

BAB II

TINJAUAN PUSTAKA

2.1 Penelitian Terkait

Hasil penelitian terdahulu disajikan dalam bentuk tabel untuk memudahkan perbandingan antara penelitian yang telah dilakukan sebelumnya dengan penelitian yang akan dilakukan oleh penulis.

Tabel 2.1 Penelitian Terdahulu

No	Peneliti	Metode / Teknologi	Hasil Penelitian	Perbedaan dengan Penelitian Ini
1	Rahadin dkk. (2026)[9]	<i>Natural Language Processing</i> (NLP), Python, <i>Node.js</i> , <i>WhatsApp Business API</i> , Telegram	<i>Chatbot</i> berhasil memberikan respons otomatis dengan akurasi <i>intent</i> di atas 85%, waktu respons kurang dari 3	Penelitian tersebut menggunakan NLP konvensional dan <i>API WhatsApp</i> standar, sedangkan penelitian ini menggunakan Local <i>Large Language Model</i> (Gemma) untuk pemahaman bahasa yang lebih kompleks dan <i>Evolution API</i> untuk

No	Peneliti	Metode / Teknologi	Hasil Penelitian	Perbedaan dengan Penelitian Ini
		Bot <i>API</i> , dan MySQL	detik, dan mendapatkan skor <i>usability</i> 84,6 (Kategori <i>Excellent</i>)	integrasi <i>WhatsApp</i> yang lebih fleksibel tanpa ketergantungan pada <i>cloud</i> pihak ketiga demi keamanan data desa.
2	Hidayat dkk. (2025)[10]	<i>Research and Development</i> (R&D). <i>Retrieval-Augmented Generation</i> (RAG). Zephyr-7B-beta	Sistem berhasil memberikan jawaban dengan akurasi tinggi dan skor kesamaan (<i>similarity</i>)	Penelitian yang dilakukan oleh Hidayat dkk. (2025) menitikberatkan pada optimalisasi layanan informasi akademik di lingkup perguruan tinggi dengan mengintegrasikan <i>platform</i> Telegram. Perbedaan mendasar dengan penelitian ini

No	Peneliti	Metode / Teknologi	Hasil Penelitian	Perbedaan dengan Penelitian Ini
		(Hugging Face). BAAI/bge-Base-en-v1.5. Python, LlamaIndex (SimpleDirectoryReader, VectorStoreIndex), dan <i>python-telegram-bot</i>.	<i>score</i>) mencapai 100% pada pengujian sampel. Waktu respons rata-rata sistem adalah sekitar 60 detik untuk memproses pertanyaan dari sederhana hingga kompleks.	terletak pada lokus dan arsitektur teknologi yang digunakan; di mana penelitian ini berfokus pada pelayanan publik di tingkat pemerintahan desa dengan mengimplementasikan <i>Local Large Language Model</i> (LLM) melalui model Gemma guna menjamin kedaulatan data. Selain itu, penelitian ini menggunakan <i>Evolution API</i> untuk integrasi <i>WhatsApp</i> dan dibangun di atas <i>NestJS</i> untuk mencapai skalabilitas sistem yang lebih modern

No	Peneliti	Metode / Teknologi	Hasil Penelitian	Perbedaan dengan Penelitian Ini
			Metode RAG terbukti efektif mengatasi keterbatasan LLM dengan memberikan konteks dokumen PDF panduan akademik yang valid.	dibandingkan penelitian terdahulu.

No	Peneliti	Metode / Teknologi	Hasil Penelitian	Perbedaan dengan Penelitian Ini
3	Hasbi, Imanda, & Fauzan (2025)[11]	<i>Natural Language Processing (NLP)</i> menggunakan <i>Large Language Model (LLM)</i> dengan alur kerja Setting Table Schema, Semantic Search, Rank Result, dan Natural Language Interface for	<i>Chatbot</i> berhasil merespons pertanyaan bahasa alami dengan tingkat akurasi 100% berdasarkan 8 skenario pengujian. Evaluasi <i>User Experience</i> Questionnaire (UEQ)	Model LLM: Penelitian tersebut menggunakan LLM umum (via LangChain), sedangkan penelitian Anda secara spesifik menggunakan <i>Local LLM</i> Gemma yang berjalan secara privat. Integrasi <i>WhatsApp</i> : Penelitian tersebut menggunakan integrasi <i>WhatsApp</i> (tidak disebutkan library spesifiknya), sementara Anda menggunakan <i>Evolution API</i> yang dikenal lebih stabil.

No	Peneliti	Metode / Teknologi	Hasil Penelitian	Perbedaan dengan Penelitian Ini
		<i>Databases</i> (NLIDB). <i>Framework</i> LangChain, <i>database</i> MySQL, dan integrasi platform <i>WhatsApp</i> .	menunjukka n nilai "Excellent" pada semua skala, terutama pada aspek Kejelasan (2,08) dan Ketepatan (2,00).	

Berdasarkan tabel di atas, kebaruan (*novelty*) dari penelitian ini terletak pada penggunaan *Local Large Language Model* (Gemma) yang di-host secara mandiri menggunakan *Ollama*, dipadukan dengan *Evolution API* untuk manajemen sesi *WhatsApp* yang lebih handal, serta dibangun di atas arsitektur *NestJS* yang mendukung *skalabilitas multi-tenant* bagi pelayanan publik di tingkat desa.

2.2 Landasan Teori

Landasan teori pada penelitian ini mencakup teori-teori dasar dalam bidang rekayasa perangkat lunak dan sistem informasi yang digunakan sebagai fondasi keilmuan dalam perancangan sistem. Teori-teori tersebut meliputi :

2.2.1 Pelayanan Publik Tingkat Desa

Pelayanan publik merupakan kegiatan atau rangkaian kegiatan dalam rangka pemenuhan kebutuhan pelayanan sesuai dengan peraturan perundang-undangan bagi setiap warga negara atas barang, jasa, dan/atau pelayanan administratif yang disediakan oleh penyelenggara pelayanan publik. Pada tingkat desa, pelayanan publik menjadi ujung tombak interaksi langsung antara pemerintah dan masyarakat.

Berdasarkan Undang-Undang Nomor 25 Tahun 2009 tentang Pelayanan Publik, setiap penyelenggara pelayanan publik wajib memenuhi standar pelayanan yang meliputi prosedur pelayanan, waktu penyelesaian, biaya pelayanan, produk layanan, sarana dan prasarana, serta kompetensi pelaksana. Dalam konteks desa, layanan yang diberikan umumnya berupa administrasi kependudukan seperti pembuatan surat keterangan domisili, surat pengantar, serta informasi terkait program bantuan dan kegiatan desa.

Namun, dalam implementasinya, pelayanan publik di tingkat desa masih menghadapi berbagai kendala, seperti keterbatasan sumber daya manusia, keterbatasan waktu operasional, serta rendahnya pemanfaatan teknologi informasi. Hal ini menyebabkan proses pelayanan menjadi kurang

efisien dan sering kali menimbulkan antrean panjang. Selain itu, kebutuhan informasi masyarakat yang semakin meningkat menuntut adanya sistem pelayanan yang dapat memberikan informasi secara cepat, akurat, dan mudah diakses kapan saja.

Dengan adanya perkembangan teknologi digital, konsep pelayanan publik mulai bertransformasi menuju pelayanan berbasis elektronik (*E-Government*). Salah satu bentuk implementasinya adalah penggunaan sistem otomatis seperti *chatbot* untuk memberikan layanan informasi kepada masyarakat secara *real-time* tanpa dibatasi oleh waktu dan tempat.

2.2.2 Large Language Model (LLM) dan Gemma.

Large Language Model (LLM) merupakan model kecerdasan buatan yang dirancang untuk memahami, memproses, dan menghasilkan bahasa alami manusia dalam skala besar[12]. LLM dibangun menggunakan arsitektur *deep learning*, khususnya *transformer*, yang memungkinkan model untuk memahami konteks dalam teks secara lebih mendalam dibandingkan metode *Natural Language Processing* (NLP) tradisional[13].

LLM dilatih menggunakan data dalam jumlah besar sehingga mampu melakukan berbagai tugas seperti menjawab pertanyaan, merangkum teks, menerjemahkan bahasa, hingga melakukan percakapan interaktif dengan pengguna[14]. Dalam konteks *chatbot*, LLM memungkinkan sistem untuk memberikan respons yang lebih natural, fleksibel, dan kontekstual dibandingkan dengan *chatbot* berbasis aturan[15].

Salah satu model LLM yang dapat digunakan adalah Gemma, yaitu model *open-weight* yang memungkinkan pengguna untuk menjalankan model secara lokal. Berbeda dengan LLM berbasis *cloud* seperti *OpenAI* atau *Gemini*, penggunaan model lokal memberikan beberapa keunggulan, antara lain:

1. Menjamin keamanan data karena seluruh data diproses secara lokal tanpa dikirim ke server pihak ketiga, sehingga dapat mengurangi risiko kebocoran informasi..
2. Meningkatkan privasi pengguna dengan memastikan seluruh proses pemrosesan data dilakukan di dalam lingkungan sistem yang dikelola sendiri.
3. Memberikan kontrol yang lebih fleksibel terhadap model sehingga pengguna dapat melakukan penyesuaian sesuai kebutuhan, termasuk proses *fine-tuning* dan pengaturan akses data.
4. Mendukung operasional secara *offline* sehingga sistem tetap dapat berfungsi tanpa ketergantungan pada koneksi internet eksternal.

Dalam konteks pelayanan publik desa, penggunaan *Local LLM* seperti Gemma sangat penting untuk menjaga kerahasiaan data kependudukan serta meningkatkan kepercayaan masyarakat terhadap sistem digital yang digunakan.

2.2.3 Prompt Engineering

Prompt Engineering merupakan teknik dalam merancang dan menyusun instruksi yang diberikan kepada model LLM agar menghasilkan

output yang sesuai dengan kebutuhan pengguna. *Prompt* yang baik akan membantu model memahami konteks, tujuan, serta batasan dari pertanyaan yang diajukan[16].

Dalam implementasi *chatbot* pelayanan publik desa, *Prompt Engineering* memiliki peran yang sangat penting untuk memastikan bahwa sistem:

1. Memberikan jawaban yang relevan dengan konteks pelayanan desa.
2. Menghindari informasi yang tidak akurat atau bersifat halusinasi.
3. Menggunakan bahasa yang mudah dipahami oleh masyarakat umum.
4. Membatasi jawaban hanya pada informasi yang telah ditentukan.

Beberapa teknik dalam *Prompt Engineering* yang umum digunakan antara lain:

1. *Instruction-Based Prompting*, yaitu teknik yang memberikan instruksi secara jelas kepada model mengenai peran, tugas, atau tujuan yang harus dijalankan.
2. *Contextual Prompting*, yaitu teknik yang menyertakan konteks tambahan seperti data, dokumen referensi, atau informasi pendukung lainnya untuk membantu model menghasilkan jawaban yang lebih relevan.
3. *Few-shot Prompting*, yaitu teknik yang memberikan beberapa contoh pertanyaan dan jawaban sebagai acuan agar model dapat memahami pola respons yang diharapkan.

4. *Constraint Prompting*, yaitu teknik yang memberikan batasan tertentu kepada model agar jawaban yang dihasilkan tetap sesuai dengan topik dan kebutuhan pengguna.

Dengan penerapan *Prompt Engineering* yang tepat, *chatbot* dapat memberikan jawaban yang lebih akurat, konsisten, dan sesuai dengan kebutuhan pelayanan publik.

2.2.4 Framework NestJS

NestJS merupakan *Framework Backend* berbasis *Node.js* yang menggunakan *TypeScript* sebagai bahasa pemrograman utama. *Framework* ini dirancang untuk membangun aplikasi *server-side* yang efisien, skalabel, dan mudah dikelola dengan pendekatan arsitektur modular.

NestJS mengadopsi konsep *dependency injection*, yang memungkinkan pengelolaan komponen aplikasi secara terstruktur dan terpisah. Selain itu, *NestJS* juga mendukung penggunaan arsitektur berbasis *Model View Controller* (MVC) maupun *microservices*, sehingga cocok untuk pengembangan aplikasi yang kompleks.

Beberapa keunggulan penggunaan *NestJS* antara lain:

1. Memiliki struktur kode yang terorganisir sehingga memudahkan proses pengembangan dan pemeliharaan sistem.
2. Berbasis *TypeScript* yang memberikan keamanan tipe data serta membantu meningkatkan kualitas kode.

3. Mendukung *skalabilitas* tinggi sehingga cocok untuk pengembangan aplikasi berskala besar.
4. Menyediakan dukungan *middleware* dan *interceptor* yang mempermudah pengolahan *request* dan *response*.
5. Mudah diintegrasikan dengan berbagai *API* dan layanan eksternal yang dibutuhkan sistem.

Dalam penelitian ini, *NestJS* digunakan sebagai *Backend* utama untuk mengelola logika aplikasi, integrasi dengan *Evolution API*, serta komunikasi dengan model LLM. Penggunaan *NestJS* diharapkan dapat menghasilkan sistem yang *robust* dan mudah dikembangkan di masa depan.

2.2.5 PostgreSQL dan Redis

Dalam pengembangan sistem *Backend* yang kompleks seperti *chatbot* pelayanan publik, pemilihan teknologi penyimpanan data menjadi aspek penting untuk menjamin performa, keandalan, dan skalabilitas sistem. Pada penelitian ini digunakan dua jenis sistem penyimpanan, yaitu *PostgreSQL* sebagai basis data relasional dan *Redis* sebagai penyimpanan *in-memory* untuk kebutuhan *caching* dan manajemen sesi.

PostgreSQL merupakan sistem manajemen basis data relasional yang bersifat *open-source* dan memiliki reputasi tinggi dalam hal stabilitas, keamanan, serta kepatuhan terhadap standar SQL. *PostgreSQL* mendukung berbagai fitur lanjutan seperti transaksi ACID (*Atomicity*, *Consistency*, *Isolation*, *Durability*), *indexing*, serta kemampuan pengolahan data yang kompleks.

Dalam konteks penelitian ini, *PostgreSQL* digunakan untuk menyimpan data yang bersifat *persisten*, seperti:

1. Data pengguna (masyarakat yang berinteraksi dengan *chatbot*)
2. Riwayat percakapan (*chat history*)
3. Data layanan desa (informasi administratif, prosedur, dan persyaratan)
4. Log aktivitas sistem

Penggunaan *PostgreSQL* memberikan keuntungan dalam menjaga konsistensi dan integritas data, terutama untuk data pelayanan publik yang membutuhkan keakuratan tinggi. Selain itu, *PostgreSQL* juga dapat diintegrasikan dengan ORM (*Object Relational Mapping*) seperti Prisma atau *TypeORM* yang umum digunakan dalam *Framework NestJS*, sehingga mempermudah proses pengelolaan *database*.

Di sisi lain, *Redis* merupakan sistem penyimpanan data berbasis *key-value* yang bekerja secara *in-memory*, sehingga memiliki kecepatan akses yang sangat tinggi dibandingkan dengan *database* tradisional. *Redis* umumnya digunakan untuk kebutuhan *caching*, *session management*, serta penyimpanan data sementara.

Dalam penelitian ini, *Redis* dimanfaatkan untuk:

1. Mendukung mekanisme *caching* yang memungkinkan penyimpanan hasil respons *chatbot* yang sering diakses sehingga dapat mempercepat waktu respons sistem.

2. Menyediakan fitur *session management* untuk menyimpan konteks percakapan pengguna sehingga interaksi dengan *chatbot* dapat berlangsung secara berkelanjutan (*stateful*).
3. Mendukung implementasi *queue* atau *message broker* yang dapat digunakan untuk mengelola antrean proses ketika sistem mengalami beban kerja yang tinggi.

Kombinasi antara *PostgreSQL* dan *Redis* memberikan arsitektur penyimpanan yang optimal, di mana *PostgreSQL* menangani data yang bersifat permanen dan kritis, sedangkan *Redis* menangani data sementara yang membutuhkan akses cepat. Pendekatan ini sangat sesuai untuk sistem *chatbot* yang memerlukan respons *real-time* sekaligus menjaga konsistensi data dalam jangka panjang.

Dengan demikian, penggunaan *PostgreSQL* dan *Redis* dalam penelitian ini diharapkan dapat meningkatkan performa, efisiensi, serta keandalan sistem *chatbot* pelayanan publik desa yang dikembangkan.

2.2.6 Knowledge Base

Knowledge Base atau basis pengetahuan merupakan kumpulan informasi, data, fakta, aturan, dan dokumen yang disusun secara terstruktur untuk mendukung proses penyimpanan, pengelolaan, serta penyajian informasi dalam suatu sistem. Pada sistem berbasis kecerdasan buatan, *Knowledge Base* berfungsi sebagai sumber pengetahuan yang digunakan oleh sistem untuk menjawab pertanyaan, memberikan rekomendasi, atau menyelesaikan permasalahan sesuai dengan konteks yang diberikan.

Knowledge Base merupakan komponen utama dalam sistem berbasis pengetahuan yang berisi fakta dan aturan yang digunakan untuk mendukung proses pengambilan keputusan. Dalam perkembangannya, konsep *Knowledge Base* tidak hanya digunakan pada sistem pakar, tetapi juga pada *chatbot* dan sistem berbasis *Artificial Intelligence* (AI) yang memanfaatkan data sebagai sumber informasi utama.

Pada sistem *chatbot*, *Knowledge Base* berperan sebagai repositori informasi yang menjadi acuan dalam menghasilkan respons kepada pengguna. Informasi yang tersimpan dapat berupa dokumen, *Frequently Asked Questions* (FAQ), Standar Operasional Prosedur (SOP), data layanan, maupun informasi lainnya yang relevan dengan kebutuhan pengguna. Ketika pengguna mengajukan pertanyaan, sistem akan melakukan pencarian terhadap data yang tersedia pada *Knowledge Base* untuk menemukan informasi yang paling sesuai sebelum menghasilkan jawaban.

Dalam penelitian ini, *Knowledge Base* digunakan sebagai sumber informasi utama bagi *Private Chatbot* Pelayanan Publik Desa. Basis pengetahuan yang digunakan berasal dari dokumen pelayanan desa, Standar Operasional Prosedur (SOP), persyaratan administrasi, informasi layanan publik, serta data lain yang berkaitan dengan kebutuhan masyarakat. Data tersebut dikelola oleh administrator melalui sistem yang telah disediakan sehingga informasi yang diberikan *chatbot* tetap akurat, relevan, dan dapat diperbarui sesuai kebutuhan.

Penerapan *Knowledge Base* pada *chatbot* pelayanan publik memberikan beberapa manfaat, di antaranya meningkatkan konsistensi informasi yang diberikan kepada masyarakat, mempercepat proses pencarian informasi, mengurangi beban kerja perangkat desa dalam menjawab pertanyaan yang berulang, serta mendukung layanan informasi yang dapat diakses selama 24 jam. Dengan adanya *Knowledge Base* yang terstruktur, *chatbot* dapat memberikan respons yang lebih tepat dan sesuai dengan informasi resmi yang dimiliki oleh pemerintah desa.

2.2.7 Artificial Intelligence (AI)

Artificial Intelligence (AI) atau kecerdasan buatan merupakan cabang ilmu komputer yang berfokus pada pengembangan sistem atau mesin yang mampu meniru kemampuan berpikir manusia, seperti belajar, memahami, menganalisis, mengambil keputusan, dan menyelesaikan permasalahan. AI dirancang untuk memungkinkan komputer melakukan tugas-tugas yang umumnya membutuhkan kecerdasan manusia sehingga dapat membantu meningkatkan efisiensi dan efektivitas dalam berbagai bidang.

Menurut John McCarthy, *Artificial Intelligence* adalah ilmu dan teknik dalam membuat mesin cerdas, khususnya program komputer yang mampu melakukan tugas yang memerlukan kecerdasan manusia. Seiring perkembangan teknologi, AI telah diterapkan pada berbagai bidang, seperti kesehatan, pendidikan, industri, keuangan, pemerintahan, serta layanan publik.

Secara umum, *Artificial Intelligence* memiliki beberapa kemampuan utama, yaitu pembelajaran (*learning*), penalaran (*reasoning*), pemecahan masalah (problem solving), pemrosesan bahasa alami (*Natural Language Processing*), dan pengambilan keputusan (*decision making*). Kemampuan-kemampuan tersebut memungkinkan sistem AI untuk memahami data yang diterima, mengidentifikasi pola, serta menghasilkan respons atau tindakan yang sesuai berdasarkan informasi yang tersedia.

Salah satu perkembangan penting dalam bidang AI adalah munculnya teknologi *Natural Language Processing* (NLP), yaitu kemampuan sistem untuk memahami, mengolah, dan menghasilkan bahasa manusia. Teknologi ini menjadi dasar pengembangan *chatbot* modern yang mampu berinteraksi dengan pengguna menggunakan bahasa alami. Dengan memanfaatkan NLP, *chatbot* dapat memahami maksud pertanyaan pengguna dan memberikan jawaban yang relevan sesuai konteks percakapan.

Dalam beberapa tahun terakhir, perkembangan *Artificial Intelligence* semakin pesat dengan hadirnya *Large Language Model* (LLM), yaitu model kecerdasan buatan yang dilatih menggunakan data dalam jumlah besar untuk memahami dan menghasilkan teks secara lebih alami. LLM memungkinkan *chatbot* memberikan respons yang lebih kontekstual, informatif, dan menyerupai komunikasi manusia dibandingkan *chatbot* berbasis aturan (*rule-Based chatbot*).

Pada penelitian ini, *Artificial Intelligence* digunakan sebagai teknologi utama dalam pengembangan *Private Chatbot* Pelayanan Publik Desa. Sistem memanfaatkan model *Large Language Model* (LLM) Gemma untuk memahami pertanyaan yang diajukan masyarakat melalui *WhatsApp* dan menghasilkan jawaban berdasarkan informasi yang tersedia pada *Knowledge Base*. Dengan penerapan AI, *chatbot* diharapkan mampu memberikan layanan informasi yang lebih cepat, akurat, dan mudah diakses oleh masyarakat tanpa harus bergantung sepenuhnya pada petugas pelayanan desa.

2.2.8 REST API

REST API (*Representational State Transfer Application Programming Interface*) merupakan suatu arsitektur komunikasi yang digunakan untuk memungkinkan pertukaran data antara dua atau lebih sistem melalui jaringan internet. *REST API* memungkinkan aplikasi yang berbeda untuk saling berkomunikasi dan bertukar informasi menggunakan protokol HTTP dengan format data yang umum digunakan, seperti JSON (*JavaScript Object Notation*).

Konsep REST pertama kali diperkenalkan oleh Roy Fielding dalam disertasinya pada tahun 2000 sebagai gaya arsitektur perangkat lunak yang dirancang untuk sistem terdistribusi. REST memiliki beberapa karakteristik utama, yaitu bersifat *stateless*, menggunakan antarmuka yang seragam (*uniform interface*), serta memanfaatkan sumber daya (*resource*) yang diidentifikasi melalui *Uniform Resource Identifier* (URI).

Dalam implementasinya, *REST API* menggunakan metode HTTP untuk melakukan operasi terhadap suatu *resource*. Metode HTTP yang umum digunakan antara lain:

1. *GET*, digunakan untuk mengambil atau membaca data dari server.
2. *POST*, digunakan untuk menambahkan atau membuat data baru pada server.
3. *PUT*, digunakan untuk memperbarui seluruh data yang telah ada.
4. *PATCH*, digunakan untuk memperbarui sebagian data tertentu.
5. *DELETE*, digunakan untuk menghapus data dari server.

REST API banyak digunakan dalam pengembangan aplikasi modern karena memiliki beberapa kelebihan, seperti mudah diimplementasikan, ringan, fleksibel, serta mendukung integrasi antar platform. Dengan menggunakan format JSON sebagai media pertukaran data, *REST API* dapat digunakan oleh berbagai jenis aplikasi, baik aplikasi *web*, *mobile*, maupun sistem berbasis layanan lainnya.

Pada penelitian ini, *REST API* digunakan sebagai mekanisme komunikasi antara *frontend*, *Backend*, dan layanan eksternal yang terintegrasi dalam sistem. *Backend* yang dibangun menggunakan *Framework NestJS* menyediakan berbagai *endpoint API* yang digunakan untuk mengelola data pengguna, *Knowledge Base*, *autoreply*, dan data layanan publik desa. Selain itu, *REST API* juga digunakan dalam proses integrasi dengan *Evolution API* sebagai *WhatsApp Gateway* serta komunikasi dengan layanan *Large Language Model (LLM)* untuk menghasilkan respons *chatbot*.

Dengan penerapan REST API, proses pertukaran data dalam sistem dapat dilakukan secara terstruktur, efisien, dan mudah dikembangkan. Hal ini memungkinkan setiap komponen sistem bekerja secara terintegrasi sehingga mendukung kinerja *Private Chatbot* Pelayanan Publik Desa dalam memberikan layanan informasi kepada masyarakat secara cepat dan akurat.

2.2.9 Evolution API dan WhatsApp Gateway.

WhatsApp merupakan salah satu aplikasi pesan instan yang paling banyak digunakan di Indonesia, sehingga menjadi media yang efektif untuk menyampaikan layanan publik berbasis digital[17]. Untuk mengintegrasikan sistem *chatbot* dengan *WhatsApp*, diperlukan sebuah *Gateway* atau API yang dapat menghubungkan aplikasi *Backend* dengan layanan *WhatsApp*.

Evolution API merupakan salah satu solusi yang dapat digunakan untuk membangun *WhatsApp Gateway* secara fleksibel. API ini memungkinkan sistem untuk mengirim dan menerima pesan secara otomatis, serta mengelola interaksi pengguna secara *real-time*.

Beberapa konsep utama dalam penggunaan *Evolution API* antara lain:

1. Mendukung mekanisme *webhook* yang memungkinkan sistem menerima notifikasi secara otomatis ketika terdapat pesan masuk dari pengguna, sehingga *Backend* dapat memproses pesan dan memberikan respons secara langsung.

2. Menyediakan fitur *session management* untuk mengelola koneksi antara server dan akun *WhatsApp* agar tetap aktif, stabil, dan dapat digunakan secara berkelanjutan.
3. Mendukung proses *message handling* yang memungkinkan pengolahan pesan masuk serta pengiriman balasan secara otomatis.
4. Menyediakan fitur *automation* yang memungkinkan pengiriman pesan secara otomatis berdasarkan kondisi atau aturan tertentu yang telah ditentukan.

Dengan integrasi *Evolution API*, *chatbot* dapat diakses langsung melalui *WhatsApp* tanpa memerlukan aplikasi tambahan. Hal ini meningkatkan kemudahan akses bagi masyarakat serta mempercepat proses adopsi teknologi.

2.2.10 Retrieval-Augmented Generation (RAG)

Retrieval-Augmented Generation (RAG) merupakan metode yang menggabungkan kemampuan pencarian informasi dengan kemampuan generasi teks dari *Large Language Model* (LLM). Metode ini dikembangkan untuk mengatasi keterbatasan LLM yang sering menghasilkan informasi yang tidak akurat atau tidak sesuai dengan data aktual.

Pada pendekatan RAG, sistem tidak hanya mengandalkan pengetahuan yang telah dipelajari oleh model saat proses pelatihan, tetapi juga mengambil informasi dari sumber data eksternal yang relevan sebelum menghasilkan jawaban. Informasi tersebut dapat berasal dari dokumen, basis pengetahuan, maupun *database* yang dimiliki oleh organisasi.

Secara umum, mekanisme RAG terdiri dari dua tahapan utama, yaitu:

1. *Retrieval* (Pencarian Informasi)

Tahap ini bertujuan untuk mencari dan mengambil informasi yang paling relevan dari sumber data berdasarkan pertanyaan atau permintaan pengguna.

2. *Generation* (Generasi Jawaban)

Informasi yang telah diperoleh pada tahap *retrieval* kemudian diberikan sebagai konteks kepada LLM untuk menghasilkan jawaban yang lebih akurat, relevan, dan sesuai dengan data yang tersedia.

Penerapan RAG memberikan beberapa keuntungan, antara lain:

1. Meningkatkan akurasi jawaban dengan memanfaatkan informasi yang berasal dari sumber data aktual.
2. Mengurangi risiko terjadinya *hallucination* pada model LLM.
3. Memungkinkan pembaruan informasi tanpa perlu melakukan pelatihan ulang model.
4. Menjamin bahwa jawaban yang diberikan sesuai dengan basis pengetahuan yang dimiliki sistem.
5. Meningkatkan kepercayaan pengguna terhadap informasi yang dihasilkan oleh *chatbot*.

Dalam penelitian ini, konsep RAG digunakan untuk mengintegrasikan *chatbot* dengan basis pengetahuan pelayanan publik desa. Ketika pengguna mengajukan pertanyaan, sistem akan mencari informasi yang relevan dari

data layanan desa yang tersimpan pada *Knowledge Base*. Informasi tersebut kemudian digunakan sebagai konteks bagi model LLM untuk menghasilkan jawaban yang lebih akurat, konsisten, dan sesuai dengan informasi resmi yang dimiliki pemerintah desa.

2.2.11 E-Government

E-Government (Electronic Government) merupakan pemanfaatan teknologi informasi dan komunikasi oleh instansi pemerintah untuk meningkatkan kualitas layanan publik, efektivitas administrasi pemerintahan, serta transparansi dalam penyelenggaraan pemerintahan. Konsep *E-Government* bertujuan untuk mengubah proses pelayanan konvensional menjadi pelayanan berbasis digital yang lebih cepat, efisien, dan mudah diakses oleh masyarakat.

Menurut World Bank, *E-Government* merupakan penggunaan teknologi informasi oleh lembaga pemerintah yang memiliki kemampuan untuk mentransformasikan hubungan antara pemerintah, masyarakat, pelaku bisnis, dan pihak-pihak terkait lainnya. Penerapan *E-Government* tidak hanya berfokus pada digitalisasi layanan, tetapi juga pada peningkatan kualitas tata kelola pemerintahan (*good governance*).

Secara umum, implementasi *E-Government* dapat diklasifikasikan ke dalam beberapa model interaksi, yaitu:

1. *Government to Citizen* (G2C), yaitu layanan yang diberikan pemerintah secara langsung kepada masyarakat, seperti pelayanan administrasi

kependudukan, penyampaian informasi publik, dan pengaduan masyarakat.

2. *Government to Business* (G2B), yaitu layanan pemerintah yang ditujukan kepada pelaku usaha, seperti perizinan usaha dan pengadaan barang atau jasa.
3. *Government to Government* (G2G), yaitu pertukaran informasi dan koordinasi antarinstansi pemerintahan.
4. *Government to Employee* (G2E), yaitu layanan yang diberikan pemerintah kepada aparatur atau pegawai pemerintah.

Penerapan *E-Government* memberikan berbagai manfaat, antara lain:

1. Meningkatkan efisiensi dan efektivitas pelayanan publik.
2. Mempermudah akses masyarakat terhadap informasi dan layanan pemerintah.
3. Meningkatkan transparansi dan akuntabilitas penyelenggaraan pemerintahan.
4. Mengurangi biaya operasional pelayanan publik.
5. Meningkatkan partisipasi masyarakat dalam proses pemerintahan.

Dalam konteks pemerintahan desa, penerapan *E-Government* menjadi salah satu upaya untuk meningkatkan kualitas pelayanan kepada masyarakat melalui pemanfaatan teknologi digital. Salah satu bentuk implementasinya adalah penggunaan *chatbot* berbasis kecerdasan buatan yang dapat memberikan informasi dan layanan kepada masyarakat secara otomatis, cepat, dan dapat diakses selama 24 jam. Dengan demikian, penerapan *chatbot*

pelayanan publik desa sejalan dengan tujuan *E-Government* dalam mewujudkan pelayanan publik yang lebih efektif, efisien, dan berorientasi pada kebutuhan masyarakat.

2.2.12 *React*

React merupakan pustaka *JavaScript* yang dikembangkan oleh Meta *Platforms* untuk membangun antarmuka pengguna pada aplikasi *web*. *React* dirancang untuk mempermudah pengembangan aplikasi yang bersifat interaktif dan dinamis melalui pendekatan berbasis komponen.

Dalam *React*, antarmuka aplikasi dibangun menggunakan komponen-komponen yang dapat digunakan kembali, sehingga proses pengembangan menjadi lebih terstruktur dan efisien. Setiap komponen memiliki fungsi dan tampilan yang dapat dikelola secara terpisah, sehingga memudahkan proses pemeliharaan dan pengembangan aplikasi.

Salah satu keunggulan *React* adalah penggunaan Virtual DOM (*Document Object Model*), yaitu mekanisme yang memungkinkan *React* memperbarui bagian tertentu dari halaman *web* tanpa harus memuat ulang seluruh halaman. Dengan demikian, performa aplikasi dapat ditingkatkan dan pengalaman pengguna menjadi lebih baik.

Selain itu, *React* mendukung pengembangan *Single Page Application* (SPA), yaitu aplikasi *web* yang dapat menampilkan dan memperbarui konten secara dinamis tanpa perlu melakukan perpindahan halaman secara penuh.

Kemampuan ini menjadikan *React* banyak digunakan dalam pengembangan aplikasi modern yang membutuhkan antarmuka responsif dan interaktif.

Pada penelitian ini, *React* digunakan sebagai teknologi *frontend* untuk membangun antarmuka sistem administrasi *chatbot*. *React* berfungsi untuk menampilkan data, mengelola interaksi pengguna, serta berkomunikasi dengan *Backend* melalui *REST API* sehingga pengguna dapat mengelola *Knowledge Base*, data layanan, dan konfigurasi *chatbot* secara lebih mudah dan efisien.

2.2.13 *Tailwind CSS*

Tailwind CSS merupakan *Framework CSS* yang menerapkan pendekatan *utility-first* dalam proses pengembangan antarmuka pengguna. Berbeda dengan *Framework CSS* konvensional yang menyediakan komponen siap pakai, *Tailwind CSS* menyediakan kumpulan *utility class* yang dapat digunakan secara langsung untuk mengatur tampilan elemen, seperti warna, ukuran, margin, padding, tipografi, dan tata letak.

Pendekatan *utility-first* memungkinkan pengembang membangun desain antarmuka secara lebih fleksibel tanpa harus menulis banyak kode *CSS* secara terpisah. Setiap elemen dapat dikustomisasi langsung melalui kombinasi *utility class* yang tersedia sehingga proses pengembangan menjadi lebih cepat dan konsisten.

Tailwind CSS juga mendukung konsep *responsive design* yang memungkinkan tampilan aplikasi menyesuaikan berbagai ukuran layar, baik

pada perangkat desktop, tablet, maupun smartphone. Selain itu, *Tailwind CSS* memiliki fitur optimasi yang dapat menghapus *class* yang tidak digunakan sehingga ukuran file CSS menjadi lebih kecil dan performa aplikasi lebih baik.

Dalam pengembangan aplikasi modern, *Tailwind CSS* sering digunakan bersama *React* karena mampu mempercepat proses pembuatan antarmuka yang responsif dan mudah dipelihara. Kombinasi kedua teknologi tersebut memungkinkan pengembang membangun aplikasi *web* yang memiliki tampilan konsisten, modern, dan nyaman digunakan.

Pada penelitian ini, *Tailwind CSS* digunakan untuk membangun dan mengatur tampilan antarmuka sistem administrasi *chatbot*. Penggunaan *Tailwind CSS* bertujuan untuk menghasilkan desain yang responsif, konsisten, dan mudah dikembangkan sehingga dapat meningkatkan pengalaman pengguna dalam mengelola sistem *chatbot* pelayanan publik desa.

2.2.14 Software Development Life Cycle (SDLC)

Software Development Life Cycle (SDLC) adalah suatu metode yang digunakan sebagai pedoman dalam proses pengembangan perangkat lunak secara sistematis, terstruktur, dan berurutan. Metode ini mencakup serangkaian tahapan yang dimulai dari perencanaan, analisis kebutuhan, perancangan sistem, implementasi, pengujian, hingga pemeliharaan perangkat lunak[18]. Melalui penerapan SDLC, proses pengembangan dapat

berlangsung secara lebih terarah sehingga menghasilkan sistem yang sesuai dengan kebutuhan pengguna.

Menurut Pressman (2015), SDLC merupakan sebuah kerangka kerja yang menjelaskan berbagai aktivitas yang dilakukan selama siklus pengembangan perangkat lunak, dimulai dari proses identifikasi kebutuhan pengguna hingga tahap pemeliharaan sistem setelah diimplementasikan. Penerapan SDLC bertujuan untuk menghasilkan perangkat lunak yang berkualitas, meminimalkan risiko kesalahan selama proses pengembangan, serta memastikan sistem mampu memenuhi kebutuhan dan harapan pengguna.

Secara umum, SDLC terdiri atas beberapa tahapan utama, yaitu analisis kebutuhan, perancangan sistem, implementasi, pengujian, dan pemeliharaan. Masing-masing tahapan memiliki fungsi dan aktivitas yang berbeda, tetapi saling berhubungan dalam membentuk proses pengembangan perangkat lunak yang efektif dan menghasilkan sistem yang berkualitas.

Berbagai model SDLC telah dikembangkan untuk menyesuaikan karakteristik dan kebutuhan suatu proyek, di antaranya Waterfall, Prototype, Spiral, Rapid Application Development (RAD), Agile, dan Scrum[18]. Pada penelitian ini, model yang digunakan adalah Waterfall karena memiliki tahapan yang bersifat sistematis dan berurutan, sehingga sesuai untuk pengembangan sistem yang kebutuhan dan ruang lingkupnya telah ditentukan secara jelas sejak awal penelitian.