BAB II

TINJAUAN PUSTAKA DAN DASAR TEORI

2.1 Tinjauan Pustaka

Penelitian ini memusatkan perhatian pada pengembangan *Chatbot* dengan memanfaatkan sejumlah sumber yang relevan terkait penggunaan *Dialogflow* API sebagai *Natural Language Processing* (NLP) dan teknologi *websocket*. Berikut adalah beberapa penelitian yang telah dilakukan dalam konteks topik ini.

Tabel 2. 1 Penelitian Terdahulu

NO	Penulis	Data	Teknologi	Hasil
1	(Harahap & Liza, 2020)	Penggunaan *Dialogflow* sebagai *natural language processing* (NLP) untuk kebutuhan fitur *chatbot* Aplikasi *HelpDesk* Kantor Pajak Pratama Binjai.	*Dialogflow*	*Dialogflow* dapat digunakan sebagai *natural language processing* (NLP) untuk kebutuhan fitur *chatbot*.
2	(Somya, 2018)	Pengembangan aplikasi berbasis web *chatting* menggunakan teknologi *websocket*	*Socket.io, Codeigniter, Framework foundation, oracle*	*Socket.IO* cocok untuk aplikasi *chatting*, sebagai komunikasi *realtime.*
3	(Wijaya & Pebriantara, 2018)	Pembuatan aplikasi *mobile chatbot* untuk pembelajaran dengan mengintegrasikan ke layanan pembelajaran.	*Dialogflow, Moodle, MySQL, Android native programming language*	*Dialogflow* dapat digunakan untuk membangun program *chatbot* untuk pembelajaran dengan *LMS Moodle*

4	(Rifai, 2020)	Pembuatan *chatbot* telegram untuk informasi akademik menggunakan *DIalogflow.*	*Laravel, Dialogflow, MongoDB*	*Dialogflow* dapat digunakan untuk membangun *chatbot* telegram
5	(Kamal & Andika, 2022)	Pembuatan aplikasi *chatbot* menggunakan d*ialogflow* untuk menangani komplain dari konsumen dan menyimpan sebagai laporan ke Google *Sheet API*	*Dialogflow, Google Sheet API, FastAPI*	*Dialogflow* dapat digunakan untuk membuat *chatbot* yang dapat digunakan untuk mengumpulkan data sebagai sebuah laporan dalam bentuk *Spreadsheet.*
6	(Usulan, 2024)	Pengambangan aplikasI *chatbot* menggunakan d*ialogflow* dengan *websocket* sebagai teknologi komunikasi data	*Dialogflow, NodeJS, TypeScript, NestJS, Socket.io, NextJS,* PostgreSQL, Docker	*Dialogflow* dapat digunakan dengan menggunakan *websocket* untuk membuat *chatbot*

Penelitian ini bertujuan untuk mengembangkan aplikasi chatbot berbasis website dengan menggunakan *dialogflow* sebagai teknologi *Natural Language Processing* (NLP) dan websocket sebagai sarana komunikasi *real-time*. Sebelumnya, telah terbukti bahwa *dialogflow* dapat digunakan sebagai NLP dalam pembuatan aplikasi *chatbot* berdasarkan penelitian sebelumnya. Selain itu, penggunaan *websocket* sebagai komunikasi *real-time* juga telah dikonfirmasi melalui penelitian sebelumnya oleh (Somya, 2018) yang membahas pengembangan aplikasi *chat* berbasis web di PT Pura Barutama Kudus dan menggunakan *Socket.IO* sebagai teknologi untuk komunikasi *real-time*.

Namun, perbedaan penting dari penelitian ini dengan penelitian-penelitian sebelumnya adalah fokus khusus pada pengembangan aplikasi *chatbot* berbasis website yang memanfaatkan *dialogflow* sebagai teknologi NLP dan *websocket* sebagai sarana komunikasi *real-time*. Penelitian ini melibatkan implementasi *dialogflow* untuk memahami bahasa manusia dengan alami dan penggunaan *websocket* untuk menyediakan komunikasi *real-time* antara *chatbot* dan pengguna. Melalui penelitian ini, diharapkan dapat dikembangkan aplikasi *chatbot* berbasis *website* yang interaktif, responsif, dan efektif dalam menyediakan layanan kepada pengguna.

2.2 Dasar Teori

1. Chatbot

Chatbot adalah program komputer yang dirancang untuk berkomunikasi dengan manusia melalui antarmuka percakapan, baik itu dalam bentuk teks atau suara. *Chatbot* berfokus pada interaksi percakapan dengan pengguna. Ini memungkinkan pengguna untuk berkomunikasi dengan sistem secara alami, mirip dengan berkomunikasi dengan manusia.

Pada pengaplikasiannya *chatbot* dapat digunakan untuk berbagai fungsi diantaranya yaitu memberikan informasi, membantu pengguna menyelesaikan tugas, melakukan reservasi, atau memberikan dukungan pelanggan *(Customer Service)*. Selain itu chatbot juga dapat diintegrasikan dengan berbagai *platform*, termasuk situs web, aplikasi seluler, dan pesan instan. Integrasi ini

memungkinkan pengguna berinteraksi dengan *chatbot* di berbagai konteks.

Kemampuan *chatbot* dalam memberikan respon cepat dan konsten terhadap pertanyaan, permintaan, atau masalah pelanggan tentunya ini menjadi sebuah potensi besar untuk dimanfaatkan oleh seorang pembisnis dalam menunjang kinerja daripada *customer service* (Harisi & Hiwono, 2024). Dengan pemanfaatan *chatbot* tersebut harapannya mampu meningkatkan kepuasan pelanggan dalam memenuhi kebutuhannya.

2. *Natural Language Processing (NLP)*

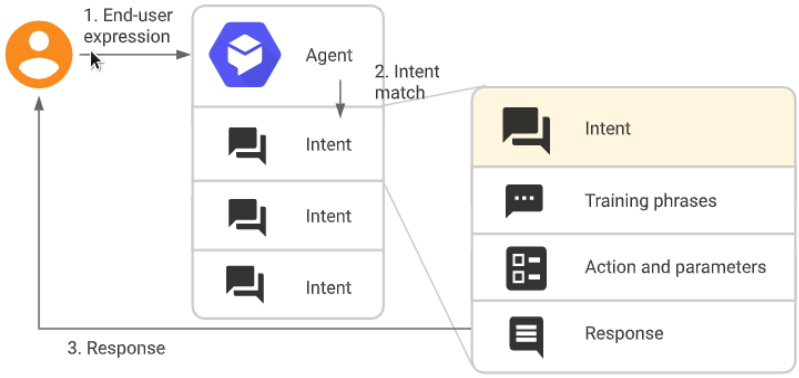
Natural Language Processing (NLP) adalah cabang ilmu komputer yang terfokus pada penerapan kecerdasan buatan *(AI)* dalam memahami, menganalisis, dan memanipulasi bahasa manusia secara alami. Menurut (Jurafsky & James H. Martin, 2008) dalam buku *"Speech and Language Processing" Natural Language Processing (NLP)* melibatkan interaksi antara manusia dan komputer melalui bahasa manusia, mencakup pemahaman semantik, analisis sintaksis, dan pengenalan entitas dalam percakapan, seperti yang diterapkan dalam *chatbot*.

Natural Language Processing (NLP) mencakup berbagai pendekatan, termasuk linguistik komputasi, pemodelan bahasa manusia berbasis aturan, serta penerapan model statistik, *machine learning*, dan *deep learning*. Integrasi teknologi ini memungkinkan program komputer untuk memproses baik bahasa lisan maupun tertulis, memahami makna dari input bahasa, dan menafsirkan niat pembicara. Dengan demikian, *Natural Language Processing (NLP)*

memainkan peran krusial dalam memfasilitasi interaksi yang lebih alami antara manusia dan sistem komputer.

3. Dialogflow

Dialogflow adalah platform pengembangan yang memanfaatkan kecerdasan buatan untuk memahami dan memproses percakapan antara pengguna dan sistem. Dengan memanfaatkan teknologi pemrosesan bahasa alami atau *Natural Language Processing (NLP)*, *dialogflow* memungkinkan pembangunan *chatbot* yang responsif dan dapat digunakan untuk berbagai aplikasi. *Platform dialogflow* terdapat 2 hal penting yang perlu diketahui, yaitu *agent dan intent*.



Gambar 2. 1 Alur Proses Dialogflow

Agent adalah entitas yang digunakan untuk mengorganisir dan mengelola model percakapan. *Agent* merespons permintaan pengguna dengan mengaktifkan intent dan memberikan respons yang sesuai. Sedangkan *Intent* adalah niat atau tujuan yang diidentifikasi oleh *dialogflow* dari percakapan

pengguna. Setiap *intent* berkaitan dengan serangkaian tindakan yang harus dilakukan oleh *agent*.

Menurut (Grabowski, 2023) dalam artikel *"What Is DialogFlow and Why Do You Need It?" dialogflow* dianggap sebagai salah satu *platform* yang sangat efisien dan mudah diintegrasikan, karena mendukung berbagai fitur canggih yang memungkinkan interaksi alami antara pengguna dan *chatbot*.

4. NodeJS

NodeJs adalah sebuah runtime untuk bahasa pemrograman *javascript* agar dapat berjalan pada sisi server. *NodeJS* bergantung pada *V8 Engine* yaitu sebuah mesin yang dikembangkan oleh google agar dapat memahami bahasa *javascript* pada sisi *server* karena selama ini *javascript* hanya bisa dipahami oleh *browser* saja.

Menurut (Casciaro & Luciano, 2014) dalam buku *"Node.js Design Patterns"* dijelaskan bahwa dengan menggunakan *nodejs, javascript* dapat dijalankan pada *server side. Node.js* menggunakan sistem modular yang memungkinkan pengembang untuk memisahkan dan mengatur kode ke dalam modul-modul yang terpisah. Modul dapat diimpor atau ekspor untuk digunakan kembali untuk mempermudah pengembangan.

Package Manager bawaan *Node.js* adalah *Node Package Manager* (NPM) yang memungkinkan pengembang mengelola dependensi proyek dengan mudah. *Node Package Manager* (NPM) menyediakan akses ke ribuan *package* modul yang dapat digunakan.

NodeJS menggunanakan model *I/O (Input/Output) non-blocking* yang memungkinkan *server* untuk menangani banyak koneksi secara bersamaan tanpa harus menunggu operasi proses I/O selesai. Ini membuat aplikasi *Node.js* menjadi lebih efisien dalam penanganan banyak permintaan secara bersama

5. TypeScript

TypeScript adalah bahasa pemrograman *open source* yang dikembangkan oleh Microsoft sebagai *superset* dari *JavaScript.* Dengan menggunakan *TypeScript*, pengembang dapat menentukan jenis data variabel, parameter fungsi, dan nilai kembali fungsi. Ini sangat membantu untuk mendeteksi kesalahan selama proses kompilasi dan meningkatkan kejelasan kode. Selain itu *typeScript* mendukung fitur-fitur modern *javaScript* seperti *class, module,* dan *arrow functions* yang akan membantu dalam proses pengembang dan mengadopsi praktik pengembangan yang lebih canggih dan terstruktur.

TypeScript hanya diperlukan pada saat pengembangan *software* saja, pada akhirnya kode *typeScript* akan dikonversi ke *javaScript* saat siap untuk dijadikan produksi. Maka dari itu, *typeScript* memiliki *typeScript compiler*

(*tsc*), yang bertugas mengkonversi kode *typeScript* menjadi *javaScript*. Dengan begitu pengembang dapat menghasilkan kode yang dapat dijalankan di lingkungan *javaScript*.

Menurut (Stemmler, 2019) dalam artikel *"When to use TypeScript: a detailed guide through common scenarios"* menjelaskan berdasarkan pengalaman penulis dalam menggunakan bahasa pemrograman *typeScript* yang pada awalnya *sentimen* hingga pada akhirnya melakukan refactor dari aplikasi yang dibangun menggunakan *JavaScript* ke *TypeScript* untuk pengembangan jangka panjang yang lebih baik, karena *Static Typing* yang sangat membantu menutup celah.

6. NextJS

Nextjs adalah sebuah *framework web* yang digunakan untuk membangun aplikasi web pada sisi *frontend* menggunakan bahasa pemrograman *JavaScript* atau *TypeScript*. *Nextjs* dikembangkan oleh Vercel. *Nextjs* memiliki berbagai fitur yang berguna untuk pengembangan aplikasi web modern seperti *client-side rendering (CSR)*, *server-side rendering (SSR), static site generation (SSG), dynamic routing, pre-rendering* dan optimasi performa.

Server-Side Rendering (SSR) adalah cara untuk melakukan *rendering* sebuah halaman pada sisi *server* sebelum dikirimkan ke *browser*, dengan begini akan meningkatkan performa dan pengalaman pengguna karena komponen yang perlu disajikan ke *browser* sudah dilakukan *rendering* di *server*.

Client-Side Rendering (CSR) adalah bagaimana sebuah halaman atau komponen yang membutuhkan pembaruan cepat tanpa harus merender ulang halaman keseluruhan, memberikan responsifitas dan pengalaman interaktif.

Static Site Generation (SSG) adalah cara halaman di *generate* menjadi *static* html yang dapat dilakukan *caching* dan diakses dengan cepat. Ini Dilakukan pada tahap *build NextJS* projek ke produksi.

Menurut (Ossera, 2021) dalam artikel *"Next.js for CTOs. What is Next.js used for & why we love it"* dengan beberapa pilihan jenis *rendering* pengembang bisa menentukan halaman yang statis maupun *rendering* dari sisi *server* jika ada *concern* pada SEO untuk meningkatkan hasil mesin pencari atau bisa juga melakukan opsi *rendering* di sisi *browser* agar *website* menjadi lebih interaktif.

7. NestJS

NestJS adalah *framework server-side* yang dibangun di atas *Node*.js dan *TypeScript*. Dengan menggunakan konsep pemrograman berorientasi objek dan dekorator, *NestJS* menyediakan struktur modular yang kuat untuk membangun aplikasi *server side* yang *scalable*.

NestJS menggunakan struktur modular untuk mengorganisir kode, memungkinkan pembagian tugas dan tanggung jawab yang jelas di seluruh aplikasi. Menurut (Romik, 2024) dalam artikel *"Why Choose NestJS As Your Backend Framework*?", *NestJS* adalah *framework* yang *open source, extensible, versatile, progressive* untuk membuat *backend system* yang sesuai kebutuhan.

Dalam *NestJS* terdapat *Middleware* dan pipes digunakan untuk memproses permintaan *HTTP* di tingkat *middleware*, memungkinkan manipulasi *request* dan response sebelum mencapai *handler endpoint*. *NestJS* menggunakan konsep *provider* dan *dependency injection* untuk manajemen ketergantungan dan penyediaan layanan antar modul.

8. Docker

Docker adalah sebuah *platform* yang digunakan untuk menjalankan kode program atau aplikasi untuk dijalankan di *server* maupun lokal. Menurut (Khadeer, 2019) dalam artikel *"Learn Docker Beginner to Expert", Docker* adalah sebuah *Platform as a Service* yang digunakan pada *OS-level* virtualisasi untuk *mendeliver software*. *Docker* akan memisahkan aplikasi yang berjalan ke dalam sebuah tempat yang biasa disebut dengan *container*. *Docker Container* adalah unit perangkat lunak yang menyatukan aplikasi dan semua dependensinya, termasuk *library*, konfigurasi, dan *runtime*, dalam satu paket yang dapat dijalankan secara konsisten di berbagai *environment*.

Selain itu, pada *docker container juga* terdapat komponen penting yaitu *Docker Image* yang merupakan *template* yang berisi sistem operasi dan aplikasi yang sudah dikonfigurasi. Image ini digunakan untuk membuat kontainer. *Docker Image* terbuat dari *dockerfile* yang merupakan sebuah *file* teks yang berisi instruksi untuk membangun *docker image*. Instruksi ini mencakup langkah-langkah seperti menginstal perangkat lunak,

mengkonfigurasi lingkungan, dan menentukan bagaimana aplikasi harus berjalan.

9. PostgreSQL

PostgreSQL adalah sistem manajemen basis data relasional (RDBMS) *open-source* yang menggunakan Structured Query Language (SQL) sebagai bahasa utamanya untuk mengelola dan memanipulasi data. Sebagai RDBMS, PostgreSQL menyimpan data dalam bentuk tabel yang dapat saling berelasi, memungkinkan untuk representasi data yang terstruktur dan hubungan yang kompleks antara entitas yang berbeda.

PostgreSQL dikenal dengan kepatuhannya terhadap standar SQL dan fitur-fitur canggih yang mendukung integritas data, transaksi ACID (*Atomicity, Consistency, Isolation, Durability*), serta kemampuan untuk menangani beban kerja yang berat. Selain itu, PostgreSQL mendukung berbagai tipe data, termasuk tipe data khusus dan tipe data yang ditentukan oleh pengguna, yang menjadikannya fleksibel untuk berbagai aplikasi.

Menurut Momjian (2001), PostgreSQL dikembangkan dengan fokus pada ekstensibilitas dan kepatuhan terhadap standar yang memungkinkan-nya

untuk tetap relevan dalam berbagai perkembangan teknologi basis data. Dalam sebuah studi oleh Stonebraker et al., (2018) yang diterbitkan dalam Communications of the ACM, PostgreSQL diakui sebagai salah satu RDBMS paling stabil dan dapat diandalkan, yang ideal untuk aplikasi yang memerlukan integritas data tinggi dan performa yang konsisten.

10. Redis

Redis (Remote Dictionary Server) adalah sistem penyimpanan data struktur kunci-nilai (key-value) yang berjalan di memori (in-memory) dan bersifat open-source. Redis mendukung berbagai struktur data seperti string, hash, list, set, sorted set, bitmaps, hyperloglogs, dan geospatial indexes. Redis sering digunakan sebagai cache, database, dan broker pesan (message broker). Menurut Salvatore Sanfilippo, pencipta Redis, pada redis terdapat API yang sederhana penggunaanya dan dapat menangani tugas-tugas yang kompleks secara internal, Redis memungkinkan pengembang yang terfokus pada business logic daripada arsitektur penyimpanan data