

BAB II

TINJAUAN PUSTAKA

2.1 Teori Dasar

Menurut Edward Feigenbaum (dikenal sebagai "Bapak Sistem Pakar"), sistem pakar adalah “*a computer program designed to simulate the problem-solving behavior of a human expert*”, yaitu program komputer yang dirancang untuk menirukan perilaku pemecahan masalah seorang pakar.

2.1.1 Artificial Intelligence (AI)

Artificial Intelligence (AI) merupakan cabang ilmu komputer yang berfokus pada pembuatan sistem atau mesin yang mampu melakukan tugas yang biasanya memerlukan kecerdasan manusia. AI mencakup berbagai teknologi seperti *machine learning*, *deep learning*, dan *natural language processing*. Dalam beberapa tahun terakhir, perkembangan AI semakin pesat karena dukungan data besar (big data) dan peningkatan kapasitas komputasi. Hal ini memungkinkan AI digunakan dalam berbagai sektor, mulai dari kesehatan, pendidikan, hingga industri manufaktur.

Salah satu perkembangan yang menonjol dalam bidang kecerdasan buatan(AI) merupakan aplikasi di bidang kesehatan, teknologi ini telah digunakan untuk mendiagnosa penyakit, memprediksi penyebaran wabah , dan membantu dalam pengembangan obat-obatan. Contohnya, algoritma *deep leaarning* dapat menganalisis citra medis dengan tingkat akurasi yang sangat tinggi, bahkan melampaui kemampuan manusia dalam beberapa situasi. Di samping itu AI berkontribusi dalam meningkatkan efisiensi operasional rumah sakit melalui sistem manajemen yang berbasis data (Pakpahan, 2021).

Meskipun menawarkan beberapa manfaat yang signifikan, penerapan AI juga dihadapkan pada sejumlah tantangan dan risiko etika. Masalah seperti bias dalam algoritma, perlindungan privasi data, serta tanggung jawab hukum terhadap keputusan yang diambil oleh sistem AI menjadi perhatian yang perlu ditanggapi dengan serius. Oleh karena itu, penting untuk memiliki regulasi yang jelas dan prinsip-prinsip etika yang kuat supaya penggunaan AI tetap aman dan adil bagi semua pihak (Harahap, 2025).

Di masa depan, AI akan menjadi bagian penting dalam kehidupan manusia, dengan kombinasi teknologi seperti IoT dan *blockchain* yang dapat mengubah cara kita bekerja dan berinteraksi. Kolaborasi antara ilmuwan, pembuat kebijakan, dan masyarakat diperlukan untuk memastikan perkembangan AI yang positif dan bertanggung jawab (Ifedayo, Olugbade, and Hamid 2025).

AI mencakup berbagai teknik dan metode, termasuk pembelajaran mesin (*machine learning*), pengolahan bahasa alami (*natural language processing*), pengenalan pola (*pattern recognition*), logika inferensial (*inferential reasoning*), dan sistem pakar (*expert systems*). Metode-metode ini digunakan untuk membangun algoritma dan model yang dapat mengambil keputusan atau melakukan tugas tertentu berdasarkan data yang diberikan.

2. Sistem Pakar

Sistem pakar merupakan sistem yang mencoba menerapkan pengetahuan manusia ke komputer sehingga komputer dapat memecahkan masalah seperti yang biasanya dilakukan oleh para ahli. Tujuan utama dari sistem pakar merupakan meniru pekerjaan para pakar dalam memecahkan masalah tertentu. Sistem pakar

bahkan memungkinkan orang biasa untuk memecahkan masalah kompleks yang tidak dapat diselesaikan tanpa bantuan pakar (Dwi Kurnia, Andryana, and Gunaryati 2021).

Secara umum, terdapat beberapa jenis sistem pakar yang memiliki fungsi berbeda-beda. Pertama, sistem pakar diagnostik digunakan untuk mendiagnosis masalah berdasarkan gejala yang ada, seperti pada dunia medis atau perbaikan mesin. Kedua, sistem pakar interpretasi berfungsi untuk menafsirkan data yang kompleks dan tidak langsung, misalnya dalam analisis citra satelit atau radiologi. Ketiga, sistem pakar prediktif digunakan untuk meramalkan suatu kejadian berdasarkan data dan pola yang telah ada, seperti dalam prediksi cuaca atau pasar saham. Keempat, sistem pakar perencanaan digunakan untuk menyusun strategi atau langkah-langkah dalam mencapai suatu tujuan, contohnya dalam manajemen proyek atau perencanaan logistik. Kelima, sistem pakar monitoring berperan dalam mengamati kondisi secara real-time dan memberikan peringatan apabila terjadi penyimpangan, seperti pada sistem pengawasan pasien atau sistem keamanan. Terakhir, sistem pakar instruksional berfungsi sebagai alat bantu belajar yang dapat memberikan panduan atau pelatihan kepada pengguna. Masing-masing jenis sistem pakar ini memainkan peran penting dalam berbagai bidang dan membantu manusia dalam membuat keputusan yang lebih cepat dan akurat.

Sistem pakar memiliki kemampuan untuk merekomendasikan tindakan atau perilaku pengguna untuk meluncurkan sistem dengan benar dan akurat. Tujuan sistem pakar merupakan membuat kesimpulan atau deskripsi dari informasi asli (data mentah). Keputusan dibuat berdasarkan hasil observasi dari analisis gambar,

pengenalan kata hingga ucapan, pembacaan tanda, dll. Sistem pakar merupakan program komputer yang meniru proses berpikir dan pengetahuan pakar dalam memecahkan masalah tertentu. Sistem pakar dapat diterapkan di bidang kesehatan (Kesumaningtyas and Handayani 2020). Mampu mendiagnosis penyebab malfungsi dengan benar berdasarkan gejala yang dapat diamati dalam situasi kompleks. Aplikasi sistem pakar banyak diminati di berbagai kalangan sejak tahun 1950, dengan cakupan yang luas daerah yang besar. Sistem Pakar pada organisasi ditujukan untuk menambah nilai, meningkatkan produktivitas serta bidang manajerial dapat mengambil keputusan dengan cepat dan tepat.

2.1.2 *Naive Bayes*

Naive Bayes merupakan algoritma pembelajaran yang mewakili setiap contoh (data) sebagai konjungsi nilai atributnya. *Naive Bayes* hanya dapat mengklasifikasikan data yang terbatas nilai fungsi target (kelas) itu berarti nilai kelas tidak bisa kontinu.

Naive Bayes telah banyak digunakan di banyak pembelajaran mesin karena kesederhanaan dan efisiensinya. baru-baru ini, *Naive Bayes* yang diatur telah dikembangkan untuk meningkatkan kinerja klasifikasi dengan menyeimbangkan kemampuan generalisasi dan daya pembeda pengklasifikasi secara otomatis melalui optimalisasi penurunan gradien menggunakan umpan balik klasifikasi (Wang, Ren, and Bai 2023).

Naive Bayes sangat cocok untuk klasifikasi biner dan multikelas, Juga dikenal sebagai pengklasifikasi *Naive Bayesian*, metode ini menerapkan teknik klasifikasi objek prospektif dengan menetapkan pengidentifikasi kelas ke kasus/kontribusi

menggunakan probabilitas bersyarat. Probabilitas bersyarat merupakan ukuran probabilitas suatu peristiwa berdasarkan peristiwa lain (dihipotesiskan, diasumsikan atau dibuktikan).

Metode *Naive Bayes* pernah di gunakan untuk mendeteksi penyakit pencernaan pada pasien. *Naive Bayes Classifier* merupakan suatu klasifikasi berpeluang sederhana berdasarkan aplikasi teoriema *Bayes* dengan asumsi antar variabel penjelas saling bebas (independen) yaitu kehadiran atau ketiadaan dari suatu kejadian tertentu dari suatu kelompok tidak berhubungan dengan kehadiran atau ketiadaan dari kejadian lainnya. Secara umum, teorema *Bayes* dinyatakan sebagai:

$$P(A|B) = \frac{P(A|B) \cdot P(A)}{P(B)} \quad \textbf{Rumus 2. 1 Rumus Probabilitas}$$

Dalam notasi ini $P(A|B)$ berarti peluang kejadian A bila B terjadi dan $P(B|A)$ peluang kejadian B bila A terjadi.

Dalam proses perhitungan dengan menggunakan metode *naive bayes* terdapat langkah – langkahm yang dapat dilakukan merupakan sebagai berikut :

- 1) Mencari nilai *prior* untuk tiap-tiap kelas dengan menghitung rata-rata tiap kelas.

$$P = \frac{x}{A} \quad \textbf{Rumus 2. 2 Mencari nilai prior}$$

Keterangan :

P = Nilai Prior

X = Jumlah data tiap kelas

A = Jumlah data seluruh kelas

- 2) Mencari nilai Likelihood untuk tiap-tiap kelas.

$$L = \frac{F}{B} \quad \textbf{Rumus 2. 3} \text{ Mencari nilai } \textit{likelihood}$$

Keterangan :

L = Nilai Likelihood

F = Jumlah data feature tiap kelas

B = Jumlah seluruh fitur tiap kelas

3) Mencari nilai posterior dari tiap-tiap kelas menggunakan persamaan.

$$P = (c | a) = P(c) \times P(a | c) \quad \textbf{Rumus 2. 4} \text{ Mencari nilai } \textit{posterior}$$

Keterangan :

P(c) = Nilai Prior tiap kelas

P(a|c) = Nilai Likelihood

2.1.3 *Forward Chaining*

Metode *forward chaining* bersifat *data-driven*, artinya sistem bekerja berdasarkan data atau fakta yang tersedia untuk menemukan semua kemungkinan kesimpulan. Keunggulan dari pendekatan ini merupakan kemampuannya dalam mengungkap banyak solusi atau skenario yang mungkin terjadi dari suatu kondisi awal. Dalam bidang medis, misalnya, *forward chaining* digunakan untuk mendiagnosis penyakit berdasarkan gejala-gejala yang dilaporkan oleh pasien. Setiap gejala menjadi fakta yang diproses untuk memunculkan kemungkinan diagnosis secara bertahap melalui aturan yang sudah ditentukan sebelumnya.

Namun, tantangan dari penggunaan *forward chaining* merupakan ketika basis data atau jumlah aturan terlalu besar, proses inferensi bisa menjadi lambat dan kurang efisien. Oleh karena itu, beberapa penelitian telah mencoba mengoptimalkan algoritma ini agar dapat bekerja lebih cepat dan akurat, terutama

dalam sistem pakar skala besar. *Forward chaining* tetap relevan dalam pengembangan sistem pakar modern, terutama jika dikombinasikan dengan teknik optimasi pencarian dan pembobotan aturan (Anam, Irawan, and Homaidi 2023).

2.1.4 Manfaat Sistem Pakar

Salah satu manfaat utamanya merupakan kemampuannya dalam memberikan diagnosis atau solusi terhadap masalah kompleks secara cepat dan akurat. Hal ini sangat berguna dalam situasi di mana ketersediaan pakar terbatas atau waktu respon yang cepat sangat dibutuhkan, seperti dalam bidang kesehatan, pertanian, dan teknik.

Selain itu, sistem pakar dapat meningkatkan efisiensi operasional dengan mengurangi ketergantungan pada tenaga ahli secara langsung. Dengan memanfaatkan basis pengetahuan yang telah dikumpulkan dari para pakar, sistem ini dapat memberikan rekomendasi atau tindakan yang konsisten tanpa kelelahan atau bias manusia. Sebagai contoh, dalam bidang kesehatan gigi, sistem pakar telah digunakan untuk membantu dalam mendiagnosa penyakit berdasarkan gejala yang dialami pasien.

Salah satu studi yang mendukung manfaat ini merupakan penelitian oleh Zaelani dan Winarti (2023) yang mengembangkan sistem pakar untuk mendiagnosa penyakit gigi dan mulut menggunakan metode *Naive Bayes*. Mereka menyatakan bahwa "sistem pakar dapat membantu masyarakat dalam mendiagnosa penyakit secara cepat dan akurat tanpa harus selalu bergantung pada dokter spesialis". Hal ini menunjukkan bahwa sistem pakar tidak hanya meningkatkan efisiensi layanan,

tetapi juga memperluas akses informasi medis kepada masyarakat luas (Zaelani and Winarti, 2023).

2.1.5 Metode Pengembangan Sistem

System Development Life Cycle (SDLC) merupakan metode terstruktur yang digunakan untuk mengembangkan dan mengelola sistem informasi dengan tujuan menghasilkan sistem yang berkualitas, efektif, dan sesuai kebutuhan pengguna. SDLC terdiri dari beberapa tahapan, yaitu perencanaan, analisis sistem, perancangan, pengembangan, pengujian, implementasi, dan pemeliharaan. Setiap tahap memiliki peran penting dalam memastikan bahwa sistem yang dibangun dapat berjalan dengan baik dan memenuhi harapan pengguna. Dalam pelaksanaannya.

Dalam pengembangan sistem informasi, terdapat berbagai jenis metode SDLC yang dapat digunakan sesuai dengan kebutuhan proyek. Salah satu metode yang paling tradisional merupakan *Waterfall*, di mana setiap tahapan dilakukan secara berurutan mulai dari perencanaan hingga pemeliharaan. Sementara itu, metode *Agile* menawarkan pendekatan yang lebih fleksibel dan iteratif dengan siklus kerja singkat untuk mengakomodasi perubahan kebutuhan secara cepat. Spiral menggabungkan elemen pengulangan dan analisis risiko, cocok untuk proyek yang kompleks dan berisiko tinggi. V-Model menekankan hubungan erat antara tahap pengembangan dan pengujian, dengan setiap langkah pengembangan memiliki pasangan tahap verifikasi. Ada juga metode *Iterative and Incremental*, yang membangun sistem secara bertahap melalui serangkaian iterasi, serta metode *Prototyping*, yang fokus pada pembuatan prototipe untuk mendapatkan umpan balik

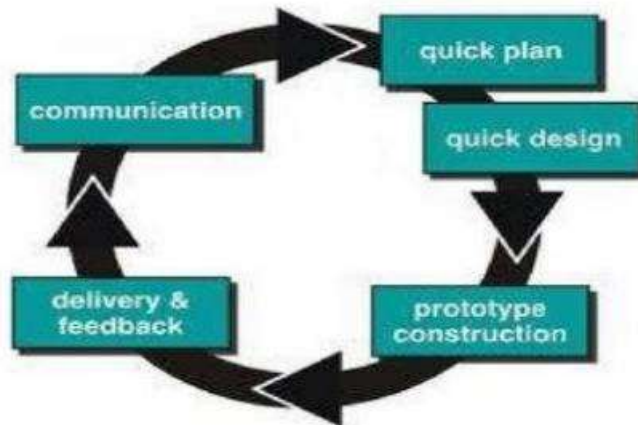
pengguna sebelum sistem akhir dikembangkan (Destiani et al. 2022). Untuk proyek kecil dengan kebutuhan yang tidak jelas, metode Big Bang terkadang digunakan, meskipun risikonya cukup tinggi. Pemilihan metode yang tepat sangat penting agar proses pengembangan sistem berjalan efektif dan sesuai dengan tujuan bisnis. untuk sistem pakar kesehatan cocok yang mana

Metode *prototyping* sangat cocok digunakan dalam pengembangan sistem pakar untuk mendeteksi penyakit. Hal ini disebabkan oleh karakteristik kebutuhan sistem yang sering melibatkan aspek subjektif, seperti gejala, tingkat keparahan, hingga rekomendasi penanganan yang bervariasi antar individu. Dengan menggunakan metode *prototyping*, pengembang dapat membuat model awal sistem yang langsung diuji oleh para ahli seperti dokter atau psikolog, sehingga mendapatkan umpan balik cepat untuk perbaikan dan penyempurnaan sistem. Selain itu, *prototyping* memungkinkan pengembangan antarmuka pengguna yang lebih intuitif dan nyaman, yang sangat penting dalam aplikasi yang berhubungan dengan kondisi mental pasien. Oleh karena itu, penggunaan metode *prototyping* dapat meningkatkan efektivitas, akurasi, dan penerimaan sistem pakar untuk penyakit depresi.

2.1.5.1 Prototyping

Untuk mengembangkan sistem pakar diagnosa kesehatan mental penyakit depresi, metode *prototyping* sangat efektif karena memungkinkan pengembang dan pakar medis berkolaborasi secara iteratif. Melalui pendekatan ini, versi awal sistem dapat diuji oleh pengguna untuk mendapatkan umpan balik yang digunakan dalam

penyempurnaan sistem. Hal ini memastikan bahwa sistem akhir benar-benar memenuhi kebutuhan dan harapan pengguna.



Gambar 2. 1 Sistem Prototyping

Sumber : (Dokumen, Kecamatan, and Kurniati 2021)

Tahapan-tahapan di atas merupakan proses dalam metode pengembangan sistem *Prototyping*, yang digunakan untuk membangun perangkat lunak secara iteratif berdasarkan masukan langsung dari pengguna. Berikut merupakan uraian lebih ringkas dan terstruktur dari masing-masing tahapan:

a. Communication

Tahap awal ini melibatkan komunikasi langsung antara *developer* dan *klien*. Tujuannya merupakan untuk memahami kebutuhan dasar, tujuan sistem, dan cakupan umum dari perangkat lunak yang akan dikembangkan. Di sini, pihak pengguna dapat menyampaikan ekspektasi awal terhadap sistem, sementara pengembang mulai membangun pemahaman terhadap domain masalah.

b. *Quick Plan*

Pada tahap ini dilakukan perencanaan cepat (*quick planning*) berdasarkan informasi yang diperoleh dari tahap komunikasi. Rencana ini menjadi fondasi dalam merancang struktur awal sistem, dan meskipun bersifat kasar, itu mencakup aspek-aspek penting untuk mendukung pembuatan prototipe secara efisien.

c. *Modelling Quick Design*

Fokus dari tahap ini merupakan pembuatan desain awal dari sistem, khususnya yang berkaitan dengan antarmuka pengguna dan interaksi dasar yang bisa segera diuji. Desain ini bersifat *low-fidelity* dan digunakan sebagai visualisasi awal sistem agar pengguna dapat memahami bagaimana sistem akan berfungsi.

d. *Construction of Prototype*

Di sini prototipe awal dari perangkat lunak mulai dibangun berdasarkan desain sebelumnya. Prototipe ini mencerminkan fungsi-fungsi inti sistem meskipun belum sempurna. Tujuannya merupakan memberikan bentuk nyata yang bisa dilihat, diuji, dan dievaluasi oleh pengguna.

e. *Delivery & Feedback*

Setelah prototipe diselesaikan, ia didistribusikan ke pengguna untuk diuji coba. Pengguna kemudian memberikan umpan balik mengenai kesesuaian sistem dengan kebutuhannya. Masukan ini digunakan untuk memperbaiki dan menyempurnakan sistem sebelum tahap akhir pengembangan.

2.2 Variable

Variabel merupakan salah satu unsur yang penting karena suatu proses pengumpulan fakta atau pengukuran dapat dilakukan dengan baik, sehingga dapat dirumuskan variabel penelitian dengan tegas. Proses perumusan variabel ini diawali dari perumusan konsep tentang segala sesuatu yang menjadi sasaran penelitian. Konsep yang dimaksud merupakan istilah dan definisi yang digunakan untuk menggambarkan secara abstrak tentang kejadian dan keadaan suatu kelompok atau individu tertentu yang menjadi sasaran penelitian.

Gangguan kesehatan mental (*Mental Health Disorder*) yaitu salah satu *syndrome* ataupun pola perilaku psikologi seseorang yang akan menyebabkan ketidakmampuan saat melakukan pekerjaan. Karena beragam kesulitan dan dilema yang muncul, banyaknya masalah yang tidak dapat dihadapi seseorang dalam kehidupan dapat mengganggu jaringan saraf otak manusia dan menyebabkan penyakit mental. Tingkat psikologis seseorang tergantung pada pengalaman emosional, perilaku, lingkungan maupun latar belakang pendidikan keluarga. (Dwi Kurnia, Andryana, and Gunaryati 2021). Berikut variabel yang dipakai dalam penelitian sistem pakar gangguan kesehatan mental meliputi :

1. Variabel Gejala dan Tanda Gangguan Kesehatan Mental

Ini termasuk gejala fisik dan psikologis yang dialami oleh seseorang yang menderita gangguan kesehatan mental. Beberapa contoh gejala yang umum pada gangguan kesehatan mental meliputi perubahan suasana hati, gangguan tidur, ketakutan dan kecemasan yang berlebihan, perubahan berat badan, dan gangguan memori.

2. Variabel Diagnosa

Tingkat depresi dapat diklasifikasikan menjadi tiga kategori utama, yaitu depresi ringan, sedang, dan berat, berdasarkan jumlah, intensitas, dan durasi gejala yang dialami seseorang. Depresi ringan ditandai dengan beberapa gejala yang muncul namun masih memungkinkan penderita menjalani aktivitas sehari-hari meski dengan sedikit kesulitan. Depresi sedang menunjukkan gejala yang lebih banyak dan intens, mulai mengganggu fungsi sosial atau pekerjaan secara nyata. Sementara itu, depresi berat mencakup gejala yang sangat mengganggu, sering kali disertai dengan hilangnya kemampuan untuk menjalankan aktivitas harian secara normal, serta mungkin muncul pikiran untuk menyakiti diri sendiri atau bunuh diri. Tingkat keparahan ini penting untuk menentukan pendekatan penanganan dan intervensi yang tepat.

3. Variabel aturan (*Rule Variabel*)

Variabel *rule* untuk menentukan tingkat depresi didasarkan pada jumlah gejala utama dan gejala tambahan yang dialami seseorang. Untuk depresi ringan, seseorang harus mengalami 2 dari 3 gejala utama disertai 1–2 gejala tambahan, dan tidak boleh ada gejala berat. Depresi sedang ditandai dengan adanya 2 dari 3 gejala utama ditambah 3–4 gejala tambahan. Sedangkan pada depresi berat, seseorang harus menunjukkan ketiga gejala utama sekaligus, serta memiliki lebih dari 4 gejala tambahan. Aturan ini berfungsi untuk memandu sistem pakar dalam mengklasifikasikan tingkat depresi berdasarkan kombinasi dan jumlah gejala yang dilaporkan pengguna.

4. Variabel Skor atau Bobot

Dalam metode *Naive Bayes*, variabel bobot (bobot probabilitas) merepresentasikan seberapa besar kontribusi masing-masing gejala terhadap kemungkinan terjadinya suatu tingkat depresi tertentu (ringan, sedang, atau berat). Setiap gejala diberi nilai probabilitas berdasarkan seberapa sering gejala tersebut muncul pada masing-masing kategori depresi, dengan asumsi bahwa semua gejala bersifat independen satu sama lain. Ketika sistem menerima masukan gejala dari pengguna, metode ini menghitung peluang untuk setiap kategori depresi dengan mengalikan bobot masing-masing gejala yang relevan, lalu memilih kategori dengan probabilitas tertinggi sebagai hasil diagnosis. Penggunaan bobot ini membuat *Naive Bayes* efektif dalam menangani ketidakpastian dan mempercepat proses klasifikasi dalam sistem pakar.

2.3 *Software* Pendukung

Software, atau perangkat lunak dalam bahasa Indonesia, merupakan kumpulan instruksi, program, dan data yang digunakan untuk mengontrol dan mengoperasikan komputer atau perangkat elektronik lainnya. *Software* merupakan salah satu komponen penting dari sistem komputer, yang bekerja sama dengan perangkat keras (*hardware*) untuk menghasilkan fungsionalitas yang diinginkan. Berikut merupakan *software* pendukung dalam perancangan sistem pakar berbasis Android:

2.3.1 Aplikasi

Aplikasi merupakan perangkat lunak yang dirancang untuk membantu pengguna dalam menjalankan tugas-tugas tertentu pada perangkat elektronik seperti komputer, tablet, atau *smartphone*. Aplikasi dapat berupa program untuk pengolahan kata, pengeditan gambar, hingga permainan. Dalam konteks perangkat *mobile*, aplikasi *mobile* merupakan perangkat lunak yang dikembangkan khusus untuk berjalan pada perangkat seperti *smartphone* dan tablet, memungkinkan pengguna untuk mengakses berbagai layanan dan informasi secara praktis. Aplikasi *mobile* biasanya tersedia melalui platform distribusi seperti *Google Play Store* untuk *Android* dan *App Store* untuk *iOS*.

Penelitian yang membahas peran aplikasi *mobile* dalam dunia kesehatan merupakan studi oleh Nurkholis dan Saputra (2021) yang menyatakan bahwa "*e-health* berbasis *mobile* dapat meningkatkan layanan klinik dan memudahkan masyarakat untuk mendapatkan layanan kesehatan." Penelitian ini menyoroti bagaimana aplikasi pada *smartphone* dapat digunakan sebagai media untuk meningkatkan efisiensi dan aksesibilitas layanan kesehatan di klinik (Nurkholis and Saputra 2021).

2.3.2 Android

Android merupakan sistem operasi berbasis Linux yang dirancang untuk perangkat bergerak layar sentuh seperti *smartphone* dan tablet. Awalnya dikembangkan oleh Android Inc., kemudian diakuisisi oleh Google pada tahun 2005. Android menyediakan platform terbuka bagi para pengembang untuk menciptakan aplikasi mereka sendiri, yang kemudian dapat didistribusikan melalui

berbagai platform distribusi seperti *Google Play Store*. Dengan sifatnya yang *open-source* dan fleksibel, Android telah menjadi sistem operasi *mobile* paling populer di dunia, digunakan oleh berbagai produsen perangkat keras di seluruh dunia.



Gambar 2. 2 Andorid

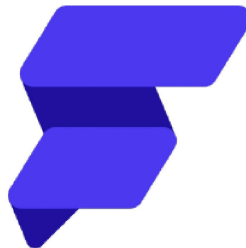
Sumber : (<https://blogs.powercode.id/>)

Di dunia kesehatan, Android telah menjadi platform utama untuk aplikasi yang membantu pasien dalam mengelola kesehatan mereka. Aplikasi berbasis Android digunakan untuk berbagai tujuan, mulai dari pemantauan kondisi medis, pengingat obat, hingga komunikasi dengan tenaga medis. Keunggulan Android dalam hal keterjangkauan, kemudahan akses, dan ketersediaan aplikasi medis membuatnya menjadi alat yang efektif dalam memberikan solusi kesehatan digital yang lebih mudah diakses oleh masyarakat luas.

2.3.3 FlutterFlow

Android Studio merupakan *Integrated Development Environment* (IDE) resmi untuk pengembangan aplikasi Android. Berbasis editor kode dan alat developer yang andal dari IntelliJ IDEA . Android merupakan salah satu sistem operasi yang sedang terkenal saat ini. Kelebihan android dibandingkan sistem operasi *smartphone* lainnya merupakan android berbasis *open source code* sehingga memudahkan para pengembang untuk menciptakan dan memodifikasi aplikasi atau

fitur-fitur yang belum ada di sistem operasi android sesuai dengan keinginan mereka sendiri (Putra et al. 2024).



Gambar 2. 3 *FlutterFlow*

Sumber : (<https://github.com/FlutterFlow>)

Dalam proses perancangan, *FlutterFlow* memudahkan pembuatan halaman seperti login, registrasi, dan dashboard. Komponen seperti tombol, formulir input, dan navigasi dapat disusun secara visual. *FlutterFlow* juga menawarkan fitur animasi, responsivitas, dan penyesuaian gaya. Selain itu, platform ini dapat terhubung langsung dengan *backend* seperti *Supabase* untuk *autentikasi* dan pengelolaan data real-time. Setelah selesai, aplikasi dapat diekspor sebagai kode *Flutter* asli untuk pengembangan lebih lanjut di editor seperti VS Code atau Android Studio.

2.3.4 *Supabase*

Supabase merupakan platform *backend open-source* yang menyediakan berbagai layanan penting untuk mendukung pengembangan aplikasi, seperti *autentikasi* pengguna, penyimpanan file, dan layanan basis data (Valerian Romero and Fahrudin 2023). Dibangun di atas *PostgreSQL*, *Supabase* menawarkan API otomatis dan *real-time*, sehingga ideal untuk aplikasi modern yang memerlukan

respons cepat dan efisien. Sering dianggap sebagai alternatif *open-source* untuk *Firebase*, *Supabase* memiliki layanan yang serupa.

Dalam perancangan aplikasi, *Supabase* berfungsi sebagai penghubung antara *frontend* dan *database*. Contohnya, ketika pengguna mendaftar atau *login* melalui aplikasi yang dibuat dengan *FlutterFlow*, permintaan tersebut dikirim ke *Supabase* untuk diverifikasi. *Supabase* juga menyediakan sistem penyimpanan file yang membantu mengelola dokumen, gambar, atau file lain yang diunggah oleh pengguna.



Gambar 2. 4 *supabase*

Sumber : (<https://nasriadzlani.medium.com>)

Supabase mendukung *query* data *real-time* dan autentikasi berbasis email/password, serta *OAuth* (Google, GitHub), dengan sistem manajemen pengguna yang lengkap. Kelebihan lainnya termasuk dokumentasi yang menyeluruh, kemudahan integrasi dengan alat pengembangan modern, dan fleksibilitas operasional baik secara lokal maupun *cloud*. Dengan *Supabase*, developer dapat fokus pada logika bisnis tanpa perlu membangun *backend* dari awal.

Supabase juga dilengkapi dengan fitur analitik yang memungkinkan developer untuk memantau penggunaan dan performa aplikasi Android secara efisien. Dengan kemampuan menghasilkan laporan dan metrik yang relevan, pengguna dapat membuat keputusan berdasarkan data untuk meningkatkan pengalaman pengguna.

Selain itu, *Supabase* memiliki komunitas yang aktif memberikan dukungan, memudahkan developer baru dalam proses pembelajaran. Dengan semua fitur ini, *Supabase* menjadi pilihan yang menarik bagi pengembang yang ingin menciptakan aplikasi Android yang scalable dan efisien tanpa harus berurusan dengan kerumitan pengembangan backend.

2.3.5 *PostgreSQL*

PostgreSQL merupakan sistem manajemen basis data relasional (RDBMS) yang kuat, *open-source*, dan banyak digunakan dalam berbagai skala pengembangan aplikasi. Sistem ini mendukung operasi SQL standar dan menawarkan fitur canggih seperti *indexing*, *foreign key*, transaksi ACID, dan integritas data. Dalam konteks *Supabase*, *PostgreSQL* berperan sebagai inti dari penyimpanan dan pengelolaan data.



Gambar 2. 5 *PostgreSQL*

Sumber : (<https://icon-icons.com>)

Untuk pembuatan database skala besar, *postgresql* menjadi solusi yang lebih baik karena beberapa query lebih unggul dibandingkan *MySQL* (Saouda and Barokah 2022) . Dalam pengembangan aplikasi, *PostgreSQL* bertanggung jawab

untuk menyimpan data penting, seperti akun pengguna, hasil *input*, log aktivitas, dan informasi lainnya. Data tersebut disusun dalam tabel relasional yang saling terhubung dan dapat diakses melalui perintah SQL. *PostgreSQL* sangat ideal untuk aplikasi yang memerlukan struktur data kompleks dan performa tinggi.

Salah satu keunggulan utama *PostgreSQL* yaitu skalabilitasnya yang tinggi, serta kompatibilitasnya dengan berbagai platform pengembangan. Dengan menggunakan *PostgreSQL* sebagai database utama dalam *Supabase*, developer tidak perlu mengatur server database secara manual, tetapi tetap dapat memanfaatkan kekuatan penuh *PostgreSQL* melalui layanan yang disediakan oleh *Supabase*.

Dengan demikian, *PostgreSQL* menjadi fondasi yang andal dalam manajemen data aplikasi, memungkinkan pengembang untuk fokus pada pengembangan fitur dan logika bisnis tanpa khawatir tentang penyimpanan dan pengelolaan data. Keberadaan *PostgreSQL* dalam ekosistem *Supabase* menjadikannya solusi efisien bagi pengembang yang mencari performa optimal dan kemudahan dalam pengelolaan data. Kemampuan untuk menangani transaksi besar dan operasi kompleks menjadikannya pilihan yang tepat untuk aplikasi skala besar, *PostgreSQL* terus berkembang dan beradaptasi dengan kebutuhan pengembang modern.

2.3.6 App Diagrams.net

Pada September 2021, aplikasi "app.diagrams.net" mengacu pada *website* *Diagrams.net*, sebelumnya dikenal sebagai *draw.io*. *Diagrams.net* merupakan alat diagram *online* yang populer dan gratis yang memungkinkan pengguna untuk

membuat berbagai jenis diagram, seperti *flowchart*, diagram alur proses, diagram ER (*Entity-Relationship*), diagram UML (*Unified Modeling Language*), dan banyak lagi. Fungsi utama app.diagrams.net merupakan sebagai berikut:



Gambar 2. 6 diagrams.net

Sumber : (<https://miro.medium.com/>)

1. Membuat Diagram: Pengguna dapat membuat berbagai jenis diagram melalui antarmuka yang intuitif. Alat ini menyediakan beragam bentuk dan simbol untuk membantu menghasilkan diagram yang akurat dan informatif.
2. Kolaborasi: Diagrams.net memungkinkan kolaborasi secara real-time, sehingga beberapa pengguna dapat bekerja bersama pada diagram yang sama sekaligus. Fitur ini sangat bermanfaat ketika tim perlu berkolaborasi dalam menciptakan diagram yang kompleks.
3. **Penyimpanan dan Ekspor:** Diagrams.net mendukung penyimpanan proyek baik secara lokal maupun di *cloud storage* seperti *Google Drive*, *OneDrive*, *Dropbox*, dan lainnya. Pengguna juga bisa mengekspor diagram dalam berbagai format file, termasuk PNG, JPG, PDF, SVG, XML, dan banyak lagi.
4. Integrasi: Diagrams.net dapat terintegrasi dengan berbagai aplikasi dan platform, seperti *Google Workspace*, *Microsoft Teams*, *Jira*, *Confluence*, dan *GitHub*. Integrasi ini memungkinkan pengguna untuk bekerja lebih

efisien dan membagikan diagram dengan mudah di lingkungan kerja yang sudah ada.

5. Bebas dan Open Source: Diagrams.net adalah aplikasi sumber terbuka, yang berarti kode sumbernya dapat diakses oleh siapa saja. Ini memungkinkan pengguna untuk mengunduh, memodifikasi, dan menginstal aplikasi secara gratis sesuai dengan lisensi perangkat lunak terbuka yang berlaku.

Catatan bahwa perangkat lunak dan aplikasi sering mengalami perkembangan dan perubahan seiring waktu, sehingga mungkin ada beberapa perubahan pada fungsi dan fitur app.diagrams.net sejak pengetahuan saya yang terakhir pada September 2021. Jika Anda tertarik menggunakan app.diagrams.net, disarankan untuk mengunjungi situs resminya atau melihat sumber daya terbaru untuk informasi yang paling terkini.

2.4 UML (*Unified Modeling Language*)

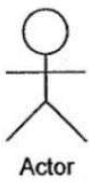
UML yaitu notasi standar *de facto* untuk perangkat lunak yang mewakili grafis. Diagram UML digunakan dalam analisis, konstruksi, dan pemeliharaan sistem perangkat lunak. Sebagian besar, diagram UML menangkap sebuah pandangan abstrak dari (bagian dari) sistem perangkat lunak. Tujuan utama diagram UML merupakan untuk berbagi pengetahuan tentang sistem antara pengembang. Kualitas tata letak diagram UML memainkan peran penting dalam pemahaman (Bergström et al. 2022). UML awalnya dibuat oleh grup manajemen objek, dengan versi pertama 1.0 dirilis pada Januari 1997. UML juga dapat didefinisikan sebagai bahasa standar untuk memvisualisasikan, mendesain, dan mendokumentasikan

sistem atau dikenal juga sebagai bahasa standar untuk menulis sketsa perangkat lunak.

2.4.1 Use Case Diagram

Use case diagram merupakan deskripsi fungsi dari sebuah sistem dari perspektif pengguna. *Use case* diagram bekerja dengan cara mendeskripsikan tipikal interaksi antara *user* sebuah sistem dengan sistemnya sendiri melalui sebuah cerita bagaimana sebuah sistem dipakai. Sebuah *use case* dapat menggambarkan jenis interaksi antara pengguna sistem dan sistem. Kasus penggunaannya mudah dipelajari. Pada tahap awal pemodelan, diperlukan diagram yang dapat menggambarkan tindakan operator dengan fungsi dari sistem itu sendiri, seperti yang terdapat pada *use case* (Ramdany et al. 2024.). Berikut simbol *use case* yang umum digunakan:

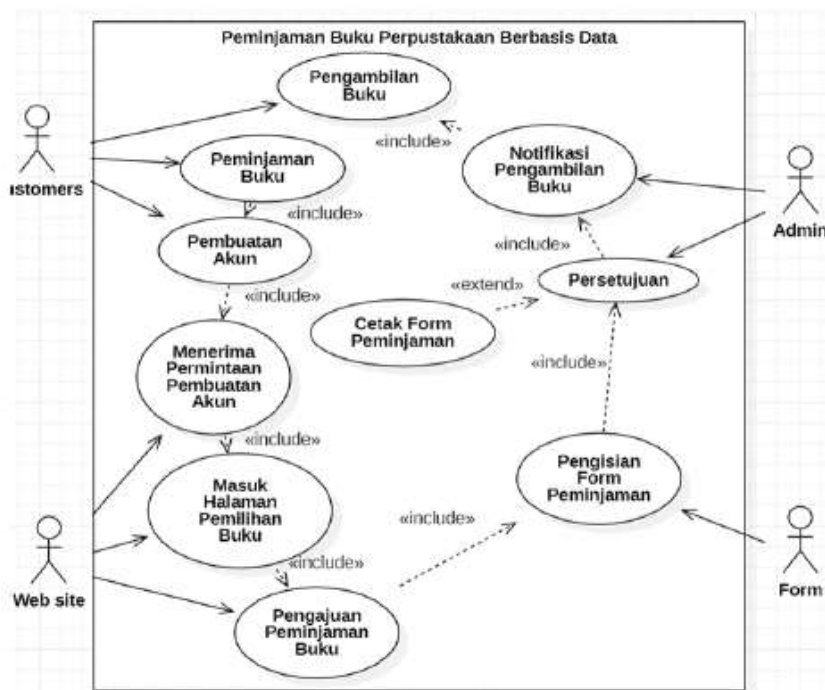
Tabel 2. 1 Simbol *Use Case*

Simbol	Keterangan
Aktor/<i>Actor</i> 	Orang proses, atau system lain yang berinteraksi dengan sistem informasi yang akan dibuat itu sendiri, jadi walaupun simbol dari <i>actor</i> merupakan gambar orang, tapi actor belum tentu orang, biasanya dinyatakan menggunakan kata benda diawali <i>frase name actor</i>.
	Fungsi yang disediakan sistem sebagai unit yang saling bertukar antar unit atau actor, biasanya

	dinyatakan dengan menggunakan kata kerja diawali <i>frase name use case</i>
Asosiasi berarah/Directed Association →	Hubungan generalisasi dan spesialisasi (umum-khusus) antara dua buah use case dimana fungsi yang satu merupakan fungsi yang lebih umum dari yang lainnya.
Association —	Komunikasi antar <i>actor</i> dan <i>use case</i> yang berpartisipasi pada <i>use case</i> atau <i>use case</i> memiliki dengan <i>acto</i> .

Sumber : (Penelitian, 2025)

Berikut contoh penerapan *Use Case* diagram:



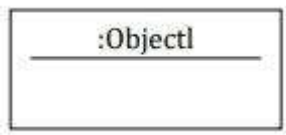
Gambar 2. 7 Contoh penggunaan dalam *Use Case*

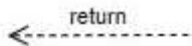
Sumber : (Ramdany, 2024.)

2.4.2 Sequence diagram

Sequence diagram merupakan diagram yang menjelaskan bagaimana objek berinteraksi selama periode waktu tertentu. Urutan dapat menggambarkan urutan atau langkah-langkah yang harus dilakukan untuk menghasilkan sesuatu, seperti yang ditunjukkan dalam diagram *use case* (Ramdany et al. 2024). Berikut simbol *Sequence Diagram* yang umum digunakan:

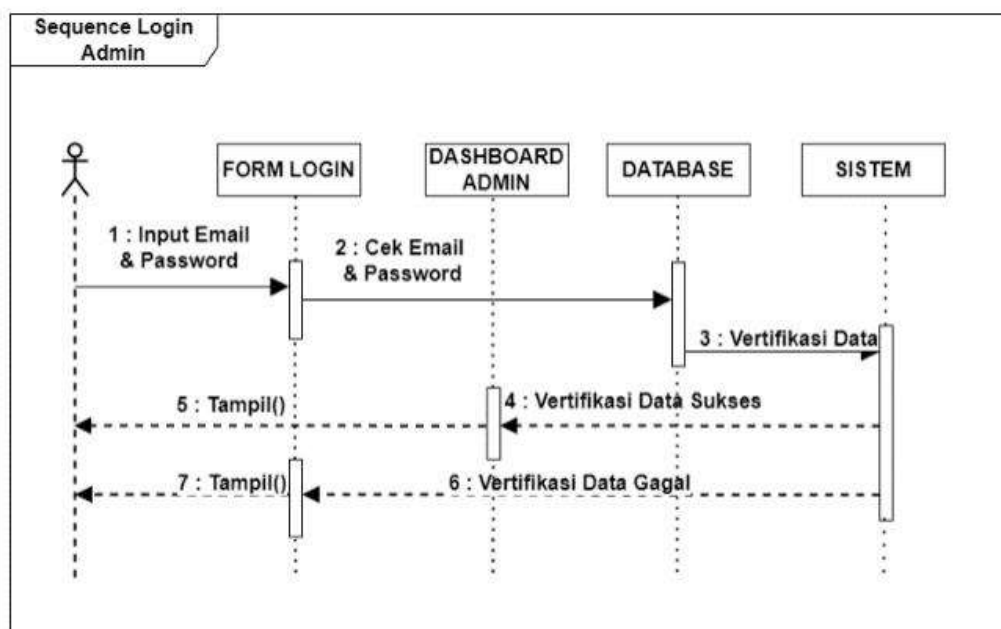
Tabel 2. 2 Simbol *Sequence Diagram*

Simbol	Keterangan
Aktor 	<i>Actor</i> juga dapat berkomunikasi dengan <i>Object</i> , maka <i>Actor</i> juga dapat di urutan sebagai kolom.
Garis Hidup/life line 	<i>Lifeline</i> mengindikasikan keberadaan sebuah <i>object</i> dalam basis waktu. Notasi untuk <i>Lifeline</i> merupakan garis putus-putus Vertikal yang ditarik dari sebuah <i>Object</i> .
Object 	<i>Object</i> merupakan instance dari sebuah class dan dituliskan tersusun secara horizontal. Digambarkan sebagai sebuah class (kotak) dengan nama object didalamnya yang diawali dengan sebuah titik normal.

Activation 	<i>Activation</i> dinotasikan sebagai sebuah kotak segi empat yang di gambar pada sebuah Lifeline mengindikasikan sebuah objek yang akan melakukan sebuah aksi.
	Untuk menotasikan relasi antara dua komponen.
<i>Association</i> 	Mengindikasikan komunikasi antar objek.

Sumber : (Penelitian, 2025)

Berikut contoh penerapan *Sequence Diagram*:



Gambar 2. 8 Contoh penerapan dalam Sequence Diagram

Sumber : (Penelitian, 2025)

2.4.3 Activity diagram

Activity Diagram (diagram aktivitas) merupakan diagram yang menggambarkan aliran fungsionalitas dari sistem. Pada tahap pemodelan bisnis, diagram aktifitas dapat digunakan untuk menunjukkan aliran kerja bisnis (Kesumaningtyas and Handayani 2020). Dapat juga digunakan untuk menggambarkan aliran kejadian (*flow of events*). Diagram aktivitas atau diagram aktivitas dalam bahasa Indonesia merupakan diagram yang dapat digunakan untuk memodelkan berbagai proses dalam suatu sistem. Sebagai sekumpulan proses yang mengontrol suatu sistem dan digambarkan secara vertikal. Diagram aktivitas merupakan contoh diagram UML dalam pengembangan *use case*. (Maiyendra, 2021). Berikut simbol *activity diagram* yang umum digunakan:

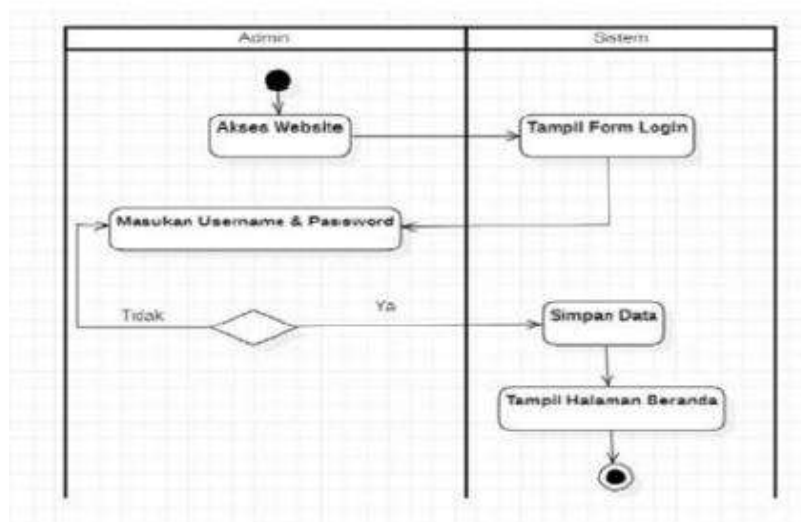
Tabel 2. 3 Simbol *Activity* Diagram

Simbol	Keterangan
Status awal 	Status awal aktivitas sistem, sebuah activity diagram memiliki sebuah status awal.
Status Akhir 	Status Akhir yang dilakukan sistem, sebuah diagram aktivitas memiliki sebuah status akhir.
Aktivitas 	Aktivitas yang dilakukan sistem, aktivitas biasanya diawali dengan kata kerja.
<i>Swimlane</i> 	Memisahkan organisasi bisnis yang bertanggung jawab terhadap aktivitas yang terjadi.

<i>Association</i> 	Menambah transisi dari suatu aktivitas ke aktivitas yang lain.
---	--

Sumber : (Penelitian, 2025)

Berikut contoh penerapan *Activity Diagram*:



Gambar 2. 9 Contoh penggunaan dalam *Activity Diagram*

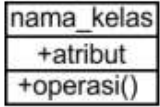
Sumber : (Ramdany, 2024)

2.4.4 Class diagram

Diagram kelas atau *class diagram* merupakan diagram yang digunakan untuk merepresentasikan kelas sebagai paket sehingga memenuhi beberapa persyaratan paket yang akan digunakan nantinya. Namun pada *class diagram*, desain model dibagi menjadi dua bagian. Skema kelas pertama dikembangkan dari model domain, yang merupakan abstraksi dari database. Diagram kelas kedua merupakan bagian dari modul program pola MVC (*Model View Controller*), yang memiliki batas kelas sebagai antarmuka kelas, kelas kontrol sebagai lokasi pencarian

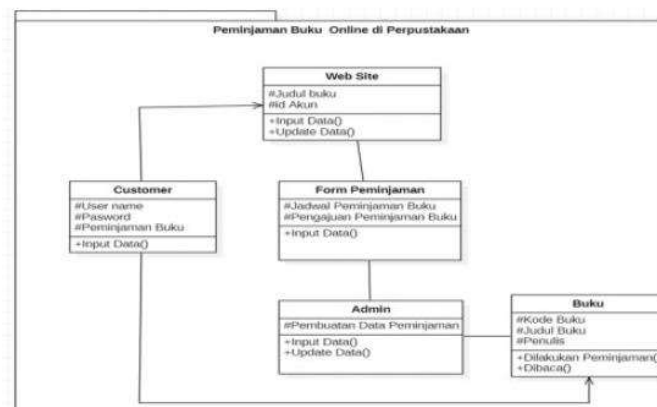
algoritma, dan unit kelas sebagai tabel dalam *database* dan kueri program (Maiyedra, 2021). Berikut simbol *Calss Diagram* yang umum digunakan:

Tabel 2. 4 Simbol *Calss Diagram*

Simbol	Keterangan
Kelas 	Kelas pada struktur sistem, tiap kelas memiliki nama, atribut, dan operation atau method.
Asosiasi/Association 	Relasi antar kelas dengan makna umum, asosiasi biasanya juga disertai dengan multiplicity
Asosiasi berarah/DirectedAssociation 	Relasi antar kelas dengan makna kelas yang satu digunakan oleh kelas yang lain, asosiasi biasanya juga disertai dengan <i>multiplicity</i> .
Generalisasi 	Relasi antara kelas dengan makna generalisasi-spesialisi (umum khusus)

Sumber : (Penelitian, 2025)

Berikut contoh penerapan *Class Diagram*:



Gambar 2. 10 Contoh penerapan dalam Class Diagram

Sumber : (Rabbani, Jamaluddin, and Solehudin 2023)

2.5 Penelitian Terdahulu

Penelitian terdahulu merupakan bagian penting dalam suatu karya ilmiah yang berfungsi untuk menggambarkan sejauh mana topik yang diteliti telah dikaji oleh peneliti lain. Dalam penelitian terdahulu merangkum nama peneliti, judul penelitian, masalah penelitian, metode penelitian serta hasil dari penelitian. Dalam penelitian ini, banyak mengangkat tentang penerapan metode *Naïve Bayes* dan *forward chaining* dalam pengembangan sistem pakar untuk diagnosis penyakit

1. Jurnal Teknologi dan Sistem Komputer (JTSiskom), Penelitian ini dilakukan oleh Riky Yoga Perkasa dan Nurul Fuad dengan judul "*Expert System For Diagnosis Of Diseases In Cat Using The Naive Bayes Method*" yang terakreditasi Sinta 2 dan diterbitkan pada volume 8 nomor 2 Oktober 2023. Masalah utama yang diangkat merupakan kesulitan dalam mendiagnosis penyakit pada kucing secara akurat dan cepat. Metode yang digunakan merupakan pengembangan sistem pakar berbasis web dengan pendekatan *Naïve Bayes*, PHP, dan *MySQL* untuk menganalisis data gejala dan penyakit kucing. Hasil penelitian menunjukkan bahwa sistem yang dikembangkan mampu mendiagnosis penyakit kucing dengan tingkat akurasi mencapai 91%, serta memberikan deskripsi dan solusi sebagai langkah awal pengobatan (Yoga Perkasa and Fuad, 2023).
2. Dengan judul "Implementasi Sistem Pakar Diagnosa Bau Mulut Penyakit gigi Menggunakan Metode *Naive Bayes* Berbasis Web" dan ISSN 2985-4172 yang dilakukan oleh Akbar Zaelani dan Wiwin Winarti pada tahun 2023. Masalah utama yang diangkat merupakan rendahnya tingkat edukasi dan pemeliharaan

kesehatan gigi masyarakat, sehingga banyak penyakit gigi dan mulut tidak terdeteksi secara dini. Metode yang digunakan merupakan pengembangan sistem pakar berbasis web dengan algoritma *Naive Bayes*, yang mengumpulkan data gejala dari pengguna dan menganalisisnya untuk memberikan diagnosis cepat dan akurat. Hasil penelitian menunjukkan bahwa sistem mampu membantu mendeteksi penyakit gigi dan bau mulut secara dini, memudahkan pengguna dalam memahami diagnosis, dan meningkatkan efisiensi proses diagnosis melalui platform web (Zaelani and Winarti, 2023).

3. Penelitian ini dilakukan oleh Farhan Dwiramadhan dan rekan, dengan ISSN 2587-3586, berjudul "Sistem Pakar Diagnosa Penyakit Kulit Kucing Menggunakan Metode *Naive Bayes* Berbasis Web". Masalah utama yang diangkat merupakan kebutuhan sistem diagnosis penyakit kulit pada kucing secara akurat dan efisien. Metode yang digunakan merupakan kombinasi *Naive Bayes* dan *Certainty Factor*, dengan pengembangan berbasis *website* menggunakan PHP dan *Sublime Text*. Hasil penelitian menunjukkan bahwa sistem pakar yang dibangun mencapai tingkat keakuratan diagnosis hingga 92%, dan perbandingan antara kedua metode menunjukkan potensi penerapan dalam diagnosis dini penyakit kulit pada kucing (Dwiramadhan, Wahyuddin, and Hidayatullah 2022).
4. Penelitian ini dilakukan oleh Awwaluddin Abdul Qohhar dan Moh Noor Al Azam dengan ISSN P-ISSN: 2622-6901 dan E-ISSN: 2622-6375, berjudul "Website Sistem Pakar Diagnosa Penyakit Kulit dengan *Forward Chaining* (Studi Kasus: Puskesmas Tanggulangin)". Masalah utama yang diangkat

merupakan kebutuhan untuk meningkatkan efisiensi dan akurasi diagnosis penyakit kulit melalui sistem berbasis *website*. Metode yang digunakan merupakan pendekatan kuantitatif dan desain penelitian eksploratif, dengan tahapan studi literatur, pengumpulan data melalui wawancara dan studi kasus, serta pengembangan sistem berbasis *framework Laravel* dan metode *forward chaining*. Hasil penelitian menunjukkan bahwa sistem yang dikembangkan mampu membantu proses diagnosis penyakit kulit secara lebih efisien dan akurat, serta dapat dikembangkan lebih lanjut dengan penambahan data penyakit dan peningkatan antarmuka pengguna (Abdul Qohhar et al. 2024).

5. Penelitian ini dilakukan oleh Aldo Rio Prayoga bersama tim dari Jurusan Sistem Informasi/Informatika, Fakultas Teknologi Komunikasi dan Informatika, Universitas Nasional, dengan ISSN 2548-9771/EISSN 2549-7200. Judul penelitian merupakan "Sistem Pakar Diagnosa Penyakit dan Hama Tanaman Pepaya Menggunakan Metode *Forward Chaining* dan *Naïve Bayes*". Masalah utama yang diangkat merupakan kurangnya wawasan petani dalam penanganan hama dan penyakit pepaya, sehingga diperlukan sistem pakar untuk memudahkan diagnosis. Metode yang digunakan merupakan gabungan dari *Forward Chaining* dan *Naïve Bayes*, yang bekerja secara berurutan untuk mengidentifikasi penyakit dan hama secara probabilistik. Hasil penelitian menunjukkan bahwa sistem ini mampu membantu petani dalam mendeteksi penyakit dan hama pada tanaman pepaya secara akurat dan efisien (Rio Prayoga et al. 2021).

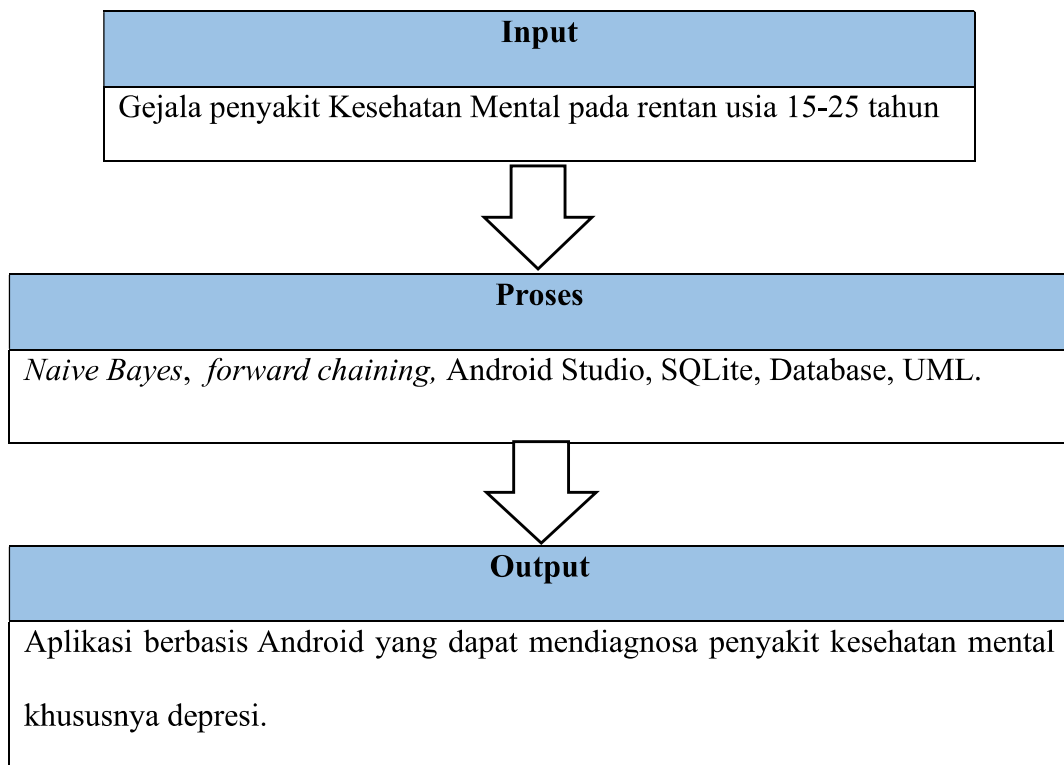
6. Gangguan kesehatan mental (*Mental Health Disorder*) yaitu salah satu *syndrome* ataupun pola perilaku psikologi seseorang yang akan menyebabkan ketidakmampuan saat melakukan pekerjaan. Karena beragam kesulitan dan dilema yang muncul, banyaknya masalah yang tidak dapat dihadapi seseorang dalam kehidupan dapat mengganggu jaringan saraf otak manusia dan menyebabkan penyakit mental. Tingkat psikologis seseorang tergantung pada pengalaman emosional, perilaku, lingkungan maupun latar belakang pendidikan keluarga. Akibat ketidakseimbangan antara jumlah penderita gangguan mental dengan jumlah dokter spesialis gangguan mental, banyak penderita gangguan mental harus menunggu lama untuk mendapatkan dokter spesialis. Selain itu, berkonsultasi dengan dokter spesialis penderita penyakit gangguan kesehatan mental juga perlu membayar biaya yang cukup mahal. Solusi yang tepat untuk mendiagnosa gejala awal gangguan mental merupakan menggunakan sistem pakar dan algoritma genetika. Suatu populasi individu dikelola kemudian mendeskripsikan kandidat solusi suatu optimasi kesehatan mental pada seseorang merupakan konsep dasar algoritma genetika ini. Sistem ini dibuat bertujuan mengetahui nilai optimal dari berbagai macam keluhan gangguan kesehatan mental dan menganalisa karakteristik metode algoritma genetika. Pemilihan metode algoritma genetika dalam hal ini berdasarkan penelitian terdahulu yang menunjukkan presentase keakuratan metode ini cukup tinggi.(Dwi Kurnia, Andryana, and Gunaryati 2021)
7. Sistem pakar ini merupakan suatu sistem yang dapat membantu dalam mendiagnosa penyakit kejiwaan pada pasien jiwa khususnya penyakit depresi.

Depresi ialah salah satu masalah kesehatan jiwa. Masalah ini sangat penting karena seseorang dengan penyakit depresi akan mengalami penurunan produktivitas. Tujuan penelitian ini merupakan menerapkan Metode *Naive Bayes* dalam mendiagnosa serta mengklasifikasian penyakit kejiwaan dengan batasan usia mulai dari 15-25 tahun. Membangun sebuah sistem pakar (*expert system*) untuk mendiagnosa gangguan kejiwaan pada pasien jiwa dengan gejala yang dialami dengan Metode *Naive Bayes*. Penerapan metode *Naive Bayes* dalam mendiagnosa serta mengklasifikasian penyakit kejiwaan diperuntukan bagi pasien dengan usia mulai dari 15-25 tahun keatas dimana prosesnya dilakukan dalam 3 tahapan yaitu menghitung nilai prior, menghitung nilai likelihood dan nilai posterior. Hasil dari penelitian mengetahui klasifikasi dan jenis penyakit kejiwaan khususnya jenis penyakit depresi. Mengolah data menjadi informasi dengan menggunakan metode *Naive Bayes* dan basis pengetahuan (informasi dari pakar, data rekam medis poli jiwa, buku, dan jurnal) sehingga sistem yang akan dibuat tepat sasaran. Setelah itu melakukan tahapan perancangan sistem. *Naive Bayes* merupakan Metode untuk mengklasifikasikan probabilitas sederhana yang didasarkan pada *Teorema Bayes*. (Jain Barus and Simangunsong 2021).

2.6 Kerangka Pemikiran

Kerangka pemikiran merupakan rencana atau struktur yang digunakan untuk mengorganisasi ide dan informasi dalam sebuah proyek atau penelitian. Hal ini bertujuan untuk memudahkan pemikiran, analisis, dan pengembangan sebuah konsep atau ide. Kerangka pemikiran dapat membantu peneliti atau pengembang

untuk mengidentifikasi masalah, mengembangkan solusi yang sesuai, dan memastikan bahwa setiap elemen penting telah dipertimbangkan. Dalam pengembangan sistem pakar, kerangka pemikiran dapat membantu mengidentifikasi masalah atau tantangan yang ingin diselesaikan. Kerangka pemikiran yang dibangun dalam sistem ini dimulai dari *input* berupa data gejala penyakit kesehatan mental yang dialami oleh individu dalam rentang usia 15 hingga 25 tahun. Gejala-gejala tersebut dikumpulkan dan dijadikan variabel utama yang akan dianalisis dalam sistem. Selanjutnya, pada tahap proses, data gejala tersebut diolah menggunakan kombinasi metode dan perangkat bantu. Metode *Naive Bayes* digunakan untuk menghitung probabilitas kemunculan penyakit berdasarkan gejala yang ada, sedangkan metode *forward chaining* diterapkan untuk melakukan inferensi atau penelusuran logika dari fakta ke kesimpulan diagnosis. Untuk membangun aplikasi ini digunakan Android Studio sebagai platform pengembangan aplikasi *mobile*, sedangkan *SQLite* dan *database* berfungsi untuk menyimpan data pengguna, data gejala, dan hasil diagnosis. Perancangan struktur sistem dilakukan menggunakan UML (*Unified Modeling Language*) agar pengembangan aplikasi menjadi lebih terarah dan sistematis. Setelah seluruh proses selesai, *output* yang dihasilkan merupakan sebuah aplikasi berbasis Android yang dapat mendiagnosa penyakit kesehatan mental, khususnya mendeteksi depresi berdasarkan gejala yang diinputkan oleh pengguna. Aplikasi ini diharapkan dapat membantu pengguna dalam mengenali kondisi mental mereka lebih dini dan mengambil langkah pencegahan yang sesuai.



Gambar 2. 11 Kerangka pemikiran

Sumber : (Penelitian, 2025)