BAB II

LANDASAN TEORI

2.1 Kajian Penelitian Terdahulu

Berikut ini adalah beberapa referensi dari penelitian-penelitian sebelumnya yang memiliki topik tentang *chatbot*. Referensi-referensi ini akan digunakan untuk mendukung penelitian yang sedang dilakukan.

2.1.1 Penelitian Gerald Santoso, Johan Setiawan, dan Agus Sulaiman (2023)

Berikut adalah penelitian yang dilakukan Gerald Santoso, Johan Setiawan, dan Agus Sulaiman pada tahun 2023 dengan judul "*Development of OpenAI API-Based Chatbot to Improve User Interaction on the JBMS Website*".

Tabel 2.1 Penelitian Gerald Santoso, Johan Setiawan, dan Agus Sulaiman

Judul penelitian	
Development of OpenAI API-Based Chatbot to Improve User Interaction on the JBMS Website	
Hasil Penelitian	**Ruang Lingkup**
Chatbot yang dapat membantu *user* dari *website JBMS* dengan menjawab pertanyaan dan	- Chatbot yang dihasilkan akan membantu *user JBMS* dalam menemukan panduan

memberikan panduan sesuai dengan kebutuhan user.	penggunaan *website JBMS*. - Chatbot hanya bisa diakses dan digunakan di *website JBMS.*
Tujuan	
Menyediakan layanan *chatbot* yang dirancang untuk membantu user dari *Website JBMS (Journal of Business, Management, and Social Studies)* untuk memperoleh layanan yang mereka butuhkan, misalnya mencari artikel jurnal yang relevan, mendapatkan panduan prosedur untuk menjadi author, serta memperoleh panduan tata cara mengunggah artikel ke *website*.	
Perbedaan dengan penelitian yang dilakukan	
- Pada penelitian ini *chatbot* dapat diakses dan digunakan di *website JBMS*. pada penelitian yang dilakukan aplikasi dapat diakses dan diinstal pada perangkat *Android* dan *iOS*. - Pada penelitian ini *chatbot* hanya menerima *input* berupa teks. Pada penelitian yang dilakukan chatbot dapat menerima *input* berupa teks dan gambar.	

2.1.2 Penelitian Muhammad Alifyan Zulkarnain, Muhammad Fajri Raharjo dan Meylanie Olivya (2020)

Berikut ini adalah penelitian yang dilakukan oleh Muhammad Alifyan Zulkarnain, Muhammad Fajri Raharjo dan

Meylanie Olivya pada tahun 2020 dengan judul "Perancangan Aplikasi *Chatbot* Sebagai Media *E-Learning* Bagi Siswa".

Tabel 2.2 Penelitian Muhammad Alifyan Zulkarnain, Muhammad Fajri Raharjo dan Meylanie Olivya

<table>
<tr><td colspan="2">Judul penelitian</td></tr>
<tr><td colspan="2">Perancangan Aplikasi Chatbot Sebagai Media E-Learning Bagi Siswa</td></tr>
<tr><td>Hasil Penelitian</td><td>Ruang Lingkup</td></tr>
<tr><td>Bot telegram yang dapat menjawab pertanyaan yang diajukan oleh siswa berdasarkan materi yang Sudah ditambahkan ke sistem oleh guru.</td><td>- Chatbot menjawab pertanyaan berdasarkan materi yang sudah didaftarkan oleh guru.
- Chatbot yang dihasilkan hanya dapat diakses melalui telegram.</td></tr>
<tr><td colspan="2">Tujuan</td></tr>
<tr><td colspan="2">Menyediakan layanan chatbot yang mendukung proses belajar-mengajar antara guru dan siswa dengan menjawab pertanyaan-pertanyaan yang diajukan oleh siswa. Chatbot ini tersedia selama 24 jam dan dapat diakses melalui aplikasi Telegram.</td></tr>
<tr><td colspan="2">Perbedaan dengan penelitian yang dilakukan</td></tr>
<tr><td colspan="2">- Pada penelitian ini chatbot hanya dapat diakses melalui aplikasi telegram. Pada penelitian yang akan dilakukan aplikasi merupakan</td></tr>
</table>

aplikasi yang dapat diinstal di *Android* dan *iOS*.

- Pada penelitian ini *chatbot* hanya dapat menjawab pertanyaan berdasarkan materi yang sudah ditambahkan oleh guru. Jika materi belum terdaftar maka akan menampilkan pesan *error*. Pada penelitian yang akan dilakukan, *chatbot* dapat menjawab pertanyaan apapun yang bertopik pemrograman.

2.1.3 Penelitian Anish Balaji Murshetwar, Yash Sunil Amane, Aryan Shirish Karanjkar, dan Shubham Ananda Bhilare (2024)

Berikut ini adalah penelitian yang dilakukan oleh Anish Balaji Murshetwar, Yash Sunil Amane, Aryan Shirish Karanjkar, dan Shubham Ananda Bhilare pada tahun 2024 yang berjudul "*Integration of OpenAI API for Multilingual Voice Chatbot Systems*".

Tabel 2.3 Penelitian Anish Balaji Murshetwar, Yash Sunil Amane, Aryan Shirish Karanjkar, dan Shubham Ananda Bhilare

Judul penelitian	
Integration of OpenAI API for Multilingual Voice Chatbot Systems.	
Hasil Penelitian	**Ruang Lingkup**
Chatbot yang dapat menerima *input* berupa suara dan dapat memberikan respon yang tepat	- *Chatbot* yang dihasilkan berupa aplikasi berbasis *website*.

terhadap *input* yang diberikan.	- *Input* yang diterima hanya *input* suara.
Tujuan	
Mengembangkan *chatbot* yang mampu menerima *input* suara dan memberikan respon yang sesuai serta kontekstual berdasarkan *input* tersebut.	
Perbedaan dengan penelitian yang dilakukan	
- Pada penelitian ini *chatbot* hanya bisa menjawab pertanyaan user tetapi tidak ada percakapannya. Sedangkan pada penelitian yang akan dilakukan, user dapat melakukan percakapan dan diskusi dengan *chatbot*. - Pada penelitian ini *chatbot* menerima *input* berupa suara. Pada penelitian yang akan dilakukan *chatbot* menerima *input* berupa teks dan gambar.	

2.1.4 Penelitian Guntoro, Loneli Costaner, dan Lisnawita (2020)

Berikut ini adalah penelitian yang dilakukan oleh Guntoro, Loneli Costaner, dan Listnawita pada tahun 2020 yang berjudul "Aplikasi *chatbot* untuk layanan Informasi dan Akademik Kampus Berbasis *Artificial Intelligence Markup Language* (AIML)".

Tabel 2.4 Penelitian Guntoro, Loneli Costaner, dan Lisnawita

<table>
<tr><td colspan="2">Judul penelitian</td></tr>
<tr><td colspan="2">Aplikasi chatbot untuk layanan Informasi dan Akademik Kampus Berbasis Artificial Intelligence Markup Language (AIML)</td></tr>
<tr><td>Hasil Penelitian</td><td>Ruang Lingkup</td></tr>
<tr><td>Chatbot yang membantu memberikan informasi seputar pendaftaran di Universitas Lancang Kuning</td><td>- Chatbot hanya dapat memberikan informasi terkait, alamat kampus, syarat pendaftaran, langkah pendaftaran, program studi, jalur kuliah, berapa biaya kuliah, dan cara mendaftar.
- Sistem yang dihasilkan berbasis website.</td></tr>
<tr><td colspan="2">Tujuan</td></tr>
<tr><td colspan="2">Membangun sebuah aplikasi chatbot yang memiliki fungsi sebagai wahana informasi kampus dan akademik, yang dapat dimanfaatkan oleh khalayak umum maupun oleh civitas akademika Universitas Lancang Kuning.</td></tr>
<tr><td colspan="2">Perbedaan dengan penelitian yang dilakukan</td></tr>
<tr><td colspan="2">- Pada penelitian ini chatbot yang dihasilkan dapat diakses menggunakan website. Pada penelitian yang akan dilakukan,</td></tr>
</table>

aplikasi tersedia untuk platform *Android* dan *iOS*.

- Pada penelitian ini chatbot hanya terbatas menjawab pertanyaan seputar pendaftaran di Universitas Lancang Kuning. Pada penelitian yang akan dilakukan *chatbot* dapat menjawab berbagai pertanyaan yang bertopik pemrograman.

2.1.5 Penelitian Mikko Lempinen, Emilia Pyyny, dan Arttu Juntunen (2023)

Berikut ini adalah penelitian yang dilakukan oleh Mikko Lempinen, Emilia Pyyny, dan Arttu Juntunen pada tahun 2023 yang berjudul "*Chatbot for Assessing System Security With OpenAI GPT-3.5*".

Tabel 2.5 Penelitian Mikko Lempinen, Emilia Pyyny, dan Arttu Juntunen

Judul penelitian	
Chatbot for Assessing System Security With OpenAI GPT-3.5	
Hasil Penelitian	**Ruang Lingkup**
Chatbot yang dapat membantu user seputar *cyber security*.	- *Chatbot* yang dihasilkan berupa aplikasi *website*. - *Chatbot* hanya membantu *user* dalam hal *cyber security.*
Tujuan	

Mengembangkan *chatbot* yang dapat membantu *user* dalam menjaga keamanan sistem mereka dengan cara menganalisa *log* dari sistem, melakukan blokir alamat IP, serta melakukan *restart server*.
Perbedaan dengan penelitian yang dilakukan
- Pada penelitian ini *chatbot* hanya membantu *user* dalam hal keamanan sistem. Pada penelitian yang akan dilakukan *chatbot* dapat membantu *user* dalam hal pemrograman. - Pada penelitian ini *chatbot* dapat diakses melalui *website*. Pada penelitian yang akan dilakukan *chatbot* akan berupa aplikasi *Android* dan *iOS*. - Pada penelitian ini *chatbot* hanya menerima *input* berupa text. Pada penelitian yang akan dilakukan *chatbot* dapat menerima *input* teks dan gambar.

2.2 Dasar Teori

Berikut ini adalah beberapa teori-teori terkait yang diambil dari beberapa referensi yang membahas kecerdasan buatan dan teori seputar pengembangan sistem.

2.2.1 Kecerdasan Buatan

Menurut Kristanto (2003), konsep kecerdasan buatan diartikan sebagai sebuah cabang ilmu komputer yang berfokus secara khusus pada pengembangan sistem yang didesain untuk memiliki kemampuan bertindak secara inteligensi dan otonom. Fondasi utama dari kecerdasan

buatan terletak pada landasan pengetahuan, yaitu pemahaman yang mendalam mengenai domain spesifik yang dikumpulkan melalui proses pembelajaran dan akumulasi pengalaman.

Dalam praktiknya, kecerdasan buatan memanfaatkan perangkat lunak dan perangkat keras untuk mereplikasi cara kerja manusia. Aktivitas-aktivitas yang dijadikan contoh mencakup proses berpikir logis (penalaran), kapasitas visual (penglihatan), pembelajaran, pemecahan masalah, dan pemahaman bahasa manusia (bahasa alami). Teknologi kecerdasan buatan sendiri merangkum beragam area, semisal robotika, visi komputer, pemrosesan bahasa alami, identifikasi pola, jejaring saraf tiruan, identifikasi suara, dan sistem berbasis pakar (Simarmata, 2006).

2.2.2 *Machine Learning*

Menurut Batta (2020), *machine learning* atau pembelajaran mesin didefinisikan sebagai implementasi kecerdasan buatan yang memungkinkan sistem untuk belajar secara otomatis dari data dan melaksanakan tugas tertentu tanpa memerlukan instruksi pemrograman yang eksplisit. Rebala et al. (2019) menjelaskan bahwa *machine learning* merupakan ranah dalam ilmu komputer yang berkonsentrasi pada pengembangan algoritma dan metode untuk menangani problematika kompleks secara otomatis, yang sulit dipecahkan menggunakan metode pemrograman konvensional. Dalam prosesnya, algoritma *machine learning* mengidentifikasi pola dalam data atau dataset untuk menciptakan aturan. Yu dan He (2019) mengklasifikasikan metode dalam *machine learning*

menjadi empat jenis berdasarkan pendekatan pembelajarannya, yaitu:

a. *Supervised Learning*, metode ini menggunakan data masukan dan keluaran dari dataset untuk memetakan dan membuat prediksi atau klasifikasi. Metode ini belajar dari dataset latih dan menerapkan pola yang dipelajari pada dataset uji.
b. *Unsupervised Learning*, metode ini digunakan untuk mengelompokkan data yang tidak dilabeli. *Unsupervised learning* bekerja dengan mengeksplorasi data untuk menemukan struktur dan mengidentifikasi kelas data, menggunakan teknik seperti estimasi kepadatan dan reduksi dimensi, yang penting dalam *computer vision*.
c. *Semi-supervised Learning*, metode ini merupakan gabungan dari *supervised* dan *unsupervised learning*, menggunakan data berlabel dan tidak berlabel. Ini berguna untuk situasi di mana data berlabel terbatas dan data tidak berlabel lebih banyak.
d. *Reinforcement Learning*, Metode ini berkaitan dengan pengambilan keputusan berurutan untuk mencapai hasil terbaik. Reinforcement learning belajar dari struktur data dan mengadaptasi perilaku baru melalui pengalaman trial and error untuk mendapatkan reward.

2.2.3 *Neural Network*

Neural Network merupakan bagian dari *Machine Learning* yang dirancang untuk meniru kemampuan pemrosesan otak manusia. *Neural network* bekerja dengan cara mengalirkan data melalui lapisan-lapisan dari neuron buatan. Dalam proses ini, setiap *neuron* menerima *input*,

memprosesnya melalui fungsi aktivasi, dan mengirimkan *output* ke lapisan berikutnya. Tujuan utamanya adalah untuk menghasilkan prediksi atau keputusan berdasarkan *input* yang diberikan, mirip dengan cara otak manusia memproses informasi. Dalam *neural network* terdapat berbagai komponen penting. Komponen-komponen yang umumnya terdapat pada setiap jaringan saraf, yaitu:

a. *Input*, merupakan data yang dimasukkan ke dalam model guna keperluan pembelajaran dan pelatihan.
b. *Weight*, fungsi dari *weight* adalah untuk mengatur variabel berdasarkan tingkat kepentingan dan pengaruhnya terhadap kontribusi.
c. *Transfer function* (fungsi transfer), pada fungsi transfer, semua input akan dikompilasi dan diintegrasikan menjadi satu variabel *output*.
d. *Activation function* (fungsi aktivasi), fungsi aktivasi bertugas menentukan apakah neuron tertentu perlu diaktifkan, berdasarkan pentingnya *input* neuron tersebut dalam proses membuat prediksi.
e. *Bias*, berfungsi untuk mengubah nilai yang dihasilkan oleh fungsi aktivasi.

Saat ini ada beberapa jenis dari *Neural Network*:

a. Recurrent Neural Networks (RNNs)

RNN menggunakan prediksi yang telah dipelajari sebelumnya untuk membantu membuat prediksi dengan akurasi tinggi.

b. Long Short Term Memory Network (LSTM)

LSTM menambahkan struktur tambahan, atau *gates,* ke RNN untuk meningkatkan kemampuan memori.

c. *Convolutional Neural Networks (CNNs)*

CNN merupakan jenis *feed-forward network* yang digunakan untuk analisis gambar dan pemrosesan bahasa. Terdapat lapisan konvolusional tersembunyi yang membantu CNN dan mendeteksi pola. CNN menggunakan fitur seperti tepi, bentuk, dan tekstur untuk mendeteksi pola.

d. *Transformer Neural Networks (TNNs)*

TNN menggunakan mekanisme *"attention"* untuk memproses urutan data secara paralel, tanpa bergantung pada struktur berulang seperti RNN. Model ini efisien dalam menangani tugas-tugas yang memerlukan pemahaman kontekstual dari data masukan, seperti *Natural Language Processing (NLP).*

2.2.4 *Natural Language Processing* (NLP)

Natural Language Processing (NLP) adalah penerapan dari ranah ilmu komputer, khususnya dalam area linguistik komputasional, yang didesain untuk mengkaji bagaimana komputer berelasi dengan bahasa manusia. Fokus utama NLP adalah untuk menanggulangi kerumitan dalam pemahaman bahasa manusia, termasuk gramatika dan semantiknya, dan mengkonversikannya menjadi format formal yang mampu diproses oleh komputer (James Putstejovsky, 2012).

Sebelum sebuah model dapat dirancang, beberapa tujuan pembuatan model harus didefinisikan, misalnya tujuan yang ingin dicapai dengan pembuatan model, jenis corpus atau dataset yang dibutuhkan, dan

apa yang diharapkan dari model tersebut. Pengumpulan dataset memerlukan beberapa teknik pra-pemrosesan teks agar dataset yang awalnya tidak terstruktur dapat menjadi terstruktur. Ada beberapa langkah dalam pra-pemrosesan, yaitu:

1. *Tokenization*

   Proses memecah *string* menjadi unit-unit teks independen yang memiliki makna. Ini melibatkan pembagian teks menjadi kalimat atau kata-kata (Sanger & Feldman, 2007).

2. *Part-of-Speech Tagging*

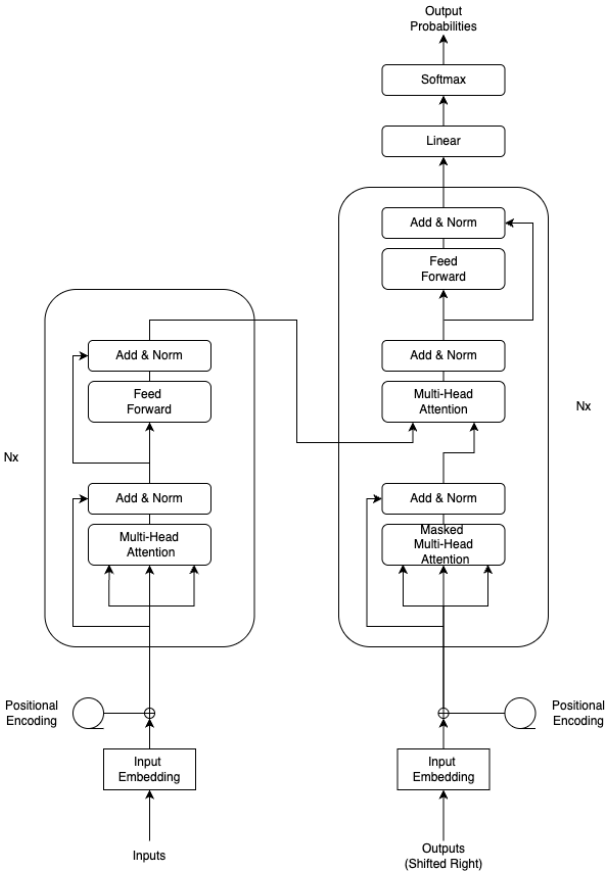
   Proses menandai kata atau kalimat dengan tag POS (*Part of Speech*).Tag ini dapat berupa kata kerja, kata benda, kata sifat, atau aturan tata bahasa lainnya (Sanger & Feldman, 2007).

3. *Syntactical Parsing*

   Proses menganalisis sintaksis suatu kalimat berdasarkan teori tata bahasa tertentu. Umumnya, analisis ini dibagi menjadi dua kategori: dependensi dan konstituensi (Sanger & Feldman, 2007).

2.2.5 *Transformers Neural Networks*

Vaswani et al. (2017) mengembangkan sebuah arsitektur yang dikenal sebagai *transformer*. Arsitektur ini secara signifikan meningkatkan kinerja *state-of-the-art* dalam *Neural Machine Translation (NMT)* tanpa memerlukan lapisan rekurensi. Berbeda dengan RNN, di mana kata-kata dalam sebuah kalimat diproses secara berurutan, transformer memproses semua kata dalam kalimat secara bersamaan dan paralel.



Gambar 2.1 Arsitektur transformer

(Vaswani et al., 2017)

Arsitektur Transformer tersusun atas dua komponen utama, yaitu *encoder* (di sisi kiri pada Gambar 2.2) dan *decoder* (di sisi kanan pada Gambar 2.2). *Encoder* dibangun dari dua sub-lapisan. Sub-lapisan pertama adalah *multi-head attention*, yang memberikan kemampuan kepada *encoder* untuk memusatkan perhatian pada terminologi spesifik sekaligus memahami konteks keseluruhan dari *input*. Sub-lapisan kedua adalah

jejaring *feedforward komprehensif* yang menghasilkan *vektor output*, yang kemudian dialirkan ke *decoder*. *Transformer* berperan sebagai kerangka dasar bagi model-model unggulan semacam *BERT* (*Bidirectional Encoder Representation from Transformers*), GPT (*Generative Pre-trained Transformer*), dan lain sebagainya, yang telah menetapkan tolok ukur baru dalam pelbagai tugas NLP.

2.2.6 GPT

Menurut dokumentasi dari OpenAI, *GPT (Generative Pre-trained Transformer)* adalah sebuah model pembelajaran mesin yang menggunakan teknologi deep learning, khususnya arsitektur transformer, untuk menghasilkan teks yang mirip dengan teks yang ditulis manusia. Model ini dilatih dengan dataset besar yang berisi berbagai jenis teks untuk memahami dan menghasilkan bahasa secara alami. Tujuan utama dari GPT adalah untuk memfasilitasi interaksi yang lebih efektif antara manusia dan komputer, serta untuk mendukung berbagai aplikasi, seperti penerjemahan bahasa, perangkat lunak asisten, dan lain-lain.

Hingga saat ini, OpenAI telah mengembangkan beberapa versi dari GPT. Model pertama adalah GPT, dilanjutkan dengan GPT-2 yang diperkenalkan pada tahun 2019 dengan peningkatan kapasitas dan kemampuan yang lebih baik. Model ini terkenal akan kemampuannya untuk menghasilkan teks yang koheren berdasarkan prompt yang diberikan. Kemudian, di tahun 2020, OpenAI meluncurkan GPT-3, GPT-3 memiliki 175 miliar parameter, membuatnya sangat poten dalam menghasilkan teks yang sangat mirip dengan tulisan manusia, dan memiliki kemampuan yang lebih luas dalam aplikasi praktis dibandingkan dengan

model sebelumnya. Lalu pada Maret 2023 OpenAI meluncurkan GPT-4. Model ini merupakan penerus GPT-3 dan memiliki peningkatan dalam hal ukuran, kompleksitas, dan kemampuan pengertian serta generasi teks.

2.2.7 GPT-4-Turbo

Menurut dokumentasi dari OpenAI, GPT-4-Turbo merupakan iterasi terbaru dari model kecerdasan buatan yang dirancang oleh OpenAI, yang merupakan evolusi dari seri GPT sebelumnya. Perbaikan yang signifikan pada model ini mencakup peningkatan efisiensi dan kecepatan dalam menghasilkan teks, sambil mempertahankan atau bahkan meningkatkan kemampuan pemahaman dan produksi teks yang relevan serta kontekstual. Model ini menggunakan teknologi terbaru dalam pemrosesan bahasa alami untuk menyediakan respons yang lebih akurat dan natural, mendukung pemakaian dalam berbagai aplikasi mulai dari pembantu virtual hingga pengembangan konten otomatis. GPT-4-Turbo dirancang untuk lebih responsif dan dapat diandalkan, membuka peluang baru dalam interaksi manusia dan mesin yang lebih efektif dan personal.

2.2.8 *OpenAI*

OpenAI adalah sebuah organisasi penelitian dan pengembangan *Artificial Intelligence (AI)* yang didirikan pada tahun 2015 dengan misi untuk memastikan bahwa kecerdasan buatan umum (*Artificial General Intelligence - AGI*) dapat memberikan manfaat bagi seluruh umat manusia. Awalnya didirikan sebagai organisasi *non-profit*, *OpenAI* telah berkembang menjadi entitas hibrida dengan sayap komersial untuk mendanai penelitian

intensifnya dalam skala besar (Brockman et al., 2016). Organisasi ini berfokus pada penelitian jangka panjang dan pengembangan *platform AI* yang aman dan kuat. *OpenAI* telah menjadi pelopor dalam pengembangan *Large Language Models - LLMs*, terutama melalui seri model *Generative Pre-trained Transformer* (GPT). Model-model ini dilatih pada korpus data teks dan kode yang sangat besar, memungkinkan mereka untuk memahami, menghasilkan, dan memanipulasi bahasa manusia dengan tingkat kefasihan yang belum pernah terjadi sebelumnya (Brown et al., 2020).

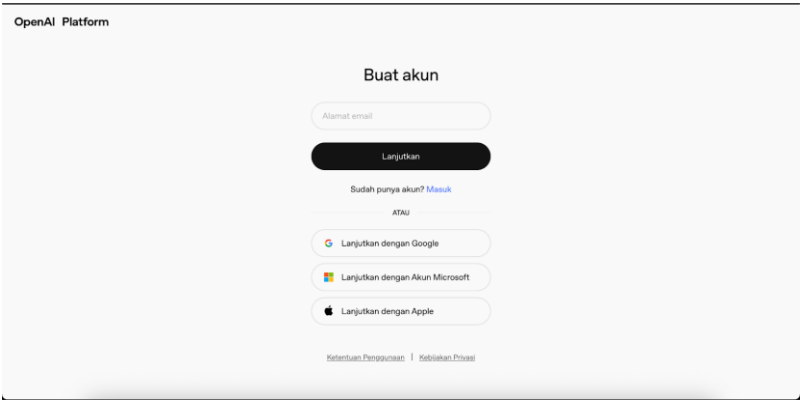
Kemampuan model seperti *GPT-3* dan *GPT-4* untuk melakukan penalaran, menjawab pertanyaan, meringkas teks, dan bahkan menulis kode telah membuka peluang baru dalam berbagai bidang, termasuk pengembangan perangkat lunak, pendidikan, dan otomatisasi proses bisnis (Floridi & Chiriatti, 2020). Dengan menyediakan akses ke model-model canggih ini melalui API, *OpenAI* memungkinkan pengembang untuk mengintegrasikan kemampuan *AI* tingkat lanjut ke dalam aplikasi mereka sendiri tanpa harus membangun dan melatih model tersebut dari awal, sebuah proses yang memerlukan sumber daya komputasi dan keahlian yang sangat besar (Eloundou et al., 2023).

Untuk menggunakan API dari OpenAI, seorang pengembang perlu mengikuti beberapa langkah untuk mendapatkan akses dan mengelola autentikasi.

1. Pendaftaran Akun

   Langkah pertama adalah membuat akun di *platform OpenAI*. Pengguna harus mengunjungi situs web resmi OpenAI (*platform.openai.com*) dan mendaftar

menggunakan email atau metode autentikasi lain yang disediakan. Proses ini biasanya melibatkan verifikasi email untuk mengaktifkan akun.



Gambar 2.2 Halaman register website OpenAI

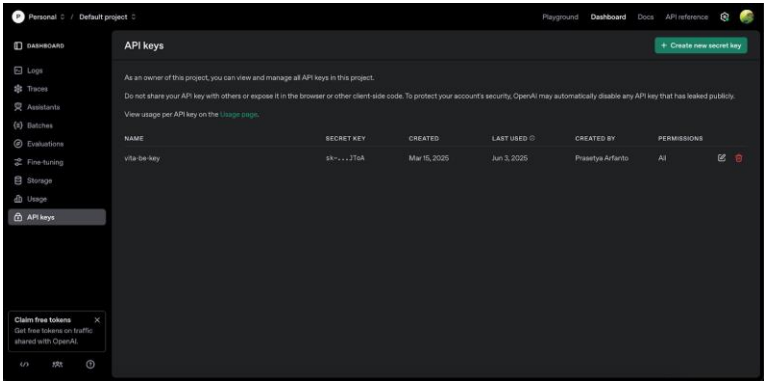
2. Mendapatkan *API Key*

   Setelah akun berhasil dibuat dan diaktifkan, *API Key* dapat diperoleh. *API Key* adalah sebuah token rahasia yang unik yang berfungsi sebagai kredensial untuk mengautentikasi permintaan ke API OpenAI. Kunci ini harus dijaga kerahasiaannya dan tidak boleh diekspos di kode sisi klien atau repositori publik (OpenAI, 2024).

   Cara mendapatkan kunci API:

   a. Masuk ke akun OpenAI Anda.
   b. Navigasi ke bagian "API Keys" pada dasbor akun Anda.
   c. Klik tombol "Create new secret key".

d. Sebuah kunci baru (misalnya, sk-xxxxxxxxxxxxxxxxxxxxxxxx) akan dibuat. Kunci ini hanya akan ditampilkan satu kali. Sangat penting untuk menyalin dan menyimpannya di tempat yang aman, seperti environment variable atau layanan manajemen rahasia.



Gambar 2.3 Halaman dashboard untuk mendapatkan *API keys*

Saat melakukan panggilan ke *API Chat Completions*, terdapat beberapa parameter yang dapat disesuaikan untuk mengontrol perilaku model dan output yang dihasilkan. Pemahaman terhadap parameter-parameter ini sangat penting untuk mengoptimalkan hasil sesuai kebutuhan aplikasi (Ziegler et al., 2022).

Beberapa parameter kunci antara lain:

1. *model*: (Wajib) ID dari model yang ingin digunakan, misalnya gpt-4, gpt-4-turbo, atau gpt-3.5-turbo. Pemilihan model akan mempengaruhi kualitas, kecepatan, dan biaya (Brown et al., 2020).

2. *messages*: (Wajib) Sebuah array dari objek pesan yang merepresentasikan percakapan sejauh ini. Setiap objek pesan memiliki dua properti:
3. *role*: Peran dari pengirim pesan. Bisa berupa *system* (untuk memberikan instruksi tingkat tinggi), *user* (untuk pesan dari pengguna akhir), atau *assistant* (untuk respons dari model).
4. *content*: Isi dari pesan teks.
5. *temperature*: Mengontrol tingkat keacakan atau "kreativitas" dari output. Nilainya berkisar antara 0.0 hingga 2.0. Nilai yang lebih tinggi (misalnya, 0.8) akan membuat output lebih acak dan kreatif, sedangkan nilai yang lebih rendah (misalnya, 0.2) akan membuatnya lebih fokus, deterministik, dan seringkali lebih faktual (Holtzman et al., 2019).
6. *max_tokens*: Jumlah maksimum token (kata atau potongan kata) yang dapat dihasilkan dalam respons. Ini berguna untuk membatasi panjang dan biaya respons.
7. *top_p*: Sebuah alternatif untuk temperature yang juga mengontrol keacakan. Parameter ini menggunakan teknik nucleus sampling, di mana model hanya mempertimbangkan token dengan total probabilitas top_p. Nilai 1.0 berarti mempertimbangkan semua token, sedangkan nilai yang lebih rendah (misalnya, 0.1) akan menghasilkan output yang sangat terbatas dan fokus.
8. *stream*: Jika diatur ke true, API akan mengirimkan

respons secara bertahap (token per token) seiring dengan dihasilkannya, bukan menunggu seluruh respons selesai. Ini sangat berguna untuk aplikasi interaktif seperti chatbot untuk memberikan umpan balik yang lebih cepat kepada pengguna.

Untuk integrasi *API* dari *OpenAI* dengan *Golang*, *OpenAI* tidak secara resmi menyediakan *library* (SDK) untuk Golang, namun komunitas telah mengembangkan beberapa *library* populer yang sangat menyederhanakan proses integrasi. Salah satu *library* yang umum digunakan adalah *go-openai* dari sashabaranov. Berikut adalah langkah-langkah dasar untuk mengintegrasikan API OpenAI di Golang:

1. Inisialisasi Proyek dan Instalasi *Library*

   Buat proyek Golang baru dan instal *library* go-openai:

```
go mod init nama-proyek-anda
go get github.com/sashabaranov/go-openai
```

Segmen Program 2.1

Inisialisasi Proyek dan Instalasi *Library*

2. Konfigurasi *Client*

Buat sebuah *client OpenAI* dengan menggunakan *API Key* yang telah diperoleh dari website OpenAI. *API Key* sebaiknya disimpan di dalam environment variable.

```
package main

import (
    "context"
    "fmt"
    "log"
    "os"

    "github.com/sashabaranov/go-openai"
)

func main() {
    // Memuat API Key dari environment variable
    apiKey := os.Getenv("OPENAI_API_KEY")
    if apiKey == "" {
        log.Fatal("OPENAI_API_KEY environment
variable tidak diatur")
    }

    // Membuat klien baru
    client := openai.NewClient(apiKey)

    // ... (kode untuk memanggil API)
}
```

Segmen Program 2.2

Kode program inisialisasi client OpenAI

3. Membuat *Request* ke *API Chat Completions*

Endpoint Chat Completions adalah endpoint yang paling kuat dan fleksibel, dirancang untuk model-model terbaru seperti GPT-4 dan GPT-3.5-Turbo. *Endpoint* ini menerima serangkaian pesan sebagai input.

```
// (melanjutkan dari kode sebelumnya)

    // Membuat permintaan chat completion
    resp, err := client.CreateChatCompletion(
        context.Background(),
        openai.ChatCompletionRequest{
            Model: openai.GPT_4_TURBO,
            Messages: []openai.ChatCompletionMessage{
                {
                    Role:
openai.ChatMessageRoleUser,
                    Content: "Buatlah sebuah fungsi
sederhana di Golang untuk menjumlahkan dua angka.",
                },
            },
        },
    )

    if err != nil {
        log.Fatalf("Error saat memanggil
ChatCompletion API: %v", err)
    }

    fmt.Println(resp.Choices[0].Message.Content)
```

Segmen Program 2.3

Kode program untuk mendapatkan chat completion dari OpenAI

2.2.9 *Chatbot*

Salah satu aplikasi dari NLP adalah *chatbot*. Menurut Adriyani (2004), chat dapat didefinisikan sebagai komunikasi verbal. *Chatbot* adalah sebuah program komputer yang dirancang untuk mengimitasi percakapan manusia dengan mereplikasi kecerdasan buatan,

seperti yang dijabarkan oleh Shawar & Atwell (2002). *Chatbot* merupakan implementasi dari area-area keilmuan seperti *natural language processing*, *machine learning*, rekayasa perangkat lunak, dan kecerdasan buatan. Objektifnya adalah untuk meniru dialog manusia melalui medium teks atau suara, dengan mengaplikasikan aturan yang ditetapkan sebelumnya atau melalui pemanfaatan kecerdasan buatan. *Chatbot* berpotensi memperoleh pembelajaran dari pola percakapan untuk menyajikan respons yang relevan terhadap permintaan tertulis atau lisan. Selain itu, *chatbot* mempunyai potensi konektivitas dengan sumber data eksternal untuk menyediakan informasi atau layanan sesuai dengan kebutuhan pengguna, contohnya cuaca, berita, atau reservasi hotel. *Chatbot* bisa dikembangkan untuk berbagai fungsi dan kebutuhan, namun ada beberapa tipe utama chatbot berdasarkan cara mereka memproses permintaan.

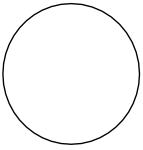
a. *Button-Based Chatbot*, tipe ini membatasi interaksi pengguna dengan menyediakan tombol yang sudah terprogram untuk respons tertentu. Kelemahannya adalah kurangnya fleksibilitas dalam menerima input bebas dari pengguna.
b. *Keyword Recognition-Based Chatbo*t, *Chatbot* ini merespons berdasarkan kata kunci yang dikenali dalam permintaan pengguna. Meskipun lebih fleksibel dibandingkan *button-based*, kelemahannya adalah dapat bingung jika menerima kata kunci yang sama dalam konteks yang berbeda.
c. *Contextual Chatbot,* ini adalah tipe *chatbot* yang paling maju, menggunakan *machine learning* dan kecerdasan buatan untuk memahami dan mengingat konteks percakapan. *Chatbot* ini dapat

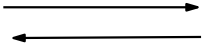
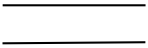
mempelajari dan menyesuaikan respons berdasarkan interaksi sebelumnya, memungkinkan respons yang lebih akurat meskipun kata-kata dalam permintaan berbeda tetapi memiliki arti yang sama. Tipe ini memerlukan pengembangan dan pembaruan *knowledge base* secara berkala untuk meningkatkan kemampuannya.

2.2.10 *Data Flow Diagram*

Muhammad Robith Adani (2021) mendefinisikan Data Flow Diagram (DFD) sebagai representasi visual yang memperlihatkan pergerakan data di dalam suatu alur proses atau sistem informasi. Dalam sebuah DFD, termuat informasi mengenai data yang masuk (input) dan data yang keluar (output) dari setiap tahapan proses. DFD juga memiliki kegunaan untuk mengkomunikasikan desain sistem, memberikan gambaran umum sistem secara komprehensif, serta memfasilitasi proses perancangan model sistem.

Tabel 2.6 Simbol-simbol dalam Data Flow Diagram

No.	Simbol	Nama	Keterangan
1		Proses transformasi	Proses perubahan data dari data *input* menjadi data *output*
2		Sumber dan tujuan data	User atau admin yang mengirimkan da

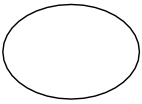
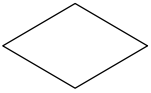
			ta ke dalam sistem dan juga menerima beberapa data dari sistem.
3		Arus data	Arus data dari setiap proses.
4		Penyimpanan data	Tempat data disimpan.

Sumber : Al-Bahra bin Ladjamudin versi Yourdan, De Marco, dan lainnya. (2005:72)

2.2.11 ERD (*Entity Relationship Diagram)*

Menurut Meilina Eka (2023) ERD merupakan representasi grafis dari struktur data yang mencerminkan interaksi antar entitas dalam suatu sistem. Dalam diagram ini, tergambar entitas beserta atribut-atributnya, serta keterkaitan antar entitas dengan tingkat detail yang tinggi. Penggunaan ERD tidak terbatas pada relasi 1:1 (*one to one*), 1:M (*one to many*), atau M:M (*many to many*). ERD memiliki karakteristik khususnya sebagai suatu alat untuk mengilustrasikan interaksi antar tabel dalam sebuah basis data.

Tabel 2.7 Simbol-simbol dalam ERD

No.	Gambar	Nama	Keterangan
1.		*Entity*	*Entity* atau Entitas adalah suatu agregat dari entitas-entitas yang memiliki identifikasi yang khas dan dapat dibedakan secara jelas antara satu dengan lainnya, yang merepresentasikan suatu keberadaan yang konkret serta memiliki karakteristik pembeda yang signifikan jika dibandingkan dengan entitas-entitas lainnya
2.		*Attribute*	Atribut merupakan elemen-elemen yang mendeskripsikan karakteristik dari entitasnya. Untuk *primary key* akan diberikan *underscore* pada nama atribut nya.
3.		*Relationship*	Hubungan antar entitas yang berbeda.
4.		*Link*	Garis yang menggambarkan hubungan antar entitas dengan relasi

Sumber : Al-Bahra bin Ladjamudin (2005:149) Analisis dan Desain Sistem Informasi. Yogyakarta : Graha Ilmu

2.2.12 *Android*

Android merupakan sebuah platform sistem operasi yang dikhususkan untuk perangkat telepon seluler dan tablet. Proyek ini pertama kali digagas oleh perusahaan teknologi asal Amerika Serikat, *Android Inc.*, pada tahun 2003 dengan tujuan awal adalah menciptakan sistem operasi bagi kamera digital. Akan tetapi, di tahun 2004, arah proyek berubah menjadi pengembangan sistem operasi untuk smartphone. Pada tahun 2005, *Google Inc*. membeli *Android Inc.*, dan tim *Android* memutuskan untuk membangun proyek mereka berdasarkan sistem operasi sumber terbuka *Linux* (*open source Linux*).

Pada tanggal 5 November 2007, *Google* menyampaikan pengumuman mengenai pembentukan *Open Handset Alliance*, sebuah gabungan dari berbagai perusahaan teknologi dan telekomunikasi yang beranggotakan perusahaan seperti *Intel*, *Motorola*, *NVIDIA*, *Texas Instruments*, *LG Electronics*, *Samsung Electronics*, *Sprint Nextel*, dan *T-Mobile*. Konsorsium ini memiliki tujuan untuk mengembangkan serta mempopulerkan *Android* sebagai sistem operasi sumber terbuka gratis yang mendukung penggunaan aplikasi pihak ketiga. Perangkat yang berbasis *Android* menggunakan koneksi jaringan nirkabel untuk fitur-fitur semacam pencarian *Google* sekali sentuh, *Google Docs*, dan *Google Earth*.

Telepon seluler pertama yang mengimplementasikan sistem operasi baru ini adalah *T-Mobile G1*, yang dikeluarkan pada 22 Oktober 2008. Pada tahun 2012, *Android* berhasil menjadi sistem operasi paling

populer untuk perangkat seluler, melebihi dominasi *iOS* milik *Apple*. Hingga tahun 2020, kurang lebih 75% perangkat seluler di dunia menjalankan *Android* (Erik Gregersen, 2024).

2.2.13 *iOS*

iOS adalah sistem operasi *multi platform* yang diciptakan dan dikembangkan oleh Apple Inc. untuk iPhone. Sebelumnya, platform ini juga mendukung iPad dan iPod Touch. Perangkat lunak ini didasarkan pada Mac OS X (setelah 2016 dikenal sebagai macOS), yang digunakan Apple sejak 2001 untuk komputer Macintosh. iOS juga menjadi basis untuk watchOS, tvOS, dan iPadOS yang mendukung Apple Watch, Apple TV, dan iPad. Pada tahun 2022, iOS masih menjadi sistem operasi mobile kedua yang paling banyak diinstal di dunia.

Perilisan iOS pada iPhone generasi pertama tahun 2007 membawa kemajuan besar dalam teknologi multitouch yang memungkinkan pengguna mengoperasikan komputer dengan tekanan kontak. Pembaruan sistem tahun 2008 memperbolehkan pengguna membeli aplikasi baru dari pengembang pihak ketiga, yang membuat iPhone menjadi sukses secara finansial dan salah satu produk konsumen paling signifikan abad ke-21. Keberhasilan iPhone melahirkan pasar baru untuk aplikasi smartphone yang mengubah cara orang bersosialisasi, berbelanja, dan bekerja.

Fitur baru yang inovatif terus muncul dalam pembaruan iOS berikutnya, seperti FaceTime dalam iOS 4, Siri dalam iOS 6, dan Apple Pay dalam iOS 8. Meskipun tidak semua fitur baru iOS mengalami peluncuran

yang mulus, Apple terus meningkatkan layanan, seperti Apple Maps setelah kritik atas kinerja yang buruk. Pertumbuhan terus-menerus dari kemampuan komputasi membawa kemampuan iPhone semakin dekat dengan Mac, memungkinkan Apple untuk merilis fitur Mac OS baru bersama versi iOS-nya. (Adam Volle, 2024)

2.2.14 *React Native*

React Native adalah sebuah kerangka kerja (*framework*) yang didasarkan pada *JavaScript* dan dimanfaatkan untuk membuat aplikasi bergerak yang dapat berjalan di dua platform sistem operasi sekaligus, yakni *Android* dan *iOS*. *Framework* ini pertama kali diperkenalkan oleh *Facebook* pada tahun 2015 dan memiliki sifat sumber terbuka (*open source*) (Rony Setiawan, 2021).

2.2.15 *Flutter*

Bamai Uma (2023) menjelaskan bahwa *Flutter* merupakan sebuah kerangka kerja sumber terbuka (*open source framework*) yang diciptakan oleh *Google* guna membuat aplikasi *multi-platform* dengan hanya menggunakan satu basis kode (*codebase*). Dengan *Flutter*, aplikasi dapat dirancang untuk berbagai platform, termasuk *Android*, *iOS*, *Desktop*, dan *Website*. Dalam proses pembuatannya, *Flutter* memakai bahasa pemrograman *Dart*.

2.2.16 Golang

Menurut Muhammad Robith Adani dari Sekawan Media (2021), bahasa pemrograman *Go*, yang lazim pula dikenal dengan sebutan *Golang*, merupakan suatu jenis bahasa yang secara khusus dirancang dan dikembangkan oleh *Google* bekerja sama dengan para pakar komputasi Ken Thompson, Robert Griesemer, dan Rob Pike pada tahun 2009. Motivasi utama dari inisiasi pengembangannya adalah untuk menciptakan sebuah bahasa yang superior dalam aspek kecepatan, tingkat kepercayaan (reliabilitas), skalabilitas, dan tingkat kesederhanaan. *Golang* tergolong ke dalam jenis bahasa yang mengimplementasikan pengetikan data secara terstruktur dan memproduksi kode biner yang siap untuk proses kompilasi. Lebih lanjut, *Golang* diadaptasi dari kerangka dasar bahasa pemrograman *C* guna mengakomodasi kebutuhan era abad ke-21. Bahasa *Go* mempunyai kapabilitas untuk merancang dan membangun beragam jenis aplikasi, website, dan produk perangkat lunak lainnya.

2.2.17 *Application Programming Interface* (API)

Application Programming Interface (API) merupakan sebuah antarmuka yang didefinisikan secara formal dan menyediakan sekumpulan fungsi, prosedur, serta protokol yang memungkinkan suatu komponen perangkat lunak untuk berinteraksi dan bertukar informasi dengan komponen perangkat lunak lainnya (Fielding, 2000; Pautasso et al., 2008). Dalam konteks sistem terdistribusi dan arsitektur berorientasi layanan *(Service-Oriented Architecture - SOA)* serta *microservices,* API memainkan peran krusial sebagai "kontrak" yang mengatur bagaimana layanan-layanan

independen dapat berkomunikasi dan berkolaborasi (Erl, 2005; Newman, 2015). API memungkinkan modularitas dan *loose coupling* antar komponen, di mana detail implementasi internal suatu layanan tidak perlu diketahui oleh layanan lain yang menggunakannya.

Salah satu gaya arsitektur API yang dominan saat ini adalah REST (*Representational State Transfer*), yang diperkenalkan oleh Roy Fielding (2000). RESTful API memanfaatkan protokol HTTP/HTTPS standar untuk komunikasi. Interaksi dalam RESTful API umumnya bersifat *stateless* dan mengikuti pola *request-response* antara *client* dan *server*:

1. *Client:* Aplikasi atau layanan yang membutuhkan akses ke sumber daya.

   Request: Client mengirimkan permintaan HTTP ke server. Permintaan ini mencakup:

   a. *HTTP Method*: Menunjukkan operasi yang diinginkan terhadap sumber daya (misalnya, *GET* untuk mengambil, *POST* untuk membuat, *PUT* untuk memperbarui, *DELETE* untuk menghapus) (Fielding, 2000).
   b. URI *(Uniform Resource Identifier) / Endpoint:* Alamat unik yang mengidentifikasi sumber daya di server.
   c. *Headers:* Metadata tambahan mengenai permintaan (misalnya, tipe konten, informasi autentikasi).
   d. *Body* (Opsional): Data yang dikirim ke server, biasanya dalam format seperti JSON atau XML, terutama untuk metode POST dan PUT (Laitkorpi et al., 2009).
   e. *Server:* Sistem yang mengelola sumber daya dan

memproses permintaan.

2. Response: Server mengirimkan kembali respons HTTP ke klien, yang mencakup:

a. Status Kode HTTP: Mengindikasikan hasil dari permintaan (misalnya, 200 OK, 201 *Created*, 400 *Bad Request*, 404 *Not Found*, 500 *Internal Server Error*) (Fielding et al., 2014).
b. *Headers:* Metadata tambahan mengenai *response.*
c. *Body* (Opsional): Representasi dari sumber daya yang diminta atau hasil operasi, seringkali dalam format JSON.

Dalam konteks RESTful API, *endpoint* adalah URI spesifik yang digunakan klien untuk mengakses atau memanipulasi sumber daya tertentu di server. Desain *endpoint* yang baik biasanya berfokus pada *nouns* (kata benda) yang merepresentasikan sumber daya, bukan *verbs* (kata kerja) yang merepresentasikan aksi (Richardson & Ruby, 2007). Struktur endpoint yang konsisten dan intuitif merupakan salah satu karakteristik API yang dirancang dengan baik (Bloch, 2006). Sebagai contoh:

1. GET /users: Mengambil daftar pengguna.
2. POST /users: Membuat pengguna baru.
3. GET /users/{userId}: Mengambil detail pengguna dengan ID tertentu.
4. PUT /users/{userId}: Memperbarui data pengguna dengan ID tertentu.

JSON (*JavaScript Object Notation*) telah menjadi format data

standar *de facto* untuk pertukaran data dalam API web modern karena sifatnya yang ringan, mudah dibaca manusia, dan mudah diproses oleh mesin (Crockford, 2006; Pautasso et al., 2008). JSON merepresentasikan data dalam bentuk pasangan kunci-nilai (*key-value* pairs) dan array, yang memfasilitasi serialisasi dan deserialisasi objek data di berbagai bahasa pemrograman.

Contoh *response* JSON dari sebuah API:

```
{
  "status": "success",
  "data": {
    "id": "user123",
    "username": "vita_programmer",
    "email": "vita@example.com",
    "preferences": {
      "theme": "dark",
      "notifications": true
    }
  }
}
```

Segmen Program 2.4

Contoh *response* JSON

Penggunaan JSON dalam API mendukung interoperabilitas yang lebih baik antar sistem yang heterogen dibandingkan format yang lebih berat seperti XML (Laitkorpi et al., 2009).

Golang (Go) sering dipilih untuk pengembangan API karena efisiensi kompilasi, performa *runtime* yang tinggi, dukungan konkurensi yang kuat melalui *goroutine* dan *channel*, serta pustaka standar net/http

yang komprehensif untuk membangun server HTTP (Donovan & Kernighan, 2015).

Meskipun detail implementasi akan sangat bergantung pada framework atau pustaka yang digunakan (jika ada), prinsip dasar pembuatan endpoint API di Golang menggunakan pustaka net/http melibatkan:

Mendefinisikan Struktur Data (Structs): Merepresentasikan entitas data yang akan ditransfer. Anotasi tag pada field struct (json:"fieldName") digunakan untuk mengontrol bagaimana struct tersebut di-encode ke atau di-decode dari JSON.

```
// User struct merepresentasikan data pengguna
type User struct {
    ID       string `json:"id"`
    Username string `json:"username"`
    Email    string `json:"email"`
}
```

Segmen Program 2.5

Contoh pembuatan *Struct* di Golang

Membuat Fungsi Handler (*Handler Functions*): Setiap *endpoint* akan dipetakan ke sebuah fungsi handler. Fungsi ini menerima *http.ResponseWriter* (untuk mengirim respons) dan *http.Request (untuk mengakses detail permintaan). Di dalam handler, logika bisnis diimplementasikan, data diambil atau dimanipulasi, dan respons (biasanya dalam format JSON) dikirim kembali ke klien.

```
import (
    "encoding/json"
    "net/http"
)

func getUserHandler(w http.ResponseWriter, r
*http.Request) {
     user := User{
                ID: "user123",
                Username: "vita_programmer",
                Email: "vita@example.com"
            }
    w.Header().Set("Content-Type",
"application/json")
    w.WriteHeader(http.StatusOK)
    json.NewEncoder(w).Encode(user)
}
```

Segmen Program 2.6

Contoh kode program API handler di Golang

Melakukan Routing (Multiplexing): Mendaftarkan fungsi handler ke path URL (endpoint) dan metode HTTP yang sesuai. Ini dapat dilakukan menggunakan http.ServeMux bawaan atau router/framework pihak ketiga untuk fungsionalitas yang lebih canggih.

```
func main() {
    mux := http.NewServeMux()
    mux.HandleFunc("/api/v1/users", func(w
http.ResponseWriter, r *http.Request) {
        if r.Method == http.MethodGet {
            getUserHandler(w, r)
        } else if r.Method == http.MethodPost {
            createUserHandler(w, r)
        } else {
            http.Error(w, "Metode tidak diizinkan",
http.StatusMethodNotAllowed);
        }
    })
    // ... (definisi endpoint lain)
}
```

Segmen Program 2.7

Pembuatan routing di Golang

Menjalankan Server HTTP: Menggunakan http.ListenAndServe untuk memulai server agar dapat menerima dan memproses permintaan HTTP yang masuk.

```
import (
    "log"
    "net/http"
)

func main() {
    mux := http.NewServeMux()
    // ... (registrasi handler seperti di atas)
    mux.HandleFunc("/api/v1/users", getUserHandler)
    log.Println("API server berjalan pada port
:8080")
    if err := http.ListenAndServe(":8080", mux); err
!= nil {
        log.Fatalf("Tidak dapat memulai server:
%s\n", err)
    }
}
```

Segmen Program 2.8

Integrasi handler yang sudah dibuat dengan routing

Pengembangan API yang robust juga melibatkan pertimbangan lain seperti autentikasi, otorisasi, *rate limiting, logging,* penanganan error yang baik, dan dokumentasi API (misalnya menggunakan OpenAPI Specification) (Masse, 2011). *Framework web* di *Golang* seperti *Gin, Echo,* atau *Chi* sering digunakan untuk menyederhanakan banyak aspek ini.

Secara keseluruhan, API adalah fondasi penting dalam pengembangan perangkat lunak modern, memungkinkan sistem untuk berinteraksi secara efisien dan fleksibel (Oreilly & Lanyon, 2019). Desain API yang baik, yang mengikuti prinsip-prinsip seperti yang diuraikan dalam

REST, sangat penting untuk keberhasilan integrasi sistem dan pengembangan aplikasi yang *scalable*.

2.2.18 *MySql*

MySQL adalah sebuah *Relational Database Management System - RDBMS open-source* yang sangat populer dan banyak digunakan dalam pengembangan berbagai jenis aplikasi, mulai dari aplikasi web skala kecil hingga sistem perusahaan yang besar (DuBois, 2013). Sebagai sebuah RDBMS, MySQL mengorganisir data ke dalam tabel-tabel yang terstruktur. Setiap tabel terdiri dari *rows* dan *columns*, di mana setiap baris merepresentasikan sebuah entitas data unik (seperti data seorang pengguna) dan setiap kolom merepresentasikan atribut dari entitas tersebut (seperti nama, email, atau tanggal lahir). Hubungan antar tabel dapat didefinisikan menggunakan *primary key* dan *foreign key*, yang memungkinkan integritas dan konsistensi data tetap terjaga (Garcia-Molina, Ullman, & Widom, 2008).

MySQL menggunakan bahasa kueri standar, yaitu *Structured Query Language* (SQL), untuk melakukan semua operasi terhadap basis data. SQL adalah bahasa deklaratif yang memungkinkan pengguna untuk mendefinisikan, memanipulasi, dan mengontrol data di dalam RDBMS (Chamberlin & Boyce, 1974). Arsitektur MySQL yang berbasis *client-server* memungkinkan aplikasi (*client*) untuk terhubung ke server MySQL melalui jaringan, mengirimkan *query* SQL, dan menerima hasilnya. Hal ini membuatnya sangat fleksibel untuk digunakan dalam arsitektur aplikasi terdistribusi.

Operasi dasar yang dapat dilakukan pada data dalam sebuah basis data dikenal dengan akronim CRUD, yang merupakan singkatan dari *Create, Read, Update,* dan *Delete*. Operasi ini adalah fondasi dari hampir semua aplikasi yang berinteraksi dengan basis data (Korth, Silberschatz, & Sudarshan, 2010).

1. *Create* (Membuat Data)

   Operasi *Create* berfungsi untuk menambahkan data baru ke dalam sebuah tabel. Dalam SQL, operasi ini dilakukan menggunakan perintah *INSERT INTO*. Perintah ini menspesifikasikan nama tabel, kolom-kolom yang akan diisi, dan nilai-nilai yang akan dimasukkan.

   Sintaks Dasar:

```
INSERT INTO nama_tabel (kolom1, kolom2, kolom3)
VALUES (nilai1, nilai2, nilai3);
```

Segmen Program 2.9

Format *Query Insert*

Contoh (Menambahkan pengguna baru):

```
INSERT INTO Users (nickname, email, password,
birth_date)
VALUES ('vita_user', 'user@example.com',
'hashed_password', '2004-02-17');
```

Segmen Program 2.10

Contoh *Query Insert*

2. *Read* (Membaca Data)

   Operasi *Read* berfungsi untuk mengambil atau membaca data dari satu atau lebih tabel. Ini adalah operasi yang paling sering

digunakan dan dilakukan dengan perintah *SELECT*. Perintah *SELECT* dapat dikombinasikan dengan berbagai klausa seperti *WHERE* (untuk memfilter data berdasarkan kondisi tertentu), *JOIN* (untuk menggabungkan data dari beberapa tabel), *ORDER BY* (untuk mengurutkan hasil), dan *LIMIT* (untuk membatasi jumlah baris yang diambil).

Sintaks Dasar:

```
SELECT kolom1, kolom2 FROM nama_tabel WHERE kondisi;
```

Segmen Program 2.11

Format *Query Select*

Contoh Praktis (Mengambil data pengguna berdasarkan email):

```
SELECT id, nickname, email, birth_date FROM Users
WHERE email = 'user@example.com';
```

Segmen Program 2.12

Contoh Query Select

3. *Update* (Memperbarui Data)

   Operasi *Update* berfungsi untuk memodifikasi atau mengubah data yang sudah ada di dalam tabel. Perintah yang digunakan adalah *UPDATE*, yang akan mengubah nilai pada kolom tertentu untuk baris yang memenuhi kondisi pada klausa *WHERE*. Sangat penting untuk selalu menggunakan klausa *WHERE* pada perintah *UPDATE* untuk menghindari perubahan data pada semua baris di dalam tabel.

   Sintaks Dasar:

```
UPDATE nama_tabel
SET kolom1 = nilai_baru1, kolom2 = nilai_baru2
WHERE kondisi;
```

Segmen Program 2.13

Format *Query Update*

Contoh Praktis (Mengubah nickname pengguna):

```
UPDATE Users
SET nickname = 'vita_new_user'
WHERE id = 123;
```

Segmen Program 2.14

Contoh *Query Update*

4. *Delete* (Menghapus Data)

   Operasi *Delete* berfungsi untuk menghapus satu atau lebih baris data dari sebuah tabel. Perintah yang digunakan adalah *DELETE FROM*. Sama seperti *UPDATE*, sangat krusial untuk menggunakan klausa *WHERE* untuk menentukan baris mana yang akan dihapus. Jika klausa *WHERE* dihilangkan, semua data di dalam tabel akan terhapus.

   Sintaks Dasar:

```
DELETE FROM nama_tabel WHERE kondisi;
```

Segmen Program 2.15

Format *Query Delete*

Contoh Praktis (Menghapus pengguna berdasarkan id):

```
DELETE FROM Users WHERE id = 123;
```

Segmen Program 2.16

Contoh *Query Delete*

Penguasaan operasi CRUD ini adalah kemampuan fundamental dalam pengembangan aplikasi berbasis basis data. MySQL, dengan implementasi SQL yang kuat dan performa yang andal, menyediakan platform yang solid untuk melakukan operasi-operasi ini secara efisien.

2.2.19 Evaluasi Aplikasi

Evaluasi aplikasi merupakan tahap krusial dalam siklus pengembangan perangkat lunak untuk mengukur keberhasilan suatu sistem dari perspektif pengguna. Salah satu metode yang paling umum digunakan adalah penyebaran kuesioner terstruktur. Kuesioner memungkinkan pengumpulan data kuantitatif mengenai persepsi pengguna terhadap berbagai aspek aplikasi. Desain kuesioner dalam penelitian ini didasarkan pada beberapa model dan kerangka kerja evaluasi yang telah mapan dalam literatur interaksi manusia-komputer (*Human-Computer Interaction - HCI*) dan sistem informasi.

Kerangka kerja utama yang diadopsi adalah model T*echnology Acceptance Model (TAM)* yang dikembangkan oleh Davis (1989) dan standar ISO 9241-11 tentang *Usability*. TAM mengemukakan bahwa penerimaan pengguna terhadap suatu teknologi sangat dipengaruhi oleh dua faktor utama: Persepsi Kegunaan (*Perceived Usefulness*) dan Persepsi Kemudahan Penggunaan (*Perceived Ease of Use*). Sementara itu, ISO 9241-11 mendefinisikan *usability* sebagai sejauh mana suatu produk dapat digunakan oleh pengguna tertentu untuk mencapai tujuan tertentu dengan

efektivitas, efisiensi, dan kepuasan dalam konteks penggunaan tertentu.

Berdasarkan kerangka kerja tersebut, pertanyaan-pertanyaan dalam kuesioner penelitian ini dikelompokkan ke dalam beberapa konstruk evaluasi berikut:

1. Kemudahan Penggunaan (*Ease of Use*)

   Konstruk ini mengukur persepsi pengguna mengenai seberapa mudah mereka dapat menggunakan dan berinteraksi dengan aplikasi tanpa mengalami kesulitan yang berarti. Ini adalah pilar utama dalam model TAM dan *usability* (Davis, 1989; Nielsen, 1993). Pertanyaan yang dirancang untuk mengukur aspek ini adalah:

   a. *"Aplikasi Vita mudah digunakan."*
   b. *"Antarmuka aplikasi Vita mudah digunakan dan mudah dipahami."*

   Pertanyaan-pertanyaan ini bertujuan untuk mengevaluasi aspek *learnability* (kemudahan untuk dipelajari) dan kejelasan antarmuka aplikasi. Menurut Nielsen (1993), sistem yang baik harus mudah dipelajari sehingga pengguna dapat dengan cepat mulai mengerjakan tugas mereka.

2. Kualitas Performa dan Respons (*Performance and Response Quality*)

   Untuk aplikasi interaktif seperti *chatbot*, kualitas performa, terutama kecepatan dan akurasi respons, adalah faktor kunci yang mempengaruhi pengalaman pengguna (Radziwill & Benton, 2017).

Konstruk ini mengukur kemampuan teknis aplikasi dalam memberikan layanan yang cepat dan relevan. Pertanyaan yang dirancang untuk mengukur aspek ini adalah:

a. *"Aplikasi Vita mampu merespon dengan cepat pertanyaan yang saya berikan."*
b. *"Vita mampu menjawab pertanyaan saya dengan akurat, relevan, dan kontekstual."*
c. *"Vita mampu mengenali gambar yang saya kirimkan dan memberikan respon yang sesuai dan kontekstual."*

Kecepatan *response* (latensi) secara langsung berkaitan dengan efisiensi dalam standar ISO 9241-11. Sementara itu, akurasi, relevansi, dan kemampuan menangani input multimodal (teks dan gambar) merupakan cerminan dari efektivitas sistem dalam membantu pengguna mencapai tujuannya (Jia, 2009).

3. Persepsi Kegunaan (*Perceived Usefulness*)

Konstruk ini, yang berasal dari model TAM, mengukur sejauh mana pengguna percaya bahwa menggunakan aplikasi akan meningkatkan kinerja atau produktivitas mereka (Davis, 1989). Dalam konteks penelitian ini, kegunaan diukur melalui kemampuan aplikasi "Vita" untuk membantu *programmer* dalam menyelesaikan tugas mereka. Pertanyaan yang dirancang untuk mengukur aspek ini adalah:

a. *"Vita membantu saya menyelesaikan masalah atau tugas saya dengan lebih cepat."*

Pertanyaan ini secara langsung mengukur dampak aplikasi

terhadap efisiensi kerja pengguna, yang merupakan inti dari persepsi kegunaan.

4. Kepuasan Pengguna (*User Satisfaction*)

   Kepuasan adalah komponen afektif dari evaluasi dan merupakan salah satu dari tiga pilar *usability* menurut ISO 9241-11. Konstruk ini mengukur perasaan subjektif dan kenyamanan pengguna setelah berinteraksi dengan aplikasi (Laugwitz et al., 2008). Pertanyaan yang dirancang untuk mengukur aspek ini adalah:

   a. *"Saya merasa nyaman menggunakan aplikasi Vita."*

b. *"Saya puas dengan performa aplikasi Vita secara keseluruhan."*

   Kepuasan pengguna seringkali dianggap sebagai hasil akhir dari kombinasi kemudahan penggunaan, kegunaan, dan performa aplikasi yang baik. Ini adalah indikator holistik dari keberhasilan pengalaman pengguna (*User Experience*) secara keseluruhan.

2.3 Gambaran Umum Obyek Penelitian

Gambar 2.3 Logo PT Living Fitness Indonesia

PT Living Fitness Indonesia merupakan salah satu *start-up* yang bergerak di bidang *corporate wellness solution,* PT Living Fitness Indonesia didirikan pada tahun 2019 dan beralamat di Jl. Hayam Wuruk No.27,

RT.14/RW.1, Kb. Klp. Kecamatan Gambir, Jakarta Pusat. *Corporate wellness solution* adalah program atau inisiatif yang diterapkan oleh perusahaan untuk meningkatkan kesehatan dan kesejahteraan karyawan mereka. Program ini mencakup berbagai kegiatan dan manfaat, seperti program kebugaran, pemeriksaan kesehatan, workshop manajemen stres, edukasi nutrisi, dan dukungan kesehatan mental. Tujuannya adalah untuk meningkatkan kesehatan keseluruhan karyawan, yang dapat menghasilkan peningkatan produktivitas, pengurangan biaya kesehatan, dan peningkatan kepuasan kerja. Dalam menjalankan programnya PT Living Fitness Indonesia mengembangkan aplikasi mobile menggunakan *React Native*, yang mana aplikasi ini dapat mengakses data pada *fitness tracker* yang digunakan oleh *user*, umumnya fitness tracker ini berupa *smartband* atau *smartwatch.* Setiap minggunya PT Living Fitness Indonesia akan memberikan *challenge* kepada client-client nya berupa minimum target aktivitas seperti minimal langkah per-harinya atau minimal waktu latihan per-harinya, setelah *user* menyentuh target aktivitas tertentu, maka *user* akan mendapatkan poin harian, poin harian ini akan terus dikalkulasi sampai batas waktu *challenge* berakhir. Setelah *challenge* berakhir, *user* yang mencapai target poin akan mendapatkan reward khusus dari perusahaannya. Dan poin yang sudah dikumpulkan dapat ditukar reward seperti token listrik, pulsa, saldo dana, dan lain-lain.