai tujuan tertentu. Dengan kemampuan otonom, sistem ini dapat melakukan persepsi, penalaran, dan tindakan secara independen guna untuk mencapai tujuan klinis tertentu, seperti deteksi anomali pada tanda vital [20].

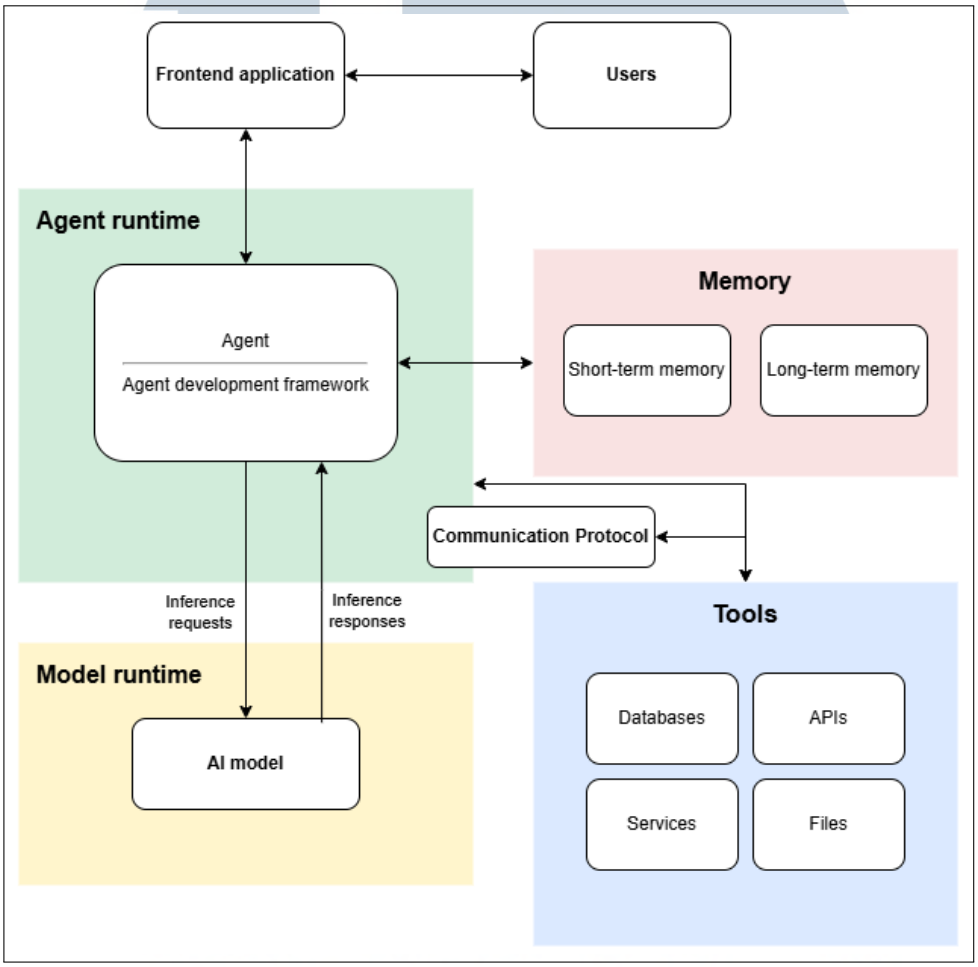
Cara kerja *Agentic AI* didasarkan pada lima karakteristik utama yaitu:

1. *Autonomous Operation*: adanya kemampuan untuk bertindak secara independen untuk mencapai tujuan.
2. *Goal-Oriented Behavior*: mengejar sebuah tujuan spesifik sehingga seringkali memerlukan beberapa langkah penalaran.
3. *Perception and Adaptation*: mampu untuk memahami lingkungan mereka dan menyesuaikan strategi berdasarkan umpan balik *real-time*.
4. *Tool Utilization*: dapat memanggil serta menggunakan berbagai alat untuk menyelesaikan sebuah tugas.
5. *Learning and Improvement*: agen dapat belajar dari pengalaman untuk menyempurnakan performa.

Gambar 2.1 mengilustrasikan integrasi komponen teknis yang mendukung kelima karakteristik utama. *Autonomous operation* direpresentasikan oleh blok *agent runtime*. Agen ini bekerja menggunakan *framework* tertentu untuk mengambil keputusan secara mandiri. Karakteristik *goal-oriented behavior* direpresentasikan melalui alur *inference requests* (panah ke blok kuning) yang merupakan proses bolak-balik antara agen dan *AI model*. Hal ini menunjukkan langkah-langkah penalaran (*reasoning*) untuk memecahkan tugas demi mencapai tujuan spesifik.

Aspek *perception and adaptation* direpresentasikan oleh *communication protocol* dan *short-term memory*. Dalam hal ini, agen memahami lingkungan

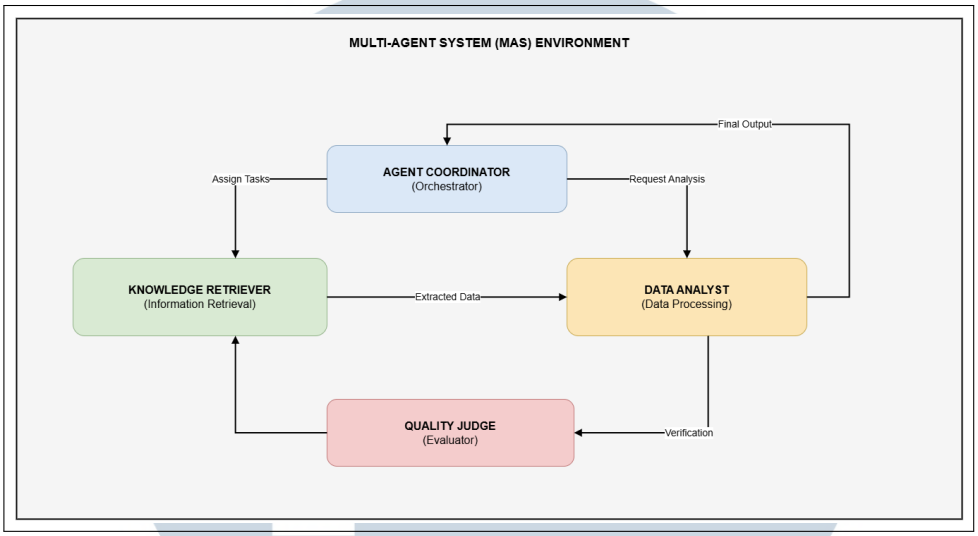
melalui data dari *users* maupun *tools*, lalu menyesuaikan strategi berdasarkan konteks terkini. *Tool Utilization* terlihat pada blok *tools* yang menghubungkan agen secara langsung dengan *databases*, *APIs*, serta layanan eksternal lainnya. Terakhir, aspek *learning and improvement* direpresentasikan oleh komponen *memory* melalui *long-term memory* yang menyimpan pengalaman masa lalu untuk menyempurnakan performa agen di masa mendatang.



Gambar 2.1. Arsitektur Agentic AI
Sumber: [21]

Terdapat *multi-agent systems* (MAS), kumpulan dari berbagai agen AI khusus dirancang untuk bekerja sama dalam suatu sistem yang terintegrasi untuk menyelesaikan masalah. Dalam sistem ini, MAS memecah masalah yang besar menjadi sub-masalah yang lebih kecil. Setiap agen akan berfokus pada suatu sub-masalah dan menyelesaikan sub-masalah tersebut secara bersamaan. Untuk memastikan interaksi antar agen lancar, terdapat sebuah agen pengawas yang telah

ditentukan sebelum mengoordinasikan aktivitas masing-masing agen [22]. Ilustrasi MAS terdapat pada Gambar 2.2.



Gambar 2.2. Ilustrasi MAS
Sumber: [23]

MAS bekerja melalui koordinasi terstruktur antara beberapa agen spesialis untuk menyelesaikan tugas yang kompleks secara otomatis. Alur dimulai dari *Agent Coordinator* yang berfungsi sebagai orkestrator utama; ia menerima instruksi dari pengguna, merencanakan strategi penyelesaian, dan membagi tugas kepada agen lainnya. Selanjutnya, *Knowledge Retriever* bertugas melakukan pencarian informasi dari berbagai sumber literatur atau basis data untuk menyediakan konteks yang akurat bagi sistem. Informasi tersebut kemudian diolah oleh *Data Analyst*, sebuah agen generatif yang mampu menulis dan mengeksekusi kode secara langsung untuk memproses data mentah menjadi wawasan ilmiah atau teknis. Terakhir, hasil analisis dievaluasi oleh *Quality Judge* menggunakan metode perbandingan berpasangan (*pairwise comparison*) untuk memastikan kualitas dan konsistensi sebelum hasil akhir diberikan kembali kepada pengguna. Seluruh proses ini membentuk sebuah siklus umpan balik yang memungkinkan sistem menyempurnakan hipotesis atau solusi secara berulang (iteratif).

2.5 N8N

N8N merupakan sebuah platform otomasi berbasis *open-source* yang memungkinkan pengguna untuk membuat serta mengelola alur kerja *workflow* secara otomatis. Platform ini mendukung integrasi dengan lebih dari 800 API dan layanan pihak ketiga, seperti Open AI GPT-4, sehingga memungkinkan pengguna untuk mengembangkan agen berbasis AI yang canggih tanpa menulis kode. N8N menggunakan konsep berbasis *node*. Setiap *node* merepresentasikan operasi atau tugas tertentu, seperti pengumpulan data, transformasi data, interaksi dengan API hingga penyimpanan data. Dengan menghubungkan *node-node* tersebut, pengguna dapat membentuk sebuah alur kerja [24]. Visualisasi dari sistem berbasis *node* tersebut dapat dilihat pada Gambar 2.3, yang menunjukkan bagaimana berbagai operasi dihubungkan untuk membentuk satu kesatuan alur kerja.

Gambar 2.3. Antarmuka N8N yang menunjukkan rangkaian *node* dalam sebuah *workflow*.

N8N mendukung alur kerja berbasis peristiwa (*event-driven workflows*. Alur kerja ini akan berjalan apabila dipicu oleh kejadian eksternal seperti pesan masuk. Selain itu, N8N juga mendukung alur kerja terjadwal (*scheduled workflows*) yang berjalan pada waktu tertentu. Kedua jenis alur kerja ini mendukung fleksibilitas dalam merancang otomasi yang bersifat responsif maupun periodik. Kemudian untuk menyeimbangkan kemudahan dalam penggunaan, N8N digunakan melalui pendekatan *drag-and-drop*. Adapun pengembang yang ingin memperluas fungsionalitas dapat melakukan penambahan skrip kustomisasi. Dengan demikian, platform ini menjadi sebuah solusi penting bagi organisasi yang ingin melakukan otomatisasi proses bisnis maupun mengembangkan sebuah *AI Agent* [24].

Integrasi dengan *Large Language Model* berfungsi sebagai mesin penalaran (*reasoning engine*) yang memungkinkan agen memecah tugas kompleks secara mandiri dan melakukan inferensi logika melalui pola *Reasoning and Acting* (ReAct). Melalui mekanisme ini, agen dapat melakukan dekomposisi masalah dan pengambilan keputusan dinamis berdasarkan konteks dari data yang tidak terstruktur. Hal ini diperkuat dengan adanya *conditional node* untuk percabangan logika eksplisit serta fitur penyimpanan data yang memungkinkan agen mempertahankan konteks dari interaksi sebelumnya [23].

Integrasi dengan *Large Language Model* juga membuat agen dapat berinteraksi dengan bahasa alami sehingga dapat digunakan untuk *chatbot*, hingga pengambilan keputusan. Selain itu, terdapat *conditional node*, sebuah node yang membuat agen dapat meniru proses berpikir manusia melalui percabangan logika berdasarkan input tertentu. *AI Agent* juga dapat mempertahankan konteks dari interaksi sebelumnya melalui fitur penyimpanan data [24].

Kemampuan integrasi API dalam N8N membuat *AI Agent* dapat berinteraksi dengan berbagai sistem eksternal seperti *database* dan *email*. Hal ini mendukung otomasi lintas aplikasi secara luas. Dengan fleksibilitas yang diberikan, pengguna dapat mengembangkan berbagai jenis *AI Agent* seperti *chatbot* layanan pelanggan, asisten pemasaran, hingga sistem automasi data untuk kebutuhan bisnis [25].

Pemilihan N8N sebagai platform utama untuk mengorkestrasi sistem *Agentic AI* pada penelitian ini didasarkan pada keunggulannya dalam menggabungkan kesederhanaan visual (*low-code visual building*) dengan fleksibilitas tingkat pengembang (*developer-grade flexibility*) jika dibandingkan dengan alternatif seperti LangGraph (LangChain), AutoGen, atau CrewAI. Berbeda dengan *code-first SDKs* seperti LangGraph dan CrewAI yang menuntut keahlian pemrograman mendalam serta manajemen kode dasar yang rumit, N8N menyediakan ekosistem dengan lebih dari 1.000 integrasi bawaan yang mempercepat otomatisasi alur kerja [26].

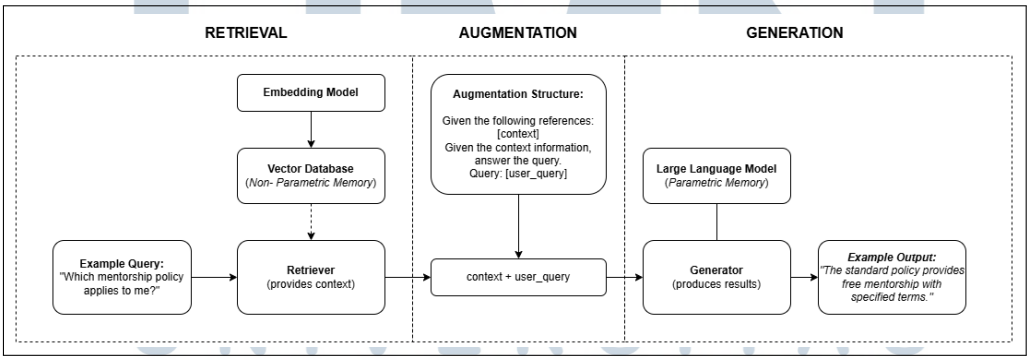
Melalui fitur *Agent-to-Agent workflows*, N8N memungkinkan pembentukan arsitektur multi-agen secara hierarkis maupun pararel tanpa perlu membangun alat khusus (*custom tools*) dari nol. Selain itu, platform ini menawarkan sistem pemantauan (*debugging*) dan evaluasi (*evals*) bawaan yang memberikan transparansi penuh terhadap eksekusi data, serta dukungan *human-in-the-loop* untuk memberikan umpan balik manual pada titik pemeriksaan kritis. Meskipun demikian, pemilihan n8n membawa konsekuensi *trade-off* berupa keterbatasan fleksibilitas penulisan kode tingkat rendah (*low-level control*) atas manajemen

memori dan arsitektur kognitif granular yang biasanya dimiliki oleh kerangka kerja murni seperti LangGraph [26].

Selain itu, jika dibandingkan dengan CrewAI yang unggul dalam pembentukan tim agen berbasis peran (*role-based teams*) untuk kolaborasi otonom pada tugas-tugas kreatif, N8N lebih berfokus pada struktur otomatisasi proses bisnis konvensional. Namun, *trade-off* tersebut dinilai dapat diterima karena fokus utama penelitian ini adalah menjamin stabilitas konektivitas data, efisiensi eksekusi token model, serta transparansi alur kerja klinis di mana N8N terbukti menjadi pemimpin yang jelas di antara opsi platform visual lainnya [26].

2.6 Retrieval-Augmented Generation (RAG)

Retrieval-Augmented Generation merupakan sebuah arsitektur yang mengintegrasikan mekanisme pencarian informasi eksternal menggunakan *Large Language Models* (LLMs). RAG dirancang untuk mengatasi keterbatasan pada model generatif statis, seperti menghasilkan informasi yang tidak akurat atau "halusinasi". RAG digunakan terutama pada tugas-tugas yang membutuhkan pengetahuan mendalam dan faktual [27]. Dengan menggunakan RAG, sistem akan dapat melakukan pencarian dinamis pada basis pengetahuan tertentu untuk mengambil dokumen yang relevan sebelum menghasilkan jawaban. Arsitektur RAG sederhana dapat dilihat pada Gambar 2.4.



Gambar 2.4. Arsitektur RAG
Sumber: [28]

RAG bekerja dengan menggabungkan kemampuan generatif dari *Large Language Model* (LLM) dengan pengetahuan eksternal yang diambil dari basis data terpisah. Pendekatan ini memanfaatkan dua kekuatan yaitu, *parametric memory* yang berupa informasi yang tersimpan dalam parameter model LLM dan

non-parametric memory yang berupa informasi di luar model, seperti dokumen organisasi atau Wikipedia untuk menyelesaikan tugas-tugas. Proses kerja RAG dalam sebuah *pipeline* terdiri tiga bagian. Pertama, *retrieval* (pengambilan), tahap ini dimulai ketika pengguna memberikan *query*. Setelah menerima *query*, sistem kemudian melakukan pencarian informasi yang paling relevan di dalam *vector database* dengan menggunakan metode *similarity scoring* antara vektor kueri dengan dokumen yang berada di basis data [28].

Konteks yang ditemukan pada tahap *retrieval* digunakan di tahap selanjutnya yaitu, *augmentation* (augmentasi). Tahap *augmentation* menggunakan konteks sebelumnya untuk memperluas *query* asli pengguna. Sistem menggunakan *augmentation template* yang menentukan bagaimana *query* dari pengguna digabungkan dengan konteks tersebut. Secara eksplisit, instruksi diberikan kepada model untuk menggunakan informasi yang disediakan, misalnya: "Gunakan konteks berikut: [konteks]. Berdasarkan informasi tersebut, jawablah pertanyaan: [kueri]". Tahap akhir dari RAG adalah *generation* (generasi), sebuah proses saat generator mengambil *query* yang telah diperkaya dengan informasi tambahan untuk menghasilkan sebuah jawaban baru [28].

Dalam konteks *agentic AI*, *Retrieval-Augmented Generation* (RAG) digunakan sebagai mekanisme utama untuk memperoleh dan memanfaatkan pengetahuan eksternal secara dinamis. *Agentic AI* berperan sebagai *controller* yang mengorkestrasi seluruh *pipeline* RAG, yaitu *retrieval*, *augmentation*, dan *generation*. Pada tahap *retrieval*, agen menentukan *query* untuk mengambil informasi relevan dari *vector database*. Selanjutnya, pada tahap *augmentation*, hasil *retrieval* disusun menjadi konteks dalam *prompt*. Kemudian, pada tahap *generation*, model bahasa menghasilkan jawaban yang lebih akurat berdasarkan konteks tersebut. Selain itu, *agentic AI* mendukung proses iteratif melalui mekanisme *think–act–observe*, sehingga agen dapat mengevaluasi dan menyempurnakan hasil secara adaptif [29].

Adapun pada konteks dunia kesehatan, implementasi RAG menjadi sangat penting karena adanya tuntutan terhadap konsistensi faktual dan keamanan pasien yang sangat tinggi. RAG bekerja dengan mengambil bukti klinis dari sumber eksternal yang tepercaya untuk memperkuat jawaban yang dihasilkan oleh LLM. Hal ini dilakukan agar saran atau klasifikasi medis didasarkan pada literatur terkini [30]. Proses ini memungkinkan sistem AI untuk beradaptasi dengan domain medis yang spesifik secara cepat sehingga menghasilkan sebuah mekanisme pendukung keputusan yang dinamis [27, 30].

2.7 MongoDB

MongoDB adalah sistem manajemen basis data NoSQL (*Not Only SQL*) yang bersifat *document-based*, artinya tersusun atas koleksi dan dokumen. Terdapat empat karakteristik utama MongoDB:

1. Data disimpan dalam koleksi dan dokumen, bukan tabel. Format yang digunakan adalah *Binary JSON* (BSON), yaitu versi biner dari file JSON.
2. Basis data MongoDB tidak memiliki struktur data yang tetap, sehingga skema bersifat sangat fleksibel dan dapat berubah sesuai kebutuhan aplikasi.
3. MongoDB mendukung replikasi data melalui *replica set* untuk menjaga ketersediaan data (*high availability*).
4. Dokumen MongoDB mendukung penggunaan *lists*, *pointers*, *embedded arrays*, atau *nested documents* yang menyederhanakan akses data.

Kode 2.4 menunjukkan implementasi penyimpanan data kesehatan dalam format NoSQL menggunakan MongoDB. Di dalamnya, terdapat sebuah *resource* FHIR *Bundle* disimpan sebagai dokumen BSON yang memiliki identitas unik berupa `_id` (*ObjectId*). Struktur ini memperlihatkan sifat *nested object* atau objek bersarang, di mana elemen medis seperti `status`, `category` (*vital-signs*), dan `code` disusun secara hierarkis tanpa memerlukan tabel relasional yang kaku. Hal ini memungkinkan sistem untuk menangani variasi data medis dengan fleksibilitas tinggi namun tetap mempertahankan standar interoperabilitas internasional.

```
{
  "_id": "ObjectId('69e97dbc307168a820078749')",
  "resourceType": "Bundle",
  "type": "collection",
  "entry": [
    {
      "resource": {
        "resourceType": "Observation",
        "status": "final",
        "category": [
          {
            "coding": [
              {
                "system": "http://terminology.hl7.org/
   CodeSystem/observation-category",
                "code": "vital-signs",
                "display": "Vital Signs"
              }
            ]
          }
        ],
        "code": {
          "coding": [
            {
              "system": "http://loinc.org",
              "code": "8867-4",
              "display": "Heart rate"
            }
          ]
        },
        "subject": {
          "reference": "Patient/undefined"
        },
        "effectiveDateTime": "2026-04-23T02:02:22.543Z",
        "valueQuantity": {
          "value": 72,
          "unit": "bpm",
          "system": "http://unitsofmeasure.org",
          "code": "/min"
        }
      }
    }
  ]
}
```

Kode 2.4: Implementasi penyimpanan data di MongoDB

Dalam konteks AI modern, terdapat MongoDB Atlas, sebuah platform yang dapat menangani *vector search* (pencarian vektor). MongoDB Atlas memungkinkan penyimpanan *vector embeddings*, yaitu representasi numerik dari data tekstual atau multimodal dalam ruang matematis berdimensi tinggi. Alih-alih hanya mencari kata kunci (*keyword*), sistem ini melakukan pencarian berdasarkan makna semantik menggunakan algoritma *Approximate Nearest Neighbor* (ANN).

Implementasi *Retrieval-Augmented Generation* menggunakan MongoDB Atlas bekerja dengan menggabungkan dokumen eksternal yang relevan ke dalam

proses generasi jawaban oleh *Large Language Model* (LLM). Alur kerjanya dimulai dengan memecah dokumen menjadi potongan kecil (*chunking*) menggunakan algoritma seperti *RecursiveCharacterTextSplitter*, yang kemudian diubah menjadi vektor dan disimpan dalam indeks pencarian MongoDB. Ketika pengguna memberikan pertanyaan, sistem akan melakukan *Similarity-Based Retrieval* untuk mengambil potongan konteks yang paling relevan, lalu menggabungkannya ke dalam *prompt* terstruktur agar LLM menghasilkan jawaban yang akurat.

2.8 Confusion Matrix

Confusion Matrix merupakan sebuah tabel dua dimensi yang digunakan untuk mengevaluasi kinerja sebuah model klasifikasi dengan membandingkan hasil prediksi dengan kondisi sebenarnya. Tabel ini terdiri dari empat komponen utama, yaitu *True Positive* (TP), *True Negative* (TN), *False Positive* (FP), dan *False Negative* (FN) [31]. Dalam evaluasi sistem medis, penggunaan tabel ini sangat krusial untuk membedakan antara kesalahan diagnosis seperti *false negative* dengan jenis kesalahan klasifikasi yang lain untuk menjamin keselamatan pasien. Integrasi komponen evaluasi ini menjadi sebuah dasar yang valid untuk menentukan seberapa konsisten dan akurat sistem cerdas yang dibangun dalam memberikan hasil [32]. Adapun representasi tabel *confusion matrix* ditunjukkan pada Tabel 2.3.

Tabel 2.3. *Confusion matrix*

	Prediksi: Positif	Prediksi: Negatif
Aktual: Positif	*True Positive* (TP)	*False Negative* (FN)
Aktual: Negatif	*False Positive* (FP)	*True Negative* (TN)

Keterangan:

1. *True Positive*: sistem memprediksi positif dan kondisi aktualnya positif.
2. *True Negative*: sistem memprediksi negatif dan kondisi aktualnya negatif.
3. *False Positive*: sistem memprediksi positif namun kondisi aktual sebenarnya negatif.
4. *False Negative*: sistem memprediksi negatif namun kondisi aktual sebenarnya positif.

Berdasarkan keempat komponen dalam tabel *confusion matrix*, digunakan beberapa metrik untuk mengevaluasi performa model secara menyeluruh [32].

Metrik-metrik tersebut meliputi akurasi, presisi, sensitivitas, spesifisitas dan F1-score. Akurasi merupakan metrik untuk menunjukkan seberapa sering model membuat prediksi yang benar dari seluruh data. Metrik presisi merupakan metrik untuk menunjukkan seberapa banyak prediksi positif yang benar dari seluruh prediksi positif. Rumus metrik akurasi terdapat pada Rumus 2.1, sedangkan rumus metrik presisi terdapat pada Rumus 2.2.

Metrik sensitivitas / *recall* merupakan metrik yang digunakan untuk mengukur proporsi kasus positif yang sebenarnya yang berhasil diprediksi dengan benar oleh model. Rumus untuk metrik sensitivitas ditunjukkan oleh Rumus 2.3. Metrik spesifisitas digunakan untuk mengukur proporsi kasus negatif yang diprediksi dengan benar oleh model. Metrik F1-score digunakan untuk mengevaluasi kinerja model dengan mempertimbangkan metrik presisi dan sensitivitas (*recall*) secara bersamaan. Rumus metrik spesifisitas terdapat pada Rumus 2.4, sedangkan rumus metrik F1-score terdapat pada Rumus 2.5.

1. Akurasi (*Accuracy*):

$$Accuracy = \frac{TP + TN}{TP + TN + FP + FN} \tag{2.1}$$

2. Presisi (*Precision*):

$$Precision = \frac{TP}{TP + FP} \tag{2.2}$$

3. Sensitivitas / *Recall*:

$$Recall = \frac{TP}{TP + FN} \tag{2.3}$$

4. Spesifisitas (*Specificity*):

$$Specifity = \frac{TN}{TN + FP} \tag{2.4}$$

5. F1-Score:

$$\text{F1-Score} = 2 \times \frac{Precision \times Recall}{Precision + Recall} \tag{2.5}$$

Tabel 2.4 merupakan sebuah contoh penggunaan *confusion matrix* untuk melakukan analisis komprehensif terhadap performa model klasifikasi yang telah diuji. Terdapat 239 data positif yang berhasil diprediksi dengan benar sebagai *true positive* dan 199 data negatif yang berhasil diidentifikasi secara tepat sebagai *true negative*. Namun, terdapat 52 data negatif yang salah diprediksi sebagai positif (*false positive*) dan 25 data positif yang salah diprediksi sebagai negatif (*false negative*). Distribusi angka ini memberikan gambaran bahwa meskipun model memiliki tingkat kesalahan, frekuensi kegagalan dalam mendeteksi data positif (*false negative*) lebih rendah dibandingkan kesalahan dalam memprediksi data negatif (*false positive*).

Efektivitas sistem ini dibuktikan melalui pencapaian akurasi sebesar 85.05%, yang menandakan bahwa mayoritas data telah diklasifikasikan ke dalam kategori yang sesuai. Nilai presisi sebesar 82.13% menjelaskan bahwa sebagian besar prediksi positif sistem memiliki ketepatan yang tinggi, sementara nilai *recall* yang menyentuh angka 90.53% menegaskan kemampuan model dalam menyaring seluruh data positif yang tersedia agar tidak terabaikan. F1-Score sebesar 86.14% mengonfirmasi bahwa model tidak mengalami ketimpangan performa antara presisi dan *recall*. Hal ini menunjukkan stabilitas model yang sangat baik dalam mendeteksi dan memprediksi, sehingga sistem ini dianggap reliabel untuk diimplementasikan pada skenario pengolahan data.

Tabel 2.4. Ilustrasi *confusion matrix*

	Prediksi: Positif	Prediksi: Negatif
Aktual: Positif	239	25
Aktual: Negatif	52	199