BAB 2
LANDASAN TEORI

2.1 *Helpdesk*

Sistem *helpdesk* di perusahaan berfungsi sebagai pusat layanan yang menangani permasalahan yang dihadapi oleh pengguna, baik karyawan maupun pelanggan, terkait dengan teknologi informasi. *Helpdesk* di perusahaan bertujuan untuk meningkatkan kinerja tim IT dan menyediakan solusi yang cepat dan tepat bagi pengguna yang menghadapi masalah teknologi, baik perangkat keras maupun perangkat lunak. Sistem ini tidak hanya membantu dalam penyelesaian masalah tetapi juga berfungsi untuk mengoptimalkan operasional dan efisiensi di perusahaan dengan menyediakan saluran komunikasi yang terstruktur dan terorganisir[2].

Helpdesk merupakan sebuah titik pusat dari suatu permasalahan yang dilaporkan kemudian dikelola atau dikoordinasi oleh departemen yang bertanggung jawab. Jika dilihat dari perspektif yang lebih luas, *helpdesk* dapat juga disebut bagian inti dari suatu fungsi layanan yang bertanggung jawab untuk membawa beberapa sumber daya sehingga berguna untuk mengatasi sebuah masalah[5].

2.2 *Chatbot*

Chatbot adalah program komputer yang menirukan percakapan manusia dengan menggunakan dua metode dan algoritma kecerdasan buatan: *Natural Language Processing (NLP)* dan *Machine Learning*. *Chatbot* merupakan bentuk interaksi antara manusia dan komputer yang mengaplikasikan sistem kecerdasan buatan untuk meningkatkan pengalaman berinteraksi. Istilah lain yang sering digunakan untuk *chatbot* meliputi *artificial conversation entity, interactive agents, smart bots, digital assistant*, dan *conversational system*[6].

Terdapat dua jenis utama *chatbot* yang digunakan dalam berbagai aplikasi. Pertama, *chatbot* berbasis aturan (*rule-based chatbot*), yang bekerja dengan pola interaksi yang sudah ditentukan sebelumnya. *Chatbot* berbasis aturan diprogram dengan serangkaian respons yang sudah ditentukan sebelumnya yang akan bereaksi berdasarkan input pengguna tertentu. Jenis ini lebih sederhana dan cocok untuk skenario yang tidak memerlukan pemahaman mendalam[7]. Kedua, *chatbot* berbasis kecerdasan buatan *(AI-powered chatbot*), yang memanfaatkan teknologi pemrosesan bahasa alami (NLP) dan pembelajaran mesin (*machine learning*)

untuk memahami konteks dan memberikan respons yang lebih dinamis. *Chatbot* berbasis kecerdasan buatan menggunakan algoritma pembelajaran mesin yang canggih untuk meningkatkan kemampuannya dalam memahami dan memprediksi niat pengguna, sehingga lebih adaptif terhadap interaksi yang kompleks[8]. Dengan demikian, *chatbot*, baik yang berbasis aturan maupun kecerdasan buatan, memberikan solusi yang efektif untuk meningkatkan efisiensi komunikasi dan layanan dalam berbagai sektor.

2.3 *Preprocessing Query, Tokenization,* dan *Stopword Removal*

Dalam penelitian ini, menggunakan tiga cabang dari *Natural Languange Processing (NLP)*, yaitu *preprocessing query*, *tokenization*, dan *stopword removal*. Penggunaan cabang dari *Natural Languange Processing (NLP)* memiliki tujuan untuk melakukan pengelolaan teks diubah menjadi bentuk token. Karena input pertanyaan dan solusi masih dalam bentuk teks, hal ini membutuhkan proses dari *preprocessing query*, *tokenization*, dan *stopword removal* untuk data bisa dikelola menggunakan algoritma *Token-Based Similarity Algorithm*.

Dalam konteks perusahaan, digunakan untuk memproses masukan pengguna dalam bentuk teks, seperti laporan masalah dalam sistem *helpdesk*. Tahap pertama adalah *preprocessing query* yang bertujuan untuk mempersiapkan data teks agar lebih mudah diproses oleh sistem. Pada tahap ini, *query* pengguna melalui serangkaian proses pembersihan, seperti menghapus karakter yang tidak diperlukan, memformat teks agar seragam, dan mengonversi teks menjadi bentuk yang konsisten, seperti mengubah huruf kapital menjadi huruf kecil. Tahapan *preprocessing* yang tepat memungkinkan sistem untuk lebih akurat dalam memahami maksud pengguna serta mengurangi kebingungannya dalam menafsirkan input yang tidak terstruktur[9].

Selanjutnya, tahap kedua adalah *tokenization* memecah teks menjadi unit-unit kecil yang disebut token, yang bisa berupa kata, frasa, atau simbol. Tokenisasi memungkinkan sistem untuk menganalisis teks secara lebih granular, sehingga memungkinkan pencocokan yang lebih presisi antara masalah yang dilaporkan oleh pengguna dan solusi yang ada dalam *database*. Tokenisasi berfungsi untuk mengidentifikasi batas-batas antar token dalam teks. Pada bahasa seperti Inggris, batas token relatif mudah dikenali melalui spasi antar kata. Selain itu, ada beberapa manfaat utama dari proses *tokenization*. Pertama, tokenisasi dapat meningkatkan konsistensi data dengan memastikan bahwa teks yang diolah

memiliki format yang seragam, sehingga algoritma dapat menginterpretasikan data secara lebih efektif. Kedua, tokenisasi meningkatkan efisiensi pemrosesan dengan memecah teks menjadi elemen-elemen yang relevan, yang pada gilirannya mengurangi beban komputasi. Ketiga, tokenisasi mendukung akurasi pencocokan dengan memungkinkan sistem untuk mengidentifikasi elemen-elemen kunci dalam teks[10].

Tahap terakhir adalah *stopword removal* dapat meningkatkan efisiensi dan akurasi pemrosesan teks. *Stopword removal* dapat membantu mengurangi kebisingan dalam teks, sehingga sistem dapat lebih fokus pada kata-kata kunci yang relevan dengan tujuan analisis, seperti dalam kasus pemrograman atau dokumentasi teknis. Kata-kata seperti kata penghubung dan kata ganti sering kali dianggap tidak relevan untuk analisis teks karena tidak memberikan kontribusi signifikan terhadap pemahaman konteks. Misalnya, kata-kata seperti *"and", "or", "but"* (kata penghubung), dan *"it", "this", "that"* (kata ganti) termasuk dalam kategori *stopwords*. Penghapusan kata-kata tersebut memungkinkan sistem untuk lebih fokus pada kata-kata yang lebih bermakna dan relevan dengan konteks yang lebih spesifik. Teknik ini sangat penting untuk aplikasi seperti pencarian informasi, analisis sentimen, dan pengklasifikasian teks. Penelitian ini juga menyoroti pentingnya menggunakan daftar *stopword* yang disesuaikan dengan domain tertentu, mengingat penggunaan kata umum dapat bervariasi antar domain. Oleh karena itu, disarankan untuk menyesuaikan dan mengoptimalkan daftar *stopword* agar sesuai dengan kebutuhan spesifik aplikasi. Secara keseluruhan, *stopword removal* yang dilakukan dengan bijaksana dapat meningkatkan kinerja sistem, baik dalam hal waktu pemrosesan maupun kualitas hasil analisis[11].

2.4 *Token-Based Similarity Algorithm*

Token-Based Similarity Algorithm adalah teknik yang digunakan untuk mengukur tingkat kesamaan antara dua teks dengan membandingkan token atau kata-kata yang terkandung dalam masing-masing teks. Dalam proses ini, teks pertama dan kedua diubah menjadi token (unit terkecil yang dapat diinterpretasikan) menggunakan metode tokenisasi. Setelah itu, algoritma ini mengukur kesamaan antara dua teks tersebut dengan membandingkan token yang ada pada masing-masing teks. Salah satu pendekatan yang umum digunakan untuk menghitung *token-based similarity* adalah menggunakan *Jaccard similarity* dan *Cosine similarity*. *Jaccard similarity* mengukur kesamaan antara dua set token

dengan membandingkan jumlah token yang sama dengan jumlah total token unik yang ada dalam kedua teks, sementara *Cosine similarity* mengukur kesamaan dengan menghitung sudut antara dua vektor representasi token dalam ruang vektor dimensi tinggi. Pendekatan ini sering diterapkan dalam berbagai bidang, seperti aplikasi sistem pencocokan solusi dalam *chatbot helpdesk* atau aplikasi lain yang membutuhkan pencocokan teks berbasis kata kunci[12].

Persamaan 2.1 merupakan persamaan dari logika *Token-Based Similarty Algorithm* yang menggunakan perbandingan antara jumlah token cocok dengan total token *input*. Logika yang digunakan dalam sistem *chatbot helpdesk* mencari *similarity* dari *list issue* yang ada di *csv* yang paling mendekati dengan token *input user*. Sistem akan melakukan perhitungan ketika *input user* sudah menjadi token, sistem akan menghitung jumlah token yang sama pada *issue* yang sesuai dengan *category* dan *subcategory input user*.

$$TKSA = \frac{JumlahTokenCocok}{TotalTokenInput} \times 100 \quad (2.1)$$

2.5 *Framework Next.js*

Next.js adalah sebuah *framework* berbasis *React* yang memungkinkan pengembang untuk membangun aplikasi web yang cepat dengan fitur-fitur seperti *server-side rendering (SSR), static site generation (SSG)*, dan *API routes*. Salah satu keunggulan utama dari *Next.js* adalah kemampuannya untuk mempermudah pengembangan aplikasi web dengan pengaturan yang minimal, memungkinkan pengembang untuk fokus pada pengembangan fitur tanpa harus khawatir tentang konfigurasi yang rumit. Berdasarkan dokumentasi resmi *Next.js, framework* ini memungkinkan *rendering* sisi server yang meningkatkan performa dan *SEO (Search Engine Optimization)* dengan menyediakan halaman web yang sudah ter-*render* ketika diterima oleh browser pengguna[13]. *Next.js* juga menawarkan solusi otomatis untuk pengoptimalan kode, seperti pembagian kode dinamis (code splitting) dan pemuatan sumber daya secara efisien.

Dalam konteks perusahaan, *Next.js* telah banyak digunakan untuk membangun aplikasi berbasis web yang membutuhkan performa tinggi dan pengalaman pengguna yang lancar. Penggunaan *Next.js* dalam pengembangan aplikasi *fitness* dapat meningkatkan waktu muat halaman, sehingga meningkatkan pengalaman pengguna dan kepuasan pelanggan. Selain itu, *framework* ini mendukung pengembangan aplikasi dengan fitur-fitur seperti penerapan metode

caching, yang memungkinkan pengembang untuk mengoptimalkan pengambilan data secara efisien dan aman, sebuah hal yang sangat relevan dalam konteks perusahaan yang bergantung pada pengambilan data untuk mendukung keputusan bisnis dan meningkatkan kinerja aplikasi[14]. Secara keseluruhan, *Next.js* menjadi pilihan yang kuat untuk perusahaan yang ingin membangun aplikasi web modern dengan kebutuhan kinerja yang tinggi dan pengelolaan sumber daya yang efisien. Penggunaannya dalam pengembangan aplikasi di perusahaan dapat mempercepat proses pengembangan sambil memastikan aplikasi tetap optimal dan mudah dipelihara.Penggunaannya dalam pengembangan aplikasi di perusahaan dapat mempercepat proses pengembangan sambil memastikan aplikasi tetap optimal dan mudah dipelihara[13]

2.6 *Testing*

Dalam penelitian ini menggunakan dua *testing*, yaitu menggunakan proporsi 70:30 untuk melakukan testing dengan cara input secara manual dan melakukan pembuatan *form Black box Testing* dan *White Box Testing*.

2.6.1 Proporsi 70:30

Pemilihan rasio 70:30 untuk pembagian data pelatihan dan pengujian dalam penelitian ini merujuk pada rekomendasi dari studi sebelumnya. Performa suatu model pembelajaran dapat bervariasi tergantung pada rasio pembagian data yang digunakan. Berdasarkan analisis statistik yang dilakukan dalam studi tersebut, rasio 70:30 terbukti optimal untuk melatih dan menguji model secara seimbang, karena data pelatihan yang cukup besar dapat memberikan informasi yang cukup untuk membangun model yang baik, sementara data pengujian yang cukup banyak dapat mengukur performa model secara efektif[15].

pembagian data dengan rasio 70:30 antara data pelatihan dan data pengujian memberikan keseimbangan yang optimal dalam proses pelatihan dan evaluasi model. Dalam studi tersebut, pembagian data yang lebih besar untuk pelatihan tidak menyebabkan model terlalu menyesuaikan diri dengan data pelatihan, sementara data pengujian yang cukup banyak memungkinkan pengujian model secara lebih efektif. Oleh karena itu, rasio ini dianggap ideal karena menyediakan data pelatihan yang cukup untuk akurasi yang baik tanpa mengorbankan kualitas pengujian model, yang sangat penting dalam evaluasi performa model[16].

2.6.2 *Black box Testing dan White Box Testing*

Black Box Testing dan *White Box Testing* merupakan dua metode utama dalam pengujian perangkat lunak yang memiliki perbedaan signifikan dalam pendekatan dan fokus ujiannya. *Black Box Testing* berfokus pada pengujian fungsionalitas sistem tanpa mempertimbangkan struktur internal atau logika pemrograman yang ada di balik aplikasi. Pengujian ini dilakukan dengan memverifikasi apakah sistem memenuhi persyaratan yang telah ditetapkan berdasarkan input yang diberikan dan *output* yang dihasilkan. *Black Box Testing* sering digunakan pada tahap pengujian penerimaan pengguna dan pengujian fungsionalitas, di mana tujuan utamanya adalah untuk memastikan bahwa sistem bekerja sesuai dengan yang diinginkan oleh pengguna tanpa memerlukan pemahaman terhadap kode sumbernya. Pengujian ini memungkinkan penguji untuk mendeteksi cacat dalam fungsionalitas aplikasi, antarmuka pengguna, dan kompatibilitas sistem[17].

Sebaliknya, *White Box Testing* melibatkan pengujian internal perangkat lunak dengan menganalisis struktur dan kode sumber aplikasi. Pendekatan ini menekankan pada verifikasi logika dan alur eksekusi program, termasuk pengujian jalur kode dan pengujian unit untuk memastikan bahwa setiap bagian kode berfungsi sesuai dengan spesifikasi.*White Box Testing* memberikan penguji wawasan lebih mendalam tentang bagaimana sistem beroperasi di dalam, memungkinkan identifikasi masalah potensial pada tingkat struktur kode, serta meningkatkan keamanan dan efisiensi aplikasi. *White Box Testing* umumnya digunakan pada tahap pengujian unit dan integrasi untuk memastikan bahwa setiap fungsi dalam aplikasi berjalan dengan benar dan aman[17].